COLLABORATIVE OBJECT DETECTORS ADAPTIVE TO BANDWIDTH AND COMPUTATION

Juliano S Assine[⋆] *J.C.S. Santos Filho*[†] *Eduardo Valle*[⋆]

[⋆] RECOD.ai Lab [†]WissTek Lab
School of Electrical and Computer Engineering, UNICAMP, Brazil

ABSTRACT

In the past few years, mobile deep-learning deployment progressed by leaps and bounds, but solutions still struggle to accommodate its severe and fluctuating operational restrictions, which include bandwidth, latency, computation, and energy. In this work, we help to bridge that gap, introducing the first configurable solution for collaborative object detection that manages the triple communication-computation-accuracy trade-off with a single set of weights. Our solution shows state-of-the-art results on COCO–2017, adding only a minor penalty on the base EfficientDet-D2 architecture.

Index Terms— Collaborative intelligence, split computing, configurable neural networks, bandwidth adaptive, compression

1. INTRODUCTION

In both raw number and market share, most of today's computing devices are mobile or embedded [1]. Given their ubiquity, it can hardly be overstated the impact of deploying on them cutting-edge computer-vision models, based on deep learning. However, such deployment poses serious challenges, due to computation, energy, and memory constraints.

A recent trend to answer those challenges is *collaborative intelligence* [2], which divides deep-learning inference across multiple devices. In particular, *split computing* [3] computes the first layers of a deep model in the local device, and delegates the remaining layers to a remote server (Fig. 1). While promising, those techniques face additional hurdles posed by limited and fluctuating network bandwidth and latency. Thus, before becoming practical for widespread consumer deployment, solutions have to adapt to multiple — and varying — network constraints.

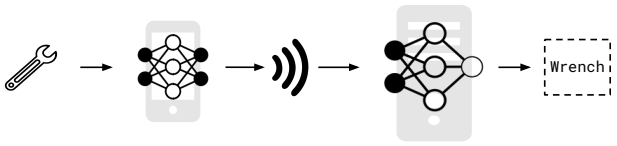

Fig. 1. Sketch of how machine learning with split computing works: part of the inference happens on the local device, and part happens on a remote server.

Traditional approaches for adapting to those varying constraints are based on multiple trained models, requiring multiple sets of weights that are switched to memory on the fly, severely increasing storage needs, power consumption, and latency during reconfiguration [4].

The **main contribution** of this work is proposing a *configurable split object-detector* with a single set of weights. Our model is the first full-neural (i.e., without non-differentiable layers) split object detector adaptable for bandwidth with a single set of weights. It is also the first split-computing vision model flexible on the fly for both bandwidth and computation. It achieves state-of-the-art performance in all three configuration axes: computation, bandwidth, and precision.

2. RELATED WORK

This is not an exhaustive survey, but a brief review of recent literature. We refer the reader to Bajić *et al.* [2] and Matsubara *et al.* [5], who recently compiled the challenges and literature for the collaborative intelligence/split computing paradigm. For a broader look into edge deep learning, with discussions on the topic, we recommend the work of Wang [6, Section V].

Early works in the field tended to focus on the feasibility of the concept, showing that split computing outperformed plain offloading of the compressed input [7, 8, 9]. Bandwidth-wise, split computing was already a strong contender since unsupervised deep compression was known to surpass classical compression [10]. However, computation-wise, its feasibility was far from clear since deep compression lacks the extensive rate-distortion-complexity literature that is available for image and video coding [11]. Eshratifar *et al.* [12] were one of the first to prove the viability of the concept with a simple framework of autoencoding+JPEG compression, analyzed under a power consumption model for wireless networks [13]. Their work showed clear gains over plain offloading in terms of bandwidth, latency, *and* device power consumption. Soon after, literature focused on the computational requirements for a given bandwidth constraint, mostly for image classification, whose simple sequential architectures allow for straightforward splitting schemes [12, 14, 15].

