From perception to action using observed actions to learn gestures

Wolfgang Fuhl¹

Received: 17 October 2019 / Accepted in revised form: 8 August 2020 / Published online: 24 August 2020
© The Author(s) 2020

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

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 interaction techniques: Some are keyboard, touch, voice, mouse, gestures, or gaze Fuhl et al. (2016, (2017a, (2017b, (2018b)). 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.

✉ Wolfgang Fuhl
wolfgang.fuhl@uni-tuebingen.de

¹ Eberhard Karls Universität Tübingen, Sand 14, Tübingen, Germany

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.

The human being is capable of learning from observations 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 application, 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 Paramythis et al. (2010).

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 is a type of interaction in which the computer uses a video or motion source. Here, we summarize the work that has already been 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) and 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 actions it is able to reproduce.

2.2 Learning from observations

Research regarding observational learning also addresses 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 Fuhl et al. (2018a). 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 consist 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 ImageNet (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.
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 1 shows five recorded scenarios. The first three shows 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

Fig. 1 Five different user movements of the same subject using our web camera

Fig. 2 Performing an action based on the thumbs-up gesture

Fig. 3 Starting a browser while performing a gesture starts the data collection and training

arm reaching toward the power button). In the following, we will name these two time-dependent gestures simple behavior.

The first part of our system is the classification of simple behavior and gestures, which are shown in Fig. 2. 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 Fig. 2, 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 Fig. 2 is turning the radio on.

The online training starts when a user toggles an observed action. In Fig. 3, 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 backward (1,5 Seconds). This means the first additional valid example goes one frame backward 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 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 with just only one valid example per class. The same amount is added from the zero class ($2 \times$ 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 ImageNet (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 were 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 ≈ 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 Fig. 4. The CNN predicted each 500ms and the input time window was therefore set to 15 frames. All recording session were aligned to 30 min at 30 fps by removing the last frames of the video. As can be seen in Fig. 4, wrong predictions are only done to the class zero which is the do nothing class. This means that the top row in Fig. 4 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 with this Fig. 5 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 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.

Fig. 4 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

	0	1	2	3	4	5	
0	70314 97.7%	20 0.0%	51 0.1%	44 0.1%	45 0.1%	24 0.0%	99.7% 0.3%
1	0 0.0%	195 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	0 0.0%	350 0.5%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	0 0.0%	375 0.5%	0 0.0%	0 0.0%	100% 0.0%
4	0 0.0%	0 0.0%	0 0.0%	0 0.0%	379 0.5%	0 0.0%	100% 0.0%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	203 0.3%	100% 0.0%
	100% 0.0%	90.7% 9.3%	87.3% 12.7%	89.5% 10.5%	89.4% 10.6%	89.4% 10.6%	99.7% 0.3%
	0	1	2	3	4	5	
Output Class	Target Class						

Fig. 5 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

		Confusion Matrix						
Output Class		0	1	2	3	4	5	
		69951 97.2%	18 0.0%	43 0.1%	38 0.1%	41 0.1%	22 0.0%	99.8% 0.2%
1		41 0.1%	196 0.3%	1 0.0%	2 0.0%	0 0.0%	0 0.0%	81.7% 18.3%
2		96 0.1%	0 0.0%	355 0.5%	1 0.0%	1 0.0%	0 0.0%	78.2% 21.8%
3		79 0.1%	1 0.0%	2 0.0%	378 0.5%	1 0.0%	0 0.0%	82.0% 18.0%
4		97 0.1%	0 0.0%	0 0.0%	0 0.0%	381 0.5%	1 0.0%	79.5% 20.5%
5		50 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	203 0.3%	80.2% 19.8%
		99.5% 0.5%	91.2% 8.8%	88.5% 11.5%	90.2% 9.8%	89.9% 10.1%	89.4% 10.6%	99.3% 0.7%
		0	1	2	3	4	5	
		Target Class						

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 Sect. 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 Fig. 6, the results improved in comparison with Fig. 4 again without wrong action executions. This means that

Fig. 6 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

		Confusion Matrix						
Output Class		0	1	2	3	4	5	
		70347 97.7%	11 0.0%	16 0.0%	13 0.0%	12 0.0%	7 0.0%	99.9% 0.1%
1		0 0.0%	213 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
2		0 0.0%	0 0.0%	394 0.5%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
3		0 0.0%	0 0.0%	0 0.0%	390 0.5%	0 0.0%	0 0.0%	100% 0.0%
4		0 0.0%	0 0.0%	0 0.0%	0 0.0%	383 0.5%	0 0.0%	100% 0.0%
5		0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	214 0.3%	100% 0.0%
		100% 0.0%	95.1% 4.9%	96.1% 3.9%	96.8% 3.2%	97.0% 3.0%	96.8% 3.2%	99.9% 0.1%
		0	1	2	3	4	5	
		Target Class						

all misclassifications are wrongly assigned to the do nothing class 0 (Top row in Fig. 6), 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 (Fig. 1), which changes in the next experiment. Here, we use the initially trained models from the first experiment (Fig. 4) 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 Fig. 7, 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 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 Fig. 8, the classification results significantly improved for each action class. In addition, no

