Deep Convolutional Autoencoder for Radar-Based Classification of Similar Aided and Unaided Human Activities

Mehmet Saygın Seyfioğlu – University of Economics and Technology, Turkey
Ahmet Murat Özbayoğlu – University of Economics and Technology, Turkey
Sevgi Zubeyde Gürbüz – University of Alabama

Citation of published version:
Seyfioğlu, M., Özbayoğlu, A., Gürbüz, S. (2018): Deep Convolutional Autoencoder for Radar-Based Classification of Similar Aided and Unaided Human Activities. *IEEE Transaction on Aerospace and Electronic System*, 54(4).

DOI: https://doi.org/10.1109/TAES.2018.2799758

Deep Convolutional Autoencoder for Radar-Based Human Activity Recognition

Mehmet Saygın Seyfioğlu, *Student Member, IEEE,* Ahmet Murat Özbayoğlu, *Member, IEEE,* Sevgi Zubeyde Gurbuz, *Senior Member, IEEE*

***Abstract*—Radar-based activity recognition is a problem that has been of great interest due to applications such as border control and security, pedestrian identification for automotive safety, and remote health monitoring. This work seeks to show the efficacy of micro-Doppler analysis to distinguish even those gaits whose micro-Doppler signatures are not visually distinguishable. Moreover, a 3-layer, deep convolutional autoencoder (CAE) is proposed, which utilizes unsupervised pre-training to initialize the weights in the subsequent convolutional layers. This architecture is shown to be more effective than other deep learning architectures, such as convolutional neural networks (CNN) and autoencoders (AE), as well as conventional classifiers employing pre-defined features, such as support vector machines (SVM), random forest (RF) and extreme gradient boosting (Xgboost). Results show the performance of the proposed deep CAE yields a correct classification rate of 94.2% for micro-Doppler signatures of 12 different human activities measured indoors using a 4 GHz continuous wave radar - 17.3% improvement over SVM.**

***Index Terms*—radar, micro-Doppler, gait recognition, deep learning, convolutional autoencoder, neural networks**

I. Introduction

RADAR-based human gait recognition has been of great interest due to its relevancy to problems of border control and security [1] [2], pedestrian recognition for automotive safety [3], and fall detection for assisted living [4]. Recently, however, the potential for micro-Doppler analysis to distinguish highly similar human gait has come to attention due to the increasingly smaller size and lower cost of software-defined and wireless radar platforms that are readily available, which have made possible applications of indoor radar. For years, indoor monitoring has been most often accomplished using video surveillance devices; however, in health applications, where privacy both at home and in hospital or assisted living facilities is of paramount importance, radar offers distinct advantages. Radar has the ability to facilitate gait recognition remotely, and in little or no light, without the potential of exposing the human body. Although initial application of radar to assisted living has focused on fall detection [5] [6] [7], investigation of radar for fall risk assessment and disease monitoring requires the identification of daily indoor activities, which, especially in the case of elderly, often result in highly similar micro-Doppler signatures.

Manuscript received April 26, 2017.

M.S. Seyfioğlu and A.M. Özbayoğlu are with the Department of Electrical and Electronics Engineering, TOBB University of Economics and Technology, Ankara, Turkey (e-mail: msseyfioglu@etu.edu.tr, mozbayoglu@etu.edu.tr).

S.Z. Gurbuz is with the Department of Electrical and Computer Engineering, University of Alabama, Tuscaloosa, AL, USA (e-mail: szgurbuz@ua.edu).

Conventional techniques for micro-Doppler analysis involve first extracting some set of pre-defined features [8]. These many be physically intuitive features [9], such as Doppler bandwidth and envelope; transform-based computational features, such as discrete cosine coefficients [10] or cepstral coefficients; or speech-processing inspired features, such as mel-frequency cepstral coefficients [11] or linear predictive coding coefficients [12]. After dimension reduction or feature selection [13] [14], a fixed set of features is supplied to a classifier, such as support vector machines (SVM), among others, to process a set of test data. Such techniques have been shown in a variety of works [15] [16] to yield high classification performance.

However, as the number and similarity between classes increases, the performance of classifiers using pre-defined features is significantly degraded [17]. Deep learning, on the other hand, has recently emerged as a powerful technique for classifying imagery, enabled by the immense computational power of modern GPUs. In one of the first works on radar micro-Doppler classification with deep learning in 2016, Kim and Moon [18] utilized a convolutional neural network (CNN) with 3 convolutional layers and 20 filters to classify 7 different activities: running, walking, walking while holding a stick, crawling, boxing while moving forward, boxing while standing in place and sitting still. The network was trained in a supervised fashion with 756 spectrogram images, yielding a classification performance of 90.9% - an accuracy that was roughly the same as that previously reported using physical features and SVM [16]. Later, Jokanovic [19] used a 2-layer autoencoder (AE) structure with a total of 120 data samples to classify 4 classes of activities (falling, sitting, bending, and walking) with an accuracy of 87%.

While the spatially localized convolutional filtering of CNN's are advantageous in capturing local features of input images, the neural network is randomly initialized prior to supervised training. As a result, it is possible that the gradient descent process implemented during training may find a less optimal local minimum, depending upon where the initialization began. In contrast, autoencoders directly learn features from unlabeled data in an unsupervised fashion. Especially in cases when signatures are highly similar, such that a pre-defined feature set capable of distinguishing classes is not readily apparent, unsupervised pre-training that learns features from the data itself can discover nuances in the data that in fact improves classification performance.

A key challenge in applying deep learning to the classification of RF signals, however, is the small amount of data

available in contrast to the millions of images on the internet that could be used to test deep learning in other fields, such as visual data processing. Thus, another important benefit of unsupervised pre-training is that it effectively functions as a regularizer, preventing the network from potentially overfitting the data [20]. A disadvantage of autoencoders, however, is that they fail to capture two-dimensional, spatial variations in the data [21].

This work proposes the use of a deep 3-layer convolutional autoencoder (CAE), which essentially combines the benefits of CNNs and AEs by first using unsupervised pre-training [22] to initialize the network near a good local minima and provide regularization, followed by supervised fine-tuning of the convolutional layers to extract spatially localized features [21] [23]. Moreover, a filter concatenation technique [24] is employed in which different sized filters, namely 3x3 and 9x9, are concatenated to take advantage of multi-level feature extraction. In this work, the classification performance of the proposed CAE architecture is compared with that attainable using a CNN, an AE, and multi-class SVM for the problem of discriminating 12 classes of aided and unaided indoor human activities, such as potentially encountered in assisted living environments.

In Section II, the experimental test set-up and pre-processing steps used to prepare the micro-Doppler dataset used in this study is presented. In Section III and IV, details on optimal selection of pre-defined features for SVM, and optimal DNN architectures are given. In Section V, results for classification performance are compared for each architecture. Finally, in Section VI, important conclusions are discussed.

II. Radar Micro-Doppler Measurements

Radar measurements of human activity were made in an indoor laboratory environment spanning a range of 1 – 5 meters using an NI-USRP 2922 model software-defined radio platform programmed to transmit a continuous wave signal at 4 GHz. Two SAS-571 antennas having a 48° azimuthal beam width were mounted along with the USRP 1 meter above the ground. Measurements were taken such that the direction of motion were directly aligned with the center of the antenna beam pattern. Each gait sample was measured separately over a different run to ensure statistical independence in samples. A picture of the radar system is shown in Figure 1.

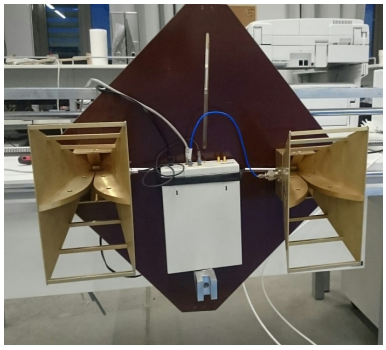


Fig. 1: Configuration of the radar hardware.

A total of 11 different people were used as test subjects to collect 1007 gait samples spanning 12 different classes. Table I summarizes the number of data samples collected per class. The gait for each class was enacted as follows:

1) Walking – medium speed, with fully extended 2-arm swing
2) Jogging – medium speed, two arms held bent at elbows, shortened swing
3) Limping – left foot dragging on ground behind right foot
4) Walking with a cane – "candy-cane" style, single-poled, metal cane used to restrict motion of one arm
5) Walking with walker – two wheeled, metal walker restricting arm swing of both arms
6) Walking with crutches – two metal crutches, one leg bent at knee
7) Crawling – slow advancement on hands and knees
8) Creeping – military-style motion with belly on ground
9) Wheelchair – wheels turned manually with both hands
10) Falling – pretended to "trip" on an object and fall forward onto a mat
11) Sitting – quick motion to sit on a chair
12) Falling from chair – falling sideways off chair on to mat.

