Micro-Doppler RADAR Spectrogram Classification of Targets with Convolutional Neural Networks

Lynn, Nicolas; Yerachmiel, Moshe

Department of Engineering, Tel Aviv University, Israel

October 11, 2020

Abstract: Deep learning-based RADAR classification has become a key technology in defense and surveillance. In this paper, different convolutional neural network architectures are tested for the task of classifying RADAR spectrogram targets as either human or another mammal based on micro-Doppler signatures, a task made difficult by the similar patterns of motion of a human's and mammal's four limbs. The efficacy of a dilated convolutional layer and a Fourier layer is tested in the context of spectrogram classification. A generative adversarial network is also proposed for generating fake RADAR spectrograms in order to correct class imbalances. Finally, an alternative to rudimentary oversampling of spectrogram images is proposed in an additional effort address class imbalances. Using the proposed MNNet, an objective classification ROC-AUC of 0.902 is obtained. Implications of this work extend to the defense and security industries that depend on passive, computational monitoring of remote regions using ground surveillance RADAR systems. Data used to train the models was provided by the MAFAT Challenge and was sponsored by the Directorate of Defense Research and Development under the Israeli Ministry of Defense.

Key words: Doppler, micro-Doppler, machine learning, convolutional neural network, surveillance, RADAR, ground surveillance monitoring, GAN, image generation.

1 Introduction

Object classification has important applications in numerous fields, among which is Doppler RADAR sensing. RADAR is a technology that enables object detection and tracking, and is therefore used extensively in civilian and military applications. With ground surveillance RADAR remote areas can be monitored without direct visual observation. RADAR sensors are an attractive choice due to their ability to collect effective data regardless of ambient visibility. Generally, information collected with RADAR is much less redundant than its image or video surveillance counterparts in the sense that the return provides a higher density of actionable insight on the targets while avoiding unimportant visual characteristics [1].

Doppler RADAR technology works by emitting radio-waves at specific frequencies in a targeted direction and receiving the echo of those signals as they bounce off of objects. Doppler RADAR obtains velocity information on a target by analyzing the way that a target's motion alters the returned radio-waves' frequency. The Doppler Shift is the frequency difference between the observed and emitted signal. From this comparison the radial components of a target's motion within its field of view (FOV) can be obtained with high accuracy. The returned signal is captured as In-Phase/Quadrature (I/Q) complex representation, which contains amplitude and phase data. Because this raw signal contains most important information in its frequencies, a more descriptive spectral representation must be obtained. A spectrogram is a heatmap or image representation of the power spectral density (PSD) of a signal at consecutive time points concatenated together. Such an image representation is what can be leveraged by deep learning.

Figure 1: Ground Doppler RADAR surveillance system. *Source: M-Landarch Company Limited*

There have been considerable machine learning (ML) efforts made with respect to RADAR classification tasks[2] [1]. Traditional ML efforts to classify actions and targets from RADAR returns rely on the development of handcrafted features, usually based on signal processing and statistics. While

these methods achieve good performance, they are inherently limited by the high level of expertise needed in the field of RADAR signals in order to develop usable features. Additionally, these methods are not sufficiently robust to different sensor specifications, geographies, and signal-to-noise ratios (SNRs).

Recent advancements have made deep neural networks an attractive alternative. Because domain expertise is not as important a prerequisite for developing image classification neural networks, models can be built with much more flexibility and scalability[1]. With convolutional neural networks (CNN), explicit feature development is avoided all-together and performance is improved significantly; these are two fundamental reasons as to why neural networks have become an innovative frontier for RADAR image classification.

While classifying target actions based on their spectral return is well established, the differentiation between specific objects, such as an animal from a human, is much more difficult for reasons soon to be discussed. Regardless, the ability to differentiate such targets is important in its own right for both defense and consumer applications.

1.1 Raw RADAR Return

While detailed knowledge on RADAR technology is less essential for the classification network developer, there are some basic principles that are important to keep in mind. The raw return a Doppler RADAR provides is a signal in I/Q representation as data-time signal samples. Each returned sample is made up of the reception time of the return signal divided into a specified number of bins (in this data 128 bins are used). This is also referred to as fast-time. If the log of the absolute value of the Fast Fourier Transform of the fast-time return is calculated, the PSD of the signal is obtained. By adjoining the PSDs of several time steps together, the change in power of each frequency bin can be observed over time in the form of a heatmap or spectrogram. This image representation of frequency characteristics is what is used as input data throughout this work. Samples of a spectrogram can be seen in 9 and 12.

1.2 Micro-Doppler Spectrogram

RADAR developments in the last twenty years include the ability to track subjects of interest (as is widely used in flight control settings), action classification of subjects (whether a human is running or sitting), and target classification of subjects (whether the RADAR is observing a private jet or a jet fighter). The two latter tasks leverage information from what is known as micro-Doppler signatures. While the most prominent information extracted from RADAR observation is the general movement of a large body – such as a human torso or the fuselage of a plane – there are less prominent items that are attached to the object of interest and that may be moving independently. More precisely, micro-Doppler refers to the movement *within* a target rather than the general motion of the target as a whole. For example, the movement of a car is quite different than the movement of its wheels.

A more relevant scenario is a torso of a human which moves monotonously in one direction while the same subject's appendages are swinging in alternating directions. Using this same example, the torso's movement would have the most defined track line on a spectrogram; yet the arms' movement would be present as well, though more subtly. The spectrographic information encoding the movement of appendages or free-moving components is what ultimately lends uniqueness to the spectrogram of different targets. When it comes to similar subjects such as humans and dogs (both with a prominent torso and four appendages), the micro-Doppler signatures becomes even more difficult to differentiate. Capturing these features is not trivial and requires methods capable of extracting subtleties, making CNNs an excellent option.