Fig. 7 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

	0	1	2	3	4	5	
0	70354 97.7%	112 0.2%	213 0.3%	199 0.3%	188 0.3%	105 0.1%	98.9%
1	0 0.0%	107 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100%
2	0 0.0%	0 0.0%	230 0.3%	0 0.0%	0 0.0%	0 0.0%	100%
3	0 0.0%	0 0.0%	0 0.0%	182 0.3%	0 0.0%	0 0.0%	100%
4	0 0.0%	0 0.0%	0 0.0%	0 0.0%	210 0.3%	0 0.0%	100%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100 0.1%	100%
	100%	48.9%	51.9%	47.8%	52.8%	48.8%	98.9%
	0.0%	51.1%	48.1%	52.2%	47.2%	51.2%	1.1%
	0	1	2	3	4	5	
	Target Class						

Fig. 8 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

		Confusion Matrix						
Output Class	0	70311 97.7%	18 0.0%	41 0.1%	36 0.1%	39 0.1%	18 0.0%	99.8% 0.2%
	1	0 0.0%	193 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
	2	0 0.0%	0 0.0%	398 0.6%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
	3	0 0.0%	0 0.0%	0 0.0%	388 0.5%	0 0.0%	0 0.0%	100% 0.0%
	4	0 0.0%	0 0.0%	0 0.0%	0 0.0%	367 0.5%	0 0.0%	100% 0.0%
	5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	191 0.3%	100% 0.0%
		100% 0.0%	91.5% 8.5%	90.7% 9.3%	91.5% 8.5%	90.4% 9.6%	91.4% 8.6%	99.8% 0.2%
		0	1	2	3	4	5	
		Target Class						

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 Figs. 4 and 5) and that the online adaption with the proposed data collection improves the result (Figs. 6 and 8). This is also true for new environments (Comparison of Figs. 7 and 8).

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 (Fig. 4).

Figure 9 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 Fig. 9, the room changes take place in minutes 15 and 20. This is 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 (Fig. 9). 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

Fig. 9 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

and already knows the new room. All in all, it can be seen in Fig. 9 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 Fig. 9, there were no wrongly performed actions in all recordings.

The number of training phases can be seen in Fig. 10 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 Fig. 10 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 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 Sect. 8.

Fig. 10 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

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 Sect. 6, which concerns the fixed time window of 500 ms. 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 solution 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.

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

Funding Open Access funding provided by Projekt DEAL.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the Twenty-First International Conference on Machine Learning. ACM, p. 1 (2004)
- Alastalo, A.T., Kaajakari, V.: Intermodulation in capacitively coupled microelectromechanical filters. *IEEE Electron Device Lett.* **26**(5), 289–291 (2005)
- Arce, F., Valdez, J.M.G.: Accelerometer-based hand gesture recognition using artificial neural networks. In: *Soft Computing for Intelligent Control and Mobile Robotics*. Springer, pp. 67–77 (2010)
- Ausubel, D.P., Novak, J.D., Hanesian, H., et al.: *Educational Psychology: A Cognitive View*, vol. 6. Holt, Rinehart and Winston, New York (1968)
- Basanta, H., Huang, Y.P., Lee, T.T.: Using voice and gesture to control living space for the elderly people. In: *2017 International Conference on System Science and Engineering (ICSSE)*. IEEE, pp. 20–23 (2017)
- Bloom, B.S.: *Human Characteristics and School Learning*. McGraw-Hill, New York (1976)
- Chen, Q., Georganas, N.D., Petriu, E.M., et al.: Real-time vision-based hand gesture recognition using haar-like features. In: *Instrumentation and Measurement Technology Conference Proceedings*. Citeseer, pp. 1–6 (2007)
- Corera, S., Krishnarajah, N.: Capturing hand gesture movement: a survey on tools, techniques and logical considerations. *Proceedings of chi sparks* (2011)
- Dardas, N.H., Georganas, N.D.: Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. *IEEE Trans. Instrum. Meas.* **60**(11), 3592–3607 (2011)
- Deng, J., Dong, W., Socher, R., Jia, L., Li, K., Fei-fei, L.: Imagenet: a large-scale hierarchical image database. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2009a)*
- Deng, J., Dong, W., Socher, R., Jia, L., Li, K., Fei-fei, L.: Imagenet: a large-scale hierarchical image database. In: *CVPR (2009b)*
- Dixit, D.S.K., Shingi, M.N.S.: Implementation of flex sensor and electronic compass for hand gesture based wireless automation of material handling robot. *Int. J. Sc. Res. Publ.* **2**(12), 1 (2012)
- Dragan, A.D., Srinivasa, S.S.: Online customization of teleoperation interfaces. In: *RO-MAN, 2012 IEEE*, IEEE, pp. 919–924 (2012)
- Francke, H., Ruiz-del Solar, J., Verschae, R.: 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 (2007)
- Fuhl, W., Tonsen, M., Bulling, A., Kasneci, E.: Pupil detection in the wild: an evaluation of the state of the art in mobile head-mounted eye tracking. *Mach. Vis. Appl.* **27**, 1275–1288 (2016)

