Pedestrian Pose Recognition in Traffic for Autonomous Driving

Project Report for APPRAS, Summer Semester 2023

Maximilian Bartels
Technische Universität Berlin
Berlin, Germany
m.bartels@campus.tu-berlin.de

Mingming Lyu
Technische Universität Berlin
Berlin, Germany
mingmin.lyu@campus.tu-berlin.de

Robert Strehlow
Technische Universität Berlin
Berlin, Germany
strehlow@campus.tu-berlin.de

Deniz Mert Tecimer
Technische Universität Berlin
Berlin, Germany
tecimer@campus.tu-berlin.de

***Abstract*—Accurately recognizing pedestrian behavior can enhance the safety of autonomous vehicles. This paper tackles this challenge by breaking it down into two main tasks: recognizing actions and understanding commands. To accomplish these tasks, the paper develops models that use temporal information including LSTM, TCN, ST-GCN, and RA-GCN, utilizing 2D body keypoints obtained through OpenPose video processing. These models can effectively predict common actions like walking and hand gestures, as well as standard commands such as "stop", "go right", and "follow me". Impressively, the average accuracy for recognizing each action exceeds 85%.**

***Index Terms*—OpenPose, action recognition, command recognition, skeleton data, LSTM, graph convolutional network**

I. Introduction

Being able to recognize and classify a human's pose might be easy for another human, but it is certainly not an easy task for a machine, although carrying a lot of non-verbal information about the human's future actions, could be helpful in the machine's decision making.

In this report, we will present our methods and results to create a pose recognition system, with our main focus being pedestrian poses in the traffic domain. Recognizing pedestrian poses, especially when done by an autonomous vehicle (AV), can have a large influence on the behavior of the driver or AV. This could improve the overall security, not only for the occupants of the vehicle but for the surrounding pedestrians as well. For example: Being able to detect a person who is about to cross the street unexpectedly, can result in an emergency break for the vehicle that was about to approach the person. Introducing another safety step in the form of an automatic driving assistant can potentially reduce the number of traffic accidents involving pedestrians.

Recognizing pedestrian poses is also necessary to reach the highest level of autonomy, defined by the Society of Automotive Engineers[15], which requires no human intervention at all. This includes the necessity to understand authorized traffic controllers (ATC), whose main responsibility is to control the traffic of a locally defined area by commonly used hand gestures.

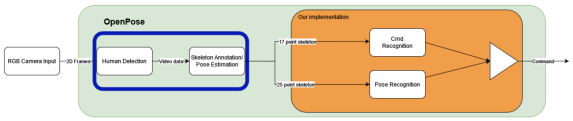

Fig. 1. The planned project pipeline

Differentiating between these two main problem cases of pedestrian pose recognition, we decided to split the general problem into two separate tasks, one dealing with classifying a more general use case called "Action Recognition", which information is mostly used in a passive manner, while the other tasks deals with the classification of common command gestures used to actively influence the AV's behavior. The reason for this task separation is the underlying semantic complexity between the two tasks. As a machine, knowing when to act in a situation based on the surrounding pose information could require much more information than available during the course of this project. Therefore, whether to act or not as an AV will be solely determined by the different use cases we have defined in the form of two separate problem definitions.

We first defined a pipeline in order to achieve the task in multiple steps using different models and planned milestones to track our improvements gradually. The pipeline in figure 1 introduces an input module and preprocessing module to obtain the body keypoints powered by OpenPose [11]. These keypoints are fed into command and pose recognition modules, whose outputs are used to compute the final output.

However, we aimed to provide a variety of base experiments with different settings so that the performance of different approaches can be examined. As the pose and command recognition tasks are similar and are both skeleton classification problems, we tested different approaches that can be used as

a base in the future.

In order to solve the problems at hand, we created our milestones as shown in figure 2. We did state-of-the-art research until the first milestone and got familiar with the features and the usage of necessary tools. At the same time, we decided on the datasets and the models we wanted to use. Until the second milestone, we conducted our first experiments with basic classification and created the project pipeline for the input-to-output data flow. Before working on the third milestone, based on the feedback from the second milestone we updated our model structure and fine-tuned some models. At the last milestone, we deployed our model and tested the pipeline on our test car Christina I.

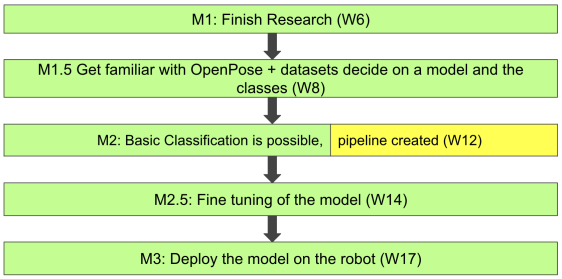

Fig. 2. Project Milestones

II. Pose Estimation

For the development of a pose recognition system, it is necessary to retrieve a generalized representation of a person's pose, which can be represented in a uniform coordinate space. Assuming an already existing object recognition system can detect multiple persons from an RGB image, a pose classification model applied directly to this image data could be heavily dependent on certain image features uncorrelated to the present pose shown by the human. To overcome this issue, a pose estimation can be performed, to transform the detected pose into a skeleton-based view. A skeleton in this context consists of multiple connected body key points to represent a simplified human skeleton.

