Noname manuscript No.
(will be inserted by the editor)

From perception to action using observed actions to learn gestures

Wolfgang Fuhl

Received: date / Accepted: date

Abstract Pervasive computing environments deliver a multitude of possibilities for human-computer interactions. Modern technologies, such as gesture control or speech recognition, allow different devices to be controlled without additional hardware. A drawback of these concepts is that gestures and commands need to be learned. We propose a system that is able to learn actions by observation of the user. To accomplish this, we use a camera and deep learning algorithms in a self-supervised fashion. The user can either train the system directly by showing gestures examples and perform an action, or let the system learn by itself. To evaluate the system, five experiments are carried out. In the first experiment initial detectors are trained and used to evaluate our training procedure. The following three experiments are used to evaluate the adaption of our system and the applicability to new environments. In the last experiment the online adaption is evaluated as well as adaption times and intervals are shown.

Keywords Gestures, Supervised Learning, Neuronal Network Adaption, Neuronal Network, Online Adaption

Acknowledgements Work of the authors is supported by the Institutional Strategy of the University of Tübingen (Deutsche Forschungsgemeinschaft, ZUK 63).

1 Introduction

Computers in our daily environments are versatile. There exist notebooks, smartphones, desktop computers, cars, intelligent lighting, and multi-room entertainment systems to name only a few. Each device offers a variety of

Eberhard Karls Universität Tübingen
Sand 14
Tel.: +49-7071-29-70492
E-mail: wolfgang.fuhl@uni-tuebingen.de

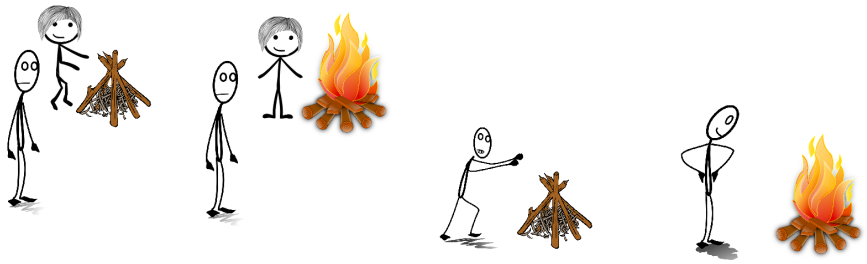

Fig. 1: Learning from observations.

interaction techniques: Some are keyboard, touch, voice, mouse, gestures, or gaze. Each is consistent in itself, yet different with regard to the usability. Meaning often, the time to acquaint oneself to all the features and proper usability becomes laborious, leading to errors and frustration.

An example of onerous device acquaintance is gesture-based control; when the user learns the pre-programmed gestures. There are some disadvantages in this context however because the gestures may be unusual for humans, making the use of the interaction technique uncomfortable. Another disadvantage of the pre-programmed gesture-based control is that it is impossible to use if any fingers or arms are injured. Additionally, it also affects people who suffer from physical limitations. In the area of voice control, all dialects can be problematic (Simpson and Levine, 2002). With this interaction technique, it is also necessary to learn the words to control the computer as well as the user has to get used to the commands to feel comfortable.

In contrast, the human being can learn by observing another person without any explicit interaction, which is known as observational learning, a concept of the social learning theory (Bandura et al., 1961). Figure 1 shows an example of this theory. One person observes another as she lights a pile of wood with flint pieces. Then, she also builds a pile of wood and ignites it with sparks by striking the flint together.

The human being is capable of this because the human brain is a marvel and capable of the extraordinary. However, its capacity and functionality are limited. We absorb information through the sensory organs, which send signals to be processed in the sensory cortex and further relayed to many other brain structures. How long we store information depends not only on its importance, but also how important we perceived it (Bloom, 1976; Rao and Gagie, 2006). A rough categorization is auditory, haptic, and perceptual learning (Ausubel et al., 1968). In this paper, we focus on perceptual learning from the computer's point of view. Meaning, the computer learns to execute an action by only receiving visual input and the status of the action.

Therefore, we conducted two experiments where the computer learns by observing the user. In the first experiment, the user trained the computer explicitly to execute an action. Therefore, the user made gestures in front of the camera and executed an action on the computer (opening an applica-

tion, pressing a key etc.). In the following three experiments the adaption of our system is evaluated based on additional training examples as well as the adaptability to new environments. The last experiment evaluates the online usage.

This visual learning is possible due to breakthroughs in the area of machine learning (LeCun et al., 1998). Computers already outperform humans in many visual tasks (LeCun et al., 2015; Szegedy et al., 2016), and with the advent of fine-tuning, they are able to learn new things quickly (Yosinski et al., 2014).

2 Related Work

We categorize the related work in two parts. The first part is hand gesture control, since it a type of interaction in which of the computer uses a video or motion source. Here, we summarize the work that has already be done in this area. The second part is a summary of machine learning approaches for learning from observations that are also used in our system and mainly comes from the field of robotics.

