Development of a Lightweight Model for Detecting Solitary-Bee Buzz Using Pruning and Quantization for Edge Deployment

Ryo Yagi
yagi-ryo143@g.ecc.u-tokyo.ac.jp
The University of Tokyo
Tokyo, Japan

David Susič
David.Susic@ijs.si
Jožef Stefan Institute
Ljubljana, Slovenia

Maj Smerkol
maj.smerkol@ijs.si
Jožef Stefan Institute
Ljubljana, Slovenia

Miha Finžgar
miha.finzgar@senso4s.com
Senso4s
Trzin, Slovenia

Anton Gradišek
anton.gradisek@ijs.si
Jožef Stefan Institute
Ljubljana, Slovenia

Abstract

Passive acoustic monitoring is increasingly applied in studies of pollinators, both for biodiversity assessment and for the conservation of endangered species. A major challenge is that continuous recording generates large volumes of audio data, making centralized processing impractical. Edge computing offers a promising alternative, provided that the models are optimized for resource constraints of edge devices while maintaining acceptable performance and efficiency. In this work, which is our initial study of the edge computing approach, we developed and evaluated compact classifiers for detecting buzzes of solitary bees, extending previous work on acoustic monitoring. We systematically apply pruning and quantization to multiple models, exploring a range of compression settings. Performance is assessed in terms of mean F1-score and on-disk size under both cross-validation and leave-one-location-out protocols. Results indicate that substantial reductions in model size can be achieved with a minimal loss of performance, and that the optimal trade-offs depend on the evaluation setting; for example, in cross-validation, a 25.2 MiB baseline reaches 96.2% F1, while a 0.062 MiB model attains 92.5%, achieving an approximately 400-fold reduction in size with less than a 4-percentage-point drop. By analyzing the Pareto front of F1 vs. model size trade-offs, we identify configurations that balance robustness and resource constraints. Our early findings demonstrate the feasibility of deploying edge-ready acoustic models for scalable pollinator monitoring.

Keywords

edge deployment, lightweight model, pruning, quantization, bees

1 Introduction

Bees are widely recognized as major pollinators - animal pollinators including honey bees contribute to yield in 75% of key crop species and an estimated 35% of global crop production [1]. This indicates the importance of pollinator monitoring and protection.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
Information Society 2025, Ljubljana, Slovenia
© 2025 Copyright held by the owner/author(s).
https://doi.org/10.70314/is.2025.skui.6788

Passive acoustic monitoring (PAM) is a non-invasive approach that continuously records environmental sound with deployed microphones to monitor animal activity. Because it reduces manual surveys and can operate continuously across space and time—even at night and under inclement weather—it has gained attention as a cost-effective biodiversity monitoring technology. PAM has been widely adopted for multiple taxa such as birds and bats; in ornithology, for example, the deep-learning system BirdNET [2] is already used operationally to identify species from passively collected field recordings. PAM is also applied to bee behavior monitoring: in social bees (such as honeybees or bumblebees), microphones and accelerometers placed inside or outside hives enable non-invasive, continuous surveillance of queen presence, swarming cues, and robbing [3]. For solitary bees, recordings at the entrance of nesting boxes are used to detect buzzing and to characterize presence/absence and activity rhythms [4].

In acoustic approaches for bee state monitoring, machine learning has been widely used to automatically determine activity and behavioral states from audio recordings. Prior work includes both classical machine-learning pipelines and deep-learning methods. Classical approaches such as SVM, k-NN, and Random Forests have been shown to be practical and effective [5, 6]. Meanwhile, several studies suggest that CNN-based deep learning models achieve superior performance compared with traditional machine-learning methods [7, 8].

However, if all long-term, continuous PAM recordings are uploaded to the cloud, features such as mel spectrograms and MFCCs are extracted there, and then analyzed using machine learning or deep learning models, the resulting data volumes become extremely large, which, in a centralized cloud-only workflow, (i) inflates communication cost by requiring all long-duration audio to be uploaded [9], (ii) raises privacy concerns as incidental human speech can accumulate in the cloud [10], (iii) introduces round-trip latency for feature extraction and inference that impedes timely detection, and (iv) exposes scalability limits as storage and compute demands grow with multi-site, long-term deployments. To address these issues, we developed a high-accuracy, lightweight deep model designed for edge deployment, capable of on-device preprocessing and inference for recorded audio. Here, the term lightweight refers to memory (both RAM and storage), but in a broader view it also refers to CPU/GPU requirements, latency requirements, and even battery constraints, which is beyond the scope of this paper. In our intended operation, audio is processed on-device and only the result is sent to the cloud, enabling multi-site, long-term monitoring with reduced storage cost and latency, while preserving privacy and power efficiency.

