IET Radar, Sonar & Navigation

Research Article

IET Journals
The Institution of Engineering and Technology

Deep CNNs as a method to classify rotating objects based on monostatic RCS

ISSN 1751-8784
Received on 26th September 2018
Revised 18th January 2019
Accepted on 20th February 2019
E-First on 25th April 2019
doi: 10.1049/iet-rsn.2018.5453
www.ietdl.org

Eric Wengrowski[1], Matthew Purri[1] ✉, Kristin Dana[1], Andrew Huston[2]

[1]Department of Electrical and Computer Engineering, Rutgers University, New Brunswick, USA
[2]Radar Group, Lockheed Martin, Moorestown, USA
✉ *E-mail: matthew.purri@rutgers.edu*

Abstract: Radar systems emit a time-varying signal and measure the response of a radar-reflecting surface. In the case of narrowband, monostatic radar signal domain, all spatial information is projected into a radar cross-section (RCS) scalar. The authors address the challenging problem of determining shape class using monostatic RCS estimates collected as a time series from a rotating object tumbling with unknown motion parameters under detectability limitations and signal noise. Previous shape classification methods have relied on image-like synthetic aperture radar or multistatic (multiview) radar configurations with known geometry. Convolutional neural networks (CNNs) have revolutionised learning tasks in the computer vision domain by leveraging images and video rich with high-resolution two-dimensional (2D) or 3D spatial information. They show that a feed-forward CNN can be trained to successfully classify object shape using only noisy monostatic RCS signals with unknown motion. They construct datasets containing over 100,000 simulated RCS signals belonging to different shape classes. They introduce deep neural network architectures that produce 2% classification error on testing data. They also introduce a refinement network that transforms simulated signals to appear more realistic and improve training utility. The results are a pioneering step toward the recognition of more complex targets using narrowband, monostatic radar.

1 Introduction

When illuminated with a narrowband radar signal, an object reflects incident energy and the reflectance depends on the object's geometry and material properties. The amount of energy that is reflected directly back toward the source of illumination is a function of its monostatic radar cross-section (RCS). As an object changes orientation, the RCS changes as well. We wish to classify the 3D shape of objects based only on a time series of monostatic RCS as the object moves according to force-free rigid body motion. Our set of target objects includes right circular cones, right circular cylinders, rectangular planes, spheroids, and trapezoidal prisms. The target object set varies in size with respect to a geometric parameter for each class (e.g. radius and height variation for cylinders). The chosen geometric properties in the test set are selected by radar wavelength so that each object is modelled as a perfect electrical conductor. Labelled data, i.e. RCS of known objects, are required to train and test our supervised classifier. We create a large dataset of geometric objects and their corresponding RCS time-series signals.

To simulate real-world conditions, the input signals for testing are corrupted by Gaussian noise and *Swerling dropout*. The Swerling model [1] is a standard method for determining the detectability of an object based on signal-to-noise ratio (SNR) and waveform characteristics. The instantaneous probability of detecting each object at given time is explicitly included in order to make the performance closer to real-world operation. If the SNR at a given time point is too small, a real-world radar system may be unable to separate the object from the noise and will, therefore, be unable to detect the object and estimate its RCS (Fig. 1).

A subset of the generated signals is used to train a feed-forward convolutional neural network (CNN) classifier. We employ an end-to-end learning architecture, where signal features and the classifier are jointly solved for. The inputs are a series of RCS samples over time as the object rotates through free space. These objects belong to one of four shape families, illustrated in Fig. 2. When the rotation is simple and follows a known path (as shown in Fig. 3, top row), the problem is trivial. However, the problem becomes substantially more difficult when the motion parameters are unknown (see Fig. 3, bottom row).

In this work, we successfully classify the shape family for rotating objects with unknown roll rates, tumble rates, and unknown initial orientations. We train deep neural network classifiers that return the probability of each signal belonging to each shape family. The deep learning training and testing are implemented using PyTorch, a machine learning and optimisation library for the Python programming language [2]. The support vector machine (SVM) and decision tree (DT) algorithms are implemented using the SciPy library for the Python programming

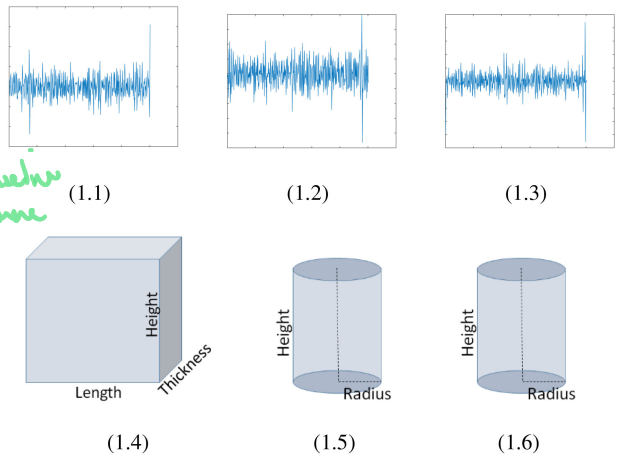

Fig. 1 *Our goal is to correctly predict object shape family from a noisy monostatic RCS signal. RCS is highly sensitive to motion, and the rotation rates and viewing angles are unknown to the classifier. For example, objects may be rotating very fast or very slow about multiple axes. These signals contain added white Gaussian noise and a Swerling detection model, where the probability of detection is smaller for lower RCS values results in missing data points. A CNN is used to learn the separating features that accurately recognise each object class overcoming the challenge of noisy data, missing data, and unknown trajectories*

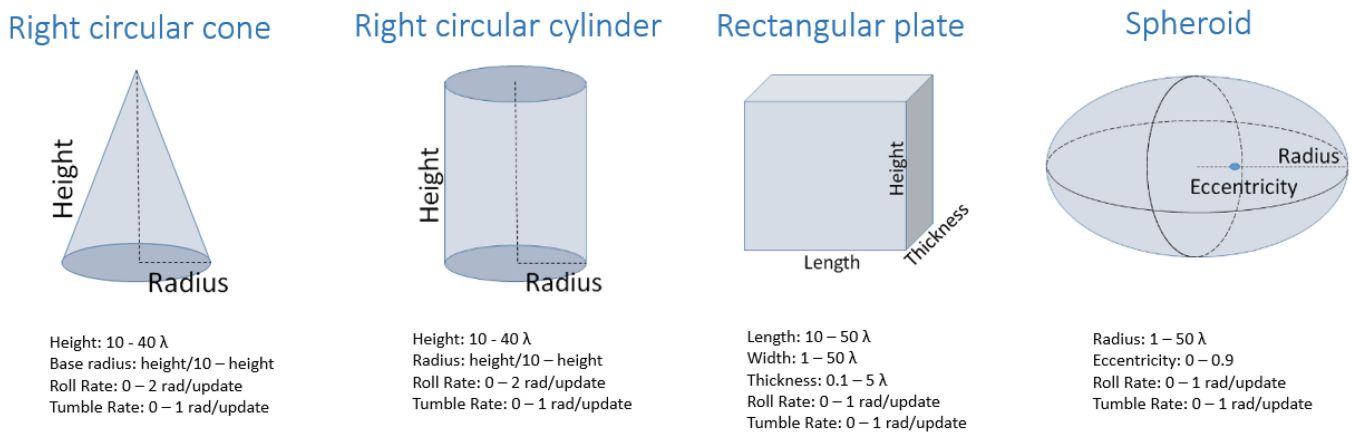

Fig. 2 *Four shape families correspond to four target classes in our classifier. Each shape class has a range of geometric parameters and motion parameters. The parameter ranges are listed under each shape. λ is the wavelength of the incident radar signal*

