Deep Learning for SAR Image Classification

Chapter · January 2020

DOI: 10.1007/978-3-030-29516-5_67

4 authors:

Anas Hasni
National Institute of Statistics and Applied Economics

Majdoulayne Hanifi
Université Internationale de Rabat

Chaimae Anibou
Mohammed V University of Rabat

Mohamed Nabil Saidi
National Institute of Statistics and Applied Economics

Deep learning for SAR image classification

Hasni Anas[1], Hanifi Majdoulayne[2], Anibou Chaimae[3], and Saidi Mohamed Nabil[1]

[1] Institut National de Statistique et d'Economie Appliqueé,Laboratoire SI2M,Rabat
[2] Université Internationale de Rabat,Faculté d'Informatique et de Logistique,TICLab Rabat
[3] Université Mohammed V Agdal,Departement de Physique, Faculté de Science, Rabat

Abstract. Deep Learning algorithm has recently encountered a lot of success especially in the field of computer vision. The current paper aims to describe a new classification method applied to synthetic aperture radar (SAR), We used transfer learning followed by fine tuning methods in such a classification schematic; Pre-trained architectures on ImageNet database was used; VGG 16 was indeed used as a feature extractor and a new classifier was trained based on extracted features; the last three convolutional blocks of the VGG16 were then fine tuned ; Dataset used is the Moving and Stationary Traget Acquisition and recognition (MSTAR) data; We've reached a final accuracy of 97.91% on Ten (10) different classes.

Keywords: Deep Learning, Convolutional Neural Networks, Transfer learning, Fine tuning, Synthetic Aperture Radar.

1 Introduction

Synthetic aperture radar (SAR) can operates on multiple weather conditions and can produce large dimension images. The produced images are affected by a multiplicative noise known as speckle noise, Interpretation and understanding of SAR images is very difficult and extremely complex. Variety of approaches are being developed in order to make the understanding of SAR images less time-consuming and more practical and surpass linked difficulties as a consequence [1]. In recent years, Deep Learning algorithm and especially Deep Convolutional Neural Networks [2] (Convnets) has contributed in the successful progress of many Computer Vision tasks [3], such as classification, detection, and localization [4]. Deep Convnets, in the opposite of traditional classification tasks [1], are capable of automatically extracting features from images with the help of convolution and pooling layers. In order to build powerful architectures, Convnets need to be trained on very large amount of data which is not the case of the MSTAR[5] data. This paper attempts to improve SAR images classification task accuracy using the VGG 16[6] Network as a feature extractor and train a newly defined classifier based on the extracted features in a first step, and then in a second step fine tune the last three convolutional layers of the VGG 16 alongside the fully connected classifier in order to highly improve SAR images classification accuracy.

2 Methodology

Convnets consists of multiple layers of Convolution and Pooling and a last Fully connected layer for Classification (Fig. 1). In order to train such a deep architecture, large amount of data is needed. In the case of the MSTAR data we used 2746 images for training and 2425 images for testing; Each one contain ten classes (Tab. 1)

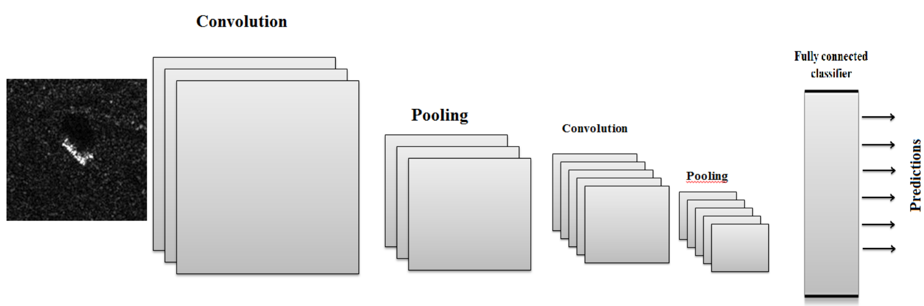

Fig. 1. Example of a Convolution Neural Network

Classes	training samples	testing samples
2S1	299	274
BMP2	233	195
BRDM_2	298	274
BTR60	256	195
BTR70	233	196
D7	299	274
T62	298	273
T72	232	196
ZIL131	299	274
ZSU_23_4	299	274

Table 1. MSTAR Data

Visual samples from the MSTAR data are shown in Fig.2

First, the loaded images into the VGG16 convnet do not undergo any preprocessing algorithms.

