Continual learning via probabilistic exchangeable sequence modelling

Anonymous authors
Paper under double-blind review

Abstract

Continual learning (CL) refers to the ability to continuously learn and accumulate new knowledge while retaining useful information from past experiences. Although numerous CL methods have been proposed in recent years, it is not straightforward to deploy them directly to real-world decision-making problems due to their computational cost and lack of uncertainty quantification. To address these issues, we propose CL-BRUNO, a probabilistic, Neural Process-based CL model that performs scalable and tractable Bayesian update and prediction. Our proposed approach uses deep-generative models to create a unified probabilistic framework capable of handling different types of CL problems such as task- and class-incremental learning, allowing users to integrate information across different CL scenarios using a single model. Our approach is able to prevent catastrophic forgetting through distributional and functional regularisation without the need of retaining any previously seen samples, making it appealing to applications where data privacy or storage capacity is of concern. Experiments show that CL-BRUNO outperforms existing methods on both natural image and biomedical data sets, confirming its effectiveness in real-world applications.

1 Introduction

Continual learning (CL) enables an intelligent system to develop and refine itself adaptively in accordance with real-world dynamics by incrementally accumulating and exploiting knowledge gained from previous experience without the need to train any new model from scratch (Hassabis et al., 2017). The main challenge CL has to tackle is known as *catastrophic forgetting*, which refers to previously learned knowledge being drastically interfered by new information (McClelland et al., 1995; McCloskey and Cohen, 1989). In order to deliver accurate and trustworthy predictions, a CL model in practice should, on one hand, be able to integrate new knowledge efficiently based on the stream of new inputs from dynamic data distributions (learning plasticity) and, on the other hand, maximally retain relevant information from the past and prevent catastrophic forgetting (memory stability). The competition between these two conflicting objectives is known as *stability-plasticity dilemma*, which has been widely studied from both biological and computational perspectives (Ditzler et al., 2015; Parisi et al., 2019).

Numerous methods and strategies have been developed to address the CL problem under different scenarios (Wang et al., 2024): Regularisation-based methods (Kirkpatrick et al., 2017; Schwarz et al., 2018; Wang et al., 2021) aim to retain knowledge from history by introducing regularisation terms to balance the old and new tasks. Such regularisation can either be at a parameter level, e.g. minimising changes to key model parameters (under some data-driven parameter-wise importance measure) as new tasks are being learnt (Aljundi et al., 2018; Zenke et al., 2017), or at a functional level using e.g. knowledge distillation to prevent the model's performance on previous tasks from drastic deterioration (Michel et al., 2024; Rudner et al., 2022; Titsias et al., 2019). Architecture-based CL approaches (Thapa and Li, 2024; Gurbuz and Dovrolis, 2022; Mallya et al., 2018; Dhar et al., 2019) seek to mitigate catastrophic forgetting and inter-task interference by allocating specific sets of parameters to different tasks. Wortsman et al. (2020); Xue et al. (2022); Kang et al. (2022) use binary masks to select and isolate the subset of dedicated parameters for different tasks in a fixed-size model. Hung et al. (2019); Draelos et al. (2017) accommodate additional tasks by dynamically expanding the

model architecture, providing additional model capacity before the model "saturates" as more incremental tasks are introduced. Another popular approach is rehearsal-based methods (Robins, 1995) where historical information is retained by approximating and recovering historical data distributions. Wen et al. (2024); Shim et al. (2021); Aljundi et al. (2019); Rebuffi et al. (2017) retain distributional information by selecting and storing representative samples from the old data. Despite its conceptual simplicity, such approaches known as *experience replay* can be infeasible due to storage or data privacy constraints. *Generative replay* methods (Petit et al., 2023; Chen et al., 2022; Egorov et al., 2021; Shin et al., 2017) address this issue by summarising old data distributions as generative models, and replay generated data instead of the actual samples.

In practice, experience-replay methods such as Jha et al. (2024) and Wen et al. (2024) are computationally intensive and require storing historical data as part of their memory states, which could be infeasible in many applications due to privacy or storage concerns. Generative (Petit et al., 2023; Gopalakrishnan et al., 2022; Wu et al., 2018) and regularisation-based CL methods (Dohare et al., 2024; He and Zhu, 2022; Li and Hoiem, 2017) avoid the need of storing historical data, but are also computational costly and lack statistically principled updating or uncertainty quantification schemes. These limitations make it difficult to deploy them directly to real-world problems. To address these limitations, we propose Continual Learning Bayesian RecUrrent Neural mOdel (**CL-BRUNO**), a probabilistic, Neural Process-based CL model that performs scalable and tractable Bayesian update and prediction. CL-BRUNO utilises deep generative models, and extends the exchangeable sequence modelling framework given by Korshunova et al. (2020) (Conditional-BRUNO) to provide a versatile probabilistic framework capable of performing probabilistic label and task-identity estimation, and handling different CL scenarios such as task- and class-incremental learning using a common, likelihood-based updating scheme. This means that users can handle different types of CL problems using a single CL-BRUNO model in a statistically principled fashion, allowing knowledge to be accumulated more efficiently. Our proposed method uses *generative replay* (Shin et al., 2017; Wu et al., 2018) as a regularisation to prevent catastrophic forgetting without the need of retaining previously seen samples, making it more appealing in real-world applications where data privacy or storage capacity is of concern. Numerical experiments show that CL-BRUNO outperforms existing methods on both natural image and biomedical datasets, confirming its effectiveness. This paper is structured as follows: We first give technical background in Sec 2, then describe the proposed CL-BRUNO model in Sec 3. We highlight its connection with existing works in Sec 4, and report numerical experiments in Sec 5[1]. We conclude the paper with a brief discussion.

2 Background

In this section, we give the technical background relevant to our proposed method.

2.1 Normalising flow

A discrete Normalising flow (NF) (Rezende and Mohamed, 2015; Dinh et al., 2016; Papamakarios et al., 2017) models a continuous probability distribution by transforming a simple-structured base distribution (e.g. isotropic multivariate Normal) to the more complex target using a bijective transformation T parameterised as a composition of a series of smooth and invertible mappings $f_1, ..., f_K$ with easy-to-compute Jacobians. This T is applied to the "base" random variable $z_0 \sim p_0$, where $z_0 \in \mathbb{R}^D$ and p_0 is the known base density. Let $z_k = f_k \circ f_{k-1} \circ ... \circ f_1(z_0)$ for $k = 1, ..., K$. By applying change of variable repeatedly, the final output z_K has density $p_K(z_K) = p_0(z_0) \prod_{k=1}^{K} |\det J_k(z_{k-1})|^{-1}$, where J_k is the Jacobian of the mapping f_k. The final density p_K can be used to approximate target distributions with complex structure, and one can sample from p_K easily by applying $T = f_K \circ f_{K-1} \circ ... \circ f_1$ to $z_0 \sim p_0$. In order to evaluate p_K efficiently, we are restricted to transformations f_k whose $\det J_k(z)$ is easy to compute. For example, Real-NVP (Dinh et al., 2016) uses the following family of transformations: For $m \in \mathbb{N}$ such that $1 < m < d$, let $z_{1:m}$ be the first m entries of $z \in \mathbb{R}^D$, let $\times$ be element-wise multiplication and let $\mu_k, \sigma_k : \mathbb{R}^m \to \mathbb{R}^{D-m}$ be two neural nets. The smooth and invertible transformation $y = f_k(z)$ for each step k in a Real-NVP is defined as

$$y_{1:m} = z_{1:m}, \quad y_{m+1:d} = \mu_k(z_{1:m}) + \sigma_k(z_{1:m}) \times z_{m+1:d} \tag{1}$$

[1]Code reproducing the reported results can be found in https://anonymous.4open.science/r/clbruno-C0DE.

The Jacobian J_k of f_k is lower triangular, hence $\det J_k(z) = \prod_{i=1}^{D-m} \sigma_{ik}(z_{1:m})$, where $\sigma_{ik}(z_{1:m})$ is the ith entry of $\sigma_k(z_{1:m})$. Continuous normalising flows (Chen et al., 2018; Onken et al., 2021; Lipman et al., 2023) further extend model flexibility by replacing the series of transformations by a vector field continuously indexed by pseudo-time.

2.2 Conditional-BRUNO

Korshunova et al. (2020) proposed Conditional Bayesian Recurrent Neural model (C-BRUNO), a Neural Process (Garnelo et al., 2018; Kim et al., 2019; Xu et al., 2024) which models exchangeable sequences of high-dimensional observations conditionally on a set of labels. Given a set of feature-label pairs $\{X_i, h_i\}_{i=1}^N$ where $X_i \in \mathbb{R}^D$ is a feature vector and h_i its corresponding label, C-BRUNO assumes that the joint distribution $p(X_1, ..., X_N | h_1, ..., h_N) = p(X_{\pi(1)}, ..., X_{\pi(N)} | h_{\pi(1)}, ..., h_{\pi(N)})$ for any permutation π. It models $p(X_1, ..., X_N | h_1, ..., h_N)$ by first transforming each feature vector X_i into a latent variable $\mathbf{z}_i = f_\theta(X_i; h_i)$, where $f_\theta(X; h) : \mathbb{R}^D \to \mathbb{R}^D$ is a bijective conditional Real-NVP depending on label h. Denote $z_i^{(d)}$ as the dth entry of $\mathbf{z}_i$. It then models each dimension d of the exchangeable latent sequence $\{\mathbf{z}_i\}_{i=1}^N$ as an independent 1-D Gaussian process with $\mathrm{Var}(z_i^{(d)}) = \nu^{(d)}$ and $\mathrm{Cov}(z_i^{(d)}, z_j^{(d)}) = \rho^{(d)}$ for all $i, j = 1, ..., N$, where $0 < \rho^{(d)} < \nu^{(d)}$ are trainable covariance parameters. In other words, C-BRUNO assumes that $p(\mathbf{z}_1, \ldots, \mathbf{z}_N) = \prod_{d=1}^D p_d(\{z_n^{(d)}\}_{n=1}^N)$ where $p_d(\{z_n^{(d)}\}_{n=1}^N) = \mathcal{N}(\{z_n^{(d)}\}_{n=1}^N; \mathbf{0}_N, \Sigma_d)$, $\mathbf{0}_N$ is a zero vector of length N, and Σ_d is a $N \times N$ covariance matrix with diagonal element $\nu^{(d)}$ and off-diagonal element $\rho^{(d)}$. Suppose the bijective transformation f_θ and the covariance parameters $\{\rho^{(d)}, \nu^{(d)}\}_{d=1}^D$ have been chosen. C-BRUNO generates a new sample from the predictive distribution $p(X^* | h^*, X_{1:N}, h_{1:N})$ conditioned on new label h^* and previously seen feature-label pairs by first sampling $\mathbf{z}^* \sim p(\mathbf{z}^* | \mathbf{z}_{1:N})$ and then computing $X^* = f_\theta^{-1}(\mathbf{z}^*; h^*)$. The predictive likelihood $p(X_{N+1} | h_{N+1}, X_{1:N}, h_{1:N})$ of a new pair (X_{N+1}, h_{N+1}) can be evaluated in a similar fashion. Thanks to the specific covariance function used in C-BRUNO, sampling from and evaluating $p(\mathbf{z}^* | \mathbf{z}_{1:N})$ using the recursive formula given in Korshunova et al. (2020) has time complexity $\mathcal{O}(N)$ instead of the general case $\mathcal{O}(N^3)$. This greatly improves the scalability of C-BRUNO as it scales linearly with both sample size N and data dimension D.

