Deep Neural Networks Efficient Inference
- lecture notes -

Edouard Yvinec[1,2] , Kevin Bailly[1,2]
Sorbonne Université[1], CNRS, ISIR, f-75005, 4 Place Jussieu 75005 Paris
Datakalab[2], 114 boulevard Malesherbes, 75017 Paris
ey@datakalab.com

October 18, 2022

Table of Contents

SORBONNE UNIVERSITÉ ISIR INSTITUT DES SYSTÈMES INTELLIGENTS ET DE ROBOTIQUE

Datakalab

Importance of Academic Actuality

(a) ResNet [16] downloads (tf-hub) - accuracy 76.15%

(b) EfficientNet B0 [32] downloads (tf-hub) - accuracy 77.10%

Efficient inference starts with efficient selection of the base model !

Image Classification

Table: ImageNet - Runtime on V100 - New references: Cait [34] and EffNet V2 [31] - Old reference: ResNet [16]

Architecture	type	accuracy	params	Flops	Runtime
ResNet 50	ConvNet	76.15	25M	4B	33ms
EfficientNet V2 S	ConvNet	83.9	22M	8.8B	24ms
CaiT xxs24	Transformers	78.52	12M	2.5B	16ms
EfficientNet V2 XL	ConvNet	87.3	208M	94B	100+ms
CaiT m36	Transformers	85.1	270.9M	53.7B	100+ms

Object Detection

Table: MSCOCO dataset – Runtime on V100 – New references: Dual-Swin-L (HTC) [22] and YOLOv7-E6 [35] – Common references: EfficientDet [33] and DETR [6] – Old reference: Faster-RCNN [28]

Architecture	type	AP_{test}/AP_{val}	params	Flops	Runtime
Faster-RCNN [28]	ConvNet	44	60M	246G	172ms (M40)
EfficientDet-D0 [33]	ConvNet	34.6/33.5	3.9M	2.5B	16ms (Titan V)
EfficientDet-D7 [33]	ConvNet	52.2/51.8	52M	325B	262ms (Titan V)
DETR DC5-R101 [6]	CNN/Trans.	- / 44.9	60M	253G	100ms
Dual-Swin-L [22]	Transformers	59.4/59.1	453M	-	600+ms
YOLOv7-E6 [35]	ConvNet	56.0 / 55.9	97M	515G	18ms

SORBONNE UNIVERSITÉ ISIR INSTITUT DES SYSTÈMES INTELLIGENTS ET DE ROBOTIQUE

Datakalab

Keypoint Estimation

Table: MS COCO - new references: VitPose [37] and RSN-50 [4] - old reference: CMU Pose [5] - speed measured on GPU A100

Architecture	type	AP	params	GFlops	Runtime
CMU Pose [5]	ConvNet	61.8	26M	-	1-ms
ViTPose-B [37]	Transformers	75.8	86M	-	1ms
RSN-50 [4]	ConvNet	74.7	25.7M	6.4	-
ViTPose-B [37]	Transformers	79.1	632M	-	4ms
4× RSN-50 [4]	ConvNet	79.2	111.8M	61.9	-

Semantic Segmentation

Table: Cityscapes dataset – New reference: ViT-Adapter-L [9] – efficient references: EfficientPS [25] and SFNet-R18 [21] – Old reference: DeepLabv3-R103 [7]

Architecture	type	mIoU	params	Flops	Runtime
DeepLabv3-R103 [7]	ConvNet	81.3	58M	—	—
ViT-Adapter-L [9]	Transformer	85.2	327M	—	—
SFNet-R18 [21]	ConvNet	80.4	13M	—	40ms GTX 1080Ti
EfficientPS [25]	ConfNet	84.2	40.89M	433B	166ms Titan RTX

Machine translation

Table: WMT2014 English-French dataset – New reference: 60L-12L ADMIN [24] – efficient reference: LightConv [36] – Old reference: Seq2seq [29]