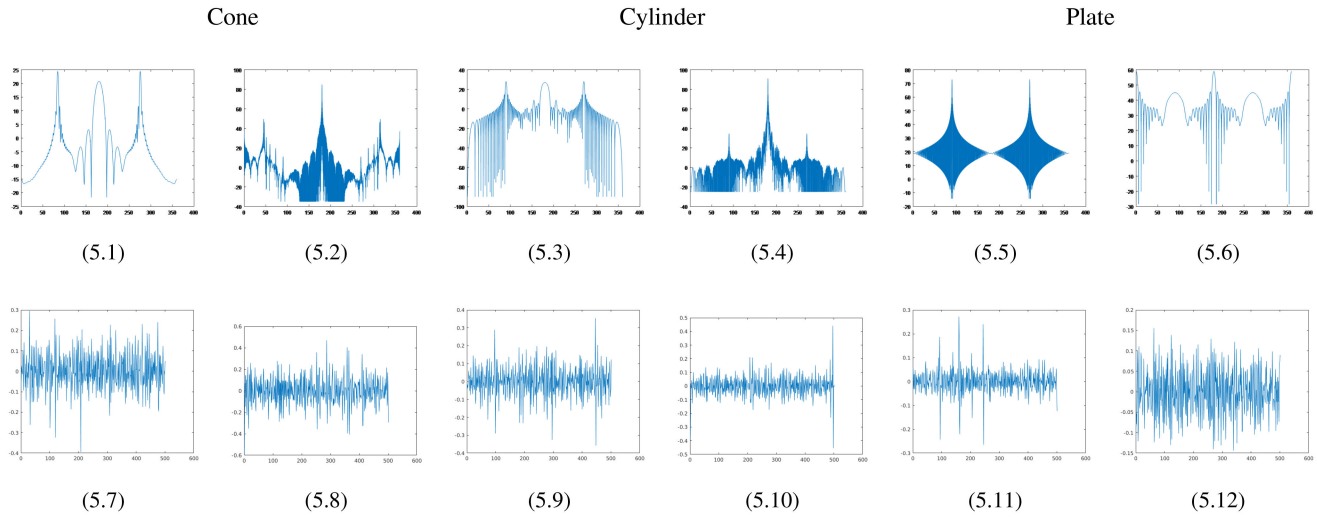

Fig. 3 *There is tremendous variation among the cone, cylinder, and plate RCS signals on the top row. Those signals have rotation about a fixed axis at a relatively slow speed and zero noise. The bottom row features two more realistic cone, cylinder, and plate RCS signals. The salient features present in the top examples are now gone*

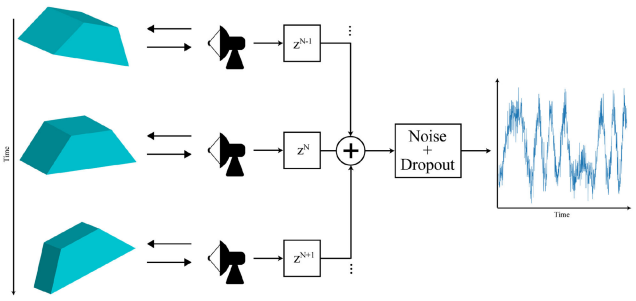

Fig. 4 *POFacets library was used to generate RCS signals from geometric shape models. Generalised Euler motion, additive Gaussian noise, and Swerling 2 dropout are then incorporated to generate the final signal*

language [3]. To our knowledge, our methods are the first application of deep learning for object shape classification using monostatic radar signals (Fig. 4).

2 Related work

Producing an accurate representation of a target object's narrowband monostatic RCS is a challenging problem. Radar specific properties such as wavelength and sampling rate, as well as object-specific properties such as surface material, shape, and motion may dramatically influence the resulting RCS time series. In this application, the objects under investigation are geometrically simple, convex shapes with uniform material construction. The incident energy wave is assumed to be a simple plane wave. The environment is not modelled, except for the addition of Gaussian noise. Owing to these constraints, the physical optics (PO) approximation is appropriate to produce realistic returns. Open-source RCS signal generation tools such as the MATLAB toolbox POFacets are readily available [4] and have been used to approximate RCS of aircraft models [5].

A powerful new class of supervised machine learning algorithms called CNNs leverage optimisation to learn complex latent features for robust classification. This family of algorithms is called *deep learning* when networks contain many convolutional layers. In 2012, a CNN significantly outperformed all other algorithms on the object classification dataset ImageNet [6] and CNNs have become the algorithm of choice for image recognition in computer vision [7–11].

Traditional neural networks have been used for radar classification tasks for decades, often derived from architectures developed for speech recognition such as the time-delay neural network [12, 13]. Early work on neural networks for processing radar signals was applied to identify the number and type of radar emitters in a simulated multisource environment [14]. Pulse-train radar signal classification and source identification remain a topic of active research [15, 16]. Another recent challenge for neural networks in radar is the identification of radar jamming signals [17, 18]. Traditional neural networks have been applied to synthetic aperture radar imagery for ground terrain classification [19] and crop classification [20]; microwave radar for classifying pedestrians and vehicles [21]; Doppler radar for identifying human breathing [22]; ground penetrating radar for the classification of geological structures [23]; and forward scattering radar for identifying very small marine targets [24].

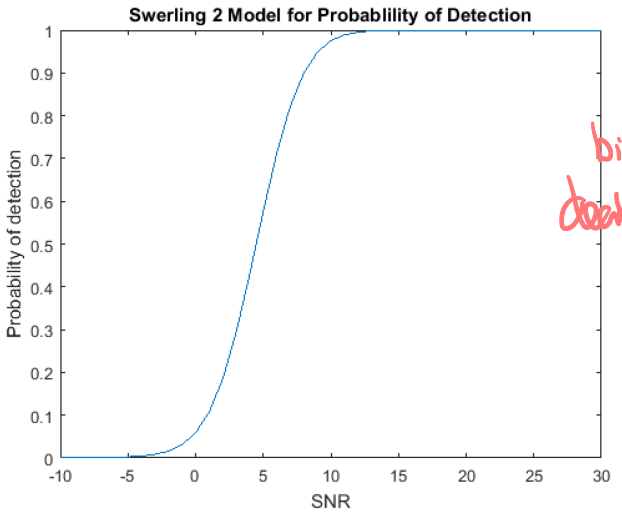

Fig. 5 *Swerling detectability is an important parameter in our model. As the RCS SNR decreased, so does the probability of detection. According to the above graph, SNRs of 25 and 15 dB provide almost no dropped measurements. However, for SNR = 5 dB, the probability of detection drops significantly, to roughly 50%. The RCS measurements with the lowest magnitude have a greater likelihood of being dropped to 0. Although Swerling dropout did have a major effect on our results, it often preserves larger RCS values in the time-series signal, and the larger RCS values are expected to play a more substantial role in feature selection*

While traditional neural networks have been used widely in radar classification tasks, modern deep learning and CNNs are beginning to take hold in recent applications [25–29]. The success of the two-dimensional (2D) CNNs on standard colour images has translated well into radar applications. While most deep learning networks are designed for 2D imagery and can be directly applied to radar-based imagery; however, the RCS time-series signals in our work are 1D signals. In fields such as natural language processing [30] and medical applications [31], 1D CNNs have provided successful classification. In this work, we leverage successful deep networks for 2D image recognition but adapt the networks to the 1D monostatic RCS signals.

