# Practical work

As stated in the introduction, this paper aims to generate radar micro-Doppler responses of human movement using GAN for better and faster training of new motion classifiers. This chapter will describe in detail the procedure for processing radar signals and their preparation for the dataset, as well as the restrictions imposed by the pre-trained motion classifier. After that, the architecture of the used GAN will be described, as well as the improvements of the training algorithm. The last paragraph of this chapter contains the experimental results obtained using a pre-trained classifier for classifying movements. These motion classifiers would then be used at bus stops where they would detect pedestrians endangering traffic safety and send warning signals to oncoming drivers.

# Dataset preparation

The pre-trained movement classifier introduces certain limitations of the shape and values of the training data. The input to the classifier should be a spectrogram obtained by a Short-time Fourier Transform (STFT) done in 128 points with a duration of 0.5s.

As the sampling frequency of the radar is 2.4 kHz, we get 1200 samples for 0.5s of signal duration. The classifier is implemented to classify the input signal into three classes:

* No movement
* Start of movement
* Movement

For such a robust classification, not all time samples are required, and spectrograms that are undersampled in time with a factor of 4 are brought to the classifier input, and the dimensions of the spectrograms are reduced from 128x1200 to 128x300. This reduction of spectrogram dimensions, in addition to reducing the memory required for their storage, also improves the generation of synthetic data because the spectrogram dimensions are more proportionate (which will be discussed in more detail in the following sections). Except for the change in the dimensions of the spectrogram, spectrogram values were normalized using formula 5.3. The motive for normalization of values is the stability and performance of GAN because without normalization to the interval [-1,1] the difference between the minimum and maximum signal value is of the order of, which significantly complicates data generation.

(5.1)

(5.2)

(5.3)

# Radar data

As mentioned earlier, training data for the GAN network was obtained from radar signals recorded as part of the HORIS project within the Fraunhofer Institute [1] [2]. The data was recorded with the FMCW radar, whose carrier frequency is 24 GHz, with a bandwidth of 250 MHz, duration of the modulated pulse (chirp) of 50μs and PRI 52μs. From the previous data, it can be calculated that this radar has a range resolution ΔR of 0.632 m. This radar recorded 60,000 pulses, of which a radar response of a total duration of 31.2 seconds was later compiled.

Radar data was collected in a controlled environment with three different scenarios. For each scenario, fifteen measuring sessions were taken. In each scenario, there is a person who initially stands still and then moves towards the radar. When the person reaches a marked place near the radar, he stops, turns around, and returns to the starting position. The difference between these scenarios is the distance of the person moving from the radar, where for the first scenario the distance is 5m, the second 10m, and the third 15m. Figure 5.1 shows a simulated walk of a human skeleton using information about a person's distance obtained during the shooting of the second scenario.

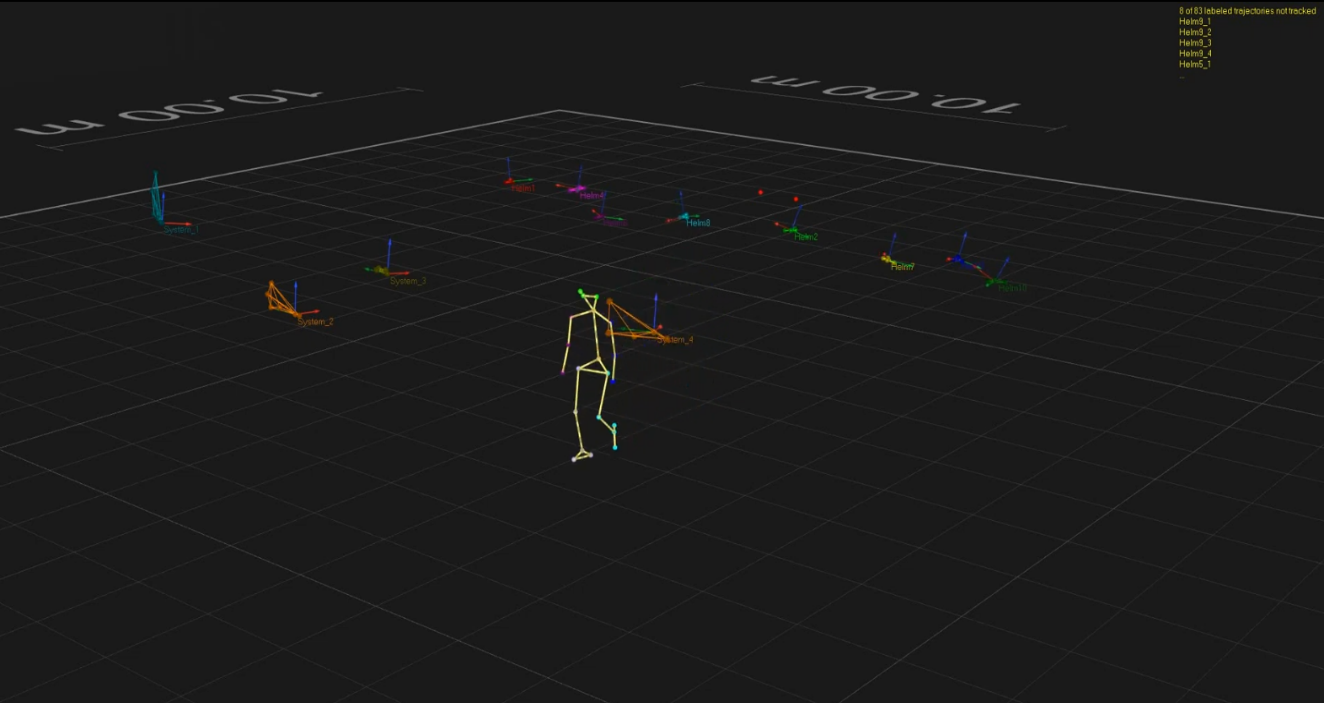


Figure 5.1 – *Human walk simulation* [2]

Each radar signal is represented by a four-dimensional complex array. The data within the signal is divided into three-dimensional cells representing distance, elevation, and azimuth while the last dimension of the array represents the time for a given three-dimensional cell. The size of the vector, ie. the number of cells into which one radar signal is divided depends on the radar and its resolution. For this particular radar there are 17 azimuth cells (with a resolution of 3.97 °), 4 elevations cells, and 74 range cells (with a resolution of 0.632 m) which with 60,000 samples in time. makes arrays of fixed size [17,4,74,60000].

Since one range cell represents the space of 0.632 m through which the person moves, the distance of the person from the radar is obtained by multiplying the ordinal number of the cell by the range resolution. Thus, for the first scenario, the range cells of interest are cells 2 to 9 (the recording was not done precisely and in many cases, the subject stood a little further than 5 m in relation to the radar). For the second scenario, interest cells are cells 8 to 17 and for the third scenario, cells 17 to 25. Due to the distance of the subject in the second and third scenarios, the signal-to-noise ratio (SNR) is lower compared to the first scenario. This can be seen in Figures 5.2, 5.3, and 5.4 where the beginning of the movement is shown, ie. range cells 8, 17, and 25, respectively. Due to the large impact of the noise, the last two scenarios were rejected as candidates for generating the training dataset, and so only the first scenario and range cells 2-9 were taken for the training dataset.

