Time-of-Flight Based Interior Sensing for Automotive Applications

Optimization of Neural Network Based Training Methods with Limited Data

von der Fakultät für Elektrotechnik, Informationstechnik und Medientechnik
der Bergischen Universität Wuppertal genehmigte

Dissertation

---

zur Erlangung des akademischen Grades
eines Doktors der Ingenieurwissenschaften

von

M. Sc. Patrick Weyers

aus

Witten

Wuppertal 2021

Tag der Prüfung: 27.10.2021
Hauptreferent: Prof. Dr.-Ing. Anton Kummert
Korreferent: Prof Dr.-Ing. Anselm Haselhoff

Vorwort

Die vorliegende Dissertation entstand an der Fakultät für Elektrotechnik, Informationstechnik und Medientechnik am Lehrstuhl für Allgemeine Elektrotechnik und Theoretische Nachrichtentechnik der Bergischen Universität Wuppertal. Meine Forschungstätigkeit fand in Kooperation mit der Abteilung für Interior Sensing der Firma Aptiv Services Deutschland GmbH im Rahmen eines Promotionsstipendiums der Bergischen Universität Wuppertal statt.

An dieser Stelle möchte ich allen Menschen meinen Dank aussprechen, die mich bei der Erstellung meiner Dissertation unterstützt haben.

Ich möchte Herrn Professor Anton Kummert für die Betreuung und Begutachtung, sowie für die diversen Möglichkeiten meine Forschungsergebnisse auf Konferenzen zu präsentieren, danken.
Ebenfalls möchte ich Professor Anselm Haselhoff für die Begutachtung in seiner Tätigkeit als Korreferrent danken.

Mein besonderer Dank gilt Alexander Barth für seinen Einsatz und die enorme Unterstützung bei der Durchführung meines Promotionsvorhabens.

Weiterhin gilt mein Dank allen KollegInnen bei Aptiv, insbesondere David Schiebener, sowie meinen Mitdoktoranden Lukas Hahn und Martin Alsfasser, die mich auf meinem Weg immer mit offenen Ohren und Anregungen unterstützt haben.

Von Herzen danke ich meinen Eltern Sabine und Klaus, ohne die mein gesamter Werdegang so nicht möglich gewesen wäre.
Aus vollem Herzen danke ich Lynn-Sophia, die mich während meiner gesamten Promotionszeit mit Verständnis und Motivation unterstützt hat.

Kurzfassung

Das Ziel dieser Arbeit ist die Entwicklung von neuen Systemen zur Innenraumüberwachung von Fahrzeugen basierend auf künstlichen neuronalen Netzwerken und Time-of-Flight Bilddaten. Diese Entwicklungen beinhalten Methoden zum Trainieren künstlicher neuronaler Netze, um den Zustand und verschiedene Aktivitäten des Fahrers in einem Fahrzeug zu überwachen. Die Innenraumüberwachung und Fahrerbeobachtung wird mit steigender Automatisierung der Fahrzeuge immer relevanter. Anwendungen, die die Aufmerksamkeit des Fahrers überwachen oder Aktivitäten des Fahrers, seien dies Aktivitäten, die zu den Fahraufgaben oder anderen Aktivitäten gehören, erkennen, können die Sicherheit und den Komfort eines Fahrzeuges aller Automatisierungsstufen verbessern.

Um den Bereich des Fahrersitzes eines Autos zu überwachen, wurden Time-of-Flight Kameras verwendet, die in das Dachmodul mehrerer Testfahrzeuge eingebaut sind. Ein Vorteil dieser Kameras ist, dass die Bilder der verwendeten Kamera unabhängig von der äußeren Beleuchtung sind. Da es nur wenige öffentliche Datensätze zur Innenraumüberwachung von Fahrzeugen mit Time-of-Flight Kameras gibt, wurden sämtliche verwendete Daten in den Testfahrzeugen aufgenommen. Dies macht effiziente Trainingsmethoden für künstliche neuronale Netze, um von wenig Daten zu lernen, umso relevanter, da so der Aufwand der Generierung eines Datensatzes für diese Systeme reduziert werden kann.

Zuerst wird ein System zur Belegungserkennung des Fahrersitzes und zur Klassifikation des Zustandes des Fahrers beschrieben. Dafür wird eine hierarchische Multi-Label Struktur und Fehlerfunktion definiert, welche die Klassifikationsergebnisse des Systems robuster machten. Danach wird ein System vorgeschlagen, um Aktionen und Objektinteraktionen des Fahrers zu erkennen. Das Training und die Klassifikationsraten des zugrundeliegenden Netzwerkes werden durch eine zeitliche Augmentierung und die Reduzierung des Eingabe-Merkmalsraums durch Sequenzen von 3D Körper Keypoints und

Bildausschnitten von Händen verbessert. Folgend wird ein drittes System zur Erkennung von Aktionen des Fahrers in kontinuierlichen Videodaten vorgestellt, welches nur mit abgeschlossenen, kurzen Sequenzen von Aktionen trainiert wurde. Dafür werden Techniken zur Erstellung von künstlichen kontinuierlichen Beispielsequenzen von Aktionen des Fahrers erläutert. Darüber hinaus wird eine Trainingsmethode vorgeschlagen, mit derer Wissen des Netzwerkes aus vorherigen Trainingsschritten für aktuelle Trainingsschritte von Rekurrenten Neuronalen Netzwerken verwendet werden kann.

Abstract

The goal of this work is the development of new interior sensing systems based on artificial neural networks and Time-of-Flight image data. This development includes training methods to monitor the driver's states and activities inside a vehicle. Interior sensing and driver monitoring gets more and more relevant as vehicle automation increases. Applications that monitor the attention of the driver or detect activities related or not related to the driving tasks can enhance the safety of a vehicle of all stages of vehicle automation, as well as infotainment applications.

To monitor the driver's seat region of a car, Time-of-Flight cameras mounted in the roofs of several test cars are used. One advantage of these cameras is that the images of the used cameras are independent from ambient illumination. As only few datasets for interior sensing applications with Time-of-Flight sensor data are available, all used data was recorded in the test cars, which makes efficient training methods for artificial neural networks to learn from less training examples relevant, in order to reduce the effort of dataset generation for such systems.

First, a system to detect the driver's seat occupancy and driver's state from single images is described. For this, a hierarchical multi-label structure and loss is defined, which makes classifications more robust. Secondly, a system for driver action and object interaction recognition is suggested. The training and classification rates of the underlying classification network are enhanced by a systematic temporal augmentation technique and the reduction of the input feature space, by using 3D body keypoint and hand patch sequences. Following, a third system to classify actions of the driver in continuous video streams, while trained only on short isolated action sequences is developed. For this, techniques to systematically create artificial continuous action examples are proposed. Furthermore, a training method is suggested to integrate knowledge of previous training steps into the training of recurrent neural networks.

Contents

List of Abbreviations

Abbreviation	Description
Acc	Accuracy
AP	Average Precision
CNN	Convolutional Neural Network
CTC	Connectionist Temporal Classification Loss
FN	False Negative
FP	False Positive
GRU	Gated Recurrent Unit
IR	Infrared
KP	Key Point
LSTM	Long Short-Term Memory
mAP	Mean Average Precision
MLP	Multilayer Perceptron
PR	Precision Recall
RELU	Rectified Linear Unit
RNN	Recurrent Neural Network
SVM	Support Vector Machine
TN	True Negative
ToF	Time-of-Flight
TP	True Positive

CHAPTER 1

Introduction

Image based driver monitoring plays an increasingly important role in the automotive industry. On the one hand, applications like gesture control systems enhance the comfort of the car occupants as well as they increase their safety by introducing more intuitive and less distractive methods to interact with the infotainment system of vehicles. On the other hand, existing driver monitoring applications increase the occupant's safety by monitoring the awareness of the driver via head and eye tracking.
Studies show that distracted driving, especially phone usage, is still a growing risk factor in road traffic. Phone usage during driving increases the risk of a crash from four up to 23 times. Moreover, the reaction time of the driver using a phone while driving is about 50% slower than the reaction time of a no-distracted driver as reported by the authors of [Wor11], [Wor18].
In the near future, semi-automated car drivers will still be responsible for monitoring the environment and to intervene in the driving actions of the car, if necessary. Recognising different activities of the driver can be helpful to draw the attention of the driver back to the road. Nevertheless, the driver will be allowed to devote his attention to other things than the driving task, as the automation of the vehicles increases. However, in situations where the car needs to hand over control to the driver it is necessary to recognise if the driver is able to take over control. Even in semi-automated vehicles, the takeover time highly depends on the state of the driver as described by the authors of [Eri17]. Advanced driver monitoring systems can adaptively inform or warn the driver, based on the driver's current state and attentiveness to the driving task. Moreover, possibly unintended deactivation of the autopilot of a highly automated vehicles can be prevented, if the current activity or state of the driver is known. Thus, monitoring the state and activity of the driver remains relevant for highly automated vehicles.

1.1 Objective of the Thesis

The objective of this thesis is to develop efficient driver monitoring systems based on Time-of-Flight image data and artificial neural network models. Gathering data for training these deep learning based systems is often time consuming and costly, because many different example images or image sequences need to be recorded and annotated in order to train neural networks which generalize to unseen data examples. Moreover, the networks need to be computationally efficient, in order to be able to be integrated in products for the automotive industry, as computational resources need to be power efficient. Thus, small and efficient artificial neural network architectures need to be trained for these systems. Training these networks is therefore challenging as the networks must be small enough to run in real-time on embedded hardware, as well as the amount of available data is very limited, and generating more data is time consuming and costly.
This thesis proposes different driver monitoring applications which can enhance the safety and comfort of the driver and passengers of a vehicle. In any case, such systems are subject to certain previously mentioned conditions under which they can be efficiently developed and used. Therefore, this thesis addresses training methods for artificial neural networks, to increase the classification performance of driver monitoring systems.

1.2 Driver Monitoring Applications

In this thesis, three driver monitoring applications based on Time-of-Flight image data are described. For each application, new enhancements for training the underlying artificial neural networks are presented, evaluated and discussed. These methods include a loss for enhancing multi-label classification with hierarchical structures, input space feature reduction for action recognition, a systematic temporal augmentation technique for augmenting action sequences, as well as a method to efficiently train a continuous action classification system with closed isolated action sequence examples only. Gathering the datasets for training and validating the applications was part of the author's work.

The three presented applications are highly relevant driver monitoring applications covering current issues in the automotive industry. These applications are:

1. Occupancy and driver state classification
2. Driver action and object interaction recognition
3. Body movement detection of the driver

The first application deals with occupancy detection of the driver seat as well as recognising different driver states the driver can adopt while sitting in the driver's seat. This application is discussed in chapter 4. Detecting the occupancy of a driver's seat is a highly relevant issue insofar as near future semi-autonomous cars still need to be operated by a trained driver. Distinguishing between an adult, a child or objects present in the driver's seat is crucial for the safety of the vehicle. Moreover, the ability to detect the current state of the driver can be used to estimate the time the driver probably needs to take over control of an autonomously driving car. Furthermore, the knowledge about the driver's state can be used to draw the attention of the driver back to the road or prevent unintentional deactivation of autopilot functions. Gathering this knowledge about the occupancy of the driver's seat and the driver from camera images can further reduce the number of sensors in a car, like pressure sensors in the driver's seat or sensors for detecting if the driver has his hands on the steering wheel.

The second application, discussed in chapter 5, addresses the detection of different actions and object interactions of drivers from image sequences. Detecting actions like leaving the car or unstrapping the seat belt can be used to warn the driver, if possible dangers are detected by other sensors outside the vehicle, if detected early enough. Moreover, knowledge about objects and the way the driver is interacting with them can refine the understanding of the distraction of the driver. This can enhance predictions about the probable takeover time even more. Distinguishing between a driver just holding a smartphone or typing on a phone can be used to adaptively draw the attention of the driver back to the road, based on the object or the kind of interaction with an object.

The third system is discussed in chapter 6 and analyzes different body movements and positions of the driver. Knowledge about the movements of the driver can be crucial to understand if the driver is able to drive a car in his current seating position. Moreover, comfort features, like adaptively

controlling the in-cabin illumination, can be implemented based on body movements of the driver or other passengers.

1.3 Structure of the Work

This work is structured in seven chapters. After the introduction, the mathematical fundamentals relevant for this work are described in chapter 2. This includes descriptions about essential artificial neural network structures and training methods, as well as evaluation metrics for classifiers and image processing methods. Moreover, the hardware environment is described in this chapter.

Chapter 3 addresses related works that have significant influence on the developed concepts which are presented in this thesis. First, related papers are discussed which deal with driver monitoring in general. Furthermore, state of the art concepts for training artificial neural networks in general are presented as well as results that apply these concepts for driver monitoring for single images or image sequences. Additional references are provided at the beginning of the chapters 4, 5 and 6

In chapter 4 a concept for detecting the occupancy of the driver seat and different driver states is introduced. For this, a hierarchical label structure is presented to create a fallback system for classifications with low confidence or misclassifications of fine grained classes. A loss function was developed, to directly integrate this hierarchical structure into the training of a multi-label classification network.

An additional system for analyzing several actions a driver can perform is described in chapter 5. In this chapter, different strategies to train and optimize action recognition systems on image sequence data are discussed. These strategies include an augmentation strategy for systematically varying the pace of sequence examples. Moreover, different methods for reducing the input feature space are discussed and evaluated.

Chapter 6 covers the topic of training continuous action recognition networks on isolated action scenes to recognise the body movements of the driver. While some concepts to improve the efficiency of the training and classification performance of the networks, discussed in chapter 5, can be used, other optimization methods are not applicable. A concept is presented that can handle the internal hidden states of recurrent neural networks to

efficiently provide knowledge about class transitions present in continuous data while trained only on single isolated action examples without explicit class transition information.

Finally, chapter 7 summarises the presented driver monitoring systems and methods to optimize the training procedure, even if only few data examples are available. Furthermore, suggestions for further enhancements of the discussed systems and methods are given.

1.4 Main Contributions

The main contributions of this work are the development of the three artificial neural network based driver monitoring systems. This development includes the proposed training methods to enhance the robustness of the classification results while trained on limited data.

For the occupancy and driver state classification described in chapter 4, these contributions are:

- A classification system to detect various occupancy and driver states,
- More robust classifications through hierarchical classification,
- Integration of a fallback option for the driver state monitoring,
- Extension of existing multi-class classification concepts for multi-label classification.

The contributions which accompany the driver action and object interaction recognition described in chapter 5 are:

- Development of a more lightweight and flexible system to recognise drivers actions and object interaction from image sequences compared to currently available systems,
- An advanced augmentation technique to augment the time component of training sequences in a more structured way than existing methods,
- Reduction of the input feature space to 3D body keypoints and hand patches of the driver to focus on most important parts of the scene for this task,
- Further enhancements of the reduced feature space by rotation and size normalization of the hand patches.

The contributions of the body movement detection system described in chapter 6 are:

- A system to recognize driver body movement actions from image sequences,
- A systematical way to concatenate image sequences to simulate continuous data for the training phase of the system,
- A new method to integrate previously calculated hidden states of a recurrent neural network to enhance the training efficiency for training a continuous action recognition system only with short isolated action sequences.

CHAPTER 2

Fundamentals

The methods and systems presented in this work are based on different machine learning and image signal processing fundamentals. These technical fundamentals of the proposed methods are described in this chapter.
This chapter is structured as follows. First, an overview of the used deep learning concepts is given in section 2.1 including feedforward and recurrent network concepts as well as optimization methods to train the artificial neural networks. In chapter 2.2 the different metrics and methods to evaluate classification systems, which are used in this work, are described. Following, in chapter 2.3 the fundamentals to calculate optical flow images are outlined. Finally, the used hardware environment is described in chapter 2.4.

2.1 Deep Learning Fundamentals

Deep learning describes a domain of machine learning methods for training artificial neural networks with representation learning. Besides unsupervised and reinforced learning, supervised learning represents one of the main training methods in deep learning. The goal of the supervised learning methods is to autonomously learn features from examples and match predefined classes for classification tasks or approximate functions in regression tasks. This work focuses on the supervised classification tasks with artificial neural networks. The artificial neural networks consist of multiple artificial neurons arranged in different ways. The most common way to connect these neurons are in a feedforward, fully connected way, resulting in a multi-layer perceptron. For image classification, convolutional layers have proven to be highly effective and deliver state of the art results for image classification tasks. To analyse sequences, recurrent neural networks were developed to integrate a memory to the neural networks regarding previous calculations for the calcu-

lations of the current timestep. For training these network models, gradient descent with backpropagation is the preferred method since modern hardware is able to deliver the required computational power to calculate the operations of the networks in an acceptable time. State of the art networks are usually trained with a huge amount of data as described by the authors [Kri12]. Because such amount of data is often not available or too time consuming to gather and annotate, different methods for optimizing and regularizing training and augmenting example images were developed, in order to train artificial neural networks with less data while still generalizing to data examples which are not used in the training process.

2.1.1 Multi-Layer Perceptron

The multilayer perceptron, or fully connected network, is a feedforward network which originates from the perceptron presented in [Ros57] and comprises multiple perceptrons with continuous nonlinearities. The output of the fully connected network with one hidden layer z, displayed in figure 2.1, can be calculated by

$$y_k(\mathbf{x}, \mathbf{W}) = \sigma \left(\sum_{j=1}^{M} w_{kj}^{(2)} h \left(\sum_{i=1}^{N} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right), \tag{2.1}$$

with the network input vector $\mathbf{x}$ and the network weight matrix $\mathbf{W} = (\mathbf{W}^{(1)}, \mathbf{W}^{(2)})$. The function $h(\cdot)$ represents a nonlinear activation function which is usually choosen to be a sigmoid function

$$h_{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

or a rectified linear activation function (RELU)

$$h_{RELU}(x) = \begin{cases} x & \text{if} \quad x > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2.3}$$

Finally a nonlinear classification activation function ω is applied to the outputs. This activation function can again be a sigmoid function 2.2 or a softmax function

$$\sigma_{softmax}(\mathbf{y})_j = \frac{e^{y_j}}{\sum_i e^{y_i}} \tag{2.4}$$

to create pseudo probabilities of the output values.

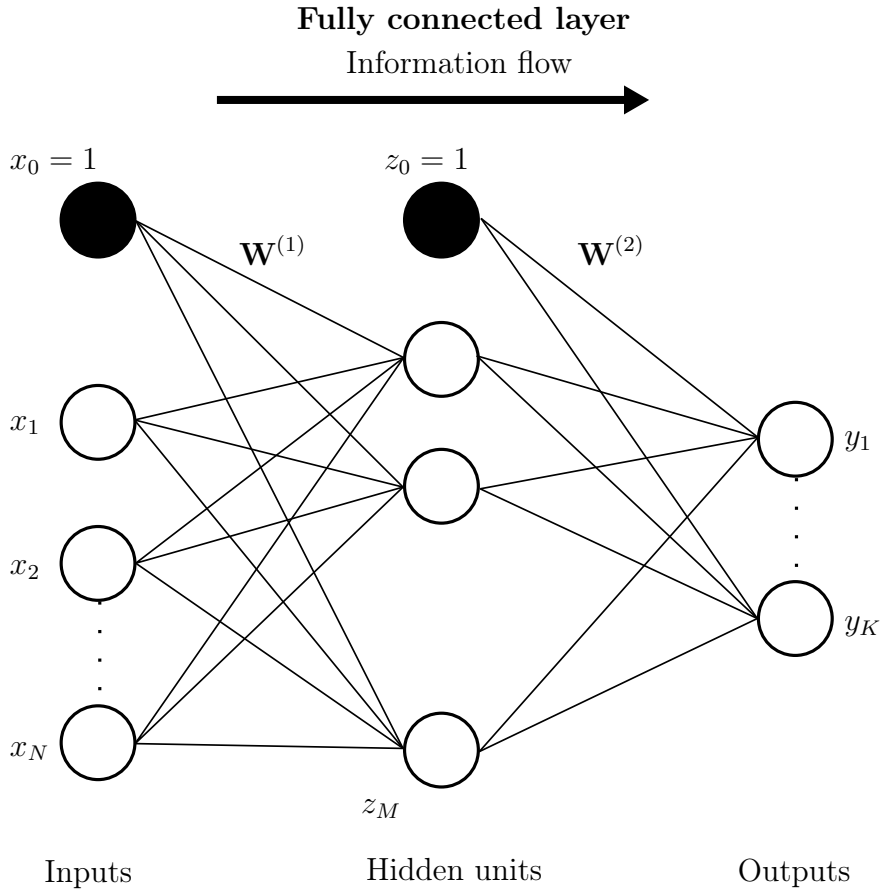


Figure 2.1: Fully connected layer with the input nodes x, one hidden layer with the hidden nodes z, the output layer with the nodes y and weight matrices $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$. The weights w_{0x} of the weight matrices $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ represent the bias of the network.

2.1.2 Convolutional Neural Networks

For Convolutional Neural Networks (CNNs) many different network architectures and concepts exist. A CNN for classification tasks usually consists of multiple convolutional layers, nonlinear activation functions, pooling layers and a fully connected network. A concept of a CNN with two convolutional layers, two pooling layers and a fully connected network for image classification is visualized in figure 2.2.

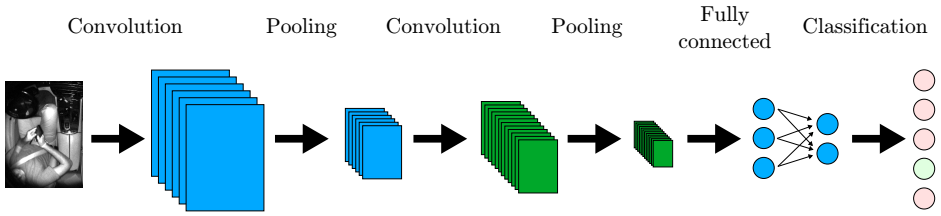


Figure 2.2: Common CNN architecture with fully connected layer to classify images.

Convolutional layer The principle of a convolutional layer and reusing weights of a neural network was first introduced by LeCun [LeC89] and is visualized in figure 2.3 and mathematically expressed by

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n). \tag{2.5}$$

This formula actually expresses a cross-correlation between the input matrix I and the convolution kernel K, but is still called a convolution in modern literature in the context of deep learning. However, as the weights of the kernel K are learned the flipping is unnecessary [Bis06], [Ian16].
In modern CNNs, a convolutional layer produces multiple convolutional operations in parallel resulting in one output matrix, often called feature map, per convolutional kernel in a convolutional layer. Afterwards a nonlinear activation function, like the activation functions described in chapter 2.1.1, are applied to the feature maps.

Pooling Pooling is another important computation block in CNNs. The pooling layer reduces the size of feature maps by applying a pooling function within a rectangular neighborhood of the feature maps. The most common pooling method is the max pooling introduced by [Zho88], where only the maximum value of the rectangular neighborhood is reported, resulting in a smaller feature map. This method helps the network to be invariant to small translations, which is necessary, as classifications often do not depend on small translations in the input.

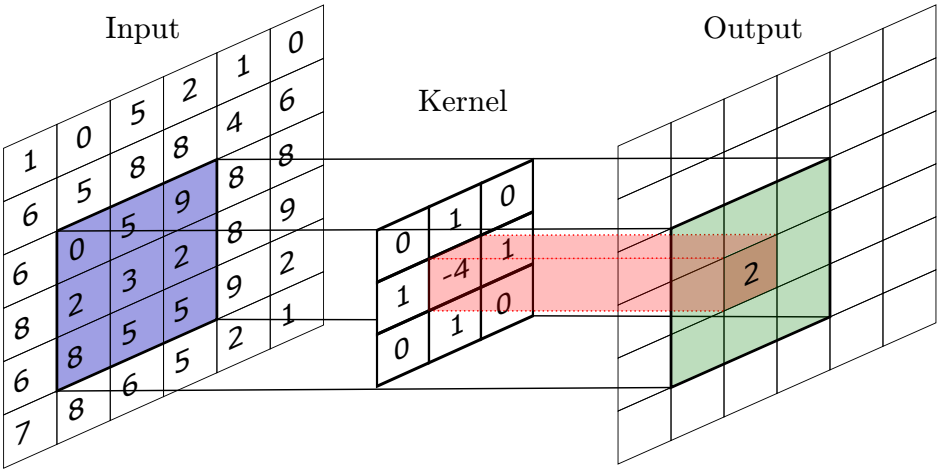


Figure 2.3: Two dimensional convolution

2.1.3 Recurrent Neural Networks

Recurrent neural networks (RNN) are a kind of neural networks that have connections to previous network layers, creating feedback loops. Moreover, with RNNs a network structure was introduced, which contains an internal memory. In contrast to feedforward neural networks, this memory enables the RNN to consider previous input examples at runtime.

Long-Short-Term-Memory

A commonly used variant of a RNN is the so-called *Long-Short-Term-Memory*-RNN (LSTM), which was introduced by the authors of [Hoc97]. The LSTM solved the vanishing gradient problem that occurs when training RNNs on long sequences. The error gradient decreases with every iteration it gets propagated backwards through the network, resulting in a gradient that has no valuable information left. The cell state, introduced by the authors, is strictly regulated, which prevents the vanishing of the gradient. A schematic visualization of an LSTM cell is shown in figure 2.4. A LSTM contains three gates which direct the information flow inside the LSTM cell. The output of the *Forget Gate*, the *Input Gate* and the *Output Gate* are calculated by

$$y_t^{forget} = \sigma(W_{xf} \cdot x_t + b_{xf} + W_{hf} \cdot h_{t-1} + b_{hf}), \tag{2.6}$$

$$y_t^{input} = \sigma(W_{xi} \cdot x_t + b_{xi} + W_{hi} \cdot h_{t-1} + b_{hi}), \tag{2.7}$$

$$y_t^{output} = \sigma(W_{xo} \cdot x_t + b_{xo} + W_{ho} \cdot h_{t-1} + b_{ho}), \tag{2.8}$$

with σ denoting the sigmoid function and $\odot$ denoting the Hadamard product.

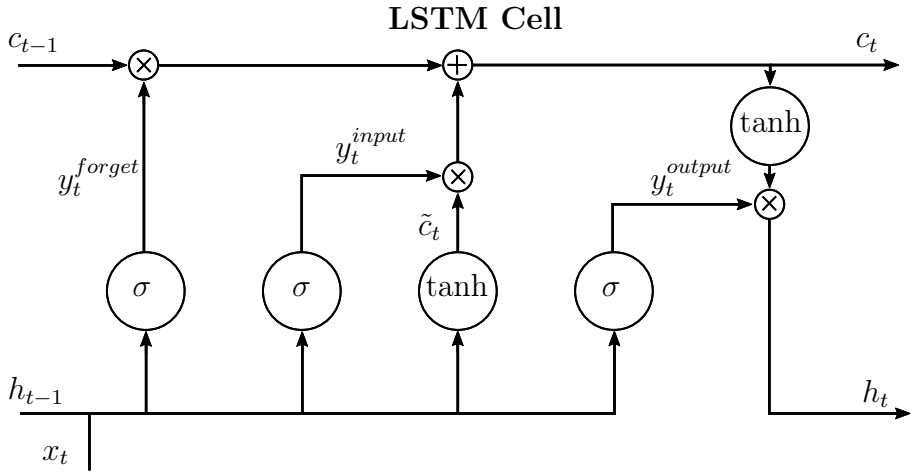


Figure 2.4: Structure of an LSTM cell

The output of the LSTM cell splits up in the cell state c_t and the hidden state h_t can which can finally be calculated by

$$c_t = y_t^{forget} \odot c_{t-1} + y_t^{input} \odot \tilde{c}_t \tag{2.9}$$

$$h_t = y_t^{output} \odot tanh(c_t), \tag{2.10}$$

$$\text{with } \tilde{c}_t = tanh(W_{xc} \cdot x_t + b_{xc} + W_{hc} \cdot h_{t-1} + b_{hc}). \tag{2.11}$$

Gated Recurrent Unit

A gated recurrent unit (GRU) is a variation of an LSTM, which was introduced by [Cho14]. The authors combined the *Forget Gate* and the *Input Gate* as well as the hidden state and the cell state, resulting in a more lightweight recurrent network structure with fewer parameters than the LSTM. The used gates in the GRU are called *Update Gate* z_t and *Reset Gate* r_t. The formulas for calculating the hidden state of a single hidden unit are described with

$$z_t = \sigma\left(W_{xz} \cdot x_t + W_{hz} \cdot h_{t-1}\right), \tag{2.12}$$

$$r_t = \sigma\left(W_{xr} \cdot x_t + W_{hr} \cdot h_{t-1}\right), \tag{2.13}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t, \tag{2.14}$$

$$\text{with } \tilde{h}_t = tanh(W_{xh} \cdot x_t + W_{hh} \cdot (r_t \odot h_{t-1}). \tag{2.15}$$

2.1.4 Optimization Through Backpropagation

In order to reduce a cost function E and update the weights of a neural network, the partial derivative of the cost function with regards to a set of weights $\mathbf{W} = (W^1, ..., W^l)$ is calculated. With the chain rule of calculus,

$$\frac{\delta E}{\delta w_{ij}^l} = \frac{\delta E}{\delta z_i^l} \frac{\delta z_i^l}{\delta w_{ij}^l} \tag{2.16}$$

$$\text{with } z_i^l = \sum_{j=1}^{m} w_{ij}^{l-1} a_j^{l-1} + w_{i0}^l \tag{2.17}$$

can be formulated with z^l as the input to layer l, the activation $a^l = h(z^l)$ and with m as the number of neurons in layer $l-1$.
Further,

$$\frac{\delta z_i^l}{\delta w_{ij}^l} = a_j^{l-1} \tag{2.18}$$

can be formulated calculating the derivative.
Finally,

$$\frac{\delta E}{\delta w_{ij}^l} = \frac{\delta E}{\delta z_i^l} a_k^{l-1} \tag{2.19}$$

denotes the final gradient.
This way, all partial derivatives of E with respect to the weights $\mathbf{W}$ can be calculated and an update step of the weights

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \eta \nabla E \tag{2.20}$$

with the step size η can be performed.

2.2 Metrics and Test Procedures to Evaluate Classification Systems

To evaluate a classifier multiple different metrics exist. Each metric evaluates the performance of a classifier differently. Thus, different metrics can be used to interpret the performance of the classifier differently. The classification results of a binary classifier, which distinguishes between a positive and a negative class can be displayed in a confusion matrix 2.1. This confusion matrix contains the correctly classified *True-Positive* (TP) examples, the mistakenly positive classified *False-Positive* (FP) examples, the mistakenly negative classified *False-Negative* (FN) examples and the correctly negative

Label \ Prediction	positive	negative
positive	TP	FN
negative	FP	TN

Table 2.1: Structure of a confusion matrix for a binary classifier

Label \ Prediction	positive	negative
positive	$\frac{TP}{TP+FN}$	$\frac{FN}{TP+FN}$
negative	$\frac{FP}{FP+TN}$	$\frac{TN}{FP+TN}$

Table 2.2: Structure of a confusion matrix with relative numbers for a binary classifier

classified *True-Negative* (TN) examples. An example of a binary confusion matrix with this absolute values is displayed in table 2.1.

The confusion matrix can be displayed with relative values instead of absolute values as well in order to better visualize the classification distribution. The entries of the relative confusion matrix are the values of the absolute confusion matrix normalized with all values of the related row as displayed by table 2.2.

The predictions of a multiclass classifier can be displayed similarly with the multiple different classes.

Accuracy

One commonly used metric to evaluate a classifier's performance is the accuracy. The accuracy

$$ACC = \frac{TP + TN}{TP + FP + FN + FP} \tag{2.21}$$

measures the portion of correct classified examples. For a multiclass classifier the accuracy is calculated by

$$ACC_{mult} = \frac{\text{Correctly classified examples}}{\text{Total number of examples}}. \tag{2.22}$$

As long as the different classes are evenly balanced, the accuracy score is a meaningful measure for the performance of a classifier. However, as soon as

the different classes are imbalanced, the accuracy score might give a misleading impression of the classifier's performance. To use the accuracy anyway, the class imbalance can be included in the accuracy score by calculating the mean of the class wise accuracy by

$$ACC_{balanced} = \frac{1}{|L|} \sum_{l \epsilon L} \frac{TP_l}{TP_l + FN_l} \tag{2.23}$$

with L as the set of classes.

Precision and Recall Metrics

Two commonly used error rates are the precision and the recall, which are both calculated per class. The precision

$$\text{precision}_l = \frac{TP_l}{TP_l + FP_l} \tag{2.24}$$

measures the rate of how many positive predicted examples are truly positive. The recall

$$\text{recall}_l = \frac{TP_l}{TP_l + FN_l} \tag{2.25}$$

measures the rate of how many positive examples of a class are correctly classified as this class.

When considering different thresholds for a class to count as positive class a precision recall curve can be drawn, showing the tradeoff between the precision and recall of the classifier's results depending on the different thresholds. An example of a precision recall-curve is shown in figure 2.5. The overall performance of the classifier is given by the area under the precision recall curve often called the average precision. A higher area under the curve means overall higher precision and recall scores. The performance of a classifier calculating random predictions about the classes is displayed by the dotted line. As the curve of a random classifier might look differently, the area under the curve is 0.5 for a binary classifier, predicting random results. A combined performance measure of the precision and the recall is given by the F1-score

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.26}$$

which calculates the harmonic mean of both measurements.

For a multiclass classifier for each class one precision recall curve can be calculated representing the evaluation of this class. In order to calculate the metrics by class, the evaluation needs to be done in a one versus all manner

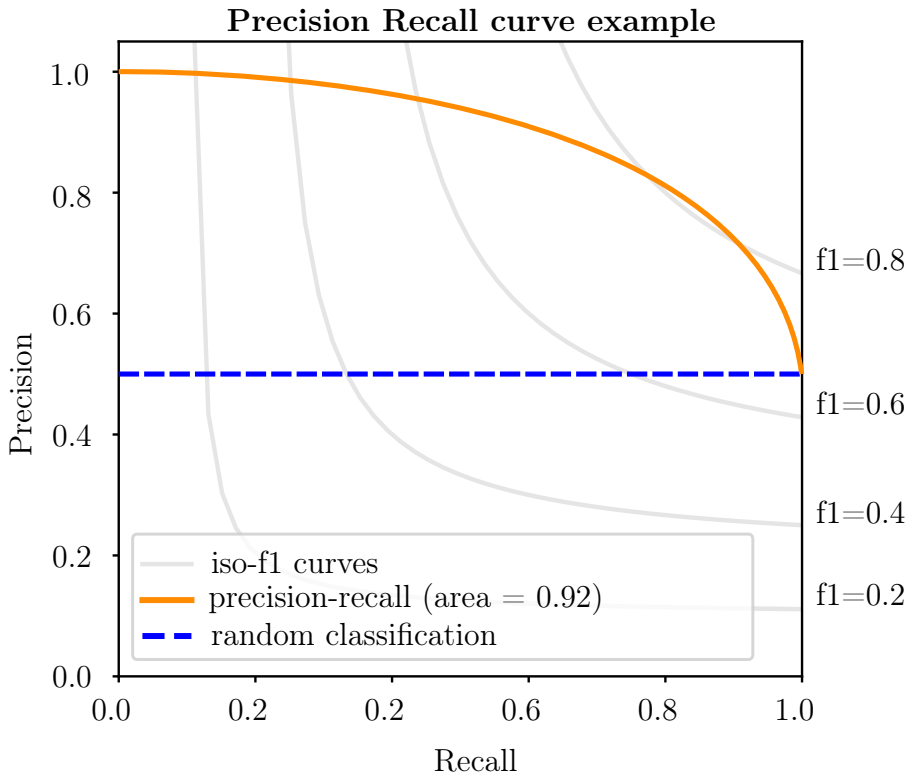


Figure 2.5: Precision recall curve of a binary classifier. The dashed line represents a precision recall curve of a binary classifier with random classification results. The orange line represents the precision recall curve of a trained classifier. Depending on the desired behaviour of the classifier to have a high precision, a high recall or a specific balance between both scores, a threshold can be determined to meet the desired behaviour of the classifier on the evaluated data.

for each class. The average precision of a class given by a randomly predicting multiclass classifier equals $\frac{1}{N}$, with N as the total number of classes. To calculate an overall measurement for the multiclass classifier, the measurements can be combined in different ways. One way to combine the measures for multiple classes is to calculate the micro-average of the scores. The micro-average sums the measurements and divides them by the total number of measurement, for example of the precision scores

$$precision_{micro} = \frac{\sum_{l \epsilon L} TP_l}{\sum_{l \epsilon L} TP_l + FN_l}. \tag{2.27}$$

A micro precision recall curve can be calculated as well, by quantifying the score on all classes jointly. However, this metric does not consider class imbalances of the data and could result in a skewed impression of the classifiers performance, if the classes are imbalanced. To equally weight the different classes the macro-average of the scores

$$precision_{macro} = \frac{1}{|L|} \sum_{l \epsilon L} precision_l \tag{2.28}$$

can be calculated by calculating the scores per class and average them over all classes instead of all examples.

2.2.1 N-Fold Cross Validation

To evaluate the performance of a classifier the classifier is ideally tested and evaluated on independent test data. This test data should not be used to train or validate the system and is processed by the system only once to get the evaluation metric scores. However, sometimes the dataset is too small to exclude a representative part of the data to be used as test data only. In these cases the system can be evaluated with a n-fold cross validation. To do this, the dataset needs to be split in n equal parts. The class distribution for each class should be the same for each split. The system is then trained n times, each time without one of the splits and evaluated on the left out split. Once the trained systems are evaluated, the mean of each left out evaluation score is computed to evaluate the overall performance of the classifiers.

2.3 Optical Flow

The optical flow of an image sequence describes the translational movement of objects between images. This translational movement can be caused by the object actually moving or by camera movements. If the camera has a fixed position, the translational movement described by the optical flow images results from actual object movement and not from camera movement.

One way to calculate the dense optical flow is the Farneback's method [Far03]. This method for calculating dense optical flow from two consecutive images approximates each pixel neighbourhood by a polynomial

$$f_1(x) = x^T A_1 x + b_1^T x + c_1 \tag{2.29}$$

and constructs a new signal by introducing a global displacement to the polynomial

$$f_2(x) = f_1(x - d) = x^T A_1 x + b_2^T x + c_2. \tag{2.30}$$

This displacement can further be calculated between two frames

$$d = -\frac{1}{2} A_1^{-1}(b_2 - b_1) \tag{2.31}$$

$$\text{with} \quad b_2 = b_1 - 2A_1 d \tag{2.32}$$

and equals the optical flow between those.

Figure 2.6 shows an example of the calculated optical flow of two images. It can be seen that the strongest movement happens in the vertical axis as

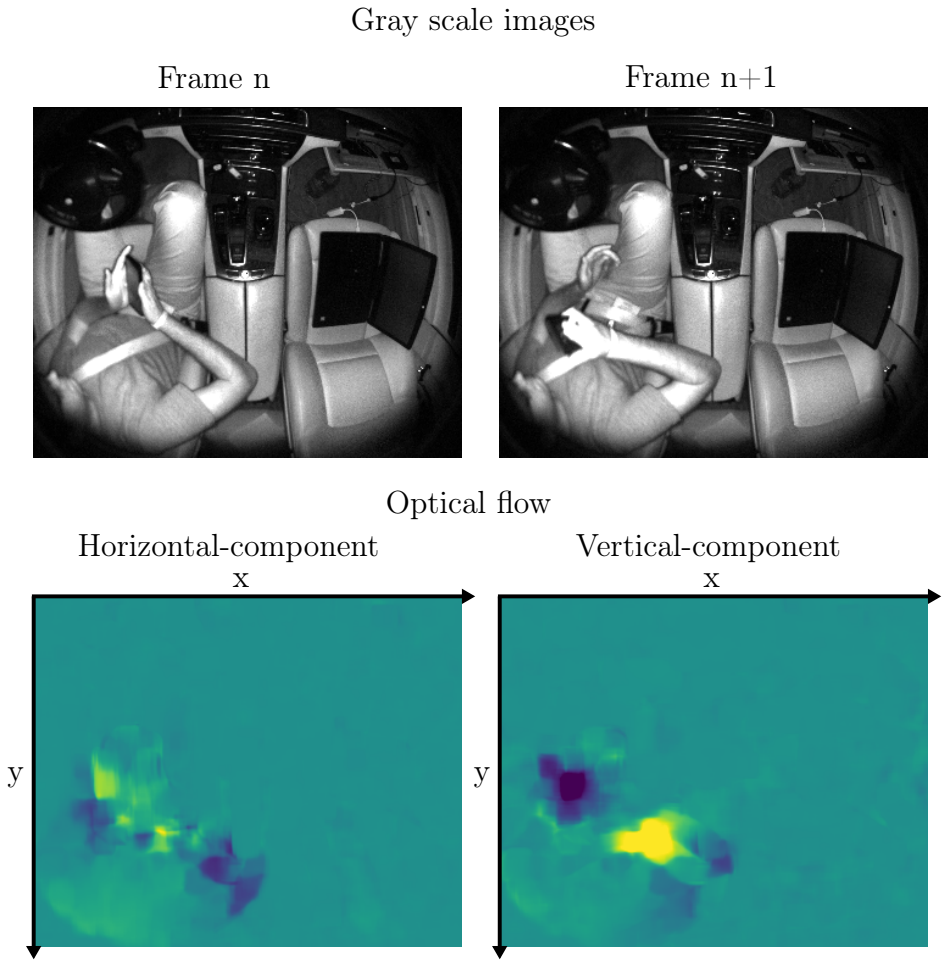


Figure 2.6: Example of optical flow images. Top: Original gray scale images. Bottom: Horizontal- and vertical-components of the optical flow calculated from the original grayscale images.

the person lifts his right arm to the head. This movement is captured by the vertical-component of the optical flow images, while less movement happens in the horizontal axis captured by the horizontal-component of the optical flow images. Around the person little to no movements happen resulting in near zero values in the optical flow images. A minimal movement is still detected in these areas as the image noise contributes to the optical flow calculation as well.

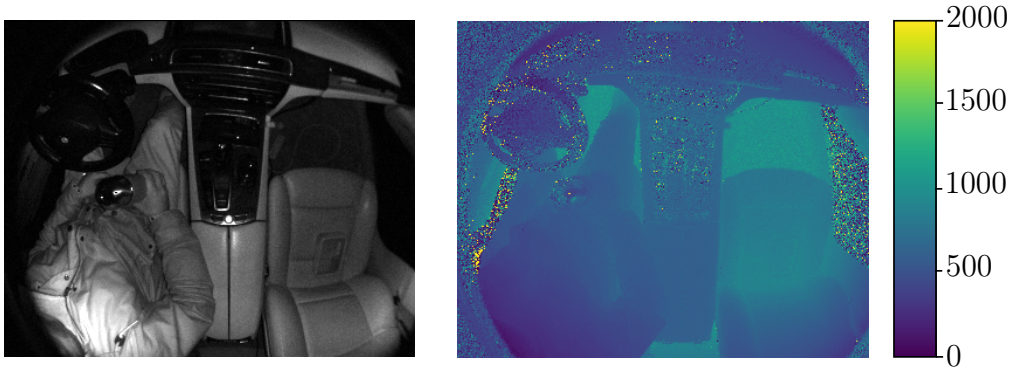


Figure 2.7: Time-of-Flight image data examples. Left: Gray scale image example. Right: Color encoded depth image example. Dark pixel represent shorter distances to the camera, light pixel represent larger distances to the camera.

2.4 Hardware and Environment

The camera used in this work to capture image data is a Time-of-Flight (ToF) camera, which is capable of capturing both a gray-scale image and a depth map of the scene. ToF cameras are widely used in several domains by now and have the advantage that they can capture images of a scene regardless of the ambient light. Following, the principle of calculating the distance from the camera to the environment which is employed by the used Time-of-Flight camera is outlined. Furthermore a short overview of the general environment in which the camera operates is given.

2.4.1 Time-of-Flight Depth Measurement

To calculate the distance from the camera to the environment a light emitting diode illuminates the environment with near infrared light (IR) pulses. The emitted light is reflected by the surface of the environment and measured by a sensor with photo diodes. With

$$d = \frac{t_D}{2} \cdot c \tag{2.33}$$

the distance to the surface can then be calculated based on the time the emitted light needs to travel to the surface and back to the sensor t_D and the speed of light $c = 3 \cdot 10^8\ m/s$.

However, the required hardware to emit and measure these IR light pulses is quite costly. A different approach to measure the distance of the environ-

ment is to measure the phase shift ϕ between the emitted and measured light of modulated light. This phase shift can further be used to calculate the distance with

$$d = \frac{c \cdot \varphi}{4 \cdot f_m \cdot \pi} \tag{2.34}$$

and the frequency of the modulated, emitted light f_m. The diodes emit light in the near infrared spectrum in order to be invisible for the human eye and therefore not being a source of disturbance for the driver. With additional measurements without the diodes illuminating the scene in between the ambient light emitted from other sources as the diodes of the camera, the measurements of the sensors resulting from the ambient light can be subtracted from the measurements with the diodes illuminating the scene. This results in measurements independent from ambient illumination, e.g. the illumination caused by daylight.
In addition to the depth image, the camera also generates a gray-scale image of the scene. Figure 2.7 displays an example of the two output images of the ToF camera. At the left side an IR amplitude image is shown, while on the right side the corresponding depth map is displayed.

2.4.2 Environment

The cameras are recording images of the front seats of several cars of different brands and models and are integrated in a special roof module which is located in the roof of the cars right in front of the rear mirror. Figure 2.8 shows an image of the front seats of one of the test cars. At the roof of the car, in front of the rear mirror, the camera location is highlighted. The camera is integrated in a way to monitor the front seats of the car from above. The example images of the ToF camera, displayed in figure 2.7, show both front seats, as well as the driver, the steering wheel, the dashboard and partly the doors of the car on the driver's side, and on the passenger side. The visibility of the driver's head depends on the height of the driver and is mostly not visible completely due to the location of the camera and the camera's angle of view. Depending on the car and the installation position, the images vary between the different cars not only that another car interior can be seen, but also that the rotation and translation of the interior visible in the images changes slightly. Figure 2.9 shows images of 4 different cars each with slightly different camera positioning. It can be seen that the camera angle differs for each of the four cars. Moreover, the visibility of the interior in the gray-scale image highly depends on the material and the color of the interior, resulting in very dark scenes for the second and the fourth example.

Figure 2.8: Example of the ToF camera location in one of the test cars. The camera is installed in the roof module of the car in such a way that it observes the front seats.

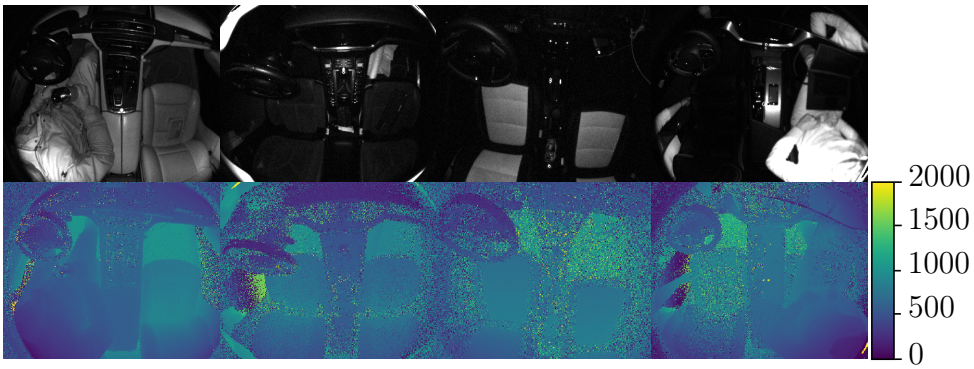


Figure 2.9: Differences of different cars and camera positions.

CHAPTER 3

Related Work

In recent years many driver monitoring systems were proposed for different kinds of applications. These applications are mainly safety applications to detect fatigue or rate the attentiveness of the driver. Besides these safety applications, methods to enhance the in-vehicle infotainment experience were proposed as well.
In this chapter, previous works related to driver monitoring and deep learning are described, on which the methods described in this work are based. First in 3.1 an overview of different driver monitoring applications is given. In 3.2 related deep learning approaches and contributions using deep learning approaches for interior sensing are presented. Finally an overview of related action recognition systems and action recognition systems for interior sensing are described in 3.3.

3.1 Driver Monitoring

To detect fatigue, papers like [Hor04], [Zha06], [Gan06], [Du08] analyse hand crafted features of the driver's eye states. These features are then further analysed to calculate the eye closure and blink rates, as they are essential indicators for these works. Later, learned features are commonly used to detect fatigue of the driver, like in the papers of [Li14], [Sha19], [Bor17], [Bor20]. These learned features are used for automatic head and eye localization, pose estimation and classification of the driver's state. Moreover the authors of [Fri18] analyse driver's eye pupils with a 3D-CNN to estimate the cognitive load and therewith the driver's overall performance.

Beside driver's fatigue, the mood of the driver is essential for the driving style, as described in [Abd16]. The authors therefore try to classify the driver's mood, based on video and audio recording of different drivers.

The papers of [Ohn14b], [Mol15b] analyse image sequences of hands to recognise different hand gestures in an in-vehicle environment.

Moreover, the papers of [Geb19] focuses on predicting the driver's intention by analysing image sequences of the driver.

Currently, one of the most important factors for the safety of a vehicle is the driver's attention as mentioned in chapter 1. Most works focus on detecting the driver's inattention by analysing the driver's pose and trying to detect object interactions with non driving related objects, like cellphones. For this, a variety of approaches exist. The authors of [Hoa16], [Yan16a], [Mas18], [Era19], [Mar19] use CNNs to classify whether the driver is attentive, or deals with other activities instead of driving. Most works, however, reduce the input feature space from full images of the driver to focus on specific attributes. Some papers, like [Cra15], [Yan16b], [Era19], [Xin19] segment the driver in camera images to reduce the feature space. Other papers focus on features like the head position, head rotation and gaze attributes [Liu15], hand features [Ohn13a], [Ohn13b], [Hoa16], or combine different head, hand and object features [Che07], [Ohn14a], [Era19]. Beside the mentioned features of a driver that can be used to classify the driver's state, some papers analyse the driver's body pose reduced to keypoints representing the locations of the driver's main body parts, like the head, shoulder, elbow and hand positions. The papers of [Dem09], [Alb18], [Tor19] focus on detecting these keypoints in an in-vehicle environment. While the work of [Mar18], [Mar19] uses only the driver's body keypoints and their relations, the paper of [Beh18] combines the keypoint locations with other features, while the work of [Era19] uses keypoints to cut out the driver's head and hands from the original image.

To analyse the driver's state and activities, the proposed methods rely on different kinds of sensor data. While some papers rely on RGB image data [Yan16a], [Beh18], [Mas18], [Geb19], [Geb19], some papers use illumination invariant image data like infrared images [Dem09], [Mol16], [Yan16a], [Liu16], [Mar19] or depth sensors [Dem09], [Li14], [Xu14], [Ohn14b], [Mol16], [Cra15], [Tor19], [Mar19] for driver monitoring applications.

3.2 Deep Learning for Image Classification and Single Image Interior Sensing

With the rise of deep learning in the recent years, new opportunities for all image related applications arose. By means of modern computer technology, larger and more complicated artificial neural network structures, such as the papers of [Kri12], [Sim14b], [Sze15], [Sze16b], [Sze16a], [He16], can be trained to analyse image data in even shorter time. Together with continuously growing datasets like [Kri12], [Lin14] and the possibility to store and process more and more data, deep learning applications were widely used and developed for many different tasks during recent years. This effectiveness on processing image data with CNNs has made deep learning driven applications an essential part for interior sensing and driver monitoring.
While the authors of [Yan16a] classify the driver's posture with a CNN from images showing the complete driver side of a vehicle from the passenger side, most papers addressing driver monitoring are based on preprocessing in-cabin image data.

Human Pose Estimation and Detection Networks

Beside classifying images, artificial neural networks can be used to estimate human poses and detect objects in images. To estimate human poses and detect different body parts in images with a trained neural network, different approaches exist. The authors of [Tos14] formulate the task as a CNN-based regression problem. First, a CNN extracts features from an image with a person. These features are then processed by a fully connected network to predict the x and y coordinates of different body parts of the person in the image. Further, these coordinates are used to crop out sub images from the original image, to refine the predicted coordinates with the same network. Other papers addressing human pose estimation also use CNNs to calculate features from images to predict the location of human joints in images, but calculate the final prediction from the feature maps without fully connected networks. The authors of [Wei16] train the CNN to generate a feature map for each body joint. Each detected body joint is represented by a $2D$ Gaussian peak at the position in the feature map, where the body joint is located in the original image. To address the vanishing gradient problem, which occurs when training neural networks with many layers, the authors train multiple stages which calculate refined feature maps based on the output of the previous stage, as well as the original image.
These techniques are further used to detect body joints of drivers in an in-

vehicle environment. The authors of [Alb18] use a motion tracking system, to generate 3D training data for human pose estimation in an in-vehicle mock-up environment. These 3D points are used to train a convolutional neural network with a fully connected layer to predict the 3D positions of the driver's joints from depth images. The authors of [Tor19] also use depth images to locate different driver's body parts. First, a VGG network ([Sim14b]) extracts features from depth images. Following, three different CNN based branches calculate heat maps for localising the body parts, part affinity fields to refine the predictions and the visibility of different joints. As human pose estimation is a very strongly investigated area, most driver monitoring systems detecting human body parts rely on pre-trained human pose estimation networks to detect different body parts in in-vehicle environments such as the papers proposed by [Beh18], [Era19], [Mar19].

Detecting different objects in images differs from localizing body parts, as object detection requires not only the position of the object in the images, but the spatial boundaries in the image plane as well. Therefore, object detection networks like the Faster-RCNN presented by the authors of [Ren15] propose the location, the height and width of objects in images. While the Faster-RCNN detects and classifies objects in images in a two step network architecture, the authors of [Red17] use a CNN based network to do bounding box regression and object classification at the same time. Moreover, the authors create a hierarchical label structure with mutually exclusive classes for each hierarchy level for the classification task, to create a fallback mechanism for the network. To avoid misclassifications for fine-grained classes, the next higher class, relative to the hierarchy, can be predicted if the network is uncertain about the classification of an object.

Using this kind of region proposal networks, the authors of [Hoa16] use a semantic detection network to detect the driver's hands, face and the car's steering wheel in two different camera images. The intersection of a detected hand and the steering wheel is used to classify, if this hand is located on the steering wheel or not. Moreover, the intersection between a hand and a face is used as a measurement to decide if the driver is taking a phone call or not.

Combined Approaches

The paper of [Yan16b] uses a similar approach for detecting more general driver actions. The authors introduce a method for driver activity classification by replacing the generic region algorithms with a Gaussian mixture model for skin segmentation. This results in an approach, which has, through the skin segmentation, access to additional information about parts of the driver's body. The approach described by [Xin19] also uses a Gaussian mix-

ture model for skin segmentation to separate pixels related to the driver from the background. The so obtained foreground image of the driver is then processed by various CNN architectures, like an AlexNet [Kri12], a GoogLeNet [Sze15] and a ResNet [He16] to distinguish between driver activities like normal driving, mirror checking or texting on a smartphone. The paper published by [Era19] uses an ensemble of neural networks to detect different distractions, like talking to a passenger, phone usage or drinking. Besides the full image, the authors use detection networks for the driver's hands and face, to create hand and face cut outs of the driver's image as well as a skin segmentation algorithm to separate the skin related pixels from the non skin related pixels, resulting in five images showing the full scene, the driver's face, the driver's hands and the driver's skin respectively. A CNN is trained to classify each kind of input image and the input image combination of the hand and face images. Different CNN architectures, like an AlexNet [Kri12], a VGG [Sim14b], an InceptionNet [Sze16b], and a ResNet [He16], are tested for classifying the different inputs. Finally the weighted ensemble of the different classifiers predicts the class probabilities.

3.3 Action Recognition and Driver Activity Recognition

Action recognition datasets

Beside image recognition, deep learning applications are also commonly used to recognise actions in image sequences. Similarly to the image recognition datasets, the number of datasets for action recognition is growing. While most action recognition datasets originate from movies or YouTube videos [Kue11], [Abu16], [Car17], some datasets are specially recorded for the task of action recognition in a specific environment [Sha16], [Mar19]. The authors of [Mar19] gathered a dataset for the specific task of driver's action recognition in an in-vehicle environment, containing multiple typical actions performed by a driver with multiple persons as drivers. The dataset consists of short three second clips of the driver recorded with a RGB camera, a depth camera and multiple IR cameras from different viewpoints, as well as calculated body keypoints of the driver. Additionally, the authors test several state of the art action recognition networks, pre-trained on related data, on their dataset.
Although many different datasets for action recognition exist and the datasets contain many different video sequences, the sequences are mostly distributed between many different classes, resulting in notably less examples per class than available examples for single image recognition tasks. This factor and

the circumstance that gathering and annotating action recognition data is time consuming complicates deep learning based action recognition tasks.

3D-CNNs

The papers of [Tra15], [Ji12] extend the concept of a CNN with an additional dimension, creating 3D-CNNs, to cover the additional temporal dimension in image sequences. Later, [Car17] enhances the idea of the 3D-CNNs by inflating an InceptionNet with batch normalization ([Iof15]) to a 3D-CNN. The concept of 3D-CNNs was picked up by the authors of [Mol15a] to recognise hand gestures and was further developed to recognize hand gestures in an in vehicle environment from RGB, depth and radar images in [Mol15b]. Image streams of each sensor type are recorded, as soon as the radar recognises a certain amount of movement directly in front of the sensors. Once the sensor does not recognise any movements, the recorded images are stacked per timestep. As the network requires a fixed length input, the stream length is re-sampled with nearest-neighbor interpolation. Finally a 3D-CNN calculates features from the image sequence which are then classified by a fully connected network.

Action Recognition with Optical Flow Images

A different way to recognise actions from image sequences is to calculate optical flow images from the image sequences to encode the movement information of multiple images in single images. The approach described by [Sim14a] introduces a two stream CNN network for action recognition, splitting up the task in a spatial and a temporal component. While the spatial stream performs action recognition with a CNN on single images of the sequence, the temporal part performs action recognition with a second CNN on multiple optical flow images, calculated from the images of the sequence. The classification results of both streams are then fused to get a final classification for the sequence.

Recurrent Neural Networks

Another concept for action recognition from image sequence data is to extract features from images and analyse these features with recurrent neural networks, like the authors of the papers of [Don15], [Yue15]. In their papers, features from single images of the sequences are computed by CNNs. These extracted features from each timestep of the sequence are then processed by a RNN to classify the frames of the sequence by considering the previously seen frames.

A similar approach is introduced by the authors of [Mol16], extending their previously described approach for driver's hand gesture recognition ([Mol15b]). Instead of classifying large blocks of image sequences each showing one hand gesture performed by a driver with a 3D-CNN, the authors now extract features of smaller image sequence blocks which are further processed by a RNN. To detect the different instances of a single action, a connectionist temporal classification loss (CTC) is used to prevent the network from recognising a single action multiple times.

Skeleton Based Action Recognition

While most papers, which aim to recognise actions with neural networks, rely on directly analysing image data, some works reduce the input features space for the networks by using skeleton data or object positions calculated from the frames of the image sequences. The approach of [Ché15] contains such a detection network for human body poses. This detection network calculates the main body keypoints of humans in the image sequences. These body keypoints are then used to generate cut outs of the full body and important parts of the visible humans, like the hands and the upper body, resulting in multiple sub images per frame. Additionally, the optical flow is calculated between consecutive images of a sequence. Similarly to the approach of [Sim14a], the sub images are processed in two streams, one spatial stream and one temporal stream. The concept of using the human pose described by body keypoints for action recognition is further used by [Liu16], [Sha16], which use the generated skeleton data to classify action with LSTM networks. The authors of [Das18] extend the approach of classifying human actions with skeleton data processed by an LSTM network with image cut outs of important body parts. The image cut outs are processed separately to the skeleton data and fused with the classification results of the LSTM network for each timestep of the sequence. The approach presented by [S K19] combines skeleton data with the results of an object detection network to detect human object interactions.
The concept of using human pose information to detect secondary tasks performed by car drivers while driving is described by [Mar18]. The authors trained a three stream LSTM network on 3D skeleton data of the driver to distinguish between classes like drinking, phone call, phone use or driving. The 3D skeleton data was pre-calculated by triangulating $2D$ skeleton information, generated by a human pose estimate network from multiple viewpoints. The three streams consist of a temporal stream, analysing the driver's movements over time, a spatial stream, analysing the spatial relation of the different body parts in a fixed time frame, and a context stream,

analysing the relations of the driver's body parts with parts of the car's interior. The approach of [Beh18] aims to recognise different actions with an in-vehicle environment with skeleton data, object positions and features calculated from the original images. The authors use a network trained for human pose estimation and object detection to generate skeleton data of the driver and localize different objects in RGB images showing the driver from the passenger side of the car. Between the different skeleton parts and the object locations, contextual descriptors between two elements are calculated, like the distance and the orientation to one another in the $2D$ plane. Additionally, a pre-trained VGG network ([Sim14b]) extracts features from the RGB images. A multi stream LSTM network analyses the skeleton data, the object location data, the contextual data and the features generated by the VGG and is finally classified by a softmax classifier for each frame.

CHAPTER 4

Improving Interior Sensing and Driver Monitoring with Hierarchical Classification

Driver monitoring applications need to be robust in order to meet automotive safety requirements. If safety critical systems misjudge the situation, a critical situation might not be detected or detected too late. Even non critical situations can become critical if a safety system misjudges the current circumstances beyond a critical point. Therefore, these systems need fallback strategies to weaken the outcome of misclassifications. Moreover, datasets containing examples for specific tasks are usually not available and need to be gathered during the development of the system. Collecting the necessary data is usually time-consuming and expensive. Therefore, it is advantageous if such systems can be trained on less data than usually required for deep learning based systems while still resulting in a robust solution meeting the performance requirements.
Hierarchical classification of the driver seat region can enhance the robustness of driver monitoring systems as well as making the training process more efficient with less data. The hierarchy integrates a natural fallback option for classes with low classification confidence to the system. Moreover, classes in higher layers of the hierarchy can be trained with more data examples compared to a flat label approach.
Most methods for hierarchical classification suggest applying weights to the loss function of a classifier as proposed by the authors of [Bin09], [Den10], [Den11], [Cha15]. These papers show that, considering the taxonomies of classes, by integrating the distance of misclassifications to the correct class in the hierarchical class tree into the loss function, the performance of SVM classifiers can be improved. However, these concepts are only applicable for multi-class classification tasks to increase the loss for misclassifications depending on the distance to the correct class. Multi-label classification tasks

with explicit classifications for each branch junction, as suggested by the authors of [Cai04], would not benefit from this approach, as parts of the classifier would be penalized during training which has no influence on the part of the network which made a false decision in a higher layer of the classification tree.
The loss functions, introduced in those papers, are mainly based on the 01-loss to train support vector machines. This loss is however not suitable for training an artificial neural network, as no gradients, which are required to adjust the weights of a neural network, can be computed by replacing correct classification with a loss of 0 and an incorrect classification with a loss of 1.
A commonly used approach for hierarchical classification with artificial neural networks is to construct a network adjusted to the hierarchical structure of the labels. The authors of [Cer14] use an MLP for each layer of the hierarchy for multi-label hierarchical classification. The MLP's output of a certain hierarchical level is the input to the following level's MLP, creating a stacked neural network with an MLP for each level of the hierarchy. This paper is extended by [Weh18] with an additional global classifier, classifying all classes at once. The results of the local and global predictions are then fused to generate a final prediction.
The paper proposed by [Li16] aims to classify human attributes as the gender and the clothing style from images. The authors use a detection network to find regions of interest in images. These regions are hierarchically separated in categories like a person, the head of a person, a group of persons or the whole scene. Several different classification networks then calculate predictions from the hierarchical ordered regions for the attributes, which are fused to generate a prediction about the different attributes. The different stages can bring different levels of context to the classifiers, supporting the final classification.
A different approach is used by the authors of [Red17] to hierarchically classify images. The classification part of the network calculates probabilities with a softmax function for each set of hyponyms of a parental node in the hierarchy. Hyponyms are more finely distinguishable subclasses of more general parental class. The probabilities along a path of the hierarchical tree are multiplied to create a conditional prediction for each class node. During training, the contribution of non-active layers of the tree to the loss are set to zero, since these branches do not provide any relevant information for the training.

Occupancy Classes	
Empty	Empty driver's seat
Object	Driver's seat is occupied by an object
Person	Driver's seat is occupied by a person

Driver's State Classes	
In Position	The driver is sitting in a driving position
Out of Position	The driver has taken a position in which he cannot drive normally
Hand on Wheel	The driver has at least on hand on the steering wheel
Hands off Wheel	The driver has no hand on the steering wheel
Object Interaction	The driver interacts with an object
no Object Interaction	The driver is not interacting with an object

Table 4.1: Classes for the hierarchical occupancy and driver's state classification

The presented method extends the paper of [Red17] by integrating a multi-label advancement to the hierarchical loss function for training a driver state monitoring system. The proposed system and methods were published by the author of this work in [Wey18].

This chapter is structured as follows. First, an overview of the gathered dataset and its hierarchical structure is given in chapter 4.1. Following, a method for training a neural network with hierarchical multi-label data is presented in chapter 4.2. An overview of the neural network used to classify the different driver states is described in chapter 4.3. The proposed method is evaluated in chapter 4.4. Finally, a summary and conclusion of the proposed system is given in chapter 4.5.

4.1 Hierarchical Occupancy and Driver State Data

The dataset for classifying the occupancy of the driver's seat and different states of the driver consists of multiple images recorded in multiple different cars with different persons and objects. Each example image shows the driver's seat region of a car in different occupancy states and, if occupied by a person, with several states of the driver. The examples were annotated manually and are composed of an amplitude image and a depth image each. Gathering the dataset, including recording and annotating the data, was part of the presented work.

The dataset consists of examples showing the different categories used for the occupancy classification and driver state monitoring. The different classes are listed in table 4.1. The dataset subdivides into the occupancy categories *Empty* for an empty seat, an object laying on driver's seat (*Object*) and a person sitting in driver's seat (*Person*). The driver's state classes are composed of examples of the classes of a person sitting in the driver's seat in a normal driving position (*In Position*) and a non-driving position (*Out of*

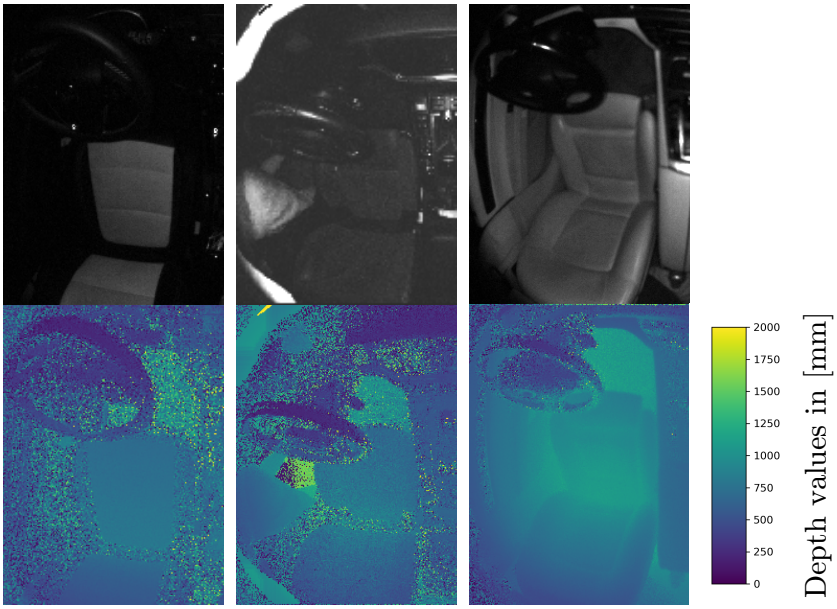
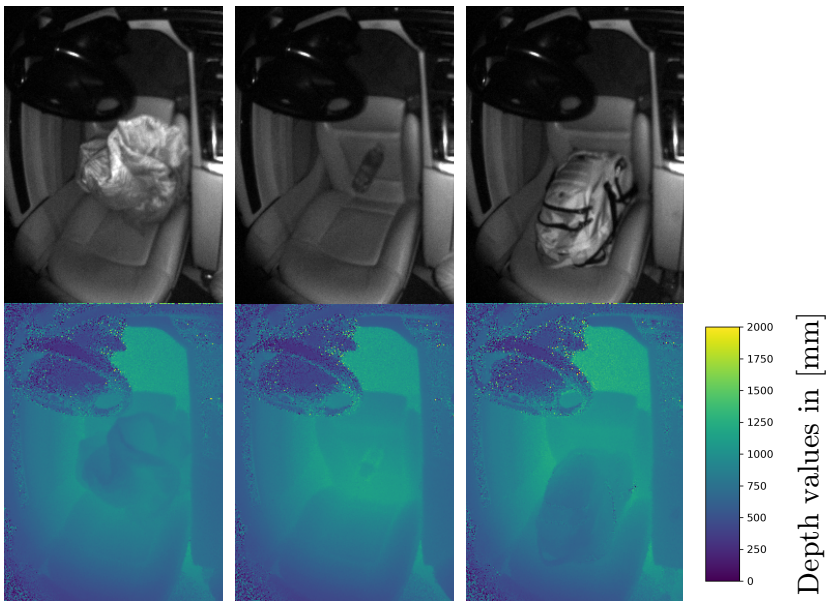


Figure 4.1: Examples of empty driver seats. Top row: amplitude images. Bottom row: Depth images with the corresponding color bar showing the color to distance coding in mm.