Multistatic radar systems utilise a set of receivers and transmitters to create multiple 1D RCS signals of a target object. In prior work, multistatic RCS signals are classified individually using CNNs [25, 32] and the average of multiple CNNs [33] for multistatic contextual target signatures. The monostatic system addressed in our work contains a single collocated receiver–transmitter pair, compared with multistatic systems which have one or more spatially separated receivers and transmitters. The classification problem of monostatic RCS signals is particularly challenging since the signals do not contain contextual information from multiple sources.

3 Generating RCS signals

The first step in RCS classification is generating 3D models of our target objects. The parameters of these objects are listed in Fig. 2. About 128 geometric models were generated, each corresponding to one of four shape classes in the primary experiments. For each of the 3D models, POFacets is used to generate narrowband monostatic RCS values. In the case of monostatic radar, we assume that the radar source and receiver are at the same location. The radar frequency is kept constant. It is important to note that in the PO model, RCS behaviour depends only on the size of the object in wavelengths. Thus, we can arbitrarily set the chosen frequency to 0.3 GHz while preserving the general behaviour of any wavelength. Since the 3D model parameters are scaled by wavelength, this allowed for unit shape size parameters. POFacets is used to generate narrowband monostatic RCS responses, sensitive to object rotation parameterised by $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. The mapping is done by specifying an angular sweep from 0° to 180° at high sampling intervals (0.1°). Symmetry about the shapes allows us to simulate to a maximum rotation of 180°.

Table 1 Generation parameters for A4 and B4 datasets

Parameters	A4	B4
number of classes	4	4
tumble rate, rad/sec×max	0.015, 0.1, 0.5	0.015, 0.1, 0.5, **1**
roll rate, rad/sec×max	0.015, 0.1, 0.5	0.015, 0.1, 0.5, **1**
SNR, dB	25, 15	25, 15, **5**
viewing vector angle, deg	0, 20	0, 20, **40**, **60**
Swerling model	2	2
probability of false alarm	0.0001	0.0001
number of pulses	10	10
signal length (samples)	501	501
number of signals	121,320	363,960

The bold values highlight the additional parameters used for dataset creation.

3.1 Generalised Euler motion

Once an RCS map had been generated, a motion path is drawn over the surface and the map is being interpolated. The target objects are assigned tumble, roll, and initial rotation angle. The initial conditions are then propagated following the physics of rigid body motion in the presence of no external forces (free motion). A quaternion model is used to generate the motion path parameterised by θ and ϕ over the precomputed 2D RCS map. The roll and tumble parameters are bound by the values described in Fig. 2. For each shape class, the centre of mass and moment of inertia are calculated and used for the simulation of realistic, geometry-dependent object motion.

3.2 Randomisations in motion parameters

It would be relatively easy to classify RCS signals from objects at the integer-valued roll, tumble, and viewing angle. To make the problem more realistic and challenging, randomisations were applied to the values of each parameter. A random variable x with $\mu = 1$ and $\sigma = 0.5$ was multiplied with the viewing angle (θ and ϕ), tumble rate, and rotation rate for each signal. The random variation allows for the construction of a database, where the same 2D RCS map could be used to generate multiple signals. The ability to scale motion parameters with random jitter allowed the creation of a nearly equal number of signals between the four classes, even though there were more 3D models created for plates. CNN performance is generally improved when there is an equal number of training examples in all classes.

3.3 Update rate, Swerling, Gaussian noise, gradients, and pyramids

A realistic radar model has a finite update rate. The number of samples as an object rotates is related to the update rate (in hertz) and the rotation rates (in radians/second). In this paper, the kinematic bounds of the objects are defined in radians/update, thus the performance of a highly sampled signal that rotates quickly is the same as for as if it were rotating more slowly with a corresponding decrease in radar update rate. The motion parameters are specified in radians per update. The radar update rate is arbitrarily set to 1 Hz. To simulate realistic distortions of each RCS value, Gaussian noise and a Swerling detectability model are incorporated into each RCS signal. The addition of Gaussian noise transforms the RCS from a truth value to an estimate. The specific parameters can be found in Table 1 (see Fig. 5).

To summarise, the objects under test have complex motion with tumble, roll, and variable viewing angles, yielding complex time series of RCS estimates. The signals are noisy and have missing data points. Each RCS signal dataset contains variable values for each of the aforementioned parameters. Therefore, the same classifier is expected to correctly label RCS signals from objects moving at highly varied speeds in highly varied motion paths with different amounts of noise.

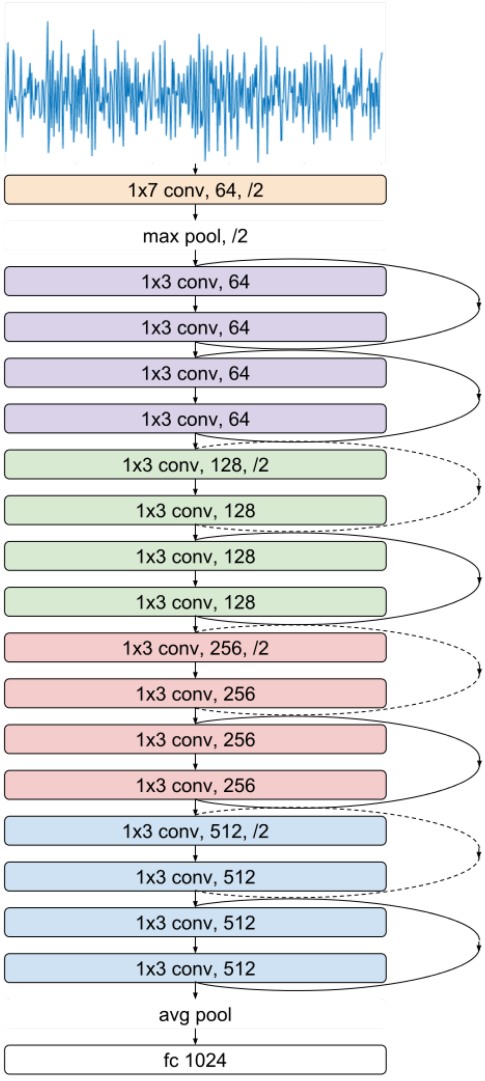

Fig. 6 *18-layer convolutional network is trained to analyse a noisy RCS signal. The architecture is strongly inspired by ResNet [10]. Skip connections are shown as curved arrows. Unlike ResNet, batch normalisation is incorporated into the model*

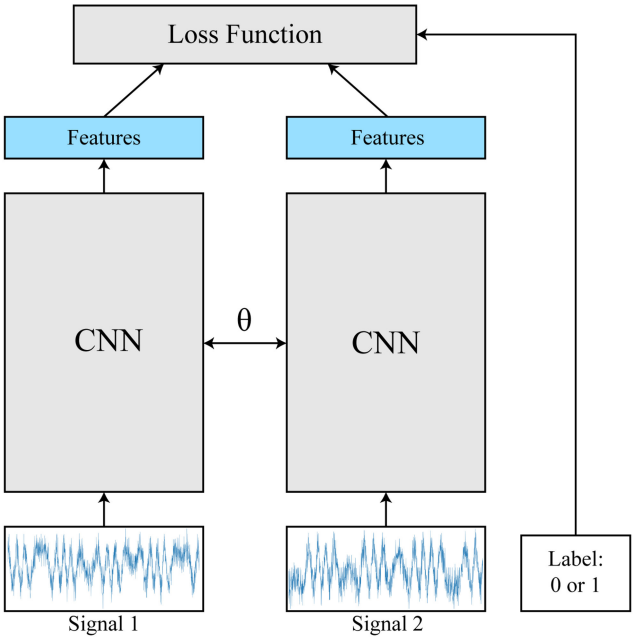

Fig. 7 *Two signals are fed into two CNNs with shared parameters. The output feature vectors are compared via the siamese network loss function 1. The target label is equal to one if the two signals belong to the same class and zero otherwise*