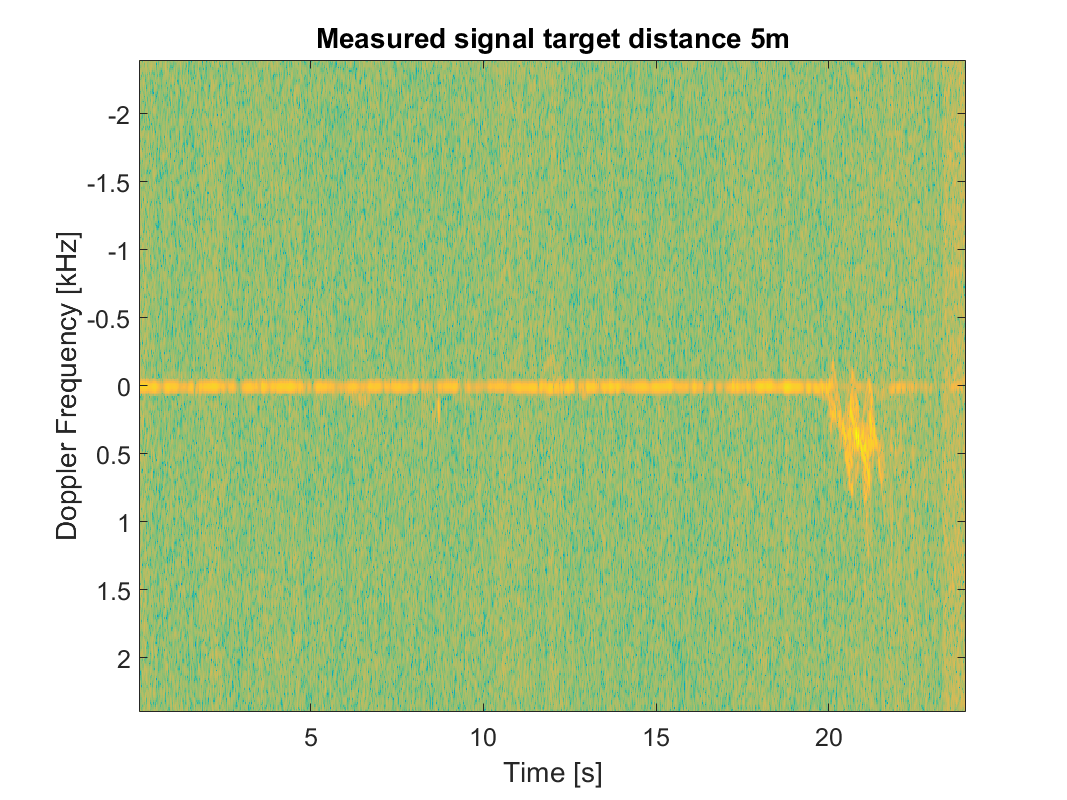


Figure 5.2 – *Micro-Doppler response of the first scenario (5m distance)*

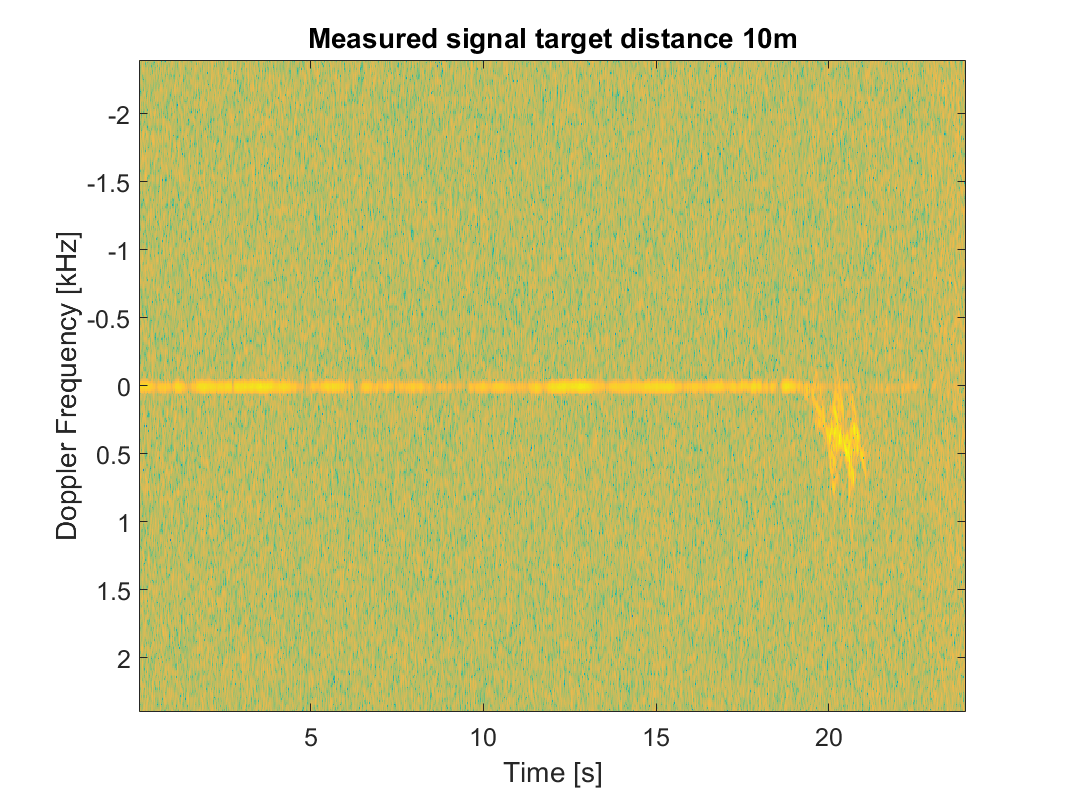


Figure 5.3 – *Micro-Doppler response of the second scenario (10m distance)*

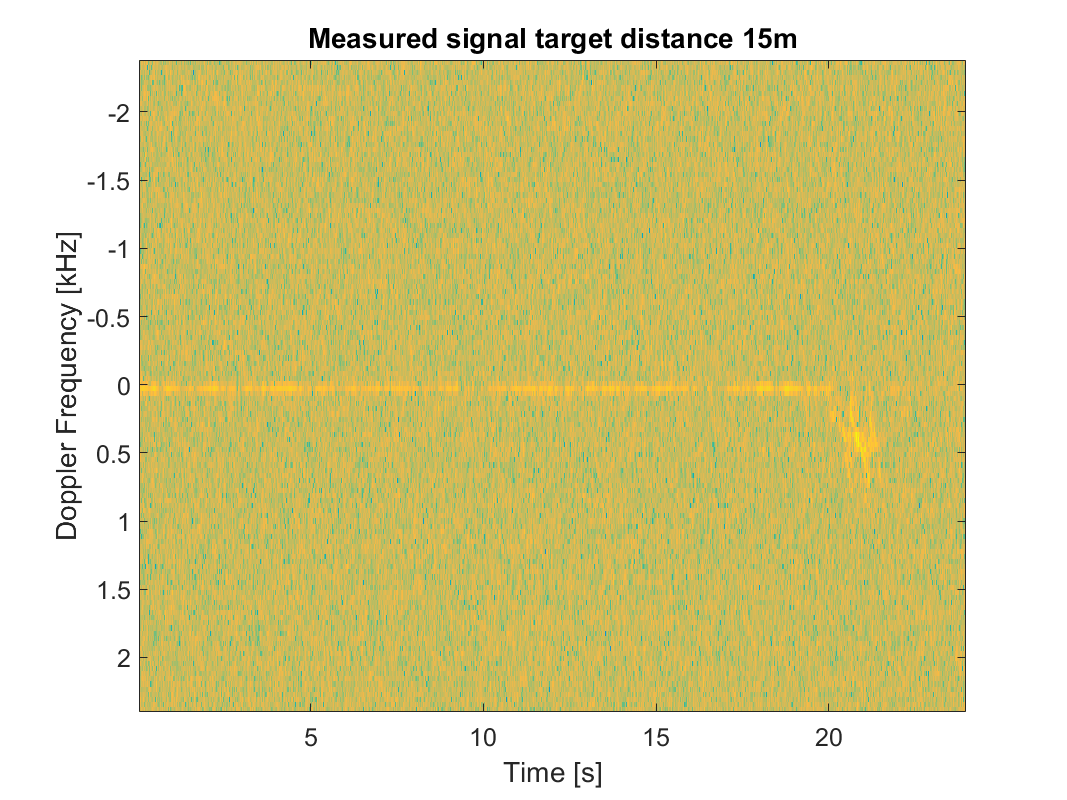


Figure 5.4 – *Micro-Doppler response of the third scenario (15m distance)*

Except for the selection of range cells, it was necessary to select the appropriate azimuth and elevation cells. During the movement of the person, it walks through more than one azimuth and elevation cell, similar to the case of different scenarios, the impact of the noise is much greater in cells that the person only partially occupies. In Figure 5.5 the influence of the selection of different azimuth cells for the same elevation and range cells can be seen. From Figure 5.5 it can be concluded that in addition to the influence of noise on the response, the cells that person partially occupies have less pronounced response amplitudes, ie. the response is "less rich" in micro-Doppler frequencies. Due to all the above, azimuth cells, 8,9 and 10 were chosen for the dataset.

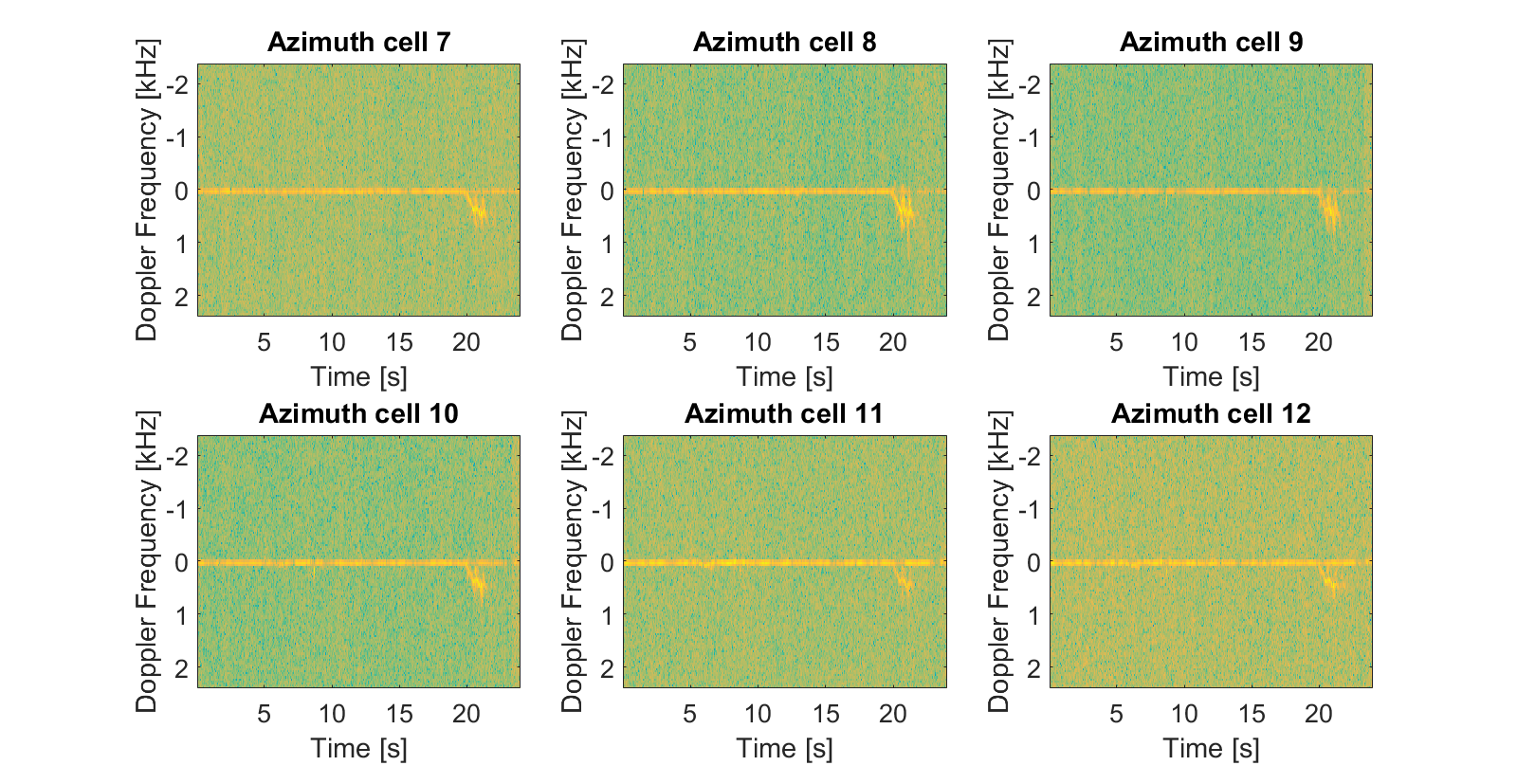


Figure 5.5 – *Selection of azimuth cells*

In Figure 5.6, spectrograms for different elevation cells are shown. The figure shows that elevation cell 3 has the clearest micro-Doppler response (most sinusoids in response) and is the least affected by noise, so for generating dataset signals from elevation cell 3 are used.

