DEEP LEARNING BASED SAS IMAGE CLASSIFICATION WITH SUPPLEMENTARY INFORMATION OF IMAGING GEOMETRY

Narada Warakagoda Norwegian Defence Research Establishment (FFI), Kjeller, Norway
Øivind Midtgaard Norwegian Defence Research Establishment (FFI), Kjeller, Norway

1 INTRODUCTION

Automatic Target Recognition (ATR) in underwater environments is an important part of Maritime Mine Counter Measures (MMCM). Classification of objects in synthetic aperture sonar (SAS) images is one of the main tasks in such ATR systems and deep learning has proven highly successful in this task[1–3].

Inspiration for such deep learning based systems is mostly drawn from the field of optical image classification. However, unlike the common optical images, SAS images data often include additional information, e.g. the imaging geometry. As this supplementary information can provide important clues about the scene, it may benefit SAS image processing tasks such as object detection and classification. Our earlier work[4] explored techniques to exploit imaging geometry as an additional source of information for improving the classification performance of SAS images with deep neural networks. In this paper, we extend those techniques and experimentally evaluate them.

Even though several parameters including ground/slant range, sensor altitude, seafloor slope and object pose are required to fully define the SAS imaging geometry, we considered only the range (ground/slant) and sensor altitude as they are readily available. Thus, in our case, we represent geometry information using a 2-D vector (*range, altitude*).

There are different techniques for utilizing the imaging geometry information in a deep learning system. Encouraged by our previous work[4], we chose to consider imaging geometry as a constraint on the space of the internal feature maps of the deep neural network. This would require transforming the 2-D geometry vector to a high dimensional feature map and we use a sonar simulator to assist this transformation. Training with such a constraint can be seen as a kind of *manifold regularization* problem[5] and the constraint can be included as a term of the loss function used in training.

We considered supplementary information on imaging geometry in the context of classifying bottom objects into the classes of cylinder, truncated cone, wedge and clutter.

Experiments with the above regularization based approach of training showed that classification performance can indeed be improved by making use of imaging geometry information[4].One disadvantage of using supplementary information for regularization is that the method would only be applicable for training. It would be desirable if the supplementary information can be used to assist the testing process as well. Therefore, in this work we propose a method for using information of imaging geometry also in testing and evaluate it experimentally.

This paper is organized as follows: Section 2 describes the background techniques relevant for this work. Section 3 gives a description of the task including the dataset and training/evaluation procedures. Details of the experiments including the model architectures are presented in Section 4. Results are reported in Section 5 and finally in Section 6 conclusions are drawn.

2 BACKGROUND

2.1 Deep Learning

Deep Learning has been immensely successful in many tasks related to artificial intelligence including computer vision. There are a number of well known deep Convolutional Neural Network (CNN) architectures that are pre-trained with optical images and usually available in the public domain. The architecture known as Inception-Resnet-v2 is one of them and all the experiments in this work were conducted using this architecture as the backbone[7]. The reason for this choice is that it was among the best in optical image classification while keeping the number of parameters and computational cost at a reasonable level.

In the past couple of years, several superior neural networks have emerged, especially those based on Transformers[8]. These networks may lead to better performances, and hence the choice of Inception-Resnet-v2 is by no means optimal.

We also apply *transfer learning*, meaning that the pre-trained neural network on optical images is trained again on our own SAS image data.

2.2 Regularization

Machine learning indirectly aims to maximize the performance of a model on a test set by optimizing the model on a training set. In order to prevent the model from overfitting to the training set and hence losing its ability to perform well on the test test, one can restrict the optimization on the training set. This is done by introducing biases to the training problem which in effect expresses preferences for some solutions over the others[9]. This process is known as regularization. One of the most widely used regularization techniques is *weight decay* and in this case the Euclidean norm of the parameter vector is minimized during training, having a loss function of the form

$$L(\mathbf{w}, \mathbf{x}) = L'(\mathbf{w}, \mathbf{x}) + \lambda ||\mathbf{w}||^2$$

where L' is the loss function before regularization, $\mathbf{w}$ is the model parameter vector, $\mathbf{x}$ is training data and λ is a constant. In this case the regularization term is a simple function of the model parameters. It can, however, be a more general function of the model parameters as well as training data $\mathbf{x}$:

$$L(\mathbf{w}, \mathbf{x}) = L'(\mathbf{w}, \mathbf{x}) + \lambda G(\mathbf{w}, \mathbf{x})$$