The exchangeability assumption in C-BRUNO includes the conditionally identically and independently distributed (conditionally i.i.d.) assumption as a special case, which provides a more flexible modelling framework to reason about future samples based on the observed ones with minimal additional computational cost. This additional flexibility is appealing in many CL scenarios where new datasets arrive incrementally in batches. Modelling samples in each dataset as an exchangeable sequence (as supposed to i.i.d.) allows users to better capture and aggregate distributional information such as sample sizes and inter-sample correlations, which are useful for probabilistic prediction and inference, in a natural and statistically principled fashion.

3 Method

In this section, we give technical details of our proposed method. We start by specifying the notation. Suppose there is a sequence of tasks labelled by $t = 1, 2, \ldots, T$. Assuming all tasks are classification problems with potentially different or disjoint label spaces (extensions to regression problems are straightforward). We suppose each task $t = 1, 2, \ldots, T$ is associated with dataset $\mathcal{D}_t = \{\mathcal{X}_t, \mathcal{Y}_t\}$ where $\mathcal{X}_t = \{X_{i,t}\}_{i=1}^{N_t}$ is the feature set, $X_{i,t} \in \mathbb{R}^d$ is the feature vector of the i-th sample in the t-th task, $\mathcal{Y}_t = \{Y_{i,t}\}_{i=1}^{N_t}$ is the label set, $Y_{i,t} \in \{1, ..., C_t\}$ is the i-th sample's corresponding label, and N_t, C_t are the sample size of $\mathcal{D}_t$ and the number of distinct labels in the t-th task respectively. Similar to Korshunova et al. (2020), for each dataset $\mathcal{D}_t = \{\mathcal{X}_t, \mathcal{Y}_t\}$, we assume the corresponding density function $p_t(\mathcal{X}_t, \mathcal{Y}_t)$ factorises as

$$p_t(\mathcal{X}_t, \mathcal{Y}_t) = p(\mathcal{X}_t | t, \mathcal{Y}_t) p(\mathcal{Y}_t | t) = p(X_{1,t}, ..., X_{N_t,t} | t, Y_{1,t}, ..., Y_{N_t,t}) \prod_{i=1}^{N_t} p(Y_{i,t} | t) = \prod_{i=1}^{N_t} p(X_{i,t} | t, Y_{1:i,t}, X_{1:i-1,t}) p(Y_{i,t} | t),$$

and $\mathcal{X}_t$ is an exchangeable sequence of feature vectors conditioned on both task identity t and label set $\mathcal{Y}_t$. Note that $\mathcal{Y}_t$ are generated i.i.d. from some task-specific label prior $p(\cdot | t)$ under our distributional assumptions.

We propose Continual Learning BRUNO (CL-BRUNO) under the distributional assumptions given above. The key motivation of CL-BRUNO is to formulate continual learning problems as modelling streams of data (one for each task), which may come in batches, as a collection of exchangeable sequences. From this perspective, once N samples from a stream of data (i.e. a continual learning task) have been observed, users can then extract distributional information from historical observations by modelling the joint distribution $p(X_{1:N,t}, Y_{1:N,t})$, and predict a new sample and/or the associated label by inferring the one-step-ahead conditional distribution $p(X_{N+1,t}, Y_{N+1,t}|X_{1:N,t}, Y_{1:N,t})$ or $p(Y_{N+1,t}|X_{1:N+1,t}, Y_{1:N,t})$. Specifically, CL-BRUNO consists of two modules: a C-BRUNO model that models feature vectors $\mathcal{X}_t$ from different task t as exchangeable sequences, and an approximate Bayesian inference pipeline for label and task identity estimation. Let $\hat{p}(\mathcal{X}_t|t, \mathcal{Y}_t)$ be a C-BRUNO model that approximates $p(\mathcal{X}_t|t, \mathcal{Y}_t)$, the true conditional distribution of feature vectors $\mathcal{X}_t$ given $(t, \mathcal{Y}_t)$. CL-BRUNO aims to train a collection of C-BRUNO models $\{\hat{p}(\mathcal{X}_t|t, \mathcal{Y}_t)\}_{t=1,2,\dots}$ incrementally over an expanding range of $(t, \mathcal{Y}_t)$ by continually learning generative models $\hat{p}(\mathcal{X}_{t'}|t', \mathcal{Y}_{t'})$ from new datasets $\mathcal{D}_{t'}$, while preventing those it has learnt from previous tasks from being interfered or disrupted: If the incrementally trained C-BRUNO models could give $\hat{p}(\mathcal{X}_t|t, \mathcal{Y}_t)$ that well approximates $p(\mathcal{X}_t|t, \mathcal{Y}_t)$ for all previously seen tasks t throughout the continual learning process, then users could handle queries such as estimating the task identity associated with a new feature vector X^*, or computing $\hat{p}(Y^*|X^*, t, \mathcal{D}_t)$, the approximate posterior distribution of label Y^* associated with a new X^* from the t-th task based on historical data at any stage of the continual learning process in a statistically principled fashion.

3.1 Incremental learning scheme

In this section, we discuss the parametrisation and incremental training procedure of CL-BRUNO. Thanks to the i.i.d. assumption on labels $\{\mathcal{Y}_t\}_{t=1}^T$ in equation 2, the task-specific label priors $p(Y|t)$ can be estimated directly using their population proportions. For the remainder of the section, we focus on learning the conditional distribution of $\mathcal{X}_t$. Specifically, the C-BRUNO model used in our proposed method consists of a Conditional Real-NVP[2] $f_\theta(\cdot; t, Y_t) : \mathbb{R}^D \to \mathbb{R}^D$ parameterised by θ as a bijective transformation that depends on both task identity t and label Y_t. We parameterise each task identity t and each of the unique labels $\{1, \dots, C_t\}$ in the t-th task as trainable embeddings $r_t, s_{j,t} \in \mathbb{R}^l$ respectively for $j = 1, \dots, C_t$ and $t = 1, 2, \dots$. For the rest of the paper, the embeddings $\{r_t, \{s_{j,t}\}_{j=1}^{C_t}\}_{t=1}^T$ are viewed as parts of the Conditional Real-NVP parameter θ. For $t = 1, 2, \dots$, let $\mathbf{z}_{i,t} = f_\theta(X_{i,t}; t, Y_{i,t})$ for all $i = 1, \dots, N_t$, $\mathbf{z}_t = \{\mathbf{z}_{i,t}\}_{i=1}^{N_t}$, and $p_{\lambda_t}(\mathbf{z}_t)$ be the task-specific latent distribution parametrised by $\lambda_t = \{\nu_t^{(d)}, \rho_t^{(d)}\}_{d=1}^D$ as described in Sec 2.2. Denote $\boldsymbol{\lambda} = \{\lambda_t\}_{t=1,2,\dots}$ the set of latent distribution parameters.

Learning from scratch

If there is no historical information or previously seen data, then we can extract and retain information in these datasets by training a series of C-BRUNO models that approximate $p(\mathcal{X}_t|t, \mathcal{Y}_t)$ for all $t = 1, \dots, T$ by minimising the negative log likelihood

$$
\begin{aligned}
L(\theta, \boldsymbol{\lambda}; \{\mathcal{D}_t\}_{t=1}^T) &= -\sum_{t=1}^{T} \overbrace{\log p_{\theta,\lambda_t}(\mathcal{X}_t|t, \mathcal{Y}_t)}^{\text{task-wise log likelihood}} \\
&= -\sum_{t=1}^{T}\sum_{i=1}^{N_t} \log p_{\theta,\lambda_t}(X_{i,t}|Y_{i,t}, t, X_{1:i-1,t}, Y_{1:i-1,t}) \\
&= -\sum_{t,i=1}^{T,N_t} \log\left(p_{\lambda_t}(\mathbf{z}_{i,t}|\mathbf{z}_{1:i-1,t}) \left|\det \frac{\partial f_\theta(X_{i,t}; t, Y_{i,t})}{\partial X_{i,t}}\right|\right),
\end{aligned} \tag{2}
$$

as suggested by Korshunova et al. (2020) where $p_{\theta,\lambda_t}(\mathcal{X}_t|t, \mathcal{Y}_t)$ denotes the C-BRUNO approximation to $p(\mathcal{X}_t|t, \mathcal{Y}_t)$. Now, suppose $(\hat{\theta}, \hat{\boldsymbol{\lambda}}) = \arg\min_{\theta,\boldsymbol{\lambda}} L(\theta, \boldsymbol{\lambda}; \{\mathcal{D}_t\}_{t=1}^T)$, and $\hat{\mathbf{z}}_{i,t} = f_{\hat{\theta}}(X_{i,t}; t, Y_{i,t})$ for all $i = 1, \dots, N_t$ and $t = 1, \dots, T$. Then for any task $t = 1, \dots, T$, $p(X^*|Y^*, t, \mathcal{D}_t)$, the predictive distribution of new

[2]It is straightforward to replace it with more sophisticated alternatives (Chen et al., 2018; Lipman et al., 2023; Shi et al., 2024). We do not consider them here for sake of conceptual simplicity.

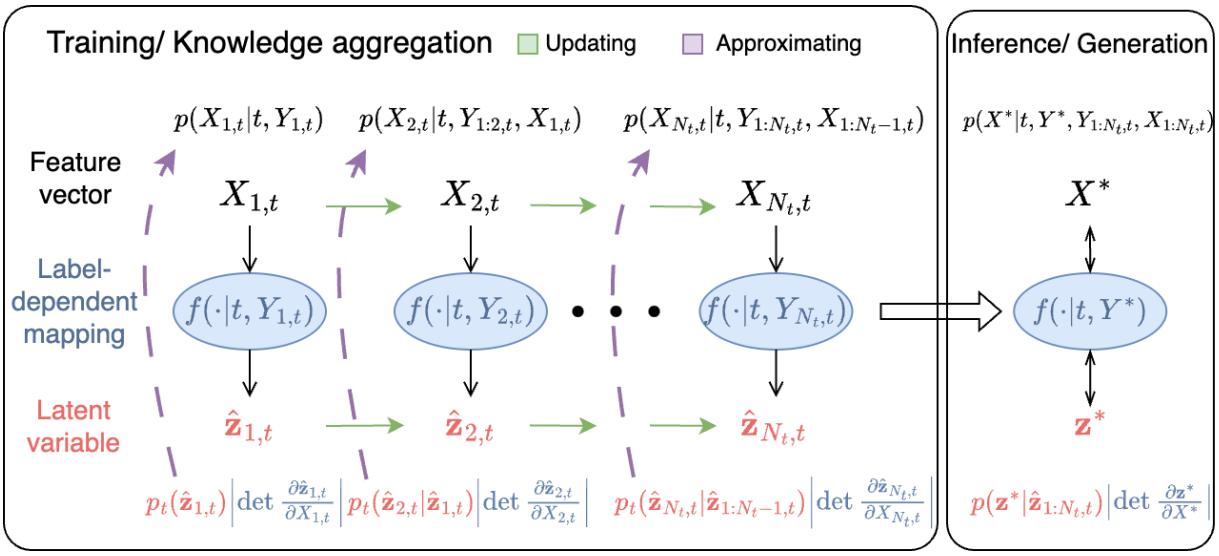


Figure 1: **Schematic illustration of how C-BRUNO learns the sequence distribution** $p(\mathcal{X}_t|\mathcal{Y}_t) = \prod_{i=1}^{N}(X_{i,t}|X_{1:i-1,t}, Y_{1:i,t})$**.** For each feature vector $X_{i,t}$ in the sequence, C-BRUNO first transform it to the corresponding latent variable using the label-dependent mapping $\hat{\mathbf{z}}_{i,t} = f(X_{i,t}|t, Y_{i,t})$, then approximates the one-step-ahead conditional $p(X_{i,t}|X_{1:i-1,t}, Y_{1:i,t})$ by $p_t(\hat{\mathbf{z}}_{i,t}|\hat{\mathbf{z}}_{1:i-1}) \left|\det \frac{\partial \hat{\mathbf{z}}_{i,t}}{\partial X_{i,t}}\right|$. Exchangeability is guaranteed by the specific covariance function in the latent distribution $p(\hat{\mathbf{z}}_{1:N_t})$. In the generation/inference phase, given a label Y^*, a new latent variable $\mathbf{z}^*$ is first generated from $p(\cdot|\hat{\mathbf{z}}_{1:N,t})$, a multivariate Gaussian whose mean and covariance depend on the observed sequence, and then transformed to the generated feature vector $X^* = f^{-1}(\mathbf{z}^*|t, Y^*)$ under label Y^*.

observation X^* given label Y^* and historical data $\mathcal{D}_t$, can be summarised by a generative C-BRUNO model with bijective mapping $f_{\hat{\theta}}$ and multivariate Gaussian predictive distribution $p_{\hat{\lambda}_t}(\cdot|\hat{\mathbf{z}}_{1:N_t,t})$. See also Fig 1 for a schematic illustration.

