Deep Convolutional Neural Networks as a Method to Classify Rotating Objects based on Monostatic Radar Cross Section

Eric Wengrowski\textsuperscript{1,*}, Matthew Purri\textsuperscript{1,*}, Kristin Dana\textsuperscript{1}, and Andrew Huston\textsuperscript{2}
*(equally contributing co-primary authors)

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 scalar Radar Cross Section (RCS) value. We address the challenging problem of determining an object’s shape class using RCS estimates from a rotating object collected as a time series. A monostatic signal is difficult to recognize when the subject is tumbling with unknown motion parameters under detectability limitations and signal noise. Previous shape classification methods have relied on image-like synthetic aperture radar (SAR) or multistatic (multiview) radar configurations with known geometry. Deep learning and convolutional neural networks (CNNs) have revolutionized machine learning and classification tasks in the computer vision domain. But vision tasks usually leverage images and video rich with high-resolution 2D or 3D spatial information. We show that a feed-forward convolutional neural network can be trained to successfully classify an object’s shape using only a noisy monostatic RCS signal and unknown quaternion motion parameters. We construct datasets that contain over 100,000 simulated monostatic RCS signals belonging to different shape classes. We introduce deep neural network architectures that produce 2% classification error on a testing dataset. We also introduce a refinement network that transforms signals generated by a simulator so they appear more realistic and have greater utility in training. To our knowledge, we are the first to apply modern deep learning techniques to target shape classification using noisy monostatic radar signals. The results are a pioneering step toward the recognition of more complex targets using narrowband, monostatic radar.

I. 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 RCS (Radar Cross Section). As an object changes orientation, the RCS changes as well. We wish to classify the 3D shape of objects based 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 family 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 modeled as a Perfect Electrical Conductor (PEC). Since frequency was a free parameter, it was arbitrarily set to 300MHz, so wavelength would be 1 meter, and all geometric parameters would be scaled to 1 meter. Labelled data, i.e. RCS of known objects, are required to train and test our supervised classifier. To generate training and testing data, we use POFacets \cite{2}, a MATLAB tool for simulating RCS with respect to viewing angle. For each object, a 2D RCS reference image map is computed for each 3D object model using POFacets. The parameters of the RCS map are the viewing angles $θ$ and $ϕ$ in standard spherical coordinates. The object is modeled as rigid body rotations using a specified initial orientation, roll rate and tumble rate. Using the orientation of the object in spherical coordinates over time and the reference 2D RCS map, a time series of RCS signals is generated which represents monostatic radar measurements over time. Several unique motion paths are generated for

\textsuperscript{1}Rutgers University, Department of Electrical and Computer Engineering
\textsuperscript{2}Lockheed Martin Radar Group, Moorestown, NJ
Fig. 2: The 4 shape families correspond to 4 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. \( \lambda \) is wavelength of the incident radar signal.

To simulate real-world conditions, the input signals for testing are corrupted by Gaussian noise and Swerling dropout. The Swerling Model [29] is a standard method for determining the detectability of an object based on SNR and waveform characteristics. The instantaneous probability of detecting each object at at given time is explicitly included in order to make the performance closer to real world operation. If the Signal to Noise Ratio (SNR) at a given time point is too small, a real-world radar system may be unable to separate the object from noise and will therefore be unable to detect the object and estimate its RCS.

A subset of the generated signals are used to train a feed-forward convolutional neural network 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 1 of 4 shape families, illustrated in Figure 2. When the rotation is simple and follows a known path (as shown in Figure 4, bottom row), the problem is trivial. However, the problem becomes substantially more difficult when the motion parameters are unknown (see Figure 4, top row).

The neural network classifier returns the probability of each signal belonging to 1 of 4 shape families. The number of RCS time series signals used for training and testing varied by the experiments discussed in Section IV. Experiments with the addition of a fifth class with a small number of training measurements tests the networks ability to add training classes.

In this work, we successfully classify the shape family for rotating objects with unknown roll rates, tumble rates, and unknown initial orientations. The classifier training and testing is implemented in Torch and PyTorch, machine learning libraries for Lua and Python.

II. 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 modeled, except for the addition of Gaussian noise. Due to these constraints, the physical optics (PO) approximation is appropriate to produce realistic returns. Open source RCS signal generation tools such as POFacets are readily available [2]. POFacets is a Matlab toolbox that has been used in many applications. e.g. Touzopoulo's et al. [32] recently created 3D models of aircraft from images and then generated an RCS signal of those aircraft with POFACETS.