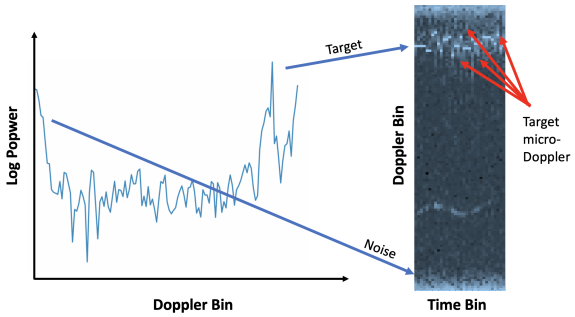

Figure 3: Signal To Spectrogram

1.3 Convolutional Neural Network

Convolutional neural networks are computational architectures consisting of several layers of filters and non-linear functions. A classification CNN is applied to an image and its filters are used to convolve the input into feature maps. These filters are tensors of parameters or weights that are learned and adjusted by the network as it is trained on a dataset; after the forward pass where an image is fed through the network, a loss can be calculated and back-propagated to each weight of each filter in the network, adjusting those values appropriately. Ideally, the feature maps resulting from the convolutions contain information that can then be used by the progression of the network to make a proper prediction. Non-linear activations typically follow convolutional steps in order to break the network's linearity. Such activation nonlinearities include ReLU and hyperbolic tangent functions. Different CNN architectures include variations in filter sizes and properties, network depths, convolutional layer types, activation functions, the employment of skip-connections, and parameter initialization methods. In the end, the outputs of the final convolution in a classification network are compressed into a one-dimensional classification vector.

1.4 EfficientNet

EfficientNet [3] is one variation of a CNN which was published in 2019 along with seven models, all of which were pretrained on ImageNet, named B0-B7. The paper re-examines the scaling of convolutional neural networks and defines a compound scaling method which ties the networks depth, width and resolution in a closed formula.

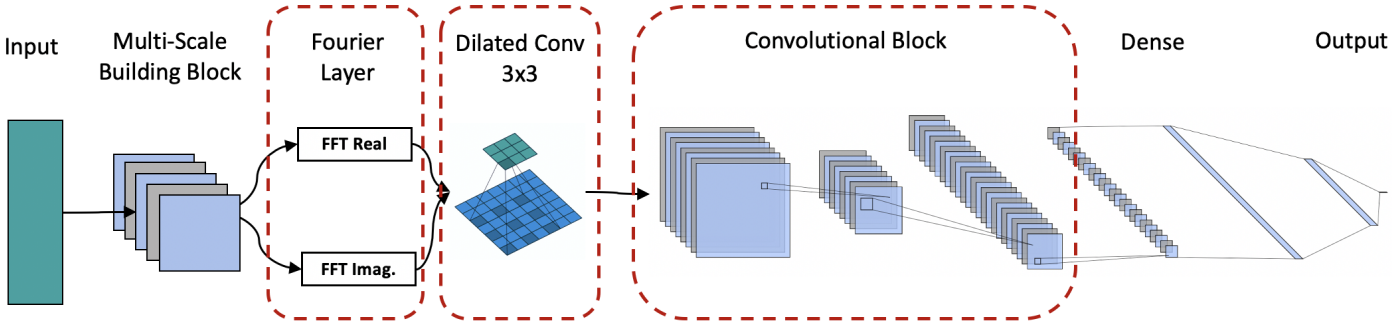

Figure 2: End-to-end F-ConvNet architecture as proposed by [1] with the layers of interest outlined.

$$
\begin{aligned}
depth : d &= \alpha^{\phi} \\
width : w &= \beta^{\phi} \\
resolution : r &= \gamma^{\phi} \\
\alpha \cdot \beta^2 \cdot \gamma^2 &\approx 2 \\
\alpha \geq 1, \beta \geq 1, \gamma &\geq 1
\end{aligned}
\tag{1}
$$

The methods applied found the most appropriate parameter ϕ that satisfies a certain memory and computation target while maximizing the accuracy on the data and scaling up to form the seven variants. This method is considered the state of the art in the ImageNet classification challenge.

2 Related Works

2.1 F-ConvNet

As previously mentioned in Section 1, there has been considerable investigative effort made in the development of action classification networks for RADAR systems based on micro-Doppler features[2][1]. One significant inspiration in the development of this work is the system proposed by Ye and Chen: F-ConvNet[1]. The proposed fourier layer is a unique convolutional layer capable of replacing the hard-coded function of a Short Time Fourier Transform (STFT) with learnable parameters. Two convolutional branches are initialized – using a customized algorithm described in Section 4.2.2 – that convolve raw input time signals into an imitation of respective imaginary and real components associated with the frequency representation of the RADAR return. By making the parameters to the Fourier convolutional layer learnable, they are able to transform the raw RADAR signal into a domain optimized for the classification task. Their proposed network has a reported accuracy of 98.5% when classifying raw RADAR returns into one of seven different activities using an unspecified RADAR dataset. However, F-ConvNet is not intended to work with spectrogram inputs and therefore is used in this paper strictly as a baseline. The proposed Fourier layer is extended upon in our work not as a direct physical translation of its original use, but as an attempt to extend the time-frequency axis of the spectrograms. Figure 2 show F-ConvNet as originally proposed.

2.2 Dilated Convolutional Networks

The efficacy of employing dilated convolutional layers is highlighted in several works[1][2]. Due to their ability to increase the receptive field of feature maps, it is suggested that dilated convolutional layers are effective in micro-Doppler RADAR spectrogram classification.

