Resource-efficient Object Detection by Sharing Backbone CNNs

Werner Bailer and Hannes Fassold
DIGITAL – Institute for Information and Communication Technologies
JOANNEUM RESEARCH Forschungsgesellschaft mbH
Steyrergasse 17, 8010 Graz, Austria
Email: {firstname.lastname}@joanneum.at

***Abstract*—The detection of objects in image and video has made huge progress in recent years due to the use of deep convolutional neural networks (DNNs), with some network architectures becoming de-facto standards. This paper addresses the problem of sharing a backbone CNN for different tasks, for example, to enable detection of additional classes when an already trained network is available. When using multiple such neural networks, sharing a backbone can save inference time and memory consumption. We study sharing a common backbone between neural networks trained for different tasks (logoness and text block detection) based on Yolo v3. We provide results on the impact of different lengths of the shared backbone on performance and resource efficiency.**

I. Introduction

The detection of objects in image and video has made huge progress in recent years due to the use of deep convolutional neural networks (DNNs). For several tasks, some network architectures such as VGG [**?**], ResNet [**?**] or Darknet [**?**] have become a de-facto standard. Those networks are adapted to specific problems using transfer learning or used as part of a larger architecture.

This paper addresses the problem of sharing a backbone for different tasks, for example, to enable the detection of additional classes when an already trained network is available. Often, the need for specific classes arises at a later stage, and there are benefits not to retrain the existing classifier with additional classes: First, the training effort will be smaller than for just retraining a classifier for all classes. Second, the performance of the combined classifier might be lower, and third, even if the overall performance does not decrease, the classifier is likely to perform differently in some cases. This issue of reproducibility and consistency of DNN-based classifiers has for example been discussed [**?**]. When using multiple such neural networks, sharing a backbone can save inference time and memory consumption. The latter is especially important when processing large images, e.g., 360° scenes [**?**].

We study sharing a common backbone network between neural networks trained for different tasks, based on Yolo v3 [**?**]. Using different lengths of the common backbone, and retraining for the rest for the specific tasks, we provide results showing the possible performance and resource gain trade-offs.

The rest of this paper is organized as follows. Section II discusses related work. The approach for sharing the backbone and performing partial retraining is described in Section III, and Section IV presents the results.

II. Related work

There is some existing work on sharing base models, and incremental neural network representations. The authors of [**?**] aim at object detection across domain boundaries, but without the need to include detectors specifically for each domain. They propose a domain-attentive detector, and analyze for this purpose the activations of object detectors on different datasets. They conclude that the early layers are more relevant for domain adaptation. In [**?**] an approach for merging two trained networks by sharing weights is proposed, in order to run multiple tasks efficiently. However, this approach requires fine-tuning for all tasks, i.e., does not leave the base network unmodified.

A recent paper [**?**] has analyzed the statistics of weights in differences of models, obtained from retraining a model or training multiple specializations of a base model using transfer learning. The conclusion is that significantly better compression can be achieved due to the higher sparsity and lower value ranges of weights. An approach for transfer learning called fixed model reuse has been proposed [**?**], which can be extended to be a shared base model for multiple applications, enabling efficient incremental representations for services that require models for multiple related tasks (e.g., different image classification problems). An incremental approach to neural network quantization has been proposed [**?**], which iterates cycles performing weight partition, group-wise quantization and re-training. The approach can be used to trade-off model size and performance loss, thus creating models with increasing size and performance, that benefit from being deployed incrementally, also in order to reduce latency before the initial model is available.

Domain adaptation (while staying with the same set of classes) is not in scope of our work, which aims to add detection capabilities for additional classes. We also aim to maintain stability and reproducibility of the already supported classes when adding additional ones.

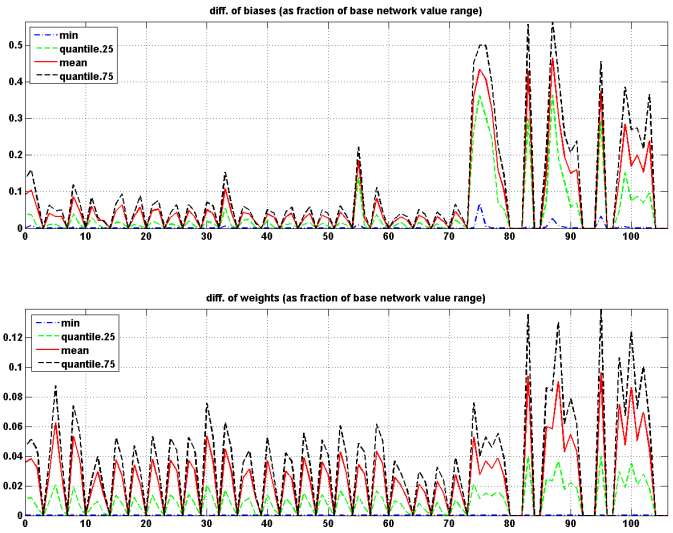

Fig. 1. Relative differences of biases and weights for the logo model.