Having the data in a graph structure, containing relative distances between specific joints, allows a classification model to focus on the positioning of the body key points for a specific pose. Not only does the skeleton representation transform the data into a more uniform space eliminating possible external influences, but its numerical form is also more suitable as input data for the upcoming neural network models.

Since there already exist pose estimation frameworks [2], [11], we decided to use an open source pose estimation framework called "OpenPose"[1]. This framework will help us to analyze images and video frames and extract body key points from input data in real-time. It will also serve as an analytical tool to retrieve body key points from already existing datasets, as well as self-recorded videos.

[1]https://github.com/CMU-Perceptual-Computing-Lab/openpose

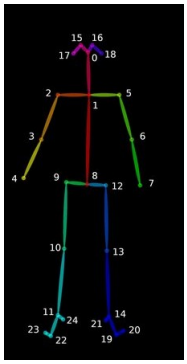

Fig. 3. OpenPose generated 25 body keypoints [10]

OpenPose creates a skeleton for each detected human, where each skeleton consists of 25 body key points, as Fig 3 shows. In our implementation, the Python API of OpenPose was used to integrate and extend its functionality into our own project. Additional functions that we created included a real-time input stream, which connects to an available camera, analyzing the last recorded image frame, as well as a video analyzer to extract the skeleton data from input videos or images.

The computational requirements of OpenPose are reasonably high, resulting in long execution times when analyzing large amounts of input data. Therefore, the usage of a GPU supporting the CUDA[2] framework is highly recommended when analyzing large datasets. For this reason, the JAAD dataset[7], which will be used in the upcoming sections, was analyzed by OpenPose with an NVIDIA RTX 3090. Analyzing all 346 videos in this dataset took about 3 hours.

III. Action Recognition

The term *action recognition* is not well defined and is often used as a synonym for pose estimation - a process to infer body information from an image or video while an action is performed (for further details see the pose estimation section) - in this paper we want to classify actions based on skeleton data.

A. Dataset

Our base dataset is the JAAD dataset, which contains pedestrian and driver behavior and the point of crossing. It contains 346 videos with various lengths (5-10s) with 30 FPS and they claim that the videos represent everyday situations in urban driving scenarios. Each pedestrian has a behavior annotation (walking, crossing, nodding, etc.) which is utilized as classes for our models. Each pedestrian is assigned one or multiple behavior annotations, which is why our models employ multi-label classification. Additional tags like weather or demographic information about the pedestrian are not used, as well as any information about the driver [12].
Since the JAAD dataset does not contain human-skeleton information about the recognized pedestrians, we have to run OpenPose on it to get the body keypoints we want to utilize in

[2]https://developer.nvidia.com/cuda-toolkit

our approach. A body keypoint was assigned to a pedestrian if it lies within the bounding box given by the dataset. Since OpenPose has problems with pose estimation if the person is not in the center of the picture or is not clearly visible, some of the pedestrians were not recognized. Another drawback of the JAAD dataset is its imbalance, while nearly every recognized person in the JAAD dataset is walking (85.52 %) the distribution of the other labels is even worse as only 0.29 % of the pedestrians show a hand gesture and only 0.08 % are nodding into the direction of the car, the looking label is with an occurrence of 18.6 % on the same level as the walking label.

To handle this problem, we decided to use additional human action datasets. Given the constraints of our approach (see. next section) the datasets must have either the same or translatable multi-labels as the JAAD dataset or must be easy to relabel to fit into our pipeline. Therefore, widely used datasets such as the *HACS Dataset* [17] or *Stanford 40 actions* [4] were not usable for us since they contain a great number of videos with more than one human and no information which person is the subject of the label. Using videos with only one person in it would require that we watch each video ourselves, which would require a lot of time with uncertain results that would go beyond the scope of this work.

Under these circumstances we decided to use a subset of the action database of the KTH first presented in a paper from 2004 [13] which originally contained "six types of human actions (walking, jogging, running, boxing, hand waving and hand clapping) performed several times by 25 subjects in four different scenarios." [13]. As we only wanted additional data for our imbalanced labels we only used the waving and clapping videos since many of the JAAD pedestrians are walking further. We decided not to use the boxing label since it is a typical gesture for a pedestrian. The remaining videos were labeled as showing a hand gesture, not walking, looking in the direction of the car/camera, and not nodding.

B. Models

One of the specifics of human action recognition is its possible temporal component (e.g. hand waving where the gesture is only recognized over the movement as well as nodding). To account for this property we decided to use time-sensitive models (LSTMs, TCN and ST-GCN). Further, as pedestrian poses don't limit themselves to one action per time but perform combinations of them in every moment simultaneously we have to handle them in an instance resulting in a multi-label problem.

