Smartphone Location Identification and Transport Mode Recognition Using an Ensemble of Generative Adversarial Networks

Lukas Günthermann
L.Gunthermann@sussex.ac.uk
University of Sussex
Sensor Technology Research Centre
Brighton, England

Ivor Simpson
I.Simpson@sussex.ac.uk
University of Sussex
Predictive Analytics Lab
Brighton, England

Daniel Roggen
daniel.roggen@ieee.org
University of Sussex
Sensor Technology Research Centre
Brighton, England

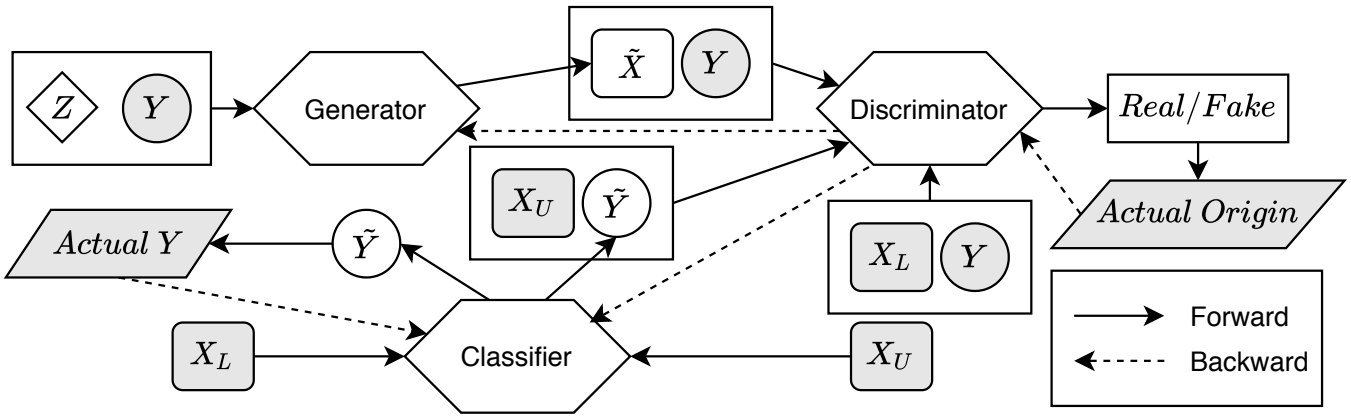


Figure 1: This network was used in the classification process. Dashed backward lines represent target variables, which are compared against predicted ones to perform gradient descent. The discriminator is trained to distinguish between *real* data-label pairs originating from the training data and *fake* data, which includes synthetic data created by the generator and unlabelled data with predicted labels. The discriminator serves as adversarial for the other networks, guiding the generator to create more realistic samples and the classifier to predict more accurate labels. A detailed description can be found in Sec. 3.

ABSTRACT

We present a generative adversarial network (GAN) approach to recognising modes of transportation from smartphone motion sensor data, as part of our contribution to the Sussex-Huawei Locomotion-Transportation (SHL) recognition challenge 2020 as team *noname*. Our approach identifies the location where the smartphone of the test dataset is carried on the body through heuristics, after which a location-specific model is trained based on the available published data at this location. Performance on the validation data is 0.95, which we expect to be very similar on the test set, if our estimation of the location of the phone on the test set is correct. We are highly confident in this location estimation. If however it were wrong, an accuracy as low as 30% could be expected.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning approaches**; • **Human-centered computing** → *Ubiquitous and mobile devices.*

KEYWORDS

Human activity recognition, deep learning, generative adversarial networks, mobile computing

ACM Reference Format:
Lukas Günthermann, Ivor Simpson, and Daniel Roggen. 2020. Smartphone Location Identification and Transport Mode Recognition Using an Ensemble of Generative Adversarial Networks. In *Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM*

UbiComp/ISWC '20 Adjunct, September 12–16, 2020, Virtual Event, Mexico

ACM ISBN 978-1-4503-8076-8/20/09…$15.00
https://doi.org/10.1145/3410530.3414353

International Symposium on Wearable Computers (UbiComp/ISWC '20 Adjunct), September 12–16, 2020, Virtual Event, Mexico. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3410530.3414353

1 INTRODUCTION

The SHL recognition challenge 2020 consists in identifying 8 modes of locomotion and transportation (being still, walking, running, cycling, being in a bus, in a car, on a train or in the subway), based on the motion sensor data in a smartphone. The challenge makes use of a subset of the the SHL dataset [1, 2].

