LOW-COMPLEXITY FIXED-POINT CONVOLUTIONAL NEURAL NETWORKS FOR AUTOMATIC TARGET RECOGNITION

Hassan Dbouk*, *Hanfei Geng**, *Craig M. Vineyard*†, *and Naresh R. Shanbhag**

*Dept. of Electrical and Computer Engineering, University of Illinois at Urbana Champaign
†Sandia National Laboratories

ABSTRACT

There has been growing interest in developing neural network based automatic target recognition systems for synthetic aperture radar applications. However, these networks are typically complex in terms of storage and computation which inhibits their deployment in the field, where such resources are heavily constrained. In order to bring the cost of implementing these networks down, we develop a set of compact network architectures and train them in fixed-point. Our proposed method achieves an overall $984\times$ reduction in terms of storage requirements and $71\times$ reduction in terms of computational complexity compared to state-of-the-art convolutional neural networks for automatic target recognition (ATR), while maintaining a classification accuracy of $> 99\%$ on the MSTAR dataset.

***Index Terms*—** deep learning, neural networks, automatic target recognition, synthetic aperture radar, quantization

1. INTRODUCTION

The design of automatic target recognition (ATR) systems based on synthetic aperture radar (SAR) imagery has been an active area of research for decades. SAR-based ATR systems are usually deployed on airborne vehicles, where always-on detection is required and compute resources are limited. A typical ATR system needs to process large volumes of imagery in two steps: first, it needs to identify regions of interest (ROI) from the original scene and second, it needs to detect whether specific targets are present in the ROI. This imposes strict energy and latency constraints on the ATR algorithm to achieve real-time decision making, as the same detection algorithm needs to be repeated for every ROI, and for every new scene. Moreover, the accuracy of ATR systems cannot be compromised, as they are often involved in mission critical applications. Therefore, the design of high accuracy low complexity ATR systems is of utmost importance.

Recently, deep neural networks (DNNs) have been successfully deployed in various computer vision tasks [1,2] due to their ability to learn complex representations from data. The performance of convolutional neural networks (CNNs) for SAR ATR has also been well studied [3–8]. However these, with the exception of [8], focus on maximizing classification accuracy without considering network complexity. CNNs require millions of parameters and up to billions of operations to generate a single inference, making them unsuitable for latency- and energy-constrained applications such as SAR ATR. The authors of [8] propose using spiking neural networks, and utilize IBM's TrueNorth chip [9] to accelerate such networks. However, as a consequence of the architectural limitations in their approach, the best model in [8] achieves a classification accuracy of 95.67% on the standard MSTAR dataset.

Our work addresses the problem of designing low complexity neural networks for ATR using networks with a *compact architecture* and *fixed-point quantization* that can be trained to achieve $> 99\%$ classification accuracy on the MSTAR dataset.

1.1. Contributions

In this paper, we present the design of low complexity networks for ATR in a systematic fashion, with minimal loss in classification accuracy. We impose an accuracy constraint of achieving a classification accuracy of at least 99% on the standard MSTAR dataset. Our contributions are: 1) we design compact neural networks for SAR-based ATR using *pointwise* convolutions [2]; 2) we further reduce the complexity by using *reduced precision fixed-point* and study the impact of varying bit precision on classification accuracy; 3) we quantify the efficiency of our designs by adopting first order complexity measures: *computational* and *representational* costs as suggested in [10,11]; and 4) we benchmark our implementation against the state-of-the-art (SOTA) neural networks for ATR on the standard MSTAR dataset. To ensure a fair comparison, we train networks using the *same training setup*.

The rest of the paper is organized as follows: Section 2 presents the necessary background material, Section 3 explains our proposed network design methodology. Section 4 demonstrates the results of our method. Section 5 concludes this paper.

This work is supported by a grant from Sandia National Laboratories and DARPA FRANC Program.

2. BACKGROUND

2.1. Depthwise Separable Convolutional Neural Networks

In CNNs, convolutional layers serve as feature extractors. The standard 2D convolution operation (Conv), parameterized by $(K \times K \times C \times M)$, convolves an input feature map $x \in \mathbb{R}^{C\times H\times W}$, where C is the number of input channels each with dimension $H \times W$, with a set of M kernels $\{w_m\}_{m=1}^{M}$, where $w_m \in \mathbb{R}^{C\times K\times K}$, to generate an output feature map $y \in \mathbb{R}^{M\times E\times F}$, with M channels each with dimension $E \times F$. To implement convolutional layers, one needs to be able to store MCK^2 parameters and perform $EFMCK^2$ multiply and accumulate (MAC) operations.

