Assessment of Human Activity Classification Algorithms for IoT Devices

Gianluca Ciattaglia, Linda Senigagliesi, and Ennio Gambi(✉)

Università Politecnica delle Marche, 60131 Ancona, Italy
{g.ciattaglia,l.senigagliesi,e.gambi}@univpm.it

Abstract. Human activity classification is assuming great relevance in many fields, including the well-being of the elderly. Many methodologies to improve the prediction of human activities, such as falls or unexpected behaviors, have been proposed over the years, exploiting different technologies, but the complexity of the algorithms requires the use of processors with high computational capabilities. In this paper different deep learning techniques are compared in order to evaluate the best compromise between recognition performance and computational effort with the aim to define a solution that can be executed by an IoT device, with a limited computational load. The comparison has been developed considering a dataset containing different types of activities related to human walking obtained from an automotive Radar. The procedure requires a pre-processing of the raw data and then the feature extraction from range-Doppler maps. To obtain reliable results different deep learning architectures and different optimizers are compared, showing that an accuracy of more than 97% is achieved with an appropriate selection of the network parameters.

Keywords: Automotive radar · Deep learning · Human activities · IoT

1 Introduction

Automatic human activity classification is an active research field with applications in various contexts, such as smart monitoring, video retrieval, ambient assisted living, surveillance, etc. Mainstream techniques rely on multiple sensors placed in different parts of the human body [23], on sensors of smartphones and wearable devices (such as wristbands and watches) [25], while contactless technologies are assuming a growing relevance, thanks to their ability to provide constant monitoring without the need of placing sensors in precise positions. Among these methodologies, Radar technology has been recently applied to identify people on the basis of their gait characteristics [16,21] and gestures recognition [15,22].

Activity classification is usually performed by applying machine learning and deep learning techniques. The rise of convolutional neural networks (CNN) has resulted in the development of categorization networks such as AlexNet [12] and VGG [18]. AlexNet that is the first DNN (deep neural network) published in

S. Spinsante et al. (Eds.): HealthyIoT 2022, LNICST 456, pp. 170–181, 2023.
https://doi.org/10.1007/978-3-031-28663-6_13

2012 significantly improved identification accuracy (10% higher) in comparison to traditional approaches on ImageNet's 1000 class. Since then, literature has focused on both, creating networks accurate and designing efficient in terms of computational cost. However, there is a lot of literature that discuss new architectures in terms of layer composition performance. There are few papers [1] which evaluate factors related to computational cost like execution time, memory usage, etc., and more relevantly, how computational efficiency impacts precision.

In this paper, we conduct different experiments to assess the benefits and drawbacks of the application of several existing deep neural networks (DNNs) on human activity classification. In particular, we consider the selection of 6 DNNs originally developed for image recognition and we evaluate their performance on images derived from the raw signal of an automotive FMCW Radar and then processed to obtain range-Doppler images as final output. Classification results are obtained considering the dataset in [5], made up of 171 images associated to three different activities, i.e., fast walk, slow walk and slow walk with hands in pockets. Two of these activities are very similar, so they are sometimes merged in order to achieve better results in terms of accuracy and loss functions. The dataset has been created with participants with different characteristics, such as age, sex, height and weight. Subjects walked back and forth the Radar used in the experiments within a distance of 12 m. Similar comparative studies have been developed in [2,13] on a very large dataset of plant diseases, while we test DNNs for human activity classification, which involves the resolution of different issues, starting from the small dimension of our dataset.

The rest of the paper is organized as follows. In Sect. 2 hardware and software used in our experiments are detailed. In Sect. 3 are introduced the DNN architectures. The performance metrics and the described results are finally presented in Sect. 4. Final considerations and remarks are provided in Sect. 5.

2 Materials and Methods

2.1 FMCW Radar

Frequency Modulated Continuous Wave (FMCW) Radar technology has recently improved its presence on the market due to its growing application in the automotive field. Devices that apply this technology are able to detect targets and to measure their velocity and angle of arrival. A simplified block scheme of this type of sensors is depicted in Fig. 1.

Angular information can be obtained with a Multiple-In-Multiple-Out (MIMO) sensor. In communications MIMO is generally used to improve the data throughput, while in Radar systems this technology is used to obtain the angle information exploiting the different phases of the echoed signals [6,11,19]. Thanks to the characteristics of radar sensors working with carrier frequencies between 77 to 81 [GHz] and given the wavelength involved, it is possible to obtain very interesting and precise information about the targets. In particular, the micro-Doppler extracted from the processing of these types of Radar signals can be exploited for classification purposes [14,16].

