A Multi-task Supervised Compression Model for Split Computing

Yoshitomo Matsubara [*]
University of California, Irvine
yoshitom@uci.edu

Matteo Mendula
University of Bologna
matteo.mendula@unibo.it

Marco Levorato
University of California, Irvine
levorato@uci.edu

Abstract

Split computing ($\neq$ split learning) is a promising approach to deep learning models for resource-constrained edge computing systems, where weak sensor (mobile) devices are wirelessly connected to stronger edge servers through channels with limited communication capacity. State-of-the-art work on split computing presents methods for single tasks such as image classification, object detection, or semantic segmentation. The application of existing methods to multi-task problems degrades model accuracy and/or significantly increase runtime latency. In this study, we propose Ladon, the first multi-task-head supervised compression model for multi-task split computing.[1] Experimental results show that the multi-task supervised compression model either outperformed or rivaled strong lightweight baseline models in terms of predictive performance for ILSVRC 2012, COCO 2017, and PASCAL VOC 2012 datasets while learning compressed representations at its early layers. Furthermore, our models reduced end-to-end latency (by up to 95.4%) and energy consumption of mobile devices (by up to 88.2%) in multi-task split computing scenarios.

1. Introduction

Learning compressed representations in a supervised manner has been empirically shown effective, specifically for training deep learning models for split computing (SC) [21]. Split computing is an efficient distributed inference technique for resource-constrained edge computing systems, where either executing entire models on mobile devices or fully offloading tasks to cloud/edge servers may not be a feasible or optimal solution [19, 20]. In split computing, the first layers of a deep learning model (*i.e.*, encoder) are executed by the mobile device, while the remaining layers' computation are offloaded to an edge/cloud server via a wireless communication channel with a constrained capacity. We remark that, different from distributed training such as

federated learning [24] or split learning [37], models for split computing are trained offline and only executed over multiple machines at runtime.

A recent study [23] conducts comprehensive benchmark experiments of supervised compression for split computing (SC2) and empirically demonstrates that supervised compression outperforms input compression (*e.g.*, codec-based and neural-based image compression methods for input images) and feature compression (*e.g.*, introducing autoencoders [13] to intermediate layers) for single downstream tasks. However, the study discusses single task scenarios only, and lacks latency and energy consumption evaluations from recent real-world embedded computers such as autonomous drones and smartphones for time-sensitive applications [6].

In this work, we propose Ladon, the first end-to-end multi-task supervised compression model for split computing that can serve multiple tasks in a fast, energy efficient manner. Following the SC2 benchmark [23], we assess the proposed model in terms of encoder size, compressed data size, and model accuracy, compared to popular lightweight models. Besides the metrics from the SC2 benchmark, we consider end-to-end latency and energy consumption of mobile devices for multi-task split computing, where we execute multiple tasks given an input sample *e.g.*, run image classification, object detection, and semantic segmentation tasks for an input image in resource-constrained edge computing systems.

Our contribution in this work is three-fold:

1. We propose the first supervised compression model designed for multi-task split computing and empirically assess its predictive performance specifically for image classification (ILSVRC 2012), object detection (COCO 2017), and semantic segmentation (PASCAL VOC 2012) tasks [8, 15, 30], using popular lightweight models as competitive baselines.

2. Based on resource-constrained edge computing systems, we evaluate average end-to-end latency of the baselines and the Ladon model in multi-task scenarios. Our Ladon model takes advantage of its single, lightweight multi-task encoder and saves up to 95.4% of the end-to-end latency for the baselines.

---

[1]Code and models are available at https://github.com/yoshitomo-matsubara/ladon-multi-task-sc2

3. Using NVIDIA Jetson devices, we assess energy consumption of the baseline and proposed methods on the mobile devices in the same multi-task scenarios. Our approach successfully reduces the energy consumption by $65.0 - 88.2\%$.

2. Preliminaries

In split computing [21], resource-constrained edge computing systems consist of three components: 1) a weak mobile (local) device; 2) a wireless communication link with limited capacity between 1) and 3); and 3) a compute-capable edge/cloud server. Local devices have limited computing resources and battery constraints *e.g.*, NVIDIA Jetson devices. The wireless communication link can achieve a limited (and volatile) data rate due to power and bandwidth contraints and impaired propagation. To make an example, a low-power communication technology, LoRa, has a maximum data rate of 37.5 Kbps.

In this setting, from an execution perspective it is preferable for the mobile devices to offload as much computing load as possible to the edge server, thus minimizing local computing (LC) load and energy consumption. However, in order to achieve low end-to-end latency, it is also essential to reduce the size of the data transmitted over the limited wireless channel to support offloading. Intuitively, the more we compress data, the more we sacrifice a model's predictive performance *e.g.*, model accuracy. In general, it is easier to aggressively compress the data at late layers of a deep learning model (*e.g.*, the penultimate layer) without significant loss of model accuracy, which, however, results in high encoding cost at the mobile devices. This results in a known three-way tradeoff between encoding cost, data size, and model accuracy in the context of supervised compression for split computing (SC2) [23].

3. Related Work

Split computing approaches have been actively studied in different research communities [21], and compared with local computing and full offloading baselines for resource-constrained edge computing systems. The former runs the full model on a weaker local device, and the latter transmits sensor data (*e.g.*, an image) to a stronger edge server via limited wireless connection channels and runs the model. Note that split computing is different from split learning [37], as split computing only involves online inference of offline trained models.

A multi-task model for collaborative inference, called Chimera, is proposed in [26], where two variants of the model are discussed. One of the models serves two different image classification tasks (gender and attribute) for 32×32 pixel face images from the CelebA dataset [17], and the other serves two different pixel-wise predictions (semantic segmentation and depth estimation) for NYUv2 dataset [33]. However, this approach is designed under the implicit assumption that a reference model to be trained "naturally" contains at least one layer whose output data size is smaller than the input data size, which is called a bottleneck. Such a strong assumption not only limits data compression gain, but also implies non-trivial computing cost on weak mobile devices as effective splittable layers only occur later in computer vision models. In fact, it is shown in their study that their approach requires to execute at least the first 34 layers in their model to produce smaller data than the input image.