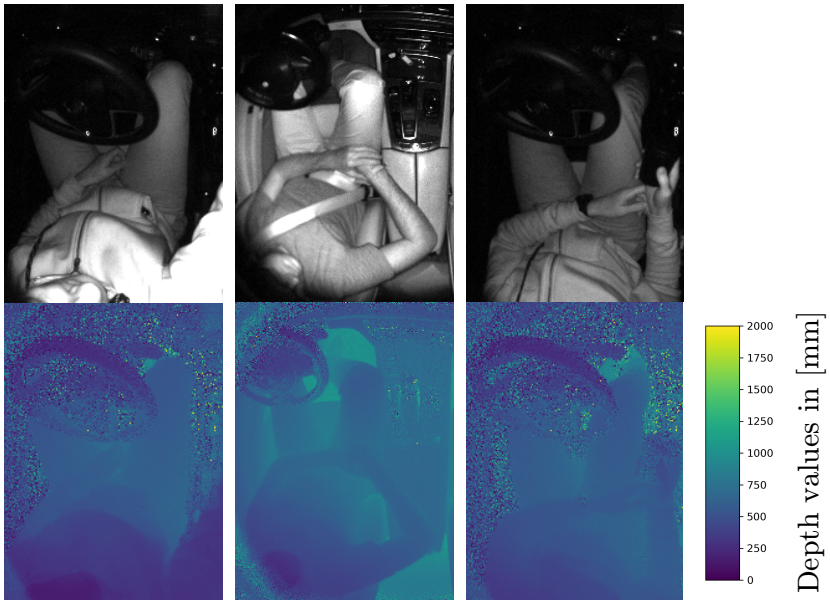
Position), a person sitting in driver's seat holding the steering wheel (*Hands On Wheel*) or not holding the steering wheel (*Hands off Wheel*) and a person sitting in the driver's seat interacting with an object (*Object Interaction*) or not interacting with an object (*No Object Interaction*).

Figure 4.1 shows examples of empty driver seats recorded in different cars. It can be seen, that the camera angle differs depending on the car. Moreover, the interior of the cars can be recognised differently well in the amplitude images as well as in the depth images, depending on the reflectance of the different materials of the interior. For example, the driver's seat is easier to recognize in the first amplitude image than in the second, while it is the other way around for the depth images. Moreover, in the first two depth images the image noise is clearly more visible compared to the third depth image.

Figure 4.2 shows two different categories of the driver's seat occupancy. In figure 4.2a the driver seat is occupied by different objects. The first image shows a wadded jacket. While the jacket is clearly visible in the amplitude image, it might not be recognised as a jacket in the depth image. However, it can be seen from the depth image that the driver seat is not empty. The same applies for the third image tuple, showing a driver seat with a backpack on it. While the backpack is clearly visible in the amplitude image, the depth image only gives a rough idea of what object is located in the driver's seat. The second image tuple shows a bottle laying in the driver seat. The bottle can be recognised as an object in the amplitude image. In the depth image the bottle is hard to recognise as the bottle is quite small and the depth values differ very little from those of the driver's seat. Figure 4.2b shows images where persons are located in the driver's seats. The persons are clearly recognisable in the amplitude and depth images.



(a) Examples of objects on the driver seats.



(b) Examples of persons on the driver seats.

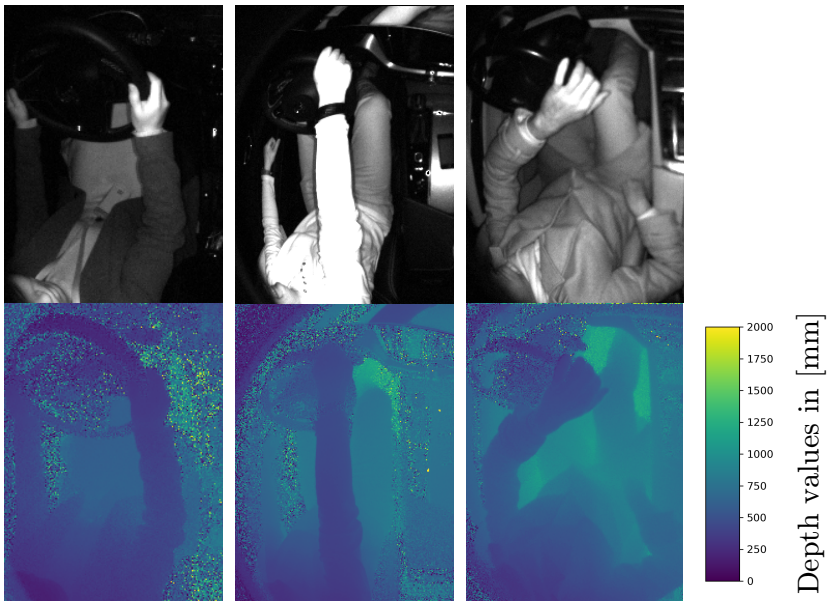
Figure 4.2: Examples of occupied driver seats

While the preceding examples show the different occupancy possibilities, the driver's seat, figure 4.3 and figure 4.4 show examples of the different driver states available in the dataset.

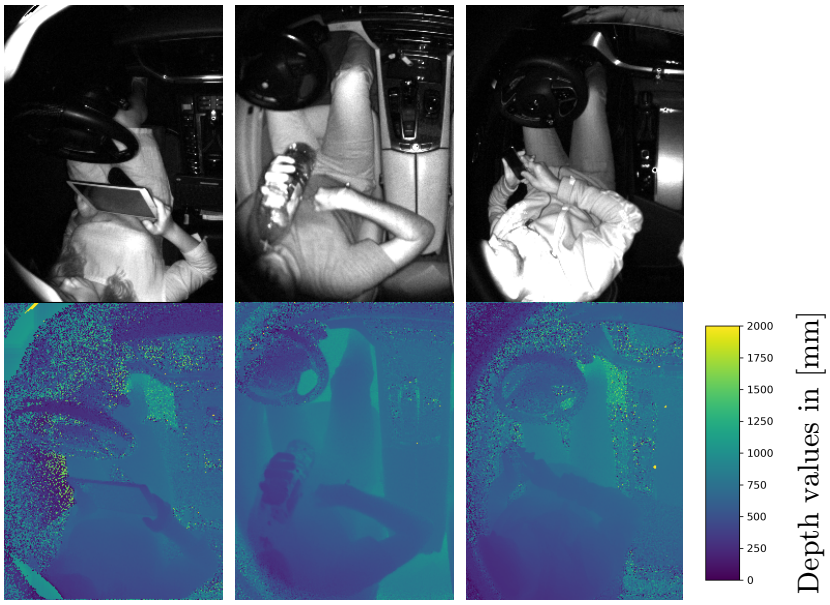
Figure 4.3a shows examples of the drivers holding the steering wheel of the car with at least one hand. While both arms of the drivers can be recognised easily in the amplitude images as well as in the depth images of the first and third image tuples, the driver's left arm is harder to localise in the depth image of the second example. This arm is clearly visible in the amplitude image as it stands out clearly from the background. In contrast, the arm nearly mixes with the background as it is located very close to the driver's door and quite thin, due to the distance to the camera, compared to the other arm in the depth image.

The second state that the driver can adopt is when the driver holds or interacts with an object such as a bottle or a smartphone. Figure 4.3b shows three examples of a driver interacting with different objects. While the objects are clearly visible in the amplitude image, only the tablet and the bottle shown in the first two examples can be recognised in the depth images. The smartphone the driver is holding in the third example cannot precisely be identified from the depth images as the depth values of the driver's hands and the smartphone mix. Merely the small area of the smartphone's surface shows a different structure than the driver's hands and might be an indicator to distinguish between the hands and the smartphone in this example.

The third state a driver can adopt in the dataset shows the driver in a non-common seating position where he might not be able to drive the car. Figure 4.3c shows examples of three non-common driving positions. The first example shows the driver leaning toward the glove compartment at the passenger side. In the second example, the driver leans towards the rear seats and in the third example, the driver leans toward the driver's side door. The driver and his positions in the driver's seat are clearly visible in the amplitude and depth images. These states of the driver are not mutually exclusive and can occur in all possible combinations.

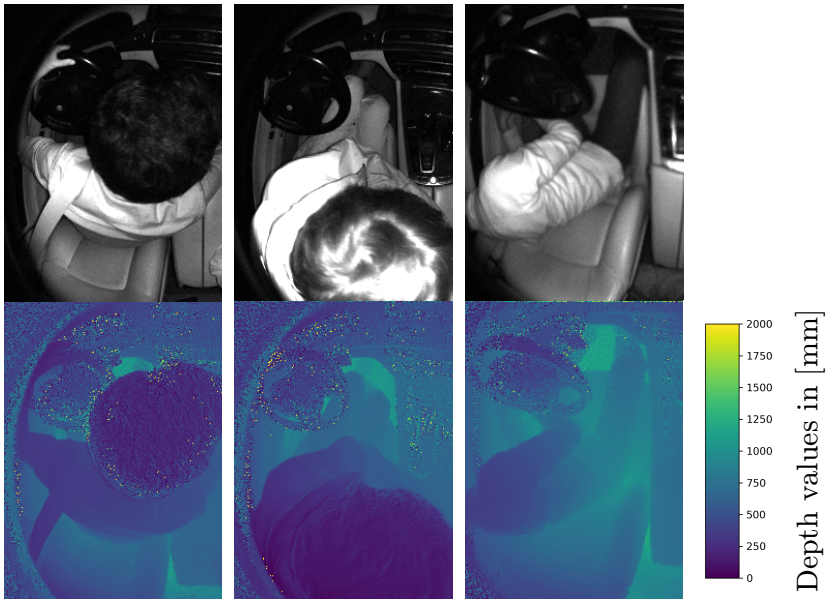


(a) Examples of persons sitting in the driver's seats holding with at least one hand the steering wheel.



(b) Examples of persons sitting in the drivers seat interacting with an object.

Figure 4.3: Examples of categories for different driver states.



(c) Examples of persons sitting in the drivers seat in a non-driving position.

Figure 4.3: Examples of categories for different driver states.

Figure 4.4 shows examples of the possible state combinations. The first example shows the driver holding the steering wheel with the left hand and a smartphone with the right hand. Because of the reflecting surface of the smartphone, it can be distinguished clearly from the hand in the depth image, unlike in the example shown in figure 4.2a. The second example shows again the driver holding the steering wheel with the left hand. This time, the driver additionally leans towards the rear seats. The third combination, the driver holding an object and seated in a non-common position, is shown in the third example. All three states the driver can adopt are simultaneously shown in the fourth example. The driver holds the steering wheel with the left hand while holding a smartphone with the right hand and leaning towards the rear seats. State combinations with other objects, different non-common seating positions of the driver and variations of the hands holding the steering wheel are present in the dataset.

The classes, as defined in this dataset, can be structured in a hierarchical way. This hierarchy is displayed in figure 4.5. The top layer of the hierarchy is defined by the occupancy of the driver's seat and can either be empty or occupied. The next hierarchical layer is defined by what occupies the driver's seat. In the presented dataset, this can be an object or a person. The third layer is defined by the different drivers states.

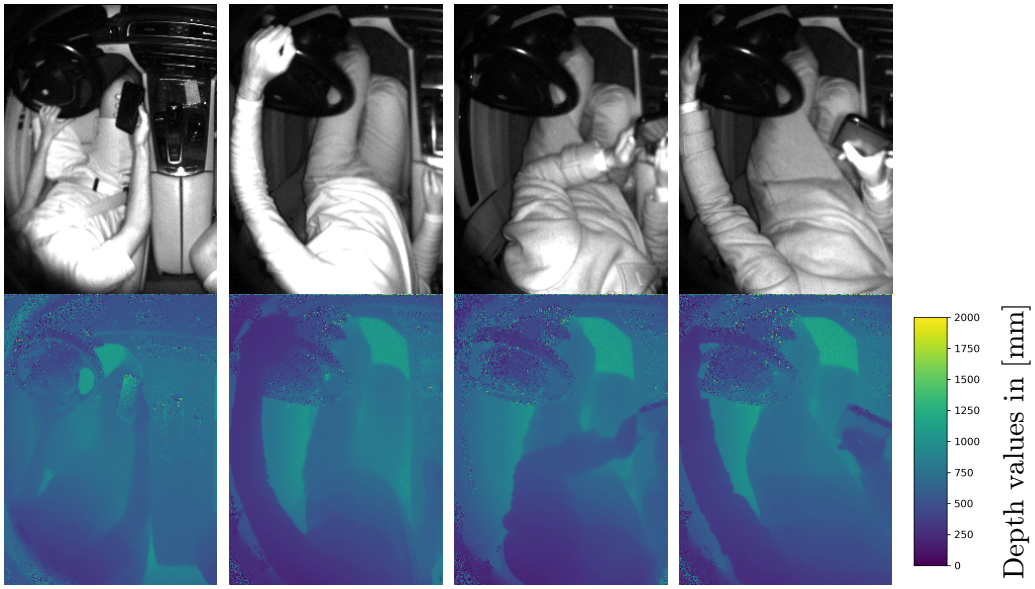


Figure 4.4: Examples of the driver in multiple states at once

The dataset, which was used to evaluate the published paper in [Wey18], was extended by additional images. Moreover, the overall number of images was reduced, by sorting out images which were too similar to each other, resulting in an advanced dataset with less redundant image examples. Nevertheless, the class imbalance of the dataset remains because the data is structured hierarchically, which entails a natural class imbalance between classes in higher layers of the hierarchy compared to classes in lower layers of the hierarchy. Figure 4.6 shows the class distribution of the dataset. The hierarchically induced class imbalance occurs because of the fact that every example of a layer in the hierarchical tree adds to the parental class as well. The second factor contributing to the class imbalance is that some classes do not benefit from more examples and collecting valuable examples of these classes is too time consuming compared to the benefit gained from more data for these classes. For color images more examples could help the system to adapt to different conditions as different lightning. However, for ToF data the ambient light is subtracted in the imaging process and therefore the data is to a large extend independent from external lightning conditions. For example, the class *Empty* was recorded in multiple cars with different driver seat and steering wheel positions. Adding more examples of this class would add images to the dataset very similar to already existing examples. For color images different lightning conditions could enhance the variation

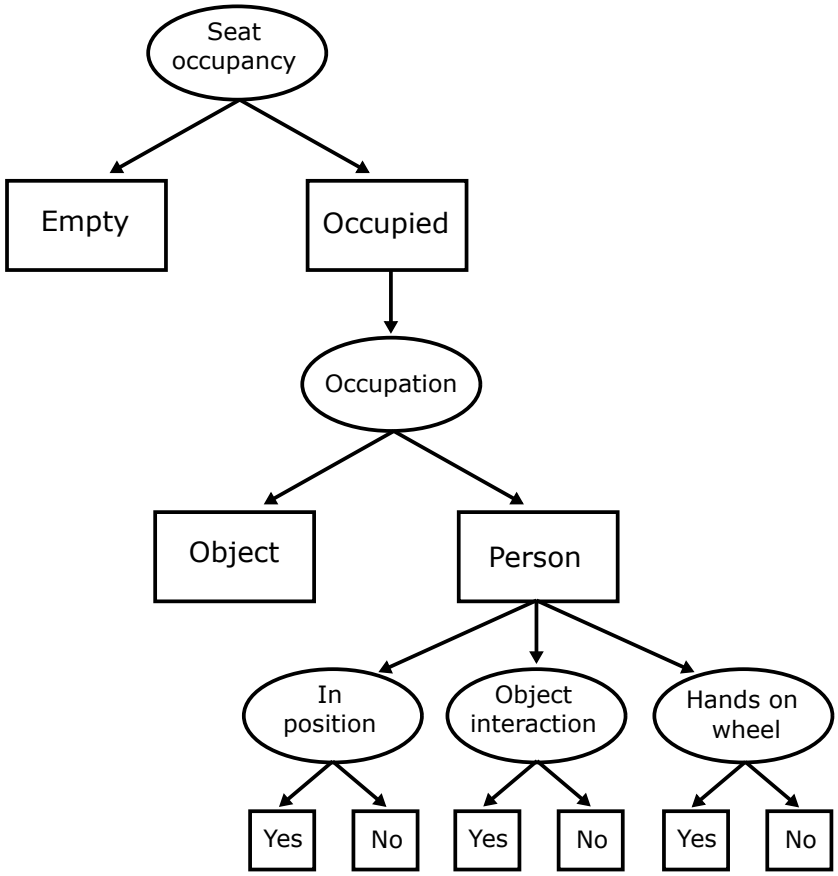


Figure 4.5: Hierarchical label structure for driver state sensing with three hierarchical layers

of the examples additionally. Similarly, the class *Object* does not benefit from more examples with the same objects used for recording the examples. However, adding more examples of different objects would certainly make the new examples valuable for training a classifier, though being very time consuming to record and annotate.

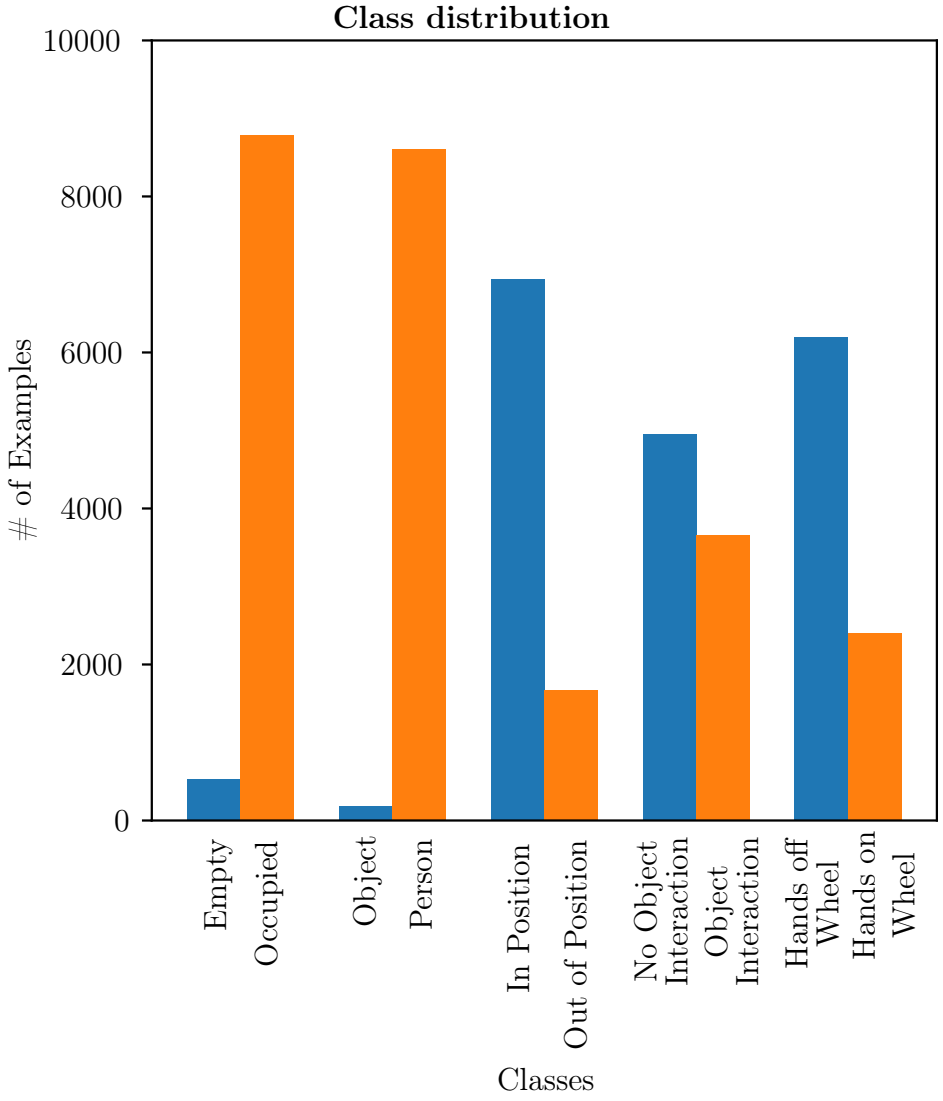


Figure 4.6: Dataset class distribution of the hierarchical driver's seat occupancy and driver state dataset.

4.2 Hierarchical Multi-Label Classification

Structuring data in a hierarchical way is a common method in several domains to arrange the data in a structured and reusable way. When classifying hierarchically structured data the classifier's performance can be enhanced by predicting a more general parental class when the more specific subclass is not recognised accurately enough. The following method is a multi-label expansion to the paper proposed by the authors of [Red17]. In contrast to this paper, the proposed method uses binary classifications for each class decision. Moreover, multiple branches of the hierarchical tree can be active simultaneously by introducing a hierarchical tree structure for multi-label classes.

4.2.1 Hierarchical Tree Structure

In order to add the described hierarchy to a classification system, two conditions must be met. First, the labels of the examples must be able be structured up hierarchically. This means that each label must have a label that describes a supercategory of the current label. Secondly, each decision between classes needs to be binary. Figure 4.7 shows the general required structure of the labels. Each decision node results in exactly two class nodes. The class nodes can have an arbitrary number of decisions. The described hierarchical tree consists of *Decision nodes* and *Class nodes*. Each decision node needs to have exactly two *Class nodes* and represents a binary decision between these classes. The number of decision nodes branching off a class node can vary between zero and an arbitrary number.

The *Decision nodes* each represent one output neuron of the neural network, while the *Class nodes* represent the class decision based on the predicted value of the parental *Decision node* neuron. The output value of a class depends additionally on the output values of each class on the path from the root of the hierarchical tree to the current class. The output of a *Decision node* represented by the vector $\mathbf{d}$ containing all decision nodes d_i and the corresponding *Class nodes* vector $\mathbf{c}$ containing the class decisions c_i on the path to the current node is calculated with

$$P(\mathbf{d} = \mathbf{c}|I) = P(d_0 = c_0|I) \cdot \prod_{i=1}^{r-1} P(d_i = c_i|d_{i-1} = c_{i-1}, I), \tag{4.1}$$

whereas r denotes the number of elements in the path from the root to the

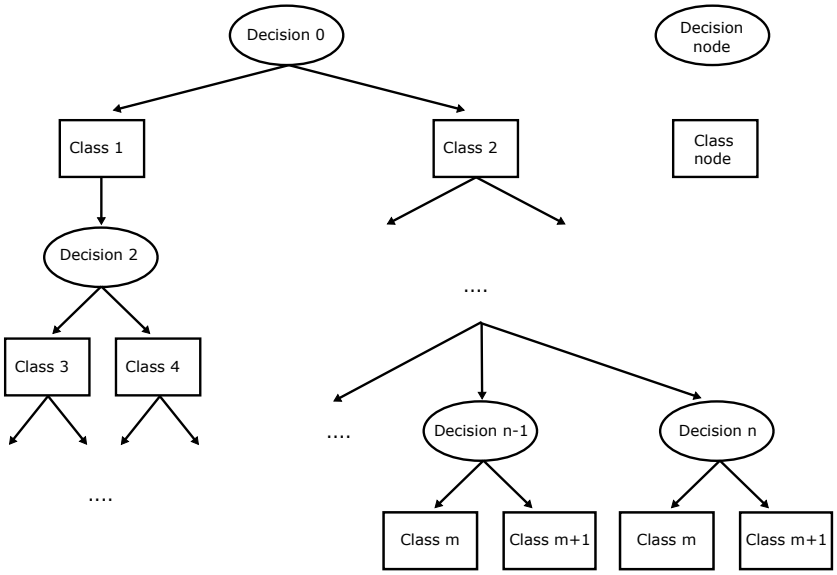


Figure 4.7: Hierarchical label structure concept with binary classes consisting of decision and result nodes

current *Decision node*. Therefore,

$$\begin{aligned} P(\{SeatOccupancy, Occupation, ObjectInteraction\} = \{Occupied, Person, Yes\}|I) = \\ P(SeatOccupancy = Occupied|I) \\ \cdot\ P(Occupation = Person|SeatOccupancy = Occupied, I) \\ \cdot P(ObjectInteraction = Yes|Occupation = Person, I) \end{aligned} \tag{4.2}$$

calculates the output of the node indicating that a person is interacting with an object.
The output of a positive class directly corresponds to the output of the related output neuron. The output of a negative class is computed by one minus the output of the corresponding positive class. Figure 4.8 and 4.9 show an example for calculating the output values of the output neuron corresponding with the object interaction displayed by equation 4.2.
The outputs for the negative classes y_i^0 are calculated by $1 - y_i^1$ with y_i^1 as the corresponding positive output. The hierarchical outputs $\widetilde{y}$ are calculated by propagating the outputs according to the hierarchy.

4.2.2 Label Structure and Masked Loss

Applying this hierarchical structure to data examples entails the circumstance that some branches and classes in the tree structure are not active

	Normalized Network Output	Negative Class	Positive Class	Output Propagated	
Occupied	0.990	0.010	0.990	0.010	0.990
Person	0.900	0.100	0.900	0.099	0.891
In Position	0.910	0.090	0.910	0.080	0.811
Object Interaction	0.150	0.850	0.150	0.757	0.134
Hands on Wheel	0.590	0.410	0.590	0.365	0.526

Figure 4.8: Example values for calculating the output of the neuron indicating the interaction with an object.

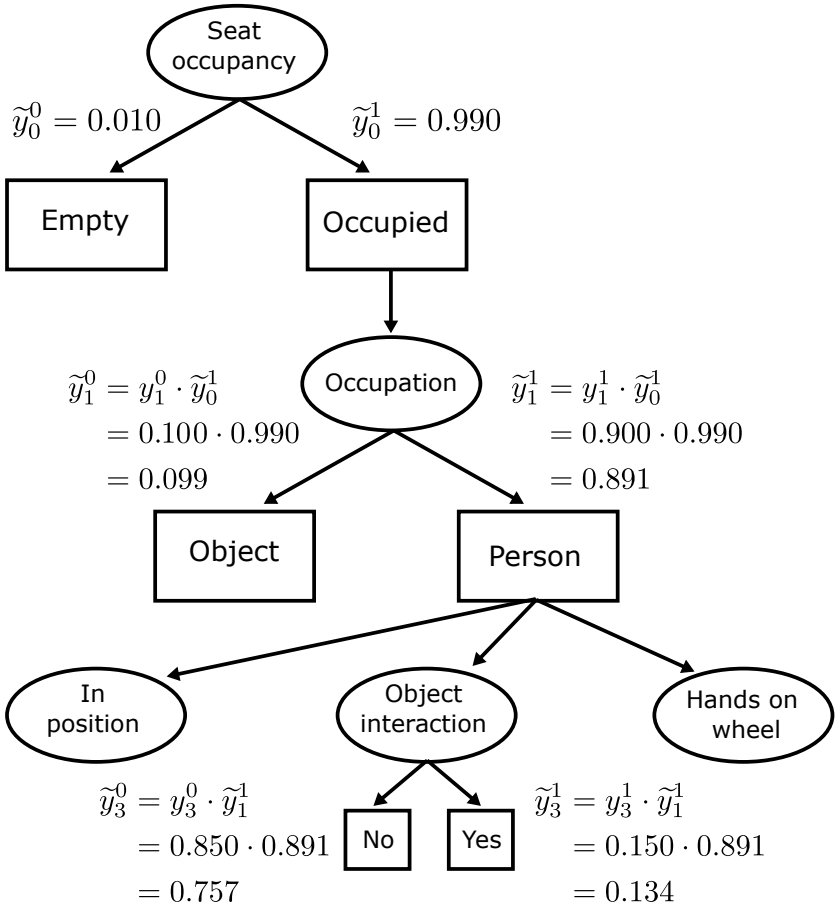


Figure 4.9: Example hierarchy for calculating the output of the neuron indicating the interaction with an object.

and should be ignored during training. For example, if an image contains an object on the driver's seat, the outputs of the neurons related to the classes *Out of Position*, *Object Interaction* and *Hands on Wheel* should not contribute to the training of the network. These branches should be ignored

for this example during training and therefore need a masking mechanism to ignore the outputs of these neuron during training. This mechanism can consist of a label structure with an additional label for the true values y, tagging the classes which need to be ignored during training. This structure can look as follows:

$$y_i = \begin{cases} 0, & \text{for negative class} \\ 1, & \text{for positive class} \\ -1, & \text{masked class} \end{cases} \tag{4.3}$$

Here, the label -1 denotes that the corresponding output of a neuron should be ignored during training. For the previous example of an object laying on the driver's seat the label vector would look like this:

$$\mathbf{y} = \begin{bmatrix} \text{S. Occupied} & \text{Occupation} & \text{In Pos.} & \text{Object Int.} & \text{H. on Wheel} \\ 1 & 0 & -1 & -1 & -1 \end{bmatrix}$$

The first label element indicates that the driver's seat is occupied while the second label element denotes that the driver's seat is occupied by an object. The last three elements are labeled with the ignore label -1 to indicate that these classes should be ignored during training.
In order to ingore the elements marked with an ignore label, the cost function used to train the network must be adjusted. Following, the binary cross entropy

$$E_{xentropy}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_i y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \tag{4.4}$$

is used as the cost function and adapted to the presented label structure.

With y_i as the true label of the i^{th} output neuron calculated from the current example and $\hat{y}_i$ as the predicted output of the i^{th} output neuron. To ignore the influence of the elements labeled with the ignore label, the loss function is modified to

$$E_{ign}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_i \mathbb{1}_{y \neq -1}(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})) \tag{4.5}$$

4.2.3 Multi-Label Classification

In comparison to most classification approaches, where one mutually exclusive class is correct, multi-label classification refers to a system where multiple classes can be active at once. In contrast to the approach proposed by the authors of [Red17], where one class in each hierarchical layer is mutually exclusive, the proposed system can have multiple active classes in one layer of the hierarchy.
In a multi-label classification network, one output neuron for each class is required, similar to a single label classification system. Since the classes are not mutually exclusive, the output values of the network need to be normalized independently. One common way to normalize the values of neurons is by applying the sigmoid function

$$\sigma(x_i) = \frac{1}{1 + e^{-x_i}} \tag{4.6}$$

to every element x_i of the output vector $\mathbf{x}$. This way, the output values of the network are normalized to values in the range of $[0, 1]$ and can be processed meaningfully by the cross entropy. Each of these normalised values represents a binary decision between two classes, e.g. whether the driver's seat is *Empty* or *Occupied*, in the hierarchical tree. A prediction towards 0 denotes a decision of the negative class and therefore the left *Class node*, while a prediction towards 1 denotes a decision towards positive class and the right *Result node* of a *Decision node*. The final output values are processed by propagating the outputs through the networks as proposed by equation 4.1.

4.3 Driver State Monitoring System

The driver state monitoring system consists of a Time-of-Flight camera monitoring the front seats of a vehicle and a classification system to recognise the current occupancy and driver state from the camera images. Figure 4.10 shows the complete driver state monitoring system. The Time-of-Flight camera records amplitude and depth images. These images are cropped in order to show only the driver's seat region. The artificial neural network to detect occupancy and driver states in an in-vehicle environment consists of a small CNN followed by a fully connected neural network to classify images of the driver side of a car. Examples of these images are shown and discussed in chapter 4.1. Figure 4.11 and 4.12 show the structure of the network pipeline. First, the input image size is scaled in order to reduce the computational

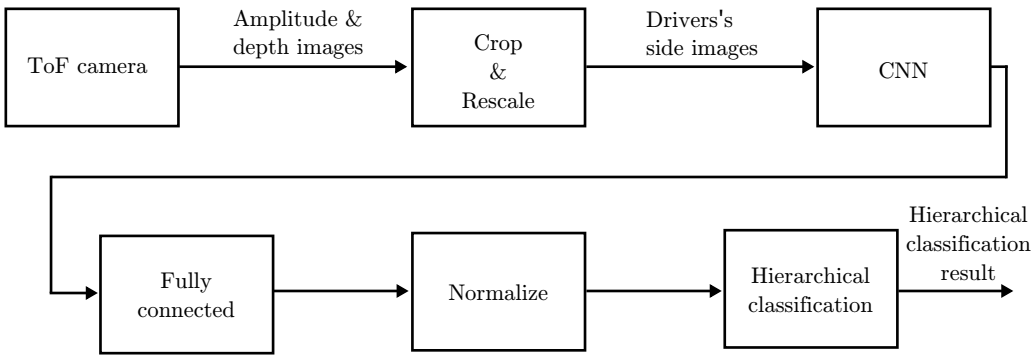


Figure 4.10: The information flow chart of the driver state monitoring system with hierarchical multi-label classification.

effort for the neural network. Then, a CNN extracts features from the scaled image, which are further processed by a fully connected neural network. The CNN consists of three convolutional layers with 8, 12 and 18 kernels with a kernel size of 3×3 each and pooling layers in between. The fully connected network has one hidden layer with 24 neurons and an output layer. The resulting output vector is processed by a sigmoid function in order to normalize the output values. The final classification result is calculated as described in section 4.2.1. An example for this calculation of an object laying on the driver's seat is shown in figure 4.12. The outputs for the negative class are calculated from the normalized network output. Following, the outputs are propagated through the network based on the hierarchical label structure as described in chapter 4.2.1. The branches of the class tree which are not relevant, as the parental node or an even higher related node is not predicted to be active, are finally marked as irrelevant accordingly. In this example, the last three values of the output are irrelevant because the network predicts an occupied seat with the first output and an object on the driver's seat with

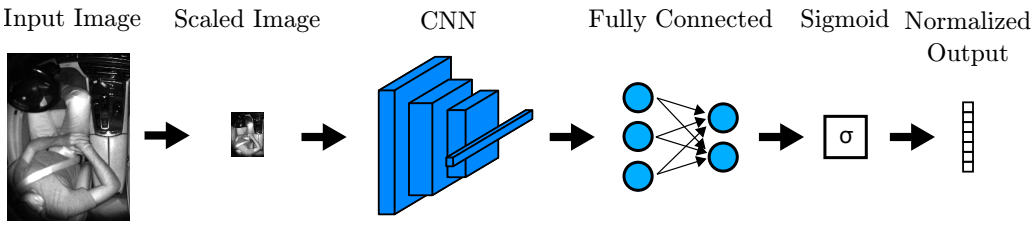


Figure 4.11: Network structure for hierarchical multi-label interior state sensing. Cropped images of the front seats of a car showing the driver's side are re-scaled and fed to a CNN. The extracted features are classified with a fully connected network and a sigmoid function.

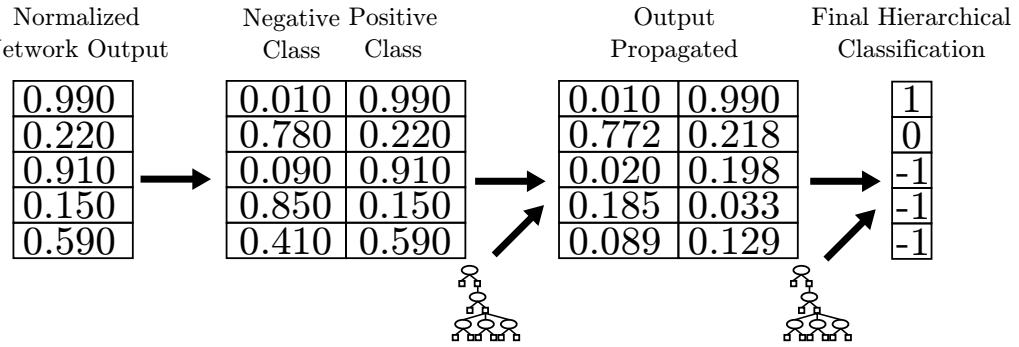


Figure 4.12: Example for the final classification based on the hierarchical label structure. The normalized network output is split up to represent the class outputs for each class. These outputs are propagated accordingly to the hierarchical label structure to get the final classification result.

the second output.

4.4 Driver State Sensing Results

The proposed method is evaluated by a five fold cross validation. Thus, the dataset was split in five parts with roughly equal size. For each split about one fifth of each class subset was selected. To prevent that too similar examples are located both in the training and validation set, examples which origin from the same recording are placed in the same split.

To compare the proposed method to other multi-label classification methods, the approach proposed by the authors of [Red17] was adjusted to be able to predict multi-label classification. For this purpose, the structure of the hierarchy was changed in a way that each class can have multiple mutually independent sets of hyponyms, i.e. several more specific subclasses, instead of one set. Each set of hyponyms represents a binary decision related to the decision tree presented in chapter 4.1. This approach is further denoted as *multi-label Yolo*.

In addition, the class structure was flattened to train a classifier with a one-hot-encoding, where each possible class combination is represented by its own class. Flattening the proposed class structure results in 10 mutually exclusive classes: *Empty*, *Object*, *Person (In Position + No Object Interaction + No Hands on Steering Wheel*), *Person (In Position + No Object Interaction + Hands on Steering Wheel)* and all other combinations of posi-

	Acc		F1		mAP	
	mean	std	mean	std	mean	std
Amplitude						
Hierarchical multi-label	**0.884**	0.023	**0.881**	0.084	**0.918**	0.011
Multi-label Yolo	0.864	0.006	0.871	0.078	0.888	0.011
Flat encoding	0.721	0.064	0.759	0.148	0.551	0.060
Depth						
Hierarchical multi-label	**0.885**	0.023	**0.888**	0.082	0.915	0.016
Multi-label Yolo	0.876	0.028	0.885	0.072	**0.919**	0.009
Flat encoding	0.717	0.066	0.764	0.140	0.583	0.022
Both						
Hierarchical multi-label	**0.889**	0.023	**0.893**	0.080	0.920	0.008
Multi-label Yolo	0.882	0.030	0.890	0.078	**0.922**	0.013
Flat encoding	0.728	0.066	0.770	0.144	0.588	0.023

Table 4.2: Five fold cross validation macro average results of the trained networks for occupancy and driver state detection with amplitude, depth and combined input images

tioning (*In and Out of Position*), object interaction (*No Object Interaction and Interaction*) and hands on the steering wheel (*Hands on and Hands off Steering Wheel*) of a person sitting in the driver's seat.

For the networks trained with the hierarchical structures, the classes are evaluated individually and the scores are combined at macro-level, while the flattened approach is evaluated in a one vs. all manner. The networks are trained on amplitude images, depth images and both image types combined and evaluated and compared with the balanced accuracy, F1-score and mAP.

Table 4.2 shows the cross evaluation results of the different approaches. The flat approach performs worst compared to the proposed hierarchical and multi-label Yolo approach for all metrics and input types. The poor performances of this approach results from the circumstance that some single classes and class combinations are significantly more frequent than other class combinations, resulting in an even heavier class imbalance than the class imbalance present in the multi-label dataset. The networks trained with the flat class label approach achieve the best performance when trained with the combination of amplitude and depth images as input. When training the networks with amplitude images, the proposed hierarchical multi-label approach performs best regarding the overall metrics compared to the multi-

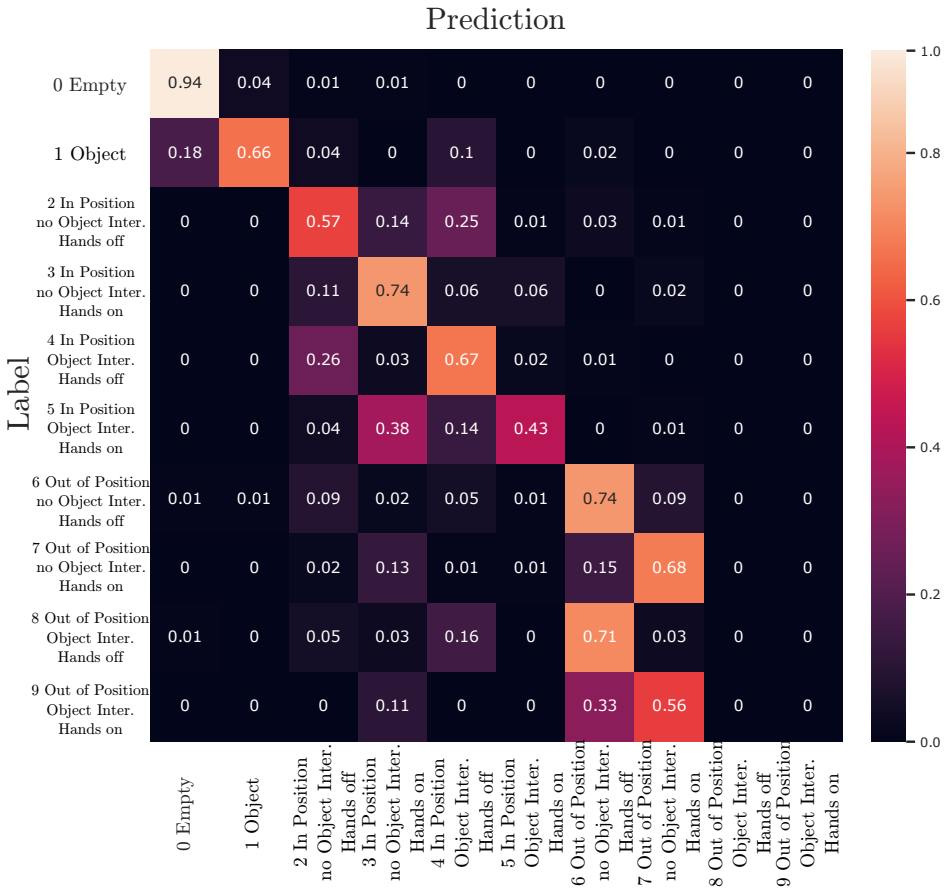


Figure 4.13: Relative confusion matrix of the validation results computed by the networks trained with the flat class label approach and both input image types combined.

label Yolo and the flat approach. This approach achieves similar results when trained on depth image data, while the performance of the networks trained with the multi-label Yolo is raised when trained on depth image data to a comparable level regarding the hierarchical multi-label approach. When using both data types as input to the networks the results of both approaches differ only slightly.

Figure 4.13 shows the confusion matrix of the validation classification results of the networks trained with the flat label structure. Some classes, like *Empty*, a person sitting in a driving position with at least one hand on the steering wheel and a person being out of a driving position are accurately recognised. Furthermore, the classes containing object interactions are most likely confused with the same driver state combination without

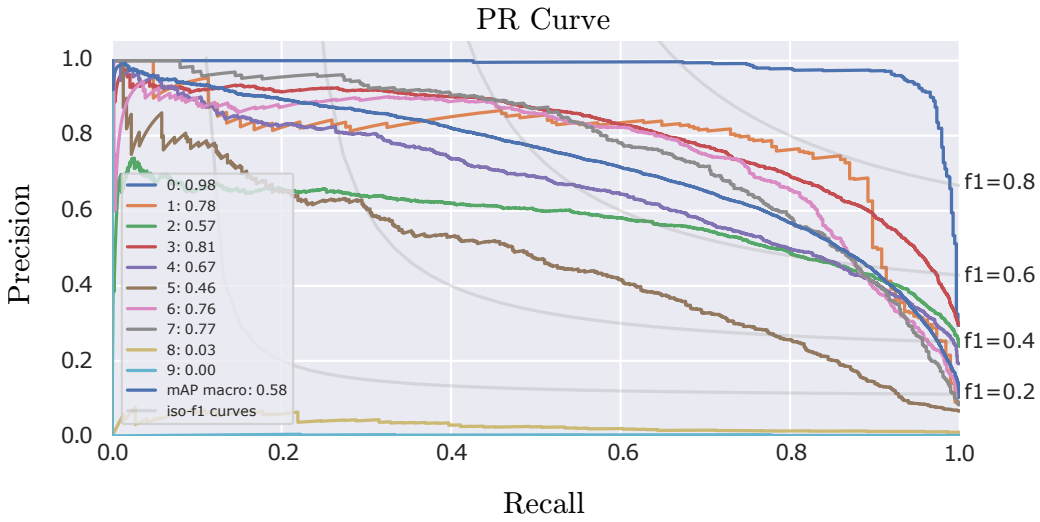


Figure 4.14: Precision recall curves of the validation results computed by the networks trained with the flat class label approach and both input image types combined.

the object interaction. Moreover, the last two classes containing the driver state combinations *Out of Position* and *Object Interaction* as well as the state combination *Out of Position*, *Object Interaction* and *Hands on Wheel* are not recognised at all. In the examples of these classes the *Object Interaction* is most likely not recognised, resulting in confusion with the classes with same driver state combination, but without the *Object Interaction*. The two classes representing these class combinations are furthermore the classes which are most underrepresented in the dataset.

The precision recall curves of the class predictions from the validation data are shown in figure 4.14. The curves show that the networks are most uncertain about the class predictions for the classes with the least recognised classes described before. With the predictions for the class *Empty* the networks are most certain, which can also be seen at the confusion matrix scores for this class.

The balanced accuracy per class of the validation data computed by the networks trained with the hierarchical approaches and both input image types combined is shown in table 4.3 and table 4.4. For the classes *Empty*, *Occupied*, *Object* and *Person* the networks trained with the proposed hierarchical multi-label structure show higher balanced accuracy scores compared to networks trained with the modified multi-label Yolo approach. For the

	Class							
Method	Empty		Occupied		Object		Person	
	Acc	Std	Acc	Std	Acc	Std	Acc	Std
Hierarchical multi-label	**0.958**	0.074	**0.958**	0.074	**0.923**	0.048	**0.991**	0.006
Multi-label Yolo	0.942	0.082	0.939	0.081	0.869	0.097	0.976	0.023

Table 4.3: Balanced accuracy per class results of the validation data computed by the networks trained with hierarchical label structure and both input image types combined.

	Class											
Method	InPosition		OutOfPosition		OffWheel		OnWheel		NoInteraction		ObjectInteraction	
	Acc	Std	Acc	Std	Acc	Std	Acc	Std	Acc	Std	Acc	Std
Hierarchical	**0.922**	0.014	0.890	0.023	0.872	0.015	0.870	0.016	0.778	0.032	0.732	0.064
Yolo	0.918	0.014	**0.893**	0.026	**0.875**	0.018	**0.873**	0.013	**0.787**	0.046	**0.748**	0.062

Table 4.4: Balanced accuracy per class results of the validation data computed by the networks trained with hierarchical label structure and both input image types combined (continued).

different driver states the modified multi-label Yolo approach shows better results for the classes describing the object interaction. The results for the remaining classes are comparable between both approaches.

The mAP scores of both hierarchical approaches are shown in table 4.5 and table 4.6. When comparing both approaches, it can be seen that the proposed hierarchical multi-label approach scores slightly better for the classes *Empty*, *Occupied* and *Person*, while scoring markedly better for the class *Object*. The driver state classes are classified with comparable mAP scores between both approaches for most of the classes. Merely the class *Object Interaction* is more reliable classified by the modified multi-label Yolo approach.

The remaining confusion matrices and precision recall curves of the validation results of the networks trained with the flat label approach are shown in chapter A.

	Class							
Method	Empty		Occupied		Object		Person	
	Acc	Std	Acc	Std	Acc	Std	Acc	Std
Hierarchical multi-label	**0.991**	0.014	**1.000**	0.000	**0.851**	0.089	**1.000**	0.000
Multi-label Yolo	0.986	0.018	0.999	0.002	0.814	0.070	**1.000**	0.001

Table 4.5: Average precision scores per class of the validation data computed by the networks trained with hierarchical label structure and both input image types combined.

	Class											
Method	InPosition		OutOfPosition		OffWheel		OnWheel		NoInteraction		ObjectInteraction	
	Acc	Std	Acc	Std	Acc	Std	Acc	Std	Acc	Std	Acc	Std
Hierarchical multi-label	0.988	0.006	0.919	0.027	**0.938**	0.026	0.911	0.020	0.913	0.046	0.684	0.070
Multi-label Yolo	**0.990**	0.003	**0.930**	0.017	**0.938**	0.033	**0.917**	0.020	**0.920**	0.042	**0.722**	0.062

Table 4.6: Average precision scores per class of the validation data computed by the networks trained with hierarchical label structure and both input image types combined (continued).

4.5 Summary and Conclusion of the Hierarchical Multi-Label Driver Monitoring

In this chapter a system for recognising the occupancy of the driver's seat and various states that a driver may adopt is presented. For this purpose a dataset was recorded and annotated consisting of Time-of-Flight image data showing the driver's seat area of a car. To train a neural network efficiently for the task of multi-label classification with this data, a hierarchical multi-label structure is proposed to directly integrate a class hierarchy into artificial neural network training. Every decision node in the hierarchy is represented by a binary class decision for the network, resulting in a classification structure, with an integrated fall back strategy for uncertain predicted classes. Moreover, the proposed hierarchical methods allow it to ignore parts of the classification that are irrelevant for training other parts of the network.

In contrast to a flat label hierarchy, it is not required that every possible class combination is encoded as a single class, which can result in an unnecessarily confusing label structure, as well as heavy class imbalances within the dataset if many different class combinations are possible. A class imbalance within the dataset is not preventable for hierarchical structured labels, as parental nodes in the hierarchy always contain all examples corresponding to the child nodes. Nevertheless, this class imbalance is usually not as prominent as in the flat hierarchical structure if the data is gathered carefully.

The results show that integrating a hierarchy to the training and classification process can increase the performance of classifiers significantly, compared to a flat label approach, if multiple classes can be active at the same time. Two similar hierarchical structures to integrate hierarchical classification to a single network were proposed. While the proposed hierarchical multi-label approach shows better results for classifying the driver's seat occupancy and driver's states for networks trained on amplitude images, the method proposed by [Red17] works comparably well, when extended to multi-label classification on a combination of amplitude and depth images.

Overall it was shown that a hierarchical multi-label classification can increase the performance of a classifier by focusing only on relevant information during training. Moreover, a hierarchy yields a natural fallback option for classification tasks, through the sequential data structure.

CHAPTER 5

Suggestion of an Action and Object Interaction Recognition System for Driver Monitoring

In order to gain a deeper understanding of the driver's activities using in-cabin cameras, the action and object interaction classification aims to recognize different activities of the driver as well as the driver's interaction with different objects from image sequences.

As the proposed application is targeted to run on an embedded processor in a vehicle, the system's computational effort needs to be as low as possible. Moreover, the recording and annotating data sequences for training the system is time consuming and costly. Therefore, a lightweight network architecture and new training procedures needs to be designed to train the system efficiently with the available data and process the input data fast and confidently.

To classify different driver activities the authors of [Ohn14a] detect the location of interesting regions in images showing the interior of the front seats of a car. Interesting regions are the region where the steering wheel, the instruments or the gear shift are present. These subimages are further analysed to detect the presence of hands. Moreover, the position of the driver's head along facial landmarks are located in images from a camera facing the driver. Instead of detecting interesting regions of the interior the authors of [Hoa16] directly detect hands and objects to further analyse geometric features of the detected regions and put them into context. Image regions of the driver's face and binary images of the driver's skin are analysed along with hand images with an ensemble of convolutional neural networks to detect driver distractions in the paper of [Era19]. The authors of [Beh18] analyse the connections between detected body key points of the driver and object locations.

The system aims to enhance the driver monitoring application presented in chapter 4 with a functionality to bring images of a sequence into context to classify action sequences performed by a driver. This extension of functionality allows the system to distinguish between fine grained actions that a system that can only infer about the driver's condition through individual images cannot. As most actions or object interactions of a driver are either performed with the full body or the hands, the input features of an action and object interaction system can be reduced to the driver's body pose and cut outs of the driver's hands.

In contrast to the previously introduced papers, the presented system focuses on analysing the driver's body movements based on the driver's $3D$ body key points only. As object interactions are mainly performed with the driver's hands, sub images of the driver's hands are analysed alongside the driver's $3D$ body key points to recognize fine grained object interactions. This combination of features reduces the input feature space significantly, while still preserving all in this context relevant input features, which allows smaller, computationally more efficient networks to be used.

Some papers [Yan16b], [Xin19], [Era19] use skin color detection for image segmentation to localize body parts such as hands. However, as most skin color detection methods rely on color images these methods are not suitable for Time-of-Flight images. Skin segmentation in amplitude images is also error prone due to similarities with other materials (e.g. fabrics, leather, etc.). Moreover, it is not uncustomary that the skin of the driver is visible at all due to clothes that cover him completely. Not detecting any skin regions of the driver would be problematic for the system if it depends on this information. Therefore, the proposed system does not rely on any skin detection mechanism.

Even though the presented work focuses on driver's actions and object interactions, the methods are not restrained to be used only for recognizing the actions and object interactions of a driver, but for passengers or different settings beyond interior sensing as well.

The dataset used to train and validate this system was recorded in different car interiors with multiple people performing predefined actions and object interactions. Gathering the video data and annotating was part of the presented work. The described system and methods origin from the paper proposed by the author of this work in [Wey19].

Driver Actions	
Nothing	Default class showing a driver doing normal body movements related to driving
Enter	Driver enters vehicle
Leave	Driver leaves vehicle
Strap	Driver fastens seatbelt
Unstrap	Driver unbuckles seatbelt
Object Interactions	
Phone Idle	Driver holds a phone
Phone Interaction	Driver interacts with a phone (e.g. typing)
Phone Call	Driver takes a phone call
Bottle Idle	Driver holds a bottle
Bottle Interaction	Driver interacts with a bottle (e.g. opening, closing)
Drinking	Driver drinks from a bottle

Table 5.1: Action and object interaction labels and descriptions. The labels subdivide in the two categories *Driver Actions* and *Object Interactions*.

At the beginning of this chapter, an overview of the gathered dataset is given in section 5.1. In section 5.2, a new method for augmenting the time component of sequence data is presented and evaluated. The methods for generating the driver's body key points and hand sub images are presented in section 5.3. Following, the system, including the network architecture, is presented in section 5.3.5. The proposed system is evaluated in section 5.3.6. Finally, section 5.4 summarises the proposed training methods and results.

5.1 Action and Object Interaction Recognition Data

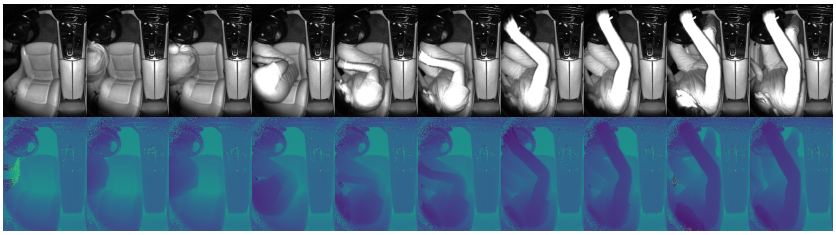
The dataset used to classify actions and object interactions consists of multiple sequences each showing one action or object interaction performed by a driver of a car. The images have the same field of view of the car's front seats as the dataset recorded for the hierarchical occupancy and driver state classification described in chapter 4.1. Each sequence was recorded in one of the available cars and was labeled manually.

The action label categories are listed in table 5.1 and subdivide in the actions *Nothing*, entering the car (*Enter*), leaving the car (*Leave*), fastening the seat belt (*Strap*) and unstrapping the seat belt (*Unstrap*). The object interaction focuses on the interaction with two objects, namely a phone and a bottle. This interaction categories subdivides into the categories of holding the object (*PhoneIdle*, *BottleIdle*), interacting with the object (*PhoneInteraction*, *BottleInteraction*) and using the object (*PhoneCall*, *Drinking*). The class *PhoneInteraction* shows sequences where the driver is using a phone in a way where he is typing on the display or the keyboard of the phone, whereas the class *PhoneCall* explicitly shows the driver leading the phone to one ear with one of his hands. Similarly, the class *BottleInteraction* shows the driver opening and closing bottles, while the class *Drinking*, although this class may also fall into the category *BottleInteraction*, gets its own category. Other possible interactions that can be performed with a phone or a bottle are not part of the dataset, but can be integrated easily.

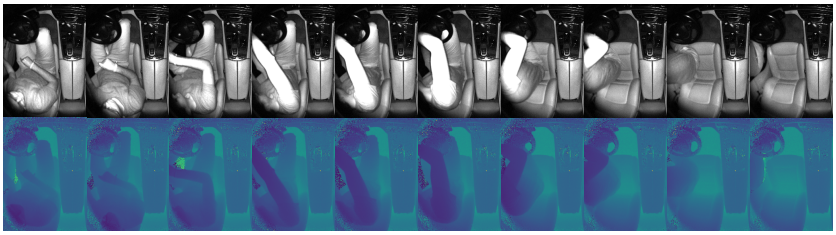
Figure 5.1, figure 5.2 and figure 5.3 show example scenes of the actions and object interactions of the dataset. A selection of amplitude images as well as the corresponding depth images of a sequence are shown to visualize the different classes used for the action and object interaction recognition.

Figure 5.1a shows a sequence of the category *Nothing*. At the beginning a driver is sitting on the driver's seat, with the hands on the jacket. In the further course of the scene, the driver lifts the arms to grab the steering wheel while moving the upper body in different directions. This example shows the default case where a driver is present, but performs none of the mentioned actions.

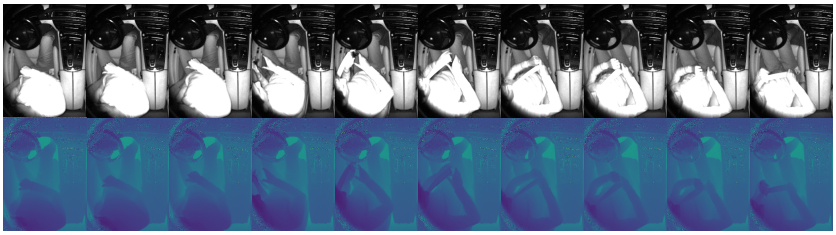
(a) *Nothing* sequence,



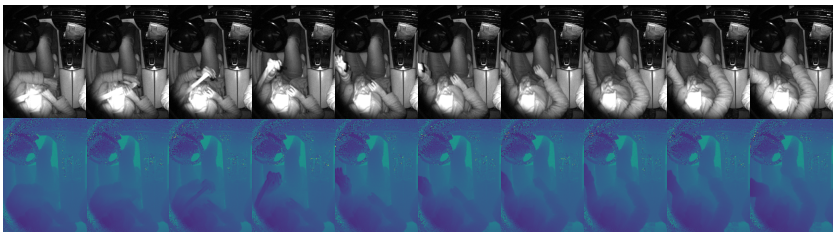
(b) Enter sequence,



(c) Leave sequence,



(d) Fasten the seat belt sequence,



(e) Unstrapping the seat belt sequence.

Figure 5.1: Driver actions.

An example of a driver entering the car is shown in figure 5.1b. In the beginning of the sequence, the driver's seat is empty. From the second frame on, parts of the driver are visible as he enters the car at the driver's side. In the process of the scene, the driver, as he enters the car, gets more and more visible, until he sits completely in the driver's seat.

Next, figure 5.1c shows a driver leaving the driver's seat. The scene starts with the driver moving his left hand towards the driver's door. Following, the driver leaves the car, by grabbing the steering wheel with the right hand and pulling himself out. Finally, the driver's seat is empty and only a small part of the driver is still visible in the driver's door frame.

A scene where the driver fastens the seat belt is shown in figure 5.1d. The driver reaches behind his left shoulder to grab the seat belt. Then, he pulls the seat belt from the pillar loop, forwards the tongue of the seat belt to the right hand and locks it at the buckle of the seat belt.

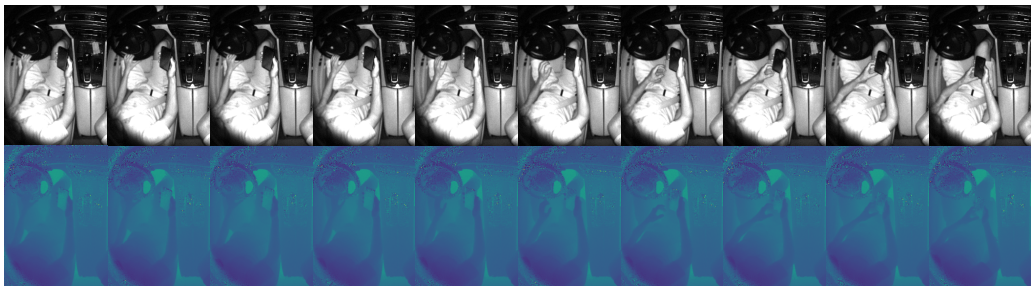
Figure 5.1e shows an example of a driver unstrapping the seat belt. The driver unlocks the tongue of the seat belt from its buckle with the left hand and leads it back to its pillar loop.

Figure 5.2 shows one example for each phone interaction class, respectively.

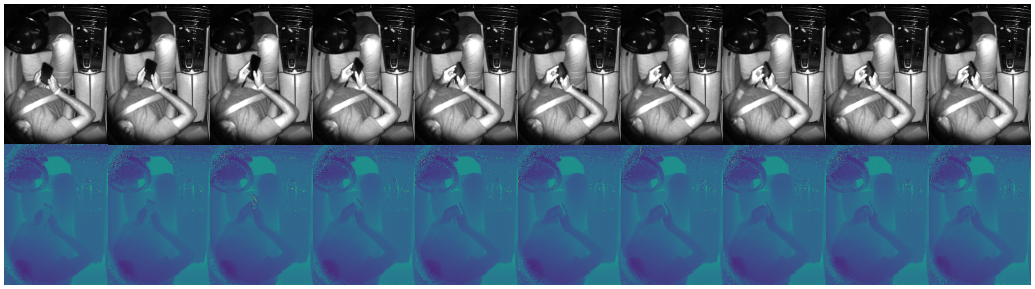
An example of the driver just holding a phone is shown in figure 5.2a. The driver sits in the driver's seat in a normal position while holding a phone. In the beginning, the driver holds the phone in the right hand while the left hand grabs the steering wheel. In the whole scene, the phone is only held in the hand and not being used in terms of direct interacting, like typing on it.

The direct interaction with a phone is shown in figure 5.2b. This time, both hands hold the phone for the complete sequence while the driver's fingers are present at different locations of the display, indicating that the driver is typing on the phone and therefore directly interacting with it.

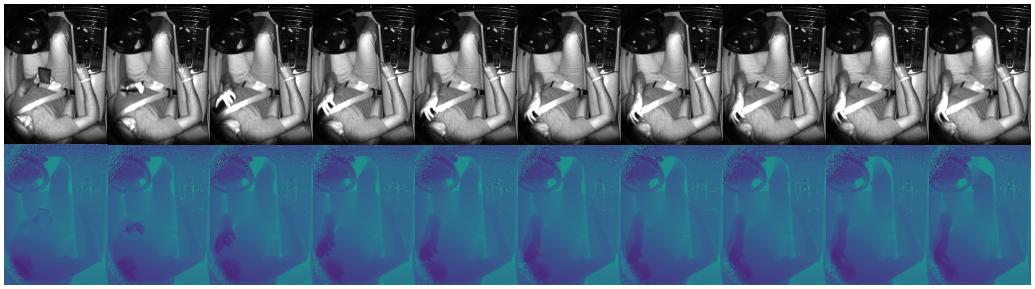
An example sequence of the driver taking a phone call is shown in figure 5.2c. The example starts with the driver holding a phone in the left hand. In the following frames, the driver leads the phone towards his left ear.



(a) Holding a phone sequence,

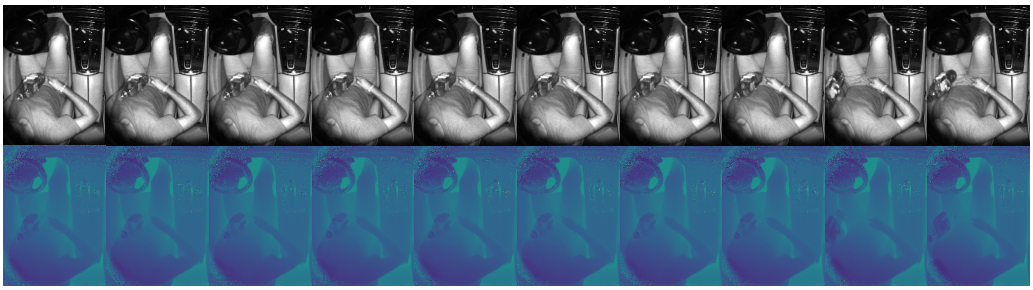


(b) Interacting (typing) with a phone sequence,

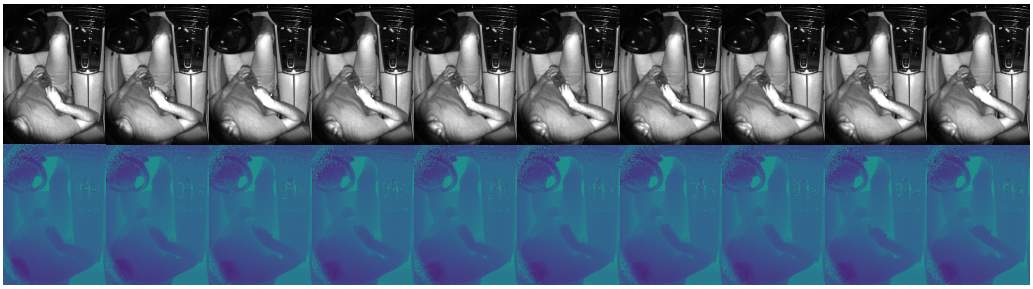


(c) Taking a phone call sequence.

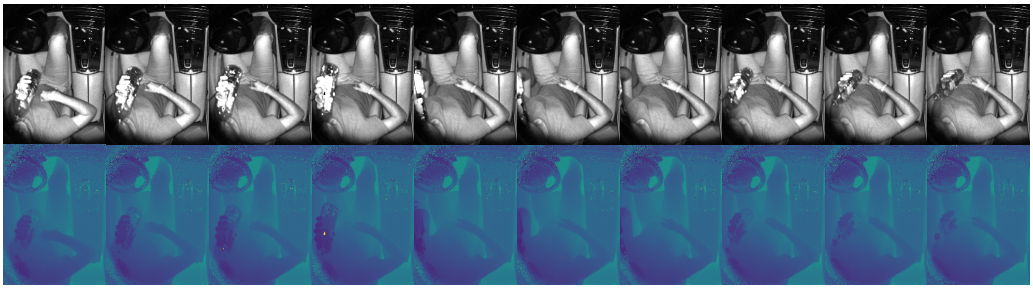
Figure 5.2: Driver phone interactions.



(a) Holding a bottle sequence,



(b) Interacting (opening) a bottle sequence,



(c) Drinking sequence.

Figure 5.3: Driver bottle interactions.

Examples of the different interaction classes with a bottle are shown in figure 5.3. The first example, displayed in figure 5.3a, shows the driver holding a bottle with the left hand at different locations in the scene.

Secondly, figure 5.3b shows the driver opening the screw cap of the bottle with the right hand, while the bottle is located in the left hand.

An example of the driver drinking from a bottle is shown in figure 5.3c. The driver holds the bottle in the left hand, lifting it in order to drink in the progression of the scene and finally lowering it again. It can be seen, that most part of the left arm is not visible as the arm is lifted. The bottle is also only partly visible as it is lifted and only gets visible again as the hand with the bottle is lowered again.

The sequence length varies from 30 frames per sequence to up to 180 frames per sequence. Some recordings of the classes *Nothing*, *Phone idle*, *Phone interaction* and *Bottle idle* are significantly longer than the maximum sequence length of 180 frames in the dataset as it is more convenient to record this kind of data in longer sequences. However, if a recording of a class exceeds 180 frames, the recording is subdivided into multiple sequences with a maximum sequence length of 180 frames.

The class distribution of the dataset is shown in figure 5.4. The classes imbalance results from the data acquisition. Some of the sequences were especially recorded for this system while others were used from already existing recordings. As some classes like *Nothing* or *Phone idle* occur more naturally and are easier to record, some classes like *Enter* or *Strap* require more recording and labeling effort, resulting in unevenly balanced classes. Moreover, examples for a subset of the classes were initially recorded for different tasks. Nevertheless, as they display actions used for the proposed task they were included in the dataset, increasing the class imbalance even more.

