FLAME: FEEDFORWARD LATENT TASK INFERENCE FOR CONTINUAL META-LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual-meta learning promises rapid adaptation to new tasks without forgetting old ones. Despite significant progress, two major challenges remain. First, existing works mainly considered MAML as the meta-learner where adaptation is limited to gradient descent and meta-knowledge is encoded in initial model weights, creating bottlenecks as the task landscape becomes complex. Second, existing works largely operate in task-aware settings that rely on known task boundaries and identities to pair context/query samples, while the few task-agnostic variants forgo the bi-level meta-objective. We address these two challenges with FLAME, a *feedforward meta-model* with *latent task inference* for continual adaptation. FLAME uses a feedforward meta-learner to rapidly adapt either the classifier layer or, in a nested formulation, the feature extractor of a DNN. It then leverages the feedforward task embedding to detect task boundaries and estimate task identities, approximating bi-level meta-optimization in a *task-agnostic* fashion. The continually-consolidated task relations in the replay reservoir further enables recognition of known *vs*. novel tasks, allow continual improvements on recurring tasks over various lengths of local stationary windows. We demonstrated the improved performance of FLAME in comparison to both task-agnostic and task-aware baselines, in domain- and class-incremental learning across image and text datasets.

1 INTRODUCTION

Despite significant growth, deep learning continues to face two central challenges: the ability to learn similar tasks with only a few examples (Vinyals et al., 2017), and the ability to incrementally learn one task at a time without forgetting (Wickramasinghe et al., 2024). Meta-learning addresses the former challenge (Hospedales et al., 2022), yet its training assumes a stationary distribution of known tasks, preventing incremental learning over streaming tasks (Son et al., 2023). Continual learning addresses the latter challenge (Ven & Tolias, 2019), yet it typically optimizes a single model across data distributions, limiting adaptability to diverse tasks when few-shot examples are available. Continual meta-learning combines these two paradigms to meta-learn few-shot adaptation in streaming tasks (Son et al., 2023): while related works come under two umbrella terms of online or continual meta-learning (O/CML), their differences are often about data and use scenarios with mutually applicable methodologies. We thus consider both and the two central challenges they share.

First, most existing O/CML works build on continual extensions of MAML (model-agonostic meta-learning) (Finn et al., 2017) and its variants. Consider meta-learning as the extraction of *what-how* meta-knowledge for fast adaptation: *i.e., what* task few-shot examples represent and *how* to best adapt to that task. In MAML, the *how* (adaptation mechanism) is predetermined as iterative gradient-descent (GD) optimization, and the *what* (task recognition) is the GD-updated network parameters: the meta-knowledge learned is thus limited to a weight initialization, without explicit means to recognize tasks or task-relations, producing a bottleneck as the complexity of task landscape increases. Indeed, many existing O/CML approaches have attempted to resolve this challenge, mostly by learning a cluster of weight initializations at the weight space (Wu et al., 2023a; Zhang et al., 2021; Yao et al., 2019).

Second, because the signature *bi-level meta-objective* requires task identities to pair context and query samples, most O/CML works operate under a *task-aware* setting where both task-transition boundaries and task identities are known to the learner. In the few attempts of *task-agnostic* O/CML (neither task boundary nor identifier is known), this bi-level meta-objective is either only applied to

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

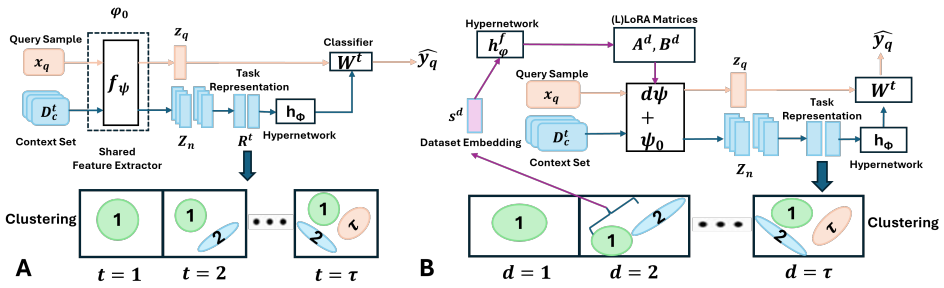

Figure 1: Overview of FLAME: A. Feedforward adaptation of DNN classifier layers, *e.g.*, in domain-incremental settings. B. Nesting classifier-adaptation in feedforward generation of (L)LoRA matrices to adapt DNN feature extractors, *e.g.,* in class-incremental settings.

the *streaming* task with forgetting mitigated by meta-model regularization (He et al., 2019; Harrison et al., 2020), or revised to *gradient alignment* such that the model adapted to the *streaming* task still applies to replayed *past* task samples (Riemer et al., 2019; Joseph & Gu, 2021). In other words, *task-agnosticism* is enabled by compromising the bi-level objective characteristic of meta-learning.

We address these two longstanding O/CML challenges with FLAME: a feedforward meta-learner with latent task inference for continual adaptation, with the key following contributions:

- For the *what*, FLAME learns feedforward extraction of latent task-embeddings from few-shot examples, with continual clustering at this latent space to consolidate task relations. This allows estimation of task boundaries and identities for *task-agnostic* bi-level meta-optimization. It also recognizes *known vs. novel* tasks to allow continual improvements on recurring tasks over local-stationary windows of various lengths.
- For the *how*, FLAME learns fast adaptation in the form of a hyper-network, offering more expressive and efficient adaptation mechanism compared to GD. We demonstrate this feedforward mechanism for either adapting only the classifier layer of a DNN (Fig. 1A, *e.g.,* in domain-incremental settings) or, in a nested fashion, simultaneously generating LoRA matrices for adapting the feature extractor with convolutional- and transformer-based backbones (Fig. 1B, *e.g.*, in class-incremental settings) across vision and text datasets.

We evaluated FLAME in both domain-incremental and class-incremental problem settings, streaming across nonstationary datasets where neither task boundaries nor identities were known. We included representative baselines from both O/CML literature, to separately delineate the benefit of the feedforward meta-learner (*vs*. MAML) and the bi-level meta-optimization enabled by task identification (*vs*. existing task-agnostic mechanisms). We further demonstrated benefits of task-relation consolidation in enabling continual improvements of *recurring* tasks, with extensive ablation over the length of local stationary windows, gradual transition boundaries, and task-inference hyperparameters.

2 Related Works

Table 6 of Appendix A presents a complete summary of related O/CML works based on several key axes of design choices including: task awareness, meta algorithmic prior, meta-optimization objective, and continual learning (CL) strategies. Below we summarize those most related to FLAME.

MAML-based O/CML approaches: The vast majority of existing continual O/CML methods build on MAML-style meta-learners as an algorithmic prior for fast adaptation. Many operate in a *task-aware* setting where both task boundaries and identities are known. These knowledge allow the models to either perform bi-level meta-objectives on all tasks leveraging a replay buffer of past tasks (Finn et al., 2019; Acar et al., 2021; Yao et al., 2019; 2020), or adapt the model to the streaming task while mitigating forgetting by model regularization (Yap et al., 2020; Zhang et al., 2021; Wu et al., 2023a). A notable focus in this line of models is to move beyond a single meta-initializatio by clustering in the DNN weight space (Yao et al., 2019; Zhang et al., 2021; Wu et al., 2023a) or constructing task-specific pathways within the DNN (Yao et al., 2020).

To operate in a *task-agnostic* fashion without task identities, many O/CML methods replace the bi-level meta-objective with an objective of gradient alignment (Joseph & Gu, 2021; Riemer et al., 2019), via per-parameter learning rate modulation (Joseph & Gu, 2021), Reptile (Nichol & Schulman, 2018) style first order meta update (Riemer et al., 2019), or regularized inner-adaptation in MAML (Rajasegaran et al., 2022). While enabling task-agnostic O/CML, maintaining non-conflicting updates may become difficult in complex task landscapes. To date, only a small number of task-agnostic O/CML models achieve bi-level meta-objectives, via prediction-based task classification (Rajasegaran et al., 2020), or dynamic selection of task-specific parameters (Jerfel et al., 2019).

FLAME shares the motivation to achieve *task-agnostic* bi-level meta-objectives, achieved via latent task inferences. The latter also shares the motivation to model task relations via clustering, but at the latent space of inferred task embeddings rather than the high-dimensional weight space.

Feedforward O/CML: Feedforward meta-learners have received limited attention in O/CML. PGMA (Hu et al., 2018) presents a parameter generator that uses the encoded representation of each individual sample to generate a subset of weight parameters for an otherwise-shared DNN classifiers, where forgetting is mitigated by generative replay. MERLIN (Joseph & Balasubramanian, 2020) uses a VAE to model the distribution of task-specific DNN weight parameters *after* they are optimized to streaming tasks, where forgetting is mitigated by replaying previously-learned weight parameters. CML-BGNN (Luo et al., 2020) utilizes a Bayesian framework to continually update a memory-augmented graph neural network to model both inter-task and intra-task relations for meta-learning. All of these existing models operate in a task-aware setting knowing task boundaries and identities.

FLAME shares the motivation in learning feedforward adaptation of task-specific DNNs, but advances these existing works by inferring task relations at the latent-space without requiring task identities.

General DNN adaptation: Outside O/CML, feed-forward meta-learners represent a vibrant area of research that learns a feedforward mapping from context samples to weight parameters of task-specific DNNs (Gordon et al., 2018; Zaheer et al., 2018) or directly the prediction of query samples (Hochreiter et al., 2001; Mishra et al., 2018; Requeima et al., 2020). A significant body of research studies questions such as how to derive context embeddings (Zaheer et al., 2018) or how to adapt task-specific DNNs (Gordon et al., 2018), including the use of hypernetworks (Ha et al., 2017).

Outside the scope of meta-learners, there has also been increasing interest in efficient adaptation of (large) DNNs, such as bias fine-tuning (Ben-Zaken et al., 2022), bottleneck-adapters (Houlsby et al., 2019; Wu et al., 2023b), and the adaptation of a difference weight $d\mathbf{W}$ (Guo et al., 2020). A notable success is LoRA (Hu et al., 2021) and its extension to CNNs (Ali et al., 2025) that reduce $d\mathbf{W}$ to low-rank matrices. LoRA-based meta-learning (Cheng et al., 2025) has shown context-modulation without capacity saturation often seen in high-dimensional feedforward meta-learners.

Unfortunately, these advances are mostly un-explored in O/CML. FLAME bridges this gap but further extends these advances by 1) demonstrating their versatility in adapting both DNN feature extractors and classsifiers, 2) leveraging their latent task embeddings to infer task identities and relations, and via which to 3) enable continual meta-optimization in a *task-agnostic* fashion.

3 Methodology: CML with Feedforward Latent Task Inference

For clarify, we start by defining several terminologies for the problem settings we operate under.

Task presentation settings – task-based *vs*. task-free: This defines how training data is structured and delivered to the learner. In a task-based setting, data is presented as explicitly separated, sequential tasks, each over a dedicated phase (Aljundi et al., 2019; Ven et al., 2025). In a task-free setting, data arrives as a continuous, unsegmented stream with no explicit boundaries between tasks (Ven et al., 2025; Aljundi et al., 2019). Similar to existing O/CML works, we consider the *task-based* setting, where we follow the assumption of *local stationarity* but ablate its effect on FLAME in Section 4.4.

Task identity settings – task-agnostic *vs*. task-aware: This defines whether the learner has access to task boundary and identity information. In a task-aware setting, the learner is provided with task identities at both training and inference time (Ven & Tolias, 2019). In a task-agnostic setting, neither task boundary nor identities information is available to the learner at any stage (Zeno et al., 2019; Lesort et al., 2022). FLAME operates under the *task-agnostic* setting.

Domain- *vs*. class-incremental: In "domain-incremental" setting, classes remain fixed across datasets but label-orthogonal domain information shifts (Ven & Tolias, 2019). In "class-incremental" setting, classes presented to the learners are changing (*e.g.*, standard N-way-K-shot classifications) within each dataset, across sequentially presented datasets. We use both settings to showcase two different types of DNN adaptation strategies within the general concept of feedforward meta-learners.

3.1 FEEDFORWARD *What-How* META-MODELS

FLMAE learns the inner-loop of the bi-level meta-optimization as a feedforward process: to extract *what* task current context samples comes from, and *how* to adapt the DNN to this task.

Adaptation of DNN classifers: In "domain-incremental" settings, we adapt only the classifier layer.

What: Consider a N-way-K-shot context set $\mathcal{D}_c^t = \{(\mathbf{x}_{c,i}, y_{c,i})\}_{i=1}^{N\times K}$ for task t and a shared feature extractor $f_\psi : \mathcal{X} \to \mathbb{R}^F$. We simply use f_ψ to obtain feature embedding $\mathbf{z}_{c,i} = f_\psi(\mathbf{x}_{c,i})$ for each context sample and aggregate them class-wise, such that for each class n, the set of embeddings is $Z_n = \{\mathbf{z}_{c,i} \mid y_{c,i} = n\}$ with cardinality $|Z_n|$. We compute the class representation $\mathbf{r}_n^t \in \mathbb{R}^F$ by averaging across $\mathbf{z}_{i,c} \in Z_n$, and stack them to form a task representation $\mathbf{R}^t \in \mathbb{R}^{F\times N}$.

How: We employ a small hypernetwork h_ϕ to generate a task-specific classifier layer from $\mathbf{R}^t$. More specifically, we feed each column $\mathbf{r}_n^t$ into h_ϕ to produce a class-specific weight vector $\mathbf{w}_n^t \in \mathbb{R}^F$ for task t. The resulting weight matrix $\mathbf{W}^t \in \mathbb{R}^{F\times N}$ define a linear classifier that predicts the class label for a query sample $\mathbf{x}_q$ in task t on its feature $\mathbf{z}_q = f_\psi(\mathbf{x}_q)$ extracted the feature extractor f_ψ.

Nested adaptation of DNN feature extractors and classifiers: In "class-incremental" settings, we nest the adaptation of classifiers for the N-way-K-shot classification, within the adaptation of feature extractors from learnable dataset-specific embeddings, as illustrated in Fig. 1B.

What: At the lower-level tasks of N-way-K-shot classifications, the extraction of task embeddings follows the description earlier. At the higher-level task of distinguishing datasets, we design learnable embedding vectors $\mathbf{s}^d$ specific to a dataset d. The decision of allocating new embeddings or re-use known embeddings will be tied to the task-relation modeling to be described in Section 3.2.

How: The adaptation of feature extractor is built on LoRA for transformers (Hu et al., 2021) and LLoRA (Ali et al., 2025) for CNNs. We employ a hypernetwork h_φ^f to generate (L)LoRA matrices $\mathbf{A}^d$ and $\mathbf{B}^d$ from $\mathbf{s}^d$, with which we adapt the feature extractor f_{ψ^d} for dataset d as $\psi^d = \psi_0 + act(\mathbf{A}^d\mathbf{B}^d)$, φ_0 shared across all datasets. The classifier adaptation then utilizes this dataset-specific f_ψ^d. We elaborate on this adaptation for different CNN and transformer backbones in Appendix B.6.

3.2 CONTINUAL META-LEARNING WITH LATENT TASK INFERENCE

In a stationary setting where all datasets are presented at once to the meta-learners, the performance of the adapted DNN on query samples across all tasks will be used to optimize the meta-learner. To continually approximate this bi-level meta-optimization over streaming tasks, we focus on the inference of task identities and task relations at the latent space of inferred task-embeddings.

Task boundary detection: Consistent with prior CML works (He et al., 2019; Caccia et al., 2021), we define the most recent $N \times K$ observations as the *context set* for the streaming task. As a result, when the underlying data distribution drifts, we expect not only a distribution shift at the task embedding space, but also a decline in the predictive accuracy of the adapted DNN due to context-query mismatch. To flag such shifts, we use a self-calibrating two-channel rule: loss and embedding. For the loss, let L_l be the context-conditioned loss at global batch index l and $\bar{L}_{l-1:l-m}$ the moving average of the loss over the past m batches. We score the residual $r_l = L_l - \bar{L}_{l-1:l-m}$ as: $z_l^{\text{loss}} = \big(r_l - \text{med}(r)\big)/\big(1.4826\,\text{MAD}(r)\big)$, with the median and maximum absolute deivation (MAD) taken over the trailing R residuals. For the embedding, let μ_l be the mean embedding of the query batch and $d_l = \mu_l - \frac{1}{m}\sum_{i=1}^{m}\mu_{l-i}$ its deviation from the mean of the past m batches. We whiten d_l by its $1.4826\,\text{MAD}$ over the trailing R deviation vectors, and score its root-mean-square, e_l, as $z_l^{\text{emb}} = \big(e_l - \text{med}(e)\big)/\big(1.4826\,\text{MAD}(e)\big)$. We declare a dataset boundary whenever $S_l = (z_l^{\text{loss}})^+ + (z_l^{\text{emb}})^+ > \kappa$. We use $\kappa = 6.5$, $m = 5$, $R = 100$ for all experiments.