The entropic student [22] is a supervised compression method that combines concepts of neural image compression and knowledge distillation for split computing. The method offers a sharable learned encoder and can serve multiple computer vision tasks. A comprehensive benchmark study on supervised compression for split computing (SC2) [23] empirically shows that the entropic student is currently the best SC2 approach for three challenging computer vision tasks: ILSVRC 2012 (image classification) [30], COCO 2017 (object detection) [15], and PASCAL VOC 2012 (semantic segmentation) [8]. We consider the approach as the strongest baseline to compare with this study from the perspective of the three-way tradeoff illustrated earlier. While its learned encoder can serve multiple tasks, the subsequent modules are independently trained on each of the target tasks and is not optimized as a multi-task model at runtime. This results in a large computing load at the edge server if the input image is to be analyzed to extract different outputs.

4. Ladon - Proposed Approach -

In this section, we describe our proposed Ladon model, highlighting the differences from the entropic student [22] in Fig. 1. Both the entropic student and our Ladon models are supervised compression models for split computing. Supervised compression is defined as learning compressed representations for supervised downstream tasks [23].

4.1. Problem Formulation

In general, a supervised compression model is trained for a single task (*e.g.*, image classification) [7, 18, 22, 34] and has a deterministic mapping $\mathbf{x} \mapsto \hat{\mathbf{z}} \mapsto \hat{\mathbf{y}}$, where $\mathbf{x}$, $\hat{\mathbf{z}}$, and $\hat{\mathbf{y}}$ indicate the input data, compressed representation, and prediction, for a given supervised downstream task. Note that $\hat{\mathbf{z}}$ contains more relevant information with supervised tasks rather than information to reconstruct the original input data $\mathbf{x}$, which is different from neural image compression [2, 3] that learns a compressed representation of the input data $\mathbf{x}$ and produces the reconstructed data $\hat{\mathbf{x}}$ such that $\mathbf{x} \approx \hat{\mathbf{x}}$ in an unsupervised manner.

We define f_θ as an encoder with learned parameters θ to transform an input image $\mathbf{x}$ to the corresponding compressed data $\hat{\mathbf{z}}$ *i.e.*, $f_\theta(\mathbf{x}) = \hat{\mathbf{z}}$. Similarly, we define g_ϕ as the remain-
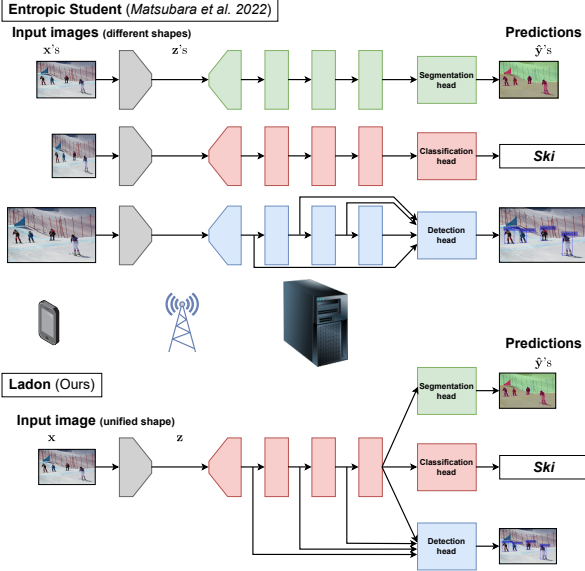
Figure 1. Entropic student (top) vs. our Ladon (bottom) in multi-task scenario. Gray module (encoder) is trained in a task-agnostic way. Red, green, and blue modules are trained for image classification, semantic segmentation, and object detection tasks respectively. Entropic student's encoder serves multiple downstream tasks, but all its modules except the encoder are trained independently, using different image preprocessing pipelines such as resizing and cropping. Our Ladon model shares a single image preprocessing pipeline and most of its parameters (except those of task-specific heads) across the downstream tasks at run time. While Entropic Student is designed to run three separate inferences per image, the Ladon model runs a single inference to serve the three tasks.

ing layers in the supervised compression model to produce a task-specific prediction $\hat{\mathbf{y}}$ including task-specific learned parameters ϕ and the encoded data $\mathbf{z}$ from the local device *i.e.*, $g_\phi(\hat{\mathbf{z}}) = \hat{\mathbf{y}}$.

At runtime, the encoder f_θ runs on the mobile device and then transmits encoded data $\hat{\mathbf{z}}$ to an edge server via a wireless channel. Once the edge server receives the encoded data $\hat{\mathbf{z}}$, the remaining layers (including the decoder) g_ϕ are executed on the edge server to produce the prediction $\hat{\mathbf{y}}$.

Shared encoder vs. end-to-end multi-task model As shown in Fig. 1 (top), the Entropic Student method trains a single encoder that serves multiple supervised tasks, and then trains the subsequent modules for each of the individual tasks. Those models may share some of the network architectures between the tasks (*e.g.*, ResNet-50 [10] as a backbone in [22]), but their parameters are separately optimized for each of the tasks. Thus, in multi-task scenarios, the entropic student method requires multiple models to be deployed on the edge server, which increases memory footprint and computing cost. Our Ladon model addresses the limitation and

shares not only the network architecture, but also learned parameters of a backbone model between the tasks. As illustrated in Fig. 1 (bottom), the Ladon model is designed to minimize task-specific components and enable to simultaneously serve multiple tasks. *I.e.*, our model produces multi-task predictions given an input image at runtime.

