Introducing and Benchmarking a One-Shot Learning Gesture Recognition Dataset

Panagiotis Kasnesis(✉), Christos Chatzigeorgiou, Charalampos Z. Patrikakis, and Maria Rangoussi

Department of Electrical and Electronics Engineering, University of West Attica, Egaleo, Greece
{pkasnesis,chrihatz,bpatr,mariar}@uniwa.gr

Abstract. Deep learning techniques have been widely and successfully applied, over the last five years, to recognize the gestures and activities performed by users wearing electronic devices. However, the collected datasets are built in an old fashioned way, mostly comprised of subjects that perform many times few different gestures/activities. This paper addresses the lack of a wearable gesture recognition dataset for exploring one-shot learning techniques. The current dataset consists of 46 gestures performed by 35 subjects, wearing a smartwatch equipped with 3 motion sensors and is publicly available. Moreover, 3 one-shot learning classification approaches are benchmarked on the dataset, exploiting two different deep learning classifiers. The results of the benchmark depict the difficulty of the one-shot learning task, exposing new challenges for wearable gesture/activity recognition.

Keywords: Datasets · Deep learning · Wearable gesture recognition · One-shot learning

1 Introduction

Gestures are a very natural and intuitive way for humans both to interact with an electronic device (e.g., TV) and to control it [18]. As a result, gesture-based interfaces based on arm, hand or finger motions are becoming increasingly popular [9]. To this end, wrist-worn devices are widely used for recording motion data signals; they are effective and unobtrusive devices which have been used successfully in several domains, such as smart home environments [16], smart factory [26], physical security [11]. They are also exploited for effective gesture-based human-machine interaction [5,23,31].

In order to accurately recognize the body movements and translate the results into commands, after having established a one-to-one correspondence between gestures and commands to action, these gesture-based interfaces are based on machine learning algorithms [11]. In particular, Deep Learning (DL) algorithms, such as Convolutional Neural Networks (CNNs; ConvNets) and Long-Short Term

Z. Deze et al. (Eds.): BDTA 2020/WiCON 2020, LNICST 371, pp. 104–116, 2021.
https://doi.org/10.1007/978-3-030-72802-1_8

Memory (LSTM) have been efficiently applied to human gesture/activity recognition over the last five years [12,20,22,25]. However, these algorithms demand large datasets to be trained on; moreover, only few works have examined their knowledge transfer capabilities [1,21,29], especially when it comes to one-shot learning [8].

A disadvantage of existing approaches is that the algorithms have to be trained on large source datasets, such as those reported in [4,24], before evaluating their knowledge on the target dataset. On the other hand, although, there do exist publicly available small datasets [7,17] that contain dozens of different gestures, the wrist-worn devices used for gesture monitoring suffered from low sampling rate (i.e., 10 Hz) and time windows (i.e., 1 s), making it almost impossible to train a DL model on them. Moreover, the scientific interest in applying neural networks to small-data tasks has been increased the last two years [3,10], where new deep learning paradigms, such as Neural Tangent Kernel, have been proposed. As a results, a standard benchmark for learning from few examples, such as Omniglot [15], is therefore lacking in the human activity/gesture recognition domain.

In the present paper, we introduce a one-Shot lEaNning geSture rEcognition Dataset[1] (SENSED); it is a dataset consisting of 46 gestures, including existing English characters, numbers and symbols performed (hand-written on a surface) only once by 35 subjects, who are wearing a commercial smartwatch. SENSED has been deliberately designed to be small to avoid arbitrary downsampling of existing large datasets, and is split into train, validation and test sets to enable a fair comparison between the machine learning algorithms [2]. Moreover, SENSED is benchmarked by 3 one-shot learning classification approaches. In particular, the contribution and innovation of this work is summarized in the following:

1. To the best or our knowledge, SENSED is the first wearable-based gesture recognition dataset particularly built for one-shot learning.
2. Three deep learning approaches for one-shot learning were applied to SENSED with promising results.
3. The collected dataset is made publicly available, to be used as a benchmark for one-shot learning sensor-based gesture recognition.

The rest of this paper is organized as follows. Section 2 elaborates on the data collection procedure, while Sect. 3 describes in detail all the investigated one-shot learning network architectures. Section 4 describes the experimental setup for data processing and Sect. 5 presents and discusses the obtained results of our experiments. Finally, Sect. 6 concludes the paper and proposes future research steps.