We stress that even though $p_{\hat{\lambda}_t^{(T)}}(\cdot|\hat{\mathbf{z}}_{1:N_t,t})$ is a multivariate Gaussian distribution whose mean and covariance depend on $\hat{\mathbf{z}}_{1:N_t,t}$, evaluating or sampling from it does not require access to $\hat{\mathbf{z}}_{1:N_t,t}$, see also Eqn (7) in Korshunova et al. (2020). As a result, once the generative model has been trained, users can use it as a proxy of $\{\mathcal{D}_t\}_{t=1}^{T}$, and handle queries without the need of revisiting any historical data. This learning strategy allows users to retain knowledge and information from different tasks without storing any sample from any dataset. However, under a CL setup, such a learning scheme via joint likelihood maximisation is not feasible as datasets $\mathcal{D}_t$s are observed in a sequential fashion, and we do not necessarily have access to any previously seen datasets. On the other hand, sequentially maximising $L(\theta, \boldsymbol{\lambda}; \mathcal{D}_t), t = 1, 2, \ldots$ naively whenever a new dataset $\mathcal{D}_t$ arrives will drastically disrupt what it has learnt from historical data due to *catastrophic forgetting* (McCloskey and Cohen, 1989), rendering it unusable for any prediction task except for the most recent one. To address this issue, our proposed CL-BRUNO uses *generative replay* to prevent catastrophic forgetting through both distributional and functional regularisation. We demonstrate our incremental learning strategy under two common CL scenarios known as Task- and Class-Incremental learning. Other scenarios such as Domain- or Instance- Incremental learning (Wang et al., 2024) can be handled in a similar fashion.

Task-incremental learning

Suppose our CL model has been trained on T tasks. Let $f_{\hat{\theta}^{(T)}}$, $\{p_{\hat{\lambda}_t^{(T)}}(\cdot|\hat{\mathbf{z}}_{1:N_t,t})\}_{t=1}^{T}$ be the trained normalising flow and predictive latent distributions respectively. Suppose now a new task $T+1$ and the associated dataset $\mathcal{D}_{T+1}$ arrive. Our goal is to update the generative model so that it adapts to $\mathcal{D}_{T+1}$ while retaining the historical knowledge the old generative model has learnt regarding the previous T tasks. This scenario is known as *Task-Incremental learning* (TIL) (Van de Ven and Tolias, 2019).

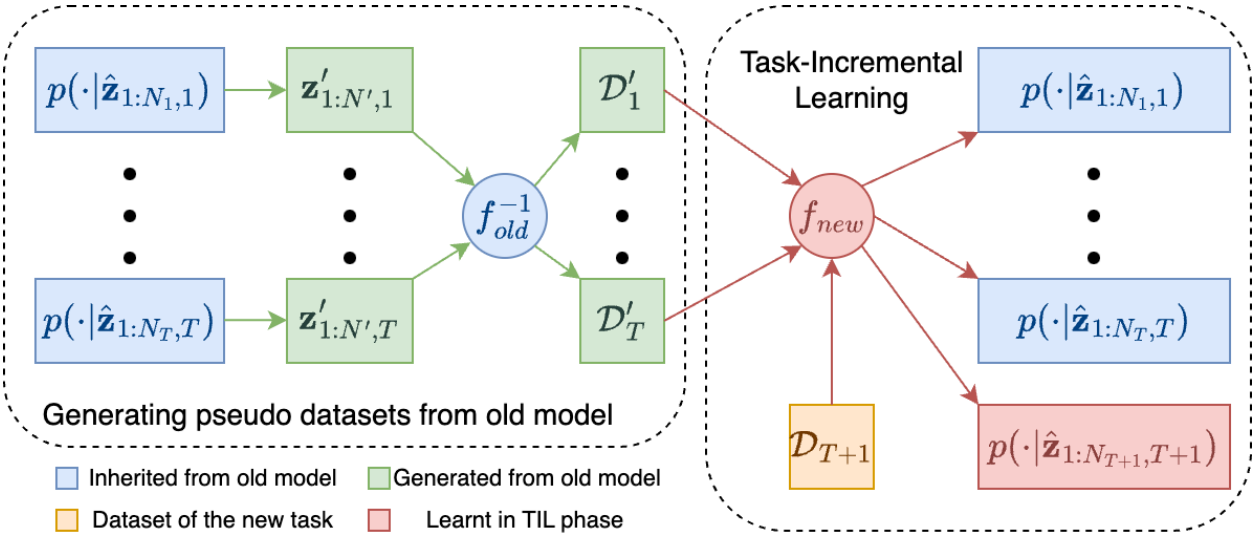


Figure 2: **Schematic illustration of TIL in CL-BRUNO.** Pseudo datasets are generated from the previous latent predictive distributions $p(\cdot|\hat{\mathbf{z}}_{1:N_t,t})$ and the bijective mapping f_{old}. Note that in the TIL phase, the new bijective mapping f_{new} learns to 1) map the new dataset $\mathcal{D}_{T+1}$ to a series of latent variables and compute the corresponding latent predictive $p(\cdot|\hat{\mathbf{z}}_{1:N_{T+1},T+1})$ (i.e. learning from new data) and 2) map the pseudo-datasets $\hat{\mathcal{D}}_t$ back to latent variables that resemble samples drawn from the previous latent predictive distributions $p(\cdot|\hat{\mathbf{z}}_{1:N_t,t})$ (i.e. retaining learnt knowledge).

To achieve this goal, we update the old CL-BRUNO by estimating 1) a new bijective transformation f_θ parametrised by θ, and 2) λ_{T+1} for the latent distribution of the $T+1$-th task. The rest of the old CL-BRUNO model (i.e. latent distributions associated with previous tasks) are kept unchanged. Specifically, we first generate pseudo datasets $\mathcal{D}'_t = \{X'_{j,t}, Y'_{j,t}\}_{j=1}^{N'}$ for each previously seen task $t = 1, 2, \ldots, T$ from the old CL-BRUNO model as follows. Denote N' the size of pseudo datasets, for each $\mathcal{D}'_t$ and $j = 1, \ldots, N'$, the pseudo label $Y'_{j,t}$ associated with the t-th task is sampled according to the population label proportion of $\mathcal{D}_t$, then the pseudo feature vector $X'_{j,t}$ is generated by first drawing latent variable $\mathbf{z}'_{j,t} \sim p_{\hat{\lambda}_t}(\cdot|\hat{\mathbf{z}}_{1:N_t,t})$, then set $X'_{j,t} = f_{\hat{\theta}^{(T)}}(\mathbf{z}'_{j,t}; t, Y'_{j,t})$. Once the T pseudo datasets based on the old CL-BRUNO has been generated, one can then update the CL-BRUNO by updating parameters $\{\theta, \lambda_{T+1}\}$ using the joint likelihood optimisation approach in Eq equation 2 based on the augmented dataset $\{\mathcal{D}'_t\}_{t=1}^T \cup \mathcal{D}_{T+1}$. See Fig 2 for a schematic illustration. To further prevent the historical knowledge in $f_{\hat{\theta}^{(T)}}$ from being disrupted by the new dataset $\mathcal{D}_{T+1}$, we adopt the functional regularisation technique in Wu et al. (2018) that penalises the L_2 distance between outputs of the old and new models generated from the same set of input noises. Specifically, for a new bijective f_θ parametrised by θ, the regularisation takes the form

$$R(\theta; \hat{\theta}^{(T)}, \{\mathcal{D}'_t\}_{t=1}^T) = \sum_{t,i=1}^{T,N'} ||f_\theta^{-1}(\mathbf{z}'_{i,t}; t, Y'_{i,t}) - X'_{i,t}||_2^2, \tag{3}$$

where $f_\theta^{-1}(\mathbf{z}'_{i,t}; t, Y'_{i,t})$ and $X'_{i,t}$ are the outputs of the new and old normalising flows based on the common noise $\mathbf{z}'_{i,t}$. Combining the augmented datasets and the functional regularisation, the new generative model f_θ and the base distribution parameter λ_{T+1} of the new task $T+1$ under the TIL scenario is chosen by solving $\min_{\theta,\lambda_{T+1}} L_{\text{TIL}}(\theta, \lambda_{T+1})$ where $L_{\text{TIL}}(\theta, \lambda_{T+1})$ is the regularised joint negative log likelihood

$$L_{\text{TIL}}(\theta, \lambda_{T+1}) = \overbrace{L(\theta, \lambda_{T+1}; \mathcal{D}_{T+1})}^{\text{neg log likelihood of new task}} + \underbrace{\alpha_1 L'(\theta, \{\hat{\lambda}_t^{(T)}\}_{t=1}^T; \{\mathcal{D}'_t\}_{t=1}^T)}_{\text{distributional regulariser}} + \underbrace{\alpha_2 R(\theta; \hat{\theta}^{(T)}, \{\mathcal{D}'_t\}_{t=1}^T)}_{\text{functional regulariser}},$$

where

$$L'(\theta, \{\hat{\lambda}_t^{(T)}\}_{t=1}^T; \{\mathcal{D}'_t\}_{t=1}^T) = -\sum_{t,j=1}^{T,N'} \log\left[p_{\hat{\lambda}_t^{(T)}}(f_\theta(X'_{j,t}; t, Y'_{j,t})|\hat{\mathbf{z}}_{1:N_t,t})\right] - \sum_{t,j=1}^{T,N'} \log\left[\left|\det \frac{\partial f_\theta(X'_{j,t}; t, Y'_{j,t})}{\partial X'_{i,t}}\right|\right],$$

is the negative log likelihood associated with the T pseudo datasets and $\alpha_1, \alpha_2 > 0$ controls the strength of regularisation. Note that in comparison with Eqn equation 2, the latent distributions in L' are informed by historical samples $\{\hat{\mathbf{z}}_{1:N_t,t}\}$ as we expect f_θ to be able to map $X'_{j,t}$ back to latent variables that resemble samples drawn from $p(\cdot|\hat{\mathbf{z}}_{1:N_t,t})$. See also Fig 2.