This challenge is an evolution of the previous 2018 [3] and 2019 challenges [4]. This year's edition consists in developing a model that will operate on a phone location on the body which is unknown. It can be one of four locations: smartphone in the hand, in the trousers pocket, in a front shirt pocket, or in a handbag/backpack. The available labelled data include motion data from these 4 locations for a user called user 1; a smaller validation dataset with motion data from these 4 locations for a "target" user, which is a combination of data of user 2 and 3. The unlabelled test data is from the target user, and from an unknown location.

Our contribution to this challenge is:

- A method to identify which is the unknown location of the smartphone in the test data, based on examining similarities in the feature space and the frequency domain.
- Based on this identified location, we train a location-specific model from the train and validation sets. We use for this a generative adversarial network in order to improve our prediction accuracy by including the unlabelled test data into the training.

An ensemble of classifiers is used to prevent overfitting. To our knowledge, this is the first usage of a GAN network on this challenge. We suppose that this approach should perform better than location-independent models if the discovery of the unknown phone location in the target dataset is correct.

2 RELATED WORK

SenseGAN

GANs, as proposed by Goodfellow [5], involve two deep neural networks competing with each other in a minimax game: The generator network G samples noise z from a Gaussian normal distribution to create synthetic samples. The discriminator network G distinguishes between given *real* samples and the *fake* ones created by the generator.

Yao et al. further developed this concept by adding a classifier C into the GAN [6]. The classifier predicts labels for given data points, the generator creates synthetic samples based on a given label, and the discriminator aims to tell apart *real* from *fake* data-label pairs. A data point x is a sequence of sensor recordings and is referred to as "sample" from now on. Together with a corresponding label y, they form a data-label pair, shortened to "pair" from now on.

The training is based on two given datasets: X_L with corresponding one-hot labels Y and data X_U with unknown labels. In addition, during training, additional data is created in two ways: i) A synthetic sample $\tilde{x}$ aiming at resembling the distribution of a given class y is generated by the generator; ii) a sample x_U from the unlabelled dataset is associated with a label $\tilde{y}$ predicted by the classifier, which attempts to trick the discriminator into believing that the prediction is a real label. Only the pairs (X_L,Y) are considered *real* pairs. Unlabelled data with predicted labels (X_U,$\tilde{Y}$) and data generated for given labels ($\tilde{X}$,Y) are considered *fake* pairs.

The overall purpose of this architecture is to improve the classifier performance, by training it to predict labels for samples in a way that fools the discriminator into believing them to be equal to the labelling in the original data. This works better if the discriminator performs well, therefore the generator exists as adversary for the discriminator.

Dataset

The data used in this challenge stems from the SHL Locomotion and Transportation dataset [1]. This dataset is a publicly available dataset designed to evaluate methods to recognise modes of transportation and locomotions from smartphone sensors.

The SHL dataset comprises the recordings of the sensors of four smartphones placed in the hand, in the trouser's pocket (hips), in the shirt pocket (torso), and in a backpack/handbag (bag) of 3 users, who engaged in 8 different transportation and locomotion activities: being still, walking, running, cycling, driving in a car, in a bus, on a train or on the subway. All the motion sensor modalities are sampled at 100Hz. This dataset has been used to understand the information content in the various data channels [2, 7] and it was used in machine learning challenges in 2018 and 2019 [3, 4].