- Fuhl, W., Santini, T., Kasneci, E.: Fast and robust eyelid outline and aperture detection in real-world scenarios. In: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV). IEEE, pp. 1089–1097 (2017a)
- Fuhl, W., Santini, T., Kasneci, E.: Fast camera focus estimation for gaze-based focus control. (2017b). arXiv preprint arXiv:1711.03306
- Fuhl, W., Castner, N., Zhuang, L., Holzer, M., Rosenstiel, W., Kasneci, E.: Mam: Transfer learning for fully automatic video annotation and specialized detector creation. In: Proceedings of the European Conference on Computer Vision (ECCV) (2018a)
- Fuhl, W., Eivazi, S., Hosp, B., Eivazi, A., Rosenstiel, W., Kasneci, E.: Bore: boosted-oriented edge optimization for robust, real time remote pupil center detection. In: Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications, pp. 1–5 (2018b)
- Gidaris, S., Komodakis, N.: Dynamic few-shot visual learning without forgetting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4367–4375 (2018)
- He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
- Hoo-Chang, S., Roth, H.R., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D., Summers, R.M.: Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE Trans. Med. Imaging* **35**(5), 1285 (2016)
- Hussein, A., Gaber, M.M., Elyan, E., Jayne, C.: Imitation learning: a survey of learning methods. *ACM Comput. Surv. (CSUR)* **50**(2), 21 (2017)
- Ijspeert, A.J., Nakanishi, J., Schaal, S.: Movement imitation with nonlinear dynamical systems in humanoid robots. In: Proceedings of IEEE International Conference on Robotics and Automation, 2002. ICRA'02, vol 2. IEEE, pp. 1398–1403 (2002)
- Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
- Lamberti, L., Camastra, F.: Real-time hand gesture recognition using a color glove. In: International Conference on Image Analysis and Processing. Springer, pp. 365–373 (2011)
- LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proc. IEEE* **86**(11), 2278–2324 (1998)
- LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. *Nature* **521**(7553), 436 (2015)
- Li, Z., Jarvis, R.: Real time hand gesture recognition using a range camera. In: Australasian Conference on Robotics and Automation, pp. 21–27 (2009)
- Li, Y., Cao, Z., Wang, J.: Gazture: design and implementation of a gaze based gesture control system on tablets. In: Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, vol. 1, no. 3, p. 74 (2017)
- Liu, Y., Gupta, A., Abbeel, P., Levine, S.: 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 (2018)
- Nielsen, M., Störring, M., Moeslund, T.B., Granum, E.: A procedure for developing intuitive and ergonomic gesture interfaces for HCI. In: International Gesture Workshop. Springer, pp. 409–420 (2003)
- Pandit, A., Dand, D., Mehta, S., Sabesan, S., Daftary, A.: A simple wearable hand gesture recognition device using imems. In: International Conference of Soft Computing and Pattern Recognition, 2009. SOC-PAR'09. IEEE, pp. 592–597 (2009)
- Paramythias, A., Weibelzahl, S., Masthoff, J.: Layered evaluation of interactive adaptive systems: framework and formative methods. *User Model. User Adap. Int.* **20**(5), 383–453 (2010)
- Pomerleau, D.A.: Efficient training of artificial neural networks for autonomous navigation. *Neural Comput.* **3**(1), 88–97 (1991)
- Rao, S.M., Gagie, B.: Learning through seeing and doing: visual supports for children with autism. *Teach. Except. Child.* **38**(6), 26–33 (2006)
- Ross, S., Gordon, G., Bagnell, D.: 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 (2011)
- Schaal, S.: Is imitation learning the route to humanoid robots? *Trends Cogn. Sci.* **3**(6), 233–242 (1999)
- Simpson, R.C., Levine, S.P.: Voice control of a powered wheelchair. *IEEE Trans. Neural Syst. Rehabil. Eng.* **10**(2), 122–125 (2002)
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826 (2016)

- Wang, R.Y., Popović, J.: Real-time hand-tracking with a color glove. *ACM TOG* **28**(3), 63 (2009)
- Yang, J., Xu, Y., Chen, C.S.: Human action learning via hidden Markov model. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **27**(1), 34–44 (1997)
- Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: *Advances in Neural Information Processing Systems*, pp. 3320–3328 (2014)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Wolfgang Fuhl was born in Freudenstadt, Germany in 1983. He received the Dipl.-FH degree in computer science, M.Sc. degree in computer science and his Phd from the University of Tübingen Germany, in 2011, 2014, and 2019 respectively. Currently he is a Postdoc at the University of Tübingen Germany in the Human Computer Interaction groupe. His research focusses on the methods of machine learning as well as computer vision. He is also interested in the application of machine learning methods to industrial challenges and in the field of human computer interaction.