2.1 Hand gesture control

Research in the field of hand gesture based human-computer interaction Francke et al. (2007); Dardas and Georganas (2011) uses different sensory systems to develop a fast, reliable, and general gesture classification. Previously, an accelerometer for the measurement of the movement was used as the sensory system (Arce and Valdez, 2010). Afterward, a neuronal network was trained to classify the gesture of the subject (Arce and Valdez, 2010). This work was enhanced using Micro-Electro-Mechanical Systems (MEMS) combined with a wearable glove (Pandit et al., 2009) and also using gyroscopes (Dixit and Shingi, 2012). Since those systems are rather expensive and complex, gloves with imprinted patterns for recognition were developed in (Wang and Popović, 2009). The gesture classification was done based on a video stream using computer vision algorithms. This approach was improved using hand detection, feature extraction, and vector quantization (Lamberti and Camastra, 2011). Earlier work in the field of image-based gesture recognition was with the use of Hidden-Markov-Models (Yang et al., 1997) in combination with color gloves or Haar-like features (Chen et al., 2007). Besides technical obstacles like reliability, speed, and costs, hand gesture interaction must also address the intuitiveness of and the comfort for the user (Corera and Krishnarajah, 2011). The first problem of gesture control in terms of intuitiveness and comfort is the lack of a standardized vocabulary (Corera and Krishnarajah, 2011). In addition, most users would prefer to define their own gestures to perform certain tasks (Li and Jarvis, 2009). Both are necessary to cope with pervasive computing environments and interaction comfort for the user (Li and Jarvis, 2009; Nielsen et al., 2003; Alastalo and Kaajakari, 2005). Modern approaches consist

of hybrid interaction technologies, such as gestures and gaze (Li et al., 2017) or voice (Basanta et al., 2017). The goal is to improve the overall comfort of the user by combining the advantages of different interaction approaches.

The system presented in this paper focuses on user comfort. It cannot accomplish the task of learning complex gestures or behavior in a way to reproduce them, rather it can learn to interpret visual input and to perform an action. The beauty comes from the natural way our system learns, which is called perceptual learning for humans. Users are visually observed and paired to their actions. Here, the actions are on or off decisions, thus simple processes it is able to reproduce.

2.2 Learning from observations

Research regarding observational learning also address imitation learning, which also apply to computer learning (Hussein et al., 2017; Liu et al., 2018). In imitation learning, information about the behavior of the teacher is extracted. This information is used to learn a mapping between the demonstrated behavior and the actions to be performed by the computer (Hussein et al., 2017). It is mainly used in the steering of robots (Schaal, 1999; Ijspeert et al., 2002) and can be split into two categories. The first category is behavioral cloning. Here the behavior is provided as consecutive actions (Pomerleau, 1991; Ross et al., 2011) and the training is done in a supervised fashion. The second category is inverse reinforcement learning, where the training is done based on a reward function (Abbeel and Ng, 2004). Both categories of imitation learning are usually demonstrated and executed in the same context. But there is also work that has studied the imitation of a demonstration with a different context (Dragan and Srinivasa, 2012; Gidaris and Komodakis, 2018).

In our scenario, the data consists of the video stream and the action state (on or off). Therefore, our approach can be assigned to the former category. For training, we use fine tuning (Yosinski et al., 2014; Hoo-Chang et al., 2016) of a deep neuronal network for image classification (Krizhevsky et al., 2012), which was trained on ImagNet (Deng et al., 2009a).

3 Contribution of this work

The contribution of this work is a learning approach for the creation and adaptation of machine learning based human computer interaction systems. The system was evaluated with ten users in five experiments and based on the experiences gained in these experiments, existing limitations are discussed. Furthermore, possible fields of application for human computer interaction for existing software will be discussed and new possibilities are identified. The following is a list of the contribution of this work.

1 Learning approach for creating human computer interaction systems by the user.

(a) Thumbs-up (b) Fist (c) Hand (d) Headphone (e) Monitor

Fig. 2: Five different user movements of the same subject using our web camera.

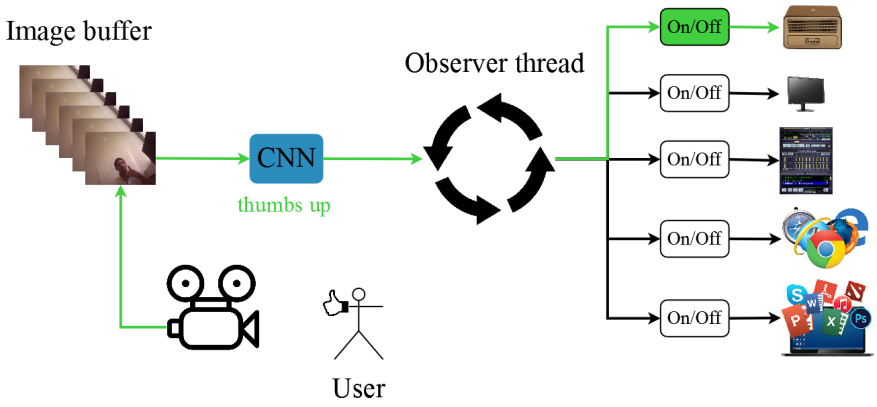

Fig. 3: Performing an action based on the thumbs-up gesture.