Task-relational reservoir buffer: To prevent catastrophic forgetting, we employ experience replay with a reservoir buffer. To enable context-query pairing for replayed samples, we enrich the reservoir with estimated task identities and relations. To truly consolidate knowledge about known tasks, we employ a Bayesian Gaussian mixture model (GMM) to continually construct clusters from task embeddings of reservoir samples. At each detected task boundary t, embedding of context samples (in the reservoir and streaming buffer) are derived based on their previously-estimated pseudo task labels. To update the GMM, previous mixture components and their weights are used to initialize the new clustering. If the streaming-task embeddings form a distinct cluster, a new GMM component is added; its final weight indicates whether the task is genuinely new (high weight) or recurring (low weight). Components whose weights fall below a certain threshold are pruned to prevent uncontrolled growth. After the GMM update, the reservoir is rebalanced: with Q_t mixture components and a buffer capacity of M, each component retains M/Q_t samples, discarding any excess. This effectively approximates the data distribution up to any detected task boundary as:

$$p_{\text{GMM}}(\mathbf{x}) = \sum\nolimits_{l=1}^{Q_t} \pi_l p^l_{\text{GMM}}(\mathbf{x}), \quad p^l_{\text{GMM}}(\mathbf{x}) \sim \mathcal{N}\big(\mathbf{x}|\boldsymbol{\mu}_l, \boldsymbol{\sigma}_l\big) \tag{1}$$

where the mixing coefficient π_l is uniformly set to be $1/Q_t$. The cluster assignment acts as pseudo task labels, to facilitate context-query matching when samples are drawn from the reservoir. As the GMM update is triggered by a detected task boundary, the number of GMM components typically grow in Q_t and, in both domain- and class-incremental settings, correspond to dataset-level of tasks.

Compared to boundary-based estimation of task identifers, this clustering strategy recognizes known *vs*. novel tasks at a detected task boundary to continually consolidate knowledge about known tasks. It can prevent buffer fragmentation and help maintain a balanced buffer for recurring and rare tasks.

Continual bi-Level meta-optimization: We now optimize the feedforward meta-model $\mathcal{M}_w$ by:

$$\hat{w} = \arg\max_{w} \sum\nolimits_{l=1}^{Q_t} \pi_l \mathbb{E}_{p^l_{\text{GMM}}(\mathbf{x},y)}[\log p_{\theta^l = \mathcal{M}_w(\mathcal{D}^l_c)}(y|\mathbf{x}) \tag{2}$$

where $\log p(y|\mathbf{x})$ is approximated by the cross-entropy loss. The adaptation of $\theta^l = \mathcal{M}_w(\mathcal{D}^l_c)$ is as described in Section 3.1, where $\theta = \mathbf{W}$ for adapting only the classifier, or $\theta = \{\mathbf{W}, \mathbf{A}, \mathbf{B}\}$ if the feature extracted is also adapted. This loss is accumulated over p_{GMM} in the reservoir buffer as defined in Section 3.2, approximating the bi-level meta-objective for the feedforward meta-model.

4 Experiments

Baselines: For task-aware MAML models, we included FTML with a single weight initialization (Finn et al., 2019), HSML (Yao et al., 2019), ACML (Wu et al., 2023a) and VC-BML (Zhang et al., 2021) with a cluster of initialization weights. For task-agnostic MAML models, we included iTAML (Rajasegaran et al., 2020) with bi-level meta-objective and La-MAML (Joseph & Gu, 2021) with gradient-alignment objective, both with a single weight initialization: we were not able to include the clustering-based approach in (Jerfel et al., 2019) due to the absence of public implementations. For feedforward CML, we included PGMA (Hu et al., 2018), MERLIN (Joseph & Balasubramanian, 2020) and CML-BGNN (Luo et al., 2020), all task-aware. Additional details are in Appendix B.

Metrics: We measured 1) **learned accuracy (LA)** (Riemer et al., 2019) for each task after it was first learned, measuring a model's plasticity, 2) **retained accuracy (RA)** (Lopez-Paz & Ranzato, 2017) for each task as the training continues to other tasks, measuring a model's ability to mitigate forgetting, and 3) **backward transfer and interference (BTI)** defined by the average change in LA and RA.

4.1 Domain-Incremental CML

We considered two popular benchmark datasets, MNIST-Rotations (Lopez-Paz & Ranzato, 2017) and MNIST-Permutations (Kirkpatrick et al., 2017), for simulated domain shifts among datasets.

Streams of non-recurring tasks: We first adopted the standard setting with a sequence of 20 non-recurring tasks on each dataset. On MNIST-Rotations, each task was obtained by rotating the original MNIST dataset by a fixed amount between 0 and 180 degrees. On MNIST-Permutations, each task was obtained by applying a fixed permutation to the original MNIST dataset. Each task had the same 10 classes and 60000 samples. We used a total buffer size of 30000 samples for replay.

Table 1: Quantitative results on MNIST-Rotations and MNIST-Permutations. Best performance in bold (and second-best underlined); * denotes statistically significant improvements (Welch's t-test, $p < 0.05$). △: MAML-based; □: Feedforward. Empty: task-aware; solid: task-agnostic.

METHOD	ROTATIONS			PERMUTATIONS		
	RA	LA	BTI	RA	LA	BTI
FTML△	88.45 ± 0.51	94.20 ± 0.42	-5.48 ± 0.42	59.98 ± 1.20	94.15 ± 0.13	-33.50 ± 2.45
HSML△	75.67 ± 1.25	83.77 ± 1.37	-8.12 ± 0.89	50.37 ± 0.98	85.09 ± 0.76	-34.25 ± 0.61
ACML△	87.32 ± 0.27	93.83 ± 0.11	-6.50 ± 0.39	80.33 ± 0.32	93.34 ± 0.04	-13.00 ± 0.29
VC-BML△	83.59 ± 0.46	94.93 ± 0.04	-11.33 ± 0.45	77.00 ± 1.25	94.09 ± 0.12	-17.08 ± 1.14
iTAML▲	91.68 ± 0.17	89.89 ± 0.08	1.78 ± 0.23	88.24 ± 0.18	87.10 ± 0.07	1.19 ± 0.08
La-MAML▲	83.49 ± 0.54	70.48 ± 0.58	13.01 ± 0.31	73.70 ± 0.63	48.34 ± 0.01	25.36 ± 0.64
PGMA□	91.75 ± 0.55	93.76 ± 0.27	-2.02 ± 0.40	57.77 ± 0.61	**95.30 ± 0.15***	-37.52 ± 0.70
MERLIN □	82.14 ± 0.65	81.80 ± 0.15	0.01 ± 0.18	85.06 ± 0.12	85.08 ± 0.11	-0.02 ± 0.21
CML-BGNN □	84.71 ± 0.21	91.52 ± 0.27	-1.35 ± 0.74	83.68 ± 0.27	89.59 ± 0.26	-5.33 ± 0.99
FLAME■	**94.94 ± 0.14***	**94.97 ± 0.13**	-0.02 ± 0.19	**89.72 ± 0.20***	91.87 ± 0.05	-2.15 ± 0.23
METHOD	ROTATIONS-RECURRING			PERMUTATIONS-RECURRING		
	RA	LA	BTI	RA	LA	BTI
FTML△	84.16 ± 0.65	91.43 ± 0.39	-7.26 ± 0.76	82.06 ± 0.97	89.17 ± 0.43	-7.11 ± 0.83
HSML△	59.38 ± 5.63	82.08 ± 3.14	-22.70 ± 6.46	71.39 ± 0.44	84.34 ± 0.22	-28.88 ± 0.06
ACML△	85.93 ± 0.08	91.97 ± 0.13	-6.04 ± 0.13	86.45 ± 0.20	91.32 ± 0.26	-1.37 ± 4.69
VC-BML△	82.68 ± 0.61	**92.41 ± 0.22***	-9.73 ± 0.32	87.77 ± 0.33	**92.37 ± 0.06***	-4.60 ± 0.22
iTAML▲	89.98 ± 0.25	83.42 ± 0.13	6.56 ± 0.11	90.01 ± 0.19	81.72 ± 0.27	8.29 ± 0.11
La-MAML▲	77.56 ± 0.48	58.33 ± 1.03	19.23* ± 0.45	73.17 ± 0.75	37.95 ± 0.32	35.22 ± 0.36
PGMA□	89.63 ± 0.84	73.59 ± 0.52	16.04 ± 0.40	81.66 ± 0.65	26.92 ± 0.15	54.74* ± 0.44
MERLIN□	81.97 ± 1.03	81.75 ± 0.49	0.22 ± 1.05	85.41 ± 0.32	85.01 ± 0.18	0.40 ± 0.38
CML-BGNN□	89.51 ± 0.66	90.34 ± 1.35	-1.80 ± 0.25	88.70 ± 0.27	90.22 ± 0.04	-1.52 ± 0.23
FLAME■	**93.67 ± 0.06***	91.88 ± 0.21	1.79 ± 0.25	**91.90 ± 0.15***	88.56 ± 0.16	3.34 ± 0.30

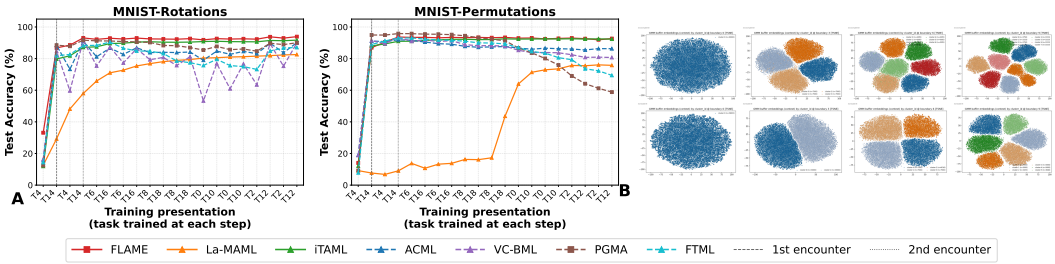

Figure 2: **(A)** Test accuracy of selected task (dashed line: first encounter; dotted line: recurrence) over the task sequence. **(B)** t-SNE visualization of reservoir task embedding color-coded by cluster identities for non-recurring (top) vs. recurring (bottom) task sequence on MNIST-Rotations.

Table 1 summarizes quantitative results on both datasets. Within task-aware MAML-baselines (△), clustering of weight initializations did not seem to help significantly. Within task-agnostic MAML-baselines (▲), the use of gradient alignment as the meta-objective (La-MAML) significantly under-performed than iTAML with bi-level meta-objectives. Performances of feedforward PGMA (□) were among the best baselines on MNIST-Rotations, although it deteriorated in MNIST-Permutations. As a task-agnostic approach, FLAME's delivered competitive **LA** and significantly-improved **RA** in comparison to even task-aware baselines. Across all models, FLAME also demonstrated the most stable **BTI**, showcasing its ability to continually infer and adapt without forgetting.

Streams of recurring tasks: To test FLAME's task-relational reservoir, we considered streams of recurring tasks in both datasets. We selected ten distinct tasks from each dataset and split them to present to the meta-model in an intertwined fashion. Quantitative results are also summarized in Table 1. The relative performance among the baselines were consistent to that observed in non-recurrent task sequences, where FLAME showed competitive **LA**, significantly-improved **RA**, and consistent **BTI** in comparison to all baselines. Furthermore, note that most baselines' **RA** was sensitive to the dataset or even the task sequence within the same dataset. In comparison, FLAME's RA remained stable across settings, speaking to its versatility in adapting to diverse task landscapes.

Table 2: Mean meta-test accuracy on all tasks *after* learning is completed on the task sequence. The reported accuracies therefore represent **LA** for the last task and **RA** for the previous three.

	Forward Order (Non-Stationary)				Reverse Order (Non-Stationary)			
Algorithms	**VGG-Flowers**	**miniImageNet**	**CIFAR-FS**	**Omniglot**	**Omniglot**	**CIFAR-FS**	**miniImageNet**	**VGG-Flowers**
FTML △	75.21 ± 0.17	45.09 ± 0.87	57.21 ± 0.13	95.51 ± 0.11	95.16 ± 0.25	58.31 ± 1.02	46.27 ± 0.58	76.01 ± 0.28
HSML △	62.21 ± 0.86	49.34 ± 2.08	61.48 ± 1.57	93.86 ± 2.31	94.59 ± 0.65	$\underline{60.94 \pm 0.83}$	$\underline{48.67 \pm 0.39}$	64.31 ± 0.05
ACML △	60.89 ± 1.84	37.50 ± 0.56	45.47 ± 0.62	**99.56 ± 0.06***	88.88 ± 1.31	54.25 ± 1.15	44.29 ± 0.15	**77.91 ± 0.58**
VC-BML △	60.40 ± 4.32	38.06 ± 1.23	47.57 ± 2.15	97.38 ± 1.02	87.17 ± 2.20	53.42 ± 1.88	43.39 ± 0.02	76.77 ± 0.74
iTAML ▲	70.72 ± 2.96	39.16 ± 1.47	50.79 ± 2.14	94.97 ± 0.33	94.46 ± 0.35	50.70 ± 0.79	39.36 ± 0.53	64.77 ± 0.38
La-MAML ▲	**76.96 ± 0.57**	48.91 ± 0.95	$\underline{61.58 \pm 0.33}$	96.42 ± 0.12	$\underline{96.09 \pm 0.16}$	60.35 ± 0.83	48.66 ± 0.48	$\underline{76.91 \pm 0.31}$
PGMA □	47.66 ± 1.54	37.90 ± 1.48	49.52 ± 2.96	94.90 ± 0.45	86.95 ± 0.94	37.81 ± 0.30	34.16 ± 0.85	55.23 ± 0.51
CML-BGNN □	24.27 ± 2.77	23.60 ± 1.85	23.76 ± 2.68	32.03 ± 5.63	33.93 ± 8.22	23.41 ± 4.28	23.94 ± 5.25	23.76 ± 6.75
FLAME■	$\underline{76.52 \pm 0.48}$	**54.73 ± 1.40***	**67.23 ± 0.30***	$\underline{97.98 \pm 0.30}$	**97.78 ± 0.09***	**64.45 ± 4.91**	**52.40 ± 3.68**	76.02 ± 0.40

Table 3: Mean meta-test accuracy on all five text datasets *after* learning was completed on the task sequence. The reported accuracies thus represent **LA** for the last task and **RA** for the previous four.

	Order 1 (Non-Stationary)					Order 2 (Non-Stationary)				
Algorithms	**Yelp**	**AGNews**	**DBpedia**	**Amazon**	**Yahoo**	**DBpedia**	**Yahoo**	**AGNews**	**Amazon**	**Yelp**
FTML △	$\underline{70.32 \pm 1.86}$	$\underline{93.12 \pm 0.45}$	99.52 ± 0.24	66.55 ± 1.48	84.79 ± 0.47	$\underline{99.64 \pm 0.09}$	82.29 ± 1.27	$\underline{92.56 \pm 0.68}$	68.26 ± 1.11	70.93 ± 0.34
iTAML ▲	46.78 ± 0.79	72.48 ± 1.09	94.28 ± 0.12	43.75 ± 0.61	61.86 ± 0.62	96.81 ± 0.42	66.75 ± 1.43	74.76 ± 1.45	43.47 ± 1.07	46.26 ± 0.29
La-MAML ▲	67.92 ± 1.02	92.46 ± 0.38	$\underline{99.57 \pm 0.16}$	$\underline{66.98 \pm 0.22}$	$\underline{86.02 \pm 0.43}$	99.54 ± 0.05	$\underline{82.78 \pm 1.71}$	92.46 ± 0.73	$\underline{69.59 \pm 1.90}$	$\underline{71.94 \pm 1.88}$
FLAME■	**71.74 ± 1.55**	**93.51 ± 0.57**	**99.62 ± 0.13**	**68.96 ± 0.91**	**87.22 ± 0.66**	**99.73 ± 0.13**	**85.10 ± 1.75**	**93.42 ± 0.42**	**71.80 ± 0.73**	**74.71 ± 0.53**

Fig. 2A examines the test-accuracy of selected recurring tasks along the task sequence. In the **MNIST-Rotations** example, in between the recurrence of a task, FLAME and iTAML were the only two high-performing models able to maintain test performance on the task and continue improving it upon recurrence; in comparison, models like ACML and VC-BML experienced a deterioration of performance before re-learning upon recurrence. In the **MNIST-Permutations** example, FLAME and iTAML showed strong retained performance in comparison to deterioration observed in other baselines. Additional examples in Fig. 6 (Appendix C.2) showed that the gap between iTAML and FLAME increased for tasks presented later in the sequence, suggesting the bottleneck of a single weight initialization in iTAML compared to FLAME's feedforward meta-learner.

Fig. 2B provides tSNE visualizations of the task embedding of reservoir samples at selected task boundaries on MNIST-Rotations for non-recurring (top) and recurring (bottom) task sequences, colored by the GMM cluster identifier. The contrast of these two sequences demonstrated the ability of FLAME's task-relational reservoir to automatically recognize known *vs*. novel tasks as they stream in. The embedding visualization of the complete sequence is presented in Appendix C.1.

4.2 Class-Incremental CML

Vision datasets: We considered a common benchmark streaming four datasets (VGG-Flower, miniImagenet, CIFAR-FS, Omniglot), in two opposite orders of streaming following (Wu et al., 2023a). For each dataset, each task was 5-way 5-shot with 32 tasks per meta-loss update and a total of 2000 updates. We emploed a four-conv layer-based CNN as the backbone and used a replay buffer size of 30000. Table 2 lists the *final* performance of each model on each dataset *after* completing the task sequence. Fig. 7 and Table 7-8 report their complete performance throughout the task sequence. PGMA was not included due to poor initial performance observed (<30% retained performance).