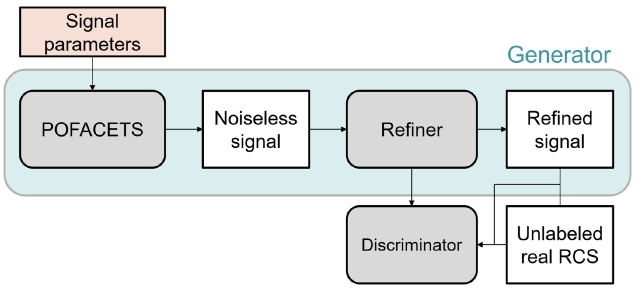

Fig. 8 *Refiner network and the discriminator work in a similar adversarial manner as a GAN. The refiner optimises the simulated signals to look more like the unlabelled realistic data while the discriminator tries to distinguish the difference between the refined and realistic signals*

4 Experiments

Two datasets are created using the methods described. One is used for training and the other for evaluation/testing. The parameters used to create this dataset are listed in Table 1. The datasets in this paper are named A4 and B4 because they both contain four classes but have different parameter values.

All experiments were run on a Ubuntu 16.04 machine with 32 GB of random access memory, a Xeon E5-1620 v4 at 3.5 GHz × 8 central processing unit, a Samsung 860 EVO SSD, and an Nvidia Titan X (Maxwell edition) graphics PU. The PyTorch and SciPy library versions used for training and evaluation are 0.1 and 1.1, respectively.

4.1 Residual network

Our 1D residual network (RN) architecture is inspired by He *et al.* [10]. 2D 3 × 3 convolutional filters were replaced by 1D 3 × 1, 5 × 1, and 7 × 1 filters but the original block module structure and skip connections are maintained. See Fig. 6 for a detailed view of the 18-layer network architecture. The RN was run over 30 epochs and updated using the Adam [34] optimiser with a learning rate of 0.001. Unlike the original implementation of ResNet, batch normalisation is done during training to avoid overfitting. The batch size for training is 128 signals for all models, except for the 152-layer RN due to GPU memory constraints and is instead run with a batch size of 32 signals. The learning rate is decayed by 70% if the current validation accuracy does not improve compared with the average of the previous five validation accuracies. The network with the lowest validation error is saved and used to evaluate the test data. The 18-layer RN requires 5 min to train while the 152-layer RN requires nearly 3 h to train. The time required to evaluate a signal with the listed hardware is on the order of tens of microseconds, allowing real-time signal classification (Figs. 7 and 8).

4.2 Expanding the A4 dataset

In secondary tests, we expand our four class dataset to include a new trapezoidal prism class. We augment the dataset to answer the question of how our model performance would be affected by the addition of a smaller class of signals. This object is selected such that it closely resembles one of the original classes, i.e. the plate class. One trapezoidal prism class model was created. The new dataset distribution is recorded in Table 2. The number of signals for the new class is significantly lower than the other classes. We call this dataset A5 because it contains the same motion parameters as A4 but has an extra shape class.

4.3 Siamese network

Our initial hypothesis was that our RN would misclassify signals belonging to the class with the fewest instances, confusing them with one of the larger classes. If we assume one class will be confused, the loss function will be minimised by misclassifying signals in the smallest class. To test our hypothesis, we compare

Table 2 Number of each respective model in the A5 dataset

	Cone	Cylinder	Plate	Sphere	Trapezoidal prism
train (#)	24,201	25,189	29,041	19,311	2,258
test (#)	1871	2038	2214	1623	254
train, %	24.2	25.2	29.0	19.3	2.3
test, %	23.4	25.4	27.3	20.3	3.2
models	11	12	30	5	1

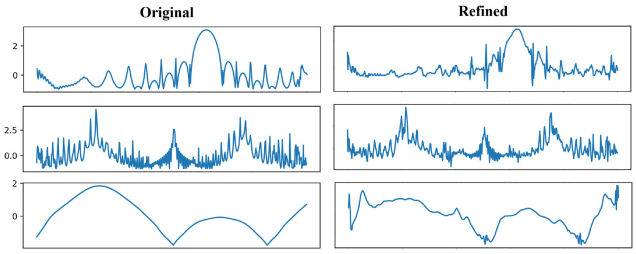

Fig. 9 *Some examples of signals pre- and post-refinement. The structure of the signal is maintained but pseudo noise is added to the original signal from the refiner network*

the performance of the RN with a siamese network. A siamese network consists of two feature extractor modules, each outputting a lower dimensional, compared with the original input, feature vector. The goal of our siamese network is to cluster signals from the same class in close proximity while moving signals from different classes farther apart in feature space. This network is chosen such that the smaller class is less likely to be grouped with another class. The feature extractor modules share the same parameter so that the output vectors can be compared symmetrically. The 18-layer RNs are used as the feature extractors in the siamese architecture. As with our other trained CNNs, the siamese network is trained using the Adam optimiser with batch sizes of 128 signals for 30 epochs. The learning rate was also initialised and adjusted congruently. The comparator or loss function requires a margin hyperparameter to separate signals of different classes

$$L = \sum_{i}^{N} y^i \parallel x_1^i - x_2^i \parallel_2^2 + (1 - y^i) \max(0, m - \parallel x_1^i - x_2^i \parallel_2^2) \quad (1)$$

The loss function encourages signals in feature space synthesised from the same type of model to converge while forcing signals in feature space belonging to different models farther apart. A CNN generates a fixed length feature representation of the input signal from learnt feature extractors. The similarity between feature representations of two signals, x_1 and x_2, is measured with the $L2$ distance metric. The binary label $y = 1$ if the signals are from the same shape primitive model then, and $y = 0$ if the signals are not from the same primitive. Signals from the same shape primitives are forced closer in feature space, whereas signals from different shape primitives are forced apart if the distance between the feature representations is closer than the margin m. Since the network requires two signals, evaluation is computed by measuring the similarity between a test signal and a set of signals from the training dataset. Several methods were attempted as classifiers but ultimately a nearest neighbour (NN) classifier performed with the greatest accuracy. An input signal first passes through the feature extractor network to produce the corresponding test signal feature vector. The test feature vector is compared with a set of training feature vectors. The most similar feature vector to the test feature vector assigns its label to the test vector. Other methods such as a k-NN with $k > 1$ and an SVM were also used but did not perform as well.

4.4 Robustness test

In order for the classifier to be utilised in real-world applications, it must make accurate predictions on signals with previously unseen distortions. Signal distortions such as occlusion, saturation, and clutter can affect monostatic RCS signals. Occlusion, in this work, is defined as zeroing a subset of a signal's RCS values. Clutter is defined as random amplitude spikes at random locations within a signal. Saturation or clipping is a hard cut off at a set threshold that limits a signal's amplitude. Subsampling is the removal of a random contiguous section of a signal. Occlusion differs from subsampling because occluded signals have the same number of samples after the distortion is applied, unlike signal subsampling. As a robustness test, the network is trained on dataset A4 which only contains signals distorted by noise and Swerling dropout. The trained network then evaluates a test set of the A4 dataset that is distorted by one of the previously mentioned distortions. The degree of distortion is varied in each test, e.g. the test signals are saturated to 75% of their maximum amplitude. The residual architecture can receive signals of various dimensions as its input because of an average pooling layer before the end of the feature extractor module. Subsampling is implemented by circularly shifting the signal by a random integer and then setting the last n elements to zero.

