Bayesian Adaptation of Network Depth and Width for Continual Learning

Jeevan Thapa [1] **Rui Li** [1]

Abstract

While existing dynamic architecture-based continual learning methods adapt network width by growing new branches, they overlook the critical aspect of network depth. We propose a novel non-parametric Bayesian approach to infer network depth and adapt network width while maintaining model performance across tasks. Specifically, we model the growth of network depth with a beta process and apply drop-connect regularization to network width using a conjugate Bernoulli process. Our results show that our proposed method achieves superior or comparable performance with state-of-the-art methods across various continual learning benchmarks. Moreover, our approach can be readily extended to unsupervised continual learning, showcasing competitive performance compared to existing techniques.

1. Introduction

Continual learning, also known as incremental or lifelong learning, entails observing a sequence of tasks and incrementally acquiring new knowledge while sustaining performance levels for previously learned tasks. It has become imperative in settings with evolving data distributions, limited resources, and privacy concerns, necessitating the retention of past knowledge without storing the past data (Parisi et al., 2019). Despite advancements in addressing catastrophic forgetting (French, 1999) in deep neural networks, existing approaches to continual learning still face limitations.

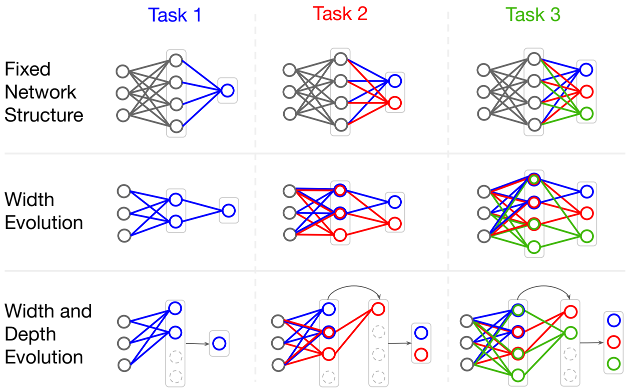


Figure 1: *Top Row*: Both the number of hidden layers (i.e., depth) and the number of neurons (i.e., width) per layer are kept constant for fixed network structure. *Middle Row*: Width evolution methods expand width by adding or activating more neurons with a pre-determined depth. *Bottom Row*: Our method enables both network depth and width to evolve as stochastic processes based on the varying task complexity.

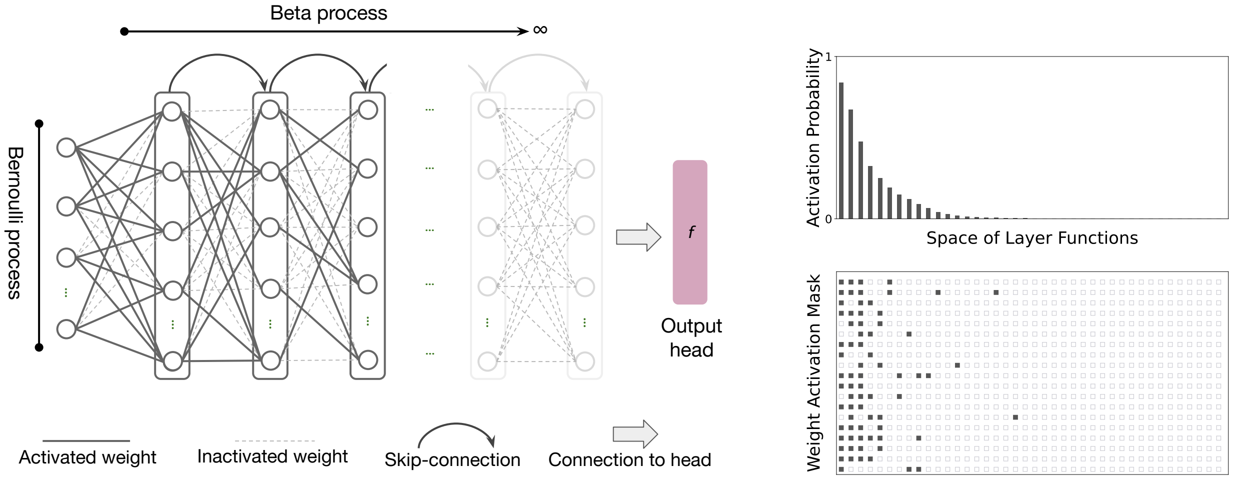
An effective strategy for continual learning involves the utilization of dynamic architecture architecture, which grows the model structure dynamically as new tasks arise. Approaches exemplified in (Rusu et al., 2016; Yoon et al., 2018) augment the model's width and incorporate distinct sub-networks for individual tasks. Despite their considerable success in mitigating forgetting, these methods cannot adjust the network depth—a crucial factor for overall model performance, whereas recent advances in deep learning highlight the significance of depth for optimal network performance (He et al., 2015; Brown et al., 2020).

Another successful method for continual learning involves utilizing the natural sequential Bayes approach. VCL (Nguyen et al., 2018) employs the posterior of neural network weights for the current task t as the prior for the next task $t + 1$ to overcome catastrophic forgetting while also enabling knowledge transfer in both forward and backward directions. This sequential Bayes method is a type of prior-focused or regularization-based method. Another Bayesian continual learning method (Adel et al., 2020) introduces task-specific weight adjustments. However, the fixed network structures limit the available model capacity for the ever-evolving lifelong scenario.

Recent works such as HIBNN (Kessler et al., 2021) and IBPCL (Kumar et al., 2021) combine the powers of these two worlds and extend the sequential Bayes on network parameters to incorporate model structure inference. However,

[1] College of Computing and Information Sciences, Rochester Institute of Technology, Rochester, New York, USA. Correspondence to: Rui Li <rxlics@rit.edu>.

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



(a) Bayesian structure inference (b) Beta process and its conjugate Bernoulli process

Figure 2: Demonstration of our Bayesian inference framework. (a) We model the growth of network depth with a beta process, enabling it to go to infinity in theory. Skip connections allow us to feed the outputs of the last activated hidden layer to the output head. The activated weights in each layer are depicted with dark-shaded connections, while dropped weights are illustrated with faint dashed connections. (b) A visualization of a beta process (top) and its conjugate Bernoulli process (bottom): the bars in the top row represent per-layer activation probabilities derived from the stick-breaking construction of the beta process. The bottom shows layer-wise weight activation masks (in columns) sampled from the conjugate Bernoulli process, where each column corresponds to flattened layer-wise weight activation masks. The activated weights are represented by solid dark squares, and inactivated weights are depicted by empty squares.

they still focus on extending network width, i.e., the number of active neurons in a hidden layer, with an Indian Buffet Process (IBP) (Griffiths & Ghahramani, 2011) prior, as in Figure 1. Given the importance of model depth in determining neural network performance, continual learning models with a fixed depth still face limitations as tasks are incrementally available, resulting in performance bottlenecks.

We thus propose a novel non-parametric Bayesian framework that enables both network depth and width to evolve simultaneously for continual learning. Specifically, we model the growth of neural network depth as a beta process and generate a conjugate Bernoulli process for drop-connect regularization on network width. We further develop an efficient estimator for joint inference of network parameters and network structures. We demonstrate the framework's compatibility with different types of backbone networks and showcase performance improvement over existing methods across benchmark datasets. Our method can be readily applied to both supervised and unsupervised learning scenarios. Extensive experiments show that by continuously updating network weights and adapting network depth and width across tasks, our method achieves superior or comparable performance to the existing state-of-the-art models. We also show competitive performance on unsupervised continual image generation.

2. Related Works

Current continual learning methods can be broadly classified into three categories: regularization-based methods (Zenke et al., 2017; Kirkpatrick et al., 2016; Nguyen et al., 2018; Ahn et al., 2019; Adel et al., 2020; Rudner et al., 2022); replay-based techniques (Shin et al., 2017; Rolnick et al., 2019; Hayes et al., 2020; Acharya et al., 2020; Buzzega et al., 2020); and dynamic architecture based approaches (Rusu et al., 2016; Yoon et al., 2018; Serrà et al., 2018; Yan et al., 2021).

Regularization-based methods: Weight regularization-based methods aim to control the variations in the network weights across different tasks. EWC (Kirkpatrick et al., 2016) uses Fisher Information Matrix to find important weights for each task, and penalizes changes for those weights. VCL (Nguyen et al., 2018) employs a sequential Bayesian approach in the Bayesian neural network and regularizes changes in weight distribution over tasks. Other approaches like (Ahn et al., 2019; Adel et al., 2020) follow the similar line of work. Similarly, within the sequential Bayesian framework, SFSVI (Rudner et al., 2022) diverges from weight regularization, opting for function space regularization. This requires an additional dataset as inducing points to regularize the function space across tasks. Although the sequential Bayesian approach applied in VCL and similar works offers a principled framework for continual learning, the model's capacity remains bounded by the fixed network structures.

Dynamic architecture-based methods: To address the bottleneck of fixed model capacity, dynamic architectures grow network structures to accommodate shifts with new tasks. Specifically, some methods (Rusu et al., 2016; Yoon et al., 2018) grow separate branches width-wise for each task. HAT (Serrà et al., 2018) learns task-specific hard attention masks on the neuron activations. Similarly, SPG (Konishi et al., 2023) uses task-specific soft parameter masks in gradients during training. And, UCB (Ebrahimi et al., 2020) samples task-specific substructures using signal-to-noise ratio for each weight in the Bayesian neural network. However, these methods are not able to adapt the network depth. Similarly, Sequential Bayes approaches, such as HIBNN (Kessler et al., 2021) and IBPCL (Kumar et al., 2021), apply IBP priors on the network structure and adapt the network width in light of data. IBPCL applies the IBP prior to select features from the previous layer for current node activation, and it learns task-specific masks by sampling from the IBP posterior. HIBNN employs an IBP prior to infer node activations for each data point and extends it to a hierarchical IBP (Thibaux & Jordan, 2007) to enable regular model structure. Yet, these methods can adapt network width only. On the other hand, network architecture adaptation methods for batch learning (Cortes et al., 2017; Antorán et al., 2020; KC et al., 2021; Nazaret & Blei, 2022) cannot be straightforwardly applied to address the unique challenges posed by continual learning.

Unsupervised continual learning: While Bayesian continual learning such as VCL (Nguyen et al., 2018) and IBPCL (Kumar et al., 2021) show success in unsupervised settings by adapting the network widths of variational auto-encoders (VAEs), other recent unsupervised continual learning works (Zhai et al., 2019; Ye & Bors, 2023) still have fixed network structures.