Similar to previous results, FLAME's **LA** was on par with and sometimes weaker than VC-BML and ACML. It however demonstrated the strongest **RA** for all datasets and across both settings. Interestingly, among the baselines, the best-performing methods changed between the domain- *vs*. class-incremental setting: while iTAML and PGMA in general showed the most competitive **RA** in Table 1, La-MAML and HSML were more successful in Table 2. This highlighted the versatility of FLAME in remaining consistently strong, especially in retaining performance, in both settings.

To investigate the scalability of FLAME, we further confirmed its ability to continually adapt VIT-S/16 network from the DINOv3 family as the backbone. Results are provided in Appendix C.4.

Text datasets: To test FLAME in non-image modalities, we chose the five text-classification datasets (AGNews, DBPedia, Yelp, Amazon and Yahoo) from (Zhang et al., 2016), sampling 16 3-way 3-shot tasks per meta-loss update and 2000 meta updates per dataset. We used a BERT-based uncased backbone for this experiment and the same 30000-size memory buffer. We chose FTML, iTAML

Table 4: Comparison of feedforward FLAME and its MAML variant on the stream of four vision datasets. TTA-T: time-to-adapt (TTA) T tasks at test time; time-to-train (TTT); Accuracy is reported as per-task accuracy at the end of the task sequence. MAML-g: g refers to number of inner steps.

	Cost			Accuracy after training [%]			
Model	TTA-1 [s]	TTA-10 [s]	TTT [min]	VGG-Flowers	miniImageNet	CIFAR-FS	Omniglot
MAML-1	0.0081 ± 0.0011	0.0783 ± 0.0104	181.8 ± 5.4	73.58 ± 0.46	44.31 ± 1.43	55.67 ± 1.19	94.62 ± 0.55
MAML-5	0.0409 ± 0.0022	0.3936 ± 0.0221	584.9 ± 17.9	$\mathbf{76.88 \pm 0.12}$	49.26 ± 1.45	61.53 ± 1.06	$\mathbf{98.16 \pm 0.26}$
FLAME	$\mathbf{0.0077 \pm 0.0008}$	$\mathbf{0.0232 \pm 0.0015}$	$\mathbf{111.8 \pm 4.6}$	76.60 ± 0.59	$\mathbf{54.91 \pm 0.41}$*	$\mathbf{67.53 \pm 0.55}$*	98.10 ± 0.14

Table 5: Ablations of the feedforward meta-model optimized by gradient-alignment (A-GEM, La-MAML), or using regularization to address forgetting (Meta-BGD).

Method	Rotations			Permutations		
	RA	LA	BTI	RA	LA	BTI
FF A-GEM	81.61 ± 1.04	75.20 ± 1.97	6.41 ± 1.72	53.79 ± 1.01	39.95 ± 1.63	13.84 ± 2.15
FF La-MAML	81.70 ± 0.18	72.69 ± 0.19	9.00 ± 0.05	67.54 ± 0.07	58.55 ± 0.14	8.99 ± 0.20
FF Meta-BGD	90.10 ± 1.13	81.04 ± 2.01	9.05 ± 0.98	74.62 ± 1.69	49.86 ± 3.21	24.77 ± 3.05
FLAME	$\mathbf{94.94}$* $\pm \mathbf{0.14}$	$\mathbf{94.97}$* $\pm \mathbf{0.13}$	-0.02 ± 0.19	$\mathbf{89.72}$* $\pm \mathbf{0.20}$	$\mathbf{91.87}$* $\pm \mathbf{0.05}$	-2.15 ± 0.23

and La-MAML as baselines for their model-agnostic nature that does not require significant re-implementation. Final results in Table 3 and complete results throughout the task sequences in Table 10-11 continued to show strong **RA** performance by FLAME in comparison to the baselines.

4.3 Ablations of Key Methodological Elements in FLAME

Ablation of meta-learners: To ablate the choice of meta-learners, we replaced the feedforward meta-learner in FLAME with MAML, trained with MAML-1/-5 (1/5 iner steps) on the four vision datasets, both with oracle task boundaries provided to isolate effect of the meta-learners. Table 4 shows that FLAME significantly outperformed MAML-5 in two datasets while similar in the other two, and that feedforward adaptation is significantly more efficient in test time adapation even compared to MAML-1. FLAME also scaled much more efficiently to the number of tasks at adaptation.

Ablation of meta-objectives: Since feedforward meta-models were not well represented in existing baselines, we performed additional ablations to retain FLAME's feedforward meta-model but replace its bi-level meta-objectives with 1) gradient alignment realized with A-GEM (Chaudhry et al., 2018) and that used in La-MAML (Joseph & Gu, 2021), and 2) meta-model regularization realized with Bayesian Gradient Descent (BGD) (He et al., 2019). Results summarized in Table 5 demonstrated that both components, especially bi-level meta-optimization, were critical to FLAME's success.

Ablation of task inference mechanisms: To confirm the effectiveness of FLAME's task inference mechanism, we replaced it with oracle task information and results in Table 12 (Appendix C.6.1) showed that FLAME approaches the oracle results. Appendix C.6.1-C.6.2 also include detailed results of boundary detection and task clustering from main experiments presented in Section 4.1-4.2.

To further isolate the benefit of clustering-based task-relation consolidation, we set up a stream of tasks with frequently *vs*. rarely recurring tasks and compared FLAME with a variant that retained its boundary detection mechanism but simply assigned a new task label at each boundary. Fig. 11 in Appendix C.6.3 showed that, as known tasks frequently recurred, the performance of the variant on the rare task was affected while the use of latent clustering allowed FLAME continue to improve on the rare task, demonstrating the importance of consolidating task relations to avoid buffer fragmentation.

4.4 Stress Testing Latent Task Inference: Boundary Detection and Clustering

We now stress test the two main elements used by FLAME to realize task-agnostic CML: detection of task-boundaries, and cluster-based estimation of task identities. The former is measured by precision (tracking false positives) and recall (tracking false negatives) of boundary detection. The latter is measured by micro purity (MP, tracking merging of tasks within a cluster) and inverse micro purity (IMP, tracking splitting of a task into multiple clusters).

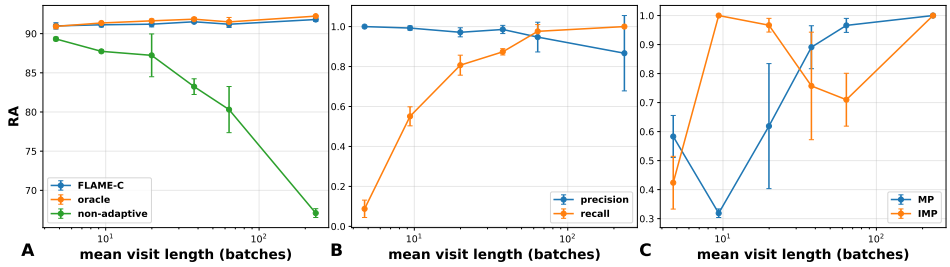

Figure 3: Stress testing FLAME as local stationary window length increased from an average of 5 to 200 batches. A: RA of FLAME *vs*. using oracle task information (oracle) or considering all tasks as one (non-adaptive). B: Boundary detection metrics. C: Latent clustering metrics.

Sensitivity analyses of hyperparameters: Appendix D.1-D.2 ablated the sensitivity of FLAME to hyperparameters in both components. These results showed that FLAME was robust in detecting task transition boundaries, and that clustering was robust when tasks were distinct. Merging of tasks could happen in datasets with similar tasks (*e.g.,* MNIST-Rotations): this however did not affect FLAME's performance which considered the tasks too similar to warrant model adaptation.

Effect of α-stationarity: We stress-tested the assumption of local stationarity on FLAME, following the definition of α-stationarity in (Caccia et al., 2021) where α is the probability the current task continues at the next timestamp. We trained FLAME on four-task sequences from MNIST-Rotations at $\alpha = 0.8, 0.9, 0.98, 0.099$ and 0.9999, corresponding to an average stationary window length of $5, 10, 25, 50, 100$, and 200 batches. Fig. 3A compares FLAME (blue) to its two references: task-aware version using oracle task labels (orange) as a performance upper bound; and non-adaptive version considering all four tasks as one task (green) as a performance lower bound as if all task boundaries are missed. Interestingly, at lower α (short stationary windows), the gap between these two versions was small, suggesting an approximation towards streaming of stationary distributions and reduced challenge of forgetting. In these settings, although FLAME had low recall in task boundaries (B) as well as task merging in the replay buffer (C), there was little impact on its performance. In comparison, as α increased, forgetting became a more significant challenge as shown by the deterioration when treating all tasks as one (green). In these settings where task inference became important, FLAME showed robust boundary detection (B) and well as task inference (C), retaining comparable performance to the oracle version throughout the α values tested. Appendix D.4 includes additional boundary detection results over task sequences at representative values of α.

Effect of local mixing boundaries: We tested FLAME's performance under gradual transition boundaries. Extending the idea of *boundary-local mixing* in (Sun et al., 2026), we varied the percentage of new task samples for the j-th batch within W number of transition batches as $\rho_j = j/(W+1)$, for $j = 1, \ldots, W$. A larger W thus introduces the new task more slowly. We swept $W \in \{0, \ldots, 5\}$ across five seeds. As shown in Fig. 14 on MNIST-Rotations, FLAME's boundary detection precision was not affected while recall decreased and plateaued $\sim 80\%$ near $W = 5$ and a delay of boundary detection by ~ 15 batches, resulting in a less than 1% decrease in RA performance.

Additional ablations on boundary detection rules, buffer-context size, replay cost are in Appendix E.

5 Conclusions and Discussion

We present FLAME to demonstrate the benefits of 1) feedforward meta-learners and 2) latent task inference, in allowing continual adaptation to streaming tasks. Our results demonstrated FLAME's ability to improve over existing works without any knowledge about task boundaries or identities, and its versatility across model backbones, local stationary windows, and gradual transition boundaries. FLAME does have several **limitations.** Task embedding clustering is currently performed separately from the meta-model, which misses the opportunity to directly model and infer the task embedding as a mixture of distributions. Although with strong retained accuracy, FLAME showed comparable learning accuracy to existing baselines and could be further improved in that aspect. Finally, FLAME is currently tested under task-based learning settings: given its strong robustness demonstrated across ablated local stationary windows, future works can explore its potential in task-free learning settings.

AI USE STATEMENT

In this work, we used generative AI tools to aid in and polish writing (e.g., appendix sections, tables), for retrieval and discovery (e.g., finding related work), for research ideation and execution (e.g., implementing the methodology and ablations), and for drafting sections of the paper. We did not use generative AI tools to generate synthetic datasets or to prove mathematical claims; the remaining required disclosure categories are not applicable to this work. All AI-assisted work was reviewed by the authors: LLM-generated code was verified and tested for correctness by the first author, and AI-generated plots and AI-polished text were reviewed by the first author. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All authors have read and adhere to the ICLR Code of Ethics. This work proposes [a continual meta-learning method] and evaluates it exclusively on publicly available benchmark datasets ([e.g., Omniglot, Mini-ImageNet]), used in accordance with their respective licenses. No human subjects were involved, no new data were collected or released, and no personally identifiable information was used; IRB approval was therefore not required. We are not aware of any conflicts of interest or sponsorship that influenced this work. While improvements in learning efficiency could in principle be applied to a wide range of downstream applications, we do not foresee direct harmful uses or fairness concerns specific to our method beyond those inherent to the underlying benchmark datasets.

REPRODUCIBILITY STATEMENT

(This section is **recommended** and does not count toward the page limit.)

It is important that the work published in ICLR is reproducible. Authors are strongly encouraged to include a paragraph-long Reproducibility Statement at the end of the main text (before references) to discuss the efforts that have been made to ensure reproducibility. This paragraph should not itself describe details needed for reproducing the results, but rather reference the parts of the main paper, appendix, and supplemental materials that will help with reproducibility. For example, for novel models or algorithms, a link to an anonymous downloadable source code can be submitted as supplementary materials; for theoretical results, clear explanations of any assumptions and a complete proof of the claims can be included in the appendix; for any datasets used in the experiments, a complete description of the data processing steps can be provided in the supplementary materials. Each of the above are examples of things that can be referenced in the reproducibility statement.

AUTHOR CONTRIBUTIONS

If you'd like to, you may include a section for author contributions as is done in many journals. This is optional and at the discretion of the authors.

ACKNOWLEDGMENTS

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

REFERENCES

Durmus Alp Emre Acar, Ruizhao Zhu, and Venkatesh Saligrama. Memory Efficient Online Meta Learning. In Proceedings of the 38th International Conference on Machine Learning, pp. 32–42. PMLR, July 2021. URL https://proceedings.mlr.press/v139/acar21b.html.

Munsif Ali, Leonardo Rossi, and Massimo Bertozzi. CoLoR-GAN: Continual Few-Shot Learning with Low-Rank Adaptation in Generative Adversarial Networks, October 2025. URL https://arxiv.org/abs/2510.13869v1.

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-Free Continual Learning, August 2019. URL http://arxiv.org/abs/1812.03596. arXiv:1812.03596 [cs.CV].

Elad Ben-Zaken, Shauli Ravfogel, and Yoav Goldberg. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models, September 2022. URL http://arxiv.org/abs/2106.10199. arXiv:2106.10199 [cs].

Massimo Caccia, Pau Rodriguez, Oleksiy Ostapenko, Fabrice Normandin, Min Lin, Lucas Caccia, Issam Laradji, Irina Rish, Alexandre Lacoste, David Vazquez, and Laurent Charlin. Online Fast Adaptation and Knowledge Accumulation: a New Approach to Continual Learning, January 2021. URL http://arxiv.org/abs/2003.05856. arXiv:2003.05856 [cs.AI].

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient Lifelong Learning with A-GEM. September 2018. URL https://openreview.net/forum?id=Hkf2_sC5FX.

Bo Cheng, Xu Wang, Jinda Liu, Yi Chang, and Yuan Wu. MeTA-LoRA: Data-Efficient Multi-Task Fine-Tuning for Large Language Models, October 2025. URL https://arxiv.org/abs/2510.11598v1.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks, July 2017. URL http://arxiv.org/abs/1703.03400. arXiv:1703.03400 [cs].

Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online Meta-Learning, July 2019. URL http://arxiv.org/abs/1902.08438. arXiv:1902.08438 [cs].

Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard Turner. Meta-Learning Probabilistic Inference for Prediction. September 2018. URL https://openreview.net/forum?id=HkxStoC5F7.

Demi Guo, Alexander M. Rush, and Yoon Kim. Parameter-Efficient Transfer Learning with Diff Pruning, December 2020. URL https://arxiv.org/abs/2012.07463v2.

David Ha, Andrew M. Dai, and Quoc V. Le. HyperNetworks. February 2017. URL https://openreview.net/forum?id=rkpACe1lx.

James Harrison, Apoorva Sharma, Chelsea Finn, and Marco Pavone. Continuous Meta-Learning without Tasks, October 2020. URL http://arxiv.org/abs/1912.08866. arXiv:1912.08866 [cs.LG].

Xu He, Jakub Sygnowski, Alexandre Galashov, Andrei A. Rusu, Yee Whye Teh, and Razvan Pascanu. Task Agnostic Continual Learning via Meta Learning, June 2019. URL http://arxiv.org/abs/1906.05201. arXiv:1906.05201 [stat].

Sepp Hochreiter, A. Steven Younger, and Peter R. Conwell. Learning to Learn Using Gradient Descent. volume 2130, pp. 87–94, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-42486-4 978-3-540-44668-2. doi: 10.1007/3-540-44668-0_13. URL http://link.springer.com/10.1007/3-540-44668-0_13. Book Title: Artificial Neural Networks — ICANN 2001 Series Title: Lecture Notes in Computer Science.

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, September 2022. ISSN 1939-3539. doi: 10.1109/TPAMI.2021.3079209. URL https://ieeexplore.ieee.org/document/9428530/.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-Efficient Transfer Learning for NLP, June 2019. URL http://arxiv.org/abs/1902.00751. arXiv:1902.00751 [cs].

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models, June 2021. URL https://arxiv.org/abs/2106.09685v2.

Wenpeng Hu, Zhou Lin, Bing Liu, Chongyang Tao, Zhengwei Tao, Jinwen Ma, Dongyan Zhao, and Rui Yan. Overcoming Catastrophic Forgetting for Continual Learning via Model Adaptation. September 2018. URL https://openreview.net/forum?id=ryGvcoA5YX.

Ghassen Jerfel, Erin Grant, Thomas L. Griffiths, and Katherine Heller. Reconciling meta-learning and continual learning with online mixtures of tasks, June 2019. URL http://arxiv.org/abs/1812.06080. arXiv:1812.06080 [cs].

Joel Joseph and Alex Felix Gu. La-MAML: Look-ahead Meta Learning for Continual Learning, ML Reproducibility Challenge 2020. January 2021. URL https://openreview.net/forum?id=d0svLMnvzWK.

K. J. Joseph and Vineeth N. Balasubramanian. Meta-Consolidation for Continual Learning, October 2020. URL http://arxiv.org/abs/2010.00352. arXiv:2010.00352 [cs].

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences, 114(13):3521–3526, March 2017. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1611835114. URL http://arxiv.org/abs/1612.00796. arXiv:1612.00796 [cs].

Timothée Lesort, Massimo Caccia, and Irina Rish. Understanding Continual Learning Settings with Data Distribution Drift Analysis, July 2022. URL http://arxiv.org/abs/2104.01678. arXiv:2104.01678 [cs.LG].