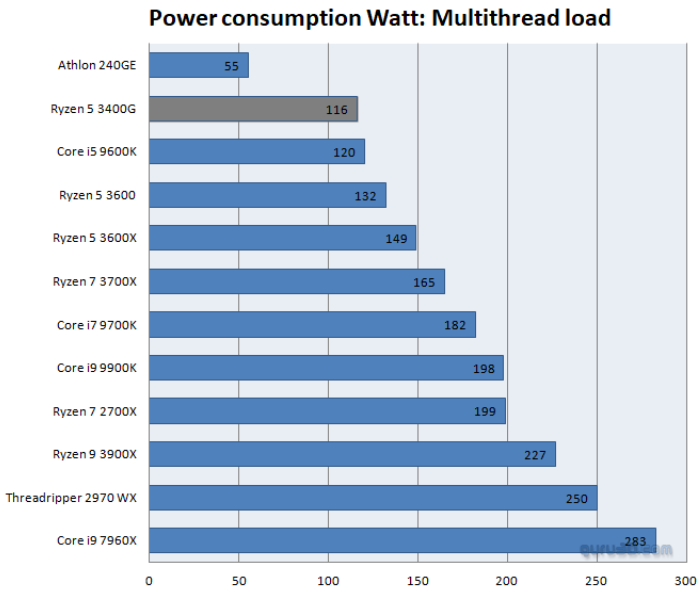
Architecture	type	BLEU	params	Runtime
Seq2seq [29]	LSTM	34.81	5*384M	1700 w/sec (C++ on GPU)
LightConv [36]	ConvNet	43.1	213M	3.9 sent/sec (En-Gr, P100, batch one)
60L-12L ADMIN [24]	Transformer	43.8	262M	—

SORBONNE UNIVERSITÉ ISIR INSTITUT DES SYSTÈMES INTELLIGENTS ET DE ROBOTIQUE

Datakalab

Outline

Standard Home PC consumption



(a) CPU power consumption

Appliance	Power Rating	Hours per Day	Common Power Use in a Day	Percentage of Power for the Day
Split System Air Con	1200W	6	6 kWh	18.0%
Pool Pump	1100W	8	8.8 kWh	26.5%
Electric Hot Water	3600W	1.5	5.4 kWh	16.2%
Electric Cooktop	2400W/element	1	4.8 kWh	14.4%
Fridge	150W	12	1.8 kWh	5.4%
Toaster	900W	0.2	0.18 kWh	0.5%
Microwave	1200W	0.2	0.24 kWh	0.7%
Kettle	2400W	0.2	0.48 kWh	1.4%
TV	200W	5	1.0 kWh	3.0%
Sound System	60W	4	0.24 kWh	0.7%
Phone Chargers	15W x 2	5	0.15 kWh	0.5%
Laptop	100W	2	0.2 kWh	0.6%
Combined Lighting	130W (LED)	5	0.65 kWh	2.0%
Bathroom Fan	60W	0.5	0.03 kWh	0.1%
Washing Machine	2400W	4 hrs / week	1.37 kWh	4.1%
Standby Appliances	120W	16	1.92 kWh	5.8%
TOTAL			**33.3 kWh per Day**	

(b) Home appliances power consumption

Figure: Modern PCs with last gen components draw 1000Watt under heavy workloads such as deep learning trainings.

SORBONNE UNIVERSITÉ ISIR INSTITUT DES SYSTÈMES INTELLIGENTS ET DE ROBOTIQUE

Datakalab

Cloud Computing

NVIDIA Tesla V100	NVIDIA RTX 3090
11 in / 267 mm	12 in / 313 mm
No outputs	1x HDMI, 3x DisplayPort
2x 8-pin	2x 8-pin
Dual-slot	Triple-slot
250 W	350 W

(a) V100 (industrial) vs RTX 3090 (personal)

Table 3. A100 Tensor Core Input / Output Formats and Performance vs FP32 FFMA.