Unified preprocessing Notably, different computer vision tasks have different requirements during the image preprocessing phase (see Table 2 and Fig. 1). For image classification tasks, a popular preprocessing at runtime is a combination of resizing an image to fixed patch size and center-cropping the resized image with a smaller fixed patch size (*e.g.*, 224×224 pixels). Object detection and semantic segmentation tasks usually prefer more flexible resizing (keeping height and width ratio of the image) and do not use cropping since model predictions such as bounding box and pixel-wise predictions need to be overlaid on the input image. As illustrated in Fig. 1, the entropic student method trains different subsequent modules with a unique preprocessing pipeline for each of the tasks while our Ladon model is optimized with a unified preprocessing pipeline so that the model can simultaneously serve multiple tasks given an input image without model accuracy degradation.

4.2. Model Implementations

We implement our Ladon model using two backbone models pretrained on the ILSVRC 2012 dataset: ResNet-50 (#params: 25.6M) [10] and ResNeSt-269e (#params: 110M) [38]. Specifically, we modify each of the backbone models and replace its first layers (including its first residual block) with supervised compression encoder-decoder architectures [22]. The modified backbone architectures serve an image classification task for ILSVRC 2012 (ImageNet) dataset and are reused as part of Faster R-CNN with feature pyramid network (FPN) [14, 29] and DeepLabv3 [4] for object detection and semantic segmentation tasks (COCO 2017 and PASCAL VOC 2012 datasets). Our code repository and trained PyTorch [27] models are publicly available.[1]

4.3. Training

We introduce three steps to train our Ladon model. We describe its hyperparameters in the supplementary material.

Step 1: Pre-training encoder-decoder Following [22], we begin by training the modified backbone model using the original pretrained backbone model as a teacher model for the ILSVRC 2012 dataset. We minimize a loss function

$$\mathcal{L}_{\text{pre}}(\mathbf{x}) = \underbrace{\sum_{i \in \mathbb{I}} ||\mathbf{h}_i^t(\mathbf{x}) - \mathbf{h}_i^s(\mathbf{x})||_2^2}_{\text{distortion}} - \beta \underbrace{\log p_\phi(f_\theta(\mathbf{x}) + \epsilon)}_{\text{rate}} \quad (1)$$

$$\epsilon \sim \text{Unif}(-\tfrac{1}{2}, \tfrac{1}{2}),$$

Device	CPU	GPU	RAM
NVIDIA Jetson Nano	Quad-core ARM Cortex-A57	128-core NVIDIA Maxwell	4 GB
NVIDIA Jetson Xavier NX	6-core NVIDIA Carmel 64-bit ARMv8.2	384-core NVIDIA Volta	8 GB
Laptop computer	Intel Core i9-11950H	NVIDIA RTX A500 Mobile	32 GB

Table 1. Device specifications

Task	Index	Name	#Params.	Size	Input Shape	Task metric
ILSVRC 2012 (IC)	1	MobileNetV3 (MNv3) Large 1.0 [12]	5.5M	21.0 MB	(3, 224, 224)	Acc. 74.0
	2	MobileNetV2 [32]	3.5M	13.5 MB	(3, 224, 224)	Acc. 71.9
	3	MNASNet 1.3 [36]	6.3M	24.2 MB	(3, 224, 224)	Acc. 76.5
	4	MNASNet 0.5 [36]	2.2M	8.5 MB	(3, 224, 224)	Acc. 67.7
COCO 2017 (OD)	5	SSD300 w/ VGG-16 [16]	35.6M	136 MB	(3, 300, 300)	mAP 25.1
	6	SSD Lite w/ MNv3 [12]	5.2M	20.0 MB	(3, 320, 320)	mAP 21.3
	7	Faster R-CNN w/ MNv3 + FPN [29]	19.4M	74.1 MB	(3, 320+, 320+)	mAP 22.8
PASCAL VOC 2012 (SS)	8	DeepLabv3 w/ MNv3 [4]	11.0M	42.2 MB	(3, 416+, 416+)	mIoU 73.4
	9	LRASSP w/ MNv3 [12]	3.2M	12.4 MB	(3, 416+, 416+)	mIoU 73.3

Table 2. Accuracy [%], mean average precision (mAP) [%], and mean intersection over union (mIoU) of baseline models for ILSVRC 2012 (IC: image classification), COCO 2017 (OD: object detection), and PASCAL VOC 2012 (SS: semantic segmentation), respectively

where $\mathbb{I}$ is a set of indices for teacher-student layer pairs. f_θ and g_ϕ indicate encoder and decoder of a student model, respectively. p_ψ is a prior probability model of quantized representations (the entropy model) and used for both the encoder and decoder [3]. $\mathbf{h}_i^t(\mathbf{x})$ and $\mathbf{h}_i^s(\mathbf{x})$ indicate embeddings of the i-th pair of teacher and student layers for a given model input $\mathbf{x}$. β is a hyperparameter that controls a rate-distortion tradeoff (*e.g.*, compressed data size and predictive performance). Note that for our Ladon model, $\mathbf{h}_1^s(\mathbf{x})$ in training is an output of its decoder *i.e.*, $\mathbf{h}_1^s(\mathbf{x}) = g_\phi\left(f_\theta\left(\mathbf{x}\right) + \epsilon\right)$.

Step 2: Fine-tuning decoder and subsequent modules
Once the pre-training of the encoder-decoder module is completed, we fix the parameters of the encoder and fine-tune the decoder and the subsequent modules in the modified backbone models. The purpose of this fine-tuning step is to make their parameters reusable for relevant downstream tasks. In this study, we fine-tune the modules by a standard knowledge distillation [11] on the ILSVRC 2012 dataset given that ImageNet pre-training is empirically demonstrated to speed up convergence of CNN models on the target task [9].

Step 3: Fine-tuning other task-specific modules Following the step 2, we freeze all the parameters of the model and introduce other task-specific heads as the Ladon's additional branches (see Fig. 1). Specifically, we attach object detection and semantic segmentation modules to the model and fine-tune the newly attached task-specific heads. Due to the nature of supervised compression (Section 4.1), the additional tasks should be related to the first supervised task.

