MetaCLBench: Benchmarking the On-Device Adaptation Cost of Meta-Continual Learning

Sijia Li[1]* **Young D. Kwon**[2]* **Lik-Hang Lee**[3] **Pan Hui**[4]
[1]University of Surrey [2]University of Cambridge
[3]The Hong Kong Polytechnic University [4]HKUST (Guangzhou)
sijia.li@surrey.ac.uk ydk21@cam.ac.uk

Abstract

Meta-Continual Learning (Meta-CL) enables models to learn new classes from a few labelled samples, which makes it attractive for IoT applications where labelling is costly. However, existing evaluations report accuracy and largely ignore whether adaptation is feasible on resource-constrained hardware. We present *MetaCLBench*, a benchmark that evaluates Meta-CL methods for accuracy together with the deployment-critical costs of learning (peak memory, end-to-end latency and energy) on real edge devices with 512 MB to 4 GB of RAM. We evaluate six Meta-CL methods across three architectures (CNN, YAMNet, ViT) and five image and audio datasets. Depending on the dataset, up to three of the evaluated methods run out of memory on a 1 GB device, which sharply narrows the set of deployable options. LifeLearner reaches near-oracle accuracy while using 2.54–7.43× less energy than the oracle baseline, and larger pretrained architectures do not consistently improve the Meta-CL accuracy–efficiency trade-off. We distil these findings into deployment guidelines and release the framework.[1]

1 Introduction

Internet-of-Things (IoT) systems increasingly require machine-learning models to adapt after deployment as environments, users and sensing conditions evolve [5, 32]. Continual Learning (CL) addresses this setting by learning from sequentially arriving data while retaining previously acquired knowledge, thereby mitigating catastrophic forgetting [43]. However, conventional CL can be difficult to deploy on edge devices because adaptation must operate under tight memory, computation and energy budgets while labelled samples for newly emerging classes are often scarce [35, 36].

Meta-Continual Learning (Meta-CL) combines meta-learning with continual adaptation to address this challenge. Through meta-training on source tasks, Meta-CL methods learn representations or adaptation mechanisms that enable rapid learning of unseen classes from only a few labelled examples while preserving previously acquired knowledge [3, 22]. Existing approaches such as OML [22], ANML [3], and their AIM-enhanced variants [38] have demonstrated strong continual-learning accuracy, and LifeLearner [35] adds hardware-aware latent replay with compression. Nevertheless, existing evaluations remain largely centred on predictive performance, leaving a practical question insufficiently explored: *does a Meta-CL method that achieves high accuracy remain viable when adaptation must run on the edge device itself?* Peak memory, adaptation latency and energy can change which methods are actually deployable. Prior work has typically studied either algorithmic performance or the system cost of individual on-device learning methods, rather than benchmarking both under a common Meta-CL protocol.

To investigate this question, we present *MetaCLBench*, a deployment-oriented benchmark for jointly evaluating Meta-CL accuracy and system cost. Rather than introducing a dataset, it provides a common protocol across representative Meta-CL methods, image and audio sensing tasks, three

*Equal contribution.
[1]https://github.com/theyoungkwon/MetaCLBench

40th Conference on Neural Information Processing Systems (NeurIPS 2026). Workshop: TS-LIMITS.

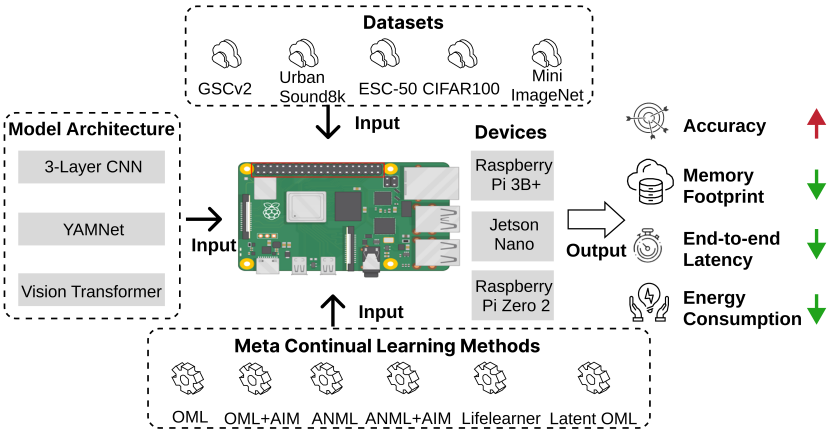


Figure 1: Overview of *MetaCLBench*: six Meta-CL methods and three architectures evaluated on five datasets and three edge devices, reporting accuracy, peak memory, latency and energy.

architecture families and real edge platforms from a 512 MB Raspberry Pi Zero 2 to a 4 GB Jetson Nano. For every configuration we measure accuracy, peak learning-time memory, end-to-end adaptation latency and energy consumption.

Our evaluation protocol and experiments show that algorithmic performance alone is insufficient to determine deployment viability. Concretely, we make three contributions. (i) We benchmark six Meta-CL methods over five datasets, three architectures and three device tiers, reporting peak learning-time memory, end-to-end latency and energy alongside accuracy, which prior Meta-CL evaluations often neglect. (ii) We introduce Latent OML, which uses LifeLearner's latent replay to OML-based ViT and YAMNet backbones and makes latent replay applicable beyond the CNN/ANML setting of prior work. (iii) We find that accuracy ranking does not predict deployability: AIM-enhanced methods fail with out-of-memory errors on a 1 GB device, latent-replay methods reach near-oracle accuracy at a fraction of the energy, and larger pretrained backbones do not consistently improve the accuracy–efficiency trade-off. We distil these results into practical guidelines (Section 4).

2 *MetaCLBench*

2.1 Benchmark Protocol

Task setting: *MetaCLBench* evaluates Meta-CL in a class-incremental few-shot setting. For each dataset, the class set is partitioned into disjoint meta-training and meta-testing classes, $C_{train} \cap C_{test} = \emptyset$. Meta-training is performed only on C_{train}, while deployment is evaluated on previously unseen classes from C_{test}. During meta-testing, classes arrive sequentially as a stream $S = (T_1, \ldots, T_N)$, and at most 30 labelled samples are provided for adapting to each new class. The learner must acquire the new class while retaining knowledge of classes encountered earlier in the stream.

Benchmark design: *MetaCLBench* evaluates this adaptation process along two complementary dimensions: **learning performance** and **deployment cost**. Rather than evaluating Meta-CL only on server-class hardware, we execute the meta-testing stage on real edge devices and measure the resources required during continual adaptation. Figure 1 summarises the benchmark across methods, datasets, architectures, and hardware platforms. Full task-construction and training details are provided in Appendix A.5.

2.2 Benchmark Coverage

MetaCLBench covers six representative Meta-CL algorithms, two sensing modalities, three model architectures, and three edge-device classes. We evaluate OML [22], ANML [3], their AIM-enhanced variants [38], and two latent-replay methods, LifeLearner [35] and Latent OML, which adopts LifeLearner's latent-replay mechanism for OML-based pretrained backbones, without its compression module. During meta-testing, the feature extractor remains frozen, latent activations are stored for replay, and only the OML prediction network is updated. Full implementation details are provided

Table 1: Representative CNN-based results on the Raspberry Pi 3B+: accuracy, peak learning-time memory, end-to-end latency and energy. Full results in Table 3.

Dataset	Method	Acc. (%) ↑	Memory ↓	Latency ↓	Energy ↓
CIFAR-100	ANML	27.2	39.7 MB	309 s	1.44 kJ
	ANML+AIM	34.6	1,093 MB	6,390 s	29.43 kJ
	Oracle ANML	44.5	39.9 MB	1,866 s	8.55 kJ
	LifeLearner	45.2	15.45 MB	374 s	1.71 kJ
GSCv2	ANML+AIM	71.0	608.2 MB	394.2 s	1.80 kJ
	Oracle ANML	71.2	10.2 MB	569.7 s	2.60 kJ
	LifeLearner	71.3	3.4 MB	75.6 s	0.35 kJ
ESC-50	ANML+AIM	23.3	2,305 MB	OOM	OOM
	Oracle ANML	43.5	1,167 MB	4,120 s	20.3 kJ
	LifeLearner	38.1	316.3 MB	1,445 s	7.8 kJ

in Appendix A.2, together with pretrained, raw-update and oracle baselines. Five datasets span image and audio sensing (see Table 2 in Appendix A.3): CIFAR-100 [31] and MiniImageNet [63] for images, and GSCv2 [66], UrbanSound8K [54] and ESC-50 [47] for audio. Three architecture families are used: the lightweight CNN of prior Meta-CL work, YAMNet [19] for audio and ViT [9] for images. Following prior findings [3, 16], CNN experiments use ANML-based methods and the pretrained YAMNet and ViT backbones use OML-based methods, so cross-architecture comparisons are deployment-level observations rather than controlled ablations. On-device meta-testing runs on a Raspberry Pi Zero 2 (512 MB), a Raspberry Pi 3B+ (1 GB, about 600 MB available) and a Jetson Nano (4 GB). Algorithm descriptions, preprocessing, model configurations and task construction are in Appendices A.2–A.5.

2.3 Deployment Metrics and Measurement

Learning performance is classification accuracy after continual adaptation to the meta-test stream. To complement predictive performance, *MetaCLBench* measures three deployment-critical system metrics during on-device learning. **Peak memory** is analytically estimated as the total learning-time memory requirement, including model parameters, optimiser states, intermediate activations, and replay samples; it is deterministic, so no variance is reported. We allocate an additional 1 GB of swap space. We report OOM only when a run crashes even with swap enabled; **End-to-end latency** measures the time from loading the pretrained/meta-trained model through adaptation over the complete meta-test stream. **Energy consumption** measures the corresponding device energy over the same interval using an external USB power meter and is computed by integrating measured power over execution time. Thus, all methods are evaluated under the same end-to-end adaptation boundary rather than inference cost alone.

Meta-training is performed offline on a server, whereas all deployment measurements correspond to meta-testing on the target edge hardware. Software-stack, quantisation, power measurement, and detailed hyperparameter settings are provided in Appendix A.5.

3 Results and Findings

Accuracy and deployment viability. Table 1 reports accuracy, peak memory, latency and energy for representative methods on the Pi 3B+; results for all methods and architectures are in Table 3 (Appendix B). Ranking methods by accuracy alone is misleading once deployment constraints are considered. ANML improves over the pretrained baseline on most datasets and AIM can add further accuracy: on CIFAR-100, accuracy rises from 27.2% (ANML) to 34.6% (ANML+AIM). The same change raises peak adaptation memory from 39.7 MB to 1,093 MB, and ANML+AIM requires 1,562 MB on MiniImageNet, 2,593 MB on UrbanSound8K and 2,305 MB on ESC-50, beyond what the 1 GB device can provide; these configurations fail with out-of-memory errors. Latent-replay methods sit at the opposite end. LifeLearner reaches 45.2% on CIFAR-100 with 15.45 MB of peak memory, against 44.5% and 39.9 MB for Oracle ANML, and on ESC-50 reaches 38.1% with 316 MB where Oracle ANML needs 1,167 MB for 43.5%. Latent OML transfers the pattern to pretrained backbones: with ViT on CIFAR-100 it matches Oracle OML (36.5% vs 36.1%) at 4.8× lower latency and 5.2× lower energy, for a small memory premium from the latent replay buffer, and with YAMNet

on GSCv2 it reaches 70.4% against 70.1% for Oracle OML with much lower latency (236 s vs 1,685 s) and energy (1.17 kJ vs 8.3 kJ; Table 3). Learning-time memory is thus a first-order constraint: once optimiser state, activations, auxiliary modules and replay storage are counted, accurate methods can become unusable.

Latency and energy trade-offs. Adaptation cost separates methods further. End-to-end latency on the Pi 3B+ is dominated by backpropagation (61%), with inference at 30%, model loading 5% and data handling 4%, so methods that update more parameters or add trainable modules pay proportionally. LifeLearner adapts on CIFAR-100 in 374 s and 1.71 kJ against 1,866 s and 8.55 kJ for Oracle ANML, and on GSCv2 in 75.6 s and 0.35 kJ against 569.7 s and 2.60 kJ; across the five datasets this is a 2.54–7.43× energy reduction at near-oracle accuracy. Raw full-network updates are prohibitive: Raw ANML needs 11,424 s and 52.5 kJ on CIFAR-100, and Raw OML with ViT needs 22,968 s and 103 kJ. AIM-enhanced methods add cost through the extra activations and matrix operations of the attention modules (ANML+AIM: 6,390 s and 29.43 kJ on CIFAR-100). Methods of similar accuracy therefore occupy very different points on the accuracy–energy frontier.