A. Micro-Doppler Signatures

In this work, spectrograms were used to represent the time-frequency distribution of the measured micro-Doppler signature. Spectrograms are defined as the modulus of the short-time Fourier transform (STFT):

$$STFT(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-j\omega n} \tag{1}$$

where *x[n]* is the received signal and *w[m]* is a window function. In this work, a hamming window with length of 2048 samples, 4096 FFT points, and 128 samples overlap were utilized. Each spectrogram was then cropped to a duration of 4 seconds, converted to grayscale, normalized between 0 and 1 and saved as an image. To reduce dimensionality, the resulting images were then down-sampled from a size of 656x875 pixels to 90x120 pixels.

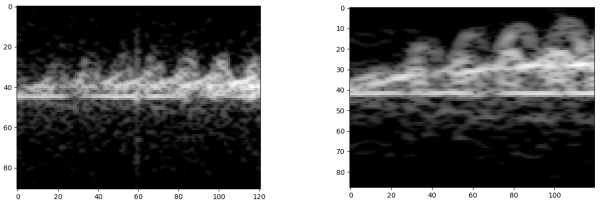
B. Mitigating Effects of Class Imbalance

One issue with data set generation that is frequently overlooked is class imbalance. It is quite common, due to practical constraints on collecting data from multiple subjects, that the number of data samples for each class are not equal. Imbalance in the dataset can cause algorithms to be biased towards the classes having more data. Although the imbalance of 3:1 in this data set is not as severe as that seen in other applications, such as 100:1 or even 100,000:1, rather than truncating the data to achieve balance, in this work, the Synthetic Minority Oversampling Technique (SMOTE) [25] [26] is applied on the training dataset to prevent imbalanced learning and avoid overfitting. The SMOTE algorithm equalizes class data by oversampling minority classes with "synthetic" samples. First, the pixels of the 2-dimensional spectrogram are converted into a concatenated 1-dimensional vector. Then, the synthetic samples are generated by taking the difference between the

TABLE I: Number of Samples Per Class (F.C denotes falling from chair)

Class	Walking	Jogging	Crawling	Limping	Cane	Falling	Wheelchair	Crutches	Sitting	Walker	F.C.	Creeping
Samples	71	72	74	104	123	53	149	74	50	121	60	56

sample under consideration and its nearest neighbor. This difference is multiplied by a random number between 0 and 1, and added to the sample under consideration. This procedure can be generalized for k-nearest neighbors and N% oversampling. For example, consider a case where 5-nearest neighbors are used for 200% oversampling. Two of the 5 nearest neighbors are chosen and random samples are generated in the direction of each nearest neighbor selected. The SMOTE algorithm was used in conjunction with 10-fold cross validation by first randomly selecting 90% of the data for training and the remaining 10% for testing. The SMOTE algorithm is then applied on the training data only, using 5-nearest neighbors and varying the amount of oversampling so as to equalize the number samples in each class. Only real, measured data is used in the testing process. Figure 2 shows two examples of SMOTE-generated synthetic spectrograms for walking and jogging, which may be compared to the measured spectrograms for each class shown in Figure 3.



(a) Synthetic walking spectrogram (b) Synthetic jogging spectrogram

Fig. 2: SMOTE-generated synthetic spectrograms: (a) walking and (b) jogging.

III. Classification with Pre-Defined Features

One of the most often used methods in the literature for the classification of micro-Doppler signatures is classification with pre-defined features - features that are extracted through a fixed computation process. A plethora of features [8] have been defined in the literature, including physical, transform-based, and speech-processing inspired features. A wide range of these features is considered in this work, as defined next.

A. Feature Definitions

Physical features are direct measurements of properties of the spectrogram or cadence velocity diagram (CVD) and relate to human gait parameters. The CVD [27] [28] [29] is defined as the Fourier transform of the spectrogram along each frequency bin

$$\Delta(v,\omega) = \left| \sum_{n=0}^{N-1} STFT(n,\omega) e^{\frac{-j2\pi nv}{N}} \right| \tag{2}$$

and provides a measure of how often the different velocities repeat (i.e. cadence frequencies). The 13 physical features utilized in this work are:

1) Bandwidth of torso response
2) Mean of the torso response
3) Minimum value of the upper envelope
4) Maximum value of the upper envelope
5) Mean value of the upper envelope
6) Minimum value of the lower envelope
7) Maximum value of the lower envelope
8) Mean value of the lower envelope
9) Overall Doppler bandwidth
10) Difference between upper and lower envelope averages
11) Gait frequency, i.e. fundamental frequency of CVD
12) Second harmonic of CVD
13) Third harmonic of CVD

Transform-based features, as computed from the first 10 coefficients of the discrete-cosine transform (DCT) of the micro-Doppler frequency, $x[n]$, as

$$C[k] = h[k] \sum_{t=0}^{T-1} x[n] \cos\left[\pi(n+\frac{1}{2})\frac{k}{T}\right] \tag{3}$$

where T is the length of the observed radar signature, $k \in [0, n-1]$, and $h[k]$ is defined as

$$h[k] = \begin{cases} \sqrt{\frac{1}{T}} & for\ k = 0 \\ \sqrt{\frac{2}{T}} & otherwise \end{cases} \tag{4}$$

Along with the first 3 cepstral coefficients, and 101 linear predictive coding coefficients (LPC) are also used as features in this work. LPC and cepstral coefficients are features originally proposed in speech processing literature, but which have also found utility in micro-Doppler analysis. LPC's are computed by representing this signal as the linear combination of past values:

$$\hat{x} = \sum_{k=1}^{p} a[k]x[n-k] \tag{5}$$

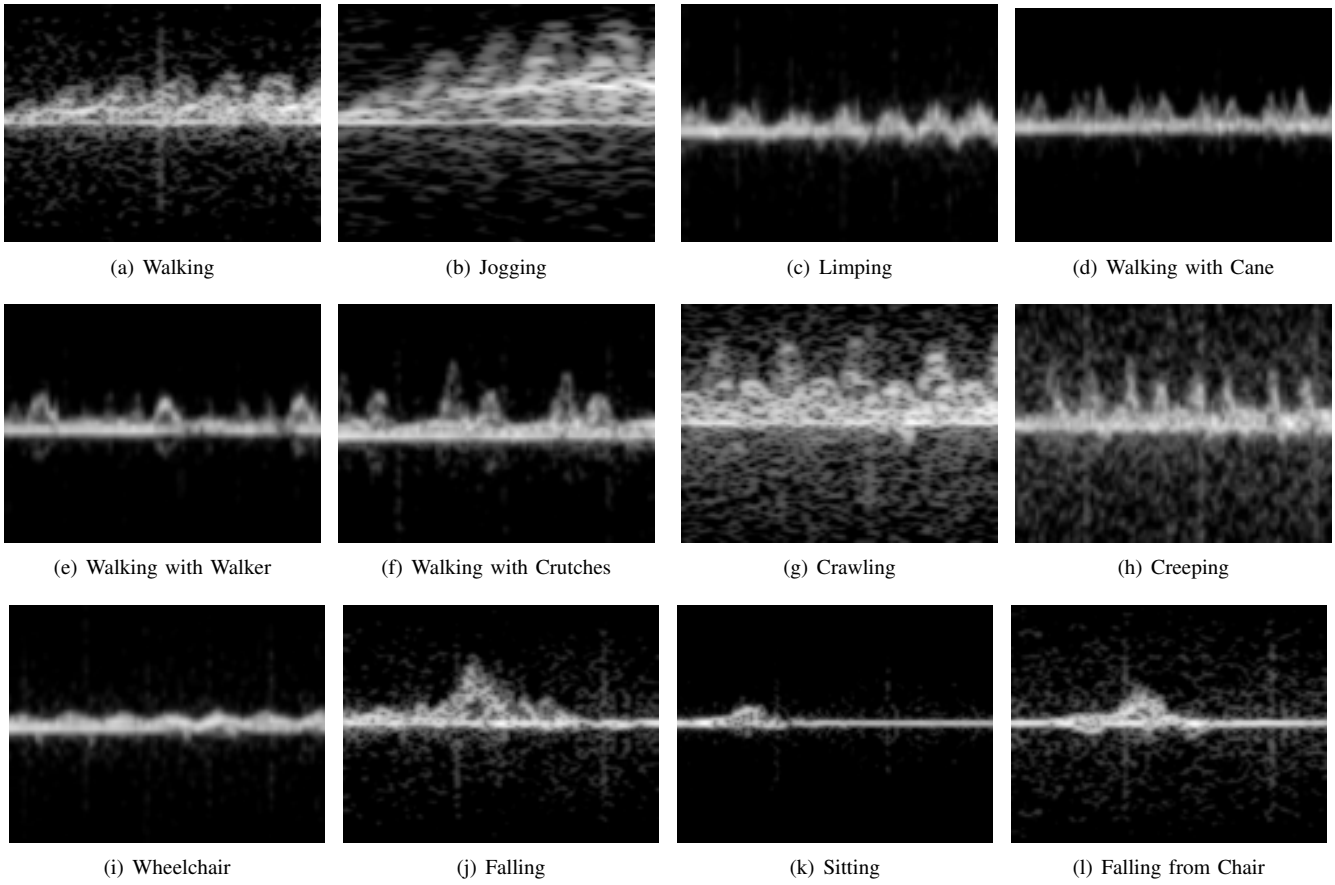
where $a[k]$ are the LPCs and p is the total number of LPCs. To compute the LPCs, the difference between the model in (5) and the true signal – i.e, the error $e[n] = x[n] - \hat{x}[n]$ – is sought to be minimized. Many methods can be employed for this minimization, such as computing the autocorrelation followed by a Levinson-Durbin recursion.