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, pp. 6470–6479, Red Hook, NY, USA, December 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4. URL https://dl.acm.org/doi/10.5555/3295222.3295393.

Yadan Luo, Zi Huang, Zheng Zhang, Ziwei Wang, Mahsa Baktashmotlagh, and Yang Yang. Learning from the Past: Continual Meta-Learning with Bayesian Graph Neural Networks. Proceedings of the AAAI Conference on Artificial Intelligence, 34(04):5021–5028, April 2020. ISSN 2374-3468. doi: 10.1609/aaai.v34i04.5942. URL https://ojs.aaai.org/index.php/AAAI/article/view/5942.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A Simple Neural Attentive Meta-Learner, February 2018. URL http://arxiv.org/abs/1707.03141. arXiv:1707.03141 [cs].

Alex Nichol and John D. Schulman. Reptile: a Scalable Metalearning Algorithm. arXiv: Learning, March 2018. URL https://www.semanticscholar.org/paper/Reptile%3A-a-Scalable-Metalearning-Algorithm-Nichol-Schulman/672c586880655dc544474280a6e086c1fc901c85.

Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. iTAML: An Incremental Task-Agnostic Meta-learning Approach, March 2020. URL http://arxiv.org/abs/2003.11652. arXiv:2003.11652 [cs].

Jathushan Rajasegaran, Chelsea Finn, and Sergey Levine. Fully Online Meta-Learning Without Task Boundaries, February 2022. URL http://arxiv.org/abs/2202.00263. arXiv:2202.00263 [cs].

James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E. Turner. Fast and Flexible Multi-Task Classification Using Conditional Neural Adaptive Processes, January 2020. URL http://arxiv.org/abs/1906.07697. arXiv:1906.07697 [stat.ML].

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to Learn without Forgetting by Maximizing Transfer and Minimizing Interference, May 2019. URL http://arxiv.org/abs/1810.11910. arXiv:1810.11910 [cs].

Jaehyeon Son, Soochan Lee, and Gunhee Kim. When Meta-Learning Meets Online and Continual Learning: A Survey, November 2023. URL https://arxiv.org/abs/2311.05241v2.

Guiquan Sun, Xikun Zhang, Jingchao Ni, and Dongjin Song. DRIFT: A Benchmark for Task-Free Continual Graph Learning with Continuous Distribution Shifts, June 2026. URL http://arxiv.org/abs/2605.12998. arXiv:2605.12998 [cs.LG].

Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning, April 2019. URL http://arxiv.org/abs/1904.07734. arXiv:1904.07734 [cs].

Gido M. van de Ven, Nicholas Soures, and Dhireesha Kudithipudi. Continual Learning and Catastrophic Forgetting. pp. 153–168. 2025. doi: 10.1016/B978-0-443-15754-7.00073-0. URL http://arxiv.org/abs/2403.05175. arXiv:2403.05175 [cs.LG].

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching Networks for One Shot Learning, December 2017. URL http://arxiv.org/abs/1606.04080. arXiv:1606.04080 [cs].

Buddhi Wickramasinghe, Gobinda Saha, and Kaushik Roy. Continual Learning: A Review of Techniques, Challenges, and Future Directions. IEEE Transactions on Artificial Intelligence, 5 (6):2526–2546, June 2024. ISSN 2691-4581. doi: 10.1109/TAI.2023.3339091. URL https://ieeexplore.ieee.org/document/10341211.

Bin Wu, Jinyuan Fang, Xiangxiang Zeng, Shangsong Liang, and Qiang Zhang. Adaptive Compositional Continual Meta-Learning. In Proceedings of the 40th International Conference on Machine Learning, pp. 37358–37378. PMLR, July 2023a. URL https://proceedings.mlr.press/v202/wu23d.html.

Junde Wu, Wei Ji, Yuanpei Liu, Huazhu Fu, Min Xu, Yanwu Xu, and Yueming Jin. Medical SAM Adapter: Adapting Segment Anything Model for Medical Image Segmentation, December 2023b. URL http://arxiv.org/abs/2304.12620. arXiv:2304.12620 [cs].

Huaxiu Yao, Ying Wei, Junzhou Huang, and Zhenhui Li. Hierarchically Structured Meta-learning, November 2019. URL http://arxiv.org/abs/1905.05301. arXiv:1905.05301 [cs].

Huaxiu Yao, Yingbo Zhou, Mehrdad Mahdavi, Zhenhui Li, Richard Socher, and Caiming Xiong. Online Structured Meta-learning, October 2020. URL https://arxiv.org/abs/2010.11545v1.

Pauching Yap, Hippolyt Ritter, and David Barber. Bayesian Online Meta-Learning. October 2020. URL https://openreview.net/forum?id=ucEXZQncukK.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep Sets, April 2018. URL http://arxiv.org/abs/1703.06114. arXiv:1703.06114 [cs].

Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. Task agnostic continual learning using online variational bayes. arXiv preprint arXiv:1803.10123, 2019.

Qiang Zhang, Jinyuan Fang, Zaiqiao Meng, Shangsong Liang, and Emine Yilmaz. Variational Continual Bayesian Meta-Learning. In Advances in Neural Information Processing Systems, volume 34, pp. 24556–24568. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/cdd0500dc0ef6682fa6ec6d2e6b577c4-Abstract.html.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level Convolutional Networks for Text Classification, April 2016. URL http://arxiv.org/abs/1509.01626. arXiv:1509.01626 [cs.LG].

A Literature Summary

We present a complete summary of related works in Table 6 based on several key axes of design choices including: task awareness (task-aware or -agnostic), meta algorithmic prior (MAML or feedforward), meta-optimization objective (bi-level or gradient alignment), and CL strategies (replay or regularization). Because O/CML often involves different evaluation settings, we further annotate the experimental data setting (stationary or non-stationary) and evaluation setting (current or *all* task performances) under which each of these methods was originally presented.

As can seen from the summary table, FLAME contributes a unique solution in CML by proposing a feedforward meta-learner with latent task inference to enable bi-level meta-optimization in a task-agnostic fashion.

Table 6: Comparison of CML methods across design axes.

Model	Task awareness	Meta-learner	Meta-objective	CL strategy	Data setting	Evaluation
FTML	Aware	MAML – None	Bi-level	Exact Replay	Stationary	Adapted model on current task
MOML	Aware	MAML – None	Bi-level	Regularization	Stationary	Adapted model on current task
HSML	Aware	MAML – Weight Clusters	Bi-level	None	Non-stationary	Adapted model on all tasks
OSML	Aware	MAML – Model Pathways	Bi-level	Exemplar Replay	Stationary	Adapted model on current task
BOML	Aware	MAML – None	Bi-level	Regularization	Both	Adapted model on all tasks
VC-BML	Aware	MAML – Weight Clusters	Bi-level	Regularization	Non-stationary	Adapted model on all tasks
ACML	Aware	MAML – Weight Clusters	Bi-level	Regularization	Non-stationary	Adapted model on all tasks
FOML	Agnostic	MAML – None	Bi-level	Exemplar Replay	Stationary	Adapted model on current task
iTAML	Agnostic	MAML – None	Bi-level	Exemplar Replay	Stationary	Pre-adapted to all tasks
DPMM	Agnostic	MAML – Weights Clusters	BI-level	None	Stationary	Adapted model on all tasks
MER	Agnostic	MAML – None	Gradient Alignment	Exemplar Replay	Stationary	Latest model on all tasks
La-MAML	Agnostic	MAML – None	Gradient Alignment	Exemplar Replay	Stationary	Latest model on all tasks
OSAKA	Agnostic	MAML – None	Bi-level	None	Non-stationary	Latest model on current task
MetaBGD	Agnostic	MAML-like – None	Bi-level	Regularization	Stationary	Adapted model on all tasks
PGMA	Aware	Feedforward – None	Implicit Bi-level	Generative replay	Stationary	Latest model on all tasks
MERLIN	Aware	Feedforward – None	Implicit Bi-level	Generative replay	Stationary	Adapted model on all tasks
MOCA	Agnostic	General – None	Implicit Bi-level	Regularization	Stationary	Latest model on current task
CML-BGNN	Aware	Feedforward – None	Implicit Bi-level	Generative replay	Stationary	Adapted model on current task

B Implementation Details

B.1 Experiment Detail for Domain-incremental Setting

In domain-incrementa setting, all models were trained for one full epoch per task (234 gradient steps, corresponding to 60,000 samples at a batch size of 256 with the final partial batch dropped). To prevent catastrophic forgetting, we used a reservoir of 30,000 samples for FLAME and all other baselines. The size of the amortized context set was fixed at $n \times k = 90$ support samples for all domain-incremental experiments.

For the **non-recurring** task stream, tasks were presented in a single randomized (seed=0) order drawn without replacement from the full set of 20 tasks. For the **recurring** task strem, a subset of 10 original tasks was selected, with each task split into two half-epoch presentations and arranged into a schedule of 20 total presentations, ensuring no task appears back-to-back with itself. In both settings, boundary detection was enabled via the additive mechanism of Section 3.2, with $\kappa = 6.5$, $m = 5$, $R = 100$, and the optimizer was reset at each detected boundary. Three seeds were used for all reported results.

B.2 Model Architecture for Domain-incremental Setting

Common Baseline Configs

- Per new task number of samples: 60,000
- Batch size: 256
- Number of batches: 234 (drop last incomplete batch)
- Buffer size: 150 per class per task (30,000 in total)
- Context size: 90
- **Input:** 784-dimensional flattened pixel vector (28×28)
- **Shared Feature extractor:** Linear$(784, 100) \to$ ReLU $\to$ Linear$(100, 100) \to$ ReLU $\to \mathbb{R}^{100}$
- **Amortized Classifier:** 1 classifier layer of 10 nodes; weight matrix shape 100×10
- Optimizer: AdamW, learning rate $\eta = 10^{-3}$
- LR scheduler: MultiStep decay, milestone at step 200, $\gamma = 10^{-5}$
- Seeds: 0, 10, 20

B.3 Experiment Detail for Class-incremental Setting

four-dataset non-stationary image classification: The class-incremental experiments consider a sequence of four visually distinct few-shot classification datasets presented one after another: VGG-Flowers, miniImageNet, CIFAR-FS, and Omniglot. In the **"Forward"** ordering, datasets are encountered in the sequence listed above; in the **"Reverse"** ordering, the sequence is inverted. Each dataset constitutes a distinct task domain, yielding the four boundaries per run.

Training follows an episodic protocol. At each iteration or meta-loss update, $T = 32$ episodes are sampled in parallel from the currently active dataset, with each episode composed of a 5-way 5-shot support set (25 context samples) and 15 query samples per class (75 queries per episode). Each dataset is trained for 2,000 such iterations, and 100 evaluation episodes are used at test time. The task-aware reservoir buffer holds up to 30,000 samples, shared across all observed datasets. Boundary detection is performed using the additive rule of Section 3.2, $S_l = (z_l^{\text{loss}})^+ + (z_l^{\text{emb}})^+ > \kappa$, with $\kappa = 6.5$, $m = 5$, $R = 100$, and the optimizer is reset upon each detected boundary. Three seeds were used for all reported results.

five-dataset non-stationary text classification: The class-incremental text dataset experiments consider, a sequence of five text classification datasets from distinct domains (Zhang et al., 2016) presented one after another: AGNews (news topics, 4 classes), Yelp (review ratings, 5 classes), DBpedia (Wikipedia entity types, 14 classes), Amazon (review ratings, 5 classes), and Yahoo Answers (question topics, 10 classes). We use two orderings: **Order 1** Yelp $\to$ AGNews $\to$ DBpedia

$\rightarrow$ Amazon $\rightarrow$ Yahoo; and **Order 2** DBpedia $\rightarrow$ Yahoo $\rightarrow$ AGNews $\rightarrow$ Amazon $\rightarrow$ Yelp; Each dataset constitutes a distinct task domain, yielding four dataset switches per run.

Training follows an episodic protocol. At each iteration or meta-loss update, $T = 16$ episodes are sampled in parallel from the currently active dataset, with each episode composed of a 3-way 5-shot support set (15 context samples) and 15 query samples per class (45 queries per episode); episodes are 3-way because the smallest dataset, AGNews, has only four classes. Each dataset is trained for 2,000 such iterations, and 100 evaluation episodes are used at test time. The task-relational buffer holds up to 30,000 samples, shared across all observed datasets. Boundary detection is performed using the additive rule of Section 3.2,, which fires when the sum of the one-sided robust z-scores of the context-only loss and of the query-embedding drift exceeds $k = 6.5$, with a moving-average window of 5 steps and a reference window of 100 steps, and the optimizer is reset upon each detected boundary. Three seeds were used for all reported results.

B.4 Model Architecture for Class-incremental Setting: Vision Datasets

Common Baseline Configs The shared feature extractor for the four-dataset non-stationary setting is a 4-layer convolutional network. Each block applies a 3×3 convolution (64 filters, same padding), followed by Batch Normalization, ReLU activation, Dropout (algorithm specific), and 2×2 max pooling. For 28×28 inputs, four pooling operations yield a 64-dimensional flattened representation. The amortized classifier is identical to the MNIST setting. Detailed hyperparameters shared across all baseline configurations and both dataset orderings are listed below.

- **Input:** $28 \times 28 \times 3$ (RGB, normalised)
- **Feature extractor:** $[\text{Conv}(3\times3, 64) \rightarrow \text{BN} \rightarrow \text{ReLU} \rightarrow \text{Dropout}(\text{Algorithm Specific}) \rightarrow \text{MaxPool}(2\times2)] \times 4 \rightarrow \mathbb{R}^{64}$
- LR scheduler: None
- Episode configuration: N-way $= 5$, k-shot $= 5$, query $= 15$
- Tasks per iteration: 32 episodes for meta-loss calculation
- Iterations per dataset: 2,000 meta-updates per dataset
- Replay buffer: 30,000 samples shared across all observed datasets
- Boundary detection: causal moving average window of 50 steps, threshold $\upsilon = 0.1$
- Dataset order (forward): VGG-Flowers $\rightarrow$ miniImageNet $\rightarrow$ CIFAR-FS $\rightarrow$ Omniglot
- Dataset order (reverse): Omniglot $\rightarrow$ CIFAR-FS $\rightarrow$ miniImageNet $\rightarrow$ VGG-Flowers
- Seeds: 0, 10, 20

Baseline Specific Configs:

FLAME

- **Amortized classifier:** 5-way prototype vectors mapped to classifier weights via four independent 2-layer ELU networks
- Optimizer: AdamW
- lr: 0.0001
- Dropout in Feature Extractor Conv Layer: dropout probability = 0.5

iTAML

- optimizer: SGD
- lr: 0.03
- beta: 1.0
- r: 5 (inner steps)
- lr_inner: 0.1

VC-BML

- optimizer: Adam
- lr: 0.001
- num_components: 4
- num_mc_samples: 5
- lr_inner: "0.05, 0.1, 0.1, 0.1" (forward order)
- nstep_inner: "3, 3, 3, 1" (forward order)
- outer_kl_scale: 1×10^{-5}
- inner_kl_scale: 0.01

ACML

- optimizer: Adam
- lr: 0.001
- num_components: 4
- num_mc_samples: 5
- lr_inner: "0.05, 0.1, 0.1, 0.1" (forward order)
- nstep_inner: "3, 3, 3, 1" (forward order)
- outer_kl_scale: 1×10^{-5}

FTML

- lr: 0.001
- optimizer: Adam
- lr_inner: "0.05, 0.1, 0.1, 0.1" (forward order)
- n_inner_steps: "3, 3, 3, 1" (forward order)
- Parameter count: $113{,}157$

La-MAML

- lr: 0.001
- optimizer: Adam
- lr_alpha: 0.01
- alpha_init: 0.1
- n_inner_steps: 3
- grad_clip: 1.0

HSML

- lr: 0.01
- optimizer: Adam
- lr_inner: "0.001, 0.001, 0.001, 0.001"
- n_inner_steps: 5
- hidden_dim: 128
- n_leaves: 4
- n_mid: 2
- emb_loss_weight: 0.01

B.5 Model Architecture for Class-incremental Setting: Text Datasets

Common Baseline Configs The shared feature extractor for the five-dataset non-stationary text classification setting is a pretrained BERT-base-uncased encoder with 12 Transformer layers and hidden size 768, which is kept frozen throughout training. Its hidden and attention dropout are disabled, so that the encoder is deterministic between the support and query passes of an episode. Inputs are WordPiece token sequences truncated to 256 tokens, and the final-layer [CLS] representation yields a 768-dimensional feature vector. Detailed hyperparameters shared across all baseline configurations and all four dataset orderings are listed below.