The data used in this work is a subset of the complete dataset called the SHL Challenge 2020 dataset. It contains 20 channels and is provided in the form of 5 second-long windows. These windows are shuffled and therefore do not represent a continuous time series. The channels are: accelerometer, gyroscope, magnetometer, orientation in quaternions, gravity, linear acceleration, and ambient pressure. The training data X_{Train} consists of 196072 windows captured on all four locations of user 1. The validation data X_{Val} offers 28789 windows from both, user 2 and 3, again for all four locations. The testing data X_{Test}, which is supposed to be classified in this challenge, is made out of 57573 windows

from user 2 and 3, captured at an unknown location out of the four ones.

Past Approaches

The SHL challenge in 2018 consisted of training and test data from one user in the form of 1 minute-long windows [3]. The three best submissions consisted of two deep learning (DL) architectures and one classical machine learning (ML) approach. The SHL challenge in 2019 included training, validation, and test data from one user in the form of 5 second-long windows, whereas the test data was located at one position and the training data was only available at the other three locations [4]. This time classical ML approaches drastically outperformed DL approaches, this was concluded to be caused by the mismatch between training and test data. This shows the importance of correctly identifying the sensor location to avoid this mismatch.

We found one approach which could be considered similar to ours: Team DB used an Adversarial Autoencoder to utilize unlabelled test data in the 2019 challenge [8]. The approach did not seem to be successful as it scored a performance of only 31.5% on the test data [4]. To our knowledge, we are submitting the first prediction for the SHL challenge which is based on a GAN architecture.

3 METHODS

Feature Extraction

For each window we compute the mean average, standard deviation, mean-crossing-rate (mcr), kurtosis, and skewness since this simple set of statistical features has shown to enable accurate classification [9]. The resulting feature vector has a length of 100 (20 channels x 5 features). Each feature is normalised individually in the range -1 and 1 at this stage, and the rest of this work (except Frequency Domain Similarity) uses only the normalised feature values.

Location Identification

In order to identify the target location used in the test data we applied two evaluation techniques: One is concerned with similarities in the feature space and the other one with similarities in the frequency domain. The results strongly suggest that 'Hips' is the target location, thus X_{Val} and X_{Train} were reduced to these subsets.

Feature Space Similarity. For this approach, we calculated the mean feature vector across all windows of a location-specific subset. The result were nine vectors, one for the test data ($\bar{x}_{Test}$), four for each location in the validation data ($\bar{x}^{(n)}_{Val}$), and four for each location in the training data ($\bar{x}^{(n)}_{Train}$). We evaluated X_{Val} and X_{Train} independently from each other. We took the four location vectors together with $\bar{x}_{Test}$ and normalised each feature by transforming it into the range between 0 and 1 in order to equalise the significance of all features.

$$d\left(\bar{x}_{Test}, \bar{x}^{(n)}_{Loc}\right) = ||\bar{x}_{Test} - \bar{x}^{(n)}_{Loc}|| \quad (1)$$

We calculated the euclidean distance d between the test vector $\bar{x}_{Test}$ and a subset vector $\bar{x}^{(n)}_{Loc}$ as shown in Eq. 1.

Table 1: Distance d between X_{Test} and subset vectors from datasets (a) X_{Train} and (b) X_{Val}.

(a)

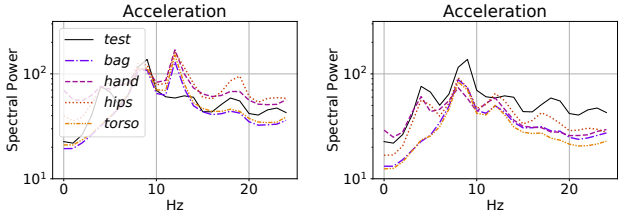
Location	Distance
Bag	6.02
Hand	6.39
Hips	4.85
Torso	5.02

(b)

Location	Distance
Bag	6.39
Hand	5.27
Hips	5.15
Torso	6.31

The results of the validation vector in Tab. 1b show that the locations 'Hand' and 'Hips' are the closest to the test vector, with a minimal advantage of 'Hips'. The results of the less significant training vector in Tab. 1a show that the locations 'Hips' and 'Torso' are the closest to the test vector, with a slightly bigger advantage of 'Hips' than before.