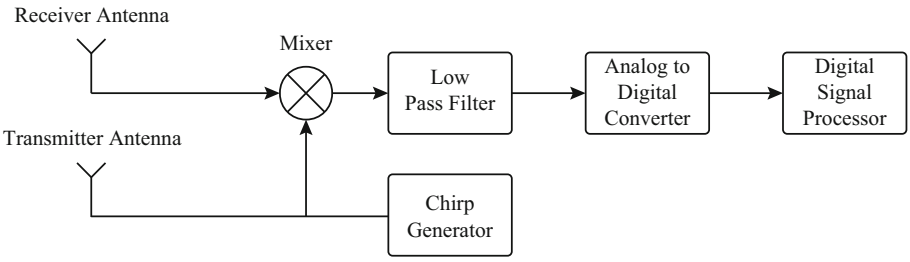

Fig. 1. FMCW Radar block scheme.

The dataset considered in the present work consists of range-Doppler maps, each of these images containing the micro-Doppler information related to different activities. The data processing necessary to obtain the maps from raw data is as follows. The considered sensor is the AWR1642, which is connected to the DCA1000 FPGA. A second board is used to configure the Radar sensor, collect the data, and stream the raw samples on the computer [8,9]. AWR1642 uses two-transmitter and four-receiver MIMO antennas. This means that the analog-to-digital converters sample four beat signals, which are then summed together. The summation improves the signal-to-noise ratio of the maps. At this point, only one signal is obtained and it is then reorganized into a Fast-Time/Slow-Time matrix. The Fast-Time represents the sample collection time (i.e., the sampling time of the analog-to-digital converters), while the Slow-Time represents the pulse repetition interval. An example of this map is reported in Fig. 2.

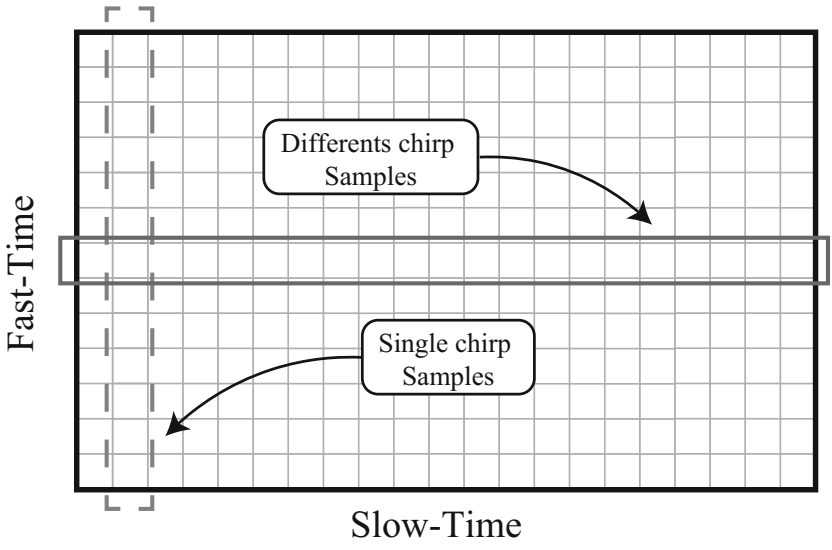

Fig. 2. Example of Fast-Time/Slow-Time matrix.

From these data matrices, it is possible to obtain the range-Doppler maps by performing a Fast Fourier Transform (FFT) along the Fast-Time axis and along the Slow-Time axis. The range-Doppler maps can be therefore used to train and test the classification algorithms. An example of the result of the whole pre-processing operation is shown in Fig. 3.