2 Learning approach for the adaptation of human computer interaction systems by the user.
3 Extensive evaluation of the system in five experiments.
4 Identification of possible fields of application and the perspective of the approach for existing software.
5 Identification of limitations and possibilities for further research.

4 Method

The used recording setup consists of a common RGB web camera with 30 frames per second (fps) in front of a desktop computer with a 19-inch monitor. For the camera, we set the capture resolution to 1280×960 and downscaled it to 227×227, which is the input size of the CNN.

Figure 2 shows five recorded scenarios. The first three show the user gestures thumbs-up, fist, and the hand with spread fingers (*high -five* gesture). For simple user behavior (can also be seen as a gesture based on a time series of frames), we used the actions of putting on headphones and turning the monitor on/off (as seen by the arm reaching towards the power button). In the following we will name these two time dependent gestures simple behaviour.

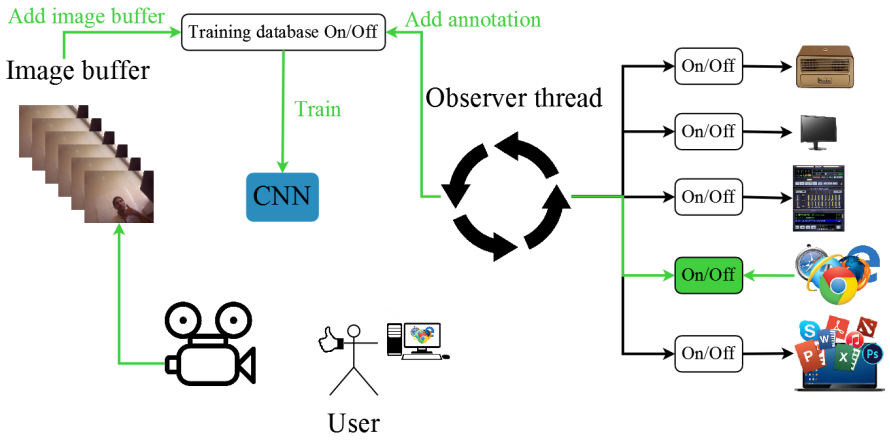

Fig. 4: Starting a browser while performing a gesture starts the data collection and training.

The first part of our system is the classification of simple behavior and gestures, which are shown in Figure 3. When the user wants to start an application that is assigned to a task or an action (On/Off box). He performs a gesture, the thumbs-up in Figure 3, which is captured by the camera. Each frame is stored in the image buffer. On each new image, the Convolutional Neuronal Network (CNN) classifies, based on a time window, if an action has to be performed. The action selected for the gesture thumbs-up in Figure 3 is turning the radio on.

The online training starts when a user toggles an observed action. In Figure 4, the user starts his browser and performs the thumbs-up gesture. The observer thread recognizes this state change and initiates the data collection and training. First, the current frame and its predecessors are combined into one input package (based on the time window size) together with the action number. This package is stored in the database as a valid example for this action. Forty-five additional valid examples are also created by shifting the current buffer index one frame backwards (1,5 Seconds). This means the first additional valid example goes one frame backwards in time and the second additional valid example two frames etc. The remaining images in the image buffer are also grouped based on the window size and added to the database as negative examples (do nothing class or class zero). For the time window size, we run the CNN in parallel (batch mode) and multiplied the probabilities (output of the last fully connected layer).

The online training starts after the collection of the new data samples. For data augmentation, we used 0–30% percent of noise, flipping, cropping and shifting the image up to 20% of the image width and height. Both values are determined randomly for each selected image in each iteration. Therefore, the CNN never sees the same image twice. For the batch generation, we computed

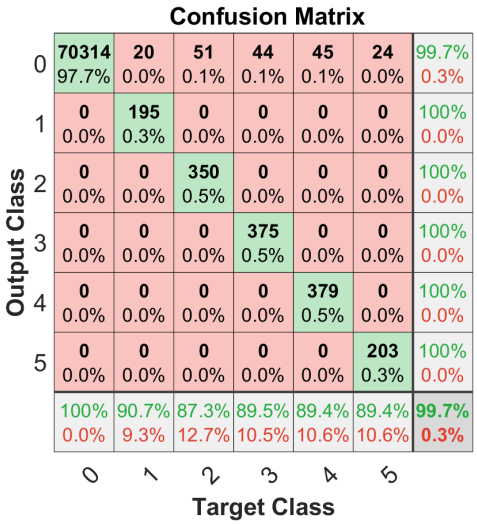

Fig. 5: Confusion matrix of the classification results with the proposed batch balancing for an initial training with four examples per class. The classes are the three gestures (fist (1), hand (2), and thumbs-up (3)) and the two movement sequences (turning the monitor on/off (4) and putting on the headphones (5)) in addition to the do nothing class 0.