4.5 Refiner network

This section is inspired by the work done by Shrivastava *et al.* [35], where the authors train a refiner network to make generated images appear more realistic. This network resembles a generative adversarial network (GAN) [8], where a generating network tries to create 'realistic' data and a discriminator network decides whether the data is real or fake. The generator network iteratively improves the generated image while the discriminator network learns to more accurately discern the real and fake data apart. Instead of generating data from a noise distribution, as with the classic GAN example, a refiner network converts simulated data into data that more resembles the realistic data. In this work, we use a refiner network to make our simulated RCS signals look like simulated signals with added noise. The refiner network maintains the structure of our signal while adding features to make it appear more like the signals with noise. The parameters used for the simulated dataset are similar to A4 dataset, except that no noise is added to the signal and rotation and roll rates are decreased (Fig. 9).

The refiner network is a three-layer CNN that takes a simulated signal as input and outputs a refined signal of the same size. The discriminator network is a five-layer CNN that receives the refined signal as input and outputs a vector probability map. The probability map determines which parts of the input signal appear realistic to the discriminator. The refiner and discriminator networks have separate loss functions and are trained iteratively. The refiner network's loss function is a combination of the distance between the input signal and the generated signal and the likelihood that the discriminator believes that the refined signal is real. The discriminator network's loss function is a combination of the likelihood that the discriminator believes that the refined signal is real and the likelihood that the discriminator is unsure that the real data is real. Both networks are trained for 50 epochs with the Adam optimiser. For each epoch, the refiner network is trained twice while the discriminator is trained only once.

5 Results and discussion

In this section, we explore the performance of our trained CNNs on our generated datasets. We also compare different architecture performances using an augmented dataset, investigate the

robustness of our classifier, and explore improving our simulated data post-generation.

5.1 Classification on A4 and B4 datasets

Several RNs with layer depths as shown in Fig. 10 are trained as described in the experiments section, on both A4 and B4 datasets. Best performance is achieved using the 152-layer RNs, with classification error scores of 2.5 and 2.0% on datasets A4 and B4, respectively, as shown in Fig. 10. While the general trend implies that deeper networks perform better, this is not always true. The 101-layer network performs slightly worst than the 50-layer and 152-layer networks for both the A4 dataset (2.9% versus 3.0% versus 2.5% for A4) and the B4 dataset (2.1% versus 2.2% versus 2.0%). Since all of these networks were trained with the same data, hyperparameters, and appropriately scaled architecture for the given depths, it is difficult to explain this fluctuation in test performance. Test performance saturates for the 18-layer network, and performance changes only slightly for larger networks. As network size increases, so does the ability to learn more complex features. However, larger networks also have a propensity to overfit if the dataset used for training is not sufficiently large and representative of the distribution of each class. When overfitting occurs, training accuracy will continue to improve while test accuracy continues to degrade. Since Fig. 10 features test error, and the A4 and B4 datasets are sufficiently large, the networks are likely not overfitted but at saturation for test accuracy given the complexity of useful signal features. Likely, the small deviations in test performance stem from each network converging on different local minima in the optimisation plane. Initial conditions and when training is stopped may have effects on which minima a network is likely to converge on.

Models trained on the B4 dataset perform better than models trained on the A4 dataset across all network depths. As a baseline, a neural network and non-residual CNN were trained and evaluated on the A4 dataset with the corresponding test errors, 29.5 and 6.1%. The neural network contains six layers, dropout, and non-linear layers. Increasing the number of layers in the neural network did not significantly improve results. The non-residual CNN contained 18 layers and is trained with the same training parameters described in the experiments section. When the number of layers in the non-residual convolutional network was increased, performance plateaued and then began to degrade.

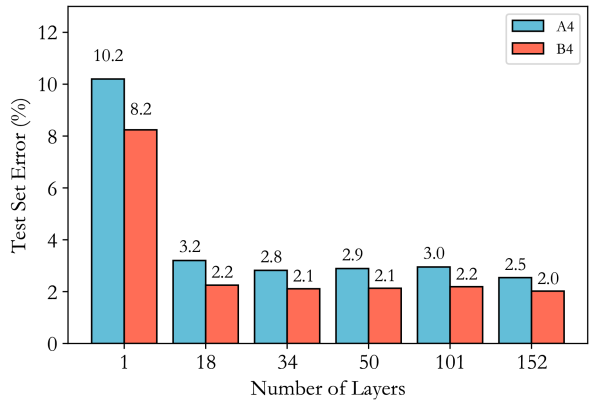

Fig. 10 *Several RNs of different lengths are evaluated on both the A4 and B4 datasets. As the number of layers in the architecture increases, the test error on either dataset decreases but only achieves marginal improvement past a depth of 18 layers*

Our classification results for the RNs may appear counter intuitive at first glance since CNNs typically perform worst on datasets that have more variation. Datasets with more variation are simply more difficult to learn because CNN will have to learn specific filters to deal with that variation. Not only does the B4 dataset contain more signals but it contains faster roll and tumble rates. The faster roll and tumble rates for our signals actually increase the amount of information per sample because the models we use to generate our signals have large distinct edges and smooth surfaces. If instead the models used had rough surfaces and less distinct edges, information would be lost by increasing the roll and tumble rates. The B4 dataset also contains signals with lower SNR rates and more varied viewing angles, which decrease the amount of information within the signals. Regardless of the size of the network, test performance on the B4 dataset was greater than on the A4 dataset. It was for this reason that the A4 dataset was selected to create new datasets and to further train/test our models. If a more difficult dataset is used, then there will be a clearer distinction between the results of more advanced networks.

In addition to neural networks, we assess the performance of other machine learning classification algorithms such as SVMs and DTs on the A4 dataset. The SVM algorithm utilises the radial basis function kernel with a gamma value equal to reciprocal of the number of input features. Multiple one-against-one classifiers are aggregated to form the final SVM classifier. As for the DT, the Gini criterion is used to measure the quality of the split in the tree and decision nodes are randomly chosen to be further split. The minimum number of samples to be a leaf node is set to five, and the minimum number of samples required to split a decision node is two. Unlike deep learning algorithms, features must be manually crafted for the SVM and DT classifiers to attain optimal performance. For comparison, the SVM and DT classifiers are trained and evaluated on the complete length signals from the A4 dataset and achieve accuracies of 55.7 and 48.5%, respectively, as shown in Table 3. Common signal statistics (SS) such as minimum and maximum are combined with low-order cumulants [36] to form a representation of the RCS signals. This representation improves on the previous accuracy of the classifiers to 86.3 and 84.9%. Following the work of Byl *et al.* [37] and Zhang *et al.*[38], more complex descriptive features such as Fourier transform frequency responses and wavelet transform coefficients are used to represent the signals. Specifically, the fast Fourier transform generates frequency coefficients and the discrete wavelet transform (DWT) symmetrically pads signals during the transform in order to avoid inaccurate calculation of the DWT. The first 50 coefficients from each transform are concatenated to form the feature vector representation. This method, which we call transform representations (TRs), is combined with the SS features to achieve accuracies of 74.2 and 82.8%. For reference, our one-layer CNN (1L-CNN) has a test accuracy of 89.8% on the A4 dataset, see Fig. 10.