Generalisation across edge devices. Repeating the GSCv2 measurements on the Pi Zero 2, Pi 3B+ and Jetson Nano gives consistent relative behaviour despite their different compute and memory capacities (Table 4, Appendix B.5). LifeLearner takes 76–84 s and 0.35–0.39 kJ across the three devices, whereas Oracle ANML takes 570–633 s, a latency gap of roughly 7.5× on each. Hardware changes absolute runtime, but the dominant overheads come from the learning algorithm itself, so memory-intensive full-update and auxiliary-module approaches remain less attractive than latent adaptation on every platform.

Pre-training, replay and architecture. Pre-training before meta-training matters most for audio: on UrbanSound8K and ESC-50, it raises the average accuracy of the CNN-based methods, and the effect is largest for the strongest methods, with Oracle accuracy on ESC-50 rising from 12.5% to 42.3% and latent replay from 15.6% to 38.1% (Figure 7). Replay frequency shows diminishing returns: one or two replay epochs of the latent methods reach the accuracy Oracle attains after five to ten, and accuracy saturates beyond five, so extra replay buys latency and energy rather than accuracy (Appendix C.2). Architectural sophistication does not consistently help. On CIFAR-100, Oracle ANML with the CNN reaches 44.5% against 36.1% for Oracle OML with ViT, while on MiniImageNet ViT is better (51.2% vs 43.8%). For audio, YAMNet often lowers peak memory but does not consistently lower latency or energy (Table 3).

4 Discussion

Learning-time cost is the bottleneck. *MetaCLBench* shows that Meta-CL deployment is a joint learning-and-systems problem rather than an accuracy-only selection problem. Compact inference models become expensive during adaptation because optimiser state, activations, replay storage and additional trainable modules inflate memory and computation, which is why AIM-enhanced and raw-update variants can be competitive in accuracy yet impractical on constrained devices. How adaptation is performed matters as much as which representation is learned: latent replay reduces what must be stored and updated and yields consistently better memory, latency and energy trade-offs, and further ablations show that replay beyond a few epochs brings diminishing accuracy at additional cost (Appendix C.2) and that audio tasks respond to pre-training and architecture differently from image tasks (Appendix C.4).

Deployment guidelines. The measurements reduce to four guidelines (Appendix C.5). Available runtime memory, not nominal RAM, decides viability: roughly 600 MB of the Pi 3B+'s 1 GB is usable, so AIM-enhanced methods should be excluded below about 1 GB available and become viable only at 2 GB or more when accuracy is paramount. Latent-replay methods, LifeLearner on CNNs and Latent OML on pretrained backbones, give the best accuracy–efficiency trade-offs on every device we tested. Raw full-network updates should be avoided outright: on CIFAR-100 they are over 30× slower than the latent methods. Pre-training before meta-training is essential for audio tasks.

Limitations and future work. The hardware range (512 MB to 4 GB, CPU only) omits micro-controllers [25, 26, 34] and dedicated accelerators [8]; extending *MetaCLBench* to both is left for future work. CNN and pretrained backbones use different Meta-CL heads (ANML versus OML), so architecture comparisons are deployment-level observations rather than controlled ablations; running both heads on every backbone would separate the two effects (Appendix C.6).

5 Conclusion

We presented *MetaCLBench*, a benchmark that evaluates Meta-CL by the cost of adapting on the device, not only by accuracy. Across six methods, five image and audio datasets, three architectures and three device tiers, learning-time memory and adaptation cost render otherwise competitive methods impractical, while latent replay, including the Latent OML variant introduced here for pretrained backbones, reaches near-oracle accuracy at a fraction of the energy.

References

[1] Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. Uncertainty-based continual learning with adaptive regularization. *Advances in neural information processing systems*, 32, 2019.

[2] Sercan Arik, Jitong Chen, Kainan Peng, Wei Ping, and Yanqi Zhou. Neural voice cloning with a few samples. *Advances in neural information processing systems*, 31, 2018.

[3] Shawn Beaulieu, Lapo Frati, Thomas Miconi, Joel Lehman, Kenneth O Stanley, Jeff Clune, and Nick Cheney. Learning to continually learn. *arXiv preprint arXiv:2002.09571*, 2020.

[4] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, P Dokania, P Torr, and M Ranzato. Continual learning with tiny episodic memories. In *Workshop on Multi-Task and Lifelong Reinforcement Learning*, 2019.

[5] Jagmohan Chauhan, Young D Kwon, Pan Hui, and Cecilia Mascolo. Contauth: Continual learning framework for behavioral-based user authentication. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(4):1–23, 2020.

[6] Jagmohan Chauhan, Young D Kwon, and Cecilia Mascolo. Exploring on-device learning using few shots for audio classification. In *2022 30th European Signal Processing Conference (EUSIPCO)*, pages 424–428. IEEE, 2022.

[7] Ali Cheraghian, Shafin Rahman, Pengfei Fang, Soumava Kumar Roy, Lars Petersson, and Mehrtash Harandi. Semantic-aware knowledge distillation for few-shot class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2534–2543, 2021.

[8] Anish Das, Young D. Kwon, Jagmohan Chauhan, and Cecilia Mascolo. Enabling On-Device Smartphone GPU based Training: Lessons Learned. *arXiv:2202.10100 [cs]*, February 2022. URL http://arxiv.org/abs/2202.10100.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[10] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Computer vision–ECCV 2020: 16th European conference, Glasgow, UK, August 23–28, 2020, proceedings, part XX 16*, pages 86–102. Springer, 2020.

[11] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

[12] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[13] Bernd Fritzke. A growing neural gas network learns topologies. *Advances in neural information processing systems*, 7, 1994.

[14] Jiangpeng He and Fengqing Zhu. Online continual learning for visual food classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2337–2346, 2021.

[15] Michael Hersche, Geethan Karunaratne, Giovanni Cherubini, Luca Benini, Abu Sebastian, and Abbas Rahimi. Constrained few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9057–9067, 2022.

[16] Nithin Holla, Pushkar Mishra, Helen Yannakoudakis, and Ekaterina Shutova. Meta-learning with sparse experience replay for lifelong language learning. *arXiv preprint arXiv:2009.04891*, 2020.

[17] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

[18] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 831–839, 2019.

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

[20] Shell Xu Hu, Da Li, Jan Stühmer, Minyoung Kim, and Timothy M Hospedales. Pushing the limits of simple pipelines for few-shot learning: External data and fine-tuning make a difference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9068–9077, 2022.

[21] Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. *Advances in neural information processing systems*, 32, 2019.

[22] Khurram Javed and Martha White. Meta-learning representations for continual learning. *Advances in neural information processing systems*, 32, 2019.

[23] Ghassen Jerfel, Erin Grant, Tom Griffiths, and Katherine A Heller. Reconciling meta-learning and continual learning with online mixtures of tasks. *Advances in neural information processing systems*, 32, 2019.

[24] Saurav Jha, Martin Schiemer, Franco Zambonelli, and Juan Ye. Continual learning in sensor-based human activity recognition: An empirical benchmark analysis. *Information Sciences*, 575:1–21, October 2021. ISSN 0020-0255. doi: 10.1016/j.ins.2021.04.062. URL https://www.sciencedirect.com/science/article/pii/S0020025521003911.

[25] Hong Jia, Young Kwon, Alessio Orsino, Ting Dang, DOMENICO TALIA, and Cecilia Mascolo. TinyTTA: Efficient Test-time Adaptation via Early-exit Ensembles on Edge Devices. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 43274–43299. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/4c454d34f3a4c8d6b4ca85a918e5d7ba-Paper-Conference.pdf.

[26] Hong Jia, Young D. Kwon, Dong Mat, Nhat Pham, Lorena Qendro, Tam Vu, and Cecilia Mascolo. UR2M: Uncertainty and Resource-Aware Event Detection on Microcontrollers. In *2024 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10, 2024. doi: 10.1109/PerCom59722.2024.10494467.

[27] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, pages 1–1, 2019. ISSN 2332-7790.

[28] Rohit Keshari, Mayank Vatsa, Richa Singh, and Afzel Noore. Learning structure and strength of cnn filters for small sample size training. In *proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9349–9358, 2018.

[29] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[30] Jedrzej Kozerawski and Matthew Turk. Clear: Cumulative learning for one-shot one-class image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3446–3455, 2018.

[31] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[32] Young D Kwon, Jagmohan Chauhan, Abhishek Kumar, Pan Hui, and Cecilia Mascolo. Exploring System Performance of Continual Learning for Mobile and Embedded Sensing Applications. In *ACM/IEEE Symposium on Edge Computing*. Association for Computing Machinery (ACM), 2021.

[33] Young D. Kwon, Jagmohan Chauhan, and Cecilia Mascolo. FastICARL: Fast Incremental Classifier and Representation Learning with Efficient Budget Allocation in Audio Sensing Applications. In *Proc. Interspeech 2021*, pages 356–360, 2021.

[34] Young D. Kwon, Jagmohan Chauhan, and Cecilia Mascolo. YONO: Modeling Multiple Heterogeneous Neural Networks on Microcontrollers. In *2022 21st ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 285–297, 2022. doi: 10.1109/IPSN54338.2022.00030.

[35] Young D. Kwon, Jagmohan Chauhan, Hong Jia, Stylianos I. Venieris, and Cecilia Mascolo. LifeLearner: Hardware-Aware Meta Continual Learning System for Embedded Computing Platforms. In *Proceedings of the 21st ACM Conference on Embedded Networked Sensor Systems*, SenSys '23, page 138–151, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704147. doi: 10.1145/3625687.3625804. URL https://doi.org/10.1145/3625687.3625804.

[36] Young D. Kwon, Rui Li, Stylianos Venieris, Jagmohan Chauhan, Nicholas Donald Lane, and Cecilia Mascolo. TinyTrain: Resource-Aware Task-Adaptive Sparse Training of DNNs at the Data-Scarce Edge. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 25812–25843. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/kwon24c.html.

[37] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[38] Eugene Lee, Cheng-Han Huang, and Chen-Yi Lee. Few-shot and continual learning with attentive independent mechanisms. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9455–9464, 2021.

[39] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017. doi: 10.1109/TPAMI.2017.2773081.

[40] Ji Lin, Ligeng Zhu, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, and Song Han. On-device training under 256kb memory. *Advances in Neural Information Processing Systems*, 35:22941–22954, 2022.

[41] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2021.10.021. URL https://www.sciencedirect.com/science/article/pii/S0925231221014995.

[42] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[43] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989. doi: https://doi.org/10.1016/S0079-7421(08)60536-8. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

[44] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[46] Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replay for real-time continual learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10203–10209. IEEE, 2020.

[47] Karol J Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1015–1018, 2015.

[48] Amal Rannen, Rahaf Aljundi, Matthew B. Blaschko, and Tinne Tuytelaars. Encoder based lifelong learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[49] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[50] Sandra Servia Rodríguez, Cecilia Mascolo, and Young D. Kwon. Knowing when we do not know: Bayesian continual learning for sensing-based analysis tasks. *CoRR*, abs/2106.05872, 2021. URL https://arxiv.org/abs/2106.05872.

[51] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[52] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[53] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

[54] Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. A dataset and taxonomy for urban sound research. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 1041–1044, 2014.

[55] Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Abhishek Kumar, Rogerio Feris, Raja Giryes, and Alex Bronstein. Delta-encoder: an effective sample synthesis method for few-shot object recognition. *Advances in neural information processing systems*, 31, 2018.

[56] Wei Shen, Ziqiang Shi, and Jun Sun. Learning from adversarial features for few-shot classification. *arXiv preprint arXiv:1903.10225*, 2019.

[57] Guangyuan Shi, Jiaxin Chen, Wenlong Zhang, Li-Ming Zhan, and Xiao-Ming Wu. Overcoming catastrophic forgetting in incremental few-shot learning by finding flat minima. *Advances in neural information processing systems*, 34:6747–6761, 2021.

[58] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

[59] Sen Song, Kenneth D Miller, and Larry F Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature neuroscience*, 3(9):919–926, 2000.

[60] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 403–412, 2019.

[61] Xiaoyu Tao, Xinyuan Chang, Xiaopeng Hong, Xing Wei, and Yihong Gong. Topology-preserving class-incremental learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIX 16*, pages 254–270. Springer, 2020.

[62] Selvarajah Thuseethan, Sutharshan Rajasegarar, and John Yearwood. Deep continual learning for emerging emotion recognition. *IEEE Transactions on Multimedia*, 24:4367–4380, 2021.

[63] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

[64] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A Comprehensive Survey of Continual Learning: Theory, Method and Application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8): 5362–5383, 2024. doi: 10.1109/TPAMI.2024.3367329.

[65] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

[66] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[67] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 374–382, 2019.