5. Experiments

We conduct comprehensive experiments using various computing devices and heterogeneous lightweight baselines.

5.1. Experimental Settings

Following [23], we consider the three challenging computer vision datasets for predictive performance evaluations: ILSVRC 2012[2] (image classification) [30], COCO 2017[3] (object detection) [15], and PASCAL VOC 2012[4] (semantic segmentation) [8]. For end-to-end latency evaluations, we simulate resource-constrained edge computing systems, considering two devices with unbalanced computing resources (one weak mobile device and one strong edge server) and a limited wireless communication link (data rate: 100 Kbps) or LoRa [31] - a protocol for low-power communications, and its maximum data rate is 37.5 Kbps. Table 1 summarizes computing devices we consider for end-to-end latency evaluations. Making a few pairs of the computing devices, we assess end-to-end latency in multi-task scenarios for each of the baseline methods and our proposed approach.

5.2. Baselines

Table 2 summarizes lightweight single-task models that we compare with our Ladon (multi-task model) in terms of predictive performance for image classification, object detection, and semantic segmentation tasks. To highlight

---

[2] https://image-net.org/challenges/LSVRC/2012/
[3] https://cocodataset.org/#detection-2017
[4] http://host.robots.ox.ac.uk/pascal/VOC/voc2012/

differences from our end-to-end multi-task model at runtime (see Fig. 1), we consider Entropic Student [22], which uses a shared encoder and separate image preprocessing pipelines for different tasks, as an additional baseline when discussing end-to-end latency and local energy consumption.

5.3. Evaluation Metrics

Our evaluations are based on the benchmark framework for split computing in [23], which adopts three key metrics: 1) encoder size, 2) compressed data size, and 3) model accuracy. Additionally, we consider end-to-end latency and energy consumption of mobile devices in multi-task scenarios. To estimate energy consumption, we employ Energon [25], a power monitoring tool compliant to the Prometheus standard [28]. Energon integrates with the built-in power monitoring registries of Jetson devices and x86 Intel-based laptop architectures. To convert instantaneous power consumption into Joules, we use Simpson's rule for numerical integration [1, 35] over the same number of inferences for each experimental configuration. We warm up the devices before conducting latency and power consumption experiments to stabilize GPU behavior [5]. Using the five different evaluation metrics, we comprehensively discuss the proposed method with respect to our baselines.

6. Results

This section discusses results of the experiments. Section 6.1 presents the evaluation results in terms of static SC2 metrics: task-specific rate-distortion tradeoff (compressed data size and predictive performance) and total encoder size for multi-task scenarios in split computing. Sections 6.2 and 6.3 show the end-to-end latency and the energy consumption for local devices, respectively.

6.1. Static SC2 Metrics

Figures 2 - 4 show task-specific rate-distortion tradeoffs for our Ladon models referring to lightweight models as local computing baselines. We confirm that Ladon models outperform many of the models designed for mobile execution in terms of task-specific prediction performance. For the semantic segmentation task, the Ladon models achieve comparable predictive performance with respect to the baselines. It is understandable that the Ladon models encounter a more difficult challenge in the semantic segmentation task due to the lossy compression of the representation as the task requires pixel-level predictions while image classification and object detection are image-level and box-level predictions.

Besides the rate-distortion tradeoff, the SC2 benchmark [23] suggests encoder size as an additional static evaluation metric. It is notable that the encoder sizes of our Ladon models are only $0.543 - 0.935$ MB, which are approximately $0.268 - 2.29\%$ of total model size for multi-task local computing configurations (Tables 2 and 3).
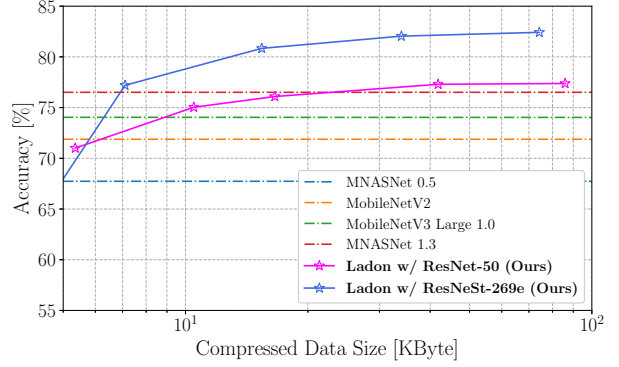

Figure 2. ILSVRC 2012: tradeoff between compressed data size and model accuracy
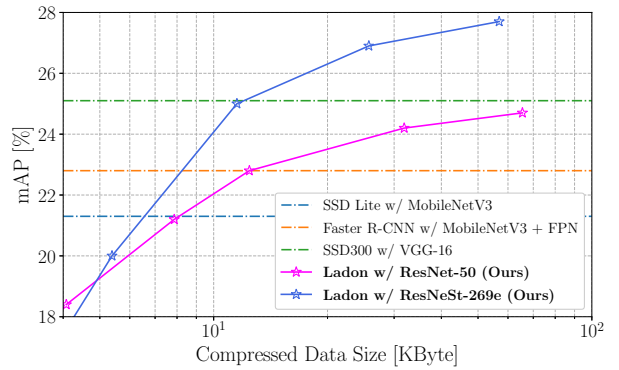

Figure 3. COCO 2017: tradeoff between compressed data size and mean average precision (mAP)
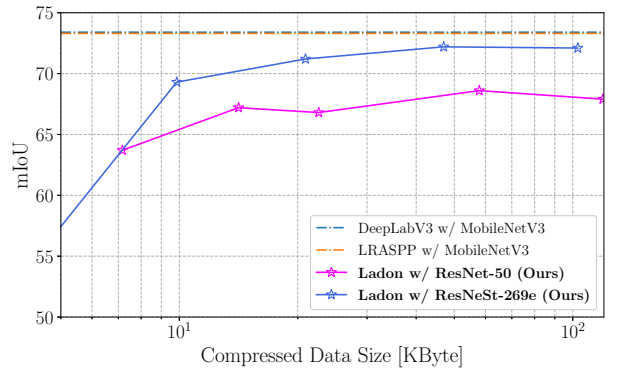

Figure 4. PASCAL VOC 2012: tradeoff between compressed data size and mean intersection over union (mIoU)

6.2. End-to-end Latency

With 30 different baseline configurations and 10 configurations for Ladon models (see Table 3), we discuss end-to-end latency evaluations based on devices listed in Table 1.

