Training Convolutional Neural Networks for Translational Invariance on SAR ATR

Malmgren-Hansen, David; Engholm, Rasmus ; Østergaard Pedersen, Morten

Published in:
Proceedings of EUSAR 2016: 11th European Conference on Synthetic Aperture Radar

Publication date:
2016

Document Version
Peer reviewed version

Link back to DTU Orbit

Citation (APA):
Malmgren-Hansen, D., Engholm, R., & Østergaard Pedersen, M. (2016). Training Convolutional Neural Networks for Translational Invariance on SAR ATR. In Proceedings of EUSAR 2016: 11th European Conference on Synthetic Aperture Radar (pp. 459-462). IEEE.

Training Convolutional Neural Networks for Translational Invariance on SAR ATR

David Malmgren-Hansen, Technical University of Denmark, dmal@dtu.dk, Denmark
Rasmus Engholm, Terma A/S, rae@terma.com, Denmark
Morten Østergaard Pedersen, Terma A/S, mdp@terma.com, Denmark

Abstract

In this paper we present a comparison of the robustness of Convolutional Neural Networks (CNN) to other classifiers in the presence of uncertainty of the objects localization in SAR image. We present a framework for simulating simple SAR images, translating the object of interest systematically and testing the classification performance. Our results show that where other classification methods are very sensitive to even small translations, CNN is quite robust to translational variance, making it much more useful in relation to Automatic Target Recognition (ATR) in a real life context.

1 Introduction

Compared to classification studies on natural images, Synthetic Aperture Radar (SAR) datasets are often smaller and contain more biases. One of very few public available SAR Automatic Target Recognition (ATR) datasets is the Moving and Stationary Target Acquisition and Recognition (MSTAR) that was collected during the 1990s. MSTAR contains several subsets with different tasks to solve. One of these is the 10 target classification task where 10 specific vehicles must be recognized. Although it is an interesting problem this does not contain much class variance. A more realistic scenario could be how well general target classes like "tanks" that might contain many different instances can be recognized from other classes of vehicles like "cars" or "busses".

The targets in MSTAR are densely sampled along azimuthal object rotation, but far most images are taken at 15° and 17° depression angles with almost no background variation.

Recent advances in Convolutional Neural Networks (CNNs) for computer vision problems makes it interesting to study the benefits of these algorithms on SAR problems as well. This has been done for the MSTAR dataset by Morgan in [3]. In [3] a CNN classifier is applied to the 10 target classification task from MSTAR and it achieve results similar to other state of the art methods benchmarked on this problem, e.g. by [4], [5], [7] and [8]. Due to the sparsity of object representation in the data, there are more investigations to be made in order to find the best classification models.

In this article we first show how to generate a simulated dataset where the amount of translational variance is bigger than for the MSTAR dataset (see Section 2). We then describe the design of the CNN used as well as the classification algorithms for comparison (see Section 3). Finally we show the results of the comparison (Section 4) and conclude that the convolution layers in a CNN can make it more invariant to translational variance, and superior to other machine learning classification algorithms (Section 5).

2 Data Simulation

For our SAR image simulations, the geometry consists of a CAD model placed on a corrugated surface representing the terrain. The surfaces are generated by selecting points in an ground grid (x-y plane) and then picking the vertical displacement (the z component) from a normal distribution with a given variance. The ground planes vertical displacement is then low-pass filtered to give a smooth and more natural terrain.

A highly simplified optical algorithm is used to trace multiple rays and find the intersection points between the incident signal and the model and/or terrain. The cell value in the image corresponds to the range / cross-range value of the intersection point. Here a simple diffuse scattering contribution is found from the angle between the surface normal at the intersect point and the incident ray direction. Based on whether a ray intersects with the model or terrain, the value of the cell is scaled with a scattering cross section representing the material (metal for the target and soil for terrain).

Parameter values for the simulation model, such as material reflection coefficients and background variation, have been adjusted according to images from the MSTAR dataset. This is done by manually defining boundaries in an MSTAR image for background and target, and then analyse the values from each region. The relationship between the median from the two sets of values was used to define the relation between the reflection coefficients for terrain and vehicle model in the simulation. The terrain topology variation in our simulation is adjusted so histograms from background pixels in a MSTAR image and in a simulated image is most alike.