the batch size based on the number of action classes (not the zero class). Each action class has always two valid examples per batch: So as to improve the generalization in comparison to just only one valid example per class. The same amount is added from the zero class (2× number of action classes). Therefore, for five action classes, we have a batch size of 20: Ten of the action classes and ten from the do nothing or zero class. In the following we refer to this structure of the batch as batch balancing. This batch creation was used to reduce misclassifications which are assigned to the wrong action class. This means that it is favored that our system does not perform an action instead of the wrong action. The fine tuning was performed with a learning rate of $1e^{-5}$. In addition, we set the learning rate of the convolution layers to 0. We used the ResNet34 (He et al., 2016) architecture pre-trained on ImagNet (Deng et al., 2009a) and replaced the last two fully connected (FC) layers. Therefore, our last layers are FC with 1024 neurons, a rectifier linear unit (ReLu) followed by the last FC with 6 neurons. The online training was stopped if the average loss value was saturated. Since the loss value for convolutional neuronal networks is shaky we smoothed it using a window function of five iterations. In addition, this value was multiplied by one hundred and then rounded to a whole number to avoid the floating point inaccuracy. Based on this signal the saturation was detected if three consecutive values are equal.

5 Evaluation

In this paper, we focus on perceptual learning from the computer's point of view. Meaning, the computer learns to execute an action by only receiving visual input and the status of the action. In the beginning the users trained the computer explicitly to execute an action by performing gestures in front of the camera and executed an action on the computer (opening an application, pressing a key etc.). Each user provided four examples for each type of action. The actions in our experiment are starting the WinAmp music player after putting on the headphones, turning the monitor on, showing a sad smiley (assigned to a fist gesture), playing a hello sound (hand gesture) and showing a happy smiley (thumbs-up gesture). Those examples where used to fine tune a Convolutional Neuronal Network (CNN) (LeCun et al., 1998; Yosinski et al., 2014) which was initially trained on ImageNet (Deng et al., 2009b). This fine tuning took $\approx$ 20 minutes for the initial training phase. After that the subject could do what they wanted for half an hour in front of the camera. This means that the users were still limited to the gestures and simple behavior to perform an action on the computer but they could start and use any application on the computer and perform the gestures/simple behavior in any order and at any time. The ground truth generation for each recording was performed by the user executing the action on the computer which was written to a CSV file. Our CNN was running in parallel writing the performed actions to an additional CSV file.

5.1 Experiment 1: Evaluation of the batch balancing

For the first evaluation we recorded ten test subjects with two sessions each. Each session lasted about one hour and included the training and the sample presentation as well as the half hour in front of the camera without restrictions. The results can be seen in the first confusion matrix in Figure 5. The CNN predicted each 500ms and the input time window was therefore set to 15 frames. All recording session where aligned to 30 minutes at 30 fps by removing the last frames of the video. As can be seen in Figure 5 wrong predictions are only done to the class zero which is the do nothing class. This means that the top row in Figure 5 represents all predictions to the do nothing class. As can be seen 20 examples of the action class 1 are wrongly predicted as the do nothing class.

In comparison to this Figure 6 shows the results without our batch balancing (50% of a batch consisted of do nothing class examples the other 50% of the batch where randomly chosen from action classes with two examples per action class). As can be seen the wrong predictions to the do nothing class are less compared to our batch balancing approach. However, there are misclassification between the action classes which lead to malfunction. In the second row (action class one) it can be seen that the first action is executed 41 times for the do nothing class as well as once for action 2 and twice for action 3. For

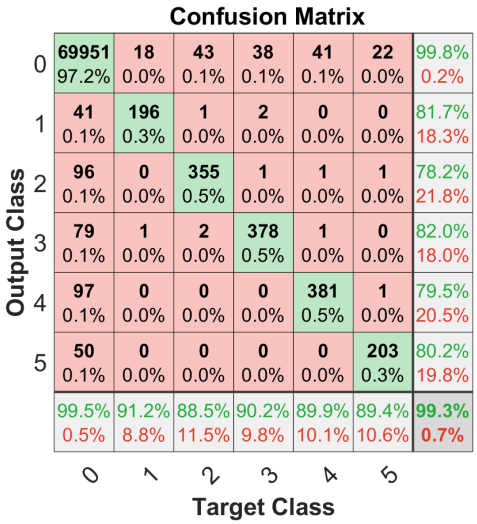

Fig. 6: Confusion matrix of the classification results without batch balancing for an initial training with four examples per class. The classes are the three gestures (fist (1), hand (2), and thumbs-up (3)) and the two movement sequences (turning the monitor on/off (4) and putting on the headphones (5)) in addition to the do nothing class 0.

a user, this malfunction is very unpleasant, as unwanted actions are carried out. In comparison, it is better if the program does nothing and the user can repeat his gesture.

Since the repetition of gestures is also unpleasant if it has to be done too often, our system adapts itself. As an example we assume that the user executes the gesture for action 1 which is not detected by our system. The user then opens the sad smiley image. Our setup recognizes that an observed action was performed which was not recognized by the system. Therefore, new training samples are generated as described in Section 4 and the CNN is adapted online. This example brings us to our second experiment which is the online adaption.