5.2 Classification on A5 dataset

The siamese network structure has been used on a variety of tasks such as signature matching and facial identification with high performance [39, 40]. This type of network performs most effectively when the number of classes in a dataset is large and the number of data per class is relatively low. The architecture's unique comparator function forces input from the same class to cluster in

Table 3 Accuracy performances of SVM, DT, single-layer CNN, and RN algorithms on the A4 dataset

	SVM	DT	1L-CNN	RN18	RN152
original	0.557	0.485	0.898	0.968	0.975
SS	0.863	0.849	—	—	—
TR	0.707	0.607	—	—	—
TR + SS	0.742	0.828	—	—	—

Leftmost column represents the signal features that were used by each classification algorithm. Common SS represents feature vectors comprised of the mean, standard deviation, and extremum of a signal. TRs represent feature vectors comprised of coefficients from the Fourier and Wavelet transforms of a signal. Since CNNs learn a feature representation, only the original signals are used as input.

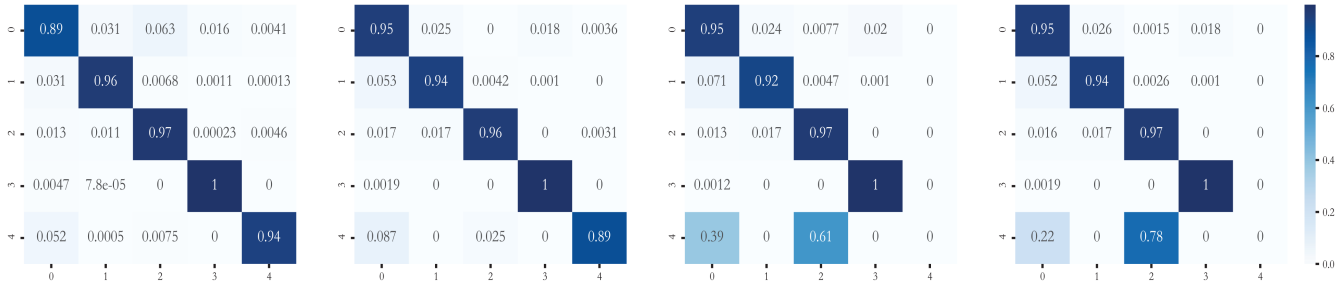

Fig. 11 *Confusion matrices for all siamese networks and the single RN. Confusion matrices starting from the left to the right belong to the single network, the siamese network with the NN, siamese network with k-NN and siamese network with SVM. The classes are enumerated as (0) cone, (1) cylinder, (2) plate, (3) spheroid, and (4) trapezoidal prism*

Table 4 Accuracy performance comparison between a single RN and a siamese network with an NN classifier on the A5 dataset

Class	Single network			Siamese network + NN		
	Precision	Recall	F1-score	Precision	Recall	F1-score
cone	0.94	0.89	0.91	0.92	0.95	0.94
cylinder	0.96	0.96	0.96	0.95	0.94	0.95
plate	0.94	0.97	0.95	0.99	0.96	0.98
spheroid	0.98	1.00	0.99	0.98	1.00	0.99
trapezoidal prism	0.93	0.94	0.93	0.95	0.89	0.92

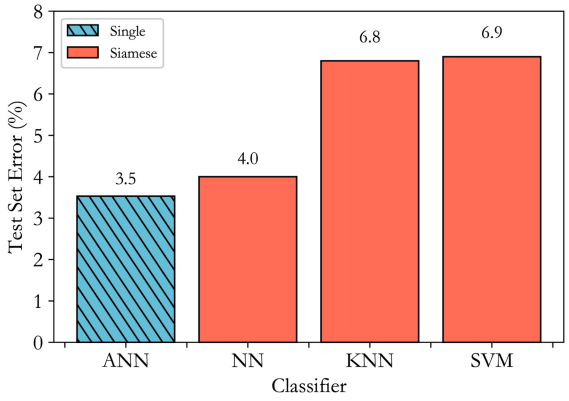

Fig. 12 *Single RN's performance on the A5 test dataset is compared with the performance of three siamese networks with various classification layers. ANN stands for the artificial neural network, NN is the nearest neighbour, k-NN is the k-nearest neighbour and SVM stands for support vector machine*

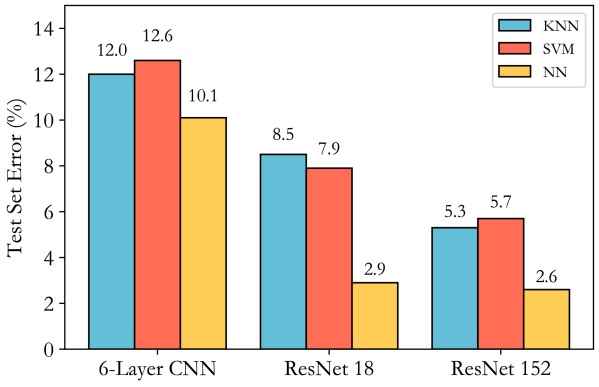

Fig. 13 *Three different classifier modules are compared after a CNN feature extractor of varied depths. The NN classifier achieves the highest overall accuracy consistently across all architectures tested*

high-dimensional space and input from different classes to be farther apart in high-dimensional space. The loss function for a typical CNN classifier is the negative log-likelihood function which does not contain any constraint on how far apart the output vectors of the feature extractor module are. The A5 dataset contains the same set of parameters as A4 but includes an additional geometric model of a trapezoidal prism. The additional class contains only one model and makes up a small portion of the total signals in the A5 dataset (Fig. 11).

The A5 dataset is a superset of the A4 dataset but augmented with an additional and easily confused shape class. The results of this experiment are shown in Table 4. The single RN outperforms all types of siamese networks in terms of overall accuracy as shown in Fig. 12. Initially it appears that the lack of clustering term in the objective function does not reduce performance on the A5 dataset; however, the CNN could maintain high accuracy even while misclassifying all of the signals in the newest class. To further investigate this result the precision, recall, and the F1-score of each class is calculated and shown in Table 4. The siamese networks with the *k*-NN and SVM classifiers misclassified the trapezoidal prism class in every case. The single RN and the siamese network with the NN classifier were both able to correctly classify the trapezoidal prism class a majority of the time.

In Table 4, we can see that the F1-score for the trapezoidal class is greater in the single network section than the siamese network section. Overall the average F1-score across classes is 0.948 and 0.956 for the single network and siamese network, respectively. If we weigh the F1-score by the number of signals per class, there is an even larger difference in performance. The weighted F1-scores of the single network and siamese network are 0.947 and 0.959, respectively. It appears that the single network showed high performance on the trapezoidal prism class because it misclassified more of the signals in the cone class. The siamese network with the NN classifier performs well because the feature extractor module is better able to separate the clusters for each class. Intuitively, we expect the *k*-NN and SVM classifiers to outperform the NN classifier but our results in Fig. 13 suggest otherwise. The dimensionality of the output vector from the feature extractor module may be a potential reason that the NN classifier performs better. As the number of dimensions increase, the *k*-NN algorithm tends to perform worst due to the increasing space in between points.