As a first step toward edge-based bee monitoring with PAM, we designed and evaluated a lightweight CNN specialized for solitary-bee buzz detection (binary classifier distinguishing between buzz and no-buzz). To compress the model, we applied compression techniques such as structured pruning and int8 post-training quantization when appropriate, and we quantified the size–accuracy trade-offs under edge-oriented constraints.

2 Methodology

2.1 Dataset

We used the dataset collected for the purpose of the study by Susič et al. [4]. This dataset comprises acoustic recordings from nesting boxes of solitary bees (predominantly Osmia spp.) collected through a citizen-science project carried out in the Bela Krajina region in the southeastern Slovenia. The recordings were gathered from March 15 to May 26, 2023, resulting in 62 long recordings across seven sites, with a mean duration of 6 ± 2.5 hours per recording. For the purpose of this study, three recordings in total were randomly selected from different locations.

The recordings were converted to mono-channel audio, segmented into 4 s windows with 2 s overlap, transformed into Mel spectrograms (128×128) configured to cover 50–1450 Hz, and standardized using the mean and standard deviation across the dataset. For labeling, two annotators inspected the spectrograms and assigned buzz=1 or no-buzz=0.

2.2 Neural Network Architecture

We addressed binary detection of solitary-bee buzzing from Mel spectrograms. With memory-constrained edge deployment, we evaluate four lighter CNNs compared to the ResNet-9 used in [4]. Specifically, we consider MobileNetV2 [11] and three custom lightweight architectures named BeeNet1, BeeNet2, and BeeNet3, that adopt a depthwise separable convolutional design similar to MobileNetV1 [12]. Model sizes and parameter counts are summarized in Table 1 and the architectural details of the BeeNet variants are provided in Table 2. In all architectures, each convolutional layer is followed by batch normalization, BatchNorm, and ReLU activation, whereas dw stands for depthwise convolution. For MobileNetV2, we use the standard backbone and adapt it to spectrograms by converting the first convolution to a 1-channel input and replacing the final linear layer with a $1280 \rightarrow 2$ classifier. All other layers remain identical to the original MobileNetV2.

While the ResNet-9 approach achieves an F1-score exceeding 95% under five-fold cross-validation on the dataset [4], its 25.2 MiB size renders its deployment on a memory-limited edge devices impractical. Accordingly, we designed and configured compact CNNs (MobileNetV2 and the BeeNet family) and, as detailed below, applied quantization and pruning to systematically evaluate the accuracy–model-size trade-off.

The aim of this study is to clarify accuracy as a function of model size and the effects of lightweighting techniques under strict model-size constraints assuming deployment on MCUs. Accordingly, we adopt a lightweight and relatively simple architecture, with the smallest model containing approximately 6k parameters.

Table 1: Parameter counts and model sizes of the models used in this study

	ResNet-9	Mobilenetv2	Beenet 1	Beenet 2	Beenet 3
Parameters (k)	6585.5	2225.9	50.2	17.6	6.4
Model size (MiB)	25.2	8.7	0.215	0.084	0.036

2.3 Model Compression Methods

Deploying deep neural networks on memory-constrained edge devices necessitates model compression. We examined two complementary techniques: quantization and pruning.

2.3.1 Quantization. Quantization maps floating-point weights and activations to low-bit integers, thereby reducing model size and computation at inference. Here, we adopted post-training quantization (PTQ) and converted the trained network to int8 without additional training. We used the QNNPACK backend in PyTorch for ARM targets. To minimize both saturation (clipping) and rounding error under the 8-bit representation and mitigate accuracy degradation, we performed calibration with up to 300 batches of representative inputs to estimate the scale and zero-point.