	INPUT OPERANDS	ACCUMULATOR	TOPS	X-factor vs. FFMA	SPARSE TOPS	SPARSE X-factor vs. FFMA
V100	FP32	FP32	15.7	1x	-	-
	FP16	FP32	125	8x	-	-
A100	FP32	FP32	19.5	1x	-	-
	TF32	FP32	156	8x	312	16x
	FP16	FP32	312	16x	624	32x
	BF16	FP32	312	16x	624	32x
	FP16	FP16	312	16x	624	32x
	INT8	INT32	624	32x	1248	64x
	INT4	INT32	1248	64x	2496	128x
	BINARY	INT32	4992	256x	-	-
	IEEE FP64		19.5	1x	-	-

(b) A100 (industrial) vs V100 (industrial)

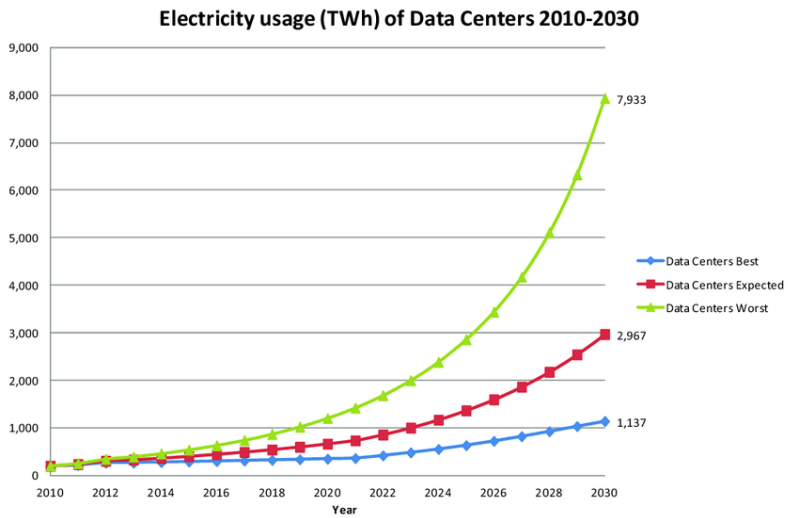
Figure: Industrial GPUs draw less power than personal ones.

source : nvidia technical report

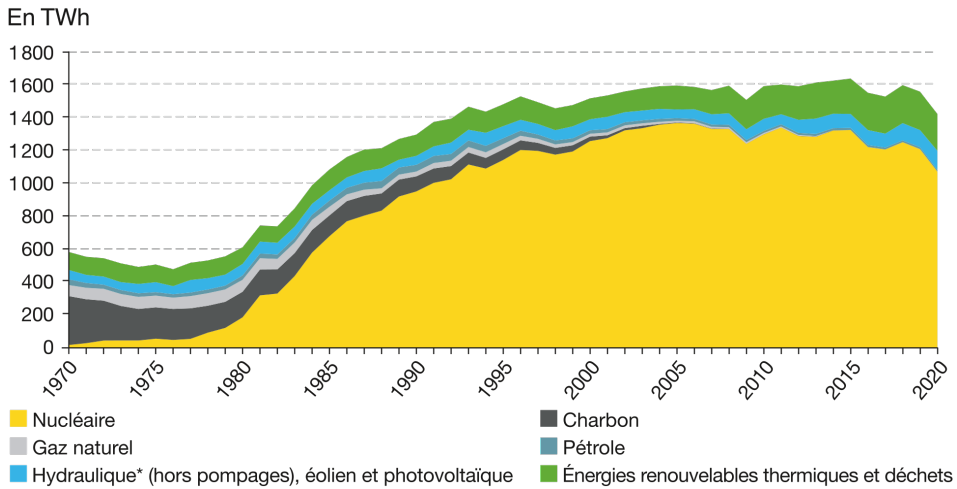
SORBONNE UNIVERSITÉ ISIR INSTITUT DES SYSTÈMES INTELLIGENTS ET DE ROBOTIQUE

Datakalab

Cloud Computing



(a) cloud energy consumption [1]



(b) French power production

Figure: Industrial GPUs draw less power than personal ones.

