MEMORY CONSTRAINED DYNAMIC SUBNETWORK UPDATE FOR TRANSFER LEARNING

Aël Quélennec **Pavlo Mozharovskyi** **Van-Tam Nguyen** **Enzo Tartaglione**
LTCI, Télécom Paris, Institut Polytechnique de Paris
19 Place Marguerite Perey, 91120 Palaiseau, France
{name.surname}@telecom-paris.fr

ABSTRACT

On-device neural network training faces critical memory constraints that limit the adaptation of pre-trained models to downstream tasks. We present MeDyate, a theoretically-grounded framework for memory-constrained dynamic subnetwork adaptation. Our approach introduces two key innovations: LaRa (Layer Ranking), an improved layer importance metric that enables principled layer pre-selection, and a dynamic channel sampling strategy that exploits the temporal stability of channel importance distributions during fine-tuning. MeDyate dynamically resamples channels between epochs according to importance-weighted probabilities, ensuring comprehensive parameter space exploration while respecting strict memory budgets. Extensive evaluation across a large panel of tasks and architectures demonstrates that MeDyate achieves state-of-the-art performance under extreme memory constraints, consistently outperforming existing static and dynamic approaches while maintaining high computational efficiency. Our method represents a significant step towards enabling efficient on-device learning by demonstrating effective fine-tuning with memory budgets as low as a few hundred kB of RAM.

1 Introduction

The exponential growth of deep neural networks has fundamentally transformed artificial intelligence, enabling breakthrough performances across diverse domains including Computer Vision [20, 42, 30], Natural Language Processing [46, 45], and Speech Recognition [10, 31]. This remarkable progress stems from the continuous scaling of model complexity, with the number of parameters doubling every 8 to 17 months since the advent of the "Large Scale Era" marked by AlphaGo's release in 2015 [41]. While this trend further demonstrates the intrisic generalization potential of deep learning, it raises profound ecological and technical concerns. Training and exploitation of these architectures require very high energy consumption, and their deployment in real-world environments is impossible without extensive compression, leading to performance degradation and creating an increasingly critical tension between model capabilities and practical deployment constraints.

The proliferation of Internet of Things (IoT) devices and the growing demand for Edge AI applications have intensified the need for efficient on-device solutions. Traditional approaches to edge deployment follow a paradigm of offline training on powerful hardware, followed by model compression and deployment for inference-only applications on resource-constrained devices. This research domain encompasses five principal methodologies, namely quantization techniques for precision reduction, low-rank decomposition for parameter compression, architectural innovations for compact model design, knowledge distillation for teacher-student learning, and network sparsification (commonly referred to as pruning) for structural optimization [5, 9]. However, this paradigm suffers from fundamental limitations as models trained on static datasets inevitably experience performance degradation when deployed in dynamic real-world environments due to *data drift* phenomena [39]. The inability to adapt and learn continuously after deployment limits the practical viability of edge AI systems, particularly in applications requiring personalization, privacy preservation, and real-time adaptation to evolving data distributions [16].

The primary obstacle preventing widespread adoption of on-device learning lies in the prohibitive computational and memory demands of backpropagation, as it requires storing intermediate activations and computing gradients, leading to memory requirements that can exceed device capabilities by several orders of magnitude. This has moti-

vated exploration of alternative learning paradigms, including the Forward-Forward algorithm [15], hyperdimensional computing [48], and gradient-free optimization methods [36]. However, these approaches consistently underperform compared to traditional backpropagation-based techniques, creating a performance-efficiency trade-off that has yet to be satisfactorily resolved.

Recent advances in memory-constrained transfer learning have demonstrated that strategic subnetwork selection offers a promising path toward bridging this performance gap. Pioneering work by [26] showed that fine-tuning within extreme memory budgets (256kB) is feasible through static subnetwork pre-selection. Improving on this concept, Quélennec *et al.* [38] propose to study the **Tra**ining **Dy**namics (TraDy) of network fine-tuning, thus yielding an approach that leverages the heavy-tailed behavior of stochastic gradients and the architectural consistency of layer importance across downstream tasks to dynamically resample channels between epochs, within pre-selected layers. Their method demonstrates that the synergistic combination of strategic layer pre-selection and dynamic channel selection enables state-of-the-art performance even under extreme memory constraints. Activation compression represents an orthogonal approach to memory-efficient training that targets the storage of intermediate representations rather than parameter selection, as demonstrated by Nguyen *et al.* [32] who compress activations stored for backpropagation. Despite achieving considerable memory compression and computational speedup, this method does not address the fundamental question of which network components to train, defaulting to a heuristic that updates only the final layers. Building upon Quélennec *et al.* insights, this work presents a comprehensive extension of the TraDy strategy in the form of **Me**mory-constrained **Dy**namic subnetwork upd**ate** (MeDyate). Our main contributions can be summarized as follows.

- We propose an improved layer ranking by leveraging the the heavy-tailed behavior of stochastic gradient (Sec. 3.2).
- We demonstrate that, for a given downstream task, the channel topology remains stable during training (Sec. 3.3).
- Given memory constraints, only a subset of relevant channels can be updated at a time. By dynamically adjusting the channels updated between epochs, performance improves compared to static approaches. We propose an adaptive algorithm that dynamically samples channels within key layers, assigning update probabilities based on their importance (Sec. 3.4).
- We test MeDyate in typical transfer learning setups, under extreme memory constraints, observing that it can achieve state-of-the-art performance across multiple efficient architectures (Sec. 4.3). Our approach allows us to drastically reduce FLOPs, weight and activation memory during training while demonstrating superior performance compared to other similar strategies.

2 Related Works

Activation Map Compression. A critical observation for memory-efficient training is that activation maps (the outputs of each layer after non-linearity application) occupy significantly more memory space than parameters during backpropagation, as they are essential for computing weight derivatives [3]. This insight has motivated a dedicated research direction focused on compressing activation maps using techniques adapted from weight compression literature, including quantization [11, 34], sparsification [21, 17], entropy encoding [13], wavelet transform methods [12] and most notably, Nguyen *et al.*'s application of tensor decomposition strategies, allowing for drastic reduction of memory usage alongside acceleration of the training process [32]. While MeDyate is primarily designed as a strategy for surgically selecting which parameters to fine-tune, it naturally induces activation sparsity, thereby contributing to this research domain as well.

Gradient Pruning. The concept of gradient pruning describes the selection of a specific subnetwork to train during backpropagation, while the remainder of the network remains frozen. It differs from classical pruning by maintaining the complete network architecture intact during inference, while selectively modifying the backpropagation phase through criterion-based gradient computation. Gradient pruning can either be applied to reduce the memory footprint in constrained environments, or to accelerate training while preserving on-task performance [50, 2, 24, 49, 29]. Within the transfer learning context, Lee *et al.* [23] provide particularly relevant insights by framing gradient pruning as a regularization mechanism, demonstrating that network blocks exhibit task-dependent contributions to downstream performance. In their work, they observe that such contribution can either be constructive or destructive with respect to the task and is predicted by the ration of gradient norm to parameter norm.

On-Device Learning. Regarding our design of the MeDyate algorithm, three lines of work have largely influenced our approach, each presenting strategies for gradient pruning in memory-constrained environments applied to

pre-trained architecture fine-tuning.
The foundational work by Lin *et al.* [26] combine selective parameter updating, in the form of Sparse Update schemes (SU), alongside operator reordering and quantization-aware scaling to enable fine-tuning on extreme edge devices. While demonstrating that memory-efficient subnetworks can achieve acceptable downstream performance, SU suffers from significant practical limitations: determining adequate sparse configurations requires computationally expensive offline analysis of accuracy contributions followed by evolutionary search for each network-budget combination, and the resulting static selections are applied uniformly across all downstream tasks under the implicit assumption that optimal layer-channel configurations remain fixed throughout training.
Kwon *et al.*[22] addressed some of these limitations by enhancing adaptability across architectures, datasets, and memory budgets. They achieve this through Fisher information computation on downstream task activations to rank layers and channels, followed by reweighting based on parameter count and MAC operations. However, this approach introduces a fundamental contradiction as computing Fisher information for all network channels requires more memory than the gradient computation it seeks to optimize. Moreover, as for SU, the selected subnetwork remains static during fine-tuning.
Finally, Quélennec *et al.* [38] developed a theoretically-grounded framework for dynamic subnetwork selection, building on evidence-based analysis rather than empirical observations. Their work established that stochastic gradients exhibit heavy-tailed behavior during transfer learning and that layer importance remains architecturally consistent across downstream tasks, enabling a-priori layer selection. Within these pre-selected layers, they implemented random channel sampling that dynamically resamples between epochs, demonstrating consistent performance improvements over static selection approaches. While providing valuable theoretical insights into gradient distribution patterns and validating the benefits of adaptive selection strategies, we believe that their layer and channel selection strategies can be improved to yield increased on-task performance. Building upon these foundational insights, our extended framework further analyzes the underlying dynamics of transfer learning in deep neural networks. We thus introducing MeDyate, an adaptive channel selection strategy that addresses the limitations of previous approaches while maintaining strict memory constraints for both parameters and activations.

3 Method

In this section, we present our comprehensive framework for parameter-efficient fine-tuning under extreme memory constraints. We begin by establishing the mathematical foundations and notation in Sec. 3.1, providing the formal basis for our approach. In Sec. 3.2, we present an alternative methodology for layer ranking through the introduction of a more suitable metric, while Sec. 3.3 analyzes the stability of channel topology throughout training, demonstrating how channel importance distributions remain stable during the fine-tuning process. Building on these observations, Sec. 3.4 introduces our core dynamic channel sampling strategy, MeDyate, which enables efficient transfer learning within strict memory budgets through stochastic channel resampling between epochs. Finally, Sec. 3.5 examines the properties of our algorithm in terms of both search space exploration and computational complexity.