The cepstrum, *c[n]*, is defined as the inverse DFT of the log magnitude of the DFT of the received radar return *x[n]*:

$$c[n] = F^{-1}[log|F[x[n]|] \tag{6}$$

where $F[.]$ is the Fourier transform.

To summarize, a total of 127 features is extracted: 10 DCT, 3 cepstral, 13 physical and 101 LPC.



(a) Walking (b) Jogging (c) Limping (d) Walking with Cane

(e) Walking with Walker (f) Walking with Crutches (g) Crawling (h) Creeping

(i) Wheelchair (j) Falling (k) Sitting (l) Falling from Chair

Fig. 3: Examples of measured spectrograms for each human activity class.

B. Feature Selection and Classifier Comparison

Due to the curse of dimensionality, utilizing all possible features does not necessarily guarantee the greatest performance. Dimension reduction [30] and feature selection [8] [13] [31] methods have been shown to yield significant improvements in classification accuracy. In this work, the Sequential Backward Elimination (SBE) method for feature selection is employed [32]. SBE is a wrapper method, which searches to find the combination of features that yield the greatest accuracy for a specific classifier. SBE initializes the feature space by utilizing all features then starts to remove features iteratively. This procedure is recursively repeated until a specified number of features have been selected.

The performance of three different classifiers is compared for different numbers of features, as selected by SBE: multi-class SVM, a popular baseline used in the literature, random forest [33] [34] [35] and extreme gradient boosting (xgboost) [36], two other classifiers have been recently reported to give good results. The performance for SVM was compared for three different kernels - linear, polynomial, and radial basis function (RBF) - among which the linear kernel was found to yield the best results. The model parameters for the xgboost and random forest classifiers was optimized using a grid search over two parameters: the number of trees in the forest and maximum depth of the tree. The best results were achieved for random forest with a 50 trees and a depth of 20, and for xgboost with 50 trees and a depth of 10.

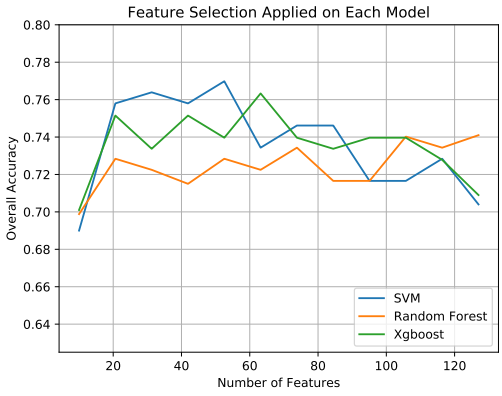


Fig. 4: Comparison of feature selection results for SVM, random forest and xgboost classifiers.

After optimization of classifier parameters, performance was compared as a function of number of features, as shown in Figure 4. The best performance was achieved by SVM with a linear kernel using 50 features selected with SBE; namely, the bandwidth of the torso response, mean torso frequency, mean of the upper envelope, mean of the lower envelope, first 2 CVD

features, first 2 cepstral coefficients, 37 LPC coefficients and 5 DCT features. The effect of feature selection is somewhat less apparent in RF and Xgboost classifiers, in part because tree-based models employ impurity based feature rankings, implicitly providing for feature selection during training.

IV. Deep Learning Architectures

Deep learning is a method for machine learning that recently has experienced a resurgence due to the increased computing capabilities offered by modern GPU's and advances in algorithms. Deep neural networks build upon past research on artificial neural networks (ANNs) by increasing the overall size of the network using many layers of neurons. Each neuron is formed by linearly weighting multiple inputs supplied to an activation function. Formerly, it was common for sigmoid or hyperbolic tangent functions to be used for activation; however, network size was limited by what is known as the "vanishing gradient" problem. Neural networks are trained by using a gradient descent algorithm during backpropagation, which functions to minimize a pre-defined loss function. However, during backpropagation the error decreases as it flows through each layer, making training slow or even impossible as the number of layers increases. This problem was solved through the recent proposal of using rectified linear units (ReLU) as activation functions [37]. ReLU are mathematically defined as $f(x) = max(0, x)$ and have an output of zero for negative input, but a linear output for positive input. By not squashing the data between 0 and 1, ReLU has been shown to prevent the vanishing gradient problem and has the additional advantage of enabling a sparse representation of the data when then network is initialized randomly [38]. In this way, ReLU units have enabled the design of modern-day "deep" networks yielding incredible performance gains in the classification of massive datasets.

In particular, convolutional neural networks became popular when an 8-layer architecture, AlexNet [39], won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) in 2012. Subsequently, a 16-layer CNN architecture, called VGG-Net [40], and a 152-layer architecture built by Microsoft called ResNet [41], won the competition in 2014 and 2016, respectively. Current research in deep learning involves processing over millions images or videos into thousands of classes.

This has led researchers in the radar community to also experiment with deep learning architectures to classify RF signals. However, a fundamental challenge in applying deep neural networks to RF signal classification is the limitation in data set size. RF data measurements are much more difficult, time consuming, and costly to collect, especially for commercial ground surveillance radar systems. It is not practical to collect millions of radar micro-Doppler signatures and thus algorithms must be designed so as to avoid overfitting – the memorization of small data sets by high-complexity deep architectures. Thus, although a 14-layer deep CNN architecture has been recently proposed for human multi-target gait classification [42], the use of 4-layers [43] or less has been more common. In this work, we consider 3-layer CNN and AE architectures for comparison with the proposed 3-layer CAE architecture.

A. Autoencoder

An autoencoder (AE) is a feed-forward neural network that aims to reconstruct the input at the output under certain constraints. In other words, for a given input vector of x, the AE tries to approximate $h_w(x) \approx x$. In 2006, an unsupervised pre-training algorithm was proposed for initializing the weights and biases of autoencoders [44] that was highly effective when only a small number of labeled training samples were available [23] [45]. Autoencoders implement unsupervised pre-training by first encoding and then decoding the inputs.

For a given input vector x, the encoder computes a non-linear mapping of the inputs as

$$e_i = \sigma(Wx_i + b). \tag{7}$$

Here, σ denotes a non-linear activation function, W denotes weights and b denotes the biases of the encoder. The encoded features are then decoded to reconstruct the given input vector x using

$$z_i = \sigma(\widetilde{W}e_i + \widetilde{b}). \tag{8}$$

Here $\widetilde{W}$ and $\widetilde{b}$ denotes weights and biases of the decoder. During unsupervised pre-training, the network tries to minimize the reconstruction error

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}(x_i - z_i)^2 \tag{9}$$

by adjusting its weights and biases $\theta = [W, b, \widetilde{W}, \widetilde{b}]$. To prevent the network from learning the identity function, a sparsity parameter is added to the cost function. This parameter forces the network to learn the correlation between the given input vectors [46]. After adding the sparsity parameter, the cost function, thus, becomes

$$argmin_\theta\ J(\theta) = \frac{1}{N}\sum_{i=1}^{N}(x_i - z_i)^2 + \beta\sum_{j=1}^{h} KL(p||p_j). \tag{10}$$

Here, h denotes the number of neurons in the hidden layer, β denotes sparsity proportion and $\sum_{j=1}^{h} KL(p||p_j)$ denotes the Kullback-Leibler (KL) divergence between the Bernoulli random variables with mean p and p_j respectively. The KL divergence between two random variables is given as follows:

$$KL(p||p_j) = plog(\frac{p}{p_j}) + (1-p)log(\frac{1-p}{1-p_j}), \tag{11}$$

where p_j denotes the activation of jth neuron in the hidden layer and p is the desired average activation value. Since h is the number of neurons in the hidden layer, the KL divergence term forces hidden unit activations within the proximity of p.

