TOWARDS DISTILLATION OF DEEP NEURAL NETWORKS FOR SATELLITE ON-BOARD IMAGE SEGMENTATION

F. de Vieilleville [1, *], A. Lagrange [1], R. Ruiloba [1], S. May [2]

[1] AGENIUM Space, Toulouse, France – (francois.devieilleville, adrien.lagrange, rosa.ruiloba)@agenium.com
[2] CNES, Centre National d'Etudes Spatiales, France – stephane.may@cnes.fr

KEY WORDS: Deep learning, parameters reduction, ablation study, low rank approximation, distillation

ABSTRACT:

Cubesats platforms expansion increases the need to simplify payloads and to optimize downlink data capabilities. A promising solution is to enhance on-board software, in order to take early decisions, automatically. However, the most efficient methods for data analysis are generally large deep neural networks (DNN) oversized to be loaded and processed on limited hardware capacities of cubesats. To use them, we must reduce the size of DNN while accommodating efficiency in terms of both accuracy and inference cost. In this paper, we propose a distillation method which reduces image segmentation deep neural network's size to fit into on board processors. This method is presented through a ship detection example comparing accuracy and inference costs for several networks.

1. INTRODUCTION

Nowadays, cubesats platforms appear to be an attractive and low-cost tool to acquire image data from outer space. However, downlink data capabilities are the bottleneck of the cubesats platforms. It is thus necessary to reduce the volume of data to downstream either by data compression or by data selection. The compression ratio required for image payloads is too high to be obtained by existing compression methods (Buciluǎ et al., 2006; Frosst and Hinton, 2017; Hinton et al., 2015). On-board, feature extraction based on deep learning (DL) (Greenland et al., 2018) provides, by now, the most efficient solution for upstream data reduction. Moreover, it can provide high-value information directly exploitable. However, the most efficient DL models are usually cumbersome due to ensembling methodologies, e.g. hundreds of millions of parameters in the case of the winners of Airbus Ship Detection challenge proposed on Kaggle[1], and are not compatible with the limited performances of the processors available on board.

Some missions integrate specific intelligence which can decide on storage or scenes acquisition regarding cloud detection, e.g. Forward Looking Imager (Greenland et al., 2018). On-board features extraction allows to decide on board if data should be stored and downlinked or not. It can also directly supply the information requested without downloading the full data image. Data reduction by DL-based feature extraction on board will improve the acquisitions capabilities for a defined bandwidth budget and will optimize data downlink providing useful information (Soukup et al., 2016).

Instead of relying on ground operator, decisions can be made on board, with efficient algorithms. This aims to increase the payload's autonomy, to release burden on ground segments and to reduce costs (since less operators are required) for creating valuable products.

The point here is that all these specific, long, tedious and complicated developments in terms of methodology, software, integration and verification can be replaced by embedded intelligence. For example, improving the reactivity may allow a simple constellation to continuously monitor events and to eventually downlink acquired/processed data when faced with predefined observed activities. Such scenario could also be compatible with satellites being interconnected and allowing downlink for the best positioned satellite on available reception stations. Moreover, the image processing chain on-board can be reprogrammable. The goal of the system design should be to make possible the replacement of the DL components by a new DL network, devoted to other feature extraction required by the mission. Thus, the same cubesat platform could be used for different user-driven missions contributing to optimize mission cost.

However, existing state-of-the-art deep neural networks (DNN) which perform the best on earth observation (EO) tasks are huge networks (several hundred million parameters). They are often aggregates of several models and require a lot of computation power to be run (Bianco et al., 2018). On the other hand, hardware (HW) for computation, in satellites with an image payload, is very limited (Lentaris et al., 2018) due to power consumption restrictions (between 1 and 10 W against 250 W for a NVIDIA Titan X) and dissipation problems. For on-board processing, the reduction of the computational burden linked to inference is the priority (Abdelouahab et al., 2018; Qi et al., 2018).

