FPGA-Based Realtime Detection of Freezing of Gait of Parkinson Patients

Patrick Langer[1], Ali Haddadi Esfahani[1], Zoya Dyka[1], and Peter Langendörfer[1,2(✉)]

[1] IHP GmbH, 15326 Frankfurt (Oder), Germany
[2] BTU Cottbus-Senftenberg, 03046 Cottbus, Germany
langendoerfer@ihp-microelectronics.com

Abstract. In this paper we report on our implementation of a temporal convolutional network trained to detect Freezing of Gait on an FPGA. In order to be able to compare our results with state of the art solutions we used the well-known open dataset Daphnet. Our most important findings are even though we used a tool to map the trained model to the FPGA we can detect FoG in less than a millisecond which will give us sufficient time to trigger cueing and by that prevent the patient from falling. In addition, the average sensitivity achieved by our implementation is comparable to solutions running on high end devices.

Keywords: Freezing of Gait · Temporal convolution models FPGA based implementation · Tool assisted implementation · Body worn sensor nodes

1 Introduction

Detecting Freezing of Gait (FoG) and triggering countermeasures like cueing is an important issue with respect of quality of life of Parkinson patients. Parkinson patients are often so afraid of falling due to FoG, that their social life seriously suffers as they do no longer leave their flats. A proper device for detecting FoG needs to be wearable and, to avoid stigmatization, as unobtrusive as possible. The time for detecting FoG and triggering the cueing is limited by the normal time for taking a step which is about 300 ms. As the device needs to get some sensor data to do the assessment and then to trigger the cueing, the whole process should be limited to about 50 ms. This leads to strict hard real time requirements and extremely short processing time. In addition, cueing may not be triggered too often to avoid that the patient becomes too familiar and it is no longer helping if FoG really occurs. The occurrence of such habituation effects needs to be investigated in further research and long-term tests with patients.

Contributions of This Paper. We report on our FPGA implementation that provides extremely good parameters with its quite good average sensitivity well above 78, comparable to what is reported in literature for high end devices

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2022
Published by Springer Nature Switzerland AG 2022. All Rights Reserved
M. Ur Rehman and A. Zoha (Eds.): BODYNETS 2021, LNICST 420, pp. 101–111, 2022.
https://doi.org/10.1007/978-3-030-95593-9_9

and a detection time of far less than 1 ms. The paper is structured as follows. Section 2 presents an overview of current research for FoG detection and its requirements. Furthermore, recent methods for the implementation of neural networks on FPGAs are discussed. Section 3 gives details about our own implementations and methods. Our experimental results are discussed in Sect. 4. Section 5 provides our conclusion and presents an outlook for our further research.

2 Related Work

2.1 Overview of Recent Methods of Detecting FoG

When it comes to Freezing of Gait (FoG), different algorithms have been used to identify an FoG event from sensor data (usually acceleration sensors). For example, in [8,17,18,30], threshold analysis methods are used, normally by applying those methods on extracted statistical features. Based on that, [3,24] employ Support Vector Machines (SVM) for FoG detection on sensor data. With the breakthrough of Machine Learning (ML) and Deep Learning (DL) methods, more recent research applied those technologies on FoG datasets to achieve new state of the art results. [15] tested different machine learning techniques for FoG detection. In [29], a Convolutional Neural Network (CNN) was trained on the raw sensor data and was used as an end-to-end classifier for FoG detection. As FoG detection is based on time series data usually from accelerometers, [31] claims that better results for FoG detection can be achieved when neural network architectures are used that are especially targeted at time series data. Historically this is associated with Recurrent Neural Networks (RNNs) [4], [14]. Especially Long short-term memory (LSTM) Units have achieved state of the art results on time series problems, such as speech recognition [10] or Human Activity Recognition (HAR) [19]. As such, [31] propose a combined CNN-LSTM architecture for FoG detection. The CNN is intended to learn the necessary feature extraction from the raw sensor data, whereas the LSTM shall learn the time-based dependencies in order to classify a time series and decide whether an FoG event occurred. A similar architecture was proposed by [26]. They evaluated a CNN combined with fully connected layers (Multilayer Perceptron, MLP) and a CNN combined with LSTM. They evaluated their models by means of sensitivity, specificity, area under the curve (AUC), geometric mean (GM) and equal error rate (EER). The CNN-LSTM achieved slightly better results (e.g. 0.849 vs. 0.844 sensitivity and the same for specificity). Thus, the authors state that the CNN-LSTM is the better architecture and should be used for future experiments.