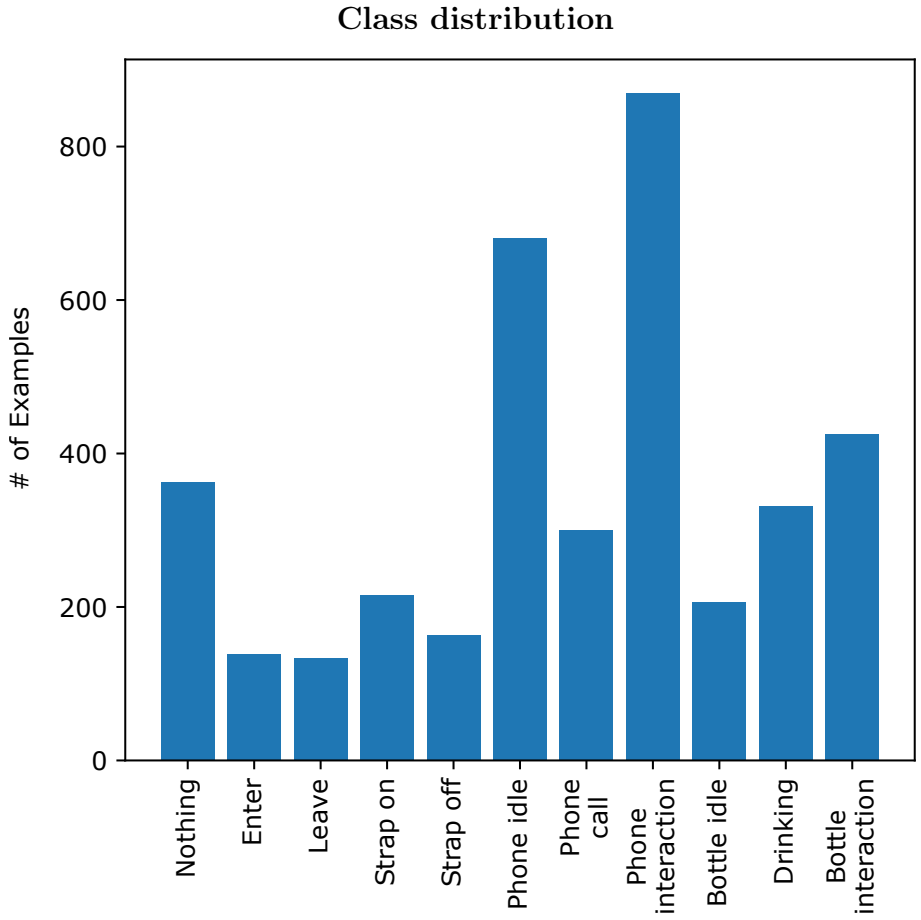


Figure 5.4: Dataset distribution for action and object interaction recognition

5.2 Time Augmentation

As the dataset is comparatively small for training neural networks for action recognition, the data needs to be augmented in order to get more variance to the training data. Augmenting sequence data cannot only be done in a spatial way, but in the time component as well. For this, jittering is applied to the image sequence by the authors of [Sim14a], [Mol16]. The authors of [Car17] randomly select the starting frame of a sequence to augment the time component. Here, a method is described to speed up and slow down sequences at different phases of the sequences systematically. The augmentation method is based on randomly calculated sine-curves, each representing the speed of a sequence at different parts of the sequence. This speed defines where frames of the sequence to augment are skipped or added to artificially

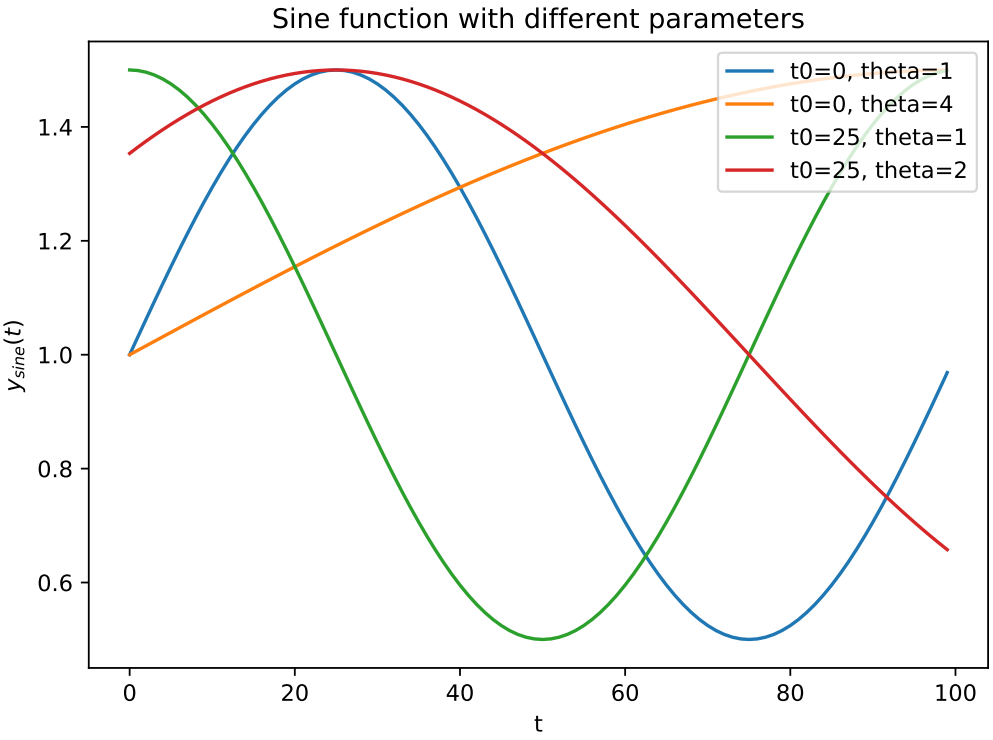


Figure 5.5: Sine curve examples with different parameters

accelerate or decelerate the sequence and thus creating different ways of e.g. performing an action in a sequence.

5.2.1 Sequence Speed Variation Function

To augment the time component of the sequences, a sine function is calculated describing the speed at different parts of the sequence. The sine function is calculated with random parameters for the dilation and phase angle in predefined ranges. Moreover, the sine function

$$f_{sine}(t) = 0.5 \cdot \sin(\omega(t + t_0)) + 1, \tag{5.1}$$

$$\text{with } \omega = \frac{2 \cdot \pi}{\hat{T}} \tag{5.2}$$

is defined in a way that it has an amplitude of 0.5 and a bias of 1 to provide values between 0.5 and 1.5 which is used as an indicator of how much values of a part of a sequence are used later in order to augment it temporally. The value t represents the current frame number. The shift t_0 is selected randomly in the range of $[0, T - 1]$. To get a random dilation of the period

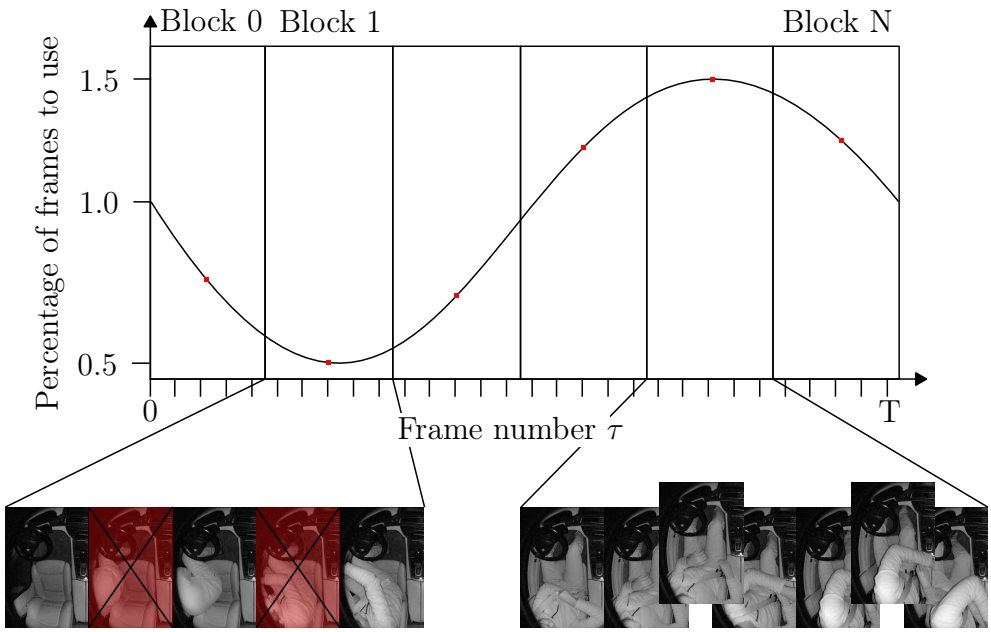


Figure 5.6: Temporal augmentation concept visualization. A sequence is split into several blocks. Each block is assigned to a part of a function which defines how many frames of the block are used.

length of the sine curve dependent on the sequence length T of the current sequence,

$$\hat{T} = T \cdot \theta \tag{5.3}$$

is calculated with T as the number of frames of the current sequence and θ as a random float number between 1 and 4 to prevent the curve from oscillating to little or to much. Figure 5.5 shows different curves with different parameters.

5.2.2 Frame Selection

To apply the calculated values of the randomly generated sine curve the sequence is separated into $N+1$ blocks. Each Block consists of n_i consecutive frames of the sequence with $i = 0, 1, ..., N$. For each block with its frame numbers the mean of the function outputs of f_{sine} for the corresponding frames is calculated. An α_i value

$$\alpha_i = \frac{1}{n_i} \sum_{\tau=\tau_i}^{\tau_i+n_i-1} f_{sine}(\tau), \tag{5.4}$$

where τ_i is the number of the fist frame in block i, is used as a factor for each block i to describe how many frames of this block are used. E.g. a factor of

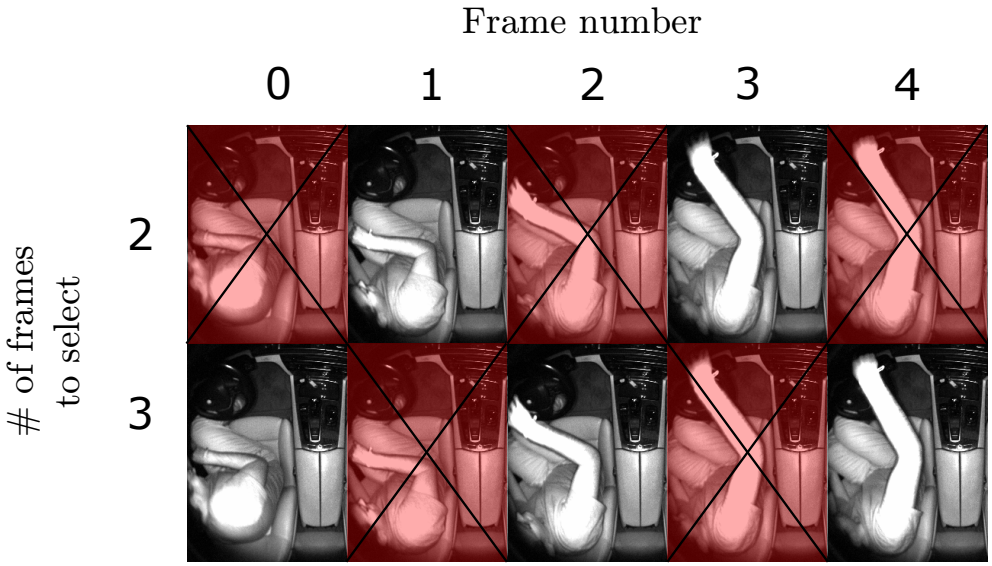


Figure 5.7: Frame selection example for selecting two and three frames of a block with five frames.

1.5 means that half of the frames of this block are doubled. Therefore, blocks assigned with a lower factor are accelerated by skipping frames, while blocks assigned with a high factor are slowed down by duplicating frames. This is visualized in figure 5.6 with five frames per block.

To select the frames in one block that should be augmented,

$$n_{aug_i} = |round(n_i \cdot \alpha_i) - n_i| \tag{5.5}$$

frames to augment are selected. From all frames of a block n_{aug_i} equidistant frames of block i are selected to ensure that each block is augmented homogeneously. For example, if two frames from a block with five frames should be selected, the second and the fourth frame would be selected. The selection of frames for a block of five frames is visualized in figure 5.7.

To choose equidistant frames of a block of frames, the algorithm 1 is executed: The function *linspace* creates a vector with ascending values from the start value to the end value with a certain steps size between each value. If $\alpha_i > 1$, the selected frames are duplicated and inserted right after their origin. If $\alpha_i < 1$, the selected frames are deleted from the sequence. In case $\alpha_i = 1$, no frames are selected and thus the current block i is not temporally augmented.

Algorithm 1 Choose equidistant frames

procedure SelectEquidistantFrames(m, n) ▷ select m numbers from n
  $start = 1 - \frac{n}{2 \cdot min(m,n)}$
  $end = n + \frac{n}{2 \cdot min(m,n)}$
  $step = min(m, n) + 2$
  $idx = round(linspace(start, end, step)$
  $results = List[0]$
  for idx in IDX **do**
    if $idx > 0$ AND $idx \leq n - 1$ **then**
      $results.append(idx)$
  return results

5.2.3 Results of Time Augmentation

To evaluate the influence of the described systematic time augmentation, multiple convolutional neural networks were trained on the full amplitude, depth and optical flow data without temporal augmentation, with random temporal augmentation and with the proposed systematic temporal augmentation to classify the different actions and object interactions. In order to evaluate the different approaches, each approach was trained five times, each time with a different validation split, resulting in a five fold cross validation. The networks trained with no temporal augmentation were trained and validated on every fourth frame of the sequences, whereas the start frame was randomly selected within the first four frames while training. For training the random and systematical temporal augmentation approaches, the frame selection was randomly selected between every third to every fifth frame. Additionally, the start frame was selected within the first frames depending on the skip selection. No temporal augmentation was applied to the validation data. For validating those two approaches, every fourth frame was selected, starting with the first frame. For the random temporal augmentation, a random selection of frames up to 20% of the sequence length were randomly skipped or duplicated. The optical flow images were calculated after the frame selection between the used consecutive frames.

The final classification result of a sequence was generated by averaging the last three network predictions of the sequence. The results of the different approaches are displayed in table 5.2. For the amplitude image sequences, the networks trained with random augmentation perform better compared to the networks trained with no temporal augmentation, while the networks

	Acc		F1		mAP	
	mean	std	mean	std	mean	std
Amplitude						
no time augmentation	0.683	0.044	0.637	0.061	0.700	0.061
random time augmentation	0.694	0.019	0.654	0.036	0.718	0.022
systematic time augmentation	**0.701**	0.033	**0.674**	0.020	**0.729**	0.017
Depth						
no time augmentation	0.731	0.046	0.697	0.047	0.747	0.043
random time augmentation	0.735	0.044	0.701	0.050	0.766	0.038
systematic time augmentation	**0.748**	0.048	**0.715**	0.055	**0.783**	0.045
Flow						
no time augmentation	0.774	0.025	0.750	0.018	0.823	0.032
random time augmentation	0.779	0.018	0.754	0.025	0.829	0.026
systematic time augmentation	**0.793**	0.018	**0.769**	0.019	**0.858**	0.026

Table 5.2: Cross validation results of networks trained with different time augmentation techniques and different input image formats.

trained with the systematic temporal augmentation perform better than the networks trained with random temporal augmentation in all three metrics. Moreover, the standard deviation of the results was reduced with both augmentation techniques compared to the results of the networks trained with no augmentation. Overall, the networks trained on amplitude image sequences perform worst, compared to the networks trained on depth or optical flow images. This might result from the small network structure of being able to learn relevant features from the amplitude image sequences as the variation in those images is higher than in the depth or optical flow images. However, using a bigger network structure resulted in overfitting of the networks to the training data.

While the mean balanced accuracy and F1-scores of the networks trained with depth image sequences with no temporal augmentation and random temporal augmentation is similar, the mAP of the networks results trained with random temporal augmentation has risen compared to the networks results with no temporal augmentation. The results of the networks with systematic augmentation exceed the other two approaches in all three metrics. Compared to the networks trained on amplitude images, the networks trained on depth images result in better balanced accuracy, F1-scores and mAP. The depth images show a reduced variance compared to the amplitude images, which is the reason for the improved results on the validation splits.

The networks trained on optical flow image sequences show the best results compared to the networks trained on amplitude and depth images. These images reduce the variance even more, because only the movements of the driver are visible. The results of the networks trained with no temporal augmentation and random augmentation show no significant differences, while the results of the network trained with systematic temporal augmentation exceed these approaches.

Figure 5.8 shows the confusion matrix of all validation results calculated by the networks trained on optical flow image sequences with systematic temporal augmentation. The predictions show that actions which require the driver to perform large movements, like entering the car, leaving the car, strapping the seat belt or taking a phone call, are recognised much more frequently than fine grained actions like actions which aim to distinguish between holding a phone or typing on a phone. This differentiation is much more difficult to recognise from optical flow images as the movements of these fine grained actions might not be captured by the optical flow.

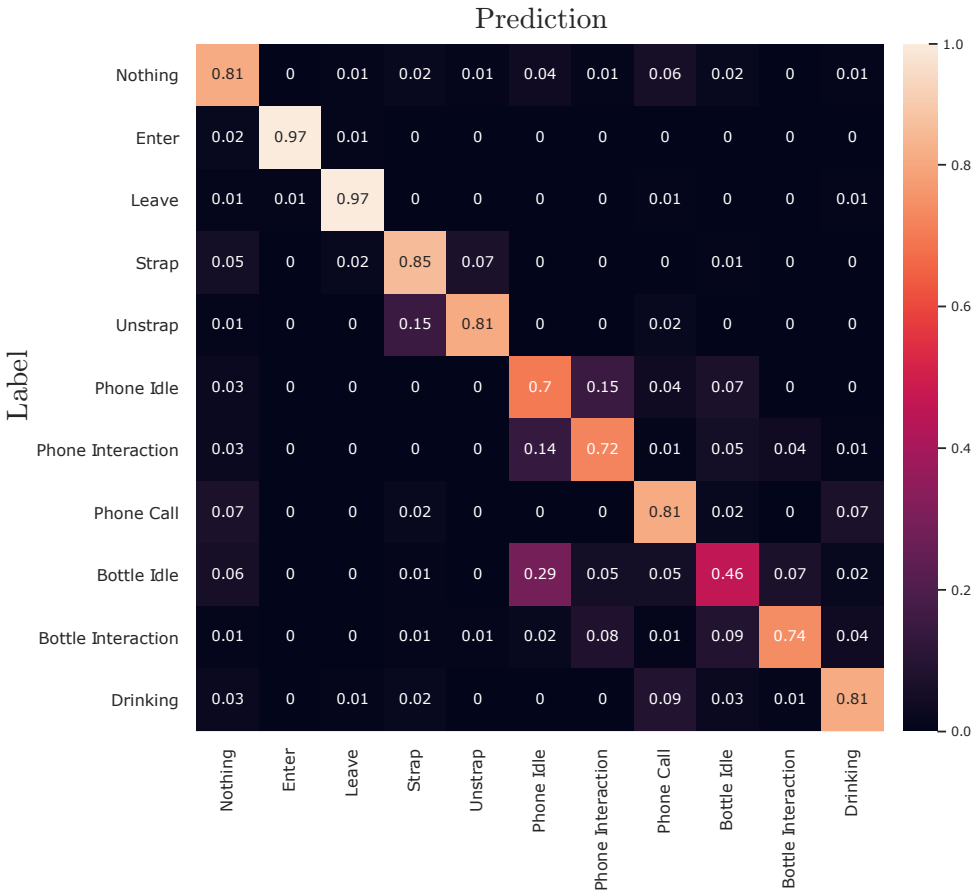


Figure 5.8: Confusion matrix of the validation predictions from the network trained on optical flow image sequences with systematic temporal augmentation.

The worst confusion happens with the class *Bottle Idle*. As optical flow images only capture the movements between consecutive frames the network might not be able to distinguish between holding a bottle and holding a phone as not much movements happens in the examples of these classes. The fact that the class *Bottle Idle* is classified in 0.29% of the cases, but the class *Phone Idle* is not recognised as that, is most likely a result of the class imbalance of the dataset. That the network recognises 0.46% of the class *Bottle Idle* can be the result of the fact that bottles are usually larger than phones and certain movements appear in the *Idle* classes which can be recognised by the network.

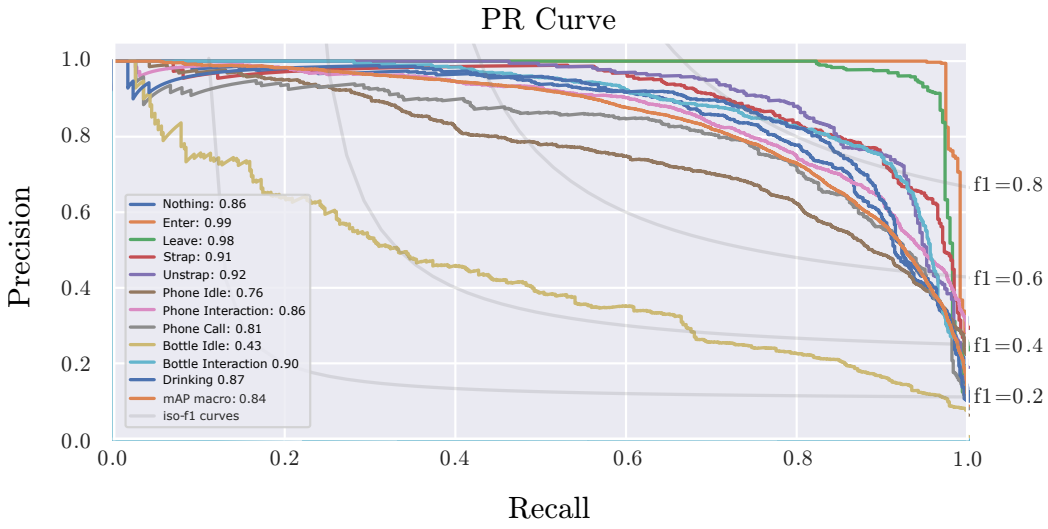


Figure 5.9: Precision recall curves of the validation predictions from the network trained on optical flow image sequences with systematic temporal augmentation.

The precision recall curves of the networks are displayed in figure 5.9. The classes *Enter* and *Leave* are recognised with average precision values of 0.99 and 0.98 with the biggest certainty. This strengthens the assumption that the classes with the most body movement of the driver are recognised most confidently by the networks. The class *Bottle Idle* is recognised with the lowest certainty as seen before. This uncertainty results in a lower certainty for the class *Phone Idle* as well, as these classes might get confused. With an average precision value of at least 0.81 the networks are much more confident in classifying the remaining classes.

Training on a combination of optical flow and amplitude or depth images is also possible to enhance the classification results. However, combining two different input formats required a significantly increased network architecture, making the approach impractical in terms of computational efficiency.

These results show that training on the right input features can be crucial for training a recurrent neural network. Moreover, applying a systematic temporal augmentation to the training process can enhance the ability of a RNN to classify image sequences.

The remaining confusion matrices and precision-recall curves of the validation results of the networks trained with the different time augmentation

techniques are shown in chapter B

5.3 Reduced Features

When analysing image sequences to recognise different human actions or human interactions with objects, image sequences provide a huge number of features to analyse. Approaches relying on artificial neural networks achieve better and better results on open access action recognition datasets like [Abu16], [Car17]. However, these networks usually consist of many stacked layers with many operations and are therefore too big and therefore too computational expensive to run in a real-time environment like a car.

In order to reduce the size of the networks the input feature space can be reduced to channel the attention of the networks directly to important features of the input image sequences. Smaller networks with less operations can then be used to classify image sequences in real-time.

One way to reduce the feature space of image sequences is to calculate optical flow images between consecutive frames. This way, the network can focus on movements in the imagery and does not need to find relevant features in the original image sequences on its own. However, optical flow images need to be calculated as well, additionally increasing the computational effort of the system.

Moreover, depending on the resolution of the images and the overall movements in the scenery, optical flow images might miss information about fine grained movements, like typing on a smartphone, as not only the fingers might move to type, but the hands and arms move as well.

A different approach to reduce the feature space of the input data can be to calculate body keypoints of a person and use these coordinates as input for the action recognition [Sha16], [Zha17], [Beh18]. Those body keypoints can further be used to crop out important parts of the images that cannot be displayed with body keypoints alone [Ché15], [Das18], [Era19]. However, calculating these body keypoints increases the overall computational cost again.

As most object interactions are performed with the hands, the hand image crops can hold valuable information about current interactions. On the contrary, the body keypoints represent the driver's body pose and can be used

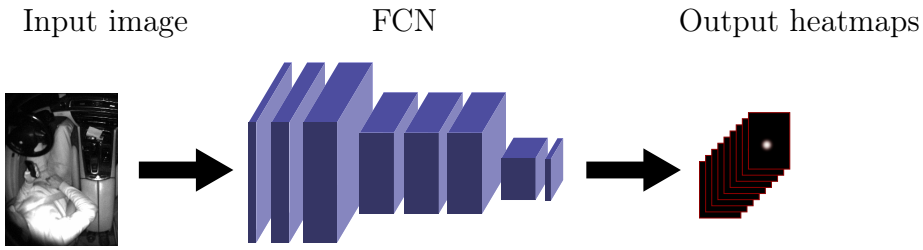


Figure 5.10: Body keypoint extraction concept with fully convolutional neural networks.

to reduce the input search space for a neural network for classifying more general actions performed with the whole body, like entering a car, while the combination of body keypoints and hand crops is suitable for combined actions like strapping the seatbelt. In contrast to the mentioned approaches, the proposed method therefore relies only on the body keypoints and the hand crop images of the driver, in order to reduce the computational effort of the neural network and focus on the most relevant features related to the task of action and object interaction classification.

5.3.1 Body Keypoints

Body keypoints are coordinates that describe the location of different body parts in images. Papers like [Dem09], [Beh18], [Mar18] showed that body keypoints can play an essential role to describe the driver's body pose. The body keypoints can either be $2D$ coordinates representing the locations in $2D$ images or they can be expressed as $3D$ coordinates, if either the network is able to calculate these, or additional information is provided with which the $2D$ coordinates can be transformed. To calculate body keypoints a fully convolutional neural network is used to detect the $2D$ locations of different body parts in the imagery. Figure 5.10 shows an example of a fully convolutional neural network extracting heat maps for the keypoint localization from an image. The neural network consists of multiple convolution layers, pooling layers and non-linearities. In contrast to a classification network, no fully-connected layers are used at the end of the network, resulting in a fully convolutional neural network which calculates heat maps, showing the prediction for the locations of one body part each. These heat maps are then further used to calculate the exact prediction of the locations of the body parts.

The final keypoint location $\vec{r}_c$ is computed by calculating the center of mass around the area of the maximum of the current keypoint heat map.

$$\vec{r}_c = \frac{1}{M} \sum_i m_i \cdot \vec{r}_i \tag{5.6}$$

$$\text{with} \quad M = \sum_i m_i. \tag{5.7}$$

As this calculation of the keypoints from the heat maps is not considering if a keypoints is visible, an additional confidence value is calculated for each keypoint, describing if the keypoint is visible or not.

The confidence of a calculated body keypoint depends on the sum of the values of the corresponding heat map in a certain range around the calculated coordinates. The sum is normalized by a normalization factor to limit the range of the confidence to $[0, 1]$. If the confidence value is smaller than a certain threshold, the keypoint is marked as not located, as the detection of the network for this body keypoint is not sufficient.

After calculating the $2D$ location and the confidence value of each body part, the $3D$ coordinates can be calculated by transforming the $2D$ coordinates by considering the depth values at these locations. Beginning with the homogeneously normalized transformation

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} \frac{X}{Z} \\ \frac{Y}{Z} \\ 1 \end{pmatrix} \tag{5.8}$$

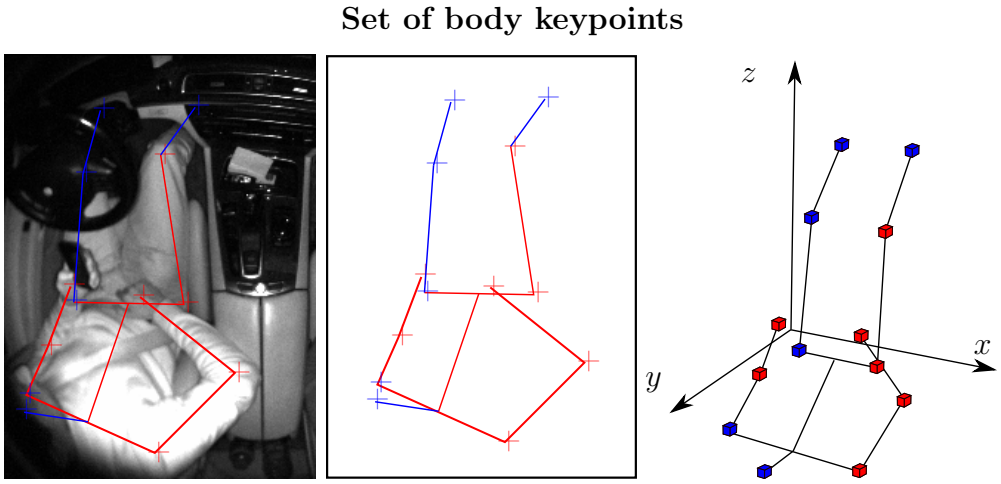
to transform image coordinates to normalized camera coordinates, the equation can be expanded to calculate the $3D$ coordinates X and Y in camera coordinates with

$$X = Z \cdot \frac{u - u_0}{f_x} \tag{5.9}$$

$$Y = Z \cdot \frac{v - v_0}{f_y} \tag{5.10}$$

if the depth value Z and the intrinsic camera parameters f_x, f_y, u_0 and v_0 are given.

Basic human body parts to analyse the drivers body pose, actions or general behaviour are the *head*, the *shoulders*, the *elbows*, the *hands*, the *hips*, the *knees* and the *feet*. These body keypoints are visualized in figure 5.11.

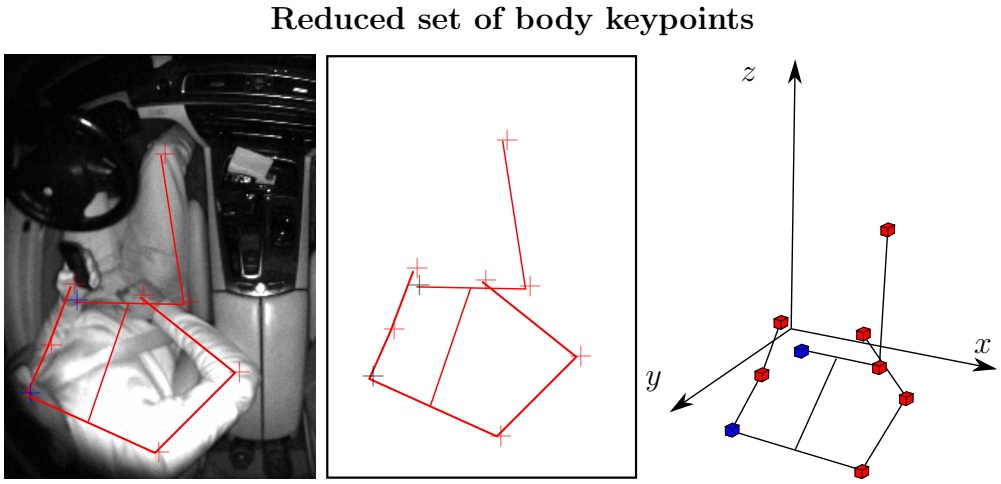


(a) Original image with body keypoints, (b) $2D$ body keypoint locations, (c) Transformed $3D$ body keypoints.

Figure 5.11: Visualization of the body keypoints of the driver.

Figure 5.11a shows an image of a driver with the body keypoint locations and their connections. Figure 5.11b illustrates the $2D$ body keypoints while figure 5.11c shows the transformed $3D$ keypoints. The red colored points illustrate the visible body keypoints, while the blue colored points illustrate the body keypoints which are not directly visible in the image.

However, some of these body parts are not visible in the current camera setting most of the time, because they are covered by the interior of the car or they are located beyond the image frame. The *left knee* is covered by the steering wheel in the majority of the cases, while both *feet* are usually located in the footwell of the car, which is not visible in this camera setting. The *head* of the driver is not captured by the camera most of the time. These four body parts are not detected by the system as they are not visible in most cases, resulting in a reduced subset of nine body parts to detect, consisting of the *shoulders*, the *elbows*, the *hands*, the *hips* and the right *knee*. Figure 5.12 shows the reduced set of body keypoints of the driver on the original image. Figure 5.12b shows the reduced set of $2D$ body keypoints while figure 5.12c shows the reduced set of transformed $3D$ keypoints. Again, the red colored points illustrate the visible body keypoints, while the blue colored points illustrate the body keypoints which are not directly visible in the image.



(a) Original image with reduced set of body keypoints, (b) Reduced set of $2D$ body keypoint locations, (c) Reduced set of transformed $3D$ body keypoints.

Figure 5.12: Reduced set of body keypoints.

Finally the nine keypoints are grouped to a keypoint-vector with three elements for each $2D$ keypoint and four elements for each $3D$ keypoint resulting in a 27 element and a 36 element long vector for the $2D$ keypoints and $3D$ keypoints respectively.

Keypoint results

To evaluate the ability of a network to classify the proposed action based only on the body keypoints of a driver, RNNs are trained to classify the 3D body keypoints of the different validation splits of the dataset. The networks consist of a fully connected layer, preprocessing the 3D body keypoints, an LSTM network and a second fully connected network to classify the body keypoints per timestep. The networks were trained with the temporal augmentation technique proposed in chapter 5.2 and evaluated on the mean prediction of the last three classifications of a sequence. The results of the cross validation are shown in table 5.3. With a mean balanced accuracy score of 0.72, a mean F1-score of 0.678 and a mAP of 0.738 the network performs comparably to the network trained on amplitude image sequences and worse than the networks trained on depth and optical flow image sequences.

	Acc		F1		mAP	
	mean	std	mean	std	mean	std
Keypoints	0.720	0.014	0.678	0.026	0.738	0.027

Table 5.3: Cross validation results of the networks trained to classify actions from 3D body keypoints of the driver.

However, actions with strongly visible movement like *Enter*, *Leave*, *Strap*, *Unstrap*, *Phone Call* are recognised quite well as displayed in the confusion matrix of the validation results in figure 5.13, whereas classes which depend on fine grained visual differences, like *Phone Idle*, *Phone Interaction* and *Bottle Idle*, which can only be distinguished by recognising a phone, moving fingers on a phone, or a bottle, are not recognised well and are most likely confused among each other. As the 3D body keypoints are not capturing these differences, the network is not able to properly distinguish between these classes.

The precision recall curves displayed in figure 5.14 show that the networks are most uncertain with the class *Bottle Idle*. Moreover, the networks are more certain for the classes with much movement of the driver.

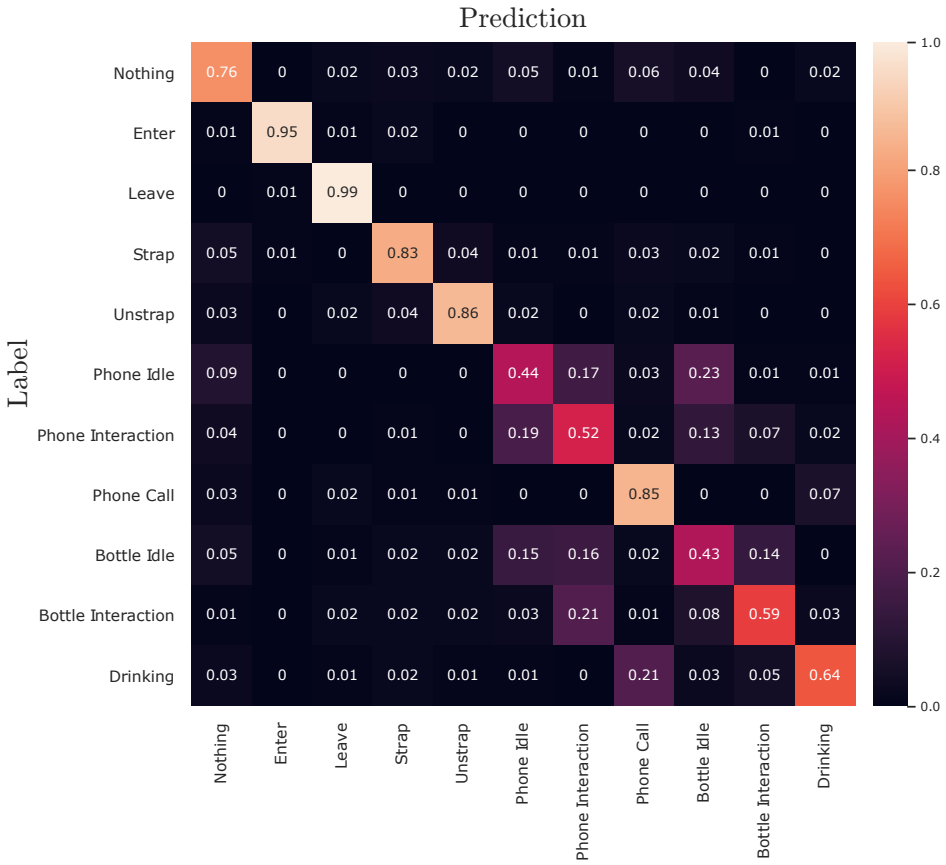


Figure 5.13: Relative confusion matrix of the cross validation results from the networks trained only on 3D body keypoints.

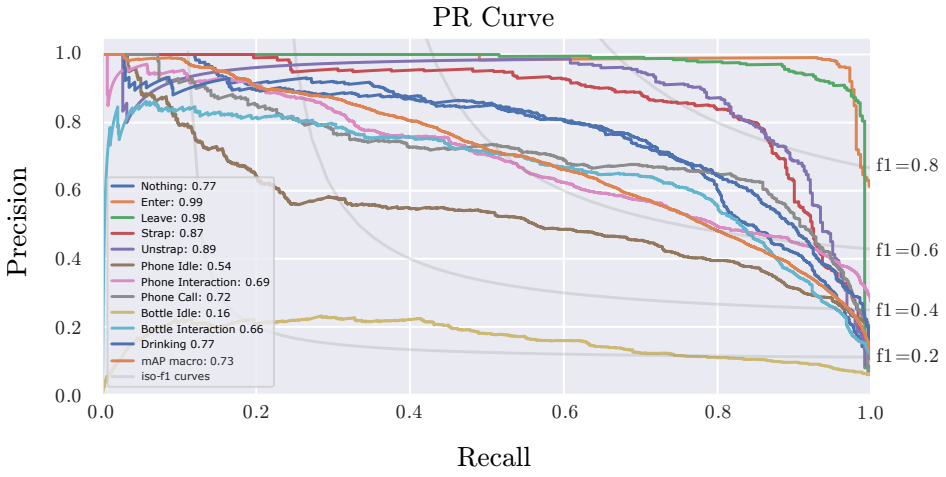
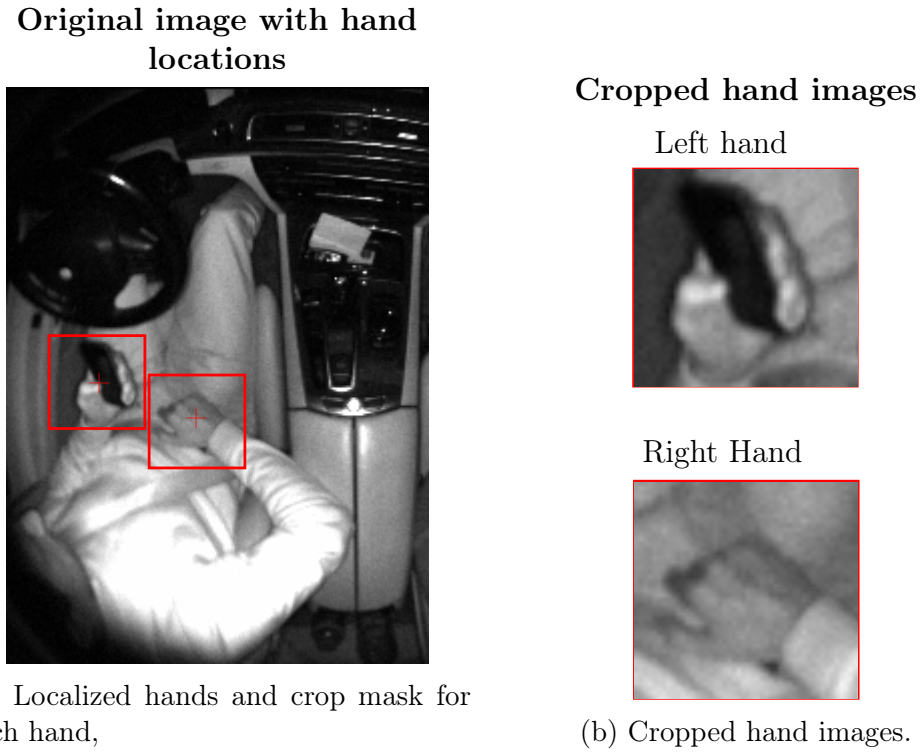


Figure 5.14: Precision recall curves of the cross validation results from the networks trained only on 3D body keypoints.

5.3.2 Hand Crops

As most important object interactions of the driver, like interacting with a smartphone or handling a bottle, involve the hands of the driver, most meaningful features to recognise these interactions are found in the image area of the driver's hands. In order to reduce the image search space and to recognise these interactions, hand crops of a specific size are cropped out of the original image. Of all $2D$ keypoints found by the keypoint localizer network, the $2D$ locations of both hands are extracted and used as a reference to crop out subimages showing only one hand respectively. Figure 5.15 shows an example of the hand cropping with $2D$ hand keypoints. The detected hand locations are shown along the crop borders in figure 5.15a. The results in figure 5.15b show perfectly cropped out hands, where in the left hand a smartphone is held and the right hand is empty.



(a) Localized hands and crop mask for each hand,

(b) Cropped hand images.

Figure 5.15: Visualization of the hand crop concept.

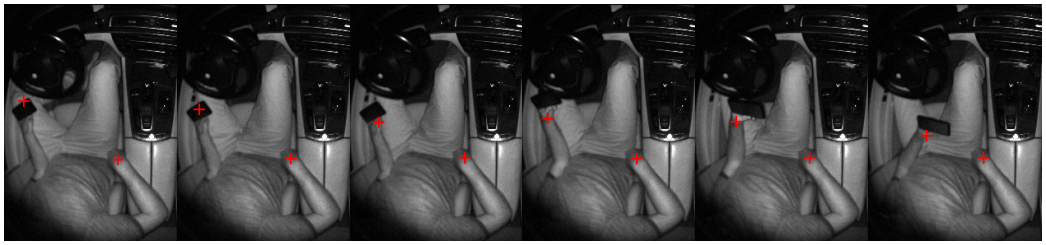
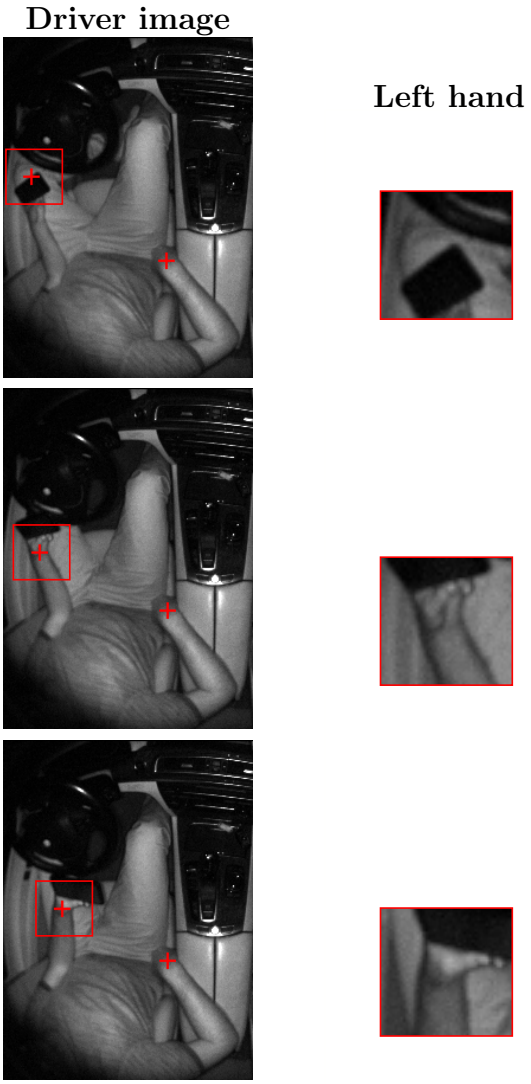


Figure 5.16: Examples of hand locations. Despite similar hand locations and visible similarity of the hands, the detected locations of the hand varies.

However, although the hand keypoints are optimally localized in the example shown in figure 5.15, not all keypoints are detected that precisely. Figure 5.16 shows six example images with the detected hand locations. Especially the detected locations for the left hand vary between the real hand location, the left wrist and the smartphone located in the left hand. This variation results in different cut outs of the hands, as seen in figure 5.17. Figure 5.17a shows three examples of hand localizations and the corresponding crop box. Figure 5.17b shows the resulting crops of the left hands. These images show the impact of the precision of the hand localization to the crop images. The variations result in different cut outs, which might display only parts of the hand or the object in the hands. As not every frame of the dataset can be labeled, the keypoints need to be generated by the keypoint detector network. Moreover, it is very likely that these variations appear in new data frames because of a different environment or camera setting and therefore need to be considered in the training process.



(a) Driver image with hand coordinates and left crop box,

(b) Cropped left hand.

Figure 5.17: Hand image crop examples. Different hand localizations result in different cut out regions of the hands.

5.3.3 Hand Crop Normalization

In order to reduce the variation of hand orientation and size resulting from different positions in the space of the interior cabin, the hand cut outs are rotated and the image size is scaled depending on a subset of the detected body keypoint locations. This ensures that the hands in the cropped out image patches have always the same orientation and size, if the depending variables are calculated correctly.

Rotation Normalization

To calculate a rotation angle to rotate the hand crops a reference point needs to be defined from which a line can be drawn to the hand location which spans an angle with the vertical axis. The most natural point to be used as a reference point is the corresponding elbow location. However, the elbow might not be able to be detected, because it is not visible. The second most natural point would be the corresponding shoulder, but this point might not be able to be detected as well. To get a more static reference point, the mean location in between those two points is calculated. If one of these points is not located, the other one is used. If both points are not located, the rotation angle is set to zero and the image will not be rotated. Figure 5.18 shows the hand point P_H, the corresponding elbow point P_E, the corresponding shoulder point P_S, the resulting reference point P_R and the resulting rotation angle σ.

Finally the image is rotated about the rotation angle in degrees

$$\sigma = -\frac{\arctan\left(\frac{P_{Rx}-P_{Hx}}{P_{Ry}-P_{Hy}}\right)\cdot 180}{\pi} \tag{5.11}$$

to normalize the orientation of the current hand.

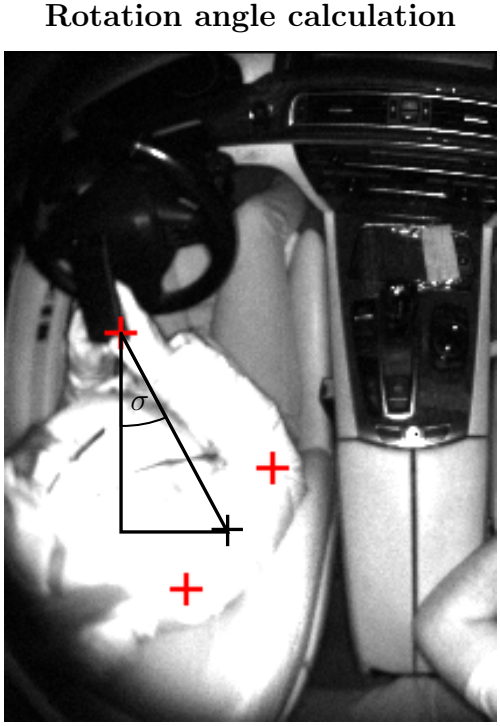


Figure 5.18: Visualization of the angle calculation for rotation normalization of the right hand of a driver. The middle point between the elbow and the shoulder of the driver's right side is used as a reference point for the rotation of the driver's right hand.

Size Normalization

Additionally to the rotation normalization, the images are scaled depending on the distance of the hands to the camera. Like this, the hands always have the same size in the images independently of their position in space of the car cabin. The distance

$$d = \sqrt{P_x^2 + P_y^2 + P_z^2} \tag{5.12}$$

of the hand to the camera is calculated from the $3D$ body keypoint of the hand. The scaling factor to scale the image size

$$s = \frac{d}{c_d} \tag{5.13}$$

is calculated with a constant c_d which defines the distance a hand needs to have to not being scaled. The image is then scaled by the calculated scaling factor s. Hand images with a keypoint too close or too far away are not scaled.

Normalized Hand Crop Examples

Cut out images of hands resulting from different images and hand localizations are shown in figure 5.19. Figure 5.19a show correctly localized cut out images of hands. The hands are located in the middle of the images and have about the same size. The wrists of the hands are located at the bottom of the images. Figure 5.19b shows different sized hand images caused by different depth measurements. The size of the hands can vary because of variations in the depth measurement. Figure 5.19c shows examples of hand cut out images with misplaced hand locations. It can be seen that the hands are not located in the middle of the images, meaning that the hand is not localized at the correct position, but slightly next to the hand. These mislocalizations result in a hand cut out image with an offset, only showing the hand, if the hand is localized close to the real hand location. Finally, figure 5.19d shows hand cut out images with hands which are located near the border of the original image. Cutting out the image in a way that the hand is localized in the middle of the cut out results in images which stick out beyond the original image border. This part of the images are padded with zeros in order to conserve the predefined image size.

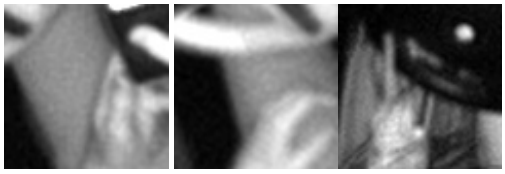
5.3.4 Reduced Feature Dataset

The dataset with reduced features originates from the dataset described in section 5.1 and consists of the files holding the body keypoints for each frame, and images showing only the normalized hand patches. The body keypoints were calculated beforehand, to reduce the computational effort during the training process. The same applies for the generation of the hand cut outs.

Examples of the reduced feature dataset are shown in figure 5.20, figure 5.21b and figure 5.22 along the original images to show the origin of the data. The shown examples of the reduced feature dataset of the actions and object interactions recognition consists of the amplitude images, the detected body keypoints of the driver, visualized by marking the coordinates in the amplitude images, and the normalized amplitude hand cut outs of each example frame directly under the original amplitude frame. Additionally, the bounding boxes of the hands cut outs of the first image are shown at the first amplitude image of each sequence example to visualize the normalization of the hand cut outs. Even though the visualized examples show only amplitude images, the corresponding depth images are also part of the reduced feature dataset.

(a) Examples of different normalized cut out images of hands,

(b) Examples of different cut out images of hands with different sizes caused by depth measurement variations,

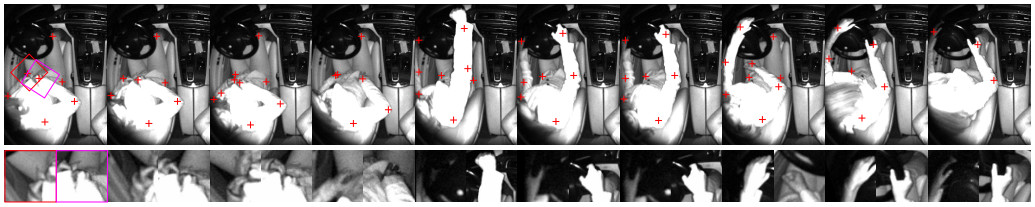


(c) Examples of different cut out images of hands with mislocated hand positions,

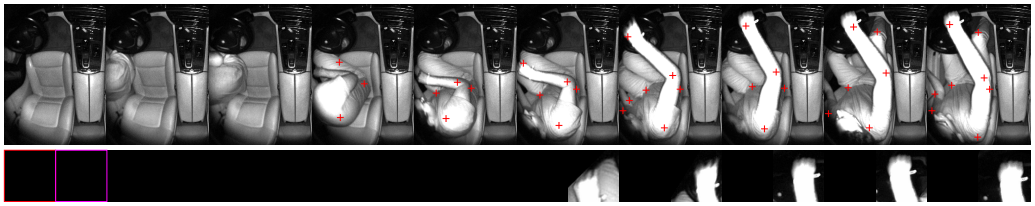
(d) Examples of different cut out images of hands located at the image border with zero padding.

Figure 5.19: Examples of different cut outs of hands with different characteristics.

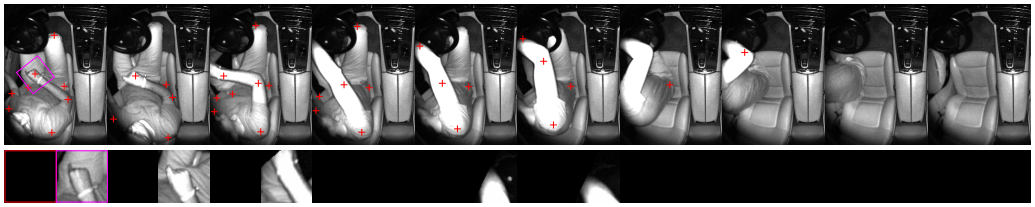
Figure 5.20a shows the reduced features for the example of a driver performing no action or object interaction. Most of the body parts with a corresponding body keypoint are visible in the scene. The left shoulder and left elbow are hidden behind the rest of the body in the first frames. Nevertheless, both are detected in the third frame roughly at their correct position. The right hip is covered by the right arm in the sixth and seventh frame. Moreover, only the left and right hand as well as the right elbow are detected by the body keypoint detection network in the last frame, as the driver leans forward.



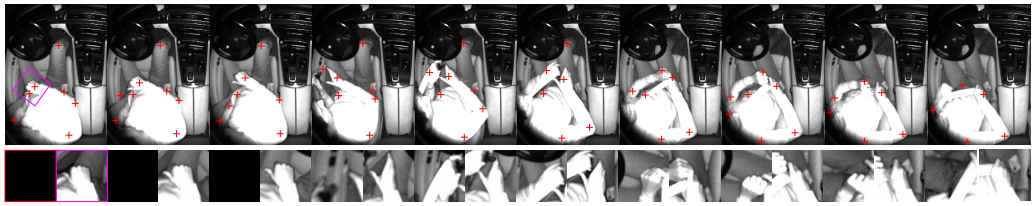
(a) Reduced features of a driver,



(b) Reduced features of a driver entering a car,



(c) Reduced features of a driver leaving the car,



(d) Reduced features of a driver fasten the seat belt,

(e) Reduced features of a driver unstrapping the seat belt.

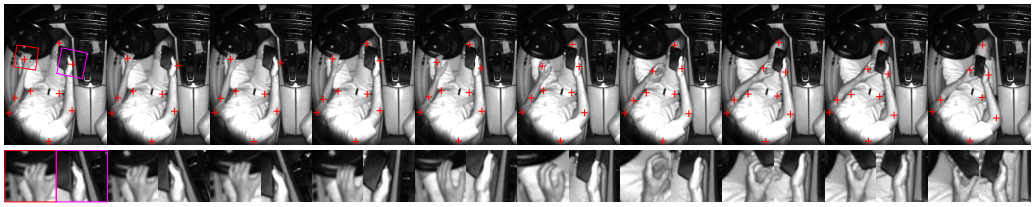
Figure 5.20: Reduced features of driver actions.

An example of the reduced feature data of a driver entering the car is shown in figure 5.20b. At first, the driver seat is empty and no body keypoints are detected and therefore no hands are cropped out. As more and more of the driver's body gets visible in the scene as the driver enters the car, more keypoints are detected until all body keypoints of the driver, except the left hand which is not visible in the images, are detected correctly. As soon as the right hand is detected it is cut out from the image and visualized underneath the original image.

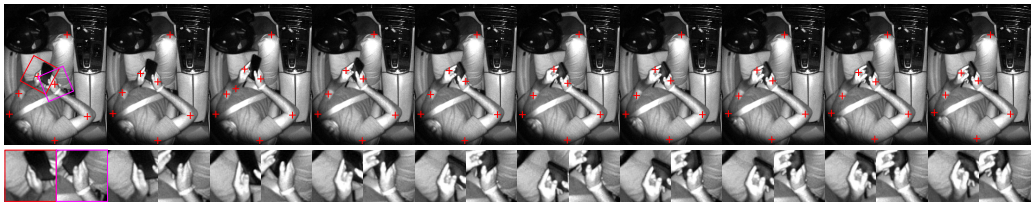
Next, figure 5.20c shows the reduced feature data of a driver leaving the driver's seat. The left hand of the driver is not visible in this scene. The right hand is correctly detected in the first three frames. Later, the right hand is detected in two images, though not visible in the scene anymore, resulting in a cut out of the right forearm. As the driver leaves the car, less body keypoints are detected until the driver is not visible anymore and the driver's seat is empty. At this point no body keypoints are detected.

A scene where the driver fastens the seat belt is shown in figure 5.20d. At the beginning, the driver reaches behind to grab the seat belt, resulting in the left hand not being visible and not being detected in the scene at this frame. As the seat belt is pulled out of the pillar loop, the left hand gets detected, although not quite precisely in the first frame the hand is visible. The left hand is detected at the right above the correct location of the hand, resulting in a cut out where the hand is in fact visible, but not in the center of the image. In the following frames, the hands are located correctly resulting in crop outs where the hands are visible completely and the handover of the seat belt tongue from the left to the right hand is visible in the cropped hand images. Following, the seat belt tongue is lead to the seat belt buckle with the right hand. In the last frame, the left hand is not detected correctly, resulting in a crop out of the lap instead of the hand. The remaining body keypoints are detected at their correct location, if their corresponding body part is visible.

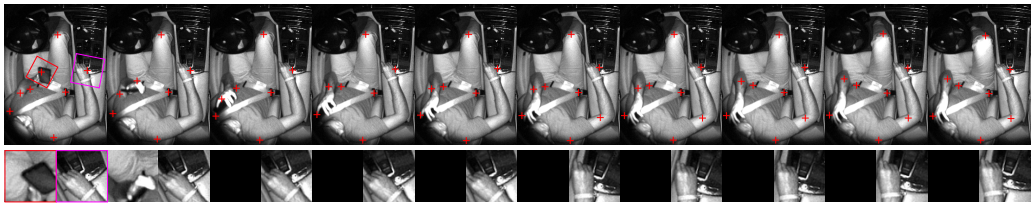
Figure 5.20e shows an example of a driver unstrapping the seat belt. In the first frame the right hand is not detected, while the left hand is detected at its wrist, resulting in an incomplete cut out of the hand. In the following frames, the hands get detected more precisely and it can be seen that the seat belt tongue is unlocked from the buckle with the left hand and lead back to its pillar loop. In the last frames, where the left hand is located near the left image border, the hand is detected at the forearm, resulting again in cut outs of the forearm.



(a) Reduced features of a driver holding a phone,



(b) Reduced features of a driver interacting (typing) on a phone,

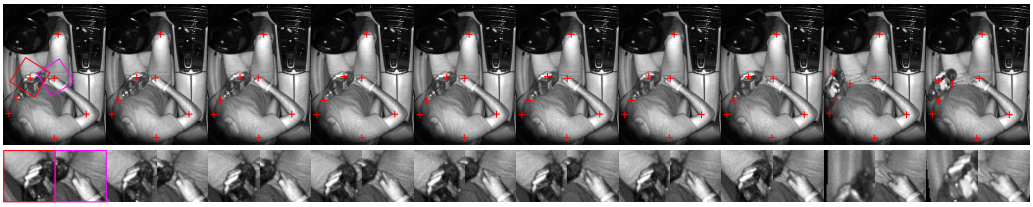


(c) Reduced features of a driver taking a phone call.

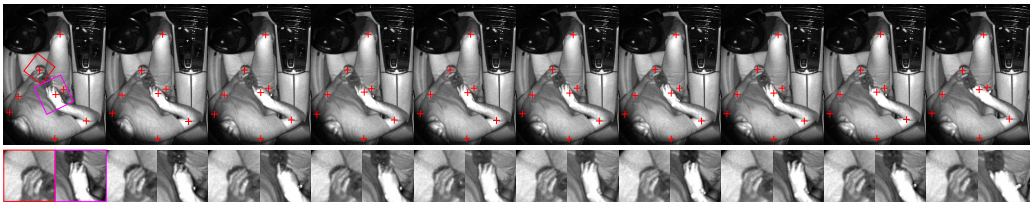
Figure 5.21: Reduced features of a driver with phone interactions.

Figure 5.21 shows one example for each phone interaction class, respectively.

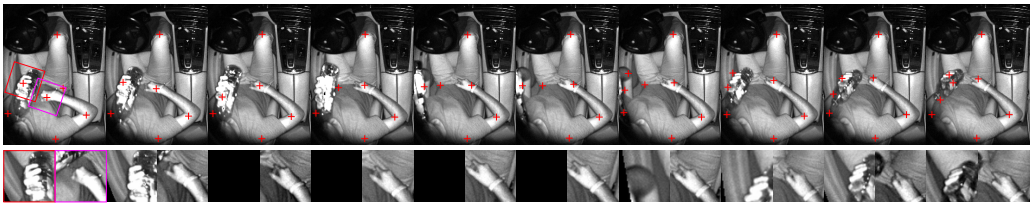
Figure 5.21a shows the reduced feature data of the driver while holding a phone. Both hands are located correctly, resulting in the hand crops showing the hands. In the course of the sequence, the right hand is moved slightly while the left hand leads towards the right hand and the phone. Both hands are located correctly in all of the shown frames, resulting in a sustained sequence of hand cut outs for both hands. The remaining body keypoints are detected correctly in all the images of the scene. Merely the right knee is not located in two frames as the view on the knee is partly blocked by the phone in these images.



(a) Reduced features of a driver holding a bottle sequence,



(b) Reduced features of a driver interacting (opening) with a bottle sequence,



(c) Reduced features of a driver drinking from a bottle.

Figure 5.22: Reduced features of a driver with bottle interactions.

The direct interaction with a phone is shown in figure 5.21b. Again, both hands are sustainably detected in the sequence. In the hand patch sequence, it is easier to see that the fingers are moving over the display of the phone, showing that the driver is directly interacting with it. The hip locations are not detected as both hips are covered by the arms of the driver.

The reduced features for the example sequence of the driver taking a phone call is shown in figure 5.21c. In the first two frames, the left hand with the phone is located correctly. From the third frame on, the left hand with the phone is not detected anymore, resulting in no crop out of this hand for these frames. In these images, the keypoints for the right hand are located at the side of the hand, resulting in slightly shifted images of this hand. As the elbow is not detected in the first images, the rotation of the cut out of the right hand depends only on the location of the right shoulder detection, resulting in a tilted hand cut out compared to the later frames where the elbow is detected and approximately located correctly.

Examples of the reduced features for the different interaction classes with a bottle are shown in figure 5.22.

The first example displayed in figure 5.22a shows the body keypoints and the hand patches of the driver while holding a bottle. Both hands are located correctly, though the left hand is mostly covered by the bottle and only single fingers are visible. The other body parts are located correctly.

Secondly, figure 5.22b shows the reduced feature data for a scene where the driver opens a bottle. Both hands are located correctly in the scene, resulting in correct crop outs of both hands. The remaining body keypoints are again located correctly.

An example of the driver drinking from a bottle is shown in figure 5.22c. At the beginning both hands are classified correctly and the cut outs of both hands show the hands. As the driver lifts the hand with the bottle to drink from it, the hand disappears from the scene, resulting in correctly not detecting the hand by the keypoint detection network. As soon as the bottle is put down, the hand is visible again and detected correctly. Therefore, the hand with the bottle is cropped out again. Similarly, the left elbow is not visible in the beginning of the scene and only gets visible and correctly detected when the arm is lifted.

5.3.5 Reduced Features Action Recognition System

The system to recognise actions and object interactions of a driver consists of a Time-of-Flight camera observing the front seats of a vehicle, different image and coordinate transformations and a network to classify the actions and objects interactions. A full overview of the system is shown in figure 5.23. The images from the Time-of-Flight camera are cropped in order to show only the driver's side. With a body keypoint detection network, 2D-body keypoints are calculated and transformed to 3D-body keypoints with the distance of the body parts to the camera obtained from the depth image as described in chapter 5.3.1. With the 2D-body keypoints and the depth image of the driver the location and rotation of the hands of the driver are known as well as the distance to the camera. The normalized hand images are cut out of the original image of the driver as described in chapter 5.3.2. The normalized hand cut outs and the 3D-body keypoints are then fed to the network. The network to classify actions and object interactions of a driver is a variant of a many-to-many CNN-RNN combination, classifying the in-

put of each timestep with respect to the previously calculated classifications. The full network concept is shown in figure 5.24.

For each timestep, a convolutional neural network extracts features from the hand patches. The network extracts spatial features from both hand patches separately, resulting in one feature vector for each hand. This CNN consists of three layers with 8, 12, and 24 kernels and a kernel size of 3×3.

The spatial features of the $3D$ body keypoint vector of the current timestep are analysed with a small fully connected network with two hidden layers with 32 and 24 neurons, also resulting in a feature vector. The extracted feature vectors for both hands are then concatenated with the extracted features from the keypoints. This combined feature vector, holding spatial information of both hands and the $3D$ body pose of the driver, is further analysed by an LSTM network (2.1.3) which consists of two LSTM cells with a size of 32 connected in series. This recurrent networks extracts temporal features from the combined feature vector at each timestep, with respect to the previously calculated combined feature vectors. Finally, the extracted temporal features are classified with a fully connected neural network with one hidden dimension with 24 neurons and an output layer with 11 neurons and a softmax classifier for each timestep. This way, the network provides predictions about the current action or object interaction at each timestep. If a hand is not visible, the corresponding feature vector elements are set to zero. Similarly, keypoints coordinates are set to zero as well, if the keypoints are not detected.

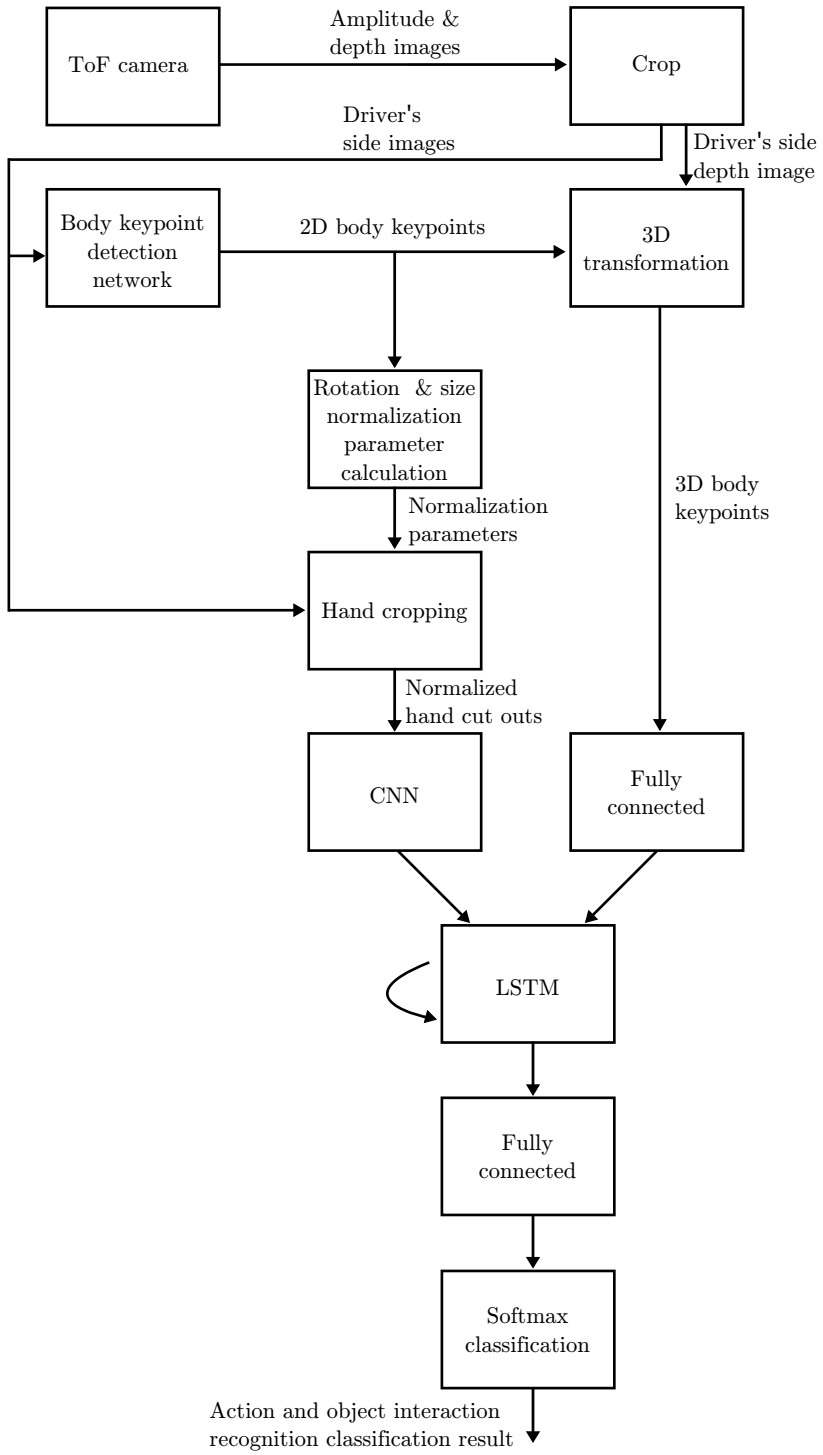


Figure 5.23: Action and object interaction recognition system overview with reduced features and normalized hand cut outs.