For the LSTM we use a deep learning approach and create a sequential model with an LSTM Layer (Output size: 64) with a tanh activation function followed by 3 dense layers (Output size: 128, 64, 32) with a ReLU activation function resulting in an output layer (Output size: 4). ReLU as an activation function was used to assure that the model can learn more complex connections, as with a linear activation function. For the loss binary cross entropy was used as well, and an averaged F1-score to take the data imbalance into account.

As LSTMs have their limitations with longer sequences of data (ca. 4 seconds for videos) [8], we also implemented two TCN approaches: one where we use the deep structure of our LSTM approach but exchange the LSTM layer with a TCN layer and shallow network consisting of only one TCN layer followed by a dense output layer.

Additionally, we also utilize the ST-GCN model, which stands for Spatio-Temporal Graph Convolutional Network. It combines both temporal and spatial information, allowing it to capture interactions between 25 body key points in video sequences. The adjacency matrix is shown in Fig. 4 with size 25 $\times$ 25, which helps leverage the spatial relations of human joints. Initially, we created a single ST-GCN layer that can later be stacked to create deeper networks. This layer takes the key points data and adjacency matrix as input, and the results pass through a temporal convolutional layer. Finally, the Parametric ReLU activation is applied. Following that, we employed another network to stack three ST-GCN layers with output sizes of 64, 32, and 64. After that, global pooling is performed to reduce the spatial dimensions, and Batch Normalization is applied to the feature tensor. A fully connected layer with an output size of 1 is used, followed by the Sigmoid activation function to produce the final output.

C. Experiments

The LSTM and TCN as well as ST-GCN Models were trained with a threshold of 0.5 and 0.6, each LSTM with an unlimited amount of steps per epoch with 11 epochs. Since TCNs consume much training data each of these models was limited to 1000 steps per epoch and 8 epochs. The batch size for each approach was 32. The ST-GCN is set to 30 epochs with a batch size of 64.

TABLE I
ACCURACY OF THE MODELS

Model	look	walk	hand gesture	nod
LSTM Threshold 0.6	84.71	87.42	99.7	99.91
LSTM Threshold 0.5	85.03	87.37	99.68	99.91
Deep TCN 0.6	85.28	87.87	99.72	99.70
Deep TCN 0.5	85.34	88.29	99.69	99.91
Shallow TCN 0.6	85.73	89.03	99.71	99.92
Shallow TCN 0.5	85.59	88.85	99.73	99.91
ST-GCN 0.5	84.52	87.48	99.75	99.95
ST-GCN 0.6	84.57	87.67	99.74	99.95

D. Results

The table above shows that the models give realistic and good results for the 'look' and 'action' labels(It's worth noting that 'action' refers to walking in our project), but the high accuracy for 'hand gesture' and 'nod' indicates overfitting. The test set was 1/3 of each dataset, which ensures that models have to deal with data that wasn't included in the training set. Given the cross-subject settings of our experiments, the cause of the unrealistic high accuracy is not caused by overfitting. The F1-score for the hand gestures indicates that the model

is accurate in its positive predictions, and captures most of the actual positive cases. Since the gestures are only limited to waving and clapping, experiments with different motions and signs. The F1-score of 0 for nodding results from the near-zero amount of positive labels in the dataset, the main reason is that the amount of nod sample is still too small. The high accuracy and F1-score for the walking label indicate that the model can actually predict pretty well if a pedestrian is walking. Regarding the looking label, the F1-score indicates that the model is better than a random guess in predicting if a pedestrian looks in the direction of the car, but is still not good. One of the reasons for this might be that only one key point is per default used for the head information, therefore, lacking additional information about the eyes, from this, we consider that looking is difficult to recognize through the existing 25 body key points alone. While it is technically possible to compute the missing eye information by changing some settings in the OpenPose configurations, it turned out that most of the pedestrians are too far away from the camera to effectively predict the positions of the eyes. To examine this further, more experiments with additional data would be necessary.

After examining the performance of the models it turns out that although all models produce similar values, the TCN models with a threshold of 0.5 deliver the best results compared to others.

TABLE II
F1-SCORE OF THE MODELS

Model	look	walk	hand gesture	nod
LSTM Threshold 0.6	0.64	0.92	0.99	0
LSTM Threshold 0.5	0.65	0.92	0.99	0
Deep TCN 0.6	0.67	0.92	0.99	0
Deep TCN 0.5	0.7	0.93	0.99	0
Shallow TCN 0.6	0.69	0.93	0.99	0
Shallow TCN 0.5	0.7	0.93	0.99	0
ST-GCN 0.6	0.64	0.92	0.99	0
ST-GCN 0.5	0.64	0.92	0.99	0

IV. COMMAND RECOGNITION

Besides action recognition and developing systems that can perceive and interpret their surroundings, directly communicating with a machine via human actions is a big research field in computer vision and intelligent systems. Autonomy mainly aims to decrease human intervention [1]. In some cases, this intervention can be helpful or is necessary for the so-called autonomous system to function correctly. In this research, we focus on cases such as recognizing commands of an authorized figure (e.g. traffic police officer directing an autonomous vehicle, or guiding delivery robots indoors to overcome the difficulties due to indoor mapping).