[68] Donghyun Yoo, Haoqi Fan, Vishnu Boddeti, and Kris Kitani. Efficient k-shot learning with regularized deep networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[69] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

[70] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

[71] Ram Zamir. A proof of the fisher information inequality via a data processing argument. *IEEE Transactions on Information Theory*, 44(3):1246–1250, 1998.

[72] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR, 2017.

[73] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

[74] Chi Zhang, Nan Song, Guosheng Lin, Yun Zheng, Pan Pan, and Yinghui Xu. Few-shot incremental learning with continually evolved classifiers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12455–12464, 2021.

[75] Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. Class-incremental learning via deep model consolidation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1131–1140, 2020.

[76] Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, Liang Ma, Shiliang Pu, and De-Chuan Zhan. Forward compatible few-shot class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9046–9056, 2022.

[77] Kai Zhu, Yang Cao, Wei Zhai, Jie Cheng, and Zheng-Jun Zha. Self-promoted prototype refinement for few-shot class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6801–6810, 2021.

Supplementary Material
MetaCLBench: Benchmarking the On-Device Adaptation Cost of Meta-Continual Learning

Table of Contents

A Detailed Benchmark Framework **11**
- A.1 Background on Continual and Meta-Continual Learning . . . 11
- A.2 Additional Baselines & Methods . . . 11
- A.3 Datasets . . . 12
- A.4 Model Architectures . . . 14
- A.5 Detailed Experimental Setup . . . 15

B Additional Results **16**
- B.1 RESULTS . . . 17
- B.2 Accuracy . . . 18
- B.3 Peak Memory Footprint . . . 18
- B.4 End-to-end Latency & Energy Consumption . . . 20
- B.5 Generalisability across IoT Hardware . . . 21

C Further Analysis and Discussion **21**
- C.1 With and Without pre-training on CNN . . . 21
- C.2 Number of replay epochs . . . 21
- C.3 Multi Compression Module. . . . 23
- C.4 Differences between image and audio datasets . . . 23
- C.5 Practical guideline . . . 23
- C.6 Limitations . . . 24
- C.7 Broader Impacts . . . 24

D Extended Related Work **24**
- D.1 Extended Related Work . . . 24
- D.2 Continual Learning . . . 25
- D.3 Meta Learning . . . 27
- D.4 Meta Continual Learning . . . 28

A Detailed Benchmark Framework

This section provides additional information on our MetaCLBench framework.

A.1 Background on Continual and Meta-Continual Learning

Existing approaches are grouped into three categories: (1) *rehearsal-based methods* that store and replay past samples [5, 46, 49], (2) *regularisation-based methods* that constrain weight updates to preserve important parameters [29, 72], and (3) *parameter isolation methods* that allocate separate model capacity for each task [21, 52, 69]. Rehearsal-based approaches achieve the strongest accuracy but require substantial labelled data [44] and impose significant memory and computational overheads [32], often exceeding what typical IoT devices can accommodate.

A.2 Additional Baselines & Methods

Pretrained: To initialize the model weights, this baseline performs conventional deep neural network (DNN) training without meta-learning procedure and/or Meta-CL methods during the meta-training phase. After that, it finetunes the model weights with a few samples during a meta-testing phase, similar to other Meta-CL methods. As Pretrained typically shows low performance, it serves as the *lower bound* in our evaluation. Also, a comparison between this baseline and other methods can reveal the effectiveness of the meta-learning procedure in a CL task.

Oracle: This baseline represents the case where it has access to all the classes at once in an i.i.d. fashion. Also, we assume that Oracle would perform DNN training for multiple epochs until the model converges. As a result, Oracle often shows superior accuracy, and thus its accuracy represents the *upper bound* of our evaluation.

OML [22]: This method constantly optimizes representations during online updates to quickly acquire predictions for new tasks, facilitating seamless and efficient CL. This approach uses CF as signals to justify the use of storage space to accommodate new tasks. The OML can accomplish CL in more than 200 consecutive classes. It is an online updating method that constantly learns representations and often outperforms rehearsal-based continuous learning methods. The fundamental difference between OML and MAML-Rep lies in their approach to updates. OML employs a single data point for each update, while MAML-Rep utilizes an entire batch. By adopting this method, OML is able to more accurately consider the impact of online learning, including the phenomenon of CF. The objective of OML is to implement the computation graph for live updates and to discover representations that maintain high accuracy even with ongoing changes. It acquires sparse representations without any inactive neurons, surpassing traditional and rehearsal-based strategies for ongoing learning. The technique entails meta-training, with an emphasis on avoiding interference and facilitating rapid adaption during online updates.

ANML [3]: This is an approach inspired by the neuromodulation process in the brain. This method accomplishes selective activation (forward pass) and selective plasticity (backward pass) in DNNs by meta-learning through a CL process within the prediction network. Optimising activation timing by controlling the activation of a predictive model based on input conditions enhances CL. It is often reported to be superior to OML, as it can be optimized to reduce both forward and backward interference. ANML's architecture comprises two parallel networks: the NM network governs the activations and plasticity of the PLN. ANML employs a unique method where it adjusts specific components of the PLN based on whether it is in the meta-training or meta-testing phase, distinguishing it from conventional methods. The results of the experiments demonstrate that ANML surpasses other approaches by a large margin. It maintains a high level of accuracy even when faced with 600 consecutive tasks. This is attributed to its capability to generate sparse yet powerful representations and effectively manage sequential learning without erasing previously acquired knowledge.

OML+AIM [38]: This is a Meta-CL approach that integrates Attentive Independent Mechanisms (AIM), which is a modular component of the OML framework. AIM increases the speed of learning a new task by selecting out the mechanism that best explains the representation of the inner loop to activate. AIM is comprised of a collection of autonomous mechanisms, with each mechanism being defined by its own unique set of parameters. Each mechanism functions as an autonomous specialist that cooperates with other specialists to solve assignments. AIM is situated between the feature

Table 2: Datasets and class splits. The benchmark spans visual and acoustic sensing, with 5 to 30 sequentially arriving meta-test classes.

Dataset	Modality	Meta-train / Meta-test
CIFAR-100 [31]	Image	70 / 30
MiniImageNet [63]	Image	64 / 20
GSCv2 [66]	Audio	25 / 10
UrbanSound8K [54]	Audio	5 / 5
ESC-50 [47]	Audio	30 / 20

extractor and the output classifier in a deep neural network. This approach incorporates higher-level data representation and employs rapidly updated weights within the inner loop of a meta-learning process. During the training process, AIM mechanisms engage in competition and focus on input representations through cross-attention. This allows them to select a limited number of mechanisms for task-specific learning.

ANML+AIM [38]: This is a Meta-CL approach that incorporates AIM into the ANML framework. The accuracy of SOTA Meta-CL continues to improve beyond the performance of the previous three approaches by integrating AIM into the existing CL framework. AIM is positioned following a feature extractor and preceding an output classifier, allowing it to effectively process intricate representations. AIM, in contrast to RIMs, does not employ LSTM-based temporal modeling, rendering it a static module. AIM's processes engage in a targeted manner according to the relevance of the information, which helps to create a concise representation and enhance the ability to adapt to new activities while reducing the risk of CF. It has demonstrated substantial enhancements in accuracy for tasks involving few-shot and ongoing learning, surpassing existing approaches without increasing the number of parameters.

LifeLearner [35]: This module is based on ANML and combines both lossy and lossless compression to achieve high compression rates and minimize footprint. The LifeLearner approach not only achieves optimal CL efficiency but also significantly reduces energy consumption and latency compared to the SOTA. LifeLearner consists of two distinct phases: meta-training, which takes place on a server to establish a strong initial weight configuration, and meta-testing, which occurs on devices to continuously acquire knowledge of new classes while retaining information from prior ones. The main characteristics consist of an enhanced compression module that minimizes memory consumption, a feature extractor that remains unchanged, and a classifier that is constantly updated through learning. And it achieves efficient learning even on resource-constrained edge devices.

Latent OML (ours): LifeLearner is built on ANML and a small CNN, so its latent-replay design has not been available for pretrained backbones. We port it to OML so that it can be used with YAMNet and ViT. The pretrained feature extractor is frozen after meta-training; during meta-testing, the latent activations of replay samples (taken at the output of the frozen extractor) are stored, and only the OML prediction network is updated on device. We build on OML rather than ANML for these backbones because ANML's neuromodulatory network is a second network that must be run at every update which adds additional computation. Latent OML changes the replay/update pathway rather than introducing a new meta-learning objective: it combines OML's prediction network with LifeLearner-style latent replay.

A.3 Datasets

To complement the relative simplicity, limited variety, and lack of generalisability of the datasets tested in previous studies, we added three audio datasets to the single-label image datasets (*CIFAR-100* [31], *MiniImageNet* [63]) from previous studies. Specifically, we used the *UrbanSound8K* [54] dataset with fewer classes, the *ESC-50* [47] dataset with more classes, and the *GSCv2* [66] dataset for keyword spotting (KWS) (see Table 2). We demonstrate trade-offs between performance and system resources on resource-constrained devices by extensively evaluating these datasets in different Meta-CL approaches. The details of each target dataset employed in our study are described below.

CIFAR-100 [31]. This dataset, developed by researchers supported by the Canadian Institute for Advanced Research, has 60,000 photos categorized into 100 classes, with each class including 600 images. The creation of this used the identical technique as CIFAR-10. The images were obtained

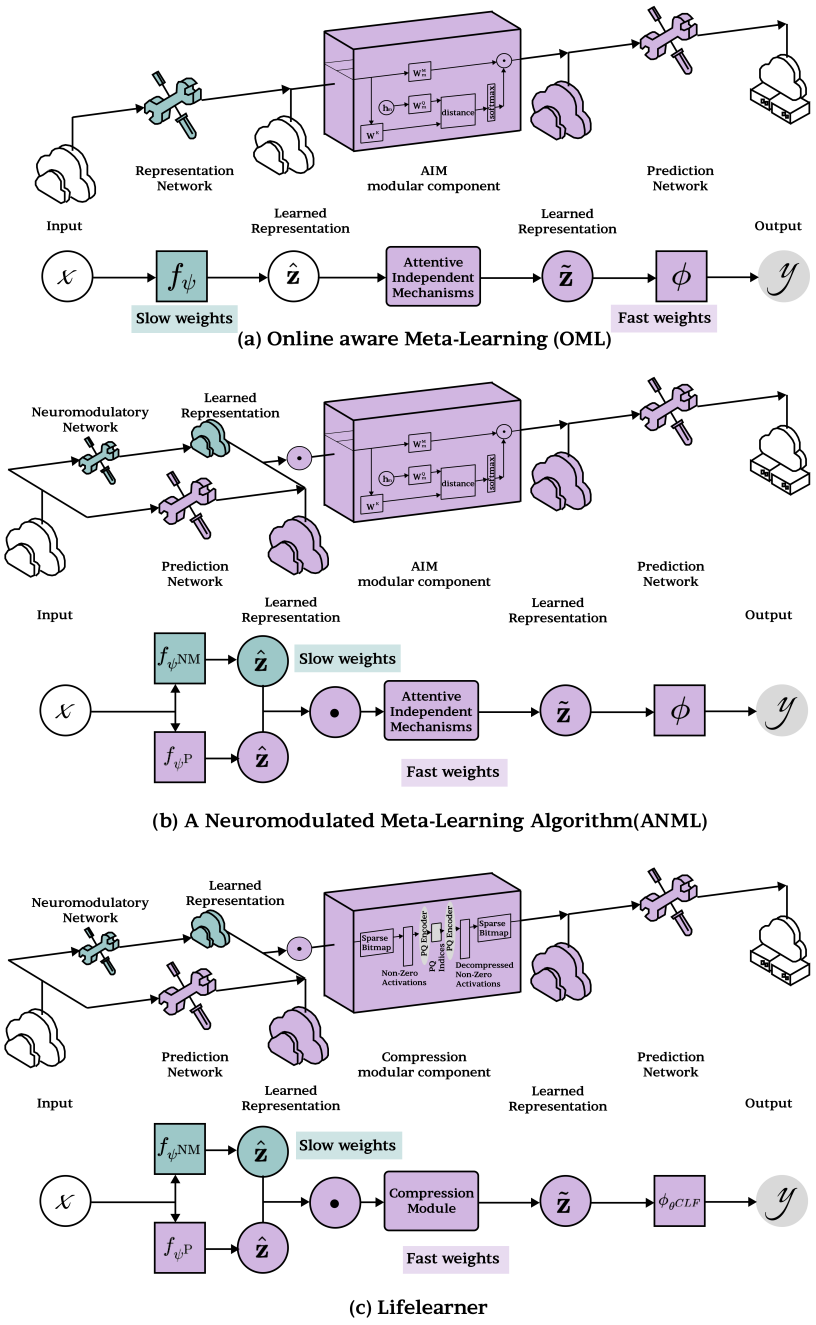


Figure 2: The illustration of the Meta-CL methods evaluated in our benchmark framework.

by utilising search phrases from WordNet, and any duplicate or irrelevant images were eliminated. The dataset is structured into 20 superclasses, with each superclass consisting of 5 associated classes. The labelers meticulously inspected and assigned labels to each image based on precise instructions, guaranteeing that each class was distinct from the classes in CIFAR-10. This ensures that CIFAR-100 may be effectively used as a negative example for CIFAR-10. The dataset is designed for the purpose of training models in object recognition, specifically focusing on small, natural photos.