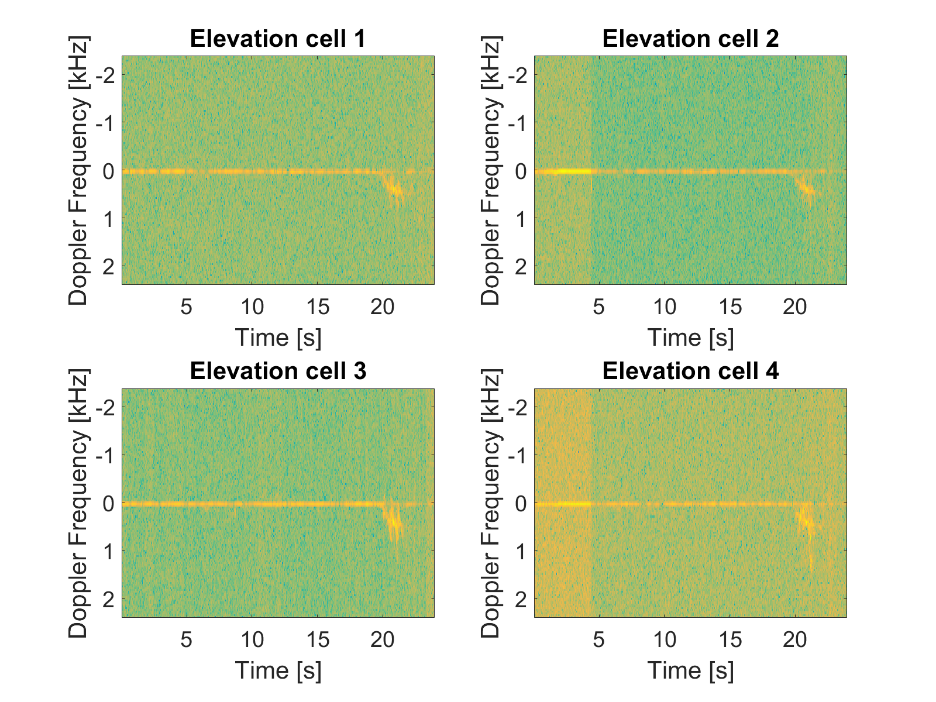


Figure 5.6 – *Selection of the elevation cell*

# Preprocessing of the dataset

In the previous point, spectrograms of raw data obtained from radar were shown without pre-processing except for shifting the frequency axis so that the Doppler frequency is equal to zero, to more easily detect the exact frequencies in the micro-Doppler response. From Figures 5.5 and 5.6, it can be seen that most of the time the person is at rest and there is no movement on the spectrogram, so it is necessary to discard that part of the signal and use only the part where there is a movement for the training set. Since it is necessary to cut the signal at different times, which is additionally disturbed by noise, the extraction of the part of the signal that contains movement is not a trivial task and therefore a special algorithm for motion detection has been developed.

Before applying the motion detection algorithm, it is necessary to prepare the signals over which it will be applied. Therefore, it is necessary to extract useful data from the raw data and transform them into the frequency domain using STFT in 128 points. Significant cells that contain motion data were previously selected and they include range cells from 2 to 9, azimuth cells 8,9, and 10, and elevation cell number 3. In this way, by extracting useful data from one radar measurement, 24 signals are obtained over which the transformation can be done, which raises the total number of usable signals from 15 to 360. After STFT, it is necessary to normalize the obtained spectrograms using formula 5.1 and save the normalized spectrogram for the application of the algorithm.

After the normalization of the data, a motion detection algorithm is applied. The idea of ​​the algorithm is to use the fact that the motion on the spectrograms is represented by sinusoids of the micro-Doppler effect, which at the shifted frequency axis are at frequencies from 0 Hz to half the sampling frequency or 1200 Hz. Ideally, without the presence of noise, the spectrogram consists of a single line representing the human walking Doppler frequency and the sinusoids of the micro-Doppler effect, and positive values ​​are found in the spectrogram at these locations. Finding the moment in which motion occurs comes down to finding positive values ​​at the appropriate frequencies in the spectrogram. When it comes to a measured signal, ie a signal disturbed by noise, it is to be expected that there are positive values ​​at other moments besides moments of the movement, at the observed frequencies. What distinguishes noise and motion is the fact that noise is random and it is unlikely that positive values ​​appear on adjacent frequencies for a long period of time on the spectrogram, as is the case with the motion that lasts a certain period of time and whose micro-Doppler effect consists of sinusoids that are also continuous on the frequency axis. In addition to the continuity of the micro-Doppler effect, the movement on the spectrogram is much more pronounced than noise (due to high SNR at a distance of 5 m) and if there are several positive values ​​at one observed frequency (due to noise), they are sorted descending by amplitude (value) and tested whether at adjacent time points on adjacent frequencies there are local maxima or positive values. If they exist, then it is most likely a matter of movement and these values ​​are connected into one unit and that unit becomes a candidate for movement, and if there are no positive values ​​on adjacent frequencies then that positive value or local maximum is due to signal noise and that point is rejected.

After finding all the units that are candidates for movement, we sort them by the times in which they occur and neighboring units merge into larger units if the difference between the end of one and the beginning of the other is less than a predetermined step value, which in this case was 0.1 s. Upon completion of the merging of adjacent units, usually, only one large unit is obtained, which represents the movement of the subject. In some cases, several smaller units can be obtained as a result of the algorithm (if the spectrogram is severely disturbed by the noise), then the algorithm is repeated with a higher step value for merging units until we get one unit or until the maximum step value for merging units is reached. If the maximum set value for merging units is reached, then the largest unit is selected as a unit representing the movement, and other smaller units are discarded. The start and end times of the selected unit are further expanded by a certain time interval (so-called buffer interval) to ensure that the motion detection algorithm includes those parts of the cell where the response intensity is slightly lower (start and end). The results of the algorithm are shown in Figures 5.7 and 5.8, where the red vertical lines delimit the whole representing the motion.

Figure 5.7 shows a part of the motion that is not covered by the motion detection algorithm. The reason for this is the break in the spectrogram and the significantly smaller amplitude of the left out part compared to the rest of the spectrogram. In Figure 5.8 we can see that the beginning of the whole does not contain a micro-Doppler effect and this is a consequence of moving, ie expanding the borders of the unit with additional time (buffer) interval.

After detecting the movement in spectrograms and extraction of that part of the spectrogram, we can proceed with the creation of the dataset. This involves cutting to the appropriate dimensions 128x300 and classifying the movement into one of two possible classes: "start of movement" and "movement". Cutting of the spectrograms was performed by dividing the spectrograms into 0.5s segments that have an overlap of 0.4s, after which each segment was classified separately and further processed. Overlapping allows more training data to be obtained from a very small initial data set.

Classification of the data was done manually, based on the spectrogram it was determined when the movement of the subject passed from the initial phase to a stable walking and this was repeated for all 15 recordings. Based on these times, segments are classified into the appropriate class if more than 50% of the segment belongs to one of the classes. Then it was necessary to additionally process the classified segment, first, it was necessary to examine whether the observed segment contains the micro-Doppler effect and if it does not contain it, it was removed from the dataset. This examination was performed by checking if there are positive maxima at frequencies other than Doppler frequency. The segment was then undersampled with a factor of 4 and stored together with its class in the training set dictionary. With undersampling one segment, four different training data for GAN were created and thus the total number of available training data was significantly increased.

With the use of a motion detection algorithm, undersampling and overlapping segments in a time out of 360 signals, 17067 training data was generated, of which 11719 belongs to the movement class while 5348 belongs to the start of movement class. This disproportion was expected because segments from only three range cells (2, 8, and 9) can be characterized as the start of the movement. Figure 5.9 contains plots of a few training data with their classes.