A powerful new class of supervised machine learning algorithms called convolutional neural networks (CNNs) leverage optimization 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 convolutional neural network significantly outperformed all other algorithms on the object classification dataset ImageNet [16] and CNNs have become the algorithm of choice for image recognition in computer vision [26], [4], [30], [7], [8].

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 [15], [17]. Early work on neural networks for processing radar signals were applied to identifying the number and type of radar emitters in a simulated multisource environment [1]. Pulse-train radar signal classification and source identification remains a topic of active research [9], [10]. Another recent challenge for neural networks in radar is the identification of radar jamming signals [27], [21]. Traditional neural networks have been applied to: SAR imagery for ground terrain classification [6] and crop classification [34]; microwave radar for classifying pedestrians and ve-
Multi-static radar systems utilize 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 [18], [28] and the average of multiple CNNs [20] for multistatic contextual target signatures. The monostatic system addressed in our work contains a single collocated receiver-transmitter pair, compared to multistatic systems which have one or more spatially separated receivers and transmitters. The classification problem of monostatic systems which have one or more spatially separated receivers and transmitters is particularly challenging since the signals do not contain contextual information from multiple sources.

III. 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 Figure 2. All models are created using POFacets, the Matlab library for radar simulation. POFacets is used to generate 128 geometric models, each corresponding to 1 of 4 shape classes in the primary experiments. For each geometric model, a monostatic, fixed-frequency 2D RCS map is created. The 2D RCS maps are parameterized by viewing angles $\theta$ and $\phi$ in standard spherical coordinates. Each of the 3D models were used for RCS signal generation. Classes with more static parameters, like plate, had more 3D models.

Once the 3D models have been created, 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 physical optics model, RCS behavior depends only on the size of the object in wavelengths. Thus we can arbitrarily set the chosen frequency to 0.3GHz while preserving the general behavior of any wavelength. Since the 3D model parameters are scaled by wavelength, this allowed for unit shape size parameters. POFacets is used to generate a 2D map of RCS response versus viewing angle $\theta$ and viewing angle $\phi$. The mapping is done by specifying an angular sweep from $0^\circ$ to $180^\circ$ at high sampling intervals (0.1$^\circ$). Symmetry about the shapes allows us to simulate to a maximum rotation of $180^\circ$.

Even for simple rotation about a fixed axis, different sets of geometric parameters, such as for the 3D cone models, produce substantially different RCS responses. The diversity of signals makes the classification task challenging, but CNNs are particularly suited for achieving a wide range of invariance over input signals. In feature-space, the distance between each of these diverse signals should be smaller than the distance between every model in each other class.

A. Generalized Euler Motion

Once an RCS map had been generated, a motion path is drawn over the surface and the map is be interpolated. The target objects are assigned an initial tumble, roll, and viewing 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 parameterized by $\theta$ and $\phi$ over the precomputed 2D RCS map. The roll and tumble parameters are bound by the values described in Figure 2.

For each shape class, the center of mass and moment of inertia are calculated and used for the simulation of realistic, geometry-dependent object motion.

B. Randomizations in Motion Parameters

It would be relatively easy to classify RCS signals from objects at integer-valued roll, tumble, and viewing angle. To make the problem more realistic and challenging, randomizations 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 ($\theta$ and $\phi$), 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 a nearly equal number of signals between the 4...
classes, even though there were more 3D models created for plates. CNN performance is generally improved when there are equal number of training examples in all classes.

C. 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 are related to the update rate (in Hz) and the rotation rates (in radians/second). In this study 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 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 Figure 7.

To summarize, 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.

IV. EXPERIMENTS

We analyze and expand upon our generated datasets by using the datasets to train and evaluate deep learning architectures. Two datasets were created according to the previous methodology. The parameters used to create these dataset are listed in Figure 7. The datasets in this paper are named A4 and B4 respectively because they both contain four classes but have different parameter values.

A. Residual Network

Our 1-dimensional residual network architecture is inspired from He et al. [7]. Two-dimensional $3 \times 3$ convolutional filters were replaced by 1-dimensional $3 \times 1$, $5 \times 1$, and $7 \times 1$ filters, but the original block module structure and skip connections are maintained. See Figure 6 for a detailed view of the 18-layer network architecture. The residual

Fig. 4: 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 2 more realistic cone, cylinder, and plate RCS signals. The salient features present in the top examples are now gone.