5.2 Experiment 2: Evaluation of the adaption

For the online adaption we repeated the experiment with all ten subjects and two sessions per subject. This time we used the initial model from the first experiment and recorded two additional examples per action class. The training reduced from the initial ≈ 20 minutes to ≈ 1 minute. As can be seen in Figure 7 the results improved in comparison to Figure 5 again without wrong action executions. This means that all misclassifications are wrongly assigned

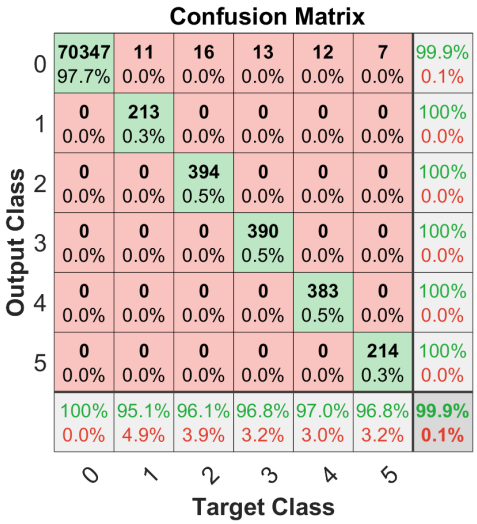

Fig. 7: Confusion matrix of the classification results after an online adaption with two additional examples per class and our batch balancing approach. The classes are the three gestures (fist (1), hand (2), and thumbs-up (3)) and the two movement sequences (turning the monitor on/off (4) and putting on the headphones (5)) in addition to the do nothing class 0.

to the do nothing class 0 (Top row in Figure 7) and no misclassification was assigned to an action class.

In these two experiments we have proven the functionality of our approach but an application in everyday life is more challenging. An important challenge is to ensure functionality in different environments. In the previous two experiments, the environment was always an office (Figure 2), which changes in the next experiment. Here we use the initially trained models from the first experiment (Figure 5) and test them on a balcony as environment. We used the same ten subjects and recorded two sessions per subject. This time one recording took ≈ 30 minutes since no examples had to be given.

5.3 Experiment 3: Evaluation in a new environment

As can be seen in Figure 8 the classification results decrease. Each action is recognized only at half of the time which is uncomfortable for the user. Since our initial model was only trained in one environment these results are expected. As in the previous experiments in which the training was performed with our batch balancing strategy, the misclassifications are always assigned to the do nothing class. Therefore, our system does not perform an unwanted action. In addition, if the user performs an observed action our system is able to adapt. This leads to the fourth experiment in which the user provides two

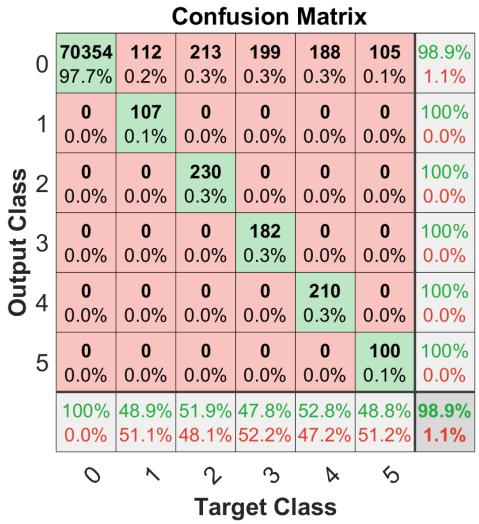

Fig. 8: Confusion matrix of the classification results for the initially trained model in a new environment with our batch balancing. The classes are the three gestures (fist (1), hand (2), and thumbs-up (3)) and the two movement sequences (turning the monitor on/off (4) and putting on the headphones (5)) in addition to the do nothing class 0.

examples for each action at the beginning and the system has to adapt to the new environment.

5.4 Experiment 4: Evaluation of the adaption to a new environment

For the online adaption in the new environment we repeated the recordings (ten subjects and two recordings per subject). This time each subject recorded two examples per action class and the model was trained for ≈ 1 minute. As can be seen in Figure 9 the classification results significantly improved for each action class. In addition, no misclassification was assigned to an action class. Therefore, our approach can effectively adapt to new environments.

So far we showed that our batch balancing approach effectively avoids the execution of an invalid action class (Comparison of Figure 5 and Figure 6) and that the online adaption with the proposed data collection improves the result (Figure 7 and Figure 9). This is also true for new environments (Comparison of Figure 8 and Figure 9).