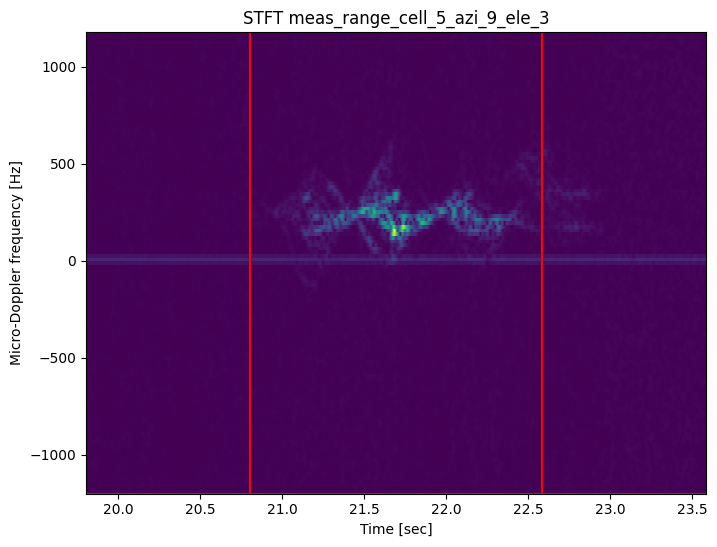


Figure 5.7 – *Output from movement catcher algorithm*

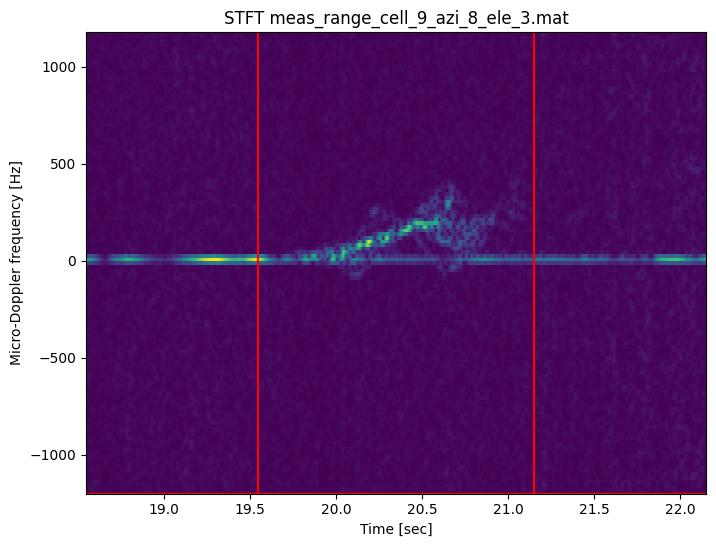


Figure 5.8 – *Output from movement catcher algorithm*

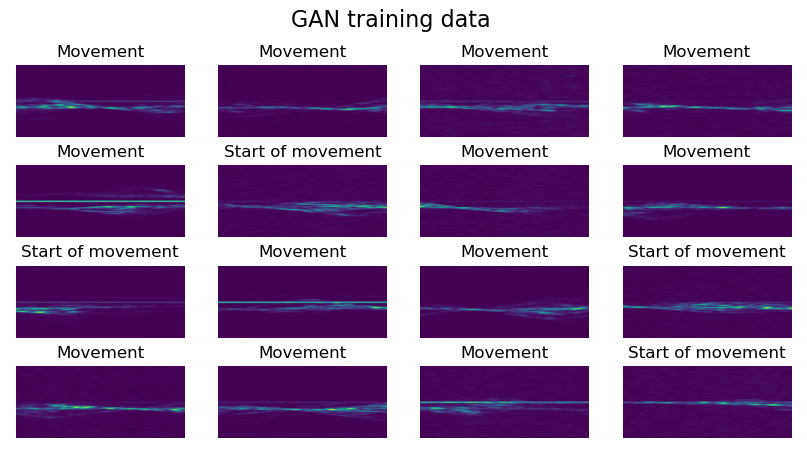


Figure 5.9 – *Dataset*

# Used network

In this section, the reasoning behind the used architecture and improvements of the training algorithm is explained in detail.

# Final GAN architecture

The foundation of the used network is a conditional DCGAN network. In order to find an architecture that gives satisfactory results in terms of generated data and training time, a few problems had to be overcome.

Firstly, the way of representing the condition had to be solved. Since there are only two data classes in the dataset, it is not necessary to make one-hot encoding vectors for their representation, it is enough to use one integer variable that has a value of 0 when the data belongs to the class "start of movement" and 1 when the data belongs to the "movement" class. After choosing the representation of the condition, it is necessary to determine one of the hyperparameters of the network, and that is the number of neurons in the embedding layer. The selected value is 75 neurons and its choice was empirical and for this value, consistently correct results were obtained regardless of the rest of the architecture.

The second problem was finding a generator architecture that does not create block artifacts when generating spectrograms. According to the recommendations from the literature [15] [20], during the training of DCGAN networks, it is necessary to increase the generated image (spectrogram in this case) in several steps and in proportionally in both dimensions, until the desired resolution is obtained. The reason for this is the gradual learning of the image features, and since the desired resolution is 128x300, the largest common divisor of the dimensions is 4, which allows a maximum of two equal increases in dimensions, where the initial dimension of the image is too large and is 32x75. Increasing the dimensions of the image within the convolution layer is done using transposed two-dimensional convolution and depends on the steps of convolution over the image [21]. Due to this limitation, it was necessary to find a different way to obtain a gradual increase in dimensions in several steps where the network will have the appropriate depth (greater than 2 layers) and will not have block artifacts at the generator output. One way is to use a convolution step that is different for each dimension but has approximate values ​​(e.g. 4x5). An example of one such generator architecture is given in Figure 5.10 and the results obtained using such an architecture are shown in Figure 5.11.

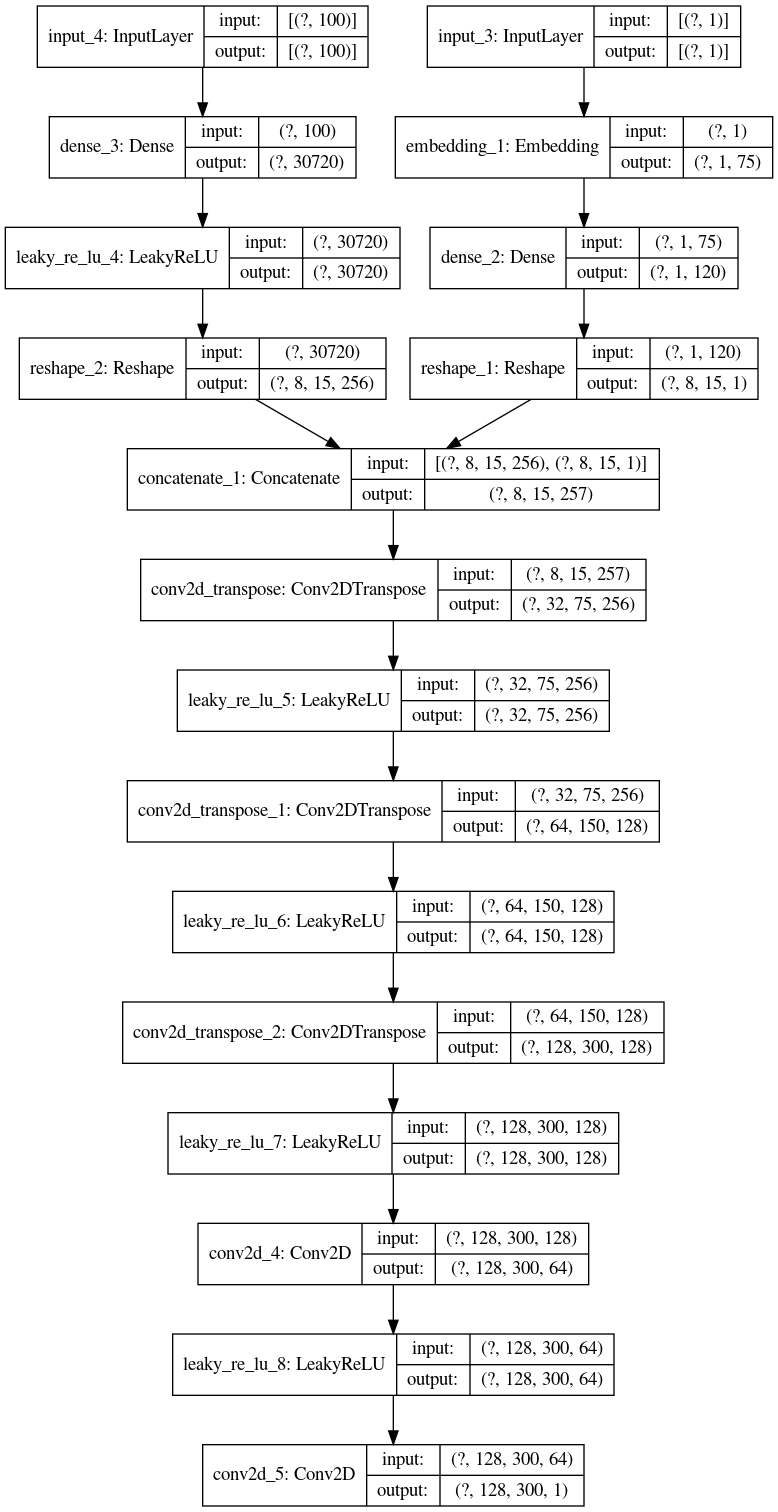


Figure 5.10 – *Architecture of the generator using uneven convolution step*

Figure 5.11 shows block artifacts in the form of vertical lines and square transitions within the spectrogram. Such artifacts were also present for different architectures that had unequal increases in image dimensions.

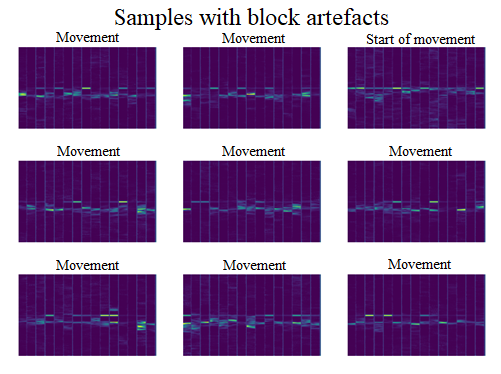


Figure 5.11 –*Generated spectrograms with block artefacts*

The second way to achieve the appropriate dimensionality of the output data is by changing the shape of one convolutional layer within the model (most often it is one of the last two layers). This approach allows the dimensions of the image to be changed without the use of convolution, which makes it possible to obtain dimensions that are difficult to achieve using transposed 2D convolution. The key for using *Reshape* layer is the number of the maps within the convolutional layer because the required number of maps to resize is calculated based on the least common denominator of the current and desired image dimensions.