In contrast, recent works on split computing for object detection show little concern for computation, focusing only on bandwidth, which reflects the challenge of splitting the branching architectures typical for the task. All existing solutions split the model before the branching stages [16, 17, 18], to avoid huge bandwidth penalties (Fig. 2).

Performance comparison across works and authors is difficult in the present literature. Even when restricting our scope to object detection, the lack of agreed datasets, protocols and metrics hinders direct comparison. In this work, we choose the challenging COCO-2017 dataset, also adopted by Matsubara *et al.* [16, 3] and Cohen *et al.* [18], but even among that limited set of works the choice of metrics varies. We follow Matsubara *et al.*, choosing the mAP@50:95 as a more challenging (and more robust) metric for object detection instead of the easier (and potentially noisier) mAP@50 chosen by Cohen *et al.*. As a result, the work of Matsubara *et al.* [16] is our main baseline.

2.1. Configurability

Existing art lacks standard terminology, employing the terms "adaptive", "dynamic", "flexible", and others with an array of meanings, ranging from relatively rigid models, trained separately (with different weight-sets) for each operational condition, until highly flexible models, able to select different computation paths on-the-fly based on each input (e.g., using early exits) [19]. In this work, we reserve the term **configurable** to models able to adapt to operational parameters (e.g., bandwidth) without loading separate sets of weights for each setting.

Bandwidth configurability appears in literature with dimensionality reduction [7], quantization [18], image/video codecs [8, 20, 12], or a combination of those techniques. A critical design choice is the number of channels in the model bottleneck, i.e., the point where the model representation is the smallest, and thus the natural point for splitting the model. As far as we know, Choi *et al.* [17] provided the only existing model with a configurable channel number at the bottleneck, but their work requires multiple server-side decoder models.

In comparison to bandwidth, **computational configurability** has been less explored. It appears in works for image compression [21], and for audio compression/generation [22], but as far as we know, this is the first work to explore it for split computing.

3. PROPOSED DESIGN

The main novelty of the proposed architecture (Fig. 2) is its encoder *configurability*. All models introduced are *bandwidth-configurable*, and several are *computation-configurable*. Another novelty is the high-performance EfficientDet [23] as base architecture, motivated by its performance in COCO-2017, reduced encoder size, and compact end-block output. We split the model at the latest possible point before branching, where the representation transmitted to the remote server is most compressible.

3.1. Configurability

To achieve configurability, we employ Slimmable Neural Networks [24], which allow dynamically configuring **the fraction of active channels** on convolutional layers via a parameter $\boldsymbol{\alpha} \in (0, 1)$. Depending on which layers are slimmable, we may achieve both bandwidth and computation configurability, or only the former. For bandwidth-only configurability, we only have to make the last layer of the encoder and the first layer of the decoder slimmable. For both configurabilities, we make all convolutions in the encoder slimmable.

While the computational impact of varying a single convolutional layer is small, the slimming factor $\boldsymbol{\alpha}$ accumulates *quadratically* when applied to neighboring layers, as the computational cost of each convolutional layer is linear both on its own number of filters and on the number of filters from its previous layer (Table 1).

3.2. Training

Our training procedure (Fig. 2) merges the advantages from Generalized Head Network Distillation [16] and from MutualNet [25]. From the latter, we employ the slimmable training with the "sandwich rule" of Yu and Huang [24], which for training with N widths per training step, picks the lowest width, the highest width, and $N - 2$ randomly chosen intermediate widths. From the former, we adopt the distillation loss minimizing the ℓ_2-norm between the *features* extracted by student and teacher at several points of the decoder, including the point where the model splits. That procedure changes from MutualNet (and from Yu and Huang), which is based on a KL-divergence distillation loss over the model outputs.