After unsupervised pre-training, the decoder is removed from the network and the remaining encoder components are trained in a supervised manner by adding a softmax classifier with 12 neurons after the encoder. The softmax classifier is a multinomial version of logistic regression. For a given input x_i, the softmax function estimates the probability that $P(y_k|x_i)$ for $k = 1, 2, ..., K$, where K denotes the number of classes (12 in this case). In other words, the probability that the input x_i belongs to the class label y_k is estimated. Mathematically, the class probability p_k can be given as

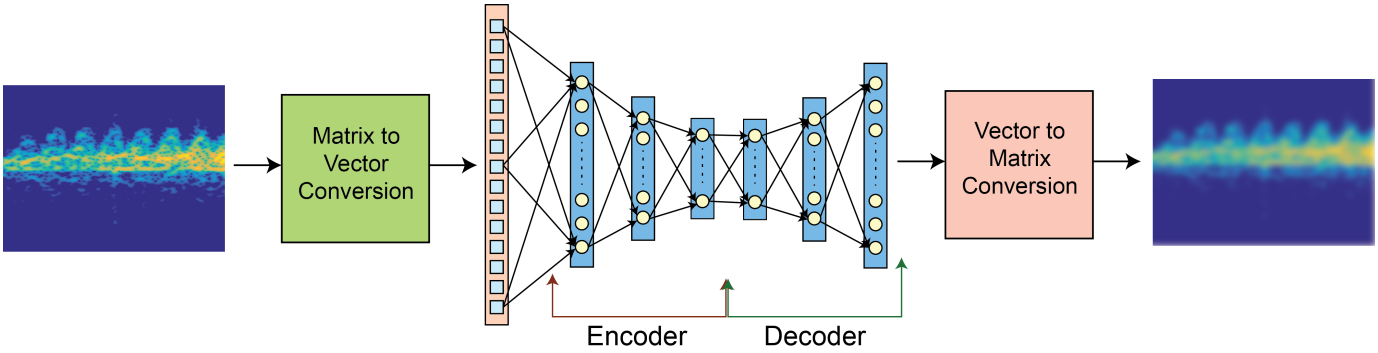


Fig. 5: The 3-layer AE, where encoder layers have 200-100-50 neurons and decoder layers have 50-100-200 neurons.

TABLE II: Parameter Optimization Table for AE (Acc. denotes accuracy)

Depth	Width	Acc. (%)	Depth	Width	Acc. (%)
1	20	74.1	4	20-50-100-200	83.4
1	50	76	4	40-100-200-400	83.1
1	100	75.5	4	100-200-400-800	79.8
2	20-50	79.8	5	20-50-100-200-400	82.7
2	50-100	81.1	5	40-100-200-400-800	82.8
2	100-200	80.9	5	100-200-400-800-1600	79.5
3	20-50-100	83.4	6	20-50-100-200-400-800	80.1
3	40-80-160	83.2	6	40-100-200-400-800-1600	78.9
3	**50-100-200**	**84.1**	6	100-200-400-800-1600-3200	78.3

$$p(y = k|x_i) = \frac{e^{\theta_k x_i}}{\sum_{k=1}^{K} e^{\theta_k x_i}}. \tag{12}$$

The weights and biases of the network, θ, can be optimized by minimizing the following cost function:

$$J(\theta) = -\sum_{i=1}^{N}\sum_{k=1}^{K} 1\{y^i = k\} log \frac{e^{\theta_k x_i}}{\sum_{k=1}^{K} e^{\theta_k x_i}}, \tag{13}$$

where $1\{.\}$ denotes the indicator function and N denotes the number of labeled samples. Equation 13 can be solved with a gradient based algorithm. This process is called fine-tuning, where the network is being trained in supervised fashion. During fine tuning, instead of mean square error, categorical cross-entropy loss is selected as the loss function.

Autoencoders can be stacked hierarchically such that upper layers receive inputs from the outputs of the layers below. A rectified linear unit (ReLU) activation function is used for non-linearity. KL divergence term for sparsity regularization is selected to be 2 and β is selected as 0.1. The optimization of both unsupervised pre-training and fine tuning is computed using the Adaptive Moment Estimation (ADAM) [47] algorithm with a learning rate of 0.001. Using a grid search to determine the optimal depth and width without overfitting (summarized in Table II), we chose to implement a 3-layer autoencoder, with layers of 200-100-50 neurons, respectively. The overall AE architecture is shown in Figure 5.

B. Convolutional Neural Network (CNN)

CNNs have recently achieved great success in image classification due to their ability to learn locally connected features. CNNs generally consist of three elements: convolutional layers, pooling layers and fully connected layers [48]. In the convolutional layer, filters are convolved with the receptive field of the input image in a sliding window style to learn data specific features. Basic features, such as lines, edges, and corners, are learned in the initial layers, while more abstract features are learned as layers go deeper. For a given matrix P, m^{th} neuron in the convolutional neural network calculates

$$M[i,j] = \sigma\left(\sum_{x=-2k-1}^{2k+1}\sum_{y=-2k-1}^{2k+1} f_m[x,y]P[i-x, j-y] + b\right), \tag{14}$$

where the origin is as defined being in the center of the image, so that the edge of the image is k-pixels from the origin pixel in either the direction of the x- or y-axis. Then, the size of each side of the image is $2k+1$, M is the activation map of the given input P, f_m is the m^{th} convolution filter and σ is the non-linear activation function. ReLU is used as the activation function.

Generally, a pooling layer follows each convolutional layer. Max-pooling is basically a non-linear down-sampling procedure, which takes the maximum of 2x2 neighborhoods of the image, and helps to reduce the computational complexity for the forward layers, as well as adding translation invariancy to the network.

Fully connected layers are used to learn the non-linear combinations of extracted features from previous layers. Dropout is recommended as a way of preventing overfitting by disabling randomly chosen neurons and their connections [49]. The dropped neurons stay inactive during the feedforward and backpropagation phases, thus forcing the network to learn different non-linear combinations of features on each epoch.

In this work, a filter concatenation technique is also applied to capture features of different resolutions from the input [24]. Two convolutional filters of different sizes are used. The

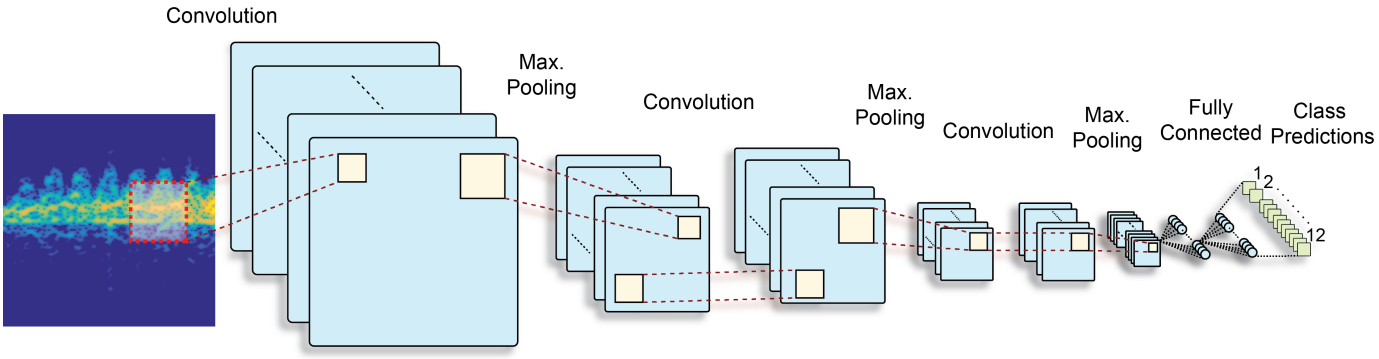


Fig. 6: The CNN architecture implemented with three convolutional layers comprised of 30 3x3 filters each, and two fully connected layers with 150 neurons/layer.

TABLE III: Parameter Optimization Table for CNN

Filter Size	Depth	Width	Acc(%)	Depth	Width	Acc(%)	Depth	Width	Acc(%)
3x3 - 9x9	1	5	80.2	3	5	88.4	5	5	87.2
	1	30	81.4	**3**	**30**	**90.1**	5	30	88.1
	1	100	81.5	3	100	90	5	100	88.4
	2	5	82.1	4	5	87.6	6	5	84.9
	2	30	86.2	4	30	89.9	6	30	85.5
	2	100	87.1	4	100	89.5	6	100	86.8
2x2 - 7x7	1	5	79.6	3	5	84.9	5	5	87.8
	1	30	80.4	3	30	88.8	5	30	88
	1	100	81.7	3	100	89.5	5	100	88.6
	2	5	83.4	4	5	86.9	6	5	85.3
	2	30	86.1	4	30	88.8	6	30	86.1
	2	100	87.6	4	100	89.3	6	100	86.2

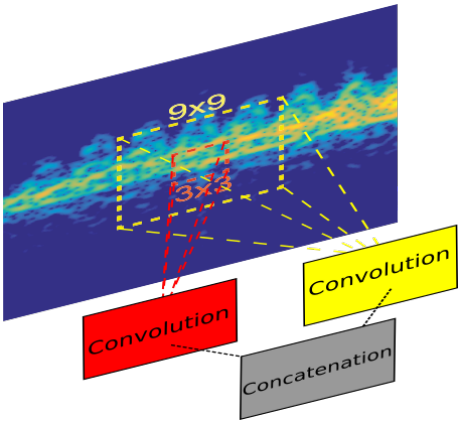


Fig. 7: Proposed filter concatenation for CNN and CAE.

larger filter captures more general features, while the smaller filter captures fine details. In this way, filter concatenation increases computational cost in return for improvement in classification performance. Figure 7 illustrates the concept of filter concatenation.