Methods exist to reduce the number of operations of these huge high-performance networks, but the achieved reduction factors with most techniques, on fully convolutional networks, do not exceed a x4, x9 factor (Han et al., 2015a). Consequently, the best DNN cannot be used with a restricted computation power and there is no other choice but to use less reliable, less efficient and smaller networks. Although these smaller networks (<5 million parameters) fit in existing computation payload, they do not bring high accuracy making them potentially useless depending on the mission requirements. For example, the works of the Aiko society have focused on a library called Mirage AI aiming at improving on-board autonomy and, more generally, mission planning. Their work has focused on the reuse of simple or highly

* Corresponding author
[1] e.g. 6th place candidate : https://www.kaggle.com/c/airbus-ship-detection/discussion/71782

efficient networks in terms of performance/number of operations, e.g. ship detection with fast detectors (SSD (Liu et al., 2016), YOLO (Redmon and Farhadi, 2018)) followed by a classification network (MobileNet, Howard et al., 2017). In this work, authors display a total size of 15MB for their networks with inference times of less than 1 second per image, the full image size and the final accuracy being not known.

One strategy to alleviate this issue is to try to extract the meaningful parts of these large and complex high-performance DNNs, possibly aggregates or ensemble models, using a teacher/student training method called distillation approach. This approach allows us to take into account what has been learned by the huge teacher model as mapping to vectors of real values and to approximate this mapping as best as we can. The interesting part is that this approach allows small architectures to reach rapidly much higher performances than conventional training (Hinton et al., 2015), even though a state-of-the-art training procedure is used such as augmentation, recent optimizer, learning rate scheduling and policies, stratification on data, batch accumulation, etc. Moreover, this approach being a first step, the other classic simplification methods (pruning, low rank approximation, quantization, weight sharing, …) can still be used. Cumulating the reduction from this approach and classic methods, it is possible to reach high enough reduction rates, e.g. (Polino et al. 2018), to bring high performances networks on board. This paper aims to demonstrate it in the specific context of image segmentation of remote sensing images.

The objective of this study is to reduce the size, in terms of number of parameters, of a given high performance network. In order to fit on cubesats processing payloads, the aim is to produce a new network with around 1 million free parameters. This size should fit in mid capacities COTS (Commercial Off-The-Shelf) and available radiation tolerant HW (Lentaris et al., 2018) used on board the satellites. This objective should be reached with a minimum accuracy loss, a maximum loss of 5% of the performances is the target of the study.

The rest of the article is organized as follows. Section 2 presents a synthetic literature review about the various approaches to reduce a DNN. In Section 3, the target segmentation task and the developed distillation method are presented. Then, experimental results are provided in Section 4 showing that distilled models are significantly smaller at the cost of very limited drop of performances. Finally, Section 5 concludes the paper and opens some perspectives to this work.

2. RELATED WORK

This section introduces the major approaches for DNN reduction in order to outline the specificities and interest of the proposed distillation method.

2.1 Pruning methods

Pruning methods have been initially developed to reduce overfitting in fully connected networks. These approaches aim at deleting connections between neurons in order to simplify the network. It is equivalent to setting a subset of the network weights to 0. There are several ways to select the weights to be nullified. The process can be based on complex optimization problem such as in (Hassibi et al., 1993), where authors use Taylor series approximation of the cost function to identify the optimal weights to delete. Other works have proposed to base the selection on heuristic criteria involving thresholding of the values. Using such methods, (Han et al., 2015b) have optimized both convolutional and fully connected layers of networks based on AlexNet (Krizhevsky et al., 2012) or VGG-16 (Simonyan and Zisserman, 2014) and obtained a reduction of the number of weights of a factor 10-12 and a reduction of processing time of a factor 3-5. These methods are very efficient to compress fully connected layers specifically, even though it means using sparse DNN which generally require the use of specialized hardware to handle sparse computation, according to the authors.