Bayesian network structure adaptation: While some recent works (KC et al., 2021; Regmi & Li, 2023) adopt stochastic processes for network adaptation, they are not designed to address catastrophic forgetting. Our adaptation method for continual learning differs from them in terms of: i) We propose a full sequential Bayesian inference to continuously co-adapt both network weights and network structure as tasks become incrementally available, rather than just generating a point estimate (i.e., MAP estimate) for the network weights. This contribution significantly enhances the model's knowledge transfer capability, as shown in our ablation study. ii) We apply binary masks directly over weights, rather than over neurons. This strategy improves the performance of mitigating forgetting, since masking out a neuron shuts off all its connected weights and leads to potential information loss. iii) We introduce a weight importance variable into our variational distribution to further customize the masks for task-incremental continual learning, which further improves knowledge retention.

3. Bayesian Continual Learning

3.1. Bayesian Neural Network

Early work on Bayesian neural networks dates back to (MacKay, 1992b;a). In contrast to deterministic neural networks, where only parameters are considered, Bayesian neural networks define a prior distribution $p(\theta)$ over network weights θ. The objective, given a dataset $\mathcal{D}$, is to infer the posterior over the weights, denoted as $p(\theta|\mathcal{D})$. Several approaches, such as Laplace Approximation (MacKay, 1992b) and Hamiltonian Monte Carlo (Neal, 2012), have been proposed for this purpose but they cannot scale to large datasets and network sizes. Variational inference (Hinton & van Camp, 1993; Graves, 2011) offers a more efficient solution by transforming the inference problem into an optimization problem. The framework introduces a variational distribution $q(\theta)$ within a variational family $\mathcal{Q}$ to approximate the true posterior $p(\theta|\mathcal{D})$, and this approximation is optimized as:

$$q(\theta) = \arg\min_{\tilde{q}\in\mathcal{Q}} \mathbb{KL}[\tilde{q}(\theta)||p(\theta|\mathcal{D})]$$

Specifically, Bayes-by-Backprop (Blundell et al., 2015) improves upon (Graves, 2011) by formulating an unbiased estimator for gradients using the re-parameterization trick. Our work extends this line of development to formulate a continual learner that sequentially learns the posterior over weights as well as the network structure.

3.2. Sequential Bayes

Given a sequence of tasks $\{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_T\}$, Bayesian inference can be readily applied to continual learning. The posterior of θ after observing t tasks can be obtained as:

$$p(\theta|\mathcal{D}_1, ..., \mathcal{D}_t) \propto p(\theta|\mathcal{D}_1, ..., \mathcal{D}_{t-1})p(\mathcal{D}_t\,|\theta) \quad (1)$$

The above recursion shows that the posterior for task t can be inferred by using the likelihood of the current data $\mathcal{D}_t$ and treating the posterior learned from the past $t-1$ tasks as the prior. With θ representing only network weights, VCL (Nguyen et al., 2018) leverages this approach to sequentially update the distribution of the model weights given different tasks. IBPCL (Kumar et al., 2021) and HIBNN (Kessler et al., 2021) apply the procedure to both the model weights and width with different prior distributions over them. We take a step further to jointly adapt the network depth and width with two distinct yet related stochastic process priors.

4. Network Depth and Width Adaptation for Continual Learning

Our non-parametric Bayesian continual learning method models the growth of network depth with a beta process and adapts its width through drop-connect regularization based on a conjugate Bernoulli process. Since the exact posterior inference is intractable, we design a stochastic variational inference procedure to jointly update the posteriors for both network structures (i.e., depth and width) and network weights given new tasks.

4.1. Bayesian Inference for Network Structures

For brevity, we first describe the neural network structure as a likelihood and specify stochastic process priors without task information from continual learning scenarios.

4.1.1. Likelihood Formulation

We formulate a neural network with infinite depth, denoting the output $\mathbf{h}_l$ of layer l, with $l = 1, \ldots, \infty$, as:

$$\mathbf{h}_l = \sigma\left(\left(\mathbf{W}^{(l)} \odot \mathbf{Z}^{(l)}\right)\mathbf{h}_{l-1}\right) + \mathbf{h}_{l-1} \tag{2}$$

where, σ represents a non-linearity operation, $\mathbf{W}^{(l)} \in R^{M\times M}$ denotes a weight matrix, $\mathbf{Z}^{(l)}$ denotes a binary weight activation mask induced from a conjugate Bernoulli process for layer l, and $\odot$ denotes elementwise multiplication of the two matrices. M is the maximum number of nodes (i.e., width) in each hidden layer. We place a Gaussian prior over each weight parameter $w_{m,n}^{(l)}$ connecting $(l-1)^{\text{th}}$ layer's node m to layer l's node n, i.e. $w_{m,n}^{(l)} \sim \mathcal{N}(\mu, \sigma^2)$. The element $z_{m,n}^{(l)} = 1$ of the weight activation mask $\mathbf{Z}^{(l)}$ indicates the corresponding connection is activated and $z_{m,n}^{(l)} = 0$ deactivates the connection, as in Figure 2. For backbone networks other than fully connected ones, such as a convolutional neural network (CNN), we readily replace the weight matrices with convolutional kernel tensors and apply the binary layer-wise activation masks to filter these tensors. The skip connections allow the last activated layer to be connected to the output layer or head $f(\cdot)$ by skipping deactivated layers in between.

Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$, we thus specify the likelihood with the output layer of the neural network as:

$$p(\mathcal{D}\,|\mathcal{W}, \mathcal{Z}) = \prod_{i=1}^{N} \text{Cat}\left(\mathbf{y}_i | f(\mathbf{x}_i; \mathcal{W}, \mathcal{Z})\right) \tag{3}$$

where $\mathcal{W} = \{\mathbf{W}^{(l)}\}$ and $\mathcal{Z} = \{\mathbf{Z}^{(l)}\}$ denote weight tensors and weight activation masks across the layers, respectively.

4.1.2. Priors over Network Structures

We specify a beta process (Hjort, 1990) as a prior over the number of hidden layers l (i.e., network depth) and perform drop-connect regularization over the connections between adjacent layers (i.e., network width) with its conjugate Bernoulli process. Specifically, we compute the beta process with a stick-breaking construction (Paisley et al., 2010) as:

$$v_l \sim \text{Beta}(\alpha, \beta), \qquad \pi_l = \prod_{i=1}^{l} v_i \tag{4}$$

We sample v_l from a beta distribution with parameters α and β for each layer l. The activation probability of the l^{th} layer, denoted as π_l, is the cumulative product of the v_l samples up to the layer l. We proceed to sample the binary weight activation mask for the layer from a conjugate Bernoulli process, $z_{m,n}^{(l)} \sim \text{Ber}(\pi_l)$.

4.1.3. Structured Variational Inference

Exact inference of the weights $\mathcal{W}$ and the activation masks $\mathcal{Z}$ is intractable due to the non-linearity of neural network and $l \to \infty$. Hence, we rely on variational inference for approximate inference. Specifically, we employ structured stochastic variational inference (Hoffman & Blei, 2015), capturing the dependency between $\mathcal{Z}$ and $\mathbf{v}$. We specify the variational distribution $q(\mathbf{v}, \mathcal{Z}, \mathcal{W})$ with a truncation level K to approximate the true posterior as:

$$\begin{aligned} q(\mathbf{v}, \mathcal{Z}, \mathcal{W}) &= q(\mathbf{v})q(\mathcal{Z}|\mathbf{v})q(\mathcal{W}) \\ &= \prod_{k=1}^{K} q(v_k) \prod_{m=1}^{M}\prod_{n=1}^{M} q\left(z_{m,n}^{(k)}\right) q\left(w_{m,n}^{(k)}\right) \end{aligned} \tag{5}$$

where,

$$\begin{aligned} q(\mathbf{v}) &= \prod_{k=1}^{K} \text{Beta}(a_k, b_k), \\ q(\mathcal{Z}|\mathbf{v}) &= \prod_{k=1}^{K}\prod_{m=1}^{M}\prod_{n=1}^{M} \text{ConBer}\left(\prod_{i=1}^{k} v_k\right) \\ q(\mathcal{W}) &= \prod_{k=1}^{K}\prod_{m=1}^{M}\prod_{n=1}^{M} \mathcal{N}\left(\mu_{m,n}^{(k)}, \left(\sigma_{m,n}^{(k)}\right)^2\right) \end{aligned}$$

$\{\mu_{m,n}^{(k)}, \sigma_{m,n}^{(k)}\}$ and $\{a_k, b_k\}$ denote the variational parameters for the corresponding distributions, respectively. The truncation on the variational distribution $q(\mathbf{v})$ can be relaxed as in (Xu et al., 2019) to better approximate the true posterior with a potentially infinite number of layers. We also relax the discrete constraint for the binary activation masks $\mathcal{Z}$ with a concrete Bernoulli (Maddison et al., 2017; Jang et al., 2017).

4.2. Network Structure Adaptation

Following the sequential Bayes from Equation (1), we set the posterior obtained from the previous task $t-1$ as the prior for the current task t. The evidence lower bound (ELBO) to the log marginal likelihood of the current task t is:

$$\begin{aligned}\mathcal{L}(t) = \mathbb{E}_{q_t(\mathbf{v},\mathcal{Z},\mathcal{W})}\left[\log p(\mathcal{D}_t\,|\mathbf{v},\mathcal{Z},\mathcal{W}))\right]\\ -\mathbb{KL}\left[q_t(\mathbf{v})||q_{t-1}(\mathbf{v})\right]-\mathbb{KL}\left[q_t(\mathcal{Z}|\mathbf{v})||q_{t-1}(\mathcal{Z}|\mathbf{v})\right]\\ -\mathbb{KL}\left[q_t(\mathcal{W})||q_{t-1}(\mathcal{W})\right] \quad (6)\end{aligned}$$

where, $q_0(\mathbf{v}) = p(\mathbf{v})$, $q_0(\mathcal{Z}|\mathbf{v}) = p(\mathcal{Z}|\mathbf{v})$, and $q_0(\mathcal{W}) = p(\mathcal{W})$ are the initial priors. The first term on the RHS is the log-likelihood that adapts the network to the current task t. The first two Kullback–Leibler (KL) divergence terms on the network structure variables $\mathbf{v}$ and $\mathcal{Z}$ regularize the changes in network depth and width, and the KL on weights $\mathcal{W}$ regularizes the changes in the weight distribution. They jointly adapt the network structure and weight changes from the previous tasks to retain previously acquired knowledge. Overall, the ELBO allows a trade-off between model evolution to adapt to a new task and structural and weight regularization for knowledge retention.