MiniImageNet [63]. This dataset was formed by randomly choosing 100 classes from the complete ImageNet dataset. Each class in the Mini-ImageNet dataset consists of 600 color images, each having dimensions of 84x84 pixels. The classes were partitioned into 80 for training and 20 for testing, with the guarantee that the test classes were never exposed during the training process. This dataset is intentionally more intricate than CIFAR-10, but still feasible for contemporary machine memory. It

enables quick development and experimentation. ImageNet is an extensive visual recognition dataset comprising more than 14 million photos categorized based on the WordNet hierarchy. Every image is annotated with the corresponding thing it depicts. The dataset facilitates a range of computer vision tasks, such as object detection, classification, and segmentation. For the purpose of classification, it includes a collection of photos spanning 1,000 different categories. ImageNet's extensive size and wide range of variations have established it as a standard for training and assessing deep learning models.

UrbanSound8K [54]. The development of UrbanSound required the compilation of a wide-ranging and comprehensive dataset of urban noises. The researchers made use of the Freesound archive, which contains more than 160,000 recordings contributed by users, with a significant number of them being from metropolitan environments. They conducted a search and retrieved recordings by utilising specified class names, manually weeding out non-field recordings. The outcome of this was the acquisition of 1302 recordings, amounting to a total of 27 hours of audio. Out of these, 18.5 hours were dedicated to manually annotating sound occurrences in 10 different categories, namely air conditioner, car horn, children playing, dog bark, drilling, engine idling, pistol shot, jackhammer, siren, and street music. Every recording was marked with annotations indicating the precise start and end times of sound events, as well as their prominence, whether they were in the foreground or background. UrbanSound8K is a subset of a dataset specifically created to train sound classification algorithms. It contains 8.75 hours of audio that has been divided into 4-second segments. The taxonomy classifies sounds into four main groups: human, nature, mechanical, and music, enabling thorough and methodical examination of urban noises.

ESC-50 [47]. The primary objective of creating the ESC-50 dataset was to alleviate the limited availability of freely accessible datasets containing ambient sounds. The dataset comprises 2,000 audio clips, each tagged and lasting 5 seconds. These clips are categorized into 50 classes, which are further separated into five key categories: animal sounds, natural soundscapes and water sounds, human non-speech sounds, interior/domestic sounds, and exterior/urban noises. The selection of clips was carefully picked to prioritize the prominence of sound events while minimising the presence of background noise. The quality control process included CrowdFlower's methods and additional validation by the author, resulting in approximately twelve human categorisation entries per clip. The dataset was generated with the purpose of aiding research in the field of environmental sound categorisation, providing a reliable benchmark for the evaluation of automatic sound recognition systems. The dataset is accessible to the public and contains an additional set of 250,000 video clips without labels, which can be used for unsupervised learning experiments. The ESC-50 dataset is a valuable resource for investigating machine learning methods in the field of ambient sound classification.

GSCv2 [66]. This dataset, created by Pete Warden from Google Brain, seeks to simplify the process of training and evaluating keyword-detecting systems. The creation of the "Speech Commands" dataset entailed gathering spoken words from a varied collection of speakers through a web-based application that utilized the WebAudioAPI to record utterances. The dataset comprises 35 distinct word utterances, encompassing categories such as numbers, frequent instructions, and miscellaneous terms such as "tree" and "wow". The dataset is specifically intended for limited-vocabulary speech recognition tasks, with the objective of identifying instances where one of the target words is spoken from a selection of ten or fewer terms. In addition, the dataset contains recordings of background noise to use in training models to distinguish between speech and non-speech audio. The objective was to record audio that accurately represented the task of on-device trigger phrases, with a specific emphasis on 10 words: "Yes", "No", "Up", "Down", "Left", "Right", "On", "Off", "Stop", and "Go". Every user would activate a "Record" button, and a randomly selected word would appear on the screen for a duration of 1.5 seconds while audio was being captured. The audio snippets were saved as 16-bit single-channel PCM values at a 16 KHz rate in the form of one-second WAV files. The dataset was specifically curated to be appropriate for training and assessing keyword spotting systems, with an emphasis on offering a specialized dataset that distinguishes itself from conventional datasets utilized for complete sentence automatic speech recognition.

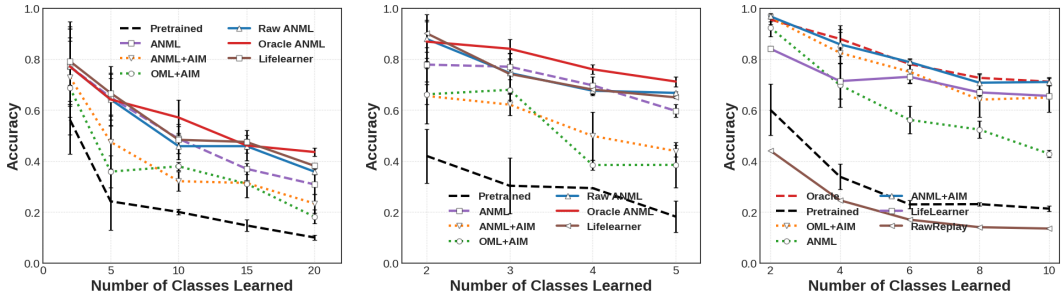
A.4 Model Architectures

Referring to previous work [40], we add optimized DNN architectures and transformer architectures intended for resource-limited IoT devices to the evaluated three-layer CNN architectures, including

YAMNet [19], and **ViT** [9]. YAMNet employs the MobileNetV1, which is a highly efficient model specifically designed for applications involving mobile and embedded vision. Depthwise separable convolutions are utilized in their construction, thereby reducing computing complexity by decomposing the normal convolution operation into separate depthwise and pointwise convolutions. This efficient structure enables the development of compact deep neural networks. MobileNets additionally incorporate two hyperparameters, namely width multiplier and resolution multiplier, that allow for adjusting the balance between model size, latency, and accuracy. Developers can tailor MobileNets to match the precise resource limitations of their application by fine-tuning these hyperparameters. MobileNets have demonstrated equivalent performance to larger, more intricate models, but with a substantially smaller size and faster computational speed. They have been effectively utilized in a wide range of activities including item identification, precise categorisation, facial features, and other applications. YAMNet provides a flexible and effective approach for implementing deep learning models on mobile and embedded devices. Moreover, MobileNets may be customized to meet unique application needs by modifying the hyperparameters, rendering them adaptable and effective solutions for on-device artificial intelligence. The Vision Transformer (ViT) is an architecture based on transformers, which has demonstrated encouraging outcomes in activities related to image identification. ViT, unlike typical Convolutional Neural Networks (CNNs), operates on inputs in a 2-dimensional format instead of a 1-dimensional series of patches. The model includes Transformer blocks, consisting of row-self-attention followed by a multi-layer perceptron (MLP), and then column-self-attention followed by an MLP, instead of the conventional self-attention followed by an MLP. ViT has undergone pre-training using extensive datasets such as ImageNet-21k and JFT-300M. It has demonstrated exceptional performance on several image recognition benchmarks, including ImageNet, CIFAR-100, and the VTAB classification suite [73], surpassing all previous records. ViT models exhibit superior memory efficiency and outperform CNNs with reduced computational resources. They can be optimized for specific tasks such as detection and segmentation, and demonstrate potential for scenarios with limited data transport. ViT is a transformer-based model that has shown impressive results in image identification tasks, particularly when it is pre-trained on extensive datasets and then fine-tuned for specific applications.

A.5 Detailed Experimental Setup

Training Detail: We adopt a pre-training procedure, similar to prior works [20, 70]. For the advanced model architectures such as ViT and YAMNet, there exist pre-trained model weights, thus we utilize them directly similar to [20]. Then, for the CNN architecture, we pre-train the CNN model with a sufficient number of epochs (i.e., until the validation loss converges), which is consistent with conventional transfer learning for DNNs [70]. Consistent with prior meta-training research [35, 38], we employed a batch size of 1 for the inner-loop updates and 64 for the outer-loop updates across 20,000 steps to ensure the accuracy of our results. To obtain a meta-trained model with the highest possible validation set accuracy, we tested with various learning rates required for both the inner and outer loops. Consequently, we established the learning rate for the inner loop (α) was set to 0.001, and the learning rate for the outer loop (β) was also set to 0.001 for all datasets. In the meta-testing phase, we assessed ten distinct learning rates for each method and reported the best accuracy. To evaluate the precision of rehearsal-based methods, we experimented with batch sizes of 8 and 16 and observed a minimal performance difference depending on the batch sizes. We selected a batch size of 8 as it requires less memory than larger batch sizes. We chose to use ANML with CNN architectures and OML with more advanced models like ViT and YAMNet. Our preliminary tests revealed minimal differences between ANML and OML within the same architecture, with ANML slightly outperforming OML in terms of accuracy. Due to this marginal difference, we focused on ANML for CNNs to streamline the analysis. For more complex architectures like ViT and YAMNet, we opted for OML, as ANML introduces additional computational overhead due to an extra layer. Our study concentrated on the performance of methods within specific architectures rather than comparing ANML and OML across all architectures. This streamlined approach allowed us to present a clearer analysis without redundant testing across combinations. Consistent with prior meta-training research, we employed a batch size of 1 for the inner-loop updates and 64 for the outer-loop updates across 20,000 steps to ensure the accuracy of our results. To obtain a meta-trained DNN with the highest possible validation set accuracy, we tested with various learning rates required for both the inner and outer loops. Consequently, we established the learning rate for the inner loop (α) was set to 0.001, and the learning rate for the outer loop (β) was also set to 0.001 for all datasets. During the ESC-50 test, we conducted five training sessions, using 1600 clips as the training set and 400 clips as



(a) ESC-50 (b) Urbansound8K (c) GSCv2

Figure 3: The accuracy of ESC-50, Urbansound8k, GSCv2 datasets using the **3-Layer CNN** architecture.

the test set. In the meta-testing phase, we assessed ten distinct learning rates for each technique and reported the most successful outcomes. Additionally, to evaluate the precision of replay systems, we experimented with batch sizes of 8 and 16. We noticed minimal fluctuations in the performance of CL. We selected a batch size of 8 because it requires less memory than larger batch sizes.

Evaluation Protocol: With reference to previous research [3], this study prioritizes the accuracy of unseen samples across new categories in CL. The accuracy rate signifies the capacity of CL to make generalisations. In addition, we analytically calculate the memory footprint required to perform CL (model parameters, optimizers, and activations memory). To obtain the peak memory footprint during the CL process, we include (1) model memory for the model itself and the weights to be updated, (2) optimizer memory for gradients, and (3) activations memory for intermediate outputs for weights update. Furthermore, we measure the end-to-end latency and energy consumption of various approaches to CL on the edge device over specified categories. To measure the end-to-end latency and energy, we include the time and energy used to: (1) load a model and(2) perform CL using all the given samples (e.g., 30) over all the given classes (e.g., 10). We test the latency and total energy expenditure for end-to-end CL by deploying MetaCLBench and baselines on an edge device, Raspberry Pi 3B+. Regarding energy, we measure the total amount of energy consumed by a device during the end-to-end CL process. This is performed by measuring the power consumption on Raspberry Pi 3B+ using a YOTINO USB power meter and deriving the energy consumption following the equation: Energy = Power × Time.

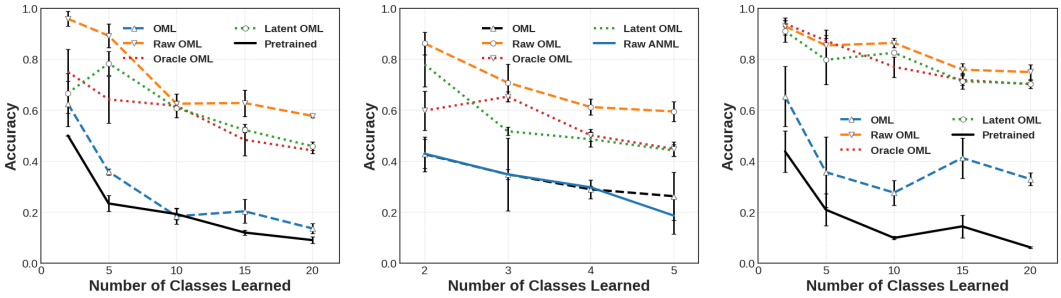
System Implementation: The initial phase of Meta-CL is the meta-training stage, which facilitates swift deployment of the model by initialising the neural network with pre-trained weights. Subsequently, the meta-testing phase incorporates optimisations compatible with hardware, enhancing system efficiency when implemented on embedded devices such as Raspberry Pi 3B+. Source code of MetaCLBench is publicly available[2].