A depthwise separable convolution (DWS-Conv) [2] factorizes convolutions to reduce the complexity of standard 2D convolutions. As explained in Figure 1, DW-Conv convolves each input channel with a 2D kernel separately, and PW-Conv is a standard 2D convolution with 1×1 kernels. Therefore, depthwise separable convolutions require $C(K^2 + M)$ parameters and $EFC(K^2+M)$ MACs which yields a $\frac{1}{M} + \frac{1}{K^2}$ reduction in parameters and MACs compared to standard 2D convolutions.

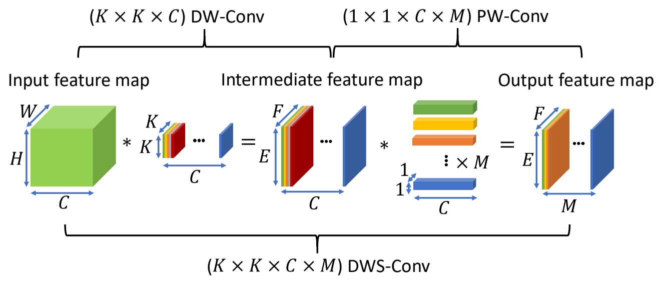
$(K \times K \times C)$ DW-Conv
$(1 \times 1 \times C \times M)$ PW-Conv
Input feature map
Intermediate feature map
Output feature map
W
H
C
K
K
C
F
E
C
1
1
C
$\times M$
F
E
M
$(K \times K \times C \times M)$ DWS-Conv

Fig. 1. A depthwise separable convolution (DWS-Conv) [2] consists of a depthwise convolution (DW-Conv) and a pointwise convolution (PW-Conv).

2.2. Complexity Metrics

The *computational cost* [10, 11] of a L-layer neural network is given by:

$$\mathcal{C}_C = \sum_{l=1}^{L} N_l D_l B_{W,l} B_{A,l} \tag{1}$$

where N_l denotes the number of D_l dimensional dot products in layer l with $B_{W,l}$ and $B_{A,l}$ being the weights and activations precisions, respectively. This cost measures the number of 1-b full adders (FAs) needed to implement the multiplications required for a given network. We assume that floating-point models have 32-b representation (single precision). We ignore the cost of exponent addition in floating-point multiplication to favor floating-point since it greatly simplifies the expression for $\mathcal{C}_C$.

The *representational cost* [10, 11] of a L-layer neural network is given by:

$$\mathcal{C}_R = \sum_{l=1}^{L} \Big(|W_l| B_{W,l} + |A_l| B_{A,l} \Big) \tag{2}$$

where $|W_l|$ and $|A_l|$ are the number of elements in the weight and activation tensors in layer l, respectively. This cost captures the number of bits needed to represent both the weights and activations of a model.

The simplicity of these metrics allows us to compare different neural networks from their precision assignment.

3. PROPOSED LOW COMPLEXITY NEURAL NETWORKS

3.1. Compact Network Architecture

We propose a family of compact network architectures parametrized by f that controls the width i.e., the number of 3D filter kernels, of each layer. Except for the first layer, we employ standard Conv layers with $K = 2$ to reduce the number of operations and storage requirements of these layers. The use of BatchNorm (BN) layers [12] allows us to train smaller models for the same classification accuracy. Furthermore, we observe empirically that the first layer requires large kernel sizes to maintain the accuracy constraint ($> 99\%$). Therefore, we choose a kernel size of $K = 9$.

For the first layer, the use of a standard Conv with f kernels $(9 \times 9 \times 1 \times f)$ will dominate ($\approx 99\%$) the computational complexity. Using DWS-Conv (see Figure 1) instead will reduce the complexity (number of operations) by $13\times - 23\times$ for the values of f used in this work. However, we observe empirically that the resultant network architectures fail to achieve an accuracy $> 99\%$. To address this issue, we propose using a *hybrid* convolutional layer composed of a standard Conv with *five* kernels $(9 \times 9 \times 1 \times 5)$ followed by a PW-Conv layer with f kernels $(1 \times 1 \times 5 \times f)$ resulting in a complexity reduction of $2.6\times - 4.6\times$ while achieving an accuracy $> 99\%$, as shown in Section 4. We also apply a central crop to the MSTAR dataset to reduce the frame size from 128×128 to 64×64 thereby further reducing the complexity of the first layer by a factor of $4.6\times$. The details of the network architecture are captured in Table 1.

