Learning to Continually Learn with the Bayesian Principle

Soochan Lee[1] **Hyeonseong Jeon**[1] **Jaehyeon Son**[1] **Gunhee Kim**[1]

Abstract

In the present era of deep learning, continual learning research is mainly focused on mitigating forgetting when training a neural network with stochastic gradient descent on a non-stationary stream of data. On the other hand, in the more classical literature of statistical machine learning, many models have sequential Bayesian update rules that yield the same learning outcome as the batch training, i.e., they are completely immune to catastrophic forgetting. However, they are often overly simple to model complex real-world data. In this work, we adopt the meta-learning paradigm to combine the strong representational power of neural networks and simple statistical models' robustness to forgetting. In our novel meta-continual learning framework, continual learning takes place only in statistical models via ideal sequential Bayesian update rules, while neural networks are meta-learned to bridge the raw data and the statistical models. Since the neural networks remain fixed during continual learning, they are protected from catastrophic forgetting. This approach not only achieves significantly improved performance but also exhibits excellent scalability. Since our approach is domain-agnostic and model-agnostic, it can be applied to a wide range of problems and easily integrated with existing model architectures.

1. Introduction

Continual learning (CL), the process of acquiring new knowledge or skills without forgetting existing ones, is an essential ability of intelligent agents. Despite recent advances in deep learning, CL remains a significant challenge. Knoblauch et al. (2020) rigorously prove that, in general, CL is an NP-hard problem. This implies that building a universal CL algorithm is impossible as long as P≠NP. To effectively tackle CL, one should first narrow down a domain and design a CL algorithm tailored to leverage a domain-specific structure. Even humans possess specialized CL abilities for specific tasks, such as learning new faces, which may not be as effective for other tasks, such as memorizing random digits. This specialization results from the evolutionary process that has optimized our CL abilities for survival and reproduction.

From this perspective, meta-continual learning (MCL) emerges as a highly promising avenue of research. Rather than manually crafting CL algorithms based solely on human knowledge, MCL aims to meta-learn the CL ability in a data-driven manner – *learning to continually learn*. Thus, we can design a general MCL algorithm and feed domain-specific data to obtain a specialized CL algorithm. MCL can be more advantageous in many practical scenarios, as it can utilize a large-scale dataset to improve the CL ability before deploying a CL agent, instead of learning from scratch.

MCL follows the bi-level optimization scheme of meta-learning: in the inner loop, a model is continually trained by a CL algorithm, while in the outer loop, the CL algorithm is optimized across multiple CL episodes. Although stochastic gradient descent (SGD) has been the primary learning mechanism in deep learning, this bi-level scheme offers the flexibility to combine neural networks with fundamentally different learning mechanisms. Specifically, we can meta-train neural networks with SGD only in the outer loop and adopt another update rule for CL in the inner loop.

In this context, the sequential Bayesian update stands out as the most promising candidate, providing an ideal framework for updating a knowledge state. While there have been a significant number of CL approaches inspired by the Bayesian updates of the posterior of neural network parameters (Kirkpatrick et al., 2016; Zenke et al., 2017; Chaudhry et al., 2018; Nguyen et al., 2018; Farquhar & Gal, 2019), they require various approximations to ensure computational tractability, which sets them apart from the ideal Bayesian update. On the other hand, we bring the Fisher-Darmois-Koopman-Pitman theorem (Fisher, 1934; Darmois, 1935; Koopman, 1936; Pitman, 1936) into the scope to point out that the exponential family is the only family of distri-

[1] Department of Computer Science and Engineering, Seoul National University, Seoul, Republic of Korea. Correspondence to: Gunhee Kim <gunhee@snu.ac.kr>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

butions that are capable of efficient and lossless sequential Bayesian update (more precise description in §2.2). Instead of dealing with the intractable posterior of complex neural networks, we consider the sequential Bayesian inference of simple statistical models that inherently come with an exponential family posterior, yielding a result identical to batch inference. While these models are immune to catastrophic forgetting by design, they are often too simple for modeling complex, high-dimensional data. Fortunately, the MCL setting allows meta-training neural networks that can work as bridges between the real world and the statistical models.

We distill this idea of combining simple statistical models and meta-learned neural networks into a general MCL framework named *Sequential Bayesian Meta-Continual Learning (SB-MCL)*. Since SB-MCL is domain-agnostic and model-agnostic, it can be applied to a wide range of problem domains and integrated with existing model architectures with minimal modifications. SB-MCL encompasses several prior works (Banayeeanzade et al., 2021; Snell et al., 2017; Harrison et al., 2018) as special cases and supports both supervised and unsupervised learning. In our extensive experiments on a wide range of benchmarks, SB-MCL achieves remarkable performance while using substantially less resources. Code is available at `https://github.com/soochan-lee/SB-MCL`.

2. Background

2.1. Meta-Continual Learning

We describe the problem setting of MCL. We denote an example (x, y) where x is an input variable, and y is a target variable, assuming a supervised setting by default. For unsupervised learning settings, one can replace (x, y) with x. A CL episode $(\mathcal{D}, \mathcal{E})$ consists of a training stream $\mathcal{D} = ((x_t, y_t))_{t=1}^T$ and a test set $\mathcal{E} = \{(\tilde{x}_n, \tilde{y}_n)\}_{n=1}^N$. The training stream is an ordered sequence of length T, and its examples can be accessed sequentially and cannot be accessed more than once. It is assumed to be non-stationary and typically constructed as a concatenation of K distinct *task* streams. Naively training a neural network on such a non-stationary stream with SGD results in catastrophic forgetting of the knowledge from the previous part of the stream. The test set consists of examples of the tasks appearing in the training stream, such that the model needs to retain knowledge of all the tasks to obtain a high score in the test set.

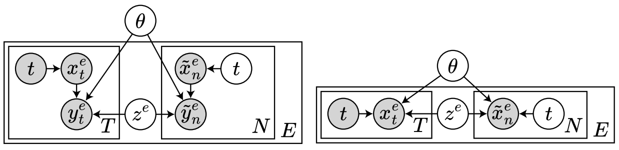
In MCL, multiple CL episodes are split into a meta-training set $\mathbb{D} = \{(\mathcal{D}^i, \mathcal{E}^i)\}_i$ and a meta-test set $\mathbb{E} = \{(\mathcal{D}^j, \mathcal{E}^j)\}_j$. During the meta-training phase, a CL algorithm is optimized across multiple episodes in $\mathbb{D}$ to produce a competent model from a training stream. The algorithm's CL capability is then measured with $\mathbb{E}$. Note that $\mathbb{D}$ and $\mathbb{E}$ typically do not share any underlying tasks since the meta-test set aims to measure the learning capability, not the knowledge of specific tasks that appear during meta-training. Note that MCL should not be confused with other specialized settings that combine meta-learning and CL (Finn et al., 2019; Riemer et al., 2019; Jerfel et al., 2019; Gupta et al., 2020; to name a few). They have different assumptions and objectives that are not compatible with MCL.

2.2. Sequential Bayesian Update of Exponential Family Posterior

The Bayes rule offers a principled way to update knowledge incrementally by using the posterior at the previous time step as the prior for the current time step, i.e., $p(z|x_{1:t}) \propto p(x_t|z)p(z|x_{1:t-1})$ (Bishop, 2006; Murphy, 2022). Therefore, the Bayesian perspective has been widely adopted in CL research (Kirkpatrick et al., 2016; Zenke et al., 2017; Chaudhry et al., 2018; Nguyen et al., 2018; Farquhar & Gal, 2019). However, prior works have focused on sequentially updating the posterior of neural network parameters, which are generally intractable to compute. Therefore, they must rely on various approximations, resulting in a wide gap between the ideal Bayesian update and reality.

Then, what kind of models are suitable for efficient sequential Bayesian updates? According to the Fisher-Darmois-Koopman-Pitman theorem (Fisher, 1934; Darmois, 1935; Koopman, 1936; Pitman, 1936), *the exponential family is the only family of distributions where the dimension of the sufficient statistic remains fixed, regardless of the number of examples.* Sufficient statistics are the minimal statistics that capture all the information in the data about the parameter of interest. Therefore, if the dimension of the sufficient statistic remains fixed, we can store all the necessary information in a fixed-size memory system. This theorem has significant implications for CL; if the model's posterior is not a member of the exponential family (as in the case of neural networks) and does not have a large enough memory system to store the ever-growing sufficient statistics, forgetting becomes inevitable. From this perspective, employing a replay buffer (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019) is an approach that aids in partially preserving sufficient statistics.

On the flip side, the theorem suggests an alternative approach; by embracing an exponential family distribution, we can store sufficient statistics within a fixed dimension, enabling efficient sequential Bayesian updates without any compromises. Although the exponential family's expressivity is limited, this challenge can be effectively addressed in MCL settings by meta-learning neural networks to reconcile the real-world data and the exponential family.



(a) Supervised MCL.

(b) Unsupervised MCL.

Figure 1. Graphical models of MCL. For each episode e, training examples (x_t^e, y_t^e) (or just x_t^e) are produced conditioned on the time step t and the episode-wise latent variable z^e.

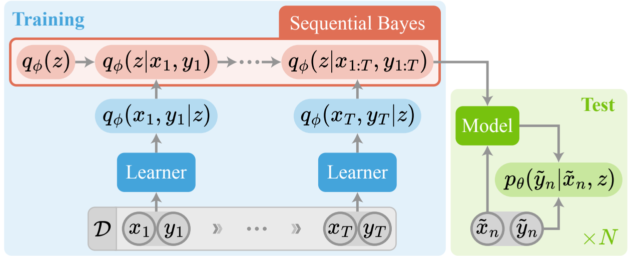


Figure 2. Schematic diagram of our SB-MCL in a single supervised CL episode. In SB-MCL, CL is formulated as the sequential Bayesian update of an exponential family posterior $q_\phi(z|x_{1:t}, y_{1:t})$. The meta-learned neural networks (the learner and the model) remain fixed during CL to protect themselves from catastrophic forgetting.

3. Our Approach: SB-MCL

3.1. The Meta-Learning Objective

Fig. 1 shows the graphical models of our MCL settings. In both supervised and unsupervised settings, there are E CL episodes. Each CL episode e has a training stream $\mathcal{D}^e$ of length T and a test set $\mathcal{E}^e$ of size N. In supervised CL settings (Fig. 1a), each example is a pair of input x and target y, and the goal is to model the conditional probability $p(y|x)$. In unsupervised settings (Fig. 1b), an example is simply x, and the goal is to model $p(x)$. For each CL episode e, we assume an episode-specific latent variable z^e that governs the entire episode. The training stream's non-stationarity, a key characteristic of CL, is expressed by the time variable t affecting the generation of x. In practice, the training stream is often constructed by concatenating multiple *task* streams, each of which is a stationary stream sampled from a distinct task distribution. Note that z^e is shared by all examples inside an episode regardless of the tasks they belong to. Under this framework, the CL process is to sequentially refine the belief state of z^e.