Edge Devices and MCU

Table: Comparison between different computing devices in terms of power consumption.

build	data-center rack	next gen home pc	home pc	ST MCU
power consumption	3000W	1000W	300W	$\approx$ 2.5W

source : ST technical report

Outline

Small backbones

Defining an efficient Neural Network starts by the right backbone. The current state of the art ranging from faster to more accurate is covered by MobileNet V3 [17] and EfficientNet V2 [31]. Their specificities are:

- depthwise convolutional layers
- silu activation functions
- usage of cut-mix and various data-augmentation
- usage of larger data corpus
- squeeze and excitation (for efficientnet)
- optimization of layers size through NAS

SORBONNE UNIVERSITÉ ISIR INSTITUT DES SYSTÈMES INTELLIGENTS ET DE ROBOTIQUE

Datakalab

Small Transformers

Transformers have achieved very impressive performance in the last few years. They should also be less costly to infer as they require way less floating point operations to run. However they still suffer from two drawbacks

- more complex intermediate operations (attention, gelu, softmax and layer normalization)
- very costly training

Although the last point has been improved in [34]

Training Cost Matters

We spend more time training models than actually inferring with them on real use-cases. Still, very few research is dedicated to lowering the cost of neural networks training. We list here some references on the matter:

- Net2Net [8] for faster training
- CaIT [34] training with fewer data for image transformers

Outline

Motivation

Many hardware only support integer inference and more specifically int8 such as micro controller units (mcu). Furthermore some larger devices leverage efficiently such representations like the Nvidia A100. This is why we are interested in quantization which consists in converting float32 operations into lower bit-width representations.

Principle

Let X be a tensor, if we note Q the quantization operator, then $Q(X) \in \{-2^{b-1}-1, \ldots, 2^{b-1}-1\}$ et $X \in]-\alpha; \alpha[$. The quantization operator is defined as follows

$$Q: \quad X \quad \mapsto \quad \left\lfloor X \times \frac{2^{b-1}-1}{\alpha} \right\rceil \in \{-2^{b-1}-1, \ldots, 2^{b-1}-1\} \tag{1}$$

The equation of the operation defined by a layer

$$\begin{array}{rccc} f: & \mathcal{A}^{d_e} & \rightarrow & \mathcal{B}^{d_s} \\ & X & \mapsto & \sigma(WX+B) \end{array} \tag{2}$$

becomes

$$\begin{array}{rccc} f: & \mathcal{A}^{d_e} & \rightarrow & \mathcal{B}^{d_s} \\ & X & \mapsto & \sigma(Q^{-1}(Q(W)Q(X))+B) \end{array} \tag{3}$$

See [19, 20]

SORBONNE UNIVERSITÉ ISIR INSTITUT DES SYSTÈMES INTELLIGENTS ET DE ROBOTIQUE

Datakalab

Symmetric vs Asymmetric

In equation 1, the quantized support is symmetric and any distribution, even an asymmetric one, is mapped to a symmetric distribution. To tackle this limitation, it is common to use a zero-point. Equation 1 becomes

$$Q: \quad X \quad \mapsto \quad \left\lfloor z + X \times \frac{2^{b-1}-1}{\alpha} \right\rceil \in \{-2^{b-1}-1, \ldots, 2^{b-1}-1\} \tag{4}$$

The zero point z centers the distribution.
See [19, 26]

Per-channel vs Per-tensor

The operators defined previously use a single scaling factor for the entire tensor. This is called **per-tensor** quantization. It is very common, in order to improve the final accuracy, to use **per-channel** quantization for weight tensors. This simply consists in using a vector of scaling factors per-output channel.
See [41, 2]

Accumulators

Following the layer definition from equation 3, if we quantize in 8 bits, for instance, then multiplying two int8 values may overflow. To tackle this limitation , it is a common practice to use accumulators with larger bit width. In other words multiplications are performed with low bit width but sums are performed with larger bit width. Very few research articles aim at solving this problem [27].