The main training parameters are $\boldsymbol{C}$, the maximum number of channels in the bottleneck, and the set of widths for which the model should learn to operate, e.g., $\boldsymbol{\alpha} \in (0.25, 0.33, 0.5, 0.66, 1.0)$. The number of widths may be larger than N, as different values will be chosen at random by the sandwich rule. Our experiments confirmed the original findings [24] that $\boldsymbol{\alpha}_{\mathbf{min}}$, the **minimum value in the set**, dominates the impact on performance.

A fine, but crucial, adaptation in moving from Faster-RCNN and Mask-RCNN (from [16]) to EfficientDet was removing the post-training batchnorm statistics used by Yu and Huang [24], as they considerably hindered the new model.

Base architecture	$\alpha = 1.0$	$\alpha = 0.25$
Mask-RCNN	7.6	0.8
EfficientDet-D2	2.6	0.6

Table 1. Full-slimmable encoders: computation in GMAC.

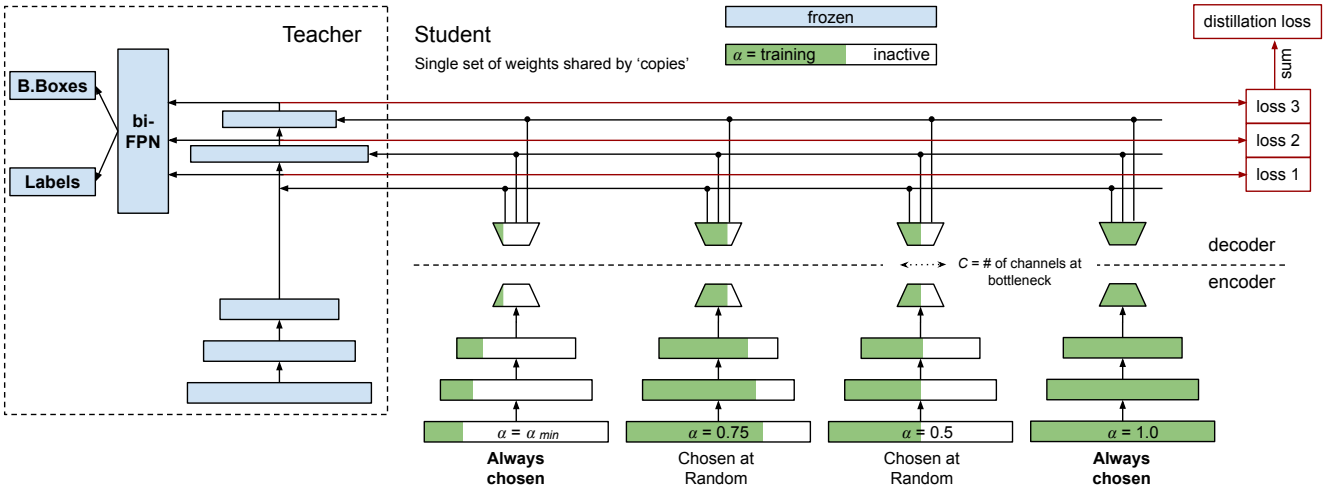

Fig. 2. Training of the proposed split slimmable architecture by distillation. N "copies" of the model (all sharing a single set of weights) are trained concurrently, for a varying number of active channels. In this case, $N = 4$.

4. EXPERIMENTS AND RESULTS

4.1. Materials and methods

We performed all experiments on the COCO-2017 dataset [26], measuring the outcomes on the validation split with the mAP@50:95 metric. This is the primary metric for the COCO challenge, which corresponds to the mean average precision, averaged for all region IoU thresholds between 50 and 95%, with a 5% step. We used the official COCO API to compute the metric. We measure computational costs using the number of Multiply-Accumulate operations (MAC), and derived multiples (e.g., GMAC=10^9 MAC).

We used the Faster RCNN and Mask RCNN models from Matsubara *et al.* [16], with the trained weights made available by them, as non-configurable baselines. We contrast those models with the performance of four bandwidth-configurable models (Faster RCNN, Mask RCNN, EfficientDet-D1, and EfficientDet-D2) and two computation- and bandwidth-configurable models (EfficientDet-D2 and Faster RCNN).