Embedded Device: The Raspberry Pi 3B+ is equipped with a quad-core ARM Cortex-A53 processor and has 1 GB of RAM. It should be noted that the available memory on the Raspberry Pi 3B+ while it is not in use is approximately 600 MB. This is because some of the memory is already being used by background processes, concurrent apps, and the operating system. Thus, we allocate an additional 1 GB of swap space. Moreover, we utilize Faiss (PQ Framework) [27] and PyTorch 1.13 (Deep Learning Framework) [45] as software platforms to create and assess the meta-training and meta-testing stages on embedded devices.

B Additional Results

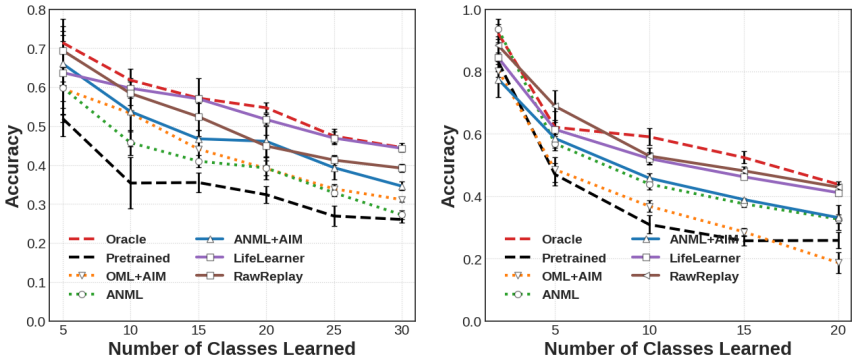
In this section, we present additional results that are not included in the main content of the paper due to the page limit.

[2] https://github.com/theyoungkwon/MetaCLBench



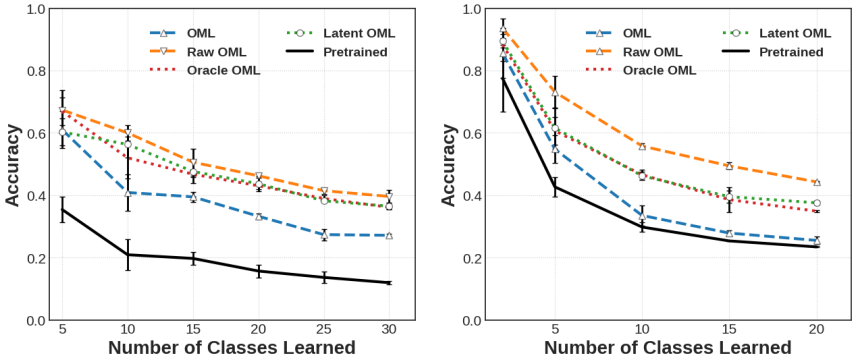
(a) ESC-50 (b) Urbansound8K (c) GSCv2

Figure 4: The accuracy of ESC-50, Urbansound8k, GSCv2 datasets using the **YAMNet** architecture.



(a) CIFAR100 (b) MiniImgenet

Figure 5: The accuracy of CIFAR100, MiniImgenet datasets using the **3-Layer CNN** architecture.



(a) CIFAR100 (b) MiniImgenet

Figure 6: The accuracy of CIFAR100, MiniImgenet datasets using the **ViT** architecture.

B.1 RESULTS

We now present deployment-oriented findings, focusing on both accuracy and deployment-critical metrics that determine whether methods are viable for IoT deployment. As previous research [3, 16] has shown that ANML often outperforms OML when using 3-layer CNN architectures, we evaluate ANML on CNN architectures. For advanced models like YAMNet and ViT, we prioritized OML evaluation due to their inherent pre-training components, which eliminate the need for additional pre-

training through ANML or the AIM module—additions that neither significantly improve accuracy nor justify their computational cost.

B.2 Accuracy

Tables 3 present accuracy across all method-dataset combinations. Oracle models (Oracle ANML for CNN, Oracle OML for YAMNet and ViT), trained on all classes simultaneously, establish upper-bound accuracy representing ideal (but unrealistic) deployment scenarios. The key question for IoT practitioners is not which method achieves the highest accuracy, but which methods achieve acceptable accuracy while remaining deployable on target hardware.

By comparing Meta-CL methods against Oracle models, we demonstrate the trade-offs between Meta-CL approaches and ideal scenarios where all data is available initially, highlighting the methods' efficiency under practical constraints. Also, "Pre-trained" approach (transfer learning applied on pre-trained weights without meta-training) shows lower accuracy, averaging 22.18% and 13.8% across the datasets for Convolutional Neural Networks (CNNs) and advanced architectures (YAMNet and ViT). This result indicates that traditional transfer learning fails to address few-shot learning challenges. In CNN architecture, baseline ANML improved upon the "pre-trained" approach with a 16.7% average accuracy gain. The AIM enhancement further increased accuracy by 2.54%. However, these improvements remained below Oracle and LifeLearner by 15.92%.

To visualize the change in accuracy as the training set grows and how each Meta-CL method compares, we provide the results of accuracy on the three architectures (3-layer CNN, YAMNet, and ViT) and six Meta-CL methods based on the five datasets as shown in Figures 3, 4, 5, and 6. The results reveal that models relying solely on pre-trained weights ("Pretrained") exhibit lower accuracy, averaging 22.18% and 13.8% across the datasets for CNNs and more advanced architectures, such as YAMNet and ViT. This result indicates that traditional transfer learning fails to address the complexities of few-shot learning scenarios. Within CNNs, baseline methods like ANML demonstrated improvement over the pretrained-only approach (with an average accuracy gain of 16.7%). This improvement was further augmented by incorporating the AIM, resulting in an additional 2.54% average accuracy gain. However, these gains still fell short of those achieved by the Oracle and LifeLearner methods (by a deficit of 15.92% in accuracy). The constrained sample size in the meta-testing phase poses a challenge to attaining high accuracy levels, even considering the theoretical upper limit of accuracy. Our research also indicated that both CNNs and advanced models experience a marked decline in performance when meta-training proceeds without prior pre-training, noting an average 13.98% improvement after pre-training. Given that advanced models intrinsically include pre-trained weights, further pre-training does not substantially increase their accuracy. Thus, pre-training is affirmed as a critical step for enhancing model accuracy. This study underscores that regardless of the method employed, the procedural sequence of pre-training followed by meta-training before evaluation is indispensable for attaining heightened accuracy levels, recognising that advanced models inherently incorporate a pre-training phase. These observations indicate a non-trivial trade-off between accuracy, memory, and computation for all the employed architectures in our work.

B.3 Peak Memory Footprint

For IoT deployment, peak memory during learning—not inference—determines viability. Hence, we conducted measurements of the maximum memory requirements for backpropagation and rehearsal samples. The backpropagation memory comprises: (1) model parameters; (2) optimizer states (gradients and momentum vectors); (3) activation (intermediate tensors used during backpropagation).

First, the results in Tables 3 reveal a critical deployment barrier: AIM-enhanced methods (ANML+AIM, OML+AIM) requires 608-2,648 MB (except for a single case), exceeding available RAM on Pi 3B+ and Pi Zero 2, due to additional parameters and activations required during training AIMs. These methods, despite strong accuracy, are *non-deployable* on typical IoT hardware.

Second, while OML, ANML, Oracle, Pre-trained, and LifeLearner demonstrate lower memory requirements, Pre-trained and ANML are less accurate. LifeLearner emerges as the optimal solution, requiring only 3.4-496 MB while maintaining accuracy comparable to Oracle. This efficiency is achieved through its compression module for latent replays, reducing activation memory and overall memory footprint, making it particularly suitable for resource-constrained devices.

Table 3: Performance and computational costs, memory footprint of six representative Meta-CL methods using three network architectures on five datasets of image and audio domains. OOM indicates an out-of-memory issue.

		CNN				ViT			
Dataset	**Method**	**Accuracy**	**Memory**	**Latency**	**Energy**	**Accuracy**	**Memory**	**Latency**	**Energy**
CIFAR100	Pretrained	**0.260**	39.69MB	**305s**	**1.44KJ**	**0.119**	**29.8MB**	**615s**	**2.96KJ**
	ANML	0.272	39.69MB	309s	1.44KJ	-	-	-	-
	ANML+AIM	0.346	**1,093MB**	6,390s	29.43KJ	-	-	-	-
	OML	-	-	-	-	0.271	29.8MB	628s	3.03KJ
	OML+AIM	0.311	834.1MB	1,481s	6.84KJ	-	-	-	-
	Raw ANML	0.392	99.5MB	**11,424s**	**52.5KJ**	-	-	-	-
	Raw OML	-	-	-	-	**0.40**	**153MB**	**22,968s**	**103KJ**
	Oracle ANML	0.445	39.93MB	1,866s	8.55KJ	-	-	-	-
	Oracle OML	-	-	-	-	0.361	29.5MB	3,672s	19.4KJ
	LifeLearner	**0.452**	**15.45MB**	374s	1.71KJ	-	-	-	-
	Latent OML	-	-	-	-	0.365	36.8MB	764s	3.76KJ
Mini ImageNet	Pretrained	0.258	474.5MB	1,198s	5.5KJ	**0.234**	336.6 MB	**2,056s**	**11.2KJ**
	ANML	0.327	474.5MB	**1,152s**	**5.3KJ**	-	-	-	-
	ANML+AIM	0.331	**1,562 MB**	**OOM**	**OOM**	-	-	-	-
	OML	-	-	-	-	0.255	336.6 MB	2,203s	11.4KJ
	OML+AIM	**0.187**	1,051 MB	1,434s	6.5KJ	-	-	-	-
	Raw ANML	0.429	897.1 MB	185,610s	810KJ	-	-	-	-
	Raw OML	-	-	-	-	0.454	**1,334 MB**	**301,456s**	**1150KJ**
	Oracle ANML	**0.438**	475.0 MB	3,414s	15.7KJ	-	-	-	-
	Oracle OML	-	-	-	-	**0.512**	337.7 MB	6,301s	31KJ
	LifeLearner	0.433	**137 MB**	1,343s	6.17kJ	-	-	-	-
	Latent OML	-	-	-	-	0.376	**336.6 MB**	2,250s	13KJ
		CNN				YAMNet			
GSCv2	Pretrained	0.213	10.16MB	**71.1s**	**0.324KJ**	**0.061**	**39.2MB**	213.6s	1.08KJ
	ANML	0.429	10.16MB	71.1s	0.324KJ	-	-	-	-
	ANML+AIM	0.710	**608.2MB**	394.2s	1.8KJ	-	-	-	-
	OML	-	-	-	-	0.329	39.2MB	**209.8s**	1.08KJ
	OML+AIM	0.649	135.2MB	157.5s	0.72KJ	-	-	-	-
	Raw ANML	**0.135**	45.72MB	**735.6s**	**3.37KJ**	-	-	-	-
	Raw OML	-	-	-	-	**0.749**	**120MB**	1,482s	6.85KJ
	Oracle ANML	0.712	10.20MB	569.7s	2.6KJ	-	-	-	-
	Oracle OML	-	-	-	-	0.701	39.7MB	**1,685s**	**8.3KJ**
	LifeLearner	**0.713**	**3.40MB**	75.6s	0.35KJ	-	-	-	-
	Latent OML	-	-	-	-	0.704	40.9MB	235.8s	**1.17KJ**
Urban Sound8K	Pretrained	**0.182**	1,382 MB	**302s**	**1.6KJ**	**0.186**	39.2 MB	912s	5.4KJ
	ANML	0.596	1,382 MB	305s	1.6KJ	-	-	-	-
	ANML+AIM	0.439	2,593 MB	**OOM**	**OOM**	-	-	-	-
	OML	-	-	-	-	0.262	39.2 MB	**910s**	5.36KJ
	OML+AIM	0.385	**2,648 MB**	OOM	OOM	-	-	-	-
	Raw ANML	0.667	1,456 MB	60,120s	279KJ	-	-	-	-
	Raw OML	-	-	-	-	0.595	**140 MB**	**120,965s**	**550KJ**
	Oracle ANML	**0.710**	1,384 MB	916s	4.5KJ	-	-	-	-
	Oracle OML	-	-	-	-	**0.729**	42.4 MB	2,501s	13.6KJ
	LifeLearner	0.650	**496 MB**	320s	1.7KJ	-	-	-	-
	Latent OML	-	-	-	-	0.442	**39.2 MB**	910s	**5.4KJ**
ESC-50	Pretrained	0.196	1,163MB	**1,359s**	**7.2KJ**	**0.090**	**39.8MB**	4,052s	19.6KJ
	ANML	0.308	1,163MB	1,374s	7.2KJ	-	-	-	-
	ANML+AIM	0.233	2,305MB	OOM	OOM	-	-	-	-
	OML	-	-	-	-	0.135	39.8MB	**3,986s**	19.6KJ
	OML+AIM	**0.181**	2,152MB	OOM	OOM	-	-	-	-
	Raw ANML	0.358	**5,005MB**	**OOM**	**OOM**	-	-	-	-
	Raw OML	-	-	-	-	**0.577**	**210.8MB**	**28,465s**	138.8KJ
	Oracle ANML	**0.435**	1,167MB	4,120s	20.3KJ	-	-	-	-
	Oracle OML	-	-	-	-	0.442	42.5MB	12,473s	**60.8KJ**
	LifeLearner	0.381	**316.3MB**	1,445s	7.8KJ	-	-	-	-
	Latent OML	-	-	-	-	0.458	44.3MB	4,536s	**22.1KJ**

