FINE-TUNING VS. FULL TRAINING OF DEEP NEURAL NETWORKS FOR SEAFLOOR MINE RECOGNITION IN SONAR IMAGES

Narada Warakagoda[a], Øivind Midtgaard[a]

[a]Norwegian Defence Research Establishment (FFI), Kjeller, Norway

Narada Warakagoda, Norwegian Defence Research Establishment (FFI), Kjeller, Norway
narada-dilp.warakagoda@ffi.no

Abstract: *Deep Convolutional Neural Networks (DCNN) is a promising approach for automatic target recognition (ATR) which effectively combines feature extraction and classification. DCNN can automatically learn the relevant features for the recognition task at hand from the training data, and thus potentially eliminate the tedious and manual feature design process distinctive for traditional classification systems. However, a huge amount of labelled data is necessary to successfully train a DCNN of typical size. Fine-tuning is a commonly applied technique when the available amount of training data is insufficient. This approach has given good results, even when the associated semantics of the fine-tuning data and original training data are significantly different. However, it is uncertain whether this approach will work well for sonar images because most of the well-known pre-trained DCNNs such as Alexnet, VGG and ResNet have been trained on optical images. The fine-tuning data and original training data will then differ not only in semantics, but also in physics of image generation.*
In this work, we evaluated fine-tuning of two popular DCNNs, AlexNet and VGG16, against full training of a smaller network on Synthetic Aperture Sonar (SAS) images. The experiments were conducted in the context of a seafloor mine recognition task where four classes were considered; Cylinder, Manta, other mines and non-mine objects. Performance of the resultant nets were evaluated using an independent test set.

Keywords: *Deep Convolutional Neural Networks, Mine recognition, Sonar Images, feature extraction, classification*

1. INTRODUCTION

Automatic target recognition (ATR) of seafloor mines in sonar images is a vital technology to increase the efficiency of mine counter-measures (MCM) operations performed with autonomous underwater vehicles (AUV) [1]. Firstly, ATR can reduce the human workload during analysis of the large data sets generated by modern high-resolution sonars. Secondly, on-board ATR processing allows the vehicle to adapt its mission plan based on analysis results, e.g. to revisit positions of recognized targets for close-up camera identification within the same mission [2]. Recognizing mines in sonar images is a challenging task, however, due to immense variations in sonar conditions, seafloor characteristics, sensing geometry and target properties. In spite of significant research efforts over the last 25 years, most existing systems for automated mine recognition struggle with an abundance of false alarms (detected non-mines), particularly in complex seafloor areas with texture or clutter [1]. Designing features that can robustly recognize the defined object classes under varied conditions is arguably the most critical part of ATR development, and has traditionally been a labor-intensive process [3].

Recently, deep convolutional neural networks (DCNN) have demonstrated superior capabilities for object classification in optical images [4],[5]. DCNNs have the major advantage that the system can automatically learn discriminative features for the current recognition task from the training data [6]. One disadvantage is, however, the requirement for a huge set of labeled data to train the network from scratch. This has triggered a significant interest in transfer learning from existing DCNNs. In [7], transferability of features at different layers is studied within the domain of optical images. Research on fine-tuning and transfer learning from optical images to other domains such as medical images has started to appear in the literature [8],[9]. For mine recognition in sonar images, it is reported in [10] that DCNNs outperform approaches that combine manual feature design and shallow classifiers. However, they opted to fully train their networks and transfer learning was not considered.

In this work, we evaluate fine-tuning of two popular DCNNs, AlexNet [4] and VGG16 [5], against full training of a smaller network on Synthetic Aperture Sonar (SAS) images. Section 2 introduces these two DCNNs, while section 3 discusses the process of fine-tuning an existing network. We describe our experimental setup in section 4 and present the results in section 5. Finally, section 6 gives the conclusions.

2. DEEP CONVOLUTIONAL NETWORKS

There are a number of DCNN architectures well known within the deep learning community. Alexnet [4] is one such architecture which in fact triggered the current interest in deep convolutional networks after its win in the ILSVRC competition [11] in 2012. It has 5 *convolution layers*, 3 *max-pooling layers* and 3 *fully connected layers*. All the convolutional layers and fully connected layers except the last one has *ReLu* non-linearity, whereas the last fully connected layer has a 1000-way *softmax* non-linearity supporting classification into 1000 classes. It has 62 million parameters and reported top 5 test error rate on the Imagenet dataset is 15.3% [4].