In the case of deep learning, the general regularization term can force outputs of some layers of the neural network layers to take a particular form dictated by the nature of G. Therefore, it can be seen as a kind of manifold regularization. This can also be seen as *multi-task learning* as there are several tasks incorporated in the loss function[6]. In this work we make use of such regularization functions.

3 TASK

Our system consists of a blob detector followed by a classifier. The blob detector outputs locations of probable objects in a full scale SAS image. Then we extract image snippets centered around these locations and consider a classification task where a given sonar snippet is classified into 4 classes: cylinder, truncated cone, wedge and clutter. The first three classes represent regular geometric shapes whereas the last one is a composite class containing spurious detections and other objects that do not belong to the first three classes.

Note that for simplicity we call the SAS image snippet a sonar image.

We implemented a baseline system that realizes the task, by adding a classification head to a Base-CNN as shown in Figure 1. In our implementation, we used the Inception- Resnet-V2 architecture[7] as the Base-CNN and a single layer fully connected neural network as the classification head. The

Base-CNN converts the input sonar image into a feature vector and transfers it to the classification head. The task of the classification head is to predict the probabilities of the pre-defined classes based on the feature vector.

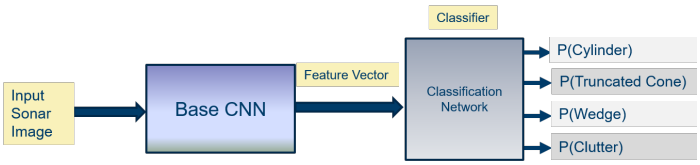

Figure 1: Baseline classification system

The base-CNN is initialized with the original parameter values pre-trained on the optical images from the Imagenet dataset[10]. Classification network parameters are initialized with random values before the whole network (Base-CNN and the Classification network) is trained on the sonar images.

3.1 Data set

The data set used in this work consists of Synthetic Aperture Sonar (SAS) images collected using a HISAS 1030 sensor[11] mounted on a HUGIN autonomous underwater vehicle*. The sonar images were first sent through a blob detector and image snippets of size 299x299 pixels were extracted around each detection. In this way, about 72000 snippets were collected, where a vast majority of the images belonged to the clutter class. More specifically, there are about 1400 cylinder images, 1200 truncated cone images and 200 wedge images, whereas about 68000 snippets are clutter images. This is clearly a highly unbalanced data set and therefore class weighting was applied during training to counter this imbalance. About 90% of the total images were used as the training set and the remaining images were set aside as the test set. The training set and test set were augmented through flipping along the across track direction and random translations. This resulted in a final training and test set sizes of 140000 and 20000 images respectively.

3.2 Training and Evaluation

Before the training procedure is started, the base-CNN is initialized with the original, pre-trained parameter values, whereas the classification network is initialized with random values. Then the whole network is trained using the training set described above. We employed the update rule of stochastic gradient descent (SGD) with momentum together with a batch-size of 25 images. In each experiment, the system was trained for 20 epochs.

For training of the baseline system, we use the cross entropy (CE) loss. That is

$$L_{\text{baseline}} = -\frac{1}{N}\sum_{i=1}^{N}\log P(C_{t_i})$$

where t_i is the target class of the i^{th} sample, $P(C_j)$ is the probability of object class j and N is the number of samples in the training set.

When modifications are introduced to the baseline architecture, the loss function is changed accordingly, essentially by adding a regularization term. Details of these modifications are described in the section on experiments.