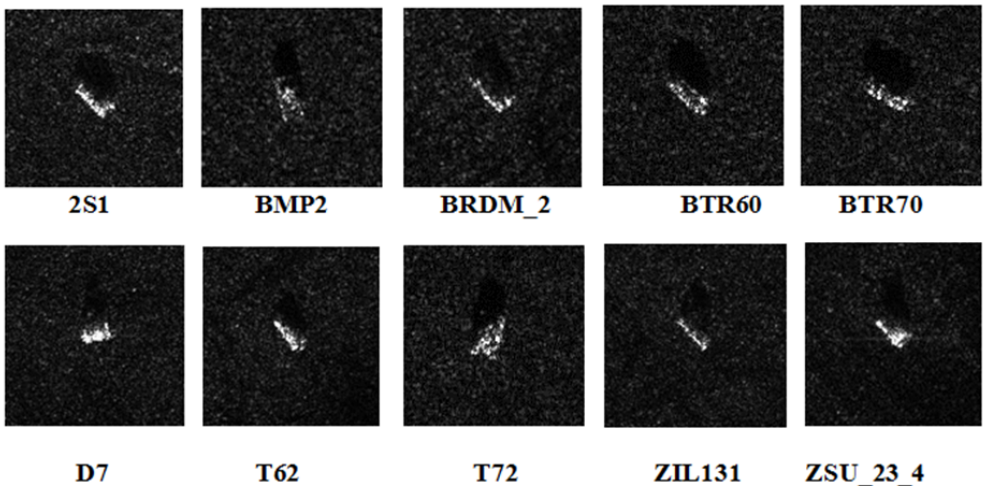

Fig. 2. Images from the MSTAR data

Following are the steps applied in order to extract features from the SAR images:

1- Load the VGG16 Network trained on the ImageNet [7] dataset (Fig.3)

2- Remove the fully connected classifier

3- Use the remaining layers as a feature extractor for both the training and testing sets (Fig.4)

4- Train a new fully connected classifier based on the extracted features (Fig.4).

The size of the extracted features is (7,7,512). The new classifier is defined by a first dense layer and 2000 neurons, a dropout layer in order to prevent overfitting and a final dense layer with 10 neurons referring to the ten classes from the MSTAR data and a softmax activation (Fig.5).

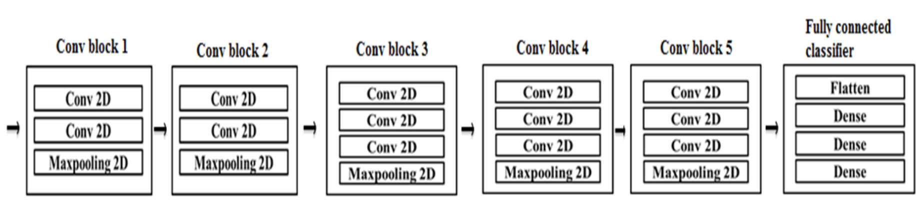

Fig. 3. VGG 16 Architecture

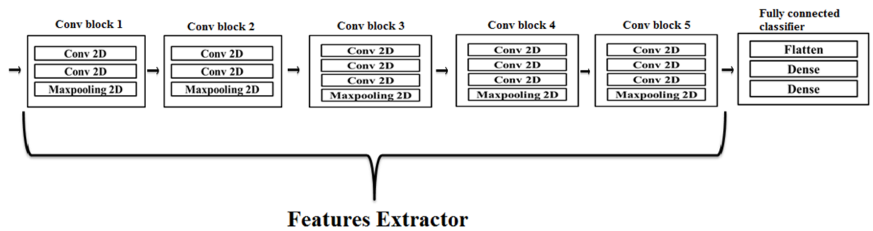

Fig. 4. Feature Extractor + New classifier

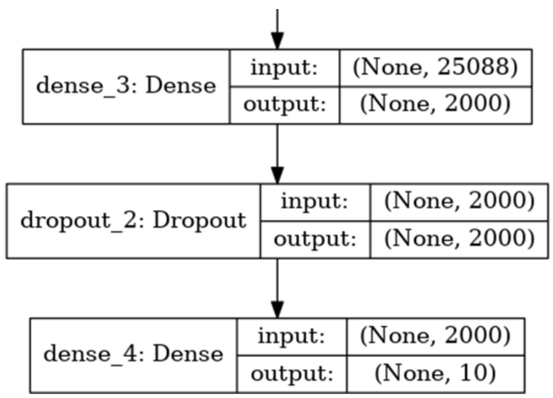

Fig. 5. Detailed architecture of the fully connected classifier

As a result for this process we reached an average test accuracy of 87.96%.