Recently, it has been shown that CNNs in fact are able to learn (long-term) dependencies on time series data, contradicting common convictions that RNNs are the logical choice for such data sets. In [20], a special 1D convolutional model was developed for raw audio generation. Recent research suggests that similar convolutional models are able to learn long-term dependencies, possibly better than recurrent models and LSTMs [6]. Those models form a new group of CNNs, called Temporal Convolutional Models (TCNs), which achieved state of the art results on different datasets [6], yet inherently being a lot more straightforward

by having a simpler architecture. In [13] it was proposed to combine TCNs and LSTMs in order to achieve better results in FoG detection (additionally, they also introduced an attentional mechanism). However, the TCN here was only used for the feature extraction, as in previous papers, not explicitly for learning dependencies in the time series data.

2.2 Requirements of FoG Detection Methods for Wearable Devices

Most of those publications especially focus on achieving better detection results and the applied methods have been trained and tested on modern GPUs. For example the mentioned TCN-LSTM architecture of [13] can be run in 0.52 ms but has only been evaluated on modern GPU (NVIDIA RTX 2060). From our understanding, there seems to be a gap in research between increasing accuracy and achieving real time capability for deployment in real world applications. While focusing on achieving better detection results, it is important to keep in mind that the proposed methods shall be deployed in wearable devices to aid parkinson patients. It is necessary that they can be run in real time efficiently. A freezing must be detected within at least 300 ms, so that an appropriate cueing signal can be issued fast enough in order to prevent patients from falling. As those wearable devices are powered by battery, the used hardware to run the neural network needs to be efficient, having an overall low power consumption.

2.3 Field Programmable Gate Arrays (FPGAs) for Inference of Neural Networks

Field Programmable Gate Arrays (FPGA) are often used as algorithm-specific hardware accelerator [5,21]. They can eventually be more efficient than generic computing units like CPUs or GPUs [7,22]. Recently, approaches to run neural networks on FPGAs are conducted, aiming for use cases in embedded applications or wearable devices. FoG detection was implemented with a specific neural network for those matters on an FPGA [16]. This architecture even has online learning capabilities, which means that the model on the FPGA can be trained continuously in action. However, the architecture itself is hardwired, it is not easily possible to switch to a more modern or sophisticated neural network. In order to speed up the development process and gain more flexibility means, to convert a neural network, trained in a common machine learning framework like TensorFlow or PyTorch, to some format applicable for the FPGAs would be needed. High Level Synthesis for machine Learning (HLS4ML) [12] is a project addressing this issues. Another approach was taken by Xilinx. They developed the so-called Deep Learning Processing Unit (DPU) [2] for their FPGAs. This is a programmable computation engine enabling FPGAs to run neural networks. Different types of DPUs with different supported layers are available, e.g. there is one for convolutional neural networks and one for recurrent neural networks. To the best of our knowledge, this technology represents the current state of the art of a generic method to deploy neural networks on FPGAs.

3 Methods

3.1 VitisAI to Run Neural Networks on FPGA

Methods of FoG detection should be executable in real-time on wearable devices and preferably only require a minimum amount of power. Therefore, we choose an FPGA as our targeted hardware and first explain how to use it to run neural networks. Xilinx provides a development environment for execution of neural networks on FPGAs. Currently, the machine learning frameworks Caffe, PyTorch and TensorFlow are supported. For this, the FPGA is configured (programmed) with Xilinx DPU. The DPU running inside the FPGA then is able to load a neural network from a file in a specific binary format (called xmodel). To generate this file from a certain model, it needs to be converted, which is a process containing two steps [1]:

1. Quantization: The trained model needs to be quantized. Currently, the DPU only supports 8-bit integer quantization. Thus, if the model was trained using 32-bit-floating-point values, those are converted to 8-bit integer values. This results in a loss of precision, possibly decreasing the accuracy of the neural network. This problem can partially be alleviated by so-called finetuning, which is also supported by the development tools.
2. Compilation: The quantized model is compiled into a binary file, which contains instructions to be run on Xilinx DPU.

Figure 1 shows the used hardware for our experiments, namely the UltraZED-EG hardware platform. It is a credit-card-sized FPGA platform based on a Xilinx XZCU3EG with some additional hardware. Additionally, a carrier card is available, which features peripheral connections, like USB or ethernet ports. The reason to use this platform is its small size and energy efficiency, but it currently does not support to execute recurrent layers and LSTM. We report on how we solved this issue in the following section.