3 Data

The data used in this work is provided by the Israeli DoD MAFAT team as a challenge and consists of four sets: a training set (1510 tracks), containing records of both human and animal Doppler returns obtained from standard geographies; an experimental set (2718 tracks) containing data of human and animal Doppler returns obtained in experimental geographies; a background auxiliary set (2837 track) containing only background/empty RADAR returns; and a synthetically-generated low SNR set (3249 tracks) which consists of tracks from the first two datasets after having been put through a transformation to add noise (the statistics of this transformation are unspecified). Due to time and computational constraints the background dataset is not used, making the aggregate dataset binary in label classification. Unique tracks were further divided into 32 pulse transmission time points (also known as slow-time segments) by the challenge organizers. The elemental unit of data in the challenge is thus a 32-time-segment-wide patch of I/Q signal reception time data. The aggregate dataset includes 10,314 tracks (7065 unique) and 137,738 segments. It is important to note that there are few real-world RADAR datasets available. This dataset is unique in the fact that it includes RADAR returns taken from active ground surveillance RADAR devices, i.e. uncontrolled environments.

The data is provided in raw complex form along with the functionalities to perform rolling Hann window smoothing, a one axis DFT, and PSD calculation.

Data sources include segments from 13 different sensors located at 7 different geographic locations. Each sensor is unique to a geographic location, though one location can contain multiple sensors. The sensor parameters and locations are not provided due to the sensitive nature of defensive radar positioning. Figures 4, 5 show the distribution of geographic locations and sensors in the datasets. In-depth data analysis revealed no significant co-variate shift between the different

attributes. Hence they were not accounted for in the preprocessing pipeline.

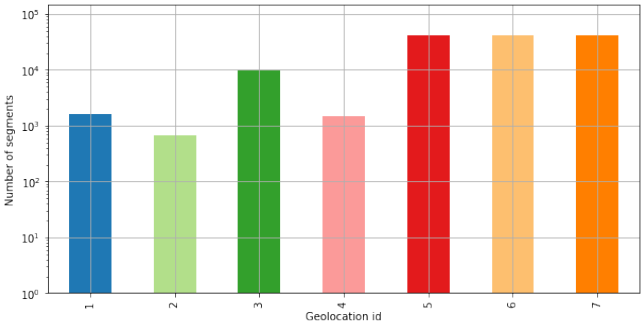

Figure 4: Distribution of segments as a function of geolocation ID. Note the log scale of the vertical axis.

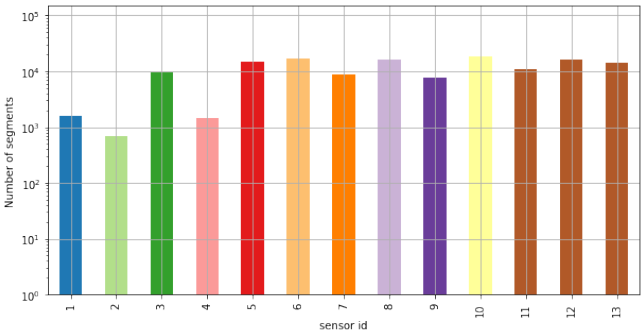

Figure 5: Distribution of segments as a function of sensor ID. Note the log scale of the vertical axis.

Each track is attributed a signal-to-noise ratio (SNR) label: *low_SNR* or *high_SNR*. There is an additional *synthetic_SNR* label reserved for the data in the synthetic set, though those can be regarded as low SNR samples. Figure 6 shows the distribution of SNR presence in the datasets. Efforts to overcome the low SNR data via generative and AE models are all detrimental to the results obtained and hence are not mentioned in this article.

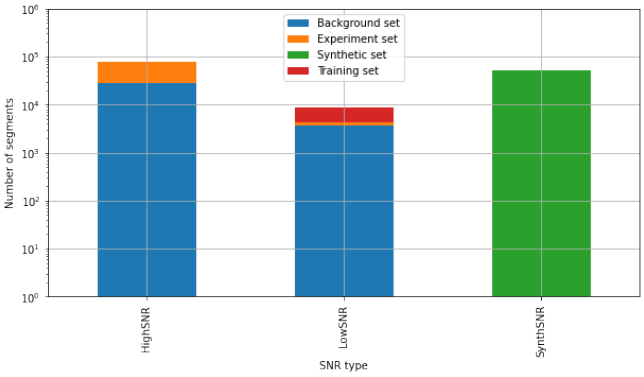

Figure 6: Distribution of segments as a function of SNR type. Note the log scale of the vertical axis.

Upon examining the target distribution across all datasets, an imbalance is highly noticeable in which class *human* outnumbers class *animal* ten-fold (99199 human, 7411 animal, 31128 background). However, there are overlapping segments in the data originating from the fact that the synthetic dataset contains tracks from both the training and the experiment dataset but with added noise. Even after accounting for this duplication and examining the source distribution, it is apparent that the unique segments representing the class *human* are ten times more abundant than class *animal*, and that most of the human segments originated in experimental geographies.

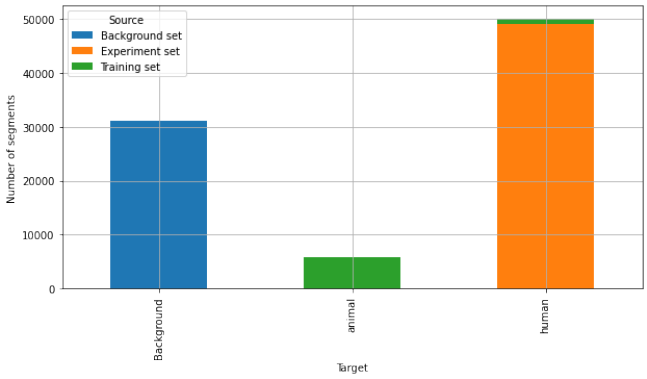

Figure 7: Targets label frequencies in the default segment division.