In Table III, we compared the effect of filter size, as well as depth and width on the resulting accuracy. Based on this analysis, we chose filter sizes of 3x3 and 9x9 for concatenation, together with 3 convolutional layers using 30 3x3 filters. After each convolutional layer, a 2x2 max pooling operation is applied. At the end of the network, the learned features are flattened to enable input to the fully connected layers. Two fully connected layers with 150 neurons in each layer are employed. After each fully connected layer, a dropout operation is applied with a probability of 0.5. Lastly, the network is connected to a softmax classifier with 12 inputs - the total number of classes. The same objective function given in Equation 13 is optimized using ADAM. The CNN architecture is shown in Figure 6.

C. Convolutional Autoencoder (CAE)

Convolutional autoencoders combine the benefits of convolutional filtering in CNN's with unsupervised pre-training of autoencoders. In contrast to the topology for autoencoders, however, instead of the fully connected layers, the encoder contains convolutional layers and the decoder contains deconvolutional layers. Deconvolutional filters may be transposed versions of the convolutional filters; or, as is done in this work, they may be learned from scratch. Additionally, each deconvolutional layer must be followed by an unpooling layer [50]. The unpooling operation is performed by storing the locations of the maximum values during pooling, preserving the values of these locations during unpooling and zeroing the rest.

Spatial locality is preserved by incorporating a convolution operation at each neuron. Thus, for a given input maxrix P, the encoder computes

$$e_i = \sigma(P * F^n + b) \tag{15}$$

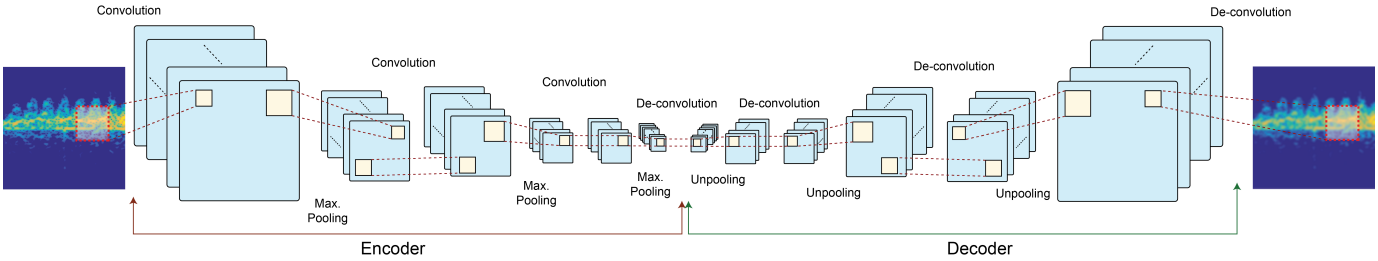


Fig. 8: Proposed CAE architecture showing convolutional and deconvolutional layers. After unsupervised pre-training the decoder part is removed and 2 fully connected layers and a softmax classifier are added at the end of encoder.

TABLE IV: Parameter optimization table for CAE

Filter Size	Depth	Width	Acc(%)	Depth	Width	Acc(%)	Depth	Width	Acc(%)
3x3 - 9x9	1	5	83.8	3	5	92.2	5	5	93.2
	1	30	84.9	**3**	**30**	**94.2**	5	30	93.4
	1	100	84.7	3	100	93.2	5	100	93.9
	2	5	88.2	4	5	94	6	5	90.1
	2	30	88.4	4	30	94.1	6	30	90.9
	2	100	89	4	100	93.7	6	100	90.3
2x2 - 7x7	1	5	82.1	3	5	92.4	5	5	92.1
	1	30	84.4	3	30	93.9	5	30	92.9
	1	100	84.5	3	100	93.8	5	100	91.2
	2	5	88	4	5	92.8	6	5	89.8
	2	30	88.5	4	30	93.2	6	30	90.5
	2	100	88.6	4	100	92.9	6	100	90.3

where σ denotes activation function, * represents 2D convolution, F^n is n^{th} 2D convolutional filter and b denotes encoder bias. To retain spatial resolution, zero padding is applied to the input matrix P. Then, the reconstruction can be obtained using

$$z_i = \sigma(e_i * \widetilde{F}^n + \widetilde{b}). \tag{16}$$

Here, z_i is the reconstruction of i^th input $\widetilde{F}^n$ denotes n^{th} 2D convolutional filter in decoder and $\widetilde{b}$ is bias of decoder. Unsupervised pre-training can be applied to the network, which aims to minimize following equation

$$E(\theta) = \sum_{i=1}^{m}(x_i - z_i)^2 \tag{17}$$

After unsupervised pre-training the unpooling and deconvolutional layers, the decoding part of the network is removed and fully connected layers as well as a softmax classifier are added at the end of the network. Then the network can be fine-tuned by optimizing Equation 13. As done with the CNN, a ReLU activation function is used, as well as the ADAM algorithm for optimization of the two fully connected layers with 150 neurons each, and dropout with a 0.5 probability on each fully connected layer.

The optimization of hyperparameters was done through grid search, as given in detail in Table IV. From these results, we chose to implement a CAE with 3 convolutional layers and populate the convolutional and deconvolutional layers with 30 3x3 and 9x9 concatenated filters. The overall structure of the proposed CAE is shown in Figure 8.

V. Results

A. Classification Accuracy

In this work, deep learning models are implemented in Python using Keras [51], which uses Tensorflow as its tensor manipulation library [52]. All classifiers described in Sections III and IV are tested using 10-fold cross-validation of the 4 GHz CW radar data set described in Section II. Each deep learning architecture is trained for 280 epochs with a minibatch size of 100. The validation accuracy is computed by setting aside 20% of training samples as the validation set, and then evaluating the model after each epoch using the validation set.

To demonstrate the value of unsupervised pre-training, as offered by the proposed CAE, we compare two cases:

1) Randomly initialization of the CNN
2) Unsupervised pre-training over 20 epochs to initialize the proposed CAE

In both cases, the learned weights are supplied as features to a multi-class SVM classifier and results are compared. We found that while random initialization of the CNN resulted in a classification accuracy of just 8.3%, the CAE using unsupervised pre-training achieved %83.4 - a drastically better result.

Of course, in the proposed approach we ultimately do not use just unsupervised pre-training, but additionally apply supervised fine-tuning and use a softmax classifier, instead of an SVM. Figure 9 compares the variation of validation accuracy as a function of epoch for all three deep learning architectures considered. Notice that after just the first few epochs, the methods that employed unsupervised pre-training, namely AE and CAE, have much higher validation accuracies of 51% and 62%, respectively, in comparison to the roughly 31% performance of the CNN. During unsupervised pre-training the

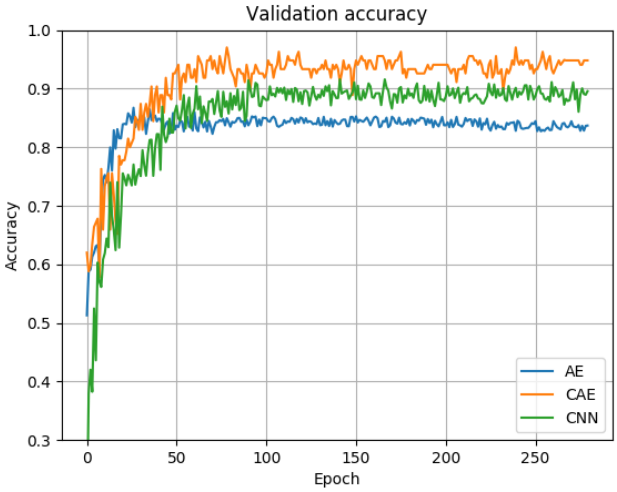


Fig. 9: Comparison of Validation Accuracies for AE, CNN and CAE

networks were trained for 3 epochs, yielding a mean square error of 0.01 and 0.5 for CAE and AE, respectively. The validation accuracy curves level off after about 75 epochs, and remain roughly constant thereafter, confirming that there is no overfitting in any of the deep architectures implemented.

Confusion matrices summarizing the classification accuracy of each technique are given in Tables V-VIII for multi-class SVM, AE, CNN, and the proposed CAE architecture, respectively, while precision and recall results for all methods are given in Table IX. In these tables, green indicates classification accuracy, red denotes high error, i.e. >20% confusion, brown denotes errors between 10% and 20%, and, lastly, yellow denotes errors lower than 10%. These results show that the performance of DNNs are between 7% to 17% higher than that achieved using a select set of pre-defined features supplied to multi-class SVM. This is a significant result as it shows the clear benefits of learning features directly from the data, especially in cases when the classes being considered are highly similar and visual difference between micro-Doppler signatures are minimal.