Another, well-known network architecture is the VGG net which came to prominence through the ILSVRC competition in 2014 [5]. VGG has different configurations and VGG16 is one of the popular configurations. It has 16 layers with trainable parameters and

about 138 million parameters. Top 5 test error rate reported for the Imagenet dataset is 7.1% [5].

3. FINE-TUNING

Training a deep neural network such as Alexnet or VGG16 which contains tens or hundreds of millions parameters is a non-trivial task that takes hours, days or even weeks. In addition, one needs a large amount of training data to be able to estimate such a large number of parameters robustly. A much cheaper approach in terms of both required amount of training data and time, is to adapt an already trained network to a new task. This process is generally known as *transfer learning*. One form of transfer learning known as *fine-tuning* is an approach where a selected set of layers of the pre-trained network is trained again using the new dataset [6],[7].

Fine-tuning of a few layers close to the output of the network is particularly interesting, because it is computationally much less expensive. However, if the new task has a number of classes different from that of the original task, then one needs to replace at least the last layer of the pre-trained network with additional layers of neurons. This set of new layers forms a simple classifier and its input can be viewed as "feature vectors" calculated by the pre-trained network.

Fine-tuning of last layers (i.e. layers close to the output) is a reasonable approach when the feature vectors calculated by the pre-trained network are meaningful for the new task. That means that the statistical properties of data on which the network is pre-trained must be similar to the properties of the new data. If that is not the case, for example when networks trained on optical images are used for sonar image classification, fine-tuning of the first layers (i.e. the layers close to the input) may be needed. Note however that we may still need new trainable layers at the output to adapt to the number of classes of the new task.

Yet another fine-tuning approach would be to re-train the whole network with new data. This approach has the highest computational cost, although it is not significantly higher than that of fine-tuning of the first layers.

4. EXPERIMENT SETUP

All three fine-tuning approaches outlined above were studied with the Alexnet and VGG-16 networks pre-trained with optical images from the Imagenet database. For full training we designed a relatively smaller network.

3.1. Data preparation

We built a train-set and a test-set using annotated images from an MCM-database which was created with the HISAS 1030 synthetic aperture sonar mounted on Hugin underwater vehicles. These images are annotated with locations of various seafloor objects including different types of mines. At each location, an area of 227x227 pixels is extracted and converted to a 3-channel RGB-image, because pre-trained networks accept only 3-channel images. This process resulted in approximately 4800 unique images, of which around 420 images were set aside for creating the test set. Note that all images in the test-set are from different missions than used in the train-set. Then both the train-set and test-set were expanded about 12 fold using an augmentation approach that involves random

translation and flipping around the along-track axis. Ultimately we had 52800 training images and 5000 test images.

For the experiments we considered 4 classes: cylinder, manta, other mines and non-mine objects. Overall we have about 25% cylinder mines, 25% manta mines, 40% non-mines and 10% other mines in our database.

4.2. Experiments

In this section we present the details of 13 different experiments we have conducted. The first three experiments deal with fine-tuning the last layers of the Alexnet. This can be considered to be equivalent to feeding features calculated by the Alexnet to a new neural network. For that purpose in these experiments, we used a 3-layer Multi-layer Perceptron (MLP) with respectively 500, 100 and 4 neurons in its 1st, 2nd and 3rd layers. In experiments named *Alex_last_pool3*, *Alex_last_fc1* and *Alex_last_fc1*, this MLP was fed with the features from the third max-pooling layer (*pool3*), first fully connected layer (*fc1*) and the third fully connected layer (*fc3*) of the Alexnet respectively [9].

In experiment 4, called *Alex_full*, we re-trained the whole Alexnet with our sonar data, where we also used a single layer adapter network to convert the 1000-way original Alexnet output to a 4-class output. In the next experiment *Alex_first1_last_1adapt*, the architecture is similar to *Alex_full*, but only the layer 1 of the Alexnet and the adapter network was updated during training. In the next two experiments, *Alex_first1_last_3adap*t and *Alex_first2_last_3adapt,* the single layer adaptor was replaced by a three layer adaptor network of the same dimensions as in experiments 1-3. In these two experiments, we updated the first layer and first two layers of the Alexnet respectively in addition to the adaptor network.

Experiments 8 and 9, named *Vgg16_last_fc1* and *Vgg16_last_fc3* are similar to the experiments 2 and 3, except that VGG16 is used instead of Alexnet for calculating features.