2.3.2 Pruning. Pruning reduces model complexity by deleting parameters deemed unimportant, thereby decreasing memory and compute complexity without retraining from scratch. Pruning can be categorized into structured and unstructured approaches. We adopted structured pruning to realize memory savings and speed-ups on commodity hardware, as unstructured sparsity typically requires specialized hardware or software support to translate sparsity into acceleration [13].

Our pruning pipeline followed Han et al. [14]: (1) train, (2) prune, and (3) retrain (fine-tune). For filter (i.e., output-channel) selection, we followed the idea of Li et al. [15], ranking convolutional filters by the L1 norm of their weights and removing those with the smallest scores. We implemented this using PyTorch's torch-pruning, configuring the MagnitudePruner with L1-based importance. The selection of filters pruned was performed globally across layers. The target was controlled by a pruning ratio p; under channel-wise pruning, the resulting parameter-reduction rate was approximately $1 - (1 - p)^2$ [16, 17].

2.4 Experimental Setup

2.4.1 Model Performance Evaluation Metrics. Because we were dealing with a class-imbalanced dataset (more no-buzz than buzz), we used the F1-score as the primary metric. F1 is the harmonic mean of precision and recall, enabling balanced assessment under imbalance.

2.4.2 Evaluation Protocols. We evaluated the buzz-detecting models using two protocols, following [4]: cross-validation (CV) and leave-one-location-out (LOLO). The first approach is a standard test in machine-learning studies whereas the second one shows how well the model generalizes to the data coming from a previously unseen location with potentially different background noise. For CV, annotated segments (4 s windows) were partitioned into five folds; models were trained on four folds and evaluated on the remaining fold, and we reported the mean F1 across folds. Stratification ensures balanced distributions of the buzz/no-buzz classes and the three locations. To mitigate temporal leakage, we performed a time-aware data split.

Table 2: The architectures of BeeNet1, BeeNet2, BeeNet3

BeeNet1			BeeNet2			BeeNet3		
Type / Stride	Filter Shape	Input Size	Type / Stride	Filter Shape	Input Size	Type / Stride	Filter Shape	Input Size
Conv / s1	$3 \times 3 \times 1 \times 32$	$128 \times 128 \times 1$	Conv / s1	$3 \times 3 \times 1 \times 32$	$128 \times 128 \times 1$	Conv / s1	$3 \times 3 \times 1 \times 32$	$128 \times 128 \times 1$
MaxPool / s2	Pool 2×2	$128 \times 128 \times 32$	MaxPool / s2	Pool 2×2	$128 \times 128 \times 32$	MaxPool / s2	Pool 2×2	$128 \times 128 \times 32$
Conv dw / s1	$3 \times 3 \times 32$ dw	$64 \times 64 \times 32$	Conv dw / s1	$3 \times 3 \times 32$ dw	$64 \times 64 \times 32$	Conv dw / s1	$3 \times 3 \times 32$ dw	$64 \times 64 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 32$	$64 \times 64 \times 32$	Conv / s1	$1 \times 1 \times 32 \times 32$	$64 \times 64 \times 32$	Conv / s1	$1 \times 1 \times 32 \times 32$	$64 \times 64 \times 32$
MaxPool / s2	Pool 2×2	$64 \times 64 \times 32$	MaxPool / s2	Pool 2×2	$64 \times 64 \times 32$	MaxPool / s2	Pool 2×2	$64 \times 64 \times 32$
Conv dw / s1	$3 \times 3 \times 32$ dw	$32 \times 32 \times 32$	Conv dw / s1	$3 \times 3 \times 32$ dw	$32 \times 32 \times 32$	Conv dw / s1	$3 \times 3 \times 32$ dw	$32 \times 32 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$32 \times 32 \times 32$	Conv / s1	$1 \times 1 \times 32 \times 64$	$32 \times 32 \times 32$	Conv / s1	$1 \times 1 \times 32 \times 64$	$32 \times 32 \times 32$
MaxPool / s2	Pool 2×2	$32 \times 32 \times 64$	MaxPool / s2	Pool 2×2	$32 \times 32 \times 64$	MaxPool / s8	Pool 8×8	$32 \times 32 \times 64$
Conv dw / s1	$3 \times 3 \times 64$ dw	$16 \times 16 \times 64$	Conv dw / s1	$3 \times 3 \times 64$ dw	$16 \times 16 \times 64$	FC / s1	1024×2	$4 \times 4 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$16 \times 16 \times 64$	Conv / s1	$1 \times 1 \times 64 \times 128$	$16 \times 16 \times 64$	Softmax / s1	Classifier	$1 \times 1 \times 2$
MaxPool / s2	Pool 2×2	$16 \times 16 \times 128$	MaxPool / s4	Pool 4×4	$16 \times 16 \times 128$			
Conv dw / s1	$3 \times 3 \times 128$ dw	$8 \times 8 \times 128$	FC / s1	2048×2	$4 \times 4 \times 128$			
Conv / s1	$1 \times 1 \times 128 \times 256$	$8 \times 8 \times 128$	Softmax / s1	Classifier	$1 \times 1 \times 2$			
MaxPool / s4	Pool 4×4	$8 \times 8 \times 256$						
FC / s1	1024×2	$2 \times 2 \times 256$						
Softmax / s1	Classifier	$1 \times 1 \times 2$						