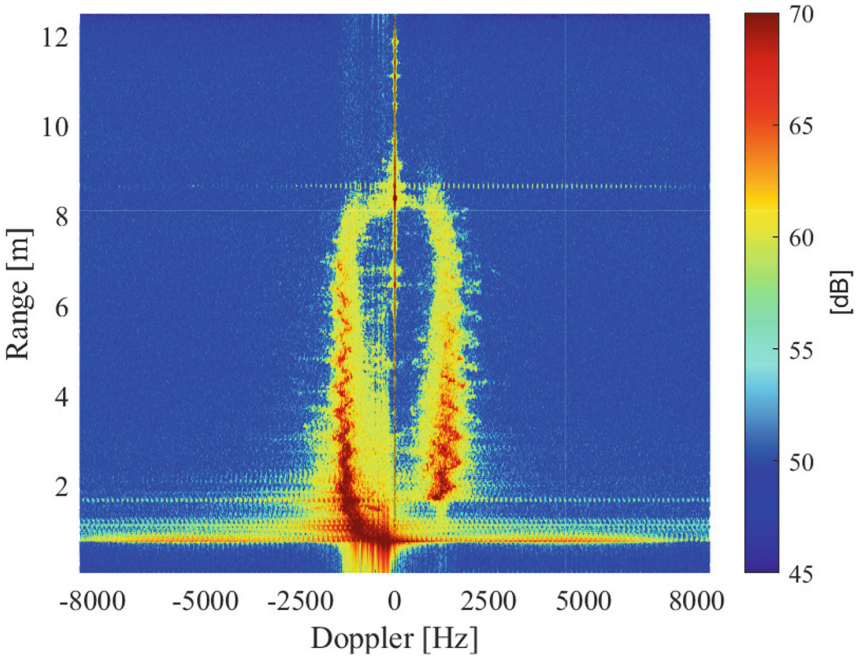

Fig. 3. Example of Range-Doppler map. Fast-Time has been converted into range distance and Slow-Time into Doppler.

2.2 Overview of CNNs

In this section, we provide a brief description of the CNNs architectures considered for our experiments. Because of the small dimension of the dataset in [5], transfer learning is applied [17]. TensorFlow package 2.3.0 is used to process neural networks with CUDA-V11.0 and cuDNN-v7.1 as the back-end.

A CNN architecture is made up of different layers, namely, the convolutional layer, pooling layer, reLU layer, fully connected layer, and loss layer. The convolutional layer is the main component of a CNN, comprising of filters (or *kernels*) that identify and forward various sorts of characteristics from the input. A pooling layer is included between consecutive convolutional layers to minimize the parameters in the network. The ReLu layer is a function that converts the negative numbers to zero.

We choose several types of networks, some of which are meant to be even more effective in terms of performance related to our small dataset. In certain

situations, the number that appears with the name of architecture represents the number of layers containing parameters to be learned (e.g., convolutional, or fully connected layers). The considered architectures are the following:

1. ResNet50
2. Inception ResNetV2
3. InceptionV3
4. VGG16
5. VGG19
6. Xception

A general overview of the different architectures is provided in the following, along with a brief description of the optimizers considered in our simulations.

The same sampling policies are applied to all the selected networks in order to have a direct fair and precise comparison. InceptionResNet-v2, Inception-v3, and Xception models require normalized images with 229 pixels, while for all the other models 224 pixels are used. Our study is focused on accuracy rate, model complexity, and inference time.

2.3 VGG

VGG16 [18] is composed by 16 levels divided into convolutional layers, max-pooling layers, and fully linked layers. It has 5 blocks and a maximum pooling layer in each block. Similarly, VGG19 contains 19 layers, with three additional convolution layers in the last three blocks. Both VGG16 and VGG19 excel in image recognition thanks to their deep layers.

2.4 InceptionV3

Deep convolutional architecture Inception V3 is frequently used for classification problems. Szegedy et al. [20] first suggested the model notion in the Google Net architecture where this model is proposed by upgrading the inception module. Each block of the Inception V3 network contains many branches of convolutions, average pooling, max pooling, concatenated, dropouts, and fully connected layers, with each block having several symmetric and symmetric construction blocks. The network contains 42 layers and 29.3 million parameters; thus, its computing cost is only around 2.5 times that of GoogleNet. The combination of a reduced parameter count and additional regularization with batch-normalized auxiliary classifiers and label-smoothing allows for training higher quality networks on small training sets.

2.5 Residual Network (ResNet)

He et al. [7] developed the ResNet models, which are based on deep architectures that have demonstrated strong convergence tendencies and convincing correctness. ResNet was created using many stacked residual units and a variety of layers. However, depending on the design, the number of operations can be changed.

Convolutional, pooling, and layers make up the residual units for all the above. ResNet is comparable to VGG net but it is eight times deeper. The ResNet 50 network has 49 convolutional layers and a fully linked layer at the end.