3.1 Notations

Following the notation conventions established by Quelennec *et al.* in their TraDy framework, we focus our mathematical formulation on 2D convolutional operations within CNNs, omitting bias parameters for clarity. It is however important to note that similar analysis naturally extends to fully-connected architectures. Our goal is to extend their methodological framework through theoretical analysis to improve upon their proposed layer and channel selection strategies. Thus, in a similar fashion to their work, we address the challenge of adapting pre-trained neural networks to downstream tasks within stringent memory budgets, operating without *a priori* knowledge of the target domain. Our objective is to identify optimal subsets of the architecture for training, maximizing adaptation performance while maintaining both parameter and activation memory consumption within device constraints. The implicit assumption made here is that the memory constraint allows the device to perform the forward pass in its entirety, while the additional memory and latency requirements of backpropagation render full network training either impossible or impractical.
Let us then write the CNN as a composition of n convolutional transformations:

$$\mathcal{F}(\mathcal{X}) = \mathcal{C}_{\mathcal{W}_n} \circ \mathcal{C}_{\mathcal{W}_{n-1}} \circ \ldots \circ \mathcal{C}_{\mathcal{W}_1}(\mathcal{X}), \tag{1}$$

where $\mathcal{X}$ represents the network input and $\mathcal{W}_i \in \mathbb{R}^{C' \times C \times D \times D}$ denotes the convolutional kernels for layer i, with C, C' and D indicating input, output channel and kernel dimensions respectively.
For any layer i, we define the input activation tensor $\mathcal{A}_i \in \mathbb{R}^{B \times C \times H \times W}$ and output tensor $\mathcal{A}_{i+1} \in \mathbb{R}^{B \times C' \times H' \times W'}$, where B denotes batch size and (H, W) represent spatial dimensions of the feature maps.
The gradient computation for layer weights follows standard backpropagation principles. The network loss $\mathcal{L}$ propa-

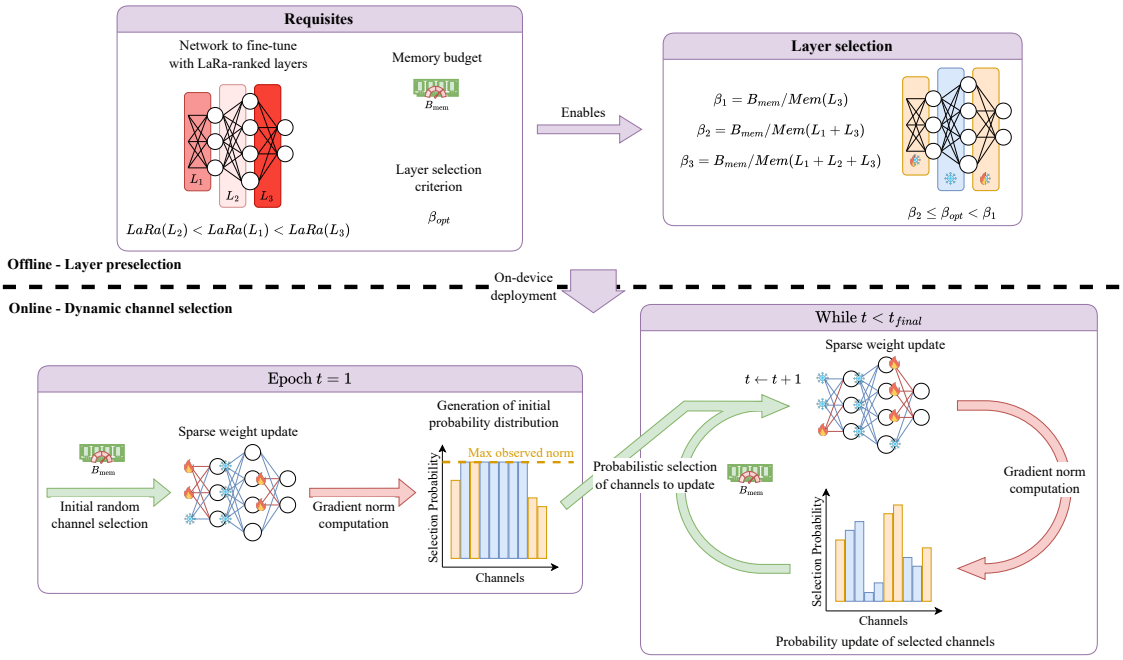


Figure 1: Overview of the MeDyate framework. The offline phase (top) shows layer pre-selection based on LaRa rankings and memory budget constraints, while the online phase (bottom) illustrates dynamic channel sampling with importance-weighted probabilities during training epochs.

gates backward from the output, generating activation gradients $\frac{\partial \mathcal{L}}{\partial \mathcal{A}_{i+1}}$ subsequently enabling weight derivative calculation:

$$\frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} = \text{conv}\left(\mathcal{A}_i, \frac{\partial \mathcal{L}}{\partial \mathcal{A}_{i+1}}\right), \tag{2}$$

with the explicit tensor formulation:

$$\left[\frac{\partial \mathcal{L}}{\partial \mathcal{W}_i}\right]_{c',c,k,l} = \sum_{b=1}^{B} \sum_{h'=1}^{H'} \sum_{w'=1}^{W'} [\mathcal{A}_i^p]_{b,c,h,w} \left[\frac{\partial \mathcal{L}}{\partial \mathcal{A}_{i+1}}\right]_{b,c',h',w'}, \tag{3}$$

where spatial indices follow $h = h' \cdot s + k \cdot d$ and $w = w' \cdot s + l \cdot d$ with stride s and dilation d, and $\mathcal{A}_i^p$ represents the appropriately padded input tensor.

Consistent with the methodology established by Quélennec *et al.* and Lin *et al.* before them, we adopt input channels as the fundamental granularity for parameter selection. This choice is motivated by three key considerations. First, operating at finer granularities would produce unstructured sparse tensors during weight derivative computation, contradicting our memory efficiency objectives. Second, coarser granularities sacrifice selection precision by increasing the likelihood of scenarios where only a subset of parameters within a block requires updating while the majority remains irrelevant, leading to inefficient memory allocation. Third, and most importantly, input channel freezing generates natural activation sparsity alongside weight sparsity which is a property unique to this dimension. When an input channel is designated for freezing prior to a training epoch, we can proactively release the corresponding activation values from memory during forward propagation rather than storing them for potential use in backpropagation, thereby achieving additional memory savings. This advantage becomes particularly significant given that activation memory typically dominates the memory footprint during backpropagation compared to parameter storage, as demonstrated by Cai *et al.* [3].

Throughout the remainder of this paper, we focus our analysis on two fundamental granularities for selective parameter updating: entire layers and individual input channels within those layers. These two levels enable efficient management of both weight and activation memory required for gradient computation. Moreover, as we will demonstrate,

efficient layer selection (or exclusion, depending on perspective) proves crucial for reducing the search space of our proposed sampling methods.

Based on equation 3, we derive analytical expressions for the memory requirements and computational complexity associated with updating a single input channel c within layer i. We define $\mathcal{C}_c^{\mathcal{W}_i} = C' \times D \times D$ as the weight memory cost and $\mathcal{C}_c^{\mathcal{A}_i} = H \times W$ as the activation memory cost for channel c. The total space complexity $(\Theta_{\text{space}})_c$ and time complexity $(\Theta_{\text{time}})_c$ for processing one input sample during backpropagation are as follows:

$$\left(\Theta_{\text{space}}\right)_c = \mathcal{C}_c^{\mathcal{W}_i} + \mathcal{C}_c^{\mathcal{A}_i}, \tag{4}$$

$$\left(\Theta_{\text{time}}\right)_c = D^2 C' H' W'. \tag{5}$$

3.2 Alternative Layer Ranking

In their work, Quélennec *et al.* invoke two foundational studies to demonstrate that stochastic gradients are implicitly compressible. The first establishes that stochastic gradient noise exhibits heavy-tailed behavior during neural network training with stochastic gradient descent (SGD), as demonstrated by Şimşekli *et al.* [43]. More precisely, the gradient noise $U_k(\mathcal{W})$ follows a symmetric α-stable distribution:

$$U_k(\mathcal{W}) = \Delta\tilde{\mathcal{W}}_k - \Delta\mathcal{W} \sim S\alpha S(\sigma), \tag{6}$$

where $\Delta\mathcal{W}$ represents the full-batch gradient, $\Delta\tilde{\mathcal{W}}_k$ the stochastic gradient computed from k samples, and $\alpha \in (0, 2]$ characterizes the tail heaviness (with decay proportional to $1/|x|^{\alpha+1}$). From equation 6, Quélennec *et al.* observe that gradients naturally incorporate heavy-tailed noise components during stochastic optimization.

Complementing this theoretical foundation, Wan *et al.* demonstrate that injecting heavy-tailed noise into weights during backpropagation renders them provably more compressible through pruning. The underlying mechanism is that heavy-tailed noise causes weight matrix columns to follow multivariate heavy-tailed distributions independently from each other. Consequently, the norm distribution becomes highly skewed: a small subset of columns exhibits disproportionately large norms while the majority remain relatively small. This concentration phenomenon means that the overall weight matrix norm is predominantly determined by a few dominant columns, creating an implicit sparse structure that aligns naturally with selective update requirements.

The natural conjunction of these two observations is that gradients, being composed of stochastic gradients and heavy-tailed noise, exhibit the same implicit compressibility properties described by Wan *et al.*. Consequently, Quélennec *et al.* propose to exploit input channel gradient norms to characterize update importance. More precisely, they reweight each channel's gradient norm by its associated memory cost as defined in equation 4, producing a metric that combines both update relevance and memory efficiency. They denote this channel-level metric as RGN (Reweighted Gradient Norm):

$$\text{RGN}_c = \frac{1}{\mathcal{C}_c^{\mathcal{W}_i} + \mathcal{C}_c^{\mathcal{A}_i}} \sqrt{\sum_{c',k,l} \left[\frac{\partial \mathcal{L}}{\partial \mathcal{W}_i}\right]^2_{c',c,k,l}}. \tag{7}$$

Quélennec *et al.* then go on to hypothesize that layers can also be pruned from the training in a similar fashion to that of channels. In order to efficiently select which layers to freeze and which layers to update, they introduce a layer-related importance metric in the form of the sum of channels RGN:

$$\left\|\frac{\partial \mathcal{L}}{\partial \mathcal{W}_i}\right\|_{\text{RGN}} = \frac{1}{\left(\Theta_{\text{space}}\right)_c} \sum_{c=1}^{C} \left\|\left(\frac{\partial \mathcal{L}}{\partial \mathcal{W}_i}\right)_c\right\|_2. \tag{8}$$

However, we contend that this metric fails to capture the holistic behavior of entire layers. As the sum of individual channel gradient norms within a layer, the metric becomes dominated by channels with high values of high gradient norm, irrespective of how many channels exhibit comparatively low norms. Moreover, since the reweighting factor in equation 7 corresponds to the memory cost of updating individual channels, and all channels within a layer share identical dimensions, this factor can be extracted from the summation as shown in equation 8. Consequently, while this metric tends to favor layers containing channels that achieve favorable trade-offs between importance and update cost, it provides limited insight into overall layer behavior.

This limitation becomes particularly problematic in our framework, where layer selection and exclusion represent complementary aspects of the same decision. When a layer is excluded, it remains frozen throughout the entire training duration; conversely, when selected, we can reasonably assume complete layer coverage over time through the stochastic sampling approaches introduced by Quélennec *et al.* and extended in this work. Given this binary layer treatment, the metric in equation 8 essentially characterizes individual channel behavior within layers rather than providing a comprehensive assessment of layer-level significance.

To address this limitation, we propose a layer-level metric that incorporates both the actual norm of the layer gradient and its complete memory cost. We denote this metric as LaRa (**La**yer **Ra**nking) and define it as:

$$\text{LaRa}_i = \frac{\left\| \frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} \right\|_2}{\sum_{c=1}^{C} \left(\mathcal{C}_c^{\mathcal{W}_i} + \mathcal{C}_c^{\mathcal{A}_i} \right)}, \tag{9}$$

where $\left\| \frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} \right\|_2$ represents the Euclidean norm of the entire layer gradient tensor. Beyond providing improved layer characterization, this metric preserves the property of layer ranking stability during fine-tuning across downstream tasks, as the theoretical inequalities established in prior work are based on Euclidean norms. In the experimental section (Sec. 4.2), we empirically validate that this new metric enables similar or superior performance with fewer layers while respecting memory budget constraints. As demonstrated in Sec. 3.4, this reduction in the number of layers requiring updates proves crucial for the design of our sampling algorithm.