Secondly, when it comes to the fine tuning part the images go under rescaling between 0-1 in addition to data augmentation via some random transformations(shear range, zoom range, horizontal flip). This helps to prevent overfitting and a better model generalization (Fig.6).

To obtain more improvement, we tried to fine tune the last three Convolutional blocks alongside the newly defined fully connected classifier from the previous training (Fig.7).

This process can be done according to the following steps:

1- Load the VGG16 and its weights

2- Add the fully connected classifier from the previous training and load its weights

3- Freeze the first two Conv blocks, and fine tune the last three conv blocks

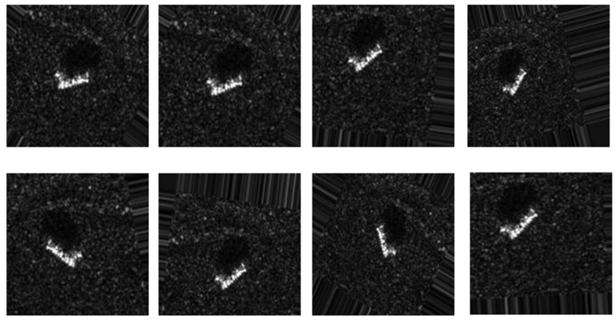

Fig. 6. Example of augmented data

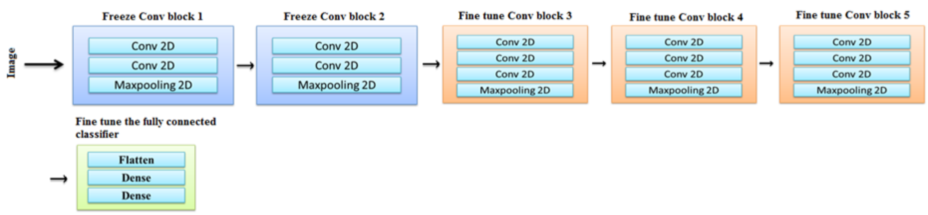

Fig. 7. Fine tuning

3 Training Approach

In this section we will describe the training process on the MSTAR database.

After loading the images into the VGG 16 Network and using it as a feature extractor, we need to evaluate the training efficiency. As a consequence we used a loss function which defines the error between the predicted classes and the ground-truth classes.

The loss function used in this paper is cross-entropy error function :

$$H(y,p) = -\sum_i (y_i log(p_i)) \tag{1}$$

y : ground-truth classes

p : predicted classes

we use alongside this function, the softmax function which produces an output probability for each class, where all outputs sum is 1.

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}} \tag{2}$$

p_i : output probability

a_i : hidden layer activations

the number of trainable parameters of the new classifier is shown in (Fig.8).

In order to speed up the training process, we used RMSprop optimization algorithm and we fixed the learning rate to 2.10^{-5}.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_11 (Flatten)         (None, 25088)             0
_________________________________________________________________
dense_21 (Dense)             (None, 2000)              50178000
_________________________________________________________________
dropout_11 (Dropout)         (None, 2000)              0
_________________________________________________________________
dense_22 (Dense)             (None, 10)                20010
=================================================================
Total params: 50,198,010
Trainable params: 50,198,010
Non-trainable params: 0
```

Fig. 8. Parameters of the new classifier

When it comes to fine tuning [8] part we used the same cross-entropy error function alongside the softmax function, and for the optimization we used the RMSprop algorithm where the learning rate in this case is set to 1.10^{-4}.

the number of trainable parameters in fine tuning part is shown in (Fig.9).

4 Experimental Results

As mentioned in the late sections, the data used is the MSTAR data which is divided into training and test sets, that each one set contains ten classes.

As first step we will show the confusion matrix; the VGG16 is used as feature extractor; As a second step we will show the confusion matrix that concerns the fine tuning. And as a third and a final step we will show the comparison between some earlier accuracies of classification tasks in relation with SAR images and the obtained accuracy in this paper.

4.1 Confusion matrix of features extraction part

The confusion matrix shown bellow concerns the first step in which we used the VGG16 as features extractor (Tab.2).

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_14 (InputLayer)        (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
sequential_13 (Sequential)   (None, 10)                50198010
=================================================================
Total params: 64,912,698
Trainable params: 64,652,538
Non-trainable params: 260,160
```

Fig. 9. Fine tuning parameters

4.2 Fine tuning confusion matrix

In this section we show the confusion matrix of the fine tuning part in odrer to improve the classification accuracy (Tab.3).