Figure 5 show the end-to-end latency when using Jetson Nano (top: CUDA OFF, bottom: CUDA ON) and laptop as mobile device and edge server, respectively. For local computing (LC) baselines, their image classification delays

Config.	IC model	OD model	SS model
LC 1	1	5	8
LC 2	2	5	8
LC 3	3	5	8
LC 4	4	5	8
LC 5	1	6	8
LC 6	2	6	8
LC 7	3	6	8
LC 8	4	6	8
LC 9	1	7	8
LC 10	2	7	8
LC 11	3	7	8
LC 12	4	7	8
LC 13	1	5	9
LC 14	2	5	9
LC 15	3	5	9
LC 16	4	5	9
LC 17	1	6	9
LC 18	2	6	9
LC 19	3	6	9
LC 20	4	6	9
LC 21	1	7	9
LC 22	2	7	9
LC 23	3	7	9
LC 24	4	7	9
SC 1	ES-IC 1	ES-OD 1	ES-SS 1
SC 2	ES-IC 2	ES-OD 2	ES-SS 2
SC 3	ES-IC 3	ES-OD 3	ES-SS 3
SC 4	ES-IC 4	ES-OD 4	ES-SS 4
SC 5	ES-IC 5	ES-OD 5	ES-SS 5
SC 6	ES-IC 6	ES-OD 6	ES-SS 6
Ours 1	Ladon (ResNet-50) w/ $\beta = 0.32$		
Ours 2	Ladon (ResNet-50) w/ $\beta = 1.28$		
Ours 3	Ladon (ResNet-50) w/ $\beta = 5.12$		
Ours 4	Ladon (ResNet-50) w/ $\beta = 10.24$		
Ours 5	Ladon (ResNet-50) w/ $\beta = 20.48$		
Ours 6	Ladon (ResNeSt-269e) w/ $\beta = 0.32$		
Ours 7	Ladon (ResNeSt-269e) w/ $\beta = 1.28$		
Ours 8	Ladon (ResNeSt-269e) w/ $\beta = 5.12$		
Ours 9	Ladon (ResNeSt-269e) w/ $\beta = 10.24$		
Ours 10	Ladon (ResNeSt-269e) w/ $\beta = 20.48$		

Table 3. End-to-end multi-task latency evaluation configurations. LC model indices in Table 2. ES-IC/OD/SS: Entropic Student (ResNet-50 / Faster R-CNN w/ ResNet-50 and FPN / DeepLabv3 w/ ResNet-50). ES-* $\{1, \cdots, 5\}$ are Entropic Student models trained with $\beta = 0.32, 0.64, 1.28, 2.56$, and 5.12, respectively [23].

are relatively small with respect to those of object detection and semantic segmentation models considered in this study. When CUDA was turned off, split computing (SC) baselines slightly outperformed local computing baselines (up to 31.3% reduction), and our multi-task models reduced the end-to-end latency by up to 82.6% and 93.6% with respect to the LC and SC baselines respectively. The local

computing baselines took more advantage of CUDA than the split computing baselines did, but our models still achieved comparable or slightly improved end-to-end latency. We confirmed similar trends in Fig. 6 where we used Jetson NX Xavier, a stronger mobile device than Jetson Nano, which made the local computing baselines even stronger. Yet, our approach reduced the end-to-end latency of the LC and SC baselines by up to 84.2% and 95.3%, respectively.

We refer readers to the supplementary material for additional evaluations that use LoRa data rate (37.5 Kbps).

6.3. Local Energy Consumption

As illustrated in Figs. 7 and 8, the overall trend is that the split computing approaches can significantly save energy consumption of mobile device in multi-task scenarios with respect to local computing even when the split computing approaches achieve higher end-to-end latency than the local computing approaches do (Figs. 5 and 6). Importantly, our Ladon models further saved energy consumption of the mobile devices. Specifically, we reduced the energy consumption of the stronger baselines (split computing) by $65.0 - 88.2\%$, which highlighted the energy efficiency of our multi-task-head supervised compression models.

7. Conclusions

Split computing emerged in research communities as a promising solution for deep learning models in resource-constrained edge computing systems. Existing studies on split computing are focused on task-specific models and assessments. In this study, we proposed Ladon, the first end-to-end multi-task supervised compression model designed to share a unified image preprocessing pipeline and network architectures across multiple tasks for efficiently serving the multiple tasks in a single inference.