Regarding convolutional layers, pruning can take several forms from deleting complete feature maps to deleting only one of the filters producing a given feature map (Li et al., 2016; Molchanov et al., 2016). The former is, for example, explored by (Li et al., 2016) where authors delete feature maps produced by filters of low L1-norm. Using a complex reduction scheme, Li et al. succeed to reduce the computational burden by 30% with a VGG-16 network (Simonyan and Zisserman, 2014) and a ResNet network (He et al., 2016). In a more general case, the reduction factor of the number of parameters in convolutional layers is closer to 2-4.

Overall, this family of methods gives interesting results but does not allow to modify/simplify the architecture of the considered networks and is limited in terms of reduction capacity. In operational context, pruning is generally complemented with other compression methods such as code book, weight sharing or Huffman coding (Han et al., 2015a). However, these additional optimizations can produce a compressed version of the network but does not reduce the number of necessary computational operations. They can even create an additional cost during inference since it is necessary to decompress the network to use it. Finally, it is also generally necessary to re-train the network after the pruning process.

Pruning methods have already been used to produce DNNs to be used in processing on-board the satellites. In the Euclid mission (Stivaktakis et al., 2019), the redshift estimation relies on a 1D CNN (Convolutional Neural Network) based on a LeNet network (LeCun et al., 1998). To reduce the memory footprint, the authors have used several methods: use single precision, prune the weights values (below a given threshold) to keep 30% of the coefficients, group weights in centroids, store them into a code book and use a smart clustering weight method. In the end, the memory footprint was reduced by a factor x54, with an increase in error rate of 0.8%. Most of the compression was done on the fully connected layers and especially the final layers. These final layers were modified in order to transform the regression problem into a classification problem with 1,000 classes. Let us note that pruning methods generally behave better in conjunction with retraining.

2.2 Low rank approximation

Pruning methods work extremely well for fully connected layers but often do not exceed 50% weight reduction for convolutional layers. Here, to reduce both computation and weights, some authors use low rank approximation on the convolutional layers. These approaches see the convolution blocks as 4D tensors and try to find low-rank approximations (for given specific norms) for them. The approximations can be found using SVD (Single Value Decomposition) like-decompositions, and some of their generalisations, or using more explicit representations. Several papers (Denton et al., 2014; Jaderberg et al., 2014; Tai et al., 2016) showed that it is possible to speedup forward computation by a factor of 2 without having to re-train the model and to reduce the number of operations by a factor of 2 in convolutional layers. Overall, the low rank constraint is an efficient regularization

technique, providing state-of-the-art accuracies (classification on Imagenet cases) but reduces weight sizes only for convolutional layers with a modest reduction factor compared to other reduction methods.

2.3 Other optimization methods

Many other methods were proposed to reduce DNN. Most of them focus mainly on the code optimization aspect. For example, quantification of weights can be performed to go from variables coded in single-precision floating-point format (float32) to variables coded in signed integer format (int8) reducing by a factor of 4 the memory footprint and opening the possibility to implement code optimization procedure such as SIMD instructions (Single Instruction Multiple Data, see (Vanhoucke et al., 2011)). Moreover, further benefits can be expected in terms of execution simplification and computing time in FPGAs (Field Programmable Gate Arrays) since int8 instructions are much simpler to implement in terms of logic gate numbers.

To go further, it was also proposed to recourse to weight sharing as performed after pruning step in previous section 2.1. This approach, proposed by (Han et al., 2015a), introduces the idea of using tabulated values that allow us to encode the high-precision values using table indices with less memory weight.

As explained before, these methods focusing on code/computation optimization can generally be used in addition to other methods focusing more on the optimization of the network architecture, with the limitations already presented (see section 2.1). It is possible to use most of them with the distillation method proposed in this paper. Studying their impact/efficiency is out of the scope of this article which is why they are not discussed in the remaining of this paper.

2.4 Teacher models

The idea of teacher models was introduced by (Buciluǎ et al., 2006). It aims at transferring the knowledge of a big teacher network in a smaller DNN by training the small DNN to predict the output of the teacher model.