The objective is to maximize the (conditional) log-likelihood of the test set $\mathcal{E}$ after continually learning the training stream $\mathcal{D}$ (superscript e is now omited for brevity). Assuming a model parameterized by θ, this objective can be summarized as

$$\sum_{n=1}^{N} \log p_\theta(\tilde{y}_n|\tilde{x}_n, \mathcal{D}) = \sum_{n=1}^{N} \log \int_z p_\theta(\tilde{y}_n|\tilde{x}_n, z) p_\theta(z|\mathcal{D})$$

in supervised settings and as

$$\sum_{n=1}^{N} \log p_\theta(\tilde{x}_n|\mathcal{D}) = \sum_{n=1}^{N} \log \int_z p_\theta(\tilde{x}_n|z) p_\theta(z|\mathcal{D})$$

in unsupervised settings, where $\tilde{x}_*$ and $\tilde{y}_*$ are the test data in $\mathcal{E}$. However, computing these objectives is generally intractable due to the integration over z. For such cases, we introduce a variational distribution q_ϕ parameterized by ϕ and derive the variational lower bounds. The bounds for the supervised and unsupervised cases are derived as

$$\begin{aligned} &\log p_\theta(\tilde{y}_{1:N}|\tilde{x}_{1:N}, \mathcal{D}) = \log p_\theta(\tilde{y}_{1:N}|\tilde{x}_{1:N}, x_{1:T}, y_{1:T}) \\ &\geq \mathbb{E}_{z \sim q_\phi(z|\mathcal{D})} \left[\sum_{n=1}^{N} \log p_\theta(\tilde{y}_n|\tilde{x}_n, z) + \sum_{t=1}^{T} \log p_\theta(y_t|x_t, z) \right] \\ &\quad - D_{\mathrm{KL}}\left(q_\phi(z|\mathcal{D}) \,\|\, p_\theta(z)\right) - \underbrace{\log p_\theta(\mathcal{D})}_{\text{const.}}, \end{aligned} \tag{1}$$

$$\begin{aligned} &\log p_\theta(\tilde{x}_{1:N}|\mathcal{D}) = \log p_\theta(\tilde{x}_{1:N}|x_{1:T}) \\ &\geq \mathbb{E}_{z \sim q_\phi(z|\mathcal{D})} \left[\sum_{n=1}^{N} \log p_\theta(\tilde{x}_n|z) + \sum_{t=1}^{T} \log p_\theta(x_t|z) \right] \\ &\quad - D_{\mathrm{KL}}\left(q_\phi(z|\mathcal{D}) \,\|\, p_\theta(z)\right) - \underbrace{\log p_\theta(\mathcal{D})}_{\text{const.}}. \end{aligned} \tag{2}$$

For more details, please refer to Appendix A.

3.2. Continual Learning as Sequential Bayesian Update

In Eq. 1 and 2, the CL process is abstracted inside the variational posterior $q_\phi(z|\mathcal{D})$, which is obtained through sequential Bayesian updates:

$$\begin{aligned} q_\phi(z|x_{1:t}, y_{1:t}) &\propto q_\phi(x_t, y_t|z) q_\phi(z|x_{1:t-1}, y_{1:t-1}), \\ q_\phi(z|x_1, y_1) &\propto q_\phi(x_1, y_1|z) q_\phi(z) \end{aligned} \tag{3}$$

$$\begin{aligned} q_\phi(z|x_{1:t}) &\propto q_\phi(x_t|z) q_\phi(z|x_{1:t-1}), \\ q_\phi(z|x_1) &\propto q_\phi(x_1|z) q_\phi(z), \end{aligned} \tag{4}$$

where Eq. 3 and 4 are respectively for supervised and unsupervised CL. In the following, we will consider only the supervised case to be concise, but the same logic can be extended to the unsupervised case. As depicted in Fig. 2, The CL process initially starts with a variational prior $q_\phi(z)$. And the *learner*, a neural network component, produces $q_\phi(x_t, y_t|z)$ for each example (x_t, y_t), which is subsequently integrated into the variational posterior

$q_\phi(z|x_{1:t}, y_{1:t})$.[1] The parameters of the prior and the learner constitute ϕ. As previously explained in §2.2, the Fisher-Darmois-Koopman-Pitman theorem implies that only exponential family distributions can perform such updates without consistently increasing the memory and compute requirement proportional to the number of examples. This property makes them ideal for our variational posterior. Note that SB-MCL does not involve any gradient descent during CL; the learner performs only the forward passes to process the training examples for sequential Bayesian updates.

As an example of exponential family distributions, we describe the exact update rule for a factorized Gaussian posterior $\mathcal{N}(z; \mu_t, \Lambda_t^{-1})$ where Λ_t is diagonal. First, the variational prior is also defined as a factorized Gaussian: $q_\phi(z) = \mathcal{N}(z; \mu_0, \Lambda_0^{-1})$. For $q_\phi(x_t, y_t|z)$, the learner outputs $\hat{z}_t$ and P_t for each (x_t, y_t), where P_t is a diagonal matrix. We consider $\hat{z}_t$ as a noisy observation of z with a Gaussian noise of precision P_t, i.e., $q_\phi(x_t, y_t|z) = \mathcal{N}(\hat{z}_t; z, P_t^{-1})$ (Volpp et al., 2021). This allows an efficient sequential update rule for the variational posterior (Bishop, 2006):

$$\Lambda_t = \Lambda_{t-1} + P_t, \quad \mu_t = \Lambda_t^{-1}\left(\Lambda_{t-1}\mu_{t-1} + P_t\hat{z}_t\right). \quad (5)$$

After training, the posterior $q_\phi(z|\mathcal{D}) = q_\phi(z|x_{1:T}, y_{1:T})$ is passed on to the test phase. During testing, the model produces outputs conditioned on the test input $\tilde{x}_n$ and z, which is compared with the test output $\tilde{y}_n$ to obtain the test log-likelihood $\mathbb{E}_{z\sim q_\phi(z|x_{1:T},y_{1:T})}[\log p_\theta(\tilde{y}_n|\tilde{x}_n, z)]$. It would be ideal if we could analytically compute it, but if this is not the case, we may approximate it by the Monte Carlo estimation (sampling multiple z's from $q_\phi(z|x_{1:T}, y_{1:T})$) or the maximum a posteriori estimation of z.

3.3. Meta-Training

During the meta-training phase, the model and the learner are meta-updated to maximize Eq. 1 or 2 with multiple CL episodes. For each episode, the CL process in §3.2 is used to obtain $q_\phi(z|\mathcal{D})$ with the learner. In contrast to SGD-based MCL, our approach does not need to process the training stream sequentially. If all the training examples are available, which is generally true during meta-training, we can feed them to the learner in parallel and combine the results with a batch inference rule instead of the sequential update rule. With the Gaussian posterior, for example, we can use the following formula instead of Eq. 5 to produce the identical result:

$$\Lambda_T = \sum_{t=0}^{T} P_t, \quad \mu_T = \Lambda_T^{-1}\sum_{t=0}^{T} P_t\hat{z}_t. \quad (6)$$

Compared to SGD-based approaches requiring forward-backward passes for each example sequentially, the meta-training of our approach can benefit from parallel processors such as GPUs or TPUs.

Once the variational posterior $q_\phi(z|\mathcal{D})$ is obtained, we use Monte Carlo approximation for the expectation w.r.t. $q_\phi(z|\mathcal{D})$ (Kingma & Welling, 2014). For the Gaussian posterior, we can utilize the reparameterization trick (Kingma & Welling, 2014) to sample z that allows backpropagation:

$$z = \mu_T + \Lambda_T^{-1/2}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I). \quad (7)$$

Conditioned on z, we run the model on the training and test examples to compute the first term in Eq. 1 or 2. This term encourages the cooperation between the model and the learner to increase the likelihood of the data. The second term is the Kullback-Leibler (KL) divergence between the variational posterior $q_\phi(z|\mathcal{D})$ and the prior $p_\theta(z)$, which can be regarded as a regularization term. We set the prior to be the same exponential family distribution, e.g., the unit Gaussian for the Gaussian posterior, which enables an analytical computation of the KL divergence. Finally, the last term $\log p_\theta(\mathcal{D})$ is a constant that can be ignored for optimization purposes.

After Eq. 1 or 2 is computed for an episode or a batch of episodes, we perform a meta-update on the model and the learner with an SGD algorithm, backpropagating through the entire episode. Unlike existing SGD-based MCL methods (Javed & White, 2019; Beaulieu et al., 2020), we do not need to calculate any second-order gradients, which is a significant advantage for scalability.

3.4. Existing Special Cases of SB-MCL

Several prior works can be considered domain-specific special cases of SB-MCL. We summarize the key characteristics in Table 1 and high-level descriptions in the following.

GeMCL (Banayeeanzade et al., 2021). GeMCL can be regarded as a specific instance of our framework in the image classification domain. It utilizes a meta-learned neural network encoder to extract an embedding vector for each image. During the training process, it maintains a Gaussian posterior for each class in the embedding space. Each Gaussian posterior is updated by the sequential Bayesian update rule whenever an example for the corresponding class becomes available. These Gaussians collectively form a Gaussian mixture model (GMM) within the embedding space. At test time, each test image is converted into an embedding vector by the same encoder, and a class is predicted by inferring the mixture component of GMM. To view GeMCL as an instance of SB-MCL, we consider the encoder as serving two roles: one as the learner and the other as a component of the model. During training, the encoder is used as the learner to update the posterior $q_\phi(z|x_{1:t}, y_{1:t})$ where z is the parameters of the GMM. At test time, the encoder transforms the test inputs into embeddings as a model com-

[1] Both $q_\phi(x_t, y_t|z)$ and $q_\phi(z|x_{1:t}, y_{1:t})$ are used as functions of z since $(x_{1:t}, y_{1:t})$ is given.

Table 1. Summary of the special cases of SB-MCL.

Method	Domain	Model structure	z	$q_\phi(z\|\mathcal{D})$
GeMCL	Classification	Encoder + GMM	GMM param.	Per-class Gaussian
PN	Classification	Encoder + GMM	GMM param.	Per-class isotropic Gaussian
ALPaCA	Regression	Encoder + Linear model	Linear model param.	Matrix normal
SB-MCL	Any domain	Any model	Any auxiliary input	An exponential family distribution

ponent, and the GMM classifies the embeddings with its parameters learned from the training phase. Banayeeanzade et al. (2021) also propose an MAP variant, which simply produces $p_\theta(\tilde{y}_n|\tilde{x}_n, z_{\text{MAP}})$ as the output. This variant has simpler computation without significant performance drops.

Prototypical Networks (Snell et al., 2017). While GeMCL is a special case of SB-MCL, it can also be seen as a generalization of the Prototypical Network (PN), which was originally proposed as a meta-learning approach for few-shot classification. Therefore, PN also falls under the SB-MCL family. While GeMCL takes a fully Bayesian approach, PN simply averages the embeddings of each class to construct a prototype vector. Since the average operation can be performed sequentially, PN can be readily applied to MCL settings. We can simplify GeMCL to PN by assuming isotropic Gaussian posteriors and an uninformative prior (Banayeeanzade et al., 2021).