LOLO assessed generalization across sites: models were trained on data from two of the three locations and evaluated on the held-out location, reporting the mean F1 across the three possible holds.

2.4.3 Hyperparameters. We trained the models with cross-entropy loss and the Adam optimizer, using a 1-cycle learning-rate schedule (maximum LR = 0.001), gradient clipping at 0.1, batch size 64, and 20 epochs. Compared to [4], the only change was increasing the number of epochs from 10 to 20. For pruning fine-tuning, we trained for 10 epochs with a fixed learning rate of 0.0001 and no scheduler. We compared the pruning ratios p of 0% (no pruning), 20%, 30%, and 50%.

3 Results

3.1 F1 vs. Model Size

For each model, we trained and evaluated a variety of combinations of pruning ratios and quantizations. Table 3 reports the mean F1 and on-disk model size (in MiB) for each setting. Figure 1 shows the plot of all configurations in the F1 – model-size plane for CV and LOLO, respectively, with the global Pareto front indicating the best trade-offs between model performance and its size denoted by a dashed line.

Even under tight memory budgets (< 100 KiB), competitive accuracy is achievable. For example, BeeNet1 (int8, p=0) attains 0.062MiB with CV F1 of 92.5% and LOLO F1 of 85.7%. Relative to ResNet-9 (float32, p=0), this represents an $\sim 400\times$ reduction in model size while keeping F1 within 4 percentage points in both protocols, which is really promising for future edge deployment.

Performance degradation from int8 quantization is small: across many settings the F1 drop is about 1 percentage point (pp). With pruning, larger models exhibit smaller accuracy losses as p increases. For example, at p=50% ResNet-9 (float32) decreases only from 96.2% to 95.1% in CV and from 89.5% to 87.6% in LOLO, a decline of $\approx$ 2 percentage points in total. By contrast, the more compact BeeNet family is more sensitive: accuracy degrades markedly with p, and at p=50% most configurations lose ≥ 4 pp.

Inspection of the global Pareto front shows that many frontier points correspond to unpruned float32 or int8 models. At a fixed memory budget, lightly pruned or unpruned lightweight architectures often achieve higher accuracy than heavily pruned larger networks, indicating that purpose-built small models are preferable to aggressive pruning under the same size constraint.

A note on MobileNetV2 at p=30%: the trained model degenerated to predicting *no-buzz* for almost all inputs. This behavior may stem from a strong structured reduction under class imbalance and warrants further investigation.

4 Conclusions

We addressed buzz detection in acoustic recordings from solitary-bee nesting boxes, aiming to develop deep-learning models suitable for memory-constrained edge deployment. We designed or selected five CNN architectures and systematically measured the performance vs. model-size trade-offs under quantization and structured pruning. As a result, we obtained sub-100 KiB models achieving F1 scores of at least 92% in CV and 85% in LOLO experiments, indicating the feasibility and strong potential of accurate on-device inference.

For future work, we plan to train the models on additional datasets that we have collected to improve robustness and to deploy the models on real edge devices. Because our compression pipeline relied on simple techniques, we anticipate further gains by adopting a broader set of compression methods, such as knowledge distillation [18], quantization-aware training (QAT) [19], and neural architecture search (NAS) [20] to optimize model architectures under memory constraints, including number and sizes of the filters.