2 Dataset Collection

The dataset is designed to contain a total of 46 characters. The first 26 are all the English capital letters, i.e., A to Z. The next ten include all the ten digits ranging

[1] https://github.com/ounospanas/sensed.

from 0 to 9. The last ten characters include some commonly used characters and math symbols: @, $, #, €, ?, !, %, *, + and =. We selected these type of gestures to be included since character and digit recognition could be useful in several domains, such as producing short texts messages, converting dynamically handwritten texts to online documents, entering security passwords without the use of a physical interface and many others. The English capitals letters were used as the train set, the numbers were used for creating the validation set and the special characters were used as the test set. In total, 35 subjects participated in the creation of the dataset and performed the gestures only once. However, not all subjects performed all characters. 25 subjects wrote the English letters, 5 subjects wrote the digits and 5 subjects wrote the remaining special characters. The above information is summarized in Table 1.

The device that was chosen for gathering the data, is the Fossil Gen 5[2]. It runs on a Snapdragon 3100 chipset[3], which is currently the latest chipset from Qualcomm for smartwatches, features 1 GB of RAM and 8 GB of storage. It comes equipped with accelerometer, gyroscope, magnetometer and heart rate sensor. For communication with other devices, it can use either Bluetooth or Wi-Fi. The hardware is managed by Wear OS[4] by Google, a modified version of Android for smartwatches and wearables. Although Wear OS targets devices with more limited hardware resources than smartphones, it has the same Application Programming Interface (API) as Android making the development process the same. The sensors that were used for creating the dataset were the 3-axial accelerometer and gyroscope. Each sensor was set to its maximum sampling rate allowed by the OS which is limited 50 Hz. For each gesture, the device captured a time window of 3 s of data (150 samples), which was enough for the subjects to perform gestures that involved a lot of hand movement.

Table 1. Dataset contents and participating subjects

Set	Characters	Subjects	No. of Chars
Train	A-Z	25	26
Validation	0–9	5	10
Test	@$€#?!%*+=	5	10
Total		35	46

In order to create an homogeneous dataset, subjects were provided with instructions on how to draw each character. The instructions included the pattern they had to follow for each character along with the starting and ending point for the symbol, the lines or curves they had to draw and the order they had to follow, as shown in Fig. 1. The starting point was shown as a blank circle

[2] https://www.fossil.com/en-us/smartwatches/learn-more/gen-5/.
[3] https://www.qualcomm.com/products/snapdragon-wear-3100-platform.
[4] https://wearos.google.com/.

and the ending point as a solid filled circle. Additionally, dashed lines showed were the subject would have to lift the hand and not draw anything. Despite the pattern limitation, subjects were free to choose the hand that they could use (left of right), to choose whether they would be sitting or standing and to select the writing surface (paper, whiteboard or simply draw the character in the air).

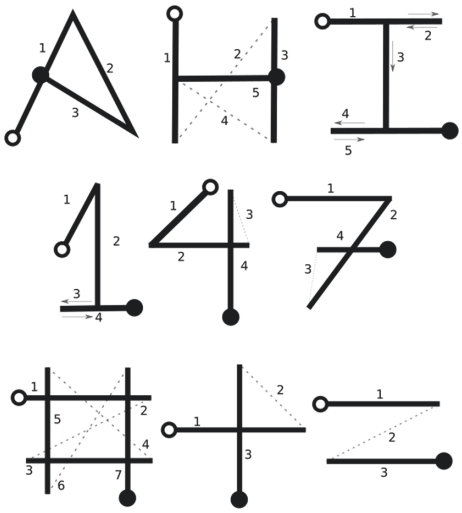

Fig. 1. A sample of the symbols used in the dataset

Two native Android applications were developed to build the dataset; one for smartphones and one for smartwatches. The smartwatch application was used to capture and save the raw data from the sensors, while the smartphone application was used for labeling the data. After wearing the smartwatch, the subject would select the character to execute from a list in the main screen of the smartphone application. Then, the device would send the symbol to the smartwatch and it would start capturing the data for the predefined duration. A vibration in the watch would signal the beginning and the ending of the capture. Afterwards, the data would be saved in Comma Separated Values (CSV) files in the smartwatch. When a subject would finish capturing the data, (s)he would tap a button on the mobile application to send the CSV files from the smartwatch to the smartphone and then another button to send the files to the server. Before actually sending the files, a random UUID (Universally Unique IDentifier) would be appended to the filename in order to easily distinguish each set. Files were deleted from both devices by tapping a button in the smartphone, before the devices was passed on the next subject. The reason for storing the output from each subject on separate file is application performance. By having small files, each write operation runs faster. Moreover, the files are sent faster between the devices and to the server. In order to avoid subjects executing the same gesture multiple

times, the background of the character in the list would change color when a gesture was performed. In addition, if the subject would tap on a character that has already executed, a dialog would appear asking him/her if (s)he wants to execute again the same gesture.