ALPaCA (Harrison et al., 2018). Originally proposed as a meta-learning approach for online regression problems, ALPaCA attaches a linear model on top of a meta-learned neural network encoder, symmetrical to PN or GeMCL that attaches a GMM for classification. In ALPaCA, the latent variable z is the weight matrix of the linear model, whose posterior is assumed to have the matrix normal distribution. Due to the similar streaming settings of online and continual learning, we can apply ALPaCA to MCL regression settings with minimal modifications.

3.5. Converting Arbitrary Models for SB-MCL

All the prior works in the previous section share a similar architecture: a meta-learned encoder followed by a simple statistical model. This configuration can be ideal if the output type is suitable for the statistical model, allowing analytic computation of the posterior. However, it is hard to apply such architectures to domains with more complex output formats or unsupervised settings where the output variable does not exist.

On the other hand, we can apply SB-MCL to almost any existing model architectures or domains, since the only modification is to be conditioned on some z whose posterior is modeled with the exponential family. Once the model is modified, a learner is added to digest the training stream into the variational posterior of z. It may share most of its parameters with the model.

While there are infinitely many ways to implement such modifications, we currently focus on perhaps the simplest approach and leave exploring more sophisticated architectures for future work. In our experiments, we define z to be a 512-dimensional factorized Gaussian variable, which is injected into the model as an auxiliary input. If the model structure follows an encoder-decoder architecture, we concatenate z with the encoder output and pass the result to the decoder. It should be noted that, despite its simplicity, a high-dimensional Gaussian can be surprisingly versatile when properly combined with neural networks. This has also been demonstrated by generative models, such as VAEs (Kingma & Welling, 2014) or GANs (Goodfellow et al., 2014), where neural networks transform a unit Gaussian variable into realistic images. While their choice of Gaussian is motivated by the convenience of sampling, ours is motivated by its robustness to forgetting.

4. Related Work

SGD-Based MCL. OML (Javed & White, 2019) employs a small multi-layer perceptron (MLP) with MAML (Finn et al., 2017) on top of a meta-learned encoder. In the inner loop of OML, the encoder remains fixed while the MLP is updated by sequentially learning each training example via SGD. After training the MLP in the inner loop, the entire model is evaluated on the test set to produce the meta-loss. Then, the gradient of the meta-loss is computed with respect to the encoder parameters and the initial parameters of the MLP to update them. Inspired by OML, ANML (Beaulieu et al., 2020) is another MCL method for image classification that introduces a component called neuromodulatory network. Its sigmoid output is multiplied to the encoder output to adaptively gate some features depending on the input. For a detailed survey of MCL and other combinations of meta-learning and CL, we refer the reader to Son et al. (2023).

Continual Learning as Sequence Modeling (CL-Seq). More recently, Lee et al. (2023) pointed out that CL is inherently a sequence modeling problem; predicting the target $\tilde{y}$ of a test input $\tilde{x}$ after a training stream $((x_1, y_1), ..., (x_T, y_T))$ is equivalent to predicting the next token $\tilde{y}$ that comes after prompt $(x_1, y_1, ..., x_T, y_T, \tilde{x})$.

From this perspective, forwarding the training stream through an autoregressive sequence model and updating its internal state, which has been called *in-context learning* in the language modeling literature (Brown et al., 2020), can be considered CL. Within MCL settings, the sequence model can be meta-trained on multiple CL episodes to perform CL. They demonstrate that Transformer (Vaswani et al., 2017) and their efficient variants (Katharopoulos et al., 2020; Choromanski et al., 2021) achieve significantly better scores compared to SGD-based approaches.

Neural Processes. While motivated by different objectives, intriguing similarities can be identified between the supervised version of SB-MCL (Eq. 1) and the neural process (NP) literature (Garnelo et al., 2018a;b). NP was initially proposed to solve the limitations of Gaussian processes, such as the computational cost and the difficulties in the prior design. It can also be considered a meta-learning approach that learns a functional prior and has been applied as a solution to the meta-learning domain (Gordon et al., 2019). Since NPs are rooted in stochastic processes, one of their primary design considerations is exchangeability: the model should produce the same result regardless of the order of the training data. To achieve exchangeability, NPs independently encode each example and aggregate them into a single variable with a permutation-invariant operation, such as averaging, and pass it to the decoder. While our sequential Bayesian update of an exponential family posterior is initially inspired by the Fisher-Darmois-Koopman-Pitman theorem, it also ensures exchangeability. Volpp et al. (2021) propose an aggregation scheme for NPs based on Bayesian principles and even suggest the possibility of sequential update, but they do not connect it to CL. To the best of our knowledge, the only connection between NPs and MCL is CNAP (Requeima et al., 2019), but it is a domain-specific architecture designed for image classification.

5. Experiments

We demonstrate the efficacy of our framework on a wide range of domains, including both supervised and unsupervised tasks. We also provide PyTorch (Paszke et al., 2019) code, ensuring the reproducibility of all experiments. Due to page limitations, we present only the most essential information; for further details, please refer to the code.

5.1. Methods

SGD-Based MCL. Due to its simplicity and generality, we test OML (Javed & White, 2019) as a representative baseline of SGD-based MCL. Although it was originally proposed for classification and simple regression, Lee et al. (2023) introduce an encoder-decoder variant of OML by stacking a MAML MLP block between the encoder and decoder, which can be used for other domains. As the main computational bottleneck of OML is the second-order gradient computation, we also test its first-order approximation (OML-Rep), following Reptile (Nichol et al., 2018).

CL-Seq. We test Transformer (TF; Vaswani et al., 2017) and Linear Transformer (Linear TF; Katharopoulos et al., 2020) imported from the implementation of Lee et al. (2023). In the case of TF, the computational cost keeps increasing as it learns more examples, which has been criticized as a major drawback limiting its scalability (Tay et al., 2022). On the other hand, Linear TF maintains a constant computational cost like other baselines and our SB-MCL, but its performance falls behind TF (Lee et al., 2023).

Offline and Online Learning. Although our work focuses on MCL, a significant number of non-meta-CL methods have been proposed. To provide a reference point to them, we report the offline and online learning scores, which are generally considered the upper bound of CL and online CL performance (Zenke et al., 2017; Farajtabar et al., 2020). For offline learning, we train a model from scratch for an unlimited number of SGD steps with mini-batches uniformly sampled from the entire training stream. Since the model is usually overfitted to the training set, we report the best test score achieved during training. For online learning, we randomly shuffle the training stream to be a stationary stream, train a model from scratch for one epoch, and measure the final test score. Note that MCL methods can outperform offline and online learning since they can utilize a large meta-training set, unlike CL methods (Lee et al., 2023).

The SB-MCL Family (Ours). We test the special cases of SB-MCL in Table 1 for their respective domains, i.e., GeMCL for image classification, ALPaCA for simple regression, and the generic variant with the factorized Gaussian variable (§3.5) for others. GeMCL and ALPaCA support the analytic calculation of posterior predictive distribution during testing. For the generic cases, we impose 512D factorized Gaussian on $q_\phi(z|\mathcal{D})$ and sample z five times to approximate $\mathbb{E}_{z\sim q_\phi(z|\mathcal{D})}[p_\theta(\tilde{y}_n|\tilde{x}_n, z)]$. In Appendix D, we also report the scores of its MAP variant that simply produces $p_\theta(\tilde{y}_n|\tilde{x}_n, z_{\mathrm{MAP}})$. The scores of MAP estimation are nearly the same as those of Monte Carlo estimation.

5.2. Benchmarks

Our experimental settings are mainly based on those of Lee et al. (2023). As the popular Omniglot dataset (Lake et al., 2015) causes severe meta-overfitting due to its small size (1.6K classes / 32K images), they repurpose CASIA (Liu et al., 2011) and MS-Celeb-1M (Guo et al., 2016) datasets for MCL. CASIA is a Chinese handwriting dataset that comprises 3.9M images of 7.4K character types, while MS-Celeb-1M contains 10M images of 100K celebrities. Using these datasets, Lee et al. (2023) test various types of supervised learning benchmarks, including both classification

Table 2. Classification results in the error rate (↓).

Method	Omniglot	CASIA	Celeb
Offline	$.300^{\pm.055}$	$.345^{\pm.045}$	$.625^{\pm.065}$
Online	$.800^{\pm.055}$	$.963^{\pm.020}$	$.863^{\pm.037}$
OML	$.046^{\pm.002}$	$.015^{\pm.001}$	$.331^{\pm.006}$
OML-Rep	$.136^{\pm.005}$	$.057^{\pm.003}$	$.660^{\pm.012}$
TF	$.014^{\pm.001}$	$.004^{\pm.000}$	$\mathbf{.228}^{\pm.003}$
Linear TF	$.125^{\pm.016}$	$.006^{\pm.000}$	$.229^{\pm.003}$
SB-MCL	$\mathbf{.008}^{\pm.000}$	$\mathbf{.002}^{\pm.000}$	$.231^{\pm.004}$

Table 3. Regression results in the loss (↓).

Method	Sine	CASIA Compl.	CASIA Rotation	Celeb Compl.
Offline	$.0045^{\pm.0003}$	$.146^{\pm.009}$	$.544^{\pm.045}$	$.160^{\pm.008}$
Online	$.5497^{\pm.0375}$	$.290^{\pm.023}$	$1.079^{\pm.081}$	$.284^{\pm.017}$
OML	$.0164^{\pm.0007}$	$.105^{\pm.000}$	$.052^{\pm.002}$	$.099^{\pm.000}$
OML-Rep	$.0271^{\pm.0012}$	$.104^{\pm.000}$	$.050^{\pm.002}$	$.105^{\pm.000}$
TF	$\mathbf{.0009}^{\pm.0001}$	$\mathbf{.097}^{\pm.000}$	$\mathbf{.034}^{\pm.001}$	$\mathbf{.094}^{\pm.000}$
Linear TF	$.0031^{\pm.0002}$	$.101^{\pm.000}$	$.068^{\pm.002}$	$.097^{\pm.000}$
SB-MCL	$.0011^{\pm.0002}$	$.100^{\pm.001}$	$.039^{\pm.001}$	$.096^{\pm.000}$

Table 4. Results of deep generative models in the loss (↓).

Method	CASIA VAE	CASIA DDPM	Celeb DDPM
Offline	$.664^{\pm.018}$	$.0451^{\pm.0022}$	$.0438^{\pm.0019}$
Online	$.862^{\pm.009}$	$.1408^{\pm.0032}$	$.2124^{\pm.0025}$
OML	$.442^{\pm.003}$	$.0353^{\pm.0001}$	$.0308^{\pm.0003}$
OML-Rep	$.454^{\pm.000}$	$.0353^{\pm.0001}$	$.0307^{\pm.0004}$
SB-MCL	$\mathbf{.428}^{\pm.001}$	$\mathbf{.0345}^{\pm.0001}$	$\mathbf{.0302}^{\pm.0004}$

and regression. Each class (e.g., character type or celebrity identity) is defined as a distinct task. High-level descriptions of each benchmark are provided below. We also provide visual illustrations of the model architectures used for each benchmark in Appendix B.