2.6 Xception

Xception [3] is based on depthwise separable convolutional layers. This neural network architecture has 36 convolutional layers. In this way, the feature extraction base of the network is formed. Except for the first and the last modules, all thirty six layers are divided into 14 modules, all of which contain linear residual connections surrounding them. In summary, the Xception design is a depthwise separable convolution layer stack with residual connections. This lets the architecture be easily modifiable and defined simply using high-level libraries.

2.7 Optimizers

Gradient descent is the most used method to improve the learning of deep neural networks. It basically updates each model parameter, verifying how it affects the objective function, and iterates until this function converges to the minimum. Stochastic Gradient Descent (SGD) is a well known variation of gradient descent. The Adaptive Gradient Algorithm (Adagrad) optimizer [4] is designed to deal with sparse data. It adapts the learning rate to the parameters, performing smaller updates (i.e. low learning rates) for parameters associated with frequently occurring features, and larger updates (i.e. high learning rates) for parameters associated with infrequent features. Adadelta is instead an extension of Adagrad developed in [24], simultaneously to Root Mean Square Propagation (RMSprop), with the aim of solving Adagrad's diminishing learning rates. Adaptive Moment Estimation (Adam) [10] is another method that computes adaptive learning rates for each parameter.

3 Proposed Methodology

As already mentioned in the previous sections, the dataset in [5] is considered. Dataset was built at Marche Polytechnic University. Different subjects of age, sex, height, and weight participate in the tests, repeating each activity (slow walk, fast walk and walk with the hands in pockets) three times. A total of 171 images of 224×224 pixels are built. The raw data obtained from the Radar are processed to obtain range-Doppler maps as described in Sect. 2.1 and feature detection is applied to smooth the signals.

The proposed methodology is conceptualized in Fig. 4. All the images are colored (RGB). Classification is performed by exploiting the range-Doppler maps. Data Augmentation is used to pre-process the range-Doppler images to remove noise or redundancy. Also, it increases accuracy and rushes the execution. For this purpose, Keras deep learning library in Python is used. When a large amount

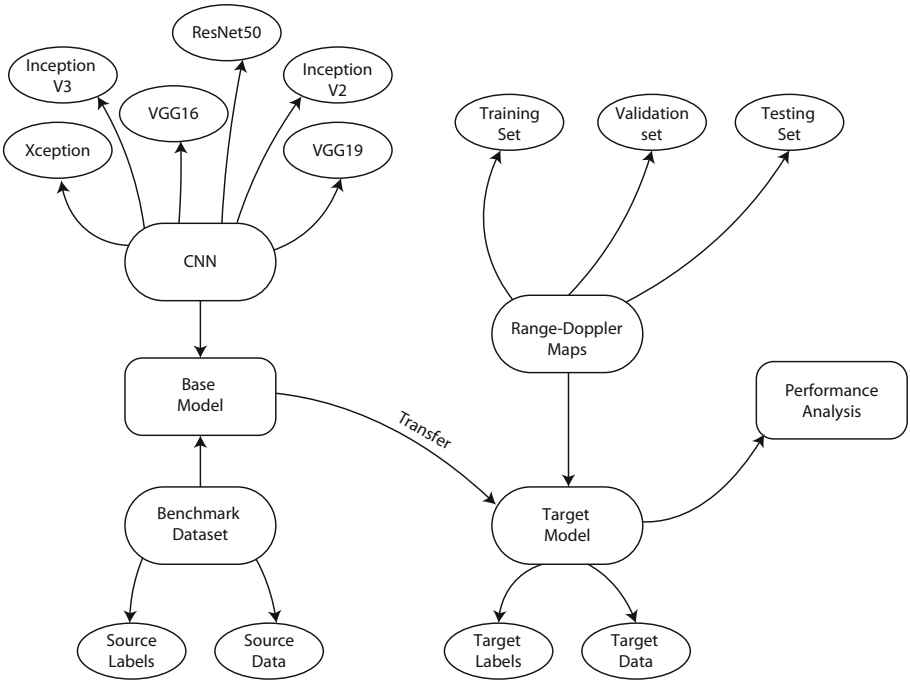

Fig. 4. Conceptual model of the proposed methodology.

of data is available, a model is built from scratch to solve a specific problem. Differently, when the dataset dimension is small, transfer learning is usually applied, and a model built by training on a larger dataset can be used as a starting point to train the network on our specific data.