This technique, denoted usually by distillation, allows to change network topology (remove layers, bridges, pooling layers) or modify elementary blocks (from convolution to depthwise convolution). It is performed in a teacher/student fashion, where each network has its own topology. Consequently, this method is compatible with both large, possibly aggregate, teacher models and small, compact student models. The aim is to find a small network mimicking the results of a large network. Distillation have already been used in various application frameworks such as acoustic models (Hinton et al., 2015) or tabular data (Tan et al., 2018). In this paper, a distillation process is developed in the specific context of image segmentation as described in the next section.

3. DISTILLATION METHOD

The proposed workflow aims at reducing the number of free parameters drastically. The aim of this paper is to simplify networks reaching very high accuracy, obtained using all the state-of-the-art training strategies without restriction. However, classic methods to reduce neural networks are not sufficient for such huge models. In fact, pruning studies show that a reduction of a factor 2-3 in the number of free parameters in convolutional layers can be achieved without significant accuracy drop (Molchanov et al., 2016) even so, these results are dependent of the involved application and architecture. A similar work can be done on linear subspace approximations (e.g. SVD on matrix) when trying to reduce the number of free parameters in linear operators, like convolution or fully connected blocks. Here, the literature reports reduction factors around 2 or 3 (Lee et al., 2019).

Then, quantization is often reduced to computations in int8 instead of float32. Despite a reduction of the memory footprint by a factor 4, this does not reduce the number of free parameters of the network. However, it does improve the global speed since less elementary operations are required when computing int8 arithmetic (Wei et al., 2019).

The objective is thus to obtain a better reduction using a distillation method, the other simplification techniques being considered complementary.

3.1 Considered workflow

The first step is to train a complex state-of-the-art DNN using the complete available data. This process should use all the conventional methods improving the results such as augmentation, recent optimization strategy, complex learning rate scheduling, etc.

The idea is to benefit from all the complex learning methods available regardless of the cost to obtain the best teacher network. As an example, it is common to find huge models with more than 200 million free parameters to obtain the best performances such as the one developed in Kaggle-like competitions.

In order to fit on cubesats processing payloads, the next step is to produce a new network with around 1 million free parameters as stated in the objective of this paper. The method adopted in this study is to use a distillation process in order to reach this objective. The distillation strategy is very straightforward as illustrated in Figure 1. The smaller student network is trained to predict the output of the teacher network using a weighted MSE loss function defined as follows:

$$MSE = \frac{1}{N}\sum_{n=1}^{N} w_n \|\boldsymbol{y}_n - \hat{\boldsymbol{y}}_n\|_2^2 \qquad (1)$$

where $\hat{\boldsymbol{y}}_n$ stands for the predicted probability vector gathering the probabilities to belong to each of the C classes, $\boldsymbol{y}_n$ for the reference probability vector produced by the teacher, w_n is the weight associated to sample n which depends of the class of the sample (in experiments, 0.9 for 'ship' and 0.1 for 'non-ship'). We do not present the formula with associated weight for each class, but it is straightforward.

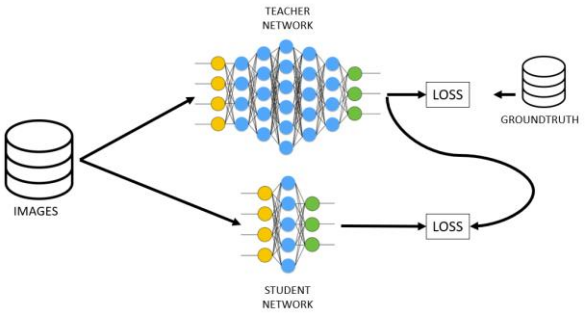

Figure 1. Distillation process

Model	Number of free parameters	Number of feature maps by layers	Training time (1x Nvidia 1080Ti)	F2 score
U-Net vanilla	31 M	64/128/256/512/1024	Long > 1 week	0.733
Ternaus16	37 M	64/128/256/512/512	Long > 1 week	0.827
Ensemble-DNN (1st place Kaggle competitor)	30 M (x10)	NA	NA	0.855
Distilled U-Net 1	4 M	64/128/256/512/1024	32 hours	0.781
Distilled U-Net 2	2 M	64/128/256/512/512	32 hours	0.780
Distilled U-Net 3	1 M	64/128/256/256/256	32 hours	0.774
Distilled U-Net 4	0.5 M	64/128/192/192/192	32 hours	0.757

Table 1. Comparison of architecture and performances of all considered models.