Fig. 5: Swerling detectability was an important parameter in our model. As the RCS SNR decreased, so did the probability of detection. According to the above graph, SNRs of 25 and 15dB provide almost no dropped measurements. But for SNR = 5dB, the probability of detection drop 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.
network was run over 30 epochs and updated using the Adam [13] optimizer with a learning rate of 0.001. Unlike the original implementation of ResNet, batch normalization is done during training to avoid overfitting. The batch size was 128 signals for all models except for the 152 layered residual network due to GPU memory constraints and was instead run with a batch size of 32 signals. The learning rate decayed by 70% if the current validation accuracy did not improve compared to the average of the previous five validation accuracies. The network with the lowest validation error was used to evaluate the test data.

B. 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 resembled one of the original classes, i.e. the plate class. Only one model for the trapezoidal prism class was created. The new dataset distribution is shown in Figure 8. The number of signals for the new class is significantly lower than the other classes.

C. Siamese Network

Our initial hypothesis was that our residual network 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 minimized by misclassifying signals in the smallest class. In order to test our hypothesis, we compare the performance of the residual network with a siamese network. A siamese network consists of two feature extractor modules, each outputting a lower dimensional, compared to the original input, feature vector. Siamese networks work by clustering 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. Residual networks were used as the feature extractors in the siamese architecture. The network was trained with similar parameters such as the Adam optimizer and using batch sizes of 128 for 30 epochs. The siamese networks used were made up of 18 layered
Fig. 9: Two signals are fed into two CNNs with shared parameters. The output feature vectors are compared via the Siamese network loss function. The target label is one if the two signals belong to the same class and zero otherwise.

The function compares the L2 distance between the two feature vectors $x_1$ and $x_2$ of the two input signals. If the binary label $y$ is equal to one then the L2 distance is considered but if $y$ is equal to zero then the loss function increases if the signals are not farther than the margin parameter $m$. Since the network requires two signals, evaluation is based on measuring the similarity between a set of signals belonging to different classes. Several methods were attempted as classifiers but ultimately a nearest neighbor 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 to 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-nearest neighbor with k greater than one and a support vector machine were also used but did not perform as well.

D. Robustness Test

In order for the classifier to be utilized 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 residual networks. The learning rate was also initialized and adjusted similarly. The comparator or loss function requires a margin hyperparameter to separate signals of different classes:

$$L = \sum \limits_{i} y^i \ast \|x^i_1 - x^i_2\|^2_2 + (1 - y^i) \ast \max(m, \|x^i_1 - x^i_2\|^2_2)$$  \hspace{1cm} (1)

The function compares the L2 distance between the two feature vectors $x_1$ and $x_2$ of the two input signals. If the binary label $y$ is equal to one then the L2 distance is considered but if $y$ is equal to zero then the loss function increases if the signals are not farther than the margin parameter $m$. Since the network requires two signals, evaluation is based on measuring the similarity between a set of signals belonging to different classes. Several methods were attempted as classifiers but ultimately a nearest neighbor 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 to 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-nearest neighbor with k greater than one and a support vector machine were also used but did not perform as well.

E. Refiner Network

This section is heavily inspired by the work done by Shrivastava et al. [25], where the authors train a refiner network to make generated images appear more realistic. This network resembles a generative adversarial network [4] 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 in a GAN network, 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 signals from POFACETS look like simulated signals with added noise. The refiner network maintains the structure of our signal 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 cutoff 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 1-dimensional residual networks performance is evaluated on signals with these unseen distortions. The network is trained on dataset A4 which only contains signals distorted by noise and Swerling dropout. The test set of A4 is distorted by one of the previously mentioned distortions and then evaluated with the residual network. 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 while the first $n$ RCS values of the original 501 RCS samples remain the same.
while adding features to make it appear more like the signals with noise. The parameters used for the simulated dataset were similar to A4 dataset except that no noise was added to the signal and rotation and roll rates were decreased.

The refiner network is a three layered convolutional neural network that takes a simulated signal as input and outputs a signal of the same size. The discriminator network is a 5 layered 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 compares the distance between the input signal and the generated signal and additionally 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 and with the Adam optimizer. For each epoch the refiner network is trained twice while the discriminator is only trained once.

V. RESULTS & DISCUSSION

In this section we explore the performance of the convolutional neural network algorithm on our generated datasets. We also compare different architectures on an augmented dataset, investigate the robustness of our classifier, and explore improving our simulated data post-generation.

To our knowledge, our methods are the first application of deep learning for monostatic radar signal classification. Monostatic signals are generated from only one transmitter and receiver pair as opposed to multistatic signals which are generated from more transmitter and receiver pairs. Classifying monostatic signals is thus more difficult because only a single viewing angle is considered and no contextual information is included. As in many fields CNNs have been used to great effect improving the performance on nearly every dataset.