PTQ techniques

There is a wide range of already trained neural networks as well as networks that are or will be trained on proprietary data. In other words, networks with correct weight values but training data is not available or available in low quantity.
To perform quantization in such instances, we use **Post-Training Quantization**.
See the practice session + [41, 2, 3, 10, 12, 13, 42, 26]

QAT techniques

When data is available, the key challenge is to handle the rounding operator. Formally, the rounding operator has a zero gradient almost everywhere which cancels out the gradient descent optimization. Straight Through Estimation (STE) is the most effective trick known to this day in **Quantization Aware training** (QAT). The idea is simple: replace the real gradient by the identity operator. See [15, 18, 23, 30, 11, 14]

Summary of Quantization

Table: PTQ performance on ImageNet

modèle	W8/A8	W4/A4	Ternaire	Binaire
ResNet 50	$\approx 100\%$	$\approx 90\%$	-	-
MobileNet V2	$\approx 100\%$	$\approx 50\%$	-	-
EfficientNet B0	$\approx 100\%$	$\approx 50\%$	-	-

Table: QAT performance on ImageNet

modèle	W8/A8	W4/A4	Ternaire	Binaire
ResNet 50	$\approx 105\%$	$\approx 100\%$	$\approx 92\%$	$\approx 90\%$
MobileNet V2	$\approx 105\%$	$\approx 90\%$	$\approx 75\%$	-
EfficientNet B0	$\approx 100\%$	$\approx 95\%$	-	-

SORBONNE UNIVERSITÉ ISIR INSTITUT DES SYSTÈMES INTELLIGENTS ET DE ROBOTIQUE

Datakalab

Outline

Principle

Pruning consists in removing computations in the neural network graph. This can be done in a **structured** manner (removing filters, neurons, ...) or in an **unstructured** manner (removing single scalar operations). The latter usually achieves higher pruning rates but is harder to actually leverage as it relies on sparse matrix computations.
All in all, pruning requires to either measure the importance of computations in order to remove the least important ones, **magnitude-based pruning** or measure similarity in order to merge similar operations, **similarity-based pruning**.

SORBONNE UNIVERSITÉ ISIR INSTITUT DES SYSTÈMES INTELLIGENTS ET DE ROBOTIQUE

Datakalab

Magnitude-based

$$
\begin{cases}
\mathfrak{F}(X) = \mathfrak{W}^{(2)}\sigma(\mathfrak{W}^{(1)}X + \mathfrak{B}^{(1)}) + \mathfrak{B}^{(2)} \\
\mathfrak{W}^{(1)} = \begin{pmatrix} W^{(1)}_{1,1} & \dots & W^{(1)}_{1,N} \\ W^{(1)}_{3,1} & \dots & W^{(1)}_{3,N} \\ \vdots & \ddots & \vdots \\ W^{(1)}_{M,1} & \dots & W^{(1)}_{M,N} \end{pmatrix} \qquad \mathfrak{W}^{(2)} = \begin{pmatrix} W^{(2)}_{1,1} & W^{(2)}_{1,3} & \dots & W^{(2)}_{1,M} \\ \vdots & \vdots & \ddots & \vdots \\ W^{(2)}_{K,1} & W^{(2)}_{K,3} & \dots & W^{(2)}_{K,M} \end{pmatrix} \\
\mathfrak{B}^{(1)} = \begin{pmatrix} B^{(1)}_1 \\ B^{(1)}_3 \\ \vdots \\ B^{(1)}_M \end{pmatrix} \qquad \mathfrak{B}^{(2)} = \begin{pmatrix} B^{(2)}_1 + W^{(2)}_{2,1}\sigma(B^{(1)}_2) \\ B^{(2)}_3 + W^{(2)}_{2,2}\sigma(B^{(1)}_2) \\ \vdots \\ B^{(2)}_K + W^{(2)}_{2,K}\sigma(B^{(1)}_2) \end{pmatrix}
\end{cases}
\tag{5}
$$

See [40]