In experiment 10, tagged *Vgg16_full*, we re-trained the whole VGG16 with a single layer, 1000-to-4-way adaptor network at the output. Experiment 11, *Vgg16_first2_last_3adapt* is identical to experiment 7, except that Vgg16 is used instead of Alexnet.

In experiments 12 and 13, we fully trained new networks from the ground up. Experiment *Newnet_7layer* used a network which has 4 convolutional layers and 3 fully connected layers. Starting from layer 1, dimensions of the convolutional layers are as follows: 17x17x1-20, 5x5x20-30, 3x3x30-26 and 3x3x26-100, where *wxhxd-n* means *n* filters of width *w*, height *h* and depth *d*. There are max-pooling layers in between all the convolutional layers, implemented using 3x3 filters. The three fully connected layers have 500, 100 and 4 neurons respectively and the first layer accepts an input of dimension 400.

The network structure in *Newnet_8layer* is similar to *Newnet_7layer* except that it has 5 convolutional layers of the dimensions 12x12x1-20, 6x6x20-20, 5x5x20-30, 3x3x30-26, 3x3x26-100.

All networks in the experiments have *rectified linear unit* (*ReLu*) activation functions [4] except for the output layer where a *softmax* nonlinearity is used [12]. We fine-tuned/trained all our networks using the *cross entropy loss* and the *back-propagation* based *gradient descent* with a *momentum* term [12]. Parameters were initialized to random values with the algorithm in [13] unless they are available from pre-trained networks. Test-set was used as the validation set and each training operation consists of either 200 or 150 epochs depending on the size of the task.

All experiments were implemented using *TensorFlow* [14] and run on a computer with one NVIDIA GTX 1080 GPU card.

5. RESULTS

Table 1 shows main results of the experiments.

No	Experiment Tag	#Trainable Parameters [millions]	Time per training epoch [sec]	Training epochs until optimum test error	Test set accuracy [%]
1	*Alex_last_pool3*	4.66	7.1	25	69.62
2	*Alex_last_fc1*	2.10	5.9	145	72.62
3	*Alex_last_fc3*	0.55	5.2	155	65.62
4	*Alex_full*	62.38	191	8	78.52
5	*Alex_first1_last_1adapt*	0.04	192	139	63.20
6	*Alex_first1_last_3adapt*	0.59	193	99	72.62
7	*Alex_first2_last_3adapt*	1.20	194	79	78.0
8	*Vgg16_last_fc1*	2.10	5.9	165	74.22
9	*Vgg16_last_fc3*	0.55	5.3	159	67.56
10	*Vgg16_full*	138.36	942	9	**84.48**
11	*Vgg16_first2_last_3adapt*	0.59	943	89	77.58
12	*Newnet_7layer*	0.30	57	199	77.70
13	*Newnet_8layer*	0.31	78	179	80.30

Table 1: Experiments and their main results

With experiments 1-3 and 8-9, we can compare the different strategies for fine-tuning the last layers. For both Alexnet and VGG16, features extracted from *fc1* lead to better classification accuracy than features extracted from *fc3*. This indicates that features closer to the output layer of the pre-trained net are less suited for fine-tuning. A possible reason is that such features are more tightly bound to the original task and hence become less generic. This pattern however does not appear across the experiments 1 and 2, even if *fc1* is closer to the output than *pool3*. An explanation might be the larger size of the fine-tuning MLP for experiment 1 (4.66 million parameters), something which leads to less generalization and hence inferior accuracy for the test set. If we compare the performance of the experiment 2 against 8 and experiment 3 against 9 we can observe that VGG16 based features lead to better fine-tuning performance than the Alexnet. This is an interesting result, because VGG16 is a better network than Alexnet for optical images and it appears that this superiority is still valid in the domain of sonar images.

Experiments 4-7 and 10-11 show how the fine-tuning of the layers close to input and full re-training perform for the Alexnet and VGG16. It is immediately clear that full re-training gives the best results over all fine-tuning approaches for both networks. Even though full training is computationally heavier (takes 191 and 942 seconds for experiments 4 and 10 respectively), the networks converge very quickly to a point corresponding to the optimum test set accuracy (under 10 training epochs for both nets). One can also note that even though the full re-training involves adaptation of very large number of parameters, overfitting does not occur and generalization ability is preserved. This means that the information these networks extracted during training on optical images is still useful in the domain of sonar images. Results of the experiments 5-7 and 11 show that fine-tuning of the layers close to the input is the second best approach to fine-tuning. However, if the number of trainable parameters is very low (experiment 5), the network is harder to be fine-tuned (takes a longer time to convergence) and the performance is weaker. This can be resolved easily by adding more trainable layers (experiments 6,7,11). The disadvantage of this approach is that it is equally computationally intensive as full re-training while results do not appear to be equally good.