File structure is uniform across subjects: it has 9 values in each row. The first field denotes the Epoch time in milliseconds, i.e. the time that the data arrived in the application, the second field includes the relative timestamp of data arrival, as reported by the Operating System (OS), the next six fields include the x, y, and z axis values of the accelerometer and the gyroscope sensors respectively, while the last field contains the label for the executed gesture. The label takes on values from 0 to 45, where 0 denotes the character "A", 25 the character "Z", 26 the character "0", 35 the character "9" and 36 to 45 the special characters in the order described above.

3 Benchmarked Deep Learning Architectures

3.1 One-Shot Learning Problem Definition

One-shot learning datasets are typically comprised of three sets: a) the training set, b) the validation set, and c) the testing set. Each one of them contains disjoint label spaces (i.e., different gestures) [14]. The main objective of few-shot learning setting is to extract transferable embeddings/features from the observed gestures included in the training set. One-shot learning is a subclass of few-shot learning setting, where the classification is done under the restriction that only a single example of each possible class is observed before making a prediction about a test sample [14]. Moreover, since SENSED contains ten unique classes in the validation and test sets, the one-shot problem is defined as 10-way.

To benchmark our dataset we investigated the three DL approaches for one-shot learning (described in the following subsections), which are based on two deep learning architectures that have been successfully applied to human activity/gesture recognition. The first one, called PerceptionNet, is a deep CNN model introduced in [12], which relies on late sensor fusion [22] using a 2D convolutional layer. The second one, called deep ConvLSTM, deploys a LSTM (Long-Short Term Memory) layer after the convolutional ones so as to fuse the sensor modalities [28].

3.2 Vanilla Embedding Extraction

The approach adopted here is the most commonly used in bibliography [21] when it comes to Transfer Learning (TL), and it is comprised of the following steps. The DL model is firstly trained on a source domain (i.e., English capital letters), just like in any vanilla classification task. After training, the weights of the trained model are "frozen" and applied to the target domain. The tensors produced by the last hidden convolutional layers, which represent all the knowledge from the input signal (called embedding), are stored. In our 10-way

one-shot learning task in order to define the class of the anchor signal, we select that embedding e^c, which represents the class C that has the minimum norm distance from the anchor signal's embedding e^1.

The L_2 norm distance between the embeddings e^1, e^2 is given by:

$$d(e^1, e^2) = \left\|(e_i^1 - e_i^2)\right\|_2^2 \tag{1}$$

3.3 Pairwise Siamese Networks

A Siamese network similar to the one reported in [6], which is an end-to-end learning approach for extracting embeddings, has, also, been implemented. In this approach, the deep neural network is replicated twice (i.e., sharing the same filters for each input gesture sample). Just like the previous approach, the produced embeddings are compared using the Euclidean distance (i.e., L_2 norm) to directly predict whether the two input samples (i.e., anchor and candidate) belong to the same gesture class. However, this approach differs from the previous one because it does not aim at finding the nearest neighbor, but feeds the produced absolute difference to a fully connected layer followed by a sigmoid activation function. As a result, the output is mapped into a single logistic unit, where different gesture signals should produce a value equal to 0 and the same gestures equal to 1. To this end, a binary cross entropy function is used as a terminal step, to define the loss, similar to [14].

The binary cross entropy loss function is given by:

$$CE_b = -(y * log(p) + (1 - y) * log(1 - p)) \tag{2}$$

where y denotes the target label and p the probability produced by the sigmoid function.

3.4 Matching Network

Another network selected for evaluation here is the Matching Network (MatchNet) [27] which has been succesfully applied to previous human activity recognition tasks [30]. It shares the same logic with Pairwise Siamese classifiers, but here the anchor signal is compared with C candidate similar motion signals (i.e., ten in our case). MatchNet was developed on top of the two selected DL models (i.e., the CNN and the deep ConvLSTM). Figure 2 displays the network architecture for the case of CNN MatchNet.