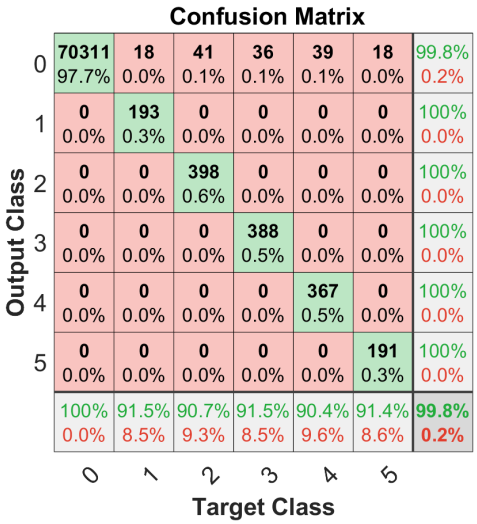

Fig. 9: Confusion matrix of the classification results for the adapted model in a new environment with our batch balancing. The classes are the three gestures (fist (1), hand (2), and thumbs-up (3)) and the two movement sequences (turning the monitor on/off (4) and putting on the headphones (5)) in addition to the do nothing class 0.

5.5 Experiment 5: Online usage evaluation

Still missing is if we can perform our adaption online, parallel to the classification. Therefore, we designed the fifth experiment. In this experiment we used two GPUs one for the classification and one for the online training. The online training is performed if there exist at least two misclassifications (The do nothing class was performed but the user executes the observed action). After the new model is trained it replaces the old model which is still used during the training time. Again we recorded the ten subjects with two sessions per subject. During each recording the subjects could do whatever they wanted and also move the table on which the PC and the camera are located. Therefore, we put the table on a rolling board with a cable reel for power supply. Initially this table was placed in a kitchen as starting location before each recording. As initial model for the classification we used those trained in the first experiment (Figure 5).

Figure 10 shows the results for the online experiment. On top the not recognized actions for each recording per minute are visualized. As can be seen, the system does not recognize many actions at the beginning. From minute 9 it runs very stable with a few drops in the detection rate. These are caused by new room changes. An example of this is recording 14. As can be seen in Figure 10, the room changes take place in minutes 15 and 20. This is

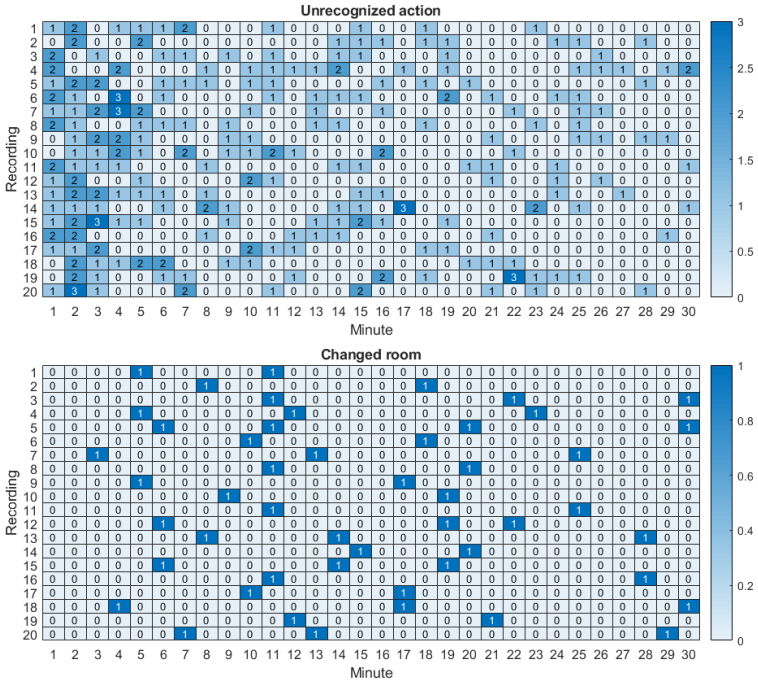

Fig. 10: The top plot shows the amount of not recognized actions for each recording and every minute. On the bottom plot the room changes per recording are shown. One means that in this minute the room change has started.

followed by a decrease in the detection rate in minutes 17 and 23. Another good example is recording 18, where a relatively early room change takes place in minute 4 (Figure 10). This is followed by a direct drop in the detection rate. As the recording progresses, the room is changed in minute 17. Here one does not see a drop in the detection rate, since the system has already adapted very well and already knows the new room. All in all, it can be seen in Figure 10 that the system is constantly improving. The central part of the results show that the system is also no longer as prone to room changes as in the beginning of the experiment. In addition to the results in Figure 10, there were no wrongly performed actions in all recordings.

The number of training phases can be seen in Figure 11 on the right plot. Please note that at least two unrecognized actions had to be present before a training phase could be started. As can be seen, each shot had a minimum of five training phases and a maximum of nine training phases. Since the training database increases in each training phase, it is also interesting to see how this affects the duration of the training phases. This can be seen in Figure 11 on the left plot. The y axis corresponds here to the average duration of a training phase in seconds and the x axis to the training phase number. It can be seen that the duration moves around the mean value of one minute, which