We adopt Monte Carlo estimation to approximate the log-likelihood and the KL divergence on Bernoulli distributions. The KL divergence on weights $\mathcal{W}$ and the beta distributions can be analytically computed. Details about the computation of the ELBO are in Appendix.

4.3. Task-Incremental Learning

We apply our method for task-incremental learning by learning task-specific masks on the network weights. Specifically, we add a weight importance parameter $\gamma^{(k)}_{m,n}$ to our variational distribution, that captures the additional importance for each connection and thus, re-define the variational distribution as $q\left(z^{(k)}_{m,n}\right) = \text{ConBer}\left(z^{(k)}_{m,n}|\text{sigmoid}\left(\gamma^{(k)}_{m,n}+\text{logit}(\pi_k)\right)\right)$.

After updating the network structure parameters for a new task t, we sample a fixed mask $\bar{\mathcal{Z}}_t$ specific to the task. We sample R layer-wise activation probabilities $\pi^{(r)}_k$s from the stick-breaking construction. Then, we incorporate the weight importance $\gamma^{(k)}_{m,n}$ to get the corresponding weight activation probability. We accumulate the weight activation probabilities as $\boldsymbol{\pi}^{(r)}_t = \left\{\text{sigmoid}\left(\gamma^{(k)}_{m,n}+\text{logit}(\pi^{(r)}_k)\right)\right\}$, we calculate the task-specific average weight activation probability $\bar{\boldsymbol{\pi}}_t$ and sample the task-specific mask as:

$$\bar{\mathcal{Z}}_t \sim \text{Ber}\left(\bar{\boldsymbol{\pi}}_t\right), \qquad \bar{\boldsymbol{\pi}}_t = \frac{1}{R}\sum_{r=1}^{R}\boldsymbol{\pi}^{(r)}_t$$

Here, we make slight abuse of notation to sample the task-specific mask from a Bernoulli distribution. To reconcile the disparity arising from the non-zero temperature in concrete Bernoulli during training, we further tune the task-specific sub-network on the dataset $\mathcal{D}_t$. Although the task-specific sub-networks are especially relevant for task-incremental learning, they can be readily extended to class-incremental learning by including an explicit task identifier.

4.4. Unsupervised Continual Learning

Our method can also be extended to generative continual learning to enable model structure adaptation. In particular, we apply our method for sequential generation in a variational autoencoder (VAE) (Kingma & Welling, 2013).

Analogous to our supervised learning setting, we introduce a variational distribution $q(\mathbf{v},\mathcal{Z},\mathcal{W})$, and a latent representation $\mathbf{s}$ for each data point $\mathbf{x}_i$. Although adapting both encoder and decoder networks is possible as in (Regmi & Li, 2023), the primary performance contribution of a VAE model relies on the decoder/generator. We thus set up task-specific deterministic encoders $q_t(\mathbf{s}|\mathbf{x}_i)$, and apply our method to infer the network structure of the decoder $p(\mathbf{x}_i|\mathbf{s},\mathbf{v},\mathcal{Z},\mathcal{W})$ only. For a fair comparison, we experiment with an adaptive decoder accompanied by smaller task-specific heads as in VCL (Nguyen et al., 2018).

The overall ELBO integrating our structural inference into VAEs for unsupervised continual learning can thus be expressed as:

$$\begin{aligned}\mathcal{L}_{\text{VAE}}(t) = -\mathbb{KL}[q_t(\mathbf{v},\mathcal{Z},\mathcal{W})||q_{t-1}(\mathbf{v},\mathcal{Z},\mathcal{W})]+\\ \sum_{i=1}^{N_t}\Bigg[\mathbb{E}_{q_t(\mathbf{v},\mathcal{Z},\mathcal{W})}\left[\mathbb{E}_{q_t(\mathbf{s}|\mathbf{x}_i)}\left[\log p(\mathbf{x}_i|\mathbf{s},\mathbf{v},\mathcal{Z},\mathcal{W})\right]\right]\\ -\mathbb{KL}\left[q_t(\mathbf{s}|\mathbf{x}_i)||p(\mathbf{s})\right]\Bigg] \quad (7)\end{aligned}$$

where, $p(\mathbf{s})$ denotes the prior distribution over latent vector $\mathbf{s}$. The first term on the RHS of Equation (7) is the KL term that regularizes decoder structure and weight distribution, and thus mitigates catastrophic forgetting. The second term is the log-likelihood. The third term is the KL over the latent representation regularizing the distribution over the VAE latent variable.

5. Experiments

We analyze the behavior of our framework across various settings on benchmark datasets. First, we illustrate how our formulation enables the evolution of structure across tasks while maintaining model performance in prior tasks. Next, we evaluate our method in task-incremental learning using different backbone networks across benchmark datasets.

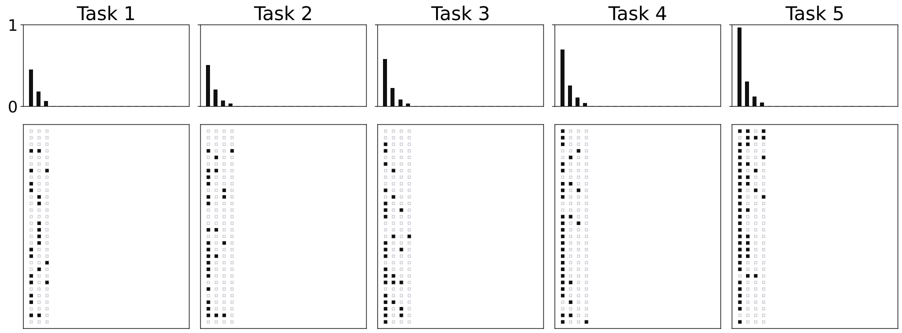


Figure 3: Evolution of the network structure in continual learning visualized through layer-wise activation probabilities (top) and weight activation masks (bottom) on split fashion MNIST. As the tasks become incrementally available, the activation probabilities increase and more weights in the corresponding layers are activated.

Table 1: Performance comparison on MNIST variants. Test accuracies are presented as percentages, averaged across five random seeds. The best results are highlighted in bold, while the second-best methods are underlined.

Methods	permuted MNIST	split MNIST	split fashion MNIST
EWC (Kirkpatrick et al., 2016)	96.86 ± 0.02	99.01 ± 0.03	97.87 ± 0.98
VCL (Nguyen et al., 2018)	97.11 ± 0.06	98.51 ± 0.07	98.61 ± 0.19
UCL (Ahn et al., 2019)	96.55 ± 0.34	99.75 ± 0.03	99.15 ± 0.16
SFSVI (Rudner et al., 2022)	97.53 ± 0.06	99.74 ± 0.05	99.12 ± 0.09
DEN (Yoon et al., 2018)	97.38 ± 0.04	99.01 ± 0.02	99.08 ± 0.07
HAT (Serrà et al., 2018)	97.48 ± 0.04	99.65 ± 0.01	98.93 ± 0.15
HIBNN (Kessler et al., 2021)	97.00 ± 0.30	98.71 ± 0.40	96.25 ± 1.97
IBPCL (Kumar et al., 2021)	**98.18 ± 0.19**	**99.79 ± 0.03**	99.27 ± 0.02
SPG (Konishi et al., 2023)	97.36 ± 0.12	99.28 ± 0.16	99.21 ± 0.03
Ours	97.98 ± 0.04	99.76 ± 0.05	**99.33 ± 0.06**

Following that, we conduct an ablation study to investigate the importance of each component in our method. Finally, we showcase the versatility of our approach by adapting it to unsupervised continual learning and class-incremental learning.[1]

5.1. Adaptive Network Structures for Continual Learning

Figure 3 demonstrates the evolution of layer-wise activations and their corresponding weight masks across tasks in a fully connected network with maximum width $M = 100$ and truncation $K = 5$ on split fashion MNIST dataset. We threshold out layer-wise masks with activation probabilities under 3% during sampling task-specific masks. The results show that as the new tasks come in, deeper layers are activated. Meanwhile, the activation probabilities of the activated shallow layers also gradually increase and their corresponding layer-wise weight masks become denser.

5.2. Applications on Fully-Connected Networks

We experiment with fully connected networks on permuted MNIST, split MNIST, and split fashion MNIST, all with 5 tasks. And, we use the final accuracy across tasks as the evaluation metric. We limit the network width to 200 for all experiments except for DEN, which grows width with new tasks. We keep other methods' hyper-parameters fixed at their best default settings, as specified in the respective papers. In our experiments, we set a truncation level K of 3. Additionally, for the baseline methods, we choose the best-performing models at depths of 1, 2, and 3.

Table 1 presents the results of our comparison. For permuted MNIST and split MNIST datasets, our method performs second only to IBPCL. Moreover, for the Fashion MNIST dataset, our method outperforms all competing methods, with IBPCL ranking second to our approach. This suggests the dominance of Bayesian regularization methods with task-specific masks over regularization-only and dynamic architecture-only methods.

We also investigate the effect of truncation level K and maximum width M settings on the performance and the

[1] The link to our codebase is `https://github.com/jt4812/bayes_struc_adap_cl`. All the details about implementations and settings are in Appendix.

Table 2: Performance evaluation with Alexnet architecture. Test accuracies are presented in percentages, averaged across five random seeds. The best results are highlighted in bold, and the second-best methods are underlined.

Methods	**CIFAR10-5**	**CIFAR100-10**	**CIFAR100-20**	**TinyImagenet-10**
EWC (Kirkpatrick et al., 2016)	81.54 ± 0.87	62.31 ± 1.09	65.20 ± 2.02	42.03 ± 0.75
VCL (Nguyen et al., 2018)	81.89 ± 2.32	64.79 ± 0.50	72.46 ± 0.82	40.07 ± 0.29
HAT (Serrà et al., 2018)	92.29 ± 0.19	**72.42 ± 0.36**	75.40 ± 0.53	44.98 ± 2.87
UCL (Ahn et al., 2019)	85.89 ± 0.42	64.80 ± 0.80	74.00 ± 0.60	46.46 ± 0.72
IBPCL (Kumar et al., 2021)	**92.53 ± 0.17**	68.84 ± 1.09	77.13 ± 0.94	48.46 ± 0.65
SPG (Konishi et al., 2023)	88.40 ± 0.40	68.94 ± 0.56	75.26 ± 0.49	49.72 ± 0.22
Ours	91.44 ± 0.16	71.83 ± 0.48	**79.93 ± 0.28**	**50.59 ± 0.42**

Note: We exclude HIBNN and DEN due to the lack of CNN implementations in their codebases. SFSVI's performance is in Appendix.