By adjusting the parameters in this way, we get a simulated image that has a visual appearance close to the real

SAR images, despite using a simple simulation procedure. In **Figure 1** an image from the MSTAR dataset can be seen together with a simulated image from our dataset.

Our simulation tool is implemented in python and GPU accelerated with the VTK toolset. It takes approximately 2 minutes to simulate an image of 192x192 pixel resolution.

(a) MSTAR image of a T62 tank. **(b)** Simulated SAR image of a leopard tank.

Figure 1: Comparison of an MSTAR image and a similar vehicle modelled with our SAR simulation algorithm.

2.1 Dataset

Our dataset is constructed from 14 CAD models from 7 different vehicle classes, so each class has two instances. When simulating the images, a background is chosen between 100 different generated examples and rotated randomly. Then a model is placed with a given rotation and images from seven different depression angles are generated. A shadow mapping technique is used to remove points not visible from a the radar position so these does not contribute to the simulation. We sample with 20°steps of azimuthal object rotation which gives us 18 images per vehicle per depression angle. This is in total 1764 images of 192x192 pixel resolution and to look most like MSTAR data our pixel spacing is 0.2 meter and resolution 0.3 meter.

CAD model examples from each target class are shown in **Figure 2** along with a simulated SAR image of the given model.

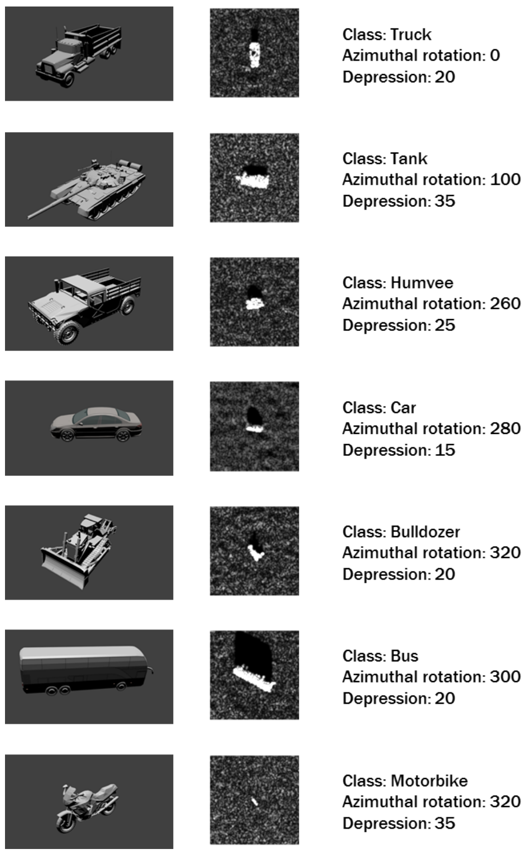

Figure 2: Examples on target instances and their simulated SAR image. Pixel values are clipped in order to show the background in the simulated images.

When training our machine learning algorithms we extract sub-images of 128x128 pixels from our dataset. This allow us to gradually extract datasets that have higher translational variance of the target. Our method is illustrated at **Figure 3**. The maximum distance away from the center which our sub image can be extracted is $\frac{192-128}{2} = 32$ pixels in each direction. We extract datasets with different random translation levels within 32 pixels. By gradually increasing the level of random translation we can measure how it effects the accuracy of the algorithms.

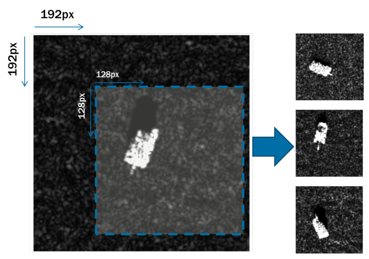

Figure 3: Illustration of how sub-image extraction gives us translational target variance.