Using the SC2 benchmark and real devices, we comprehensively assessed its performances for three challenging computer vision tasks, end-to-end latency, and energy consumption of mobile devices in multi-task scenarios. Learning compressed representations at early layers in a supervised manner for split computing, the Ladon models either outperformed or rivaled popular mobile-friendly models for ILSVRC 2012, COCO 2017, and PASCAL VOC 2012 in terms of predictive performance. Overall, our models also outperformed LC and SC baselines in terms of end-to-end latency and energy consumption of mobile devices. If energy consumption is not a concern, the LC baselines may be reasonable options as they achieved short latency in some cases at the cost of up to 8x energy consumption.
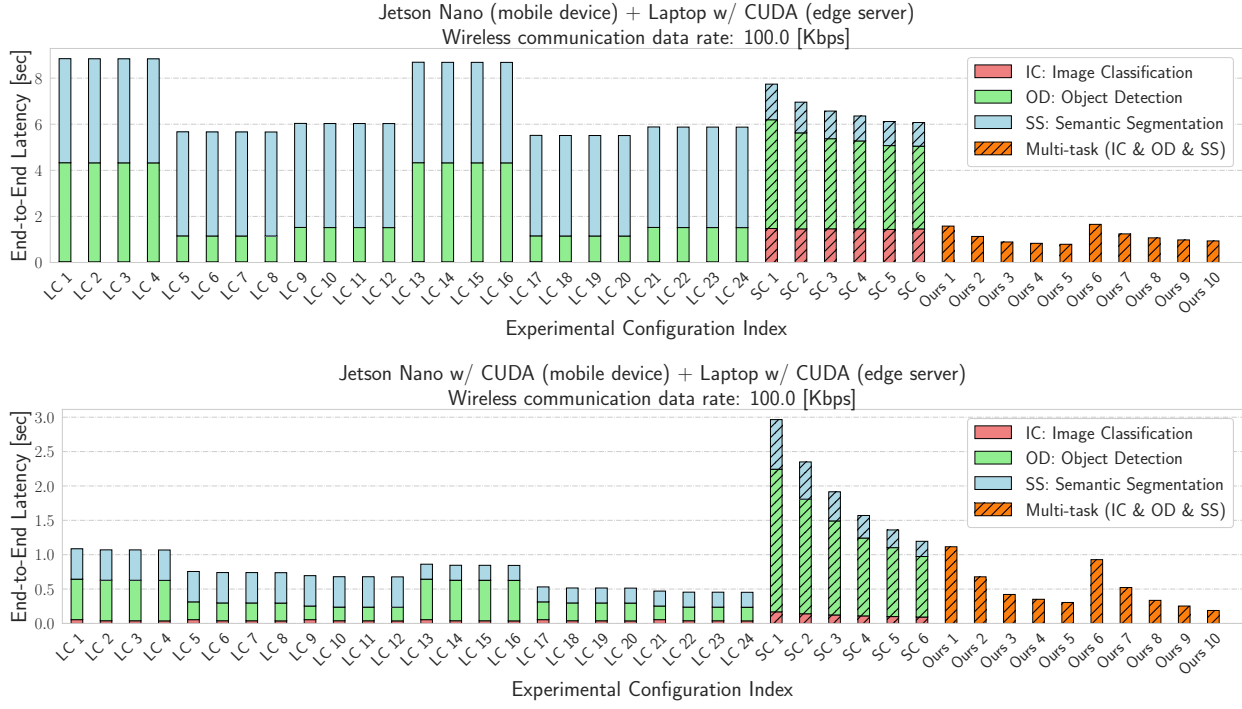
Acknowledgements

Figure 5. End-to-end latency for Jetson Nano (mobile device) and laptop with CUDA (edge server). Top/bottom: local computing without/with CUDA.
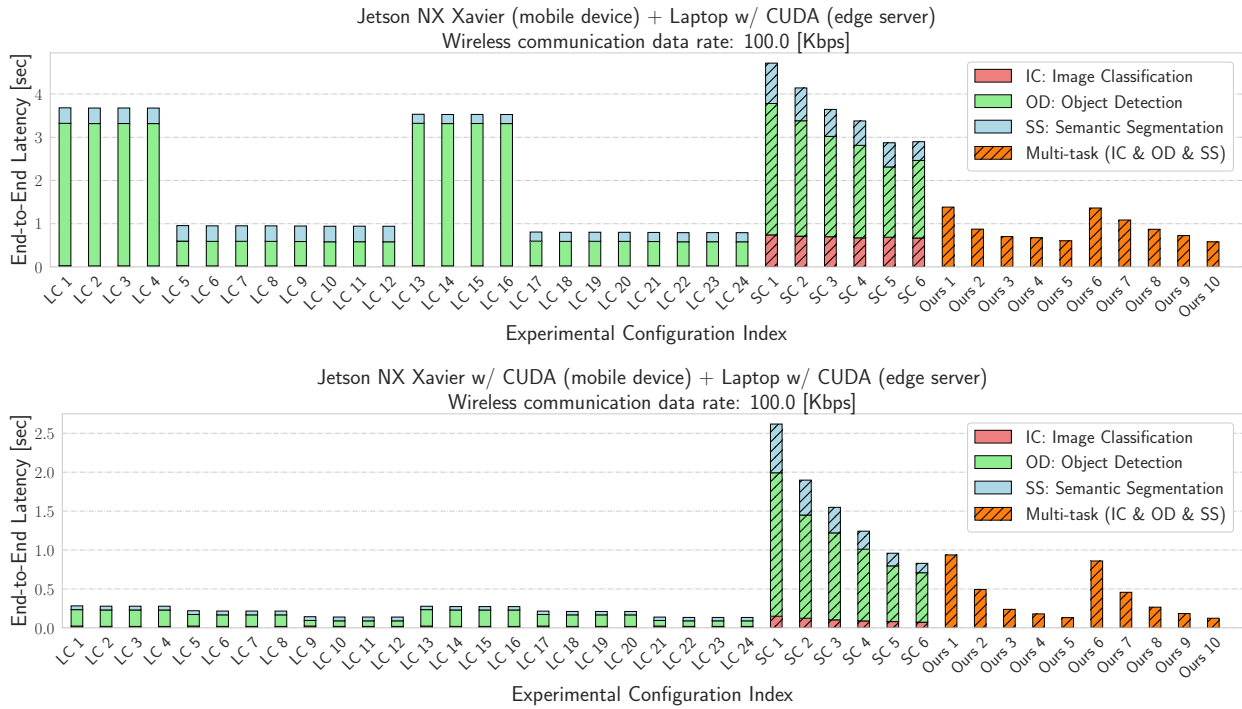