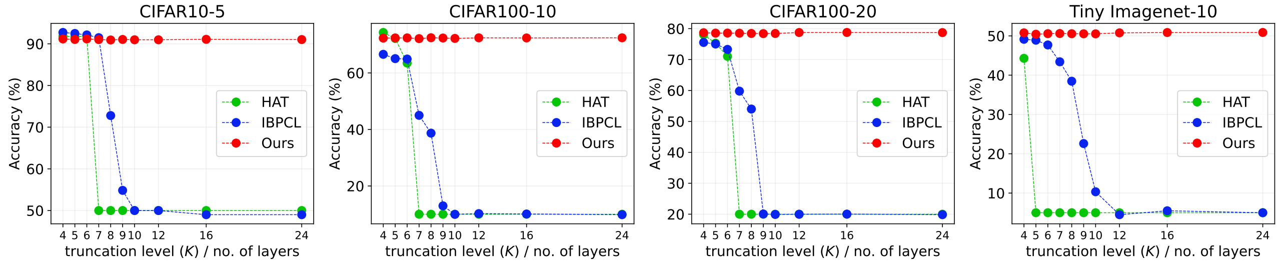


Figure 4: Performance of dynamic architecture-based methods changes with their network depths in comparison with our framework's truncation level (K).

inferred network structures of fully-connected networks in Appendix. The results show that the model performance and the inferred network depth remain stable with reasonably large truncation K and M. As the maximum network width M increases, the inferred depth decreases first and then becomes stable, while the model performance remains stable for a wide range of maximum network widths.

5.3. Applications on CNNs

We further evaluate our proposed framework with convolutional layers. Alexnet (Krizhevsky et al., 2012) is a popular choice in continual learning for vision datasets (Serrà et al., 2018; Kumar et al., 2021; Konishi et al., 2023). Hence, to compare against the existing methods, we adapt our method in the Alexnet architecture. And, to further showcase our method's capability for structure adaptation in CNN, we experiment with fully convolutional architectures.

5.3.1. Alexnet as the Backbone Network

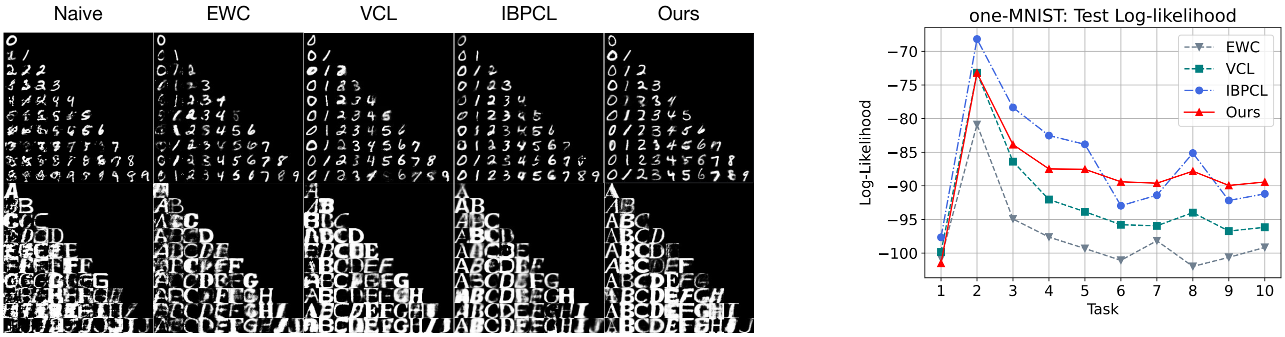
Following (Serrà et al., 2018; Kumar et al., 2021; Konishi et al., 2023), we use an architecture consisting of three convolutional layers with channel widths of 128, 256, and 512, along with two fully connected layers having a width of 2048. For our network, due to max-pooling layers after each convolutional layer, we omit the skip connections in the convolutional layers. Unlike IBPCL, we do not include any task-specific batch normalization.

The results, in Table 2, indicate the performance of our model across various vision datasets. Specifically, for CIFAR10-5, our model ranks third, behind HAT and IBPCL. In the case of CIFAR100-10, our model outperforms other methods except HAT. Interestingly, when faced with the more challenging CIFAR100-20 dataset, which entails twice the number of tasks compared to CIFAR100-10, our method surpasses the performance of all the baseline methods. Furthermore, on the relatively more difficult TinyImagenet-10 dataset, our proposed method exhibits superior performance compared to other methods.

5.3.2. Performance Comparison with Dynamic Architecture Methods

We investigate our model's capability to adapt network structure with different truncation levels compared to other dynamic architecture-based methods. For this experiment, we use a fully convolutional backbone with K convolutional layers only, corresponding to truncation level K. The first three layers have max pooling layers while the remaining $K-3$ layers do not. And, the structure inference is applied over the whole truncation.

Figure 4 shows the performance robustness of our framework and other dynamic architecture-based methods to the change of network depths. Our method is more robust to the specification of model depth/truncation levels with stable performance on CIFAR and Tiny-Imagenet datasets. The



(a) Sequential generations

(b) one-MNIST test log-likelihoods

Figure 5: Performance evaluation of our method in unsupervised continual learning. (a) The upper and lower rows consist of sequential generation results for one-MNIST and not-MNIST datasets, respectively. (b) Test log-likelihoods for one-MNIST across tasks.

performance of IBPCL and HAT is comparable with ours only for shallow network structures (i.e., $l \leq 5$). As we increase the model depth, their performance quickly drops. The performance of HAT collapses even faster. We speculate that this is because their task-specific masks accumulate sparsity across layers as the network depth increases, and this leads to a decrease in the number of complete activated pathways between the input layers and the output heads. In contrast, our model formulation alleviates the issue caused by the in-between sparse pathways with skip-connections and allows us to effectively adapt the network depth without worrying about the performance collapse caused by deep network structures.

Table 3: Test accuracies of the ablation study on CIFAR10-5 with Alexnet and fullyConv-7 network.

Methods	**Alexnet**	**fullyConv-7**
ada-st + map wt	77.71 ± 2.40	79.78 ± 0.98
ada-st + Bayes wt	84.48 ± 0.56	86.54 ± 0.79
ada-st + map wt + tsm	87.34 ± 0.74	86.55 ± 0.76
ada-st + Bayes wt + tsm	**91.44 ± 0.16**	**91.02 ± 0.29**

5.4. Ablation Study

We explore the effects of sequential Bayesian inference over weights (Bayes wt) and task-specific masks (tsm) on our network structure adaptation (ada-st) in detail by comparing with Maximum A Posteriori (map wt) estimation on weights. The results are summarized in Table 3, where the backbone networks are Alexnet and a fully convolutional network with 7 layers (fullyConv-7) with a comparable parameter count on CIFAR10-5. We observe that sequential Bayes on weight yields enhanced performance compared to the MAP estimation. The results also show that the task-specific masks improve the performance for both MAP estimation and sequential Bayes. With task-specific masks, our structure adaptation for Alexnet performs better than fullyConv-7, while the opposite results are observed in the cases without task-specific masks. Overall, our proposed approach, integrating sequential Bayesian inference on both weights and network structures with task-specific masks, outperforms the alternative configurations.

5.5. Unsupervised Continual Learning

For continual image generation, we evaluate our method on MNIST and not-MNIST against naive training, EWC, VCL, and IBPCL. We fix the latent dimension to 50 and the network width to 500. For EWC, we report the best result obtained with $\lambda = 10$. We set a truncation level of 2 to ensure a fair comparison with other methods. And, we calculate test data log-likelihoods using importance sampling with 5000 samples per data point.

The qualitative results depicting sequential image generation are illustrated in Figure 5a. It is evident that the naive method, without any retraining, experiences severe catastrophic forgetting across both datasets. Similarly, while EWC exhibits improved retention, it still demonstrates reduced performance for the initial tasks following training on subsequent tasks. In contrast, VCL, IBPCL, and our approach showcase notably enhanced generative outcomes.

For quantitative analysis, we report the test log-likelihoods shown in Figure 5b for the one-MNIST dataset. Here, we exclude the naive method due to its comparatively inferior performance. Our method outperforms EWC and VCL consistently across all tasks. Although IBPCL exhibits superior performance for the initial five tasks, our method surpasses it for the remaining tasks, except the eighth, suggesting that our model outperforms IBPCL for a longer chain of continual image generation tasks.

5.6. Case Study: Class-Incremental Learning

We showcase the application of our structure adaptation framework in a class-incremental learning scenario where the task ID is not provided in prediction. For this setting, we discard the task-specific masks and use a single-headed architecture. Therefore, we use the variational distribution without the weight importance parameter $\gamma_{m,n}^{(k)}$. We adapt the ER-ACE framework proposed in (Caccia et al., 2022). Specifically, for each new training data point, ER-ACE calculates the likelihood across the new task labels only, and for the replay data points, it computes the likelihood across the entire class labels. This strategy is successful in reducing abrupt changes in feature representations for previous tasks and helps mitigate forgetting in the class-incremental learning scenario.

We evaluate a single-headed 7-layered fully convolutional network (fullyConv-7) using the CIFAR10-5 dataset. We employ a memory replay of size 500 with reservoir sampling to sample examples for replay in future tasks. We conduct a comparison of our structure adaptation in cases involving both sequential Bayesian inference and Maximum A Posteriori for weights. Without Bayesian inference on weights and structure adaptation, our method reduces to the ER-ACE framework. Table 4 shows that by inferring the model structures given the dataset, our structure adaptation framework improves the model's performance for both weight MAP and Bayesian weight inference cases. We further observe that sequential Bayesian inference on weights enhances performance over the MAP estimation. Additionally, the combination of Bayesian weight inference with our structure adaptation leads to the best performance given the architecture and the coreset size.

Table 4: Mean Test accuracy ±1 std across tasks for class-incremental learning on CIFAR10-5 dataset with fullyConv-7 architecture with a coreset of size 500.

Methods	**Accuracy** (%)
ER-ACE	50.86 ± 2.44
ER-ACE + Bayes wt	53.16 ± 1.09
ER-ACE + ada-st	53.83 ± 1.34
ER-ACE + ada-st + Bayes wt (**Ours**)	$\mathbf{56.55 \pm 0.61}$

6. Limitations

The challenge with adding pooling layers is due to the stochastic dimensionality of the inferred layers. For the fully convolutional network, we insert the pooling layers only in the first three layers and perform depth inference over the subsequent convolutional layers. An alternative approach to pooling layers is to use strided convolutions and design the network in a way such that the feature map size (i.e., product of feature map height, width, and channels) is constant across convolutional layers.

Truncation over backbone networks for the variational distribution can be relaxed using Russian Roulette as pointed out in section 4.1.3.