Frequency Domain Similarity. For this approach, we again examined the 9 location subsets. We computed the average power spectrum of the acceleration magnitude across all windows in each subset. The result were 8 figures, each showing the plot of the power spectrum calculated on the motion data of the target location, and the plot of the motion power spectrum calculated on one of the 4 possible locations in the training and validation data.



(a) X_{Train} **(b)** X_{Val}

Figure 2: Comparing the acceleration spectrum of the $Test$ data to known locations.

The graphs were presented to six test subjects with experience in statistics. The graphs were presented in different orders: Three subject were first shown the four validation graphs and asked to simply state which one represents the test graph the best, before they were asked to do the same for

the four training graphs. This process was repeated for the other three subjects, with the only difference that they were shown the training graphs before the validation graphs to prevent question order bias [10]. All subjects gave the same answer for both sets of graphs. Out of the six test subjects, five concluded that the 'Hips' curve resembles the target curve the most. Only one subject, who saw the training graphs first, chose 'Hand'.

GAN Architecture

We implemented a solution based on previous work [9] inspired by SenseGAN [6], which is able to utilise the unlabelled test data and thus improve the accuracy for predicting the labels for the test data. Since there are no continuous time series available, the GAN operates on extracted features of the recorded windows. These extracted feature vectors x serve as data samples for this approach. The GAN architecture can be seen in Fig. 1.

Concept. A data sample x serves as input for the classifier, which predicts a corresponding label $\tilde{y} = C(x)$ in the form of a vector of normalised class probabilities. If the sample originates from the labelled subset X_L, the loss is calculated based on the actual label y as shown in Eq. 2, whereas t refers to one bit of the one-hot label.

$$L^{(C)} = min \sum_{i}^{8} [-t_i \cdot \log(C(x_L)_i)] \tag{2}$$

However, if the sample is part of X_U, the sample x and the predicted label $\tilde{y}$ are combined to serve as input to the discriminator network. In that case the higher the loss of the classifier, the better the ability of the discriminator to classify these samples as *fake* (Eq. 3).

$$L^{(C)} = min[log(1 - D(x_U, C(x_U)))] \tag{3}$$

The generator takes a random uniform distribution z and a one-hot label y as input and generates a data sample $\tilde{x} = G(z, y)$, which is supposed to make the discriminator believe that it is *real*. The length of the vector z is set to 100 (this value is not related to the number of features in x) as it is common practice [11]. The higher the loss of the generator, the better the ability of the discriminator to classify the generated samples as *fake* (Eq. 4).

$$L^{(G)} = min[log(1 - D(G(z, y), y))] \tag{4}$$

The discriminator is a binary classifier, which takes a pair as input and predicts whether it originates from the *real* dataset (x_L,y) or if it is a *fake* pair. A data-label pair is considered *fake* if either the data sample is synthetic ($\tilde{x}$,y) or the label was predicted (x_U,$\tilde{y}$) by the classifier. The loss is calculated as shown in Eq. 5.

$$\begin{aligned} L^{(D)} = max[&log(1 - D(G(z, y), y)) \\ &+ log(1 - D(x_U, C(x_U))) \\ &+ log(D(x_L, y))] \end{aligned} \tag{5}$$

The loss functions shown in Eq. 3-5 are an inverted variant of the binary cross-entropy function, turning the common minimisation problem into a maximisation one for the discriminator.

Parameters. The networks were trained using the Adam optimizer [12]. The hyperparameters are primarily based on the results of previous work [9]. Similarly, we selected the learning rate α and the exponential decay rate for the first moment estimates $\beta 1$ based on the results of a grid-search. For this grid-search we tested various combinations of α and $\beta 1$ values in twelve rounds. Each round would start with testing the generator parameters, followed by the discriminator parameters, and finished with the classifier parameters, since the last one were the most important ones. These rounds were repeated since the parameters might affect each other. One test consisted of three to four α values tested against three to four $\beta 1$ values. After each test the parameters were updated and the search space redefined based on the outcome, resulting in a manually crafted random search.