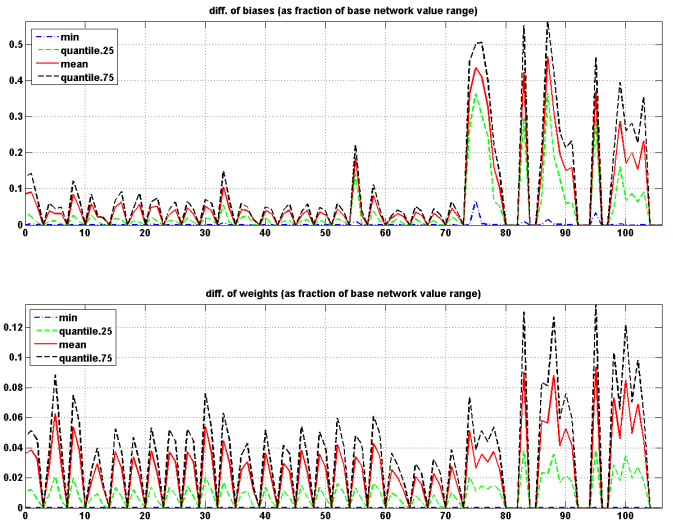

Fig. 2. Relative differences of biases and weights for the text model.

III. Shared backbone with partial retraining

We study the problem of using a pretrained Yolo v3 classifier on the MS COCO dataset[1] [**?**], and two adapted versions of the model. One is trained for text detection on the COCO-Text v2 dataset[2] and the other for logoness detection (i.e., detection of regions that contain a logo, without identifying a particular logo instance) on the Logos in the Wild dataset [**?**]. Both networks follow the same architecture as the pretrained model, but have only one output class. In addition, the shape of the anchor boxes is different for the text detector, addressing the more longitudinal nature of text areas. We trained the two variants of the network end-to-end, starting from a network pretrained on MS COCO.

We analyze the differences of biases and weights between the pretrained model and the two refined variants. Layers that have a different number of inputs or outputs (e.g., due to the different number of classes) are ignored. The differences are expressed as fractions of the base model. Figures 1 and 2 show the result of this analysis. It is apparent that there are a few layers with higher weight differences in the first few blocks of convolutional layers, around layer 30, and that the differences in terms of both weights and biases increase strongly after layer 74. In the Yolo v3 architecture, this layer marks the start of the last block of convolutional layers before the first detection block (layer 82).

The distribution of differences of weights and biases for the two models trained starting from MS COCO is similar. For comparison, we also analyze the differences between Yolo v3 trained on MS COCO, and Yolo v3 trained on the OpenImages v4 dataset [**?**]. The results are shown in Figure 3. Even though this model is trained from scratch, the overall pattern of differences is similar compared to the model trained on MS COCO, with the mean differences higher than for the logo and text models, and with significantly more and higher outliers in the differences.

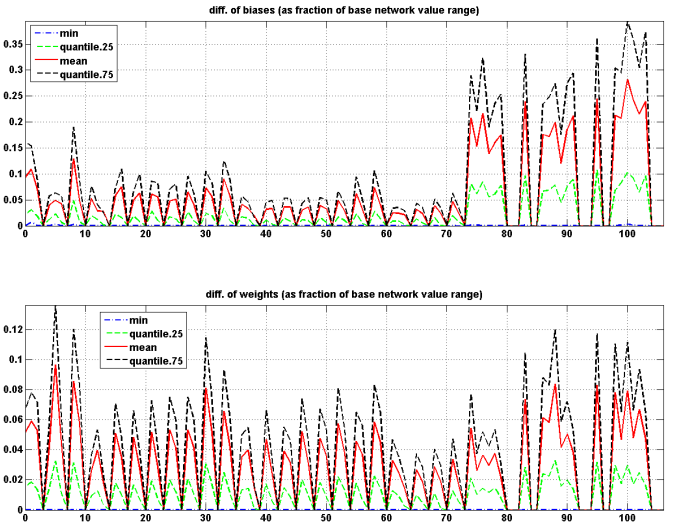

Fig. 3. Relative differences of biases and weights for the OpenImages v4 model.

It is obvious that there are strong differences in the biases and weights, in particular from layer 74 onward, and that these layers are strongly influenced by the changed targets in the detection layers. This is thus a natural point to choose the part of the network before this layer as a common backbone.