Acknowledgements

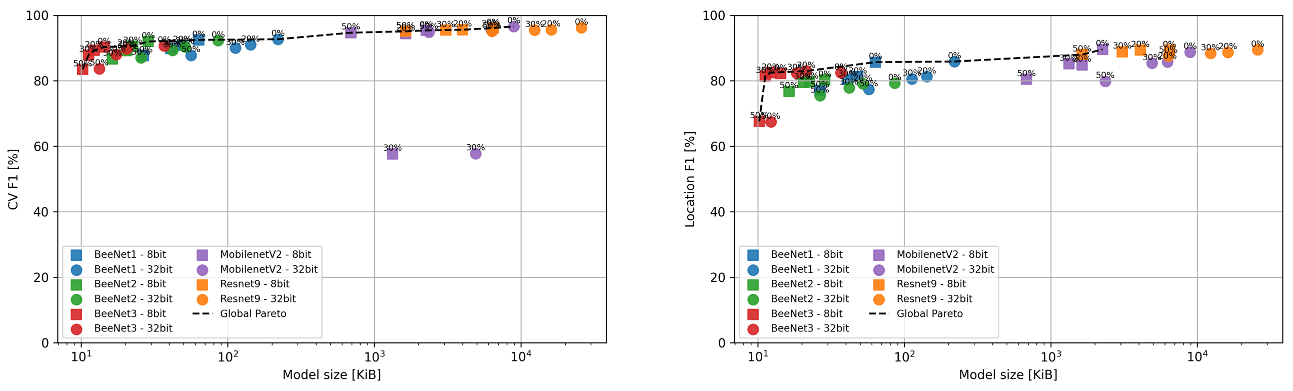
The authors acknowledge the Slovenian Research and Innovation Agency, grants PR-10495 and J7-50040, and Basic core funding P2-0209.

References

[1] Tom D Breeze, Alison P Bailey, Kelvin G Balcombe, and Simon G Potts. 2011. Pollination services in the uk: how important are honeybees? *Agriculture, Ecosystems & Environment*, 142, 3-4, 137–143.
[2] Stefan Kahl, Connor M. Wood, Maximilian Eibl, and Holger Klinck. 2021. Birdnet: a deep learning solution for avian diversity monitoring. *Ecological Informatics*, 61, 101236. DOI: https://doi.org/10.1016/j.ecoinf.2021.101236.
[3] Mahsa Abdollahi, Pierre Giovenazzo, and Tiago H Falk. 2022. Automated beehive acoustics monitoring: a comprehensive review of the literature and recommendations for future work. *Applied Sciences*, 12, 8, 3920.

Table 3: Comparison of F1 scores and model sizes with quantization and pruning applied for five CNNs (ResNet-9, MobileNetV2, and BeeNet1/2/3)

	CV								**LOLO**							
	Pruning ratio (%)								*Pruning ratio (%)*							
Model (Quant.)	0		20		30		50		0		20		30		50	
	F1 (%)	Size (MiB)	F1 (%)	Size (MiB)	F1 (%)	Size (MiB)	F1 (%)	Size (MiB)	F1 (%)	Size (MiB)	F1 (%)	Size (MiB)	F1 (%)	Size (MiB)	F1 (%)	Size (MiB)
ResNet-9 (float32)	96.2	25.2	95.6	15.7	95.5	12.1	95.1	6.2	89.5	25.2	88.7	15.8	88.4	12.0	87.6	6.2
ResNet-9 (int8)	96.1	6.3	95.6	3.9	95.5	3.0	95.1	1.6	89.4	6.3	89.4	4.0	88.9	3.0	88.0	1.6
MobileNetV2 (float32)	96.6	8.7	95.6	6.1	57.8	4.8	94.9	2.3	88.8	8.7	85.8	6.1	85.4	4.8	79.9	2.3
MobileNetV2 (int8)	95.4	2.2	94.4	1.6	57.7	1.3	94.7	0.677	89.6	2.2	84.9	1.6	85.2	1.3	80.5	0.665
BeeNet1 (float32)	92.7	0.215	91.0	0.140	90.0	0.110	87.8	0.055	85.9	0.215	81.3	0.139	80.6	0.110	77.4	0.056
BeeNet1 (int8)	92.5	0.062	90.9	0.048	89.9	0.040	87.7	0.026	85.7	0.062	81.3	0.047	80.4	0.040	77.1	0.026
BeeNet2 (float32)	92.3	0.084	90.4	0.050	89.3	0.041	87.0	0.025	79.3	0.084	79.1	0.051	77.9	0.041	75.5	0.026
BeeNet2 (int8)	92.0	0.028	90.6	0.022	89.2	0.020	86.7	0.016	80.3	0.028	79.7	0.022	79.5	0.020	76.8	0.016
BeeNet3 (float32)	90.7	0.036	89.8	0.020	88.0	0.017	83.7	0.013	82.5	0.036	83.0	0.021	82.4	0.018	67.5	0.012
BeeNet3 (int8)	90.3	0.014	89.3	0.012	87.9	0.011	83.5	0.010	82.2	0.014	82.5	0.012	81.5	0.011	67.6	0.010