Classifier. The classifier network *GAN-C* consists of an input layer with a size of 15 (three channels with each five features), one hidden layer with 256 nodes and an output layer with a size of 8 (representing each of the eight classes). The input and hidden nodes are activated by a ReLu function. The one-hot label is determined by a Gumbel-Softmax activation function [13], allowing random sampling based on the class probabilities and discrete output values. Discrete output values are crucial, since the discriminator could otherwise identify continuous labels and distinguish them from discrete *real* labels. The temperature of the Gumbel-Softmax was set to 1, since it seemed like a good trade-off between the bias in approximation and the variance in gradients. During training the following parameters for the Adam optimizer were established: $\alpha = 0.003$, $\beta 1 = 0.9$, $\beta 2 = 0.999$. The classifier accuracy A_C is evaluated on a provided validation dataset (not necessarily X_{Val}).

Generator. The generator network *GAN-G* consists of an input layer with a size of 108 (random vector with length of 100 plus eight one-hot labels), one hidden layer with 256 nodes and an output layer with a size of 15 (a sample $\tilde{x}$ consisting of three channels with five features each). The input and hidden nodes are activated by a LeakyReLu function. The output features are generated via the hyperbolic tangent function to resemble the normalized data distribution between -1 and 1. During training the following parameters for the Adam

optimizer were established: $\alpha = 0.0005$, $\beta 1 = 0.5$, $\beta 2 = 0.999$. The generator accuracy A_G is based on the ratio of generated samples which are classified as *real* by the discriminator.

Discriminator. The discriminator network *GAN-D* consists of an input layer with a size of 23 (15 features plus eight one-hot labels), one hidden layer with 256 nodes and an output layer with a size of 1 (binary representation of *real* / *fake*). The input, hidden and output nodes are activated by a LeakyReLu function. During training the following parameters for the Adam optimizer were established: $\alpha = 0.0125$, $\beta 1 = 0.75$, $\beta 2 = 0.999$. The discriminator accuracy A_D is based on the ratio of generated samples which are classified as *fake* (25%), the ratio of unlabelled samples with predicted labels classifies as *fake* (25%), and the ratio of pairs (x_L, y) classified as *real* (50%).

Training Setup

We convert the activity classes c of X_L into one-hot labels y. Due to potential class imbalances in the data, we oversample the minority classes in X_L using SMOTE [14].

Since X_{Val} stems from the same two users as X_{Test}, we valued the significance of X_{Val} way above X_{Train}. Therefore X_{Val} will serve as X_L and X_{Test} as X_U, so that *GAN-C* learns from *GAN-D* how to label the target data. In order to not waste the potential of X_{Train}, we used it to pretrain *GAN-C*. The pretraining was done with 25 epochs of label prediction, that time period achieved the peak accuracy of 50.1% validated on X_{Val}.

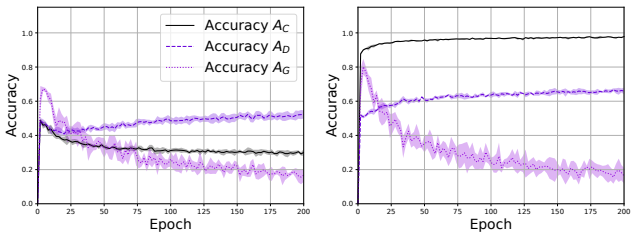


Figure 3: The development of accuracy over the training period. A_C is evaluated on (a) X_{Train} and (b) X_{Val}.

Validated both on X_{Val} and X_{Train} (Fig. 3), 50 epochs of training were established to be optimal. At that point the accuracy on X_{Val} has not converged towards 1.0 too much yet (Fig. 3b) and still maintains some of its initial pretrained accuracy on X_{Train} (Fig. 3a).