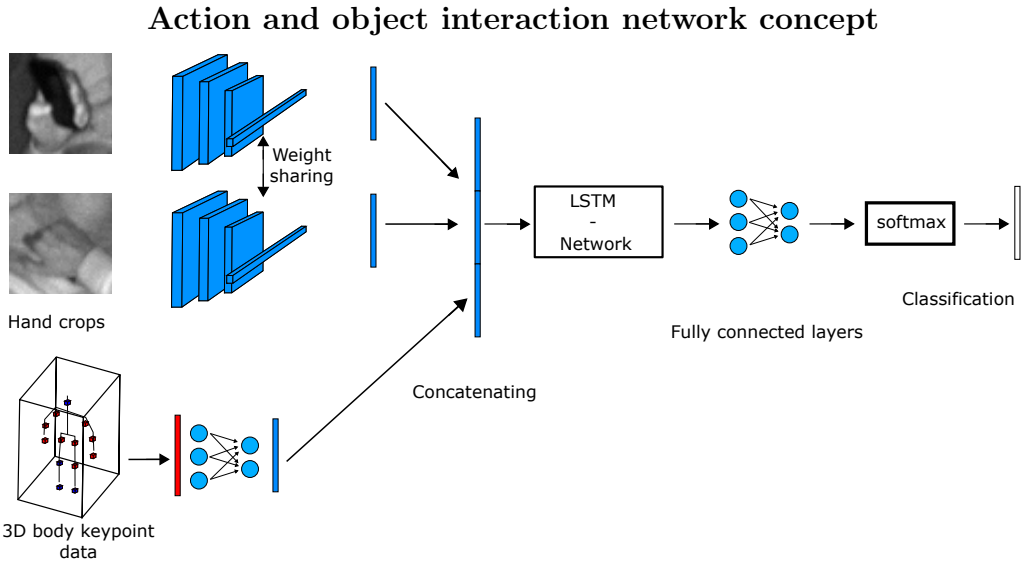


Figure 5.24: Concept of the driver action and object interaction network. The hand cut outs of the driver's hands as well as the 3D body keypoints of the driver are used as input for the action recognition network. A CNN extracts features of the hand cut outs which are combined with the processed 3D body keypoints. These combined features are further processed by an LSTM network. The output of each timestep is then classified by a fully connected network and a softmax classifier.

5.3.6 Reduced Features Action Recognition Results

To evaluate the proposed system, the networks were trained on the driver's calculated body keypoints as well as hand cut outs of the driver. The sequences were temporally augmented as described in chapter 5.2 and evaluated on the mean predictions of the last three examples of an example sequence. Two kinds of networks were trained on the different validation splits. One set of networks was trained to recognise the different action and object recognition classes from the body keypoints and the unnormalized hand crop images, while the second set of networks was trained on the body keypoints and the normalized hand crop images of the driver. Each set of networks was trained on amplitude, depth and optical flow images with the proposed systematic time augmentation described in section 5.2. The optical flow images are calculated between the image of the hand crop sequences.

The results of the proposed action and object interaction system with reduced features are shown in table 5.4. The networks trained on rotation

	Acc		F1		mAP	
	mean	std	mean	std	mean	std
Amplitude						
hand crops	0.782	0.027	0.749	0.026	0.812	0.024
normed hand crops	**0.819**	0.031	**0.788**	0.024	**0.852**	0.022
Depth						
hand crops	0.777	0.024	0.745	0.033	0.810	0.021
normed hand crops	**0.798**	0.022	**0.760**	0.034	**0.831**	0.027
Flow						
hand crops	0.716	0.012	0.683	0.028	0.734	0.036
normed hand crops	**0.727**	0.059	**0.705**	0.052	**0.782**	0.041

Table 5.4: Cross validation results of the action and object interaction recognition system with reduced features with and without normalized hand cut outs

and size normalized hand crops perform better in terms of balanced accuracy, F1-score and mAP on amplitude, depth and optical flow images compared to the networks trained on unnormalized hand crops. The network trained on body keypoints and normalized amplitude hand crop sequences shows the best results, compared to the networks trained on depth or optical flow hand crop image sequences. Reducing the image size also reduces the variance of the images, which enables the network to learn better features for the task of the action and object interaction from amplitude hand image crops. Using optical flow image crops results in the worst network performance on the validation data. As the hands are moved through the scene, the optical flow images of the hands capture rather movements around the hands than movements of the hands. Additionally, the body keypoints slightly move, even if the corresponding body part does not move in the scene, as image noise is present in all images. As a result, additional artificial movement is visible in consecutive hand cut outs which is present in the optical flow images and contributes to the worse results of the networks trained with optical flow image patches of the driver's hands. Moreover, objects cannot be recognized from optical flow images generated by moving hand crops as well as from amplitude or depth image crops.

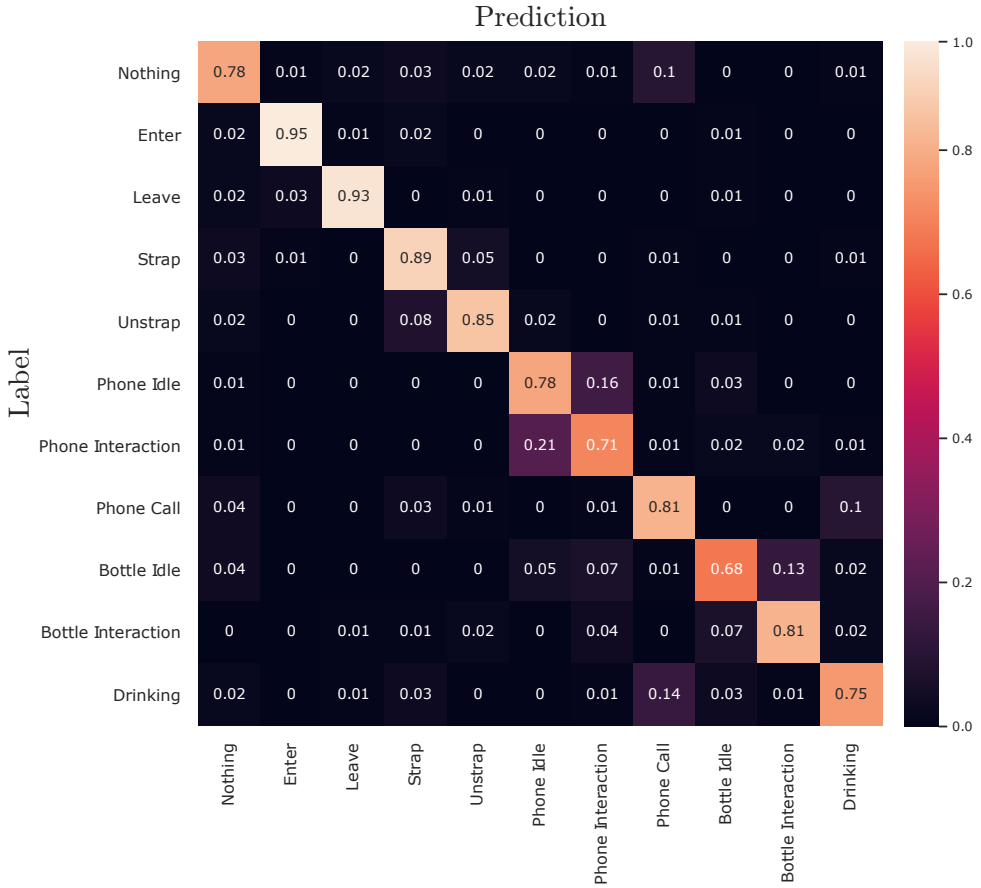


Figure 5.25: Confusion matrix of validation results from the action and object interaction recognition systems trained on 3D body keypoints and normalized amplitude hand crop images

Figure 5.25 shows the confusion matrix of the classification results from the validation examples. The least recognized class *Bottle Idle* is most likely confused by the networks with the class *Bottle Interaction*. Moreover, some examples of the classes *Phone Idle* and *Phone Interaction* are mistaken. This confusion is most likely the result of the similarity of both classes, as they can only be distinguished by recognising the finger movements over the display or keyboard of the phone. However, both classes are recognized correctly most of the time. Another confusion between classes occurs between the classes *Phone Call* and *Drinking*. This confusion might result from the similarity of the sequences, as a movement from at least one hand to the driver's head happens in both actions.

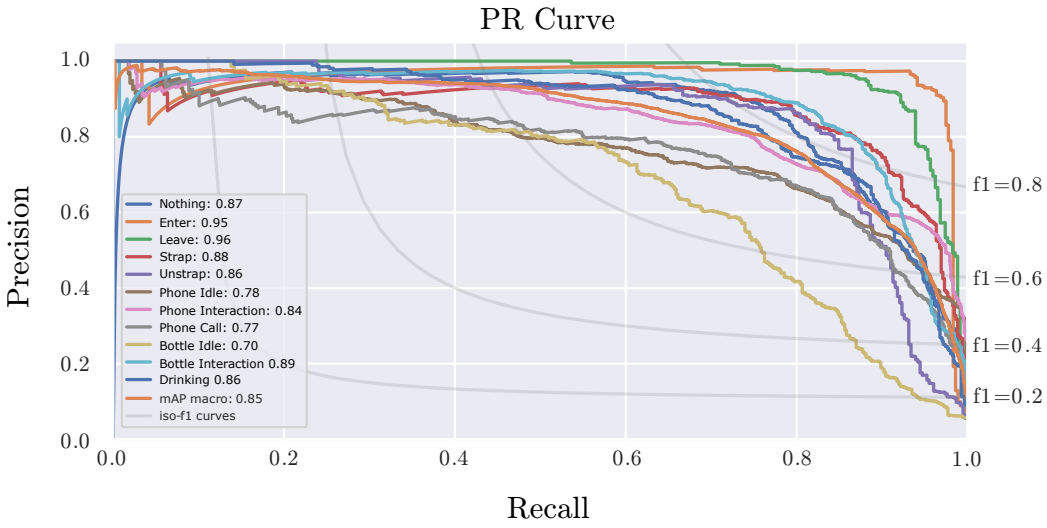


Figure 5.26: Precision recall curves of validation results from the action and object interaction recognition systems trained on 3D body keypoints and normalized amplitude hand crop images

The precision recall curves of the predictions on the validation examples are shown in figure 5.26. The curves show that the classifiers are most confident predicting the classes *Enter* and *Leave*, as these classes are the classes whose examples show the most movement of the driver. Moreover, the absence of the driver's body keypoints at some point of these sequences can be a strong feature for the classifiers to recognise these actions. With a minimal average precision of 0.70 for the class *Bottle Idle*, the classifiers are overall quite sure about the classifications.

The remaining confusion matrices and precision-recall curves of the validation results of the networks trained with the reduced dataset with and without the proposed hand normalization are shown in chapter B.

5.4 Action and Object Interaction Recognition System Summary

In this chapter, different approaches for classifying actions and object interactions performed by car drivers were proposed. As an extension to the driver's state classification described in chapter 4, the driver's state is analysed by recognising fine grained actions performed by the driver.

When using full images of the driver to train an action recognition system, the ability of the networks to classify the actions can be enhanced by selecting a proper input image format. Amplitude images showing the whole scene can be too variable for too small networks to learn the right features from them. Especially if the size of the dataset is also small, the performance of the networks can be enhanced by selecting an input image format which is less variable, like depth or optical flow images as shown in the evaluation of chapter 5.2. Moreover, it was shown that the performance of the networks can further be enhanced by augmenting the temporal component of the sequences systematically.

When training only on body keypoints of the driver, the networks are able to recognize actions with much body movement of the driver as shown in chapter 5.3.1. Fine grained object interactions cannot be recognized as well as with full images, as contextual information, like the presence of a phone or a bottle, is not existent in body keypoints. Moreover, small movements, like finger movements on a smartphone, cannot be recognised, which makes the task of recognising the interaction with a phone more difficult. Combining the $3D$ body keypoints and subimages of the hands of the driver in a network resulted in networks which are capable of recognising the proposed actions and object interaction reliably, as shown in chapter 5.3. When training the proposed system on the reduced features consisting of hand crops and $3D$ body keypoints of the driver, it is preferable to use amplitude image cut outs of the hands as this image format holds more contextual information about the current hand interactions which can be extracted by small CNNs compared to image formats like depth images or optical flow images. Moreover, normalizing the hands to always have the same rotation and size in the image crops improved the capability of the networks even more.

Table 5.5 shows a comparison of the best proposed networks. It can be seen that the networks trained on 3D body keypoints and normalized amplitude hand crops result in the best mean balanced accuracy and F1-score of

	Acc		F1		mAP	
	mean	std	mean	std	mean	std
Amplitude time augmentation	0.701	0.033	0.674	0.020	0.729	0.017
Depth time augmentation	0.748	0.048	0.715	0.055	0.783	0.045
Optical flow time augmentation	0.793	0.018	0.769	0.019	**0.858**	0.026
Keypoints	0.720	0.014	0.678	0.026	0.738	0.027
KP + normed amplitude hand crops	**0.819**	0.031	**0.788**	0.024	0.852	0.022
KP + normed depth hand crops	0.798	0.022	0.760	0.034	0.831	0.027
KP + normed flow hand crops	0.727	0.059	0.705	0.052	0.782	0.041

Table 5.5: Cross validation results of the best evaluated networks for action and object interaction recognition

the evaluated networks, while the mAP score is comparable to the networks trained on optical flow images showing the movement of the complete driver side.

In this chapter, it was shown how an action and object interaction recognition system for driver state monitoring can be enhanced by applying temporal augmentation to the training process of recurrent neural networks. Furthermore, a system was developed to reduce the input feature space to $3D$ body keypoints and hand cut outs of the driver. The $3D$ body keypoints contain valuable features for recognising different actions, if the action examples contain much movement. Fine grained actions with small movements require additional contextual information to be recognised reliable. Combining $3D$ body keypoints with amplitude image crops of the driver's hands enhanced the capability of the networks to not only recognise actions with much movement present, but to recognise fine grained object interactions as well.
Moreover, selecting the right input image format to classify the action and object interactions reliably is crucial for small neural networks. While the evaluated networks learned to recognise the presented actions best from optical flow images, when using full images, using amplitude images is preferable, when using hand cut outs as input images. This results from the circumstance, that the small neural networks can rather classify actions correctly

from full images of the driver, if unnecessary information, as present in complete amplitude images, is filtered out by calculating optical flow images which focus only on movements of the driver. However, when focusing on cut outs of the driver's hands contextual information, like the textures of the object the driver is holding, is lost by calculating optical flow images from hand crop sequences. Therefore, amplitude image cut outs are preferable for training small neural networks, if the image cut outs are chosen in a way that they cut out unnecessary information.

CHAPTER 6

Improving Continuous Online Action Recognition from Short Isolated Sequences

Classifying continuous data can be challenging, as no explicit information about the start or the end of actions in video streams is given, except the information visible in the image frames. In contrast to most action classification papers, which focus either on classifying isolated actions in video clips showing only one action or detecting specific actions in untrimmed video clips, continuous online action classification aims to recognise different actions in long sequence data, in which multiple classes can occur, as they appear. Therefore, the systems need to classify single frames or short blocks of frames based on information from previous image data.

While most papers on action recognition focus on analysing isolated action clips [Sim14a], [Car17], [Mar19], other works like the papers of [Sin17], [Zol18] focus on online action recognition in untrimmed videos and only few papers focus on continuous action recognition [Mol16]. This can be seen from the amount of different published datasets for the two tasks. Most action recognition datasets include trimmed videos showing isolated action sequences [Kue11], [Abu16], [Car17], [Goy17], [Mar19], whereas only few datasets with untrimmed videos of different actions exist [Jia14], [Cab15]. Moreover, these untrimmed action recognition datasets contain only few or no transitions between actions, making them useful for action detection of single actions in this kind of setting, but not for continuous action recognition, where class transitions are relevant. For action recognition in continuous video data also only very few datasets exist. The dataset described in the paper of [Ste13] shows videos of people preparing salads, while the dataset proposed by [Kue14] shows continuous video streams of people doing daily cooking activities. Another drawback of some action datasets is, that they

contain only few examples of each action, which makes them inappropriate for most deep learning techniques. For example, the dataset proposed by [Ste13] contains 44 to 64 example sequences per class which makes it too small to be usable for pre-training of an action recognition network.

The *Action and Object Interaction* described in chapter 5, creates the basis to recognize not only single, trimmed sequences, but to classify the frames of a continuous video stream showing different actions performed by a driver. However, some body movements of the driver might result in poses which body keypoints are hard to localize or can only be localized reliably with an excessive amount of training data, which is again time consuming to gather. Therefore, in contrast to the *Action and Object Interaction*, the proposed system works only with image sequences showing the driver's seat and optical flow image sequences showing the movements in the driver's seat region and contains no body keypoints or crops of special regions except the driver's seat region. The described system and methods were proposed by the author of this work in [Wey21].

In this chapter, an overview of the obtained data, including the recognition tasks, examples of the different classes and the datasets statistics are given in section 6.1. In section 6.2 an overview of the used network architecture is given and evaluated for a first baseline. In section 6.3 a method for improving the networks performance on continuous data by structuring the input data for training the network, is suggested and evaluated. Finally a new method for handling the hidden states while training the recurrent networks is proposed, evaluated and compared to the previously evaluated methods in section 6.4.

6.1 Isolated Action Data

The dataset consists of several image sequences showing the driver's seat. The actions in the dataset show different basic body movements of a driver.

In order to train and test a continuous action recognition system whose actions can occur in different orders, a partly new dataset was gathered. Some sequences of the *Action and Object Interaction Recognition* dataset proposed in chapter 5.1 were used along additional recorded data sequences. In contrast to the previously used action recognition dataset, the classes are explicitly chosen in a way that they can precede each other and a continuous action stream can be generated by lining up different classes. The classes

Movement Actions	
To Front	Driver leaning towards steering wheel (leaning forward)
To Right	Driver leaning towards passenger side (leaning rightward)
To Back	Driver leaning towards rear seats (leaning backward)
From Front	Driver returning from a forward leaning position
From Right	Driver returning from a rightward learning position
From Back	Driver returning from a backward leaning position
Static	
Empty	Empty driver's seat
In Position	Driver sitting in a driving position
Front	Driver leaned towards steering wheel
Right	Driver leaned towards passender side
Back	Driver leaned towards rear seats
Driver Actions	
Enter	Driver enters the car
Leave	Driver leaves the car
Strap	Driver fastens the seatbelt
Unstrap	Driver unbuckles the seatbelt

Table 6.1: Classes for action recognition from short isolated sequences. The classes subdivide in the three categories *Movement Actions* of the driver, *Static* scenes and *Driver Actions*

of the *Action and Object Recognition* dataset was not designed to be concatenated one after another as the objective of this system was to create a lightweight system for single action recognition. Some classes of this dataset can be concatenated to generate meaningful continuous label sequences, but with too little variation for training a continuous action recognition system.

The actions for the continuous action recognition are subdivided into basic body movements towards or from non-common driving positions of the driver, static scenes with little to no visible motion, and common driver actions. An overview of the labels is shown in table 6.1.

The different movement actions are: Leaning towards the steering wheel (*To front)*, leaning towards the passenger seat (*To right*), leaning towards the rear seats (*To back*), returning from the steering wheel (*From front*), returning from passenger seat (*From right*) and returning from back seat (*From back*). Examples for these classes are shown in figure 6.1.

Figure 6.1a shows a driver leaning towards the steering wheel. The driver is initially in a normal driving position leaning against the back of the driver's seat. In the progression of the scene the driver leaves his leaning position in order to lean forward to the steering wheel. Secondly, figure 6.1b shows a snippet of a scene, where the driver leaves his initial position of leaning to the back of the driver's seat towards the passenger side. Next, figure 6.1c shows the driver leaning from the initial position, leaning at the driver seat's back towards the rear seats. The starting phase of this leaning action is similar to the starting phase of the action, where the driver leans to the passenger side, as the first part of the way to the final position is the same. Figure 6.1d shows a driver returning from the steering wheel back into a driving position, leaning at the driver seat's backrest. Following, figure 6.1e shows the driver leaning from the passenger side back to the position, where he leans at the backrest of the driver's seat. Figure 6.1f shows the last movement action, namely the driver returning from a position where he leans towards the rear seat, back in a position where he sits normally in the driver's seat. Again, the end phase of this movement action is similar to the movement action, where the driver returns from the passenger side as the way the driver covers on the way back to the driver's seat, is partly the same as the way the driver covers when returning from the passenger side.

(a) To steering wheel sequence,

(b) To passenger side sequence,

(c) To rear seat sequence,

(d) From steering wheel sequence,

(e) From passenger side sequence,

(f) From rear seats sequence.

Figure 6.1: Driver movement actions.

The static scenes consist of sequences where the driver is either in a driving position (*In Position*), leaned towards the steering wheel (*Front*), leaned towards the passenger side (*Right*) and leaned towards the rear seats (*Back*). This subset of the classes includes also the class *Empty*, where the driver's seat is not occupied by a person. These classes contain little to no motion, with the exception of the class *In Position*, which contains normal driving movements of the driver. Examples of these classes are shown in figure 6.2.

Figure 6.2a shows images of a sequence with an empty driver seat. As the driver's seat is not occupied, the scene does not change in the course of the sequence. Secondly, figure 6.2b shows a driver in a normal driving position. In the beginning of the scene the driver sits on the driver's seat leaning at the backseat of the driver's seat with his hands on his lap. In the sequence of events the driver grabs the steering wheel and turns it in different directions or holds it at different positions. Following, figure 6.2c shows a driver moving in a position where he leans towards the steering wheel. Next, figure 6.2d shows a subset of images of the driver leaning towards the passenger side. While leaning in that position the driver moves slightly around. Figure 6.2e shows the driver leaned towards the rear seats.

The common driver actions includes the actions: Entering the car (*Enter*), leaving the car (*Leave*), strapping the seat belt (*Strap*) and unstrapping the seat belt (*Unstrap*). Examples of these classes are shown in figure 6.3.

Figure 6.3a shows a driver entering the car at the driver's side. The first images show the empty driver's seat, while at the driver's door movements are already visible. Following, the driver enters the car, until he is fully seated in the driver's seat. Contrary, figure 6.3b shows a driver leaving the car. In this example the driver begins this action by leaning forward towards the driver's door side to open the driver's door. The driver steps out of the car, by leaning even further towards the driver's door, until he is out, leaving an empty driver's seat. Figure 6.3c shows the driver strapping the seat belt. This scene starts with the driver reaching behind to grab the seat belt. Following, the driver pulls on the seat belt to draw it from its pillar loop and finishing the action by buckling it up. Finally, figure 6.3d shows the driver unstrapping the seat belt. This action starts with unbuckling the seat belt by the driver, followed by leading it back to its pillar loop.

(a) Empty scene sequence,

(b) In driving position sequence,

(c) Leaned to front sequence,

(d) Leaned to passenger side sequence,

(e) Leaned to rear seats sequence.

Figure 6.2: Static scene sequences.

(a) Enter sequence,

(b) Leave sequence,

(c) Strapping seat belt sequence,

(d) Unstrap seat belt sequence.

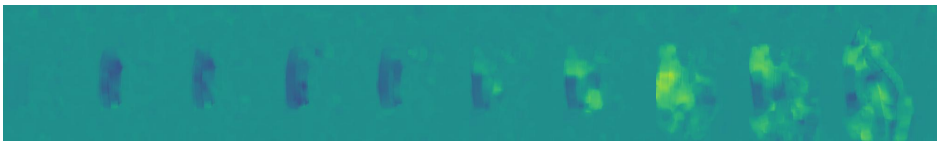
Figure 6.3: Driver actions.

The examples show snippets of the whole sequences, as the full sequences are too long to display. The length of the different sequences from the dataset varies from short sequences with 20 frames to longer sequences with up to 180 frames.

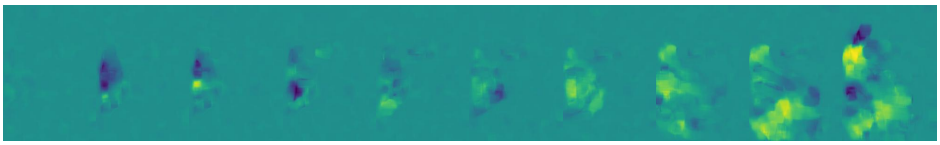
When training on optical flow images, the optical flow is calculated at runtime of the training, since the images between which the optical flow is calculated may vary. The images can be augmented spatially or the sequence can be augmented in its temporal component, leading to different optical flow images each training cycle. An example of an optical flow sequence with its origin images is shown in figure 6.4. At the top in 6.4a the origin of the flow images is shown. The x- and y-component of the optical flow is shown in 6.4b and 6.4c. In the fist two frames no movement is present. In the third frame the door opens, resulting in movement at the left part of the image, as it can be seen in the second optical flow images. Following, a person enters the car, resulting in even more movement and optical flow, as shown in the

Example of optical flow images for continuous driver action recognition

(a) amplitude sequence of a driver entering the car,



(b) x-component of the optical flow,



(c) y-component of the optical flow.

Figure 6.4: Optical flow example images.

optical flow images. The optical flow sequences is one frame shorter as the amplitude sequence, since one optical flow image is calculated between two consecutive amplitude images resulting in a sequence length of $n - 1$, if n is the number of frames of the origin sequence.

The class distribution of the dataset is shown in figure 6.5. Similar to the distribution from the action and object interaction recognition dataset described in chapter 5.1 the classes are unbalanced. This results again from the fact that some classes appear more naturally between different classes, like the class *In position*, which appears naturally between other classes.

For this dataset no body keypoints were calculated, as the body keypoint localizer network is not trained to detect body keypoints for this application. For example, the body keypoints for a driver who leans to the passenger side cannot be captures as the field of view for the keypoint localization is limited to the driver side.

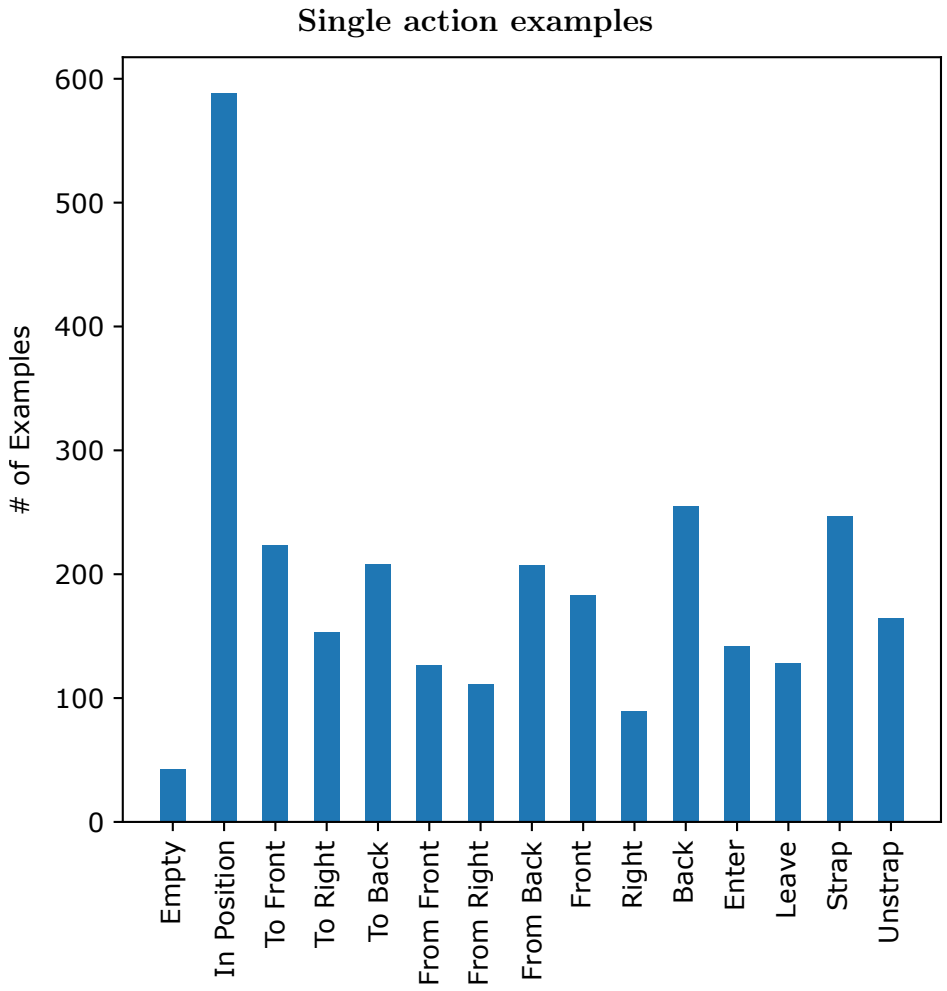


Figure 6.5: Isolated action sequences dataset distribution

6.2 Continuous Action Recognition System

The system to detect actions of a driver in a continuous video stream consists of a Time-of-Flight camera, multiple image processing nodes and an artificial neural network to classify the video stream frame wise. An overview of the system is shown in figure 6.6. The images from the Time-of-Flight camera are cropped in a way to show only the driver's seat and parts of the central console of the car. The cropped images are used to calculate optical flow images from consecutive frames of the video stream. These optical flow images are further processed by a network to recognise the actions in continuous image streams, which is a many to many convolutional and recurrent neural network combination (CNN-RNN), consisting of a convolutional neural network with three layers with 8, 12, and 18 kernels of size 3×3 with max pooling layers in between to extract spatial features from the input data, a recurrent neural network with gated recurrent units (2.1.3) of size 32 as recurrent units to extract temporal features from the previously calculated spatial features and a fully connected classification network with one hidden dimension with 32 neurons, 15 output neurons and a softmax layer to finally classify the extracted temporal features. The systems uses every fourth frame of the original sequences. Gated recurrent units as recurrent units are chosen as they contain fewer parameters and less computational operations than long-short-term memory (2.1.3). The network is visualized in figure 6.7.

The input images are optical flow image tuples calculated between consecutive frames of the input sequence. For each timestep, the CNN calculates spatial features from the input. These features are input to the RNN-

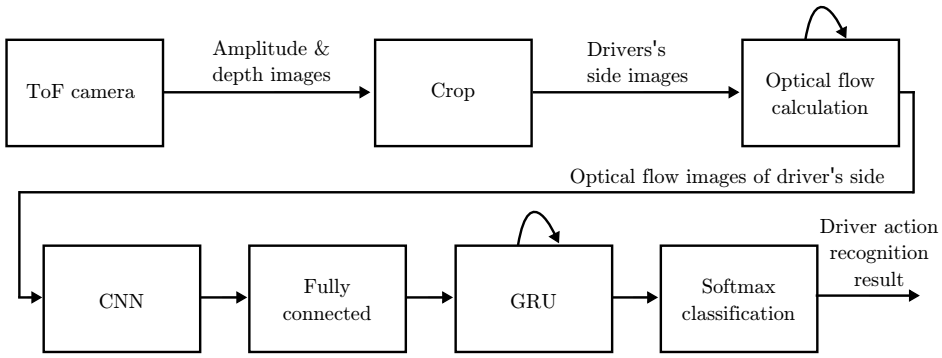


Figure 6.6: Overview of the system to recognize driver's actions in continuous video streams with optical flow image sequences. If depth or amplitude images are used as input for the CNN, the optical flow images are not calculated.

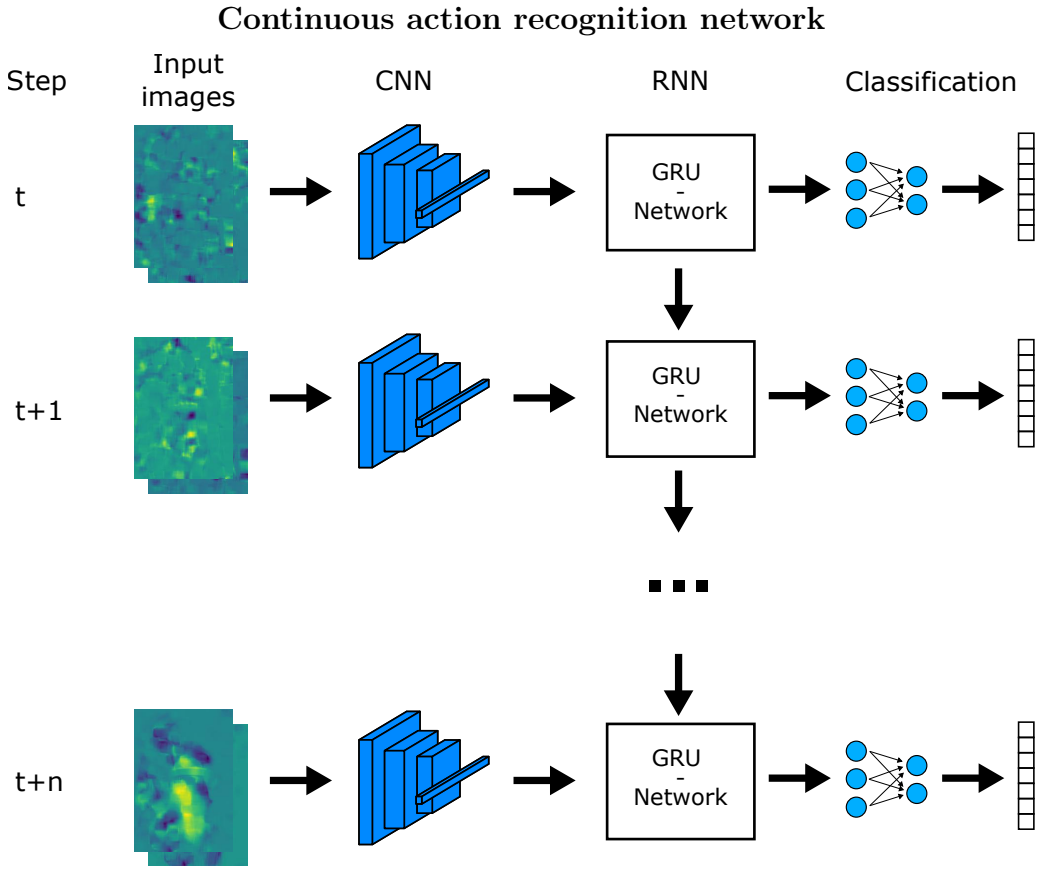


Figure 6.7: Network concept for continuous action recognition with optical flow image tuples as input. The Network consists of a CNN, a recurrent network and a fully connected network.

network, which calculates temporal features based on the spatial features of the current and previous timesteps. Afterwards the calculated temporal features are classified with a fully connected classification network to get a classification for the current timestep. The input for the next timesteps is processed in the same way, while the hidden states of the recurrent network are updated for each timestep. The input of the network can be varied by adapting the first convolutional layer of the CNN. This way, single amplitude or depth images as well as optical flow image touples can be used as input to the network.

6.2.1 Hidden State Reset Results

To define a baseline for the online action recognition network trained on closed actions, two networks for each input data format are trained with two common initialization methods. The networks are trained on amplitude and depth image sequences, as well as flow sequences calculated from consecutive amplitude images of a sequence. While testing, different reset strategies are evaluated, which define at which timesteps the hidden states of the networks are reset with the trained method. While training, the hidden states of the networks are initialized with zeros as proposed by the authors of [Cho14] and randomly generated numbers as proposed by the authors of [Zim12] for each network respectively at the beginning of each training example sequence.

	Amplitude			Depth			Flow		
	Acc	F1	mAP	Acc	F1	mAP	Acc	F1	mAP
Zeros, no reset	0.183	0.173	0.247	0.189	0.195	0.294	0.263	0.223	0.364
Zeros, min reset	0.334	0.322	0.362	0.393	0.368	0.425	0.478	0.413	0.452
Zeros, mean reset	**0.387**	**0.397**	**0.389**	**0.470**	**0.453**	**0.477**	0.499	0.455	0.478
Zeros, max reset	0.301	0.323	0.309	0.348	0.354	0.368	0.377	0.344	0.366
Random, no reset	0.296	0.299	0.330	0.269	0.257	0.344	0.371	0.328	0.463
Random, min reset	0.329	0.293	0.350	0.375	0.358	0.411	0.482	0.400	0.453
Random, mean reset	0.374	0.365	0.387	0.430	0.424	0.449	**0.509**	**0.458**	**0.500**
Random, max reset	0.327	0.332	0.326	0.364	0.374	0.381	0.436	0.404	0.442

Table 6.2: Results for networks trained with zero or random initialization evaluated with different reset strategies at test time. The hidden states of the network are not reset at test time, reset at the minimal sequence length of the training examples, the mean sequence length of the training examples and the maximum sequence length of the training examples at test time.

Besides calculating the test predictions and resetting the hidden states only at the beginning of a new test sequence, different reset strategies which are applied at test time are evaluated. These reset strategies are resetting the hidden states at the minimum sequence length, resetting the hidden states at the mean sequence length and resetting the hidden states at the maximum sequence length of the training examples. Table 6.2 shows the results calculated from the test sequences.

Resetting the hidden states only at the beginning of a new test sequence resulted in the worst classification results for the network trained with zero initialization. When applying reset strategies to this network the classification results improve. Resetting the hidden states at the mean sequence length results in the best classification results on the test data, for all input image formats. The network trained on the optical flow image sequences resulted in a balanced accuracy of 0.499, a F1-score of 0.455 and a mAP of 0.478. The network trained with random initialization of the hidden states results in better classification results on the test data when not resetting the hidden states at test time, compared to the zero initialization method without resetting. Similarly to the zero initialization networks, resetting the hidden states of the network at the mean length of the training sequences results in the best test scores for all input image formats. The results on optical flow image sequences with mean reset surpasses the results of the zero initialization network with the same configuration slightly with a balanced accuracy score of 0.509, a F1-score of 0.458 and a mAP of 0.5. Resetting the hidden states of the network at every mean training example length results in the best classification results. Moreover, using optical flow image sequences improves the capability of the network to classify continuous data when trained only on isolated sequence examples.

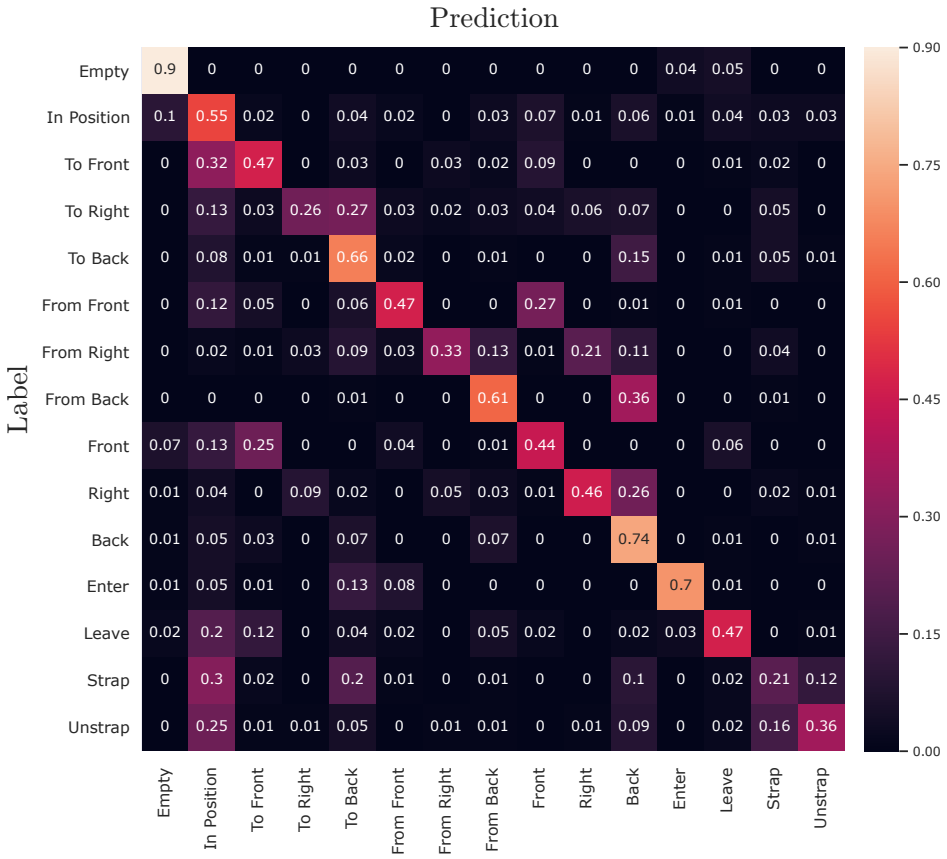


Figure 6.8: Relative confusion matrix of the action recognition network trained on flow image sequences trained with random initialization and reset at the mean sequence length at test time.

Figure 6.8 shows the confusion matrix of the test results from the action recognition network trained on optical flow image sequences with random initialization reset at the mean sequence length of the training examples. The matrix shows that the network classifications tend to be right for most of the classes, as the main classifications can be found on the main diagonal, which indicates a correct classification.

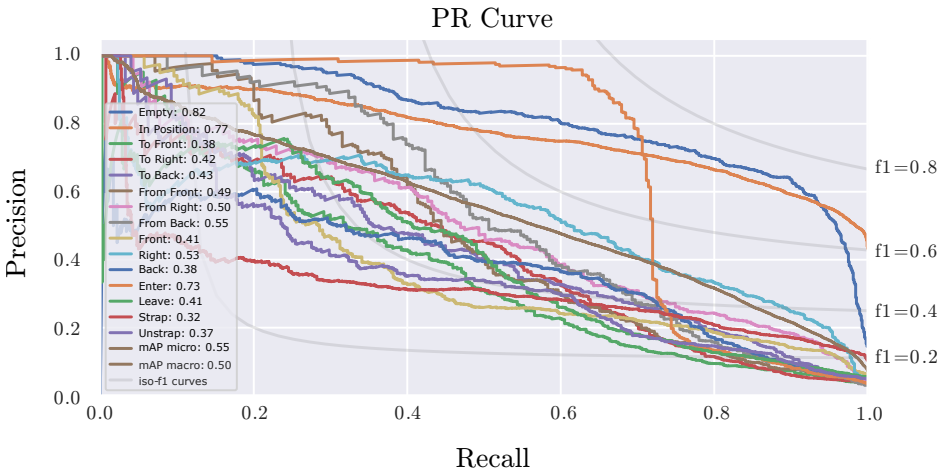


Figure 6.9: Precision recall curves of the action recognition network trained on flow image sequences and with random initialization and reset at the mean sequence length at test time.

How sure the classifier is about these classifications can be seen in figure 6.9. For the classes *Empty*, *In Position* and *Enter* the classifier reaches average precision values over 0.7 indicating that the classifier is sure about its predictions. The other classes are, despite having markedly lower average precision scores between 0.32 to 0.55, still better than random guessing, which would give an average precision score of 0.067 as 15 classes are present.

The remaining confusion matrices and precision recall curves of the test results of the networks trained with zero and random initialization and different reset strategies are displayed in appendix C. It can be seen that all networks tend to classify the frames of the continuous data as the class *In Position* if the network is not reset during test time. When resetting the hidden states at test time periodically more classes are classified correctly. However, most misclassifications are due to a class confused with the class *In Position*, which happens because of the bigger amount of different training examples of this class.

6.3 Class Transitions

In order to train a classification system to recognise actions from a continuous data stream, real continuous data is preferable to train on. However, this data for action recognition tasks is often not available and too time consuming to record and annotate. All classes and their transition behaviour need to be considered while recording. Adding a class that is not yet recorded results in re-recording the complete dataset. An alternative to use continuous data is to concatenate action sequences to artificially generate continuous data sequences. The sequences can either be concatenated randomly, or in a systematic way. In case the sequences are connected completely randomly, sequences combination might occur that would normally not occur in the real data. If the transitions between classes are known sequences of specific classes can be connected in a way that classes that are logically connected precede each other.

For this a transition list can be created, which contains all reasonable class transitions. A possible binary transition table for the introduced dataset is shown in figure 6.10. On the columns the current classes are listed, while on the rows the previous classes are listed. A 0 between a current and a previous class indicates that this class transition is unreasonable, while a 1 indicates a reasonable class transition. Those indicators also can be used as weights to alter the probability of a specific class to precede another class. For example, the class *Strap* can only be preceded by the classes *In Position* and *Mount*. This binary transition list, displaying only the possible transitions, can be extended to contain the transition possibilities. If every reasonable class is equally likely to occur before the current class the probabilities of a reasonable class to precede the current class

$$P_{W_{curr_i}}(W_{prev_j}) = \frac{\omega_{ij}}{\sum\limits_{ij} \omega_{ij}} \tag{6.1}$$

can be calculated with $P_{W_{curr_i}}(W_{prev_j})$ as the probability of class j to precede class i and $\omega_{ij} = 1$ as the class weights. The resulting transition list with probabilities is shown in figure 6.11. In contrast to the previously shown transition list, the transition indicators are replaced by probabilities, which depend on the number of possible transitions for each current class.

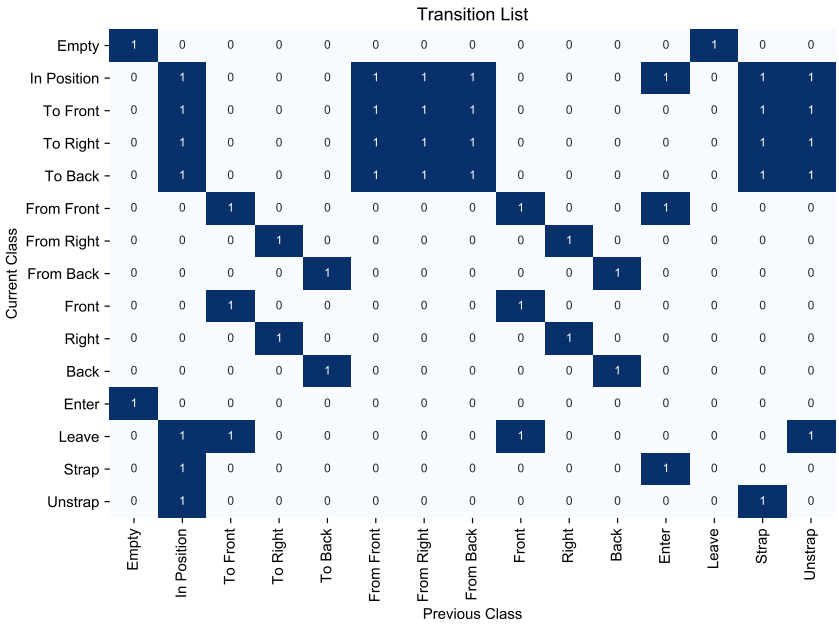


Figure 6.10: Binary transition list showing the reasonable class transitions for each class of the proposed dataset [Wey21].

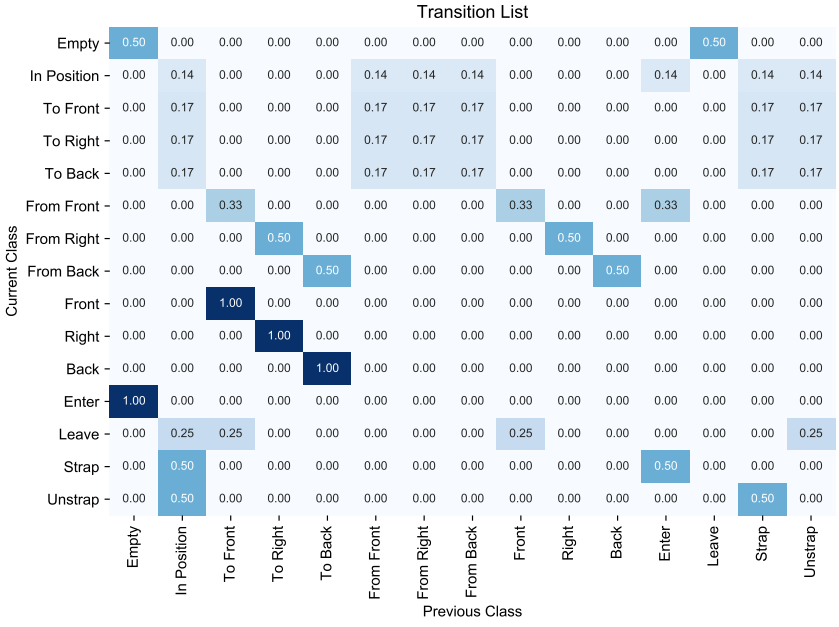


Figure 6.11: Transition list showing the probabilities for each class transition based on the number of reasonable transitions to other classes of each class [Wey21].

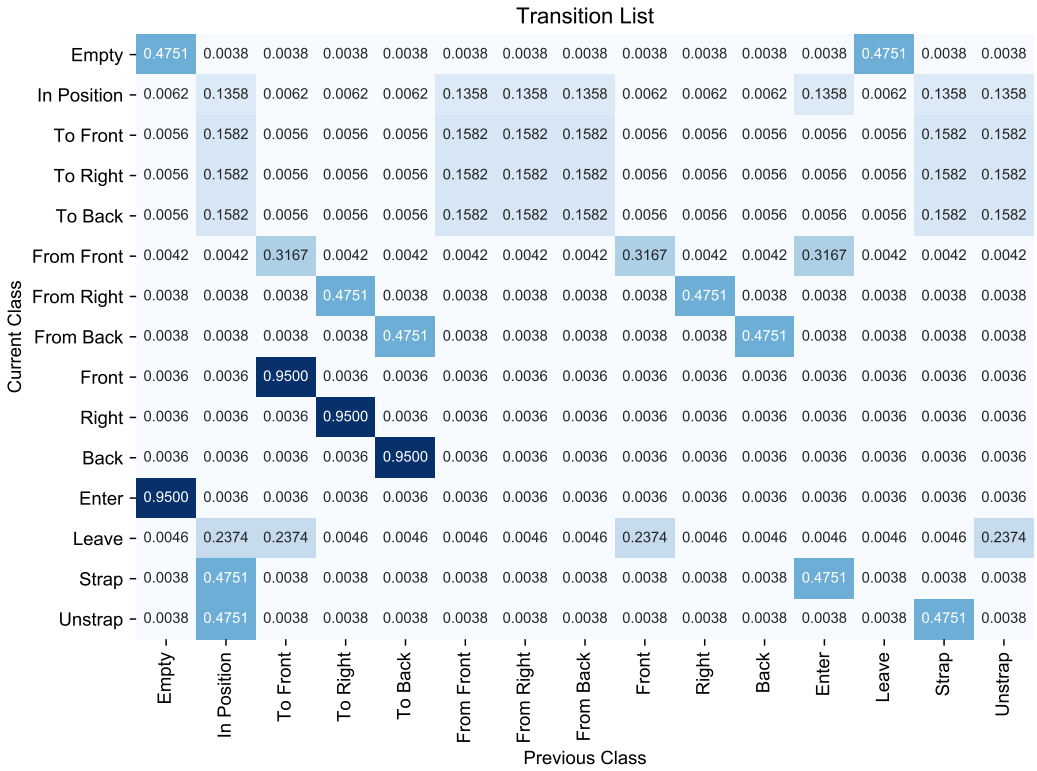


Figure 6.12: Transition list showing the probabilities of the class transitions with a minimum transition probability of choosing a non reasonable class of 0.05 [Wey21].

However, as classification systems might classify a single or multiple frames wrongly, the system still needs to be able to overcome unreasonable class transitions. To consider this possibility that an unreasonable class transition occurs in the continuous data, the transition lists can be modified with a probability that these unreasonable class transitions occur. Figure 6.12 shows this transition probability table with a minimal transition probability of 0.05 for an unreasonable transition.

6.3.1 Class Transition Results

The results of the networks trained on concatenated sequences are displayed in table 6.3. When training a network on two concatenated image sequences, the network classification error can be calculated depending on both sequences, or only on the latter one. The hidden states are initialized with random values at the beginning of each training step. Moreover, the effect of the specific class transition is evaluated. The transition probabilities are calculated as described in chapter 6.3, with a probability 0.05 of selecting an unreasonable previous class.

When training the networks on reasonable concatenated sequences, the evaluation scores improve for the networks trained on depth and optical flow image sequences, compared with the networks which are trained on the randomly concatenated sequences. Moreover, training on only the latter sequence of the concatenated sequences shows even better results than the networks trained on both concatenated sequences. This might result from the random initialization at the beginning of each training step. The network additionally tries to learn to adapt from those random values, which results in a network which is less capable of classifying continuous data. Only for the network trained on infrared image sequences the network with reasonable sequence transitions yields better results on the test data, when trained on both concatenated sequences. Overall the networks seems to learn better from depth and optical flow image sequences than from infrared image sequences, which might result from the circumstance that in infrared images the variation in imagery is higher than in depth and optical flow images. Therefore, the small networks might learn better features for this task of

	Amplitude			Depth			Flow		
	Acc	F1	mAP	Acc	F1	mAP	Acc	F1	mAP
random transition, train second half	0.415	0.410	0.444	0.416	0.428	0.472	0.669	0.607	0.704
reasonable transition, train second half	0.392	0.398	0.453	**0.477**	**0.518**	**0.561**	**0.682**	**0.649**	**0.740**
random transition, train all	0.330	0.311	0.386	0.280	0.299	0.375	0.563	0.539	0.630
reasonable transition, train all	**0.419**	**0.413**	**0.475**	0.398	0.402	0.447	0.622	0.590	0.708

Table 6.3: Test results of the networks trained on randomly and reasonably concatenated action sequences. The networks are trained on the second half of the concatenated action sequences and on all frames of the concatenated action sequences with each training method.

	Amplitude			Depth			Flow		
	Acc	F1	mAP	Acc	F1	mAP	Acc	F1	mAP
Zero initialization, mean reset	0.387	0.397	0.389	0.470	0.453	0.477	0.499	0.455	0.478
random initialization, mean reset	0.374	0.365	0.387	0.430	0.424	0.449	0.509	0.458	0.500
reasonable transition, train second half	0.392	0.398	0.453	**0.477**	**0.518**	**0.561**	**0.682**	**0.649**	**0.740**
reasonable transition, train all	**0.419**	**0.413**	**0.475**	0.398	0.402	0.447	0.622	0.590	0.708

Table 6.4: Test results comparison of the networks trained with zero and random initialization reset at the mean length of the training sequences at test time, the network trained on the second half reasonably concatenated action sequences and the network trained on all frames of reasonably concatenated action sequences.

action recognition from the more standardized imagery in depth and optical flow images, rather than from infrared images. Moreover, the networks trained on optical flow image sequences show better results than the networks trained on depth image sequences.

The results of the networks with reasonable class transitions are directly compared to the networks with a reset strategy in table 6.4. The reasonable concatenation of image sequences clearly raises the ability of the networks to classify online action sequences, while trained on trimmed action sequences only. Concatenating reasonable class examples additionally boosts the classifier's performance on continuous data. The network trained on only the latter optical flow sequence examples, which is preceded by a reasonable class with a probability of 0.95 performs best with a balanced accuracy of 0.682, an F1-score of 0.649 and an mAP of 0.74 on the continuous test examples.

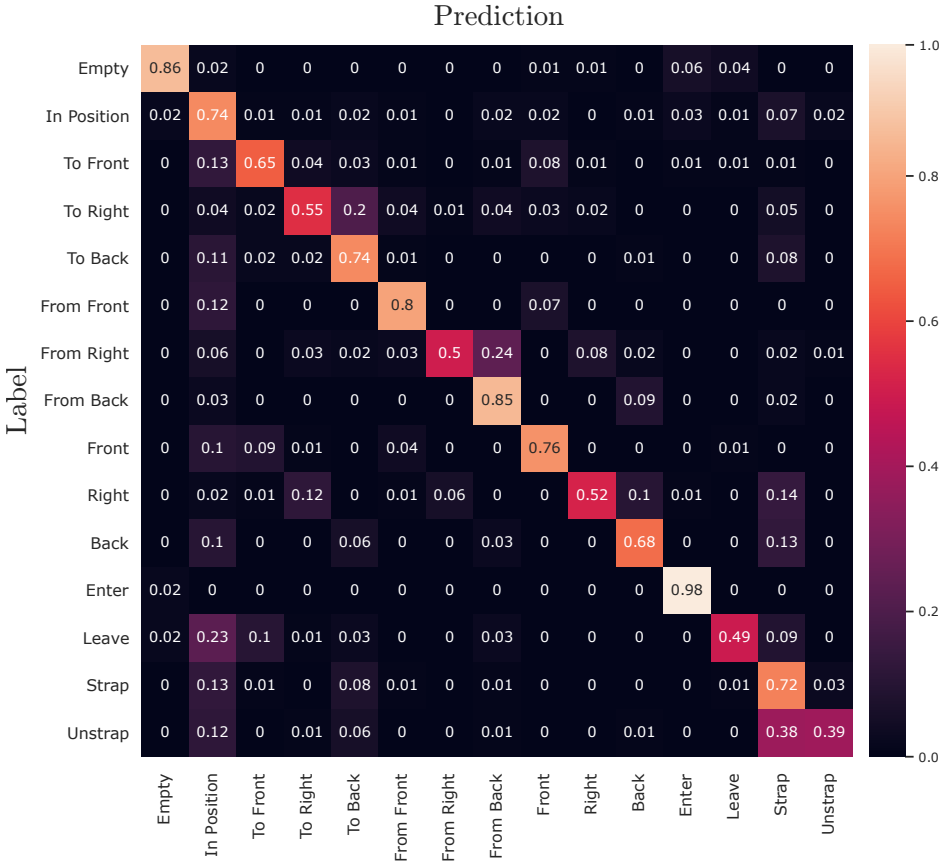


Figure 6.13: Relative confusion matrix of the action recognition network trained on the second half of reasonable concatenated optical flow image sequences.

The confusion matrix of the test results shown in figure 6.13 shows the improved performance of the classifier as well. With the reasonable concatenation approach at least half of the timesteps are classified correctly for most classes. Only the classes *Leave* and *Unstrap* are not recognised that well. The class *Leave* is most likely confused by the classifier with the class *In Position*. This might result from similarities in the beginning of the scenes of the class *Leave* with the class *In Position*. That effect is visible for some other classes as well, as those classes are also confused with this class.

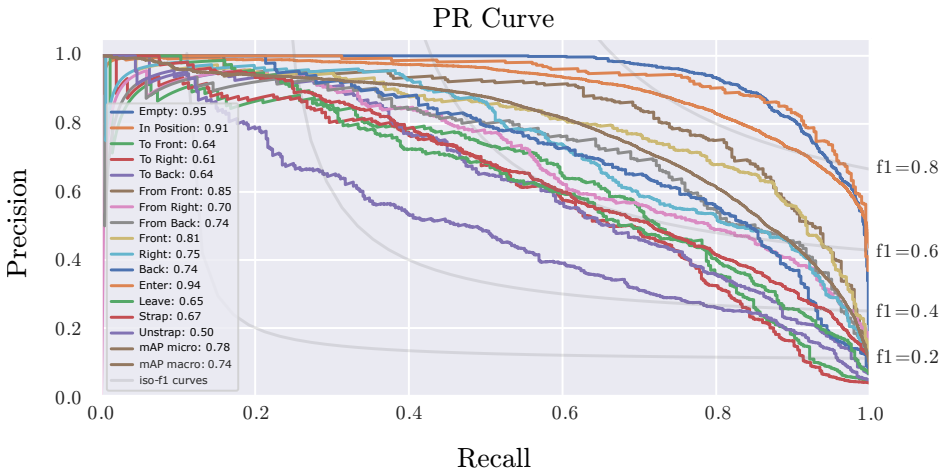


Figure 6.14: Precision recall curves of the action recognition network trained on the second half of reasonable concatenated optical flow image sequences.

The precision recall curves of this classifier's results are displayed in figure 6.14. It can be seen that the performance of the classifiers was clearly improved by training on concatenated sequences with reasonable class transitions, as the macro mAP was raised from 0.5 for the network with random initialization and mean sequence length reset to 0.74. Moreover, the lowest average precision value was raised from 0.32 to 0.5, showing, the improvement of the classifier more clearly.

The remaining confusion matrices and precision recall curves of the test results of the networks trained with the proposed transition strategies of concatenating action sequences are displayed in chapter C.

6.4 State Handling

In order to use the information from relevant previous classes to train a network, which is capable of classifying images of a continuous data stream, while trained only on isolated action sequences, a new concept for using and handling the recurrent hidden states of relevant previous examples is presented. The concept was previously published by the author in [Wey21] and is further explained in detail.

6.4.1 Recurrent States

Concatenating different sequences is the most intuitive, but for multiple reasons, not the best or an efficient way to simulate continuous data and class transitions. When training a neural network on concatenated sequences, the previous class sequence needs to be processed by the network for every training example. This can be beneficial, as the results from the previous sequences are always calculated by the most up to date neural network, in terms of training steps. However, calculating these additional sequences requires additional computational effort, as not only the current example needs to be calculated. Moreover, when training a recurrent neural network, the dimensions of the tensor need to be predefined. For a single example the sequence length is set to the maximum sequence length of the examples. Shorter sequences are padded to fit the defined sequence length. When concatenating sequences, the maximum sequence length of a training example equals the maximum sequence length of the examples multiplied by the number of concatenated sequences, resulting in more computational effort needed for each training step. An alternative to concatenating example sequences to simulate continuous data is to store the hidden states of previously processed sequences and use those stored hidden states as initialization for future training steps. This way, only current training examples and one hidden state per example need to be loaded and computed per training step.

6.4.2 Recurrent State Handling

The goal of the recurrent state handling is to define a concept to store and handle the hidden states of the recurrent units of a neural network, so that previously calculated hidden states can easily be accessed during training. The state memory is a memory block to store the hidden states of the recurrent units during training. Figure 6.15 displays the basic concept of the state handling approach for two input tensors x^0 and x^1 and three timsteps per input example. For the first input example x^0 the GRU is initialized

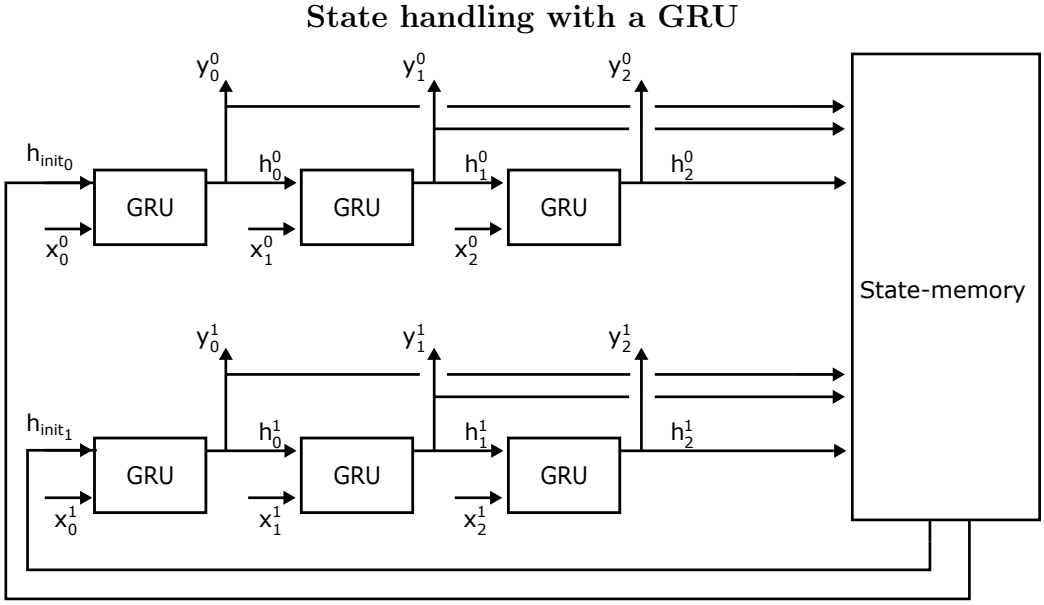


Figure 6.15: State handling with a GRU for two input examples x^0 and x^1 for three timesteps. The hidden states y of each timestep are stored in a state memory. Before starting to process new hidden states with a new input sequence, the hidden states of the GRU are initialized with a hidden state h_{init} loaded from the state memory.

with one reasonable hidden state h_{init_0} from the *State-memory*, while for the second input example x^1 a reasonable initial hidden state h_{init_1} is selected from the *State-memory*. The new calculated hidden states for each timestep h_0, h_1 and h_2 are stored in the *State-memory*.

The calculated hidden states of the training process are stored in a *State Memory* as seen in figure 6.16. This *State Memory* consists of multiple memory queues. Each memory queue handles the hidden states of one class C_i, whereas each cell of the queue stores one hidden state. The size of the queues are variable to hold only a predefined number of hidden states. Old hidden states calculated in previous training steps will therefore be deleted when the capacity of the corresponding queue is exhausted and new hidden states are calculated. This mechanism prevents that old hidden states from previous training steps are used to initialize the recurrent units as they become obsolete for future training steps. This way, hidden states calculated only from the latest training steps are stored in the *State Memory* and the hidden states selected to initialize the recurrent unit are always up to date.

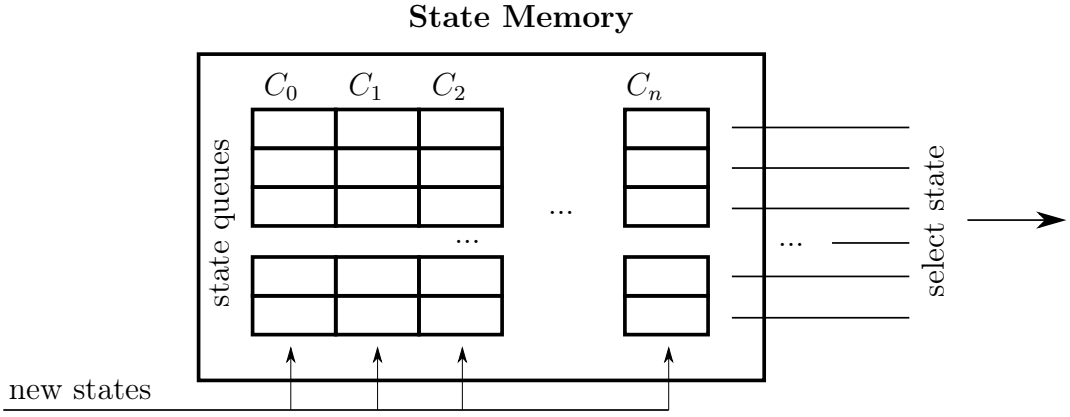


Figure 6.16: Visualization of the state memory concept. New hidden states are stored in memory queus of the respective class the hidden state originates from.

To use this state handling concept for recurrent networks with multiple recurrent layers the hidden states of each layer need to be stored jointly. To initialize the recurrent layers meaningfully all hidden states belonging to the same timestep of the same input example need to be loaded.

When using LSTMs as recurrent units the *State-memory* needs to be adapted in a way to be able to store the calculated cell states of the LSTMs jointly to the hidden states. For other recurrent network structures, which need to be initialized, the *State-memory* needs to be adapted respectively to fit the state structure of the chosen recurrent network structure.

6.4.3 State Memory Network Adaptation

To use the previously calculated hidden states of the examples, the network has additionally access to a state memory module which handles the hidden states while training the network. Figure 6.17 shows how the hidden states of the RNN-network are handled during training. The RNN-network is initialized with a hidden state of the network at the beginning of a training step. After each timestep the recently calculated hidden states of the RNN are stored in the state memory. As the training proceeds, the hidden states of the RNN-network are re-initialized with hidden states from the state memory at the beginning of each new sequence. This way, the RNN-network starts with a previously calculated hidden state configuration at the beginning of each training sequence. If the *State Memory* for a class is empty the hidden state can also be initialized with zero or random values.

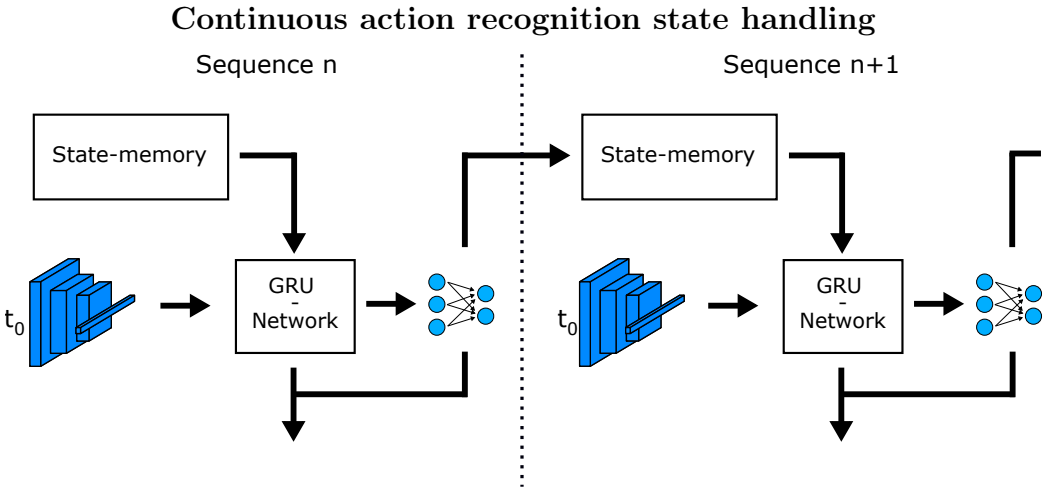


Figure 6.17: State handling concept for continuous action recognition. The calculated hidden states of each training example are stored in the state memory. For each new training example one hidden state of a previous example related to a reasonable or random class is loaded to initialize the GRU.

6.4.4 Recurrent State handling Results

The state handling approach can be applied to the training process when training on single action sequences, as well as concatenated sequences. The results of both approaches are shown in table 6.5. When training on amplitude image sequences, the state handling approach improves the test results on single actions in terms of balanced accuracy, F1-score and mAP more than trained on concatenated sequences with the state handling approach. This might result again from the circumstance that the amplitude image sequences sceneries vary and the point at which the sequences are concatenated brings too abrupt change of the scenery. Morphing or filtering the sequences at the sequence transition creates artifacts disturbing the network even more. Therefore, the previously calculated states are better for learning to classify online actions from closed action examples as the combined approach. For the depth image sequences, the state handling approach helps the network to generate better states from the first half of the concatenated sequences, which generates better test results when trained only on the second half.