Measurements were conducted to ascertain the maximum amount of RAM required for executing backpropagation and storing rehearsal samples. The memory for performing backpropagation is divided into three parts: (1) model memory, which is responsible for storing model parameters; (2) optimizer memory, which is responsible for storing gradient and momentum vectors; (3) activation memory, which is responsible for storing intermediate activations that are reused during backpropagation. Table 3 shows the various methods and the peak memory usage of our system. First, we found that although AIM improves accuracy, it takes up much memory space because there are many parameters in the AIM module. The memory required is as large as 1,562 MB for MiniImageNet and 2,648 MB for UrbanSound8K, which far exceeds the RAM capacity of embedded devices like the Pi 3B+ and causes an out-of-memory (OOM) problem. Consequently, it becomes challenging to utilize it in edge devices. OML, ANML, Oracle, Pretrained, and LifeLearner have a more minor memory requirement. However, Pretrained and ANML are less accurate. In contrast, LifeLearner is

the most satisfactory, requiring only 3.4-496 MB of very high compression rates with high accuracy comparable to Oracle.

Deployment Viability. Out-of-memory (OOM) failures occurred for ANML+AIM and OML+AIM on all datasets except GSCv2 and CIFAR-100 when deployed on Pi 3B+. This finding has immediate practical implications: IoT practitioners targeting sub-1 GB devices should exclude AIM-enhanced methods from consideration regardless of their reported accuracy.

B.4 End-to-end Latency & Energy Consumption

Table 3 also presents the operational efficiency of various Meta-CL methods on edge devices, with specific emphasis on end-to-end latency and energy consumption. The end-to-end latency comprises four components: model loading (5%), inference (30%), backpropagation (61%), and data processing overhead (4%).

First, we measured the end-to-end latency of CL methods on the Raspberry Pi 3B+ across all classes (30 samples per class). LifeLearner achieved fast end-to-end latency. Oracle exhibited higher training latency due to i.i.d. training over multiple epochs until it converges. AIM's implementation of attention mechanism selection during backpropagation increased computational overhead. LifeLearner reduces training and memory latency through latent replay and model freezing. Due to the large amount of memory required by the AIM series, crashes occurred several times during the test due to memory exhaustion. In particular, Raw ANML requires substantial latency and energy consumption (e.g., 11,424s and 52.5 kJ on CIFAR-100) as it requires full backpropagation of the entire networks.

Furthermore, the range of energy consumption is substantial: LifeLearner consumes 0.35 kJ on GSCv2 while Oracle ANML requires 2.6 kJ—a 7.4$\times$ difference. Meta-CL methods with the AIM module (ANML+AIM, OML+AIM) consume significantly more energy (e.g., 29.43 kJ for CIFAR-100) compared to more efficient methods like LifeLearner (1.71 kJ) consuming 17.2$\times$ less energy. This increased consumption directly correlates with the computational complexity introduced by the attention mechanisms in AIM, which require additional matrix operations during both forward and backward passes.

For severely resource-constrained devices (like Raspberry Pi Zero 2), LifeLearner offers the near-optimal balance between accuracy and energy efficiency (achieving near oracle accuracy at 2.54-7.43$\times$ lower energy consumption), making it ideal for battery-powered applications with limited computational resources. For devices with moderate resources (Raspberry Pi 3B+), baseline ANML represents a reasonable compromise, providing improved accuracy with only marginal energy increase over pre-trained methods. For devices with greater computational capacity (Jetson Nano), AIM-enhanced methods become viable options when application requirements prioritize accuracy over energy consumption.

The end-to-end latency is calculated by taking into account three different times: (1) The duration required for loading the pre-trained model, (2) The duration needed for sequentially training the model for each specified class, and (3) The duration needed for employing further optimisation proposed in specific Meta-CL methods such as compression method of LifeLearner or training additional module such as AIM. First, we measured the end-to-end latency of CL for all systems on the Raspberry Pi CPU for all classes for each dataset (30 samples per class). LifeLearner achieved fast end-to-end latency. Due to the large amount of memory required by the AIM series, crashes occurred several times during the test due to memory exhaustion. While Raw OML may exhibit the highest accuracy among advanced architectures, our tests have demonstrated that Raw methods also require the most computational resources in terms of latency and energy consumption. Additionally, our findings indicate that the YAMNet architecture significantly decreases capacity requirements due to its smaller size; however, it exhibits greater delays and higher energy consumption as a result of its more intricate architecture compared to CNN. As a result, we must consider factors beyond a single result alone when selecting Meta-CL methods for edge devices.

To establish generalisability, we extended our evaluation to diverse edge devices with varying computational capabilities (Jetson Nano with enhanced specifications and Pi Zero 2 with reduced capacity). Table 4 below shows the end-to-end latency and energy consumption results across these three edge devices, confirming that LifeLearner achieves a 7.5-fold reduction in latency compared to Oracle.

Table 4: End-to-end latency and energy consumption results of running different Meta-CL methods on three different edge devices

Device	Method	Latency	Energy
Raspberry Pi 3B+	Pre-trained	71s	0.32 kJ
	ANML	71s	0.32 kJ
	Oracle ANML	570s	2.60 kJ
	LifeLearner	76s	0.35 kJ
Jetson Nano	Pre-trained	79s	0.36 kJ
	ANML	79s	0.36 kJ
	Oracle ANML	633s	2.90 kJ
	LifeLearner	84s	0.39 kJ
Raspberry Pi Zero 2	Pre-trained	74s	0.33 kJ
	ANML	75s	0.33 kJ
	Oracle ANML	601s	2.64 kJ
	LifeLearner	80s	0.35 kJ

B.5 Generalisability across IoT Hardware

We extended our evaluation to diverse edge devices with varying computational capabilities (Jetson Nano with enhanced specifications and Pi Zero 2 with reduced capacity). The devices we selected span from entry-level edge AI platform (Jetson Nano: 4GB RAM, quad-core processor) to general IoT device (Pi 3B+: 1GB RAM) to highly constrained (Pi Zero 2: 0.5GB RAM) configurations, covering key points in the resource continuum of IoT platforms. By testing across this hardware spectrum, we observe how different Meta-CL methods perform under varying resource constraints, revealing which approaches are most suitable for specific hardware profiles. Our findings demonstrate that the performance patterns and resource requirements scale predictably with hardware capabilities, allowing practitioners to extrapolate to similar devices. For example, devices with RAM less than 512 MB would likely benefit from the same optimisation strategies identified for Pi Zero 2, while those with about 1 GB RAM might follow patterns observed on Pi 3B+. The table 4 below shows the end-to-end latency and energy consumption results across these three edge devices, confirming that LifeLearner achieves a 7.5-fold reduction in latency compared to Oracle.

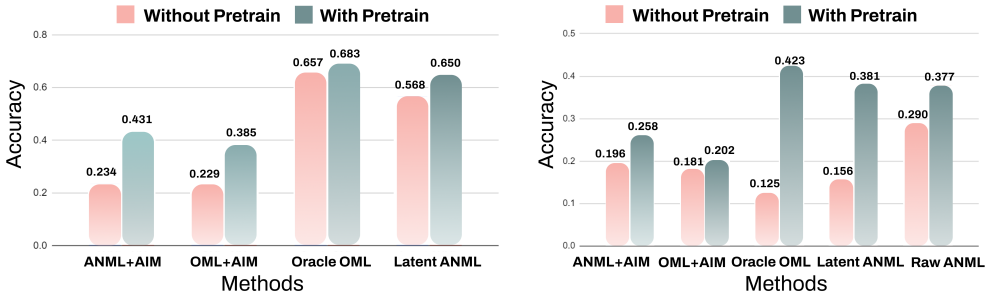
C Further Analysis and Discussion

C.1 With and Without pre-training on CNN

In this section, we examine the impact of hyperparameter variations on model accuracy. First, we investigate the influence of pre-training on the accuracy metric. Initially, we evaluated the five datasets without pre-training; the two image datasets and GSCv2 exhibited satisfactory accuracy. However, the performance on Urbansound8k and ESC-50 was suboptimal. Consequently, we adopted the strategy of pre-training as employed in advanced models, incorporating pre-training before meta-training for each method. Figure 7 illustrates that including pre-training significantly enhances model efficacy. Specifically, in the Urbansound8k and ESC-50 datasets, all Meta-CL methods demonstrated an average efficiency increase of 13.98% following pre-training. Consequently, pre-training plays a significant role in enhancing accuracy, particularly for fundamental algorithms such as ANML and OML, where accuracy without pre-training is a mere 5%. Advanced architectures intrinsically incorporate pre-training, obviating the need for additional pre-processing. Nonetheless, these sophisticated architectures, when incorporating pre-training, exhibit increased latency and energy consumption.

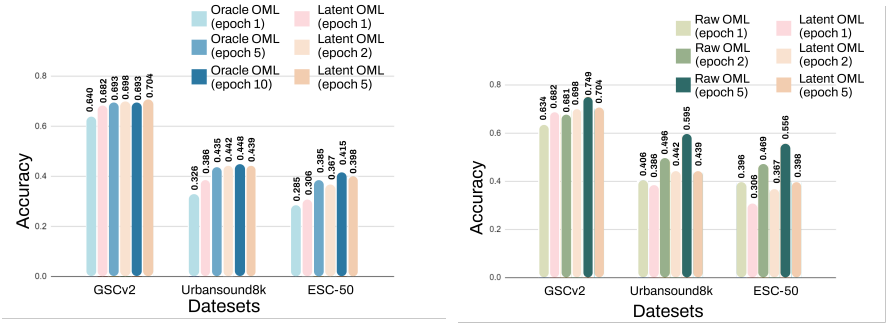
C.2 Number of replay epochs

Recognising that an increased number of replays correlates with heightened latency and energy usage, we explored the association between replay frequency and accuracy. Figure 8a indicates that Oracle OML necessitates more than five replays to attain elevated accuracy levels, thereby escalating energy expenditure throughout the training phase. Conversely, Latent OML demonstrates superior

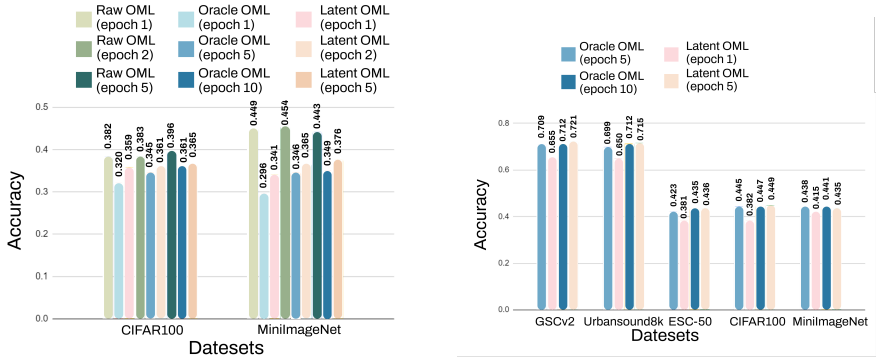


(a) Urbansound8k (b) ESC-50

Figure 7: Comparison of accuracy with and without pretraining for datasets UrbanSound8K and ESC-50 using the 3-Layer CNN architecture.