7. Conclusion

We propose a novel non-parametric Bayesian inference framework by integrating network structural adaptation into sequential Bayes for continual learning. Specifically, we model the growth of network depth as a beta process and perform drop-connect regularization for network width with a conjugate Bernoulli process. We develop an efficient estimator to jointly infer the depth and adapt the width continually. We demonstrate the versatility of our framework on different types of backbone networks in both supervised and unsupervised continual learning. The results show that it can retain performance levels that are either superior or on par with the state-of-the-art methods across various benchmark datasets. We also demonstrate that our proposed framework can be readily extended to class-incremental learning scenarios.

Acknowledgements

This work is supported by the National Science Foundation under NSF Award No.2045804. We acknowledge the Research Computing at Rochester Institute of Technology (RIT, 2019) for the computational resources.

Impact Statement

Our method is a non-parametric Bayesian framework that allows for network structure adaptation in a continual learning setting. The proposed method enhances performance in continual learning across diverse settings, making it applicable in a wide variety of real-world applications. Primarily, such methods find application in situations with limited resource constraints and privacy-critical applications, such as healthcare and medical data. Additionally, we do not foresee any potential harmful societal consequences of this work.

References

Acharya, M., Hayes, T. L., and Kanan, C. Rodeo: Replay for online object detection. In *The British Machine Vision Conference*, 2020.

Adel, T., Zhao, H., and Turner, R. E. Continual learning with adaptive weights (CLAW). In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=Hklso24Kwr.

Ahn, H., Cha, S., Lee, D., and Moon, T. Uncertainty-based continual learning with adaptive regularization. In *Advances in Neural Information Processing Systems*, pp. 4394–4404, 2019.

Antorán, J., Allingham, J. U., and Hernández-Lobato, J. M. Depth uncertainty in neural networks, 2020.

Babakniya, S., Fabian, Z., He, C., Soltanolkotabi, M., and Avestimehr, S. A data-free approach to mitigate catastrophic forgetting in federated class incremental learning for vision tasks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=3b9sqxCW1x.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural network. In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1613–1622, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/blundell15.html.

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 *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Buzzega, P., Boschini, M., Porrello, A., Abati, D., and CALDERARA, S. Dark experience for general continual learning: a strong, simple baseline. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 15920–15930. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/b704ea2c39778f07c617f6b7ce480e9e-Paper.pdf.

Caccia, L., Aljundi, R., Asadi, N., Tuytelaars, T., Pineau, J., and Belilovsky, E. New insights on reducing abrupt representation change in online continual learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=N8MaByOzUfb.

Cortes, C., Gonzalvo, X., Kuznetsov, V., Mohri, M., and Yang, S. AdaNet: Adaptive structural learning of artificial neural networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 874–883. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/cortes17a.html.

Ebrahimi, S., Elhoseiny, M., Darrell, T., and Rohrbach, M. Uncertainty-guided continual learning with bayesian neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HklUCCVKDB.

French, R. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3:128–135, 05 1999. doi: 10.1016/S1364-6613(99)01294-2.

Graves, A. Practical variational inference for neural networks. In *Neural Information Processing Systems*, 2011. URL https://api.semanticscholar.org/CorpusID:14885866.

Griffiths, T. L. and Ghahramani, Z. The indian buffet process: An introduction and review. *J. Mach. Learn. Res.*, 12(null):1185–1224, jul 2011. ISSN 1532-4435.

Hayes, T. L., Kafle, K., Shrestha, R., Acharya, M., and Kanan, C. Remind your neural network to prevent catastrophic forgetting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015. URL https://api.semanticscholar.org/CorpusID:206594692.

Hinton, G. E. and van Camp, D. Keeping the neural networks simple by minimizing the description length of the weights. In *Annual Conference Computational Learning Theory*, 1993. URL https://api.semanticscholar.org/CorpusID:9346534.

Hjort, N. L. Nonparametric bayes estimators based on beta processes in models for life history data. *Annals of Statistics*, 18(3):1259–1294, 1990.

Hoffman, M. and Blei, D. Stochastic Structured Variational Inference. In Lebanon, G. and Vishwanathan, S. V. N. (eds.), *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pp. 361–369, San Diego, California, USA, 09–12 May 2015. PMLR. URL https://proceedings.mlr.press/v38/hoffman15.html.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

Jankowiak, M. and Obermeyer, F. Pathwise derivatives beyond the reparameterization trick. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 2240–2249, 2018. URL http://proceedings.mlr.press/v80/jankowiak18a.html.

KC, K., Li, R., and Gilany, M. Joint inference for neural network depth and dropout regularization. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=S5LLQZ-yUP.

Kessler, S., Nguyen, V., Zohren, S., and Roberts, S. J. Hierarchical indian buffet neural networks for bayesian continual learning. In de Campos, C. and Maathuis, M. H. (eds.), *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pp. 749–759. PMLR, 27–30 Jul 2021. URL https://proceedings.mlr.press/v161/kessler21a.html.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, 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. *Proceedings of the National Academy of Sciences*, 114, 12 2016. doi: 10.1073/pnas.1611835114.

Konishi, T., Kurokawa, M., Ono, C., Ke, Z., Kim, G., and Liu, B. Parameter-Level Soft-Masking for Continual Learning. In *Proc. of ICML*, 2023.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K. (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Kumar, A., Chatterjee, S., and Rai, P. Bayesian structural adaptation for continual learning. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5850–5860. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/kumar21a.html.

MacKay, D. J. *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology, 1992a.

MacKay, D. J. C. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3): 448–472, 05 1992b. ISSN 0899-7667. doi: 10.1162/neco.1992.4.3.448. URL https://doi.org/10.1162/neco.1992.4.3.448.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=S1jE5L5gl.

Nazaret, A. and Blei, D. Variational inference for infinitely deep neural networks. In *International Conference on Machine Learning*, 2022.

Neal, R. M. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. Variational continual learning. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BkQqq0gRb.

Paisley, J., Zaas, A., Woods, C. W., Ginsburg, G. S., and Carin, L. A stick-breaking construction of the beta process. In *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, pp. 847–854, 2010.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2019.01.012. URL https://www.sciencedirect.com/science/article/pii/S0893608019300231.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017. URL https://api.semanticscholar.org/CorpusID:40027675.

Regmi, P. and Li, R. Adavae: Bayesian structural adaptation for variational autoencoders. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 14737–14749. Curran Associates, Inc., 2023.

RIT. Research computing services, 2019. URL https://www.rit.edu/researchcomputing/.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T. P., and Wayne, G. Experience replay for continual learning, 2019. URL https://openreview.net/forum?id=S1g2V3Cct7.

Rudner, T. G. J., Smith, F. B., Feng, Q., Teh, Y. W., and Gal, Y. Continual Learning via Sequential Function-Space Variational Inference. In *Proceedings of the 39th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2022.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *CoRR*, abs/1606.04671, 2016. URL http://arxiv.org/abs/1606.04671.

Serrà, J., Surís, D., Miron, M., and Karatzoglou, A. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, 2018. URL https://api.semanticscholar.org/CorpusID:2157345.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual learning with deep generative replay. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 2994–3003, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Thibaux, R. and Jordan, M. I. Hierarchical beta processes and the indian buffet process. In *Artificial Intelligence and Statistics*, pp. 564–571, 2007.

van de Ven, G. M. and Tolias, A. S. Three scenarios for continual learning, 2019.

Xu, K., Srivastava, A., and Sutton, C. Variational russian roulette for deep Bayesian nonparametrics. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6963–6972. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/xu19e.html.

Yan, S., Xie, J., and He, X. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Ye, F. and Bors, A. G. Continual variational autoencoder via continual generative knowledge distillation. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. ISBN 978-1-57735-880-0. doi: 10.1609/aaai.v37i9.26294. URL https://doi.org/10.1609/aaai.v37i9.26294.

Yoon, J., Yang, E., Lee, J., and Hwang, S. J. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Sk7KsfW0-.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 3987–3995. JMLR.org, 2017.

Zhai, M., Chen, L., Tung, F., He, J., Nawhal, M., and Mori, G. Lifelong gan: Continual learning for conditional image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

A. Variational Inference Framework

The Evidence Lower Bound for a mini-batch of size B of task t can be written as:

$$\mathcal{L}(t) = \frac{N_t}{B}\sum_{i=1}^{N_t} \mathbb{E}_{q_t(\mathbf{v},\mathcal{Z},\mathcal{W})}\left[\log p(\mathbf{x}_{t,i}|\mathbf{v},\mathcal{Z},\mathcal{W}))\right] - \mathbb{KL}\left[q_t(\mathbf{v})||q_{t-1}(\mathbf{v})\right] - \mathbb{KL}\left[q_t(\mathcal{Z}|\mathbf{v})||q_{t-1}(\mathcal{Z}|\mathbf{v})\right] \\ - \mathbb{KL}\left[q_t(\mathcal{W})||q_{t-1}(\mathcal{W})\right] \tag{8}$$

We approximate the log-likelihood term using Monte Carlo estimation. Using $\theta = (\mathbf{v},\mathcal{Z},\mathcal{W})$, we approximate the log-likelihood as:

$$\mathbb{E}_{q_t(\theta)}\left[\log p(\mathbf{x}_{t,i}|\theta)\right] \approx \frac{1}{S}\sum_s \log p\left(\mathbf{x}_{t,i}|\theta^{(s)}\right), \\ \text{where, } \theta^{(s)} \sim q_t(\theta) \tag{9}$$

.

We discuss the KL divergence calculation for each KL term and the sampling mechanism for the corresponding distributions in the subsequent sections.

A.1. Gaussian Weight Distribution Reparameterization

We use the reparameterization trick to sample from the variational weight distribution $w^{(k)}_{m,n} \sim \mathcal{N}\left(\mu^{(k)}_{m,n}, \left(\sigma^{(k)}_{m,n}\right)^2\right)$.

$$w^{(k)}_{m,n} = \mu^{(k)}_{m,n} + \sigma^{(k)}_{m,n} \times \epsilon, \qquad \epsilon \sim \mathcal{N}(0,1) \tag{10}$$

Let $\phi = \left(\mu^{(k)}_{m,n}, \sigma^{(k)}_{m,n}\right)$ represents the variational parameters for weight $w^{(k)}_{m,n}$. The above trick moves the stochasticity from the weight to the random variable ϵ and so allows the gradient calculation for any differentiable function $g\left(w^{(k)}_{m,n}\right)$ w.r.t ϕ.

$$\nabla_\phi \, \mathbb{E}_{q\left(w^{(k)}_{m,n}\right)}\left[g\left(w^{(k)}_{m,n}\right)\right] = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)}\left[\nabla_\phi g\left(w^{(k)}_{m,n}\right)\right]\Big|_{w^{(k)}_{m,n}=\mu^{(k)}_{m,n}+\sigma^{(k)}_{m,n}\times\epsilon} \tag{11}$$

The KL for the Gaussian distribution has a closed-form solution (Kingma & Welling, 2013).