3.2. Fixed-point Quantization via Training

A quantized model has much less storage requirements compared to a floating-point (FL) model. Moreover, fixed-point (FX) multipliers/adders are much more area/latency/power efficient than their FL counterparts. A popular method for designing FX networks is via training [13], where weights and activations are quantized in the forward path (inference) and full precision weights are used for weight updates in the backward path (training).

Layer Type	Layer Shape	Input Shape
Conv	$9 \times 9 \times 1 \times 5$	$64 \times 64 \times 1$
BN	5	$56 \times 56 \times 5$
ReLU	$-$	$56 \times 56 \times 5$
PW-Conv	$1 \times 1 \times 5 \times f$	$56 \times 56 \times 5$
BN	f	$56 \times 56 \times f$
ReLU	$-$	$56 \times 56 \times f$
MaxPool	8×8	$56 \times 56 \times f$
Conv	$2 \times 2 \times f \times 2f$	$7 \times 7 \times f$
BN	$2f$	$6 \times 6 \times 2f$
ReLU	$-$	$6 \times 6 \times 2f$
MaxPool	2×2	$6 \times 6 \times 2f$
Conv	$2 \times 2 \times 2f \times 4f$	$3 \times 3 \times 2f$
BN	$4f$	$2 \times 2 \times 4f$
ReLU	$-$	$2 \times 2 \times 4f$
Conv	$2 \times 2 \times 4f \times 10$	$2 \times 2 \times 4f$
BN	10	$1 \times 1 \times 10$
ReLU	$-$	$1 \times 1 \times 10$
FC	10×10	$1 \times 1 \times 10$
Softmax	$-$	$1 \times 1 \times 10$

Table 1. The proposed compact network architecture for SAR ATR. Parameter f controls the layer width and hence the network complexity. A Conv layer is parameterized by $(K \times K \times C \times M)$.

The main challenge in training quantized networks is in back-propagating the gradients of the quantized parameters through the non-differentiable quantization function. One simple yet successful trick to overcome this challenge is approximating the gradient using a straight through estimator (STE) [14]. The STE can be used to compute the gradient of the loss function $\mathcal{L}$ with respect to the full precision parameter x:

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial x_q} \times \frac{\partial x_q}{\partial x} \approx \frac{\partial \mathcal{L}}{\partial x_q} \times \mathbf{1}_{\{c_1 \leq x \leq c_2\}} \tag{3}$$

where $\mathbf{1}_{\{\}}$ is the indicator function, x_q is the quantized parameter, and c_1 and c_2 are clipping values needed for quantizing x. For the case of a signed quantizer needed for weights, we use $-c_1 = c_2 = c$ for some $c > 0$. As for quantizing unsigned values (needed for activations) we use $c_1 = 0$ and $c_2 = c$ for some $c > 0$. For quantizing weights in the forward path, for every layer $l \in [L]$ we use a clipping value:

$$c_{W,l} = \max(|W_l|) \tag{4}$$

thus eliminating any clipping noise. Quantizing activations is a bit more involved than quantizing weights, because the maximum absolute value can change arbitrarily from one mini-batch to another during training. To determine a suitable clipping value for activations, we leverage the properties of BN layers and the central limit theorem (CLT). For a given layer l, we clip the *post* BN activations x_l using:

$$c_{A,l} = \max_{i \in [C_l]} (\beta_l^{(i)} + 3\gamma_l^{(i)}) \tag{5}$$

where C_l is the number of channels in the activation tensor x_l and $(\beta_l^{(i)}, \gamma_l^{(i)})$ are learnable per-channel shift and scale parameters of BN. It is well known that the distribution of $x_l^{(i)}$ can be approximated as a normal distribution $\mathcal{N}(\beta_l^{(i)}, (\gamma_l^{(i)})^2)$ [12]. Therefore, using this approximation and the 3σ rule, one can show that the choice of $c_{A,l}$ in (5) guarantees:

$$\Pr\{x_l \leq c_{A,l}\} \geq 0.99865 \tag{6}$$

Note that having a fixed clipping value $c_{A,l}$ for all channels is crucial in order to ensure that the dot product operations can be implemented in FX.

Vehicle Type	Training Images (17 degrees)	Testing Images (15 degrees)
2S1	299	274
BMP2	698	587
BRDM2	298	274
BTR60	256	195
BTR70	233	196
D7	299	274
T62	299	273
T72	691	582
ZIL131	299	274
ZSU234	299	274

Table 2. MSTAR training and test sets.