The values of the selected window size are denoted by n_w, while the number of the sensor channels are depicted by n_c. Since the sensor signals in the selected architecture are stacked vertically, the number of the kernels is equal to 1. It should be noted that all the convolutional layers are followed by a ReLU activation function and a batch normalization layer, while all the pooling layers are followed by a dropout. After computing the embeddings, u for the anchor signal and v_i for the comparing signal, where $i = 1, ..., C$, we estimate the similarity score defined by:

$$similarity = \frac{u * v_i}{\|v_i\|} \tag{3}$$

Finally, the produced similarities scores are concatenated and normalized using a softmax function to compute the multi-class cross entropy loss:

$$CE_c = -\sum_{i=1}^{C} y_i \log(p_i) \tag{4}$$

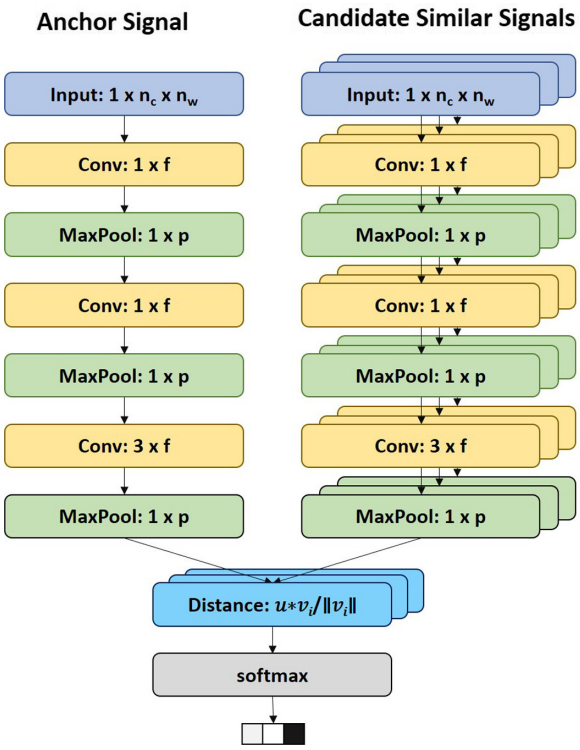

Fig. 2. CNN MatchNet architecture.

4 Experimental Set-Up

For our experiments we used a computer workstation equipped with a NVIDIA GTX 1080 Ti GPU featuring 11 gigabytes RAM, 3584 CUDA cores and a bandwidth of 484 GB/s. Python and specifically the Numpy library for matrix multiplications, data preprocessing and segmentation, was used as the programming language of choice, the scikit-learn[5] library was used for implementing the

[5] https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html.

t-SNE algorithm, and the Keras[6] high-level neural networks library with TensorFlow[7] library as backend was used for developing the model architectures. The CUDA Toolkit supported by the cuDNN[8], which is the NVIDIA GPU-accelerated library for deep neural networks, was used to accelerate the tensor multiplications. The software was installed on a 18.04 Ubuntu Linux operating system.

4.1 Data Sampling and Preprocessing

The data sampling process was done as follows. In the case of training set, the anchor gesture executed by a subject s_i was compared with 10 randomly chosen gestures executed by a subject s_j; we selected to make 2,000 10-way comparisons for each English character leading to 52,000 training samples. The same approach was followed in the cases of the validation and test sets; however, in these cases all possible 10-way combinations were taken into consideration, yielding 200 samples for each set. These are stored and published to be used for future algorithmic benchmarking.

Table 2. Examined hyperparameters

Name	Symbol	Range
Batch size	–	64, 128, 256
Learning rate	α	1e-03
Beta1	β_1	0.9
Beta2	β_2	0.999
Epsilon	ϵ	1e-08
Filter height	f_h	1–3
Filter width	f_c	7–15
Filter channels	f_c	16, 24, 32, 48, 64, 80
LSTM units	–	8, 16
Dropout probability	–	0.1–0.5
Maximum epoch	–	1000
Early stopping criterion	–	100

As a preprocessing step for all the sets, the collected gesture signals were normalized using the following equation:

$$z_i = \frac{(x_i - \mu_i)}{\sigma_i} \tag{5}$$

[6] https://keras.io/.
[7] https://www.tensorflow.org/.
[8] https://developer.nvidia.com/cudnn.

where x_i denotes the samples of sensor modality i, while μ_i, σ_i depict their corresponding mean and standard deviation values respectively, which were computed using only the training samples.