We trained all our models for 12 epochs, using the largest batch size possible for each model (6 for non-configurable, 8 for configurable models). We hand-optimized the learning rates and schedulers of all models, setting learning rates to decrease by half every 3 epochs for non-configurable and bandwidth-configurable models, and every 2 epochs for full-configurable models. All values of α used were segmented in steps of $\frac{1}{12}$, which resulted in round channel numbers. For inference, all encoder outputs were discretized with 8-bit uniform quantization. All models were implemented using PyTorch. A full implementation, including all details, is available in our source repository[1].

[1]https://github.com/jsiloto/adaptive-cob

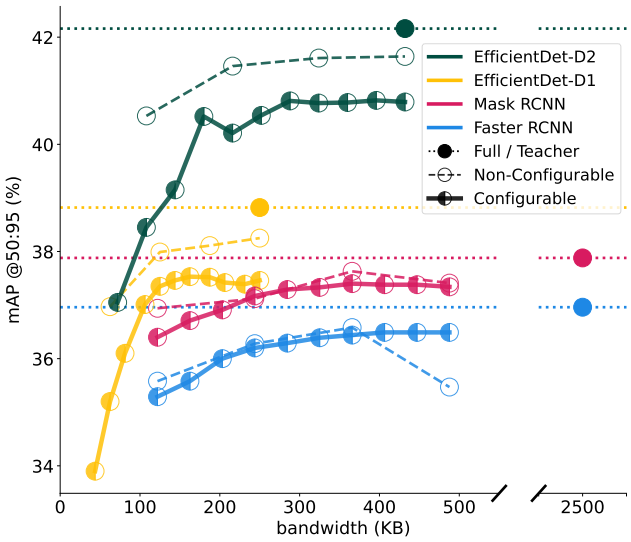

Fig. 3. Comparison of **Bandwidth-Configurable** (half-dots) and **Non-Configurable** (hollow dots) encoders. Teachers provide a baseline (full dots, horizontal dotted lines).

4.1.1. *Main results*

We showcase the performance of our bandwidth-configurable models in Fig. 3. Our results show that it is possible to make Mask RCNN and Faster RCNN bandwidth-configurable without impacting their performance. We also present EfficientDet-D1 and, especially, EfficientDet-D2 as much better base architectures in terms of mAP *vs.* bandwidth, all the while using much less computation — although there is some performance penalty in making them configurable.

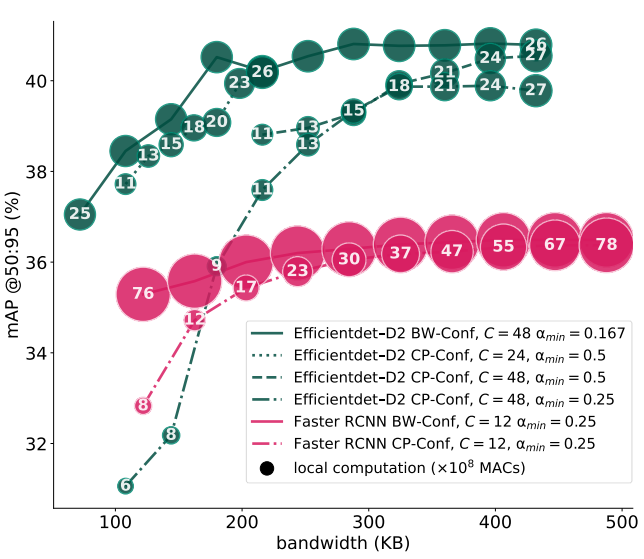

Fig. 4. Comparison between encoders that are only Bandwidth-Configurable (BW-Conf) with ones that are also Computation-Configurable (CP-Conf), showcasing local computation cost ($\mathbf{x}10^8$ MAC) in the data points' sizes and labels. While BW-Conf encoders display almost fixed computation, CP-Conf encoders are able to exploit the triple trade-off.