3.2 Architecture modification

As stated earlier, distillation is the only method allowing a change of network architecture at the cost of a special training. It is actually possible to consider any architecture as student network. In this paper, we propose to start from standard architecture such as U-Net network and to implement some architecture modifications aiming at simplifying specific layers of the networks. The substitution of conventional convolutions by depthwise separable convolutions (Chollet, 2017; Howard et al., 2017) is in particular implemented. Such approach reduces the number of free parameters of these layers by a constant factor of 9. This proposition was implemented in the experiments described in Section 4.

After this architecture simplification, the second modification proposed in this paper to reach the desired number of free parameters is the reduction of the number of features used in the output of the intermediate layers. It is a simple way to go from a specific architecture to an architecture with the desired number of parameters.

These modifications are performed on a well-established network architecture, such as U-Net for segmentation task (Ronneberger et al., 2015). Applying these simplifications allows one to reach a desired number of parameters that can be fitted into a given HW platform.

4. EXPERIMENTS

4.1 Task and dataset

The task that was used to test this reduction framework is the Airbus Ship Detection Challenge proposed on Kaggle[2]. The objective is to perform the best ships detection by segmenting remote sensing satellite images. The dataset is composed of more than 200k 3-channels RGB images of 768-by-768 pixels. Unfortunately, no information on the origin and resolution of the images is provided by Airbus. The dataset has very unbalanced classes since ships are small objects and, moreover, only around one quarter of the images contain ships for a total of around 200,000 labelled ships. Figure 2 shows a few examples of images with their associated ground truth segmentation.

A split of the dataset into train and test sets is provided by the authors of the dataset where the ground truth is only available for the training set and the results on the test can be evaluated using Kaggle website. In addition, for the presented experiments the training set was split in a train and validation set with a ratio of 80%/20%.

Figure 2. Dataset Kaggle "Airbus Ship Detection"

4.2 Compared methods

Since the goal of the experiment is to make the most performant ship detection networks suitable for execution on-board the satellites, obtaining these high-performance networks is the first step. In our experiments we started with a 37 million free parameters model based on a Ternaus16 architecture with a VGG16 encoder (Iglovikov et al., 2018). This network achieves high performances in the ADS ship detection competition, *i.e.*, 3% below the top-1 Kaggle score. Thus, it is used as teacher networks in the distillation framework. The distillation aims at reducing the number of parameters of this model.

For the network reduction part, 4 student models have been trained using the proposed distillation framework. All these networks are based on a standard architecture, specifically a U-Net architecture (Ronneberger et al., 2015). The number of parameters of this baseline architecture is gradually reduced from 4 million of free parameters to 0.5 million. Table 1 shows how the number of features, before each reduction of the spatial resolution (maxpooling layer), is gradually reduced to obtain a specific number of parameters. In the end, the network selection

[2] https://kaggle.com/c/airbus-ship-detection

depends both on the hardware capacities and the performance requirements. In terms of memory occupancy, these networks with roughly 0.5 million parameters can fit on most standard FPGA, with approximately a few megabytes of BRAM (Block Random Access Memory (RAM), dual-port RAM module present into the FPGAs). Thus, if we can maintain high performances for such networks, they will be suitable for execution in FPGA on board the satellites and achieve the use case demonstration success. It has been demonstrated by the execution of the inference of the network "Distilled U-Net 4" (0,5 million parameters) with performance 0.757 in F2 -score, in a mid-range FPGA (the one included in the Xilinx ZCU102 test board) very similar of those used on-board the satellites.