- **Input:** WordPiece token ids (bert-base-uncased tokenizer), truncated to 256 tokens
- **Feature extractor:** BERT-base-uncased $(12$ layers, 768-d, frozen, dropout $0) \rightarrow$ [CLS] $\rightarrow \mathbb{R}^{768}$
- LR scheduler: None
- Gradient clipping: maximum norm 2.0
- Episode configuration: N-way $= 3$, k-shot $= 5$, query $= 15$
- Tasks per iteration: 16 episodes for meta-loss calculation
- Iterations per dataset: 2,000 meta-updates per dataset
- Replay buffer: 30,000 samples shared across all observed datasets
- Dataset order 1: Yelp $\rightarrow$ AGNews $\rightarrow$ DBpedia $\rightarrow$ Amazon $\rightarrow$ Yahoo
- Dataset order 2: DBpedia $\rightarrow$ Yahoo $\rightarrow$ AGNews $\rightarrow$ Amazon $\rightarrow$ Yelp
- Seeds: 0, 10, 20

Baseline Specific Configs:

FLAME

- Optimizer: AdamW
- lr: 0.0001
- Dropout in Feature Extractor: none (frozen encoder, dropout probability $= 0$)
- **Amortized classifier:** 3-way prototype vectors mapped to classifier weights via four independent 2-layer ELU networks

FTML

- lr: 0.0001
- optimizer: AdamW
- lr_inner: 0.1 (all datasets)
- n_inner_steps: 1 (all datasets), second-order
- Meta-parameters: MLP head $768 \rightarrow 768 \rightarrow 768 \rightarrow 3$ (ELU) and LoRA with rank $r = 4$ and $\alpha = 1$ on the same projections as FLAME
- Task boundaries: provided (task-aware)

iTAML

- optimizer: SGD
- β: 1.0
- r: 5 (inner steps)
- lr_inner: 0.1

La-MAML

- optimizer: AdamW
- lr: 0.0001
- lr_alpha: 0.01
- alpha_init: 0.1
- n_inner_steps: 3

B.6 LoRA ADAPTATION MECHANISMS OF FLAME ACROSS BACKBONES

LLoRA for the convolutional backbone (vision datasets): The convolutional kernels of the feature extractor are adapted with LLoRA (Ali et al., 2025); For a kernel with base weight $\mathbf{W}_0 \in \mathbb{R}^{C^{\text{out}} \times C^{\text{in}} \times \kappa \times \kappa}$, the hypernetwork maps the dataset embedding $\mathbf{s}^d \in \mathbb{R}^{128}$ to two channel factors and a spatial core, $\{\mathbf{A}^d, \mathbf{B'}^d, \mathbf{M}^d\} = h^f_\varphi(\mathbf{s}^d)$ with $\mathbf{A}^d \in \mathbb{R}^{r \times C^{\text{in}}}$, $\mathbf{B'}^d \in \mathbb{R}^{C^{\text{out}} \times r}$ and $\mathbf{M}^d \in \mathbb{R}^{r \times r \times \kappa \times \kappa}$, which are composed per kernel position (u, v) as

$$\begin{aligned} \big[\mathbf{B}^d\big]_{o,j,u,v} &= act\Big(\sum_{i=1}^{r} \big[\mathbf{B'}^d\big]_{o,i} \big[\mathbf{M}^d\big]_{i,j,u,v}\Big), \\ \big[\mathbf{W}^d\big]_{o,c,u,v} &= \big[\mathbf{W}_0\big]_{o,c,u,v} + \frac{\alpha}{r}\, act\Big(\sum_{j=1}^{r} \big[\mathbf{B}^d\big]_{o,j,u,v} \big[\mathbf{A}^d\big]_{j,c}\Big), \end{aligned} \tag{3}$$

where o and c index output and input channels. With act the identity, as in all our experiments, every kernel position receives its own channel-mixing update $\frac{\alpha}{r}\mathbf{B'}^d\mathbf{M}^d[:,:,u,v]\,\mathbf{A}^d$ of rank at most r: the core $\mathbf{M}^d$ carries the spatial structure of the kernel, while $\mathbf{A}^d$ and $\mathbf{B'}^d$ are shared by all its positions. Here ψ_0 is learned from scratch and shared across datasets rather than frozen; we use $r = 16$ and $\alpha = 4$.

LoRA for the transformer backbone (text datasets): In the BERT encoder, the query, key, value and attention-output projections and the two feed-forward projections of every layer ℓ are adapted; let $\mathbf{W}_{0,\ell,\sigma} \in \mathbb{R}^{n^{\text{out}}_\sigma \times n^{\text{in}}_\sigma}$ denote the base weight of projection $\sigma \in \mathcal{S}$. The same three-factor form is used with a spatially trivial core, i.e. LLoRA with a 1×1 kernel, which reduces to LoRA (Hu et al., 2021) when $\mathbf{M}^d_{\ell,\sigma} = \mathbf{I}_r$ and act is the identity:

$$\mathbf{W}^d_{\ell,\sigma} = \mathbf{W}_{0,\ell,\sigma} + \frac{\alpha}{r}\, act\Big(act\big(\mathbf{B'}^d_{\ell,\sigma}\mathbf{M}^d_{\ell,\sigma}\big)\mathbf{A}^d_{\ell,\sigma}\Big), \qquad \{\mathbf{A}^d_{\ell,\sigma}, \mathbf{B'}^d_{\ell,\sigma}, \mathbf{M}^d_{\ell,\sigma}\}_{\sigma\in\mathcal{S}} = h^f_\varphi\big([\mathbf{e}_\ell; \mathbf{s}^d]\big), \tag{4}$$

with $\mathbf{A}^d_{\ell,\sigma} \in \mathbb{R}^{r \times n^{\text{in}}_\sigma}$, $\mathbf{B'}^d_{\ell,\sigma} \in \mathbb{R}^{n^{\text{out}}_\sigma \times r}$ and $\mathbf{M}^d_{\ell,\sigma} \in \mathbb{R}^{r \times r}$. A hypernetwork h^f_φ generates the factors of every projection of every layer. Its input concatenates a learnable layer embedding $\mathbf{e}_\ell \in \mathbb{R}^{32}$ with a dataset embedding $\mathbf{s}^d \in \mathbb{R}^{32}$ shared by all layers, and one evaluation emits the factors of all projections of layer ℓ at once, so $\mathbf{e}_\ell$ alone distinguishes the layers. The pretrained weights ψ_0 stay frozen, and the generated updates are the only dataset-specific change to the encoder; we use $r = 4$ and $\alpha = 1$.

C Additional Results on Main Experiments in Sections 4.1-4.2

C.1 Latent Task Clustering

Fig. 4 and Fig. 5 present the TSNE embedding visualization of context embeddings of boundaries 0-9 and 10-18, respectively, for recurring and non-recurring task sequences on the MNIST-Rotations dataset. Similar to previous Fig. 2B, here we can see that the GMM clustering based reservoir can recognize recurring tasks except some exception. Also, clustering based reservoir can assign some different tasks to same cluster if they are close together in the context embedding space (top row).

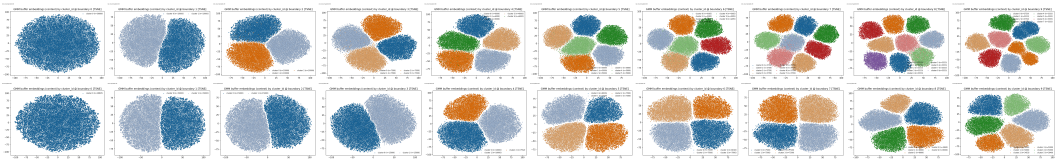

Figure 4: TSNE visualization of reservoir task embedding and the associated cluster ID (by color), for non-recurring (top) vs recurring (bottom) task sequence for task boundary 0-9 for MNIST-Rotations by FLAME.

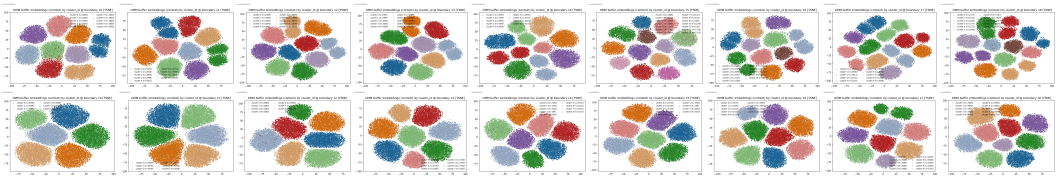

Figure 5: TSNE visualization of reservoir task embedding and the associated cluster ID (by color), for non-recurring (top) vs recurring (bottom) task sequence for task boundary 10-18 for MNIST-Rotations by FLAME.

C.2 Additional Results for Recurring Task Settings

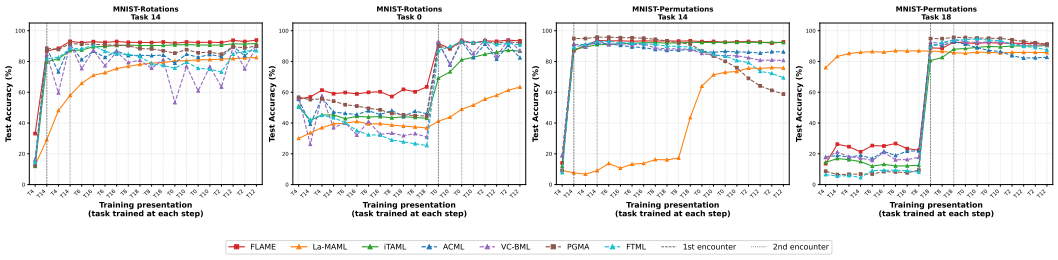

Figure 6: Test accuracy of selected tasks (dashed line: first encounter; dotted line: recurrence).

Fig. 6 expands the single-task view of Fig. 2A to four representative tasks, two per dataset, selected to contrast a task introduced early in the sequence with one introduced late; each panel traces the test accuracy of one original task across all 20 presentations, averaged over three seeds, with the vertical dashed and dotted lines marking the presentations at which that task is trained for the first and second time. Under our schedule the **MNIST-Rotations** panels show Task 14 (first trained at presentation 2) and Task 0 (presentation 13), and the **MNIST-Permutations** panels show Task 14 (presentation 2) and Task 18 (presentation 10). The FLAME–iTAML gap at the moment a task is first learned widens with the task's position in the sequence: on **MNIST-Rotations** it grows from 7.0 points on Task 14 (86.8% vs. 79.8%) to 22.7 points on Task 0 (91.8% vs. 69.1%), and on **MNIST-Permutations** from -1.1 points on Task 14 (86.9% vs. 88.0%, the one case in which iTAML is ahead) to 8.3 points on Task 18 (88.9% vs. 80.6%).

C.3 ADDITIONAL RESULT FOR CLASS-INCREMENTAL SETTING ON IMAGE DATASETS

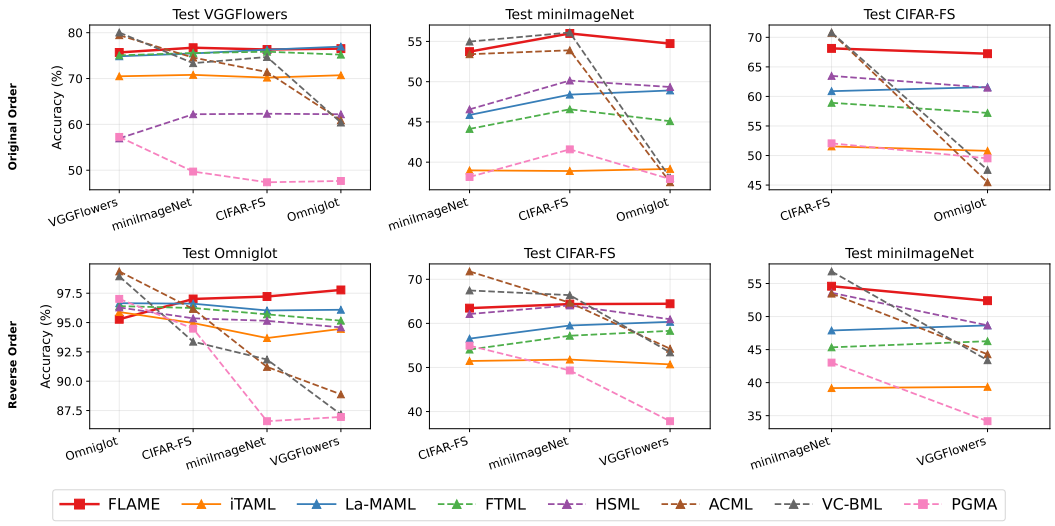

Figure 7: Changes of test accuracies on the *first three* datasets along the streaming task sequence.

Tables 7-8 list the complete test performance of all baselines alongside FLAME as the learning proceeds along the two task sequences consisting of the four datasets. Fig. 7 shows each baseline's test accuracy changes on the first three dataset boundaries along the streaming sequence.

Table 7: Performance of our FLAME and the baselines on each dataset at each meta-training stage.

Meta-Training Stage	Algorithms	VGG-Flowers	miniImageNet	CIFAR-FS	Omniglot	Average
VGG-Flowers	FTML △	75.06 ± 0.72	–	–	–	75.06 ± 0.72
	HSML △	56.94 ± 1.88	–	–	–	56.94 ± 1.88
	ACML △	79.45 ± 2.17	–	–	–	79.45 ± 2.17
	VC-BML △	80.04 ± 0.07	–	–	–	80.04 ± 0.07
	iTAML ▲	70.49 ± 0.69	–	–	–	70.49 ± 0.69
	La-MAML ▲	74.85 ± 0.55	–	–	–	74.85 ± 0.55
	PGMA □	57.27 ± 1.09	–	–	–	57.27 ± 1.09
	CML-BGNN □	82.72 ± 4.17	–	–	–	82.72 ± 4.17
	FLAME ■	75.69 ± 1.53	–	–	–	75.69 ± 1.53
miniImageNet	FTML △	75.56 ± 0.44	44.13 ± 0.53	–	–	59.84 ± 0.48
	HSML △	62.21 ± 3.45	46.56 ± 0.73	–	–	54.39 ± 2.09
	ACML △	74.57 ± 0.69	53.41 ± 0.14	–	–	63.99 ± 0.41
	VC-BML △	73.35 ± 0.40	54.98 ± 0.37	–	–	64.17 ± 0.39
	iTAML ▲	70.80 ± 0.92	38.98 ± 0.45	–	–	54.89 ± 0.68
	La-MAML ▲	75.47 ± 0.46	45.86 ± 0.42	–	–	60.66 ± 0.44
	PGMA □	49.71 ± 2.11	38.16 ± 0.57	–	–	43.93 ± 1.34
	CML-BGNN □	29.13 ± 10.90	30.60 ± 15.97	–	–	29.87 ± 13.44
	FLAME ■	76.72 ± 0.42	53.73 ± 1.04	–	–	65.22 ± 0.73
CIFAR-FS	FTML △	75.85 ± 0.43	46.57 ± 0.26	58.92 ± 0.62	–	60.45 ± 0.44
	HSML △	62.32 ± 1.46	50.14 ± 1.12	63.48 ± 1.28	–	58.65 ± 1.29
	ACML △	71.42 ± 2.57	53.90 ± 1.64	70.77 ± 1.25	–	65.36 ± 1.82
	VC-BML △	74.70 ± 0.27	56.11 ± 0.36	70.82 ± 0.95	–	67.21 ± 0.53
	iTAML ▲	70.20 ± 2.27	38.90 ± 1.71	51.54 ± 2.14	–	53.55 ± 2.04
	La-MAML ▲	76.29 ± 0.16	48.39 ± 0.84	60.89 ± 0.44	–	61.86 ± 0.48
	PGMA □	47.37 ± 2.45	41.59 ± 0.74	52.06 ± 0.82	–	47.01 ± 1.34
	CML-BGNN □	27.85 ± 6.57	30.38 ± 10.18	32.33 ± 7.10	–	30.19 ± 7.95
	FLAME ■	76.33 ± 1.32	55.99 ± 0.68	68.13 ± 0.95	–	66.82 ± 0.98
Omniglot	FTML △	75.21 ± 0.17	45.09 ± 0.87	57.21 ± 0.13	95.51 ± 0.11	68.26 ± 0.32
	HSML △	62.21 ± 0.86	49.34 ± 2.08	61.48 ± 1.57	93.86 ± 2.31	66.72 ± 1.71
	ACML △	60.89 ± 1.84	37.50 ± 0.56	45.47 ± 0.62	99.56 ± 0.06	60.85 ± 0.77
	VC-BML △	60.40 ± 4.32	38.06 ± 1.23	47.57 ± 2.15	97.38 ± 1.02	60.85 ± 2.18
	iTAML ▲	70.72 ± 2.96	39.16 ± 1.47	50.79 ± 2.14	94.97 ± 0.33	63.91 ± 1.73
	La-MAML ▲	76.96 ± 0.57	48.91 ± 0.95	61.58 ± 0.33	96.42 ± 0.12	70.97 ± 0.49
	PGMA □	47.66 ± 1.54	37.90 ± 1.48	49.52 ± 2.96	94.90 ± 0.45	57.49 ± 1.60
	CML-BGNN □	24.27 ± 2.77	23.60 ± 1.85	23.76 ± 2.68	32.03 ± 5.63	25.92 ± 3.23
	FLAME ■	76.52 ± 0.48	54.73 ± 1.40	67.23 ± 0.30	97.98 ± 0.30	74.12 ± 0.62

Table 8: Performance of our FLAME and the baselines on each dataset at each meta-training stage of the reverse order of the four datasets.