(a) Oracle& Latent under YAMNet architecture (b) Raw& Latent under YAMNet architecture



(c) ViT architecture (d) CNN architecture

Figure 8: Comparison of different Meta-CL methods regarding the number of replay epochs.

performance, requiring merely one or two replays to reach comparable accuracy, thereby diminishing system overhead. In the case of the same dataset, the efficiency of Latent OML after five replays approximates that of Oracle OML following ten replays. The same observation is evident in the CNN architecture depicted in Figure 8d; here, one or two iterations of the Latent method attain the same level of accuracy as the Oracle method after five iterations. Additionally, we note that exceeding five iterations does not significantly enhance accuracy. Figures 8band 8cillustrate that the accuracy of Raw OML within the advanced methods surpasses that of the previously tested Latent approach.

C.3 Multi Compression Module.

In CNN architectures, ANML with AIM demonstrates substantial accuracy improvements (26.0% to 34.6% on CIFAR100), albeit with increased memory requirements (39.69MB to 1,093MB). LifeLearner presents an optimal balance between accuracy and memory utilisation in resource-constrained environments. The integration of AIM, while beneficial for accuracy, introduces significant computational overhead, particularly in Raw ANML implementations (11,424s latency, 52.5KJ energy consumption on CIFAR100). Oracle ANML offers a compromise between performance enhancement and resource efficiency.

Vision Transformer (ViT) and YAMNet architectures exhibit analogous performance-resource tradeoffs. ViT implementations with OML and AIM show improved accuracy but incur substantial computational costs (22,968s latency, 103KJ energy consumption on CIFAR100). Similarly, YAMNet with AIM achieves superior accuracy (42.9% to 71.0% on GSCv2) at the expense of increased memory utilisation (10.16MB to 608.2MB). Pretrained YAMNet models demonstrate superior resource efficiency metrics.

C.4 Differences between image and audio datasets

IoT deployments span both image sensors (cameras, visual inspection systems) and audio sensors (microphones for voice interfaces, acoustic monitoring). Understanding modality-specific Meta-CL behavior is essential for system design.

First, the temporal nature of audio data creates distinct feature dependencies compared to image data. Our experimental results demonstrate that ANML achieves 59.6% accuracy on UrbanSound8K versus only 32.7% on MiniImageNet, a disparity stemming from audio's sequential characteristics being better suited for neural networks to extract temporal patterns.

Also, audio data requires specialized preprocessing steps (e.g., spectrogram conversion or MFCC feature extraction) that introduce extra complexity in Meta-CL systems. Architectures optimized for audio like YAMNet exhibit different memory-energy tradeoffs compared to visual architectures like ViT, with the former showing better energy efficiency on audio samples (5.4 kJ for UrbanSound8K vs. 11.2 kJ for MiniImageNet).

Finally, we find that audio Meta-CL is more sensitive during the pre-training phase, as in Figure7, where pre-training improved average performance by 13.98% for UrbanSound8K and ESC-50 datasets, while the improvement for image datasets was substantially smaller. This indicates that knowledge transfer in the audio domain requires stronger domain-specific foundations to work effectively in Meta-CL settings.

C.5 Practical guideline

Our extensive empirical analysis provides evidence-based guidelines for researchers and practitioners implementing Meta-CL on edge devices. While our investigation focuses on specific Meta-CL methods and may not encompass all learning approaches, our results and findings show consistent patterns across different architectures and datasets, which lead to the following key guidelines.

Resource-Aware Method Selection. Device memory during runtime (not nominal RAM) is the primary constraint determining method viability. For devices with less than 1 GB available RAM, AIM-enhanced methods should be excluded due to OOM failures observed in our experiments. LifeLearner and Latent OML provide the best balance of accuracy and resource efficiency across memory-constrained environments. For devices with sufficient memory ($\geq$2 GB available), AIM-enhanced methods become viable when accuracy is paramount.

Architecture Selection. Direct comparison across architectures is limited because CNN uses ANML-based methods while ViT and YAMNet use OML-based methods. Nevertheless, we observe that advanced architectures (ViT and YAMNet) do not necessarily guarantee better Meta-CL performance over CNN. For image tasks, results are dataset-dependent: CNN with Oracle ANML outperforms ViT with Oracle OML on CIFAR-100 (44.5% vs. 36.1%) but underperforms on MiniImageNet (43.8% vs. 51.2%). For audio tasks, YAMNet tends to achieve higher accuracy than CNN while requiring less memory. These findings challenge conventional assumptions about model complexity.

Pre-training Strategy. Pre-training before meta-training is essential, particularly for audio domains where we observed 14% average accuracy improvement (Figure 7a). As detailed in Section A.5, for CNNs, we pre-train on source classes until the validation loss converges. For YAMNet and ViT, we leverage publicly available pre-trained weights directly.

Edge Deployment Considerations. Latent methods such as LifeLearner for CNN and Latent OML for ViT/YAMNet consistently demonstrate strong performance across image and audio datasets, balancing accuracy with resource efficiency. These methods are particularly advantageous for battery-powered IoT applications, enabling real-time category integration, efficient personalised processing, and privacy-preserving local computation without additional communication overhead.

AIM-enhanced methods should be avoided in energy-constrained scenarios due to their substantial memory and high energy usage. While Raw ANML and Raw OML can sometimes achieve higher accuracy than Oracle methods, their extreme latency makes them impractical for edge deployment. For example, on CIFAR-100, Raw ANML and Raw OML require 11,424s and 22,968s, respectively, compared to 374s and 764s for LifeLearner and Latent OML – over 30$\times$ slower.

C.6 Limitations

MetaCLBench evaluates a representative but non-exhaustive set of Meta-CL methods, datasets, architectures, and edge devices. Our hardware coverage ranges from 512 MB to 4 GB RAM and does not include microcontroller-class devices or dedicated neural accelerators. In addition, architecture comparisons are not fully controlled because CNN experiments use ANML-based methods while ViT/YAMNet experiments use OML-based methods. The reported deployment costs should therefore be interpreted for the evaluated hardware/software configurations rather than as universal hardware-independent values.

C.7 Broader Impacts

MetaCLBench can support the development of more resource-efficient on-device learning systems by exposing memory, latency, and energy costs that are often omitted from algorithmic evaluations. More efficient local adaptation may reduce dependence on cloud computation and communication, potentially improving energy efficiency and keeping sensing data on-device. However, lowering the deployment cost of adaptive models may also facilitate their use in applications with privacy, surveillance, or fairness concerns. MetaCLBench evaluates deployment efficiency rather than the societal suitability of particular applications, and its results should not be interpreted as evidence that an evaluated method is safe or fair for deployment. Furthermore, conclusions are limited to the evaluated datasets and hardware configurations.

D Extended Related Work

D.1 Extended Related Work

Continual Learning: CL aims to learn new knowledge from dynamically evolving input data distributions [41, 64]. When algorithms are trained sequentially, the new learning will interfere with previous established weights, leading to CF of the previous learning [43]. Current CL approaches to the CF challenge can be divided into two categories based on task information management. The first approach incorporates regularisation terms into the loss function to facilitate knowledge of old and new models, thereby preserving previously learned knowledge [1, 39, 48, 50]. The second approach involves either original or pseudo-samples from previous tasks during new task training [4, 51, 58].

Meta Continual Learning: Meta-CL updates the model in the outer loop using learned random samples and optimizes it in the inner loop with a few samples before passing it to the outer loop [23]. Previous research on Meta-CL has primarily concentrated on image recognition applications. Online-aware Meta-Learning (OML) [22] and A Neuromodulated Meta-Learning (ANML) [3] both demonstrated high CL performance on the Omniglot dataset [37]. Based on these approaches, state-of-the-art (SOTA) Meta-CL incorporated the Attentive Independent Mechanisms (AIM) module, which can independently acquire new knowledge to enhance model performance [38]. However, due to its high processing cost, using Meta-CL is challenging on low-end edge devices [35]. LifeLearner addresses

these limitations by integrating meta-learning, rehearsal mechanisms, and resource-efficient compression, making it suitable for deploying Meta-CL applications on resource-constrained devices [35].

Continual Learning for Image and Audio Applications: CL has remarkable capabilities in sensing applications, including action recognition, audio sensing, and speech recognition [24, 32]. Its ability to update models based on continuous data streams makes it particularly valuable for mobile devices processing speech and image data. Researchers have successfully applied CL to speech and emotion recognition [62]. CL systems integrated into edge devices with audio and microphones continually learn from ambient human voices and environmental sounds to improve recognition accuracy [33]. In image recognition, CL facilitates applications such as food recognition [14], where systems must continuously learn from new instances to enhance recognition accuracy.

Prior work demonstrates CL's value for IoT sensing applications, yet does not address a fundamental question facing practitioners: *which methods can actually deploy on target hardware while achieving acceptable accuracy?* Our work fills this gap by evaluating Meta-CL methods for deployment viability, not just accuracy. We measure memory footprint, latency, and energy consumption on three device classes, revealing that several methods optimized for accuracy fail to run on typical IoT hardware. This deployment-oriented evaluation across image and audio modalities, using architectures from lightweight CNNs to ViT and YAMNet, provides actionable guidance for IoT system designers.

D.2 Continual Learning

Neural network training is difficult to shift from batch learning to a continuous learning model because neural networks are different from human brain mechanisms. Deep neural network-based systems that are continuously updated with new data suffer from a severe forgetting problem, where performance on old tasks typically drops significantly, known as CF [24, 32, 43, 44, 64]. Researchers have put forward numerous approaches to prevent CF, mostly encompassing rehearsal-based, regularisation-based, and parameter isolation-based strategies.

(1) Rehearsal-based approaches [5, 35, 49, 67] allow storing some of the early training data, i.e., old data, in a cache, which will be used for replay or prototype as new tasks are learned as a way of preventing CF. At this point, cache size, storage data selection, data storage form, and the way old and new data are mixed are all important influencing factors. Rehearsal-based approaches include memory-based, generation-based replay, and other approaches. Memory-based approaches generally necessitate a memory buffer for storing data instances or pertinent information from the previous task. This stored information is then retrieved and utilized during the learning of a new task to reinforce the knowledge acquired from the prior activity. Rebff et al. first proposed a strategy for learning both the classifier and the feature representations simultaneously during a continuous learning process, which became an incremental classifier(iCARL) based on representational learning [49]. During representation learning, iCARL is trained using a small number of data instances representative of the old category stored and instances from the new task. Wu et al. suggest that the present approaches, which demonstrate good performance on small datasets with only a few tasks, are unable to sustain their performance on huge datasets with thousands of tasks [67]. Therefore, they proposed bias-corrected (BIC) methods that focus on continuous learning from large datasets. Because there is actually a strong bias in the classification layer for each new task, BIC segregates the validation set from the amalgamation of previous and new data instances and incorporates a linear bias correction layer subsequent to the classification layer. This correction layer utilizes the validation set to rectify any bias present.

Generative replay-based methods are an alternative to memory-based methods, using generative modules that can reproduce information related to previous tasks instead of memory buffers. Shin et al. proposed a deep generative replay method (DMC) that sequentially learns multiple tasks by mimicking pseudo-data generated from previous training examples by a GAN model and pairing them with corresponding labels [58]. Nevertheless, a significant limitation is that the efficacy of the statement is contingent upon the caliber of the generator. Deep model consolidation (DMC) was proposed by Zhang et al [75]. The process begins by training a distinct model for the new class using labeled data. Subsequently, the old and new models are merged by utilising publicly accessible unlabeled auxiliary data. This approach overcomes the difficulties because of the inaccessibility of legacy data and mines these auxiliary data for rich transferable representations to facilitate incremental learning.

Generative replay methods have been shown to be effective on problems with relatively simple inputs, but for more complex input problems, such as natural images, generative replay methods may face greater challenges. While generative replay still has some advantages in class incremental learning problems using natural images, these methods rely on pre-trained networks or extensive non-incremental initialisation phases.

(2) Regularisation-based approaches [29, 72] usually impose restrictions on various model parameters and hyperparameters during the updating process to achieve the goal of consolidating knowledge learned in a straight line while learning a new task, thus mitigating CF in continuous learning. Regularisation-based approaches include methods such as optimising constraints on important model parameters to add distillation losses with respect to the model as an objective.

A common approach is to enhance the loss function by regularising previously acquired knowledge, which is a technique that gained recognition as Enhanced Weight Consolidation (EWC) was first embodied by Kirkpatrick et al. [29]. EWC introduces a novel quadratic penalty term in the loss function to limit pattern modifications to weights that were more important for the previous learning task. In order to limit the parameter to low-loss regions that are common between tasks, instead of only updating low-loss regions for new tasks by calculating the importance of the weights using the diagonal of the Fisher information matrix [71], thus mitigating the CF problem. The rebalancing uniform classifier LUCIR method was proposed by Hou et al. [18]. This method proposes three loss functions to bind the bias problem present in incremental learning due to the imbalance between old and new samples. The data is rebalanced by re-assigning weights in each iteration cycle. This means that for a few class instances, higher weights will be assigned; for the majority of class instances, lower weights will be assigned. This reallocation technique ensures that the model pays more attention to the minority class, thus improving the overall classification performance.