The networks trained on optical flow image sequences score the best when

	Amplitude			Depth			Flow		
	Acc	F1	mAP	Acc	F1	mAP	Acc	F1	mAP
state handling	**0.467**	**0.495**	**0.549**	0.471	0.493	0.594	0.691	**0.667**	0.729
state handling, concatenated, train second half	0.411	0.410	0.458	**0.522**	**0.525**	**0.614**	**0.715**	0.657	**0.755**
state handling, concatenated, train all	0.368	0.371	0.430	0.371	0.355	0.400	0.675	0.655	0.735

Table 6.5: Test results of the networks trained on single action sequences, the second half of concatenated action sequences with active state handling and all frames of concatenated action sequences with active state handling.

	Amplitude			Depth			Flow		
	Acc	F1	mAP	Acc	F1	mAP	Acc	F1	mAP
Zero initialization, mean reset	0.387	0.397	0.389	0.470	0.453	0.477	0.499	0.455	0.478
random initialization, mean reset	0.374	0.365	0.387	0.430	0.424	0.449	0.509	0.458	0.500
reasonable transition, train second half	0.392	0.398	0.453	0.477	0.518	0.561	0.682	0.649	0.740
reasonable transition, train all	0.419	0.413	0.475	0.398	0.402	0.447	0.622	0.590	0.708
state handling	**0.467**	**0.495**	**0.549**	0.471	0.493	0.594	0.691	**0.667**	0.729
state handling, concatenated, train second half	0.411	0.410	0.458	**0.522**	**0.525**	**0.614**	**0.715**	0.657	**0.755**

Table 6.6: Comparison of the networks trained with the state handling method and the network trained on the second half of concatenated image sequences with the best previously evaluated networks and training methods.

trained additionally with the state handling approach. The network trained on single actions achieves an F1-score of 0.667, scoring higher than the other two approaches, while the network trained on the latter sequence of concatenated sequences with the state handling approach scores best in terms of balanced accuracy (0.715) and mAP (0.755). This indicates that the state handling approach alone results in a better recall for training on optical flow images, while the other networks scores better in terms of precision.

The direct comparison to the previously evaluated networks is shown in table 6.6. It can be seen that training a recurrent neural network on reasonable concatenated image sequences of isolated actions results in better performing classifiers on continuous data compared to the recurrent networks trained with commonly used initialization methods and resetting the hidden

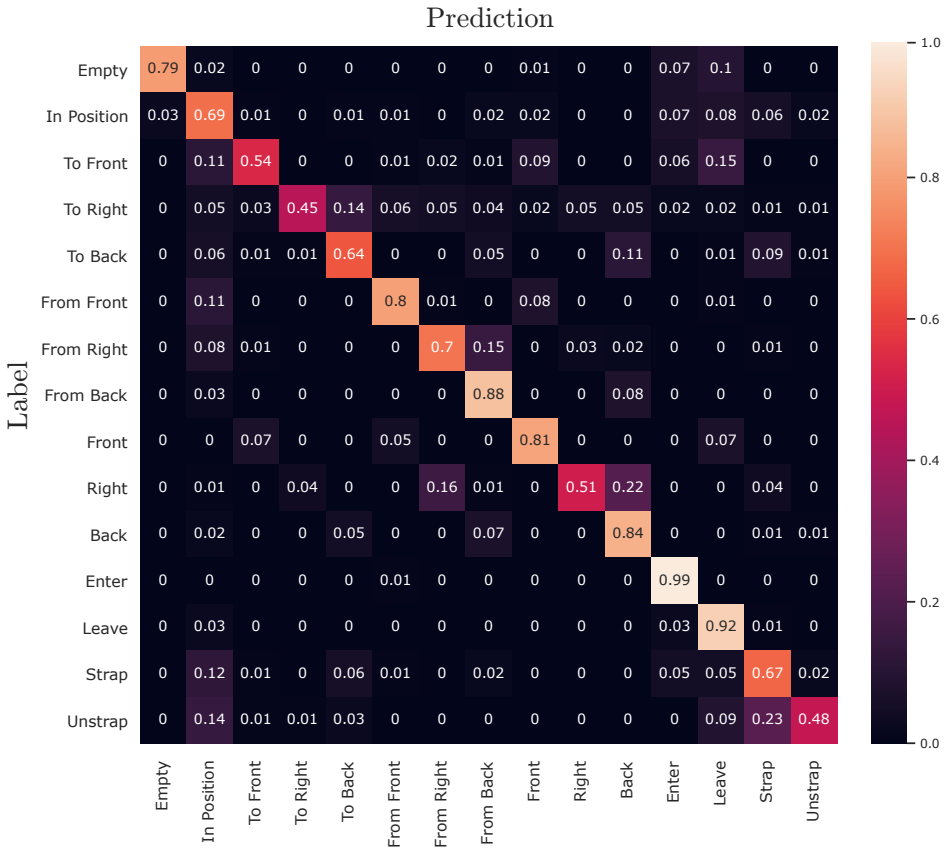


Figure 6.18: Relative confusion matrix of the action recognition network trained on the latter sequence of concatenated optical flow image sequences with reasonable state handling.

states at test time. If the networks should be trained on both concatenated image sequences or only on the latter one depends on the input data, as networks trained on input data with reduced variance like depth images or optical flow images perform better on continuous data when trained only on the latter sequences. Moreover, applying the proposed state handling approach while training the network additionally boosts the classifier's performance on continuous sequence data. The following networks are trained on optical flow image sequences only, as the previously analysed networks trained on optical flow image sequences provide significant better results on the test data as the networks trained on amplitude or depth image sequences.

When inspecting the confusion matrix of the network trained on the latter concatenated optical flow image sequences with state handling approach displayed in figure 6.18, it can be seen that classes like *From Right*, *Back*,

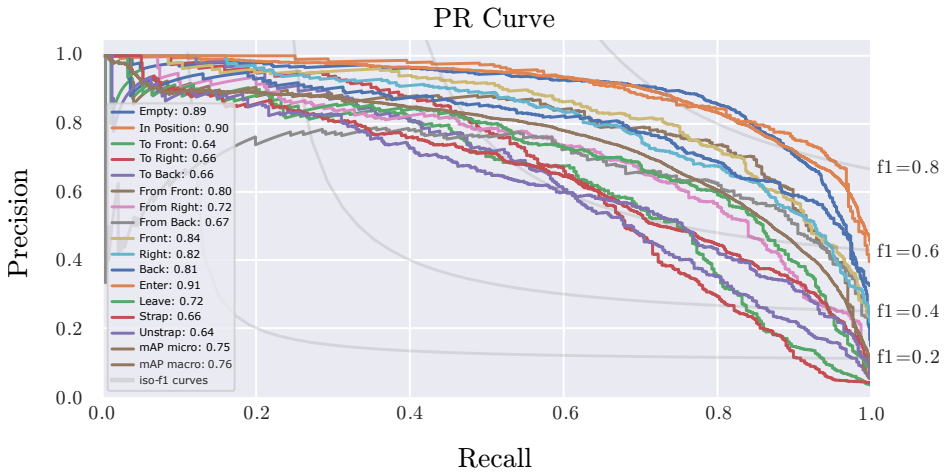


Figure 6.19: Precision recall curves of the action recognition network trained on the latter sequence of concatenated optical flow image sequences with reasonable state handling.

Leave and *Unstrap* are classified much more often correctly than with the networks trained with random initialization and hidden state reset at the mean sequence length (figure 6.8) or the network trained on the second half of reasonable concatenated optical flow image sequences without state handling (figure 6.13). Other classes like *Empty*, *In Position* and *To Front* are slightly less classified correctly and more likely confused with classes that show similarities to those classes compared to the network trained on the second half of reasonable concatenated optical flow image sequences without state handling. For instance, this effect can be seen between the classes *To Front* and *Leave*. A driver leaning forward can be recognized as a driver leaving the car, as leaning forward can be part of the leaving action. A different example is the confusion of the class *Empty* with the classes *Enter* or *Leave*, as those classes might show an empty seat for at least a short time.

Figure 6.19 shows the precision recall curve of this classifier. It can be seen that for every class the network performs far better than the networks evaluated with a reset strategy. Moreover, the minimal average precision score was raised from 0.5 of the network trained only on the latter sequence of the reasonable concatenated sequences without explicit state handling to 0.64.

Comparing the average precision scores per class of the networks trained on optical flow image sequences, shown in table 6.7, it can be seen that for

most classes the networks trained with the state handling approach results in the best average precision scores. The remaining classes score best in terms of average precision per class for the networks trained with reasonable concatenated sequences without the state handling approach. This shows that the performance of a continuous action recognition network trained on short isolated sequences can be enhanced with two of the proposed training techniques. First, by training on reasonably concatenated training sequences, and secondly by combining it with the proposed state handling approach. Combining both approaches to train the network improves the system even more.

The remaining confusion matrices and precision recall curves of the test results of the networks trained with the proposed state handling approach are displayed in chapter C.

		Average Precision												
	mAP	Empty	InPosition	ToFront	ToRight	ToBack	FromFront	FromRight	FromBack	Front	Right	Back	Enter	Leave	Strap	Unstrap
Zero initialization, mean reset	0.478	0.813	0.778	0.323	0.403	0.421	0.437	0.435	0.438	0.447	0.476	0.371	0.719	0.366	0.407	0.335
random initialization, mean reset	0.500	0.820	0.772	0.378	0.420	0.433	0.490	0.498	0.552	0.411	0.527	0.376	0.728	0.406	0.319	0.375
reasonable transition, train second half	0.740	**0.949**	0.907	0.637	0.611	0.643	**0.846**	0.703	0.739	0.807	0.751	0.743	0.941	0.651	**0.669**	0.505
reasonable transition, train all	0.708	0.904	0.887	0.610	0.563	0.545	0.841	**0.745**	0.679	0.782	0.647	0.469	**0.954**	**0.740**	0.624	0.629
state handling	0.729	0.914	**0.916**	**0.701**	0.618	0.607	0.793	0.707	**0.773**	0.826	0.764	0.619	0.870	0.655	0.632	0.543
state handling, concatenated, train second half	**0.755**	0.894	0.905	0.636	**0.661**	**0.656**	0.796	0.722	0.667	**0.838**	**0.816**	**0.807**	0.914	0.721	0.658	**0.638**

Table 6.7: Average precision score comparison of the best networks and training methods on the test data with optical flow image sequences as input.

LSTM Results In order to evaluate the performance of the proposed online action recognition network with LSTMs as recurrent units compared to GRUs as recurrent units, table 6.8 provides the results of the different networks on the test set. The best previously evaluated networks are now trained with LSTM networks instead of GRUs as recurrent units to compare the performance of both recurrent network structures. Similarly to the GRU version of the evaluated networks, the networks with LSTMs as recurrent units perform worst when trained on amplitude image sequences. While the performance of the network trained only on the latter sequence of the concatenated sequences and the state handling approach results in higher evaluation scores than the network with GRUs as recurrent units with the same training configuration, the other two training methods preform worse than the GRU networks. This might result from the fact that LSTMs contain more trainable parameters than GRUs and therefore need more training examples to be trained properly.

	Amplitude			Depth			Flow		
	Acc	F1	mAP	Acc	F1	mAP	Acc	F1	mAP
GRU, state handling	**0.467**	**0.495**	**0.549**	0.471	0.493	0.594	0.691	**0.667**	0.729
GRU, state handling, concatenated, train second half	0.411	0.410	0.458	**0.522**	**0.525**	**0.614**	0.715	0.657	**0.755**
GRU, state handling, concatenated, train all	0.368	0.371	0.430	0.371	0.355	0.400	0.675	0.655	0.735
LSTM, state handling	0.346	0.324	0.319	0.513	0.508	0.577	0.671	0.614	0.686
LSTM, state handling, concatenated, train second half	0.438	0.445	0.468	0.498	0.482	0.550	**0.730**	0.662	0.749
LSTM, state handling, concatenated, train all	0.321	0.284	0.336	0.447	0.474	0.533	0.651	0.631	0.727

Table 6.8: Comparison of LSTMs and GRUs for the recurrent units of the online action recognition network

For the networks trained on depth image sequences, the networks trained with state handling on single image sequences and trained on both concatenated image sequences with state handling perform better than their complement GRU networks. However, the GRU network trained only on the latter sequences of the concatenated sequences still scores better in terms of balanced accuracy, F1-score and mAP than the LSTM networks.

The networks trained on optical flow image sequences result in the best training scores for the LSTM networks compared to the networks trained on amplitude and depth sequences. The network trained only on the latter sequence of the concatenated sequences with state handling performs best with a balanced accuracy score of 0.73, surpassing the balanced accuracy scores of the GRU networks. The F1-score and the mAP values are slightly below the results of the GRU networks. Therefore, a network with GRUs as recurrent units performs similarly to the networks with LSTMs as recurrent units, while containing fewer parameters and less computations.

The remaining confusion matrices and precision recall curves of the test results of the networks with LSTM modules are shown in chapter C.

To bring the results into more context table 6.9 shows current state of the art action segmentation results of the papers of [Hua20] and [Che20] on the continuous action recognition datasets *50Salats* [Ste13] and *Breakfast* [Kue14]. These offline approaches segment video streams with multiple actions into action segments. The F1-scores are calculated at different overlapping limits at which the segmented action counts as correctly segmented, while the accuracy is calculated frame wise. However, since most action recognition systems that analyze continuous data segment actions in an offline approach with access to previous and future frames of the sequence, the results are not very fair to compare with the proposed approach that aims to classify frames of a sequence in real-time with access only to the previous frame. Nevertheless, it can be seen that the metric scores of the proposed approaches are in a similar range as two of the current state of the art approaches on the public available dataset.

	F1 @10	F1 @25	F1 @50	Acc
50Salats [Ste13]				
[Hua20]	0.754	0.728	0.639	0.826
[Che20]	0.820	80.1	0.725	0.832
Breakfast [Kue14]				
[Hua20]	0.575	0.540	0.433	0.650
[Che20]	0.742	0.686	0.565	0.710

Table 6.9: Two current state of the art action recognition results on the datasets *50Salats* [Ste13] and *Breakfast* [Ste13]

6.5 Continuous Online Action Recognition from Short Isolated Sequences Summary

In this chapter, new training methods for recurrent neural networks were presented to train online action recognition systems from isolated sequences with the basis of classifying trimmed driver's actions described in chapter 5. As continuous data for training online action recognition networks with many classes are very time consuming to gather the presented approaches improve the classification scores of networks trained on trimmed action sequences.

First, networks were trained on single action sequences with zero or random initialized hidden states. For testing the trained network on continuous data the hidden states were then reset in different intervals.

Secondly, a method to concatenate reasonable sequences was presented. Concatenating reasonable classes improve the classifiers performances compared to approaches trained only on single trimmed videos with different strategies to reset the hidden states at runtime.

Third, a new method to initialize the hidden states was proposed to train the recurrent networks on single isolated action sequences, without concatenating them with other sequences, reducing the computational effort of a training step, compared to the concatenation approach.

It was shown, that the results of the network trained with the proposed state handling approach and amplitude images exceeds the classification results of all other evaluated approaches trained on amplitude image sequences.

When training the networks on depth or optical flow image sequences the networks trained with the state handling method scores similar compared to the networks trained on concatenated sequences. Combining both approaches boosted the classification results on the test data for the network trained on depth images even more compared to the test results of the network trained with only one of these approaches. The network trained on concatenated optical flow image sequences with the proposed state handling approach scores slightly better in terms of balanced accuracy and mean average precision than the approaches trained with only one of these approaches.

As seen in chapter 5 the choice of the data input format essentially influences the capability of the networks to classify the proposed actions. As the action dataset is composed of actions with much body movement of the driver the results of the networks trained on optical flow images exceeds the results of the networks trained on amplitude or depth image sequences.

Overall, two practicable methods were presented to train online action recognition systems from short isolated action sequences. One big advantage of those methods to train an online action recognition system is that these short isolated action sequences are much easier to gather and annotate compared to continuous data. Moreover, using the calculated hidden states of a training sequence to initialize future training steps while training a recurrent neural network exceeds the test results of the networks trained with random or zero initialization and reset at runtime. Additionally, these networks score similar to the networks trained on concatenated image sequences, while being computationally more efficient, as only one image sequence needs to be loaded and processed instead of multiple image sequences.

CHAPTER 7

Conclusion and Outlook

In this thesis different training methods to improve Time-of-Flight image based driver monitoring systems were proposed, discussed and evaluated. For that, existing methods were refined and new methods were proposed in order to improve artificial neural network based classification systems and their training for interior sensing and driver monitoring applications. Such interior sensing and driver monitoring applications and systems are highly relevant issues in the automotive industry by now. Developing computationally efficient systems for these tasks is insofar important as several applications need to run at once on computational limited recources. Moreover, artificial neural network and deep learning based systems usually require a big amount of data in order to be trained properly and generalize to unseen data. Gathering these datasets is time consuming and costly, as the data not only needs to be recorded but annotated as well, especially for image sequence data.

In this thesis three driver monitoring systems were developed. For each system new training methods for artificial neural networks were proposed to improve the classification results of small networks trained on limited image data. This methods include a multi-label extension for hierarchical classification to integrate a fallback option for single image occupancy classification of the driver's seat as well as classifying the state of the driver. Moreover, different methods for reducing and normalizing the input feature space for action recognition were proposed, in order to train action recognition systems with limited data and computational resources. Furthermore, an augmentation technique to systematically augment the time component of sequence data is introduced. Finally, problems of training a system for continuous action recognition systems are discussed and a computational efficient solution for training continuous action recognition systems with isolated action sequence training data is proposed.

For the first application described in chapter 4 an interior sensing system based on single Time-of-Flight images was developed classifying the occupancy of a driver's seat as well as the current state of the driver if a driver is present. A hierarchical structure integrated in the classification network, providing a built-in fallback option if the classifier is unsure about a class decision of a sub domain. The proposed multi-label extended hierarchical structure improves the classification by masking out irrelevant parts of the label structure due to the hierarchical structure of the labels. This masking enables the network to learn only from the relevant classifications during training and ignore irrelevant parts. This structure not only scores with a difference of 0.325 in Accuracy, 0.123 in F1-score and 0.332 in mean average precision significantly better than an approach trained with a flat label structure, but provides more structured labels while providing a fallback option through the hierarchy. Compared to the proposed multi-label extension of the Yolo approach described by the authors of [Red17], the networks trained with the proposed approach show slightly better classification performances for the labels located higher in the hierarchy (+0.026 mean balanced accuracy, +0.011 mean average precision for the first two hierarchical layers) while the classification performance is comparable for the labels located in the lower part of the hierarchy (−0.005 mean balanced accuracy, −0.01 mean average precision for the last hierarchical layer). Moreover, it was shown that for the task of occupancy and driver state monitoring, artificial neural network based systems benefit from combined amplitude and depth images compared to the networks trained with only one of these input image types. Overall a hierarchical Time-of-Flight based system was introduced to efficiently detect the occupancy of driver seats and driver states while providing fallback options for uncertain classification decisions.

The second application, discussed in chapter 5, deals with action and object interaction recognition of the driver and enhances the capability of the *Driver's State Monitoring System* described in chapter 4 to recognise the driver's state by analysing image sequences instead of single images of the driver. A system was introduced to analyse the body movements of a driver along his hand interactions from sequence data in real-time. For this, a feature reduction method was proposed which extracts the positions of relevant body parts of the driver as well as orientation and size normalized cut outs of the hand regions. These reduced features were further used to train smaller networks as the feature space to be searched for valuable information for the actions and object interactions was reduced with the mentioned feature reduction beforehand. These networks perform markedly better than

the networks trained on sequences of images showing the complete driver seat region instead of the reduced features only. It was shown that optical flow images provide valuable information for recognising classes with strong movement, but should not be preferred for distinguishing between object interactions as only movements are present in this image format and information about the objects are most likely lost. However, optical flow images need to be calculated from two sensor images, adding an additional computation step to the overall computational cost. Moreover, a new augmentation method to systematically augment the time component of sequence data was introduced and has shown a valuable positive effect on the test classification performance of the networks when applied during training.

Finally, the third system discussed in chapter 6 aims to classify several body movements of the driver and offers the possibility to extend systems, like *Action and Object Interaction* described in chapter 5, that analyze isolated action sequences, so that they can also process continuous data. For this system a training method was suggested which is able to handle the hidden states of a recurrent neural network for more efficient training of a system classifying continuous sequence data while trained only on closed trimmed action sequences, compared to concatenating these sequences during training. Methods for training a system to classify continuous data sequences from short isolated action sequences is valuable insofar as it simplifies the process of gathering training data. Single action sequences can be recorded without having to make sure that certain class transitions are sufficiently present in the dataset. Moreover, adding new classes to the system as the transitions from and to the new classes can easily be added to the proposed transition table instead of recording new data examples with all possible class transitions present. It was shown that the trained network was able to learn these class transitions with the proposed methods with comparable results to the concatenation approach while being more efficient during training. Combining the state handling approach with the approach of systematically concatenating the training sequences improved the classification results even more. As already shown in chapter 5, training a recurrent neural network to recognise action classes with much movement benefits from training with optical flow images as input image format, compared to amplitude or depth image sequences.

Overall, three Time-of-Flight based interior sensing and driver monitoring systems as well as new training methods for artificial neural networks were presented. The proposed systems can provide valuable information about the occupancy of the driver's seat as well as the state of the driver. This

information can further be used to gain a deeper understanding of possible distractions and the readiness of the driver to take over control of the vehicle in near future semi-automated cars. Moreover, warning strategies can be adapted more precisely to the current state of the driver to draw his attention back to the road if necessary, increasing the overall vehicle's safety.
However, not only safety systems can benefit from the additional knowledge about the driver's activities in the car. Infotainment and comfort systems can be adapted to the state of the driver as well, increasing the comfort of the vehicle.
While the focus of this thesis was to develop systems to monitor the driver seat region and the driver, all presented systems can be adapted to monitor the passenger seats as well as the passengers of a vehicle. Moreover, the systems can be used in non-vehicle settings as well as with image data other than Time-of-Flight image data if appropriately adapted to the new environment.

Future systems can combine the presented approaches to be more failsafe while trained on less data than deep learning systems usually need.

To integrate the hierarchical structure proposed in chapter 4 in an action recognition system as describes in the chapters 5 and 6 the classes need to be changed in a way, that they feature a hierarchical structure, independently from the input feature format. Once the classes feature a hierarchical structure the proposed method can be implemented as described. The system can benefit similarly as the hierarchy integrates a natural fallback option to higher layers of the class hierarchy.
The approaches to train action recognition systems described in the chapters 5 and 6 can be combined as well to train a continuous action recognition system with trimmed action sequences of reduced features. For this, the system needs to meet three criteria. First, the classes must be able to be lined up meaningfully one after the other in order to artificially generate continuous sequence data. Secondly, for all relevant body poses of the driver body keypoints must be predictable. And finally, the part of the network that is responsible for bringing features of different timesteps into context need to be a recurrent neural network with storable hidden states for each timestep. If these three criteria are met, a continuous action recognition system can be trained on isolated action sequences of body keypoints and additional relevant features.

Own Publications and Patents

[Bar19] Alexander Barth and Patrick Weyers. *System and method for generating a confidence value for at least one state in the interior of a vehicle*. US Patent App. 16/196,193. June 2019.

[Wey18] Patrick Weyers, Alexander Barth, and Anton Kummert. "Driver state monitoring with hierarchical classification". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 3239–3244.

[Wey19] Patrick Weyers, David Schiebener, and Anton Kummert. "Action and Object Interaction Recognition for Driver Activity Classification". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 4336–4341.

[Wey21a] Patrick Weyers, Alexander Barth, and David Schiebener. *Method and System for Determining an Activity of an Occupant of a Vehicle*. US Patent App. 17/006,652. Mar. 2021.

[Wey21b] Patrick Weyers and Anton Kummert. "Continuous Driver Activity Recognition from Short Isolated Action Sequences". In: *Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM,* INSTICC. SciTePress, 2021, pp. 158–165.

Bibliography

[Abd16] Irman Abdic, Lex Fridman, Daniel McDuff, Erik Marchi, Bryan Reimer, and Björn Schuller. "Driver frustration detection from audio and video in the wild". In: *Proceedings of the KI* (2016), p. 237.

[Abu16] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. "Youtube-8m: A large-scale video classification benchmark". In: *arXiv preprint arXiv:1609.08675* (2016).

[Alb18] Franz Albers, Cecilia de La Parra, Jan Braun, Frank Hoffmann, and Torsten Bertram. "Schätzung der Körperpose von Autofahrern aus Tiefenbildern". In: *Proceedings 28. Workshop Computational Intelligence*. 2018, p. 197.

[Beh18] Ardhendu Behera, Alexander Keidel, and Bappaditya Debnath. "Context-driven multi-stream LSTM (M-LSTM) for recognizing fine-grained activity of drivers". In: *German Conference on Pattern Recognition*. 2018, pp. 298–314.

[Bin09] Alexander Binder, Motoaki Kawanabe, and Ulf Brefeld. "Efficient classification of images with taxonomies". In: *Asian Conference on Computer Vision*. 2009, pp. 351–362.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[Bor17] Guido Borghi, Riccardo Gasparini, Roberto Vezzani, and Rita Cucchiara. "Embedded recurrent network for head pose estimation in car". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 1503–1508.

[Bor20] Guido Borghi, Stefano Pini, Roberto Vezzani, and Rita Cucchiara. "Mercury: a vision-based framework for Driver Monitoring". In: *International Conference on Intelligent Human Systems Integration*. 2020, pp. 104–110.

[Cab15] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. "Activitynet: A large-scale video benchmark for human activity understanding". In: *Proceedings of the ieee conference on computer vision and pattern recognition*. 2015, pp. 961–970.

[Cai04] Lijuan Cai and Thomas Hofmann. "Hierarchical document categorization with support vector machines". In: *Proceedings of the thirteenth ACM international conference on Information and knowledge management*. 2004, pp. 78–87.

[Car17] Joao Carreira and Andrew Zisserman. "Quo vadis, action recognition? a new model and the kinetics dataset". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6299–6308.

[Cer14] Ricardo Cerri, Rodrigo C. Barros, and André CPLF de Carvalho. "Hierarchical multi-label classification using local neural networks". In: *Journal of Computer and System Sciences* 80.1 (2014), pp. 39–56.

[Cha15] Ju Yong Chang and Kyoung Mu Lee. "Large margin learning of hierarchical semantic similarity for image classification". In: *Computer Vision and Image Understanding* 132 (2015), pp. 3–11.

[Che07] Shinko Y. Cheng, Sangho Park, and Mohan M. Trivedi. "Multi-spectral and multi-perspective video arrays for driver body tracking and activity analysis". In: *Computer Vision and Image Understanding* 106.2-3 (2007), pp. 245–257.

[Ché15] Guilhem Chéron, Ivan Laptev, and Cordelia Schmid. "P-cnn: Pose-based cnn features for action recognition". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 3218–3226.

[Che20] Min-Hung Chen, Baopu Li, Yingze Bao, and Ghassan AlRegib. "Action segmentation with mixed temporal domain adaptation". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020, pp. 605–614.

[Cho14] 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". In: *arXiv preprint arXiv:1406.1078* (2014).

[Cra15] Céline Craye and Fakhri Karray. "Driver distraction detection and recognition using RGB-D sensor". In: *arXiv preprint arXiv:1502.00250* (2015).

[Das18] Srijan Das, Michal Koperski, Francois Bremond, and Gianpiero Francesca. "Deep-temporal lstm for daily living action recognition". In: *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2018, pp. 1–6.

[Dem09] D. Demirdjian and C. Varri. "Driver pose estimation with 3D Time-of-Flight sensor". In: *2009 IEEE Workshop on Computational Intelligence in Vehicles and Vehicular Systems*. 2009, pp. 16–22.

[Den10] Jia Deng, Alexander C. Berg, Kai Li, and Li Fei-Fei. "What does classifying more than 10,000 image categories tell us?" In: *European conference on computer vision*. 2010, pp. 71–84.

[Den11] Jia Deng, Alexander C. Berg, and Li Fei-Fei. "Hierarchical semantic indexing for large scale image retrieval". In: *2011 IEEE Conference on Computer Vision and Pattern Recognition*. 2011, pp. 785–792.

[Don15] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. "Long-term recurrent convolutional networks for visual recognition and description". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 2625–2634.

[Du08] Yong Du, Peijun Ma, Xiaohong Su, and Yingjun Zhang. "Driver fatigue detection based on eye state analysis". In: *11th Joint International Conference on Information Sciences*. 2008.

[Era19] Hesham M. Eraqi, Yehya Abouelnaga, Mohamed H. Saad, and Mohamed N. Moustafa. "Driver distraction identification with an ensemble of convolutional neural networks". In: *Journal of Advanced Transportation* 2019 (2019).

[Eri17] Alexander Eriksson and Neville A. Stanton. "Takeover time in highly automated vehicles: noncritical transitions to and from manual control". In: *Human factors* 59.4 (2017), pp. 689–705.

[Far03] Gunnar Farnebäck. "Two-frame motion estimation based on polynomial expansion". In: *Scandinavian conference on Image analysis*. 2003, pp. 363–370.

[Fri18] Lex Fridman, Bryan Reimer, Bruce Mehler, and William T. Freeman. "Cognitive load estimation in the wild". In: *Proceedings of the 2018 chi conference on human factors in computing systems*. 2018, pp. 1–9.

[Gan06] Ling Gan, Bing Cui, and Weixing Wang. "Driver fatigue detection based on eye tracking". In: *2006 6th World Congress on Intelligent Control and Automation*. Vol. 2. 2006, pp. 5341–5344.

[Geb19] Patrick Gebert, Alina Roitberg, Monica Haurilet, and Rainer Stiefelhagen. "End-to-end prediction of driver intention using 3d convolutional neural networks". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 969–974.

[Goy17] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. "The" something something" video database for learning and evaluating visual common sense". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5842–5850.

[He16] 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*. 2016, pp. 770–778.

[Hoa16] T. Hoang Ngan Le, Yutong Zheng, Chenchen Zhu, Khoa Luu, and Marios Savvides. "Multiple scale faster-rcnn approach to driver's cell-phone usage and hands on steering wheel detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2016, pp. 46–53.

[Hoc97] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[Hor04] Wen-Bing Horng, Chih-Yuan Chen, Yi Chang, and Chun-Hai Fan. "Driver fatigue detection based on eye tracking and dynamic template matching". In: *IEEE International Conference on Networking, Sensing and Control, 2004*. Vol. 1. 2004, pp. 7–12.

[Hua20] Yifei Huang, Yusuke Sugano, and Yoichi Sato. "Improving action segmentation via graph-based temporal reasoning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 14024–14034.

[Ian16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[Iof15] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[Ji12] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. "3D convolutional neural networks for human action recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.1 (2012), pp. 221–231. ISSN: 0162-8828.

[Jia14] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. *THUMOS Challenge: Action Recognition with a Large Number of Classes*. 2014.

[Kri12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[Kue11] Hildegard Kuehne, Hueihan Jhuang, Est\'\ibaliz Garrote, Tomaso Poggio, and Thomas Serre. "HMDB: a large video database for human motion recognition". In: *2011 International Conference on Computer Vision*. 2011, pp. 2556–2563.

[Kue14] Hilde Kuehne, Ali Arslan, and Thomas Serre. "The language of actions: Recovering the syntax and semantics of goal-directed human activities". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 780–787.

[LeC89] Yann LeCun et al. "Generalization and network design strategies". In: *Connectionism in perspective* 19 (1989), pp. 143–155.

[Li14] Li Li, Klaudius Werber, Carlos F. Calvillo, Khac Dong Dinh, Ander Guarde, and Andreas König. "Multi-sensor soft-computing system for driver drowsiness detection". In: *Soft computing in industrial applications*. Springer, 2014, pp. 129–140.

[Li16] Yining Li, Chen Huang, Chen Change Loy, and Xiaoou Tang. "Human attribute recognition by deep hierarchical contexts". In: *European Conference on Computer Vision*. 2016, pp. 684–700.

[Lin14] 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*. 2014, pp. 740–755.

[Liu15] Tianchi Liu, Yan Yang, Guang-Bin Huang, Yong Kiang Yeo, and Zhiping Lin. "Driver distraction detection using semi-supervised machine learning". In: *IEEE transactions on intelligent transportation systems* 17.4 (2015), pp. 1108–1120.

[Liu16] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. "Spatio-Temporal LSTM with Trust Gates for 3D Human Action Recognition". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. Cham: Springer International Publishing, 2016, pp. 816–833. ISBN: 978-3-319-46487-9.

[Mar18] Manuel Martin, Johannes Popp, Mathias Anneken, Michael Voit, and Rainer Stiefelhagen. "Body pose and context information for driver secondary task detection". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018, pp. 2015–2021.

[Mar19] Manuel Martin, Alina Roitberg, Monica Haurilet, Matthias Horne, Simon Reiss, Michael Voit, and Rainer Stiefelhagen. "Drive&Act: A Multi-Modal Dataset for Fine-Grained Driver Behavior Recognition in Autonomous Vehicles". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[Mas18] Sarfaraz Masood, Abhinav Rai, Aakash Aggarwal, Mohammad Najmud Doja, and Musheer Ahmad. "Detecting distraction of drivers using convolutional neural network". In: *Pattern Recognition Letters* (2018).

[Mol15a] P. Molchanov, S. Gupta, K. Kim, and J. Kautz. "Hand gesture recognition with 3D convolutional neural networks". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2015, pp. 1–7.

[Mol15b] P. Molchanov, S. Gupta, K. Kim, and K. Pulli. "Multi-sensor system for driver's hand-gesture recognition". In: *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*. Vol. 1. 2015, pp. 1–8.

[Mol16] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz. "Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3D Convolutional Neural Networks". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 4207–4215.

[Ohn13a] Eshed Ohn-Bar, Sujitha Martin, and Mohan Trivedi. "Driver hand activity analysis in naturalistic driving studies: challenges, algorithms, and experimental studies". In: *Journal of Electronic Imaging* 22.4 (2013), p. 041119.

[Ohn13b] Eshed Ohn-Bar and Mohan Trivedi. "In-vehicle hand activity recognition using integration of regions". In: *2013 IEEE Intelligent Vehicles Symposium (IV)*. 2013, pp. 1034–1039.

[Ohn14a] Eshed Ohn-Bar, Sujitha Martin, Ashish Tawari, and Mohan M. Trivedi. "Head, eye, and hand patterns for driver activity recognition". In: *2014 22nd International Conference on Pattern Recognition*. 2014, pp. 660–665.

[Ohn14b] Eshed Ohn-Bar and Mohan Manubhai Trivedi. "Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations". In: *IEEE transactions on intelligent transportation systems* 15.6 (2014), pp. 2368–2377.

[Red17] Joseph Redmon and Ali Farhadi. "YOLO9000: better, faster, stronger". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.

[Ren15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[Ros57] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[S K19] S. Kim, Kimin Yun, Jongyoul Park, and J. Choi. "Skeleton-Based Action Recognition of People Handling Objects". In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2019), pp. 61–70.

[Sha16] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. "NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[Sha19] S. S. Sharan, R. Viji, R. Pradeep, and V. Sajith. "Driver Fatigue Detection Based On Eye State Recognition Using Convolutional Neural Network". In: *2019 International Conference on Communication and Electronics Systems (ICCES)*. 2019, pp. 2057–2063.

[Sim14a] Karen Simonyan and Andrew Zisserman. "Two-Stream Convolutional Networks for Action Recognition in Videos". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc, 2014, pp. 568–576.

[Sim14b] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[Sin17] Gurkirt Singh, Suman Saha, Michael Sapienza, Philip Torr, and Fabio Cuzzolin. "Online Real-Time Multiple Spatiotemporal Action Localisation and Prediction". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 3657–3666. ISBN: 978-1-5386-1032-9.

[Ste13] Sebastian Stein and Stephen J. McKenna. "Combining embedded accelerometers with computer vision for recognizing food preparation activities". In: *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. 2013, pp. 729–738.

[Sze15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[Sze16a] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. "Inception-v4, inception-resnet and the impact of residual connections on learning". In: *arXiv preprint arXiv:1602.07261* (2016).

[Sze16b] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[Tor19] Helena R. Torres, Bruno Oliveira, Jaime Fonseca, Sandro Queirós, João Borges, Nélson Rodrigues, Victor Coelho, Johannes Pallauf, José Brito, and José Mendes. "Real-Time Human Body Pose Estimation for In-Car Depth Images". In: *Doctoral Conference on Computing, Electrical and Industrial Systems*. 2019, pp. 169–182.

[Tos14] Alexander Toshev and Christian Szegedy. "Deeppose: Human pose estimation via deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1653–1660.

[Tra15] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. "Learning spatiotemporal features with 3d convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4489–4497.

[Weh18] Jonatas Wehrmann, Ricardo Cerri, and Rodrigo Barros. "Hierarchical multi-label classification networks". In: *International Conference on Machine Learning*. 2018, pp. 5075–5084.

[Wei16] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. "Convolutional pose machines". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2016, pp. 4724–4732.

[Wey18] Patrick Weyers, Alexander Barth, and Anton Kummert. "Driver state monitoring with hierarchical classification". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 3239–3244.

[Wey19] Patrick Weyers, David Schiebener, and Anton Kummert. "Action and Object Interaction Recognition for Driver Activity Classification". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 4336–4341.

[Wey21] Patrick Weyers and Anton Kummert. "Continuous Driver Activity Recognition from Short Isolated Action Sequences". In: *Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM,* INSTICC. SciTePress, 2021, pp. 158–165.

[Wor11] World Health Organization et al. "Mobile phone use: a growing problem of driver distraction". In: (2011).

[Wor18] World Health Organization et al. *Global status report on road safety 2018*. Geneva, Switzerland: World Health Organization, 2018. ISBN: 9789241565684.

[Xin19] Yang Xing, Chen Lv, Huaji Wang, Dongpu Cao, Efstathios Velenis, and Fei-Yue Wang. "Driver activity recognition for intelligent vehicles: A deep learning approach". In: *IEEE Transactions on Vehicular Technology* 68.6 (2019), pp. 5379–5390.

[Xu14] Lijie Xu and Kikuo Fujimura. "Real-time driver activity recognition with random forests". In: *Proceedings of the 6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. 2014, pp. 1–8.

[Yan16a] Chao Yan, Frans Coenen, and Bailing Zhang. "Driving posture recognition by convolutional neural networks". In: *IET Computer Vision* 10.2 (2016), pp. 103–114.

[Yan16b] Shiyang Yan, Yuxuan Teng, Jeremy S. Smith, and Bailing Zhang. "Driver behavior recognition based on deep convolutional neural networks". In: *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*. 2016, pp. 636–641.

[Yue15] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. "Beyond short snippets: Deep networks for video classification". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 4694–4702.

[Zha06] Zutao Zhang and Jia-shu Zhang. "Driver fatigue detection based intelligent vehicle control". In: *18th International Conference on Pattern Recognition (ICPR'06)*. Vol. 2. 2006, pp. 1262–1265.

[Zha17] 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 (WACV)*. IEEE, 2017, pp. 148–157. ISBN: 978-1-5090-4822-9.

[Zho88] Yi-Tong Zhou and Rama Chellappa. "Computation of optical flow using a neural network". In: *ICNN*. 1988, pp. 71–78.

[Zim12] Hans-Georg Zimmermann, Christoph Tietz, and Ralph Grothmann. "Forecasting with recurrent neural networks: 12 tricks". In: *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 687–707.

[Zol18] Mohammadreza Zolfaghari, Kamaljeet Singh, and Thomas Brox. "Eco: Efficient convolutional network for online video understanding". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 695–712.

List of Figures

List of Tables

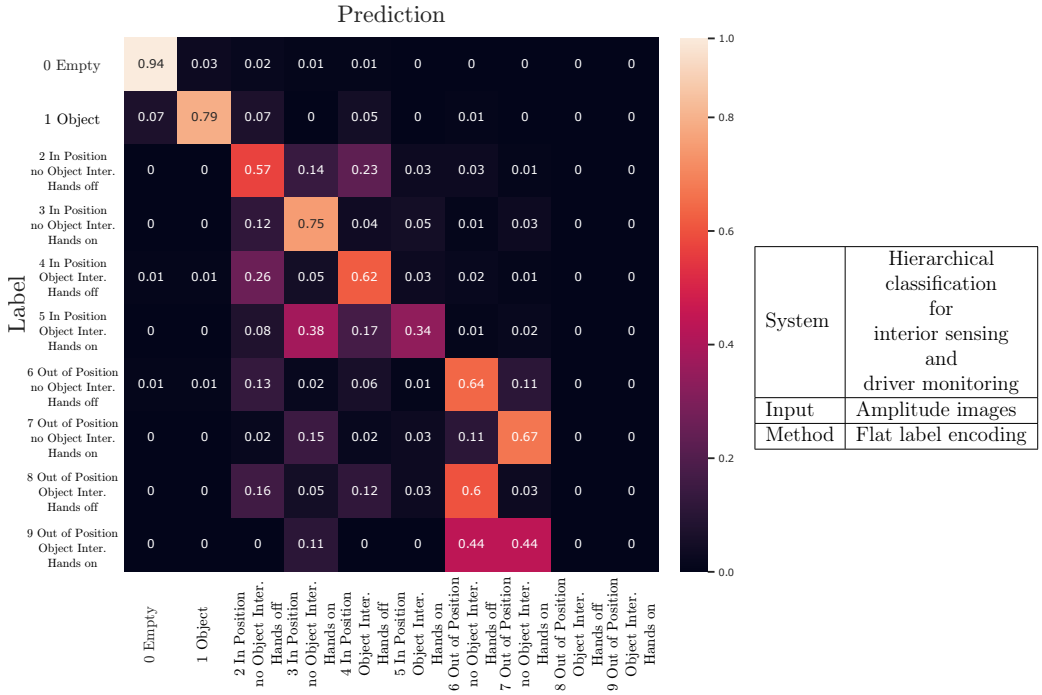
APPENDIX A

Hierarchical Classification for Interior Sensing and Driver Monitoring

Flat encoding

Following, the confusion matrices and precision recall curves of the remaining systems trained with a flat label encoding described in chapter 4.4 are shown and belong to the systems evaluated in table 4.2. These remaining system are either trained on amplitude or depth images.

Amplitude Input



System	Hierarchical classification for interior sensing and driver monitoring
Input	Amplitude images
Method	Flat label encoding

Figure A.1: Relative confusion matrix of the validation results computed by the networks trained with the flat class label approach and amplitude images as input

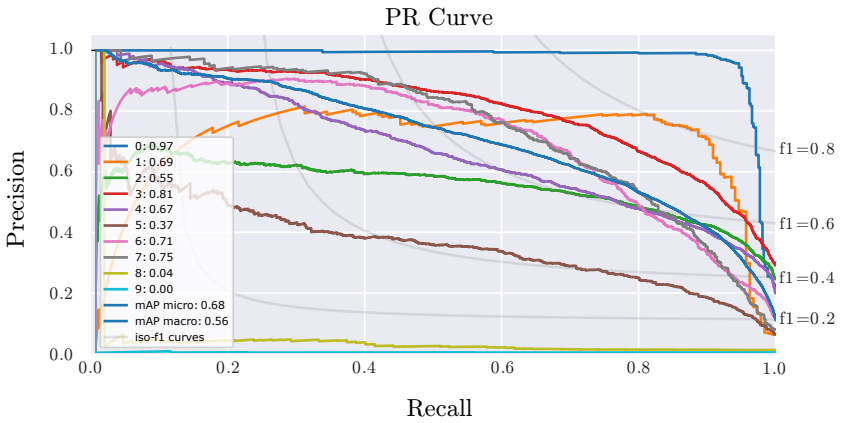
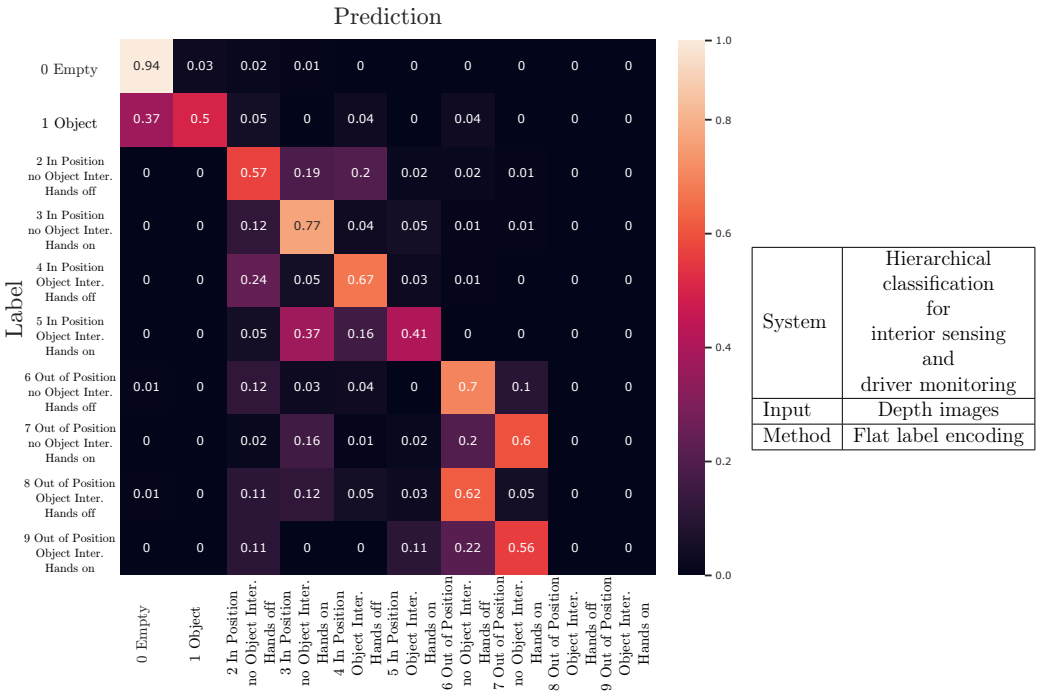


Figure A.2: Precision recall curves of the validation results computed by the networks trained with the flat class label approach and amplitude images as input

Depth Input



System	Hierarchical classification for interior sensing and driver monitoring
Input	Depth images
Method	Flat label encoding

Figure A.3: Relative confusion matrix of the validation results computed by the networks trained with the flat class label approach and depth images as input

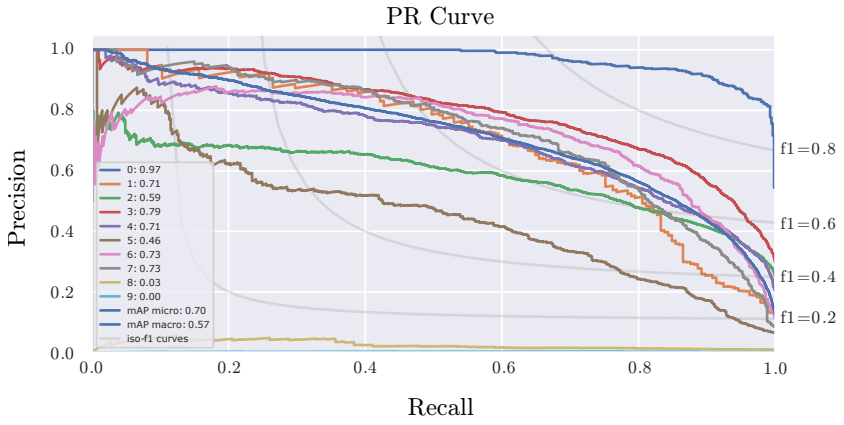


Figure A.4: Precision recall curves of the validation results computed by the networks trained with the flat class label approach and depth images as input

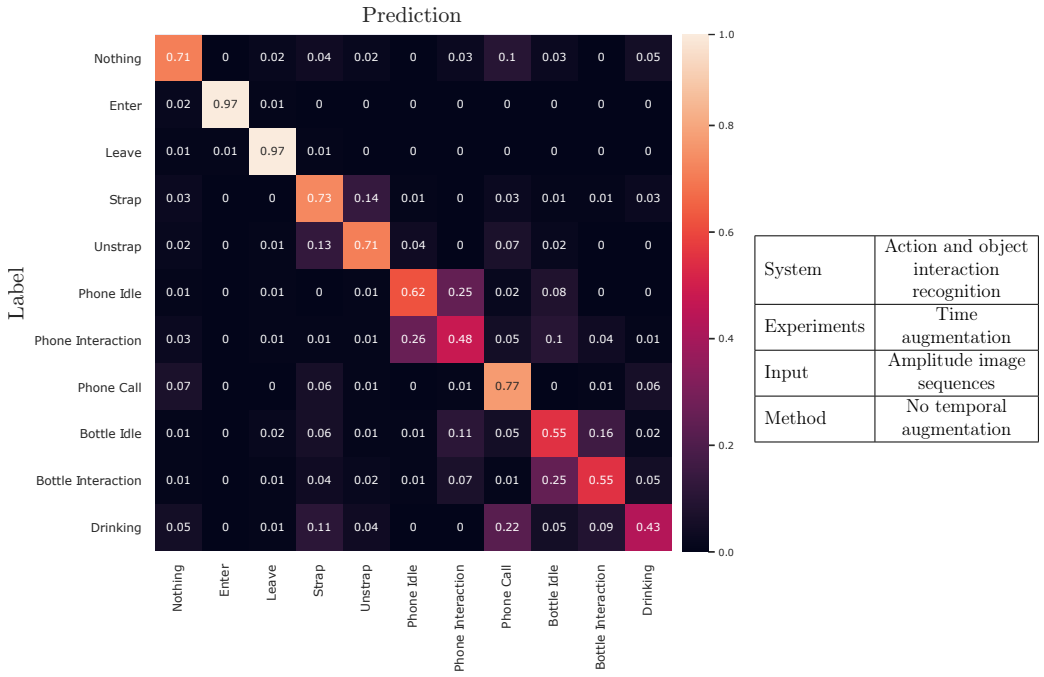
APPENDIX B

Action and Object Interaction

Time Augmentation

Following, the confusion matrices and precision recall curves of the remaining systems trained with and without the time augmentation method described in chapter 5.2.3 are shown and belong to the systems evaluated in table 5.2. The systems were trained to evaluate the time augmentation method proposed in chapter 5.2 with amplitude, depth and optical flow image sequences of the driver's seat region.

Amplitude Input



System	Action and object interaction recognition
Experiments	Time augmentation
Input	Amplitude image sequences
Method	No temporal augmentation

Figure B.1: Relative confusion matrix of the validation predictions from the network trained on amplitude image sequences without temporal augmentation

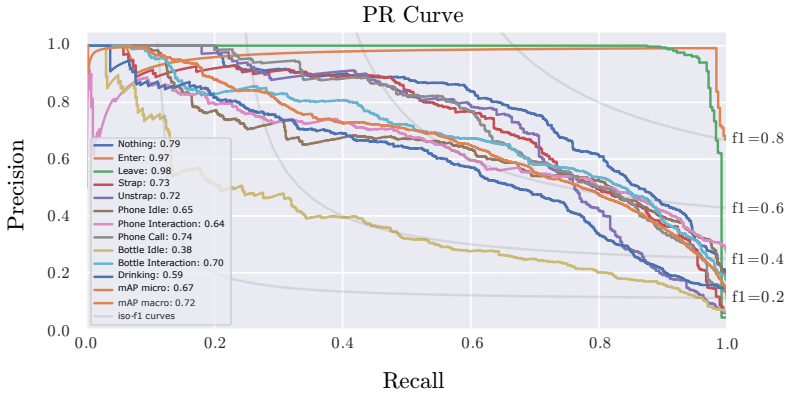
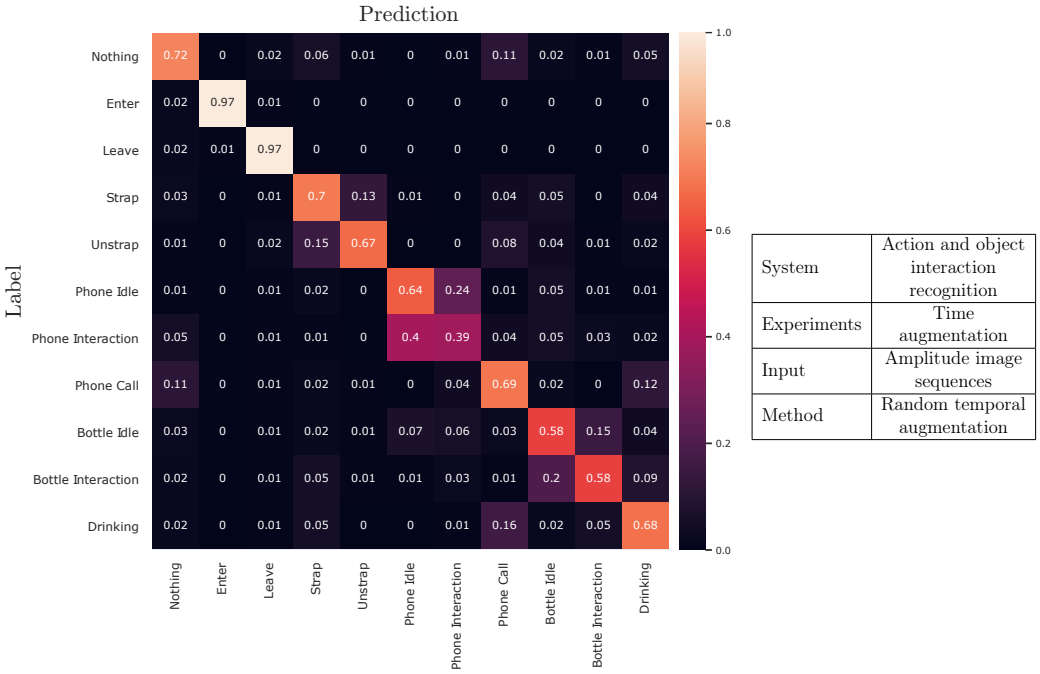


Figure B.2: Precision-Recall curves of the validation predictions from the network trained on amplitude image sequences without temporal augmentation



System	Action and object interaction recognition
Experiments	Time augmentation
Input	Amplitude image sequences
Method	Random temporal augmentation

Figure B.3: Relative confusion matrix of the validation predictions from the network trained on amplitude image sequences with random temporal augmentation

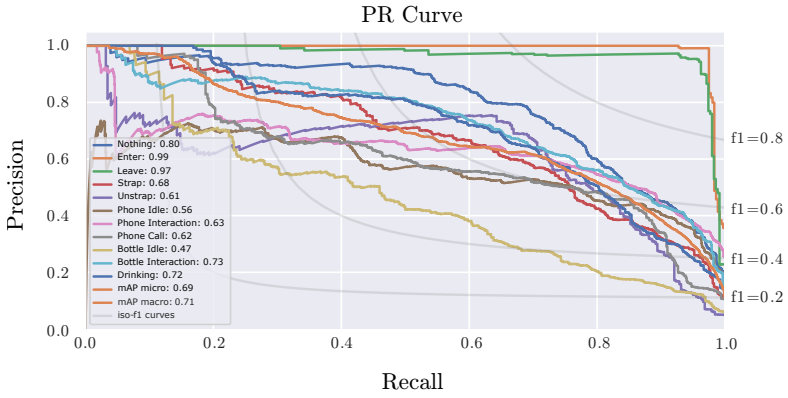
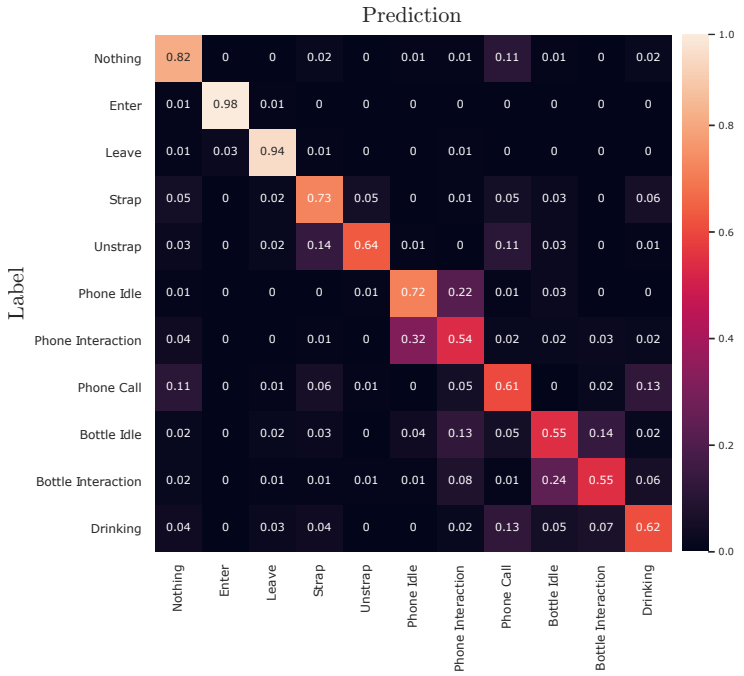


Figure B.4: Precision-Recall curves of the validation predictions from the network trained on amplitude image sequences with random temporal augmentation



System	Action and object interaction recognition
Experiments	Time augmentation
Input	Amplitude image sequences
Method	Systematical temporal augmentation

Figure B.5: Relative confusion matrix of the validation predictions from the network trained on amplitude image sequences with systematical temporal augmentation

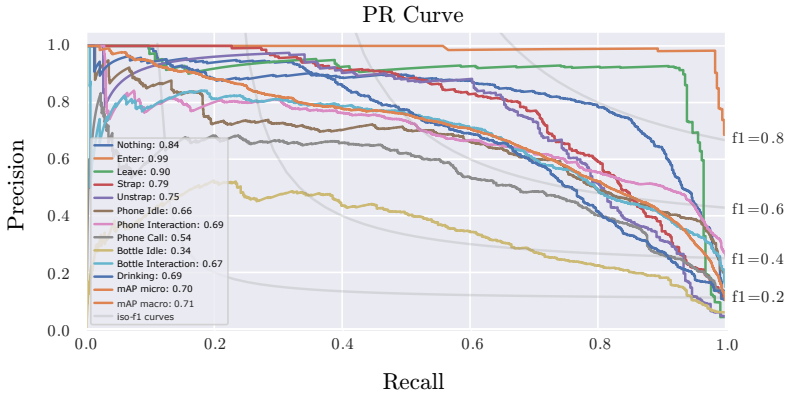
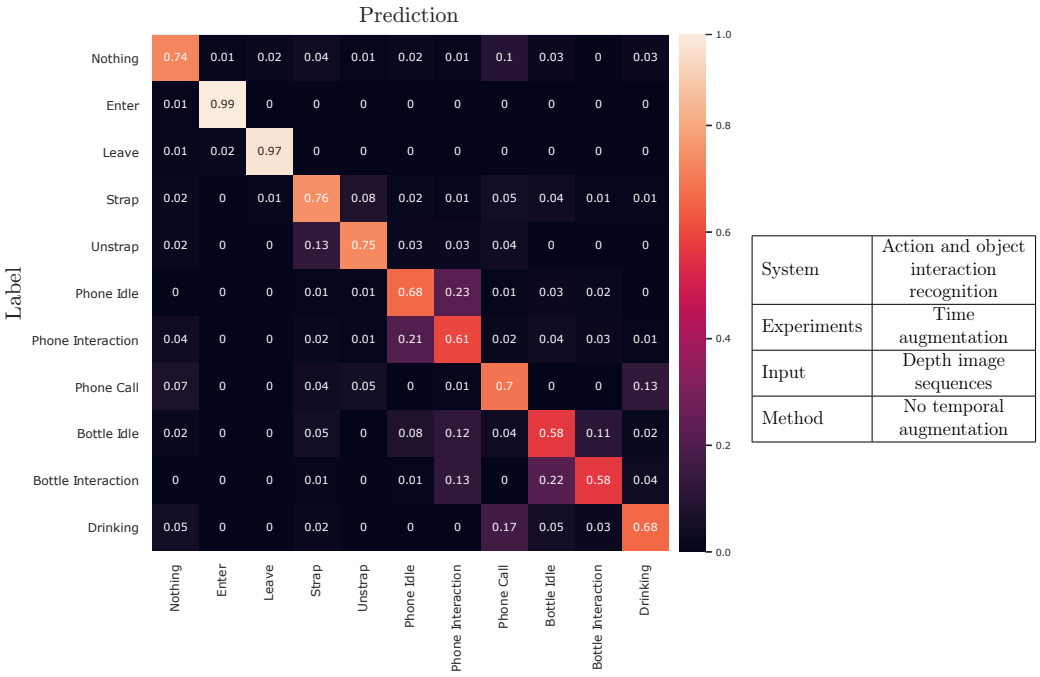


Figure B.6: Precision-Recall curves of the validation predictions from the network trained on amplitude image sequences with systematical temporal augmentation

Depth Input



System	Action and object interaction recognition
Experiments	Time augmentation
Input	Depth image sequences
Method	No temporal augmentation

Figure B.7: Relative confusion matrix of the validation predictions from the network trained on depth image sequences without temporal augmentation

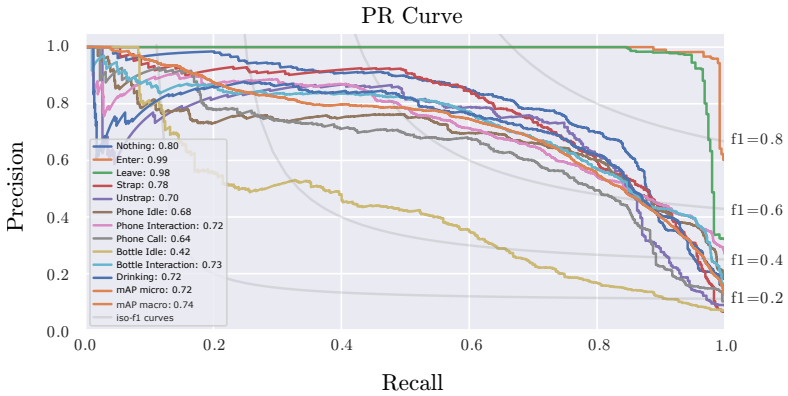
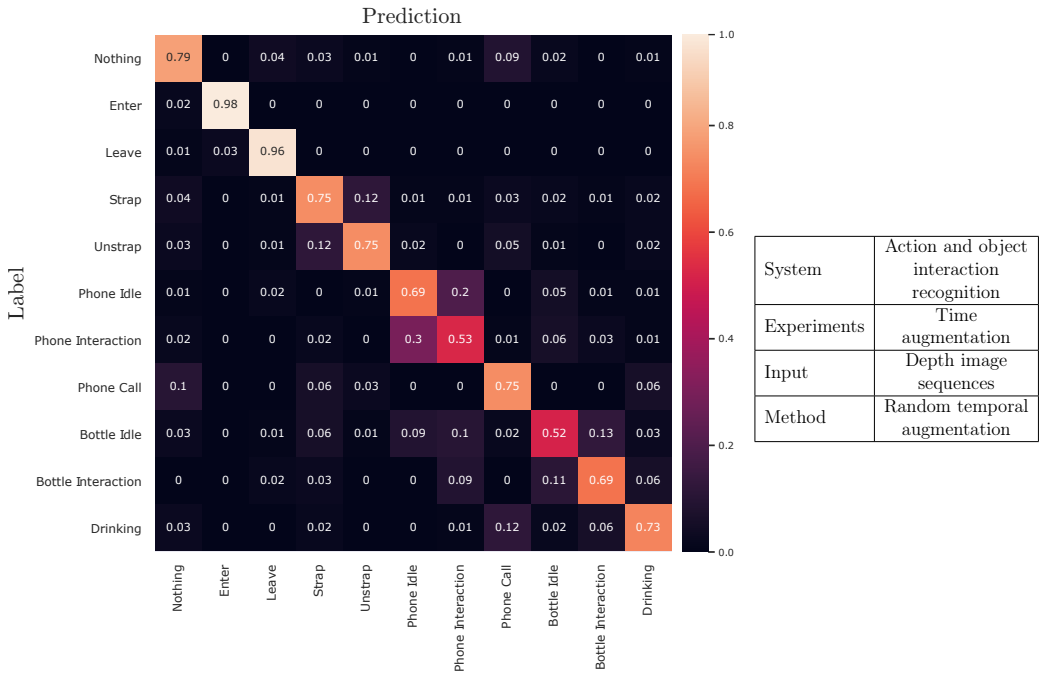


Figure B.8: Precision-Recall curves of the validation predictions from the network trained on depth image sequences without temporal augmentation



System	Action and object interaction recognition
Experiments	Time augmentation
Input	Depth image sequences
Method	Random temporal augmentation

Figure B.9: Relative confusion matrix of the validation predictions from the network trained on depth image sequences with random temporal augmentation

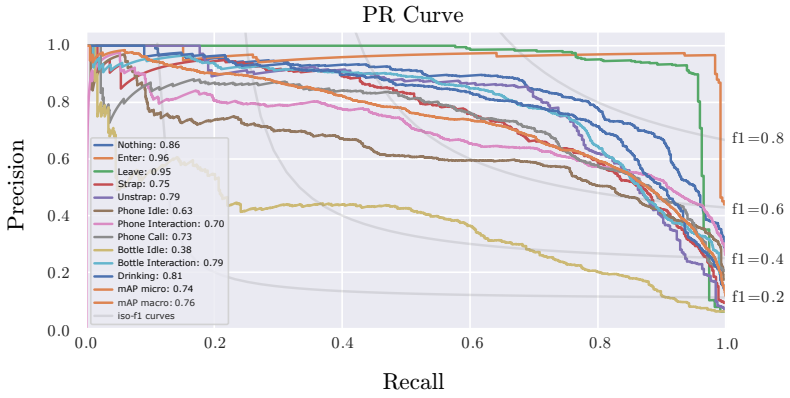
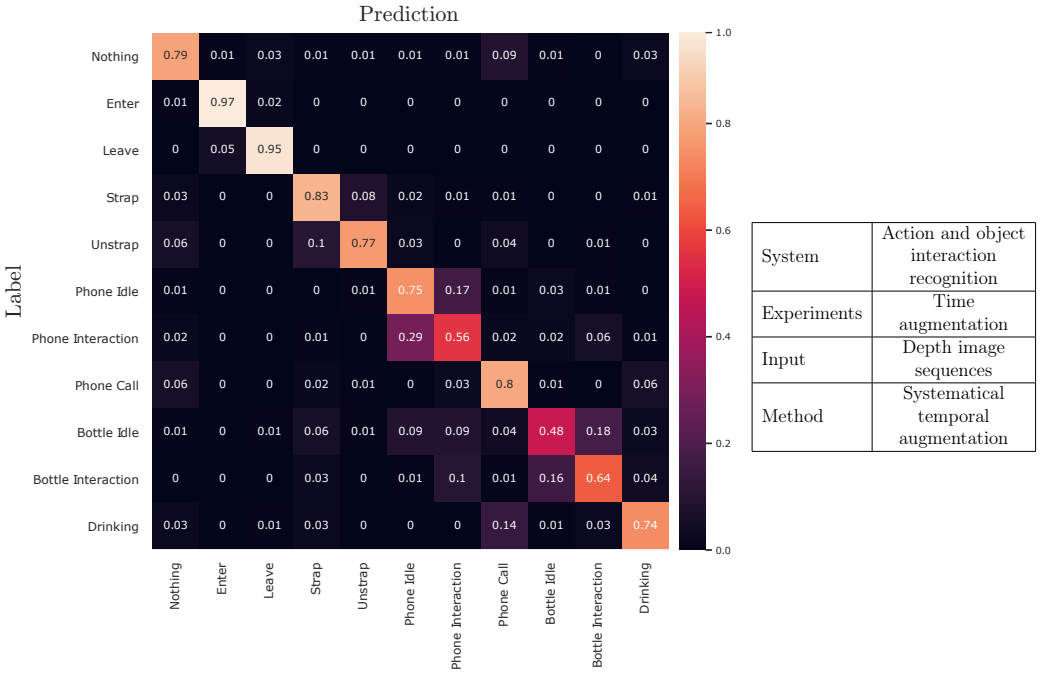


Figure B.10: Precision-Recall curves of the validation predictions from the network trained on depth image sequences with random temporal augmentation



System	Action and object interaction recognition
Experiments	Time augmentation
Input	Depth image sequences
Method	Systematical temporal augmentation

Figure B.11: Relative confusion matrix of the validation predictions from the network trained on depth image sequences with systematical temporal augmentation

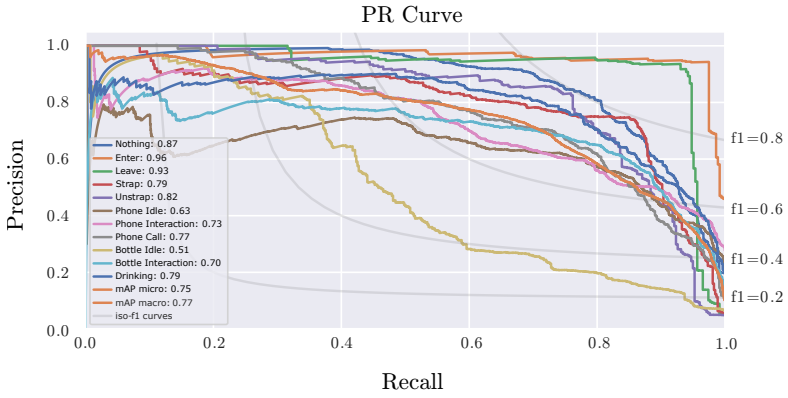
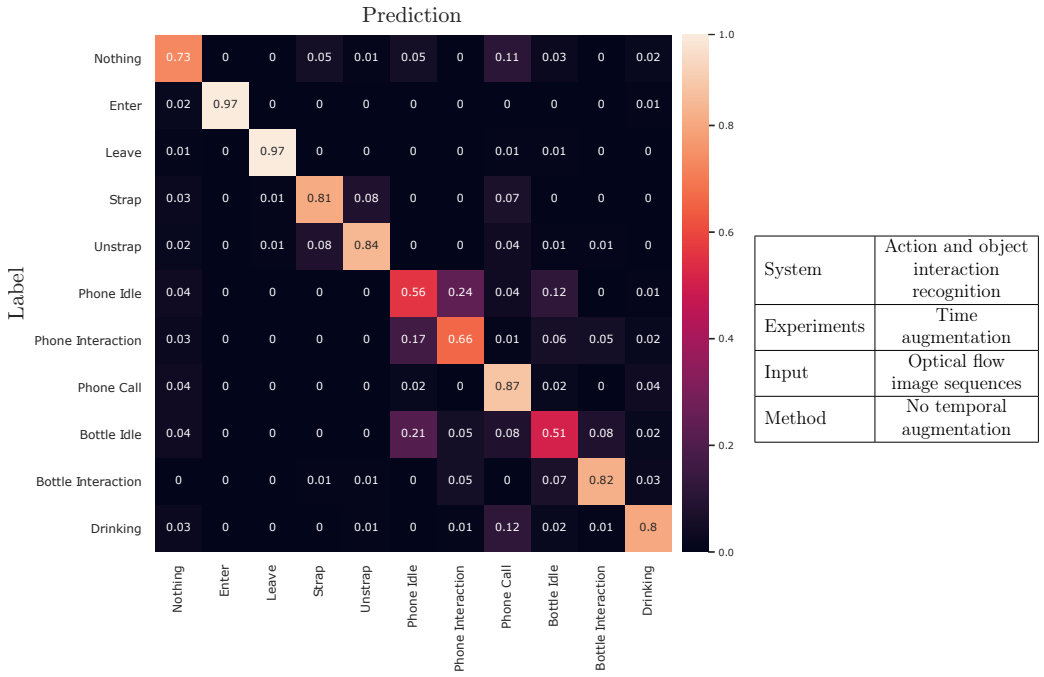


Figure B.12: Precision-Recall curves of the validation predictions from the network trained on depth image sequences with systematical temporal augmentation

Optical Flow Input



System	Action and object interaction recognition
Experiments	Time augmentation
Input	Optical flow image sequences
Method	No temporal augmentation

Figure B.13: Relative confusion matrix of the validation predictions from the network trained on flow image sequences without temporal augmentation

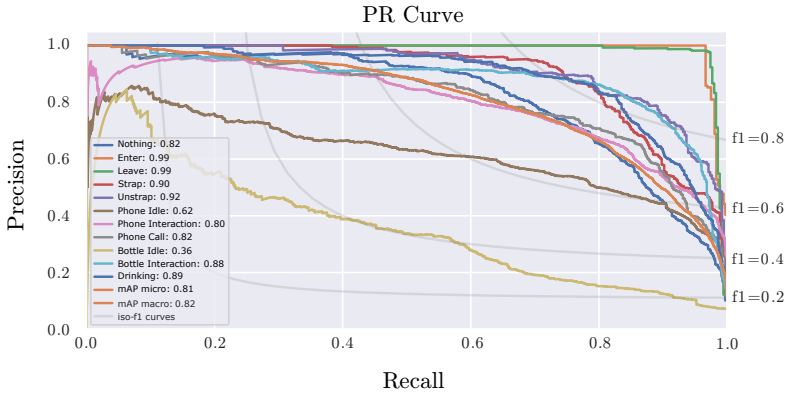
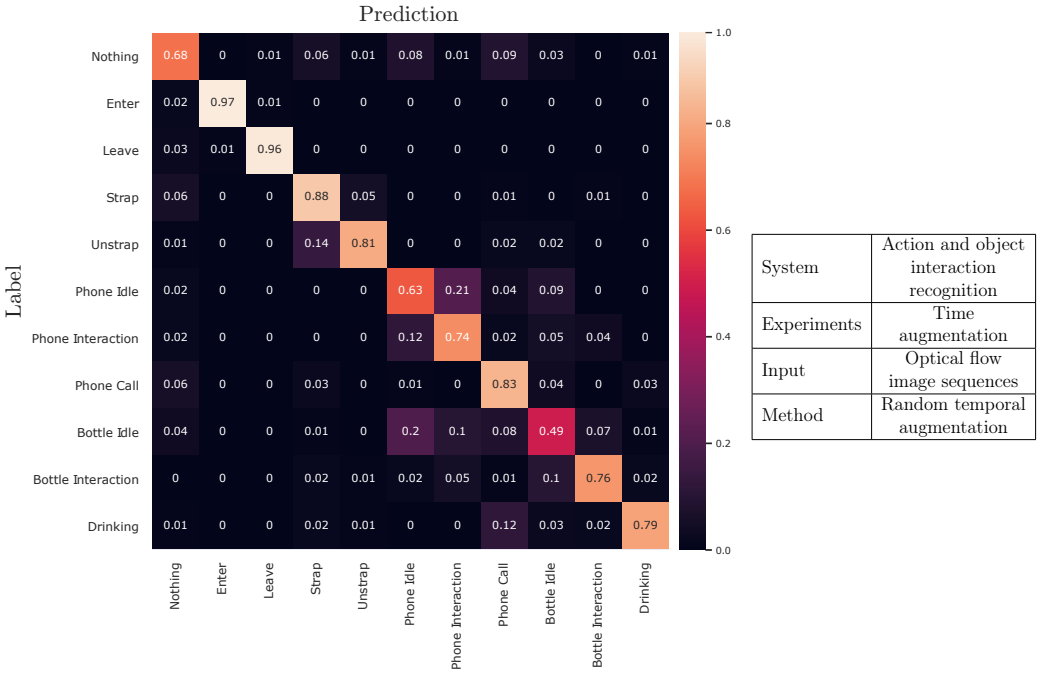


Figure B.14: Precision-Recall curves of the validation predictions from the network trained on flow image sequences without temporal augmentation



System	Action and object interaction recognition
Experiments	Time augmentation
Input	Optical flow image sequences
Method	Random temporal augmentation

Figure B.15: Relative confusion matrix of the validation predictions from the network trained on flow image sequences with random temporal augmentation

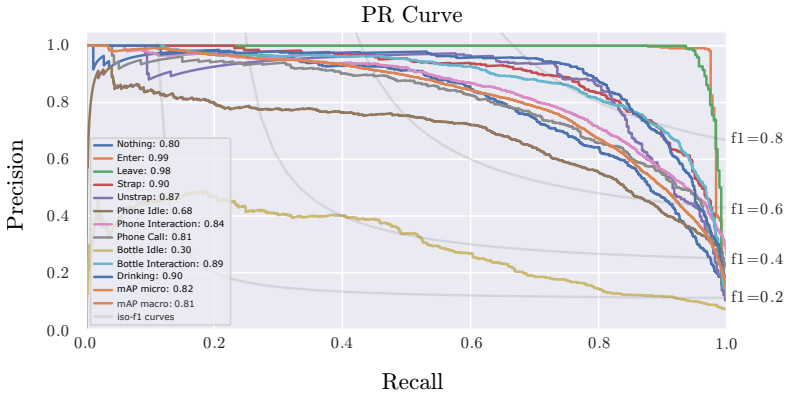
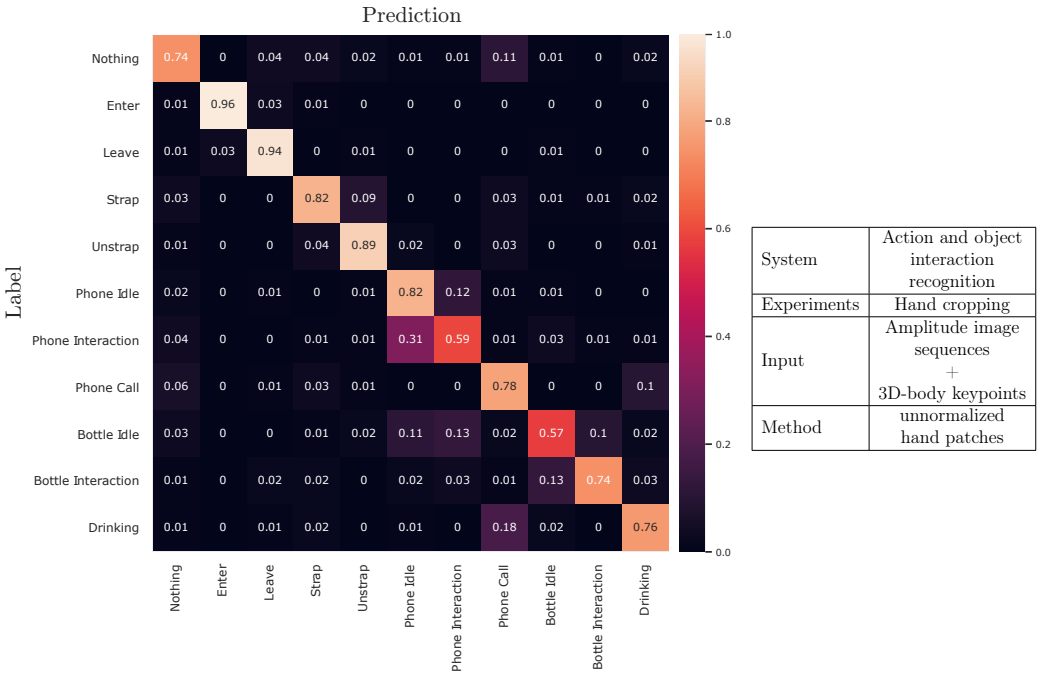


Figure B.16: Precision-Recall curves of the validation predictions from the network trained on flow image sequences with random temporal augmentation

Hand Normalization

Following, the confusion matrices and precision recall curves of the remaining systems described in chapter 5.3.6 are shown and belong to the systems evaluated in table 5.4. The networks were trained with normalized and unnormalized hand patch sequences with 3D body keypoints. The hand patches origin from the amplitude, depth and optical flow image sequences.