Among deep learning architectures, the proposed CAE architecture surpasses the performance attained with the CNN or AE. An overall classification accuracy of 94.2% was achieved with the CAE, while the CNN and AE yielded classification accuracies of 90.1% and 84.1%, respectively. This represents a 4% - 10% improvement in performance, primarily due to the exploitation of unsupervised pre-training together with the benefits of spatially localized processing offered by convolutional filtering and filter concatenation.

B. Discussion of Classification Results

The AE exhibited significant confusions between some inherently similar classes, namely:

- Using a cane with limping and using a walker,
- Falling with falling off of a chair and sitting, and
- Crawling with creeping,

while the confusion of crawling with jogging was more surprising. The CNN was able to distinguish falling off a chair and sitting, but the confusion between falling forward and falling of a chair was significantly higher. Moreover, the confusion between limping and using a walker, and using a cane; as well as using crutches, creeping, crawling, and jogging was more evident. In contrast, the proposed CAE architecture had no problems differentiating falling forward from falling off a chair, while some amount of confusion between limping-cane, cane-walker, and creeping-crawling remained. The CAE did confuse falling and sitting for 9.4% of the relevant samples, but the confusion between creeping and crawling was reduced by %8.4. Creeping is seen to have the highest in class variance due to difficulty of subjects in advancing forward on the stomach without rising upon their knees and elbows- an act that increases similarity with crawling. Moreover the confusion in creeping also reflects the difficulty in generalizing an activity that is not consistent. Overall, the proposed CAE architecture outperformed the multi-class SVM, CNN, and AE architectures for all activity classes, except creeping and falling, for which case the difference was under 1%.

C. Computational Complexity

A comparison of computation time and number of network parameters is given in Table X. These numbers are based on running all deep learning networks on a system with the following specifications: NVIDIA Tesla K80 GPU with 24 GB of VRAM, 378 GB of ram and Intel Xeon E5 2683 processor. Utilization of the GPU accelerates computations by a factor of 10 (approximately) and is essential for deep learning implementations. In contrast, conventional methods using pre-defined features are much faster and less complex, therefore, GPUs are not required and training on a CPU is feasible. The 37 seconds duration reported for SVM involves both feature extraction and classifier training. In general, the computation time will depend on the numbers of features to be selected. Results are shown for 50 features.

In contrast, deep learning algorithms require vast amounts of computational resources. The convolutional operations used by the CNN and CAE add complexity relative to the computations

TABLE V: Confusion matrix for multi-class SVM ad 50 selected features (overall accuracy is 76.9%).

%	Walking	Wheelchair	Limping	Cane	Walker	Falling	Crutches	Creeping	Crawling	Jogging	Sitting	F.C
Walking	**100**	0	0	0	0	0	0	0	0	0	0	0
Wheelchair	0	**83.9**	3.2	12.9	0	0	0	0	0	0	0	0
Limping	0	0	**72.4**	15.6	12	0	0	0	0	0	0	0
Cane	0	3.8	0	**76.9**	19.3	0	0	0	0	0	0	0
Walker	0	4.4	0	8.7	**86.9**	0	0	0	0	0	0	0
Falling	25	0	0	0	0	**55**	0	0	0	0	0	20
Crutches	0	0	0	0	0	0	**100**	0	0	0	0	0
Creeping	50	0	0	0	0	0	0	**37.5**	12.5	0	0	0
Crawling	0	14.3	0	0	0	0	0	43.1	**28.3**	14.3	0	0
Jogging	0	0	0	0	0	0	0	0	0	**95.7**	0	4.3
Sitting	0	0	0	0	0	0	0	0	0	0	**100**	0
F.C	0	0	0	0	0	0	0	0	0	0	14.3	**85.7**

TABLE VI: Confusion matrix for autoencoder (overall accuracy is 84.1%).

%	Walking	Wheelchair	Limping	Cane	Walker	Falling	Crutches	Creeping	Crawling	Jogging	Sitting	F.C
Walking	**91.1**	0	0	0	0	0	0	8.9	0	0	0	0
Wheelchair	0	**91.4**	2.9	0	4.1	0	0	1.6	0	0	0	0
Limping	0	0	**84.4**	7.3	8.3	0	0	0	0	0	0	0
Cane	0	0	10.1	**69.1**	20.8	0	0	0	0	0	0	0
Walker	0	0.1	0.1	9.1	**90.7**	0	0	0	0	0	0	0
Falling	0.5	0	0	0	0	**89.3**	0	0	0	0	0	10.2
Crutches	7	0	0	0	0	0	**91.5**	0	1.5	0	0	0
Creeping	0	0	0	0	0	0	0	**70.4**	29.6	0	0	0
Crawling	2.1	0	0	0	0	0	0	24.5	**60.9**	12.5	0	0
Jogging	0.4	0	0	0	0	0	0	0	0	**98.1**	0	1.5
Sitting	0	0	0	0	0	19.6	0	0	0	0	**80.4**	0
F.C	0	0	0	0	0	8.5	0	0	0	0	0	**91.5**

TABLE VII: Confusion matrix for CNN (overall accuracy is 90.1%)

%	Walking	Wheelchair	Limping	Cane	Walker	Falling	Crutches	Creeping	Crawling	Jogging	Sitting	F.C
Walking	**100**	0	0	0	0	0	0	0	0	0	0	0
Wheelchair	0	**100**	0	0	0	0	0	0	0	0	0	0
Limping	0	0	**86.2**	13.8	0	0	0	0	0	0	0	0
Cane	0	0	3.9	**81.9**	14.2	0	0	0	0	0	0	0
Walker	0	0	0	10.3	**89.7**	0	0	0	0	0	0	0
Falling	0	0	0	0	0	**86.4**	0	0	0	0	0	13.6
Crutches	0	0	0	0	0	0	**100**	0	0	0	0	0
Creeping	0	0	0	0	0	0	10.3	**51.2**	30.8	7.7	0	0
Crawling	0	0	0	0	0	0	0	0	**85.7**	14.3	0	0
Jogging	0	0	0	0	0	0	0	0	0	**100**	0	0
Sitting	0	0	0	0	0	0	0	0	0	0	**100**	0
F.C	0	0	0	0	0	0	0	0	0	0	0	**100**

done by a single neuron in the autoencoder; however, all of the AE neurons are fully connected, resulting in actually the greatest number of parameters, and hence, longest training time. The filter concatenation technique, which improves accuracy, also comes with a computational cost, reflected in greatly increasing the number of convolutional parameters and more than doubling the total number of parameters in the network. Unsupervised pre-training likewise improves performance but adds to the total training time required.

D. Discussion of Deep Learning of Features

One way of getting a better understanding of how the propsed CAE architecture better learns features is to visualize the convolutional filters of each layer. To do this, the the activation maximization technique [53] was employed, which works by modifying a randomly generated image such that the activation of a selected filter is maximized. Mathematically, this can be achieved by maximizing the following loss function:

$$x_d = \underset{x}{\operatorname{argmax}}(h_{ik}(\theta, x)) \tag{18}$$

where x is the randomly generated input image, θ are the weights and biases of the network and h_{ik} is the activation of the k^{th} filter in the i^{th} layer. The variable x_d is the desired result, i.e the image yields highest activation for h_{ik}. This optimization problem can be solved by using the gradient ascent algorithm:

$$x = x + \alpha \frac{\partial(h_{ik}(\theta, x))}{\partial x} \tag{19}$$

where α is the step size. The gradient $\frac{\partial(h_{ik}(\theta,x))}{\partial x}$ yields the direction in which to update the input image x in order to have the highest activation for h_{ik}. Figure 8 shows first 25 filters which have obtained by computing Equation 19 for 20 steps with $\alpha = 0.8$. The filters shown in Figure 9 have the highest loss according to Equation 18.

From Figure 9 it can be seen that the network is learning to extract basic orientations on the first layer: the 9x9 filters

TABLE VIII: Confusion matrix for convolutional autoencoder (overall accuracy is 94.2%)

%	Walking	Wheelchair	Limping	Cane	Walker	Falling	Crutches	Creeping	Crawling	Jogging	Sitting	F.C
Walking	**100**	0	0	0	0	0	0	0	0	0	0	0
Wheelchair	0	**100**	0	0	0	0	0	0	0	0	0	0
Limping	0	0	**93.1**	6.9	0	0	0	0	0	0	0	0
Cane	0	0	0	**91.1**	8.9	0	0	0	0	0	0	0
Walker	0	0	0	0	**100**	0	0	0	0	0	0	0
Falling	0	0	0	0	0	**89**	0	0	0	0	9.4	1.6
Crutches	0	0	0	0	0	0	**100**	0	0	0	0	0
Creeping	4.9	0	0	0	0	0	7.2	**65.5**	22.4	0	0	0
Crawling	0	0	0	0	0	0	0	8.5	**91.5**	0	0	0
Jogging	0	0	0	0	0	0	0	0	0	**100**	0	0
Sitting	0	0	0	0	0	0	0	0	0	0	**100**	0
F.C	0	0	0	0	0	0	0	0	0	0	0	**100**

TABLE IX: Precision (P), recall (R) and accuracy (A) values for all methods.