These 4 student networks have been trained using a stochastic gradient descent with a momentum of 0.9 and Nesterov acceleration. The learning rate was initialized at 0.01 and gradually reduced using a plateau patience of 80 epochs, *i.e.*, after 80 consecutive epochs without reaching a better minimum of the loss function, the learning rate is reduced by half. Finally, an augmentation procedure has been used during training performing standard transformations randomly on the input images (rotation, zoom, flip, illumination change).

It should be also noted that the results produced by the networks have followed a final postprocessing step. The networks produce probabilities to belong to each class but the evaluation procedure, described in the next section, requires having binary masks as outputs. The first step of the postprocessing is thus a binarization of the output using a threshold value. The thresholding operation corresponds to choosing a specific operating point on ROC (Receiver Operating Characteristic) curve of the model. In the following experiments, several thresholds have been tested and the one corresponding to the best result on the validation set have been kept.

The second and last step of the postprocessing involves performing a morphological opening on the mask (disk of radius 2 as structural element).

The simplified networks performances are evaluated on unseen data and compared to the more accurate but bigger networks: Ensemble-DNN and Ternaus16, and to a U-Net vanilla, the same architecture than student networks but learned by classical training from input images. Ternaus16 is the high-performance teacher network considered for the distillation. Ensemble-DNN is the segmentation model built by aggregating the results of several networks that get the 1st place in Kaggle challenge associated to the dataset. This latter network is not used as teacher since the way the set of results are combined in an ensembling method could alter the distillation process and this matter is out of the scope of this paper.

4.3 Results and discussion

The results obtained with the different models are show in Table 1. It is possible to compare the performances in terms of F2-score for the segmentation task. The F2-score is computed on the class of interest in this application, *i.e.*, the class *ship*. The F2-score is the metric used in the original ADS challenge and, since ground truth of the test set is not available, evaluation on the test set is done through Kaggle website which provides the F2-score as results. It is computed based on the number of true positives (TP), false negatives (FN), and false positives (FP) resulting from comparing the predicted object to all ground truth objects. In particular, a predicted object is considered a "hit" if its intersection over union with a ground truth object is greater than a given threshold. Then, the final score is generated by averaging the F2-score obtained with different thresholds (0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95) where the F2-score is computed with the following formula, with $\beta = 2$,

$$F\beta - score = \frac{(1+\beta^2).TP}{(1+\beta^2).TP + \beta^2.FN + FP} \qquad (2)$$

From the results shown in Table 1, it is possible to draw several conclusions. First of all, we can see that there is indeed a drop of performances between the teacher network Ternaus16 and the student networks. The drop of performance appears to be around 5% for the biggest distilled model (Distilled U-Net 1) and remains stable until reduces for the model of 1 million parameters (Distilled U-Net 3). It seems that below 1 million parameters performances start to decrease more constantly. However, when comparing with U-Net vanilla (version learned from scratch, without distillation) which is the original architecture of the distilled model, it is worth noting that even the performances of the distilled model with 0.5 million parameters (Distilled U-Net 4) gives better results. It proves that the distilled models can benefit from the high-performances teacher model even though if the architecture is entirely different (Ternaus 16 to U-Net).

Figure 3. Performances of U-Net 2 (2 million parameters) when learning is done from scratch.

To further assess the interest of distillation, a second experiment was carried. In this experiment only one network architecture was considered, the architecture used for the Distilled U-Net 2 with 2 million parameters. The aim was to verify the performance reachable with this architecture when directly training it from scratch. As shown in Figure 3, the results showed that it is indeed possible to reach the same accuracy but at the cost of a very time-consuming training process. The distilled version was trained 32 hours when it was necessary to train the "from-scratch" version for 20 days on the same HW (1x NVIDIA 1080 Ti).
Similarly, as a third experimental setup, the lighter models Distilled U-Net 3 and 4, with respectively 1 million and 0.5 million parameters, have been trained from scratch for 32 hours. The results obtained with these two models, referred to as "From scratch" U-Net 3 and 4, are presented in Figure 4. These results have been obtained using the same training procedure described in Section 4.2. They show that, with the same amount of training time and the same architecture, the distilled networks managed to learn faster thanks to the teacher network. More precisely, Figure 4 shows that the results of the "from scratch" models are very dependent of the postprocessing step and especially the operating point chosen on the ROC curve.