Class-incremental learning

So far we focused on task-incremental learning where all samples associated with a new task are presented to the model as a single dataset $\mathcal{D}_{T+1}$. It is not always the case in practice as samples associated with the same task may also come in batches. In particular, each batch may contain labels not seen in any previous batches. This scenario is known as *Class-Incremental learning* (CIL) (Zhou et al., 2024). We here demonstrate how CL-BRUNO handles CIL under the scenario where labels in different data batches are disjoint (the extension to the non-disjoint case is straightforward).

Our goal now is to learn from a new batch of data $\mathcal{D}_k^{(1)} = \{X_{i,k}^{(1)}, Y_{i,k}^{(1)}\}_{i=1}^{N_k^{(1)}}$ associated with a *previously seen and known* task $k \in \{1, .., T\}$ such that $Y_{i,k}^{(1)} \neq Y_{j,k}$ for all $Y_{i,k}^{(1)}$ in $\mathcal{D}_k^{(1)}$ and $Y_{j,k}$ in $\mathcal{D}_k$ (i.e. disjoint labels). Such problem can be addressed by updating the normalising flow f_θ using the same regularisation approach as in Sec 3.1. Similar to L_{TIL} in Eqn equation 4, CL-BRUNO under the CIL scenario is updated by finding a f_θ that minimizes a regularised augmented negative log likelihood

$$L_{\text{CIL}}(\theta) = \overbrace{-\log p_{\theta,\hat{\lambda}_k^{(T)}}(\mathcal{X}_k^{(1)}|k, \mathcal{Y}_k^{(1)}, \mathcal{D}_k)}^{\text{neg log lkd of new batch given old}} + \alpha_1 L'(\theta, \{\hat{\lambda}_t^{(T)}\}_{t=1}^T; \{\mathcal{D}'_t\}_{t=1}^T) + \alpha_2 R(\theta; \hat{\theta}^{(T)}, \{\mathcal{D}'_t\}_{t=1}^T),$$

where

$$\log p_{\theta,\hat{\lambda}_k^{(T)}}(\mathcal{X}_k^{(1)}|k, \mathcal{Y}_k^{(1)}, \mathcal{D}_k) = \sum_{i=1}^{N_k^{(1)}} \log\left(p_{\hat{\lambda}_k^{(T)}}(\mathbf{z}_{i,k}^{(1)}|\mathbf{z}_{1:i-1,k}^{(1)}, \hat{\mathbf{z}}_{1:N_k,k}) \times \left|\det \frac{\partial f_\theta(X_{i,k}^{(1)}; k, Y_{i,k}^{(1)})}{\partial X_{i,k}^{(1)}}\right|\right), \tag{4}$$

is the conditional likelihood of the new feature set $\mathcal{X}_k^{(1)} = \{X_{i,k}^{(1)}\}_{i=1}^{N_k^{(1)}}$ given the corresponding label set $\mathcal{Y}_k^{(1)} = \{Y_{i,k}^{(1)}\}_{i=1}^{N_k^{(1)}}$ and previous data batch $\mathcal{D}_k$ associated with the same task, and $\mathbf{z}_{i,k}^{(1)} = f_\theta(X_{i,k}^{(1)}; k, Y_{i,k}^{(1)})$ is the transformed latent variable generated by the new normalising flow f_θ. Note that given the predictive latent distribution $p_{\hat{\lambda}_k^{(T)}}(\cdot|\hat{\mathbf{z}}_{1:N_k,k})$ from the old CL-BRUNO, $p_{\hat{\lambda}_k^{(T)}}(\mathbf{z}_{i,k}^{(1)}|\mathbf{z}_{1:i-1,k}^{(1)}, \hat{\mathbf{z}}_{1:N_k,k})$ in equation 8 can be evaluated efficiently using the recursive formula in Korshunova et al. (2020) for all $i = 1, \ldots, N_k^{(1)}$ without access to $\hat{\mathbf{z}}_{1:N_k,k}$. In comparison with Eqn equation 4, the parameter of latent distribution $\hat{\lambda}_k^{(T)}$ associated with task k in L_{CIL} is taken from the old CL-BRUNO and *not updated* alongside with θ. We choose to do so as it simplifies the computation and ensures the tractability of $p_{\hat{\lambda}_k^{(T)}}(\cdot|\hat{\mathbf{z}}_{1:N_k,k})$. This choice can also be viewed as an additional regularisation that aims to mitigate the impact of catastrophic forgetting. See Supplementary material A.1 for additional discussion on L_{CIL}.

3.2 Label and task identity prediction

In the previous section, we discussed a unified incremental learning strategy for both task- and class-incremental learning. In particular, we focused on incrementally modelling the feature vectors $p(\mathcal{X}_t|t, \mathcal{Y}_t)$ for tasks $t = 1, \ldots, T$. In this section, we discuss the probabilistic inference pipeline for label and task identity prediction. Let $f_{\hat{\theta}^{(T)}}$, $\{p_{\hat{\lambda}_t^{(T)}}(\cdot|\hat{\mathbf{z}}_{1:N_t,t})\}_{t=1}^T$ be the normalising flow and predictive latent distributions trained over T tasks. Let X^* be a generic test point and Y^* be the unknown label of interest. We start from the case

where the task identity t is known. Under the assumption given in equation 2, we approximate the posterior label distribution $p(Y^*|t, X^*, \mathcal{D}_t)$ by

$$\hat{p}(Y^*|t, X^*, \mathcal{D}_t) \propto p_{\hat{\theta}^{(T)}, \hat{\lambda}_t}(X^*|t, Y^*, \mathcal{D}_t)p(Y^*|t) \propto p_{\hat{\lambda}_t}(\mathbf{z}^*_{Y^*,t}|\hat{\mathbf{z}}_{1:N_t,t}) \left|\det \frac{\partial f_{\hat{\theta}^{(T)}}(X^*; t, Y^*)}{\partial X^*}\right| p(Y^*|t),$$

for all $Y^* \in \{1, \ldots, C_t\}$, where $\mathbf{z}^*_{Y^*,t} = f_{\hat{\theta}^{(T)}}(X^*; t, Y^*)$ is the transformed latent variable associated with X^* conditioned on label Y^*, and $p(Y^*|t)$ is the prior distribution over the C_t classes associated with the t-th task.

However, the task identity associated with X^* in practice is not always available, and task identity estimation is itself a challenging problem in CL (Lee et al., 2020). Here we give an easy-to-interpret probabilistic estimate of task identity of a test point X^* under the assumption that X^* is indeed a sample from the underlying generative process of one of the T previously seen datasets $\{\mathcal{D}_t\}_{t=1}^T$ (See Supplementary material A.3 for discussion on other possibilities). Denote $p(t)$ a user specified prior over the T tasks, and $\mathcal{C}_t = \{1, \ldots, C_t\}$ the label set associated with the t-th task. We approximate the task identity distribution $p(X^* \text{ from task } t|\{\mathcal{D}_t\}_{t=1}^T)$ for $t = 1, \ldots, T$ by

$$\hat{p}(X^* \text{ from task } t|\{\mathcal{D}_t\}_{t=1}^T) \propto p(t) \sum_{Y^* \in \mathcal{C}_t} p_{\hat{\theta}^{(T)}, \hat{\lambda}_t}(X^*|t, Y^*, \mathcal{D}_t)p(Y^*|t), \tag{5}$$

which can be interpreted as the predictive likelihood of observing X^* as the next new sample given previous ones $\mathcal{D}_t$ averaged over all possible labels Y^*. In addition, if all tasks share the same label space, we can then further marginalise out the uncertainty of task identity in label prediction by

$$\hat{p}(Y^*|X^*, \{\mathcal{D}_t\}_{t=1}^T) = \sum_{t=1}^{T} \hat{p}(Y^*|t, X^*, \mathcal{D}_t) \times \hat{p}(X^* \text{ from task } t|\{\mathcal{D}_t\}_{t=1}^T). \tag{6}$$

This is particularly useful in biomedical settings where a single CL model incrementally learns to distinguish healthy vs unhealthy from a sequence of datasets collected from patients from different hospitals or regions (i.e. different tasks).

4 Related works

4.1 Neural Process continual learning

Jha et al. (2024) propose Neural Process Continual Learning (NPCL), which combines attentive Neural Processes (Kim et al., 2019) and experience replay (Chaudhry et al., 2019). NPCL uses Neural Processes to model the posterior distributions of labels $p(Y^*|t, X^*, \mathcal{D}_{prev})$ as a random function of task identity t, test point X^* and historical data $\mathcal{D}_{prev}$. In contrast, our approach uses Neural Processes to model the distribution of feature vectors $p(\mathcal{X}_t|t, \mathcal{Y}_t)$ associated with different tasks and labels. Unlike NPCL, our method leverages generative replay to retain learned knowledge without the need for storing any previous samples, making it more appealing to applications where data privacy or storage is of concern. In addition, thanks to the specific covariance function used in C-BRUNO, our proposed method scales linearly with both the sample size and the dimension of feature vector. In contrast, NPCL exhibits quadratic complexity relative to the training sample size due to the attention architecture.

4.2 Conditional generative continual learning

The generative replay strategy used in our proposed method is closely related to Continual Learning for Conditional Generation (CLCG) (Wang et al., 2024), which also aims to mitigate catastrophic forgetting by recovering previously-learned data distributions. Recent CLCG methods such as MeRGANs Wu et al. (2018), Boo-VAE (Egorov et al., 2021), Hyper-LifelongGAN (Zhai et al., 2021) and FILIT (Chen et al., 2022) use GAN (Goodfellow et al., 2020) or VAE (Kingma and Welling, 2013) as their underlying conditional generative models and require separate discriminative models for label prediction. Scardapane et al. (2020)

Type	Method	S-CIFAR100	MNIST
(a)	NPCL ($M = 500$)	0.198	-
(b)	EWC	0.092	0.441
	LwF	0.145	0.472
	SSRE	0.179	-
(c)	MeRGAN	x	0.670
	FeTriL	0.187	-
	Boo-VAE	-	0.892
	CL-BRUNO	**0.212**	**0.947**

Table 1: Classification accuracy of different CL methods on class-incremental S-CIFAR-100 dataset and task-incremental MNIST dataset. - indicates method is not applicable. x indicates method does not converge in reasonable time. (a) Experience replay, (b) Non-generative, exemplar-free and (c) Generative, exemplar-free. Best results are in **boldface**.

uses normalizing flows to retain distributional information at the feature embedding level, but still requires separate encoders and discriminators. In comparison, CL-BRUNO utilises a tractable deep generative model that supports both conditional generation and density estimation on feature vectors. This enables tractable, efficient, and statistically principled label prediction and task identity estimation without the need for separate discriminators or any other modules, improving both interpretability and computational efficiency.

5 Experiments

In this section, we demonstrate the efficacy and versatility of CL-BRUNO using two real-world biomedical datasets. We also compare CL-BRUNO with existing methods on two commonly used public benchmark image datasets, CIFAR-100 (Krizhevsky et al., 2009) and MNIST (LeCun, 1998).