Once the system, either the baseline or a modified version of the baseline is trained, its performance is evaluated on the test set.

We calculated several evaluation metrics on the test set after each training epoch.

* https://www.kongsberg.com/maritime/products/marine-robotics/autonomous-underwater-vehicles/AUV-hugin/

- **Accuracy:** This is the ratio between the number of correctly classified images and the total number of images in the test set. This metric may not represent the true performance for a highly imbalanced data set like ours, because accuracy can be quite high even when all the samples are classified into the clutter class.
- **Average Recall:** We calculated Recall averaged over all classes. i.e.

$$R_{\text{ave}} = \frac{1}{C}\sum_{i=1}^{C}\frac{n_{ii}}{\sum_{j=1}^{C} n_{ij}},$$

  where $n_{i,j}$ is the number of class i images classified into class j and C is the number of classes.
- **Average Area Under the Curve:** Unlike the previous two, this metric does not depend on a particular threshold in classification.Therefore, this is a highly suitable metric for our problem. We first create receiver operating characteristics (ROC) curves, that is the graph of the true positive rate against the false positive rate, for each of the classes. Then the area under the ROC curve (AUC) is calculated for each class and the final metric is obtained by averaging AUC values for all classes.

4 EXPERIMENTS

Our goal of the experiments was to study utilization of geometry information both in training and testing of the classifier.

4.1 Training

In training, we used the imaging geometry information to derive a regularization term. The main idea of this approach is illustrated in Figure 2. As shown in this figure, the baseline architecture (the lower part of the network) is supplemented with a couple of operations (the upper part of the network) which are used to compute a regularization term that is added to the baseline loss function. These operations take the imaging geometry parameters as inputs and compute an intermediate representation $\mathbf{Z}^B$ which is compared with the equivalent representation $\mathbf{Z}^A$ derived from the input image. The regularization term is a measure of the distance between $\mathbf{Z}^B$ and $\mathbf{Z}^A$. By adding this loss to the optimization procedure in training, we try to keep $\mathbf{Z}^A$ as close as possible to $\mathbf{Z}^B$. In this way, information contained in imaging geometry parameters are transferred to the system parameters.

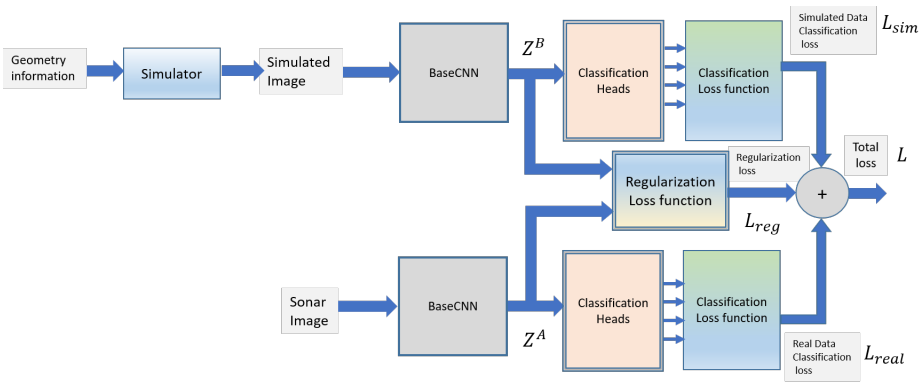

Figure 2: Training with a regularization configuration where geometry information is used to derive a regularization term

More details of calculating $\mathbf{Z}^A$ and $\mathbf{Z}^B$ are as follows: As Figure 2 shows, in computing $\mathbf{Z}^B$, imaging geometry is first used as an input to a SAS image simulator. In order to simulate a SAS image of an

object with a known geometric shape, we need ground range and sensor altitude as well as the object orientation and seafloor slope. Since only the ground range and the sensor altitude are readily available from sonar data records, the other parameters, object orientation and seafloor slope are estimated using the given input sonar image. For this purpose, a dedicated image processing algorithm is used[12]. Once all the parameters are estimated and the geometric shape of the simulated object is defined, the SAS image simulator can generate a simulated image. Then the BaseCNN network is used to transform the simulated image to a feature vector $\mathbf{Z}^B$.