5.3 Robustness metric performance

A CNN classifier's ability to handle noisy input data can be evaluated in multiple ways such as testing on a novel set of data with distortions seen in the training data or testing on a novel set of data with distortions unseen in the training data. Monostatic radar signals can have a variety of distortions in real applications such as

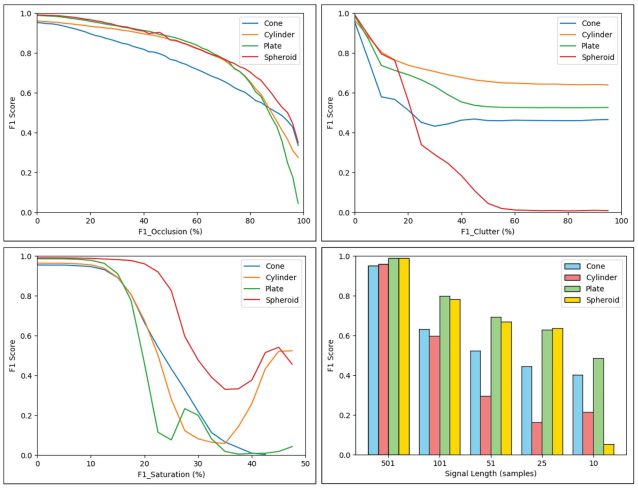

Fig. 14 *Single 18-layered RN's robustness performance is shown for several novel distortions. This benchmark is a way to compare network robustness to realistic signal distortions found RCS systems. Signal occlusion, clutter, saturation, and subsampling are the realistic distortions used for this benchmark*

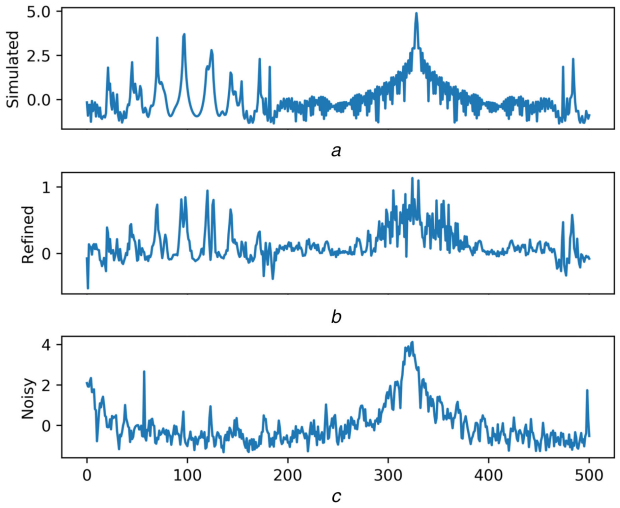

Fig. 15 *Result of the refiner network is shown above. The refiner network takes*

The signal in ***(a)*** as input and returns the signal in ***(b)***. It learns to make this transformation by observing signals with noise like the signal in ***(c)***

signal occlusion, clutter, sensor saturation, subsampling, or a combination of several. Since generating a dataset with every combination of signal distortions is unwieldy, we instead decide to evaluate our system's robustness to distortions by evaluating our model on data with distortions not seen in the training data. The results shown in Fig. 14 are the F1-score per class from a single 18-layer RN. However, the robustness results for networks with more layers are nearly identical and not presented. The evaluation set was generated via the method described in the experiments section.

The network performs remarkably well on signals that have been occluded by even 75% of the total signal, even though no dropout layers are used to train the model. Occlusion may not affect our network significantly because the rotation rates used in our dataset generation are relatively large and occasionally the shape model is rotating several times within the full window of sampling. Even if the signal is occluded significantly, some signals with high rotation rates may contain enough information for classification. However, signals generated with slower rotation rate parameters do not appear to complete rotations multiple times within a full window. For these cases, the CNN is able to discern the object within a limited viewing window. The CNN is, however, very sensitive to signal clutter, accuracy-per-class drops as soon as clutter is introduced. Clutter in this work is the addition of random peaks in a signal and CNNs are sensitive to slight distortions to input data. This distortion is similar to the distortion created by adversarial attacks such as Fast Gradient Sign Attack (FGSM) [41], except that we are adding distortions with random amplitudes at random locations. Most CNNs are not robust to adversarial attacks and it appears that clutter approximates an adversarial attack in this domain. The CNN is resilient to signal saturation up to roughly 15%, then performance decreases significantly soon after. Signals with heavy saturation begin to appear indistinguishable from each other, and the filters that the CNN uses to detect features cannot distinguish between each class. The rise in F1-score of some of the classes seems to be an artefact of the dataset instead of a feature of the network. The final distortion is subsampling the input signal. This measure is similar to the occlusion distortion but the number of total samples in the signal does not change in the occlusion distortion. The results of subsampling show that the CNN can use signals with lengths as small as 25 samples as input and achieve a reasonable F1-score. The performance halves when input size is 5% of its original length. The siamese network evaluated with the robustness metric is not included because the previously mentioned siamese testing method compares an input signal to a subset of the training data. Since the training data does not contain the distortions of the evaluation data, unsurprisingly, the siamese network performs very poorly.

5.4 Classification on the refined dataset

To compare the difference between the simulated dataset and the refined dataset, we train separate three-layered CNNs. The network's performance was evaluated by classifying simulated signals with added white Gaussian noise. The simulated signals with added noise were also used as 'real' data in the refiner network training. Overall the model's performance on the evaluation dataset is greater when the model is trained using the refined dataset by 3.5%. The accuracy of the network trained on the simulated subset A4 dataset is 86.7%, while the accuracy of the network trained on the refined dataset was 90.2%.

No simulator can perfectly model all of the nuances and variables that are required to create real data. Therefore, training a CNN on simulated data typically does not perform well on real data. This does not mean that networks should be trained with only real data because representative real data is difficult and expensive to obtain. Real data is also potentially biased in terms of representing only certain occurrences and typically few variables are able to be controlled when creating real datasets. Simulated data is useful because very large datasets can be generated easily. Adjustments can be made one variable at a time and all parameters used to create that data is known at every time step. The generative CNN called the refiner network described in Section 4 makes simulated data appear more like real data, shown in Fig. 15. Using the refined data to train a small network on a subset of our A4 dataset results in a 3.5% accuracy improvement over training using the equivalent simulated data. For that test, the only 'realistic' feature added to the 'real' data was noise. In Fig. 15, we see that the refined signal seemingly adds noise to the simulated signal but maintains the structural elements of the signal.

6 Conclusion

To the best of our knowledge, we are the first to train CNNs to classify object shape from monostatic radar signals. We expand on the MATLAB library POFacets to generate large datasets with a variety of selected parameters. Realistic motion, added noise, and Swerling dropout enhance the initial simulation generation. Utilising the latest in deep learning architecture we create a 1D RN capable of achieving test error results as low as 2–2.5% on our generated datasets. Our A4 dataset is augmented with an additional test and then evaluated with siamese network architecture. The siamese CNN does perform as well in terms of accuracy but surpasses the performance of the single RN in terms of average F1-score. The robustness of our CNN is then evaluated on signals with previously unseen realistic distortions. The single RN performs well on signals with occlusion and subsampling but performs poorly on signals with clutter and saturation. We explored

increasing the quality of the simulated signals using a state-of-the-art refiner network. Deep learning models trained on the refined signals outperform models trained on the original simulated data.

7 Acknowledgments

This project was funded by the Lockheed Martin. We thank Rowland Escritor and Kevin Vance for their substantial efforts coordinating and facilitating this project. E. Wengrowski and M. Purri are equally contributing co-primary authors.