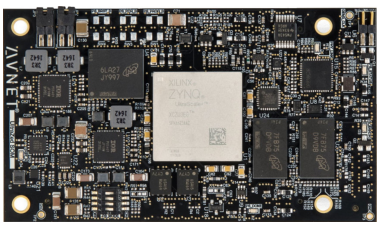

Fig. 1. UltraZED-EG. The used hardware platform for our experiments, which can used for embedded applications.

3.2 Hardware-Aware Implementation of a Temporal Convolutional Network for FoG Detection

As mentioned, usually RNNs are chosen for dealing with time series data. However, recently it has been shown that TCNs might achieve comparable or even better results [6]. When it comes to deciding which type of neural network shall be used for the implementation, a thorough analysis of benefits and drawbacks is essential. For the case of FoG detection, we especially consider the advantages of TCNs shown in Table 1 to be very important. In addition, it needs to be taken into account that TCNs potentially have a higher memory requirement during inference. When it comes to FPGA-based neural network accelerators, often memory bandwidth is the main limitation [25]. Thus, a recurrent neural network might have benefits when it comes to long input sequences or more complex problems, as the memory requirement during inference is potentially lower [6]. However, in the case of FoG detection, memory of modern FPGA systems is sufficient to run our proposed architecture very well.

Table 1. Advantages of TCNs compared to RNNs according to [6].

	TCN	RNN
Parallelism	Input sequence can be processed as a whole, convolution operations and filters are parallelizable. Especially important for FPGAs or other hardware accelerators	Each sample in the input sequence is processed one at a time (only sequential processing)
Stable gradients	Backpropagation path different from temporal direction of sequence, avoids vanishing/ exploding gradient problem. Plus strategies like skip connections etc. can be used to build deep networks just as for conventional neural networks	LSTMs reduce risk of vanishing gradient problems, exploding gradients might still be a problem [27]
Receptive field size	Flexible, can be easily expanded, e.g. by increasing filter size or dilation factor. Might lead to better possibilities to learn long term dependencies.	Cannot be flexibly changed or influenced

So, from our point of view TCN are the better choice for an FPGA implementation. Another important aspect is that RNNs are not yet supported for our targeted FPGA platforms, neither by Xilinx VitisAI nor by other alternatives such as HLS4ML [12] etc. Xilinx provides a DPU able to execute RNNs only for Alveo platforms.

Implementation Details. For our implementations, which were done in Keras, we used [23] as a reference. After training, the model was converted to the binary file needed for the FPGA. For our tests, we used VitisAI version 1.3[1]. However, there were some restrictions for the conversion of the model in terms of supported layers and operations.

- First, the mentioned version of VitisAI only supports models built with Keras functional API. Furthermore, custom layers as used in [23] cannot be used.
- Second, Conv1D operations are not supported.

The latter issue can be remedied as follows. It is possible to replace any Conv1D operation with a Conv2D operation equivalently. For example, if the Conv1D operation uses a kernel of size 3, the Conv2D operation can use kernel of size 1×3, where 1 represents the height and 3 the width. However, while Conv1D operation in Keras supports causal (asymmetric) and non-causal (symmetric) padding, Conv2D operations only support the latter. However, as Keras is based on TensorFlow, a tf.pad layer can be used to do asymmetrical padding manually. But standard TensorFlow layers are currently only supported by VitisAI for TensorFlow 1, not for TensorFlow 2.

We realized causal padding with manual padding using tf.pad layer. The conversion for the FPGA worked well, however the used tools indicated that some of the layers of the converted model might not be run on the FPGA but on the CPU of the SoC automatically. This is not the case if non-causal padding is used.

To solve these issues, we came up with our own implementation of TCN which is convertible with VitisAI to be run on the FPGA (or FPGA + CPU accordingly). The desired form of padding can be chosen as desired. Training and conversion can be done end-to-end, no manual steps are required.

4 Experiments

4.1 Dataset Description

We trained our model using the Daphnet dataset [9,11] in order to compare our own results with the state-of-the-art publications using the same data set (such as [13] or [15]). It is a publicly available dataset of movement data recordings from 10 Parkinson patients. The age of those patients ranged from 59 to 75 years (66.4 years $\pm$ 4.8 years), whereas 3 patients were females. The patients were asked to perform three walking tasks as described in [9]. Three sensor nodes placed at different locations of the body, i.e. shank, thigh, and on the lower back of the patients, were used to record the data. Each sensor acquired data at a frequency 64 Hz. A physiotherapist marked FoG events through recorded videos of the experiments. In total, 8 h 20 min of acceleration signals were recorded, during which 237 FoG events occurred. Two of the ten patients did not show any freezing, their gait appeared as normal walking.