	SVM w/F.S.			**AE**			**CNN**			**CAE**		
Gait	P	R	A	P	R	A	P	R	A	P	R	A
Walking	1	0.57	**100**	0.91	0.9	91.1	1	1	**100**	1	0.95	**100**
Wheel chair	0.84	0.79	83.9	0.91	1	91.4	1	1	**100**	1	1	**100**
Limping	0.72	0.96	72.4	0.84	0.87	84.4	0.86	0.96	86.2	0.93	1	**93.1**
Cane	0.77	0.67	76.9	0.69	0.81	69.1	0.82	0.77	81.9	0.91	0.93	**91.1**
Walker	0.87	0.74	86.9	0.91	0.73	90.7	0.9	0.86	89.7	1	0.92	**100**
Falling	0.55	1	55	0.89	0.76	**89.3**	0.86	1	86.4	0.89	1	89
Crutches	1	1	**100**	0.92	1	91.5	1	0.91	**100**	1	0.93	**100**
Creeping	0.38	0.47	37.5	0.7	0.67	**70.4**	0.51	1	51.2	0.66	0.89	65.5
Crawling	0.28	0.69	28.3	0.61	0.66	60.9	0.86	0.74	85.7	0.92	0.8	**91.5**
Jogging	0.96	0.87	95.7	0.98	0.89	98.1	1	0.82	**100**	1	1	**100**
Fastly Sitting	1	0.87	**100**	0.8	1	80.4	1	1	**100**	1	0.91	**100**
Falling from Chair	0.86	0.78	85.7	0.92	0.89	91.5	1	0.88	**100**	1	0.98	**100**
Average	0.77	0.78	76.9	0.84	0.85	84.1	0.9	0.91	90.1	**0.94**	**0.94**	**94.2**

TABLE X: Computational complexity for all methods (F.S. denotes feature selection and F.C. denotes filter concatenation)

	SVM	**SVM w/F.S.**	**AE**	**CNN**	**CNN w/F.C.**	**CAE**	**CAE w/F.C**
Total Params.	-	-	6.361.562	832.428	1.980.324	832.428	1.980.324
Conv. Params.	-	-	-	18.816	371.712	18.816	371.712
Total Time	**37 s.**	99 s.	**943 s.**	157 s.	463 s.	241 s.	747 s.

(top three rows) learn more generalized shapes, whereas the 3x3 filters (bottom two rows) learn more specific shapes. As the layers go deeper, the second and third layers learn more abstract shapes, as can be seen by the clear horizontal lines of the 3rd filter in the first row of Layer 2 and the fourth filter in the first row of Layer 3.

Also, it may be observed that some filters have learned clutter and noise. This is significant because in this work no clutter cancellation or clutter filtering has been applied to remove ground clutter, visible as a narrow band near 0 Hz Doppler frequency. In fact, past work has shown that clutter has a significant affect on classifier performance when pre-defined features are used, so that clutter filtering is typically beneficial [54]. But, in the case of deep learning architectures, and, in particular, the proposed CAE, the clutter is separated in the convolutional filtering process and effectively discarded during the learning process. Thus, the deep learning architectures are not sensitive to the affects of ground clutter, caused by furniture (tables and bookshelves) present in the indoor laboratory environment - yet another advantage of learning features in contrast to extracting pre-defined features.

VI. Conclusion

This work shows that a convolutional autoencoder architecture that takes advantage of both unsupervised pre-training as well as the localized spatial features enabled by convolutional filtering outperforms other deep architectures such as CNN, autoencoder, as well as multi-class SVM taking pre-defined features. While the performance gain over other deep learning architectures is 4.1% and 10.1%, for CNN and AE, respectively, the improvement over multi-class SVM is 17.3%. These results clearly show the advantage of deep learning, especially in the cases when the classes are highly similar. Moreover, inspection of the filters in each layer reveals that deep learning architectures can also separate components due to clutter and noise, eliminating the need for explicit ground clutter cancellation or filtering prior to classification. The high classification accuracy of 94.2% for discriminating 12 indoor activity classes involving aided and unaided human motion also shows the potential for using radar-based health monitoring systems for assisted living. Future work will involve estimation of gait parameters when walking is detected for the purposes of fall risk assessment and neuromuscular disease monitoring.

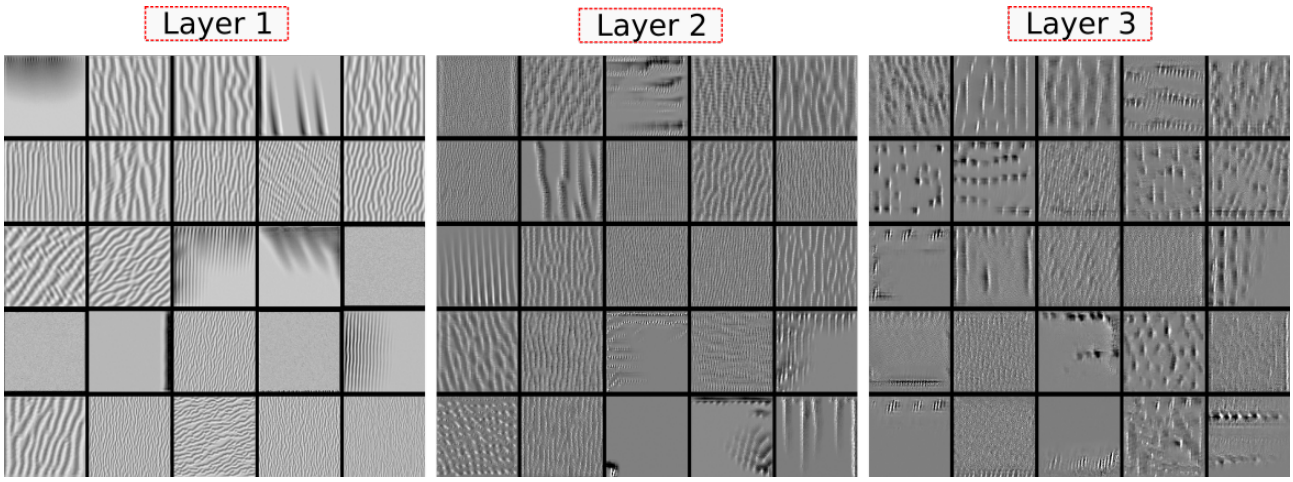


Fig. 10: Visualization of learned filters for CAE model, 25 of 64 filters are shown for each layer with the 9x9 filters in the top 3 rows and the 3x3 filters in the bottom 2 rows.

ACKNOWLEDGMENT

The authors would like to thank Ahmet Serinöz and Ezgi Pelin Altıner for their contributions in creating the dataset.

REFERENCES

[1] S. Z. Gürbüz, Ü. Kaynak, B. Özkan, O. C. Kocaman, F. Kıyıcı, and B. Tekeli, "Design study of a short-range airborne uav radar for human monitoring," in *Signals, Systems and Computers, 2014 48th Asilomar Conference on*. IEEE, 2014, pp. 568–572.

[2] F. Fioranelli, M. Ritchie, and H. Griffiths, "Classification of unarmed/armed personnel using the netrad multistatic radar for micro-doppler and singular value decomposition features," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 9, pp. 1933–1937, 2015.

[3] P. Khomchuk, I. Stainvas, and I. Bilik, "Pedestrian motion direction estimation using simulated automotive mimo radar," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 3, pp. 1132–1145, 2016.

[4] M. G. Amin, Y. D. Zhang, F. Ahmad, and K. D. Ho, "Radar signal processing for elderly fall detection: The future for in-home monitoring," *IEEE Signal Processing Magazine*, vol. 33, no. 2, pp. 71–80, 2016.

[5] C. Garripoli, M. Mercuri, P. Karsmakers, P. J. Soh, G. Crupi, G. A. Vandenbosch, C. Pace, P. Leroux, and D. Schreurs, "Embedded dsp-based telehealth radar system for remote in-door fall detection," *IEEE journal of biomedical and health informatics*, vol. 19, no. 1, pp. 92–101, 2015.

[6] B. Y. Su, K. Ho, M. J. Rantz, and M. Skubic, "Doppler radar fall activity detection using the wavelet transform," *IEEE Transactions on Biomedical Engineering*, vol. 62, no. 3, pp. 865–875, 2015.