A. Dataset

Many datasets were researched and examined. However, as denoted by the creators of the HRI-Gestures Dataset [6], most of them were created for simplex communication. Therefore, we decided to use the HRI-Gestures Dataset which enables the duplex communication when used together with our action recognition module.

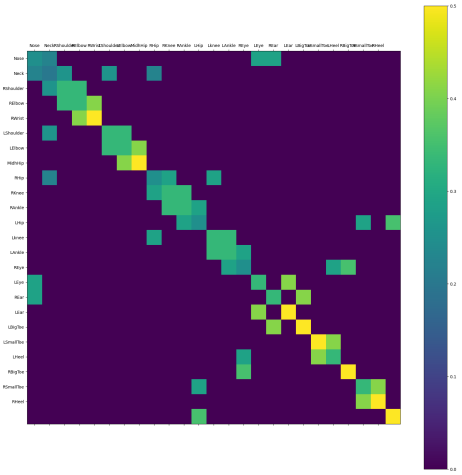

Fig. 4. Normalized adjacency Matrix, a matrix representation based on the Pedestrian Graph, representing all the connections for the 25 chosen human keypoints

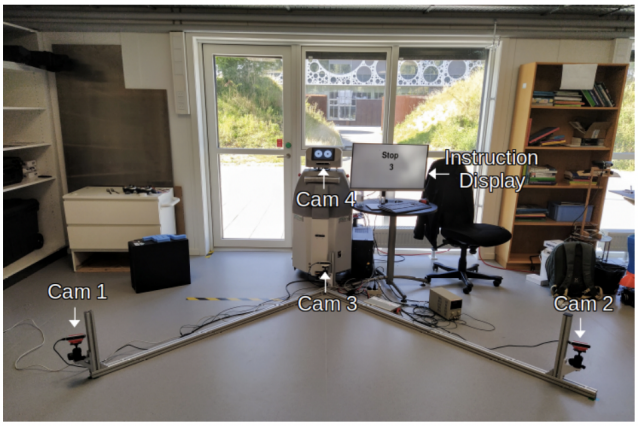

Fig. 5. The camera setup for creating the dataset [6]

The HRI-Gestures Dataset was created by recording 17 subjects performing 20 different actions(commands) 10 times with 4 different cameras located at different angles. Due to data protection regulations, they did not publish the original recordings, however, 17 body key points were published, which were extracted from the recordings via a framework proposed by Juan et al. [5].

Although the HRI gestures dataset [6] includes three-dimensional data, we decided to only use the x and y coordinates. The reason for this is the generalization purpose we would like to achieve with this model, using it on our own data, which only includes two dimensions.

The camera setup for the recordings can be viewed in the following figure 5. While reading the dataset, we discovered that some data was missing, resulting in an imbalanced dataset. The details are shared in the following sections.

Fig. 6. Stop (left), Go right (right)

Fig. 7. Skeleton example generated from the 2D joints from the dataset

The dataset contains 15 different action classes and 5 passive action classes. Active actions are ”Stop”, ”Go right”, ”Go left”, ”Come here”, ”Follow me”, ”Go away”, ”Agree”, ”Disagree”, ”Go there”, ”Get attention”, ”Be quiet”, ”Don’t know”, ”Turn around”, ”Take this”, ”Pick up” and passive actions are ”Standing still”, ”Being seated”, ”Walking towards”, ”Walking away”, ”Talking on phone”. However we do not have the recordings, you can view an example recording snapshot and an example skeleton plot in figure 6 and 7 respectively.

Even though they reference COCO Dataset [9] for using 17 key points, they do not use the same edges combining the joints. The difference is presented in the figure 8.

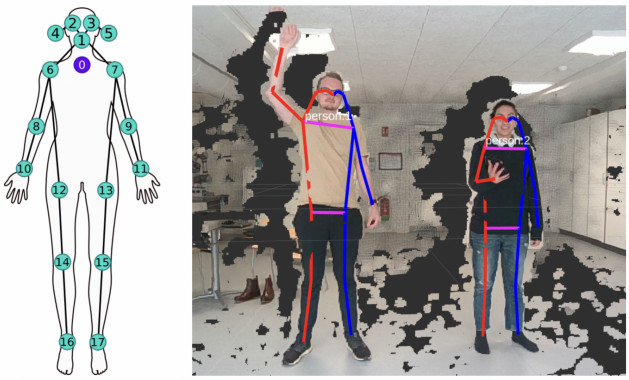

Fig. 8. The edges from COCO (left), the edges used in HRI-Gestures Dataset (right)

B. LSTM Network Experiments

As an improvement of RNNs, LSTM networks provide better results in generating predictions with sequence data [16]. It is well-known and widely used for time-series data, too. As a base method, we decided to conduct three experiments using the HRI-Gestures Dataset in a heuristically created LSTM network, which consists of an LSTM layer, followed by 3 fully connected layers making multi-class classification for 12 selected action classes. The layers have 64, 128, 64, and 12 units, respectively. The model structure is shown in figure 9. In each experiment, we used the ADAM optimizer with default parameters and the categorical cross-entropy loss as our loss function. The model was evaluated with the categorical accuracy metric while training for 100 epochs with a batch size of 32.