Image Classification. We conduct experiments with the Omniglot, CASIA, and Celeb datasets, following the setups of Lee et al. (2023). All the methods share the same CNN encoder with five convolutional layers. GeMCL is compared as an instance of SB-MCL.

Sine Regression. We adopt the synthetic sine wave regression setting from Lee et al. (2023). ALPaCA is tested as an instance of SB-MCL.

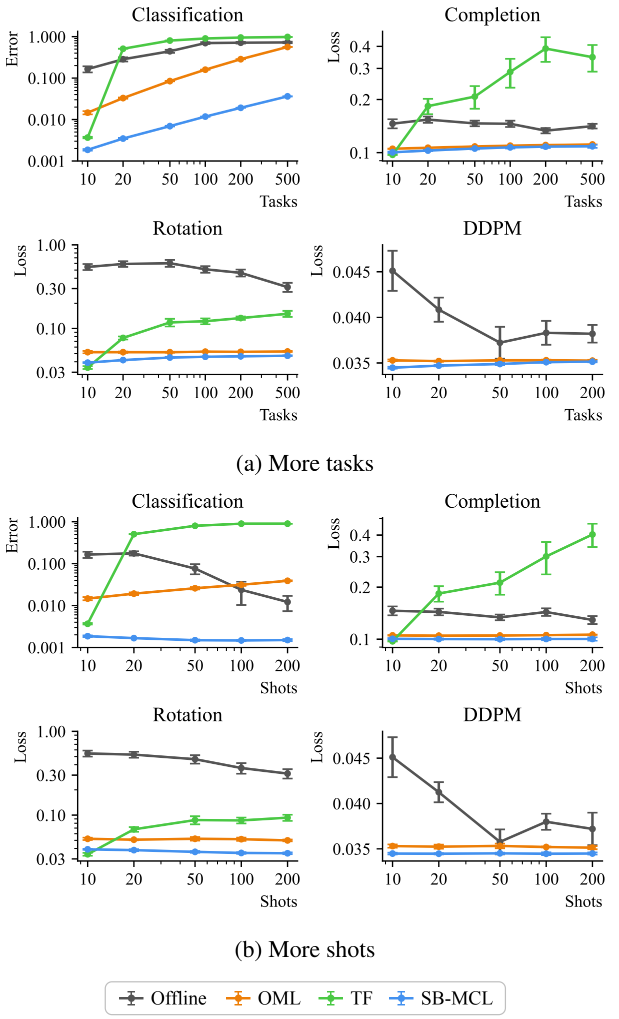
Image Completion (Compl.). x and y are an image's top and bottom halves, and each class is defined as a task. We use the convolutional encoder-decoder architecture from Lee et al. (2023). In the case of SB-MCL, we use the factorized Gaussian posterior and introduce another five-layer convolutional encoder for the learner, which produces $q_\phi(x, y|z)$ from a full training image. The model's decoder is slightly modified to take the concatenation of the encoder's output and z as input.

Rotation Prediction. A model is given a randomly rotated image x and tasked to predict the rotation angle y. Although the rotation angle is not high-dimensional, we use the generic supervised SB-MCL architecture as in the image completion task. This is due to the objective function, which is defined as $1 - \cos(y - \hat{y})$ and cannot be used for analytically computing the posterior of the linear model in ALPaCA. For the architecture, we use a convolutional encoder followed by an MLP output module. For the learner in SB-MCL, we share the same encoder in the model for encoding x and introduce a new MLP to encode y. These two encoders' outputs are concatenated and fed to another MLP to produce $q_\phi(x, y|z)$.

Deep Generative Modeling. As the first in MCL research, we evaluate MCL performance with deep generative models. We evaluate unsupervised learning performances with two types of deep generative models: variational autoencoder (VAE; Kingma & Welling, 2014) and denoising diffusion probabilistic models (DDPM; Ho et al., 2020). We use a simple convolutional encoder-decoder architecture for VAE and a U-Net encoder-decoder architecture for DDPM following Ho et al. (2020). In SB-MCL, we use a separate encoder for the learner, and z is injected into the model by concatenating it with the decoder's input. For OML, we replace the encoder's last MLP and the decoder's first MLP with a MAML MLP. Transformers are not tested in this setting since they are not straightforward to be combined with deep generative models.

Evaluation Scheme. In all MCL experiments, we meta-train the methods in a 10-task 10-shot setting: each training stream is a concatenation of 10 tasks with 10 examples each. We primarily evaluate their performance in a meta-test set with the same task-shot setting, while also measuring the generalization capability on other meta-testing setups. The hyperparameters are tuned to maximize the performance in the 10-task 10-shot settings. We report classification errors for the classification benchmarks and losses for others. Therefore, lower scores are always better. For each experiment, we report the average and the standard deviation of five runs. Within each MCL run, we calculate the average score from 512 CL episodes sampled from the meta-test set. For offline and online learning, which do not involve any meta-training, we sample an episode from the meta-test set, train the model on the training set, and measure the test score. We repeat this process 20 times and report the average and standard error of the mean.

5.3. Results and Analyses



(a) More tasks

(b) More shots

Figure 3. Generalization to longer training streams with more tasks and shots after meta-training with 10-task 10-shot on CASIA.

We present our classification, regression, and deep generative modeling results in Table 2, 3, and 4, respectively. Fig. 3 compares the generalization abilities in longer training streams, while Table 5 summarizes generalization to a different dataset. For qualitative examples and more extensive results, please refer to Appendix C and D. We discuss several notable characteristics of our SB-MCL that can be observed in the experiments.

Strong CL Performance. In the classification, regression, and generation experiments (Table 2-4), the SB-MCL family significantly outperforms SGD-based approaches and Linear TF. Its performance is comparable to TF, whose per-example computational cost constantly grows with the number of learned examples.

Table 5. Generalization to another dataset. Meta-test scores on Omniglot after meta-training on CASIA.

Method	Classification	Rotation	VAE	DDPM
OML	$.445^{\pm.020}$	$.856^{\pm.074}$	$.227^{\pm.002}$	$.027^{\pm.000}$
OML-Rep	$.496^{\pm.023}$	$.736^{\pm.010}$	$.244^{\pm.001}$	$.027^{\pm.000}$
TF	$.088^{\pm.010}$	$.850^{\pm.015}$	–	–
Linear TF	$.102^{\pm.011}$	$.931^{\pm.031}$	–	–
SB-MCL	$\mathbf{.023}^{\pm.001}$	$\mathbf{.640}^{\pm.012}$	$\mathbf{.219}^{\pm.001}$	$\mathbf{.026}^{\pm.000}$

Table 6. Meta-training time comparison. We report the time required to meta-train for 50K steps with a single A40 GPU.

Method	OML	TF	SB-MCL
Classification	6.5 hr	1.2 hr	**40 min**
Completion	16.5 hr	1.4 hr	**1.2 hr**
VAE	19 hr	N/A	**1.2 hr**
DDPM	5 days	N/A	**8 hr**

Stronger Generalization Ability. When meta-tested on longer training streams (Fig. 3) or a different dataset (Table 5), SB-MCL achieves substantially better scores than all the other baselines. Especially, TF's performance degrades catastrophically due to its poor length generalization ability, which is a well-known limitation of TF (Anil et al., 2022). Another interesting point is that TF and OML's performance can degrade even when provided with more shots and the same number of tasks as presented in Fig. 3b. This may seem counterintuitive, as providing more information about a task without adding more tasks should generally be beneficial. In SGD-based MCL, however, the longer training stream results in more SGD updates, which can exacerbate forgetting. TF's performance deteriorates even more dramatically due to length generalization failure. On the other hand, the SB-MCL family demonstrates a remarkable level of robustness in many-shot settings. As the number of shots increases, their performance even improves slightly. This observation aligns with our formulation. Since our posterior follows an exponential family distribution with fixed-sized sufficient statistics, maintaining the same number of tasks while increasing the number of shots serves only to enhance the accuracy of the variational posterior.

Superior Efficiency. In Table 6, we compare the meta-training time of the SB-MCL family against OML and TF. First of all, SB-MCL and TF are significantly faster than OML, which does not support parallel training. Parallel training is essential for utilizing parallel processors like GPUs for efficient meta-training. SB-MCL is faster than TF in all the benchmarks, demonstrating its superior efficiency due to the constant computational cost of the Bayesian update.

CL as a Matter of Representational Capacity. By design, SB-MCL yields the same results regardless of whether the training data is provided sequentially or not; in other words, *no forgetting* is theoretically guaranteed. This unique property enables new approaches to CL; instead of dealing with the complex learning dynamics of SGD on a non-stationary training stream, we can focus on maximizing the representational capacity. This includes designing better/bigger architectures and collecting more data, just like solving ordinary deep-learning problems in offline settings. Note that this has not been possible with SGD-based approaches since their CL performance is not necessarily aligned with the representational capacity due to the complicated dynamics of forgetting.

6. Conclusion

This work introduces a general MCL framework that combines the exponential family's robustness to forgetting and the flexibility of neural networks. Its superior performance and efficiency are empirically demonstrated in diverse domains. Unifying several prior works under the same framework, we aim to establish a solid foundation for the future sequential Bayesian approaches in the field of MCL. As discussed in §5.3, our framework reframes CL's forgetting issue as a matter of representational capacity. This allows us to focus on the architectural aspect, rather than the optimization aspect of preventing forgetting. Designing neural architectures for interacting with the exponential family posterior can be an exciting avenue for further research. Collecting new datasets for MCL also arises as an important future direction. While our method can benefit from large-scale data, few datasets are available for MCL research at the moment. We believe our approach can enable interesting applications when combined with appropriate datasets.

Limitation

While our framework demonstrates strong performance across various MCL tasks, it faces a fundamental limitation due to the assumption of an exponential family posterior. The equivalence between the sequential update rule and batch learning, while preventing forgetting, completely disregards the order of training data. This is acceptable and even beneficial when data order is irrelevant, as observed in the standard CL benchmarks used in our experiments. However, in real-world applications, the sequence of training data can be crucial. For instance, training data may be organized into a curriculum where acquiring new knowledge depends on previously learned information. In such scenarios, our framework may not be the optimal choice.

Our research began with the constraint of maintaining a constant memory size throughout the learning process. The Fisher-Darmois-Koopman-Pitman theorem indicates that only an exponential family posterior can prevent forgetting under this constraint. By relaxing this constraint, we could explore more flexible, non-parametric posterior distributions. We propose this as an intriguing direction for future research.

Impact Statement

This paper contributes to the field of machine learning, specifically in continual learning. While recognizing the potential societal consequences of our work, we conclude that no particular aspects demand specific highlighting.

Acknowledgements

This work was supported by Samsung Advanced Institute of Technology and the Institute of Information & communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (No. RS-2022-II220156, Fundamental research on continual meta-learning for quality enhancement of casual videos and their 3D metaverse transformation; No. RS-2019-II191082, SW StarLab; No. RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University)).