A. Classification on A4 and B4 Datasets

Several 1-dimensional residual networks are trained at various depths as described in the experiments section, on both dataset A4 and B4. As network depth is increased, classification error decreases to 2.5% and 2.0% on datasets A4 and B4 respectively, as shown in Figure 12. Although dataset B4 contains a larger variety of parameters it also contains more training examples. 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 convolutional neural network were trained and evaluated on the A4 dataset with the corresponding test errors, 29.5% and 6.1%. The neural network contains 6 layers, dropout, and non-linear layers. Increasing the number of layers in the neural network did not significantly improve results. The non-residual convolutional neural network 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 the performance plateaued and then began to degrade.

Our classification results for the 1D residual networks seem counter-intuitive at first glance, since CNNs typically perform worse on datasets that have more variation. Datasets with more variation are simply more difficult to learn because the 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 increases the amount of information per sample because the models we use to generate our signals have large distinct edges and smooth surfaces. If the models used instead 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.

![Fig. 11: 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.](image)

![Fig. 12: The Residual network was evaluated on both dataset A4 and B4. As the number of layers in the architecture increases the test error on either dataset decreases but only achieves marginal improvement past the 18 layer depth.](image)
Fig. 13: Three different classifier modules are compared after a CNN feature extractor of varied depths. The nearest neighbor classifier performed with the highest overall accuracy across all architectures tested.

Fig. 14: The single residual network’s robustness performance is shown for several novel distortions. This benchmark was a way to compare the network robustness to realistic signal distortion. Signal occlusion, clutter, saturation, and subsampling were the realistic distortions used for this benchmark.

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 the single CNN versus the siamese network.

B. Classification on A5 Dataset

The A5 dataset contains the same set of parameters as A4 but includes an additional geometric model of trapezoidal prism. The additional class contains only one model and makes up a small portion of the total signals in the A5 dataset. The single residual network outperforms the siamese network in terms of overall accuracy as shown in Figure 16. The nearest neighbor classifier performed significantly better compared to the k-nearest neighbor and support vector ma-

chine classifiers. From the confusion matrices in Figure 15 the single network classifies the trapezoidal prism class more accurately than all siamese network architectures. The siamese network with a nearest neighbor classifier correctly predicts the trapezoidal class nearly as frequently as the single network. The k-nearest neighbor and support vector machine classifiers however, consistently misclassify the trapezoidal prism class. The single network did not perform well in terms of accuracy for the cone class. The precision, recall, and F1-scores for the single network and siamese network with a nearest neighbor classifier is shown in Table I. The single network performs worst on the cone class and best on the spheroid class in terms of F1-scores. The siamese network performs worst on the trapezoidal prism class. The average F1-score scores for the single network and the siamese network were 0.948 and 0.956 respectively.

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

If the A4 dataset is augmented with a class that is unique but similar to one of the other classes in the dataset, we would expect that the CNN would be more likely to misclassify the novel class. The results of this experiment are shown in Table I. The single residual network outperforms all types of the siamese networks in terms of overall accuracy as shown in Figure 16. 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 I. The siamese networks with the k-nearest neighbor and support vector machine classifiers misclassified the trapezoidal prism class in every case. The single residual network and the siamese network with the nearest neighbor classifier were both able to correctly classify the trapezoidal prism class a majority of the time. It was surprising that the single residual network performed better on the new class than the siamese network.

In Table I 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-score of the single network and siamese network are 0.947 and 0.959 respectively. It appears that the single network showed high performance
Fig. 15: The confusion matrices for all siamese networks and the single residual network. Confusion matrices belong to the (a) single network, (b) the siamese network with nearest neighbor, (c) siamese network with k nearest neighbor, and (d) siamese network with support vector machine. The classes are enumerated as (0) cone, (1) cylinder, (2) plate, (3) spheroid, and (4) trapezoidal prism.

on the trapezoidal prism class because it misclassified more of the signals in the cone class. The siamese network with the nearest neighbor classifier performs well because the feature extractor module is more able to separate the clusters for each class. Intuitively we would expect the k nearest neighbor and support vector machine classifiers to outperform the nearest neighbor classifier. A potential reason that the nearest neighbor classifier performed better, may be a result of the dimensionality of the output vector from the feature extractor module. As the number of dimensions increase, the k nearest neighbor algorithm tends to perform worse due to the increasing space in between points.