Similarity-based

$$
\begin{cases}
\mathfrak{F}(X) = \mathfrak{W}^{(2)}\sigma(\mathfrak{W}^{(1)}X + \mathfrak{B}^{(1)}) + \mathfrak{B}^{(2)} \\
\mathfrak{W}^{(1)} = \begin{pmatrix} \frac{W^{(1)}_{1,1}+W^{(1)}_{2,1}}{2} & \cdots & \frac{W^{(1)}_{1,N}+W^{(1)}_{2,N}}{2} \\ W^{(1)}_{3,1} & \cdots & W^{(1)}_{3,N} \\ \vdots & \ddots & \vdots \\ W^{(1)}_{M,1} & \cdots & W^{(1)}_{M,N} \end{pmatrix} \quad \mathfrak{W}^{(2)} = \begin{pmatrix} W^{(2)}_{1,1} + W^{(2)}_{1,2} & W^{(2)}_{1,3} & \cdots & W^{(2)}_{1,M} \\ \vdots & \vdots & \ddots & \vdots \\ W^{(2)}_{K,1} + W^{(2)}_{K,2} & W^{(2)}_{K,3} & \cdots & W^{(2)}_{K,M} \end{pmatrix} \\
\mathfrak{B}^{(1)} = \begin{pmatrix} \frac{B^{(1)}_1+B^{(1)}_2}{2} \\ B^{(1)}_3 \\ \vdots \\ B^{(1)}_M \end{pmatrix} \quad \mathfrak{B}^{(2)} = \begin{pmatrix} B^{(2)}_1 \\ B^{(2)}_3 \\ \vdots \\ B^{(2)}_K \end{pmatrix}
\end{cases}
\tag{6}
$$

See [39, 38]

Fine-tuning

To recover the accuracy when the pruning rate is too high, fine-tuning is the standard approach. It goes from a full re-training process to a few optimization steps. This step is subject to hyper-parameter tuning like any training phase. Some pruning techniques even use several full-training in order to achieve the highest possible accuracy.

Summary of Pruning

Table: Empirical pruning performance

modèle	tache	% paramètres retirés
ResNet 56	Cifar 10	≈90%
Wide ResNet 28-10	Cifar 10	≈75%
ResNet 50	IamgeNet	≈60%
MobileNet V2	ImageNet	≈40 %

Bibliography I

[1] Anders Andrae and Tomas Edler. "On Global Electricity Usage of Communication Technology: Trends to 2030". In: Challenges 6 (Apr. 2015), pp. 117–157. DOI: 10.3390/challe6010117.

[2] Ron Banner, Yury Nahshan, and Daniel Soudry. "Post training 4-bit quantization of convolutional networks for rapid-deployment". In: (2019), pp. 7950–7958.

[3] Yaohui Cai et al. "Zeroq: A novel zero shot quantization framework". In: (2020), pp. 13169–13178.

[4] Yuanhao Cai et al. "Learning delicate local representations for multi-person pose estimation". In: ECCV. Springer. 2020, pp. 455–472.

[5] Zhe Cao et al. "Realtime multi-person 2d pose estimation using part affinity fields". In: CVPR. 2017, pp. 7291–7299.

[6] Nicolas Carion et al. "End-to-end object detection with transformers". In: ECCV. Springer. 2020, pp. 213–229.

[7] Liang-Chieh Chen et al. "Rethinking atrous convolution for semantic image segmentation". In: arXiv preprint arXiv:1706.05587 (2017).

[8] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. "Net2net: Accelerating learning via knowledge transfer". In: arXiv preprint arXiv:1511.05641 (2015).

Bibliography II

[9] Zhe Chen et al. "Vision Transformer Adapter for Dense Predictions". In: arXiv preprint arXiv:2205.08534 (2022).

[10] Yoni Choukroun et al. "Low-bit Quantization of Neural Networks for Efficient Inference.". In: (2019), pp. 3009–3018.

[11] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. "Binaryconnect: Training deep neural networks with binary weights during propagations". In: (2015), pp. 3123–3131.

[12] Jun Fang et al. "Post-training piecewise linear quantization for deep neural networks". In: (2020), pp. 69–86.

[13] Sahaj Garg et al. "Confounding tradeoffs for neural network quantization". In: arXiv preprint arXiv:2102.06366 (2021).

[14] Philipp Gysel, Mohammad Motamedi, and Soheil Ghiasi. "Hardware-Oriented Approximation of Convolutional Neural Networks". In: ICLR workshop (2016).

[15] Philipp Gysel et al. "Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks". In: IEEE transactions on neural networks and learning systems 29.11 (2018), pp. 5784–5789.