A.2. Relaxation of Bernoulli Distribution

During training, we relax the discrete constraint for $\mathcal{Z}$ by using a concrete Bernoulli (Maddison et al., 2017; Jang et al., 2017).

$$\log \text{ConBer}\left(z^{(k)}_{l,m}|\pi_k,\tau\right) = \log\tau - \tau\text{logit}\left(z^{(k)}_{l,m}\right) + \text{logit}\left(\pi^{(k)}_{l,m}\right) \\ -2\log\left(1+\exp\left(-\tau\text{logit}\left(z^{(k)}_{l,m}\right) + \text{logit}\left(\pi^{(k)}_{l,m}\right)\right)\right) \tag{12}$$

where, τ is temperature and $\text{logit}(x) = \log\frac{x}{1-x}$ is a logit function. The temperature controls the smoothness of the distribution and we recover the discrete Bernoulli as $\tau \to 0$. We generate random samples $z^{(k)}_{l,m}$ from the distribution by first sampling ϵ from a uniform distribution and passing through the function as:

$$z^{(k)}_{l,m} = \text{sigmoid}\big(\tau^{-1}\left(\text{logit}(\pi_k) + \text{logit}(\epsilon)\right)\big), \\ \epsilon \sim \text{Uniform}(0,1) \tag{13}$$

This relaxation of Bernoulli allows backpropagation of the ELBO while sampling from the distribution. The KL on Concrete Bernoulli requires Monte Carlo estimation by sampling from the variational distribution.

A.3. Implicit Reparameterization for Beta Distribution

Unlike Gaussian, an easy explicit reparameterization is not possible for beta distribution. Hence, we rely on implicit reparameterization instead.

Using the beta CDF $F_\phi(v) = \epsilon \sim \text{Uniform}(0, 1)$ as the transformation function, we apply implicit differentiation to avoid F^{-1}. Here, $\phi = (\alpha, \beta)$ represents the beta distribution parameters. Applying the total derivative on both sides leads to the gradient of v w.r.t. ϕ.

$$\nabla_\phi v = -\left(\nabla_v F_\phi(v)\right)^{-1} \nabla_\phi F_\phi(v) \tag{14}$$

This definition of $\nabla_\phi v$ allows the gradient computation of any differentiable function $g(v)$ w.r.t. the beta parameters ϕ. Especially, in the context of our lower bound 8, we are interested in a gradient of expectation of $g(v)$ under the distribution $q_\phi(v)$.

$$\nabla_\phi \mathbb{E}_{q_\phi(v)}[g(v)] = \mathbb{E}_{q_\phi(v}[\nabla_v g(v) \nabla_\phi v]\big|_{v=F_\phi^{-1}(\epsilon)} \tag{15}$$

While $\nabla_v F_\phi(v) = q_\phi(v)$, $\nabla_\phi F_\phi(v)$ does not have an analytic solution. Hence, we resort to numerical approximation for its computation (Jankowiak & Obermeyer, 2018). Sampling from beta distribution can be done with rejection sampling or by sampling from gamma distributions. The implementation for the implicit reparameterization and sampling for beta distribution is available in PyTorch (Paszke et al., 2017).

Likewise, the KL on beta distributions for $\mathbf{v}$ can be obtained as:

$$\begin{aligned}\mathbb{KL}\left[q(\mathbf{v})||p(\mathbf{v})\right] = \sum_k \log \frac{B(\alpha_0, \beta_0)}{B(a_k, b_k)} &+ (a_k - \alpha_0)\Psi(a_k) + (b_k - \beta_0)\Psi(b_k) \\ &+ (a_k - \alpha_0 + b_k - \beta_0)\,\Psi(a_k + b_k)\end{aligned} \tag{16}$$

where, B and Ψ represent beta and di-gamma functions respectively.

B. Depth Inference and Weight Posterior Redefinition

The stick-breaking construction of the beta process leads to an exponential increase in sparsity in the masks $\mathbf{Z}^{(k)}$. Consequently, we define the final activated layer L as the final layer with at least one activated weight. Considering our use of relaxed Bernoulli during model training, we introduce a threshold ϵ to compute the depth.

$$L = \max_k \left\{ k \middle| \sum_m \sum_n z^{(k)}_{m,n} > \epsilon \right\} \tag{17}$$

$\sum_l \sum_m z^{(k)}_{l,m}$ signifies the pseudo-count of activated weights during training in layer k, while serving as the true count of activated weights during testing, as we use Bernoulli distribution for predictions.

For task t, when $L \leq K$ layers are activated, the model's predictions depend only on the first L layers and not the succeeding layers. Thus the exact posterior incorporates information solely from the data for $\mathbf{W}^{(k)}$ up to $k \leq L$. As a result, the posterior of the $\mathbf{W}^{(k)}$ with $k > L$ must match the prior. We incorporate this into our variational approximation as:

$$q_t(\mathcal{W}; L) = \prod_{k=1}^{L} q_t\left(\mathbf{W}^{(k)}\right) \prod_{k=L+1}^{K} p_t\left(\mathbf{W}^{(k)}\right) \tag{18}$$

This redefinition of the posterior approximation allows the propagation of small initialization values for the variances of variational weight distribution to subsequent tasks in inactive layers. Without this adaptation, the variances in the inactive layers tend to increase, potentially hindering the training of newly activated layers for future tasks.

C. Experimental Setup

C.1. Datasets

For supervised continual learning using fully connected neural networks, three datasets are used: permuted MNIST, split MNIST, and split fashion MNIST. Permuted MNIST involves a 10-class classification problem where pixels in all task images are shuffled based on a fixed permutation. In a similar vein, split MNIST comprises five binary classification tasks presented sequentially: 0/1, 2/3, ..., to 8/9. Furthermore, split fashion MNIST involves five binary classification tasks related to clothing.

To explore convolutional neural networks, experiments are conducted with split CIFAR10-5, split CIFAR100-10, split CIFAR100-20, and split TinyImagenet-10. Split CIFAR10-5 includes a sequence of five tasks from CIFAR10, while split CIFAR100-n involves n tasks from CIFAR100. Similarly, split TinyImagenet-10 comprises ten tasks from the TinyImagenet dataset.

For unsupervised continual learning, we use two datasets: one-MNIST and not-MNIST for evaluation. Both datasets consist of 10 tasks. The one-MNIST dataset consists of sequential images for digits 0, 1, ... to 9 from the MNIST dataset and the not-MNIST dataset consists of sequential images for alphabets A, B, ..., to J.

C.2. Hyper-parameter Settings for Fully Connected Experiments

We use fully connected neural networks with a fixed width of 200 and maintain a truncation level $K = 3$. We apply LeakyReLU activation with a negative slope of 0.01. We use a batch size of 512 for all experiments, and we estimate the log-likelihood using 10 samples and 100 samples for the test. We train using the Adam optimizer its default settings $\beta_1 = 0.9, \beta_2 = 0.999$ and we re-initialize the optimizer for each task. Additionally, we found that multiplying the KL terms by a factor of 0.1 enhances model training. The minimum pseudo-count ϵ for activated weights to identify the activated layer is taken to be 0.001. For each Bayesian layer's weight, we place a standard normal $\mathcal{N}(0, 1)$ prior. And, for the first task, we initialize the standard deviations of all variational weight distributions with a small value of 0.0001. For the subsequent tasks, we employ sequential Bayes by copying the parameters of the previous task's posterior to the next task's prior. For task-incremental learning with weight importance parameter γ, we reinitialize $\gamma = 0.5$ for each new task to ensure the masks are tailored to each specific task.

We ran all the experiments five times and reported the mean and standard deviation of the final average accuracy across tasks.

C.2.1. Permuted MNIST

We use $\alpha = 75$ and $\beta = 300$ as beta priors for the permuted MNIST experiment. Additionally, we initialize the variational structure parameters as $a_k = \alpha$ and $b_k = \beta$ for the first task. The temperature values for both the prior and posterior concrete Bernoulli are kept constant at 0.3. We employ two learning rates: one for weights and another for structure. The structure learning rate is kept constant at 0.03. The weight learning rate is initialized at 0.02 and reduced by 0.85 with each new task. We train for 25 epochs for the permuted MNIST. After sampling the task-specific masks, we train for an additional 15 epochs by reducing the weight learning rate for the task by a factor of 0.1. We apply the KL mask following (Kumar et al., 2021) only to the activated weights, and we freeze the structure parameters during this step.

C.2.2. Split MNIST and Split Fashion MNIST

For task-specific mask experiments in split MNIST and split fashion MNIST, we initialize the structure priors as $\alpha = 200$ and $\beta = 300$. , for both datasets, we use a fixed structure learning rate of 0.1 and initialize the weight learning rate as 0.008 which is reduced by 0.85 for every new task. We use temperature parameter $\tau = 0.1$ for both prior and posterior concrete Bernoulli distributions. We train for 15 epochs for both datasets and an additional 10 for finetuning after sampling task-specific masks. While sampling the task-specific mask, we consider the layers with weight activation below 2.5% as unactivated, ensuring sparsely activated deeper layers are trimmed off. Other hyper-parameters follow permuted MNIST's experimental settings.

C.3. Hyper-parameter Settings for CNN Experiments

We provide the details regarding hyper-parameters for our experiments with task-specific masks in the following section. Subsequently, we elaborate on the two model architecture settings employed in the CNN experiments.

C.3.1. Split CIFAR10-5, Split CIFAR100-n and Split TinyImagenet-10

We use a batch size of 256 and set temperature $\tau = 0.1$ for all CNN experiments. We use 2 Monte Carlo samples during training and 10 samples for predictions. Following (Kumar et al., 2021), a single mask is employed per channel for each convolutional layer. We remove task-specific normalization used by (Kumar et al., 2021) for fair comparison. And, following (Konishi et al., 2023), we use 32×32 images for training and testing. In addition to hyper-parameter search, we utilize the validation set to determine the optimal model weights during training. And, we set the prior for the beta process as $\alpha = 300$ and $\beta = 200$, ensuring activation of at least the first three layers. A larger α compared to β ensures that the model activates deeper layers, as opposed to the reverse setting. Moreover, using larger values for the hyper-parameters of the beta process ensures a peakier distribution, reducing variance for the beta process.