3.3 Stability of Channel Importance Throughout Training

A fundamental insight from the TraDy framework is that channel importance distributions (also termed "channel topology" by Quélennec *et al.*) exhibit task-dependent variations that prevent *a priori* channel selection across different downstream tasks. This task-specific property implies that static pre-computed subnetwork selection will underperform compared to task-adaptive approaches. However, in memory-constrained environments that prevent full gradient computation, reliable estimation of channel importance remains a complex challenge.

As established by Quélennec *et al.*, the distribution of channel gradient norms varies significantly between downstream tasks. This observation stems from the fundamental composition of weight derivatives as expressed in equation 3, where both activation maps and activation derivatives are inherently shaped by task-specific data characteristics. The activation maps $\mathcal{A}_i$ capture features extracted at each network layer and reflect how the network processes input data, while activation derivatives $\frac{\partial \mathcal{L}}{\partial \mathcal{A}_{i+1}}$ encode task-specific loss landscape information that varies according to the downstream objective.

We observe, however, that channel topology exhibits remarkable stability throughout training within a given downstream task. More precisely, this temporal consistency unfolds through two subsequent phases:

- **Rapid Stabilization:** Channel importance distributions evolve quickly during initial training epochs but then stabilize for the remainder of the fine-tuning process. This rapid convergence suggests that the relative importance of channels becomes established early in adaptation and remains consistent as the network refines its task-specific representations.
- **Distributional Consistency:** Once stabilized, the channel topology maintains its structure throughout training, with channels preserving their relative importance rankings even as absolute gradient magnitudes may fluctuate during optimization (typically, average gradient norms decrease throughout training as a result of loss minimization).

This behavior can be explained by the fundamental nature of fine-tuning pre-trained architectures. During fine-tuning, the most significant weight modifications occur in the initial stages as the network adapts to the new loss landscape. Since fine-tuning typically assumes that downstream task features are sufficiently similar to those of the pre-training task, the network requires only targeted adaptations rather than complete relearning. In the first few epochs (or steps, depending on task similarity), loss and gradient magnitudes are relatively high as the network adjusts to new features, resulting in substantial weight changes that cause channel importance fluctuations.

Subsequently, as the network converges toward a local minimum, the rate of loss reduction decreases and absolute gradient values become smaller. Consequently, we observe overall weight stabilization with only minor adjustments. This stability propagates through the computational graph: as weights $\mathcal{W}_i$ stabilize and the input dataset remains fixed, the activations $\mathcal{A}_i$ also achieve stability. This stability then cascades to activation derivatives and ultimately to weight derivatives, which represent combinations of these stabilized components as expressed in equation 2.

This temporal stability has profound implications for memory-constrained channel selection. Since channel importance distributions remain consistent within tasks after initial stabilization, we can reliably estimate the overall channel topology through sampling approaches, even when memory constraints prevent computing gradients for all channels simultaneously.

Furthermore, in the TraDy framework Quélennec *et al.* showcased that dynamic channel selection strategies consistently outperform their static counterparts under memory constraints. This superiority of dynamic approaches naturally aligns with our temporal stability observations: if channel importance distributions remain stable over time,

then dynamic sampling can effectively explore this distribution while respecting memory limitations, ultimately approximating the performance approaches that would have full gradient knowledge.
The combination of temporal stability and dynamic sampling superiority forms the theoretical foundation for our dynamic channel selection strategy. By leveraging the stable channel topology within tasks, we can design sampling algorithms that efficiently explore channel importance space while maintaining strict memory constraints, confident that our estimates will remain representative throughout the training process.

3.4 Dynamic Chanel Sampling

The insights developed in the previous subsections directly inform the design of our dynamic channel sampling algorithm. The demonstrated stability of channel importance distributions within tasks enables us to dynamically sample channels for updating across epochs, yielding a faithful estimation of the underlying channel topology over time. Simultaneously, our alternative layer ranking methodology allows us to significantly reduce the search space for sampling by excluding inefficient layers *a priori*, thereby accelerating convergence as the memory budget is concentrated within layers known to be more beneficial for adaptation.
Building upon these foundations, we introduce several algorithmic innovations. First, while Quélennec *et al.* employed reweighted gradient norms as channel importance metrics, we contend that our LaRa-based layer exclusion enables a shift to raw gradient norms as channel importance indicators within the selected layer subset. Since LaRa already incorporates memory cost reweighting at the layer level, inefficient high-memory layers are excluded, allowing raw channel norms to serve as more direct measures of importance within the remaining efficient layers. Second, rather than deterministically selecting channels based on importance rankings, we propose a probabilistic selection strategy where each channel receives an update probability proportional to its importance. This stochastic approach ensures that all channels within selected layers receive updates at some point during training, while channels with higher importance naturally receive more frequent attention, leading to more robust and comprehensive network adaptation.
While our LaRa metric provides layer rankings, it does not inherently specify a stopping criterion for layer selection. Quélennec *et al.* employed a fixed number of layers regardless of memory budget constraints, determined by ranking layers according to their RGN values and applying a threshold that captures 97% of the cumulative gradient norm on a given downstream task. However, we argue that our more sophisticated channel selection strategy necessitates careful adaptation of the layer count K based on the available memory budget. This requirement stems from the need to balance exploration and exploitation: we must adequately capture channel topology within few epochs while avoiding excessive search space reduction that would permanently exclude relevant channels from consideration.
To address this challenge, we introduce a hyperparameter α_K that represents the ratio between the total memory budget B_{mem} and the memory footprint M_K of the K selected layers constituting our search space:

$$\alpha_K = \frac{B_{\text{mem}}}{M_K}. \tag{10}$$

This formulation enables principled adjustment of the layer search space based on memory constraints, ensuring that our sampling strategy operates within an appropriately sized parameter space. In Sec. 4.2, we detail the systematic exploration conducted to determine the optimal value α_{opt}. Given this optimal proportion and a specific memory budget, we can then adaptively select the K_{opt} layers for our search space according to the ranking established by our LaRa metric.
We present MeDyate, our dynamic subnetwork update pipeline for memory-constrained transfer learning, as detailed in Algorithm 1. Prior to training, the top K layers constituting the search space are selected such that α_K, as defined in equation 10, represents the largest value not exceeding α_{opt} (line 1).
The training process then proceeds as follows. Given a pre-trained backbone and training dataset, channels are initially sampled uniformly at random within the predefined layer set L_K, subject to the memory budget constraint (line 4). In the second epoch, we compute the gradient norms of previously selected channels (line 7) and update the sampling distribution by assigning corresponding norm values to sampled channels while setting unselected channels to the maximum observed norm value (line 9). As established in Sec. 3.4, gradient magnitudes typically decrease throughout fine-tuning. Consequently, by assigning the maximum observed norm to unseen channels, we effectively grant them maximum selection probability, thereby promoting early exploration of the channel space.
Subsequently, channels are resampled according to a probability distribution proportional to the constructed gradient norm vector, again respecting the memory budget (line 10), and the selected channels are updated (line 11). From the third epoch onward, the maximum norm assignment step (line 9) is omitted, and we iteratively resample and update channels according to the probability distribution derived from the established norm vector. Upon completion of training, we evaluate model performance on the test dataset (line 12).
Fig. 1 illustrates the operational flow of our MeDyate algorithm across its two distinct phases. In the offline phase, we assume the LaRa metric has been pre-computed through a preliminary fine-tuning run on available downstream task

Algorithm 1: MeDyate

Require: Pre-trained backbone weights $\mathcal{W}$, LaRa ranked layers L, initial empty channel norm vector N, number of epochs n, Train data D_{train}, Test data D_{test}, memory budget B_{mem}.

Hyper-Parameters: Top K layers selection criterion α_{opt}.

Initialization: Select the top K layers L_K such that $\alpha_{K-1} > \alpha_{\text{opt}}$ and $\alpha_K \leq \alpha_{\text{opt}}$.

for *epoch* $= 1, ..., n$ **do**

  if *epoch* $= 1$ **then**

    Randomly sample channels C_t within L_K along uniform probability distribution until the memory budget B_{mem} is met.

  else if *epoch* > 1 **then**

    Compute gradient norm $\left\| \left(\frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} \right)_c \right\|_2$ of channels selected at previous epoch.

    Update the corresponding norm values in N.

    if *epoch* $= 2$ **then**

      Set the norm value for non-selected channels $\bar{C}_t$ to be the maximum norm value observed, $\max_c \left[\left\| \left(\frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} \right)_c \right\|_2 \right]$.

    Resample channels C_t along probability distribution proportional to N until the memory budget B_{mem} is met.

  Update the weights of the selected channels C_t using D_{train}.

Evaluate the fine-tuned backbone using D_{test}

data. This pre-computation is justified by the stability of layer rankings across tasks, requiring only a single mock training session of a few epochs on any relevant downstream task to establish the layer importance hierarchy.
The online phase proceeds through three key stages. First, channels are initially selected uniformly at random within the pre-selected layer subset to respect memory budget constraints. Second, the probability distribution is instantiated by assigning gradient norm-proportional probabilities to previously selected channels while setting unobserved channels to the maximum observed probability to encourage exploration. Finally, the algorithm enters an iterative loop alternating between probabilistic channel selection based on the current distribution and probability distribution updates using newly computed gradient norms from the selected channels.

3.5 Convergence and Complexity Analysis

Here we analyze the number of epochs required for our MeDyate algorithm to explore the search space and provide an analysis of the computational overhead of our method.

MeDyate Convergence. Due to our assignment of maximum probability to unseen channels, we can assert that our algorithm exhibits superior search space exploration compared to a fully random approach (i.e., where every channel receives equal selection probability). We therefore propose to analyze this random approach as a worst-case scenario bound for MeDyate convergence regarding exploration potential.
At each epoch, channels are selected according to a uniform distribution such that the memory budget B_{mem} is satisfied. By design, this memory budget represents a proportion α_K of the total memory available within the search space. If we denote u_e as the proportion of unexplored search space at epoch e, we naturally obtain $u_e = (1 - \alpha_K)^e$. This expression clearly highlights the interdependence between the number of selected layers and the memory budget as key factors determining our algorithm's capacity to explore the available search space.

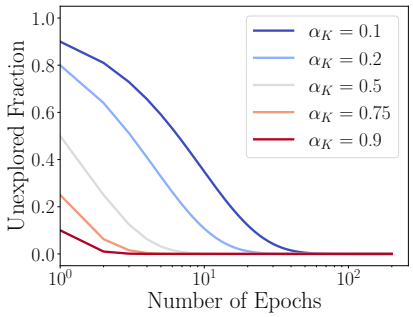


Figure 2: Evolution of u_e as a function of epochs for different α values.