Bibliography III

[16] Kaiming He et al. "Deep residual learning for image recognition". In: CVPR (2016), pp. 770–778.

[17] Andrew Howard et al. "Searching for mobilenetv3". In: (2019), pp. 1314–1324.

[18] Itay Hubara et al. "Binarized neural networks". In: NeurIPS 29 (2016).

[19] Benoit Jacob et al. "Quantization and training of neural networks for efficient integer-arithmetic-only inference". In: (2018), pp. 2704–2713.

[20] Raghuraman Krishnamoorthi. "Quantizing deep convolutional networks for efficient inference: A whitepaper". In: arXiv preprint arXiv:1806.08342 (2018).

[21] Xiangtai Li et al. "Semantic flow for fast and accurate scene parsing". In: ECCV. Springer. 2020, pp. 775–793.

[22] Tingting Liang et al. "CBNetV2: A Composite Backbone Network Architecture for Object Detection". In: arXiv preprint arXiv:2107.00420 (2021).

[23] Zhouhan Lin et al. "Neural networks with few multiplications". In: arXiv preprint arXiv:1510.03009 (2015).

[24] Xiaodong Liu et al. "Very deep transformers for neural machine translation". In: arXiv preprint arXiv:2008.07772 (2020).

Bibliography IV

[25] Rohit Mohan and Abhinav Valada. "Efficientps: Efficient panoptic segmentation". In: International Journal of Computer Vision 129.5 (2021), pp. 1551–1579.

[26] Markus Nagel, Mart van Baalen, et al. "Data-free quantization through weight equalization and bias correction". In: ICCV (2019), pp. 1325–1334.

[27] Renkun Ni et al. "Wrapnet: Neural net inference with ultra-low-resolution arithmetic". In: ICLR (2020).

[28] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: IEEE Transactions on Pattern Analysis and Machine Intelligence 39.6 (2017), pp. 1137–1149. DOI: 10.1109/TPAMI.2016.2577031.

[29] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: Advances in neural information processing systems 27 (2014).

[30] Shyam A Tailor, Javier Fernandez-Marques, and Nicholas D Lane. "Degree-quant: Quantization-aware training for graph neural networks". In: ICLR (2021).

[31] Mingxing Tan and Quoc Le. "Efficientnetv2: Smaller models and faster training". In: ICML. PMLR. 2021, pp. 10096–10106.

Bibliography V

[32] Mingxing Tan and Quoc V Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: ICML (2019), pp. 6105–6114.

[33] Mingxing Tan, Ruoming Pang, and Quoc V Le. "Efficientdet: Scalable and efficient object detection". In: CVPR. 2020, pp. 10781–10790.

[34] Hugo Touvron et al. "Going deeper with Image Transformers". In: arXiv preprint arXiv:2103.17239 (2021).

[35] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors". In: arXiv preprint arXiv:2207.02696 (2022).

[36] Felix Wu et al. "Pay Less Attention with Lightweight and Dynamic Convolutions". In: International Conference on Learning Representations. 2018.

[37] Yufei Xu et al. "ViTPose: Simple Vision Transformer Baselines for Human Pose Estimation". In: arXiv preprint arXiv:2204.12484 (2022).

[38] Edouard Yvinec et al. "RED++: Data-Free Pruning of Deep Neural Networks via Input Splitting and Output Merging". In: IEEE Transactions on Pattern Analysis and Machine Intelligence (2022).

Bibliography VI

[39] Edouard Yvinec et al. "RED: Looking for Redundancies for Data-FreeStructured Compression of Deep Neural Networks". In: NeurIPS 34 (2021), pp. 20863–20873.

[40] Edouard Yvinec et al. "SInGE: Sparsity via Integrated Gradients Estimation of Neuron Relevance". In: NeurIPS (2022).

[41] Edouard Yvinec et al. "SPIQ: Data-Free Per-Channel Static Input Quantization". In: arXiv preprint arXiv:2203.14642 (2022).

[42] Ritchie Zhao et al. "Improving Neural Network Quantization without Retraining using Outlier Channel Splitting". In: (2019), pp. 7543–7552.