C. Robustness Metric Performance

The robustness metric is designed as a means of studying how a convolutional neural network would perform on unseen data corrupted by unknown distortions. The four types of distortions were signal occlusion, clutter, saturation, and subsampling. The network was trained on the A4 dataset which does not contain any of the four distortions. The robustness results for the 18-layered residual network are shown in Figure 14. The amount of the signal occluded varies from 0% to 99%. At 0% distortion none of the signal is distorted while at 99% distortion only 1% of the signal is non-zero. The overall F1-score of the 18-layered residual network is above 0.8 at 50% signal occlusion. The steepest degradation in performance is around 85% signal occlusion. Clutter immediately begins to degrade the performance of the network. However after roughly 50% signal clutter the performance of the network plateaus for each class. The networks performance on saturated signals is essentially unaffected before 13% saturation. The average F1-score drops most significantly at 20% saturation and performance improves at 40% saturation before falling. The final distortion is the removal of signal samples from the original 501 sample length to 101, 51, 25, and 10 signal lengths. When the signals are a fifth of the original number of samples the average F1-score decreases to 0.7. Even when the test signal length is 25, the network achieves an average F1-score of roughly 0.4. At a test signal length of 10 samples the average F1-score is essentially random.

A CNN classifiers 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 signals can have a variety of distortions such as signal occlusion, clutter, sensor saturation, subsampling, etc. or a combination of several. Since generating a dataset with every combination of signal distortions is unwieldy we instead decide to evaluate our systems robustness to distortions by evaluating our model on data with distortions not seen in the training data. The results shown in Figure 14 are from the single 1-dimensional 18 layered residual network however the results for the same network with more layers is nearly identical. The evaluation set was generated via the method described in the experiments section. Surprisingly,
TABLE I: Computed precision, recall, F1-scores for the single residual network and the siamese network with the nearest neighbor classifier. The average and weighted F1-score of the siamese network is larger than the single network.

<table>
<thead>
<tr>
<th>Class</th>
<th>Single Network</th>
<th>Siamese Network+NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cone</td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td></td>
<td>0.94</td>
<td>0.89</td>
</tr>
<tr>
<td>Cylinder</td>
<td>0.96</td>
<td>0.96</td>
</tr>
<tr>
<td>Plate</td>
<td>0.94</td>
<td>0.97</td>
</tr>
<tr>
<td>Spheroid</td>
<td>0.98</td>
<td>1.00</td>
</tr>
<tr>
<td>Trapezoidal Prism</td>
<td>0.93</td>
<td>0.94</td>
</tr>
</tbody>
</table>

Fig. 16: The A5 dataset is trained and evaluated on a single Residual network and compared to three residual Siamese networks with various classifier modules. All models were given the same number of epochs to run but the training model with the highest validation accuracy was used for testing. ANN stands for artificial neural network, NN is nearest neighbor, kNN is k nearest neighbor, and SVM stands for support vector machine.

our network performs remarkably well on signals that have been occluded by even 75%. Occlusion may not affect our network significantly because the rotation rates used in our dataset generation are relatively large and sometimes the 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 FGSM [5], 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. 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 artifact 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 of samples in the signal do 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 relatively good 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 current 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, our network performs very poorly.

D. Classification on Refined Dataset

To compare the difference between the simulated dataset and the refined dataset we train separate three layered convolutional neural networks. The network’s performance was evaluated by classifying simulated signals with added white 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 only representing certain occurrences and typically few variables are able to be controlled when creating real datasets. Simulated data is useful because you can generate very large datasets easily, can make adjustments on one variable at a time, and know the values of all parameters used to create that data. The generative CNN called the refiner network described in the experiments section has been shown to make simulated data appear more like real data. 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...
outperform models trained on the original simulated data. Quality of the simulated signals using a state of the art refiner with clutter and saturation. We explored increasing the occlusion and subsampling but performs poorly on signals. The single residual network performs well on signals with realistic distortions. Evaluated on signals it previously unseen realistic distortions. The robustness of our CNN is then tested the only “realistic” feature added to the “real” data was noise. In Figure 17 we see that the refined signal seemingly adds noise to the simulated signal but maintains the structural elements of the signal.

VI. CONCLUSION

To the best of our knowledge, we are the first to train convolutional neural networks to classify object shape from monostatic radar signals. We expand upon the POFacets MATLAB package to generate large datasets with a variety of selected parameters. Realistic motion, added noise, and Swerling dropout enhance the initial simulation generation. Utilizing the latest in deep learning architecture we create a 1D residual network 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 a siamese network architecture. The siamese CNN does perform as well in terms of accuracy but surpasses the performance of the single residual network in terms of average F1-score. The robustness of our CNN is then evaluated on signals it previously unseen realistic distortions. The single residual network 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.

ACKNOWLEDGMENT

This project was funded by Lockheed Martin. We would like to thank Rowland Escorir and Kevin Vance for their substantial efforts coordinating and facilitating this project.

REFERENCES