Fig. 2 illustrates the evolution of the unexplored fraction as a function of epochs for different values of α. In practice, we have $\alpha_K \simeq \alpha_{\text{opt}}$, and our hyperparameter search detailed in Sec. 4.2 yields $\alpha_{\text{opt}} = 0.2$. Under random search conditions, this implies that 80% of channels are observed within 8 epochs, 90% within 11 epochs, and 95% within 14 epochs. As previously stated, these values represent upper bounds for exploration time, since we expect MeDyate to surpass random search performance. These rapid convergence rates are crucial, as they demonstrate that within just

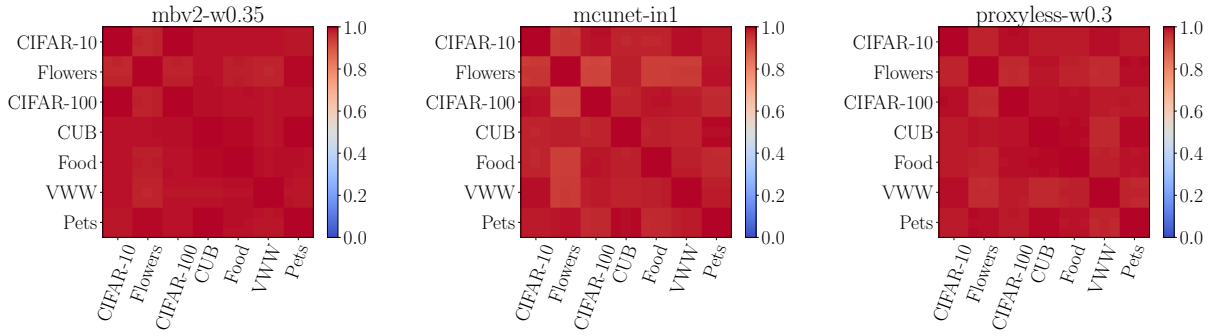


Figure 3: Spearman correlation of layer cumulated LaRa metric across seeds and datasets.

a few epochs, MeDyate can construct a comprehensive gradient representation, enabling it to achieve performance comparable to methods with full gradient knowledge.

MeDyate Computational Overhead. Compared to Lin *et al.*'s SU approach, which adds no on-device computational overhead since update schemes are pre-computed, or Quélennec *et al.*'s TraDy method, which performs random channel selection within predefined layer subsets, our MeDyate algorithm introduces more substantial computational overhead due to the necessity of computing gradient norms between epochs. We analyze this added complexity below. For a given epoch e, let ζ_e denote the set of channels selected for update and $|\zeta_e|$ its cardinality. Assuming uniform channel dimensions for simplicity, the computational complexity $(\Theta_{\text{time}})_e^{RN}$ of computing norm metrics for selected channels between epochs is:

$$(\Theta_{\text{time}})_e^{RN} = |\zeta_e| C' D^2. \tag{11}$$

The complexity of sampling one channel within a probability distribution of length N is $\mathcal{O}(\log(N))$, making the complexity of sampling $|\zeta_{e+1}|$ channels $\mathcal{O}(|\zeta_{e+1}| \log(N))$. Since $|\zeta_{e+1}| \simeq |\zeta_e|$ in practice, the total computational complexity of MeDyate for epoch e, denoted $(\mathcal{O}_{\text{time}})_e^{\text{Med}}$, becomes:

$$(\mathcal{O}_{\text{time}})_e^{\text{Med}} = |\zeta_e|(C' D^2 + \log(N)). \tag{12}$$

To contextualize this overhead, we compare it to the computational cost of computing weight derivatives for selected channels during backpropagation. Let M denote the number of steps per epoch. The computational complexity during epoch e, $(\Theta_{\text{time}})_e^{\text{grad}}$, is:

$$(\Theta_{\text{time}})_e^{\text{grad}} = M|\zeta_e| D^2 C' H' W'. \tag{13}$$

Furthermore, forward propagation and loss backpropagation exhibit similar computational complexity and involve the entire network rather than just the updated channel subset. Given the typical orders of magnitude of these variables, MeDyate's computational overhead remains negligible compared to the overall backpropagation cost. We anticipate a modest increase in inter-epoch latency, creating a trade-off between improved performance and computational overhead.

4 Experiments

4.1 Preamble

Experimental Setup. Our evaluation leverages three efficient architectures with ImageNet [8] pre-training: MobileNetV2 [40], ProxylessNAS [4], and MCUNet [25], adopting the same models used by Lin *et al.* [26] and Quélennec *et al.* for consistency. All experiments are conducted on Nvidia Tesla V100 SXM2 hardware using PyTorch 2.0.0 implementation in Python. Across all experimental configurations, we maintain classifier layer training regardless of the specific parameter selection strategy employed. Additional evaluation on transformer-based architectures is presented in the appendix.

Evaluation Datasets. We assess our approach across seven downstream tasks: CIFAR-10 [19], CIFAR-100 [19], CUB [47], Flowers [33], Food [1], Pets [35], and VWW [6].[1] We reproduce Quélennec *et al.* training protocol, employing cosine annealing schedules with 5-epoch warm-up periods [14], spanning 200 epochs for smaller datasets

[1]Pets: https://creativecommons.org/licenses/by-sa/4.0/, CC BY-SA 4.0 license; ImageNet: https://image-net.org/download.php the ImageNet license; others are not listed.

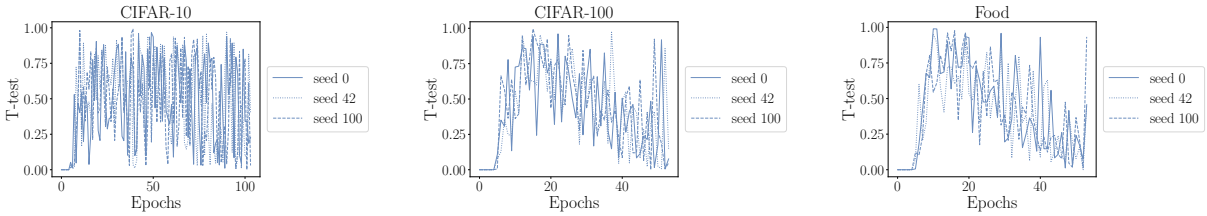


Figure 4: Evolution of channel gradient norm T-test over time, MobileNetV2 fine-tuned on 3 downstream tasks.

(CUB, Pets, Flowers), 100 epochs for CIFAR-100, and 50 epochs for larger datasets (CIFAR-100, Food, VWW). Learning rates decay from 0.125 to 0 without weight decay or dropout regularization. Statistical robustness is ensured through triple-seed evaluation, with reported metrics representing means and standard deviations across runs.

Memory Budget Configuration. To ensure fair comparative evaluation, we implement identical memory constraints to those established in Lin *et al.*'s SU framework and reproduced in Quélennec *et al.*'s TraDy framework. Our experimental design incorporates five graduated memory limitation tiers for each evaluated architecture, where each budget B_{mem} defines the total permissible memory allocation encompassing both parameter updates and activation storage requirements during training. This standardized constraint methodology enables direct performance comparisons while ensuring that observed improvements stem from algorithmic advances rather than relaxed memory limitations, thereby providing rigorous validation of our theoretical contributions under equivalent resource restrictions.

4.2 Preliminary Experiments

LaRa-based layer ranking consistency. We begin by validating that our proposed LaRa metric introduced in Sec. 3.2 preserves the architectural consistency of layer importance across downstream tasks. We define layer ranking as the ordered sequence of layers based on their LaRa values, computed according to equation 9. For each fine-tuning configuration, we accumulate the LaRa values of each layer over training epochs, forming a "layer topology" vector that characterizes the relative importance distribution across the network architecture. To assess the stability of these rankings across tasks, we calculate Spearman rank correlation coefficients between all pairwise combinations of fine-tuning experiments across our seven evaluation datasets, with three random initializations per dataset. This analysis produces a 21×21 correlation matrix for each architecture, illustrated in Fig. 3.

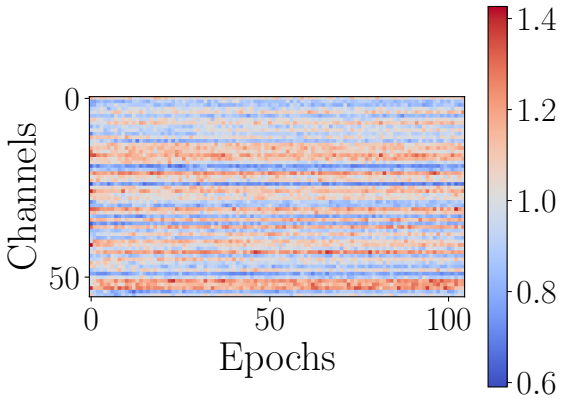
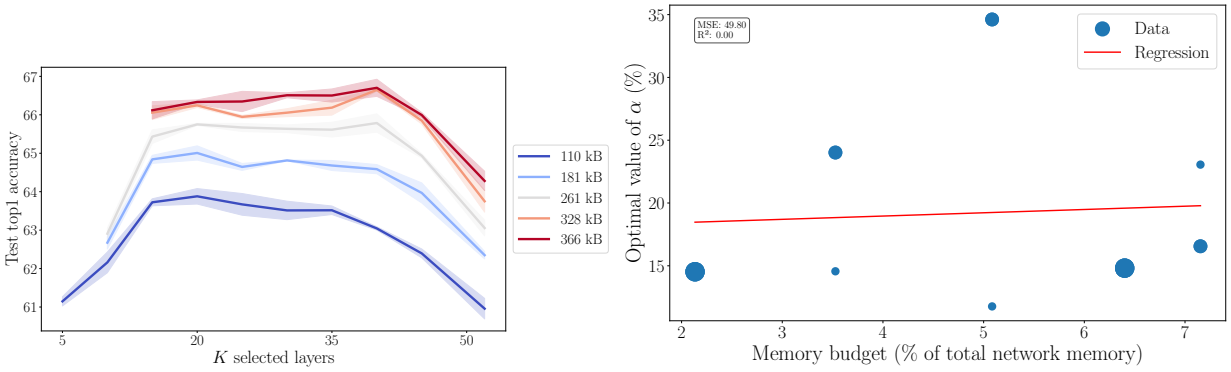


Figure 5: Evolution of channel gradient norm over time within a specific layer of a MobileNetV2 fine-tuned on CIFAR-10. Results are normalized per epoch for visualization.

The results confirm that our LaRa metric preserves the architectural consistency property established in Quélennec *et al.*'s work. Across all three network architectures, correlation coefficients between downstream task pairs consistently exceed 0.8, with the majority surpassing 0.9. These findings indicate that our LaRa metric maintains the fundamental property that layer importance rankings remain largely consistent across diverse downstream tasks, despite differences in data characteristics and task objectives. This consistency validates the key insight from Quélennec *et al.* that layer ranking can be performed *a priori* by conducting a preliminary fine-tuning run on available downstream task data (different than the target downstream task) and collecting the corresponding LaRa metrics. Similar patterns are observed for transformer architectures, as detailed in the appendix.

Channel Gradient Norm Stability. Building on the theoretical analysis presented in Sec. 3.3, we empirically examine the temporal evolution of channel topology during transfer learning across different downstream tasks. Fig. 4 illustrates this evolution by presenting T-test results comparing channel gradient norms between consecutive epochs to assess distributional similarity over time. The results reveal a clear two-phase pattern: during initial epochs, p-values are consistently equal to zero, indicating significant changes in gradient norm distributions as the network adapts to the



(a) Final MeDyate test top1 accuracies depending on the number of top K layers for different memory budgets.

(b) Regression on the optimal α_{opt} obtained for different memory budgets and seeds.