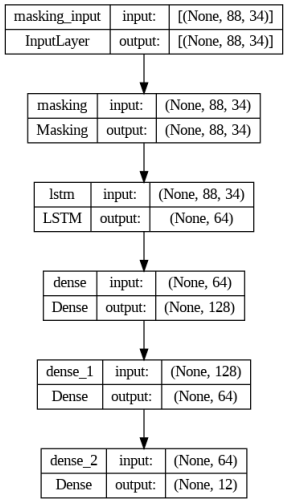

Fig. 9. The LSTM model structure

For the experiments, we decided to use 12 subsequent action classes which are more suitable for our use case. These classes are ”Stop”, ”Go right”, ”Go left”, ”Come here”, ”Follow me”, ”Get attention”, ”Be quiet”, ”Standing still”, ”Being seated”, ”Walking towards”, ”Walking away”, and ”Talking on phone”. As before mentioned, the imbalanced training data class distribution can be viewed in figure 10 which also shows that a similar amount of training data was used in each experiment. As we want to test each class with the same amount of data, we split the data accordingly.

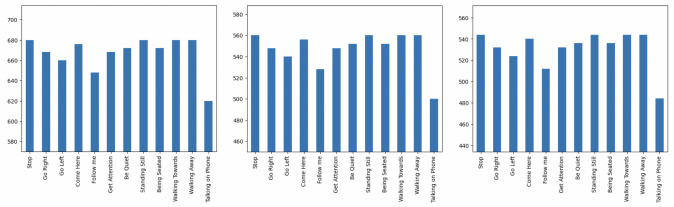

Fig. 10. The sample counts for each class for the base (left), cross-subject (middle), and cross-repetition (right) experiments

The first experiment is called the base experiment in which randomly %20 of the data is used for testing and %80 for training. The second one is the cross-subject experiment which aims to measure the subject dependency of the model. The data is split based on the subjects. The data of the 5 subjects are reserved for testing and the model is trained only with the rest. The last experiment is the cross-repetition experiment which aims to measure the performance of the model when 3 repetitions of each action recording of each subject are used for testing and the rest for training. The results of these three experiments are presented in table III.

TABLE III
EXPERIMENT RESULTS

Accuracy (%)	Base	Cross-Subject	Cross-Repetition
Train	97.13	96.13	97.30
Test	91.32	76.94	83.52
F1-Score (Weighted)	91.36	78.04	83.44

The model performance was also tested class-wise, confusion matrices and f1-score of each class are presented in figure 11 and in table IV, respectively.

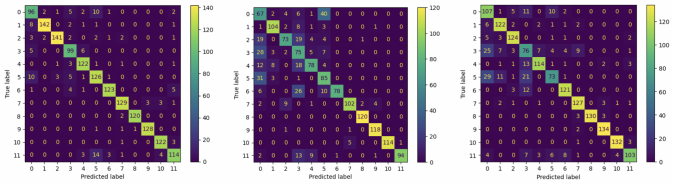

Fig. 11. Confusion matrices of the base experiment (left), cross-subject (middle), cross-repetition (right)

TABLE IV
F1-SCORES OF THE EXPERIMENTS WITH LSTM

F1-Scores	Base	Cross-subject	Cross-repetition
Stop	79.33	45.52	68.58
Go Right	94.35	86.66	86.21
Go Left	94.00	69.19	89.85
Come Here	86.08	52.44	55.07
Follow Me	92.07	71.23	86.69
Get Attention	83.72	62.27	62.93
Be Quiet	93.53	78.39	88.00
Standing Still	94.50	89.47	90.39
Being Seated	98.76	98.76	97.74
Walking Towards	97.70	96.72	96.40
Walking Away	95.68	97.43	96.70
Talking on Phone	85.39	87.44	82.73

Even though the result of the base experiment seems promising, when we compare the model performance in the cross-subject and -repetition settings, the performance drops significantly. Based on this comparison, we can conclude that the model overfitted the seen samples from the training set, such as subjects or repetitions. For instance, the model strongly confuses between Stop and Get attention samples. This might be due to the similarity between these samples. The model performs overall the best on passive classes yielding over %90 F1-Score.

Skeleton data, especially 2D body keypoints data lacks the edge information. When an LSTM model with 2D coordinates representing the joints is used for such a task, we expect the model itself to extract that information which can already be extracted and fed in that way. This problem removes the graph structure which could be utilized better. The imbalanced dataset and not having enough data might also cause overfitting on a class or underfitting overall, by not allowing the model to generalize the extracted features. This can be overcome by adding some dropout layers in the model or by upscaling the data with some augmentation methods. Some gestures contain hand movements, however, the hand keypoints are not provided and the model has only seen the wrist keypoints during training.