We use a learning rate of 0.003 for the weights in all CIFAR experiments and 0.001 for TinyImagenet-10. Additionally, we reduce the weight learning rate by a factor of 0.85 for every new task. For the structure, we maintain a constant learning rate of 0.2 for CIFAR10-5, CIFAR100-10, and TinyImagenet-10. And, for CIFAR100-20, we find that a learning rate of 0.5 works best for the structure. During shared training, we run 25 epochs, and for selective fine-tuning of the task-specific mask, we use 15 epochs for Alexnet experiments. And, we reinitialize $\gamma = 0.1$ every new task whenever used. All remaining hyperparameters, except for the model architecture, are consistent with those used in the split MNIST experiments.

For fully convolutional networks, the hyperparameters are similar to those for the AlexNet architecture. Only the differences are mentioned here. We train for 30 epochs and down-weight the KL term for the Bernoulli distribution by 1e-7, which provides better performance while maintaining stable inferred depth for reasonably large truncation. Additionally, we found structure learning rates of 0.3 and 0.45 to work best for the CIFAR100-10 and CIFAR100-20 datasets, respectively, with fully convolutional networks. After training on each task t, we directly copy each layer k's structure posterior parameters a_k^t as prior α_k^{t+1} for the next task but adjust b_k using the ratio of activated weights $\{\tilde{\pi}_k\}$ in the task t's task-specific mask. We approximate the beta sample $\tilde{v}_k = \frac{\tilde{\pi}_k}{\tilde{\pi}_{k-1}}$ and adjust $\tilde{b}_k^t = \frac{a_k(1-\tilde{v}_k)}{\tilde{v}_k}$. Finally, for L activated layers, we update the priors for the next task as

$$\beta_k^{t+1} \leftarrow \begin{cases} \tilde{b}_k^t & \text{if } 1 < k \leq L \\ b_k^t & \text{otherwise.} \end{cases}$$

C.3.2. Model Architecture

We use two different setups for CNNs - Alexnet and fully convolutional architecture. The second one allows depth inference for convolutional layers using our formulated beta-Bernoulli processes.

Alexnet Backbone: Following the popular Alexnet backbone used in (Serrà et al., 2018; Kumar et al., 2021; Konishi et al., 2023), we employ three convolutional layers with channel widths of 128, 256, and 512, followed by two fully connected layers with hidden widths of 2048. We do not use any task-specific batch normalization layers as specified in (Kumar et al., 2021). Max pooling layers are utilized after each convolutional layer. Due to spatial size reduction from max pooling and channel expansion with each convolutional layer, we avoid skipping connections in convolutional layers. Nevertheless, we apply skip connections in the fully connected layers while maintaining the same model width. The beta-Bernoulli process is then applied over the entire network depth. Consequently, we can interpret the network as having a truncation $K = 5$. The goal of the experiments involving the Alexnet architecture is to demonstrate our framework's ability to showcase dynamic expansion in a continual learning framework.

Fully Convolutional Backbone: The Alexnet architecture comprises fully connected layers after convolutional layers before connecting to the task-specific classification heads. Consequently, leveraging depth inference for convolutional layers becomes challenging. To address this, we design a fully convolutional backbone without any fully connected layers, specifically for a truncation level K with K convolutional layers. The model structure is depicted in figure 6. The first three convolutional layers have kernel sizes of 4×4, 3×3, and 2×2, and we apply max pooling after each of these three layers. However, we do not apply max pooling for the remaining $K - 3$ layers, each featuring a kernel size of 2×2. Additionally, padding is applied to ensure that the feature map sizes are the same in these layers. The last activated layer is connected to a flatten layer, followed by the fully connected classification heads. Finally, we adapt the network structure for the model by leveraging our beta-Bernoulli setup across the entire truncation. It is important to note that our focus here is not on researching the optimal model architecture for vision tasks. Instead, we aim to showcase our framework's ability to infer the network structure—both width and depth—for convolutional architectures.

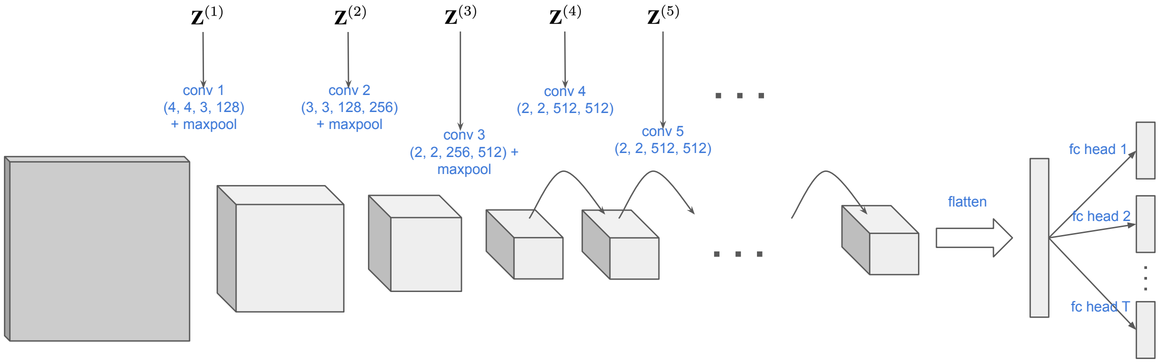


Figure 6: Application of our structure inference on the fully-convolutional network (fullyConv-K). The backbone consists of K convolutional layers and fully connected layers are present in task-specific heads only. The beta-Bernoulli process is applied across the K convolutional layers, where the stick-breaking construction is implemented with a truncation K. And, channel masks $\{\mathbf{Z}^{(k)}\}$, sampled from the conjugate Bernoulli processes, are applied on the convolutional layers.

C.4. Unsupervised Continual Learning

For all unsupervised CL experiments, we use the latent dimension of 50 and network width of 500. And, we estimate the log-likelihood using 10 training samples and use a batch size of 50. We use importance sampling with 5000 samples per data point to calculate the test data log-likelihoods. For VCL and IBPCL, we fix all other hyper-parameters at their best default settings, as specified in the respective papers. And, for EWC, we report the best result obtained with $\lambda = 10$. For all of our unsupervised continual learning experiments, we maintain a truncation level of 2 to ensure fair comparison with other methods. We set $\alpha = 30$ and $\beta = 20$ as the prior for the beta process. The temperature values for both prior and posterior distributions are kept constant at 0.1. And, we don't use any additional scale on the KL part. Similarly, we use the default settings of the Adam optimizer with a learning rate initialized at 0.0003 for both weight and structure parameters, and we reduce the weight learning rate by a factor of 0.85 for each new task. Additionally, following VCL, we reset the log standard deviation for each weight to -6 and copy means to the next task while switching to the next task. We don't use any task-specific masks in our unsupervised CL experiments.

C.5. Class Incremental Learning (CIL)

We conduct a comparison of our beta-Bernoulli processes in scenarios involving both weight distribution and Maximum A Posteriori (MAP) for weight. We employ a memory replay of size 500 with reservoir sampling and a batch size of 128. We found that $\alpha = 20$ and $\beta = 10$ yields the best results for structure inference. We evaluate with a single-headed 7-layered fully convolutional network (fullyConv-7). In all experiments, we employ the Adam optimizer with a constant learning rate of 0.001 for weights. Regarding structure inference, we use a learning rate of 0.5 for Bayesian weights and 0.001 for the deterministic case. Additionally, given the longer training iterations required for Bayesian networks, we trained them for 120 epochs, while we found 40 epochs to work best for MAP estimation. And, we use 0.2 dropout in cases without structure inference.

C.6. Hyper-parameter Tuning Setup

We leverage grid search for tuning the hyperparameters. The following shows the range of our search space.

1. $\alpha : \{1, 2, 3, 4, 5, 6, 10, 20, 50, 75, 100, 200, 300\}$
2. $\beta : \{1, 2, 3, 4, 5, 6, 10, 20, 50, 75, 100, 200, 300\}$
3. $\tau : \{0.1, 0.3, 0.75, 1, 5, 10\}$
4. weight learning rate: range-$[0.0001, 0.5]$
5. structure learning rate: range-$[0.001, 0.5]$
6. weight learning rate decay: $\{0.75, 0.85, 0.9, 1\}$

C.7. Hardware Specification

We trained and evaluated our models in NVIDIA A100 GPUs.

D. Further Results and Analysis

D.1. Comparison Against Function Space Variational Inference

Stochastic Function Space Variational Inference (SFSVI), as proposed by (Rudner et al., 2022), presents an alternative perspective on continual learning by employing function space regularization instead of weight regularization. Notably, it necessitates an additional dataset for inducing points to ensure functional regularization. SFSVI is computationally intensive, and the original configuration experiences out-of-memory issues, even when applied to CIFAR100-10 with the Alexnet backbone (refer to C.3.2). Consequently, our comparison is limited to CIFAR10-5 and CIFAR100-5. The comparative results are outlined in Table 5. To mitigate memory constraints, we reduce the memory footprint by scaling the convolutional channels of the Alexnet architecture by a factor of 0.25 for this experiment. Despite utilizing a coreset of size 200, SFSVI does not perform as well as our proposed method.

Table 5: Comparison against S-FSVI (Rudner et al., 2022) with Alexnet architecture (scaled by 0.25 factor).

Methods	**CIFAR10-5**	**CIFAR100-5**
SFSVI - 50 coreset	85.68 ± 0.88	49.90 ± 0.98
SFSVI - 200 coreset	88.14 ± 0.53	54.61 ± 1.13
Ours	**89.09** ± **0.20**	**57.37** ± **0.52**

D.2. Task-Incremental Learning on MNIST without Task-Specific Masks

We explore our method's ability to perform without task-specific masks in MNIST variants against fixed depth and width methods (EWC and VCL) and fixed depth with adaptive width method (HIBNN). Our experiments encompass both single-headed and multi-headed architectures for permuted MNIST and multi-headed cases for split MNIST. In the multi-headed experiment, task identity is required during predictions, as in task-incremental learning (van de Ven & Tolias, 2019). Conversely, the single-headed experiments are structured to eliminate the requirement for explicit task identity during prediction. The single-headed permuted MNIST experiment is an example of a domain-incremental learning scenario.

For permuted MNIST, our model outperforms all other methods for both multi-headed and single-headed cases. Also, our method performs better than competing methods in split MNIST. Accompanied by weight and structure regularization, our drop-connect regularization effectively prevents overfitting on the training data, enabling the model to generalize well across different tasks, unlike other methods.

Table 6: Evaluation of our model without task-specific mask against EWC, VCL, and HIBNN for single-headed (SH) and multi-headed (MH) of permuted MNIST and multi-headed split MNIST datasets.