The real sonar image is input to the BaseCNN and it outputs the corresponding feature vector $\mathbf{Z}^A$. Note that the same BaseCNN is used to calculate both $\mathbf{Z}^A$ and $\mathbf{Z}^B$. Further the same network is used as the classification head in both branches.

Once $\mathbf{Z}^A$ and $\mathbf{Z}^B$ are known they are used to calculate the regularization loss L_{reg}. There are several ways to define a loss based on these two quantities, but in this work we used the following:

$$L_{reg} = \frac{1}{N}\sum_{i=1}^{N}\left(l_i - \exp\left(-||\mathbf{Z}_i^A - \mathbf{Z}_i^B||\right)\right)^2,$$

where $\mathbf{Z}_i^A$ and $\mathbf{Z}_i^B$ are i^{th} feature vectors, $||\cdot||$ is the Euclidean distance, N is the number of samples in the training set and

$$l_i = \begin{cases} 1, & Z^A \text{ and } Z^B \text{ from the same object class} \\ 0, & Z^A \text{ and } Z^B \text{ from different object classes} \end{cases}$$

Minimizing L_{reg} would then lead to the Euclidean distance between feature vectors $\mathbf{Z}^A$ and $\mathbf{Z}^B$ being minimized when they are from the same class and maximized otherwise.

In addition to the regularization process, both branches try to classify the incoming feature vector into the predefined set of classes. For both these classifications we can define the cross entropy loss:

$$L_{real} = -\frac{1}{N}\sum_{i=1}^{N}\log P(C_{t_i^r}) \quad \text{and} \quad L_{sim} = -\frac{1}{N}\sum_{i=1}^{N}\log P(C_{t_i^s})$$

where t_i^r and t_i^s are target class index for the real sonar image and simulated image respectively.

We train the whole system by minimizing the total loss L given by

$$L = L_{reg} + L_{real} + L_{sim}$$

The basic steps of the training procedure is shown in Figure 3.

4.2 Testing

Our approach to using imaging geometry information in testing is illustrated in Figure 4. The architecture used in testing has two main parts: the regular processing chain of the input sonar image (the bottom part of Figure 4) and the processing chains for simulated images (top part).

As in the case of training, BaseCNN converts the input sonar image to a feature vector $\mathbf{Z}^A$. In addition, based on the geometry information and sea-floor slope and object pose extracted from the sonar image, three images are simulated for the classes of regular shaped objects, i.e. cylinder, truncated cone and wedge. BaseCNN again converts all these simulated images to feature vectors, $\mathbf{Z}_1^B$, $\mathbf{Z}_2^B$ and $\mathbf{Z}_3^B$. While the feature vector $\mathbf{Z}^A$ undergoes the normal classification procedure with the classification network to produce probabilities $P_i^C,\ i = 1, 2, 3, 4$, it also participates in distance calculation operations with $\mathbf{Z}_1^B$, $\mathbf{Z}_2^B$ and $\mathbf{Z}_3^B$.

- For all batches in the training set
  - For all sonar images in the batch
    - Calculate $\mathbf{Z}^A$
    - Get geometry information
    - Estimate sea-floor and object pose
    - If sonar image of regular shaped object
      - Simulate image of the object of the same class
      - $l = 1$
    - Else (i.e. sonar image is a clutter object)
      - Simulate image of a random class
      - $l = 0$
    - Calculate $\mathbf{Z}^B$ using the simulated object
    - Update Losses
  - End
- End