C. Richly-Activated Graph Convolutional Network (RA-GCN)

1) RA-GCN Structure: Another experiment for the command recognition task was conducted with a Richly-Activated Convolutional Neural Network, which was derived from [14]. This network consists of spatial-temporal graph convolutional networks (ST-GCN), capturing and interpreting the underlying graph structure of the input data, given as a graph of body key points. The network is also capable of introducing a simulated occlusion in the input data, removing specific joints given by the user.

Understanding the temporal shift of body key points is a key factor in extracting the correct command displayed by a pose since command gestures might include dynamic joint movement (e.g. *Follow Me, Walking Towards*). For this reason, the temporal component of the ST-GCN is used to not only learn static constellations of body key points but also the continuous movement involved in portraying the specific command. This is done by a temporal convolutional block in the model.

The spatial information, given by the graph structure from the input data, will be handled by the multiple spatial convolutional blocks. These components are designed to analyze the spatial information contained inside the graph structure given to the neural network, representing the body key points of a person.

The input data given to the RA-GCN is in the following dimension form: (C, T, V, M). C is the number of coordinates, T is the number of maximum frames, V is the number of joints a skeleton contains, and M is the maximum count of persons detected in a video sequence. Since we are dealing with command gestures and the ATC should be uniquely identified, we put ourselves under the constraint that only one person should be visible at any time in the input video sequence, hence the latter variable will always be one in our proceedings.

Another discrepancy is found between the number of detected body key points. While the HRI gestures dataset only includes 17 body key points as shown in figure 7, the skeleton produced by OpenPose includes 25 body key points. Since the 17 body key points are all included in the 25 keypoints of the OpenPose, we removed the remaining data to match the

HRI gestures dataset. Finally, we set the maximum number of frames to 90, since the videos in the dataset do not exceed this number and our own recordings match this upper boundary as well. The final dimension for our input data is therefore: $(2, 90, 17, 1)$.

The complete structure of our custom RA-GCN is as follows: The input data gets preprocessed, introducing two additional input dimensions, one for the temporal shift and another one for the occlusion, resulting in an updated dimension given by $(6, 90, 17, 1)$. The data is then sent to a multi-model stream, consisting of our example of three identical models. Each model first performs a batch normalization before training the ST-GCNs. After that, a global average pooling operation is performed before a fully connected layer into the output dimension consisting of the number of classes to be classified.

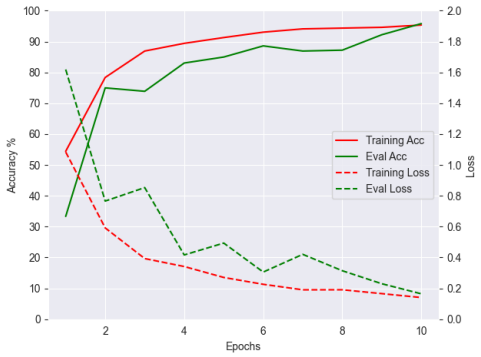

Fig. 12. Learning Curve of Cross-Subject model after 10 Epochs.

2) RA-GCN Experiments: The conducted experiments are performed with the underlying motive of deploying this model in a real-world scenario. For this purpose, we only focused on specific command gestures, namely 'Follow Me', 'Standing Still', and 'Stop'. These three commands are enough to control a basic robot by two direct, active actions with opposing intents and a passive action to not react at all.

The dataset was prepared in the exact way explained earlier and fed to the model. Since the RA-GCN has a rather complex structure, training time was heavily influenced by the used hardware. We were able to train the model on a Ryzen 9 7950X CPU, reducing training time for 10 Epochs to approximately 1 hour.

The model uses the ADAM optimizer and a cross-entropy loss function. The data gets split into batch sizes of 64. The training data gets shuffled, to reduce the effect of a structural bias in the input domain. For the experiments, we decided on a cross-subject approach, dividing the training and evaluation set based on the subjects that were recorded. This will create a better understanding of how well the model will be able to generalize its classification ability across multiple subjects.

After evaluating the results, we can see in figure 12 that the model's learning ability is very strong, with an accuracy reaching up to 95%. The evaluation accuracy approaches the training accuracy over time and finally reaches nearly the same level of accuracy as the training data. The same can be observed for the loss, where the loss during the evaluation of different subjects is slowly approaching the loss reached during the training phase. This result indicates a strong generalization of the model, being able to correctly classify the three commands ("Follow Me", "Standing Still", "Stop") performed by never-before-seen subjects.

As another experiment, we investigated the model's performance outside its learned dataset. For this purpose, we recorded our own videos displaying the trained commands in a similar matter as seen in figure 5. The recordings have been done with a resolution of 1920×1080 and 30 frames per second. The length of the recordings varies but generally does not exceed the maximum frame number set to 90. In cases where there are more frames, the remaining frames will get cut at the end of the sequence. In total, there were 70 recordings with a distribution of 26 'Follow Me', 24 'Standing Still', and 20 'Stop'. Some examples can be seen in figure 13 with their corresponding labels.