3 Classifiers

The classifiers used in our study are all end-to-end learning based models. This means they all take raw pixel values as input and there are no hand crafted feature extractors preprocessing the data for the classifier. One of the nice properties on Convolutional Neural Networks is its ability to learn feature extractors in its convolution layers and base a nonlinear decision boundary on top of this.
The data set described in Section 2.1 is split randomly in five parts on which we make a 5-fold cross-validation of all classifiers. This gives us training set sizes of 1411 samples and test sets of 353 samples for each fold. We report performance of our classifiers as the fraction of correct classifications averaged over the cross-validation. The classifiers has been chosen to span complexity from very simple classifiers (K-Nearest Neigbor) to state of the art (Convolutional Neural Networks and Random Forests). We have futhermore tested various subcomponents of the CNN, the Multi Layer Perceptron i.e. the fully connected layers in the CNN and the Softmax i.e the output layer in the CNN.

3.1 Convolutional Neural Network

Our CNN model has 5 layers with trainable parameters, 3 convolutional and 2 fully connected hidden layers. We train our model with a stochastic gradient descent (SGD) algorithm minimizing the multiclass log-loss over our 7 target classes.

Layer	kernels	kernel size	Dropout fraction
C1	18	9x9	0.25
C2	36	5x5	0.25
C3	120	4x4	0.25
F4	400	120	0.50
F5	7	400	0.25

Table 1: CNN layer sizes shown. In total the model has 138'159 parameters.

After the first two convolution layers we apply a max pooling operator to reduce the dimensionality. The first max pooling operation has a window size of 6x6 pixels and the second a window size of 4x4 pixels. After all layers we apply a Rectified Linear Unit activation function as Krizhevsky et al. in [2] showed how this can make CNN's converge faster. We found it necessary to apply the dropout technique described by Srivastava et al. in [6] to prevent overfitting. In this technique a fraction of randomly chosen network weights are omitted from the weight updates in each iteration of the SGD algorithm. Our model layer sizes together with the fraction of dropout used can be seen in **Table 1**. Considering the amount of variation in target appearance in our dataset, a training set size of 1411 samples cannot be considered a lot and it seems reasonable that measures must be taken to prevent overfitting to the training data.

3.2 Classification models

We use an implementation of Random Forest [1], an ensemble method of classification trees from the Python library scikit-learn. The primary parameters for the Random Forest is the number of trees N and the number of features per tree m. The parameters (N, m) have been found by 5-fold cross-validation to be 512 trees and 128 features per tree.

K Nearest Neighbours is the classic approach of storing all training images together with their class. At test time, euclidean distances is calculated between the given test images and all training images and the k shortest distances vote which class the test images belongs to. the k-parameter have found by 5-fold cross-validation to be optimal at $k = 1$.

A simpler neural network where the input layer size is number of pixels 128^2, one hidden layer with 128 nodes and a softmax layer to output probabilities for the 7 classes in the dataset (2'114'432 trainable parameters).

Our softmax classifier is a one layer shallow neural network normalized with a softmax function output. It is similar to the Multilayer perceptron but has only a 7 neurons with each 16384 weights giving in total 114'688 trainable parameters.

.

4 Results

At **Figure 4** our results can be seen. The test score which is the percentage of correct classifications on the test data averaged over 5-fold cross-validation procedure.

It can be seen that the Convolutional Neural Network does not decrease in performance as fast as the other algorithms when we increase translational object variance. It is not a surprise that there is some decrease in all classifiers performance, because we keep the number of training images fixed as well as the size of the classifiers. When increasing the dimensions of variance we should also expect a lower performance. The interesting fact, seen at Figure 4, is the relative performance drop between the CNN and the other classifiers. It clearly shows that CNN can be trained to have translational invariance.

5 Conclusions

There are many ways to increase the problem size shown in this article towards more realistic scenarios for SAR ATR. We have created a simulated dataset that covers some many type of variance in target appearance. By controlling the amount translational variance of target alignment we have shown how CNN are superior to others machine learning algorithms in dealing with translational variance.