Numerous pre-trained models are available under certain conditions. For our implementation, we used six pre-trained models, namely, VGG16, VGG19, InceptionResNetV2, InceptionV3, ResNet50, Xception. We fine-tuned all the models on the last layer. All 171 images are randomly generated using ImageDataGenerator. Training and testing image datasets used the same data generators. These ImageDataGenerators apply pre-processing while generating the images. We also use label encoding to get the categorical output. When the validation accuracy stops improving, early stopping is used to terminate the training process. The underlying model is then combined by a GlobalAveragePooling layer, which is used to reduce the data and prepare the model for the final classification. After that, a BatchNormalization layer is introduced for model stability and to have a faster execution, resulting in fewer training epochs. The last layer exploits a dense layer and a SoftMax activation function for extracting and categorizing the output. The weights are updated using different optimizers with an activation function of softmax in the final layer for classifying output. Optimizers are

used to update the weights. The performance of the network is measured using a categorical loss function. A dropout is added to avoid overfitting.

After verifying that a number greater than 100 epochs leads to overfitting, the models are trained using a size batch of 50 and 100 epochs on the training set and then tested on the test set. At the end we evaluate the performance of the chosen models as a function of different parameters.

4 Experiments and Results

We here present a comparison of the results obtained by applying different neural networks and different optimizers to the dataset in [5]. For the sake of reproducibility, we select a random seed equal to 1337. This choice does not influence the results, since they are averaged over multiple random selections. The dataset then is randomly split into 80% for training, 10% for validation, and 10% for testing. This is done to prevent results from repeating each time the algorithm is rerun. The considered output classes are:

- Fast walk;
- Slow walk;
- Slow walk with hands into pockets.

We first define the metrics which will be used to evaluate the performance of the proposed methodologies. They are based on the so-called *confusion matrix*, whose columns represent the predicted values for each class, while the rows represent the real values. The most common metric is the accuracy, which is defined as

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}, \tag{1}$$

where TP and TN stand for true positives and true negatives, while FP and FN are the false positives and the false negatives.

A loss function measures the probabilities or uncertainty of a prediction based on how much the prediction varies from the true value. Differently from accuracy, the loss is computed as a summation of the errors made for each sample in training or validation sets, and is not evaluated as a percentage. Loss is often used in the training process to find the "best" parameter values for the model (e.g. weights in neural network). During the training process, the goal is to minimize this value. In the following, we consider the categorical cross-entropy loss.

Another useful metric is the precision (also called positive predictive value), which corresponds to the ratio of correctly predicted positive observations to the total predicted positive observations or

$$P = TP/(TP + FP). \tag{2}$$

Recall (also known as sensitivity) is the ratio of correctly predicted positive observations to all observations in actual class

$$R = TP/(TP + FN). \tag{3}$$

Table 1. Accuracy and loss for two activities. Accuracy values are reported as percentages.

Model	VGG16		ResNet50		Inception ResNetV2		InceptionV3		Xception		VGG19	
Optimizer	ACC	LOSS	ACC	LOSS	ACC	LOSS	ACC	LOSS	ACC	LOSS	ACC	LOSS
Adam	93.75	0.17	85.71	0.32	95.92	0.63	97.96	0.11	93.97	0.13	93.75	0.29
RMSprop	93.75	0.17	81.63	0.36	97.96	0.06	97.96	0.05	97.96	0.07	93.75	0.25
SGD	87.50	0.27	67.35	0.54	89.80	0.22	95.92	0.16	91.84	0.20	81.25	0.37
Adadelta	81.25	0.48	67.35	0.51	89.80	0.27	87.76	0.25	89.80	0.25	81.25	0.50
Adagrad	93.75	0.19	89.80	0.34	97.96	0.04	95.92	0.06	97.96	0.06	93.75	0.27
Adamax	93.75	0.17	89.80	0.40	97.96	0.07	97.96	0.05	95.92	0.14	93.75	0.29

Table 2. Accuracy and loss for three activities. Accuracy values are reported as percentages.