The 57573 samples in X_{Val} were extended by 18923 samples synthesised using SMOTE. In order to balance out the influence of random disturbances, the final prediction of the test set was based on an ensemble of 20 classifiers, each independently pretrained on X_{Train} and further trained with individual generator and discriminator networks on X_{Val} as X_L and X_{Test} as X_U. We added up their continuous Gumbel Softmax output and selected the argmax as their collective assessment.

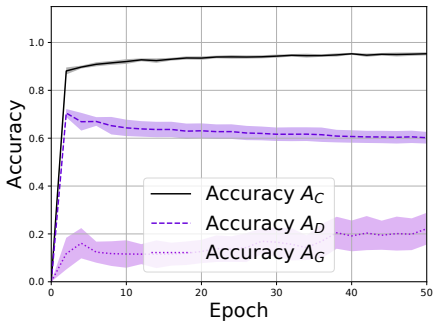


Figure 4: The development of accuracy over the training period. A_C is evaluated on X_{Val}.

Development

The GAN was created using pytorch, the source code for this project can be found on Github [15]. Training the final model took 3 hours on a Intel quad-core i7-6700 3.4GHz CPU and a Nvidia GeForce RTX 2080 Super GPU.

4 RESULTS

Table 2: Prediction accuracy of the final classifier ensemble on (a) X_{Train} and (b) X_{Val} after 50 epochs of training.

(a)

Location	A_C	σ
Bag	0.21	0.015
Hand	0.23	0.013
Hips	0.32	0.017
Torso	0.26	0.021

(b)

Location	A_C	σ
Bag	0.33	0.029
Hand	0.30	0.029
Hips	0.95	0.007
Torso	0.33	0.030

The results in Tab. 2 show that our prediction accuracy is highly depended on the correct identification of the location. Based on the results on X_{Val} in Tab. 2b, we can expect an accuracy as low as 0.3 if our identification is wrong, whereas a correct identification should result in an accuracy of no higher than 0.95, since the measured 0.95 were evaluated on the same data the network was trained on.

As can be seen in Fig. 4 the development of A_C and A_D has mostly stabilised towards the end of the training, while A_G is still increasing.

5 DISCUSSION

While the accuracy of *GAN-G* in the trial runs shown in Fig. 3 has exploded initially to slowly decrease from there on, the accuracy in the final run shown in Fig. 4, surprisingly shows a steady increase in accuracy.[1] Since there were no fundamental changes in the training methods in between these runs, the difference can only result from changes in α and $\beta 1$ caused by the constantly ongoing grid-search (each of the 12 rounds took 2-3 days). This search could very likely be continued further to improve the quality of the GAN, although it is challenging to find a equilibrium between the parameters of three networks depending on each other. Instead of a grid-search, a heuristic approach such as HyperNEAT [16] could be used.

Another approach worth investigating would be training all three networks with half of X_{Train} as X_L and the other half as X_U, or potentially forgoing the classification of unlabelled samples for the pretraining. The latter approach would involve training *GAN-D* and *GAN-G* independently from *GAN-C*, very much like the original GAN setup described in Sec. 2. However, due to potential unforeseen consequences like getting stuck in a local minima in the form of a equilibrium between *GAN-D* and *GAN-G* caused by training on X_{Train}, this approach was considered too risky without proper prior evaluation. Although X_{Val} could have been split into a training and a validation subset, we assumed that this would result in a significant decrease in classification accuracy due to the small size of the dataset.

Our approach is a location-specific activity recognition model, which is customized for the specific circumstances of this challenge and depends on the correct identification of location to achieve proper prediction accuracy. However, the training setup could easily be changed into a location-independent model, for instance by training with data from all four locations and develop a more general applicable classifier.