These findings (i.e. the imbalance in SNR types and in target distributions) inspire the creation of a specialized pipeline described in Section 4.1 in which the segments are stitched back into tracks and divided accordingly in order to create a more balanced dataset.

4 Methods

The ultimate goal of this work – beyond obtaining optimal classification efficacy – is to test the effect of classifying spectrogram targets with varying CNN architectures and with context-specific convolutional layers. It is hypothesized that both the Fourier layer and the dilated convolutional layer improve the classification efficacy of CNNs. Additionally, this paper aims to ascertain an optimal network – MNNet – for RADAR spectrogram target classification. Furthermore, this paper tests two methods that solve the class imbalance in the dataset. Finally, this experiment aims to offer insight into the ability to generalize between samples of low SNR and high SNR.

All code is implemented using *Python 3.7*. The deep learning framework used is *Pytorch*. This investigation is hosted and executed using Google-Colab Jupyter framework. For reproducibility, the random seeds are fixed and constant throughout the investigation.

4.1 Data Preprocessing

It has been shown that deep learning RADAR classification tasks have improved results when working directly with spectrogram images [4][5]. Therefore, the I/Q raw matrices provided in the datasets are transformed using Hann windowing, a Fast Fourier Transform (FFT), and normalization. The resulting spectrogram images are of dimensions 126 x 32, or 126 frequency bins along the Y axis at 32 different time steps

along the X axis. The cell values of this array represent the log power.

Next, there is an immediate need to balance the data in order to prevent the model from minimizing loss by developing a prediction bias towards the more frequent label, as discussed in Section 3. Each datapoint includes a track ID and segment ID. In order to improve the ratio between labels, a rudimentary over-sampling approach of animal-labelled datapoints or under-sampling approach of human-labelled datapoints can be used. However, over-sampling can lead to over-fitting of the repeated images and under-sampling wastes data. Instead, a new approach is developed where each datapoint with corresponding track IDs are compiled and reconstructed into continuous tracks based on consecutive segment IDs. Once all the tracks are reconstructed, they are split back into images with a width of 32 slow-time points using a sliding window. The sliding window's stride is calculated based on the ratio of label frequencies. This methodology allows for some uniqueness in the extended data as there are no two identical segments, though there are overlaps.

Additionally, considerations have to be taken to avoid overlaps between datapoints in the training and validation partitions of training. If left uncorrected, the training set would be able to peek into the validation set. Therefore, the unextended training data is first split into folds using a stratified k-folds algorithm described in Section 4.1. Within each fold, the validation indices are removed from a temporary copy of the full dataset. With the validation partition removed, the remaining training points are put through the described data extension pipeline, resulting in an improved ratio between human and animal labels.

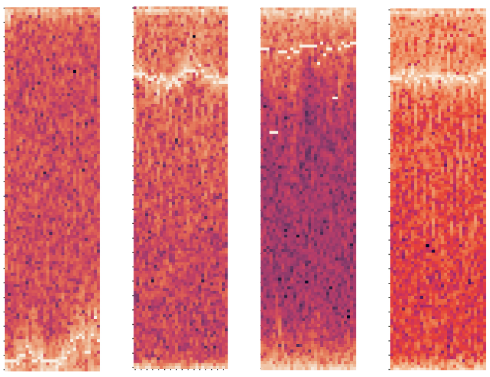

Figure 8: Animal-labelled spectrogram samples with evident velocity tracks.

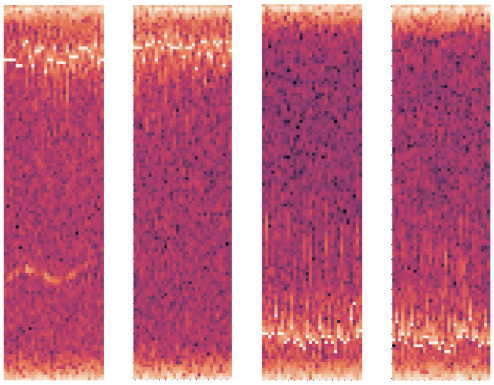

Figure 9: Human-labelled spectrogram samples with evident velocity tracks.

The final obligation for strategizing how data is handled is the need to ensure fraternal twin-type datapoints shared between the primary, synthetic, and experimental training datasets do not end up in both train and validation partitions (similar to the issue of overlap). The synthetic dataset includes many radar datapoints present in the experimental and training sets that are put through transformations to decrease SNR. Allowing for such an overlap would pollute what should be an objective validation loss metric, even if the images are not identical. For investigative purposes, training is only performed on either the low SNR-type or the high SNR-type datasets in order to test the hypothesis of whether or not one is able to generalize better to the other. The efficacy of training on each set becomes a valuable point of interest in itself since there is not an intuitive answer as to whether training on low SNR-type data will generalize better to high SNR-type data, or visa versa.

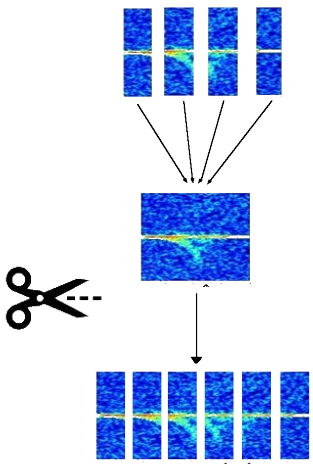

Figure 10: Re-sampling method proposed for improved network training without oversampling.

Using a stratified K-Folds method, the data is then divided into a number folds, a hyper-parameter determined during the optimization process. For each fold a model is trained for a specified number of epochs and saved to disk. This process

creates an ensemble of such models. During inference, predictions for the test data are produced across the model ensemble and a median operation is applied to produce a single probability per test point. The median function is chosen due to its inherent robustness to outlier values.