4. RESULTS

4.1. Experimental Setup

To benchmark our networks, we use the popular and publicly available moving and stationary target acquisition and recognition (MSTAR) dataset, which is the standard dataset for benchmarking SAR ATR systems. Table 2 summarizes the dataset. We use the commonly used train/test split as in [4], that is images collected at 17 degrees depression angles are used for training, and images collected at 15 degrees depression angles are used for testing. The dataset consists of grayscale images of size 128×128 distributed amongst 10 classes. All the networks were trained using vanilla SGD for 200 epochs from scratch. We used an initial learning rate of 0.01 which was divided by 10 every 50 epochs. We chose a batchsize of 140. Weight regularization was employed with a value of 9×10^{-4}. We also use data augmentation in our training setup. Specifically, we use two separate stages of elastic distortion (ED) [4] to effectively triple our training set. Both stages use Guassian kernels with standard deviations of 3 with scales set to 10 (first stage) and 20 (second stage).

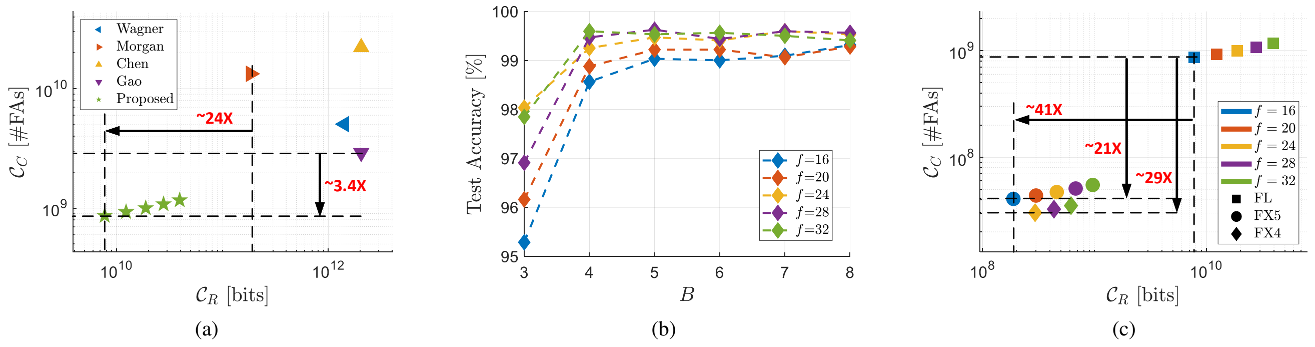

Fig. 2. Plots showing: (a) $\mathcal{C}_C$ vs $\mathcal{C}_R$ for the proposed FL compact network architectures and state-of-the-art networks, (b) the test accuracy of the proposed FX networks vs bit precision, and (c) $\mathcal{C}_C$ vs $\mathcal{C}_R$ for the FX networks and the FL networks.

4.2. Floating-point (FL) Results

We trained five FL models using our proposed compact architecture with $f \in \{16, 20, 24, 28, 32\}$ denoted by FL-f (see Table 1). We also reproduced the results of five SOTA ATR convolutional neural networks from the literature [3–7] using the same setup to ensure a fair comparison. All ten FL models achieve a classification accuracy $> 99\%$ on the MSTAR dataset, and Table 3 summarizes their performance. Figure 2a demonstrates the effectiveness (in terms of $\mathcal{C}_C$ and $\mathcal{C}_R$) of the proposed FL networks. The most compact network (FL-16) achieves a $24\times$ reduction in $\mathcal{C}_R$ and $3.4\times$ reduction in $\mathcal{C}_C$ compared to SOTA while still meeting the accuracy constraint. Note that the network in [6] is not shown in Figure 2a because it exhibits extremely large values for $\mathcal{C}_R$ ($\sim 10^{16}$) and $\mathcal{C}_C$ ($\sim 10^{12}$).

Network Architecture	Input Crop Size	Test Accuracy [%]
Prior Art		
Morgan [3]	128×128	99.72
Wagner [4]	64×64	99.56
Gao [5]	64×64	99.31
Ding [6]	128×128	99.34
Chen [7]	88×88	99.66
Proposed Networks		
FL-16	64×64	99.38
FL-20	64×64	99.47
FL-24	64×64	99.41
FL-28	64×64	99.56
FL-32	64×64	99.66

Table 3. Summary of the performance (in terms of test accuracy on the MSTAR dataset) of recently published and proposed CNNs for SAR ATR. All networks achieve a test accuracy $> 99\%$. FL-x denotes a floating-point model with $f = x$.