Model	VGG16		ResNet50		Inception ResNetV2		InceptionV3		Xception		VGG19	
Optimizer	ACC	LOSS	ACC	LOSS	ACC	LOSS	ACC	LOSS	ACC	LOSS	ACC	LOSS
Adam	60	0.66	53.3	0.74	66.7	2.2	66.7	0.51	66.67	3.44	66.67	0.75
RMSprop	66.7	0.74	53.3	0.91	80	1.1	86.7	1.49	80	1.70	73.3	0.68
SGD	60	0.94	33.3	1.08	73.3	0.66	86.7	0.56	60	0.85	53.3	0.89
Adadelta	60	1.01	53.3	0.99	53.3	0.77	60	0.65	60	0.77	60	0.97
Adagrad	53.3	0.68	60	0.82	80	0.77	66.7	0.66	73.3	0.87	53.3	0.74
Adamax	80	0.67	46.6	0.82	80	0.90	86.7	2.50	66.7	1.30	60	0.77

The F1 score is the harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall). Therefore, this score takes both false positives and false negatives into account as follows

$$F1 = 2RP/(R + P). \tag{4}$$

The accuracy and loss for two and three activities are shown in Tables 1 and 2, respectively. Different optimizers are selected for each type of CNN. In the two-activities test, we have an imbalance between the folders because the slow walk and slow walk with hands in pockets classes are merged. This gives us some fluctuation in the results of the training and validation loss. It is possible to rve that the Inception family of algorithms leads to the best results for the two activities case, especially when the RMSprop optimizer is selected. The small dimension of the dataset deeply affects the results when three different activities are considered, as evident from the high values of the loss function. This reflects the fact that there are not enough images to perform a correct classification, and the model easily tends to overfit. An accuracy of more than 86% is however achieved by InceptionV3 network is SGD optimizer, with an overall contained loss with respect to the other cases.

In Table 3 we report the precision and recall values and the F1 score for three CNNs, i.e., VGG16, ResNet50 and Inception ResNetV2. We select the optimizers which give the best results and corresponds to Adagrad, RMSprop and Adam, respectively. In general, there is no CNN or optimizers which results the best in

all cases, so a suitable configuration must be selected depending on the scenario and the parameters that we want to maximize.

Table 3. Precision, recall and F1 score for the FastWalk and SlowWalk activities.

Model	VGG16			ResNet50			Inception ResNetV2		
	P	R	F1	P	R	F1	P	R	F1
FastWalk	1	0.43	0.6	1	0.57	0.73	0.6	0.43	0.5
SlowWalk	0.75	1	0.86	0.8	1	0.89	0.71	0.83	0.77

Table 4. Execution times required by different algorithms for the three activities case. All times are reported in seconds.

Model	VGG16	ResNet50	Inception ResNetV2	InceptionV3	Xception	VGG19
Adam	30	100	38	55	107	21.04
RMSprop	52	195	75	93	150	42.4
SGD	19	68	24	30	47	17

Finally, in Tables 4 and 5 we compare the execution times required by each algorithm. Indeed, complexity and duration of running algorithms is an important parameter to take into account when dealing with IoT devices, which have limited storage and computational capacity. Algorithms have been tested using a Windows 10 Pro 64-bit (10.0, Build19042.789), Version 20H2, Intel Core I7-5500U, CPU @ 2.40 Ghz RAM 16 GB, Display NVIDIA GeForce 920M 4 GB. In this case there are no evident differences between the cases with two or three activities. In both scenarios SGD takes less time than other optimizers, while VGG19 results the fastest network. SGD in fact is a variation of the classical gradient descent, but it only computes a tiny subset or random selection of data instances, rather than the entire dataset, which is redundant and wasteful. In this way SGD needs less iterations until the objective function converges to the minimum, thus resulting faster than other optimizers, and when the learning rate is modest, it is able to achieve the same results than traditional gradient descent. Adam is instead a gradient-based optimization technique for stochastic objective functions. It computes specific adaptive learning rates for distinct parameters by combining the benefits of two SGD extensions, RMSProp and AdaGrad. Although it has a great popularity, Adam has recently been shown to fail to converge to an optimal solution in some situations, thus requiring in some cases larger execution times.

Table 5. Execution times required by different algorithms for the two activities case. All times are reported in seconds.