[1] https://github.com/Xilinx/Vitis-AI/tree/v1.3.

4.2 Performance Evaluation

Training Details. For evaluation, we used a patient dependent approach. This means that for each patient, a separate model was trained. As mentioned, two patients (patients four and ten) did not experience any FoG episodes during the recordings. Those patients were excluded from the training. The training dataset for each patient was composed of 80% of the corresponding data for the patient including all data of all other patients (except patients four and ten). The remaining 20% of the patients' data were used as validation dataset. The split of 80/20 was chosen as a it is a common split for initial evaluations. We might test additional variants in future work and might also do a R10Fold evaluation as seen in recent literature regarding FoG detection [13].

The daphnet dataset, is highly imbalanced. Therefore, common indicators such as accuracy might not be suitable to evaluate the detection performance of a model. Thus, we use sensitivity (true positive rate) and specificity (true negative rate), as used in other publications as well. We configured our architecture to use three residual blocks as described in [6], using a kernel size of 3 and 64 kernels per layer overall. For each patient (except four and ten), our model was trained five times for 1000 epochs using a learning rate of 0.001 and batch size 1000. Among all trainings and epochs, the best model for each patient was saved. Afterwards, it was quantized using Vitis AI tools and converted for FPGA. The quantized model is a TensorFlow graph and can be run on GPU as well.

4.3 Hardware Platform Details

For our experiments, an UltraZED-EG was used. It features a Xilinx XCZU3EG multiprocessor system on a chip (MPSoC). It contains 154.350 system logic cells, 216 Block RAM blocks (7.6 MB BRAM memory in total) and 360 DSP slices. It features an ARM Cortex-A53 processor aswell. In our case, the processor runs an Ubuntu based operating system including PYNQ[2]. A program on the operating system is responsible for loading the test data, feeding it to the model running on the FPGA and interpret the results. The model only needs 0.7 ms = 700 µs for execution on the FPGA. This number was consistent during the whole evaluation. The FPGA does not need any scheduling like CPUs or GPUs, thus the inference time is almost exactly the same each time the model is run.

Results on Standard Daphnet Dataset. In Table 2 we present our results of the model for each patient. We evaluated sensitivity and specificity on the original model, the quantized model running on GPU and the converted quantized model running on the FPGA.

[2] http://www.pynq.io/board.html.

Table 2. Results for dataset without augmentation

Patient	Original		Quantized		FPGA	
	Sens.	Spec.	Sens.	Spec.	Sens.	Spec.
Patient 1	0.0833	0.9987	0.0833	0.9889	0.0833	1.0000
Patient 3	0.4615	0.9760	0.4615	0.9680	0.3846	0.9640
Patient 3	0.4603	0.9514	0.5238	0.9114	0.4127	0.9114
Patient 5	0.4216	0.9394	0.3627	0.9303	0.3725	0.9394
Patient 6	0.0714	0.9943	0.0357	0.9448	0.0357	0.9946
Patient 7	0.3158	0.9968	0.3684	0.9968	0.3158	0.9935
Patient 8	0.6905	0.9052	0.4286	0.9138	0.2857	0.9483
Patient 9	0.4035	0.9663	0.4211	0.8653	0.3158	0.8384
Average	0.3635	0.9669	0.3356	0.9482	0.2758	0.9487

As can be seen, the overall specificity is quite high, 0.9669 on the original model run on GPU and still 0.9487 on FPGA. However, with this naive approach, the average sensitivity is quite low. This is due to the fact that the dataset is highly imbalanced, and the positive class (FoG event) is underrepresented. This issue was addressed by recent research. Different publications suggest using augmentation or rebalancing strategies to improve the dataset, such as [13,28].

Results on Rebalanced Daphnet Set. Due to the lack of more balanced datasets, we as well used a simple oversampling strategy to virtually rebalance the dataset in order to evaluate our model in accordance with the current state of the art, which used augmentations. The results of the augmented dataset are presented in Table 3. In further research, we aim to build our own, better balanced dataset.