Figure 6: Study of the relationship between α_{opt} and memory budget. MobileNetV2 fine-tuned on Food.

new task. Following this initial adaptation period, p-values become substantially higher, failing to reject the topology similarity hypothesis and confirming distributional stabilization. This pattern demonstrates that after the stabilization phase, channels maintaining high gradient norms relative to others preserve this ranking consistently, while channels with comparatively low norms similarly maintain their relative positions.

To further illustrate the temporal consistency of relative channel importance, Fig. 5 depicts the evolution of channel gradient norms within a specific layer during MobileNetV2 fine-tuning on CIFAR-10, with norms normalized per epoch for visualization clarity. The figure reveals distinct trajectories of consistently high and low gradient norm channels across epochs, forming stable "beams" that persist throughout training. This visualization confirms that the relative gradient norm rankings of channels remain stable over time, supporting the theoretical foundations established in Sec. 3.3.

Defining α_{opt}. We investigate the relationship between memory budget constraints and the optimal α_K value that maximizes MeDyate performance. Fig. 6a presents the evolution of final test top-1 accuracy as a function of selected layer count K when fine-tuning MobileNetV2 on the Food dataset across different memory budgets. The curves demonstrate that peak accuracy positions shift rightward as budget increases, confirming our hypothesis that optimal search space size correlates with memory constraint as presented in Sec. 3.4. Furthermore, performance typically degrades when layer selection is either too restrictive or too permissive: insufficient layers exclude potentially crucial parameters, while excessive layer inclusion dilutes the memory budget and impairs MeDyate's convergence capacity.

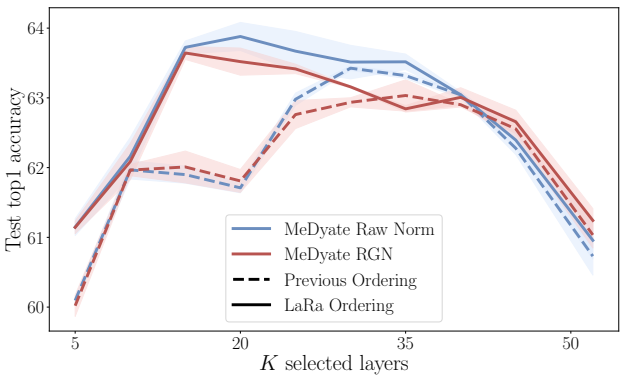


Figure 7: Performance comparison of layer ranking methods (LaRa vs. previous TraDy ordering) and channel importance metrics (raw gradient norm vs. RGN) when fine-tuning MobileNetV2 on Food dataset under memory constraints. Results show test top-1 accuracy across different numbers of selected layers K.

To quantify the relationship between B_{mem} and α_{opt}, we extract the accuracy-maximizing layer count K for each seed-budget combination from Fig. 6a, then compute the corresponding α_K values using Eq. equation 10. Fig. 6b presents these α_K values as a scatter plot against memory budgets (expressed as percentages of total network memory), with point sizes indicating the frequency of each configuration. Although regression analysis yields high variance, Fig. 6a reveals that larger memory budgets achieve relatively stable peak accuracy across a moderate range of K values, providing flexibility in α_{opt} selection. Given the accuracy degradation observed with insufficient layer selection (corresponding to larger α_K values), we establish $\alpha_{\text{opt}} = 0.2$, positioned slightly above the regression trend. This choice accounts for the discrete nature of layer selection, where we choose K such that $\alpha_K \leq \alpha_{\text{opt}} < \alpha_{K-1}$, ensuring adaptive layer selection that optimizes MeDyate performance across diverse memory constraints. Notably, while Quélennec *et al.*'s TraDy achieved peak accuracy with 35 layers under the smallest budget, our LaRa-based ranking reaches optimal performance with only 20 layers, demonstrating the effectiveness of our refined layer ranking methodology.

Validating LaRa and Raw Gradient Norm. We empirically validate the two key modifications proposed in our methodology compared to Quélennec *et al.*'s work: the LaRa layer ranking introduced in Sec. 3.2 and the raw gradient norm channel metric discussed in Sec. 3.4. Our evaluation involves fine-tuning MobileNetV2 on the Food dataset under the smallest memory budget constraint, comparing MeDyate performance across four configurations: our LaRa ranking versus the original TraDy layer ordering, each paired with either raw gradient norm or RGN as the channel importance metric.

Fig. 7 presents the final test top-1 accuracies for each configuration across varying numbers of selected layers K. The results demonstrate clear superiority of raw gradient norm over RGN across both layer ranking approaches, supporting our theoretical argument that memory cost reweighting becomes redundant when operating within pre-selected efficient layers. Furthermore, LaRa-based layer ranking consistently achieves higher peak accuracies than the previous ordering while requiring fewer layers to reach optimal performance. This dual advantage validates both the effectiveness of our holistic layer characterization approach and its practical benefits for memory-constrained sampling strategies.

4.3 Main Results

Experimental Design. We conduct a comprehensive evaluation of our MeDyate strategy across our complete experimental framework. Each channel selection approach is evaluated through 189 individual training runs, representing the full cross-product of three network architectures, seven downstream datasets, three memory budget levels, and three random seeds. Our statistical analysis employs paired T-tests to compare average final test top-1 accuracies across all experimental conditions, as visualized in Fig. 8. Each matrix cell represents a statistical hypothesis test examining whether the row strategy achieves superior mean accuracy compared to the column strategy. Complete numerical results and additional evaluations on transformer architectures are provided in the appendix (Sec. A).

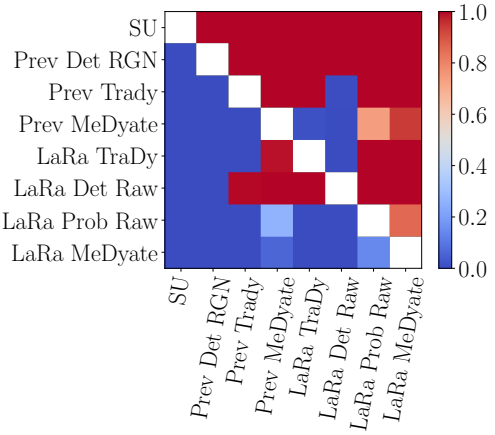


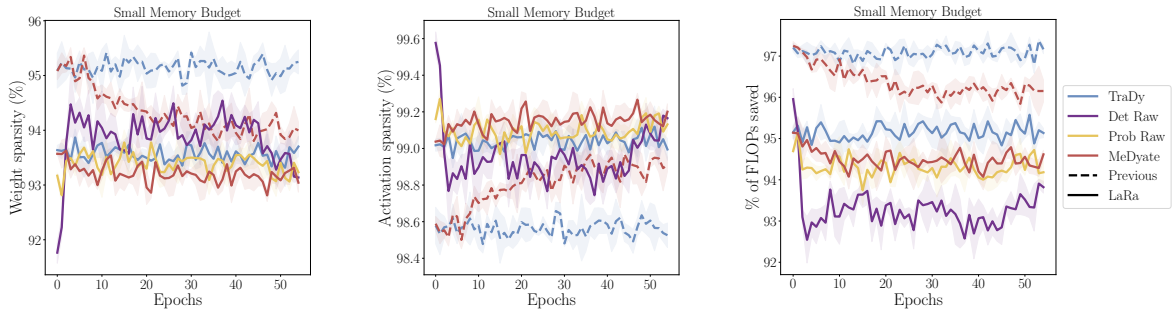
Figure 8: T-test comparisons of average final test accuracies across multiple experimental dimensions.

Evaluated Strategies. Our comparative analysis encompasses both static and dynamic selection approaches. We include Lin *et al.*'s SU method as the representative static baseline, while all other evaluated strategies employ dynamic channel resampling between epochs, consistent with the demonstrated superiority of dynamic approaches. Our nomenclature distinguishes between layer ranking methodologies: strategies prefixed with *Prev* utilize the original TraDy layer ordering with fixed layer counts, while *LaRa*-prefixed approaches employ our proposed layer ranking with budget-adaptive layer selection. We evaluate several key strategies from the TraDy framework, including deterministic RGN selection (*Det*), which leverages gradient pre-computation to select channels maximizing RGN within memory constraints, and the original TraDy approach for random channel sampling within pre-selected layers.

To isolate the contributions of our methodological components, we implement cross-combinations of layer ranking approaches with different channel selection strategies. This includes deterministic raw gradient norm selection within our LaRa framework (*Raw*), MeDyate applied with previous layer selection method, and TraDy adapted to our layer ranking system. Additionally, we evaluate a probabilistic gradient norm strategy (*Prob*) that pre-computes channel gradient norms and converts them to sampling probabilities, serving as a theoretical upper bound for MeDyate's performance by providing complete gradient knowledge during channel selection.

Discussion. The results presented in Fig. 8 support our methodological design choices. MeDyate achieves the highest performance across all strategies, with results that even seem to exceed those of the probabilistic gradient norm strategy with complete gradient information, indicating effective convergence despite memory constraints. The third-ranked performance of Prev MeDyate demonstrates that the core algorithmic principles remain effective even when combined with suboptimal layer ordering and selection, highlighting the robustness of the approach.

The strong performance of LaRa TraDy further validates our layer ranking methodology, showing that improved layer selection can enhance existing dynamic strategies. This approach presents a compelling alternative when computational overhead considerations favor simpler sampling schemes over MeDyate's gradient norm computation requirements. The consistent underperformance of raw gradient norm deterministic selection compared to its probabilistic counterparts reinforces the established advantage of stochastic sampling strategies in memory-constrained environments.



(a) Weight sparsity evolution during training.

(b) Activation sparsity evolution during training.

(c) Computational savings in weight derivative FLOPs.

Figure 9: Efficiency metrics comparison across channel selection strategies during MobileNetV2 fine-tuning on Food dataset under memory constraint. Results show evolution of sparsity levels and computational savings throughout training.

Efficiency Metrics. Our algorithm achieves these performance gains while simultaneously reducing computational overhead and maintaining high sparsity levels. Fig. 9 illustrates the evolution of key efficiency metrics when fine-tuning MobileNetV2 on the Food dataset under the smallest memory budget across representative channel selection strategies.

The sparsity analysis reveals interesting trade-offs between different approaches. While all methods achieve comparable overall sparsity levels (weight sparsity ranging from 92% to 96% and activation sparsity from 98.4% to 99.6%), LaRa MeDyate exhibits a distinct pattern of trading lower weight sparsity for higher activation sparsity. This behavior reflects the selection of channels with higher weight-to-activation memory ratios within the LaRa-selected layers. Conversely, TraDy with the previous layer policy demonstrates the opposite tendency, favoring channels with lower weight-to-activation memory ratios.

Regarding computational efficiency, Fig. 9c presents the percentage of weight derivative FLOPs saved through channel freezing during gradient computation (Eq. equation 2). Methods utilizing the previous layer policy achieve higher FLOP savings with typically one to two percentage points more saved FLOPs compared to their LaRa-based counterparts. This difference highlights a potential trade-off between accuracy optimization and computational efficiency, which may be relevant in scenarios where computational cost takes precedence over performance gains.