8 References

[1] Swerling, P.: 'Probability of detection for fluctuating targets', *IRE Trans. Inf. Theory*, 1960, **6**, pp. 269–308

[2] Paszke, A., Gross, S., Chintala, S., *et al.*: 'Automatic differentiation in PyTorch', 2017

[3] Jones, E., Oliphant, T., Peterson, P.: '{Scipy}: open source scientific tools for {Python}', 2014

[4] Chatzigeorgiadis, F.: '*Development of code for a physical optics radar cross section prediction and analysis application*' (Naval Postgraduate School, Monterey, CA, 2004)

[5] Touzopoulos, P., Boviatsis, D., Zikidis, K.C.: '3D modelling of potential targets for the purpose of radar cross section (RCS) prediction: based on 2D images and open source data'. IEEE 2017 Int. Conf. Military Technologies (ICMT), Brno, Czech Republic, 2017, pp. 636–642

[6] Krizhevsky, A., Sutskever, I., Hinton, G.E. 'ImageNet classification with deep convolutional neural networks', Adv. Neural. Inf. Process. Syst., Lake Tahoe, NV, USA, 2012, pp. 1097–1105

[7] Simonyan, K., Zisserman, A.: 'Very deep convolutional networks for large-scale image recognition', 2014, arXiv preprint arXiv:14091556

[8] Goodfellow, I., Pouget-Abadie, J., Mirza, M., *et al.*: 'Generative adversarial nets'. Advances in Neural Information Processing Systems, Montreal, Canada, 2014, pp. 2672–2680

[9] Szegedy, C., Liu, W., Jia, Y., *et al.*: 'Going deeper with convolutions'. Proc. IEEE Conf. Computer Vision and Pattern Recognition, Boston, MA, USA, 2015, pp. 1–9

[10] He, K., Zhang, X., Ren, S., *et al.*: 'Deep residual learning for image recognition'. Proc. IEEE Conf. Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 2016, pp. 770–778

[11] Huang, G., Liu, Z., Weinberger, K.Q., *et al.*: 'Densely connected convolutional networks', 2016, arXiv preprint arXiv:160806993

[12] Kouemou, G.: '*Radar target classification technologies'. Radar technology*' (InTech, Rijeka, Croatia, 2010)

[13] Lang, K.J., Waibel, A.H., Hinton, G.E.: 'A time-delay neural network architecture for isolated word recognition', *Neural Netw.*, 1990, **3**, pp. 23–43

[14] Anderson, J.A., Gately, M.T., Penz, P.A., *et al.*: 'Radar signal categorization using a neural network', *Proc. IEEE*, 1990, **78**, pp. 1646–1657

[15] Jordanov, I., Petrov, N.: 'Sets with incomplete and missing datann radar signal classification'. IEEE 2014 Int. Joint Conf. Neural Networks (IJCNN), Beijing, China, 2014, pp. 218–224

[16] Jordanov, I., Petrov, N., Petrozziello, A.: 'Supervised radar signal classification'. IEEE 2016 Int. Joint Conf. Neural Networks (IJCNN), Vancouver, Canada, 2016, pp. 1464–1471

[17] Soto, A., Mendoza, A., Flores, B.C.: 'Optimization of neural network architecture for classification of radar jamming FM signals'. *Radar Sensor Technology XXI Int. Society for Optics; Photonics*, 2017, **10188**, p. 101881H

[18] Mendoza, A., Soto, A., Flores, B.C.: 'Classification of radar jammer fm signals using a neural network'. *Radar Sensor Technology XXI Int. Society for Optics; Photonics*, 2017, **10188**, p. 101881G

[19] Hara, Y., Atkins, R.G., Yueh, S.H., *et al.*: 'Application of neural networks to radar image classification', *IEEE Trans. Geosci. Remote Sens.*, 1994, **32**, pp. 100–109

[20] Zhang, Y., Wu, L.: 'Crop classification by forward neural network with adaptive chaotic particle swarm optimization', *Sensors*, 2011, **11**, pp. 4721–4743

[21] Park, S., Hwang, J.P., Kim, E., *et al.*: 'A neural network approach to target classification for active safety system using microwave radar', *Expert Syst. Appl.*, 2010, **37**, pp. 2340–2346

[22] Rahman, A., Yavari, E., Lubecke, V.M., *et al.*: 'Noncontact Doppler radar unique identification system using neural network classifier on life signs'. IEEE 2016 IEEE Topical Conf. Biomedical Wireless Technologies, Networks, and Sensing Systems (BioWireleSS), Austin, TX, USA, 2016, pp. 46–48

[23] Szymczyk, P., Szymczyk, M.: 'Classification of geological structure using ground penetrating radar and Laplace transform artificial neural networks', *Neurocomputing*, 2015, **148**, pp. 354–362

[24] Kabakchiev, C., Behar, V., Garvanov, I., *et al.*: 'Detection, parametric imaging and classification of very small marine targets emerged in heavy sea clutter utilizing GPS-based forward scattering radar'. 2014 IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, 2014, pp. 793–797

[25] Lundén, J., Koivunen, V.: 'Deep learning for HRRP-based target recognition in multistatic radar systems'. 2016 IEEE Radar Conf. (RadarConf), Philadelphia, PA, USA, 2016, pp. 1–6

[26] Morgan, D.A.: 'Deep convolutional neural networks for ATR from SAR imagery'. Proc. Algorithms for Synthetic Aperture Radar Imagery XXII, Baltimore, MD, USA, 2015, p. 94750F

[27] Kim, Y., Moon, T.: 'Human detection and activity classification based on microDoppler signatures using deep convolutional neural networks', *IEEE Geosci. Remote Sens. Lett.*, 2016, **13**, pp. 8–12

[28] Gong, M., Zhao, J., Liu, J., *et al.*: 'Change detection in synthetic aperture radar images based on deep neural networks', *IEEE Trans. Neural Netw. Learn. Syst.*, 2016, **27**, pp. 125–138

[29] Mason, E., Yonel, B., Yazici, B.: 'Deep learning for radar'. IEEE Radar Conf. (RadarConf 2017), Seattle, WA, USA, 2017, pp. 1703–1708

[30] Zhang, X., LeCun, Y.: 'Text understanding from scratch', 2015, arXiv preprint arXiv:150201710

[31] Kiranyaz, S., Ince, T., Gabbouj, M.: 'Real-time patient-specific ECG classification by 1-d convolutional neural networks', *IEEE Trans. Biomed. Eng.*, 2016, **63**, pp. 664–675

[32] Stinco, P., Greco, M.S., Gini, F., *et al.*: 'Non-cooperative target recognition in multistatic radar systems', *IET Radar Sonar Navig.*, 2013, **8**, pp. 396–405

[33] Mathews, Z., Quiriconi, L., Böniger, U., *et al.*: 'Learning multi-static contextual target signatures'. IEEE Radar Conf. (RadarConf 2017), Seattle, WA, USA, 2017, pp. 1568–1572

[34] Kingma, D.P., Ba, J.: 'Adam: a method for stochastic optimization', 2014, arXiv preprint arXiv:14126980

[35] Shrivastava, A., Pfister, T., Tuzel, O., *et al.*: 'Learning from simulated and unsupervised images through adversarial training'. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, p. 6

[36] Xin, Z., Ying, W., Bin, Y.: 'Signal classification method based on support vector machine and high-order cumulants', *Wirel. Sens. Netw.*, 2010, **2**, p. 48

[37] Byl, M.F., Demers, J.T., Rietman, E.A.: 'Using a kernel adatron for object classification with RCS data', 2010, arXiv preprint arXiv:10055337

[38] Zhang, L., Zhou, W., Jiao, L.: 'Wavelet support vector machine', *IEEE Trans. Syst. Man Cybern. B, Cybern.*, 2004, **34**, pp. 34–39

[39] Bromley, J., Guyon, I., LeCun, Y., *et al.*: 'Signature verification using a 'siamese' time delay neural network', Adv. Neural. Inf. Process. Syst., Denver, CO, USA, 1994, pp. 737–744

[40] Sun, Y., Wang, X., Tang, X.: 'Deep learning face representation from predicting 10,000 classes'. Proc. IEEE Conf. Computer Vision and Pattern Recognition, Columbus, OH, USA, 2014, pp. 1891–1898

[41] Goodfellow, I.J., Shlens, J., Szegedy, C.: 'Explaining and harnessing adversarial examples', 2014, arXiv preprint arXiv:14126572