References

Anil, C., Wu, Y., Andreassen, A., Lewkowycz, A., Misra, V., Ramasesh, V. V., Slone, A., Gur-Ari, G., Dyer, E., and Neyshabur, B. Exploring length generalization in large language models. In *NeurIPS*, 2022.

Banayeeanzade, M., Mirzaiezadeh, R., Hasani, H., and Soleymani, M. Generative vs. discriminative: Rethinking the meta-continual learning. In *NeurIPS*, 2021.

Beaulieu, S., Frati, L., Miconi, T., Lehman, J., Stanley, K. O., Clune, J., and Cheney, N. Learning to continually learn. In *ECAI*, 2020.

Bishop, C. M. *Pattern Recognition and Machine Learning*. Berlin, Heidelberg, 2006. ISBN 0387310738.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *NeurIPS*, 2020.

Chaudhry, A., Dokania, P. K., Ajanthan, T., and Torr, P. H. S. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*, 2018.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H. S., and Ranzato, M. Continual learning with tiny episodic memories. *CoRR*, abs/1902.10486, 2019.

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlós, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D. B., Colwell, L. J., and Weller, A. Rethinking attention with Performers. In *ICLR*, 2021.

Darmois, G. Sur les lois de probabilitéa estimation exhaustive. *CR Acad. Sci. Paris*, 260(1265):85, 1935.

Farajtabar, M., Azizan, N., Mott, A., and Li, A. Orthogonal gradient descent for continual learning. In *AISTATS*, 2020.

Farquhar, S. and Gal, Y. A unifying Bayesian view of continual learning. *CoRR*, abs/1902.06494, 2019.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Finn, C., Rajeswaran, A., Kakade, S. M., and Levine, S. Online meta-learning. In *ICML*, 2019.

Fisher, R. A. Two new properties of mathematical likelihood. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 144(852):285–307, 1934.

Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D. J., and Eslami, S. M. A. Conditional neural processes. In *ICML*, 2018a.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S. M. A., and Teh, Y. W. Neural processes. *CoRR*, abs/1807.01622, 2018b.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. Generative adversarial nets. In *NeurIPS*, 2014.

Gordon, J., Bronskill, J., Bauer, M., Nowozin, S., and Turner, R. E. Meta-learning probabilistic inference for prediction. In *ICLR*, 2019.

Guo, Y., Zhang, L., Hu, Y., He, X., and Gao, J. MS-Celeb-1M: A dataset and benchmark for large-scale face recognition. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III*, 2016.

Gupta, G., Yadav, K., and Paull, L. Look-ahead meta learning for continual learning. In *NeurIPS*, 2020.

Harrison, J., Sharma, A., and Pavone, M. Meta-learning priors for efficient online Bayesian regression. In *Algorithmic Foundations of Robotics XIII, Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics, WAFR 2018, Mérida, Mexico, December 9-11, 2018*, 2018.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Javed, K. and White, M. Meta-learning representations for continual learning. In *NeurIPS*, 2019.

Jerfel, G., Grant, E., Griffiths, T., and Heller, K. A. Reconciling meta-learning and continual learning with online mixtures of tasks. In *NeurIPS*, 2019.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are RNNs: Fast autoregressive Transformers with linear attention. In *ICML*, 2020.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *ICLR*, 2014.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N. C., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796, 2016.

Knoblauch, J., Husain, H., and Diethe, T. Optimal continual learning has perfect memory and is NP-hard. In *ICML*, 2020.

Koopman, B. O. On distributions admitting a sufficient statistic. *Transactions of the American Mathematical society*, 39(3):399–409, 1936.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, 350:1332 – 1338, 2015.

Lee, S., Son, J., and Kim, G. Recasting continual learning as sequence modeling. In *NeurIPS*, 2023.

Liu, C., Yin, F., Wang, D., and Wang, Q. CASIA online and offline chinese handwriting databases. In *ICDAR*, 2011.

Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning. In *NeurIPS*, 2017.

Murphy, K. P. *Probabilistic Machine Learning: An introduction*. 2022.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. Variational continual learning. In *ICLR*, 2018.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Pitman, E. J. G. Sufficient statistics and intrinsic accuracy. In *Mathematical Proceedings of the Cambridge Philosophical Society*, number 4. Cambridge University Press, 1936.

Requeima, J., Gordon, J., Bronskill, J., Nowozin, S., and Turner, R. E. Fast and flexible multi-task classification using conditional neural adaptive processes. In *NeurIPS*, 2019.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., and Tesauro, G. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *ICLR*, 2019.

Snell, J., Swersky, K., and Zemel, R. S. Prototypical networks for few-shot learning. In *NeurIPS*, 2017.

Son, J., Lee, S., and Kim, G. When meta-learning meets online and continual learning: A survey. *CoRR*, abs/2311.05241, 2023.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. Efficient Transformers: A survey. *ACM Comput. Surv.*, 55(6), 2022. ISSN 0360-0300.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

Volpp, M., Flürenbrock, F., Großberger, L., Daniel, C., and Neumann, G. Bayesian context aggregation for neural processes. In *ICLR*, 2021.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In *ICML*, 2017.

A. Variational Bound Derivation

The derivation of the variational bound for supervised learning setup (Eq. 1) is as follows:

$$
\begin{aligned}
&\log p_\theta(\tilde{y}_{1:N} | \tilde{x}_{1:N}, \mathcal{D}) \\
&= -\log p_\theta(z | \tilde{y}_{1:N}, \tilde{x}_{1:N}, \mathcal{D}) + \log p_\theta(\tilde{y}_{1:N}, z | \tilde{x}_{1:N}, \mathcal{D}) \\
&= \mathbb{E}_{z \sim q_\phi(z|\mathcal{D})} \left[\log q_\phi(z|\mathcal{D}) - \log p_\theta(z | \tilde{y}_{1:N}, \tilde{x}_{1:N}, \mathcal{D}) + \log p_\theta(\tilde{y}_{1:N}, z | \tilde{x}_{1:N}, \mathcal{D}) - \log q_\phi(z|\mathcal{D}) \right] \\
&= D_{\mathrm{KL}} \left(q_\phi(z|\mathcal{D}) \,\|\, p_\theta(z | \tilde{y}_{1:N}, \tilde{x}_{1:N}, \mathcal{D}) \right) + \mathbb{E}_{z \sim q_\phi(z|\mathcal{D})} \left[\log p_\theta(\tilde{y}_{1:N}, z | \tilde{x}_{1:N}, \mathcal{D}) - \log q_\phi(z|\mathcal{D}) \right] \\
&\geq \mathbb{E}_{z \sim q_\phi(z|\mathcal{D})} \left[\log p_\theta(\tilde{y}_{1:N}, z | \tilde{x}_{1:N}, \mathcal{D}) - \log q_\phi(z|\mathcal{D}) \right] \\
&= \mathbb{E}_{z \sim q_\phi(z|\mathcal{D})} \left[\log p_\theta(\tilde{y}_{1:N} | z, \tilde{x}_{1:N}) + \log p_\theta(z | \tilde{x}_{1:N}, \mathcal{D}) - \log q_\phi(z|\mathcal{D}) \right] \\
&= \mathbb{E}_{z \sim q_\phi(z|\mathcal{D})} \left[\log p_\theta(\tilde{y}_{1:N} | z, \tilde{x}_{1:N}) + \log p_\theta(\mathcal{D} | z, \tilde{x}_{1:N}) + \log p_\theta(z | \tilde{x}_{1:N}) - \log p_\theta(\mathcal{D} | \tilde{x}_{1:N}) \right. \\
&\qquad\qquad \left. - \log q_\phi(z|\mathcal{D}) \right] \\
&= \mathbb{E}_{z \sim q_\phi(z|\mathcal{D})} \left[\log p_\theta(\tilde{y}_{1:N} | z, \tilde{x}_{1:N}) + \log p_\theta(\mathcal{D} | z) + \log p_\theta(z) - \log p_\theta(\mathcal{D}) - \log q_\phi(z|\mathcal{D}) \right] \\
&= \mathbb{E}_{z \sim q_\phi(z|\mathcal{D})} \left[\log p_\theta(\tilde{y}_{1:N} | z, \tilde{x}_{1:N}) + \log p_\theta(\mathcal{D} | z) \right] - D_{\mathrm{KL}} \left(q_\phi(z|\mathcal{D}) \,\|\, p_\theta(z) \right) - \log p_\theta(\mathcal{D}) \\
&= \mathbb{E}_{z \sim q_\phi(z|\mathcal{D})} \left[\sum_{n=1}^{N} \log p_\theta(\tilde{y}_n | \tilde{x}_n, z) + \sum_{t=1}^{T} \log p_\theta(y_t | x_t, z) \right] \\
&\quad - D_{\mathrm{KL}} \left(q_\phi(z|\mathcal{D}) \,\|\, p_\theta(z) \right) - \underbrace{\log p_\theta(\mathcal{D})}_{\text{const.}}
\end{aligned}
\tag{8}
$$

We can derive a similar bound for unsupervised settings (Eq. 2):

$$
\begin{aligned}
&\log p_\theta(\tilde{x}_{1:N} | \mathcal{D}) \\
&= -\log p_\theta(z | \tilde{x}_{1:N}, \mathcal{D}) + \log p_\theta(\tilde{x}_{1:N}, z | \mathcal{D}) \\
&= \mathbb{E}_{z \sim q_\phi(z|\mathcal{D})} \left[\log q_\phi(z|\mathcal{D}) - \log p_\theta(z | \tilde{x}_{1:N}, \mathcal{D}) + \log p_\theta(\tilde{x}_{1:N}, z | \mathcal{D}) - \log q_\phi(z|\mathcal{D}) \right] \\
&= D_{\mathrm{KL}} \left(q_\phi(z|\mathcal{D}) \,\|\, p_\theta(z|\mathcal{D}) \right) + \mathbb{E}_{z \sim q_\phi(z|\mathcal{D})} \left[\log p_\theta(\tilde{x}_{1:N}, z | \mathcal{D}) - \log q_\phi(z|\mathcal{D}) \right] \\
&\geq \mathbb{E}_{z \sim q_\phi(z|\mathcal{D})} \left[\log p_\theta(\tilde{x}_{1:N}, z | \mathcal{D}) - \log q_\phi(z|\mathcal{D}) \right] \\
&= \mathbb{E}_{z \sim q_\phi(z|\mathcal{D})} \left[\log p_\theta(\tilde{x}_{1:N} | z, \mathcal{D}) + \log p_\theta(z|\mathcal{D}) - \log q_\phi(z|\mathcal{D}) \right] \\
&= \mathbb{E}_{z \sim q_\phi(z|\mathcal{D})} \left[\log p_\theta(\tilde{x}_{1:N} | z) + \log p_\theta(\mathcal{D}|z) + \log p_\theta(z) - \log p_\theta(\mathcal{D}) - \log q_\phi(z|\mathcal{D}) \right] \\
&= \mathbb{E}_{z \sim q_\phi(z|\mathcal{D})} \left[\log p_\theta(\tilde{x}_{1:N} | z) + \log p_\theta(\mathcal{D}|z) \right] - D_{\mathrm{KL}} \left(q_\phi(z|\mathcal{D}) \,\|\, p_\theta(z) \right) - \log p_\theta(\mathcal{D}) \\
&= \mathbb{E}_{z \sim q_\phi(z|\mathcal{D})} \left[\sum_{n=1}^{N} \log p_\theta(\tilde{x}_n | z) + \sum_{t=1}^{T} \log p_\theta(x_t | z) \right] - D_{\mathrm{KL}} \left(q_\phi(z|\mathcal{D}) \,\|\, p_\theta(z) \right) - \underbrace{\log p_\theta(\mathcal{D})}_{\text{const.}}
\end{aligned}
$$