Methods utilising knowledge distillation [58] incorporate the concept of knowledge distillation into incremental learning by extracting knowledge from a model trained on a previous task into a model trained on a new task, therefore consolidating previously acquired knowledge. Li et al. proposed a method for learning without forgetting (LWF, Learning without Forgetting) [39]. Prior to acquiring new knowledge, LWF retains a duplicate of the preceding model parameters. Subsequently, the model acquired from the new task data instances is employed as the classifier target for the old task. The classifier for the new task is utilized with the objective of attaining the real value. The total of the losses from the new task and old task is calculated and employed as the ultimate loss. Douilaed et al. refer to continuous learning as representation learning and propose a distillation loss that becomes a summarized output distillation (POD) [10], which restricts the updating of the final output and the learned representation of the middle layer. The disadvantage of such methods is that after the first task, all the parameters of the hidden layer are frozen but representation learning is limited.

(3) Parameter isolation-based approaches [42, 69] typically involve expanding the old model on new tasks and assigning different model parameters to different tasks with varying degrees of isolation to prevent subsequent tasks from interfering with previously learned knowledge.

Parameter isolation-based approaches allocate distinct model parameters to individual tasks through the dynamic modification of the network architecture. Yoon et al. introduced a Dynamically Extensible Network (DEN) [69]that leverages acquired information from previous jobs and expands the network structure when the existing knowledge is inadequate for handling new tasks. Neurons undergo processes of addition, duplication, and separation to expand the structure of the network.

In addition to using a parameter isolation-based approach to dynamically change the network architecture, the researchers also designed a fixed network architecture. This architecture can still assign different parameters to handle different tasks. Mallya et al. proposed PackNet [42], a method that first fixes the parameters of the old task and second trains the entire neural network for each new task. After the training is completed, unnecessary parameters are released using weight-based pruning techniques and redundant spatial models are retained. Fernando et al. proposed PathNet [11]which initiates the training process by randomly selecting a few paths within the network. A race selection genetic algorithm is then employed to choose the most optimal paths for training on the given task. Afterward, for each subsequent task, the model parameters on all the paths chosen in the prior tasks are locked, while the remaining parameters are reset and trained again using the aforementioned approach to ultimately achieve the optimal path. Afterward, for each subsequent job, the model parameters on all the previously selected paths are fixed, while the remaining parameters are reset and trained again using the aforementioned method in order to ultimately choose the optimal path.

The method based on parameter isolation is simply to cope with the learning of a new task by adding parameters to the original model, but this method does not eliminate the interference between the new parameters and the old ones, nor does it ensure that the new parameters will not overwrite the old ones. Therefore, even with the parameter isolation-based approach, incremental learning is still limited by the problem of CF.

(4) Recently, preserving the structure of previously acquired knowledge has also emerged as a compelling area of research. Tao et al. proposed the Topology Preserving Class Incremental Learning (TPCIL) approach [61]to store resilient Hebbian graphs [59]in buffers for data playback effects rather than just stored instances of prior data. TPCIL is motivated by the concept derived from human cognitive research that the process of forgetting is a result of the destruction of topological structures in human memory, and the method mitigates CF by injecting topological retention terms into the loss function.

The 'regularisation and parameter isolation based approach' mentioned above sometimes does not work as well as it should. This is because the approach is not always consistent with new incremental learning and the advantages can even be negative.

D.3 Meta Learning

Meta Learning, also known as few-shot learning, is one of the hotspots of current research. Few-shot learning aims to use few training samples to recognize new classes that are not concatenable, regardless of the model's performance in recognising the base class, and its core problem is that deep models tend to be overfitted to a limited number of training samples [17]. To mitigate this issue, researchers have proposed a number of methods, which mainly include three main categories: model-based fine-tuning, data-based enhancement, and migration learning [65].

(1) Meta learning methods [2, 6, 12] based on model fine-tuning are more traditional approaches, which typically pre-train models on large-scale data and then fine-tune the existing parameters through regularisation. Because of the limited quantity of training samples, how to tune the parameters without leading to overfitting becomes a key issue, for which the researchers proposed the following solution. a. Early stopping: Arik et al. separated the validation set from the training set in order to oversee the training process, and the training will cease when there is no discernible enhancement in the performance of the validation set [2]. b. Selective updating of parameters: Keshari and his team gave a set of pre-trained filters that only learn the parameters that are multiplied by the filters, and this selective updating of the parameters effectively mitigates overfitting [28]. c. Simultaneous updating of relevant parts of the parameters: Yoo et al. group the pre-trained neural networks so that the filters are fine-tuned based on some auxiliary information using the training set via grouped backpropagation for the purpose of jointly updating each grouping using the same update information [68]. d. Using model regression networks: Kozerawski et al. mapped sample embeddings into a transformation function for classification decision boundaries [30]. The model regression network represents a transformation that is independent of the specific task, which translates parameter values acquired via training on a limited number of cases to parameter values acquired through training on a substantial number of samples. Although the model-based fine-tuning approach is simpler, the differences between the source dataset and the target dataset may lead to overfitting of the model on the target dataset during use.

(2) The fundamental problem of Meta learning is that the sample size is too small, the problem can be supplemented by data enhancement methods to increase the diversity of the samples and supplement its insufficient number of images. Delta encoders were proposed by Schwartz et al [55]. The model can extract transferable intraclass deformations between training samples of the same class and apply these increments to small samples of the new class, effectively synthesising new class samples as it goes. However, in this method, the features usually extracted by the classification network only focus on the most discriminative regions, while ignoring other less discriminative regions, which is not conducive to the generalisation of the network. To solve this problem, Shen et al. replaced the fixed attention mechanism with an indeterminate one [56]. First, the input image is extracted with features, and then the average is pooled and classified to obtain the cross-entropy loss. This loss is then used to derive the gradient for the uncertain attention mechanism, thus updating the model.

(3) Transfer learning [36, 70] is the use of knowledge already learned on one task to improve learning on another task. Its main goal is to quickly transfer what has already been learned to a new domain. a.

Metric-based learning: Vinyals et al. proposed a Matching Network, which maps training samples and validation samples to corresponding labels [63]. Then the samples are mapped into a low dimensional vector space using LSTM to compute the similarity between the new sample and each sample. Finally the predicted labels are output using the kernel density estimation function. b. Meta learning based approach: Finn et al. proposed a model-independent Model Aagnstic Meta Learning (MAML), aiming at obtaining a good general initialisation parameter for the model and achieving fast fine-tuning of the model on new tasks [12]. The strategy learns features that are applicable to a variety of tasks, allowing the model to quickly adjust the model's weights so that it can be generalized and quickly adapted to new tasks. However, its disadvantage makes it necessary to train with a sufficient number of tasks to converge. For this reason, Sun et al. proposed Meta Transfer Learning (MTL) for Few-shot learning. This method enables MAML to learn only the last layer as a classifier [60]. It is first trained on a large-scale dataset to get a full-time deep neural network and fixes the lower-level convolutional layers as feature extractors. The parameters of the feature extractor neurons are subsequently acquired through learning to ensure their rapid adaptability to the Few-shot job. When only a small amount of labeled data is used, the method can help deep neural networks converge quickly and reduce the probability of overfitting occurring. Rusu et al. proposed the LEO method based on MAML to address the uncertainty of its use of Meta learning [53]. The method first initializes the parameters of the model for each task. Its parameters are sampled from a conditional probability distribution associated with the training data. Secondly, the parameters are updated in a low-dimensional hidden space to adapt the model more efficiently.

D.4 Meta Continual Learning

Most of the traditional methods of class Continuous Learning lead to significant performance degradation when applied directly to Meta-CL. Firstly, because of the limited number of samples in the new class during Meta-CL, the model is prone to overfitting into the new class, resulting in a loss of its capacity to generalize to large sample tests. Second is the dilemma of facing a performance tradeoff between the old and new classes. Due to the fact that in the case of very few samples, it is necessary to increase the learning rate as well as enhance the gradient of the new class loss in order to learn the relevant knowledge of the new class. However, this will make it more difficult to maintain the relevant knowledge of the old class. Therefore, researchers address Meta-CL from a new cognitive perspective.

The problem of Meta-CL was raised by Tao et al. [61] at first, and they also proposed the TOpology-Preserving knowledge InCrementer (TOPIC) framework. The original version of the neural network was used for unsupervised learning during input. To accomplish the Few-shot supervised incremental learning task for each training task, the research team improved it. The improved supervised neural gas network allows for incremental growth of nodes and edges through competitive learning. Based on this, this study also designed the neural gas robustness loss function that can effectively suppress the forgetting of old categories and the neural gas adaptation function that reduces the overfitting in new categories. The TOPIC framework can alleviate the forgetting of old knowledge by stabilising the topology of the neural gas network NGs. It also improves representation learning on a small number of new category samples by making the NG grow [13]and adapt to new training samples to prevent overfitting to new categories with fewer samples.

To solve the Few-shot incremental learning problem, Cheragian et al. proposed a Semantic-aware Knowledge Distillation approach [7]. The main idea is to introduce semantic information in knowledge distillation to accomplish the preservation of old things in the process of Meta-CL. The method first transforms the labels into word vectors. The input image is mapped to the space of its label word vectors to minimize the distance between the two vectors. No additional neural network parameters need to be added as new categories emerge, thus reducing the dependence of network training on the amount of data. Shi et al. proposed a method based on the discovery of flat minima (F2M) [57] for model training in incremental Few-shot learning. An average minimum is when the model parameters are in a flat region and when the loss function value is minimized. This flat region prevents the model from overfitting and preserves previously learned knowledge when learning new data. A Continuous Evolutionary Classifier (CEC) was proposed by Zhang et al [74]. The method initially separates the feature extraction module from the classifier, where the feature extraction module is frozen to avoid generating forgetfulness and only the classifier is changed for each incremental task. The process is that whenever vectors for a new category are generated, the vectors obtained from all categories, old and new, will be aggregated together. Where each vector is a node and the nodes are

aggregated based on an attention mechanism that utilizes the invariance of the connections between the nodes to maintain the old knowledge. Finally, the graph attention model is utilized to generate the final classifier by combining the information of the emerged categories. A self-promoting prototype refinement mechanism (SPPR) was proposed by Zhu et al [77]. The technique uses the correlation matrix between newly introduced category samples and pre-existing category prototypes to revise the current prototypes. This mechanism is realized by dynamic relational projection. The heart-tired sample representations and the just-tired prototypes are mapped into the same embedding space, and the two embeddings are computed in the above-mentioned distance metric in the space to compute the space of projection matrices between them. Ultimately, this matrix space serves as a weight for prototype refinement to guide the dynamics of the prototypes toward preserving existing knowledge and enhancing the defensibility of new categories.

Recently, Zhou et al. proposed a Forward Compatible Training (FACT) [76]. It makes it easy to merge new categories and facilitates the seamless integration of these categories into the existing model. In order to enable the model to be expandable, they designate many virtual prototypes in advance within the embedding area as reserved space and optimize these virtual circles to push instances of the same class closer together, thus reserving more space for incoming new classes. Subsequently, Constrained Few Shot Incremental Learning (C-FSCIL) was proposed by Hersche et al [15]. The method first creates and updates the average prototype vector so that it serves as the average of the samples in memory. It then polarizes the prototype vector and retraces the fully connected layer for no more than a constant number of iterations. Finally, the average prototype vector is pushed so that it is quasi-orthogonal and remains close to the average prototype at the original site.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the scope, contributions, and main findings of MetaCLBench, which are supported by the experiments.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the benchmark's limitations, including its finite coverage of methods, datasets, architectures, and edge platforms.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: This paper does not include theoretical results requiring formal proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The benchmark protocol, methods, datasets, architectures, training settings, and measurement procedures are described in Section 2 and Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All datasets are publicly available and cited, and we provide access to the MetaCLBench implementation and evaluation framework.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: The paper specifies the task setting, dataset splits, architectures, devices, evaluation metrics, and detailed experimental settings.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The current results do not report error bars, confidence intervals, or statistical significance tests across repeated runs.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the edge-device specifications and provide the meta-training compute environment and implementation details in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conducted in this paper conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the potential benefits and risks of resource-efficient on-device continual learning in the appendix.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The work does not release high-risk models or datasets that require additional safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All existing datasets and pretrained models used in this work are credited to their original creators.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: MetaCLBench is released as a new benchmark framework, with its protocol, configurations, metrics, and usage documented in the paper and repository.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as an important or original component of the methodology or experiments.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.