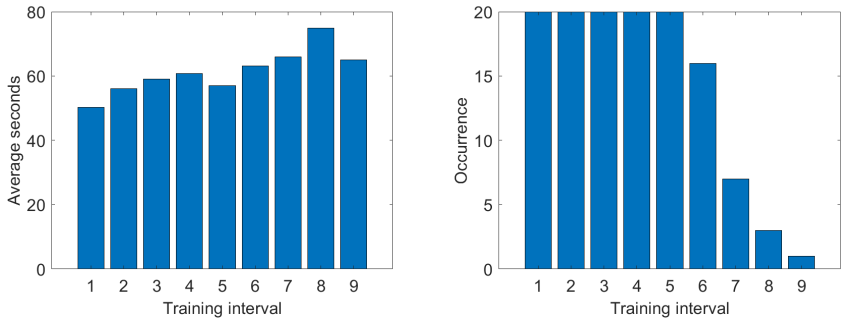

Fig. 11: The left plot shows the average training time per training-phase and on the right the amount of training-phase occurred over all recordings are shown.

increases slightly compared to the first training phase. This behavior has to be investigated in more detail in longer recordings as well as the challenge of the constantly growing training database. This is discussed in more detail in Section 8.

6 Runtime and delay

The runtime of our ResNet-34 on a NVIDIA 1050ti card is 89ms per batch (15 images). Since we only classify every half second, a delay of 589ms can occur between a gesture and an action. Of course, this is not optimal because it can be perceived by the user. In contrast to this, a smaller window leads to a more frequent use of the GPU, which in the case of a mobile device like a laptop leads to a reduced battery life. Finding an optimal window requires further experiments and depends on the field of application. This is beyond the scope of this work and will be investigated in future research.

7 Perspectives of adaptive learning

In this section, we want to show the possibilities that adaptive learning brings for the usability of applications. The first would be to give users the opportunity to improve a system as much as they can. There are already many applications like Alexa from Amazon, Google Home, gesture control for smartphones etc. but all have the disadvantage that in case of misclassification or incomprehensible input the user can do nothing but repeat them over and over again. Our approach offers a remedy and leaves it to the user to further improve the system and adapt it to himself. A further disadvantage of the already existing applications is that they cannot be adapted arbitrarily. An example of this would be a non-integrated language in a voice control system or an unfeasible

gesture for a user. Our system allows to learn any gestures and in case of a voice control, which is not evaluated in this work, our approach would support any word combination even without being able to understand the language itself. Of course, our system is not comparable to applications that have users all over the world, but we believe that our approach can improve existing systems. This is especially true for users who suffer from restrictions as well as for users who are not supported by the system due to local conditions such as a dialect. Since our system also allows to personalize the human computer interaction, the size of the machine learning model used could be reduced and thus a better runtime in addition to improved classification would be possible. This is due to the fact that the model no longer has to support all possible users in the world, but only the local user group.

8 Limitations

The first limitation of our experiments was already mentioned in Section 6, which concerns the fixed time window of 500ms. For optimum time windows, especially with regard to the application and the device used for evaluation, further experiments must be carried out. Another interesting application of our system would be the use of several users with the same model. Here one would have to either make an identification before or carry out new experiments which analyze the use with several users. Another challenge in terms of using our system in everyday life would be to use it in outdoor areas such as a park or a street on a bench. In addition, gestures that are very similar to each other must also be considered. This could be compensated by a higher input resolution in case of an error, but would result in a longer runtime. In the last experiment, a long-term analysis was also mentioned. This is particularly interesting if the user changes buildings or walks around in nature. This would clearly show the usability of the system in everyday life and would be the final step before a commercial application.

The long-term application itself also provides new challenges for our system. The first big challenge would be to limit the training database. It is not possible in a real system to store a constantly growing amount of data. One solutions here could be the use of a server but this also creates data protection challenges and also requires a stable network connection to the server. An advantage however would be that not two GPUs are necessary in order to allow the adaptation of the system. Under a limited amount of Classes and a modern GPU it is also possible to evaluate and train on a single GPU.

There are further challenges of the system which have to be evaluated but exceed the scope of this work. The last FC layer which limits the number of classes that can be learned is one of those challenges. This means, that if the last fully connected layer has 100 neurons, the model can only observe 99 actions and therefore learn 99 gestures (since one neuron is required for "no action"). This also affects the batch size for our batch creation strategy and increases the memory requirements on the GPU. In the case of the server

solution this would not be a problem, but in the case of a purely local execution of the system this is not possible indefinitely. Additional challenges would also come from the different clothing of the user, wearing glass, or changing hair style as well as changing environments. These challenges could lead to the need for larger models.

9 Conclusion

We proposed a framework which can be trained by the user. It is capable of learning gestures on its own to perform human computer interaction. The user is also able to train the framework directly by examples. We conducted an experiment to show the efficiency of our batch balancing approach. In addition, we showed that our system is able to adapt to new environments online where each challenge was additionally evaluated in independent experiments. Based on the results as well as the runtime of our system, the remaining limitations were pointed out and further possibilities for research were discussed. Possible fields of application and the improvement of existing software are also discussed.