Model	VGG16	ResNet50	Inception ResNetV2	InceptionV3	Xception	VGG19
Adam	31.2	196	45.7	57.2	114	24
RMSprop	56.1	104	89	117	215	60
SGD	19	86	39	42.1	73	24

5 Conclusions

Deep learning is a branch of artificial intelligence that is widely applied in different fields. We have considered human activity classification performed by exploiting images derived from an automotive Radar, which represents an emerging contactless technology. We have shown how different pre-trained models built using transfer learning can achieve good results to discriminate different gait velocities. In order to define a solution which is more easily implementable on IoT devices with limited capacity, we have provided a comparison between different neural networks, in terms of accuracy, loss, classification parameters and execution times. The quality of health monitoring and the detection of smaller movements, including the position of hands during walking, can be further improved by expanding the dataset dimension and including the tracking of the subjects and trajectory.

References

1. Bianco, S., Cadene, R., Celona, L., Napoletano, P.: Benchmark analysis of representative deep neural network architectures. IEEE Access **6**, 64270–64277 (2018)
2. Chellapandi, B., Vijayalakshmi, M., Chopra, S.: Comparison of pre-trained models using transfer learning for detecting plant disease. In: 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), pp. 383–387. IEEE (2021)
3. Chollet, F.: Xception: deep learning with depthwise separable convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1251–1258 (2017)
4. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res. **12**(7), 2121–2159 (2011)
5. Gambi, E., Ciattaglia, G., De Santis, A., Senigagliesi, L.: Millimeter wave radar data of people walking. Data Brief **31**, 105996 (2020). https://doi.org/10.1016/j.dib.2020.105996, https://www.sciencedirect.com/science/article/pii/S2352340920308908
6. Haimovich, A.M., Blum, R.S., Cimini, L.J.: Mimo radar with widely separated antennas. IEEE Signal Process. Mag. **25**(1), 116–129 (2007)
7. 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)

8. Instruments, T.: Awr1642 single-chip 77- and 79-ghz fmcw radar sensor. http://w3techs.com/technologies/overview/contentlanguage/all
9. Instruments, T.: Dca1000evm data capture card. http://www.ti.com/lit/ug/spruij4a/spruij4a.pdf
10. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
11. Krieger, G.: Mimo-sar: opportunities and pitfalls. IEEE Trans. Geosci. Remote Sens. **52**(5), 2628–2645 (2013)
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Adv. Neural. Inf. Process. Syst. **25**, 1097–1105 (2012)
13. Maeda-Gutierrez, V., et al.: Comparison of convolutional neural network architectures for classification of tomato plant diseases. Appl. Sci. **10**(4), 1245 (2020)
14. Pan, E.: Object classification using range-doppler plots from a high density PMCW mimo mmwave radar (2020)
15. Ryu, S.J., Suh, J.S., Baek, S.H., Hong, S., Kim, J.H.: Feature-based hand gesture recognition using an FMCW radar and its temporal feature analysis. IEEE Sens. J. **18**(18), 7593–7602 (2018). https://doi.org/10.1109/JSEN.2018.2859815
16. Senigagliesi, L., Ciattaglia, G., De Santis, A., Gambi, E.: People walking classification using automotive radar. Electronics **9**(4), 588 (2020)
17. Shu, M.: Deep learning for image classification on very small datasets using transfer learning (2019)
18. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
19. Sun, S., Petropulu, A.P., Poor, H.V.: Mimo radar for advanced driver-assistance systems and autonomous driving: advantages and challenges. IEEE Signal Process. Mag. **37**(4), 98–117 (2020)
20. Szegedy, C., et al.: Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9 (2015). https://doi.org/10.1109/CVPR.2015.7298594
21. Vandersmissen, B., et al.: Indoor person identification using a low-power FMCW radar. IEEE Trans. Geosci. Remote Sens. **56**(7), 3941–3952 (2018). https://doi.org/10.1109/TGRS.2018.2816812
22. Wang, Y., Ren, A., Zhou, M., Wang, W., Yang, X.: A novel detection and recognition method for continuous hand gesture using FMCW radar. IEEE Access **8**, 167264–167275 (2020). https://doi.org/10.1109/ACCESS.2020.3023187
23. Wu, W., Dasgupta, S., Ramirez, E.E., Peterson, C., Norman, G.J.: Classification accuracies of physical activities using smartphone motion sensors. J. Med. Internet Res. **14**(5), e130 (2012)
24. Zeiler, M.D.: Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)
25. Zhang, L., Wu, X., Luo, D.: Recognizing human activities from raw accelerometer data using deep neural networks. In: 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), pp. 865–870. IEEE (2015)