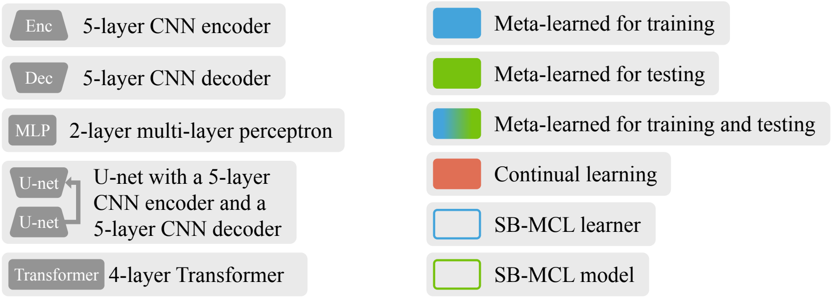
It is noteworthy that Neural Process (Garnelo et al., 2018b) instead approximates $\log p_\theta(z|\tilde{x}_{1:N}, \mathcal{D})$ of Eq. 8 with $\log q_\phi(z|\tilde{x}_{1:N}, \mathcal{D})$:

$$
\begin{aligned}
&\mathbb{E}_{z \sim q_\phi(z|\mathcal{D})} \left[\log p_\theta(\tilde{y}_{1:N} | z, \tilde{x}_{1:N}) + \log p_\theta(z | \tilde{x}_{1:N}, \mathcal{D}) - \log q_\phi(z|\mathcal{D}) \right] \\
&\approx \mathbb{E}_{z \sim q_\phi(z|\mathcal{D})} \left[\log p_\theta(\tilde{y}_{1:N} | z, \tilde{x}_{1:N}) + \log q_\phi(z | \tilde{x}_{1:N}, \mathcal{D}) - \log q_\phi(z|\mathcal{D}) \right] \\
&= \mathbb{E}_{z \sim q_\phi(z|\mathcal{D})} \left[\sum_{n=1}^{N} \log p_\theta(\tilde{y}_n | \tilde{x}_n, z) \right] - D_{\mathrm{KL}} \left(q_\phi(z|\mathcal{D}) \| q_\phi(z | \tilde{x}_{1:N}, \mathcal{D}) \right)
\end{aligned}
$$

Since we can use the Bayes rule to convert $\log p_\theta(z|\tilde{x}_{1:N}, \mathcal{D})$ into $\log p_\theta(\mathcal{D}|z, \tilde{x}_{1:N}) + \log p_\theta(z|\tilde{x}_{1:N}) - \log p_\theta(\mathcal{D}|\tilde{x}_{1:N})$, which is subsequently reduced to $\log p_\theta(\mathcal{D}|z) + \log p_\theta(z) - \log p_\theta(\mathcal{D})$ by conditional independence, such an approximation is not necessary.

B. Architecture Diagrams

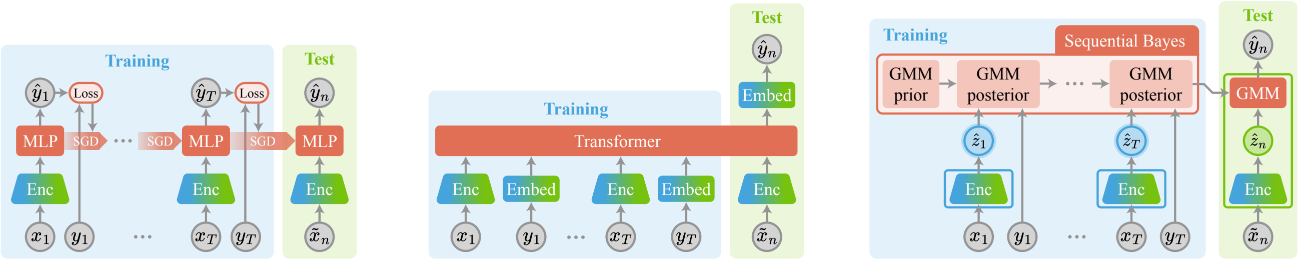
For better understanding of the architectures used in the experiments, we provide detailed diagrams for each experiment. In Fig. 4, we present the notations used in the architecture diagrams. In Fig. 5-9, we present the architectures used in the classification, rotation, completion, VAE, and DDPM experiments, respectively.



(a) Neural network components

(b) Color coding

Figure 4. Notations for architecture diagrams.

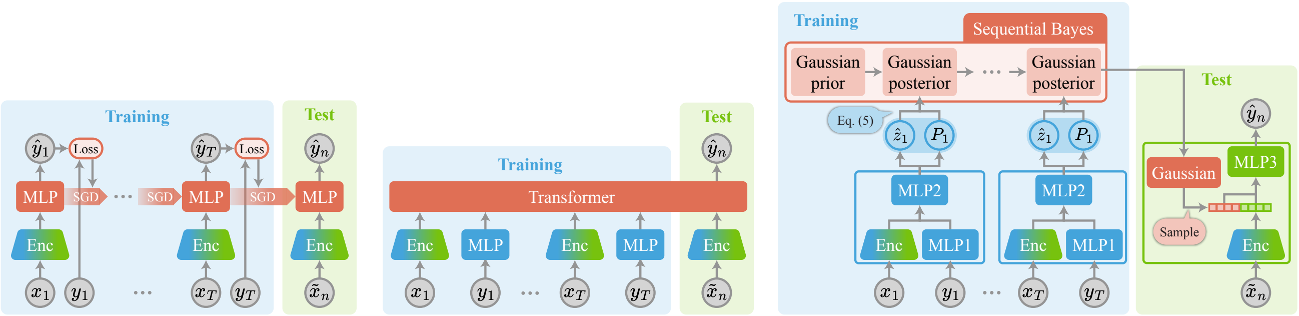


(a) OML

(b) TF

(c) SB-MCL (GeMCL)

Figure 5. Architectures for classification experiments.

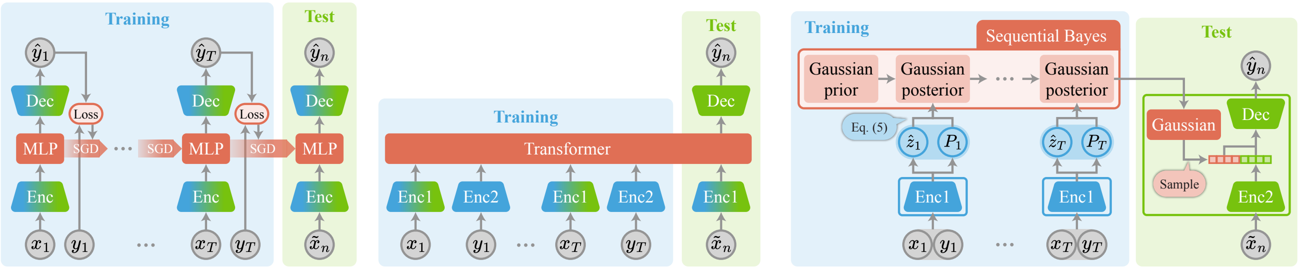


(a) OML

(b) TF

(c) SB-MCL

Figure 6. Architectures for rotation experiments.

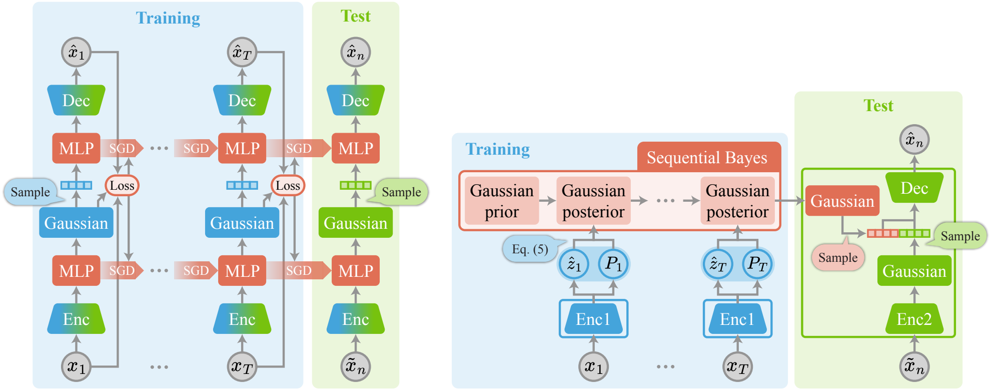


(a) OML

(b) TF

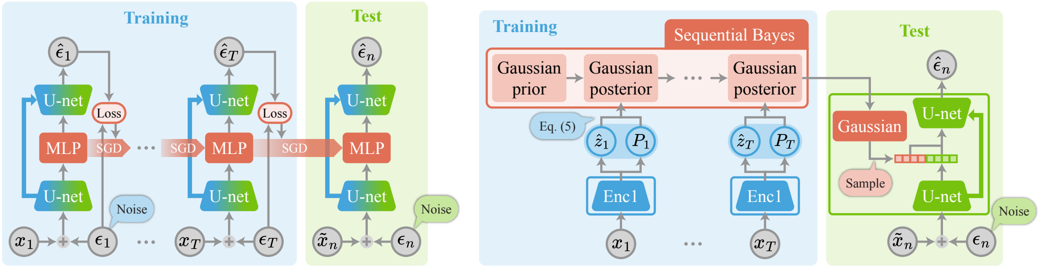
(c) SB-MCL

Figure 7. Architectures for completion experiments.



(a) OML (b) SB-MCL

Figure 8. Architectures for VAE experiments.



(a) OML (b) SB-MCL

Figure 9. Architectures for DDPM experiments.

C. Qualitative Examples of Deep Generative MCL

In Fig. 10-15, we present qualitative examples of the deep generative model experiments. For VAEs, we use a binarized CASIA dataset for easier likelihood calculation, while using unmodified CASIA and MS-Celeb-1M datasets for DDPMs. With each meta-trained MCL method, we train a VAE or DPMM on a 5-task 10-shot training stream in Fig. 10 or 11, which are sampled from the meta-test set. Then, we extract 20 generation samples for the VAE (Fig. 13) and the DDPM (Fig. 14 and 15). For the VAE, we also visualize the reconstructions of the test images in Fig. 12.

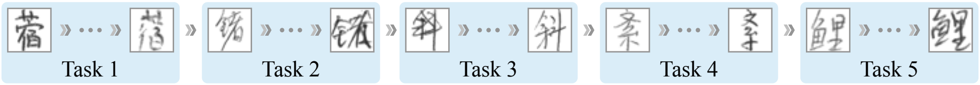


Figure 10. An example training stream from CASIA.