Fig. 13. Examples from our recordings

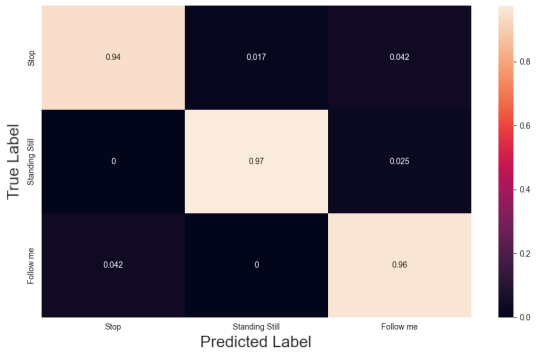

Fig. 14. Confusion Matrix of Cross-Subject Evaluation Set.

The results of the classification done by the same model as in the previous experiment on our custom-created video recordings can be seen in the confusion matrix in figure 15. As we can observe, the model has a strong confidence when observing the 'Follow Me' command. This is not the case for the commands "Standing Still" and "Stop", which seem to get confused, with a bias for predicting "Stop". The reason for this could lie in the similar distances between the body key

pointy in these two poses. The ”Standing Still” pose is done by not moving any joints during the recording, while the ”Stop” pose includes raising one arm, in most cases the right arm, and signaling a common ”Stop” gesture with the palm of the hand. Therefore, the only difference between those two poses is only indicated in two body key points, the hand and the elbow. The ”Follow Me” pose includes movement with the whole body, showing a clear distinction from the other two poses. This may result in high confidence in this command.

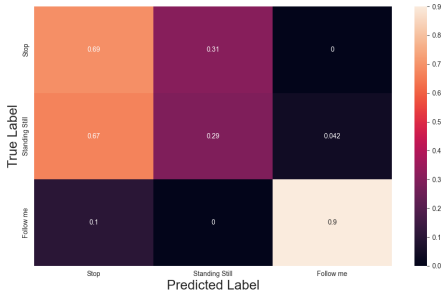

Fig. 15. Confusion Matrix of own data recordings.

Finally, the model does well within its trained domain as can be seen in figure 14, that is the HRI gestures dataset. However, being unable to obtain the original video data, it remains unclear how exactly the gestures were recorded and if there exist any specific movements or placements of joints within the three chosen commands that could influence the learning ability of the model. Other deviations from the HRI gestures dataset might include the model view perspective and the performed data normalization, which was also not well enough explained in the original paper[6] and had to be approximated on our own. Therefore we can conclude, that the RA-GCN is performing well within the dataset it was trained upon but has difficulties identifying the correct commands displayed by our own custom-created data, showing a lower confidence in predicting similar command poses.

V. Deployment

We built a small test car, called Christina I, with Raspberry Pi 4 [3]. The 2D video input is recorded through a laptop camera. The input is sent to the RA-GCN model and the detected command is sent to the car through the HTTP API. As denoted in our constraints, the car does not react to the command or perform an action based on that.

VI. Conclusion

The project consists of developing two different models for the command and action recognition tasks, following the body keypoints extraction from a video input. In this section, we want to evaluate our performance in managing the project and the results of different experiments.

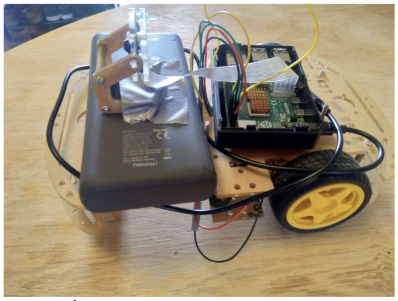

Fig. 16. Christina I

A. Experiments

We conducted different experiments for pose estimation and in order to overcome the data limitation, different sources were merged and used together for the model training. When model performances are compared, methods using TCN yield the best results both in overall accuracy and F1-score.

All of the models used in action recognition experiments failed to classify the ”nod” class. As we have very few samples from that class, it is highly possible that the model underfits this class. Randomly assigning another label has a higher possibility of making a correct prediction as our data is imbalanced. It is also important to note that the datasets were merged and the data do not come from the same distribution but from a similar one. A balanced and bigger dataset could serve better in utilizing these models.

Our aim for command recognition was to develop a model that could distinguish and recognize commands independent of the subjects it was trained on. That is why, cross-subject performance evaluations and F1-scores of the classes. Our most successful results were achieved with RA-GCN. However, when tested on our custom data, it did not perform as well as the test data from the same distribution as the training data. This could be caused by the similarity between the classes we experimented with, such as ”Standing still” and ”Stop”. They look very similar, one can even classify ”Stop” as a sub-action, which can be done while standing still. Also as ”Standing still” is not a command but a state, we could have chosen better fitting labels for our task.

Regardless of the limitations we had, we managed to create a pipeline that takes video input from a camera (a laptop camera in our case) runs the input through the model, and finally sends the output to our model car, Christina.

B. Project Management

During the project, we managed to create a group in which everyone was heard and the communication channels were established well. Each week, we mostly met twice to track our progress and plan our next steps or discuss necessary changes. This helped a lot in meeting our requirements for the project completion and our milestones defined at the beginning of the project.