Table 2. Confusion matrix 1

test set	2S1	BMP2	BRDM_2	BTR60	BTR70	D7	T62	T72	ZIL131	ZSU_23_4	accuracy
2S1	235	1	1	5	0	0	21	0	3	8	85.76%
BMP2	0	126	4	31	15	0	0	18	0	1	64.61%
BRDM_2	0	0	263	6	0	0	0	2	0	3	86.13%
BTR60	0	2	4	179	3	0	0	2	0	5	91.79%
BTR70	0	7	2	33	146	0	0	8	0	0	74.48%
D7	0	0	0	0	0	263	4	0	0	7	95.98%
T62	0	0	0	0	0	0	257	0	1	15	94.13%
T72	0	6	1	12	4	0	0	173	0	0	88.26%
ZIL131	0	0	0	0	0	2	0	0	270	2	98.54%
ZSU_23_4	0	0	0	0	0	0	0	0	0	274	100%

Table 3. Confusion matrix 2

test set	2S1	BMP2	BRDM_2	BTR60	BTR70	D7	T62	T72	ZIL131	ZSU_23_4	accuracy
2S1	258	1	0	0	0	0	1	0	14	0	94.16%
BMP2	3	183	1	0	2	0	0	5	1	0	93.84%
BRDM_2	0	0	267	2	0	0	0	2	2	1	97.44%
BTR60	1	4	0	187	1	0	0	2	0	0	95.89%
BTR70	0	0	0	0	196	0	0	8	0	0	100%
D7	0	0	0	0	0	273	0	0	1	0	99.63%
T62	0	0	0	0	0	0	271	2	0	0	99.26%
T72	0	2	0	0	0	0	0	194	0	0	98.97%
ZIL131	0	0	0	0	0	0	0	0	274	0	100%
ZSU_23_4	0	0	0	0	0	0	0	0	0	274	100%

4.3 Comparison

In order to have a clear understanding on the obtained results, we compared the classification with similar approaches that used deep learning to classify the MSTAR data such as [9] in which they used unsupervised features learning algorithm and obtained an average classification rate of 84.7%.

In another paper [10] authors replaced the fully connected layers with sparsely-connected convolutional layers under the name of All-Convolutional Networks (A-ConvNets). The average accuracy that they obtained is 99.1%

5 Conclusions and perspectives

In this work, we used as first step the VGG16 pretrained convolution neural network on the ImageNet database as a feature extractor, to classify the SAR images. We obtained an average accuracy 87.96%.

Thus, we wanted to improve this accuracy value by fine tuning, the last three convolutional blocks of the VGG16 alongside the defined classifier from the first step. As a result we obtained 97.91% as average accuracy.

As a future work, we want to use other pretrained convolutional neural network architectures, which they have more deep architecture than the VGG16.

References

1. M. N. Saidi, Ab. Toumi, A. Khenchaf, D. Aboutajdine, B. Hoeltzener, “Feature Extraction and Fusion for Automatic Target Recognition Based ISAR Images,”2009.
2. Deep Learning, Ian Goodfellow and Yoshua Bengio and Aaron Courville, MIT Press, http://www.deeplearningbook.org, 2016
3. A. Krizhevsky, I. Sutskever, Geoffrey E. Hinton, NIPS'12, Proceedings of the 25th International Conference on Neural Information Processing Systems , vol. 1. , PP 1097-1105, 2012.
4. Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, Yann LeCun, “OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks,”, arXiv:1312.6229, 2013.
5. https://www.sdms.afrl.af.mil/index.php?collection=mstar
6. Deep Learning with python, Franois Chollet, Manning, 2017
7. http://image-net.org/challenges/LSVRC/
8. K. Simonyan, A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition ,”, arXiv:1409.1556, 2014.
9. Sizhe Chen, Haipeng Wang, “SAR Target Recognition Based on Deep Learning,” , pp. 541-547, DSAA 2014 - Proceedings of the 2014 IEEE International Conference on Data Science and Advanced Analytics, 2014.
10. Haipeng Wang, Sizhe Chen, Feng Xu, and Ya-Qiu Jin,, APPLICATION OF DEEP-LEARNING ALGORITHMS TO MSTAR DATA, pp.3743-3745, 2015.