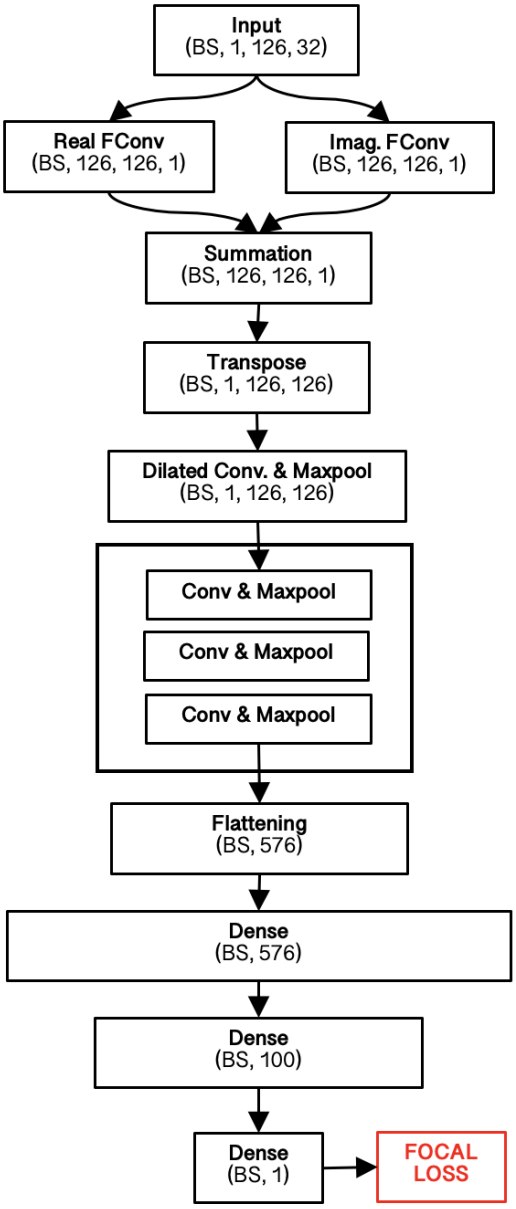

Figure 11: MNNet experimental architecture with a designated location for a Fourier layer, a dilated convolutional layer, and a convolutional block.

4.2 Proposed MNNet Architecture

MNNet is an experimental network with components that are iteratively inserted and removed. In the testing of MNNet's various architectures and convolutional layers, there are some general specifications that guide development. First, the input to all models is a 126x32 spectrogram image. The output is to be a single probability value between zero and one. Beyond this, various elements may be tested. As can be seen in Figure 11, a Fourier layer will be incorporated at the head of the network, potentially followed by a dilated convolutional layer. A convolutional block then empties into a dense network. These experimental components are inserted, excised, and altered in the different variants to be tested.

4.2.1 The Fourier Layer

The Fourier layer is proposed by [1] as a learnable alternative to STFT and is introduced in Section 2.1. However, it is intended to work on raw time series data. STFT works similarly to FFT with the distinction that STFT calculates the Fourier transform for intervals along the original signal in an effort to find the changes in frequency along different windows of the signal. STFT representations are computed separately from the network, without learnable parameters which makes the algorithm sub-optimal for deep-learning contexts which is rooted in the ability to adjust parameters. Rule-based representation methods are less effective in deep learning tasks when juxtaposed with learning-based transformations[1]. Because STFT and FFT can be looked at as a convolution in itself, it may be of value if the parameters of the kernels of these methods could be adjusted and optimized for a classification task, as is described in [1].

MNNet intends to take the functionality of the Fourier layer and apply it to a less physics-grounded intention of convolving the magnitudes of each frequency bin across all time segments into new feature maps. One way of looking at this implementation is as an attempt to understand the change in frequency characteristics of the RADAR return across time. With this the network may be able to extrapolate additional micro-Doppler features that can be used to understand the target.

The Fourier layer is comprised of two separate convolutional kernels of size 126x1x1x32. This layer accepts images with one channel and returns 126 features maps. The convolution kernel is as wide as the input image (i.e. 32 time points) and hence the dimension of the image is transformed from 1x126x32 into a feature map of 126x126x1. Because this layer has two convolutions, the two resulting feature maps are summed together into a new 126x126x1 feature map. The second and fourth dimensions of this output can then be transposed in order to treat the number of channels as the new image width. The final output of the Fourier layer is a 1x126x126 image.

The kernel weights are initialized according to the algorithm developed by [1], using sinusoidal coefficients. more details regarding the algorithm can be found in Section 4.2.2.

4.2.2 Fourier Kernel Initialization

A proper initialization method is described by [1]. Reportedly, this initialization method increases convergence speed. Instead of standard uniform distribution initialization of the convolutional weights (the default in *PyTorch*), the real convolution kernel is initialized using cosine coefficients while the imaginary convolution kernel is initialized using negative sine coefficients. For the full algorithm, please refer to the original work.

4.2.3 The Dilated Convolution Layer

Dilated convolutions have been suggested to improve RADAR classification accuracy [2][1]. Dilated convolutional layers increase the receptive field of succeeding layers. In MNNet, a single dilated convolution with a kernel size of three and a dilation of two is placed experimentally after the Fourier convolutional layer. This dilated convolution is followed by a batch normalization and Leaky ReLU activation. A padding of two and a stride of one is employed. Therefore, the output should be of identical size to the input (in this case 126x32). This layer receives a single channel and returns four feature maps. The performance of MNNet with and without a dilated convolution is explored.