[7] Q. Wu, Y. D. Zhang, W. Tao, and M. G. Amin, "Radar-based fall detection based on doppler time–frequency signatures for assisted living," *IET Radar, Sonar & Navigation*, vol. 9, no. 2, pp. 164–172, 2015.

[8] S. Z. Gürbüz, B. Erol, B. Çağlıyan, and B. Tekeli, "Operational assessment and adaptive selection of micro-doppler features," *IET Radar, Sonar & Navigation*, vol. 9, no. 9, pp. 1196–1204, 2015.

[9] Y. Kim, S. Ha, and J. Kwon, "Human detection using doppler radar based on physical characteristics of targets," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 2, pp. 289–293, 2015.

[10] P. Molchanov, J. Astola, K. Egiazarian, and A. Totsky, "Ground moving target classification by using dct coefficients extracted from micro-doppler radar signatures and artificial neuron network," in *Microwaves, Radar and Remote Sensing Symposium (MRRS), 2011*. IEEE, 2011, pp. 173–176.

[11] D. Yessad, A. Amrouche, M. Debyeche, and M. Djeddou, "Micro-doppler classification for ground surveillance radar using speech recognition tools," in *Iberoamerican Congress on Pattern Recognition*. Springer, 2011, pp. 280–287.

[12] R. J. Javier and Y. Kim, "Application of linear predictive coding for human activity classification based on micro-doppler signatures," *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 10, pp. 1831–1834, 2014.

[13] B. Tekeli, S. Z. Gurbuz, and M. Yuksel, "Information-theoretic feature selection for human micro-doppler signature classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 5, pp. 2749–2762, 2016.

[14] S. Z. Gurbuz, B. Tekeli, C. Karabacak, and M. Yuksel, "Feature selection for classification of human micro-doppler," in *Microwaves, Communications, Antennas and Electronics Systems (COMCAS), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1–5.

[15] J. Zabalza, C. Clemente, G. Di Caterina, J. Ren, J. J. Soraghan, and S. Marshall, "Robust pca micro-doppler classification using svm on embedded systems," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, no. 3, pp. 2304–2310, 2014.

[16] Y. Kim and H. Ling, "Human activity classification based on micro-doppler signatures using a support vector machine," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 5, pp. 1328–1337, 2009.

[17] M. S. Seyfioglu, S. Z. Gurbuz, M. Ozbayoglu, and M. Yuksel, "Deep learning of micro-doppler features for aided and unaided gait recognition," *IEEE Radar Conference*, May 2017.

[18] Y. Kim and T. Moon, "Human detection and activity classification based on micro-doppler signatures using deep convolutional neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 1, pp. 8–12, 2016.

[19] B. Jokanovic, M. Amin, and F. Ahmad, "Radar fall motion detection using deep learning," in *Radar Conference (RadarConf), 2016 IEEE*. IEEE, 2016, pp. 1–6.

[20] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 625–660, 2010.

[21] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," *Artificial Neural Networks and Machine Learning–ICANN 2011*, pp. 52–59, 2011.

[22] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *J. Mach. Learn. Res.*, vol. 11, pp. 625–660, Mar. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1756006.1756025

[23] O. E. David and N. S. Netanyahu, "Deeppainter: Painter classification using deep convolutional autoencoders," in *International Conference on Artificial Neural Networks*. Springer, 2016, pp. 20–28.

[24] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[25] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[26] T. D. Bufler and R. M. Narayanan, "Radar classification of indoor targets using support vector machines," *IET Radar, Sonar Navigation*, vol. 10, no. 8, pp. 1468–1476, 2016.
[27] M. Otero, "Application of a continuous wave radar for human gait recognition," in *Defense and Security*. International Society for Optics and Photonics, 2005, pp. 538–548.
[28] S. Björklund, T. Johansson, and H. Petersson, "Evaluation of a micro-doppler classification method on mm-wave data," in *2012 IEEE Radar Conference*, May 2012, pp. 0934–0939.
[29] C. Clemente, L. Pallotta, A. De Maio, J. J. Soraghan, and A. Farina, "A novel algorithm for radar classification based on doppler characteristics exploiting orthogonal pseudo-zernike polynomials," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 1, pp. 417–430, 2015.
[30] J. Zabalza, C. Clemente, and et. al., "Robust pca micro-doppler classification using svm on embedded systems," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, no. 3, pp. 2304–2310, July 2014.
[31] F. Fioranelli, M. Ritchie, S. Z. Gurbuz, and H. Griffiths, "Feature diversity for optimized human micro-doppler classification using multistatic radar," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 53, no. 2, pp. 640–654, April 2017.
[32] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.
[33] M. Ritchie, M. Ash, Q. Chen, and K. Chetty, "Through wall radar classification of human micro-doppler using singular value decomposition analysis," *Sensors (Basel, Switzerland)*, vol. 16, no. 9, 2016.
[34] I. Rosa, A. Marques, and et. al., "Classification success of six machine learning algorithms in radar ornithology," *Ibis International Journal of Avian Science*, vol. 158, no. 1, pp. 28–42, Jan. 2016.
[35] S. Gurbuz, M. Seyfioglu, and F. Liechti, "Identification of biological signals with radar," *Short Term Scientific Mission Report*, Oct. 2015. [Online]. Available: http://www.enram.eu/wp-content/uploads/2014/02/Vogelwarte_Gurbuz_STSM_October2015_REPORT.pdf
[36] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016, pp. 785–794.
[37] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.
[38] Y. Ioannou, D. Robertson, D. Zikic, P. Kontschieder, J. Shotton, M. Brown, and A. Criminisi, "Decision forests, convolutional networks and the models in-between," *arXiv preprint arXiv:1603.01250*, 2016.
[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
[42] R. Trommel, R. Harmanny, L. Cifola, and J. Driessen, "Multi-target human gait classification using deep convolutional neural networks on micro-doppler spectrograms," in *Radar Conference (EuRAD), 2016 European*. IEEE, 2016, pp. 81–84.
[43] Y. Kim and B. Toomajian, "Hand gesture recognition using micro-doppler signatures with convolutional neural network," *IEEE Access*, vol. 4, pp. 7125–7130, 2016.
[44] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.
[45] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
[46] A. Ng, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.
[47] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[48] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems 2, NIPS 1989*. Morgan Kaufmann Publishers, 1990, pp. 396–404.
[49] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
[50] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2018–2025.
[51] F. Chollet, "Keras," https://github.com/fchollet/keras, 2015.
[52] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.
[53] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, vol. 1341, p. 3, 2009.
[54] B. Çağlıyan and S. Z. Gürbüz, "Micro-doppler-based human activity classification using the mote-scale bumblebee radar," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 10, pp. 2135–2139, 2015.

M. Saygın Seyfioğlu received the B.S. and M.S. degrees in electrical and electronics engineering from TOBB University of Economics and Technology (TOBB ETU), Ankara, Turkey, in May 2015 and December 2017, respectively. From May 2015 to December 2016, he was a Teaching Assistant at TOBB ETU for a variety of classes relating to signal processing and communications. He is currently working in STM Defense Technologies Engineering and Trade Inc. as a data scientist. His research interests include deep learning, computer vision, and signal processing for radar and remote sensing.

A. Murat Özbayoğlu graduated from the Department of Electrical Engineering at METU, Ankara, Turkey in 1991, then he got his Msc and PhD degrees from Missouri University of Science and Technology, USA in 1993 and 1996, respectively. After graduation, he joined MEMC Electronics (now became SunEdison), USA as a software project engineer, programmer and analyst. In 2005, he went back to academia by joining the Department of Computer Engineering of TOBB University of Economics and Technology, in Ankara, Turkey. His research interests are machine learning, pattern recognition, deep learning, financial forecasting, computational intelligence, machine vision.

Sevgi Z. Gürbüz (S'01-M'10–SM'17) received the B.S. degree in electrical engineering with minor in mechanical engineering and the M.Eng. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1998 and 2000, respectively, and the Ph.D. degree in electrical and computer engineering from Georgia Institute of Technology, Atlanta, GA, USA, in 2009.

From February 2000 to January 2004, she worked as a Radar Signal Processing Research Engineer with the U.S. Air Force Research Laboratory, Sensors Directorate, Rome, NY, USA. Formerly holding academic positions the TOBB University of Economics and Technology in Ankara, Turkey and Utah State University, Logan, UT, USA, she is currently an Assistant Professor in the University of Alabama at Tuscaloosa, Department of Electrical and Computer Engineering. Her current research interests include radar signal processing, machine learning and pattern recognition, cognitive radar, and remote sensing.

Dr. Gurbuz is a recipient of a USAF Achievement Medal, USAF Commendation Medal, AFRL Technical Achievement Award, C.S. Draper Fellowship, National Defense Science and Engineering Fellowship, EU Marie Curie Research Fellowship, and the 2010 IEEE Radar Conference Best Student Paper Award.