4.2 Hyperparameter Tuning

While training the selected models, the fine-tuning of several hyperparameters was attempted, based on the criterion of the lowest error in the validation set. The model's accuracy on the test set, was computed subsequently, using the same criterion. Table 2 illustrates all the examined hyperparameters. It should be noted that a set of the selected hyperparameters of a model (e.g., deep ConvLSTM) used in a certain one-shot learning architecture (e.g., MatchNet), could slightly differ from the optimal ones for the same model deployed to a different one-shot learning architecture (e.g., Siamese). Regarding the optimization of each model's weights, Adam algorithm [13] was selected, with the following hyperparameters: learning rate α equal to 0.001, β_1 equal to 0.9, β_2 equal to 0.999 and ϵ equal to 1e−08. Furthermore, the batch size was set 128 and the minimum number of epochs to 1,000. The training process was automatically terminated if the best validation accuracy had not improved after 100 epochs.

Table 3. 10-way accuracy performance of the one-shot learning techniques on SENSED

Network	Val acc	Test acc
CNN TL	52.5%	51.0%
ConvLSTM TL	56.5%	55.5%
CNN Siamese	63.00%	58.25%
ConvLSTM Siamese	62.25%	56.75%
CNN MatchNet	82.0%	76.5%
ConvLSTM MatchNet	71.5%	55.0%

5 Results and Discussion

Table 3 presents the accuracy reached by each network architecture. The MatchNet built on top of the PerceptionNet model obtained the best results by far, reaching 82.0% on the validation set and 76.5% for the test set. The three convolutional layers of this model had filter sizes $1 \times 32 \times 1 \times 11$, $32 \times 48 \times 1 \times 11$ and $48 \times 64 \times 3 \times 11$ resulting in 119,056 trainable parameters. Moreover, it is worth mentioning that while the ConvLSTM produced more generalizable embeddings than the CNN ones in the case of TL, this did not hold true for the Siamese and the MatchNet architectures.

Figure 3 illustrates the confusion matrices of the CNN MatchNet algorithm for the validation and the test sets. It should be noted that the algorithm misclassified many instances of class "6" to class "9". Apart from the fact that number "9"

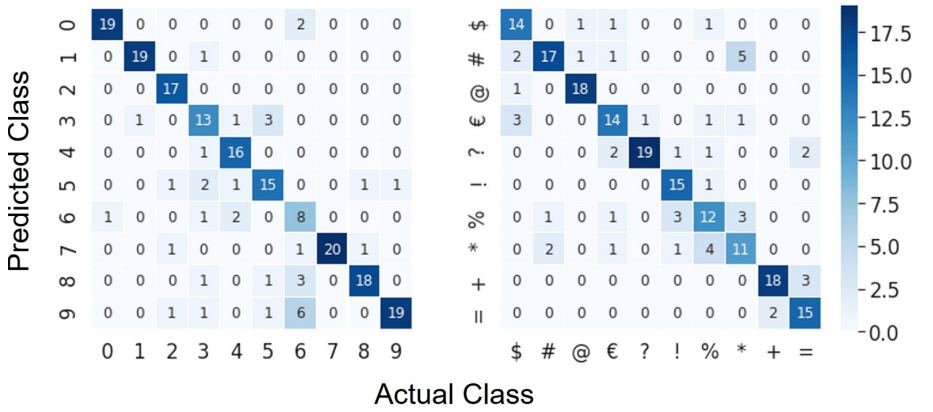

Fig. 3. Confusion matrices produced by CNN MatchNet for the validation (left) and the test (right) sets.