5.1 Pan-Cancer Atlas dataset

We here demonstrate CL-BRUNO under a CIL scenario using the Pan-Cancer Atlas (PANCAN) dataset (Hoadley et al., 2018). The PANCAN dataset consists of pre-processed RNAseq readings of $N = 10,535$ tumour samples from 33 cancer types. In our analysis, we use only the top $P = 2,000$ most variable genes as the feature vector associated with each tumour sample, and split the PANCAN dataset into 6 groups according to their cancer types: the first consists of 8 cancer types, while each of the rest consists of 5 cancer types. Cancer types are partitioned in a way such that all 6 groups of data have similar sample sizes. Our goal is to predict cancer type from the RNAseq data of a tumour under a CIL scenario, where 6 groups of data are presented to the model sequentially. In this example, we set the number of coupling layers in CL-BRUNO to be 6, the dimension of task and label embedding to be 256, size of pseudo data $N' = 128$ and regularisation strength $\alpha_1 = \alpha_2 = 1$. Additionally, we resample the pseudo-data for every gradient descent step. Each group of data is randomly split into a training set consisting of 80% of the samples and a test set containing the rest, and the performance is measured by the misclassification rate on all test sets after the model has been incrementally trained on all training sets. We compare CL-BRUNO with three exemplar-free CL methods: EWC (Kirkpatrick et al., 2017), LwF (Li and Hoiem, 2017) and MeR-GAN (Wu et al., 2018) under default or recommended settings. The misclassification rates on test sets are reported in Table 2. We also demonstrate in Fig 3(a) how CL-BRUNO retains previously learnt knowledge by reporting the misclassification rate associated with each of the 6 test sets at each incremental learning step, and compare them with results from an oracle CL-BRUNO model that has access to all historical data (i.e. trained by directly minimising Eqn 2) at each incremental learning step. Comparing with the oracle model, we see no abrupt interference or deterioration in prediction accuracy in the training steps.

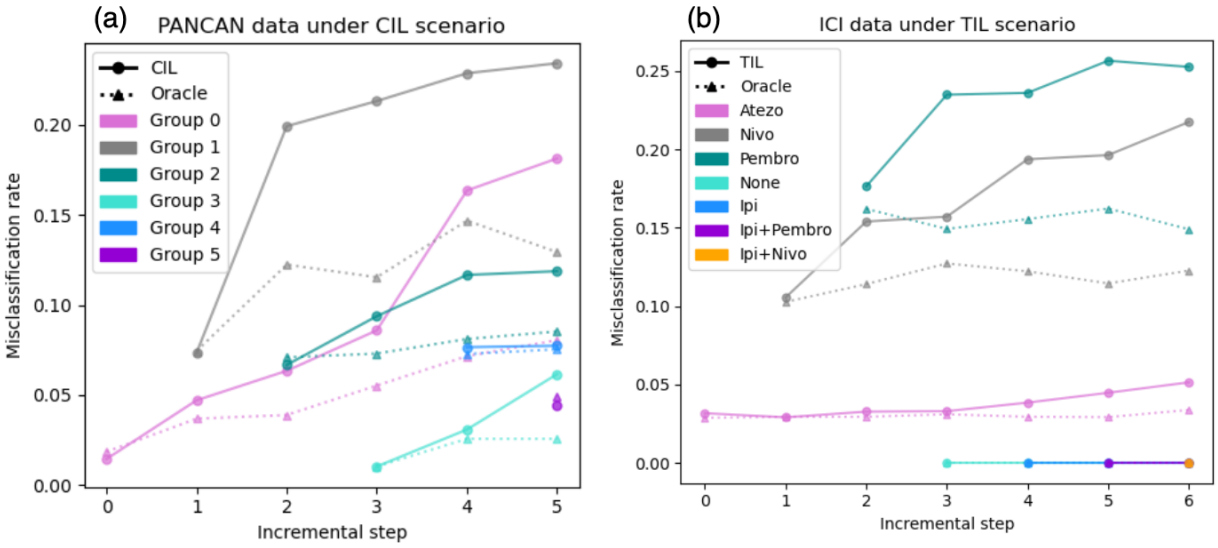


Figure 3: **Evolution of misclassification rate specific to each incremental datasets.** Each point represents the misclassification rate specific to an incremental dataset evaluated at a specific training step using the incrementally trained CL-BRUNO. Each triangle represents the same quantity given by a CL-BRUNO who has access to all historical datasets (oracle). (a) PANCAN dataset under a CIL scenario, (b) ICI dataset under a TIL scenario. Note that ICI dataset consists of tasks with only one class, which leads to zero test error.

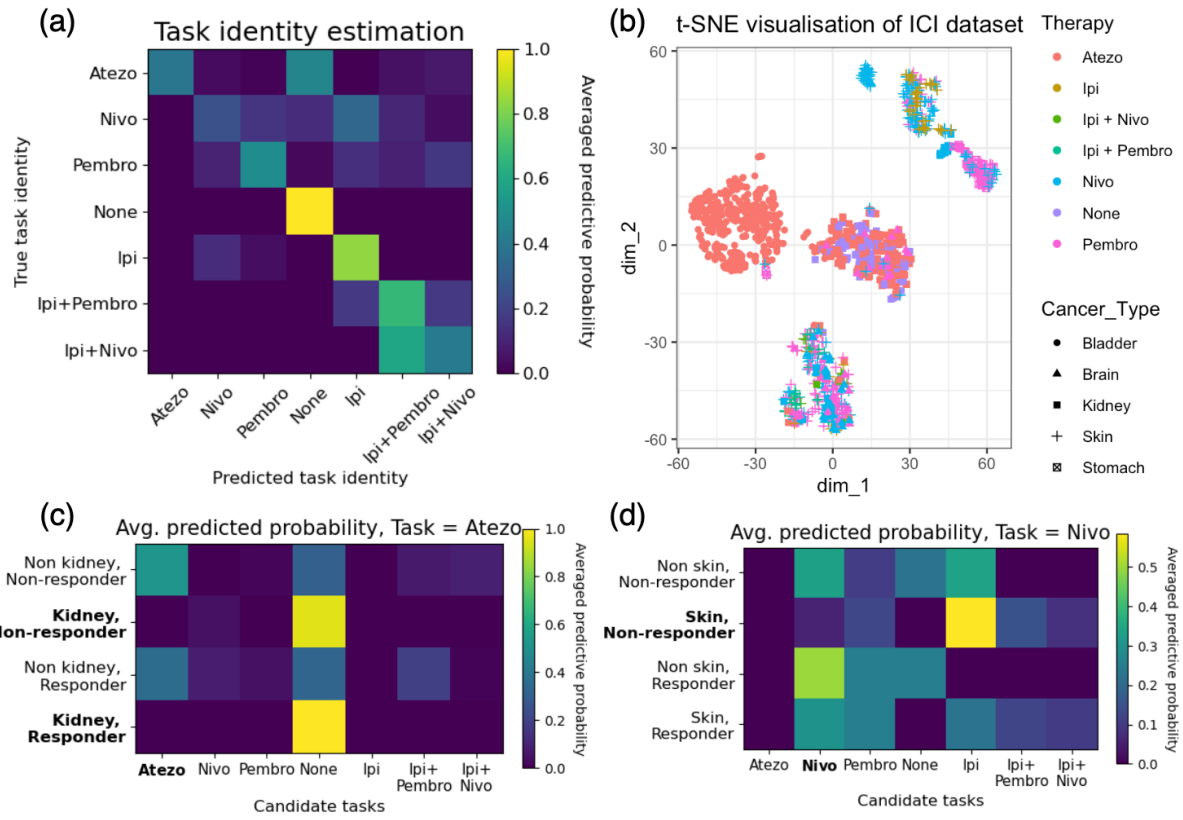
Method/Data	PANCAN	ICI
CL-BRUNO	**0.139(0.0233)**	**0.118(0.0351)**
EWC	0.569(0.0265)	0.331(0.0392)
LwF	0.322(0.0217)	0.157 (0.0433)
MeR-GAN	0.518(0.0521)	-

Table 2: **Misclassification rate of different methods on PANCAN and ICI datasets**. Results are averaged over 5 repeated runs. Standard deviations of the misclassification rates are reported in brackets. Best results are in **boldface**.

5.2 Immune Checkpoint Inhibitors dataset

We also tested CL-BRUNO under a TIL scenario using the Molecular Response to Immune Checkpoint Inhibitors (ICI) dataset (Eddy et al., 2020). The ICI dataset contains pre-processed RNAseq data from $N = 1,142$ patients' tumour samples under 7 different types of immunotherapy treatments. This included four single therapy only regimes: Atezolizumab (`Atezo`), Nivolumab (`Nivo`), Pembrolizumab (`Pembro`), Ipilimumab (`Ipi`), two combination therapies (`Ipi+Pembro`, `Ipi+Nivo`) and a non-treatment/placebo group (`None`). Our goal is again to predict cancer type given the RNAseq data. In this example, we split the dataset into 7 groups according to the therapy the patients received, and treat the classification problem under each therapy as an individual task. The 7 tasks consists of $N_1 = 524, N_2 = 224, N_3 = 218, N_4 = 89, N_5 = 42, N_6 = 32, N_7 = 13$ samples and $C_1 = 2$ ({Bladder, Kidney}), $C_2 = 3$ ({Brain, Skin, Kidney}), $C_3 = 3$ ({Stomach, Brain, Skin}), $C_4 = 1$ ({Kidney}), $C_5 = 1$ ({Skin}), $C_6 = 1$ ({Skin}), $C_7 = 2$ ({Kidney, Skin}) unique cancer types respectively. We use the same set of hyperparameters and training strategy as in the last example to train the CL-BRUNO, and compare its performance with EWC and LwF under default or recommended settings.[3] This example is challenging due to the relatively small and imbalanced sample sizes. We report the misclassification rates given by different methods in Table 2. Knowledge retention curves reported in Fig 3 (b) show no drastic performance deterioration in any task throughout the incremental training steps. We

[3]We did not include MeR-GAN here as it is designed for CIL.



olecular

Response to Immune Checkpoint Inhibitors

Figure 4: **ICI dataset**. (a): Heat map of predicted task identity. Each row corresponds to the categorical task identity distribution equation 5 averaged over test samples from each task. (b): t-SNE (Van der Maaten and Hinton, 2008) projection of the pre-processed RNAseq measurement associated with different therapy types in ICI dataset. (c): Averaged predicted probabilities for different groups of patients under treatment `Atezo`. Patients are split into four groups based on cancer type (Kidney cancer vs Non-kidney cancer) and responsiveness to treatment (Responder vs Non-responder). (d): Averaged predicted probabilities for groups of patients under treatment `Nivo`. Patients are split into four groups in a similar fashion to (c).

report the task identity estimates given by the trained CL-BRUNO model under a uniform task prior in Fig 4. We see the task identity estimates put most of the probability mass on the correct task identities (the diagonal line), indicating good prediction accuracy.

5.2.1 CL-BRUNO captures inter-task relationships in ICI dataset

In this section we demonstrate that CL-BRUNO is capable of capturing inter-task relationships governed by the underlying biological process: Pal et al. (2022) report that `Atezo` is ineffective to kidney cancers. Hence we expect samples from `Atezo` with kidney cancer to be indistinguishable from samples in `None` as both are equivalent to no treatment. To verify if CL-BRUNO captures this relationship, we first split the test set of `Atezo` into 4 subgroups depending on their cancer type (Kidney vs Non-kidney) and response to the therapy (Responder vs Non-responder), then compute the averaged predicted probabilities separately. From Fig 5 (c) we see CL-BRUNO is much more likely to assign kidney cancer samples in `Atezo` to `None` in comparison with non-kidney cancer samples regardless of the response status. This agrees with previous studies, and is also confirmed by visualisation, as we see from Fig 5 (a) that samples from `None` overlap with a cluster of kidney cancer samples in `Atezo`.