4.2.4 Standard Convolutional Networks

The selection of the convolutional block is based on the experimental results of four different vanilla CNNs; a 3-layer CNN (CNN-3), a 5-layer CNN (CNN-5), an 8-layer CNN (CNN-8), and a 10-layer CNN (CNN-10). Each convolutional block tested has a varying number of layers, all with 3x3 kernels. Each convolution is followed by a batch normalization, which help neural network stability and convergence. After batch normalization, a Leaky ReLU activation is applied as a non-linear function. Then, maximum pooling with a kernel size of 2 is employed to downsample the feature maps after most convolutional steps. All four variants are tested in isolation and the best-performing block is incorporated into MNNet. A dense network consisting of two fully connected layers feeds into a final single-node layer. The first layer typically contains 512 nodes. The second layer typically contains 100 nodes.

4.3 Optimization and evaluation

Certain hyper-parameters that are to be optimized include batch size and loss function. During preliminary iterations, batch-size and fold counts were tested. Throughout the work, a batch-size of 32 and a fold count of 5 are used. A learning rate of 10^{-3} and a weight decay of $5{\cdot}10^{-4}$ are used. All models are trained for 15 epochs, which is more than enough to allow for convergence of all models during preliminary iterations. A learning rate (LR) scheduler is implemented to decrease LR on loss plateaus with a patience of two and a multiplicative factor of 0.1. An AdaM optimizer is used as well. Finally, two criterion functions are employed: Focal Loss and Binary Cross Entropy.

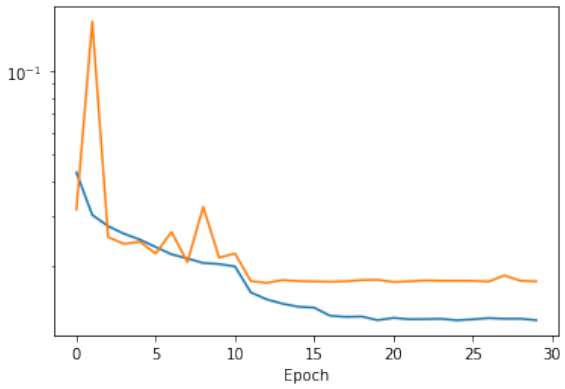

Figure 12: Loss convergence through training 30 training epochs in preliminary classification trials.

The primary loss function used is Focal Loss [6]. Focal Loss is an extension of cross-entropy loss that amplifies the penalty for hard examples and attenuates it for easy examples.

$$p_t = \begin{cases} \text{p} & \text{y=1} \\ \text{1-p} & \text{otherwise.} \end{cases} \tag{2}$$

$$FL(p_t) = -\alpha(1-p_t)^{\gamma} \cdot log(p_t)$$

With $\alpha and \gamma being 1 and 2, respectively.$

As mentioned in Section 4.1, the data is split into folds and for each fold a model is trained for a constant number of epochs (15), creating an ensemble of models. For the purpose of evaluating the performance of the ensemble, an out-of-fold approach is selected. For each model of the ensemble, the validation predictions are stored. Upon completion of the training loop, two evaluation measures are generated: the residuals of the prediction probability versus the label and the effective validation receiver operating characteristic (ROC) curve along with its area under the curve (AUC). The hyper-parameters of the training procedure are optimized using these measures.

At this point, the possibility of an imbalance in statistical properties between the entirety of the labelled training data and the unlabelled test set should be noted. This conclusion is reached due to the significant difference in performance of any given model on the test set when compared to the training and validation set. For example, it is recurrently observed that a trained model would achieve a validation AUC > 0.99 but achieve < 0.75 on the test set. This warrants the need to use a secondary optimization metric: the public test-set ROC AUC.

4.4 New sample generation

The exploratory data analysis phase reveals that the distribution of labels across the training data is imbalanced, as discussed in Section 3. For the purpose of overcoming this imbalance and making the training process more robust to the two classes present, a generative approach is selected.

To this end, a generative adversarial network [7] (GAN) is employed. Due to the odd shape and highly domain-specific modality of the input data, no pre-existing solution was found to be useful. Hence, an *ad hoc* solution is constructed. The architecture consists of an input noise vector of size 182, a

generator with two linear modules (activation - ReLU with batch normalization), four transposed convolution modules (with same activation and normalization), and a discriminator with two convolution modules (activation - LeakyReLU) and one linear module (same activation).

Optimization continues for 1000 epochs with AdaM optimizer (beta1, beta2 = 0.5,0.999 respectively), a learning rate of 10^{-4}, and batch size of 128. Throughout training batches of samples are stored in a visual format and evaluated by a human observer for sample quality and potential mode-collapse. Once an adequate quantity of generated spectrogram samples are developed, their efficacy in improving classification is tested by loading variable numbers of samples into the training set.

5 Experiments

5.1 Efficacy of CNN Blocks

In this trial, vanilla CNNs are trained using the proposed data re-sampling method and with the default Focal Loss criterion. The results of training the four vanilla CNN blocks are summarized in Table 1. As can be seen, CNN-8 performs optimally compared to shallower and deeper networks. It can be speculated that eight layers allows the network the right amount of depth to capture the dimensionality. This trial was conducted early in the investigation so that CNN-8 could be incorporated into MNNet as the permanent CNN block selection.

Model	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Ens.
CNN-10	0.849	0.879	0.885	0.886	0.876	0.881
CNN-8	0.877	0.885	0.899	0.892	0.897	0.900
CNN-5	0.868	0.861	0.884	0.879	0.888	0.879
CNN-3	0.843	0.834	0.839	0.862	0.872	0.856

Table 1: Vanilla CNN area under the ROC curve. Ens in the performance of the ensemble.