Meta-Training Stage	Algorithms	Omniglot	CIFAR-FS	miniImageNet	VGG-Flowers	Average
Omniglot	FTML △	96.40 ± 0.21	–	–	–	96.40 ± 0.21
	HSML △	96.29 ± 0.62	–	–	–	96.29 ± 0.62
	ACML △	99.36 ± 0.04	–	–	–	99.36 ± 0.04
	VC-BML △	98.94 ± 0.12	–	–	–	98.94 ± 0.12
	iTAML ▲	95.90 ± 0.12	–	–	–	95.90 ± 0.12
	La-MAML ▲	96.65 ± 0.23	–	–	–	96.65 ± 0.23
	PGMA □	97.02 ± 0.69	–	–	–	97.02 ± 0.69
	CML-BGNN □	79.62 ± 18.14	–	–	–	79.62 ± 18.14
	FLAME ■	95.28 ± 0.23	–	–	–	95.28 ± 0.23
CIFAR-FS	FTML △	96.24 ± 0.40	54.06 ± 1.22	–	–	75.15 ± 0.81
	HSML △	95.36 ± 0.54	62.12 ± 0.76	–	–	78.74 ± 0.65
	ACML △	96.16 ± 0.12	71.79 ± 2.19	–	–	83.97 ± 1.16
	VC-BML △	93.37 ± 0.41	67.47 ± 1.55	–	–	80.42 ± 0.98
	iTAML ▲	94.96 ± 0.34	51.47 ± 0.69	–	–	73.21 ± 0.51
	La-MAML ▲	96.61 ± 0.14	56.54 ± 0.73	–	–	76.58 ± 0.44
	PGMA □	94.48 ± 0.35	54.95 ± 0.28	–	–	74.72 ± 0.31
	CML-BGNN □	36.63 ± 5.76	33.80 ± 12.74	–	–	35.22 ± 9.25
	FLAME ■	97.01 ± 0.36	63.46 ± 0.62	–	–	80.23 ± 0.49
miniImageNet	FTML △	95.70 ± 0.45	57.20 ± 0.63	45.34 ± 0.70	–	66.08 ± 0.59
	HSML △	95.15 ± 0.12	64.07 ± 0.95	53.60 ± 0.41	–	70.94 ± 0.49
	ACML △	91.22 ± 1.12	64.70 ± 1.29	53.47 ± 0.75	–	69.80 ± 1.05
	VC-BML △	91.85 ± 1.74	66.41 ± 0.06	56.83 ± 0.43	–	71.69 ± 0.74
	iTAML ▲	93.68 ± 0.19	51.79 ± 1.00	39.17 ± 0.94	–	61.55 ± 0.71
	La-MAML ▲	96.03 ± 0.29	59.52 ± 0.25	47.89 ± 0.64	–	67.81 ± 0.39
	PGMA □	86.59 ± 0.67	49.32 ± 0.44	43.03 ± 0.19	–	59.65 ± 0.43
	CML-BGNN □	34.78 ± 5.07	35.52 ± 10.43	36.66 ± 9.38	–	35.66 ± 8.29
	FLAME ■	97.22 ± 0.15	64.38 ± 1.14	54.59 ± 1.45	–	72.06 ± 0.91
VGG-Flowers	FTML △	95.16 ± 0.25	58.31 ± 1.02	46.27 ± 0.58	76.01 ± 0.28	68.94 ± 0.53
	HSML △	94.59 ± 0.65	60.94 ± 0.83	48.67 ± 0.39	64.31 ± 0.05	67.13 ± 0.48
	ACML △	88.88 ± 1.31	54.25 ± 1.15	44.29 ± 0.15	77.91 ± 0.58	66.33 ± 0.80
	VC-BML △	87.17 ± 2.20	53.42 ± 1.88	43.39 ± 0.02	76.77 ± 0.74	65.18 ± 1.21
	iTAML ▲	94.46 ± 0.35	50.70 ± 0.79	39.36 ± 0.53	64.77 ± 0.38	62.33 ± 0.51
	La-MAML ▲	96.09 ± 0.16	60.35 ± 0.83	48.66 ± 0.48	76.91 ± 0.31	70.50 ± 0.44
	PGMA □	86.95 ± 0.94	37.81 ± 0.30	34.16 ± 0.85	55.23 ± 0.51	53.54 ± 0.65
	CML-BGNN □	33.93 ± 8.22	23.41 ± 4.28	23.94 ± 5.25	23.76 ± 6.75	26.26 ± 6.13
	FLAME ■	97.78 ± 0.09	64.45 ± 4.91	52.40 ± 3.68	76.02 ± 0.40	72.66 ± 2.27

C.4 Larger Backbone on Class-Incremental Image Datasets

To investigate the scalability of FLAME, we further tested replacing the 4-conv-layer backbone with the VIT-S/16 network from the DINOv3 family. We kept all of the 21M parameters of the backbone network frozen while generating LoRA parameters from dataset embedding and layer-specific embedding using a hypernetwork similar to our class-incremental text setting presented in B.6. We then ran the non-stationary class-incremental experiment with VIT-S/16 as backbone in FLAME with oracle task boundary information. Table 9 shows the comparison between final accuracy on each dataset of four-dataset CIL setting without and with utilization of LoRA adaptation. Each accuracy column is those dataset's 5-way 5-shot test accuracy after the whole stream has been trained on; RA is their mean. Both rows are evaluated from the same end-of-training checkpoint on identical episodes, the ablation differing only in whether the generated ΔW is applied to the backbone's attention/MLP projections. Mean $\pm$ std over seeds 0, 10, 20.

Table 9: End-of-training accuracy of FLAME with the frozen DINOv3 ViT-S/16 backbone on the four-dataset non-stationary stream, with and without its shared hypernetwork-generated LoRA adaptation.

	Accuracy after training [%]				
Method	VGG-Flowers	miniImageNet	CIFAR-FS	Omniglot	RA [%]
DinoV3 backbone w/o LoRA adaptation	93.82 ± 0.42	73.21 ± 0.61	85.53 ± 0.37	90.94 ± 2.07	85.87 ± 0.78
DinoV3 backbone with LoRA adaptation	$\mathbf{94.74 \pm 1.13}$	$\mathbf{84.15 \pm 0.50}$	$\mathbf{93.65 \pm 0.44}$	$\mathbf{99.36 \pm 0.04}$	$\mathbf{92.98 \pm 0.25}$

C.5 Additional results in class-incremental setting on text datasets

Tables 10-11 list the complete test performance of all baselines alongside FLAME as the learning proceeds along the two task sequences consisting of the five datasets.

Table 10: Performance of our FLAME and the baselines on each dataset at each meta-training stage of Order 1 of the five-dataset text classification stream (Yelp $\rightarrow$ AGNews $\rightarrow$ DBpedia $\rightarrow$ Amazon $\rightarrow$ Yahoo).

Meta-Training Stage	Algorithms	Yelp	AGNews	DBpedia	Amazon	Yahoo	Average
Yelp	FTML △	75.39 ± 0.41	–	–	–	–	75.39 ± 0.41
	iTAML ▲	45.61 ± 1.12	–	–	–	–	45.61 ± 1.12
	La-MAML ▲	72.65 ± 1.15	–	–	–	–	72.65 ± 1.15
	FLAME ■	68.13 ± 0.46	–	–	–	–	68.13 ± 0.46
AGNews	FTML △	72.02 ± 1.14	94.53 ± 0.25	–	–	–	83.27 ± 0.69
	iTAML ▲	45.92 ± 1.08	69.04 ± 0.54	–	–	–	57.48 ± 0.81
	La-MAML ▲	71.08 ± 0.57	94.16 ± 0.22	–	–	–	82.62 ± 0.39
	FLAME ■	72.44 ± 0.69	95.18 ± 0.08	–	–	–	83.81 ± 0.39
DBpedia	FTML △	69.94 ± 0.63	93.67 ± 0.39	99.60 ± 0.02	–	–	87.74 ± 0.35
	iTAML ▲	50.03 ± 1.87	75.78 ± 1.09	96.22 ± 1.05	–	–	74.01 ± 1.34
	La-MAML ▲	68.69 ± 1.26	92.98 ± 0.26	99.59 ± 0.15	–	–	87.09 ± 0.55
	FLAME ■	70.33 ± 0.27	93.93 ± 0.77	99.75 ± 0.12	–	–	88.00 ± 0.39
Amazon	FTML △	71.07 ± 1.27	93.08 ± 0.30	99.67 ± 0.06	69.81 ± 1.07	–	83.41 ± 0.67
	iTAML ▲	46.90 ± 0.80	72.19 ± 0.70	94.04 ± 0.73	43.95 ± 0.21	–	64.27 ± 0.61
	La-MAML ▲	71.34 ± 0.86	92.27 ± 0.16	99.61 ± 0.11	70.55 ± 0.89	–	83.44 ± 0.50
	FLAME ■	71.12 ± 0.21	93.46 ± 0.12	99.61 ± 0.18	71.68 ± 0.97	–	83.97 ± 0.37
Yahoo	FTML △	70.32 ± 1.86	93.12 ± 0.45	99.52 ± 0.24	66.55 ± 1.48	84.79 ± 0.47	82.86 ± 0.90
	iTAML ▲	46.78 ± 0.79	72.48 ± 1.09	94.28 ± 0.12	43.75 ± 0.61	61.86 ± 0.62	63.83 ± 0.65
	La-MAML ▲	67.92 ± 1.02	92.46 ± 0.38	99.57 ± 0.16	66.98 ± 0.22	86.02 ± 0.43	82.59 ± 0.44
	FLAME ■	71.74 ± 1.55	93.51 ± 0.57	99.62 ± 0.13	68.96 ± 0.91	87.22 ± 0.66	84.21 ± 0.76

Table 11: Performance of our FLAME and the baselines on each dataset at each meta-training stage of Order 2 of the five-dataset text classification stream (DBpedia → Yahoo → AGNews → Amazon → Yelp).

Meta-Training Stage	Algorithms	DBpedia	Yahoo	AGNews	Amazon	Yelp	Average
DBpedia	FTML △	99.77 ± 0.15	–	–	–	–	99.77 ± 0.15
	iTAML ▲	99.68 ± 0.08	–	–	–	–	99.68 ± 0.08
	La-MAML ▲	99.69 ± 0.13	–	–	–	–	99.69 ± 0.13
	FLAME ■	99.62 ± 0.14	–	–	–	–	99.62 ± 0.14
Yahoo	FTML △	99.79 ± 0.09	87.74 ± 0.56	–	–	–	93.76 ± 0.33
	iTAML ▲	98.75 ± 0.05	71.11 ± 1.30	–	–	–	84.93 ± 0.67
	La-MAML ▲	99.70 ± 0.10	86.61 ± 0.46	–	–	–	93.15 ± 0.28
	FLAME ■	99.70 ± 0.02	87.61 ± 0.64	–	–	–	93.65 ± 0.33
AGNews	FTML △	99.70 ± 0.11	85.41 ± 0.90	94.16 ± 0.55	–	–	93.09 ± 0.52
	iTAML ▲	98.79 ± 0.37	70.62 ± 0.39	76.52 ± 0.24	–	–	81.98 ± 0.33
	La-MAML ▲	99.63 ± 0.18	84.22 ± 0.28	93.89 ± 0.41	–	–	92.58 ± 0.29
	FLAME ■	99.75 ± 0.27	86.18 ± 0.12	94.39 ± 0.33	–	–	93.44 ± 0.24
Amazon	FTML △	99.58 ± 0.05	83.88 ± 0.90	93.23 ± 0.43	69.02 ± 1.00	–	86.43 ± 0.59
	iTAML ▲	96.31 ± 0.48	67.92 ± 0.39	75.13 ± 1.11	43.81 ± 1.99	–	70.79 ± 0.99
	La-MAML ▲	99.41 ± 0.18	85.01 ± 0.36	92.96 ± 0.33	70.17 ± 0.56	–	86.89 ± 0.36
	FLAME ■	99.67 ± 0.16	86.32 ± 1.23	93.85 ± 0.27	72.41 ± 0.37	–	88.06 ± 0.51
Yelp	FTML △	99.64 ± 0.09	82.29 ± 1.27	92.56 ± 0.68	68.26 ± 1.11	70.93 ± 0.34	82.73 ± 0.70
	iTAML ▲	96.81 ± 0.42	66.75 ± 1.43	74.76 ± 1.45	43.47 ± 1.07	46.26 ± 0.29	65.61 ± 0.93
	La-MAML ▲	99.54 ± 0.05	82.78 ± 1.71	92.46 ± 0.73	69.59 ± 1.90	71.94 ± 1.88	83.26 ± 1.26
	FLAME ■	99.73 ± 0.13	85.10 ± 1.75	93.42 ± 0.42	71.80 ± 0.73	74.71 ± 0.53	84.95 ± 0.71

C.6 Task Boundary Detection & Clustering in Main Experimental Results

C.6.1 Boundary Detection Results

Oracle baseline comparison: Table 12 shows the performance of FLAME with oracle task boundary information on non-recurring sequence of MNIST-Rotations dataset. The table also shows FLAME with the deployed task detection method closely follows the performance of the oracle counterpart.

Table 12: FLAME on MNIST-Rotations with oracle task boundaries versus boundary detection.

Method	RA	LA	BTI
FLAME (Oracle Boundary)	95.08 ± 0.09	95.07 ± 0.03	0.00 ± 0.07
FLAME (Boundary Detection)	94.94 ± 0.14	94.97 ± 0.13	-0.02 ± 0.19

Domain-incremental setting: Fig. 8 traces the two channels of the boundary detection rule of Section 3.2 and their sum along the task sequence on MNIST-Rotations (FLAME, non-recurring), with $\kappa = 6.5$, $m = 5$ and $R = 100$. The top panel shows the loss channel: the positive part of the residual $r_l = L_l - \bar{L}_{l-1:l-m}$, its robust scale $1.4826\,\mathrm{MAD}(r)$ over the trailing R residuals, and the resulting score $(z_l^{\mathrm{loss}})^+$. The middle panel shows the embedding channel: the root-mean-square whitened deviation e_l, its robust scale, and $(z_l^{\mathrm{emb}})^+$. The bottom panel shows $S_l = (z_l^{\mathrm{loss}})^+ + (z_l^{\mathrm{emb}})^+$. Both scores are colored by the true task identity, and dotted vertical lines mark the true task boundaries. Because the context set is the most recent $N \times K$ samples, the first batch of a new task is conditioned on context from the previous task, so both channels jump at every boundary. Within a task, S_l stayed at a few robust standard deviations, whereas it rose above κ at every one of the 19 boundaries. Assuming local stationarity, the rule was consulted only after the fixed local stationarity window following each detected boundary, and under this schedule all 19 boundaries were detected with no false positives and no delay. Each boundary appeared as an isolated spike, since the rule stayed silent for the next m batches while it re-estimated the location of its embedding channel.

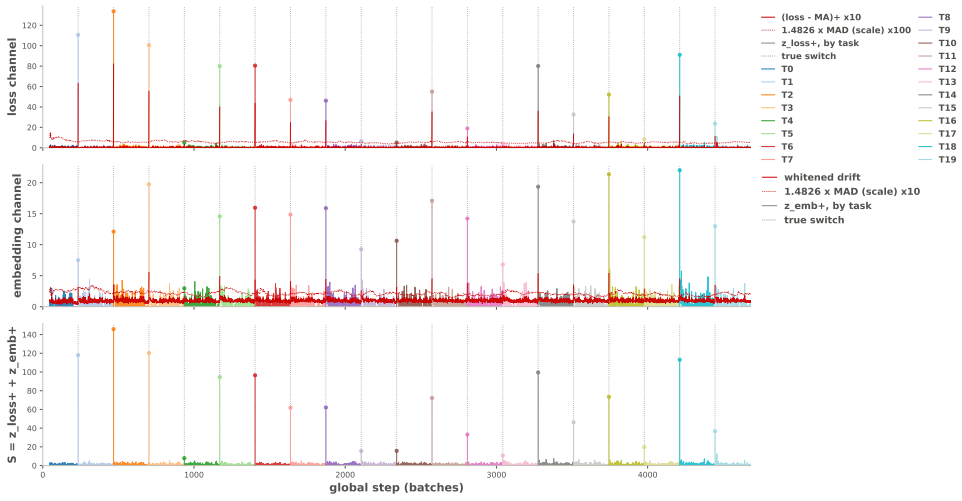

Figure 8: Boundary detection in the domain-incremental setting (MNIST-Rotations, FLAME). **Top:** loss channel, $(L_l - \bar{L}_{l-1:l-m})^+$ ($\times 10$), its robust scale ($\times 100$), and $(z_l^{\mathrm{loss}})^+$. **Middle:** embedding channel, e_l, its robust scale ($\times 10$), and $(z_l^{\mathrm{emb}})^+$. **Bottom:** the detection statistic S_l. Scores are colored by the true task; dotted lines mark the true boundaries.

Class-incremental setting: We treated each dataset as one task and applied the same rule, with the same $\kappa = 6.5$, $m = 5$ and $R = 100$ and no retuning. Fig. 9 shows the two channels and S_l for the forward stream of four vision datasets (FLAME), in the same layout as Fig. 8. Here the loss

channel alone was not reliable. When the incoming dataset was no harder than the outgoing one, the mismatch between context and query barely raised the loss, and at the miniImageNet → CIFAR-FS and CIFAR-FS → Omniglot boundaries the loss channel alone would not have cleared κ. The embedding channel carried these boundaries instead, while at the VGG-Flowers → miniImageNet boundary both channels responded. Within a dataset, S_l stayed below κ on nearly every batch, and with the rule consulted only after a fixed stationarity window following each detected boundary, all three boundaries were detected with no false positives and no delay. This was the case the additive form was designed for: a dataset shift that left one channel silent was still detected through the other.



Figure 9: Boundary detection in the class-incremental setting (four vision datasets, forward order, FLAME). Panels as in Fig. 8: loss channel (top), embedding channel (middle), and the detection statistic S_l (bottom), colored by the true dataset, with dotted lines marking the true boundaries.