Table 3. Results for dataset without augmentation

Patient	Original		Quantized		FPGA	
	Sens.	Spec.	Sens.	Spec.	Sens.	Spec.
Patient 1	0.9995	0.9558	0.8103	0.9088	0.8966	0.9171
Patient 2	0.9996	0.9880	0.9692	0.9480	0.9487	0.9480
Patient 3	0.9995	0.9257	0.9199	0.8371	0.8846	0.8886
Patient 5	0.9994	0.9394	0.8518	0.8848	0.7787	0.8394
Patient 6	0.9998	0.9517	0.7059	0.7158	0.6053	0.9196
Patient 7	0.9999	0.9643	0.8000	0.7695	0.7316	0.9123
Patient 8	0.9988	0.9483	0.7000	0.9224	0.6048	0.9569
Patient 9	0.9997	0.9798	0.9146	0.8889	0.8043	0.8923
Average	0.9953	0.9566	0.8340	0.8594	0.7818	0.9093

On this rebalanced dataset, the sensitivity is significantly higher than on the non-augmented dataset. We achieve an average sensitivity of 0.9953 and specificity of 0.9566 with our model run on GPU, which is comparable to current state of the art results. However, after quantization, sensitivity and specificity suffer a significant drop (0.8340 sensitivity and 0.8594 specificity for the quantized model run on GPU, 0.7818 sensitivity and 0.90932 specificity for converted quantized model run on FPGA). This can be explained by the loss of precision, as the quantization converts the 32-bit float model to an 8-bit integer model.

5 Conclusions

In this paper we reported on an FPGA based implementation for detecting freezing of gait of Parkinson patients. We would like to stress the following points. Our implementation achieves almost the same values for sensitivity and specificity as reported in the literature for high end devices. Even after quantization, the results are quite good. So, the use of FPGAs to allow real time detection of FoG in wearables is a feasible solution. In our discussion with clinicians, they reported that false positives, if they do not occur too often, are not an issue and that patients rather like to get a cueing more often to be reassured the system is still working. So, 100% sensitivity is not the ultimate goal. On the other hand, a very fast detection of FoG is key when it comes to trigger proper cueing to prevent the patient from falling due to FoG. Here our implementation provides extremely good parameters with its quite good sensitivity and a detection time of far less than 1 ms. The latter is the parameter that makes fall prevention by a body worn sensor node feasible. Please note that we achieved this extremely fast processing even though we used a tool to map the trained model onto the FPGA. In our future work we aim at integrating our FPGA based solution with a wireless sensor node and to run experiments with Parkinson patients together with a clinical partner. In order to improve user experience, we will also work on increasing sensitivity. The loss of precision because of quantization can possibly be alleviated by finetuning the model. Xilinx already provides support for finetuning the converted models using their development tools. We are also interested to further reduce the processing time on the FPGA.

References

1. Vitis AI user guide. https://www.xilinx.com/support/documentation/sw_manuals/vitis_ai/1_3/ug1414-vitis-ai.pdf. Accessed 21 June 2021
2. Convolutional neural network with INT4 optimization on Xilinx devices white paper (2014)
3. Ahlrichs, C., et al.: Detecting freezing of gait with a tri-axial accelerometer in Parkinson's disease patients. Med. Biol. Eng. Comput. **54**(1), 223–233 (2015). https://doi.org/10.1007/s11517-015-1395-3
4. Almqvist, O.: A comparative study between algorithms for time series forecasting on customer prediction: an investigation into the performance of ARIMA, RNN, LSTM, TCN and HMM. Ph.D. thesis, June 2019