Although the scores of OML and OML-Reptile are much worse than SB-MCL, the reconstruction results in Fig. 12 do not seem to show a significant difference. However, the generation results in Fig. 13 of OML and OML-Reptile are not properly structured, showing that OML and OML-Reptile have difficulty in training VAE on a non-stationary stream. On the other hand, the VAE with SB-MCL produces significantly better samples, demonstrating the effectiveness of our approach.

All the DDPM samples in Fig. 14 and 15 are of much higher quality compared to VAE and are hard to distinguish from real images. Since the DDPMs meta-learn general concepts from the large-scale meta-training set, they can produce high-fidelity images. The key difference to notice is whether the DDPM has learned new knowledge from the training stream. Since the training stream is from the meta-test set, it cannot produce the classes in the training stream unless it actually learns from it. Among the samples from OML and OML-Reptile, it is hard to find the classes in the training stream, suggesting that they are producing samples from the meta-training distribution. On the other hand, the DDPMs with SB-MCL produce samples remarkably similar to the ones in Fig. 10 and 11. This experiment confirms that SB-MCL can be an effective solution for

Figure 11. An example training stream from Celeb.

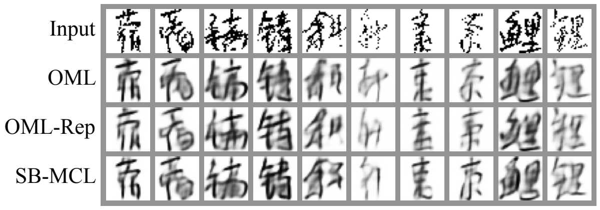
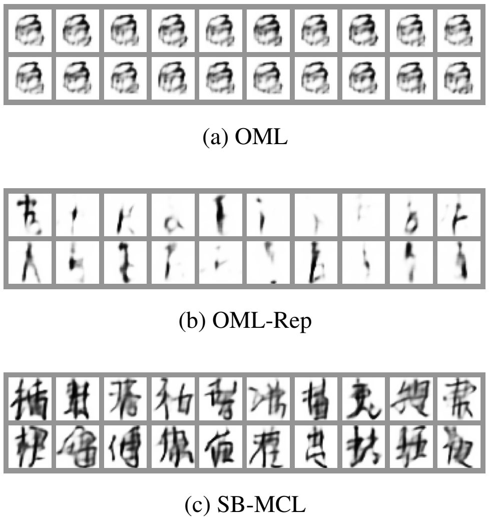


Figure 12. VAE reconstruction samples (CASIA).


(a) OML

(b) OML-Rep

(c) SB-MCL

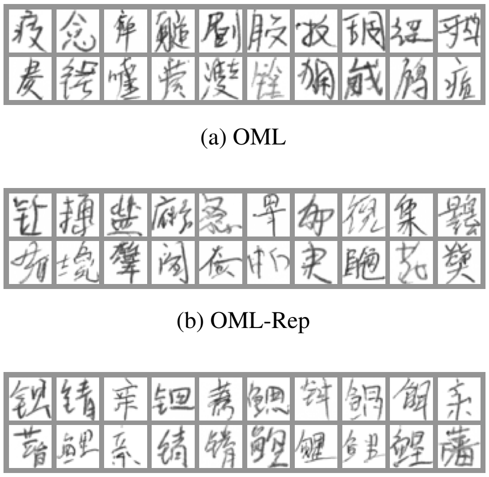
Figure 13. VAE generation samples (CASIA).

modern deep generative models.

D. Extended Experimental Results

Table 7. CASIA classification with more tasks.

Method	Tasks					
	10	20	50	100	200	500
Offline	$.165^{\pm.028}$	$.284^{\pm.033}$	$.444^{\pm.038}$	$.700^{\pm.038}$	$.714^{\pm.034}$	$.725^{\pm.031}$
Online	$.963^{\pm.020}$	$.925^{\pm.031}$	$.963^{\pm.020}$	$.963^{\pm.020}$	$.963^{\pm.013}$	$.970^{\pm.007}$
OML	$.015^{\pm.001}$	$.033^{\pm.001}$	$.085^{\pm.001}$	$.159^{\pm.001}$	$.286^{\pm.002}$	$.564^{\pm.001}$
OML-Rep	$.057^{\pm.003}$	$.104^{\pm.002}$	$.215^{\pm.004}$	$.359^{\pm.002}$	$.559^{\pm.005}$	$.796^{\pm.003}$
TF	$.004^{\pm.000}$	$.510^{\pm.001}$	$.804^{\pm.001}$	$.903^{\pm.001}$	$.952^{\pm.000}$	$.980^{\pm.000}$
SB-MCL	$.002^{\pm.000}$	$.003^{\pm.000}$	$.007^{\pm.000}$	$.012^{\pm.000}$	$.019^{\pm.000}$	$.036^{\pm.000}$
SB-MCL (MAP)	$.002^{\pm.000}$	$.003^{\pm.000}$	$.007^{\pm.000}$	$.012^{\pm.000}$	$.019^{\pm.000}$	$.036^{\pm.000}$

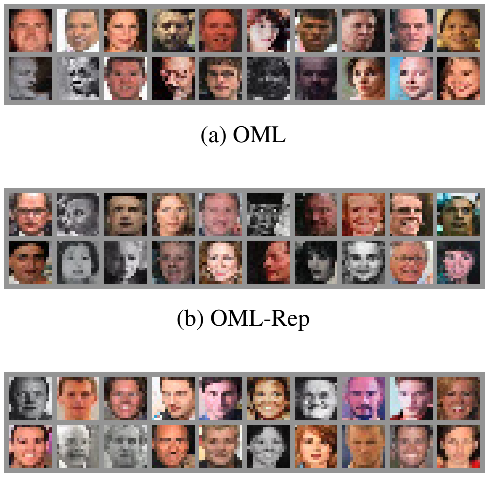


(a) OML

(b) OML-Rep

(c) SB-MCL

Figure 14. DDPM generation samples (CASIA).



(a) OML

(b) OML-Rep

(c) SB-MCL

Figure 15. DDPM generation samples (Celeb).

Table 8. CASIA classification with more shots.

Method	Shots				
	10	20	50	100	200
Offline	.165$^{\pm.028}$	.176$^{\pm.021}$	.076$^{\pm.021}$	.024$^{\pm.013}$	.012$^{\pm.005}$
Online	.963$^{\pm.020}$	.838$^{\pm.032}$	.662$^{\pm.041}$	.550$^{\pm.074}$	.388$^{\pm.065}$
OML	.015$^{\pm.001}$	.019$^{\pm.001}$	.026$^{\pm.002}$	.031$^{\pm.002}$	.039$^{\pm.001}$
OML-Rep	.057$^{\pm.003}$	.066$^{\pm.002}$	.083$^{\pm.004}$	.101$^{\pm.002}$	.121$^{\pm.003}$
TF	.004$^{\pm.000}$	.505$^{\pm.001}$	.800$^{\pm.000}$	.899$^{\pm.001}$	.899$^{\pm.000}$
Linear TF	.006$^{\pm.000}$	.530$^{\pm.010}$	.768$^{\pm.028}$	.804$^{\pm.031}$	.818$^{\pm.038}$
SB-MCL	.002$^{\pm.000}$	.002$^{\pm.000}$	.001$^{\pm.000}$	.001$^{\pm.000}$	.002$^{\pm.000}$
SB-MCL (MAP)	.002$^{\pm.000}$	.002$^{\pm.000}$	.001$^{\pm.000}$	.001$^{\pm.000}$	.001$^{\pm.000}$

Table 9. Sine classification with more tasks.

Method	Tasks					
	10	20	50	100	200	500
Offline	.005$^{\pm.000}$	.004$^{\pm.001}$	.005$^{\pm.001}$	.008$^{\pm.001}$	.036$^{\pm.008}$	.198$^{\pm.021}$
Online	.550$^{\pm.037}$	.525$^{\pm.032}$	.590$^{\pm.030}$	.549$^{\pm.031}$	.526$^{\pm.022}$	.569$^{\pm.013}$
OML	.016$^{\pm.001}$	.034$^{\pm.002}$	.082$^{\pm.001}$	.153$^{\pm.002}$	.270$^{\pm.000}$	.484$^{\pm.002}$
OML-Rep	.027$^{\pm.001}$	.054$^{\pm.002}$	.115$^{\pm.003}$	.201$^{\pm.004}$	.335$^{\pm.005}$	.559$^{\pm.003}$
TF	.001$^{\pm.000}$	.238$^{\pm.020}$	.454$^{\pm.011}$	.535$^{\pm.011}$	.586$^{\pm.013}$	.615$^{\pm.006}$
Linear TF	.003$^{\pm.000}$	.201$^{\pm.011}$	.409$^{\pm.011}$	.489$^{\pm.006}$	.526$^{\pm.003}$	.543$^{\pm.002}$
SB-MCL	.001$^{\pm.000}$	.002$^{\pm.000}$	.007$^{\pm.000}$	.020$^{\pm.000}$	.065$^{\pm.001}$	.228$^{\pm.001}$

Table 10. Sine classification with more shots.

Method	Shots				
	10	20	50	100	200
Offline	$.005^{\pm.000}$	$.003^{\pm.000}$	$.003^{\pm.000}$	$.002^{\pm.000}$	$.002^{\pm.000}$
Online	$.550^{\pm.037}$	$.446^{\pm.031}$	$.376^{\pm.031}$	$.273^{\pm.018}$	$.219^{\pm.017}$
OML	$.016^{\pm.001}$	$.018^{\pm.001}$	$.017^{\pm.001}$	$.017^{\pm.001}$	$.018^{\pm.001}$
OML-Rep	$.027^{\pm.001}$	$.027^{\pm.001}$	$.027^{\pm.002}$	$.027^{\pm.002}$	$.027^{\pm.002}$
TF	$.001^{\pm.000}$	$.152^{\pm.030}$	$.212^{\pm.044}$	$.221^{\pm.034}$	$.199^{\pm.039}$
Linear TF	$.003^{\pm.000}$	$.140^{\pm.012}$	$.212^{\pm.017}$	$.228^{\pm.026}$	$.252^{\pm.022}$
SB-MCL	$.001^{\pm.000}$	$.001^{\pm.000}$	$.001^{\pm.000}$	$.001^{\pm.000}$	$.001^{\pm.000}$

Table 11. CASIA completion with more tasks.