Amplitude Input



System	Action and object interaction recognition
Experiments	Hand cropping
Input	Amplitude image sequences + 3D-body keypoints
Method	unnormalized hand patches

Figure B.17: Relative confusion matrix of validation results from the action and object interaction recognition systems trained on 3D body keypoints and original amplitude hand crop images

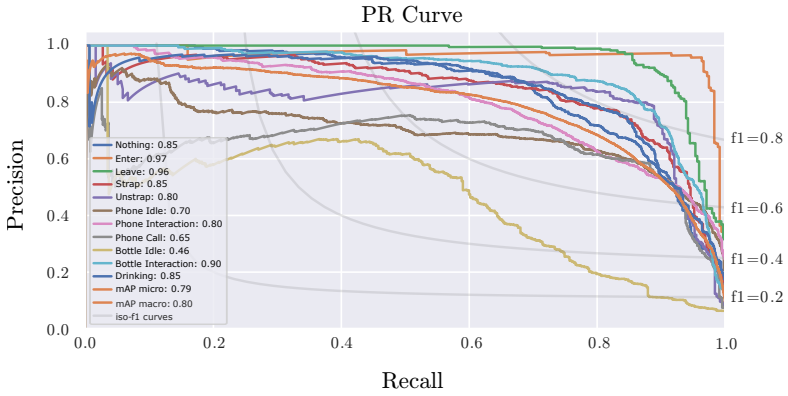
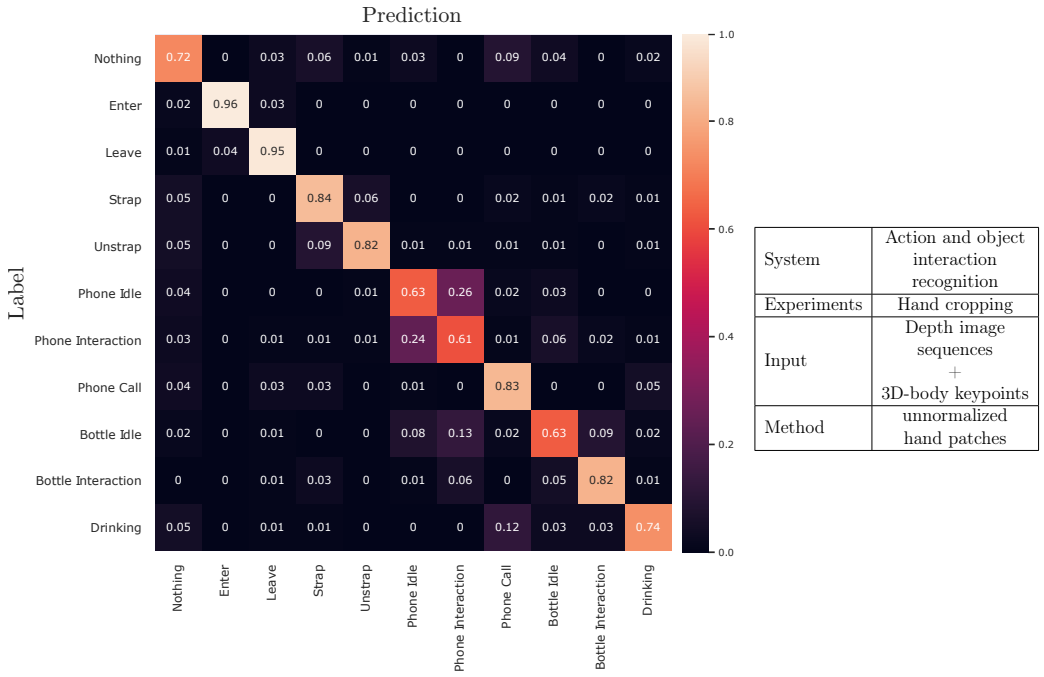


Figure B.18: Precision-Recall curves of validation results from the action and object interaction recognition systems trained on 3D body keypoints and original amplitude hand crop images

Depth Input



System	Action and object interaction recognition
Experiments	Hand cropping
Input	Depth image sequences + 3D-body keypoints
Method	unnormalized hand patches

Figure B.19: Relative confusion matrix of validation results from the action and object interaction recognition systems trained on 3D body keypoints and original depth hand crop images

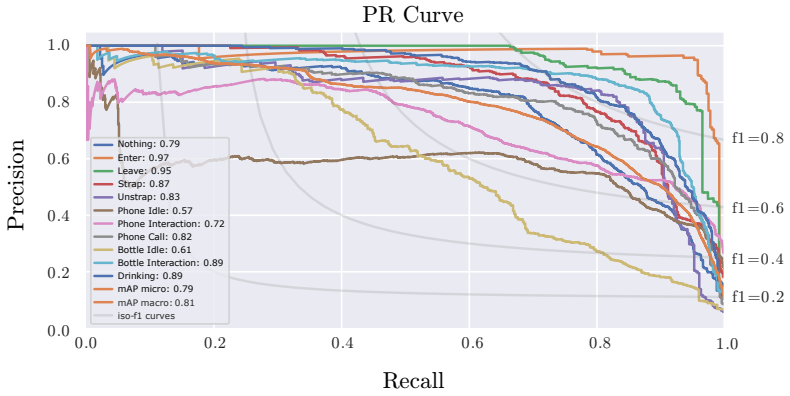


Figure B.20: Precision-Recall curves of validation results from the action and object interaction recognition systems trained on 3D body keypoints and original depth hand crop images

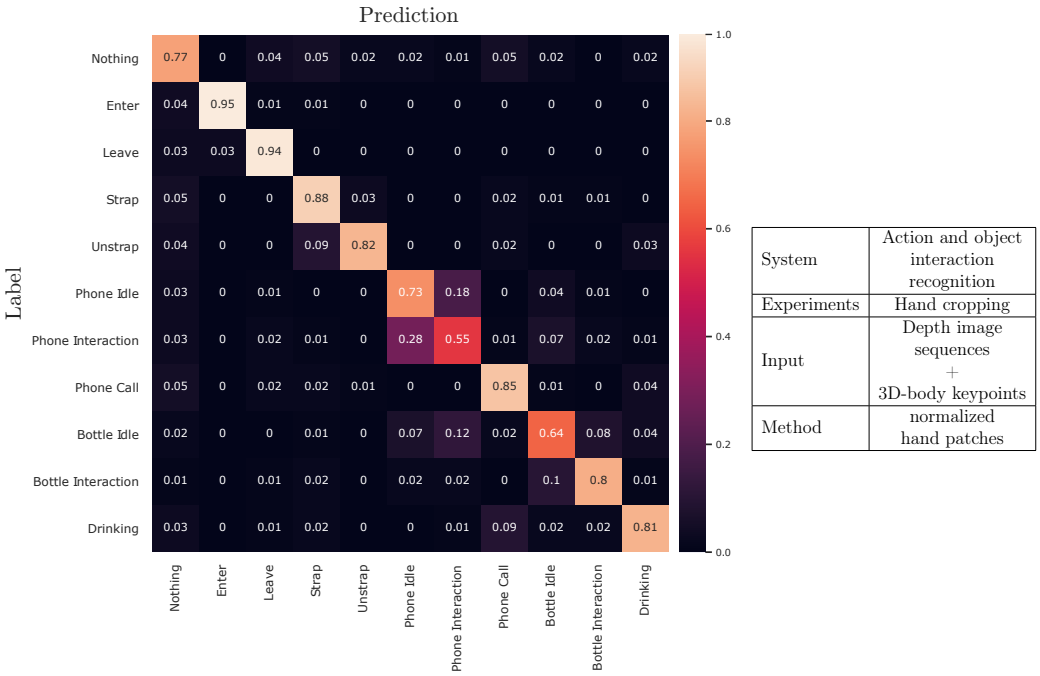


Figure B.21: Relative confusion matrix of validation results from the action and object interaction recognition systems trained on 3D body keypoints and normalized depth hand crop images

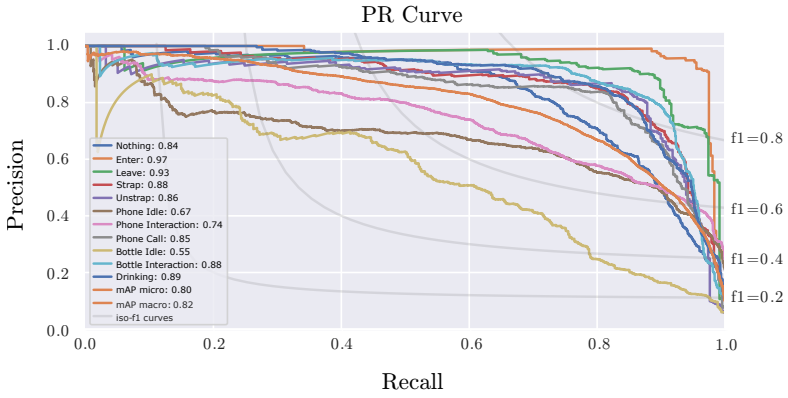
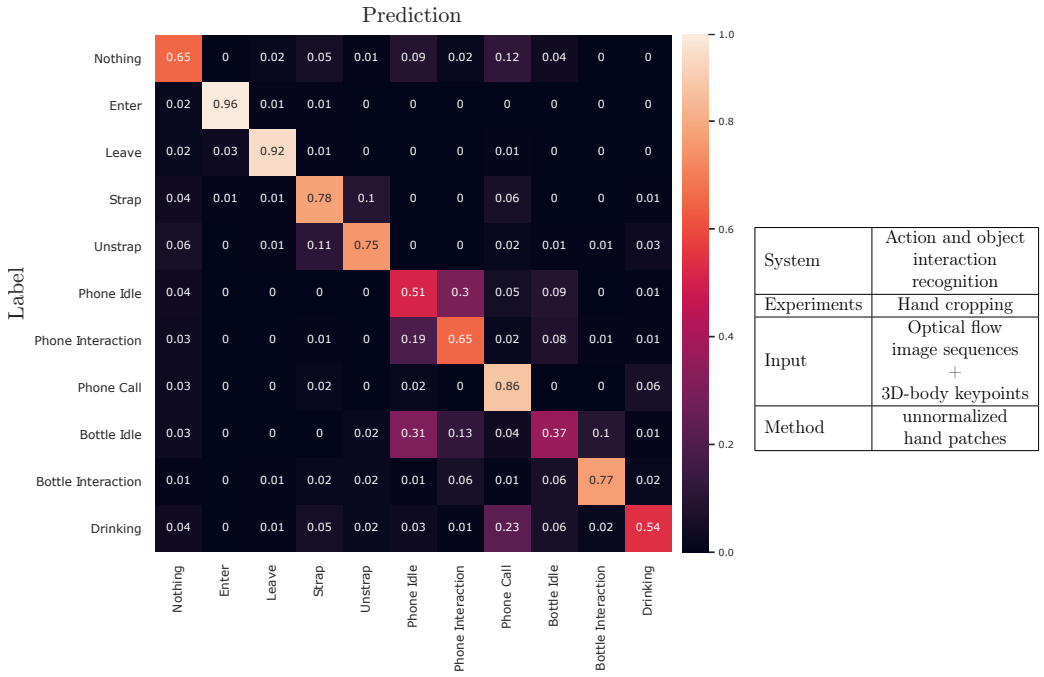


Figure B.22: Precision-Recall curves of validation results from the action and object interaction recognition systems trained on 3D body keypoints and normalized depth hand crop images

Optical Flow Input



System	Action and object interaction recognition
Experiments	Hand cropping
Input	Optical flow image sequences + 3D-body keypoints
Method	unnormalized hand patches

Figure B.23: Relative confusion matrix of validation results from the action and object interaction recognition systems trained on 3D body keypoints and original optical flow hand crop images

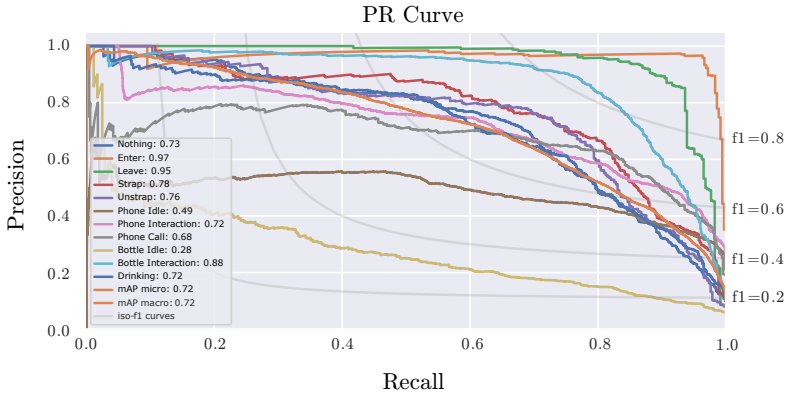


Figure B.24: Precision-Recall curves of validation results from the action and object interaction recognition systems trained on 3D body keypoints and original optical flow hand crop images

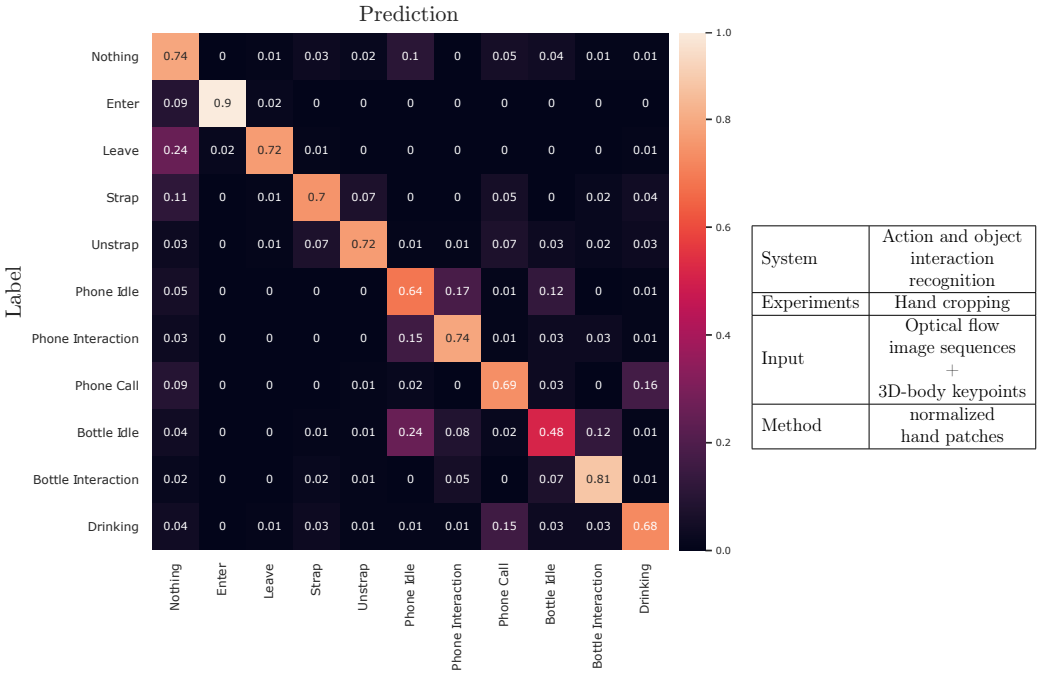


Figure B.25: Relative confusion matrix of validation results from the action and object interaction recognition systems trained on 3D body keypoints and normalized optical flow hand crop images

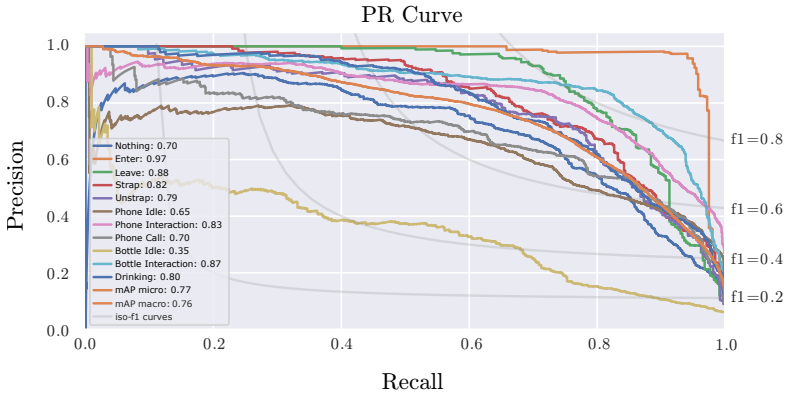


Figure B.26: Precision-Recall curves of validation results from the action and object interaction recognition systems trained on 3D body keypoints and normalized optical flow hand crop images

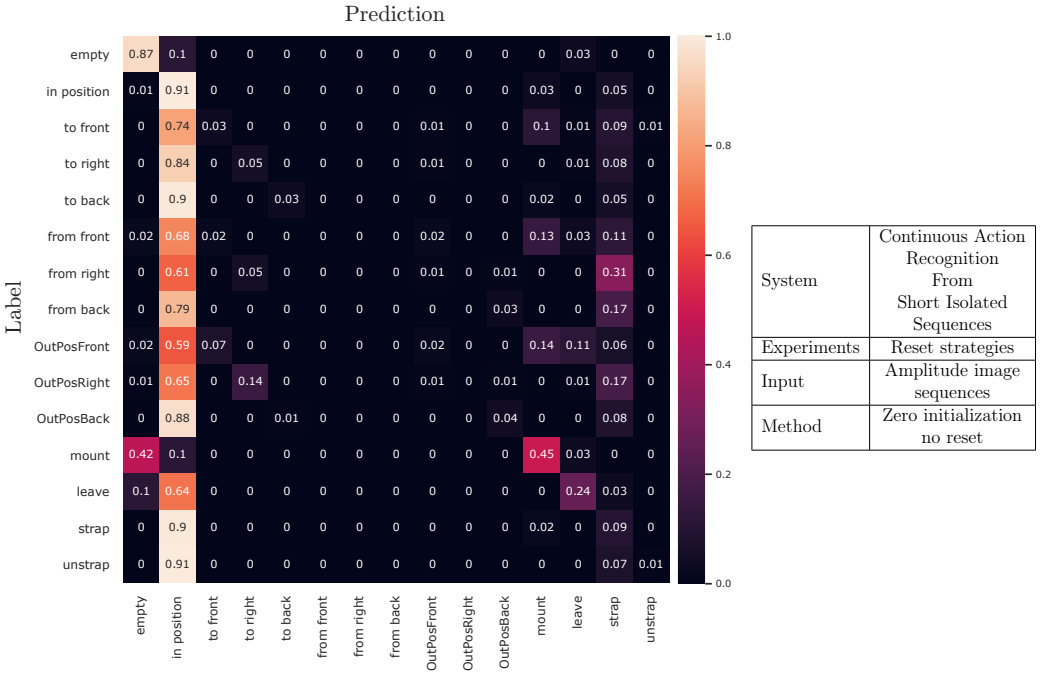
APPENDIX C

Continuous Action Recognition From Short Isolated Sequences

Reset Results

Following, the confusion matrices and precision recall curves of the remaining systems described in chapter 6.2.1 are shown and belong to the systems evaluated in table 6.2. The networks were trained with zero and random initialization and tested with the different reset strategies at test time described in chapter 6.2.1. As input to the network short amplitude, depth or optical flow image sequences were used to train the networks while longer sequences of the corresponding training input format were used for testing.

Amplitude Input



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Amplitude image sequences
Method	Zero initialization no reset

Figure C.1: Relative confusion matrix of the action recognition network trained on amplitude image sequences trained with zero initialization and no reset at runtime

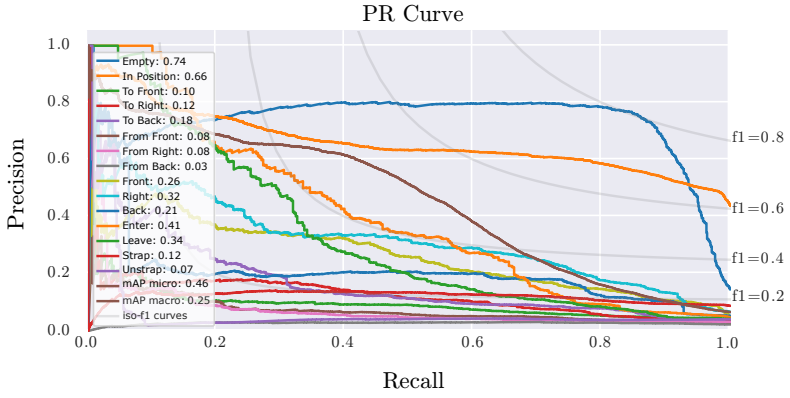
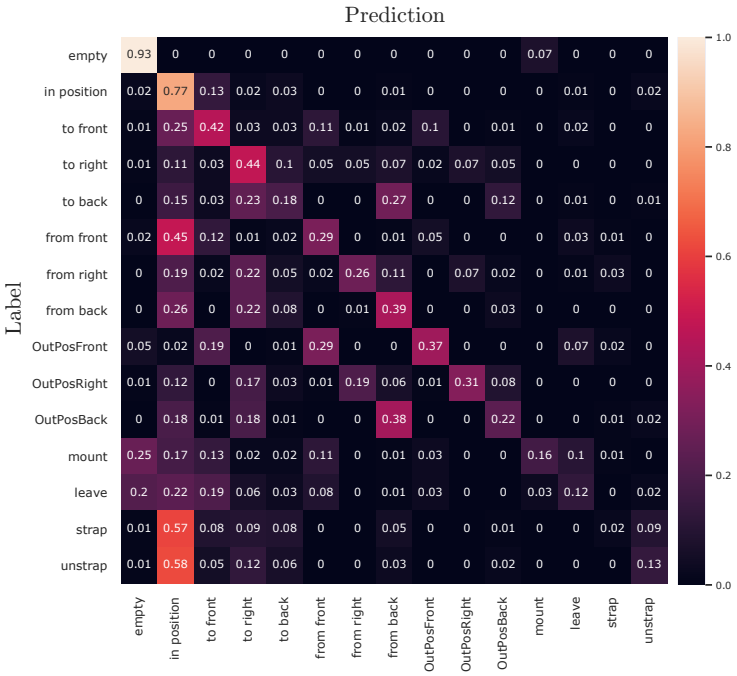


Figure C.2: Precision recall curves of the action recognition network trained on amplitude image sequences trained with zero initialization and no reset at runtime



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Amplitude image sequences
Method	Zero initialization reset at minimum sequence length

Figure C.3: Relative confusion matrix of the action recognition network trained on amplitude image sequences trained with zero initialization and reset at the minimum sequence length at runtime

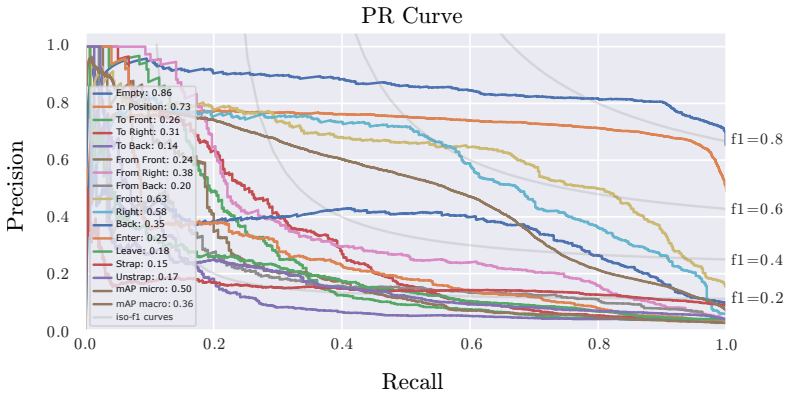
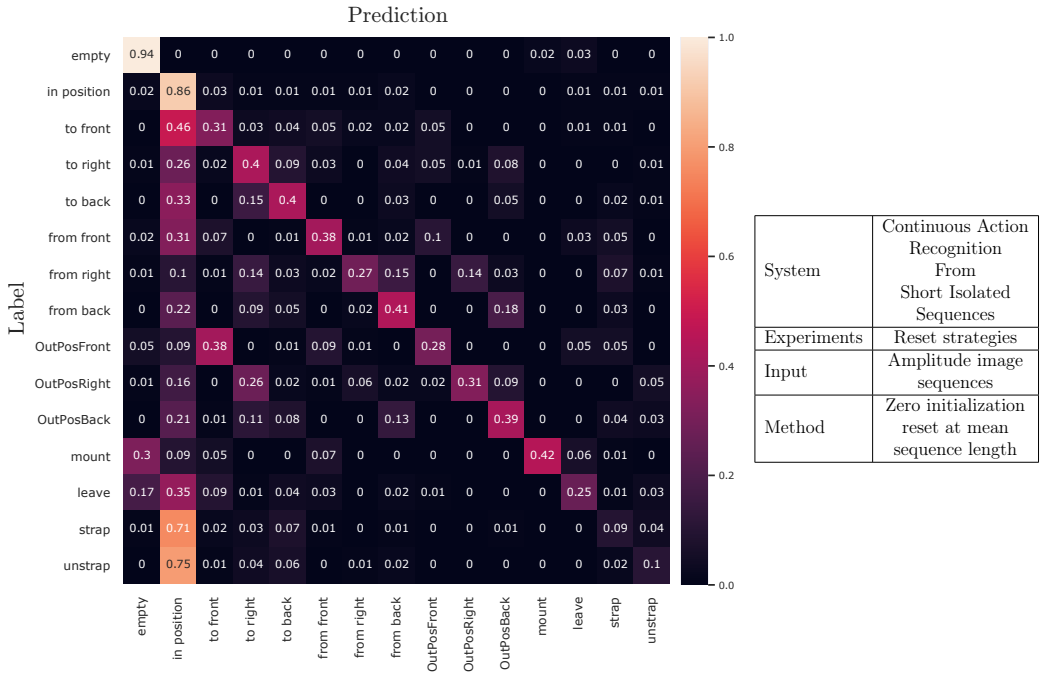


Figure C.4: Precision recall curves of the action recognition network trained on amplitude image sequences trained with zero initialization and reset at the minimum sequence length runtime



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Amplitude image sequences
Method	Zero initialization reset at mean sequence length

Figure C.5: Relative confusion matrix of the action recognition network trained on amplitude image sequences trained with zero initialization and reset at the mean sequence length at runtime

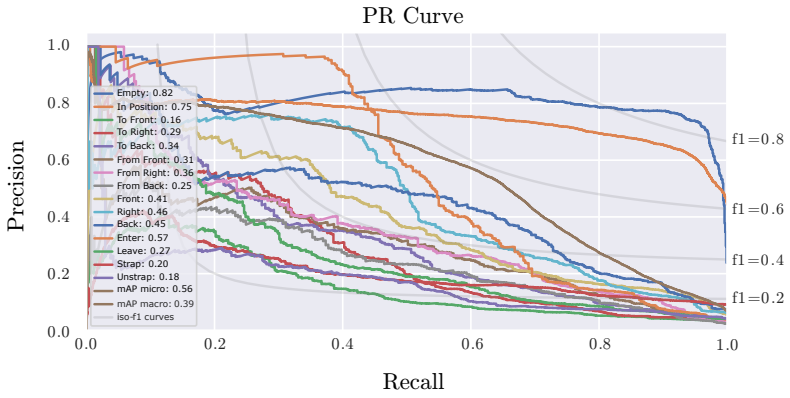
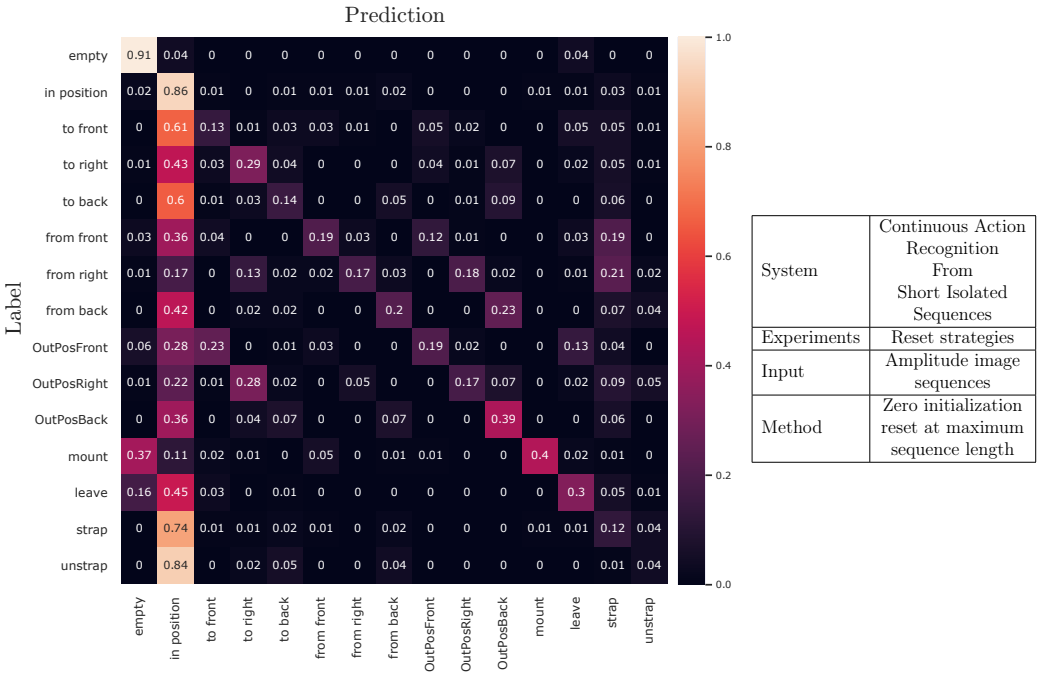


Figure C.6: Precision recall curves of the action recognition network trained on amplitude image sequences trained with zero initialization and reset at the mean sequence length runtime



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Amplitude image sequences
Method	Zero initialization reset at maximum sequence length

Figure C.7: Relative confusion matrix of the action recognition network trained on amplitude image sequences trained with zero initialization and reset at the maximum sequence length at runtime

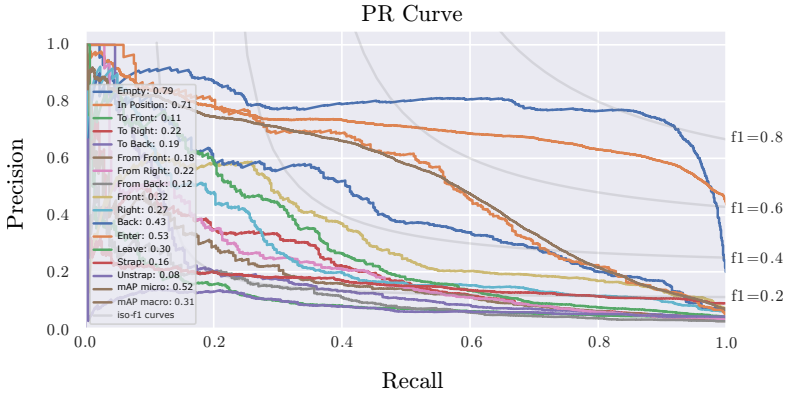
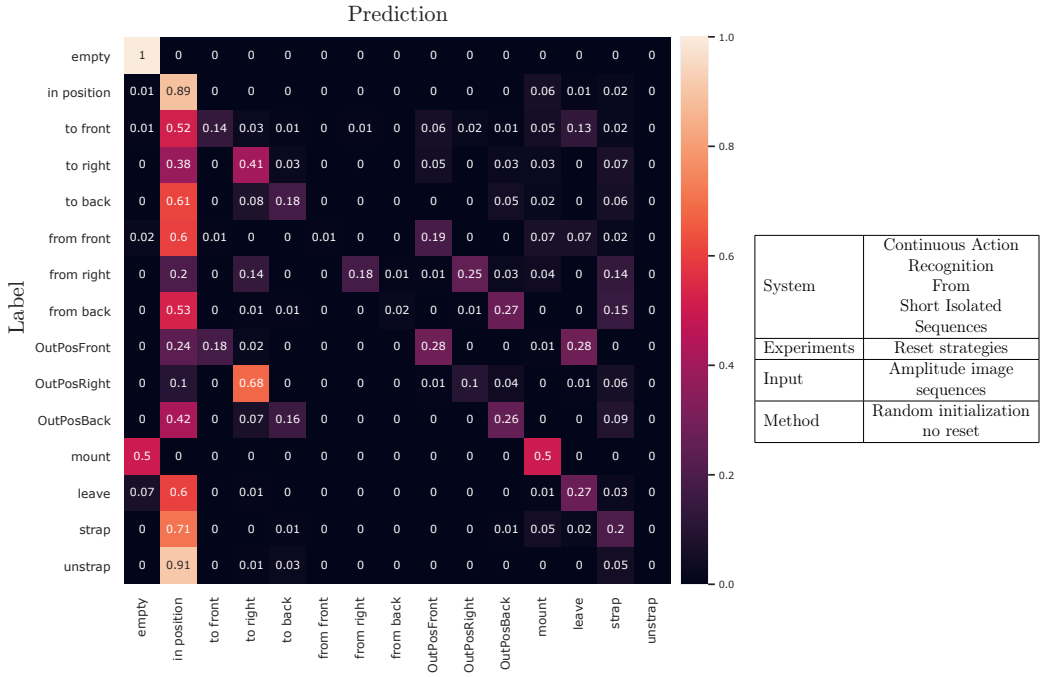


Figure C.8: Precision recall curves of the action recognition network trained on amplitude image sequences trained with zero initialization and reset at the maximum sequence length runtime



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Amplitude image sequences
Method	Random initialization no reset

Figure C.9: Relative confusion matrix of the action recognition network trained on amplitude image sequences trained with random initialization and no reset at runtime

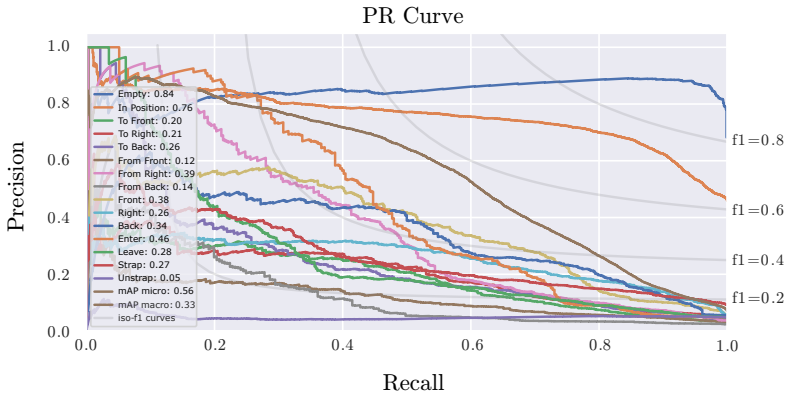
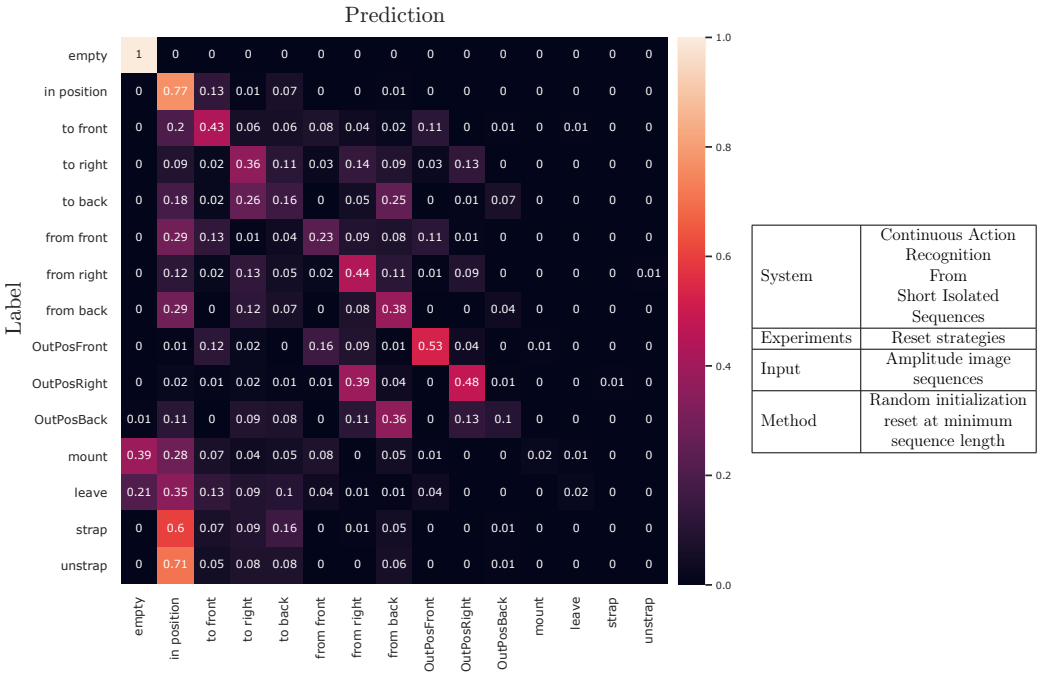


Figure C.10: Precision recall curves of the action recognition network trained on amplitude image sequences trained with random initialization and no reset at runtime



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Amplitude image sequences
Method	Random initialization reset at minimum sequence length

Figure C.11: Relative confusion matrix of the action recognition network trained on amplitude image sequences trained with random initialization and reset at the minimum sequence length at runtime

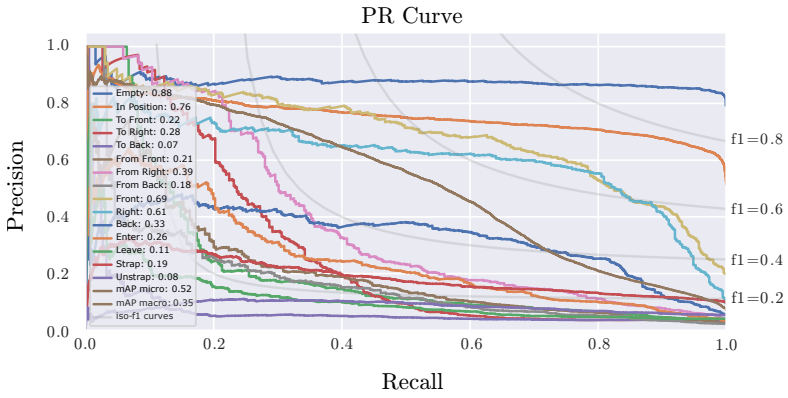
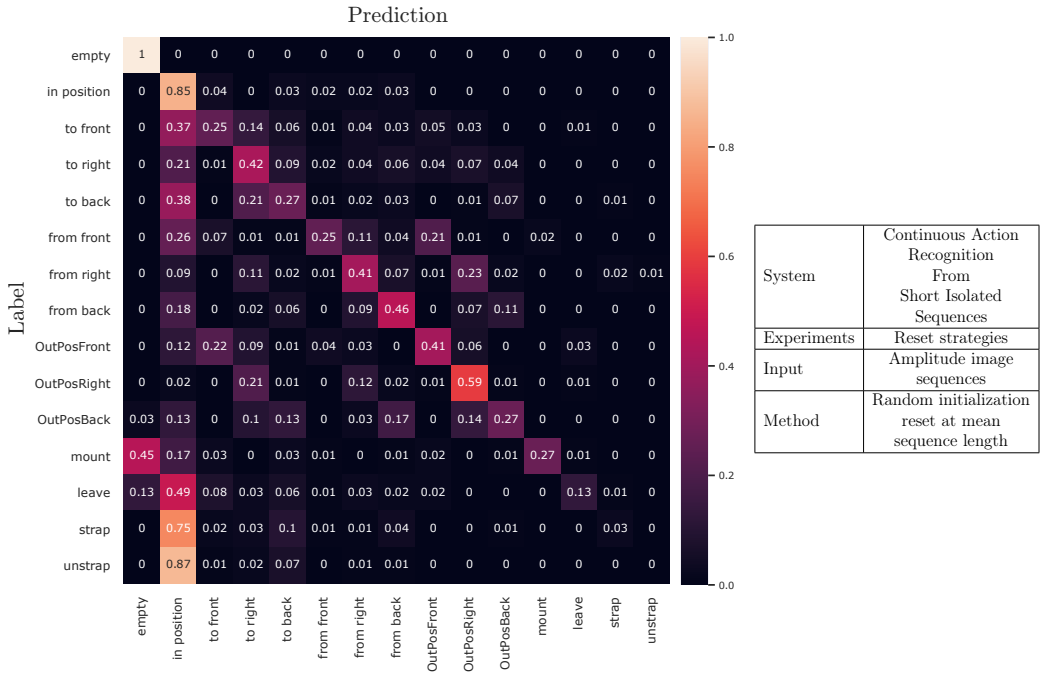


Figure C.12: Precision recall curves of the action recognition network trained on amplitude image sequences trained with random initialization and reset at the minimum sequence length runtime



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Amplitude image sequences
Method	Random initialization reset at mean sequence length

Figure C.13: Relative confusion matrix of the action recognition network trained on amplitude image sequences trained with random initialization and reset at the mean sequence length at runtime

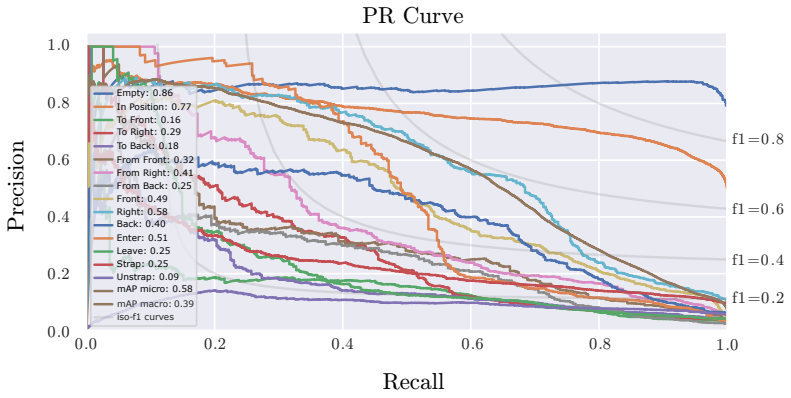
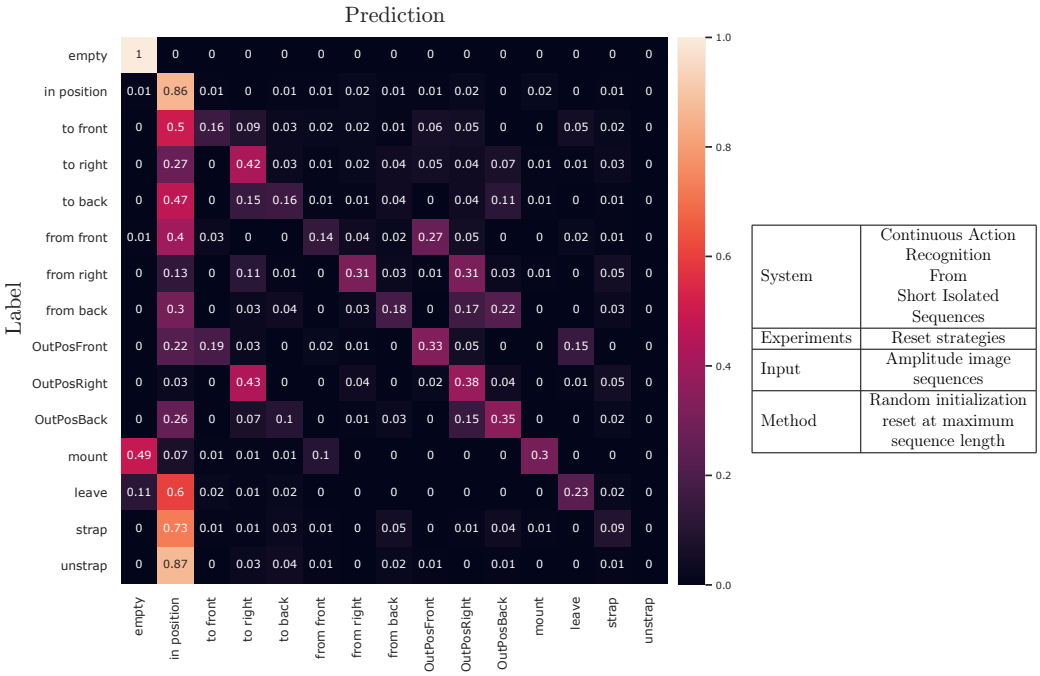


Figure C.14: Precision recall curves of the action recognition network trained on amplitude image sequences trained with random initialization and reset at the mean sequence length runtime



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Amplitude image sequences
Method	Random initialization reset at maximum sequence length

Figure C.15: Relative confusion matrix of the action recognition network trained on amplitude image sequences trained with random initialization and reset at the maximum sequence length at runtime

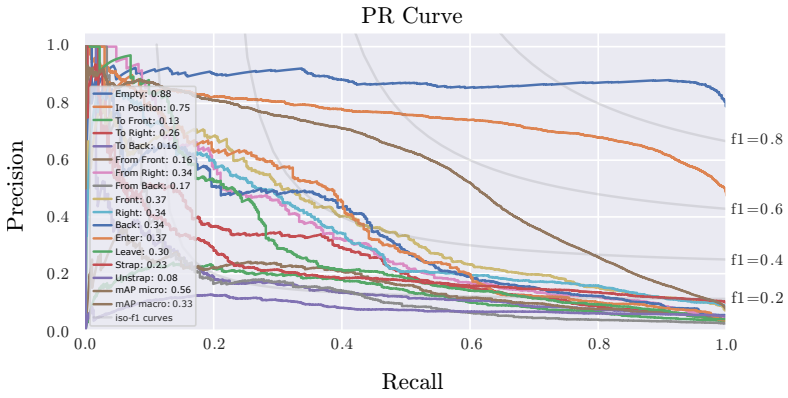
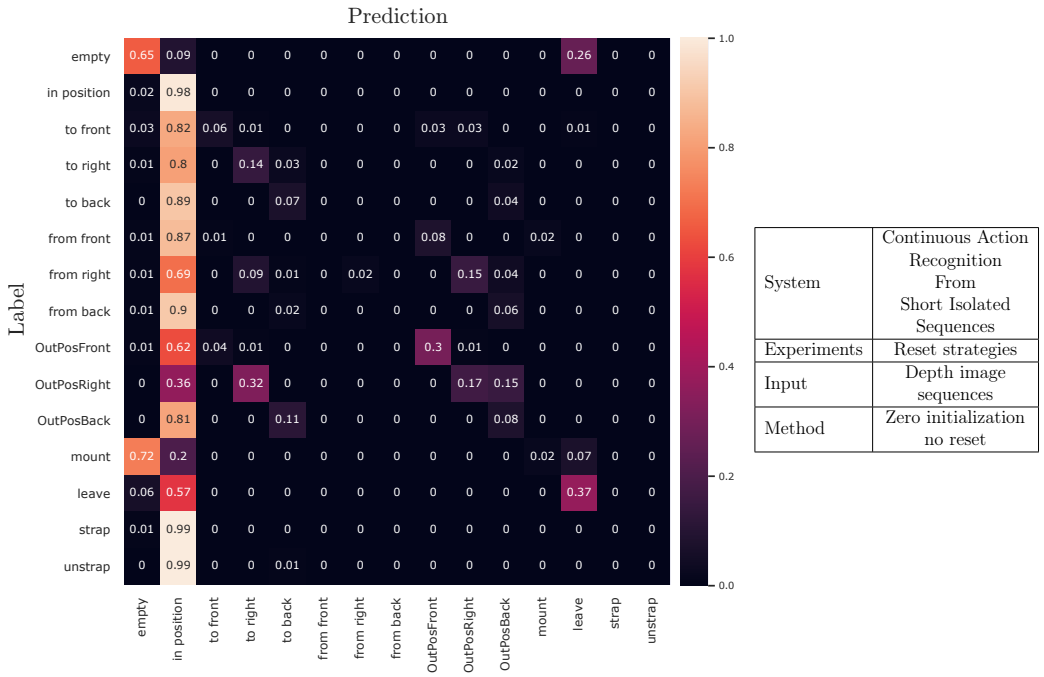


Figure C.16: Precision recall curves of the action recognition network trained on amplitude image sequences trained with random initialization and reset at the maximum sequence length runtime

Depth Input



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Depth image sequences
Method	Zero initialization no reset

Figure C.17: Relative confusion matrix of the action recognition network trained on depth image sequences trained with zero initialization and no reset at runtime

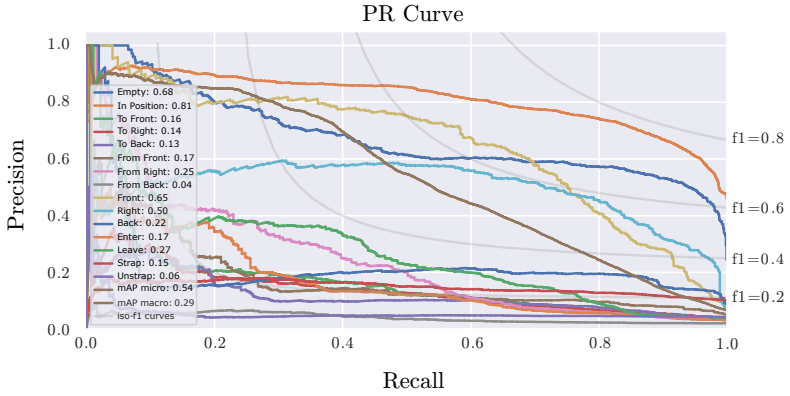
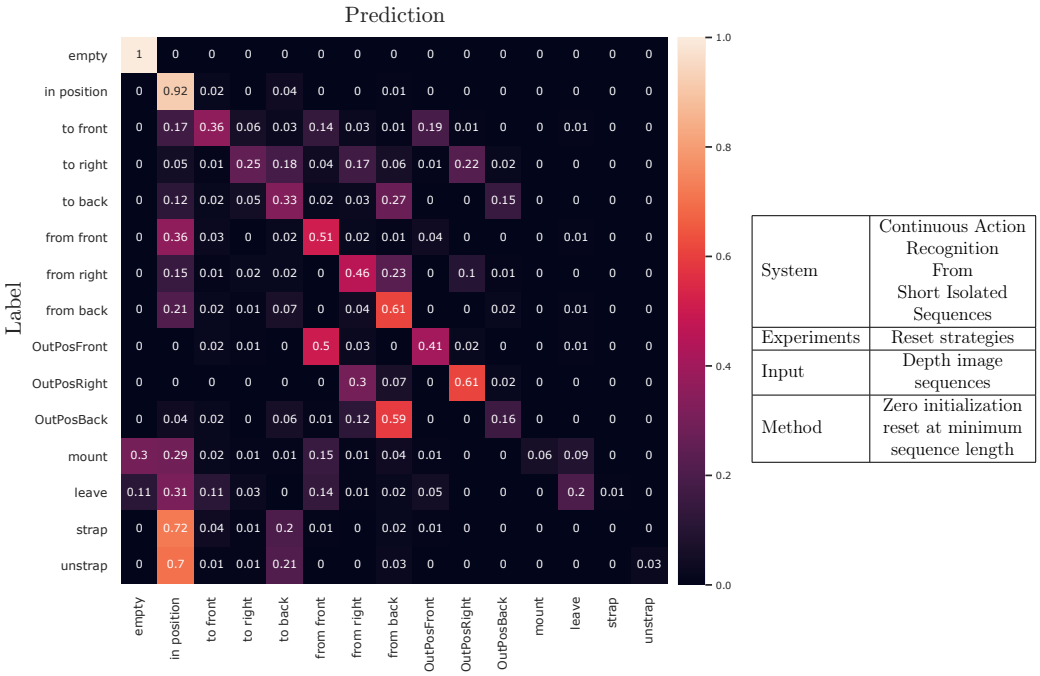


Figure C.18: Precision recall curves of the action recognition network trained on depth image sequences trained with zero initialization and no reset at runtime



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Depth image sequences
Method	Zero initialization reset at minimum sequence length

Figure C.19: Relative confusion matrix of the action recognition network trained on depth image sequences trained with zero initialization and reset at the minimum sequence length at runtime

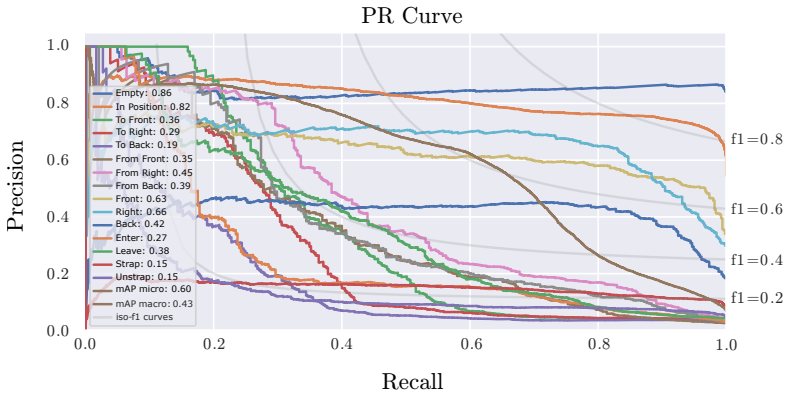
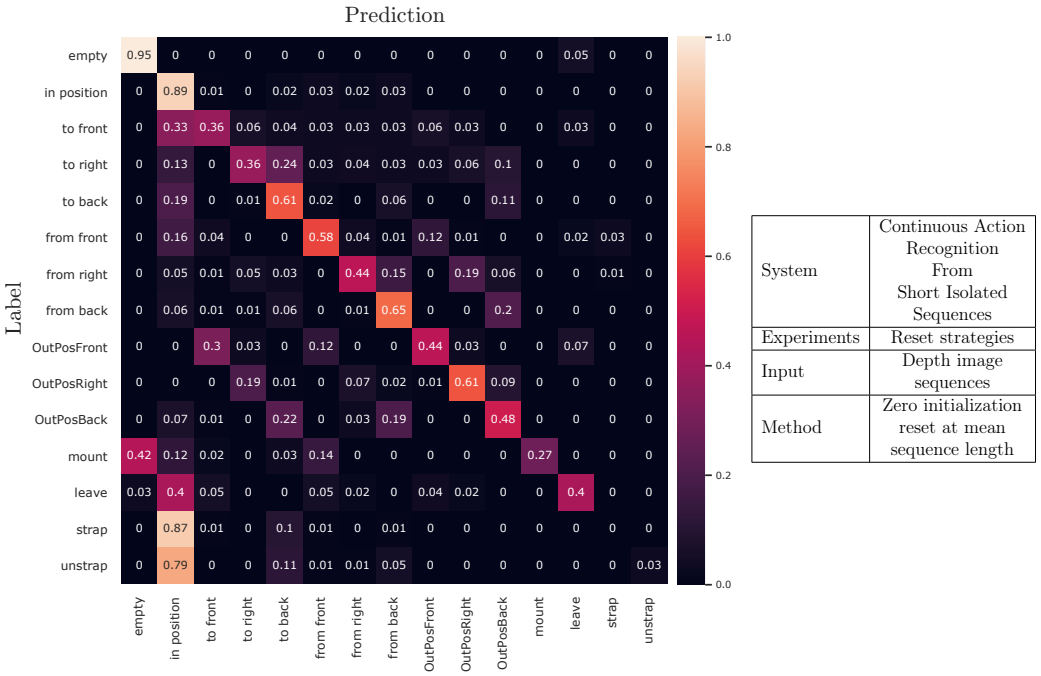


Figure C.20: Precision recall curves of the action recognition network trained on depth image sequences trained with zero initialization and reset at the minimum sequence length runtime



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Depth image sequences
Method	Zero initialization reset at mean sequence length

Figure C.21: Relative confusion matrix of the action recognition network trained on depth image sequences trained with zero initialization and reset at the mean sequence length at runtime

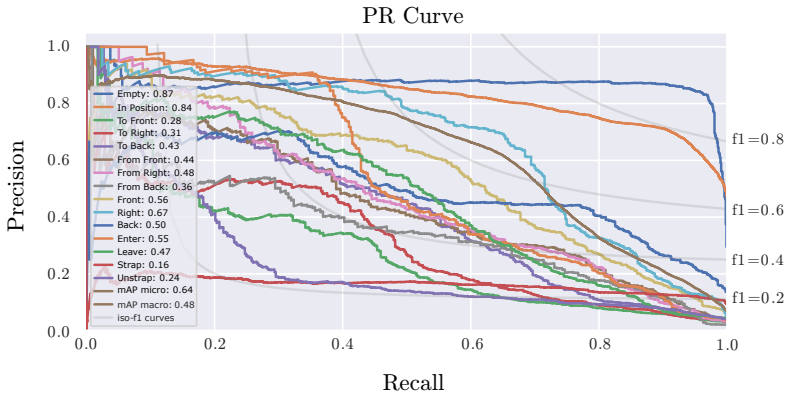
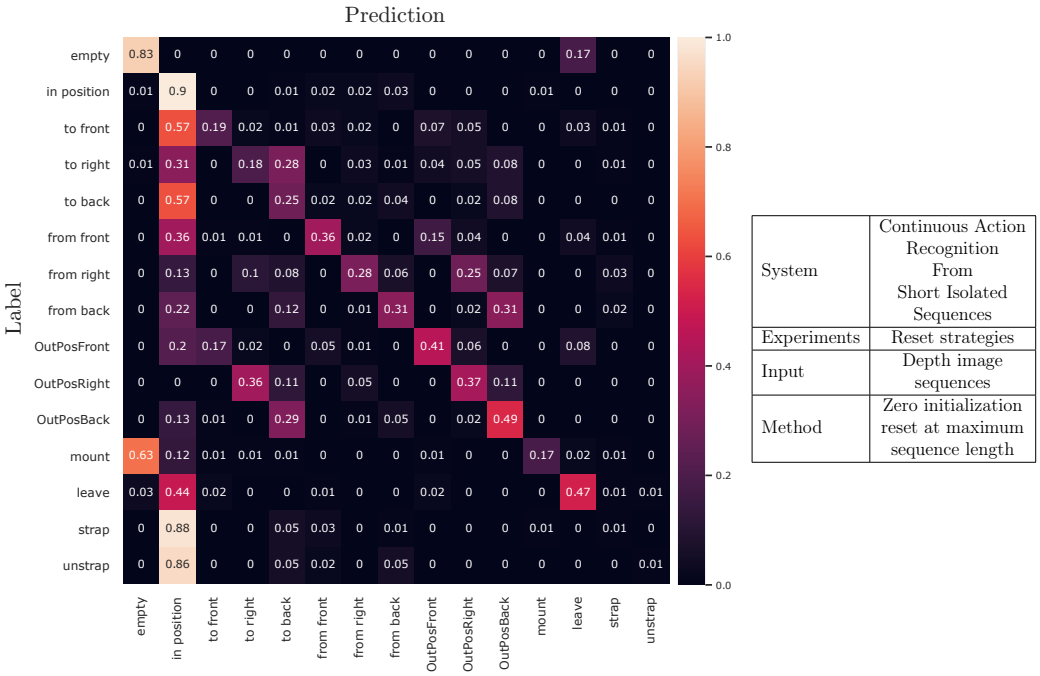


Figure C.22: Precision recall curves of the action recognition network trained on depth image sequences trained with zero initialization and reset at the mean sequence length runtime



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Depth image sequences
Method	Zero initialization reset at maximum sequence length

Figure C.23: Relative confusion matrix of the action recognition network trained on depth image sequences trained with zero initialization and reset at the maximum sequence length at runtime