5.2 Efficacy of Sophisticated Classification Networks

Three additional models are tested against each other using the proposed data resampling method and Focal Loss. The first is EfficientNet variant b0 (see section 1.4). EfficientNet-b0 is instantiated with pretrained weights which were trained on ImageNet. Though a global average pooling operation makes the network invariant to input size, it is elected to use the pretrained input size of 224x224 by padding the input segments with zeros. The second network tested is the proposed F-ConvNet (see section 2.1), adjusted to accept 126x32 spectrograms and to output one single classification rather than 7. Lastly, an MNNet variant with a Fourier layer preceding CNN-8 (inserted as the convolutional block) and without a dilated convolutional layer is tested.

The results are presented in Table 2. Interestingly, EfficientNet obtained the best result out of the three models presented, with MNNet trailing by 0.036. F-ConvNet performed the worst indicating that it cannot translate well from raw RADAR returns to spectrograms. While EfficientNet did perform best, it took significantly longer to train than MNNet.

Comparing results for CNN-8 MNNet variants with and without the Fourier layer (Table 1 and Table 2), it can be concluded that the Fourier layer has detrimental results when applied to spectrograms, decreasing AUC ROC from 0.900 to 0.852, a drop in 0.048 points.

Model	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Ens.
EffNetb0	0.817	0.898	0.896	0.867	0.881	0.888
FConvNet	0.790	0.807	0.826	0.826	0.834	0.830
MNNet	0.814	0.840	0.856	0.874	0.857	0.852

Table 2: Area under the ROC curve for different model types. Ens in the performance of the ensemble.

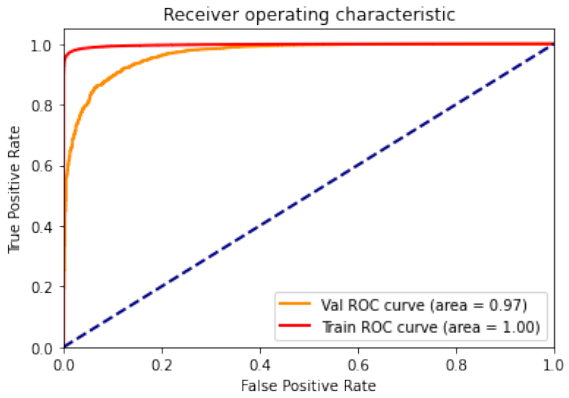

Figure 13: MNNet variant ROC AUC on training and validation data.

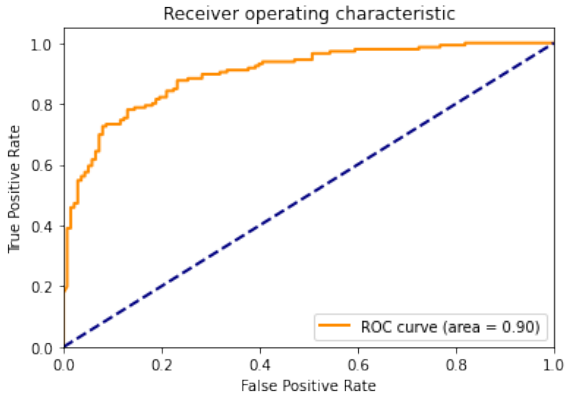

Figure 14: ROC AUC of leading CNN-8 test results.

5.3 Effects of Imbalance Corrections

Next, data preprocessing step are tested. Here the track stitching and re-sampling of segments via a sliding window approach against simple under-sampling (i.e. taking a smaller number of human segments from the dataset) and against simple oversampling (i.e. taking the animal segments multiple times is tested). Table 3 shows the area under the ROC curve for the different approaches.

Table 3 shows that the re-sampling method had a positive effect on the performance of all models tested. Using

the re-sampling method rather than the oversampling method improved results by up to 0.116 in the case of EfficientNet. Using re-sampling is still significant when compared to undersampling, though the results are not as stark. It is interesting to note that excluding large amounts of data had a relatively low negative impact on performance. The target distributions in the training and validation partitions across each data balancing process can be found in Tables 4 and 5.

Model	Oversampling	Resampling	Undersampling
EffNetb0	0.765	0.881	0.879
FConvNet	0.778	0.830	0.806
MNNet	0.818	0.852	0.827

Table 3: Area under the ROC curve for different data balancing strategies.

Label	Oversampling	Resampling	Undersampling
Train Animal	46,050	26,052	4,605
Train Human Label	49,454	49,454	5,605
Val Animal	1,151	1,151	1,151
Val Human Label	9,863	9,863	9,863

Table 4: Number of datapoints by label and partition based on the data balancing method selected with low SNR training.

Label	Oversampling	Resampling	Undersampling
Train Animal	46,050	26,052	4,605
Train Human Label	39,454	49,454	5,605
Val Animal	1,152	1,152	1,152
Val Human Label	9,994	9,994	9,994

Table 5: Number of datapoints by label and partition based on the data balancing method selected with low SNR training.

5.4 Effects of Training Data SNR Type

Next, the effects of training on low SNR-type datsets and high SNR-type datasets are compared. For reasons delineated in Section 3, training is limited to either low SNR-type or high SNR-type data, but not both.

Table 6 shows the dramatic improvement that training on low SNR-type data has over training on high SNR-type data. It can be concluded that training on low SNR-type RADAR spectrograms allows for better generalization to high SNR-type RADAR spectrograms. The inverse cannot be claimed. Intuitively, this makes some sense. If a CNN is able to extract distinct spectrographic features in the presence of noise, it stands that such a model would perform as well or even better when applied to high SNR-type data which would have more pronounced features.

Model	High SNR	Low SNR
EffNetb0	0.765	0.888
FConvNet	0.632	0.830
MNNet	0.697	0.852

Table 6: Area under the ROC curve for different SNR types.