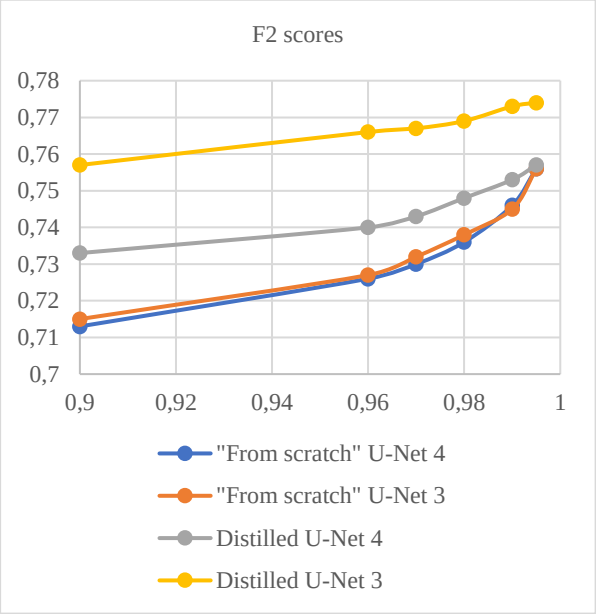

Figure 4. Comparison of performances between networks learned from scratch and distilled networks in function of the binarization threshold.

5. CONCLUSION

Distillation is a generic and re-usable workflow for simplifying DL networks for defining new Earth Observation (EO) products generated on-board the satellites. It can reduce the uptaking cost of innovative deep learning technologies for on-board use.

This paper presented a distillation strategy that allows the user to reduce the size of a network to desired number of parameters in order to fit in a specific hardware.

A use case performing segmentation of EO images was implemented and showed, in particular, that is possible to reduce drastically the number of parameters with a minimal drop of performances. Moreover, the proposed reduction framework has the advantage to be compatible with additional reduction methods (pruning, quantization, etc).

Future works include the study of the influence of additional reduction methods on the performances of the model. Additionally, the presented method is to be tested on other datasets, more specifically additional use cases featuring planes or building detection are being considered.

ACKNOWLEDGEMENTS

The authors acknowledge the support from the CNES, French government Space Agency, and specially the DSO/SI/2A department, under the contract N° 190392/00 “Smart payloads” which allowed to perform a significant part of the work presented in this paper.

REFERENCES

Abdelouahab, K., Pelcat, M., Serot, J., Berry, F., 2018. Accelerating cnn inference on fpgas: A survey. ArXiv Prepr. ArXiv180601683.

Bianco, S., Cadene, R., Celona, L., Napoletano, P., 2018. Benchmark Analysis of Representative Deep Neural Network Architectures. IEEE Access 6, 64270–64277. https://doi.org/10.1109/ACCESS.2018.2877890

Buciluǎ, C., Caruana, R., Niculescu-Mizil, A., 2006. Model compression, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 535–541.

Chollet, F., 2017. Xception: Deep learning with depthwise separable convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1251–1258.

Denton, E.L., Zaremba, W., Bruna, J., LeCun, Y., Fergus, R., 2014. Exploiting linear structure within convolutional networks for efficient evaluation, in: Advances in Neural Information Processing Systems. pp. 1269–1277.

Frosst, N., Hinton, G., 2017. Distilling a Neural Network Into a Soft Decision Tree. ArXiv171109784 Cs Stat.

Greenland, S., Ireland, M., Kobayashi, C., Mendham, P., Post, M., White, D., 2018. Development of a minaturised forwards looking imager using deep learning for responsive operations. ESA.

Han, S., Mao, H., Dally, W.J., 2015a. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. ArXiv Prepr. ArXiv151000149.