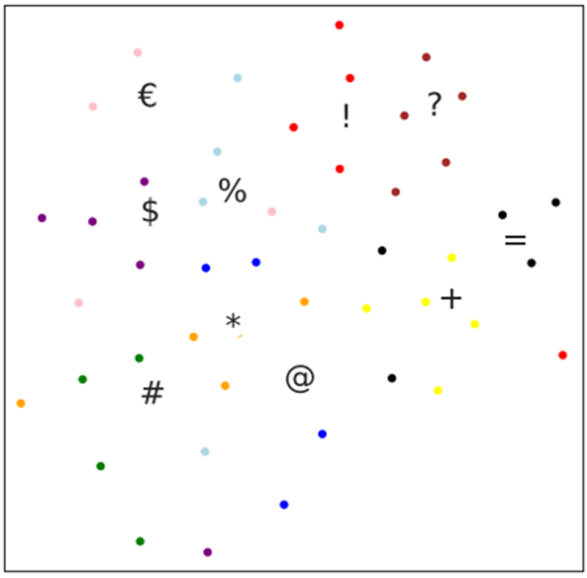

Fig. 4. t-SNE visualization of the test set's last hidden layer representations in CNN MatchNet.

is a reversed number "6", this occurred because the users were free to execute the gestures using hand, posture and writing surface of their choice. Consequently, the trained MatchNet seem to be a non-orientation-free algorithm and does not discriminate between the starting and the ending point of a gesture.

In the test set's confusion matrix some misclassification patterns can be observed. In particular, the algorithm confused a few instances of "=" with "+"

and vice versa, since they are consisted of two straight lines; it also confused instances of "*" and "#", since they both consist of six straight lines. In addition to this, "$" was misclassified three times as "€" and the model struggled to recognize class "%". All these observations, are shown in Fig. 4, as well. Figure 4 illustrates the extracted CNN MatchNet embeddings projected in the 2 dimensional space using the t-SNE algorithm [19].

Finally, despite the performance of the algorithms not being spectacular, especially if compared to those of Omniglot [14,27], the produced results are promising - even more so because SENSED is more challenging dataset. Hand-written characters presented as images are time, rotation, surface, hand and, more importantly, subject independent. Thus, the trained algorithms are expected to generalize more successfully on unseen characters. On the contrary, in the case of wearable-based gesture analysis, the DL models have to generalize against the aforementioned dependencies.

6 Conclusions

In this paper, we presented a publicly available wearable-based gesture recognition dataset, called SENSED, designed and built specifically to meet the needs of the one-shot learning setting. It is considered to be a fairly small dataset for deep learning algorithms to be trained on, since it contains only one instance of each gesture performed by each subject, while the gestures contained in the training, validation and test sets are disjoint with each other and are performed by different users. In our belief, the current dataset will assist researchers to test the generalizability of their algorithms on unseen gestures; it can also serve as a benchmark for one-shot learning.

Moreover, we have exploited the dataset to evaluate three one-shot learning techniques. The Matching Network architecture built on top of a deep CNN produced the best results, 82.0% and 76.5% accuracy on the validation and test respectively. Such recognition scores are considered to be promising for dynamic gesture-based user interactions with smart devices, as they might enable users to add moves of their own choice to smartwatches.

Exploitation of SENSED opens up a variety of new challenges and opportunities for one-shot learning. Future steps are currently being taken towards exploring more generalizable algorithms for domain-adaptation. Along a different line, data augmentation techniques based on Generative Adversarial Networks (GANS) could also be investigated.

Acknowledgements. This research is co-financed by Greece and the European Union (European Social Fund- ESF) through the Operational «Programme Human Resources Development, Education and Lifelong Learning 2014-2020» in the context of the project "On applying Deep Learning techniques to insufficient labeled sensor data for gesture recognition" (MIS 5050324).