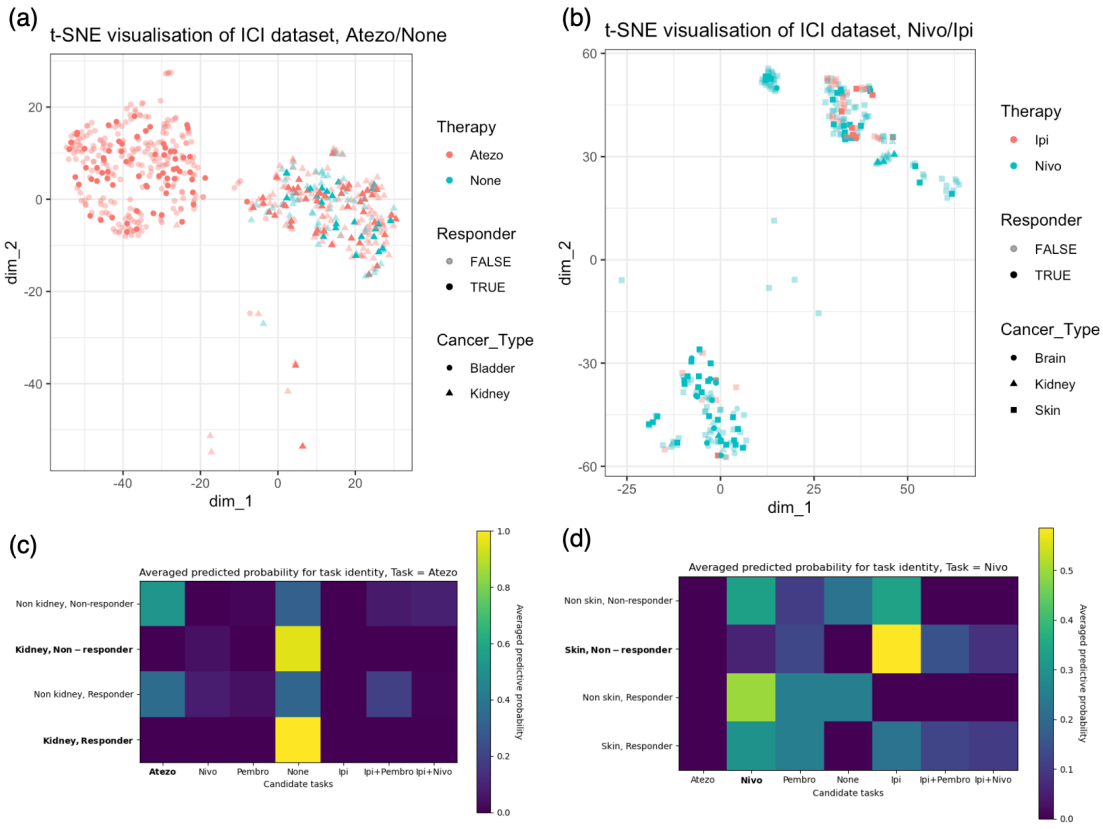


Figure 5: **Visualisation of subsets of ICI dataset** (a): t-SNE projection of samples from `Atezo` and `None`. (b): t-SNE projection of samples from `Nivo` and `Ipi`. (c): Averaged predicted probabilities for different groups of patients under treatment `Atezo`. Patients are split into four groups based on cancer type (Kidney cancer vs Non-kidney cancer) and responsiveness to treatment (Responder vs Non-responder). (d): Averaged predicted probabilities for groups of patients under treatment `Nivo`. Patients are split into four groups in a similar fashion to (c).

By the same rationale, since both `Nivo` and `Ipi` are given to patients with skin cancer, we expect non-responders in `Nivo` with skin cancer to be indistinguishable from non-responders in `Ipi`, which consists of solely patients with skin cancer. To verify if CL-BRUNO captures this relationship, we split the test set of `Nivo` and compute the averaged predicted probabilities in a similar fashion as before. From Fig 5 (d) we see non-responders in `Nivo` with skin cancer are much more likely to be classified as `Ipi` in comparison with the rest. This is supported by the visualisation in Fig 5 (b) as we can see a clear overlap between the non-responders to `Nivo` and `Ipi` with skin cancer. This unique pattern suggests that CL-BRUNO accurately captures the inter-task relationship between `Nivo` and `Ipi`, and further confirms that CL-BRUNO is capable of capturing inter-task relationships under a TIL scenario.

5.3 CIFAR100

We first evaluate CL-BRUNO in a class-incremental learning setting using the CIFAR-100 dataset. CIFAR-100 contains 3-channel images of size 32×32 from 100 classes, and each class includes 500 training images and 50 test images. In this class-incremental learning setup, we follow Lopez-Paz and Ranzato (2017) and create the sequential data set by splitting CIFAR-100 into 10 equal sized batches where each batch contains images from 10 classes.

Here we compare CL-BRUNO with a recently published experience replay-based methods (NPCL (Jha et al., 2024)), three non-generative exemplar-free methods (EWC (Kirkpatrick et al., 2017), LwF (Li and Hoiem,

2017), SSRE (Zhu et al., 2022)) and two generative exemplar-free methods (MeRGAN (Wu et al., 2018), FeTRIL (Petit et al., 2023)). For a fair comparison with existing methods, we follow Jha et al. (2024) and adopt the settings given in the Mammoth CL benchmark model (Boschini et al., 2022) for all existing methods except MeRGAN (For MeRGAN we use the default setting). For experience replay-based NPCL, we fix the buffer size $M = 500$ as recommended by the authors.

Our CL-BRUNO is specified as follows: We use a Resnet18 (He et al., 2016) as a feature extractor (i.e. using the output of the second-last layer of Resnet18, a 512-dimensional real vector, as the transformed feature of the corresponding input image) and apply CL-BRUNO to the 512-dimensional feature vectors. Specifically, the Resnet18 feature extractor here is only trained using the images from the first batch of data (10 classes) in the initial state, and is then frozen for the reminder of the class-incremental learning process. Similar strategies have been used in exemplar-free generative methods such as FeTriL (Petit et al., 2023). (In principle, we could directly specify a generative model on raw images instead of feature vectors. However, we do not consider it here as our method is motivated by biomedical rather than visual applications. In this example, we set the number of coupling layers in CL-BRUNO to be 6, the dimension of class embedding to be 128, size of pseudo data $N' = 128$ and regularisation strength $\alpha_1 = \alpha_2 = 1$. We report the classification accuracy on test set for each of the methods in Table 1.

5.4 MNIST

In addition to CIL, we also evaluate CL-BRUNO in a task-incremental learning setting using the MNIST dataset (LeCun, 1998). The MNIST dataset consists of images of handwritten digits from 0 to 9. We follow the experiment setup in Egorov et al. (2021), and split the MNIST dataset into 5 batches, where each batch consists of images associated with two digits. We view each batch as a binary classification task (0 vs 1, 2 vs 3, ..., 8 vs 9). The models need to learn these 5 tasks incrementally. In addition to the methods mentioned above, we also compare our method with Boo-VAE (Egorov et al., 2021), a VAE-based generative exemplar-free CL method, in this task-incremental learning example. For Boo-VAE, we use the default settings as described in the paper. For the rest of the methods, we use the same experiment setup discussed above. Results are reported in Table 1. We see CL-BRUNO outperforms existing methods in both experiments.

6 Conclusion

We propose CL-BRUNO, a probabilistic, exemplar-free CL model based on exchangeable sequence modelling. Compared with existing generative continual learning methods, our proposed method provides a unified probabilistic framework capable of handling different types of CL problems such as TIL and CIL, and giving easy-to-interpret probabilistic predictions without the need of training or maintaining a separate classifier. These features make our approach appealing in applications where data privacy and uncertainty quantification are of concern.

References

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. (2018). Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154.

Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. (2019). Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32.

Boschini, M., Bonicelli, L., Buzzega, P., Porrello, A., and Calderara, S. (2022). Class-incremental continual learning into the extended der-verse. *IEEE transactions on pattern analysis and machine intelligence*, 45(5):5497–5512.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H., and Ranzato, M. (2019). On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*.

Chen, P., Zhang, Y., Li, Z., and Sun, L. (2022). Few-shot incremental learning for label-to-image translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3697–3707.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. *Advances in neural information processing systems*, 31.

Dhar, P., Singh, R. V., Peng, K.-C., Wu, Z., and Chellappa, R. (2019). Learning without memorizing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5138–5146.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*.

Ditzler, G., Roveri, M., Alippi, C., and Polikar, R. (2015). Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25.

Dohare, S., Hernandez-Garcia, J. F., Lan, Q., Rahman, P., Mahmood, A. R., and Sutton, R. S. (2024). Loss of plasticity in deep continual learning. *Nature*, 632(8026):768–774.

Draelos, T. J., Miner, N. E., Lamb, C. C., Cox, J. A., Vineyard, C. M., Carlson, K. D., Severa, W. M., James, C. D., and Aimone, J. B. (2017). Neurogenesis deep learning: Extending deep networks to accommodate new classes. In *2017 international joint conference on neural networks (IJCNN)*, pages 526–533. IEEE.

Eddy, J. A., Thorsson, V., Lamb, A. E., Gibbs, D. L., Heimann, C., Yu, J. X., Chung, V., Chae, Y., Dang, K., Vincent, B. G., et al. (2020). Cri iatlas: an interactive portal for immuno-oncology research. *F1000Research*, 9.

Egorov, E., Kuzina, A., and Burnaev, E. (2021). Boovae: Boosting approach for continual learning of vae. *Advances in Neural Information Processing Systems*, 34:17889–17901.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S., and Teh, Y. W. (2018). Neural processes. *arXiv preprint arXiv:1807.01622*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

Gopalakrishnan, S., Singh, P. R., Fayek, H., Ramasamy, S., and Ambikapathi, A. (2022). Knowledge capture and replay for continual learning. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 10–18.

Gurbuz, M. B. and Dovrolis, C. (2022). Nispa: Neuro-inspired stability-plasticity adaptation for continual learning in sparse networks. *arXiv preprint arXiv:2206.09117*.

Hassabis, D., Kumaran, D., Summerfield, C., and Botvinick, M. (2017). Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258.

He, J. and Zhu, F. (2022). Exemplar-free online continual learning. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 541–545. IEEE.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hoadley, K. A., Yau, C., Hinoue, T., Wolf, D. M., Lazar, A. J., Drill, E., Shen, R., Taylor, A. M., Cherniack, A. D., Thorsson, V., et al. (2018). Cell-of-origin patterns dominate the molecular classification of 10,000 tumors from 33 types of cancer. *Cell*, 173(2):291–304.

Hung, C.-Y., Tu, C.-H., Wu, C.-E., Chen, C.-H., Chan, Y.-M., and Chen, C.-S. (2019). Compacting, picking and growing for unforgetting continual learning. *Advances in neural information processing systems*, 32.

Hyndman, R. J. (1996). Computing and graphing highest density regions. *The American Statistician*, 50(2):120–126.

Jha, S., Gong, D., Zhao, H., and Yao, L. (2024). Npcl: Neural processes for uncertainty-aware continual learning. *Advances in Neural Information Processing Systems*, 36.