6 CONCLUSION

To our knowledge, this work is the first using a GAN and an ensemble of deep learning models on the SHL recognition challenge. One unique aspect of this work is that we attempted to create a smartphone location-specific model, instead of a location-independent model. Both approaches would be suitable as the location of the smartphone is unknown, however a location-specific model may achieve higher performance.

We are highly confident in the location identification, as two distinct methods operating in the feature space and frequency domain both indicated that the target location is likely to be 'Hips'. Assuming the identification is correct, as we believe, then we estimate the performance on the test set to be similar to the performance on the validation set, i.e. 0.95. The recognition result for the testing dataset will be presented in the summary paper of the challenge [17].

[1] Generator performance is not a particular meaningful metric since it only exists in the adversarial relation to discriminator performance.

ACKNOWLEDGMENTS

We acknowledge Nvidia for their donation of a TITAN Xp.

REFERENCES

[1] H. Gjoreski et al.: The University of Sussex-Huawei Locomotion and Transportation Dataset for Multimodal Analytics With Mobile Devices. IEEE Access **6** (2018) 42592–42604
[2] L. Wang et al.: Enabling reproducible research in sensor-based transportation mode recognition with the Sussex-Huawei dataset. IEEE Access **7** (2019) 10870–10891
[3] L. Wang et al.: Summary of the Sussex-Huawei Locomotion-Transportation Recognition Challenge. In: Proc. ACM Int Joint Conf and 2018 Int Symp on Pervasive and Ubiquitous Computing and Wearable Computers, ACM (2018) 1521–1530
[4] L. Wang et al.: Summary of the Sussex-Huawei locomotion-transportation recognition challenge 2019. In: Adjunct Proc. 2019 ACM Int Joint Conf on Pervasive and Ubiquitous Computing and 2019 ACM Int Symp on Wearable Computers, ACM (2019) 849–856
[5] I.J. Goodfellow et al.: Generative adversarial networks. arXiv preprint (2014) arXiv:1406.2661
[6] S. Yao et al.: Sensegan: Enabling deep learning for internet of things with a semi-supervised framework. Proc. ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies **2**(3) (2018) 1–21
[7] L. Wang et al.: Benchmarking the SHL recognition challenge with classical and deep-learning pipelines. In: Proc. ACM Int Joint Conf and 2018 Int Symp on Pervasive and Ubiquitous Computing and Wearable Computers, ACM (2018) 1626–1635
[8] D. Balabka: Semi-Supervised Learning for Human Activity Recognition Using Adversarial Autoencoders. In: Adjunct Proc. 2019 ACM Int Joint Conf On Pervasive and Ubiquitous Computing and Int Symp on Wearable Computers, New York, NY, USA, Association for Computing Machinery (2019) 685–688
[9] L. Günthermann, A. Philippides, and D. Roggen: Improving smartphone based transport mode recognition using generative adversarial networks. (in press) (2020)
[10] A. Blankenship: Psychological difficulties in measuring consumer preference. Journal of Marketing **6**(4_part_2) (1942) 66–75
[11] A. Radford, L. Metz, S. Chintala: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint (2015)
[12] D.P. Kingma, J. Ba: Adam: A method for stochastic optimization (2014)
[13] E. Jang, S. Gu, B. Poole: Categorical reparameterization with gumbel-softmax (2016)
[14] N.V. Chawla et al.: Smote: Synthetic minority over-sampling technique. J. Artif. Int. Res. **16**(1) (2002) 321–357
[15] L. Günthermann, I. Simson, D. Roggen: SHL GAN (2020) https://doi.org/10.5281/zenodo.3921803.
[16] K.O. Stanley, D.B. D'Ambrosio, J. Gauci: A hypercube-based encoding for evolving large-scale neural networks. Artificial Life **15**(2) (2009)
[17] L. Wang et al.: Summary of the Sussex-Huawei Locomotion-Transportation Recognition Challenge 2020. In: Proc. 2020 ACM Int Joint Conf and Int Symp on Pervasive and Ubiquitous Computing and Wearable Computers, ACM (2020)