Figure 3: Pseudo code for training

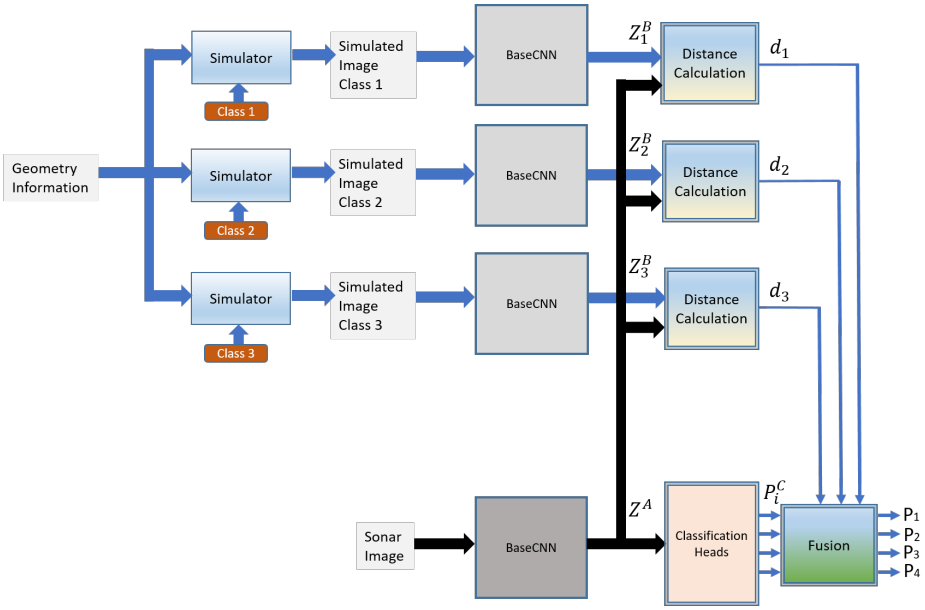

Figure 4: Network architecture for using geometry information in testing

The results of distance calculations d_1, d_2 and d_3 are given by

$$d_i = ||\mathbf{Z}^A - \mathbf{Z}_i^B||, \quad i = 1, 2, 3$$

where $||\cdot||$ is the Euclidean norm. Finally, these distances are fused with the classification probabilities $P_i^C,\ i = 1, 2, 3, 4$ to get the final class probabilities $P_i,\ i = 1, 2, 3, 4$.

In order to perform fusion, we need to convert the distance d_i to a probability like measure P_i^d. For this conversion, we use the relation

$$P_i^d = \exp(-\alpha d_i - \beta) \quad i = 1, 2, 3$$

where α and β are constants which are estimated for each sample by setting $P^d = 0.9$ for $d = d_{min}$ and $P^d = 0.1$ for $d = d_{max}$ with $d_{max} = \max_i d_i$ and $d_{min} = \min_i d_i$.

We have used a simple fusion strategy, where $P_i = \max(P_i^C, P_i^d), \quad i = 1, 2, 3$ and $P_4 = P_4^C$. The final probabilities $P_i,\ i = 1, 2, 3, 4$ are used to estimate the performance metrics, accuracy, average recall and average AUC.

5 RESULTS

We trained the baseline system and then the architecture with regularization configuration shown in Figure 2. The evaluation metrics were computed for the test set after completion of 20 epochs of training. We considered three test modes: using only the classifier output (P_i^C), only the distance measures based scores (P_i^d) and the fused classifier output and distance measures (P_i).

Table 1 shows the value of each evaluation metric in the different experiments conducted. The winner with respect to each metric is shown in bold. We can see that all the winners except for AUC come from the regularization experiments. Further, Table 1 reveals that when the classifier output is fused with the distance measures, the performances are superior to the classifier only case. However, this superiority does not appear to be significant. In addition, we can observe that distances alone can produce a respectable level of performance, even though it is inferior to classifier only performance values.