These results represent a single configuration (network, dataset, and memory budget) as metric behaviors remain consistent across experimental conditions. Complete training metrics for all configurations are provided in the supplementary materials.

5 Conclusion

In this paper, we present MeDyate, a theoretically-grounded framework for memory-constrained dynamic subnetwork adaptation that introduces the LaRa layer ranking metric and exploits channel importance stability during fine-tuning. Our extensive evaluation demonstrates consistent performance improvements over existing approaches while operating within memory budgets as low as a few hundred kB. However, a key limitation of our work is the absence of actual on-device implementation. While our algorithmic innovations show promise in experimental settings, practical deployment of dynamic channel selection on edge devices remains unvalidated, preventing us from obtaining real-world performance metrics. Future research should prioritize efficient on-device implementations to translate these theoretical advances into practical edge AI solutions.

References

[1] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*, 2014.

[2] Andrea Bragagnolo, Enzo Tartaglione, and Marco Grangetto. To update or not to update? neurons at equilibrium in deep models. *Advances in Neural Information Processing Systems*, 35, 2022.

[3] Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. Tinytl: Reduce memory, not parameters for efficient on-device learning. *Advances in Neural Information Processing Systems*, 33:11285–11297, 2020.

[4] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2019.

[5] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136, 2018.

[6] Aakanksha Chowdhery, Pete Warden, Jonathon Shlens, Andrew Howard, and Rocky Rhodes. Visual wake words dataset. *arXiv preprint arXiv:1906.05721*, 2019.

[7] Dorottya Demszky, Kelvin Guu, and Percy Liang. Transforming question answering datasets into natural language inference datasets. *arXiv preprint arXiv:1809.02922*, 2018.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 2009.

[9] Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4):485–532, 2020.

[10] Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8599–8603. IEEE, 2013.

[11] Sebastian Eliassen and Raghavendra Selvan. Activation compression of graph neural networks using block-wise quantization with improved variance minimization. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7430–7434, 2024.

[12] Shahaf E. Finder, Yair Zohav, Maor Ashkenazi, and Eran Treister. Wavelet feature maps compression for image-to-image cnns. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 20592–20606. Curran Associates, Inc., 2022.

[13] Georgios Georgiadis. Accelerating convolutional neural networks via activation map compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7085–7095, 2019.

[14] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[15] Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

[16] Ozlem Durmaz Incel and Sevda Ozge Bursa. On-device deep learning for mobile and wearable sensing applications: A review. *IEEE Sensors Journal*, 2023.

[17] Ziyu Jiang, Xuxi Chen, Xueqin Huang, Xianzhi Du, Denny Zhou, and Zhangyang Wang. Back razor: Memory-efficient transfer learning by self-sparsified backpropagation. *Advances in neural information processing systems*, 35:29248–29261, 2022.

[18] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

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

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[21] Mark Kurtz, Justin Kopinsky, Rati Gelashvili, Alexander Matveev, John Carr, Michael Goin, William Leiserson, Sage Moore, Nir Shavit, and Dan Alistarh. Inducing and exploiting activation sparsity for fast inference on deep neural networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5533–5543. PMLR, 13–18 Jul 2020.

[22] Young D. Kwon, Rui Li, Stylianos I. Venieris, Jagmohan Chauhan, Nicholas D. Lane, and Cecilia Mascolo. Tinytrain: Resource-aware task-adaptive sparse training of dnns at the data-scarce edge, 2024.

[23] Yoonho Lee, Annie S Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts. *arXiv preprint arXiv:2210.11466*, 2022.

[24] Ziyu Li, Enzo Tartaglione, and Van-Tam Nguyen. Scotti: Save computation at training time with an adaptive framework. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1443–1452, 2023.

[25] Ji Lin, Wei-Ming Chen, Yujun Lin, Chuang Gan, Song Han, et al. Mcunet: Tiny deep learning on iot devices. *Advances in Neural Information Processing Systems*, 33, 2020.

[26] 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, 2022.

[27] Yinhan Liu. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364, 2019.

[28] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[29] Bradley McDanel, Helia Dinh, and John Magallanes. Accelerating dnn training with structured data gradient pruning. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 2293–2299. IEEE, 2022.

[30] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3523–3542, 2021.

[31] Ali Bou Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, and Khaled Shaalan. Speech recognition using deep neural networks: A systematic review. *IEEE access*, 7:19143–19165, 2019.

[32] Le-Trung Nguyen, Aël Quélennec, Van-Tam Nguyen, and Enzo Tartaglione. Beyond low-rank decomposition: A shortcut approach for efficient on-device learning. *arXiv preprint arXiv:2505.05086*, 2025.

[33] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008.

[34] Zizheng Pan, Peng Chen, Haoyu He, Jing Liu, Jianfei Cai, and Bohan Zhuang. Mesa: A memory-saving training framework for transformers. *arXiv preprint arXiv:2111.11124*, 2021.

[35] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012.

[36] Danilo Pietro Pau and Fabrizio Maria Aymone. Suitability of forward-forward and pepita learning to mlcommons-tiny benchmarks. In *2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, 2023.

[37] Adam Poliak. A survey on recognizing textual entailment as an nlp evaluation. *arXiv preprint arXiv:2010.03061*, 2020.

[38] Aël Quélennec, Nour Hezbri, Pavlo Mozharovskyi, Van-Tam Nguyen, and Enzo Tartaglione. Study of training dynamics for memory-constrained fine-tuning, 2025.

[39] Berkman Sahiner, Weijie Chen, Ravi K Samala, and Nicholas Petrick. Data drift in medical machine learning: implications and potential remedies. *The British Journal of Radiology*, 96(1150):20220878, 2023.

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

[41] Jaime Sevilla, Lennart Heim, Anson Ho, Tamay Besiroglu, Marius Hobbhahn, and Pablo Villalobos. Compute trends across three eras of machine learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022.

[42] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[43] Umut Simsekli, Mert Gürbüzbalaban, Thanh Huy Nguyen, Gaël Richard, and Levent Sagun. On the heavy-tailed theory of stochastic gradient descent for deep neural networks. *arXiv preprint arXiv:1912.00018*, 222, 2019.

[44] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[45] Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*, 2019.

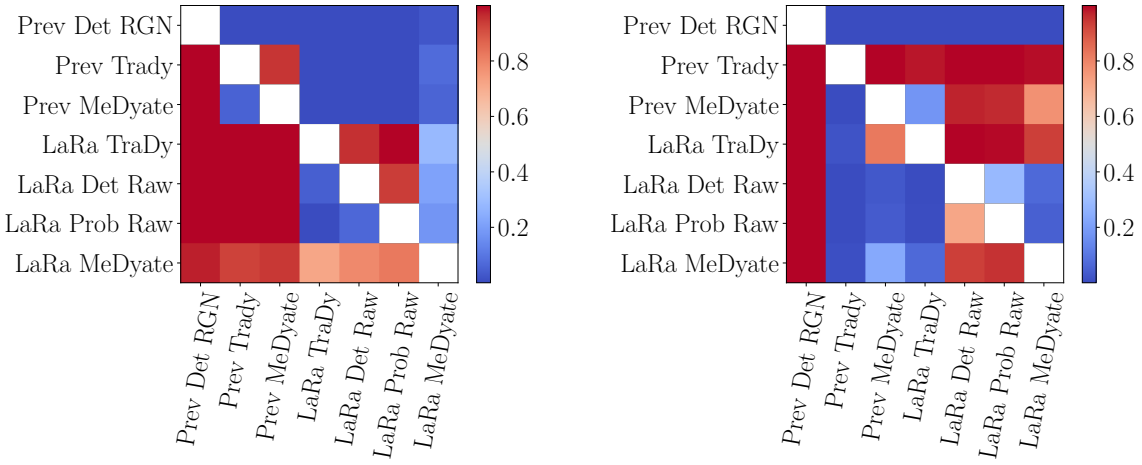
[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[47] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. Technical Report CNS-TR-201, Caltech, 2010.

[48] Junhuan Yang, Yi Sheng, Yuzhou Zhang, Weiwen Jiang, and Lei Yang. On-device unsupervised image segmentation. *arXiv preprint arXiv:2303.12753*, 2023.

[49] Xucheng Ye, Pengcheng Dai, Junyu Luo, Xin Guo, Yingjie Qi, Jianlei Yang, and Yiran Chen. Accelerating cnn training by pruning activation gradients. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16*, pages 322–338. Springer, 2020.

[50] Zhi Zhang, Qizhe Zhang, Zijun Gao, Renrui Zhang, Ekaterina Shutova, Shiji Zhou, and Shanghang Zhang. Gradient-based parameter selection for efficient fine-tuning, 2024.



(a) SwinT model

(b) BERTs models

Figure 10: T-test comparisons of average final test accuracies across multiple experimental dimensions for each group of transformer architectures.

A Appendix

A.1 Transformer Results

We extend our evaluation to transformer architectures, adopting the framework from [38]. Our experiments employ a SwinT model [28] pre-trained on ImageNet and evaluated on the seven downstream vision tasks, alongside BERT [18] and RoBERTa [27] for NLP tasks: QNLI [7], RTE [37], and SST2 [44]. Fig. 10 displays paired T-test comparisons of mean final test accuracies, mirroring the statistical methodology from Sec. 4.3. Complete numerical results appear in Sec. A.3.

Transformer architectures exhibit significantly different performance patterns compared to CNNs. The deterministic RGN-based channel selection within fixed layer subsets emerges as the dominant strategy across both architecture families, contrasting sharply with CNN results where MeDyate demonstrated clear superiority.

The SwinT results (Fig. 10a) presents an inversion of training dynamics: LaRa-based MeDyate records the lowest performance across all strategies, while the fixed RGN-ranked approach achieves second place and represents the best deployable solution. This inversion indicates that LaRa's design principles, effective for convolutional architectures, introduce counterproductive biases in vision transformers. The adaptive layer selection advantageous for CNNs appears to conflict with the operational characteristics of attention-based vision models.

BERTs architectures (Fig. 10b) show more nuanced behavior. While deterministic RGN selection maintains its lead, performance differences between strategies diminish substantially. LaRa demonstrates acceptable effectiveness in the NLP domain, with deterministic and stochastic raw norm variants achieving competitive accuracy. These findings suggest LaRa's applicability varies with task modality: reasonable for language tasks yet problematic for vision transformers.

This architectural divergence reveals fundamental distinctions in fine-tuning behavior requiring systematic analysis. We propose two primary explanatory factors. The attention mechanism's global receptive fields may fundamentally alter gradient propagation patterns relative to the local connectivity assumptions underlying LaRa's formulation for CNNs. Additionally, vision and language transformers process fundamentally different information structures: spatial hierarchies through attention versus sequential token dependencies, potentially necessitating domain-specific layer importance characterization approaches.

Future research should explore architecture-aware extensions of the LaRa framework, potentially incorporating attention-specific metrics or developing task-conditional layer ranking strategies tailored to the unique computational patterns of transformer models.