5. Andrey, G., Thirer, N.: A FPGA implementation of hardware based accelerator for a generic algorithm, November 2010. https://doi.org/10.1109/EEEI.2010.5662152
6. Bai, S., Kolter, J., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, March 2018
7. Betkaoui, B., Thomas, D.B., Luk, W.: Comparing performance and energy efficiency of FPGAs and GPUs for high productivity computing. In: 2010 International Conference on Field-Programmable Technology, pp. 94–101 (2010)
8. Bächlin, M., Hausdorff, J., Roggen, D., Giladi, N., Plotnik, M., Tröster, G.: Online detection of freezing of gait in Parkinson's disease patients: a performance characterization. In: BODYNETS 2009–4th International ICST Conference on Body Area Networks, p. 11, April 2009. https://doi.org/10.4108/ICST.BODYNETS2009.5852
9. Bächlin, M., Plotnik, M., Roggen, D., Giladi, N., Hausdorff, J., Tröster, G.: A wearable system to assist walking of Parkinson's disease patients. Methods Inf. Med. **49**, 88–95 (2009). https://doi.org/10.3414/ME09-02-0003
10. Chiu, C.C., et al.: State-of-the-art speech recognition with sequence-to-sequence models, pp. 4774–4778, April 2018. https://doi.org/10.1109/ICASSP.2018.8462105
11. Bächlin, M., et al.: Wearable assistant for parkinson's disease patients with the freezing of gait symptom. IEEE Trans. Inf. Technol. Biomed. **14**(2), 436–446 (2010)
12. Duarte, J., et al.: Fast inference of deep neural networks in FPGAs for particle physics. ArXiv arXiv:1804.06913 (2018)
13. Li, B., Yao, Z., Wang, J., Wang, S., Yang, X., Sun, Y.: Improved deep learning technique to detect freezing of gait in Parkinson's disease based on wearable sensors. Electronics **9**, 1919 (2020). https://doi.org/10.3390/electronics9111919
14. Mahmoud, A., Mohammed, A.: A survey on deep learning for time-series forecasting. In: Hassanien, A.E., Darwish, A. (eds.) Machine Learning and Big Data Analytics Paradigms: Analysis, Applications and Challenges. SBD, vol. 77, pp. 365–392. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-59338-4_19
15. Mazilu, S., et al.: Online detection of freezing of gait with smartphones and machine learning techniques (2012). https://doi.org/10.4108/icst.pervasivehealth.2012.248680
16. Mikos, V., et al.: A wearable, patient-adaptive freezing of gait detection system for biofeedback cueing in Parkinson's disease. IEEE Trans. Biomed. Circuits Syst. (2019). https://doi.org/10.1109/TBCAS.2019.2914253
17. Moore, S., MacDougall, H., Ondo, W.: Ambulatory monitoring of freezing of gait in Parkinson's disease. J. Neurosci. Methods **167**, 340–8 (2008). https://doi.org/10.1016/j.jneumeth.2007.08.023
18. Moore, S., et al.: Autonomous identification of freezing of gait in Parkinson's disease from lower-body segmental accelerometry. J. Neuroeng. Rehabil. **10**, 19 (2013). https://doi.org/10.1186/1743-0003-10-19
19. Murad, A., Pyun, J.Y.: Deep recurrent neural networks for human activity recognition. Sensors **17**, 2556 (2017). https://doi.org/10.3390/s17112556
20. Oord, A., et al.: Wavenet: A generative model for raw audio, September 2016
21. Possa, P., Schaillie, D., Valderrama, C.: FPGA-based hardware acceleration: a CPU/accelerator interface exploration. In: 2011 18th IEEE International Conference on Electronics, Circuits, and Systems, pp. 374–377 (2011). https://doi.org/10.1109/ICECS.2011.6122291
22. Qasaimeh, M., Denolf, K., Lo, J., Vissers, K., Zambreno, J., Jones, P.: Comparing energy efficiency of CPU, GPU and FPGA implementations for vision kernels, May 2019. https://doi.org/10.1109/ICESS.2019.8782524
23. Remy, P.: Temporal convolutional networks for Keras (2020). https://github.com/philipperemy/keras-tcn

24. Rodríguez-Martín, D., et al.: Home detection of freezing of gait using support vector machines through a single waist-worn triaxial accelerometer. PLoS One **12**, e0171764 (2017)
25. Shawahna, A., Sait, S.M., El-Maleh, A.: FPGA-based accelerators of deep learning networks for learning and classification: a review. IEEE Access **7**, 7823–7859 (2019)
26. Sigcha, L., et al.: Deep learning approaches for detecting freezing of gait in Parkinson's disease patients through on-body acceleration sensors. Sensors **20**, 1895 (2020). Basel, Switzerland
27. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: NIPS (2014)
28. Um, T.T., et al.: Data augmentation of wearable sensor data for Parkinson's disease monitoring using convolutional neural networks. In: Proceedings of the 19th ACM International Conference on Multimodal Interaction (2017)
29. Wang, J., Liu, Q., Chen, H.: Detection of freezing of gait for Parkinson's disease patients based on deep convolutional neural networks. Chin. J. Biomed. Eng. **36**, 418–425 (2017). https://doi.org/10.3969/j.issn.0258-8021.2017.04.005
30. Zach, H., et al.: Identifying freezing of gait in parkinson's disease during freezing provoking tasks using waist-mounted accelerometry. Parkinsonism Relat. Disord. **21** (2015). https://doi.org/10.1016/j.parkreldis.2015.09.051
31. Zhang, Y., Gu, D.: A deep convolutional-recurrent neural network for freezing of gait detection in patients with Parkinson's disease, pp. 1–6, October 2019. https://doi.org/10.1109/CISP-BMEI48845.2019.8965723