Kang, H., Mina, R. J. L., Madjid, S. R. H., Yoon, J., Hasegawa-Johnson, M., Hwang, S. J., and Yoo, C. D. (2022). Forget-free continual learning with winning subnetworks. In *International Conference on Machine Learning*, pages 10734–10750. PMLR.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. (2019). Attentive neural processes. In *International Conference on Learning Representations*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Korshunova, I., Gal, Y., Gretton, A., and Dambre, J. (2020). Conditional bruno: A neural process for exchangeable labelled data. *Neurocomputing*, 416:305–309.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

LeCun, Y. (1998). The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

Lee, S., Ha, J., Zhang, D., and Kim, G. (2020). A neural dirichlet process mixture model for task-free continual learning. *arXiv preprint arXiv:2001.00689*.

Li, Z. and Hoiem, D. (2017). Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. (2023). Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*.

Lopez-Paz, D. and Ranzato, M. (2017). Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30.

Mallya, A., Davis, D., and Lazebnik, S. (2018). Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European conference on computer vision (ECCV)*, pages 67–82.

McClelland, J. L., McNaughton, B. L., and O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419.

McCloskey, M. and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Michel, N., Wang, M., Xiao, L., and Yamasaki, T. (2024). Rethinking momentum knowledge distillation in online continual learning. *arXiv preprint arXiv:2309.02870*.

Onken, D., Fung, S. W., Li, X., and Ruthotto, L. (2021). Ot-flow: Fast and accurate continuous normalizing flows via optimal transport. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9223–9232.

Pal, S. K., Uzzo, R., Karam, J. A., Master, V. A., Donskov, F., Suarez, C., Albiges, L., Rini, B., Tomita, Y., Kann, A. G., et al. (2022). Adjuvant atezolizumab versus placebo for patients with renal cell carcinoma at increased risk of recurrence following resection (immotion010): a multicentre, randomised, double-blind, phase 3 trial. *The Lancet*, 400(10358):1103–1116.

Papamakarios, G., Pavlakou, T., and Murray, I. (2017). Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71.

Petit, G., Popescu, A., Schindler, H., Picard, D., and Delezoide, B. (2023). Fetril: Feature translation for exemplar-free class-incremental learning. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3911–3920.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR.

Robins, A. (1995). Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146.

Rudner, T. G., Smith, F. B., Feng, Q., Teh, Y. W., and Gal, Y. (2022). Continual learning via sequential function-space variational inference. In *International Conference on Machine Learning*, pages 18871–18887. PMLR.

Scardapane, S., Uncini, A., et al. (2020). Pseudo-rehearsal for continual learning with normalizing flows. In *4th Lifelong Machine Learning Workshop at ICML 2020*.

Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. (2018). Progress & compress: A scalable framework for continual learning. In *International conference on machine learning*, pages 4528–4537. PMLR.

Shi, Y., De Bortoli, V., Campbell, A., and Doucet, A. (2024). Diffusion schrödinger bridge matching. *Advances in Neural Information Processing Systems*, 36.

Shim, D., Mai, Z., Jeong, J., Sanner, S., Kim, H., and Jang, J. (2021). Online class-incremental continual learning with adversarial shapley value. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9630–9638.

Shin, H., Lee, J. K., Kim, J., and Kim, J. (2017). Continual learning with deep generative replay. *Advances in neural information processing systems*, 30.

Thapa, J. and Li, R. (2024). Bayesian adaptation of network depth and width for continual learning. In *Forty-first International Conference on Machine Learning*.

Titsias, M. K., Schwarz, J., Matthews, A. G. d. G., Pascanu, R., and Teh, Y. W. (2019). Functional regularisation for continual learning with gaussian processes. *arXiv preprint arXiv:1901.11356*.

Van de Ven, G. M. and Tolias, A. S. (2019). Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Wang, L., Zhang, M., Jia, Z., Li, Q., Bao, C., Ma, K., Zhu, J., and Zhong, Y. (2021). Afec: Active forgetting of negative transfer in continual learning. *Advances in Neural Information Processing Systems*, 34:22379–22391.

Wang, L., Zhang, X., Su, H., and Zhu, J. (2024). A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Wen, Y., Tan, Z., Zheng, K., Xie, C., and Huang, W. (2024). Provable contrastive continual learning. *arXiv preprint arXiv:2405.18756*.

Wortsman, M., Ramanujan, V., Liu, R., Kembhavi, A., Rastegari, M., Yosinski, J., and Farhadi, A. (2020). Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33:15173–15184.

Wu, C., Herranz, L., Liu, X., Van De Weijer, J., Raducanu, B., et al. (2018). Memory replay gans: Learning to generate new categories without forgetting. *Advances in neural information processing systems*, 31.

Xu, J., Dupont, E., Märtens, K., Rainforth, T., and Teh, Y. W. (2024). Deep stochastic processes via functional markov transition operators. *Advances in Neural Information Processing Systems*, 36.

Xue, M., Zhang, H., Song, J., and Song, M. (2022). Meta-attention for vit-backed continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 150–159.

Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR.

Zhai, M., Chen, L., and Mori, G. (2021). Hyper-lifelonggan: Scalable lifelong learning for image conditioned generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2246–2255.

Zhou, D.-W., Wang, Q.-W., Qi, Z.-H., Ye, H.-J., Zhan, D.-C., and Liu, Z. (2024). Class-incremental learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zhu, K., Zhai, W., Cao, Y., Luo, J., and Zha, Z.-J. (2022). Self-sustaining representation expansion for non-exemplar class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9296–9305.

A Additional discussion on CL-BRUNO

A.1 Discussion on CIL loss

In this section, we justify the choice of L_{CIL} in Sec 3.1. L_{CIL} takes the form

$$L_{\mathrm{CIL}}(\theta) = \overbrace{-\log p_{\theta,\hat{\lambda}_k^{(T)}}(\mathcal{X}_k^{(1)}|k,\mathcal{Y}_k^{(1)},\mathcal{D}_k)}^{\text{neg log lkd of new batch given old}} + \alpha_1 L'(\theta,\{\hat{\lambda}_t^{(T)}\}_{t=1}^T;\{\mathcal{D}'_t\}_{t=1}^T) + \alpha_2 R(\theta;\hat{\theta}^{(T)},\{\mathcal{D}'_t\}_{t=1}^T), \tag{7}$$

where

$$\log p_{\theta,\hat{\lambda}_k^{(T)}}(\mathcal{X}_k^{(1)}|k,\mathcal{Y}_k^{(1)},\mathcal{D}_k) = \sum_{i=1}^{N_k^{(1)}} \log\left(p_{\hat{\lambda}_k^{(T)}}(\mathbf{z}_{i,k}^{(1)}|\mathbf{z}_{1:i-1,k}^{(1)},\hat{\mathbf{z}}_{1:N_k,k})\times \left|\det \frac{\partial f_\theta(X_{i,k}^{(1)};k,Y_{i,k}^{(1)})}{\partial X_{i,k}^{(1)}}\right|\right). \tag{8}$$

Recall that under the TIL scenario, the trained CL-BRUNO does not depend on the ordering of samples in each dataset, as we assumed that samples within each task are exchangeable. Similarly, under the CIL scenario, we also expect the trained model to *not depend* on the ordering of data batches the model has been trained on (e.g. a model first trained on $\{\mathcal{X}_k,\mathcal{Y}_k\}$ and then $\{\mathcal{X}_k^{(1)},\mathcal{Y}_k^{(1)}\}$ should be the same as a model trained on the reverse order). This requirement is easily satisfied by our CL-BRUNO model under the assumption that the concatenated $\{\mathcal{X}_k,\mathcal{X}_k^{(1)}\}$ is jointly exchangeable given $\{\mathcal{Y}_k,\mathcal{Y}_k^{(1)}\}$, i.e. the two data batches $\{\mathcal{X}_k,\mathcal{Y}_k\}$ $\{\mathcal{X}_k^{(1)},\mathcal{Y}_k^{(1)}\}$ associated with the k-th task are "segments" of a sequence generated from an underlying exchangeable generative process.

In particular, under the distributional assumption above, CL-BRUNO aims to learn the joint distribution $p(\mathcal{X}_k^{(1)},\mathcal{X}_k|t,\mathcal{Y}_k^{(1)},\mathcal{Y}_k)$ of the concatenated exchangeable sequence $\{\mathcal{X}_k^{(1)},\mathcal{X}_k\}$ given the label $\{\mathcal{Y}_k^{(1)},\mathcal{Y}_k\}$ while retaining the marginal $p_{\hat{\theta}^{(T)},\hat{\lambda}_k^{(T)}}(\mathcal{X}_k|t,\mathcal{Y}_k)$ the model has learnt from historical dataset. In other words, terms in L_{CIL} associated with the k-th task can be interpreted as learning the joint

$$p(\mathcal{X}_k^{(1)},\mathcal{X}_k|t,\mathcal{Y}_k^{(1)},\mathcal{Y}_k) = p(\mathcal{X}_k^{(1)}|t,\mathcal{Y}_k^{(1)},\mathcal{X}_k,\mathcal{Y}_k)p(\mathcal{X}_k|t,\mathcal{Y}_k)$$

under the constraint that

$$p(\mathcal{X}_k|t,\mathcal{Y}_k) = p_{\hat{\theta}^{(T)},\hat{\lambda}_k^{(T)}}(\mathcal{X}_k|t,\mathcal{Y}_k).$$

Note that the first term in L_{CIL} forces the normalising flow to learn the conditional $p(\mathcal{X}_k^{(1)}|t,\mathcal{Y}_k^{(1)},\mathcal{X}_k,\mathcal{Y}_k)$ under the marginal constraint, while the regularisers in L_{CIL} related to the k-th task in Eqn equation 7 enforce the marginal constraint by penalising both distributional and functional deviation between the new $p_{\theta,\hat{\lambda}_k^{(T)}}(\mathcal{X}_k|t,\mathcal{Y}_k)$ and the old $p_{\hat{\theta}^{(T)},\hat{\lambda}_k^{(T)}}(\mathcal{X}_k|t,\mathcal{Y}_k)$. The latent distribution parameter $\hat{\lambda}_k^{(T)}$ are not updated alongside with the normalising flow parameter θ. We choose to do so as it simplifies the computation and ensures the tractability of $p_{\hat{\lambda}_k^{(T)}}(\cdot|\hat{\mathbf{z}}_{1:N_k,k})$. This choice can also be viewed as an additional regularisation that aims to mitigate the impact of catastrophic forgetting.

A.2 Likelihood-based updating scheme of CL-BRUNO

Following the discussion in Section 3, we highlight the likelihood-based training strategy of CL-BRUNO: In the task incremental learning (TIL) scenario, CL-BRUNO extracts distributional information from the dataset by directly maximising the log likelihood of the observed dataset (in contrast with e.g. evidence lower bound in VAE based method such as BooVAE (Egorov et al., 2021), or the classification loss in GAN-type models such as MeRGAN (Wu et al., 2018)). In the class-incremental learning (CIL) scenario, the loss function Eqn 7 is also derived directly from the Bayes formula, providing an intuitive and principled learning scheme.

In addition to the likelihood-based updating scheme, CL-BRUNO also offers intuitive and principled uncertainty quantification: CL-BRUNO is able to estimate probability distributions of both class- and task-identity of a test point. Note that these quantities can be interpreted as (approximate) Bayes classifiers that enjoy various desirable properties.