An example of such a generator architecture is given in Figure 5.12, and the results obtained using such an architecture are shown in Figure 5.13. From Figure 5.12 it can be seen that the initial dimension of the image is 8x8 which is doubled in each subsequent layer until it reaches the dimensions 256x256 after which the image is reshaped into an image of dimensions 128x300.

The spectrograms shown in Figure 5.13 do not have block artefacts and correspond to the appearance of the spectrograms shown in Figure 5.9, which is why this generator architecture is retained in the final model.

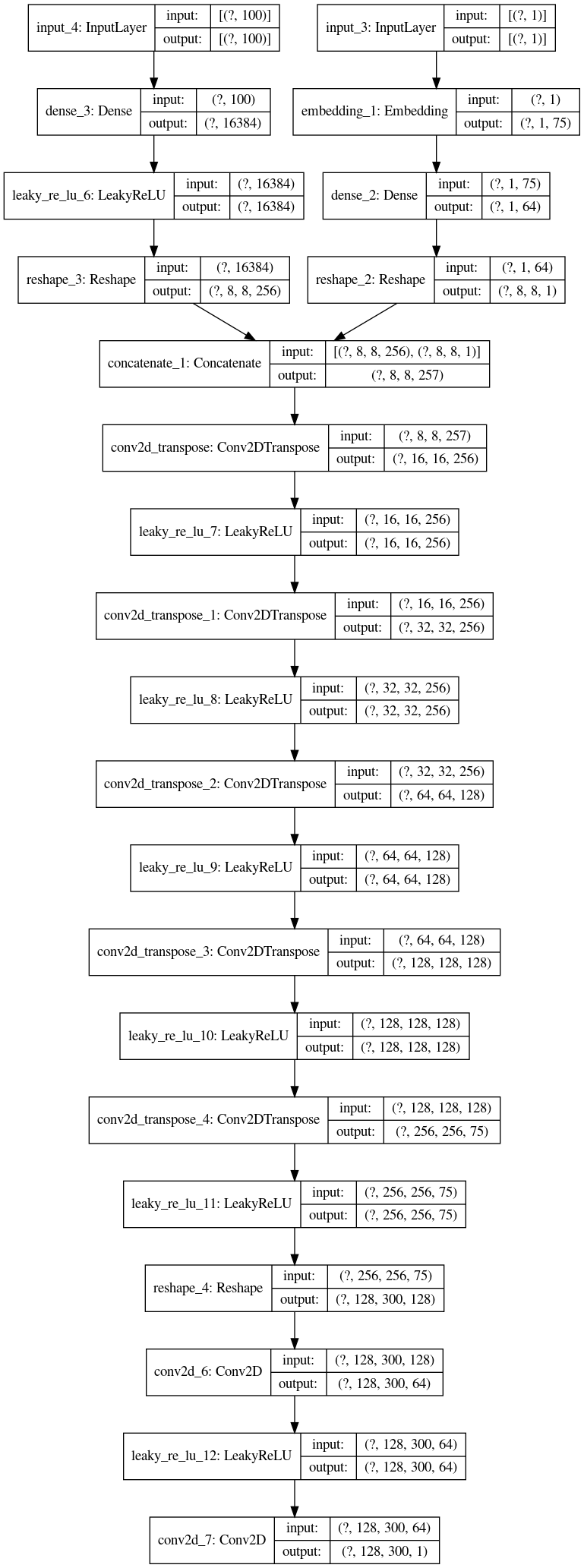


Figure 5.12 –*Architecture of the generator using even convolution step*

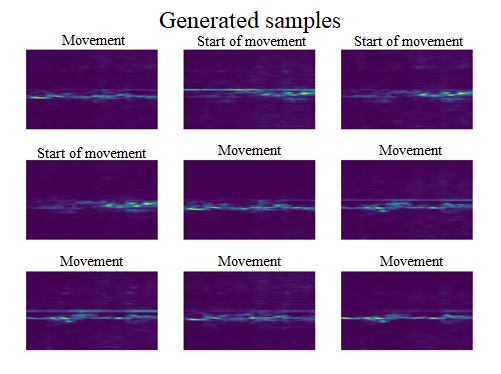


Figure 5.13 –*Generated spectrograms without block artefacts*

After the selected generator architecture, it is necessary to select discriminator architecture and generate the entire GAN network. Although the literature recommends that the architecture of the discriminator should be symmetric to the architecture of the generator, it has been shown that the use of such an architecture produces worse results compared to the use of an asymmetric architecture of the discriminator.

The discriminator architecture symmetric to the generator from Figure 5.12 is shown in Figure 5.14. It can be seen from the figure that the discriminator first increases the dimensions of the input image and then gradually decreases them to ensure symmetry with the generator. This adds a single redundant layer and increases the complexity of the entire network by adding 300,000 new parameters which further burdens the hardware on which the network is trained. In addition to the larger network model, it has been shown that a network that uses this discriminator architecture becomes overfitted faster, ie. it begins to generate one spectrogram for each of the classes regardless of the input vector passed to the generator. For these reasons, the discriminator architecture shown in Figure 5.15 was chosen. A network that uses this architecture takes twice as long to overfits, with the generated spectrograms being much clearer and richer in detail.

The architecture of the entire GAN network used is shown in Figure 5.16 and it consists of the generator shown in Figure 5.12 and the discriminator shown in Figure 5.15.