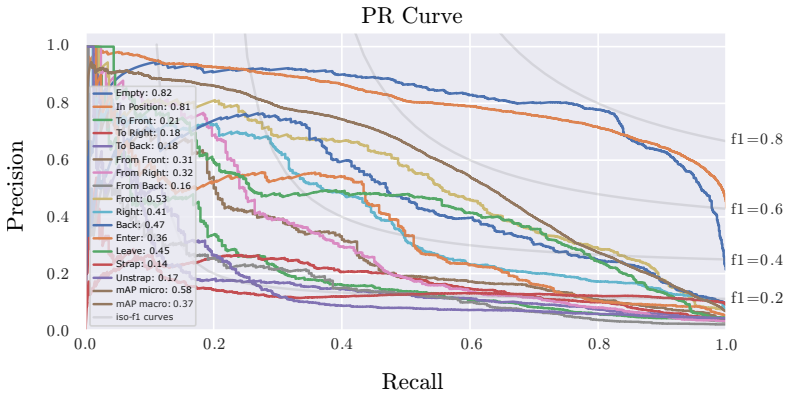
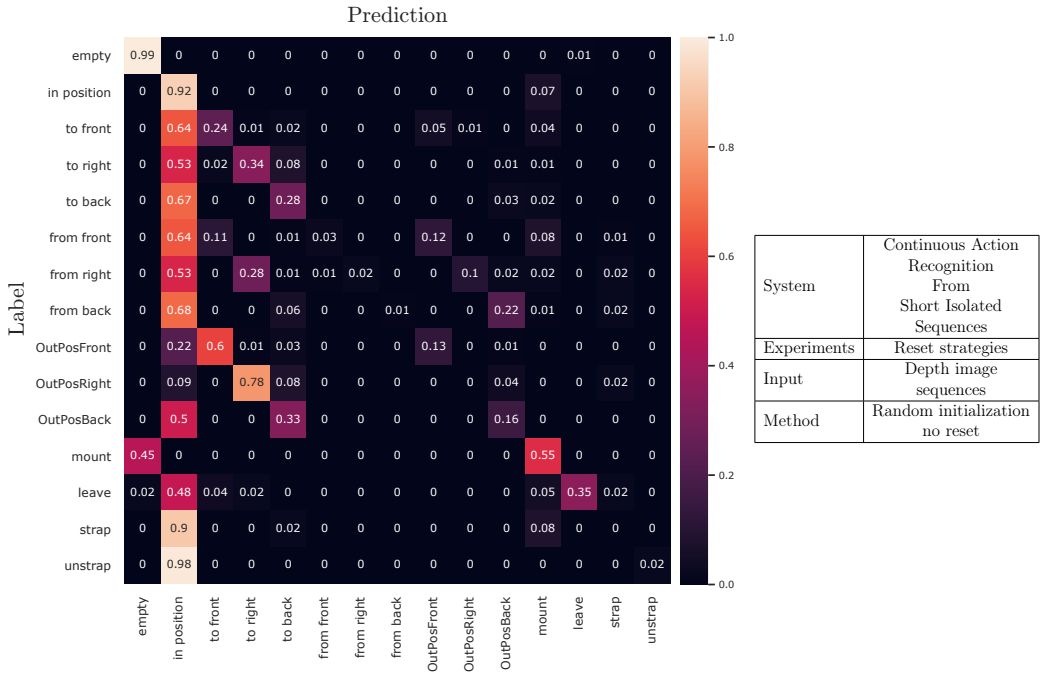


Figure C.24: Precision recall curves of the action recognition network trained on depth image sequences trained with zero initialization and reset at the maximum sequence length runtime



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Depth image sequences
Method	Random initialization no reset

Figure C.25: Relative confusion matrix of the action recognition network trained on depth image sequences trained with random initialization and no reset at runtime

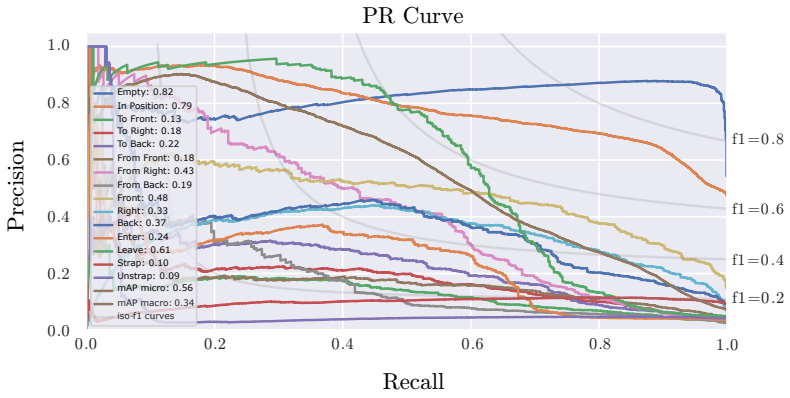


Figure C.26: Precision recall curves of the action recognition network trained on depth image sequences trained with random initialization and no reset at runtime

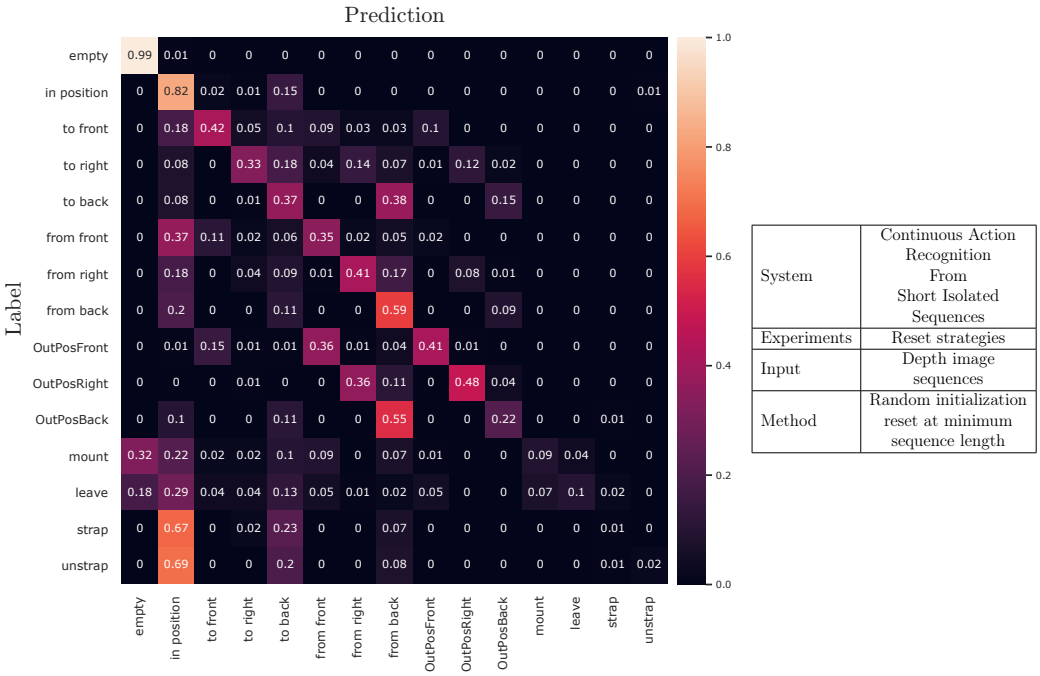


Figure C.27: Relative confusion matrix of the action recognition network trained on depth image sequences trained with random initialization and reset at the minimum sequence length at runtime

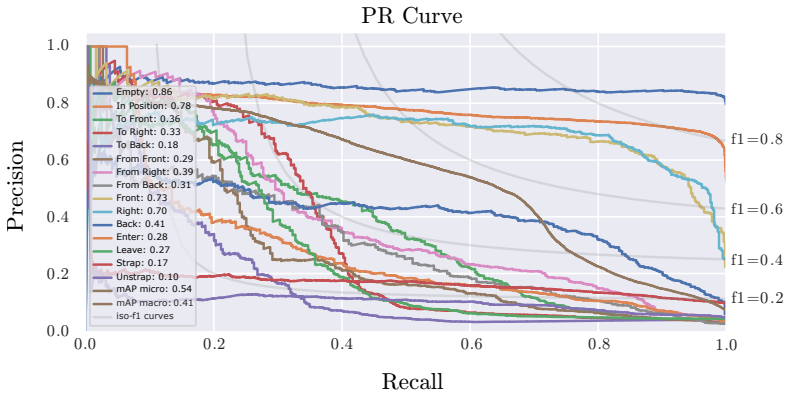
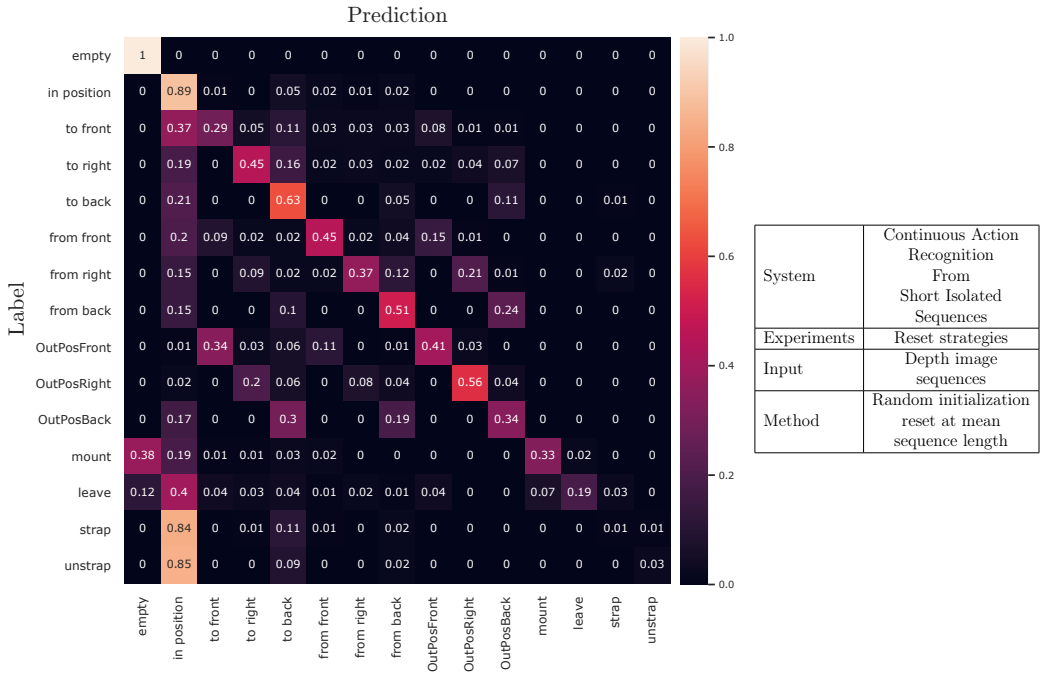


Figure C.28: Precision recall curves of the action recognition network trained on depth image sequences trained with random initialization and reset at the minimum sequence length runtime



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Depth image sequences
Method	Random initialization reset at mean sequence length

Figure C.29: Relative confusion matrix of the action recognition network trained on depth image sequences trained with random initialization and reset at the mean sequence length at runtime

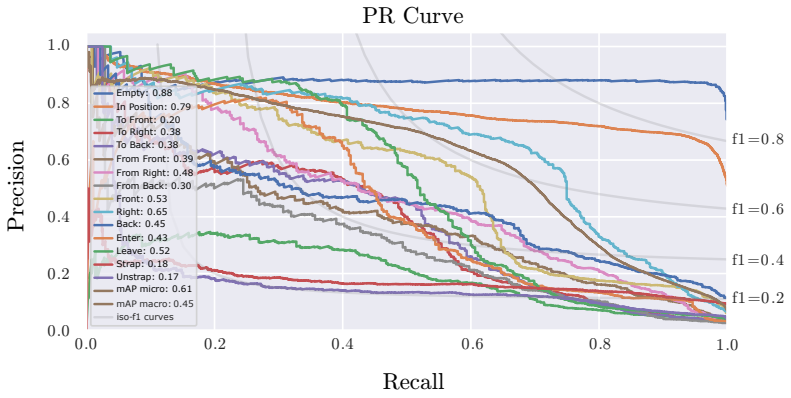
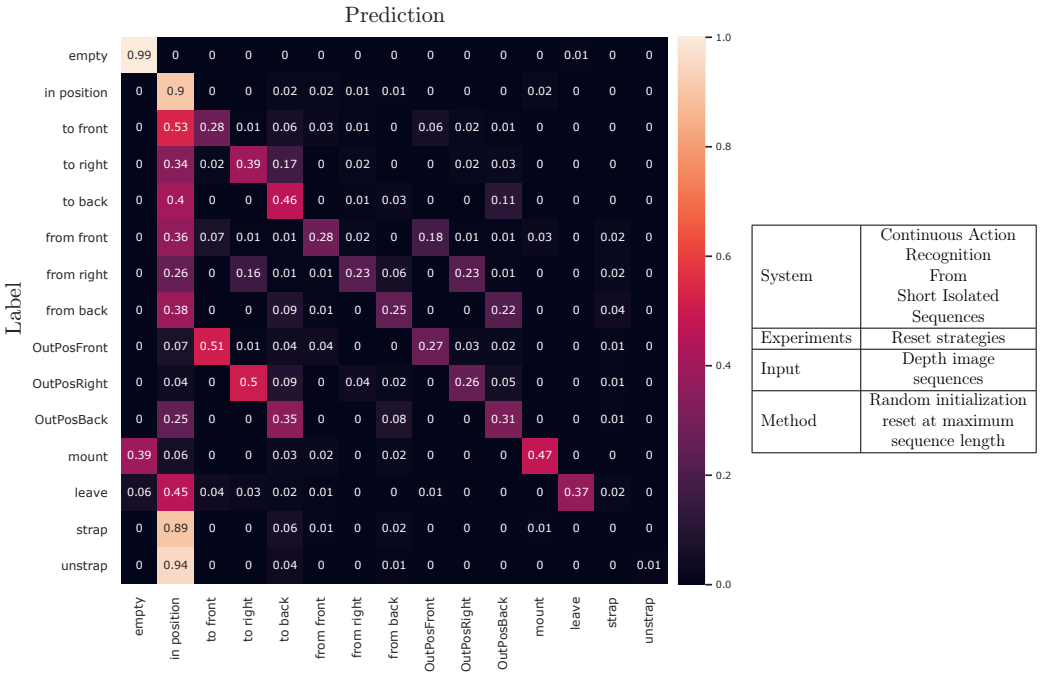


Figure C.30: Precision recall curves of the action recognition network trained on depth image sequences trained with random initialization and reset at the mean sequence length runtime



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Depth image sequences
Method	Random initialization reset at maximum sequence length

Figure C.31: Relative confusion matrix of the action recognition network trained on depth image sequences trained with random initialization and reset at the maximum sequence length at runtime

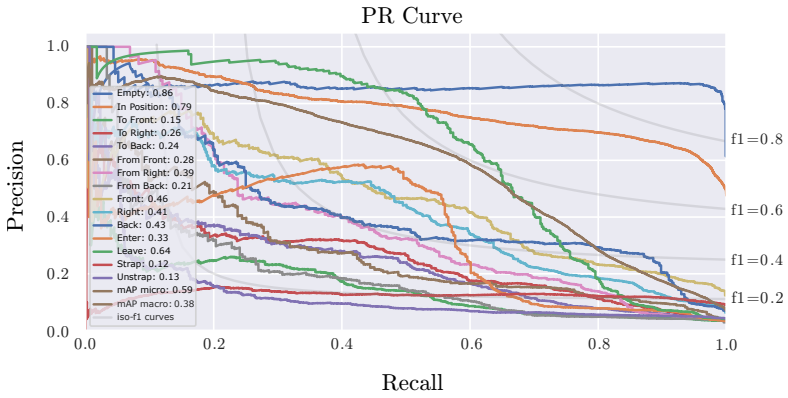
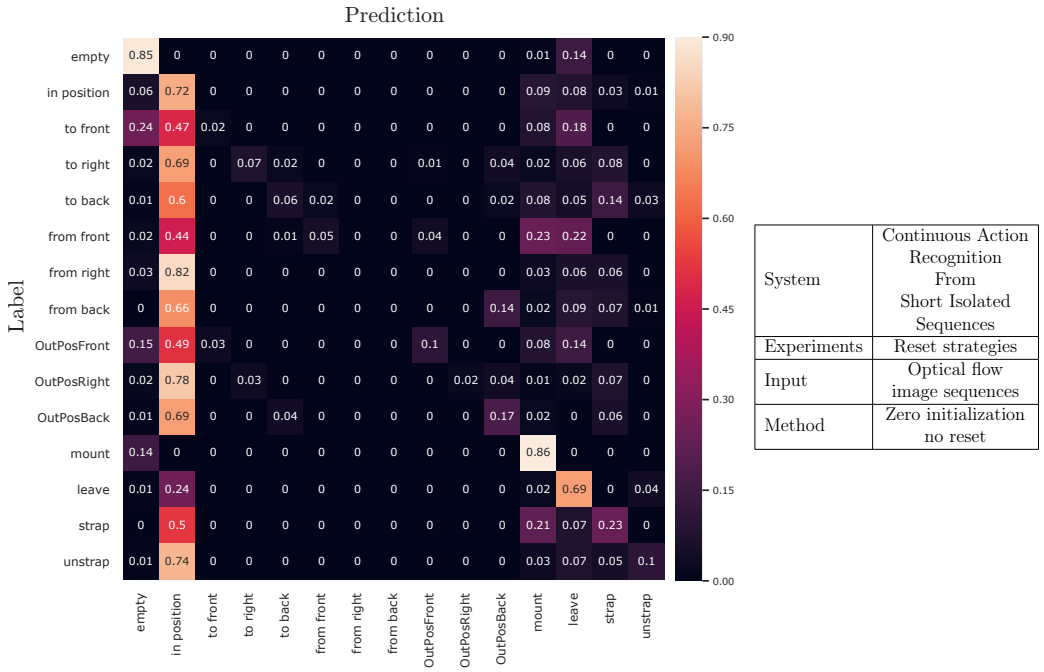


Figure C.32: Precision recall curves of the action recognition network trained on depth image sequences trained with random initialization and reset at the maximum sequence length runtime

Optical Flow Input



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Optical flow image sequences
Method	Zero initialization no reset

Figure C.33: Relative confusion matrix of the action recognition network trained on flow image sequences trained with zero initialization and no reset at runtime

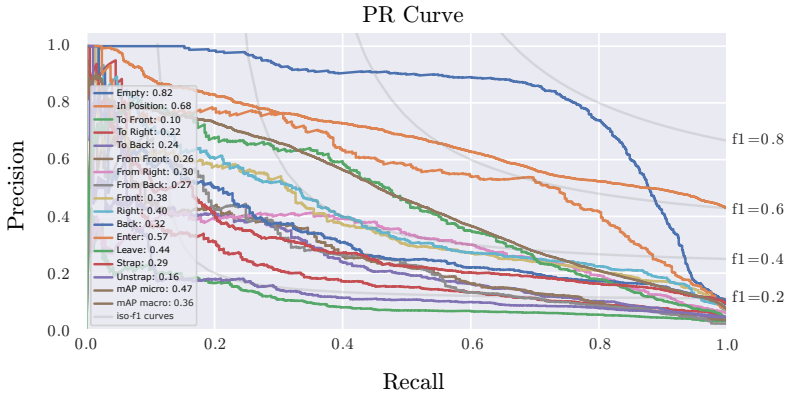
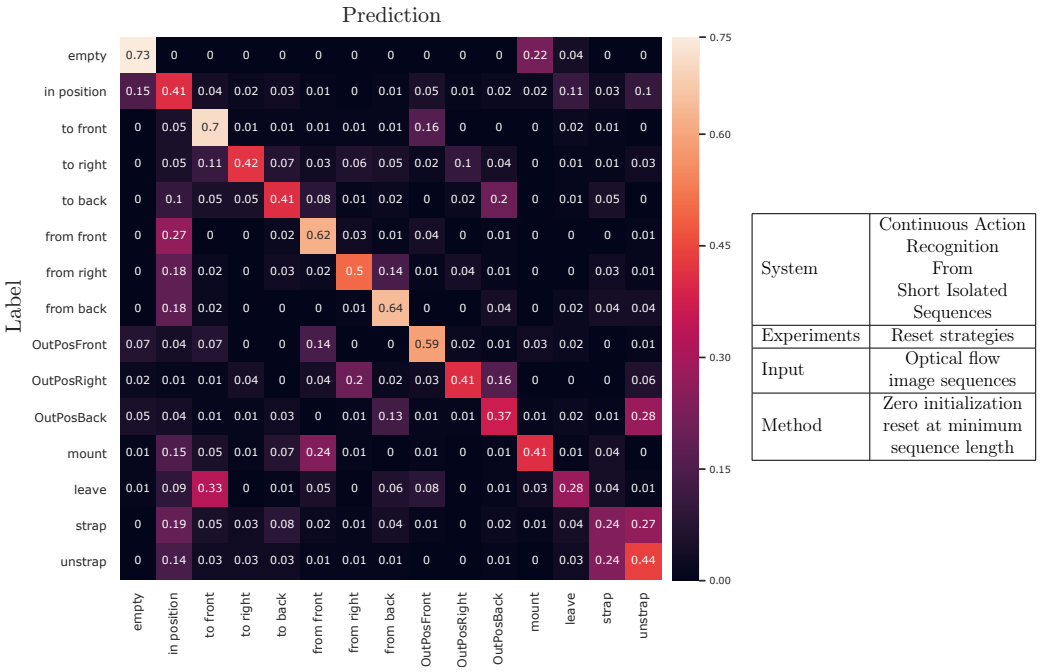


Figure C.34: Precision recall curves of the action recognition network trained on optical flow image sequences trained with zero initialization and no reset at runtime



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Optical flow image sequences
Method	Zero initialization reset at minimum sequence length

Figure C.35: Relative confusion matrix of the action recognition network trained on flow image sequences trained with zero initialization and reset at the minimum sequence length at runtime

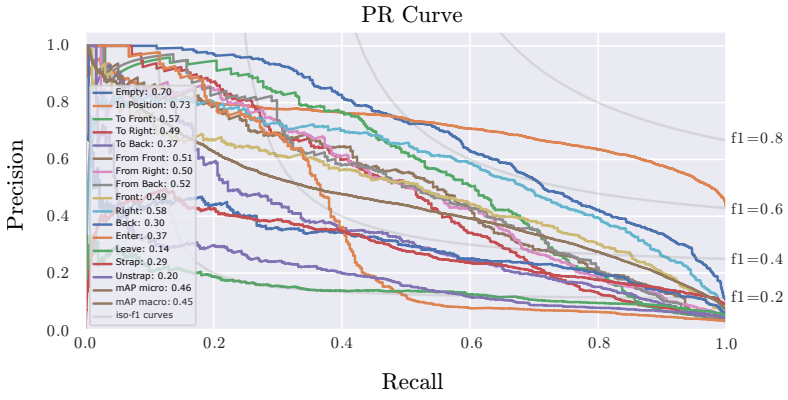
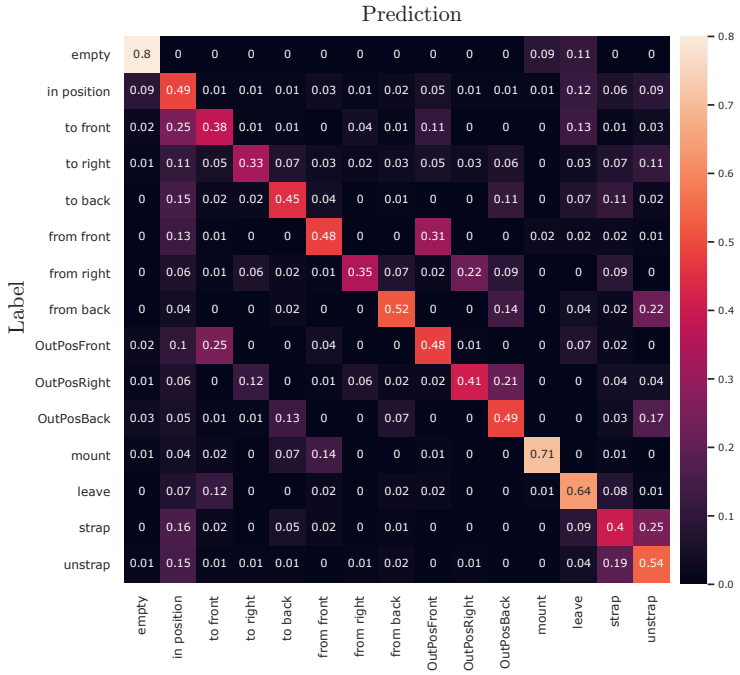


Figure C.36: Precision recall curves of the action recognition network trained on optical flow image sequences trained with zero initialization and reset at the minimum sequence length runtime



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Optical flow image sequences
Method	Zero initialization reset at mean sequence length

Figure C.37: Relative confusion matrix of the action recognition network trained on flow image sequences trained with zero initialization and reset at the mean sequence length at runtime

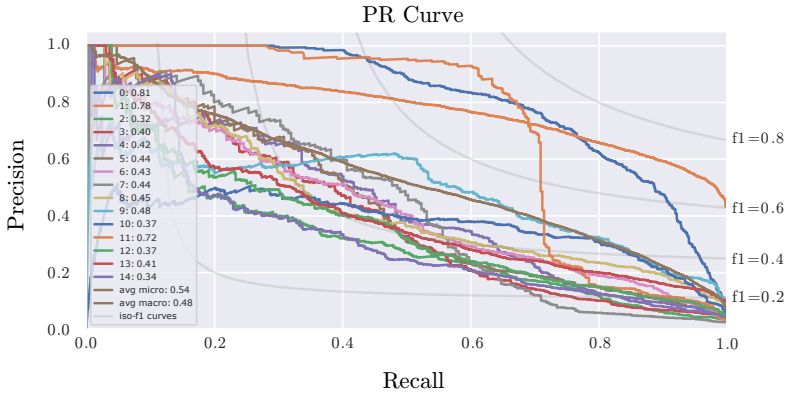
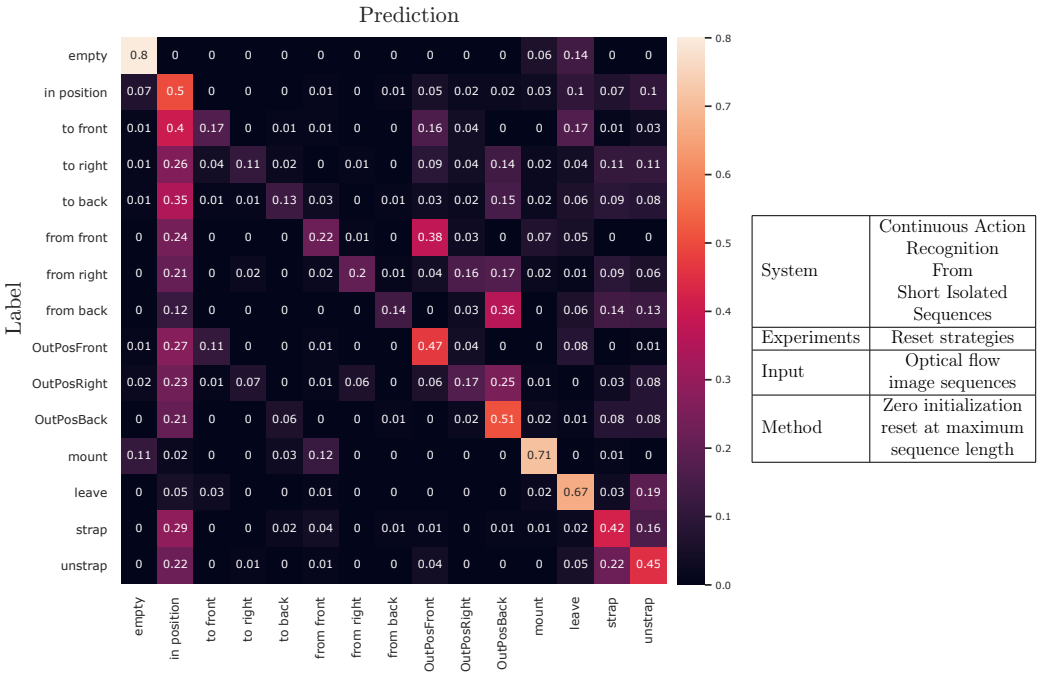


Figure C.38: Precision recall curves of the action recognition network trained on optical flow image sequences trained with zero initialization and reset at the mean sequence length runtime



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Optical flow image sequences
Method	Zero initialization reset at maximum sequence length

Figure C.39: Relative confusion matrix of the action recognition network trained on flow image sequences trained with zero initialization and reset at the maximum sequence length at runtime

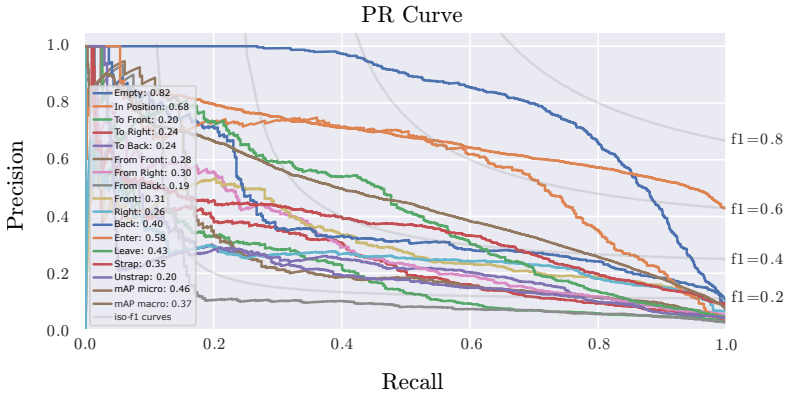


Figure C.40: Precision recall curves of the action recognition network trained on optical flow image sequences trained with zero initialization and reset at the maximum sequence length runtime

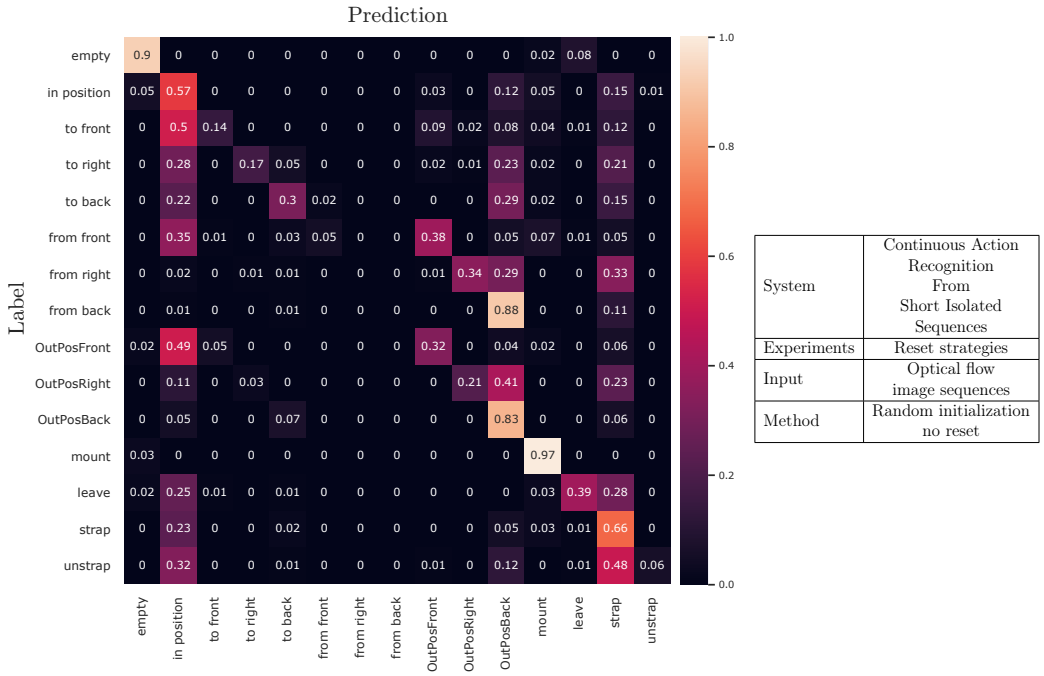


Figure C.41: Relative confusion matrix of the action recognition network trained on flow image sequences trained with random initialization and no reset at runtime

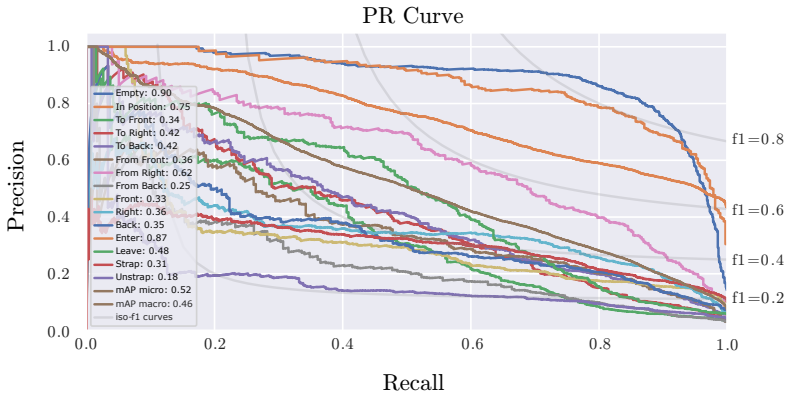
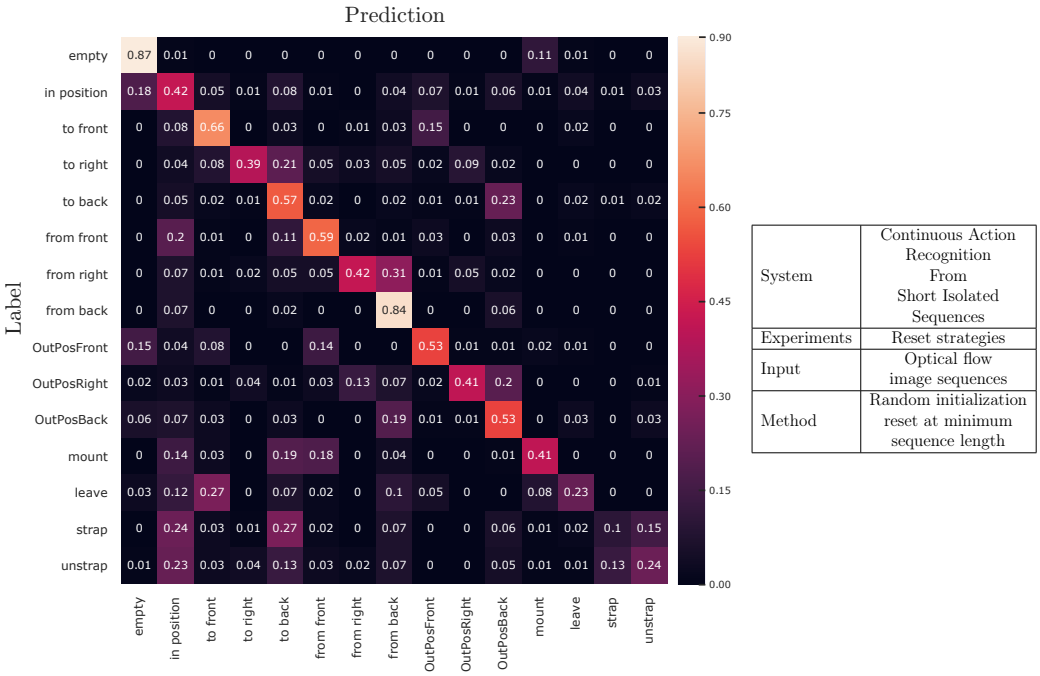


Figure C.42: Precision recall curves of the action recognition network trained on optical flow image sequences trained with random initialization and no reset at runtime



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Optical flow image sequences
Method	Random initialization reset at minimum sequence length

Figure C.43: Relative confusion matrix of the action recognition network trained on flow image sequences trained with random initialization and reset at the minimum sequence length at runtime

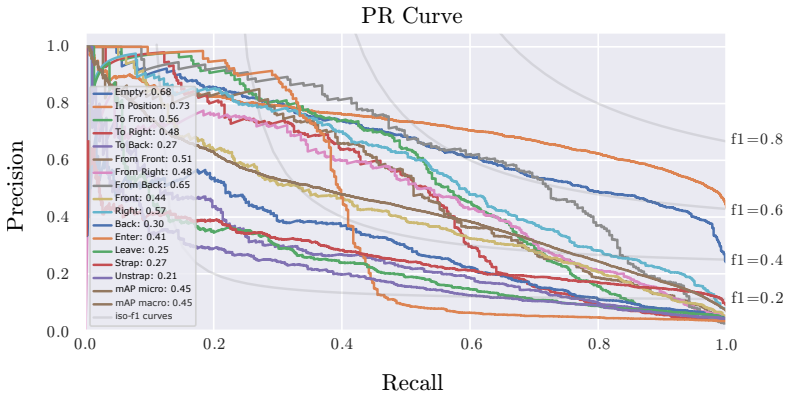
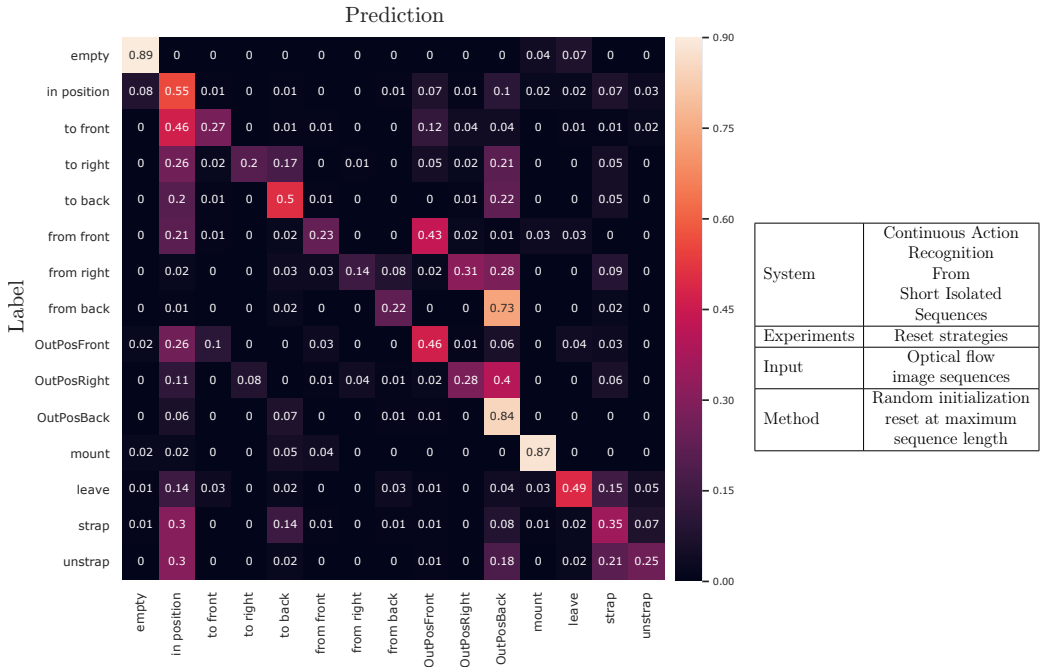


Figure C.44: Precision recall curves of the action recognition network trained on optical flow image sequences trained with random initialization and reset at the minimum sequence length runtime



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Reset strategies
Input	Optical flow image sequences
Method	Random initialization reset at maximum sequence length

Figure C.45: Relative confusion matrix of the action recognition network trained on flow image sequences trained with random initialization and reset at the maximum sequence length at runtime

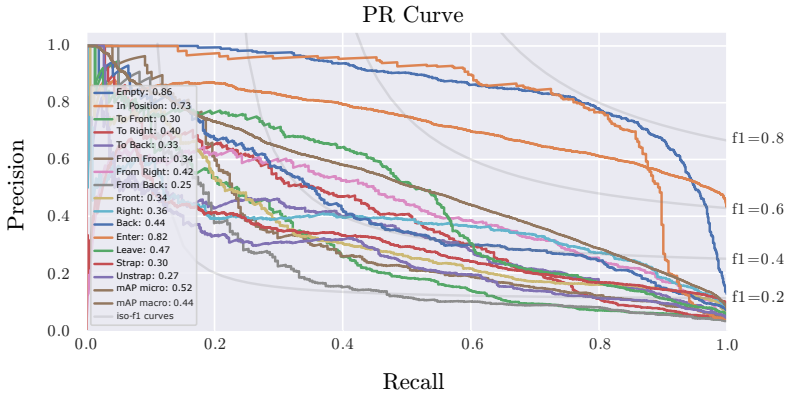


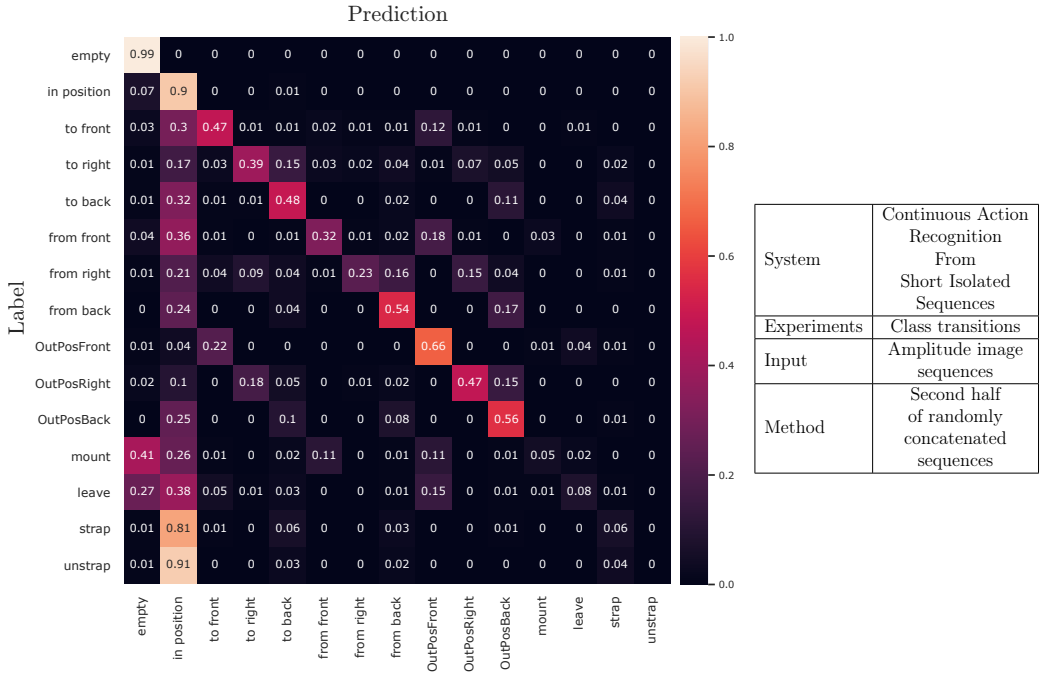
Figure C.46: Precision recall curves of the action recognition network trained on optical flow image sequences trained with random initialization and reset at the maximum sequence length runtime

Class Transition Results

Following, the confusion matrices and precision recall curves of the remaining systems described in chapter 6.3.1 are shown and belong to the systems evaluated in table 6.3. The networks were trained with different methods of concatenating short action sequences described in chapter 6.3 to simulate the training on continuous data streams.

Random Transition

Amplitude Input



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Class transitions
Input	Amplitude image sequences
Method	Second half of randomly concatenated sequences

Figure C.47: Relative confusion matrix of the action recognition network trained on the second half of randomly concatenated amplitude image sequences

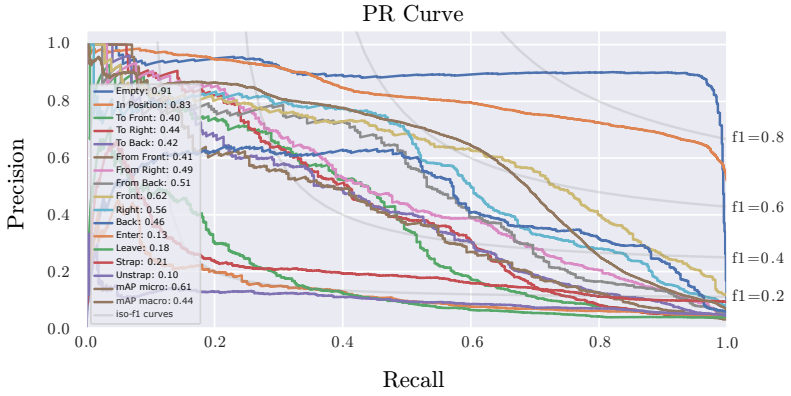
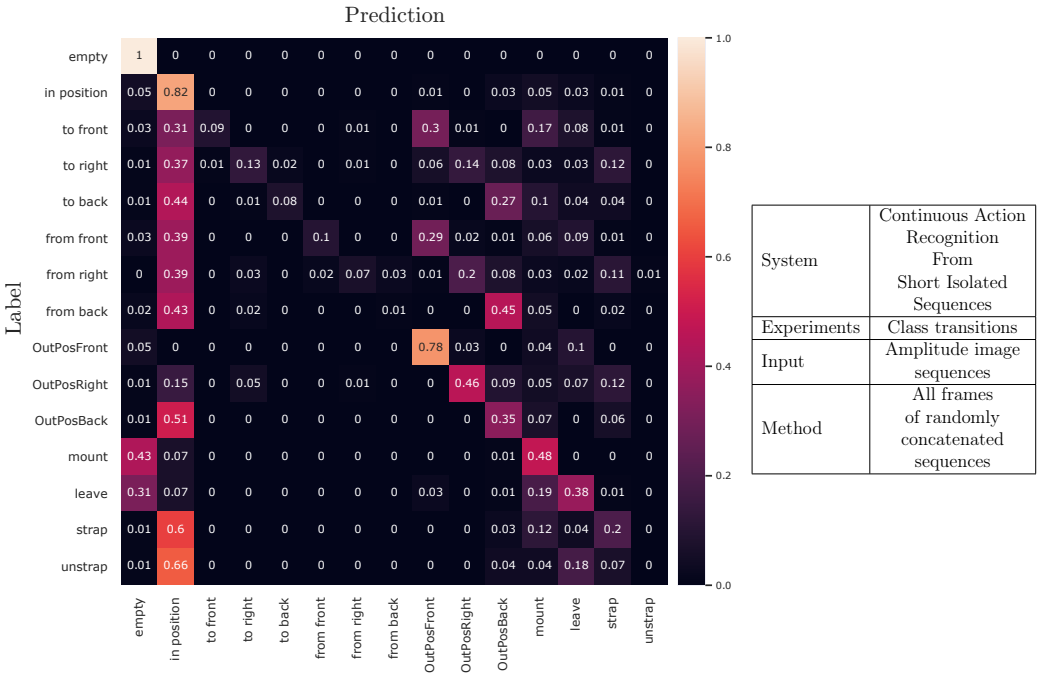


Figure C.48: Precision recall curves of the action recognition network trained on the second half of randomly concatenated amplitude image sequences



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Class transitions
Input	Amplitude image sequences
Method	All frames of randomly concatenated sequences

Figure C.49: Relative confusion matrix of the action recognition network trained on all frames of randomly concatenated amplitude image sequences

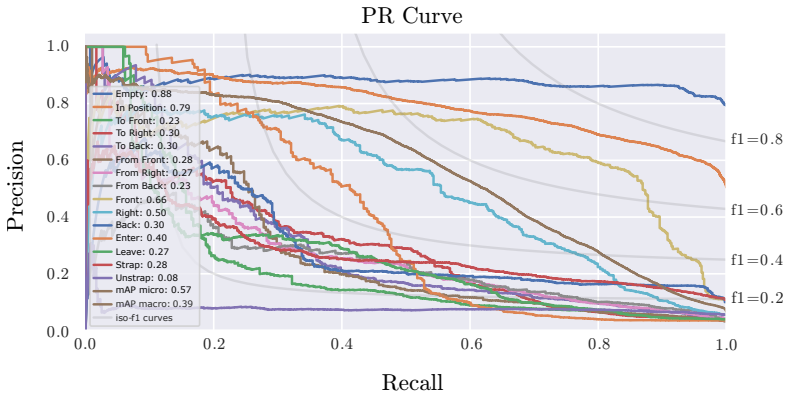
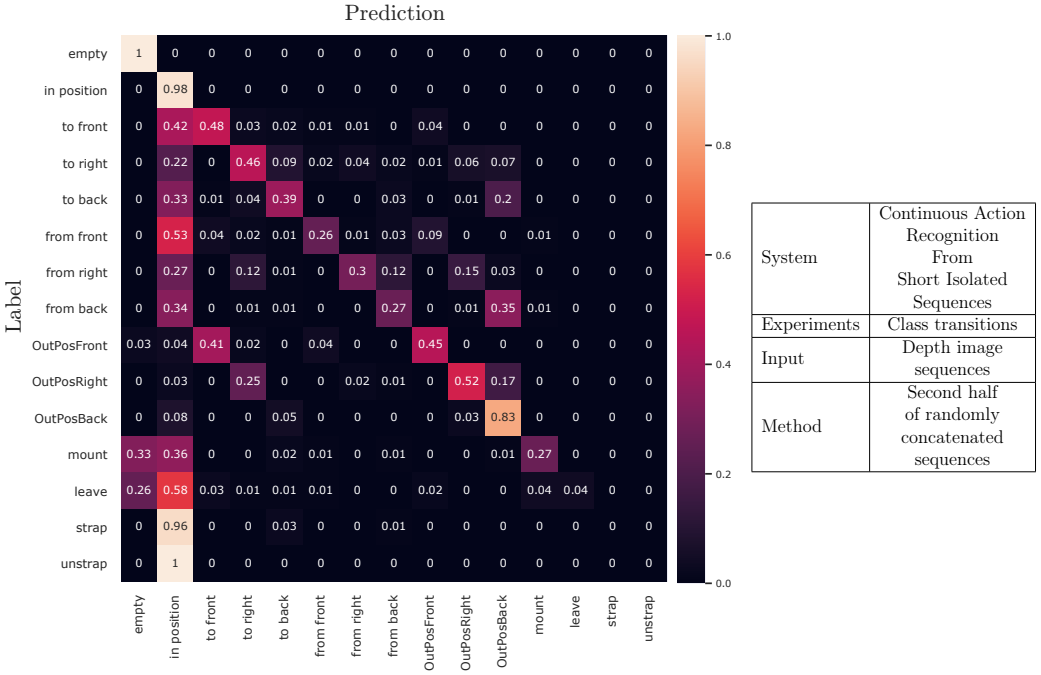


Figure C.50: Precision recall curves of the action recognition network trained on all frames of randomly concatenated amplitude image sequences

Depth Input



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Class transitions
Input	Depth image sequences
Method	Second half of randomly concatenated sequences

Figure C.51: Relative confusion matrix of the action recognition network trained on the second half of randomly concatenated depth image sequences

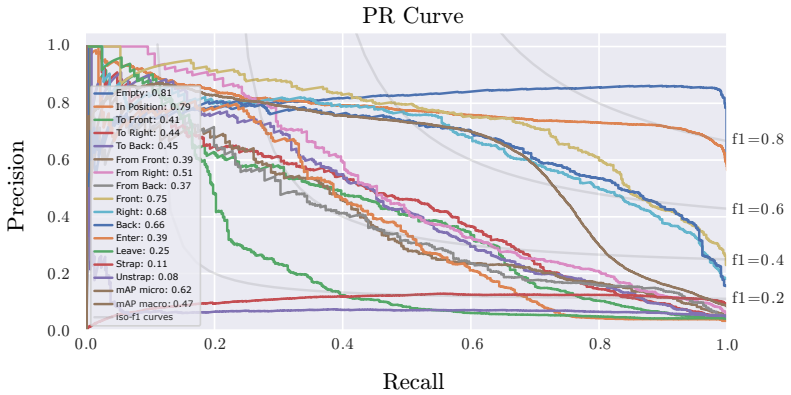
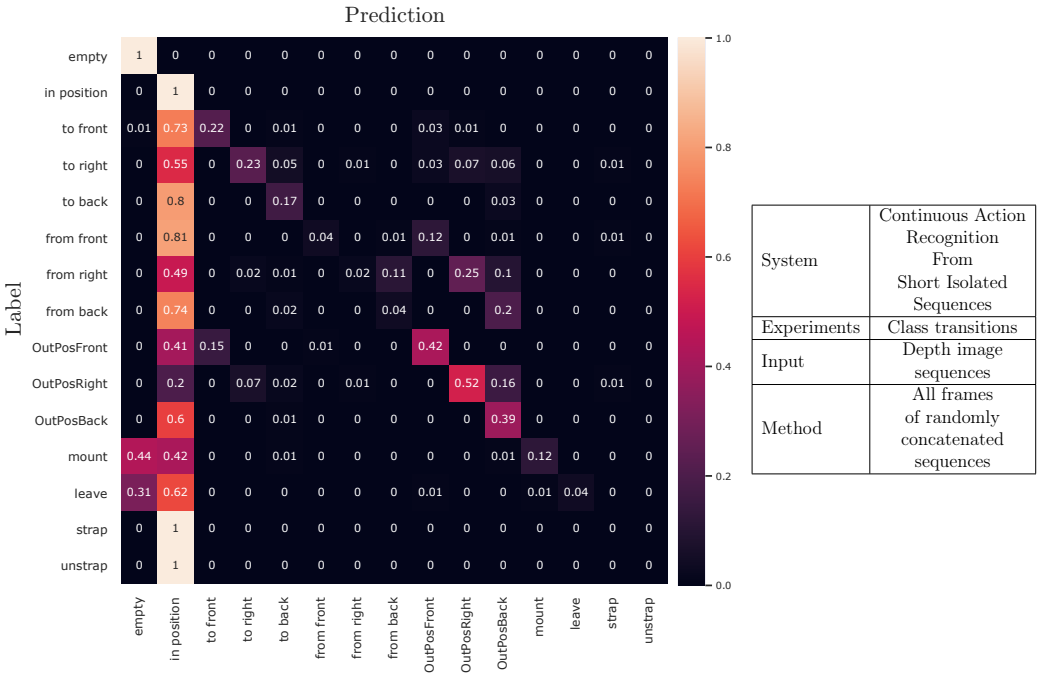


Figure C.52: Precision recall curves of the action recognition network trained on the second half of randomly concatenated depth image sequences



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Class transitions
Input	Depth image sequences
Method	All frames of randomly concatenated sequences

Figure C.53: Relative confusion matrix of the action recognition network trained on all frames of randomly concatenated depth image sequences

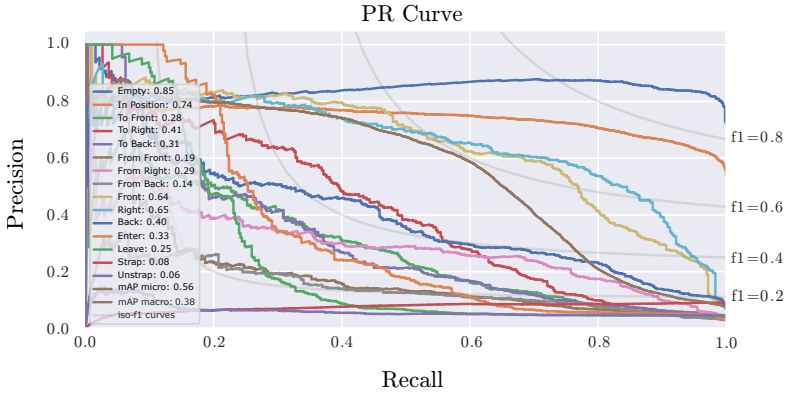
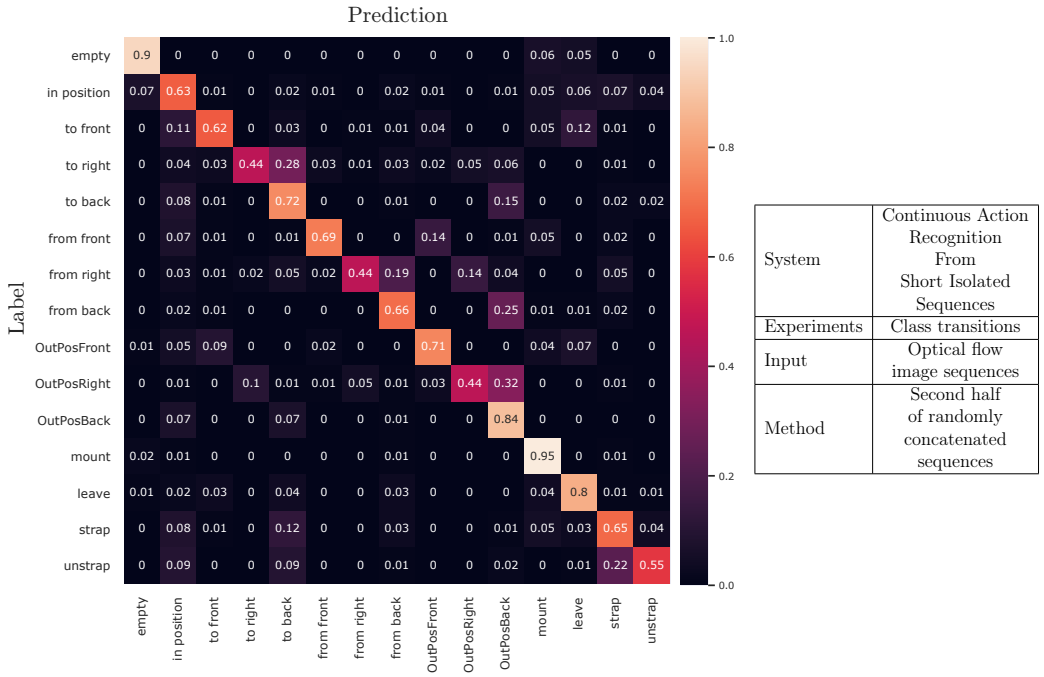


Figure C.54: Precision recall curves of the action recognition network trained on all frames of randomly concatenated depth image sequences

Optical Flow Input



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Class transitions
Input	Optical flow image sequences
Method	Second half of randomly concatenated sequences

Figure C.55: Relative confusion matrix of the action recognition network trained on the second half of randomly concatenated optical flow image sequences

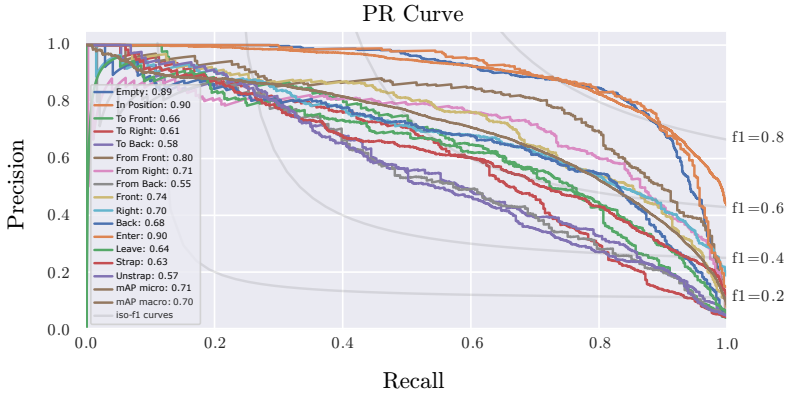
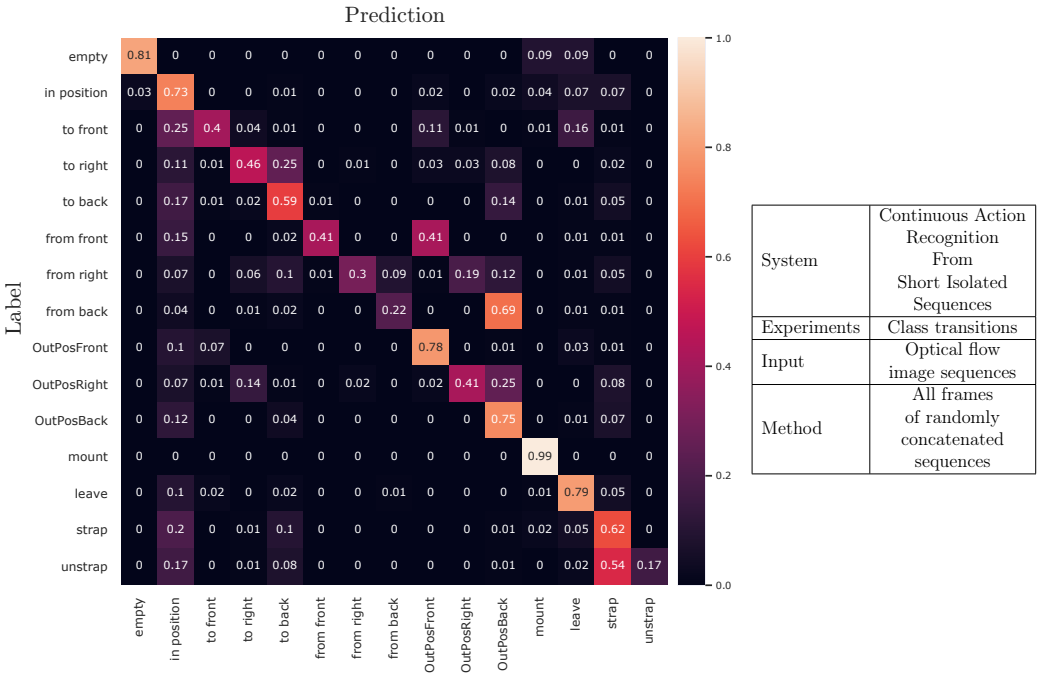


Figure C.56: Precision recall curves of the action recognition network trained on the second half of randomly concatenated optical flow image sequences



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Class transitions
Input	Optical flow image sequences
Method	All frames of randomly concatenated sequences

Figure C.57: Relative confusion matrix of the action recognition network trained on all frames of randomly concatenated optical flow image sequences

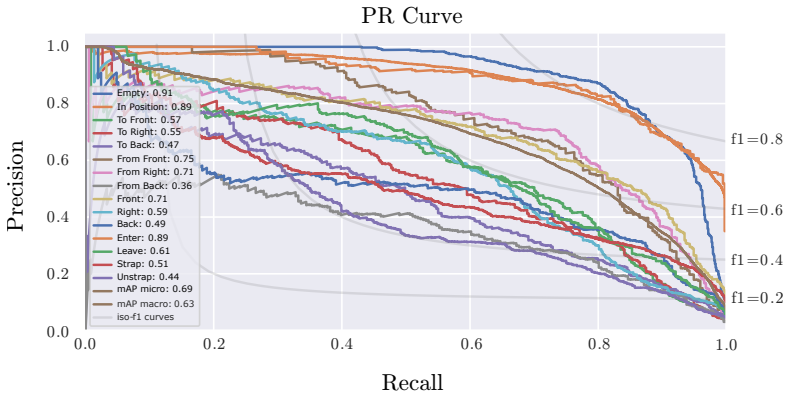
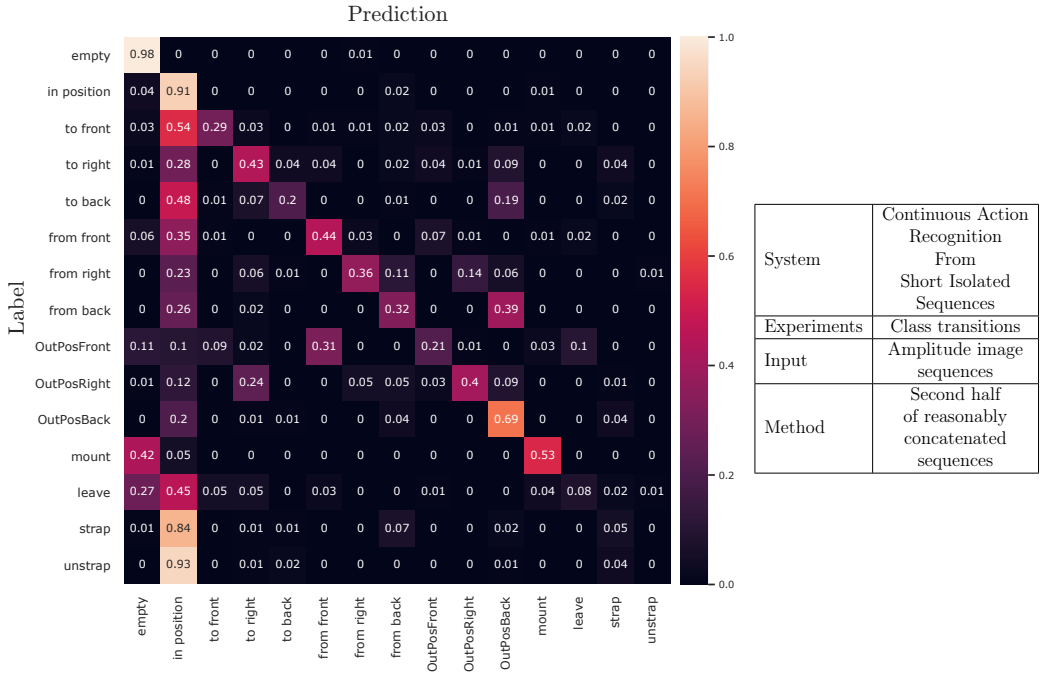


Figure C.58: Precision recall curves of the action recognition network trained on all frames of randomly concatenated optical flow image sequences

Reasonable Transition

Amplitude Input



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Class transitions
Input	Amplitude image sequences
Method	Second half of reasonably concatenated sequences

Figure C.59: Relative confusion matrix of the action recognition network trained on the second half of reasonably concatenated amplitude image sequences

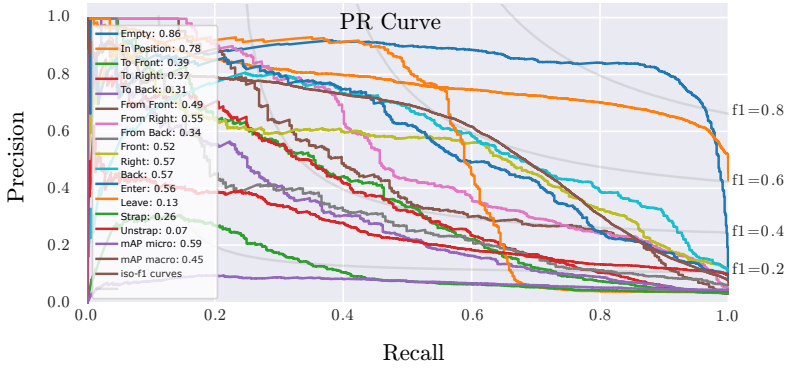
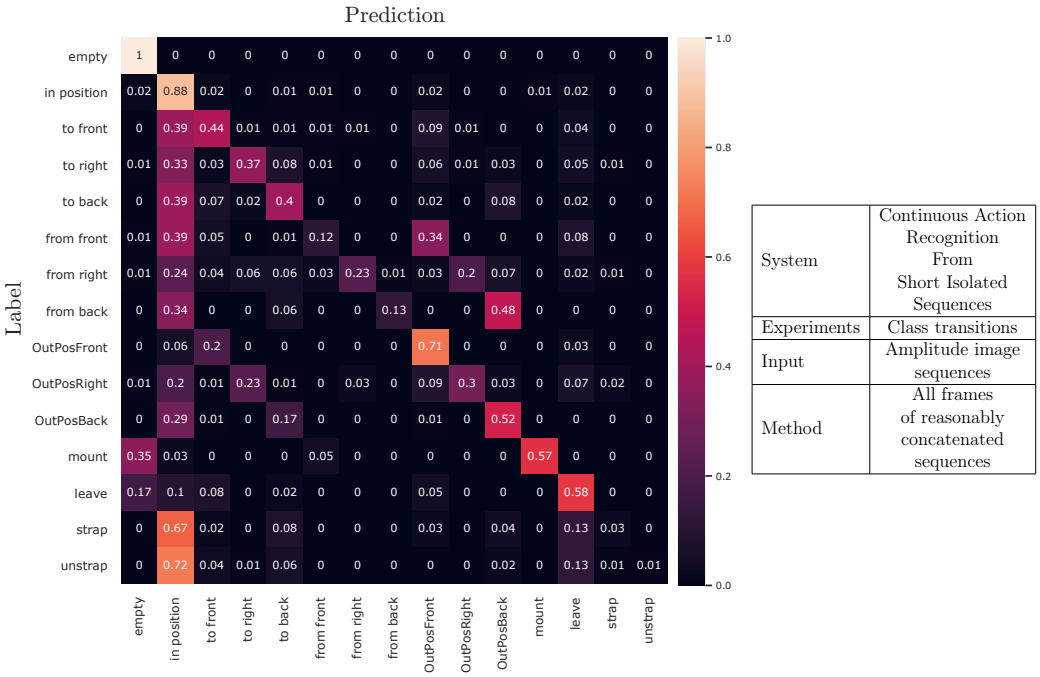


Figure C.60: Precision recall curves of the action recognition network trained on the second half of reasonably concatenated amplitude image sequences



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Class transitions
Input	Amplitude image sequences
Method	All frames of reasonably concatenated sequences

Figure C.61: Relative confusion matrix of the action recognition network trained on all frames of reasonably concatenated amplitude image sequences