In experiments 12 and 13, we see that fully-trained, relatively small networks (with number of parameters in the range of 300 000) compared to the typical DCNNs like Alexnet or VGG16 can achieve very good results. In fact, this approach gives better test set accuracy than Alexnet with any fine-tuning strategy. Because of the smaller size, these networks occupy less GPU memory and take less time to complete a training iteration. But it takes a while before the network converges to an acceptable optimum. Smaller size of the networks can also lead to better generalization and superior test-set accuracy. These experiments also show that a deeper network gives better performance (i.e. experiment 13 has better accuracy than experiment 12). The challenge with full design and training of such networks is however that a significant amount of experimentation is required to select an architecture and a set of hyper-parameters which can give good enough results.

3.2. Receiver Operating Characteristic (ROC) curves

In order to gain a deeper insight into the performance of different classifiers, we consider four detectors (cylinder, manta, other-mines and non-mines) that make use of the classifier output for detection. We show test-set ROC curves for these detectors for a selected set of experiments in Figure 2. As seen from the figure, the ROC curves are generally in agreement with the classification accuracies. For all the classes, experiment 2 (*Vgg16_full*) is clearly the best. In the case of manta mine detection however, it gets a close competition from experiment 13 (*Newnet_8layer*). Experiments on fine-tuning of last layers, experiment 2 (*Alex_last_fc1*) and experiment 8 (*Vgg16_last_fc1*) lag behind the other approaches. However, in the case of detection of non-mine objects their performance gap with other approaches is narrower. Experiment 13 (*Newnet_8layer*) and experiment 4 (*Alex_full*) have a close competition with each other. In the case of cylinder mine detection *Alex_full* is slightly better, but in detection of manta mines *Newnet_8layer* is clearly ahead. It can also be observed that in general, detection of objects of regular mine classes such as a cylinder or manta has higher areas under ROC-curves than for detection of objects of a more vaguely defined class such as non-mine objects.