C.6.2 Latent Clustering Results

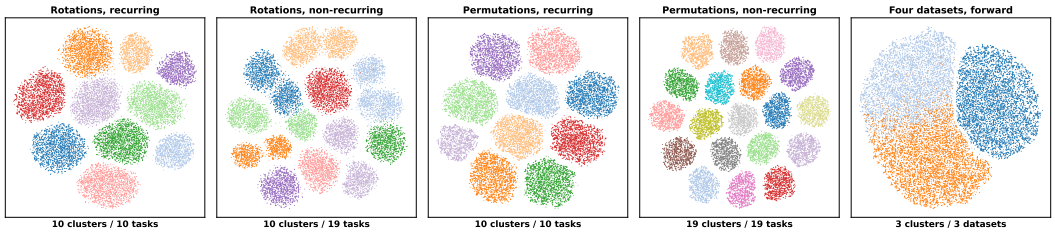

Figure 10: t-SNE of the replay-buffer context embeddings colored by GMM cluster at the last task boundary, with boundaries detected online by the deployed deetction method. Each panel reports its number of clusters and of original tasks (datasets in the four-dataset setting, whose last-boundary buffer holds VGG-Flowers, Mini-ImageNet and CIFAR-FS). Table 13 gives mean ± std over three seeds.

Figure 10 and Table 13 evaluate the pseudo-labels that GMM clustering assigned to the replay buffer at the last task boundary, with boundaries detected online by the additive rule 3.2, on MNIST-Rotations and MNIST-Permutations (recurring and non-recurring streams) and on the four-dataset setting (forward order). We scored clusters against the original task identity, so both presentations of a recurring task counted as one task. Micro-purity (MP) falls below 1 when a cluster merges several tasks, inverse micro-purity (IMP) falls below 1 when a task is split across clusters, and context contamination (CC) is the probability that a context sample drawn from a query's cluster belongs to a different task than the query.

How cleanly the GMM separated tasks depended on how related they were. On MNIST-Permutations, whose tasks are unrelated, it found exactly one cluster per task in every seed of both streams, and placed both presentations of a recurring task in the same cluster. In the four-dataset setting, it likewise recovered one cluster per dataset, and its few errors came only from the two general object-recognition datasets, Mini-ImageNet and CIFAR-FS, which exchanged a small fraction of their samples.

Table 13: Cluster quality of the replay buffer at the last task boundary under the deployed detection rule, scored against the original task identity (both presentations of a recurring task count as one task).

Stream	Rotations			Permutations			Four datasets		
	MP	IMP	CC	MP	IMP	CC	MP	IMP	CC
Recurring	0.9423 ± 0.0475	0.9962 ± 0.0052	0.0633 ± 0.0542	1.0000 ± 0.0000	1.0000 ± 0.0000	0.0000 ± 0.0000	–	–	–
Non-recurring	0.5976 ± 0.0985	0.9997 ± 0.0002	0.4562 ± 0.1271	1.0000 ± 0.0000	1.0000 ± 0.0000	0.0000 ± 0.0000	0.9604 ± 0.0021	0.9604 ± 0.0021	0.0744 ± 0.0037

When tasks were similar, the GMM merged neighboring tasks but almost never split one. On MNIST-Rotations, every merged cluster covered a contiguous range of nearby rotation angles, and merging was heavier in the non-recurring stream, whose rotations lie about half as far apart as those of the recurring stream.

These merges carried no measurable accuracy cost, since context from a rotation a few degrees away remained informative about the query's task. Consistently, across the GMM hyperparameter grid of Appendix D.2, cluster quality on MNIST-Rotations ranged from heavy merging to near-perfect separation while RA barely changed.

C.6.3 Effect of rare tasks

To further test the importance of knowledge consolidation, we further set up a stream of tasks with frequently *vs*. rarely recurring tasks. We chose four tasks from MNIST-Rotations and split each into 13 even sub-tasks. We presented to FLAME four consecutive splits of task 0 (rare task), followed by the rest three tasks in intermittent fashion through 24 task boundaries (frequently recurring tasks). We compare FLAME with a simpler variant, FLAME-B, which assigns each replay-buffer sample a pseudo task label using simply corresponding detected boundary counter, instead of the GMM clustering of Section 3.2. In both training, we used a total buffer of 10000.

Fig. 11 shows that, with FLAME-B, the test accuracy on the rare task was not able to improve and started to decrease as the other tasks frequently recurred. In comparison, with FLAME, the performance on the rare task was able to continue to increase even though it has not been revisited over frequently-switching task boundaries.

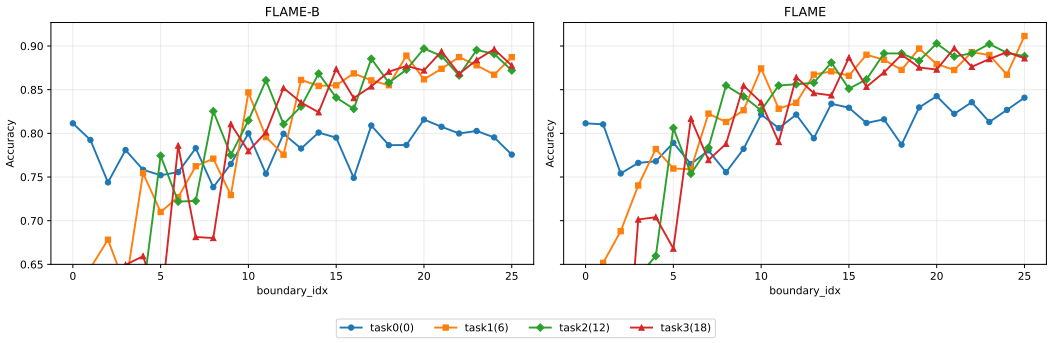

Figure 11: FLAME-B (boundary detection based) vs FLAME (GMM clustering based) buffer replay on rare task sequence where the rare task (denoted by the blue line) appeared once while other tasks appeared in a cyclic fashion.

D Additional Results on Stress-Testing Task Inference in Section 4.4

D.1 Sensitivity to Boundary Detection Hyperparameters

We swept the three hyperparameters of the boundary detection rule of Section 3.2 over a full grid of 60 configurations, $\kappa \in \{4, 5, 6.5, 8, 10\}$, $m \in \{5, 10, 25\}$ and $R \in \{50, 75, 100, 125\}$, training FLAME-B (Appendix C.6.3) separately for each configuration on the non-recurring MNIST-Rotations and MNIST-Permutations streams (seed 20). Table 14 reports, for each metric, the share of its variance explained by each hyperparameter (η^2) and by their joint effect (Jt), together with its mean and standard deviation over the grid.

Detection quality: Boundary detection was nearly exact across the grid. On MNIST-Permutations, every configuration detected all boundaries with no false positives. On MNIST-Rotations, all but two configurations did, and those two, at the most conservative settings, each missed only one boundary between two very similar rotations.

Learning outcome: The accuracy metrics varied very little across the grid, and the variation followed no single hyperparameter in a consistent direction; even the two configurations that missed a boundary kept an RA at the grid mean.

The default setting ($\kappa = 6.5$, $m = 5$, $R = 100$) therefore sat within a broad plateau, and the accuracy of FLAME-B was insensitive to the boundary detection hyperparameters over these ranges on both datasets.

Table 14: Task-boundary hyperparameter sensitivity (η^2, detection quality vs. learning outcome) and metric summary for MNIST-Rotations and MNIST-Permutations.

	Detection quality							Learning outcome						
	Sensitivity (η^2, %)				Metrics (mean ± std)			Sensitivity (η^2, %)				Metrics (mean ± std, %)		
Dataset	κ	M	R	Jt	P	R	F1	κ	M	R	Jt	RA	LA	BTI
Rotations	13.8	6.9	3.5	75.8	1.0000±0.0000	0.9982±0.0094	0.9991±0.0049	15.2	8.5	2.7	73.6	94.49±0.19	95.09±0.08	−0.60±0.22
Permutations	–	–	–	–	1.0000±0.0000	1.0000±0.0000	1.0000±0.0000	14.1	0.0	3.9	82.0	88.70±0.06	92.15±0.05	−3.44±0.01

D.2 Sensitivity to Clustering Hyperparameters

Our clustering fits a non-decreasing number of mixture components at each task boundary, so the number of components is not a hyperparameter. We swept the remaining hyperparameters of scikit-learn's `BayesianGaussianMixture`, namely $weight_concentration_prior \in \{\text{None}, 0.01, 0.05\}$, $n_init \in \{1, 3, 5\}$ and $max_iter \in \{100, 1000\}$, together with our threshold for pruning inactive components, $prune_threshold \in \{0.01, 0.05\}$: a full grid of 36 configurations on MNIST-Rotations and MNIST-Permutations. For each configuration we record the RA, LA and BTI. We evaluate cluster quality at every boundary the model trains with, i.e., after each of the first 19 tasks; the fit after the final task no longer influences training. Table 15 reports the share of variance explained by each hyperparameter (η^2) and the mean ± std of each metric over the grid. For cluster quality, η^2 is pooled over the training boundaries by accumulating sums of squares across them, so each boundary contributes in proportion to its variance, and each metric is first averaged over the training boundaries of a run.

The learning outcome of FLAME was insensitive to these hyperparameters, even where the clustering was not. On MNIST-Rotations, the clustering ranged from heavy merging to near-perfect separation across the grid, depending mainly on the threshold for pruning inactive components, and merging increased as the buffer accumulated closely spaced rotations. The accuracy metrics nevertheless barely changed, and their small differences did not follow cluster quality (Table 15): the merged rotations were too similar to warrant separate model adaptation.

Clustering was robust when tasks were distinct. On MNIST-Permutations, whose tasks are unrelated, it was near-perfect at every boundary in every configuration, and the accuracy metrics again varied little.

In both settings, the full grid changes RA by at most 0.56 points and LA by at most 0.12 points, and hence they seem to be fairly insensitive to these hyperparameters.

Table 15: GMM hyperparameter sensitivity (η^2, cluster quality vs. learning outcome) and metric summary (mean $\pm$ std over the 36-point grid) for MNIST-Rotations and MNIST-Permutations. Cluster quality is evaluated at every boundary the model trains with (after tasks 1–19 of 20): its η^2 is pooled over these boundaries

	Cluster quality									Learning outcome							
	Sensitivity (η^2, %)					Metrics (mean ± std)				Sensitivity (η^2, %)					Metrics (mean ± std, %)		
Dataset	PT	NI	WCP	MI	Jt	CR	MP	IMP	CC	PT	NI	WCP	MI	Jt	RA	LA	BTI
Rotations	57.6	16.4	1.4	1.5	23.1	0.8979±0.0534	0.9156±0.0350	0.9942±0.0064	0.0909±0.0383	2.1	44.6	3.9	2.2	47.2	95.15±0.14	95.05±0.04	0.09±0.12
Permutations	0.0	100.0	0.0	0.0	0.0	0.9991±0.0013	0.9995±0.0007	1.0000±0.0000	0.0005±0.0007	0.0	100.0	0.0	0.0	0.0	89.55±0.12	92.03±0.02	−2.48±0.14

D.3 Additional Results on Stress-Testing α-Stationarity

D.4 Additional Results on Stress-Testing α-Stationarity

Fig. 12 shows the detection statistic S_l of FLAME over the four-task stream at each value of α in Fig. 3. With long stationary windows, every task switch produced a spike far above the threshold κ and was caught without false positives. As the windows shortened, the spikes shrank and more switches fell below κ, while some also arrived when the rule was silent, during its warm-up or right after a detection. The spikes shrank not because the scales of the rule changed, but because a switch raised the context-conditioned loss less. When every task recurred within a few batches, the model never specialised to the current one, so switching to another cost little. This was the regime in which missed boundaries barely affected performance (Fig. 3A), as the stream approached a stationary mixture of the four tasks.

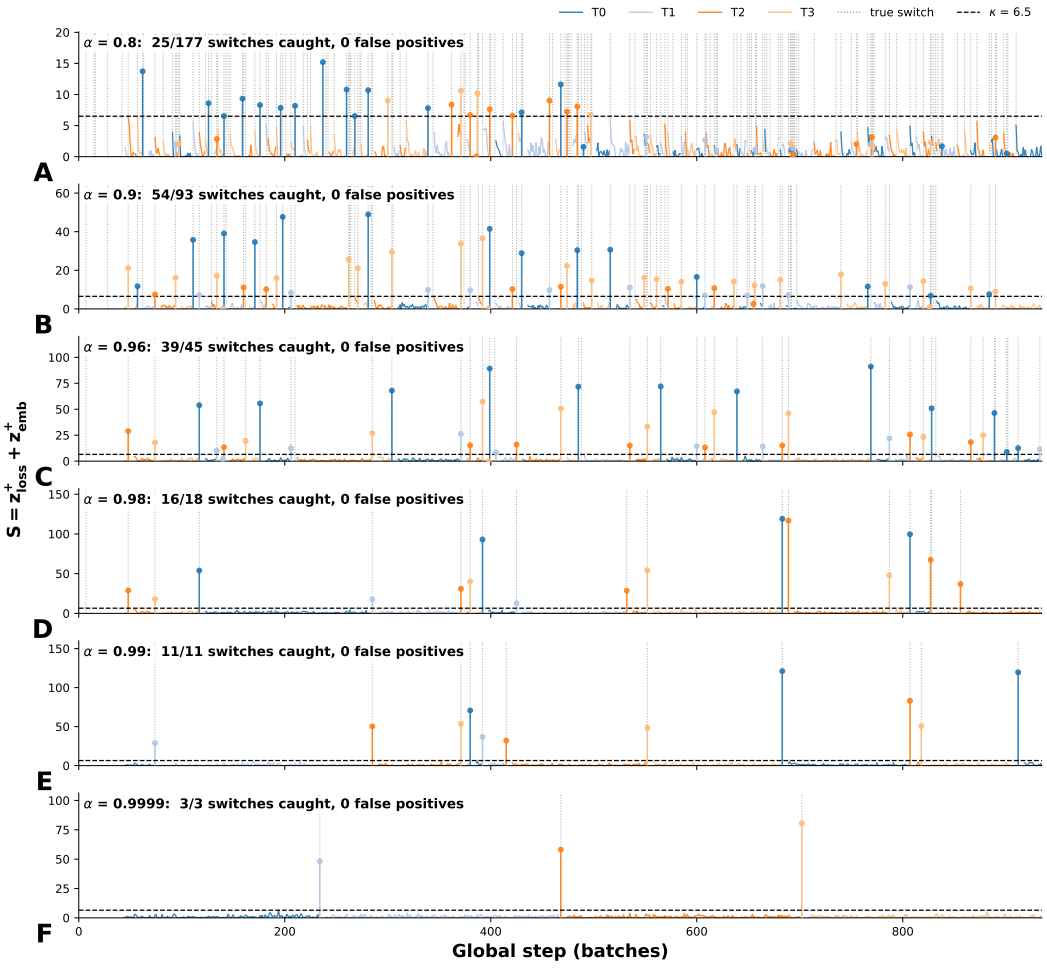

Figure 12: Boundary detection signal S_l of FLAME over the task stream at representative values of α (MNIST-Rotations, seed 20). Each row shows S_l at every batch, colored by the true task, with true task switches dotted and the threshold κ dashed; row labels give the switches caught and false positives. Averages across seeds are in Fig. 3.

D.5 Task Boundary Detection Accuracy for Gradual Boundaries

A gradual transition dilutes both channels of the detection statistic S_l (Section 3.2): when only part of a batch comes from the new task, the context-conditioned loss rises less and the query-embedding mean moves less. We probed this with FLAME-B (Appendix C.6.3) on MNIST-Rotations in two

settings: (i) scoring a single batch that mixed the old and new tasks at varying shares (Fig. 13), and (ii) training end to end on the local-mixing streams of Section 4.4 with live detection, compared against the same model given oracle boundaries (Fig. 14).

In setting (i), S_l decreased with the new-task share but stayed above κ for all but the most diluted batches, and every boundary missed on a mixed batch was caught on the pure batch that followed.

In setting (ii), precision stayed perfect for every W, since the detector re-arms only after a full presentation assuming the local stationarity assumption. Recall decreased as the first transition batch carried less of the new task, and plateaued near $W = 5$. Late detections compounded through the same refractory period, which delayed detection on average, although the median boundary was still caught within one batch. The resulting cost in RA relative to oracle boundaries stayed below one percent.

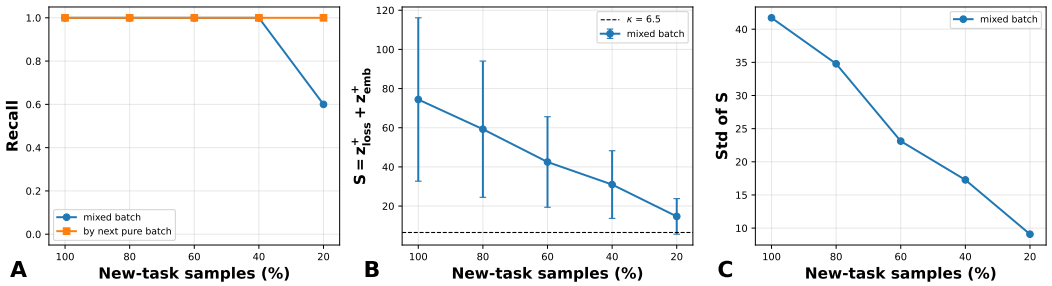

Figure 13: Single-mixed-batch ablation on MNIST-Rotations (FLAME-B C.6.3, five seeds, one boundary per seed). **A:** recall of boundary detection on the mixed batch and, cumulatively, by the pure new-task batch that follows it. **B:** mean S_l on the mixed batch ($\pm$ standard deviation); the dashed line marks κ. **C:** standard deviation of S_l on the mixed batch.