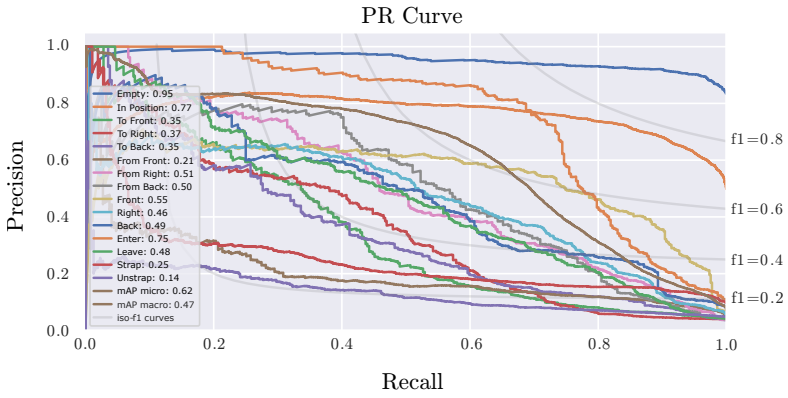
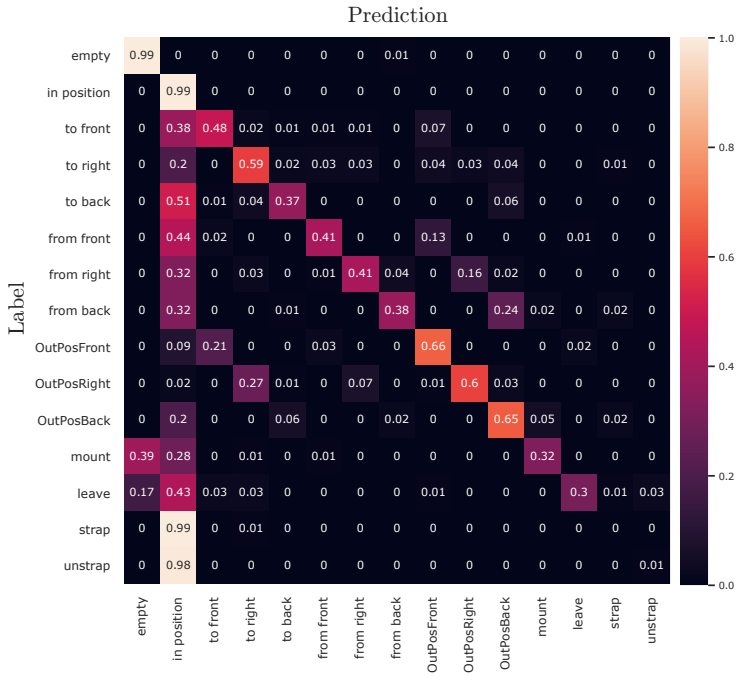


Figure C.62: Precision recall curves of the action recognition network trained on all frames of reasonably concatenated amplitude image sequences

Depth Input



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Class transitions
Input	Depth image sequences
Method	Second half of reasonably concatenated sequences

Figure C.63: Relative confusion matrix of the action recognition network trained on the second half of reasonably concatenated depth image sequences

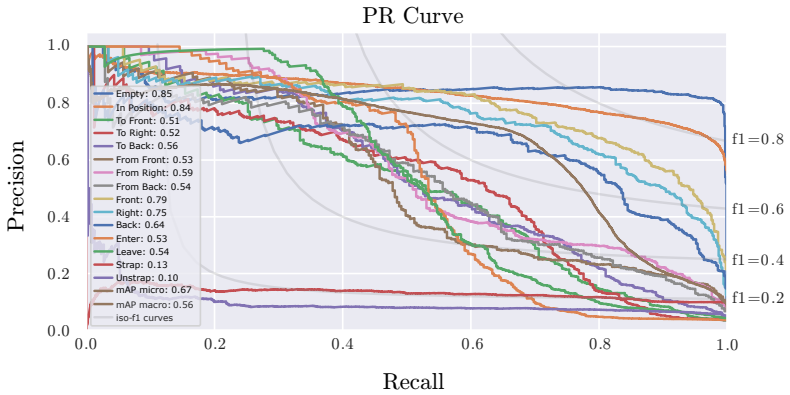
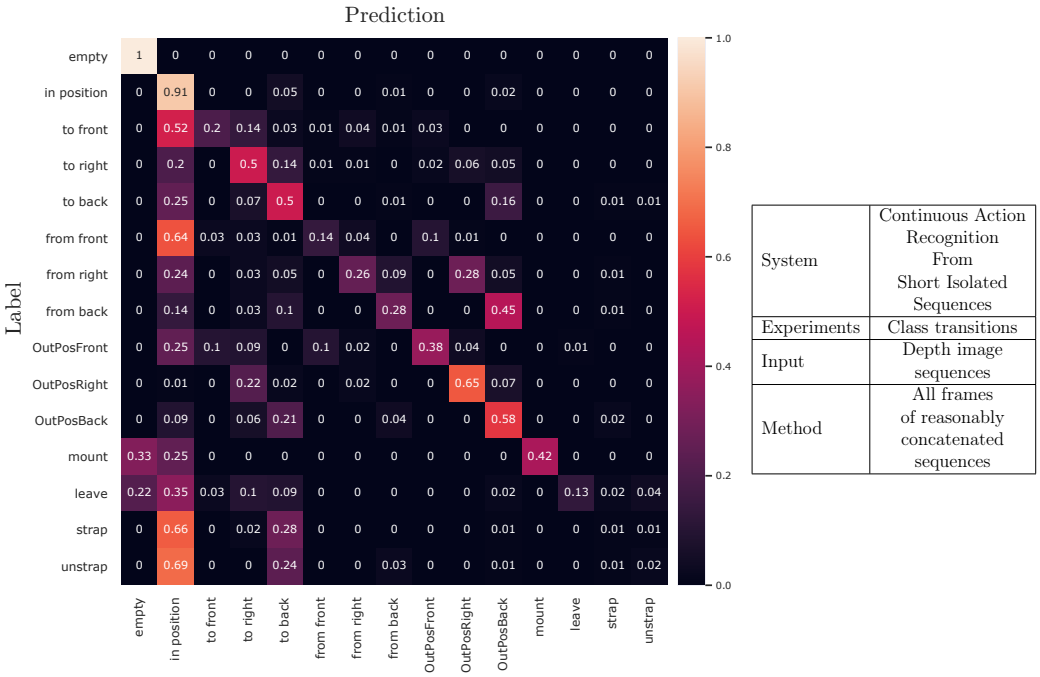


Figure C.64: Precision recall curves of the action recognition network trained on the second half of reasonably concatenated depth image sequences



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Class transitions
Input	Depth image sequences
Method	All frames of reasonably concatenated sequences

Figure C.65: Relative confusion matrix of the action recognition network trained on all frames of reasonably concatenated depth image sequences

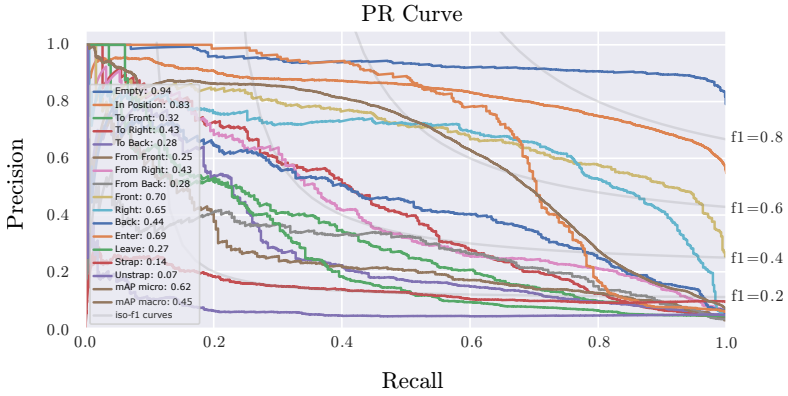
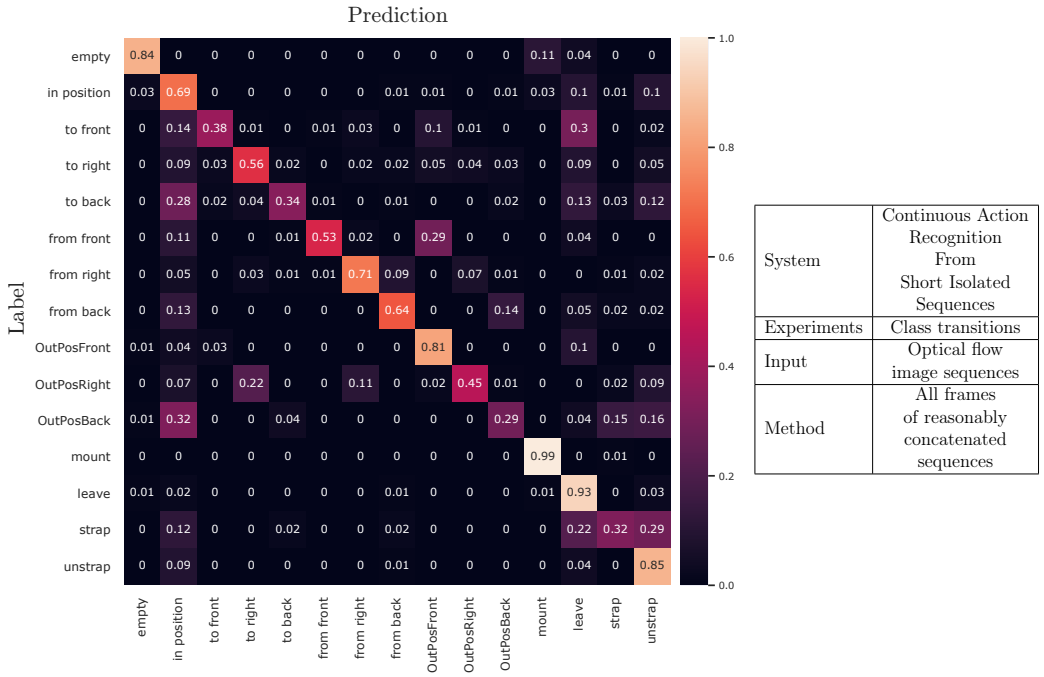


Figure C.66: Precision recall curves of the action recognition network trained on all frames of reasonably concatenated depth image sequences

Optical Flow Input



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	Class transitions
Input	Optical flow image sequences
Method	All frames of reasonably concatenated sequences

Figure C.67: Relative confusion matrix of the action recognition network trained on all frames of reasonably concatenated optical flow image sequences

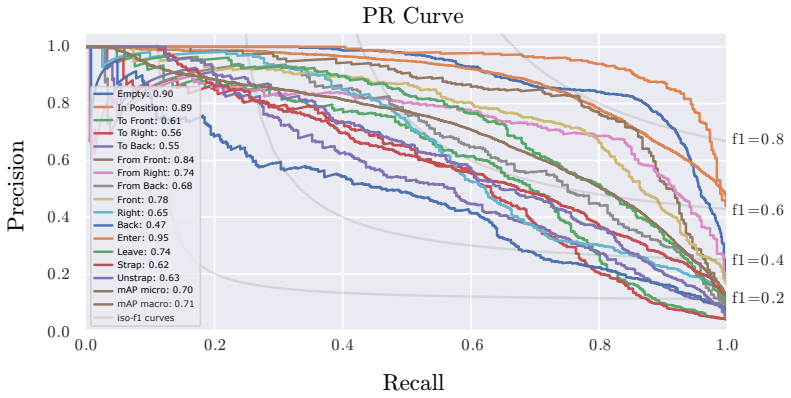
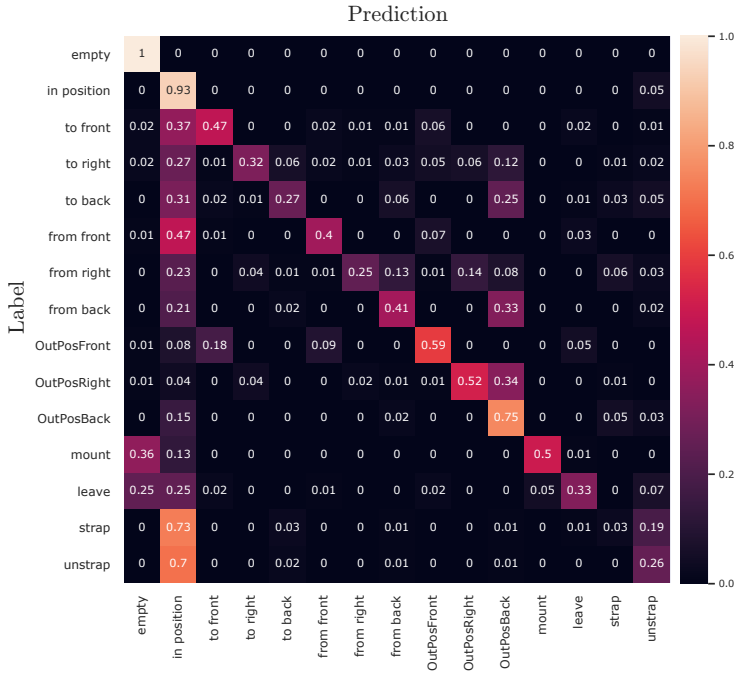


Figure C.68: Precision recall curves of the action recognition network trained on all frames of reasonably concatenated optical flow image sequences

State Handling Results

Following, the confusion matrices and precision recall curves of the remaining systems described in chapter 6.4.4 are shown and belong to the systems evaluated in table 6.5. The networks were trained with combinations of the state handling method, described in chapter 6.4 and the sequence concatenation approaches described in chapter 6.3.

Amplitude Input



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	State handling
Input	Amplitude image sequences
Method	Reasonable state handling

Figure C.69: Relative confusion matrix of the action recognition network trained on amplitude image sequences with reasonable state handling

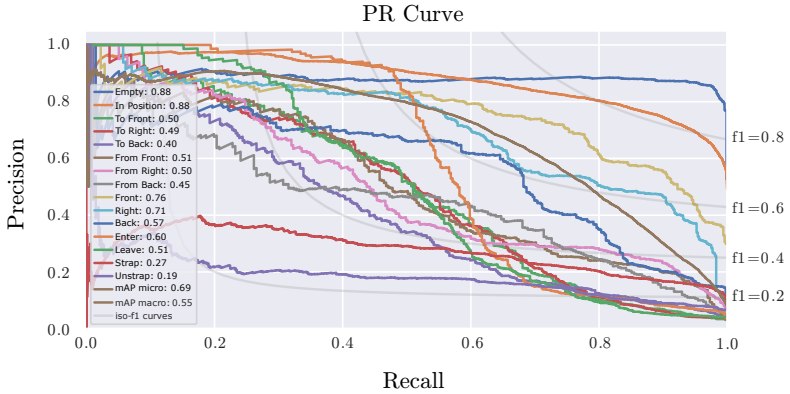
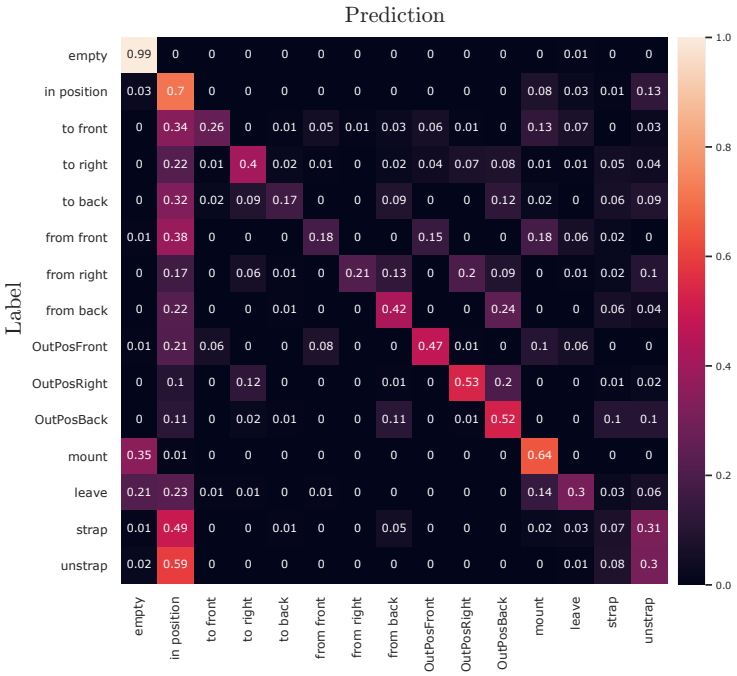


Figure C.70: Precision recall curves of the action recognition network trained on amplitude image sequences with reasonable state handling



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	State handling
Input	Amplitude image sequences
Method	Second half of reasonably concatenated sequences with reasonable state handling

Figure C.71: Relative confusion matrix of the action recognition network trained on the second half of reasonably concatenated amplitude image sequences with reasonable state handling

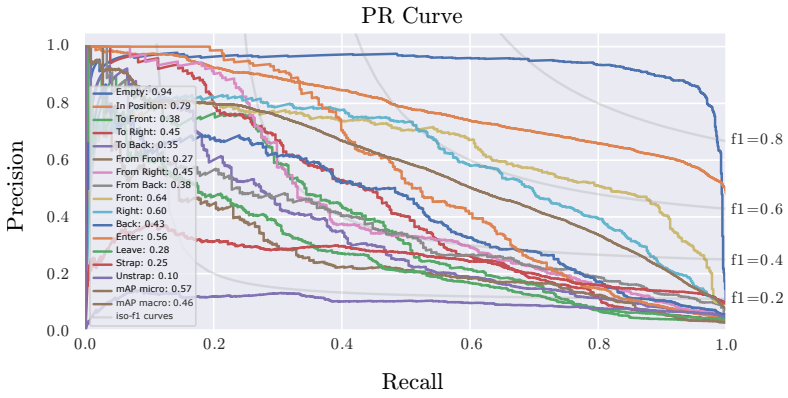
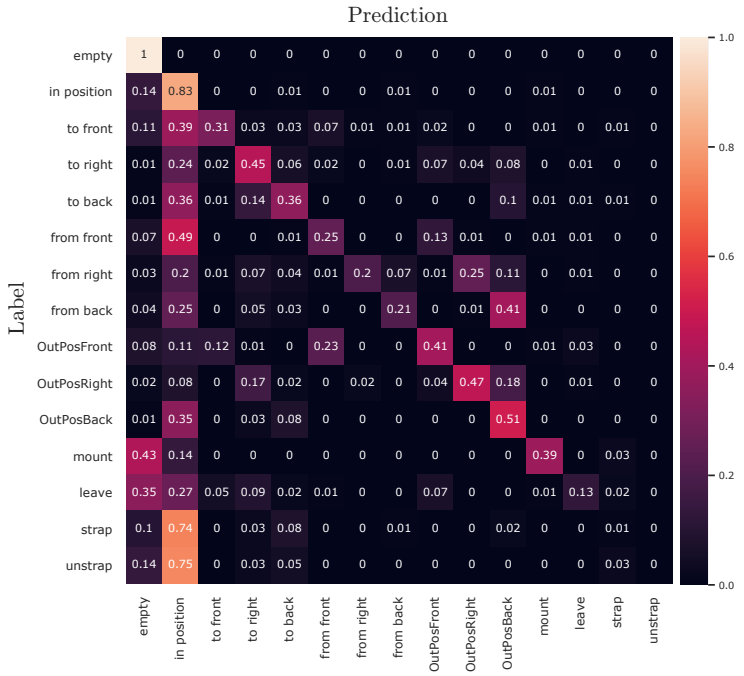


Figure C.72: Precision recall curves of the action recognition network trained on the second half of reasonably concatenated amplitude image sequences with reasonable state handling



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	State handling
Input	Amplitude image sequences
Method	All frames of reasonably concatenated sequences with reasonable state handling

Figure C.73: Relative confusion matrix of the action recognition network trained on all frames of reasonably concatenated amplitude image sequences with reasonable state handling

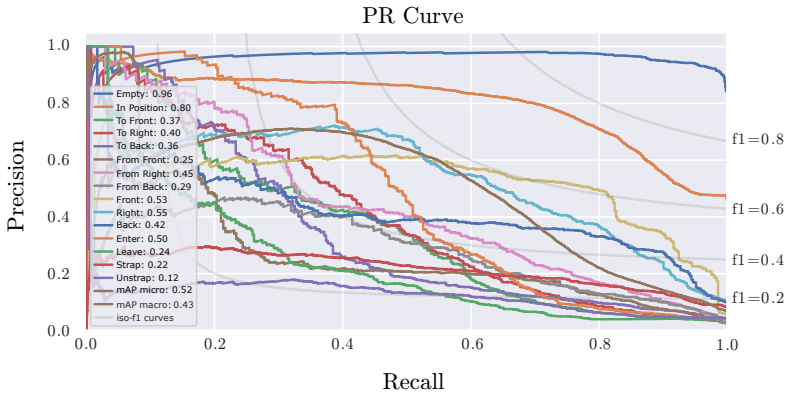
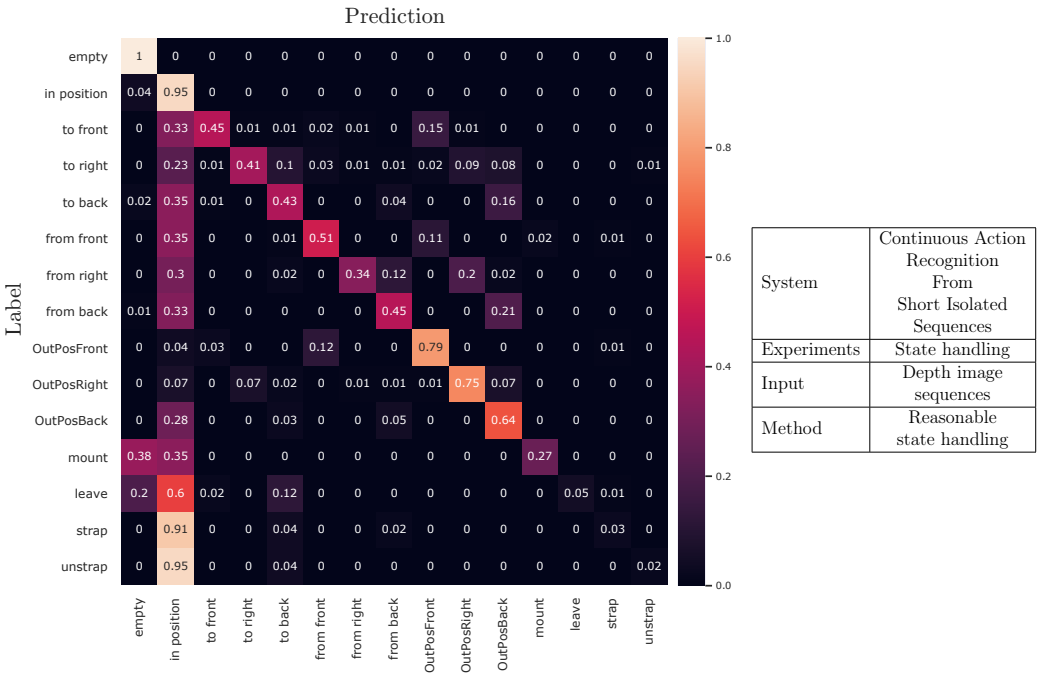


Figure C.74: Precision recall curves of the action recognition network trained on all frames of reasonably concatenated amplitude image sequences with reasonable state handling

Depth Input



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	State handling
Input	Depth image sequences
Method	Reasonable state handling

Figure C.75: Relative confusion matrix of the action recognition network trained on depth image sequences with reasonable state handling

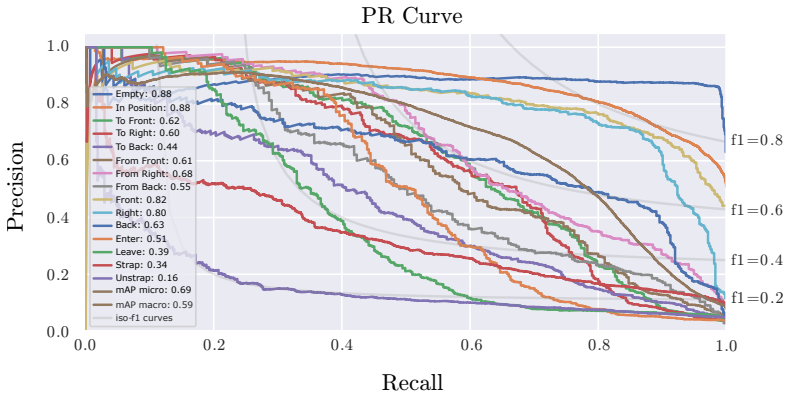
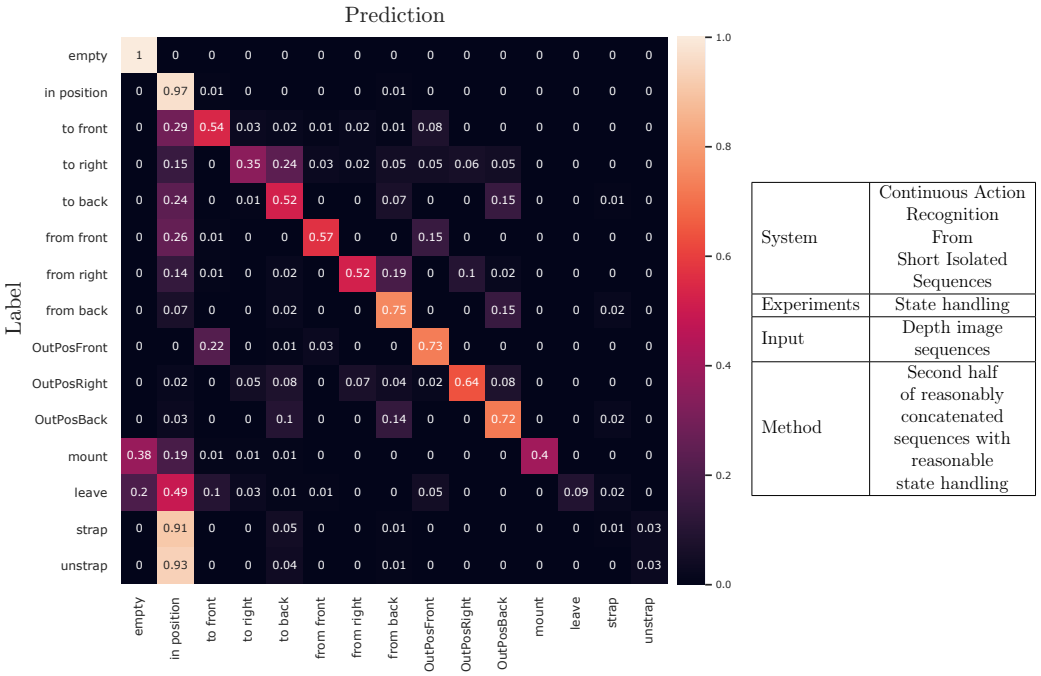


Figure C.76: Precision recall curves of the action recognition network trained on depth image sequences with reasonable state handling



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	State handling
Input	Depth image sequences
Method	Second half of reasonably concatenated sequences with reasonable state handling

Figure C.77: Relative confusion matrix of the action recognition network trained on the second half of reasonably concatenated depth image sequences with reasonable state handling

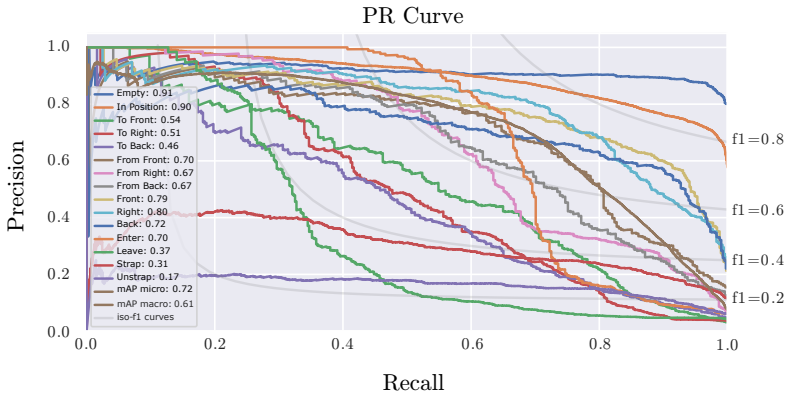
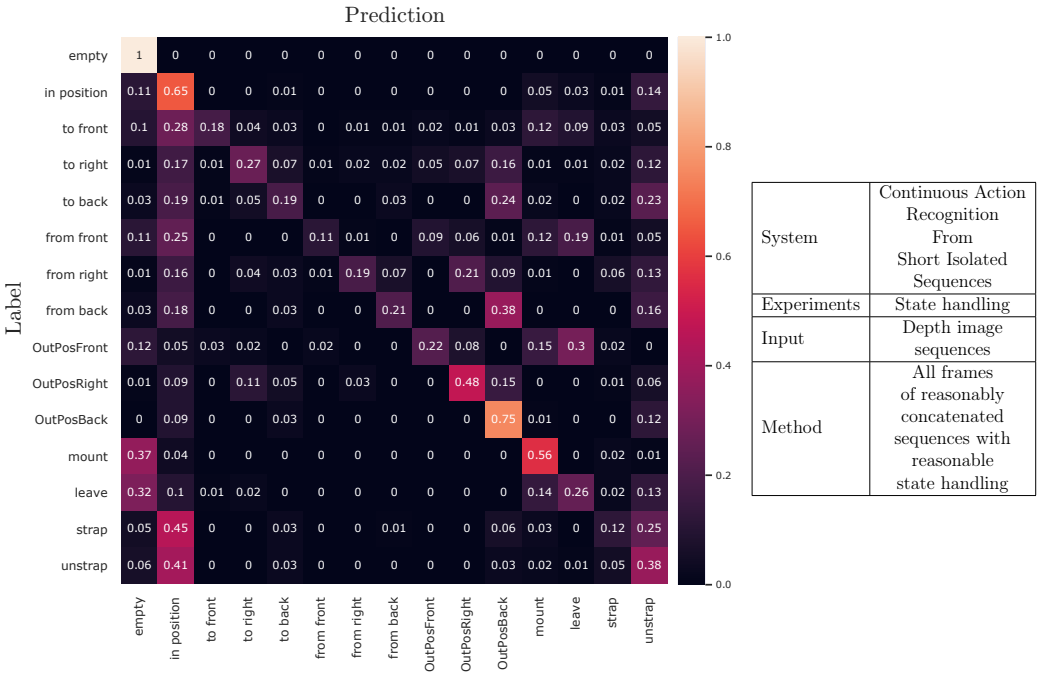


Figure C.78: Precision recall curves of the action recognition network trained on the second half of reasonably concatenated depth image sequences with reasonable state handling



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	State handling
Input	Depth image sequences
Method	All frames of reasonably concatenated sequences with reasonable state handling

Figure C.79: Relative confusion matrix of the action recognition network trained on all frames of reasonably concatenated depth image sequences with reasonable state handling

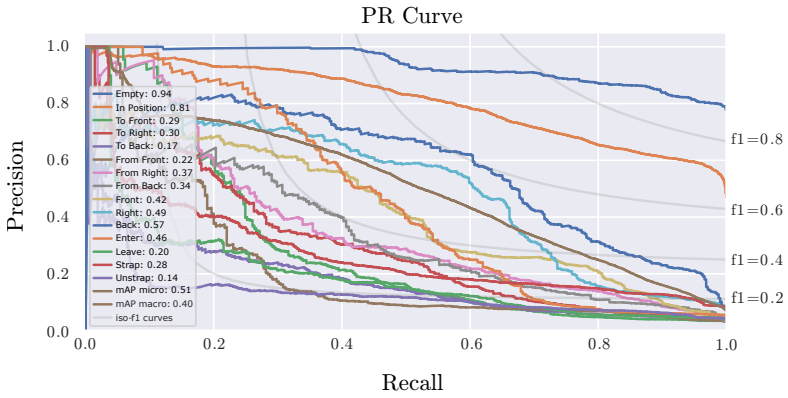
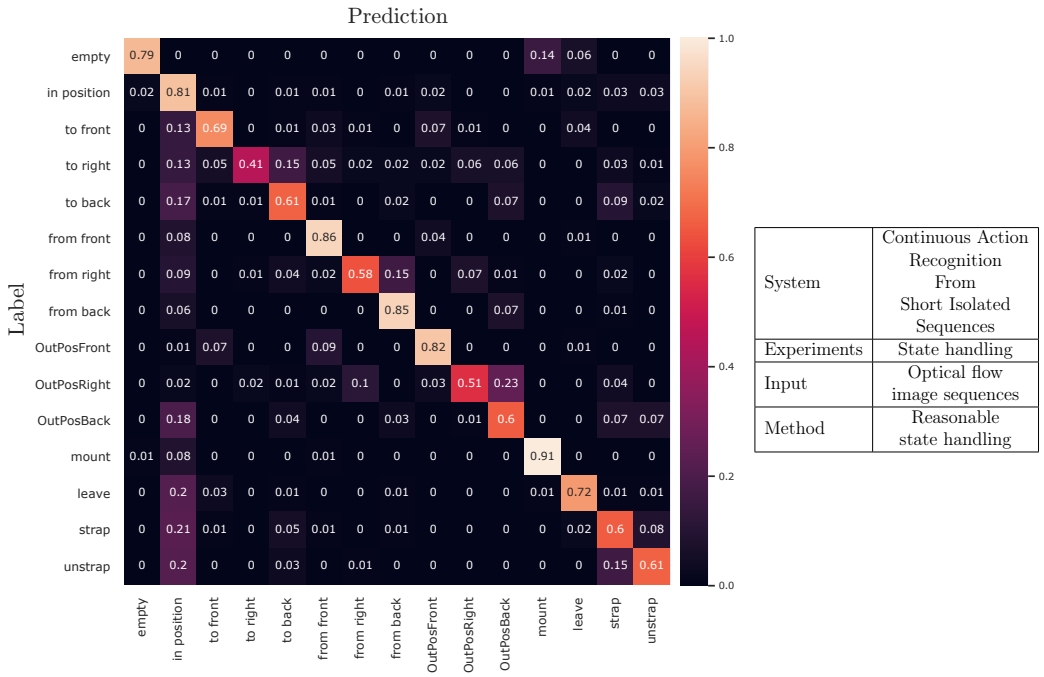


Figure C.80: Precision recall curves of the action recognition network trained on all frames of reasonably concatenated depth image sequences with reasonable state handling

Optical Flow Input



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	State handling
Input	Optical flow image sequences
Method	Reasonable state handling

Figure C.81: Relative confusion matrix of the action recognition network trained on optical flow image sequences with reasonable state handling

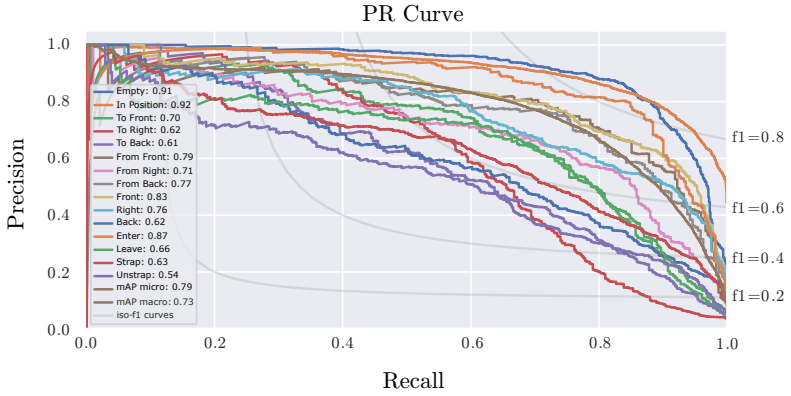
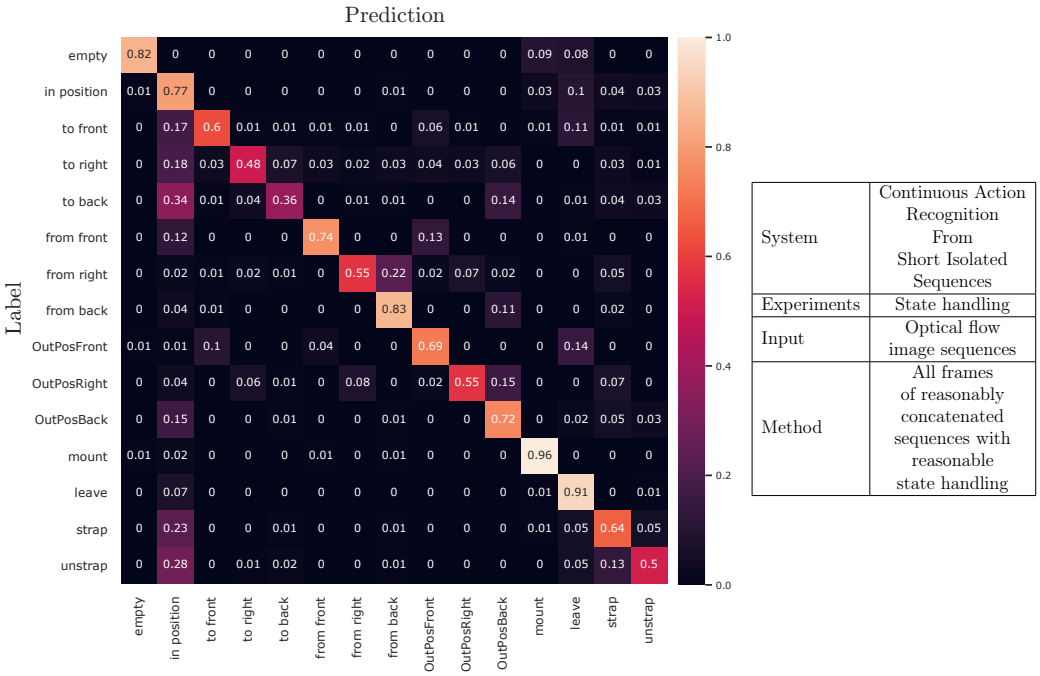


Figure C.82: Precision recall curves of the action recognition network trained on optical flow image sequences with reasonable state handling



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	State handling
Input	Optical flow image sequences
Method	All frames of reasonably concatenated sequences with reasonable state handling

Figure C.83: Relative confusion matrix of the action recognition network trained on all frames of reasonably concatenated optical flow image sequences with reasonable state handling

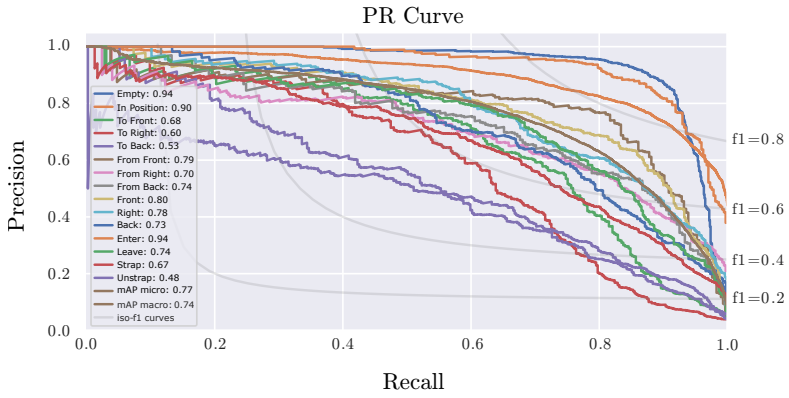
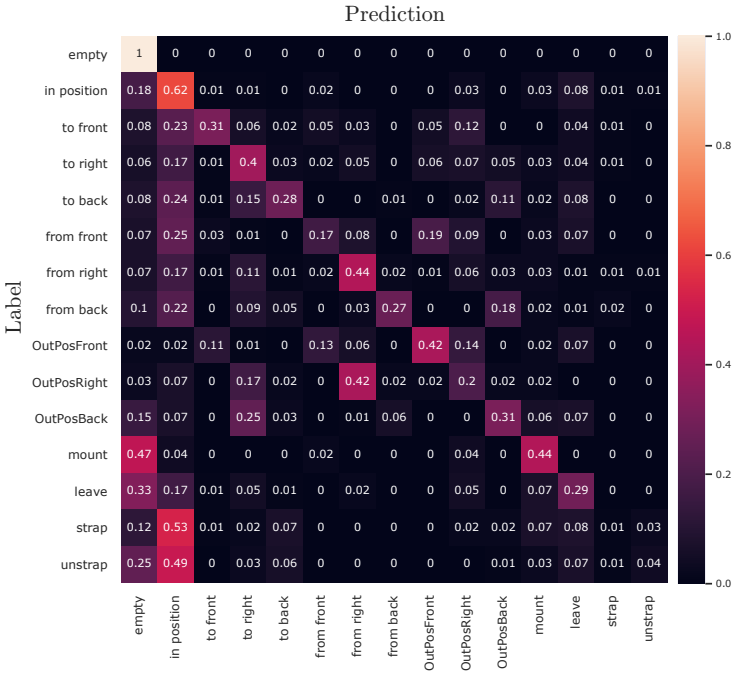


Figure C.84: Precision recall curves of the action recognition network trained on all frames of reasonably concatenated optical flow image sequences with reasonable state handling

LSTM Results

Following, the confusion matrices and precision recall curves of the remaining systems described in chapter 6.4.4 are shown and belong to the systems evaluated in table 6.8. The networks trained with reasonable state handling and action sequence concatenation described in the chapters 6.3 and 6.4 were trained with LSTM networks instead of GRU networks.

Amplitude Input



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	LSTM
Input	Amplitude image sequences
Method	LSTM with reasonable state handling

Figure C.85: Relative confusion matrix of the action recognition network with LSTM modules as recurrent units trained on amplitude image sequences with reasonable state handling

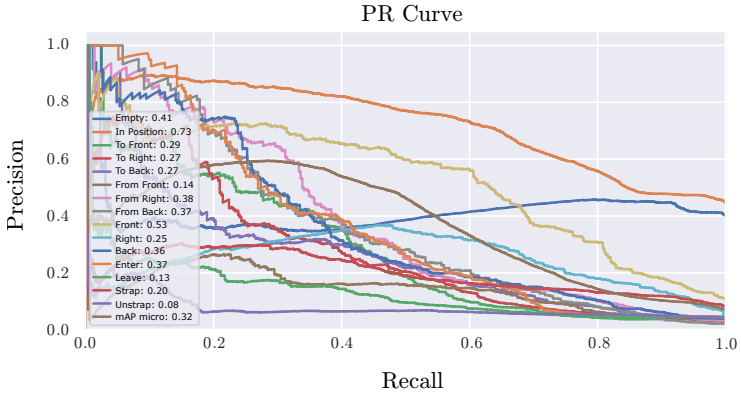
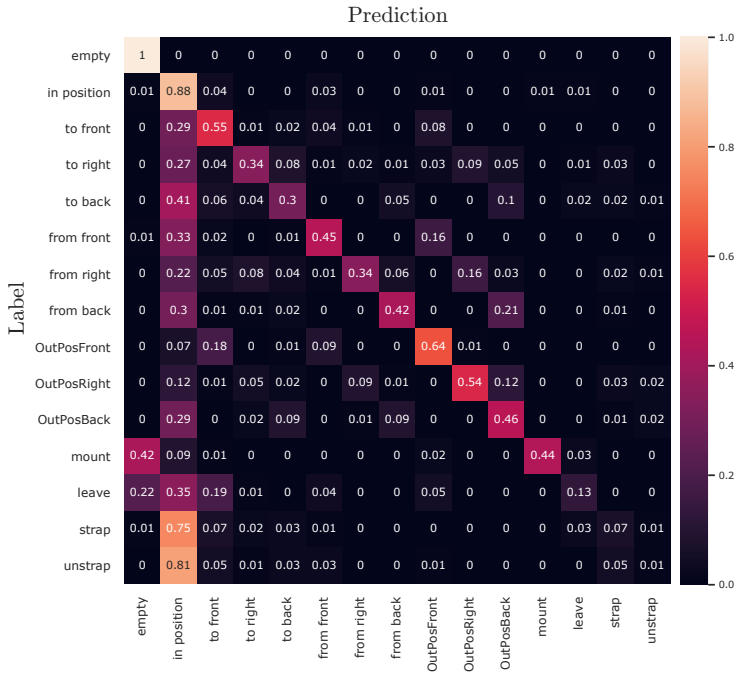


Figure C.86: Precision recall curves of the action recognition network with LSTM modules as recurrent units trained on amplitude image sequences with reasonable state handling



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	LSTM
Input	Amplitude image sequences
Method	LSTM with second half of reasonably concatenated sequences with reasonable state handling

Figure C.87: Relative confusion matrix of the action recognition network with LSTM modules as recurrent units trained on the second half of reasonably concatenated amplitude image sequences with reasonable state handling

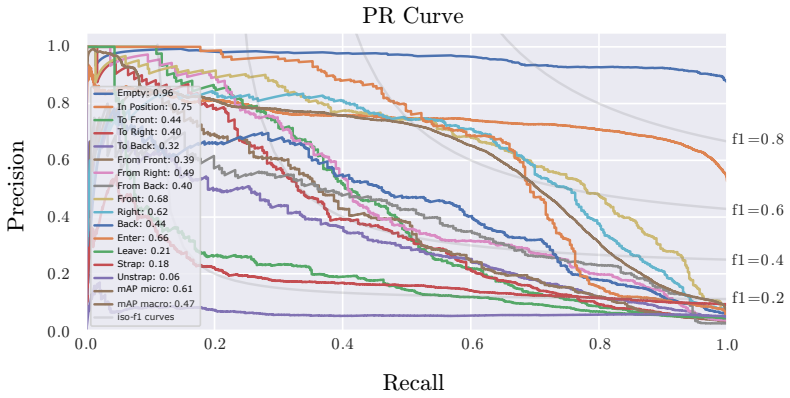
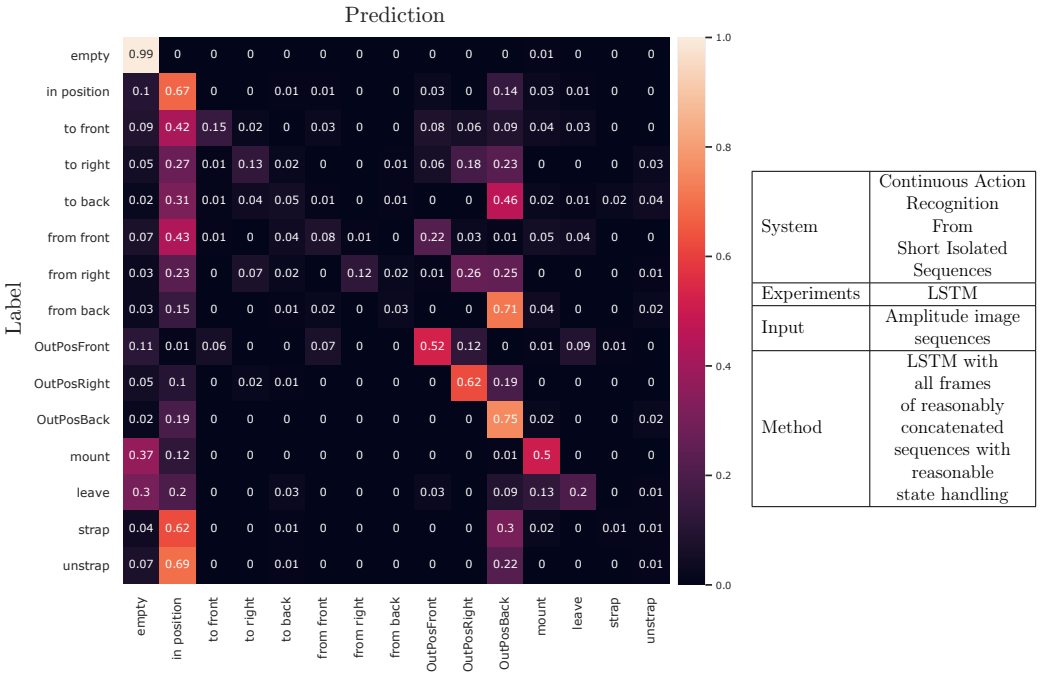


Figure C.88: Precision recall curves of the action recognition network with LSTM modules as recurrent units trained on the second half of reasonably concatenated amplitude image sequences with reasonable state handling



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	LSTM
Input	Amplitude image sequences
Method	LSTM with all frames of reasonably concatenated sequences with reasonable state handling

Figure C.89: Relative confusion matrix of the action recognition network with LSTM modules as recurrent units trained on all frames of reasonably concatenated amplitude image sequences with reasonable state handling

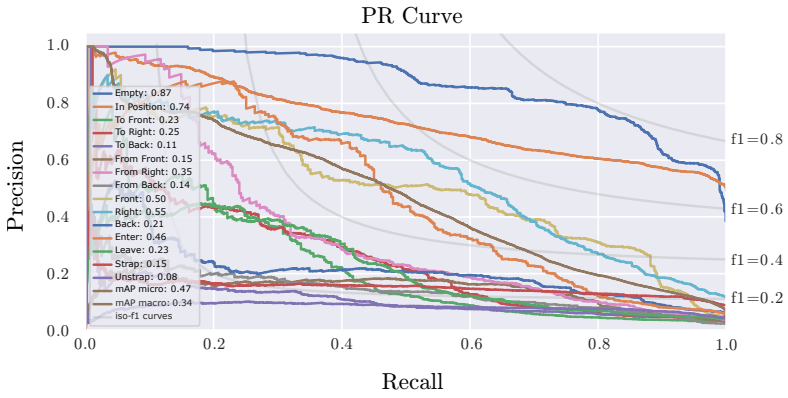
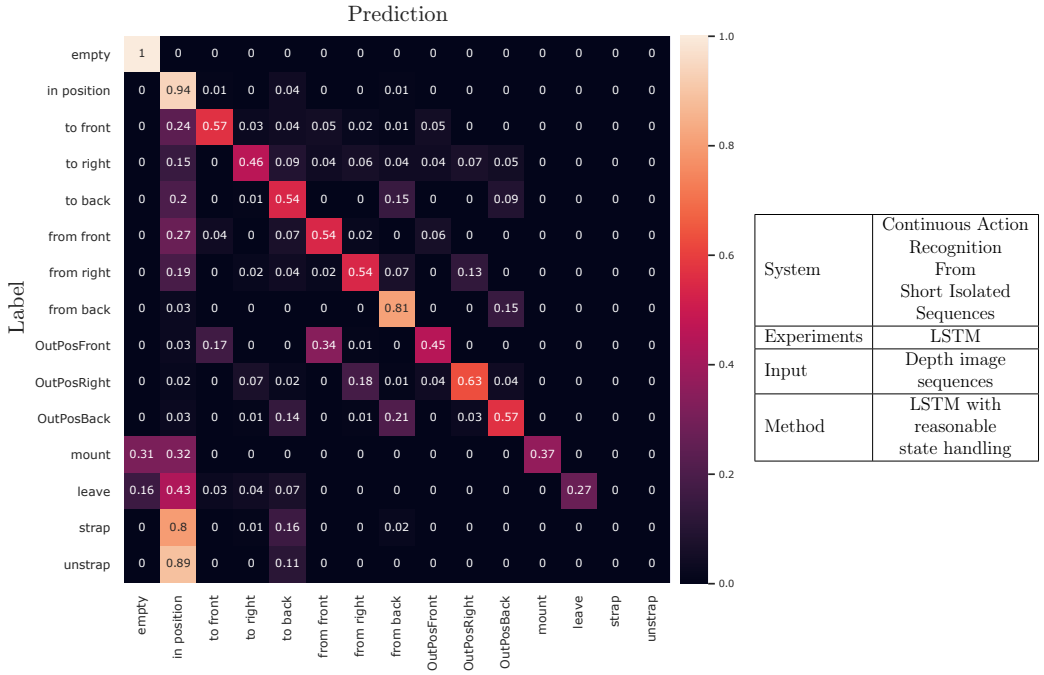


Figure C.90: Precision recall curves of the action recognition network trained with LSTM modules as recurrent units on all frames of reasonably concatenated amplitude image sequences with reasonable state handling

Depth Input



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	LSTM
Input	Depth image sequences
Method	LSTM with reasonable state handling

Figure C.91: Relative confusion matrix of the action recognition network with LSTM modules as recurrent units trained on depth image sequences with reasonable state handling

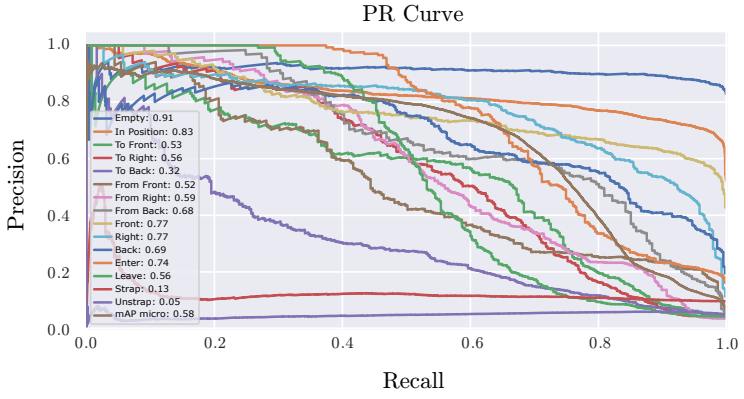
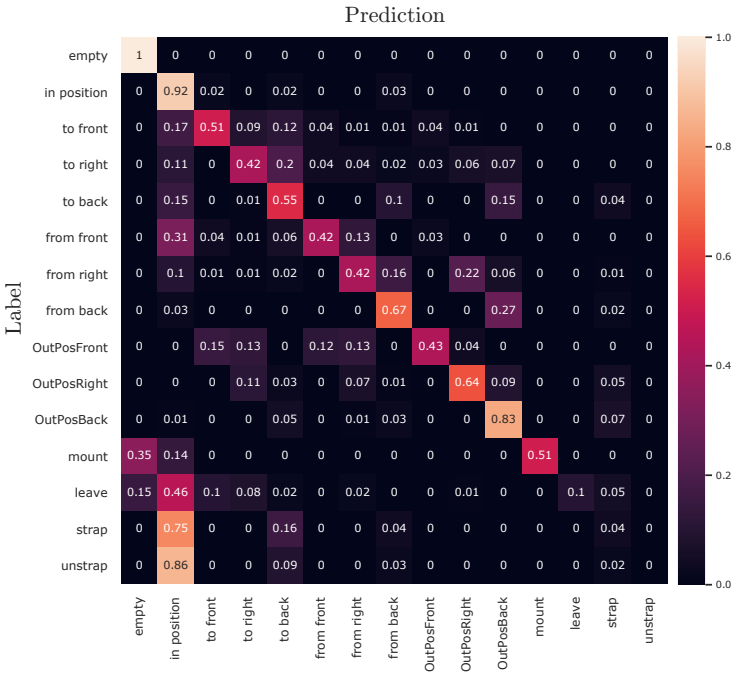


Figure C.92: Precision recall curves of the action recognition network with LSTM modules as recurrent units trained on depth image sequences with reasonable state handling



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	LSTM
Input	Depth image sequences
Method	LSTM with second half of reasonably concatenated sequences with reasonable state handling

Figure C.93: Relative confusion matrix of the action recognition network with LSTM modules as recurrent units trained on the second half of reasonably concatenated depth sequences with reasonable state handling

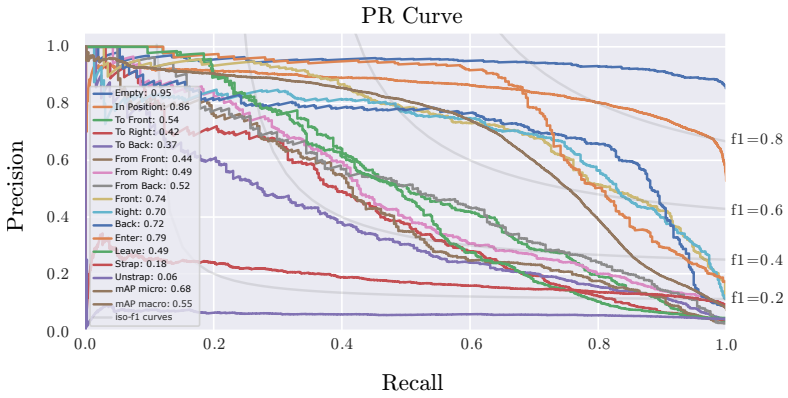
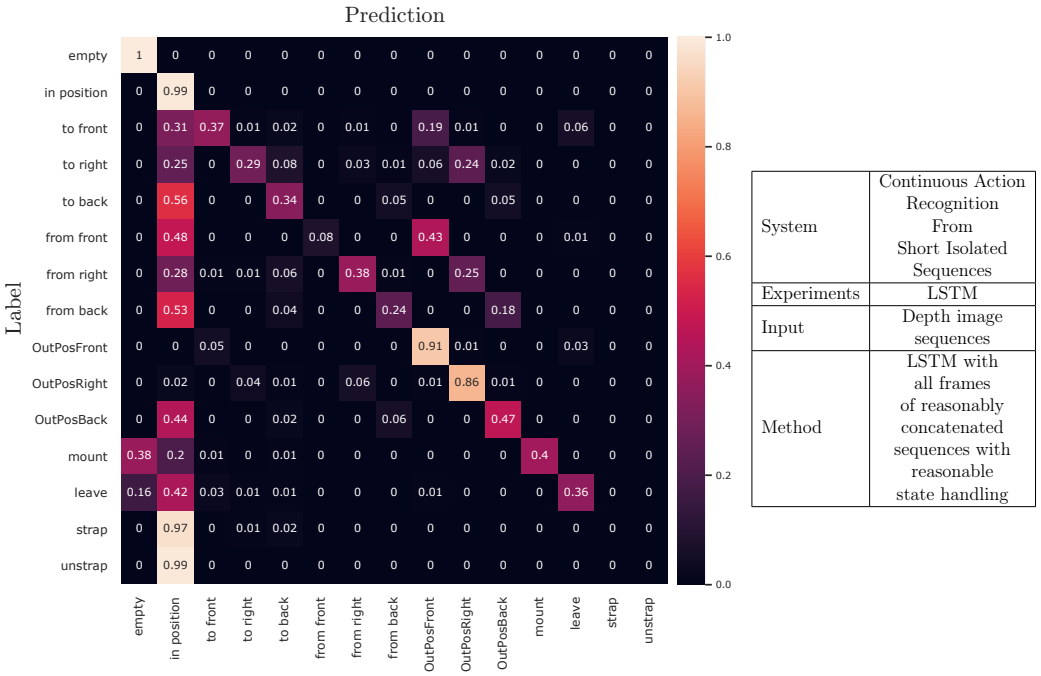


Figure C.94: Precision recall curves of the action recognition network with LSTM modules as recurrent units trained on the second half of reasonably concatenated depth image sequences with reasonable state handling



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	LSTM
Input	Depth image sequences
Method	LSTM with all frames of reasonably concatenated sequences with reasonable state handling

Figure C.95: Relative confusion matrix of the action recognition network with LSTM modules as recurrent units trained on all frames of reasonably concatenated depth image sequences with reasonable state handling

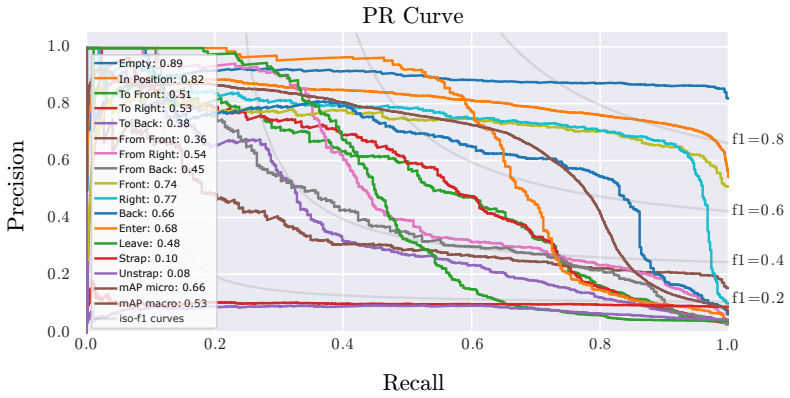
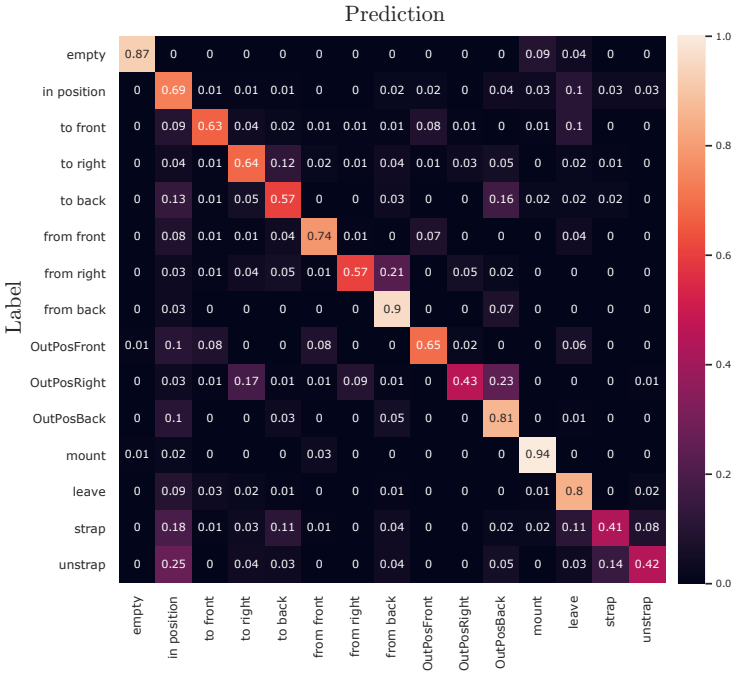


Figure C.96: Precision recall curves of the action recognition network trained with LSTM modules as recurrent units on all frames of reasonably concatenated depth image sequences with reasonable state handling

Optical Flow Input



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	LSTM
Input	Optical flow image sequences
Method	LSTM with reasonable state handling

Figure C.97: Relative confusion matrix of the action recognition network with LSTM modules as recurrent units trained on optical flow image sequences with reasonable state handling

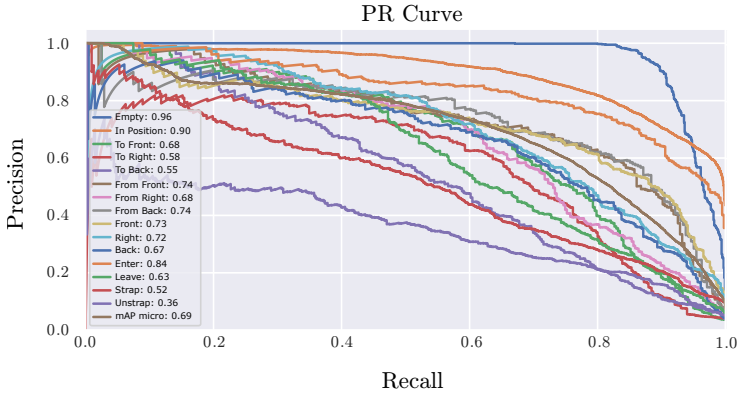
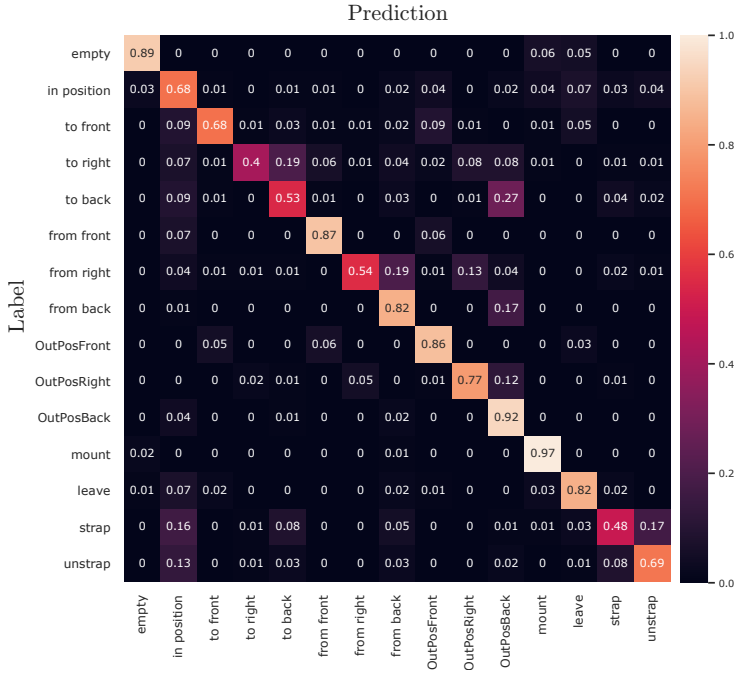


Figure C.98: Precision recall curves of the action recognition network with LSTM modules as recurrent units trained on optical flow image sequences with reasonable state handling



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	LSTM
Input	Optical flow image sequences
Method	LSTM with second half of reasonably concatenated sequences with reasonable state handling

Figure C.99: Relative confusion matrix of the action recognition network with LSTM modules as recurrent units trained on the second half of reasonably concatenated amplitude optical flow sequences with reasonable state handling

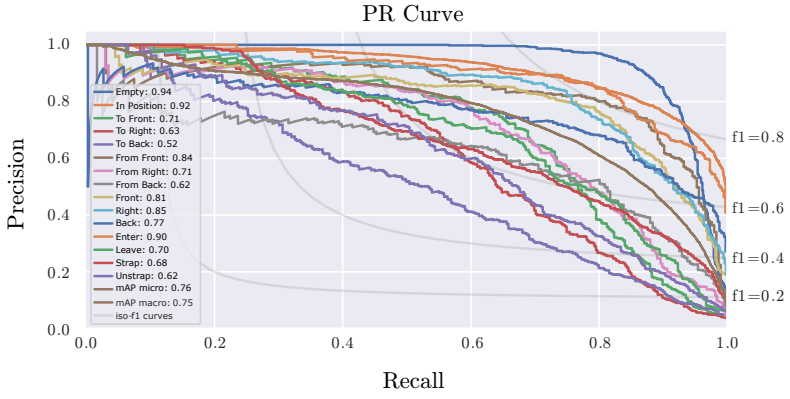
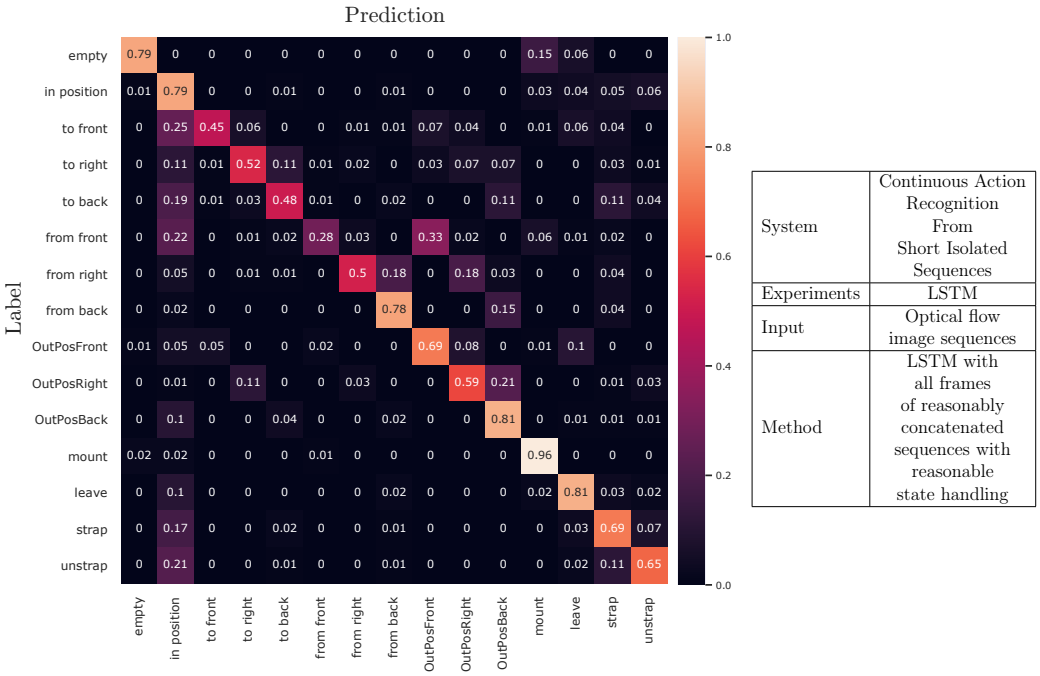


Figure C.100: Precision recall curves of the action recognition network with LSTM modules as recurrent units trained on the second half of reasonably concatenated optical flow image sequences with reasonable state handling



System	Continuous Action Recognition From Short Isolated Sequences
Experiments	LSTM
Input	Optical flow image sequences
Method	LSTM with all frames of reasonably concatenated sequences with reasonable state handling

Figure C.101: Relative confusion matrix of the action recognition network with LSTM modules as recurrent units trained on all frames of reasonably concatenated optical flow image sequences with reasonable state handling

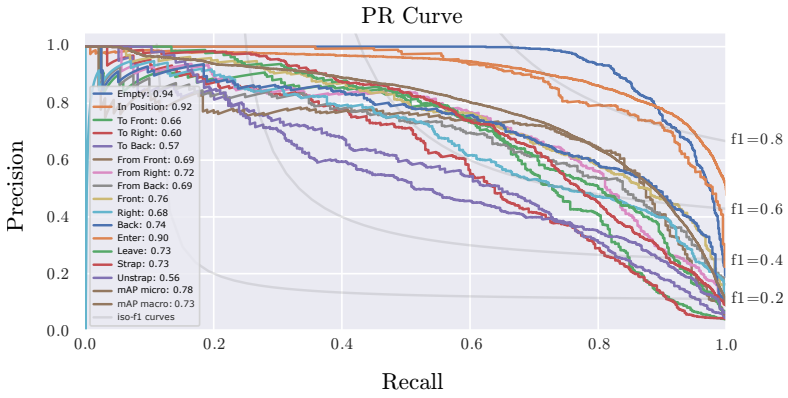


Figure C.102: Precision recall curves of the action recognition network trained with LSTM modules as recurrent units on all frames of reasonably concatenated optical flow image sequences with reasonable state handling