In the beginning, we utilized the ticketing system well, but later we did not use it much. We could have created tickets

for more detailed tasks, but we trusted our communication which is not ideal. Sometimes, we acted as two different groups while working on the action and command recognition modules which caused a separation of the main task, making the merging processes harder for us later on. However, we had the chance to apply different methods, take different actions, and cover a wider range of methods. If we had a group member who is only responsible for project management and ticketing etc., we believe that the project management and the code structure could be maintained better.

Hereby, we also would like to thank our supervisor for the fundamental motivation of this task, his guidance, and his help throughout the project.

References

[1] Chinedu Pascal Ezenkwu and Andrew Starkey. "Machine Autonomy: Definition, Approaches, Challenges and Research Gaps". In: *Intelligent Computing*. Ed. by Kohei Arai, Rahul Bhatia, and Supriya Kapoor. Springer International Publishing, 2019, pp. 335–358. ISBN: 978-3-030-22871-2.

[2] Hao-Shu Fang et al. "Alphapose: Whole-body regional multi-person pose estimation and tracking in real-time". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[3] Warren Gay. *Raspberry Pi Hardware Reference*. 1st. USA: Apress, 2014. ISBN: 1484208005.

[4] Kai-Jun Hu, He-Feng Yin, and Jun Sun. "Discriminative non-negative representation based classifier for image recognition". In: *Journal of Algorithms & Computational Technology* 15 (Sept. 2021), p. 174830262110449. DOI: 10 . 1177 / 17483026211044922.

[5] William Juel et al. "An Integrated Object Detection and Tracking Framework for Mobile Robots". In: *Proceedings of the 17th International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO,* INSTICC. SciTePress, 2020, pp. 513–520. ISBN: 978-989-758-442-8. DOI: 10.5220/0009888405130520.

[6] Avgi Kollakidou et al. "HRI-Gestures: Gesture Recognition for Human-Robot Interaction". In: *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. Ed. by Giovanni Maria Farinella, Petia Radeva, and Kadi Bouatouch. 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Online ; Conference date: 06-02-2022 Through 08-02-2022. SCITEPRESS Digital Library, 2022, pp. 559–566. DOI: 10.5220/0010871200003124.

[7] Iuliia Kotseruba, Amir Rasouli, and John K Tsotsos. "Joint attention in autonomous driving (JAAD)". In: *arXiv preprint arXiv:1609.04741* (2016).

[8] Colin Lea et al. *Temporal Convolutional Networks: A Unified Approach to Action Segmentation*. 2016. arXiv: 1608.08242 [cs.CV].

[9] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 740–755.

[10] Atchara Namburi and Thapani Hengsanankun. "Combining SVM and Human-Pose for a Vision-based Fall Detection". In: *ICIC Express Letters Part B, vol. 13, no. 11* (2022).

[11] Daniil Osokin. "Real-time 2d multi-person pose estimation on cpu: Lightweight openpose". In: *arXiv preprint arXiv:1811.12004* (2018).

[12] Amir Rasouli, Iuliia Kotseruba, and John K Tsotsos. "Are they going to cross? A benchmark dataset and baseline for pedestrian crosswalk behavior". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2017, pp. 206–213.

[13] C. Schuldt, I. Laptev, and B. Caputo. "Recognizing human actions: a local SVM approach". In: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* Vol. 3. 2004, 32–36 Vol.3. DOI: 10.1109/ICPR.2004.1334462.

[14] Yi-Fan Song, Zhang Zhang, and Liang Wang. "Richly activated graph convolutional network for action recognition with incomplete skeletons". In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, pp. 1–5.

[15] SAE Taxonomy. "Definitions for terms related to driving automation systems for on-road motor vehicles". In: *SAE: Warrendale, PA, USA* 3016 (2018).

[16] Christian Bakke Vennerød, Adrian Kjærran, and Erling Stray Bugge. "Long Short-term Memory RNN". In: *CoRR* abs/2105.06756 (2021). arXiv: 2105.06756. URL: https://arxiv.org/abs/2105.06756.

[17] Hang Zhao et al. "HACS: Human Action Clips and Segments Dataset for Recognition and Temporal Localization". In: *arXiv preprint arXiv:1712.09374* (2019).

VII. Appendix

In the following table, you will find the work distribution in our group.

Mingming Lyu	• For the report:Abstract,action recognition focusing on ST-GCN part • For the code:ST-GCN file • find suitable dataset to offset data imbalance
Deniz Mert Tecimer	• For the report: Introduction, Command Recognition Introduction, HRI-Gestures Dataset, and LSTM part, Deployment part, Conclusion and proofreading • For the code: CommandRecognition file
Robert Strehlow	• For the report: Introduction, Pose Estimation, RA-GCN part • In Code: data_parser, op_utils, and the ragcn file
Maximilian Bartels	• The Action Recognition part in the report with focus on the LSTM and TCN parts • The LSTMs and TCN models in the code • JAAD data preprocessing • HACS, Stanford 40 actions data preprocessing • Programming for the robot/pipeline