A.2 Inconclusive Strategy: Weighted Sampling

Both Quélennec *et al.*'s TRaDy and our proposed MeDyate (Sec. 3.4) employ uniform probability distributions for channel sampling, either throughout the entire fine-tuning process or at initialization. This approach implicitly assigns equal importance to all channels within the selected layer subset. Given our access to pre-computed layer importance

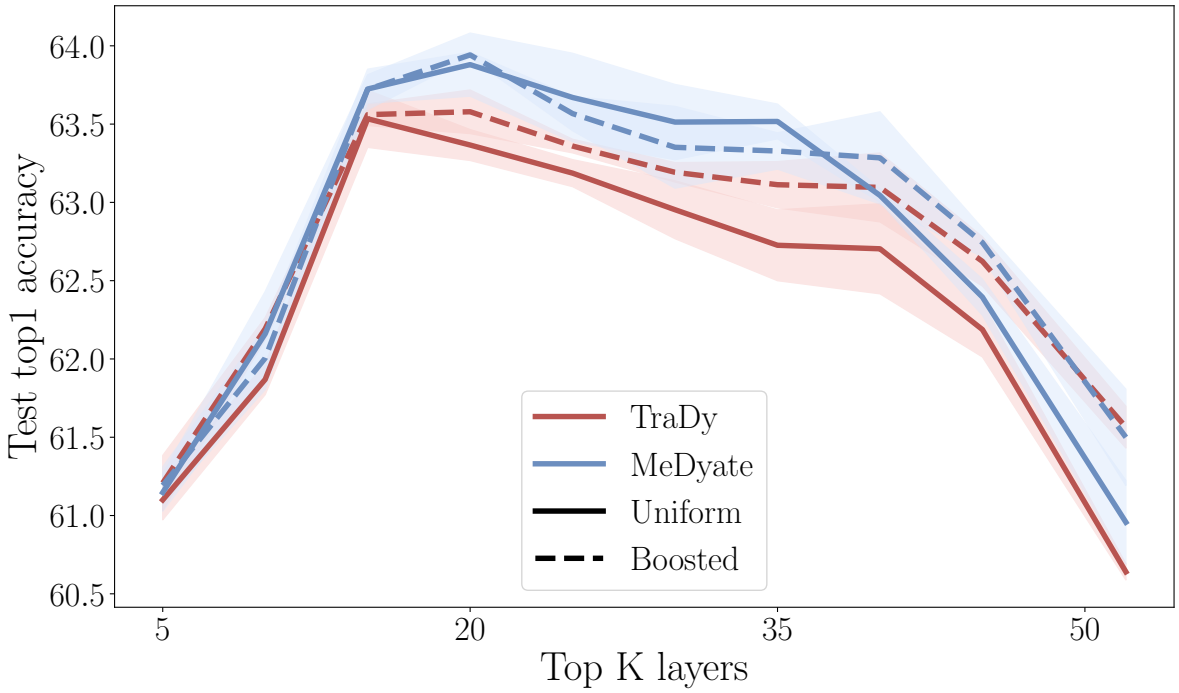


Figure 11: Effects of probability distribution in probabilistic sampling approaches with respect to the number of layers selected when fine-tuning MobileNetV2 on Food dataset under memory constraints. Results show test top-1 accuracy across different numbers of selected layers K.

through the LaRa metric (Sec. 3.2), we investigated replacing uniform sampling with a LaRa-weighted distribution, where channel selection probabilities are proportional to their corresponding layer's LaRa value. We hypothesized that this *boosting* strategy would enable more informed channel selection in TRaDy and accelerate convergence toward the optimal channel distribution in MeDyate.
Fig. 11 illustrates the evolution of final test top-1 accuracy across different numbers of selected LaRa-ranked layers K, comparing uniform and boosted variants of both TRaDy and MeDyate. The boosting mechanism primarily influences performance at larger K values, where it mitigates accuracy degradation. However, within the optimal K range that maximizes accuracy, boosting yields only marginal improvements for TRaDy and no statistically significant effect for MeDyate. Given these limited gains, we defer comprehensive exploration of layer-importance-weighted sampling to future work, focusing instead on the more impactful contributions presented in the main paper.

A.3 Full Results Tables

This section presents experimental results for the selection strategies evaluated in our study. Results from strategies presented in Quélennec *et al.*'s work are omitted since we use the same framework to run experiments thus obtaining the same results [38]. Tab. 1 displays results for CNN architectures, while Tab. 2 and Tab. 3 respectively presents results with SwinT and NLP models.

Table 1: Comparison of final top1 test accuracies between channel selection strategies over various pretrained convolutional architectures, datasets, and budgets.

Model	B_{mem}	Method	CIFAR-10	CIFAR-100	CUB	Flowers	Food	Pets	VWW	Average
MbV2-w0.35	27 946	Prev MeDyate	90.18±0.13	68.84±0.20	58.19±0.64	80.07±0.42	63.32±0.04	77.02±0.26	88.66±0.11	75.18±0.85
		LaRa TraDy	89.97±0.06	68.67±0.23	58.67±0.26	80.34±0.60	63.60±0.12	77.72±0.24	88.48±0.25	75.35±0.79
		LaRa Det Raw	89.89±0.03	68.17±0.20	58.0±0.19	80.65±0.35	62.61±0.01	77.60±0.45	88.10±0.29	75.00±0.70
		LaRa Prob Raw	90.13±0.17	68.75±0.23	58.20±0.36	80.25±0.42	63.93±0.07	77.28±0.20	88.58±0.31	75.30±0.73
		LaRa MeDyate	90.24±0.11	68.83±0.13	58.73±0.12	80.31±0.15	63.7±0.16	77.45±0.25	88.53±0.11	75.40±0.41
	66 592	Prev MeDyate	90.70±0.05	69.66±0.17	58.96±0.33	80.80±0.34	65.78±0.11	77.16±0.70	87.65±0.14	75.82±0.88
		LaRa TraDy	90.82±0.18	69.65±0.16	58.50±0.23	80.27±0.29	65.08±0.22	76.90±0.20	88.06±0.22	75.61±0.58
		LaRa Det Raw	90.57±0.12	69.22±0.16	58.62±0.08	81.10±0.62	64.58±0.26	77.24±0.38	87.98±0.05	75.62±0.80
		LaRa Prob Raw	91.02±0.09	69.59±0.28	58.87±0.24	80.68±0.51	65.38±0.25	77.00±0.59	87.72±0.24	75.75±0.93
		LaRa MeDyate	91.02±0.18	69.70±0.31	58.90±0.10	80.71±0.59	65.37±0.08	76.92±0.10	88.01±0.29	75.80±0.77
	93 696	Prev MeDyate	91.27±0.11	70.23±0.31	58.86±0.54	81.27±0.43	66.72±0.19	77.11±0.33	88.02±0.18	76.21±0.87
		LaRa TraDy	91.37±0.06	70.09±0.21	59.03±0.32	80.66±0.32	66.39±0.06	76.86±0.24	87.96±0.21	76.05±0.60
		LaRa Det Raw	90.91±0.04	69.73±0.08	58.88±0.24	81.48±0.35	65.53±0.12	77.24±0.46	87.93±0.26	75.96±0.69
		LaRa Prob Raw	91.44±0.10	70.23±0.15	59.45±0.03	81.05±0.39	66.83±0.04	77.13±0.14	87.87±0.19	76.29±0.49
		LaRa MeDyate	91.35±0.15	70.21±0.05	59.08±0.36	80.94±0.40	66.99±0.14	77.04±0.08	87.9±0.36	76.22±0.69
	1 252 320	Baseline	92.72±0.03	72.69±0.16	60.03±0.18	81.88±0.34	70.79±0.20	76.68±0.33	88.58±0.19	77.62±0.60
MCUNet-in1	15 936	Prev MeDyate	90.63±0.10	69.94±0.12	61.70±0.05	82.51±0.22	66.94±0.23	81.21±0.04	89.69±0.33	77.52±0.49
		LaRa TraDy	90.89±0.06	69.88±0.13	62.23±0.07	83.00±0.41	67.34±0.15	81.62±0.24	89.72±0.24	77.81±0.58
		LaRa Det Raw	90.28±0.14	69.66±0.28	61.83±0.63	82.65±0.40	66.29±0.19	80.85±0.34	89.67±0.15	77.32±0.91
		LaRa Prob Raw	90.91±0.18	69.74±0.20	61.75±0.39	82.61±0.19	67.33±0.26	80.98±0.26	89.55±0.21	77.55±0.66
		LaRa MeDyate	90.89±0.10	69.89±0.05	62.39±0.41	82.76±0.43	67.39±0.12	81.37±0.24	89.94±0.04	77.80±0.66
	64 832	Prev MeDyate	92.33±0.12	72.41±0.25	62.40±0.50	83.29±0.63	71.20±0.28	81.44±0.37	89.22±0.31	78.90±1.02
		LaRa TraDy	92.09±0.02	72.24±0.38	62.60±0.34	83.07±0.53	70.44±0.03	81.46±0.44	89.09±0.24	78.71±0.89
		LaRa Det Raw	91.94±0.30	71.57±0.11	62.29±0.50	82.71±0.40	69.98±0.16	80.96±0.09	89.19±0.15	78.38±0.75
		LaRa Prob Raw	92.14±0.10	72.26±0.18	62.51±0.29	82.93±0.42	70.79±0.10	81.59±0.45	89.32±0.29	78.79±0.77
		LaRa MeDyate	92.23±0.18	72.00±0.13	62.44±0.09	83.24±0.29	71.00±0.19	81.20±0.03	89.18±0.23	78.76±0.48
	112 640	Prev MeDyate	92.78±0.09	72.88±0.36	61.83±0.49	83.48±0.44	72.42±0.29	81.16±0.48	89.14±0.49	79.10±1.06
		LaRa TraDy	92.78±0.10	73.71±0.03	61.85±0.54	83.24±0.26	72.20±0.14	81.17±0.55	89.14±0.09	79.16±0.84
		LaRa Det Raw	92.51±0.19	72.88±0.20	62.14±0.12	83.16±0.34	71.68±0.11	80.78±0.28	89.59±0.29	78.96±0.62
		LaRa Prob Raw	92.78±0.23	73.46±0.21	62.06±0.30	82.96±0.23	72.55±0.11	81.15±0.19	89.50±0.06	79.21±0.54
		LaRa MeDyate	92.72±0.20	73.55±0.28	62.36±0.34	83.46±0.42	72.73±0.02	80.83±0.50	89.51±0.13	79.31±0.82
	1 309 808	Baseline	93.87±0.10	76.03±0.18	61.62±0.62	83.45±0.42	75.74±0.14	79.49±0.60	90.06±0.16	80.04±1.00
Proxyless-w0.3	25 984	Prev MeDyate	91.43±0.20	70.30±0.08	57.74±0.38	82.15±0.22	64.80±0.24	78.77±0.05	88.57±0.13	76.25±0.56
		LaRa TraDy	91.46±0.17	69.38±0.15	58.18±0.30	81.76±0.25	64.58±0.05	79.10±0.22	88.57±0.05	76.15±0.51
		LaRa Det Raw	91.26±0.11	68.59±0.11	57.87±0.10	82.11±0.31	63.40±0.07	78.91±0.16	88.65±0.18	75.83±0.44
		LaRa Prob Raw	91.53±0.24	69.39±0.12	58.14±0.28	81.89±0.40	64.66±0.24	79.27±0.52	88.74±0.17	76.23±0.82
		LaRa MeDyate	91.38±0.21	69.71±0.10	58.27±0.06	81.96±0.46	64.75±0.27	78.92±0.24	88.77±0.20	76.25±0.66
	72 960	Prev MeDyate	92.27±0.13	71.76±0.13	59.06±0.29	82.59±0.20	67.81±0.08	79.28±0.35	88.46±0.02	77.32±0.54
		LaRa TraDy	92.29±0.24	71.26±0.17	58.73±0.24	82.50±0.33	67.14±0.13	79.27±0.22	88.44±0.26	77.09±0.62
		LaRa Det Raw	92.32±0.07	71.32±0.03	58.87±0.20	83.09±0.28	67.20±0.35	78.49±0.20	88.02±0.24	77.04±0.59
		LaRa Prob Raw	92.48±0.04	71.83±0.33	59.75±0.67	82.89±0.46	67.85±0.19	79.31±0.47	88.02±0.18	77.45±1.03
		LaRa MeDyate	92.50±0.15	71.54±0.17	59.25±0.79	82.90±0.12	67.66±0.17	79.05±0.32	88.05±0.14	77.28±0.92
	101 376	Prev MeDyate	92.65±0.25	72.28±0.26	59.58±0.20	83.04±0.30	68.77±0.21	79.23±0.22	88.35±0.17	77.70±0.62
		LaRa TraDy	92.56±0.09	71.76±0.29	59.58±0.48	82.6±0.20	67.79±0.16	79.05±0.18	88.3±0.20	77.38±0.68
		LaRa Det Raw	92.37±0.19	71.36±0.31	59.34±0.61	83.25±0.46	67.35±0.07	79.00±0.45	88.30±0.27	77.28±1.00
		LaRa Prob Raw	92.61±0.15	71.62±0.12	59.91±0.26	82.99±0.28	68.32±0.14	79.31±0.36	88.47±0.45	77.60±0.73
		LaRa MeDyate	92.42±0.02	71.86±0.09	59.73±0.28	82.9±0.23	68.52±0.22	79.45±0.50	88.23±0.14	77.59±0.68
	1 162 032	Baseline	93.71±0.12	74.81±0.13	61.75±0.12	84.44±0.50	72.98±0.09	78.53±0.10	88.95±0.04	79.31±0.56