(a) Cross-validation (CV) **(b) Leave-one-location-out (LOLO)**

Figure 1: F1–model-size trade-offs with the global Pareto frontier under CV and LOLO

[4] David Susič, Johanna A. Robinson, Danilo Bevk, and Anton Gradišek. 2025. Acoustic monitoring of solitary bee activity at nesting boxes. *Ecological Solutions and Evidence*, 6, 3, e70080. DOI: https://doi.org/10.1002/2688-8319.70080.

[5] Alison Pereira Ribeiro, Nádia Felix Felipe da Silva, Fernanda Neiva Mesquita, Priscila de Cássia Souza Araújo, Thierson Couto Rosa, and José Neiva Mesquita-Neto. 2021. Machine learning approach for automatic recognition of tomato-pollinating bees based on their buzzing-sounds. *PLOS Computational Biology*, 17, 9, (Sept. 2021), 1–21. DOI: 10.1371/journal.pcbi.1009426.

[6] Antonio Robles-Guerrero, Tonatiuh Saucedo-Anaya, Carlos A. Guerrero-Mendez, Salvador Gómez-Jiménez, and David J. Navarro-Solís. 2023. Comparative study of machine learning models for bee colony acoustic pattern classification on low computational resources. *Sensors*, 23, 1. DOI: 10.3390/s23010460.

[7] Jaehoon Kim, Jeongkyu Oh, and Tae-Young Heo. 2021. Acoustic scene classification and visualization of beehive sounds using machine learning algorithms and grad-cam. *Mathematical Problems in Engineering*, 2021, 1, 5594498.

[8] Vladimir Kulyukin, Sarbajit Mukherjee, and Prakhar Amlathe. 2018. Toward audio beehive monitoring: deep learning vs. standard machine learning in classifying beehive audio samples. *Applied Sciences*, 8, 9, 1573.

[9] Carrie C Wall, Samara M Haver, Leila T Hatch, Jennifer Miksis-Olds, Rob Bochenek, Robert P Dziak, and Jason Gedamke. 2021. The next wave of passive acoustic data management: how centralized access can enhance science. *Frontiers in Marine Science*, 8, 703682.

[10] Benjamin Cretois, Carolyn M Rosten, and Sarab S Sethi. 2022. Voice activity detection in eco-acoustic data enables privacy protection and is a proxy for human disturbance. *Methods in Ecology and Evolution*, 13, 12, 2865–2874.

[11] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4510–4520.

[12] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[13] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. 2024. A survey on deep neural network pruning: taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46, 12, 10558–10578. DOI: 10.1109/TPAMI.2024.3447085.

[14] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

[15] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2016. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*.

[16] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. 2023. Depgraph: towards any structural pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[17] Gongfan Fang and contributors. 2023. Torch-pruning: structural pruning for pytorch. (2023). https://github.com/VainF/Torch-Pruning.

[18] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

[19] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2704–2713.

[20] Barret Zoph and Quoc V. Le. 2017. Neural architecture search with reinforcement learning. (2017). https://arxiv.org/abs/1611.01578 arXiv: 1611.01578 [cs.LG].