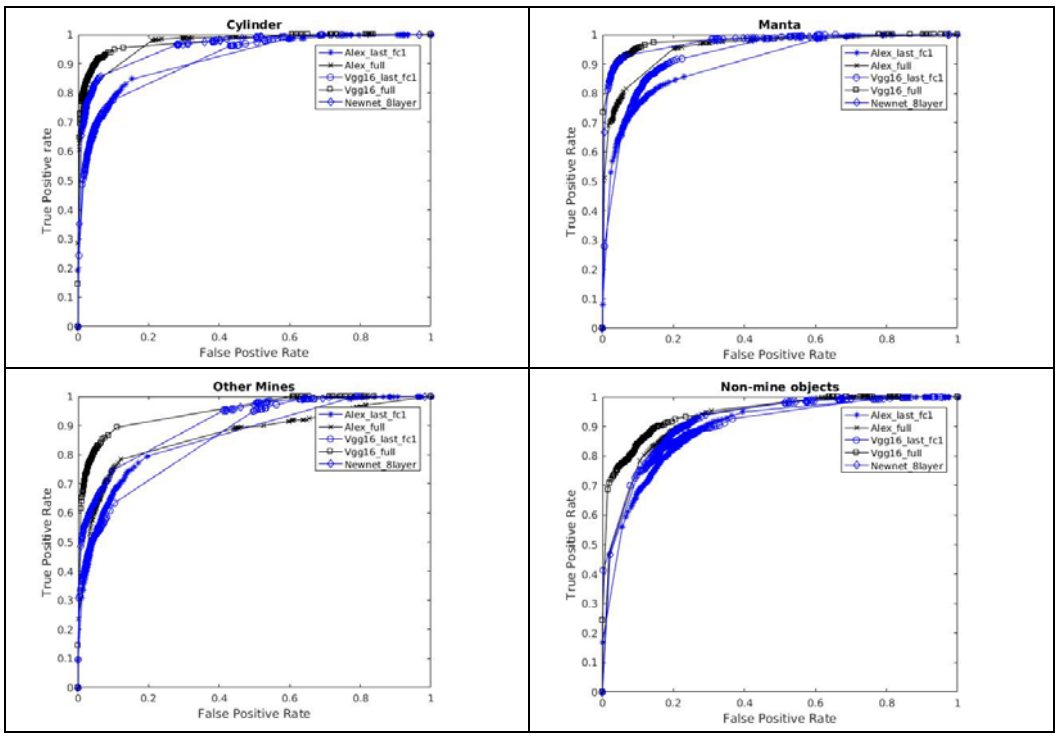

Fig.2: ROC Curves for different target classes

6. CONCLUDING REMARKS

This work demonstrates that DCNNs pre-trained with optical images can successfully be fine-tuned using sonar images. Of all approaches, the most attractive one is fine–tuning the whole network, that is re-training the whole network on sonar data with the pre-trained network parameters as initial values. This procedure appears to have a built-in regularization effect and information extracted from optical images in pre-training is transferred effectively to the domain of sonar images resulting in an excellent test-set accuracy. Fine-tuning of the layers close to the input also has the potential to deliver performances comparable to that of fully trained networks of manageable sizes. However, that approach is computationally intensive and offers no significant advantage in practise other than a way to limit overfitting. The third approach for fine-tuning, namely re-training the last layers does not appear to be promising. But such an approach still leads to reasonable results, indicating that feature extraction strategies learned on optical images are also useful for discriminating sonar images. Furthermore, we conclude that full training of networks whose sizes match the amount of available training data is also a viable approach, even though full training did not surpass the best fine-tuning accuracy in our experiments. It is still inconclusive, however, whether full training can yield better classification accuracy than the best accuracy that fine-tuning can generate.

7. ACKNOWLEDGEMENTS

We thank the Royal Norwegian Navy and Kongsberg Maritime AS for kindly contributing to the MCM database used in this study.

REFERENCES

[1] **J. Stack,** Automation for underwater mine recognition: current trends and future strategy", *In Proceedings of SPIE 8017, Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XVI*, Orlando, USA, 8017, 2011.

[2] **Ø. Midtgaard and H. Midelfart,** SITAR: A modular system for automatic target recognition in side looking sonar imagery, In *Proceedings of the 2012 International Conference on Detection and Classification of Underwater Targets*, pp. 268-270, 2014.

[3] **H. Midelfart and Ø. Midtgaard,** Robust template matching for object classification, In *Proceedings of the fourth International Conference and Exhibition on Underwater Acoustic Measurements: Technologies and Results*, Kos, Greece, pp. 2011.

[4] **A. Krizhevsky, I. Sutskever, and G. Hinton**, Imagenet classification with deep convolutional neural networks, In *Advances in Neural Information Processing Systems*, pp. 1097-1105, 2012.

[5] **K. Simonyan and A. Zisserman,** Very deep convolutional networks for large-scale image recognition, *CoRR*, abs/1409.1556, 2014.

[6] **J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darre,** DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition, In *Proceedings of the 31st International Conference on International Conference on Machine Learning (ICML)*, Beijing, China, 32, pp. I647-I655,2014.

[7] **J. Yosinski, J. Clune, Y. Bengio, and H. Lipson,** How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pp.3320-3328, 2014.

[8] **H. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D.Mollura and R. M. Summers,** Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning, *IEEE Transactions on Medical Imaging*, 35 (5), pp. 1285-1298, 2016.

[9] **N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang,** Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning?, *IEEE Transactions on Medical Imaging*, 35(5), pp. 1299-1310, 2016.

[10] **D. Williams**, Underwater target classification in synthetic aperture sonar imagery using deep convolutional neural networks, In *Proceedings of International Conference on Pattern Recognition (ICPR)*, Cancun,Mexico, pp. 2497-2502, 2016.

[11] **O. Russakovsky et al.,** ImageNet Large Scale Visual Recognition Challenge, *CoRR*, abs/1409.0575, 2014.

[12] **I. Goodfellow, Y. Bengio and A. Courville,** *Deep Learning*, MIT Press, 2016.

[13] **X. Glorot and Y. Bengio,** Understanding the difficulty of training deep feedforward neural networks, In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, Sardinia, Italy, 9, pp. 249-256, 2010.

[14] **M. Abadi et al.,** Tensoflow: Large scale machine learning on heterogeneous distributed systems, *CoRR*, abs/1603.04467, 2016.