5.5 Selection of Criterion

Next, the hypothesis that using Focal Loss improves the performance on difficult examples is tested on spectrogram images. According to prior knowledge, the natural competitor to test Focal Loss against is Binary Cross Entropy loss. The use of a dilated convolutional layer in the network before the convolutional block is also compared (see section 2.2).

Table 7 shows results are superior when using Focal Loss to train MNNet, supporting the original hypothesis. Training MNNet in this iteration excluded the Fourier layer which is show to have detrimental effects on performance in Section 5.2. Generally, it can be seen from the results that Focal Loss is a much more effective criterion for training RADAR spectrograms when compared to BCE. Interestingly, the results are not as uncontested in regards to the inclusion of a dilated convolutional layer before the convolutional block in MNNet. While including the dilated layer shows a slight improvement in performance when trained with Focal Loss, there is a slight decrease in performance when trained with BCE.

Model	Loss	Dil. Conv.	AUC
CNN-8	BCE	Yes	0.883
CNN-8	BCE	No	0.899
CNN-8	Focal Loss	Yes	0.902
CNN-8	Focal Loss	No	0.900

Table 7: Area under the ROC curve for different loss function and the inclusion of dilated convolution.

5.6 Effects of Generated Datapoints

Lastly, the effects of including new generated animal spectrograms to augment the lacking data is displayed. Table 8 shows the effects of adding differing amounts of generated datapoints to the training process.

Model	Fake datapoints	AUC
CNN-8	0	0.900
CNN-8	500	0.882
CNN-8	1000	0.881
CNN-8	3000	0.880

Table 8: Area under the ROC curve for different numbers of generated datapoints included in the training set.

As can be seen in Table 8, the addition of generated datapoints to the best performing network is detrimental to results. The inverse relationship between the number of fake data points included and model performance indicates that further development is needed to generate highly detailed spectrogram images that can improve the training of MNNet. One potential reason the fake data does not improve results could be due to the great accuracy of network, learning to capture the statistical properties of the input data to greater detail and therefore not recognizing the generated data as a valid class.

Samples of the generated data can be seen in Figure 15. While the general motifs of the spectrograms resemble those of real spectrograms in Figures 12 and 9, there is some missing track intensity.

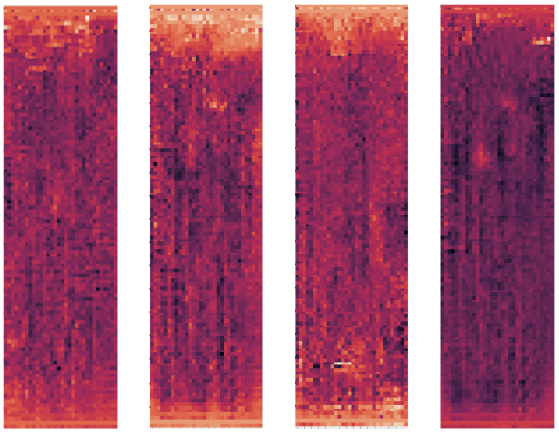

Figure 15: GAN Generated Animal-Label Spectrograms

6 Conclusions

In this paper, the classification efficacy of MNNet variant models on RADAR spectrogram images are tested and examined. The optimal results are achieved with a dilate convolutional layer and an eight-layer CNN. This may be due to an optimal number of parameters fitting well to the data complexity and dimensionallity. A dilated convolution improves RADAR spectrogram classification in some settings. This is likely due to the fact that dilated convolutions increase the receptive field of downstream filters, a property that may be beneficial when analyzing micro-Doppler features. However, these results are inconclusive as the improvement obtained is insignificant and since classification using the dilated layer deteriorated performance in other settings.

Contrary to the hypothesis, incorporating a Fourier layer into the MNNet architecture is detrimental to performance. It is interesting to note that EfficientNet, which is optimized on non-spectrogram, real-world images, out-performs solutions designed to work with RADAR data. This is a testament to the EfficientNet approach.

As evident from the experiments, a major improvement to the results comes from the data re-sampling technique which enables a simple yet effective solution to the problem of data imbalance while avoiding rudimentary oversampling.

Including generated samples of class *animal* in the training set is detrimental to results when tested on the best-performing network (MMNet with a dilated convolutional layer and CNN-8). The effect is greater as more samples are added. When fake data is used in the training of other network, a performance improvement of up to 0.01 is observed (though they perform worse than CNN-8 in general).

Finally, using low SNR data over high SNR data when training proves to have better generalization into the test set.

References

[1] Wenbin Ye and Haiquan Chen. “Human Activity Classification Based on Micro-Doppler Signatures by Multiscale and Multitask Fourier Convolutional Neural Network”. In: *IEEE Sensors Journal* 20.10 (2020), pp. 5473–5479.

[2] Tyler S Jordan. “Using convolutional neural networks for human activity classification on micro-Doppler radar spectrograms”. In: *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security, Defense, and Law Enforcement Applications XV*. Vol. 9825. International Society for Optics and Photonics. 2016, p. 982509.

[3] Mingxing Tan and Quoc V Le. “Efficientnet: Rethinking model scaling for convolutional neural networks”. In: *arXiv preprint arXiv:1905.11946* (2019).

[4] Esra Al Hadhrami et al. “Ground moving radar targets classification based on spectrogram images using convolutional neural networks”. In: *2018 19th International Radar Symposium (IRS)*. IEEE. 2018, pp. 1–9.

[5] W Max Lees et al. “Deep learning classification of 3.5-GHz band spectrograms with applications to spectrum sensing”. In: *IEEE transactions on cognitive communications and networking* 5.2 (2019), pp. 224–236.

[6] Tsung-Yi Lin et al. “Focal loss for dense object detection”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[7] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

A Appendix

For reproduceability the code for this project can be found on git. Due to computation constraints, this project was developed on Google Colab, hence parts of the code are in Jupyter notebook format.