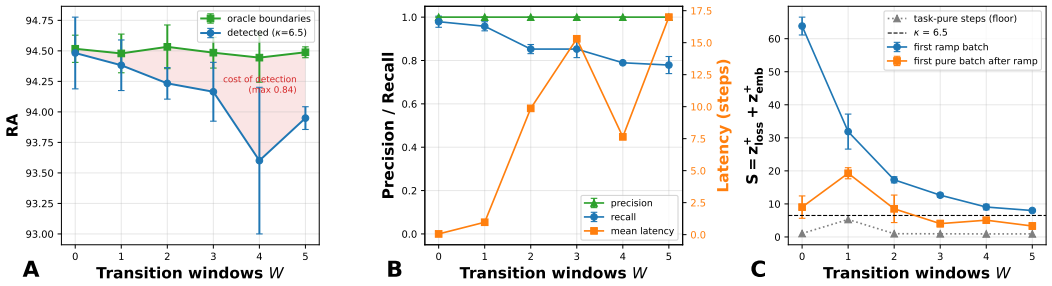

Figure 14: Local mixing over the full task stream on MNIST-Rotations (FLAME-B C.6.3, five seeds). **A:** RA with oracle *vs*. detected boundaries; the shaded area is the cost of detection. **B:** precision and recall of boundary detection (left axis) and mean detection latency in batches (right axis). **C:** S_l at the first transition batch and at the first pure batch after the transition, averaged over boundaries at which the detector was armed, with its level on task-pure batches for reference; the dashed line marks κ.

E Additional Ablation Results

E.1 Choice of Boundary Detection Rule

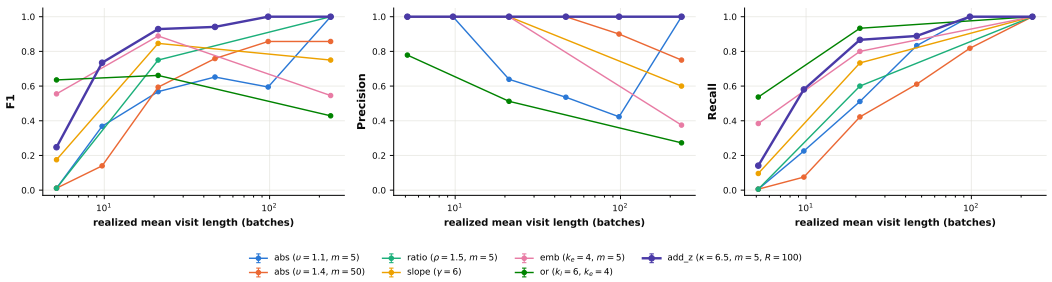

Figure 15: Boundary detection F1 (left), precision (middle) and recall (right) of seven candidate rules against the realized mean visit length of the α-stationarity stream on MNIST-Rotations (FLAME, rule consulted at every batch).

Before settling on the two-channel rule of Section 3.2, we compared it against six alternatives on the α-stationarity stream of Section 4.4: FLAME trained on four-task sequences from MNIST-Rotations at $\alpha \in \{0.8, 0.9, 0.96, 0.98, 0.99, 0.9999\}$, corresponding to a realized mean visit length of 5.1, 9.7, 21.2, 46.7, 98.0 and 234 batches (the full per-task budget at $\alpha = 0.9999$), and 177, 93, 45, 18, 11 and 3 true boundaries respectively. To measure each rule's own discrimination, the rule was consulted at every batch, with no adaptation window after a detection. Using the notation of Section 3.2, the candidates were:

- **abs:** a moving-window loss rule in the spirit of loss-based shift detection (Aljundi et al., 2019; Caccia et al., 2021), $\|L_l - \bar{L}_{l-1:l-m}\| > \upsilon$, with $(\upsilon, m) = (1.1, 5)$ and $(1.4, 50)$;
- **ratio:** the same deviation relative to the loss level, $\|L_l - \bar{L}_{l-1:l-m}\| / \bar{L}_{l-1:l-m} > \rho$, with $\rho = 1.5$, $m = 5$;
- **slope:** the one-step change against its typical size, $|L_l - L_{l-1}| > \gamma\big(|\bar{L}_{l-1:l-m} - \bar{L}_{l-2:l-m-1}| + \hat{\sigma}_l\big)$, where $\hat{\sigma}_l$ is $1.4826\,\mathrm{MAD}$ of the trailing 50 values of $|L_l - L_{l-1}|$, with $\gamma = 6$;
- **emb:** the embedding channel alone, $z_l^{\mathrm{emb}} > k_e$ with $k_e = 4$, where d_l is whitened by the standard deviation of the same $m = 5$ batch means that define its location;
- **or:** a disjunction of the two channels, $\max\big(z_l^{\Delta}/k_l,\ z_l^{\mathrm{emb}}/k_e\big) > 1$, where z_l^{Δ} is the robust standard score of $|L_l - L_{l-1}|$ over the trailing R values, with $k_l = 6$, $k_e = 4$;
- **add_z:** the additive rule of Section 3.2, $S_l = (z_l^{\mathrm{loss}})^+ + (z_l^{\mathrm{emb}})^+ > \kappa$, with $\kappa = 6.5$, $m = 5$, $R = 100$.

Fig. 15 reports the detection precision, recall and F1 of each rule. The additive rule achieved the highest mean F1 over the stationarity levels shared by all rules, and was the only rule that never caused a false positive, at any value of α. Every other rule lost precision as the visits lengthened, which is the regime where boundary detection matters most: as shown in Fig. 3A, the gap between the oracle and the non-adaptive references widened with α, so a missed or spurious boundary cost more retained accuracy there. In this regime, the additive rule caught most boundaries, and all of them at the two longest visit lengths, without a single false positive.

Two design choices drove this. First, the thresholds of abs, ratio and slope acted on the raw loss and had to be tuned to its scale, which drifts as training proceeds, whereas both terms of S_l are robust standard scores, making κ scale-free. Second, summing the two channels rewarded agreement: a genuine shift that moved both channels moderately cleared κ, whereas a fluctuation confined to one channel did not. A disjunction instead false-alarmed on either channel's fluctuations, and a conjunction could not fire when one channel stayed silent.

The additive rule was weaker only at the shortest visits, where the next boundary often arrived while the rule was silent, re-estimating the location of its embedding channel after a detection. As discussed

in Section 4.4, missed boundaries had little impact on retained accuracy in this regime, because the stream approached a stationary mixture of tasks. We therefore adopted the additive rule, with $\kappa = 6.5$, $m = 5$ and $R = 100$, for all experiments.

E.2 Effect of Buffer Size

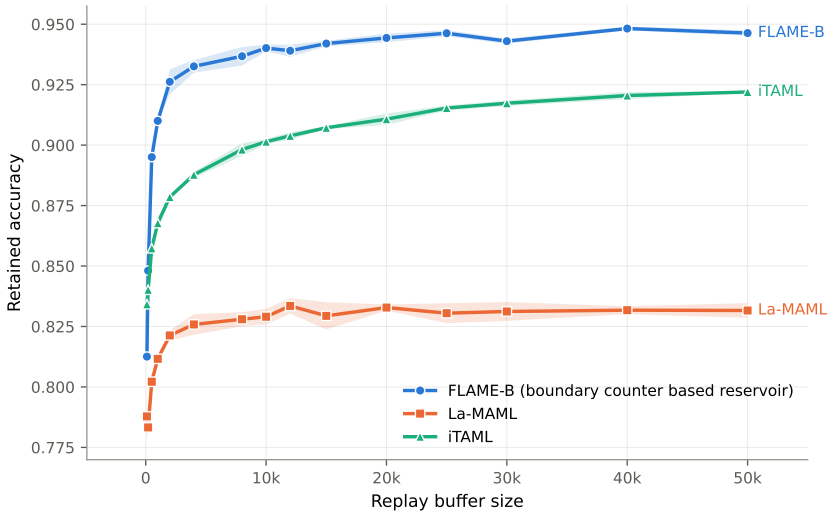

Figure 16: Retained accuracy (RA) versus replay buffer size on MNIST-Rotations with oracle task boundaries. Markers show the mean over three seeds; shaded bands show ± 1 standard deviation.

We studied how the size of the replay buffer affected the retained accuracy (RA) of FLAME-B (Appendix C.6.3), La-MAML (Joseph & Gu, 2021) and iTAML (Rajasegaran et al., 2020) on MNIST-Rotations (20 tasks) with known task boundaries, sweeping the buffer size from 100 to 50,000 samples (Fig. 16).

FLAME-B achieved the highest RA at every buffer size but the smallest. Its RA rose steeply up to about a thousand samples and then saturated, and the buffer size of 30,000 used in our main experiments lay on this plateau. La-MAML saturated earlier and at a much lower level, so its gap to FLAME-B widened as the buffer grew. iTAML improved steadily over the whole range and narrowed its gap to FLAME-B at large buffers without closing it. At La-MAML's original buffer size of 200, FLAME-B and iTAML were comparable and both well above La-MAML.

The exception was the smallest buffer, where iTAML outperformed FLAME-B. Each sample FLAME-B replays is paired with a context set drawn from buffered samples of the same task, and at 100 samples the buffer held too few samples per task to form diverse context sets. FLAME-B recovered quickly as the buffer grew.

E.3 Effect of Context Size

We also investigated the effect of context set size, the support set drawn from the previous batch used by the shared feature extractor to condition task-specific predictions, in the continual task setting. Similar to the accuracy *vs*. replay buffer size ablation, sweeping `context_size` from 10 to 200 revealed a saturating effect on final average validation accuracy across tasks: gains diminish beyond a modest context size as shown in Fig. 17. As an accuracy vs. computational cost trade-off (larger context sizes increase memory and per-step inference cost through the amortizer), we selected `context_size`$=$ 90 as our default for all main experiments.

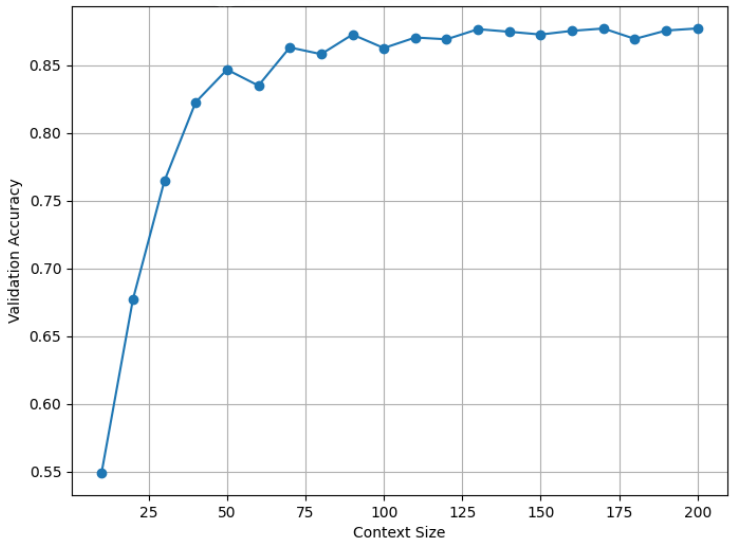

Figure 17: Accuracy vs Context Size.

E.4 Replay Memory and Compute Cost

Because FLAME maintains a clustered buffer rather than a flat reservoir, it is reasonable to ask whether its retained-accuracy advantage is bought with a larger memory footprint or a heavier replay budget. We instrumented the buffer of FLAME and La-MAML (Joseph & Gu, 2021) on MNIST-Rotations, sweeping the buffer from $1{,}000$ to $50{,}000$ samples and recording, at every task boundary, the memory the buffer holds and the wall time attributable to it, split into the per-step cost of drawing a replay batch and the periodic cost of maintaining and consolidating the buffer.

We found that memory was governed by the buffer size alone: at a given capacity both methods held essentially the same amount of data throughout the sequence (Fig. 18D, 82 MB for FLAME against 90 MB for La-MAML at $30{,}000$ samples, and 144 MB against 150 MB at $50{,}000$), because the stored samples dominate the buffer memory and the mixture state FLAME keeps alongside them is negligible. On compute FLAME was not merely comparable but substantially cheaper, spending five to eight times less total time on the buffer than La-MAML at every capacity (Fig. 18A). The reason is a difference in where the cost falls (Fig. 18B,C) rather than in how much work is done overall. La-MAML draws from the whole reservoir at every training step, so its per-draw cost grows with the buffer and is paid continuously; at $50{,}000$ samples a single draw took $5{,}116$ ms against FLAME's 50 ms, a hundredfold gap that widens with capacity. FLAME instead confines sampling to the selected clusters, which keeps the per-step cost nearly flat in the buffer size, and concentrates its work in consolidation, which is paid only at boundaries. Trading a recurring per-step cost that scales with the buffer for a periodic one proved favourable at every capacity we tested.

These measurements carry two caveats. La-MAML was run with fewer seeds than FLAME — two at the smaller capacities and one at $30{,}000$ and $50{,}000$ — so its variability at the larger buffers is understated.

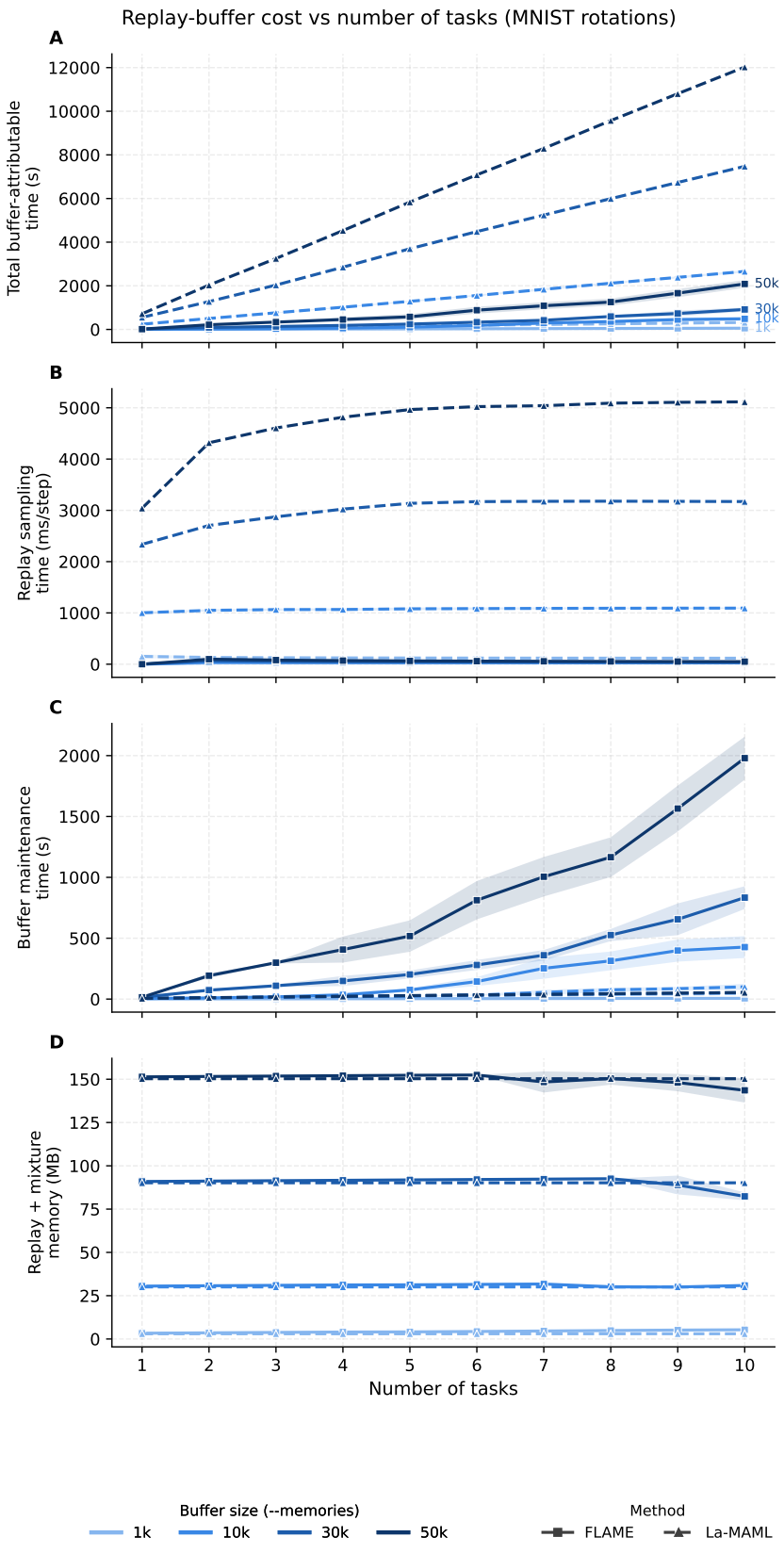

Figure 18: Replay-buffer cost against the number of tasks seen on MNIST-Rotations, at buffer sizes of 1,000, 10,000, 30,000 and 50,000 samples (light to dark). (A) total buffer-attributable wall time, (B) mean wall time of one replay draw, (C) cumulative buffer maintenance time, (D) resident replay memory. Markers show the mean over seeds; shaded bands show ± 1 standard deviation.