Method	Tasks					
	10	20	50	100	200	500
Offline	$.146^{\pm.009}$	$.154^{\pm.006}$	$.146^{\pm.005}$	$.146^{\pm.006}$	$.133^{\pm.005}$	$.141^{\pm.004}$
Online	$.290^{\pm.023}$	$.188^{\pm.007}$	$.163^{\pm.007}$	$.153^{\pm.007}$	$.153^{\pm.005}$	$.154^{\pm.003}$
OML	$.105^{\pm.000}$	$.107^{\pm.000}$	$.108^{\pm.000}$	$.110^{\pm.000}$	$.110^{\pm.000}$	$.111^{\pm.000}$
OML-Rep	$.104^{\pm.000}$	$.106^{\pm.000}$	$.107^{\pm.000}$	$.108^{\pm.000}$	$.108^{\pm.000}$	$.109^{\pm.000}$
TF	$.097^{\pm.000}$	$.183^{\pm.018}$	$.208^{\pm.031}$	$.287^{\pm.053}$	$.389^{\pm.062}$	$.347^{\pm.060}$
Linear TF	$.101^{\pm.000}$	$.125^{\pm.002}$	$.127^{\pm.002}$	$.128^{\pm.001}$	$.132^{\pm.002}$	$.132^{\pm.001}$
SB-MCL	$.100^{\pm.001}$	$.103^{\pm.001}$	$.106^{\pm.001}$	$.107^{\pm.002}$	$.108^{\pm.002}$	$.109^{\pm.002}$
SB-MCL (MAP)	$.100^{\pm.001}$	$.103^{\pm.001}$	$.106^{\pm.001}$	$.107^{\pm.002}$	$.108^{\pm.002}$	$.109^{\pm.002}$

Table 12. CASIA completion with more shots.

Method	Shots				
	10	20	50	100	200
Offline	$.146^{\pm.009}$	$.144^{\pm.006}$	$.134^{\pm.005}$	$.144^{\pm.007}$	$.129^{\pm.007}$
Online	$.290^{\pm.023}$	$.204^{\pm.008}$	$.151^{\pm.008}$	$.152^{\pm.008}$	$.156^{\pm.008}$
OML	$.105^{\pm.000}$	$.105^{\pm.000}$	$.105^{\pm.000}$	$.106^{\pm.000}$	$.106^{\pm.000}$
OML-Rep	$.104^{\pm.000}$	$.104^{\pm.000}$	$.105^{\pm.000}$	$.106^{\pm.000}$	$.107^{\pm.000}$
TF	$.097^{\pm.000}$	$.184^{\pm.019}$	$.212^{\pm.032}$	$.301^{\pm.064}$	$.403^{\pm.062}$
Linear TF	$.101^{\pm.000}$	$.123^{\pm.002}$	$.125^{\pm.002}$	$.126^{\pm.002}$	$.130^{\pm.002}$
SB-MCL	$.100^{\pm.001}$	$.100^{\pm.001}$	$.100^{\pm.001}$	$.100^{\pm.002}$	$.100^{\pm.002}$
SB-MCL (MAP)	$.100^{\pm.001}$	$.100^{\pm.001}$	$.100^{\pm.001}$	$.100^{\pm.002}$	$.100^{\pm.002}$

Table 13. CASIA rotation with more tasks.

Method	Tasks					
	10	20	50	100	200	500
Offline	$.544^{\pm.045}$	$.591^{\pm.047}$	$.603^{\pm.057}$	$.510^{\pm.046}$	$.463^{\pm.044}$	$.312^{\pm.039}$
Online	$1.079^{\pm.081}$	$.986^{\pm.073}$	$.862^{\pm.085}$	$.616^{\pm.040}$	$.810^{\pm.059}$	$.784^{\pm.029}$
OML	$.052^{\pm.002}$	$.052^{\pm.001}$	$.052^{\pm.001}$	$.053^{\pm.000}$	$.053^{\pm.000}$	$.053^{\pm.001}$
OML-Rep	$.050^{\pm.002}$	$.050^{\pm.001}$	$.052^{\pm.001}$	$.053^{\pm.001}$	$.055^{\pm.001}$	$.056^{\pm.001}$
TF	$.034^{\pm.001}$	$.077^{\pm.003}$	$.118^{\pm.012}$	$.122^{\pm.010}$	$.133^{\pm.006}$	$.150^{\pm.013}$
Linear TF	$.068^{\pm.002}$	$.078^{\pm.004}$	$.086^{\pm.003}$	$.087^{\pm.002}$	$.094^{\pm.005}$	$.091^{\pm.004}$
SB-MCL	$.039^{\pm.001}$	$.042^{\pm.000}$	$.045^{\pm.001}$	$.046^{\pm.000}$	$.047^{\pm.000}$	$.047^{\pm.001}$
SB-MCL (MAP)	$.040^{\pm.001}$	$.042^{\pm.001}$	$.045^{\pm.001}$	$.046^{\pm.000}$	$.047^{\pm.000}$	$.047^{\pm.000}$

Table 14. CASIA rotation with more shots.

Method	Shots				
	10	20	50	100	200
Offline	$.544^{\pm.045}$	$.527^{\pm.043}$	$.465^{\pm.054}$	$.365^{\pm.053}$	$.313^{\pm.040}$
Online	$1.079^{\pm.081}$	$.852^{\pm.062}$	$.916^{\pm.078}$	$.649^{\pm.062}$	$.668^{\pm.073}$
OML	$.052^{\pm.002}$	$.051^{\pm.001}$	$.052^{\pm.003}$	$.052^{\pm.002}$	$.050^{\pm.001}$
OML-Rep	$.050^{\pm.002}$	$.050^{\pm.001}$	$.047^{\pm.001}$	$.046^{\pm.001}$	$.045^{\pm.000}$
TF	$.034^{\pm.001}$	$.068^{\pm.004}$	$.087^{\pm.010}$	$.086^{\pm.007}$	$.093^{\pm.008}$
Linear TF	$.068^{\pm.002}$	$.073^{\pm.004}$	$.072^{\pm.003}$	$.075^{\pm.002}$	$.079^{\pm.006}$
SB-MCL	$.039^{\pm.001}$	$.038^{\pm.001}$	$.036^{\pm.001}$	$.035^{\pm.001}$	$.035^{\pm.001}$
SB-MCL (MAP)	$.040^{\pm.001}$	$.039^{\pm.001}$	$.036^{\pm.001}$	$.036^{\pm.001}$	$.035^{\pm.001}$

Table 15. CASIA VAE with more tasks.

Method	Tasks					
	10	20	50	100	200	500
Offline	$.664^{\pm.018}$	$.645^{\pm.027}$	$.590^{\pm.014}$	$.571^{\pm.012}$	$.594^{\pm.017}$	$.594^{\pm.012}$
Online	$.862^{\pm.009}$	$.801^{\pm.013}$	$.760^{\pm.013}$	$.775^{\pm.019}$	$.745^{\pm.007}$	$.736^{\pm.007}$
OML	$.442^{\pm.003}$	$.441^{\pm.003}$	$.440^{\pm.003}$	$.440^{\pm.003}$	$.440^{\pm.003}$	$.439^{\pm.003}$
OML-Rep	$.454^{\pm.000}$	$.455^{\pm.001}$	$.457^{\pm.001}$	$.457^{\pm.001}$	$.458^{\pm.001}$	$.459^{\pm.001}$
SB-MCL	$.428^{\pm.001}$	$.428^{\pm.001}$	$.429^{\pm.001}$	$.429^{\pm.001}$	$.429^{\pm.001}$	$.429^{\pm.001}$
SB-MCL (MAP)	$.428^{\pm.001}$	$.428^{\pm.001}$	$.429^{\pm.001}$	$.429^{\pm.001}$	$.429^{\pm.001}$	$.429^{\pm.001}$

Table 16. CASIA VAE with more shots.

Method	Shots				
	10	20	50	100	200
Offline	$.664^{\pm.018}$	$.580^{\pm.014}$	$.570^{\pm.018}$	$.564^{\pm.015}$	$.531^{\pm.014}$
Online	$.862^{\pm.009}$	$.805^{\pm.016}$	$.740^{\pm.027}$	$.780^{\pm.017}$	$.726^{\pm.017}$
OML	$.442^{\pm.003}$	$.440^{\pm.003}$	$.440^{\pm.003}$	$.440^{\pm.002}$	$.440^{\pm.003}$
OML-Rep	$.454^{\pm.000}$	$.455^{\pm.002}$	$.455^{\pm.002}$	$.456^{\pm.001}$	$.459^{\pm.001}$
SB-MCL	$.428^{\pm.001}$	$.428^{\pm.001}$	$.427^{\pm.001}$	$.428^{\pm.000}$	$.428^{\pm.002}$
SB-MCL (MAP)	$.428^{\pm.001}$	$.427^{\pm.001}$	$.428^{\pm.001}$	$.428^{\pm.001}$	$.428^{\pm.001}$

Table 17. CASIA DDPM with more tasks.

Method	Tasks					
	10	20	50	100	200	500
Offline	$.0451^{\pm.0022}$	$.0408^{\pm.0013}$	$.0372^{\pm.0017}$	$.0383^{\pm.0013}$	$.0382^{\pm.0010}$	$.0379^{\pm.0008}$
Online	$.1408^{\pm.0032}$	$.1090^{\pm.0020}$	$.0787^{\pm.0019}$	$.0698^{\pm.0016}$	$.0601^{\pm.0007}$	$.0511^{\pm.0004}$
OML	$.0353^{\pm.0001}$	$.0352^{\pm.0001}$	$.0353^{\pm.0001}$	$.0353^{\pm.0001}$	$.0353^{\pm.0001}$	$.0353^{\pm.0001}$
OML-Rep	$.0353^{\pm.0001}$	$.0353^{\pm.0001}$	$.0353^{\pm.0001}$	$.0353^{\pm.0001}$	$.0352^{\pm.0001}$	$.0352^{\pm.0001}$
SB-MCL	$.0345^{\pm.0001}$	$.0347^{\pm.0001}$	$.0349^{\pm.0001}$	$.0351^{\pm.0001}$	$.0351^{\pm.0001}$	$.0352^{\pm.0000}$
SB-MCL (MAP)	$.0345^{\pm.0001}$	$.0348^{\pm.0000}$	$.0350^{\pm.0001}$	$.0351^{\pm.0001}$	$.0352^{\pm.0000}$	$.0353^{\pm.0001}$

Table 18. CASIA DDPM with more shots.

Method			Shots		
	10	20	50	100	200
Offline	$.0451^{\pm.0022}$	$.0412^{\pm.0011}$	$.0358^{\pm.0014}$	$.0380^{\pm.0009}$	$.0372^{\pm.0018}$
Online	$.1408^{\pm.0032}$	$.1072^{\pm.0026}$	$.0826^{\pm.0029}$	$.0688^{\pm.0020}$	$.0590^{\pm.0016}$
OML	$.0353^{\pm.0002}$	$.0352^{\pm.0002}$	$.0353^{\pm.0003}$	$.0352^{\pm.0001}$	$.0351^{\pm.0002}$
OML-Rep	$.0353^{\pm.0001}$	$.0353^{\pm.0002}$	$.0352^{\pm.0001}$	$.0353^{\pm.0002}$	$.0352^{\pm.0001}$
SB-MCL	$.0345^{\pm.0001}$	$.0345^{\pm.0001}$	$.0345^{\pm.0001}$	$.0345^{\pm.0002}$	$.0345^{\pm.0001}$
SB-MCL (MAP)	$.0345^{\pm.0001}$	$.0345^{\pm.0001}$	$.0345^{\pm.0000}$	$.0344^{\pm.0001}$	$.0346^{\pm.0001}$