References

Abbeel P, Ng AY (2004) Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the twenty-first international conference on Machine learning, ACM, p 1

Alastalo AT, Kaajakari V (2005) Intermodulation in capacitively coupled microelectromechanical filters. IEEE electron device letters 26(5):289–291

Arce F, Valdez JMG (2010) Accelerometer-based hand gesture recognition using artificial neural networks. In: Soft Computing for Intelligent Control and Mobile Robotics, Springer, pp 67–77

Ausubel DP, Novak JD, Hanesian H, et al. (1968) Educational psychology: A cognitive view, vol 6. Holt, Rinehart and Winston New York

Bandura A, Ross D, Ross SA (1961) Transmission of aggression through imitation of aggressive models. The Journal of Abnormal and Social Psychology 63(3):575

Basanta H, Huang YP, Lee TT (2017) Using voice and gesture to control living space for the elderly people. In: System Science and Engineering (ICSSE), 2017 International Conference on, IEEE, pp 20–23

Bloom BS (1976) Human characteristics and school learning. McGraw-Hill

Chen Q, Georganas ND, Petriu EM, et al. (2007) Real-time vision-based hand gesture recognition using haar-like features. In: Instrumentation and Measurement Technology Conference Proceedings, Citeseer, pp 1–6

Corera S, Krishnarajah N (2011) Capturing hand gesture movement: a survey on tools, techniques and logical considerations. Proceedings of chi sparks

Dardas NH, Georganas ND (2011) Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. IEEE Transactions on Instrumentation and measurement 60(11):3592–3607

Deng J, Dong W, Socher R, jia Li L, Li K, Fei-fei L (2009a) Imagenet: A large-scale hierarchical image database. In: Proceedings of the IEEE conference on computer vision and pattern recognition

Deng J, Dong W, Socher R, jia Li L, Li K, Fei-fei L (2009b) Imagenet: A large-scale hierarchical image database. In: In CVPR

Dixit DSK, Shingi MNS (2012) Implementation of flex sensor and electronic compass for hand gesture based wireless automation of material handling robot. International Journal of Scientific and Research Publications 2(12):1

Dragan AD, Srinivasa SS (2012) Online customization of teleoperation interfaces. In: RO-MAN, 2012 IEEE, IEEE, pp 919–924

Francke H, Ruiz-del Solar J, Verschae R (2007) Real-time hand gesture detection and recognition using boosted classifiers and active learning. In: Pacific-Rim Symposium on Image and Video Technology, Springer, pp 533–547

Gidaris S, Komodakis N (2018) Dynamic few-shot visual learning without forgetting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 4367–4375

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778

Hoo-Chang S, Roth HR, Gao M, Lu L, Xu Z, Nogues I, Yao J, Mollura D, Summers RM (2016) Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. IEEE transactions on medical imaging 35(5):1285

Hussein A, Gaber MM, Elyan E, Jayne C (2017) Imitation learning: A survey of learning methods. ACM Computing Surveys (CSUR) 50(2):21

Ijspeert AJ, Nakanishi J, Schaal S (2002) Movement imitation with nonlinear dynamical systems in humanoid robots. In: Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on, IEEE, vol 2, pp 1398–1403

Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105

Lamberti L, Camastra F (2011) Real-time hand gesture recognition using a color glove. In: International Conference on Image Analysis and Processing, Springer, pp 365–373

LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11):2278–2324

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. nature 521(7553):436

Li Y, Cao Z, Wang J (2017) Gazture: Design and implementation of a gaze based gesture control system on tablets. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 1(3):74

Li Z, Jarvis R (2009) Real time hand gesture recognition using a range camera. In: Australasian Conference on Robotics and Automation, pp 21–27

Liu Y, Gupta A, Abbeel P, Levine S (2018) Imitation from observation: Learning to imitate behaviors from raw video via context translation. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp 1118–1125

Nielsen M, Störring M, Moeslund TB, Granum E (2003) A procedure for developing intuitive and ergonomic gesture interfaces for hci. In: International gesture workshop, Springer, pp 409–420

Pandit A, Dand D, Mehta S, Sabesan S, Daftery A (2009) A simple wearable hand gesture recognition device using imems. In: Soft Computing and Pattern Recognition, 2009. SOCPAR'09. International Conference of, IEEE, pp 592–597

Pomerleau DA (1991) Efficient training of artificial neural networks for autonomous navigation. Neural Computation 3(1):88–97

Rao SM, Gagie B (2006) Learning through seeing and doing: Visual supports for children with autism. Teaching Exceptional Children 38(6):26–33

Ross S, Gordon G, Bagnell D (2011) A reduction of imitation learning and structured prediction to no-regret online learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics, pp 627–635

Schaal S (1999) Is imitation learning the route to humanoid robots? Trends in cognitive sciences 3(6):233–242

Simpson RC, Levine SP (2002) Voice control of a powered wheelchair. IEEE Transactions on neural systems and rehabilitation engineering 10(2):122–125

Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2818–2826

Wang RY, Popović J (2009) Real-time hand-tracking with a color glove. ACM transactions on graphics (TOG) 28(3):63

Yang J, Xu Y, Chen CS (1997) Human action learning via hidden markov model. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 27(1):34–44

Yosinski J, Clune J, Bengio Y, Lipson H (2014) How transferable are features in deep neural networks? In: Advances in neural information processing systems, pp 3320–3328