Figure 6. End-to-end latency for Jetson NX Xavier (mobile device) and laptop with CUDA (edge server). Top/bottom: local computing without/with CUDA.
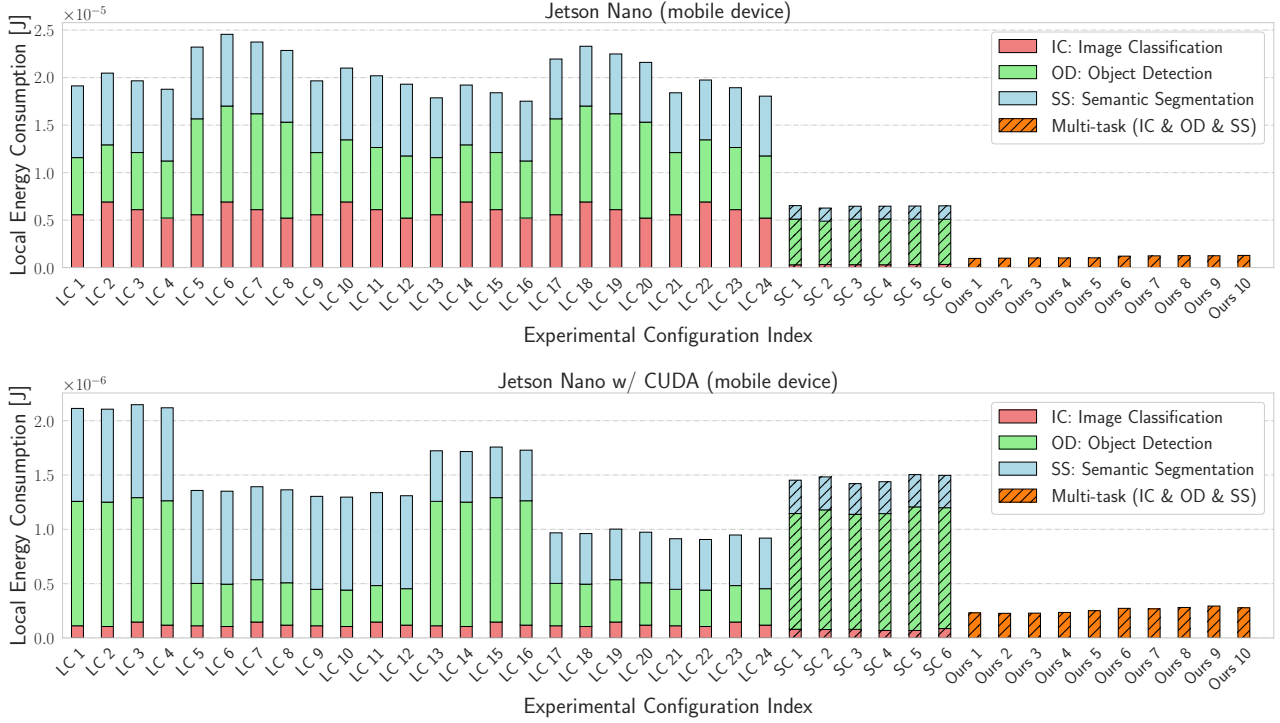
Figure 7. Energy consumption of Jetson Nano (mobile device). Top/bottom: local computing without/with CUDA.
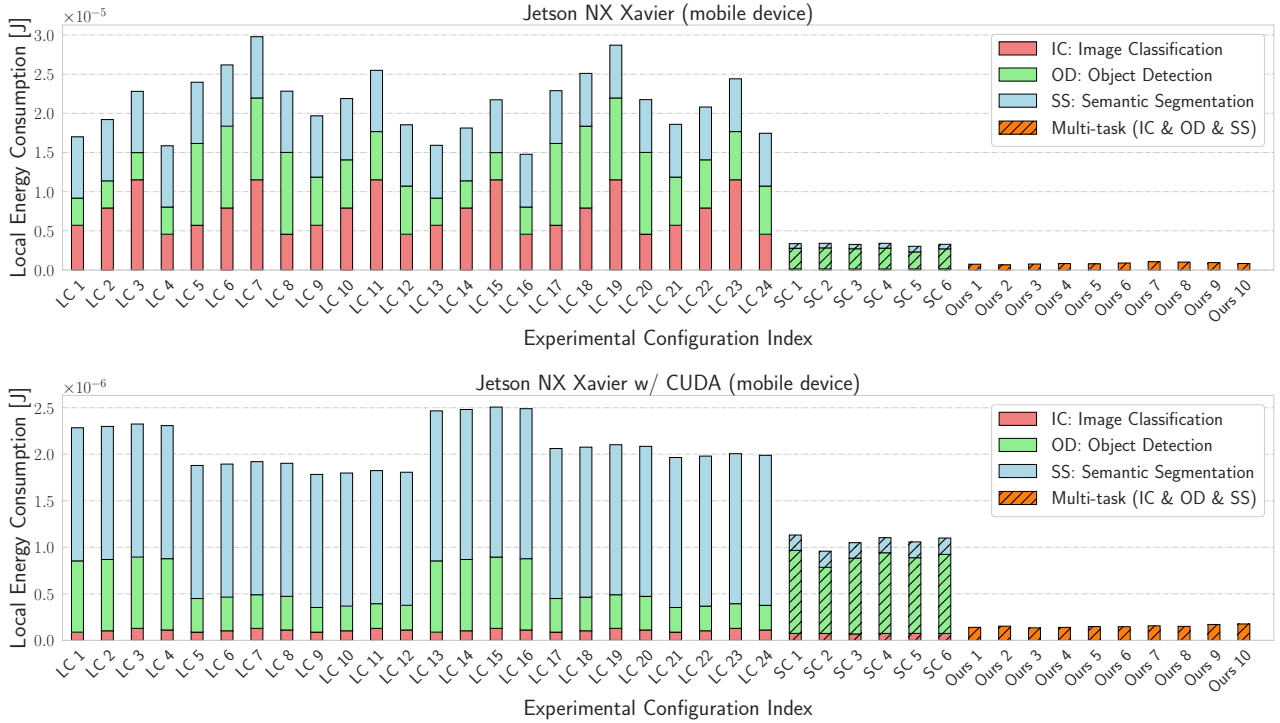

Figure 8. Energy consumption of Jetson NX Xavier (mobile device). Top/bottom: local computing without/with CUDA.