As an initial experiment, we mixed the weights in the end-to-end transferred model, using the ones from base model trained on MS COCO up to layer 74, and after that using the one from the specific model. This model performs very badly, as the end-to-end model adapts already in earlier layers, so that the weights in the later layers do not match.

We thus chose to partially retrain the model starting at layer k (experiments have been performed for $k = \{40, 48, 64, 73\}$, see below), starting from the model pretrained on MS COCO.

[1] http://cocodataset.org

[2] https://bgshih.github.io/cocotext/

As stated above, layer 74 seems to be a natural choice for retraining after it, as the differences in weights and biases increase strongly. As some performance loss can be expected, performance could be regained by reducing k, i.e. trading of resource savings and performance. However, in contrast to the difference increase after layer 74, there is no such point in the earlier layers of Yolo v3. Instead, the differences all follow a similar pattern, so that the choice of k seems arbitrary. However, we have always chosen k to be the last of a block of convolutional layers.

For the practical implementation, we created a tool for measuring the differences in weights and biases, determining statistics, and concatenating networks. When using networks for multiple tasks in inference, they should be combined in a single network. We insert thus a routing layer after the last layer of the network (i.e. layer 107 in Yolo v3), and connect it to layer k. Then we replicate the rest of the network, if needed with adjustments (e.g. different anchor boxes for different detection tasks). For example, when combining the base network capable of detecting the 80 MS COCO classes with a text block detector, we obtain a network with 141 layers in total, and twice the detection blocks for three scales. The implementation of the DarkForce framework[3] has been extended to handle multiple sets of detection heads, with possibly a different number of classes for each of them.

IV. Results

We evaluate the results by measuring the mean average precision (MAP) on the validation sets of the respective data sets (i.e., Logos in the Wild for logoness detection, COCO Text v2 for text block detection). Figure 5 provides an overview of the results. The results for $k = 0$ correspond to end-to-end training, and represent the anchor to compare against. The sharing of the backbone network results in some performance loss, rather moderate for logoness detection (just below 7%), and more significant for text block detection (around 16%). Moving k to 40 roughly halves the performance loss in both cases.

In order to further analyze why the performance loss cannot be eliminated or further reduced by moving k to earlier layers, we look again at the differences of weights and biases. Figure 4 shows these differences for the logoness detection with $k = 40$. It is obvious that the differences of the layers between 60 and 80 are quite small, i.e., most of the retraining has affected the earlier layers. This shows that some adaptation to the new problem takes place in the early layers, and can only be partly compensated in a later stage. This puts a limit to the performance that can be achieved with the shared backbone, when no information from earlier layers is adapted.

Table I provides and overview of the fraction of weights and FLOPs that are saved when the backbone is shared up to a certain layer k. While the number of weights impacts the memory consumption of the model, the number of FLOPs impacts the computational effort. It is apparent, that the number of weights in the earlier layers of the model is lower, as the convolutional layers have relatively few parameters. Conversely, as these layers work on still larger data, they are particularly computationally heavy. We can see from the results, that sharing the backbone up to layer 74 saves 2/3 of the memory and 3/4 of the computational effort of every additional model sharing the backbone network.

[3]https://github.com/pjreddie/darknet

Layer n° (k)	frac. weights	frac. FLOPs
73	0.655	0.744
64	0.401	0.663
48	0.156	0.505
40	0.092	0.424

TABLE I
RESOURCE REDUCTION BY REUSING BACKBONE UP TO A CERTAIN LAYER.

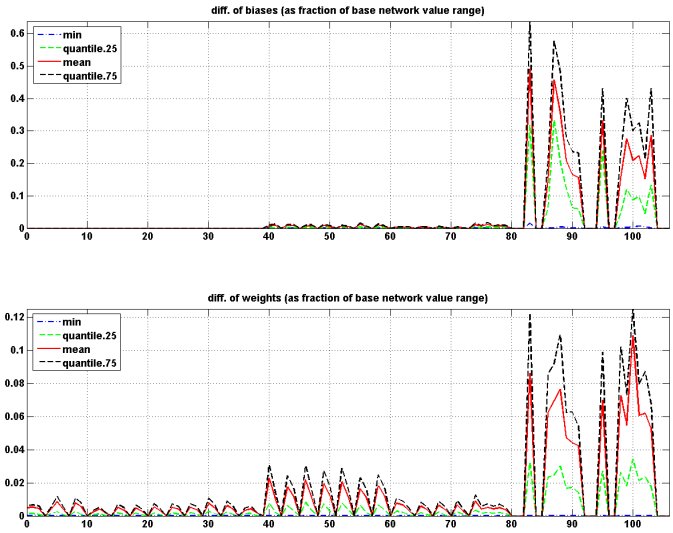

Fig. 4. Relative differences of biases and weights for the logo model with shared backbone, retrained after layer 40.