References

1. Akbari, A., Jafari, R.: Transferring activity recognition models for new wearable sensors with deep generative domain adaptation. In: 2019 18th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), pp. 85–96 (2019)
2. Arnault, A., Hanssens, B., Riche, N.: Urban sound classification: striving towards a fair comparison. arXiv abs/2010.11805 (2020)
3. Brigato, L., Iocchi, L.: A close look at deep learning with small data. arXiv abs/2003.12843 (2020)
4. Chavarriaga, R., et al.: The opportunity challenge: a benchmark database for on-body sensor-based activity recognition. Pattern Recognit. Lett. **34**, 2033–2042 (2013)
5. Choi, Y., Hwang, I., Oh, S.: Wearable gesture control of agile micro quadrotors. In: 2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), pp. 266–271 (2017)
6. Chopra, S., Hadsell, R., LeCun, Y.: Learning a similarity metric discriminatively, with application to face verification. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), vol. 1, pp. 539–546 (2005)
7. Costante, G., Porzi, L., Lanz, O., Valigi, P., Ricci, E.: Personalizing a smartwatch-based gesture interface with transfer learning. In: 2014 22nd European Signal Processing Conference (EUSIPCO), pp. 2530–2534 (2014)
8. Feng, S., Duarte, M.F.: Few-shot learning-based human activity recognition. arXiv abs/1903.10416 (2019)
9. Garber, L.: Gestural technology: moving interfaces in a new direction. Computer **46**, 22–25 (2013)
10. Jacot, A., Gabriel, F., Hongler, C.: Neural tangent kernel: convergence and generalization in neural networks. In: NeurIPS (2018)
11. Kasnesis, P., Chatzigeorgiou, C., Toumanidis, L., Patrikakis, C.Z.: Gesture-based incident reporting through smart watches. In: 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), pp. 249–254 (2019)
12. Kasnesis, P., Patrikakis, C.Z., Venieris, I.S.: PerceptionNet: a deep convolutional neural network for late sensor fusion. ArXiv abs/1811.00170 (2018)
13. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. CoRR abs/1412.6980 (2015)
14. Koch, G.R.: Siamese neural networks for one-shot image recognition (2015)
15. Lake, B.M., Salakhutdinov, R., Gross, J., Tenenbaum, J.: One shot learning of simple visual concepts. Cognit. Sci. **33**, 2568–2573 (2011)
16. Laput, G., Xiao, R., Harrison, C.: Viband: high-fidelity bio-acoustic sensing using commodity smartwatch accelerometers. In: Proceedings of the 29th Annual Symposium on User Interface Software and Technology (2016)
17. Liu, J., Wang, Z., Zhong, L., Wickramasuriya, J., Vasudevan, V.: uWave: accelerometer-based personalized gesture recognition and its applications. In: PerCom (2009)
18. Luna, M.M., Carvalho, T.P., Soares, F., Nascimento, H.A.D., Costa, R.M.: Wrist player: a smartwatch gesture controller for smart TVs. In: 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), vol. 02, pp. 336–341 (2017)

19. Maaten, L.V.D., Hinton, G.E.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**, 2579–2605 (2008)
20. Morales, F.J.O., Roggen, D.: Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. Sensors (Basel, Switzerland) **16**, 115 (2016)
21. Morales, F.J.O., Roggen, D.: Deep convolutional feature transfer across mobile activity recognition domains, sensor modalities and locations. In: ISWC 2016 (2016)
22. Münzner, S., Schmidt, P., Reiss, A., Hanselmann, M., Stiefelhagen, R., Dürichen, R.: CNN-based sensor fusion techniques for multimodal human activity recognition. In: Proceedings of the 2017 ACM International Symposium on Wearable Computers (2017)
23. Nascimento, T.H., Soares, F.A.A.M.N., do Nascimento, H.A.D., Vieira, M.A., Carvalho, T.P., de Miranda, W.F.: Netflix control method using smartwatches and continuous gesture recognition. In: 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE), pp. 1–4 (2019)
24. Reiss, A., Stricker, D.: Introducing a new benchmarked dataset for activity monitoring. In: 2012 16th International Symposium on Wearable Computers, pp. 108–109 (2012)
25. Ronao, C.A., Cho, S.B.: Human activity recognition with smartphone sensors using deep learning neural networks. Expert Syst. Appl. **59**, 235–244 (2016)
26. Villani, V., Sabattini, L., Battilani, N., Fantuzzi, C.: Smartwatch-enhanced interaction with an advanced troubleshooting system for industrial machines (2016)
27. Vinyals, O., Blundell, C., Lillicrap, T.P., Kavukcuoglu, K., Wierstra, D.: Matching networks for one shot learning. In: NIPS (2016)
28. Wang, J., Zheng, V., Chen, Y., Huang, M.: Deep transfer learning for cross-domain activity recognition. In: ICCSE 2018 (2018)
29. Wang, J., Chen, Y., Hu, L., Peng, X., Yu, P.S.: Stratified transfer learning for cross-domain activity recognition. In: 2018 IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 1–10 (2018)
30. Wijekoon, A., Wiratunga, N., Sani, S.: Zero-shot learning with matching networks for open-ended human activity recognition. In: SICSA ReaLX (2018)
31. Zhu, P., Zhou, H., Cao, S., Yang, P., Xue, S.: Control with gestures: a hand gesture recognition system using off-the-shelf smartwatch. In: 2018 4th International Conference on Big Data Computing and Communications (BIGCOM), pp. 72–77 (2018)