Han, S., Pool, J., Tran, J., Dally, W., 2015b. Learning both weights and connections for efficient neural network, in: Advances in Neural Information Processing Systems. pp. 1135–1143.

Hassibi, B., Stork, D.G., Wolff, G.J., 1993. Optimal brain surgeon and general network pruning, in IEEE International Conference on Neural Networks. IEEE, pp. 293–299.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778.

Hinton, G., Vinyals, O., Dean, J., 2015. Distilling the Knowledge in a Neural Network. ArXiv150302531 Cs Stat.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. ArXiv Prepr. ArXiv170404861.

Iglovikov, V., Seferbekov, S.S., Buslaev, A., Shvets, A., 2018. TernausNetV2: Fully Convolutional Network for Instance Segmentation., in: CVPR Workshops. pp. 233–237.

Jaderberg, M., Vedaldi, A., Zisserman, A., 2014. Speeding up convolutional neural networks with low rank expansions. ArXiv Prepr. ArXiv14053866.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems. pp. 1097–1105.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proc. IEEE 86, 2278–2324.

Lee, D., Kwon, S.J., Kim, B., Wei, G.-Y., 2019. Learning Low-Rank Approximation for CNNs. ArXiv190510145 Cs Stat.

Lentaris, G., Maragos, K., Stratakos, I., Papadopoulos, L., Papanikolaou, O., Soudris, D., Lourakis, M., Zabulis, X., Gonzalez-Arjona, D., Furano, G., 2018. High-performance embedded computing in space: Evaluation of platforms for vision-based navigation. J. Aerosp. Inf. Syst. 15, 178–192.

Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P., 2016. Pruning filters for efficient convnets. ArXiv Prepr. ArXiv160808710.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C., 2016. Ssd: Single shot multibox detector, in European Conference on Computer Vision. Springer, pp. 21–37.

Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J., 2016. Pruning convolutional neural networks for resource efficient inference. ArXiv Prepr. ArXiv161106440.

Polino, Antonio, Razvan Pascanu, et Dan Alistarh 2018. Model compression via distillation and quantization. ArXiv preprint arXiv:1802.05668, 2018.

Qi, B., Shi, H., Zhuang, Y., Chen, H., Chen, L., 2018. On-board, real-time preprocessing system for optical remote-sensing imagery. Sensors 18, 1328.

Redmon, J., Farhadi, A., 2018. Yolov3: An incremental improvement. ArXiv Prepr. ArXiv180402767.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. ArXiv150504597 Cs.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. ArXiv Prepr. ArXiv14091556.

Soukup, M., Gailis, J., Fantin, D., Jochemsen, A., Aas, C., Baeck, P., Benhadj, I., Livens, S., Delauré, B., Menenti, M., 2016. HyperScout: Onboard Processing of Hyperspectral Imaging Data on a Nanosatellite, in: Proceedings of the Small Satellites, System & Services Symposium (4S) Conference, Valletta, Malta.

Stivaktakis, R., Tsagkatakis, G., Moraes, B., Abdalla, F., Starck, J.-L., Tsakalides, P., 2019. Convolutional neural networks for spectroscopic redshift estimation on euclid data. IEEE Trans. Big Data.

Tai, C., Xiao, T., Zhang, Y., Wang, X., E, W., 2016. Convolutional neural networks with low-rank regularization. ArXiv151106067 Cs Stat.

Tan, S., Caruana, R., Hooker, G., Lou, Y., 2018. Distill-and-compare: auditing black-box models using transparent model distillation, in: Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society. ACM, pp. 303–310.

Vanhoucke, V., Senior, A., Mao, M.Z., 2011. Improving the speed of neural networks on CPUs.

Wei, X., Liu, W., Chen, L., Ma, L., Chen, H., Zhuang, Y., 2019. FPGA-Based Hybrid-Type Implementation of Quantized Neural Networks for Remote Sensing Applications. Sensors, 2019 Feb 22;19(4). pii: E924. doi: 10.3390/s19040924..