V. Conclusion

In this work we have analyzed sharing a backbone network for different visual object detection tasks, based on the Yolo v3 architecture. Results on standard data sets for logo detection and text block detection have been provided. We have studied the impact of different lengths of the shared backbone on performance and resource efficiency. At a performance decrease of about 3.5% for logoness and about 9% for text block detection, 42% of the floating point operations of each additional detector are saved. If one can accept a twice as large performance degradation, $2/3$ of the memory and $3/4$ of the floating operations can be saved.

In future work, this analysis will be extended to other visual analysis tasks, that can share the same backbone. One open question is still how to determine the best choice of the length of the shared backbone. It may depend on the specific tasks, and as our results show, it cannot be directly derived from the weight and bias differences of the fully trained networks for different tasks.

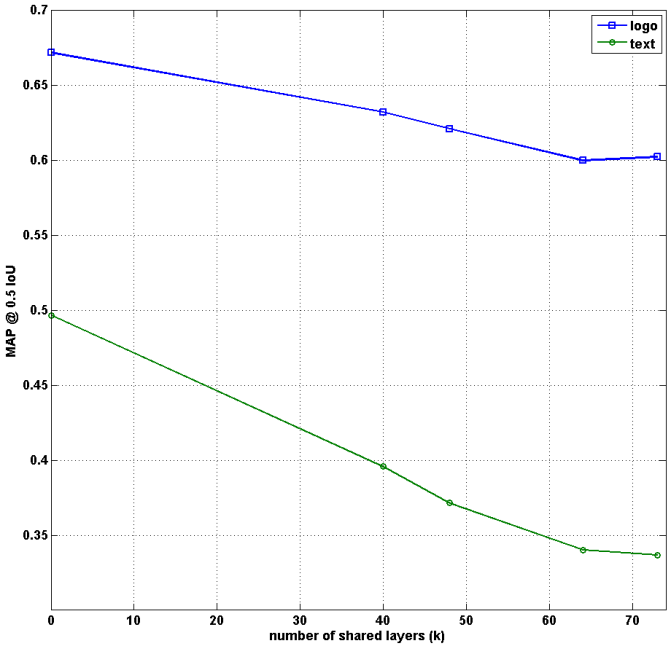

Fig. 5. Detection results (MAP @ 0.50 IoU) for logo and text detection with different number of shared layers (i.e., retrained starting at layer k).

ACKNOWLEDGMENT

This work has received funding from the European Union's Horizon 2020 research and innovation programme, under grant agreements n° 761802 MARCONI ("Multimedia and Augmented Radio Creation: Online, iNteractive, Individual") and n° 761934, Hyper360 ("Enriching 360 media with 3D storytelling and personalisation elements").

The authors would like to thank Ridouane Ghermi for his support with training the models used in the experiments.

REFERENCES

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[3] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[4] W. Bailer, "On the traceability of results from deep learning-based cloud services," in *Proceedings of the 24th International Conference MultiMedia Modeling*, Bangkok, TH, 2018.

[5] H. Fassold, "Automatic camera path generation from 360° video," in *14th International Symposium on Visual Computing*, 2019.

[6] X. Wang, Z. Cai, D. Gao, and N. Vasconcelos, "Towards universal object detection by domain attention," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7289–7298.

[7] Y.-M. Chou, Y.-M. Chan, J.-H. Lee, C.-Y. Chiu, and C.-S. Chen, "Unifying and merging well-trained deep neural networks for inference stage," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. AAAI Press, 2018, pp. 2049–2056.

[8] Z. Chen, S. Wang, D. O. Wu, T. Huang, and L.-Y. Duan, "From data to knowledge: Deep learning model compression, transmission and communication," in *2018 ACM Multimedia Conference on Multimedia Conference*. ACM, 2018, pp. 1625–1633.

[9] Y. Yang, D.-C. Zhan, Y. Fan, Y. Jiang, and Z.-H. Zhou, "Deep learning for fixed model reuse," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[10] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless cnns with low-precision weights," *arXiv preprint arXiv:1702.03044*, 2017.

[11] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[12] A. Tüzkö, C. Herrmann, D. Manger, and J. Beyerer, "Open Set Logo Detection and Retrieval," in *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications: VISAPP*, 2018.

[13] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig, and V. Ferrari, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *arXiv:1811.00982*, 2018.