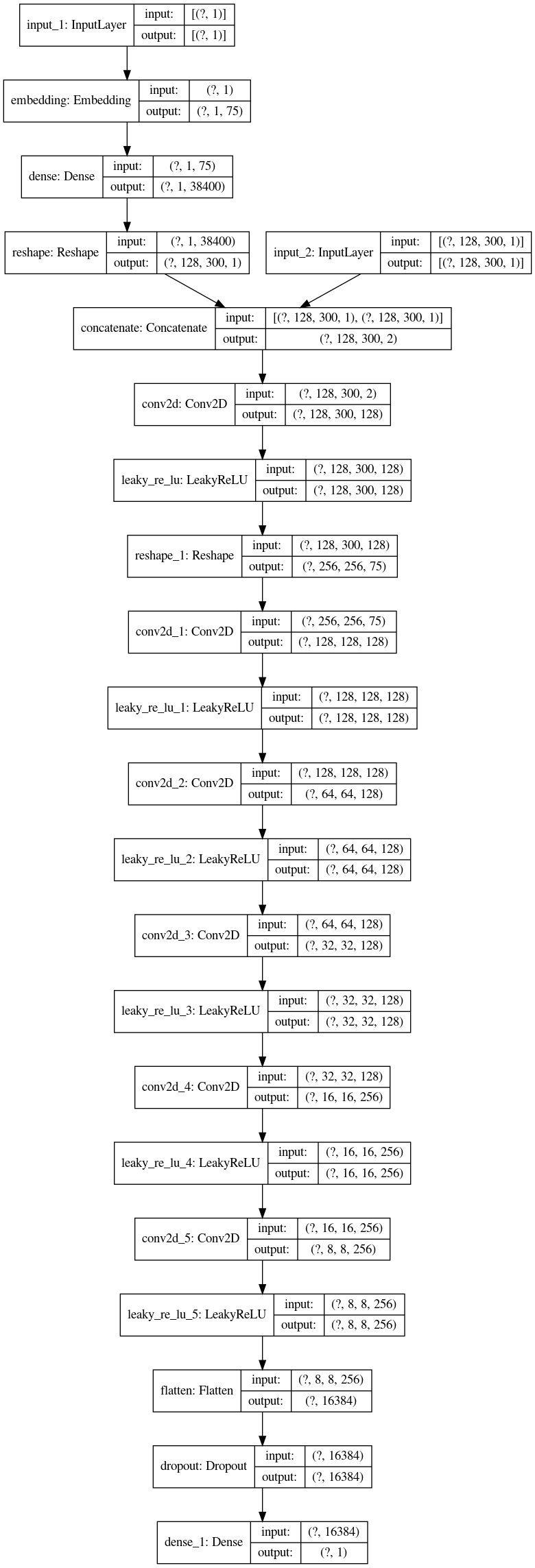


Figure 5.14 –*Architecture of the discriminator symmetrical to the generator*

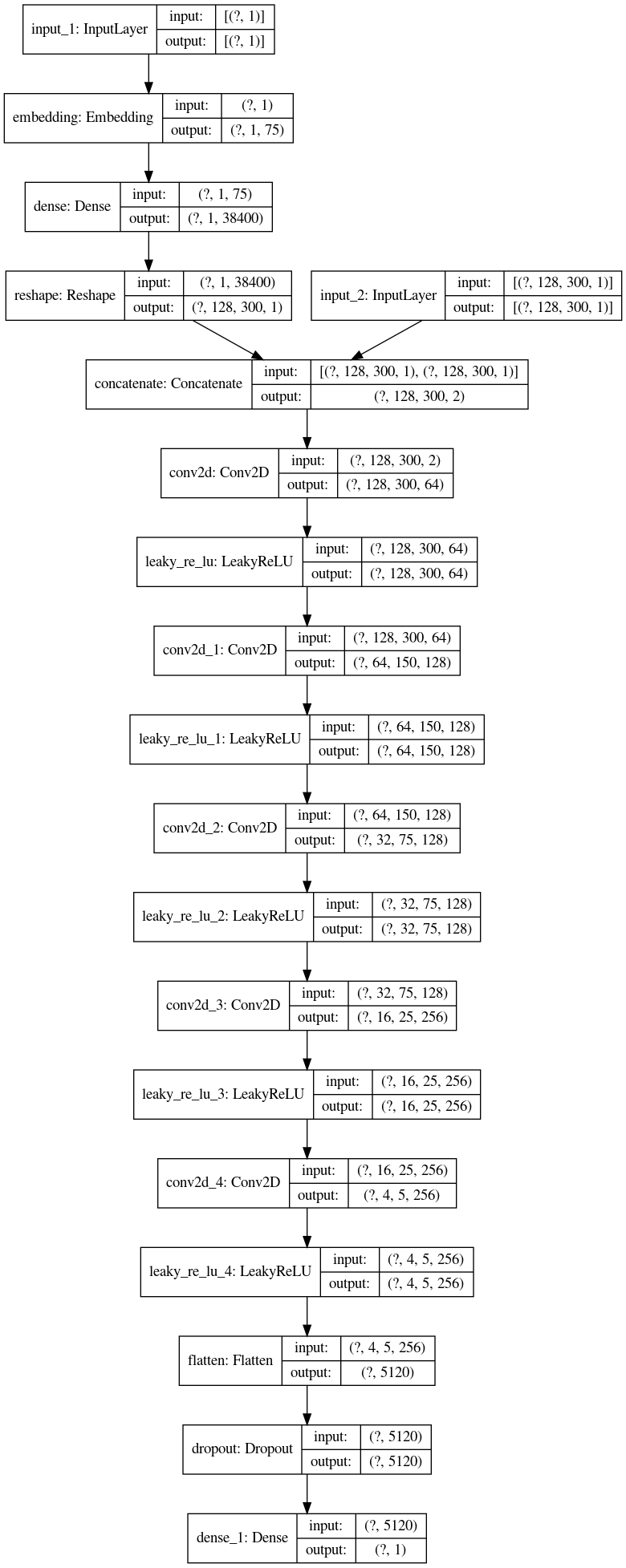


Figure 5.15 –*Architecture of the discriminator asymmetrical to the generator*

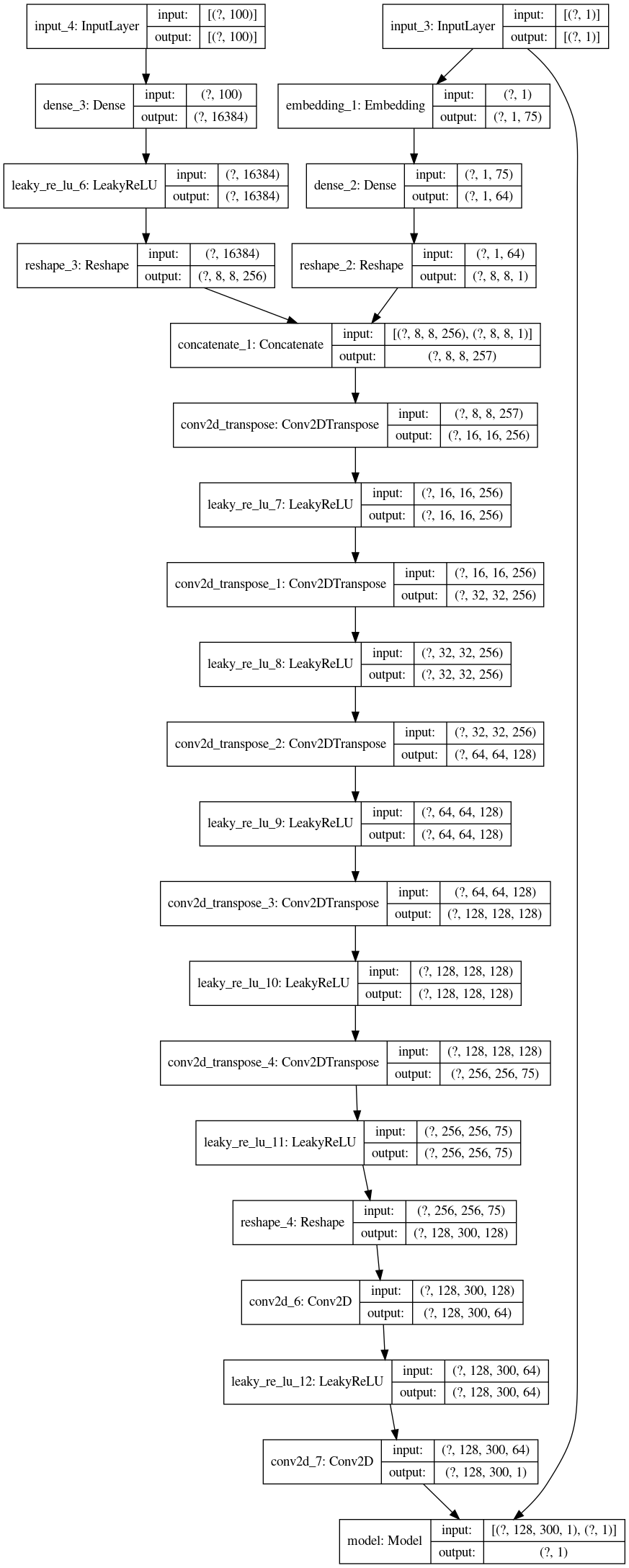


Figure 5.16 –*Architecture of the used GAN*

# Modifications to the training algorithm

The algorithm shown in Figure 4.3 represents a foundation that can be upgraded to speed up and improve the learning process, and during the preparation of this paper, the algorithm was iteratively improved.

The first improvement of the algorithm is the use of the so-called soft labels, or labels that have values that ​​are not strictly 0 or 1 for the corresponding classes, but a value in a certain interval around the originals. During the preparation of this paper, it was shown that the use of soft labels in the training of generators and discriminators gives positive results if soft labels are used for the classes "real" and "generated" data, and negative results if soft labels are used in classes "start of movement" and "movement". In addition to soft labels, noise-disrupted labels were also used to increase the robustness and accuracy of the classifier (which in this case is a discriminator). Noise-disrupted labels are labels that have several labels with incorrect values, e.g. a vector that has 100 elements 97 are correct labels (eg. "1") and 3 are incorrect (eg. "0"). In this paper, labels with values ​​within the interval [0,0.2] for the class “generated data” and labels with values ​​within the interval [0.8,1] for the class “real data” were used, as well as the possibility of wrong labels when training discriminators of 2%.

The second improvement relates to the generation of synthetic data, ie, the fact that it is necessary to select the size of the latent space, which is one of the network hyperparameters, and the distribution from which the input vector is sampled. The size of the latent space is chosen empirically, by trial and error. For the used architecture, the best results in terms of training time and generated spectrograms were obtained for a latent space of size 100. After selecting the size of the latent space, it is necessary to choose the mathematical distribution of data from which the input vector will be generated. During network training, the best results were obtained when the normal (Gaussian) distribution was used. The advantage of using a normal distribution over the use of other distributions (eg. uniform) is reflected in the faster training of the network because usable spectrograms are obtained in fewer epochs. In addition to the training speed, it was observed that for different network topologies during the training, the normal distribution gives consistently better results. In cases where the network uses a uniform distribution during training converges to generate spectrograms consisting of a single line (shows only the Doppler frequency), the same network using the normal distribution generates spectrograms that have a richer response (the micro-Doppler effect is present) and takes longer for model to converge (start generating single line for spectrograms).

The prepared dataset does not contain an equal number of samples from both data classes, more precisely, it has about twice as much data for the "movement" class as in the "start of movement" class. By randomly generating labels using the random function when preparing input vectors, the representation of both classes is equal unless certain conditions are set. Thus, the generator model would equally learn to generate spectrograms of the "movement" class and the "start of movement" class, although there is significantly more data of the "movement" class. This approach results in slower network training and poorer appearance of generated spectrograms for the "start of movement" class. When an additional condition is introduced while generating labels for synthetic data in terms of the probability of label appearance (in this case it is 31.3% for a label marked 0 and 68.7% for a label marked 1), the training time is reduced (number of required epochs) and generated the spectrograms for the class "start of movement" are more similar to those in the training set.

As GAN represents two networks merged into one, there is no clear metric based on which the network would be trained until a certain value of that metric is reached. Therefore, GAN is trained for a given number of epochs after which the training is stopped and the generator network is evaluated using a previously trained classifier. During each epoch, the training dataset is divided into smaller sets (mini-batches) after which the network is trained as shown by the algorithm in Figure 4.3 and explained in point 4.1.3. An algorithm from the Figure 4.3 process of dividing data into mini-batches contains off generating random indices of training data and there might be repetition in the training data, while in the modified algorithm that is impossible. This prevents the network from learning the order of the data and generating results based on it, as well as ensures that all data are used in the training process. An additional difference from the algorithm in Figure 4.3 is the existence of a recovery mechanism if the model begins to converge. As mentioned before, GAN is a network consisting of two separate networks and the goal of its training is to gradually improve both networks. If the generator cannot keep track of the progress of the discriminator or vice versa, the network starts to converge and the training should be stopped, the model reversed to the point where the progress of the network was equal and the training is continued. In this paper, this is made possible by monitoring the value of the loss function and the accuracy of the discriminator when classifying data into real and synthetic. If the value of the discriminator loss function is approaching zero (constantly decreasing and has a value less than 0.1) for real and synthetic data, the convergence of one of the networks in the model has occurred and it needs to be reloaded. Another indicator is the accuracy of the discriminator, which should ideally be 50% for real and 50% for synthetic data. If the discriminator manages to classify synthetic and real data with 100% accuracy, it is necessary to reload the model and increase the learning rate of the generator or reduce the learning rate of the discriminator to achieve balance in the network.