4.3. Fixed-point (FX) Results

In order to train fixed-point (FX) versions of our compact networks, we need to determine the bit precisions of all weights ($B_{W,l}$) and activations ($B_{A,l}$) $\forall l \in [L]$. A brute force search is computationally expensive, since the search space is huge and the training of each network takes much time even on GPUs. To simplify the search, we fix $B_{W,l} = B_{A,l} = B$, sweep B from 3 to 8, and for each value of B train the five proposed networks denoted by FXB-f with $f \in \{16, 20, 24, 28, 32\}$.

Figure 2b shows that for $B \geq 5$, all models achieve a test accuracy $> 99\%$ (our stated accuracy constraint). For $B = 4$, only models with $f \in \{24, 28, 32\}$ satisfy this constraint. The test accuracy degrades noticeably for $B = 3$.

Figure 2c shows that FX5-16 is the best model in terms of $\mathcal{C}_R$ achieving a $41\times$ reduction compared to the lowest complexity FL-16. As for $\mathcal{C}_C$, FX4-24 is the best model achieving a $29\times$ reduction compared to the lowest complexity FL-16.

Thus FX5-16 achieves a total $24 \times 41 = 984\times$ reduction in $\mathcal{C}_R$ and $3.4\times 21 \approx 71\times$ reduction in $\mathcal{C}_C$ compared to SOTA CNNs for ATR, while maintaining a test accuracy $> 99\%$ on the MSTAR dataset.

5. CONCLUSION

While CNNs have been studied to a great extent in various image classifications tasks, not much work has been done for ATR applications and that consider their implementation complexity. We have presented a set of compact CNN architectures for ATR coupled with a fixed-point training methodology to reduce the numerical precision required for weights and activations. The proposed reduced complexity networks are suitable for resource-constrained applications such as SAR ATR, and exhibit an inherent accuracy-complexity tradeoff via tuning the design parameters (f, B).

Future work includes mapping the proposed networks onto efficient hardware architectures, such as the deep in-memory architecture [15], to further facilitate their deployment.

6. REFERENCES

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[2] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[3] David AE Morgan, "Deep convolutional neural networks for ATR from SAR imagery," in *Algorithms for Synthetic Aperture Radar Imagery XXII*. International Society for Optics and Photonics, 2015, vol. 9475, p. 94750F.

[4] Simon A Wagner, "SAR ATR by a combination of convolutional neural network and support vector machines," *IEEE transactions on Aerospace and Electronic Systems*, vol. 52, no. 6, pp. 2861–2872, 2016.

[5] Fei Gao, Teng Huang, Jinping Sun, Jun Wang, Amir Hussain, and Erfu Yang, "A new algorithm for SAR image target recognition based on an improved deep convolutional neural network," *Cognitive Computation*, pp. 1–16, 2018.

[6] Jun Ding, Bo Chen, Hongwei Liu, and Mengyuan Huang, "Convolutional neural network with data augmentation for SAR target recognition," *IEEE Geoscience and remote sensing letters*, vol. 13, no. 3, pp. 364–368, 2016.

[7] Sizhe Chen, Haipeng Wang, Feng Xu, and Ya-Qiu Jin, "Target classification using the deep convolutional networks for SAR images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 8, pp. 4806–4817, 2016.

[8] Megan Renz and Qing Wu, "An energy-efficient embedded implementation for target recognition in SAR imageries," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2017, pp. 1–5.

[9] Steven K Essera, Paul A Merollaa, John V Arthura, Andrew S Cassidya, Rathinakumar Appuswamya, Alexander Andreopoulosa, David J Berga, Jeffrey L McKinstrya, Timothy Melanoa, Davis R Barcha, et al., "Convolutional networks for fast energy-efficient neuromorphic computing," *Proc. Nat. Acad. Sci. USA*, vol. 113, no. 41, pp. 11441–11446, 2016.

[10] Charbel Sakr, Yongjune Kim, and Naresh Shanbhag, "Analytical guarantees on numerical precision of deep neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3007–3016.

[11] Charbel Sakr and Naresh Shanbhag, "Per-tensor fixed-point quantization of the back-propagation algorithm," *arXiv preprint arXiv:1812.11732*, 2018.

[12] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[13] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio, "Binarized neural networks," in *Advances in neural information processing systems*, 2016, pp. 4107–4115.

[14] Yoshua Bengio, Nicholas Léonard, and Aaron Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[15] Mingu Kang, Sujan K Gonugondla, Ameya Patil, and Naresh R Shanbhag, "A multi-functional in-memory inference processor using a standard 6t sram array," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 2, pp. 642–655, 2018.