References

[1] Andrea Agiollo, Paolo Bellavista, Matteo Mendula, and Andrea Omicini. EneA-FL: Energy-aware orchestration for serverless federated learning. *Future Generation Computer Systems*, 154:219–234, 2024. 5

[2] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end Optimized Image Compression. *International Conference on Learning Representations*, 2017. 2

[3] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations*, 2018. 2, 4

[4] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 3, 4

[5] Thanh Tuan Dao, Jungwon Kim, Sangmin Seo, Bernhard Egger, and Jaejin Lee. A Performance Model for GPUs with Caches. *IEEE Transactions on Parallel and Distributed Systems*, 26(7):1800–1813, 2014. 5

[6] Marco Di Vaio, Paolo Falcone, Robert Hult, Alberto Petrillo, Alessandro Salvi, and Stefania Santini. Design and Experimental Validation of a Distributed Interaction Protocol for Connected Autonomous Vehicles at a Road Intersection. *IEEE Transactions on Vehicular Technology*, 68(10):9451–9465, 2019. 1

[7] Amir Erfan Eshratifar, Amirhossein Esmaili, and Massoud 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)*, pages 1–6, 2019. 2

[8] Mark Everingham, Luc Van Gool, CKI Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012). 2012. 1, 2, 4

[9] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking ImageNet Pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4918–4927, 2019. 4

[10] 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*, pages 770–778, 2016. 3

[11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. In *Deep Learning and Representation Learning Workshop: NIPS 2014*, 2014. 4

[12] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for MobileNetV3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019. 4

[13] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014. 1

[14] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017. 3

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 1, 2, 4

[16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *European conference on computer vision*, pages 21–37, 2016. 4

[17] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep Learning Face Attributes in the Wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015. 2

[18] Yoshitomo Matsubara, Sabur Baidya, Davide Callegaro, Marco Levorato, and Sameer Singh. Distilled Split Deep Neural Networks for Edge-Assisted Real-Time Systems. In *Proceedings of the 2019 Workshop on Hot Topics in Video Analytics and Intelligent Edges*, pages 21–26, 2019. 2

[19] Yoshitomo Matsubara, Davide Callegaro, Sabur Baidya, Marco Levorato, and Sameer Singh. Head Network Distillation: Splitting Distilled Deep Neural Networks for Resource-Constrained Edge Computing Systems. *IEEE Access*, 8:212177–212193, 2020. 1

[20] Yoshitomo Matsubara and Marco Levorato. Neural Compression and Filtering for Edge-assisted Real-time Object Detection in Challenged Networks. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 2272–2279, 2021. 1

[21] Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. Split Computing and Early Exiting for Deep Learning Applications: Survey and Research Challenges. *ACM Computing Surveys*, 55(5):1–30, 2022. 1, 2

[22] Yoshitomo Matsubara, Ruihan Yang, Marco Levorato, and Stephan Mandt. Supervised Compression for Resource-Constrained Edge Computing Systems. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2685–2695, 2022. 2, 3, 5

[23] Yoshitomo Matsubara, Ruihan Yang, Marco Levorato, and Stephan Mandt. SC2 Benchmark: Supervised Compression for Split Computing. *Transactions on Machine Learning Research*, 2023. 1, 2, 4, 5, 6

[24] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017. 1

[25] Matteo Mendula, Paolo Bellavista, Marco Levorato, and Sharon Ladron de Guevara Contreras. Furcifer: a Context Adaptive Middleware for Real-world Object Detection Exploiting Local, Edge, and Split Computing in the Cloud Continuum. In *2024 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 47–56. IEEE, 2024. 5

[26] Sumaiya Tabassum Nimi, Md Adnan Arefeen, Md Yusuf Sarwar Uddin, Biplob Debnath, and Srimat Chakradhar. Chimera:

Context-Aware Splittable Deep Multitasking Models for Edge Intelligence. In *2022 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 70–77. IEEE, 2022. 2

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019. 3

[28] Bjorn Rabenstein and Julius Volz. Prometheus: A next-generation monitoring system (talk). Dublin, May 2015. USENIX Association. 5

[29] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015. 3, 4

[30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 1, 2, 4

[31] Farzad Samie, Lars Bauer, and Jörg Henkel. IoT Technologies for Embedded Computing: A Survey. In *2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, pages 1–10. IEEE, 2016. 4

[32] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 4

[33] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor Segmentation and Support Inference from RGBD Images. In *Computer Vision – ECCV 2012*, pages 746–760. Springer, 2012. 2

[34] Saurabh Singh, Sami Abu-El-Haija, Nick Johnston, Johannes Ballé, Abhinav Shrivastava, and George Toderici. End-to-end Learning of Compressible Features. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 3349–3353. IEEE, 2020. 2

[35] Ronald J Tallarida, Rodney B Murray, Ronald J Tallarida, and Rodney B Murray. Area under a Curve: Trapezoidal and Simpson's Rules. *Manual of Pharmacologic Calculations: with Computer Programs*, pages 77–81, 1987. 5

[36] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. MnasNet: Platform-Aware Neural Architecture Search for Mobile. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019. 4

[37] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018. 1, 2

[38] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Haibin Lin, Zhi Zhang, Yue Sun, Tong He, Jonas Mueller, R Manmatha, et al. ResNeSt: Split-Attention Networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2736–2746, 2022. 3