A.3 Outlier detection using CL-BRUNO

In Sec 3.1, we give task-identity estimate of a generic test point X^* under the assumption that X^* is indeed drawn from the generative process of one of the previously seen $\mathcal{D}_t$s. This assumption no longer holds when X^* is e.g. an outlier. Thanks to the tractable generative modelling framework of CL-BRUNO, users can identify outliers in a straightforward fashion using e.g. level sets (Hyndman, 1996): For each task t, we first generate a pseudo dataset $\hat{\mathcal{D}}_t$ as in Sec 3.1 and discard the pseudo labels $\hat{Y}_{i,t}$, then for each generated pseudo feature vector $\hat{X}_{i,t}$, $i = 1, ..., N'$, we evaluate

$$\hat{p}_t(\hat{X}_{i,t}) = p_{\hat{\theta}^{(T)},\hat{\lambda}_t}(\hat{X}_{i,t}|t, \mathcal{D}_t) \tag{9}$$

$$= \sum_{Y^* \in \mathcal{C}_t} p_{\hat{\theta}^{(T)},\hat{\lambda}_t}(\hat{X}_{i,t}|t, Y^*, \mathcal{D}_t)p(Y^*|t), \tag{10}$$

the approximate predictive density of observing $\hat{X}_{i,t}$ as the next sample conditioned on $\mathcal{D}_t$ marginalised over all labels $Y^* \in \mathcal{C}_t$ where $p(\cdot|t)$ is the prior that generates the pseudo labels $\hat{Y}_{i,t}$. Note that by definition, each $\hat{X}_{i,t}$ is indeed a sample drawn from $p_{\hat{\theta}^{(T)},\hat{\lambda}_t}(\cdot|t, \mathcal{D}_t)$, the approximate marginal predictive distribution of feature vectors from the t-th task learnt by CL-BRUNO under the chosen label prior $p(\cdot|t)$. Let $\alpha \in (0, 1)$ and $\hat{p}_{\alpha,t}$ be the α% quantile of $\{\hat{p}_t(\hat{X}_{i,t})\}_{i=1}^{N'}$. By Hyndman (1996), level set $S_t = \{X' : \hat{p}_t(X') > \hat{p}_{\alpha,t}\}$ defines a $(1 - \alpha)$% approximate highest density region of $p_{\hat{\theta}^{(T)},\hat{\lambda}_t}(\cdot|t, \mathcal{D}_t)$. As a result, one could test if X^* is a typical feature vector associated with the t-th task in a natural and interpretable fashion by comparing $\hat{p}_t(X^*)$ with the threshold $\hat{p}_{\alpha,t}$.

B Additional experiments

We include additional synthetic and real-world examples in this section.

B.1 Synthetic data

Here we demonstrate CL-BRUNO using a synthetic dataset. We set data dimension $D = 1000$, and consider $T = 4$ tasks. Each task $t = 1, ..., T$ is associated with a classification problem with $C_t = t + 1$ distinct classes. Each synthetic dataset $\mathcal{D}_t$ consists of $N_t = 500$ samples, where labels $Y_{i,t} \sim Unif(\{1, ..., C_t\})$, and features $X_{i,t}|Y_{i,t} \sim N(\boldsymbol{\mu}_{i,t}, 0.5\mathbf{I}_D)$ with $\mathbf{I}_D$ being a $D \times D$ identity matrix, $\boldsymbol{\mu}_{i,t} = \{\mu_{i,t}^{(d)}\}_{i=1}^{D}$ and $\mu_{i,t}^{(d)} = \sqrt{t}\sin(2\pi Y_{i,t}/C_t)$ if d is odd and $\mu_{i,t}^{(d)} = \sqrt{t}\cos(2\pi Y_{i,t}/C_t)$ if d is even. See Fig () for an illustration of the synthetic datasets. We first initialise our proposed model on task 1 by training the generative on $\mathcal{D}_1$, then we incrementally present task 2 ($\mathcal{D}_2$) and 3 ($\mathcal{D}_3$) to the model. We then demonstrate the class-incremental scenario using task 4: We split $\mathcal{D}_4$ into two groups, one containing data associated with the first three out of the five classes in $\mathcal{D}_4$, and the other containing the rest two classes. We then present the two groups of data sequentially to the model. As a result, the example consists of 5 incremental training steps in total (1 for initialisation, 2 for task-incremental learning and 2 for class-incremental learning). We parameterise task ID t and each

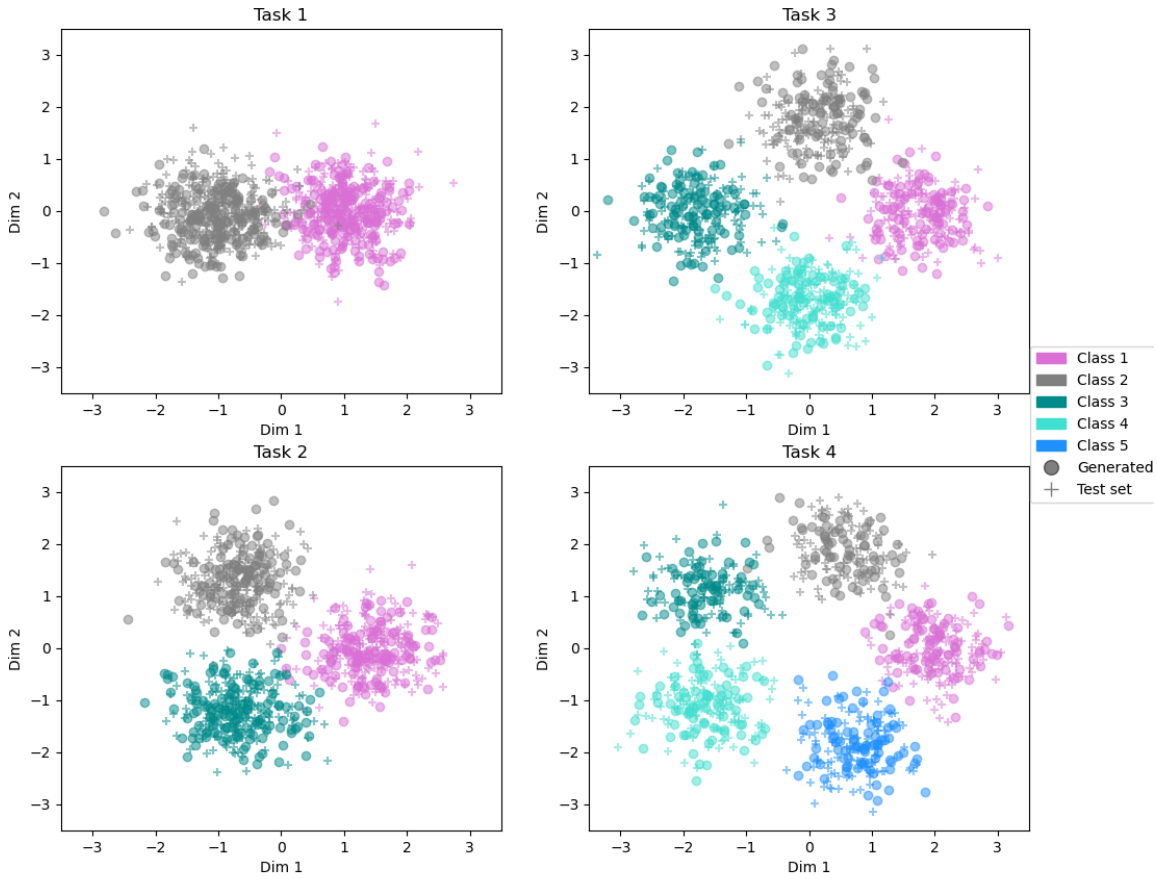


Figure 6: Scatter plots of the first two dimensions of samples in the test set (cross) and samples generated from the trained CL-BRUNO (circle) for each of the 4 tasks.

distinct label associated with task t for each task $t = 1, ..., T$ as trainable embeddings of length 16. After the training phase, we test the classification accuracy on all four tasks using separate test sets drawn from the same generative process. Note that no samples in training data $\{\mathcal{D}\}_{t=1}^{T}$ is retained in our trained model.

For all four tasks, the trained model attains $< 1\%$ misclassification rate on test sets of size 1000 samples. CL-BRUNO is also able to accurately predict task identities of samples from test sets, attaining $< 5\%$ misclassification rate on all tasks. To visualise the fit of CL-BRUNO, we report scatter plots of the first two dimension of samples from the test set and samples generated by the trained CL-BRUNO in Fig 6. We see CL-BRUNO is able to accurately capture the feature distributions associated with each class within each task. This confirms the effectiveness of CL-BRUNO.

B.2 Discussion on computational cost

We first discuss the memory overhead of CL-BRUNO in comparison with existing methods using the S-CIFAR-100 dataset as an example. Since we follow the setup in Jha et al. (2024) and Boschini et al. (2022), all methods except MeRGAN use Resnet18, which consists of 11.7M parameters, as a backbone model. In addition to the backbone model, CL-BRUNO also maintains a C-BRUNO model consisting of ∼1.38M

parameters, and 100 class embeddings (12.8K parameters). As a result, CL-BRUNO in total requires storing ∼1.39M extra parameters in addition to the backbone Resnet18 model. In comparison, the replay-based method requires storing exemplar samples from historical datasets instead of extra model parameters. In our setup where the buffer size is fixed at 500, maintaining this buffer requires storing ∼1.54M floating point numbers, which is larger than the memory required by CL-BRUNO. In addition to exemplar samples, NPCL also requires maintaining an attention-based network consisting of another ∼12.8M parameters. Among generative replay models, MeRGAN requires ∼2.15M parameters under the default setting. FeTriL has a smaller extra memory requirement (∼0.3M parameters) in addition to the backbone Resnet18 as it encodes and stores historical data sets as fixed-length vectors instead of a full generative model. Although MeRGAN and FeTriL require less memory than CL-BRUNO, we see that CL-BRUNO outperforms them in terms of classification accuracy. In addition, we would like to highlight that MeRGAN and FeTriL are only designed for discriminative CIL, while CL-BRUNO is more versatile, offering more functionalities such as task incremental learning (Sec 3.1), task identity estimation (Sec 3.2) and outlier detection (Supplementary material A.3).

We then inspect the computational cost of CL-BRUNO in terms of wall clock time using the same S-CIFAR-100 example. All examples are executed on our machine with an AMD Ryzen7 2700 CPU and NVIDIA RTX 2060 GPU. Under the experiment setup described above, CL-BRUNO takes around 2.9×10^3s to complete. In comparison, the running time of non-generative, exemplar-free LwF and EWC are around 2.4×10^3s, whereas the generative, exemplar-free FeTriL takes around 3.2×10^3s. The experience replay-based NPCL takes around 3.5×10^5s to run. We see in this example that the computational cost of CL-BRUNO is on a scale comparable to the existing exemplar-free CL models.

Furthermore, we investigate the additional computational cost incurred by sampling pseudo-samples from the reference model in computing the distributional and functional regularisers in Eqn 3, 4. In this example, generating $N' = 128$ pseudo-samples in this scenario takes around 0.017s, while one step of stochastic gradient descent takes around 0.272s. We see that the computational cost of generating pseudo-samples in terms of running time is only around 7% of gradient descent steps in the incremental training procedure. This suggests that generating samples from the reference model does not incur high computational cost. To further examine the impact of pseudo sample size N', we run CL-BRUNO under three choices of pseudo samples $N'_1 = 32, N'_2 = 64, N'_3 = 128$. The running times of the three models are 2569 s, 2726 s, and 2917 s, respectively, on our machine. We see that increasing N' by a factor of 4 (from 32 to 128) only leads to a $\sim 14\%$ increase in the running time of CL-BRUNO. This further confirms that sampling from the reference model does not drastically increase the computational cost of CL-BRUNO.