A finer-grained comparison of some of the models appears in Fig. 4, where we added computation-configurable techniques. The small labels inside the dots indicate encoder computation costs, in tenths of GMAC = 10^8 multiply-accumulate operations. Here, the same bandwidth-only-configurable Faster RCNN and EfficientDet-D2 from Fig. 3 appear as baselines (continuous lines). In the other series, we plot the results by varying the size of the bottleneck C and the minimum value for α. Fig. 4 shows that full-configuration is achievable, with a considerable reduction in computation, and modest impact on mAP, except below a certain threshold. Achieving full configuration, however, is less straightforward than bandwidth configuration. Contrarily to the bandwidth-only-configurable Efficient-D2 [BW-Conf: $(C = 48, \alpha_{\min} = 0.167)$], a single full-configurable version cannot be stretched over extended regimens of bandwidth and computation without suffering large penalties: notice how the left tail of the [CP-Conf: $(C = 48, \alpha_{\min} = 0.25)$] series drops sharply in mAP. Still, if covering such an extended range of configurations is necessary, it is possible to do it advantageously with just two sets of weights, instead of several.

Finally, in Fig. 5 we show how our bandwidth reduction could be enhanced with more aggressive quantization, still with modest impact on mAP, showcasing how improved forms of quantization (such as those presented by Cohen *et al.* [18]) could enhance our approach in both performance and configurability.

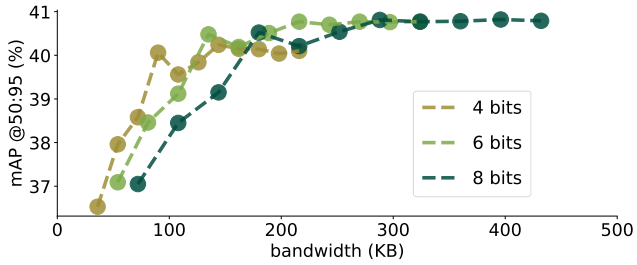

Fig. 5. Effects of feature quantization on Bandwidth-Configurable EfficientDet-D2.

5. DISCUSSION

In order to reach large-scale deployment to mobile and embedded consumers, deep learning will have to adapt to several, often varying, operational constraints. In this paper we showed how configurable architectures are a crucial step in that direction, allowing for the deployment of compact, single-weight-set models which are still robust for many operational settings.

Because both communication and computation are critical resources for mobile devices, the latter being especially relevant for attaining energy and thermal goals, bandwidth-only configuration is not enough for consumer deployment. By allowing configuration in both axes, our model is an important step in the realistic deployment of configurable object detectors to mobile consumers.

We employed a single slimming factor α for the entire encoder. Future works could explore the advantage of more factors, decoupling computation and bandwidth. In this work, the server model was kept fixed, but server-side bandwidth and computation configurability may be relevant, e.g., in high-throughput situations to avoid queuing. Collaborative on-the-fly client-server reconfiguration to maintain optimal quality-of-service is a fascinating frontier for deep computer vision.

6. ACKNOWLEDGMENTS

J. S. Assine is funded by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Finance Code 001. Eduardo Valle is partially funded by CNPq Grant 315168/2020-0. J.C.S. Santos Filho is partially funded by CNPq Grant 306241/2019-6. The RECOD.ai Lab is funded by grants from FAPESP, CAPES, and CNPq.

7. REFERENCES

[1] P. R. IMARC, "Embedded systems market size is growing at 5.6% cagr to reach 95400 million usd in 2024," Feb. 2019.

[2] I. V. Bajić, W. Lin, and Y. Tian, "Collaborative intelligence: Challenges and opportunities," in *ICASSP - IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 8493–8497.

[3] Y. Matsubara and M. Levorato, "Split computing for complex object detectors: Challenges and preliminary results," in *Proc. of the 4th Int. Workshop on Embedded and Mobile Deep Learning*, 2020, pp. 7–12.