# Experimental results

This section presents the experimental results and the ranking process of the trained networks. During the preparation of this paper, the network architecture and the training algorithm were iteratively improved with several architectures giving promising results and those networks are being directly compared in this section.

For evaluation of the networks, three parameters were used. The first parameter was the accuracy of the network during the training, ie. the fact that the discriminator was not able to classify the generated and real data into appropriate classes with 100% certainty. This parameter was met by all observed networks, with the fact that after the 200th epoch, almost all discriminators had an accuracy of 100% during classification. The second parameter represented the visual impression of the generated spectrograms, ie. whether the generated data has visible artifacts or whether the network is overfitted (saturated) and generates the same spectrograms for one class. The third parameter was the accuracy of classifying generated spectrograms into the appropriate class using a pre-trained classifier over actual data. The architecture of the classifier used is shown in Figure 5.17. The classification was performed by generating 1000 synthetic data of both classes using generator models that were recorded every two epochs during the training. The generated data are then classified into one of three possible classes, and the classification results are presented in the form of graphs showing what percentage of the generated data belongs to the corresponding class.

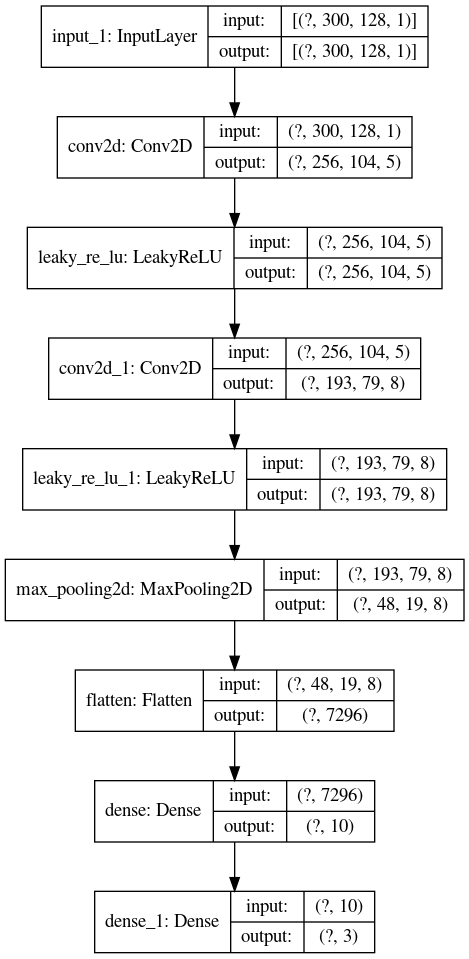


Figure 5.17 – *Architecture of the classifier*

For the network to be taken into consideration as a final network, it was necessary to fulfill the first parameter because the fast convergence to the accuracy of the discriminator of 100% gives the results of classification by the classifier shown in Figure 5.18, where after only six epochs the generator creates completely unusable data. Therefore, five different networks were taken into consideration as plausible final network models and were compared against each other based on the remaining two parameters.

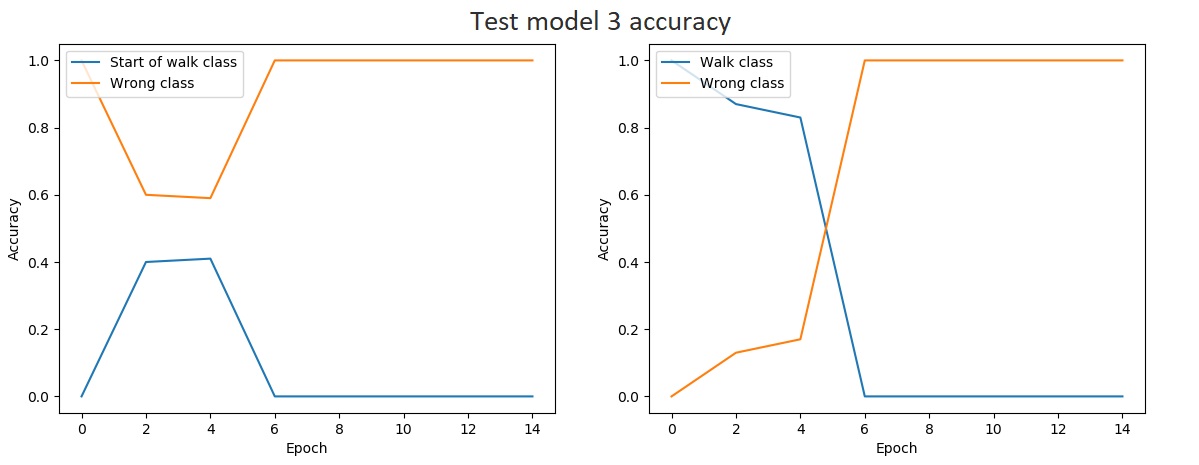


Figure 5.18 – *Classification results of converging network*

From Figure 5.19 we can see that all models have a pronounced oscillation in the accuracy of classification, especially for the class "beginning of movement". This behavior is expected for two reasons, the first reason is that the size of the dataset for class "start of movement" is twice smaller than for the "movement". Another reason is that for the class "start of movement" samples were taken from the beginning and end of the movement, which is why the micro-Doppler response can be found at both ends of the spectrogram, which further confuses the generator.

From Figure 5.19 we can see that there are three models that have been trained for more than 200 epochs, namely models 1,2, and 4. The reason for this was that during the training of these models the discriminator didn't reach an accuracy of 100% during the classification of generated spectrograms after 200 epochs and their training was continued for another 100 and 200 epochs respectively.

Based on the accuracy shown in Figure 5.19, it could be concluded that model 1 generates the best results, however, model 1 converges and generates one spectrogram for a certain class, and based on that spectrogram the classification is 100% correct or 100% wrong. Generated spectrograms by model 1 after 300 epochs of training are shown in Figure 5.20.

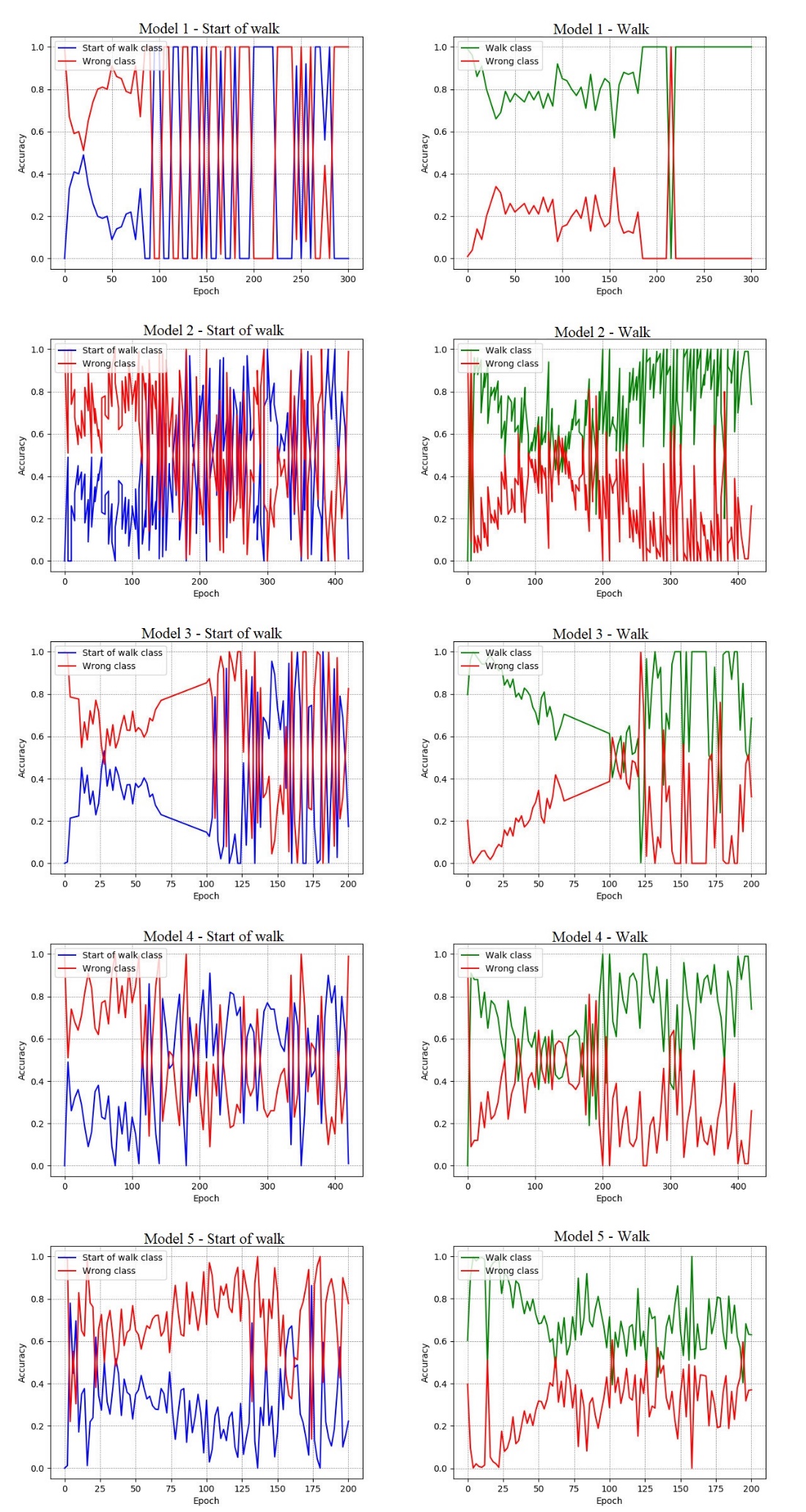


Figure 5.19 – *Comparison of the best network models*

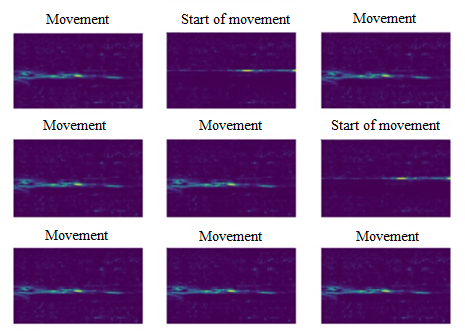
**

Figure 5.20 *– Example of generated sample by converging network*

Model 2 and Model 4 share the same network architecture (the architecture described in this paper) with the difference that during the training of Model 4 the mini-batches were manually divided and it was ensured that during the epoch all data was used exactly once, while in Model 2 the data were randomly divided into mini-batches. After 250 epochs, model 2 begins to converge, but not like Model 1 to generate one spectrogram for one class. In this case, the generator generates about 30 different spectrograms per class. Model 4 did not enter such saturation (which can also be seen from the graphs) and after 420 epochs of training, it always generated different spectrograms. The appearance of the spectrograms generated by Model 4 is shown in Figure 5.21.