Table 1: Evaluation metrics for different configurations of training and testing

Training Configuration	Test mode	Accuracy	Recall	AUC
Baseline	Classifier only	67.69	49.29	**90.19**
Regularization	Classifier only	95.92	50.92	76.41
	Distances only	95.56	41.24	65.23
	Fused classifier and distances	**95.94**	**51.27**	76.76

In Table 2, confusion matrices for the three test modes when the system is trained with regularization are shown. Rows and columns of the table are corresponding to the true classes and the predicted classes respectively. Each table entry shows the number of examples of the corresponding true class (row) that are classified to the corresponding prediction (column). As can be seen from Table 2, the fused testing mode (FCD) is the best approach for cylinder objects. It is also better than the classifier only approach for truncated cones. For the wedge shaped objects all three approaches perform equally unimpressively. On clutter objects, all three approaches perform well, but the fused mode is the most inferior one.

Table 2: Confusion matrices for the three test cases, classifier only (CO), distances only (DO) and fused classifier and distances (FCD)

	Cylinder (1)			Tr. Cone (2)			Wedge (3)			Clutter (4)		
	CO	DO	FCD	CO	DO	FCD	CO	DO	FCD	CO	DO	FCD
1	444	453	**460**	0	0	3	0	0	0	438	426	420
2	0	0	0	540	**550**	547	0	0	0	112	102	105
3	0	0	0	0	0	0	17	17	17	139	139	139
4	24	26	26	48	63	61	36	32	38	**19325**	19312	19308

6 CONCLUSION

In this work, we have studied how to exploit the additional information typically available for SAS images for improving the classification performance. The results indicate that imaging geometry parameters (sensor altitude and range) can be used to improve the classification performance by using them as a means of regularization in training. Further, we presented a method for including imaging geometry parameters also in testing. That approach led to slightly better performance metrics than the case where testing is performed only with the sonar image. Even though the improvements do not seem significant, there are encouraging signs that warrant further investigations and tuning.

ACKNOWLEDGMENTS

We thank the Royal Norwegian Navy and Kongsberg Maritime for providing HISAS data used in the work. We also thank the NATO STO Centre for Maritime Research and Experimentation for providing the MUSCLE SAS data, which was collected with funding from the NATO ACT.

References

1. D. P. Williams, “Underwater target classification in synthetic aperture sonar imagery using deep convolutional neural networks,” Proc. 2016 23rd International Conference on Pattern Recognition (ICPR), pp. 2497–2502 (2016).

2. N. D. Warakagoda and Ø. Midtgaard, “Fine-tuning vs full training of deep neural networks for seafloor mine recognition in sonar images,” Proc. Underwater Acoustics Conference and Exhibition (UACE), Skiathos, Greece, (2017).

3. C. Li, Z. Huang, J. Xu, and Y. Yan, “Underwater target classification using deep learning,” Proc. OCEANS 2018 MTS/IEEE Charleston, pp.1–5 (2018).

4. N. D. Warakagoda and Ø. Midtgaard, “Utilizing imaging geometry meta-data in classification of synthetic aperture sonar images with deep learning,” Proceedings of Meetings on Acoustics, vol. 47, (2022).

5. M. Belkin, P. Niyogi, and V. Sindhwani, “Manifold regularization: a geometric framework for learning from labeled and unlabeled examples,” Journal of Machine Learning Research, vol. 7, pp. 2399–2434, (2006).

6. M. Crawshaw, “Multi-task learning with deep neural networks: A survey,” *CoRR*, vol. abs/2009.09796, [Online]. Available: https://arxiv.org/abs/2009.09796, (2020).

7. C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” (2016).

8. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in Advances in Neural Information Processing Systems, pp. 5998–6008, (2017).

9. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, http://www.deeplearningbook.org (2016).

10. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” Proc. IEEE conference on computer vision and pattern recognition. Ieee, pp. 248–255 (2009).

11. P. E. Hagen, T. G. Fossum, and R. E. Hansen, “HISAS 1030: The next generation mine hunting sonar for AUVs,” Proc. UDT Pacific 2008 Conference Proceedings, (2008).

12. H. Midelfart and Ø. Midtgaard, “Adaptive template matching for sas images,” 10th European Conference on Underwater Acoustics (ECUA), Turkey, (2010).