Table 2: Comparison of final top1 test accuracies between channel selection strategies over various datasets, and budgets when considering a SwinT architecture

Model	B_{mem}	Method	CIFAR-10	CIFAR-100	CUB	Flowers	Food	Pets	VWW	Average
SwinT	27 946	Prev MeDyate	96.35±0.11	82.91±0.10	73.98±0.06	88.44±0.34	80.74±0.04	90.97±0.20	93.75±0.15	86.73±0.45
		LaRa TraDy	96.30±0.07	83.16±0.17	74.31±0.23	88.59±0.24	80.78±0.05	90.91±0.12	92.77±0.14	86.69±0.43
		LaRa Det Raw	96.34±0.06	82.96±0.10	74.42±0.10	88.21±0.14	80.76±0.11	90.98±0.11	92.93±0.14	86.66±0.29
		LaRa Prob Raw	96.33±0.04	83.14±0.18	74.35±0.02	88.56±0.34	80.90±0.17	90.87±0.29	92.81±0.06	86.71±0.52
		LaRa MeDyate	96.29±0.09	83.10±0.06	74.34±0.01	88.63±0.51	80.97±0.05	90.99±0.19	92.76±0.07	86.73±0.56
	112 640	Prev MeDyate	96.76±0.05	83.58±0.16	74.43±0.15	88.85±0.23	81.58±0.04	90.99±0.19	93.55±0.38	87.11±0.53
		LaRa TraDy	96.94±0.14	83.88±0.17	73.24±0.20	85.60±0.64	81.85±0.07	90.61±0.26	92.75±0.11	86.41±0.76
		LaRa Det Raw	96.69±0.12	83.49±0.12	73.69±0.32	86.46±0.15	81.30±0.10	91.12±0.27	93.19±0.14	86.56±0.51
		LaRa Prob Raw	96.79±0.15	83.71±0.13	73.89±0.06	86.15±0.28	81.59±0.16	91.15±0.19	93.04±0.15	86.62±0.45
		LaRa MeDyate	96.87±0.03	83.82±0.15	73.85±0.17	86.52±0.33	81.79±0.08	91.01±0.12	93.01±0.09	86.70±0.44
	633 859	Prev MeDyate	97.32±0.03	84.72±0.18	75.14±0.57	89.79±0.36	83.51±0.11	91.02±0.13	93.33±0.27	87.83±0.77
		LaRa TraDy	97.31±0.04	84.90±0.08	74.36±0.27	86.56±0.89	83.63±0.12	91.08±0.17	92.72±0.13	87.22±0.97
		LaRa Det Raw	97.19±0.03	84.55±0.13	74.25±0.15	87.67±0.54	83.20±0.03	91.13±0.27	92.62±0.28	87.23±0.70
		LaRa Prob Raw	97.32±0.03	84.71±0.09	74.30±0.08	87.53±0.19	83.69±0.05	91.18±0.15	92.75±0.34	87.35±0.44
		LaRa MeDyate	97.25±0.12	85.01±0.10	74.67±0.55	87.56±0.31	83.66±0.13	91.21±0.06	92.72±0.08	87.44±0.67
	2 767 686	Prev MeDyate	97.70±0.08	85.91±0.13	76.13±0.29	90.73±0.34	84.87±0.08	91.38±0.33	93.77±0.12	88.64±0.59
		LaRa TraDy	97.52±0.07	85.81±0.12	75.22±0.06	88.21±0.46	84.70±0.02	91.23±0.18	93.33±0.07	88.00±0.52
		LaRa Det Raw	97.54±0.02	85.69±0.21	75.42±0.57	90.26±0.22	84.83±0.06	91.04±0.37	93.52±0.22	88.33±0.78
		LaRa Prob Raw	97.71±0.05	85.79±0.30	75.78±0.58	89.48±0.41	84.88±0.10	91.12±0.65	93.57±0.05	88.33±1.02
		LaRa MeDyate	97.67±0.13	85.82±0.18	75.71±0.10	89.24±0.46	84.84±0.12	91.18±0.22	79.9±23.43	86.34±23.44
	31 889 952	Baseline	97.78±0.16	86.30±0.05	74.89±0.20	90.57±0.43	86.07±0.23	90.18±0.60	93.72±0.10	88.50±0.31

Table 3: Comparison of final top1 test accuracies between channel selection strategies with pretrained BERT and RoBERTa models, fine-tuned on various datasets and budgets.

Model	B_{mem}	Method	QNLI	RTE	SST2	Average
BERT	27 946	Prev MeDyate	84.38±0.06	58.24±1.82	89.37±0.13	77.33±1.83
		LaRa TraDy	84.92±0.05	57.76±2.60	89.53±0.13	77.40±2.60
		LaRa Det Raw	84.75±0.13	56.92±0.75	88.91±0.35	76.86±0.84
		LaRa Prob Raw	84.62±0.30	57.04±1.30	88.91±0.18	76.86±1.35
		LaRa MeDyate	84.58±0.07	58.48±1.65	89.18±0.24	77.41±1.67
	112 640	Prev MeDyate	84.51±0.25	58.48±0.63	89.53±0.33	77.51±0.75
		LaRa TraDy	85.16±0.19	59.09±2.18	89.68±0.11	77.98±2.19
		LaRa Det Raw	85.69±0.04	58.24±1.16	89.11±0.46	77.68±1.25
		LaRa Prob Raw	85.28±0.11	57.52±0.55	89.56±0.00	77.45±0.56
		LaRa MeDyate	85.19±0.21	56.56±0.21	89.41±0.37	77.05±0.47
	1 912 629	Prev MeDyate	86.78±0.43	55.72±1.63	90.37±0.34	77.62±1.72
		LaRa TraDy	87.43±0.30	56.68±0.36	90.56±0.48	78.22±0.67
		LaRa Det Raw	88.36±0.16	58.12±1.65	90.83±0.23	79.10±1.67
		LaRa Prob Raw	88.31±0.10	58.12±2.25	90.63±0.66	79.02±2.35
		LaRa MeDyate	87.79±0.10	55.84±1.04	90.44±0.76	78.02±1.29
	8 351 308	Prev MeDyate	89.05±0.14	61.13±1.99	91.21±0.66	80.46±2.10
		LaRa TraDy	89.13±0.32	57.76±1.88	90.75±0.18	79.21±1.92
		LaRa Det Raw	89.96±0.12	63.06±0.83	91.74±0.72	81.59±1.11
		LaRa Prob Raw	89.69±0.14	61.01±0.96	91.48±0.13	80.73±0.98
		LaRa MeDyate	89.95±0.18	61.49±1.71	90.90±0.53	80.78±1.80
	96 225 792	Baseline	90.81±0.27	62.45±1.81	91.74±0.50	81.67±1.90
RoBERTa	27 946	Prev MeDyate	89.69±0.06	57.04±0.72	93.31±0.07	80.01±0.73
		LaRa TraDy	89.57±0.29	59.33±1.82	93.35±0.11	80.75±1.85
		LaRa Det Raw	89.02±1.15	68.23±0.36	93.00±0.30	83.42±1.24
		LaRa Prob Raw	89.42±0.22	66.91±3.47	93.08±0.54	83.14±3.52
		LaRa MeDyate	89.81±0.27	63.18±4.72	93.31±0.18	82.10±4.73
	112 640	Prev MeDyate	90.05±0.09	60.41±2.29	93.39±0.18	81.28±2.30
		LaRa TraDy	89.65±0.53	62.33±2.05	93.46±0.40	81.81±2.15
		LaRa Det Raw	89.62±0.10	66.55±2.21	92.85±0.07	83.01±2.21
		LaRa Prob Raw	89.66±0.15	66.55±2.35	92.89±0.34	83.03±2.38
		LaRa MeDyate	89.77±0.33	67.75±3.28	93.27±0.13	83.60±3.30
	1 912 629	Prev MeDyate	91.40±0.06	75.45±0.72	93.92±0.53	86.92±0.90
		LaRa TraDy	90.82±0.25	69.68±0.96	93.16±0.33	84.55±1.05
		LaRa Det Raw	90.85±0.11	73.04±5.01	92.09±0.11	85.33±5.01
		LaRa Prob Raw	90.87±0.19	75.21±1.46	92.39±0.46	86.16±1.54
		LaRa MeDyate	90.60±0.47	74.61±1.85	92.97±0.07	86.06±1.91
	8 351 308	Prev MeDyate	91.52±0.41	75.21±1.63	93.16±0.46	86.63±1.74
		LaRa TraDy	90.87±0.21	74.13±2.92	93.23±0.40	86.08±2.95
		LaRa Det Raw	91.21±0.25	74.73±1.08	92.66±0.61	86.20±1.27
		LaRa Prob Raw	90.67±0.24	74.37±0.72	92.93±0.07	85.99±0.76
		LaRa MeDyate	90.49±0.78	70.52±3.51	92.97±0.40	84.66±3.62
	96 225 792	Baseline	92.31±0.14	76.41±0.55	93.16±0.92	87.29±1.08