The negative impact of the application of soft labels on the classes "start of movement" and "movement" is shown in Figure 5.22. The model that produced these spectrograms has the same architecture as Model 4, which proved to be the best in generating the required spectrograms and the only difference is in the use of soft labels. The spectrograms generated by this model are approximately the same regardless of the class passed to them, ie. spectrograms generated for the "start of movement" class resemble spectrograms generated for the "movement" class.

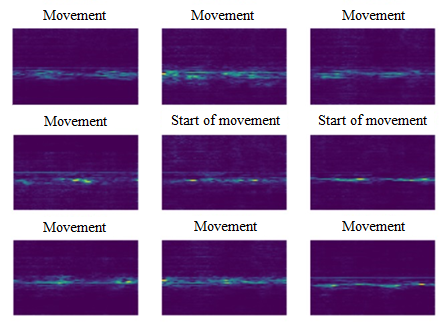


Figure 5.21 – *Example of noncoverging spectrograms*

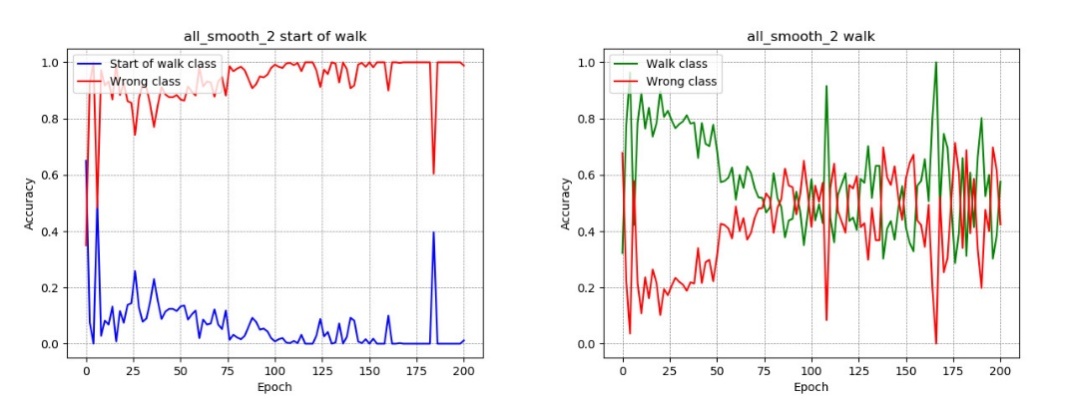


Figure 5.22 – *Accuracy of model that used soft labels for movement classes*

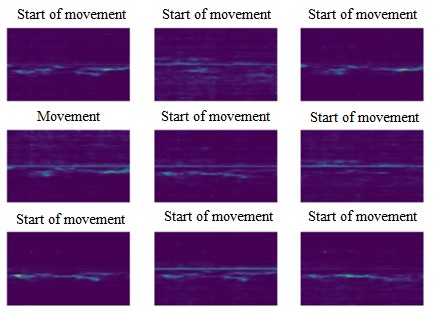


Figure 5.23 – *Spectrograms from model that used soft labels for movement classes*

# Conclusion

This paper aimed to examine the possibility of generating synthetic radar signals using machine learning to train motion classifiers in real situations. This paper represents a combination of domain knowledge of radar systems and signal processing and the knowledge of advanced machine learning concepts. The synthesis of this knowledge is mostly reflected in the quality assessment of one network model, where there is no strict metric that could be used to determine the best model, but it is rather necessary to combine several quality indicators and rank their importance.

This paper consists of a brief introduction to radar systems and the theory behind the operation of radar, how to detect an object, and the speed of a moving object. After that, the two most important types of radar, pulse, and frequency modulated radars are briefly explained.

After the introduction to radar systems, a description of the micro-Doppler effect is given, and the micro-Doppler responses of a rotating object and human gait are shown. In the same chapter applications of using the micro-Doppler effect such as recognizing human activities, recognizing the type of drone, and monitoring life signals are explained. At the very end of the chapter, the difference between the mathematical model of the micro-Doppler response of human gait and the measured response obtained by radar signal analysis was presented.

Then, the focus of the paper shifts to machine learning, more precisely generative adversarial networks - GAN, their architecture, training algorithm and application, and variations to the basic network in the form of a conditional (cGAN) model.

Within this paper, a GAN network for generating synthetic radar signals was realized. As a basis for this network, the architecture of the conditional DCGAN network with small changes in the training algorithm was used. The training data set for this network consists of radar signals recorded as part of the HORIS project for traffic safety and consists of fifteen measurements of a person walking towards the radar from a distance of 5 m, 10 m, and 15 m. The recorded signals had to be pre-processed, ie. an algorithm for detecting the movement within the signal, as well as an algorithm for cutting and normalizing the obtained, had to be implemented. To confirm the performance of the realized network, a previously trained motion classifier trained over real data was used to test whether the network generates adequate signals.

During the work on this paper, several problems were noticed that made it impossible to achieve better results:

1. Inadequate training data set: the biggest problem in the implementation of the network from this paper was an extremely small training data set. As only 15 of the 45 signals were usable due to noise disturbance, the number of different spectrograms generated from these signals is very small. In addition to the small number of training data, a big problem is the unequal representation of the data in the dataset, where there are twice as many of the spectrograms from the class "movement" than spectrograms from the class "start of movement". Changes in the training algorithm have reduced but have not completely eliminated this shortcoming.
2. Limitations introduced using the classifier: as the realization of a network that generates synthetic data is part of a larger project, it was necessary to ensure that the generated data meet certain conditions such as data dimensions. Since the generated data is a spectrogram with a micro-Doppler response, the dimensions of the spectrogram directly determine the resolution in frequency and duration of the signal, ie. the number of points in which the Fourier transform is performed and the number of samples in time. By using 128 points for FFT, a number of details in the frequency domain are lost, further reducing the diversity of training data. In addition to the lower frequency resolution of the spectrogram, by choosing a relatively long-lasting response of 0.5 s in relation to the number of points where fast Fourier transform is performed, disproportionate dimensions of the desired spectrogram are obtained, which makes it difficult to find a suitable network architecture.

In addition to the previously listed problems, the implemented network produces good results when generating data from the class "movement", where for the selected architecture over 80% of the generated data belongs to the given class. In the case of the "start of movement" class, the situation is somewhat worse and about 60% of the generated data belongs to that class. These results show that with the use of a larger dataset, it is possible to implement a GAN network that would generate unique synthetic signals that can be used to train better classifiers.

In addition to collecting a larger training dataset, in order to improve the implementation of this work, an algorithm for automated classification of cut training data (spectrograms) should be implemented (this was done manually in this paper). In addition to the automated classification of training data, it is necessary to improve the classifier by adding more classes such as "end of the movement" and "running" and increase the number of points in which FFT is performed to preserve more information about movement frequencies. By increasing the number of points of the Fourier transform, it is necessary to consider the use of longer samples, proportional to the number of FFT points. With longer samples of the spectrograms, they could contain the entire period of human movement (the assumption is that humans walk slower than 0.5s for a complete cycle of movement). Also, it is necessary to consider the possibility of using Image-to-Image Transition networks and the use of simulated spectrograms generated based on distance information from the radar (as spectrograms shown in Chapter 3).

# Literature

[1] Better protection for pedestrians, 2021 [Online] Available: <https://www.fhr.fraunhofer.de/en/press-media/press-releases/2021/horis-better-protection-for-pedestrians.html> [Accessed: 2- Apr- 2021]

[2] HORIS - High-resolution radar systems for infrastructures, 2021 [Online] Available: [https://www.iis.fraunhofer.de/en/ff/lv/dataanalytics/anwproj/horis---hochaufloesende-radarsysteme-in-der-infrastruktur.html](https://www.fhr.fraunhofer.de/en/press-media/press-releases/2021/horis-better-protection-for-pedestrians.html) [Accessed: 2- Apr- 2021]

[15] J. Brownlee, Generative Adversarial Networks with Python, 2019.

[20] T. Karras, T. Aila, S. Laine и J. Lehtinen, „Progressive Growing of GANs for Improved Quality, Stability, and Variation,“ у *ICLR*, Wien, 2018.

[21] V. Dumoulin и F. Visin, „A guide to convolution arithmetic for deep,“ Université de Montréal; Politecnico di Milano, Montreal; Milano, 2018.