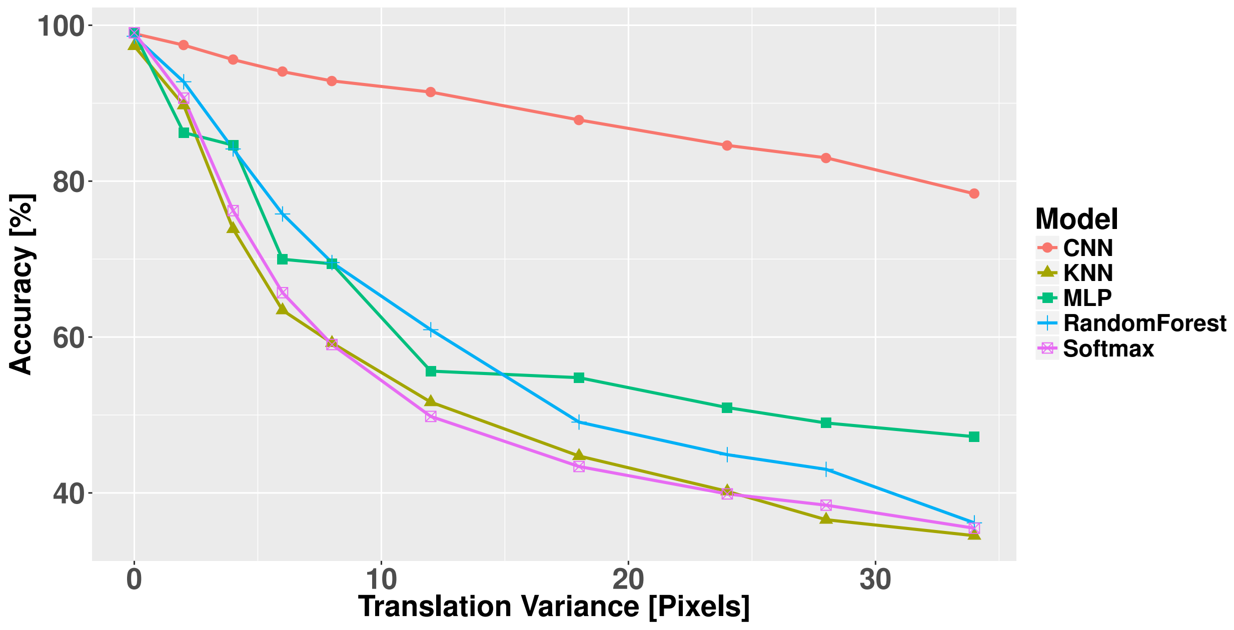

Figure 4: Test score plotted as a function of the percentage of randomly translational variance. Note that the first drop in performance happens around 3 pixels random translational variance

The impracticalities of collecting real SAR datasets that represent all natural variances of target representation for SAR ATR, are many. It is therefore very important to consider realistic problems when benchmarking algorithms on the sparse data available. We have shown that when considering the best algorithms for SAR ATR one must know the precision of placing targets consistently. Whether it is from a detection algorithm or manually extracted target patches, their displacement variability must be considered as even small inconsistent displacements can have big impact on the accuracy of a classifier. We have shown that some algorithms have a drastic decrease in performance when objects vary in position with as little as 3 pixels.

References

[1] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. pages 1097–1105, 2012.

[3] David A. E. Morgan. Deep convolutional neural networks for ATR from SAR imagery. *Proc. SPIE*, 9475:94750F–94750F–13, 2015.

[4] Joseph A O'Sullivan, Michael D DeVore, Vikas Kedia, and Michael I Miller. SAR ATR performance using a conditionally gaussian model. *Aerospace and Electronic Systems, IEEE Transactions on*, 37(1):91–108, 2001.

[5] Umamahesh Srinivas, Vishal Monga, and Raghu G Raj. SAR automatic target recognition using discriminative graphical models. *Aerospace and Electronic Systems, IEEE Transactions on*, 50(1):591–606, 2014.

[6] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Suskever, and Ruslan Slakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*.

[7] Steffen Wagner. Combination of convolutional feature extraction and support vector machines for radar ATR. In *Information Fusion (FUSION), 2014 17th International Conference on*, pages 1–6. IEEE, 2014.

[8] Qun Zhao and Jose C Principe. Support vector machines for SAR automatic target recognition. *Aerospace and Electronic Systems, IEEE Transactions on*, 37(2):643–654, 2001.