[4] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE Int. Solid-State Circuits Conf. Digest of Technical Papers (ISSCC)*. IEEE, 2014, pp. 10–14.

[5] Y. Matsubara, M. Levorato, and F. Restuccia, "Split computing and early exiting for deep learning applications: Survey and research challenges," *arXiv preprint arXiv:2103.04505*, 2021.

[6] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[7] J. S. Assine, A. Godoy, and E. Valle, "Compressing representations for embedded deep learning," *arXiv preprint arXiv:1911.10321*, 2019.

[8] H. Choi and I. V. Bajić, "Deep feature compression for collaborative object detection," in *2018 25th IEEE Int. Conf. on Image Processing (ICIP)*. IEEE, 2018, pp. 3743–3747.

[9] A. E. Eshratifar, A. Esmaili, and M. Pedram, "Towards collaborative intelligence friendly architectures for deep learning," in *20th Int. Symposium on Quality Electronic Design (ISQED)*. IEEE, 2019, pp. 14–19.

[10] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," *arXiv preprint arXiv:1611.01704*, 2016.

[11] J. Vanne, M. Viitanen, T. D. Hamalainen, and A. Hallapuro, "Comparative rate-distortion-complexity analysis of hevc and avc video codecs," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1885–1898, 2012.

[12] A. E. Eshratifar, A. Esmaili, and M. Pedram, "Bottlenet: A deep learning architecture for intelligent mobile cloud computing services," in *2019 IEEE/ACM Int. Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 2019, pp. 1–6.

[13] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4g lte networks," in *Proc. of the 10th Int. Conf. on Mobile systems, applications, and services*, 2012, pp. 225–238.

[14] M. Jankowski, D. Gündüz, and K. Mikolajczyk, "Joint device-edge inference over wireless links with pruning," in *2020 IEEE 21st Int. Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2020, pp. 1–5.

[15] J. Shao and J. Zhang, "Communication-computation trade-off in resource-constrained edge inference," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 20–26, 2020.

[16] Y. Matsubara and M. Levorato, "Neural compression and filtering for edge-assisted real-time object detection in challenged networks," in *2020 25th Int. Conf. on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 2272–2279.

[17] H. Choi, R. A. Cohen, and I. V. Bajić, "Back-and-forth prediction for deep tensor compression," in *ICASSP - IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 4467–4471.

[18] R. A. Cohen, H. Choi, and I. V. Bajić, "Lightweight compression of neural network feature tensors for collaborative intelligence," in *2020 IEEE Int. Conf. on Multimedia and Expo (ICME)*. IEEE, 2020, pp. 1–6.

[19] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *2016 23rd Int. Conf. on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 2464–2469.

[20] Z. Chen, K. Fan, S. Wang, L.-Y. Duan, W. Lin, and A. Kot, "Lossy intermediate deep learning feature compression and evaluation," in *Proc. of the 27th ACM Int. Conf. on Multimedia*, 2019, pp. 2414–2422.

[21] F. Yang, L. Herranz, Y. Cheng, and M. G. Mozerov, "Slimmable compressive autoencoders for practical neural image compression," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2021, pp. 4998–5007.

[22] Y. Xu, Y. Song, S. Garg, L. Gong, R. Shu, A. Grover, and S. Ermon, "Anytime sampling for autoregressive models via ordered autoencoding," in *Int. Conf. on Learning Representations*, 2021.

[23] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2020, pp. 10781–10790.

[24] J. Yu and T. S. Huang, "Universally slimmable networks and improved training techniques," in *Proc. of the IEEE Int. Conf. on Computer Vision*, 2019, pp. 1803–1811.

[25] T. Yang, S. Zhu, C. Chen, S. Yan, M. Zhang, and A. Willis, "Mutualnet: Adaptive convnet via mutual learning from network width and resolution," in *European Conf. on Computer Vision*. Springer, 2020, pp. 299–315.

[26] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conf. on computer vision*. Springer, 2014, pp. 740–755.