Methods	**permuted MNIST MH**	**permuted MNIST SH**	**split MNIST MH**
EWC (Kirkpatrick et al., 2016)	96.86 ± 0.02	95.84 ± 0.07	99.01 ± 0.06
VCL (Nguyen et al., 2018)	97.11 ± 0.06	96.30 ± 0.16	98.51 ± 0.07
HIBNN (Kessler et al., 2021)	97.00 ± 0.30	96.83 ± 0.09	98.71 ± 0.40
Ours (w/o task-specific mask)	**97.59** ± **0.06**	**96.87** ± **0.08**	**99.13** ± **0.04**

D.3. Model Structure Learning in Fully-Connected Network

D.3.1. Model Performance and Inferred Network Structure across Max Width (M)

To analyze the effect of network maximum width M on performance and inferred structure, we experiment on split MNIST and split Fashion MNIST datasets with different maximum widths. We set the truncation level $K = 10$. We consider a layer activated only if the activated weight percentage for the layer is more than 2.5% while sampling the task-specific mask. The results in table 7 show that as long as the maximum width is reasonably large $M > 64$, it doesn't affect the performance.

For thinner networks, we observe higher inferred depths with a higher activated weight percentage. For moderately thin networks ($M = 64$), we observe an inferred depth of either 3 or 4 depending on the sampled network structure. As we increase M, the inferred depth decreases to 3 and stabilizes with decreasing weight activation.

Table 7: Model performance with different maximum widths (M). Our model's performance becomes stable with a reasonably large width. Also, fewer layers are activated with larger widths with decreasing activated weight percentage.

Max Width (M)	**Split MNIST**			**Split Fashion MNIST**		
	Test Acc. (%)	Depth	Activated Weight (%)	Test Acc. (%)	Depth	Activated Weight (%)
16	99.41 ± 0.04	4	85.71 ± 0.37	99.19 ± 0.06	4	85.87 ± 0.39
32	99.58 ± 0.05	4	75.93 ± 0.58	99.18 ± 0.03	4	75.31 ± 0.48
64	99.63 ± 0.06	3/4	59.84 ± 0.16	99.32 ± 0.02	3/4	59.23 ± 0.32
128	99.74 ± 0.01	3	43.02 ± 0.00	99.27 ± 0.03	3	43.02 ± 0.29
200	99.72 ± 0.08	3	33.87 ± 0.00	99.33 ± 0.06	3	33.91 ± 0.04
320	99.79 ± 0.02	3	26.06 ± 0.06	99.32 ± 0.03	3	26.13 ± 0.04

D.3.2. Model Performance and Inferred Network Structure across Truncation (K)

Additionally, we investigate the effect of truncation level K in model performance and inferred depth for 5 tasks in split fashion MNIST in table 8. We experiment with a fully connected network of width $M = 200$ with varying truncation levels. We observe that the model performance remains stable even after increasing the truncation K. The inferred depth is equal to 3 in all cases with $K \geq 3$ and $M = 200$, which is the reason why we chose truncation $K = 3$ in our experiments in table 1.

Table 8: Test accuracies and inferred depth on split Fashion MNIST for network width $M = 200$ across different truncation levels.

Truncation (K)	**Test Accuracy (%)**	**Inferred Depth**
3	99.33 ± 0.06	3
7	99.33 ± 0.08	3
10	99.32 ± 0.06	3

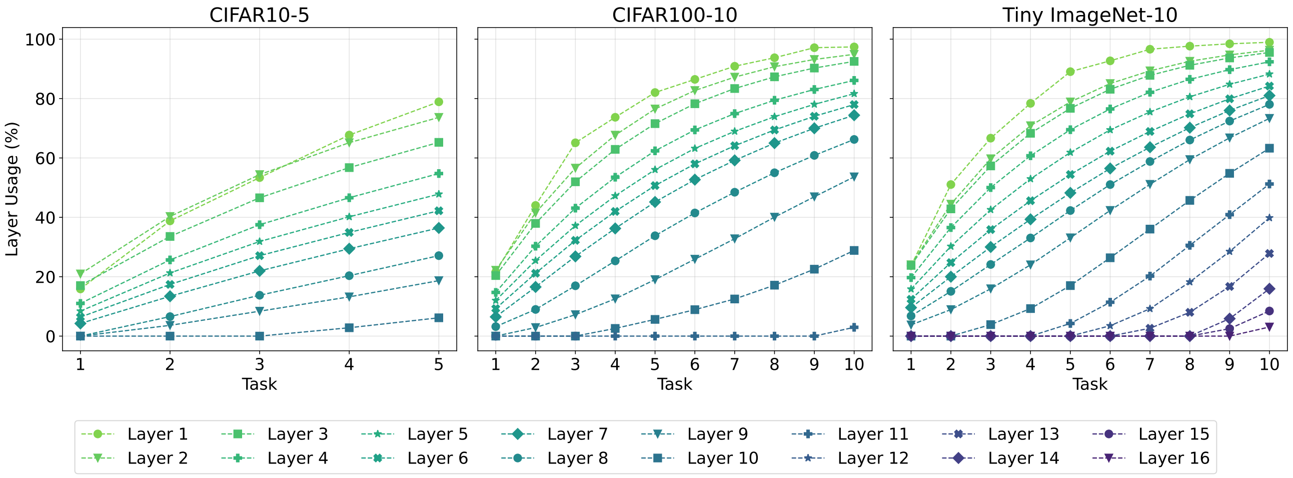


Figure 7: Cumulative model usage across tasks for a fully convolutional network with truncation $K = 24$ in CIFAR10-5, CIFAR100-10, and Tiny Imagenet-10.

D.4. Model Structure Learning in Fully Convolutional Network

Figure 7 illustrates the percentage of cumulative model usage per layer across tasks for a fully convolutional architecture with truncation $K = 24$ on the CIFAR10-5, CIFAR100-10 and Tiny Imagenet-10 datasets. We visualize only the activated

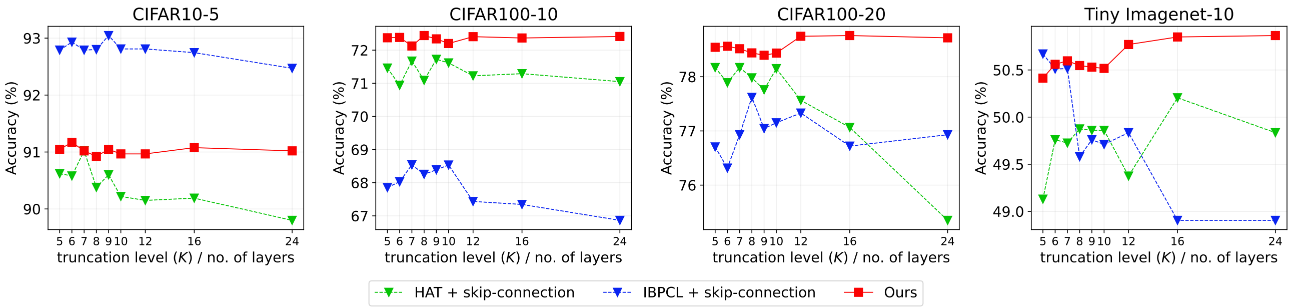


Figure 8: Performance comparison with HAT and IBPCL after adding skip-connections.

layers for each dataset. As indicated by table 9, we observe that the inferred depth matches the dataset complexity, with larger depths and more weights being inferred for more complex datasets. We observe a consistent increase in the activation of overall layer usage with each successive task.

Table 9: Depth Inference and Model Usage vs. Datasets with a fully convolutional network with truncation $K = 24$. We define a layer as activated if its weight activation is greater than 2.5% while sampling the task-specific mask.

Dataset	Inferred Depth	Cumulative Activated Weight (%)
CIFAR10-5	10	13.16%
CIFAR100-10	11	26.01%
Tiny Imagenet-10	16	35.89%
Super Imagenet-10 (Babakniya et al., 2023)	19	45.95%

Due to the larger data size, we train for longer epochs (40 for each task) and also, we set the KL coefficient for the Bernoulli distribution equal to 1e-8 for super Imagenet-10.

Similarly, Figure 9 demonstrates the Intersection over Union (IoU) between task-specific masks learned for different layers for the CIFAR100-10 dataset. We observe that task-specific masks in shallow layers have high overlap due to higher activations, while in deeper layers, the overlap is lower due to lesser activation. This is a direct outcome of the stick-breaking construction for the beta process, which induces exponentially decaying model activation with deeper layers. Additionally, it aligns with the observation that shallow layers in convolutional neural networks learn shareable low-level image features, while deeper layers learn more abstract feature representations.

D.5. Performance Stability Across Tasks

Although the mean accuracy across tasks reported in the main text captures the overall performance in continual learning, it doesn't specifically demonstrate the method's ability to retain knowledge of previous tasks without forgetting. We illustrate this qualitatively by explicitly plotting the test accuracies for the observed tasks over the training for each task. Figures 10 and 11 show the test accuracies for tiny Imagenet-10 and CIFAR variants respectively. Our method is able to sustain the test accuracies for the previous tasks while being able to learn new tasks incrementally.

D.6. Augmentation of skip-connection in HAT and IBPCL

We further investigate the HAT and IBPCL's performance by incorporating skip-connection in Figure 8. The results show that skip connections tend to improve their performance. However, our method still outperforms them for more complex datasets like CIFAR100-10/20 and tiny imagenet. Note that our method has a significant advantage for large truncation K in terms of computational cost, as the number of activated layers is fewer than the truncation due to decreasing layer activations over the layers K as shown in table 9. Hence, for each input prediction, only the activated layers need to be computed. In contrast, for HAT and IBPCL, the layer activations do not decrease over the network layers, which means deeper layers can be activated for large truncations too. Consequently, each forward pass must go through all the layers for these methods in the pre-specified network depth, unlike in our proposed method.

D.7. Time Complexity

Training our network with depth L and width M, the time complexity is $T = O(NBLM^2)$ with N training examples per epoch and B epochs. Let S denote the number of network structure samples, The time complexity of our method is linearly scalable as ST. With proper thresholding, the number of active layers L is relatively small in each sample.

The times taken for training the sequence of tasks with $L/K = 3$ for split fashion MNIST and $L/K = 10$ for split CIFAR10-5 are shown in the table below. The number of network structure samples $S = 10$ for split fashion MNIST and $S = 2$ for CIFAR10-5 for IBPCL and our method. The results are consistent with our above analysis.

Method	**split fashion MNIST** (sec)	**CIFAR10-5** (sec)
VCL	1194.82	7119.54
IBPCL	126.62	1841.66
Ours	512.84	2230.41

Table 10: Comparison of training times for Fashion MNIST and CIFAR10-5 for different methods.

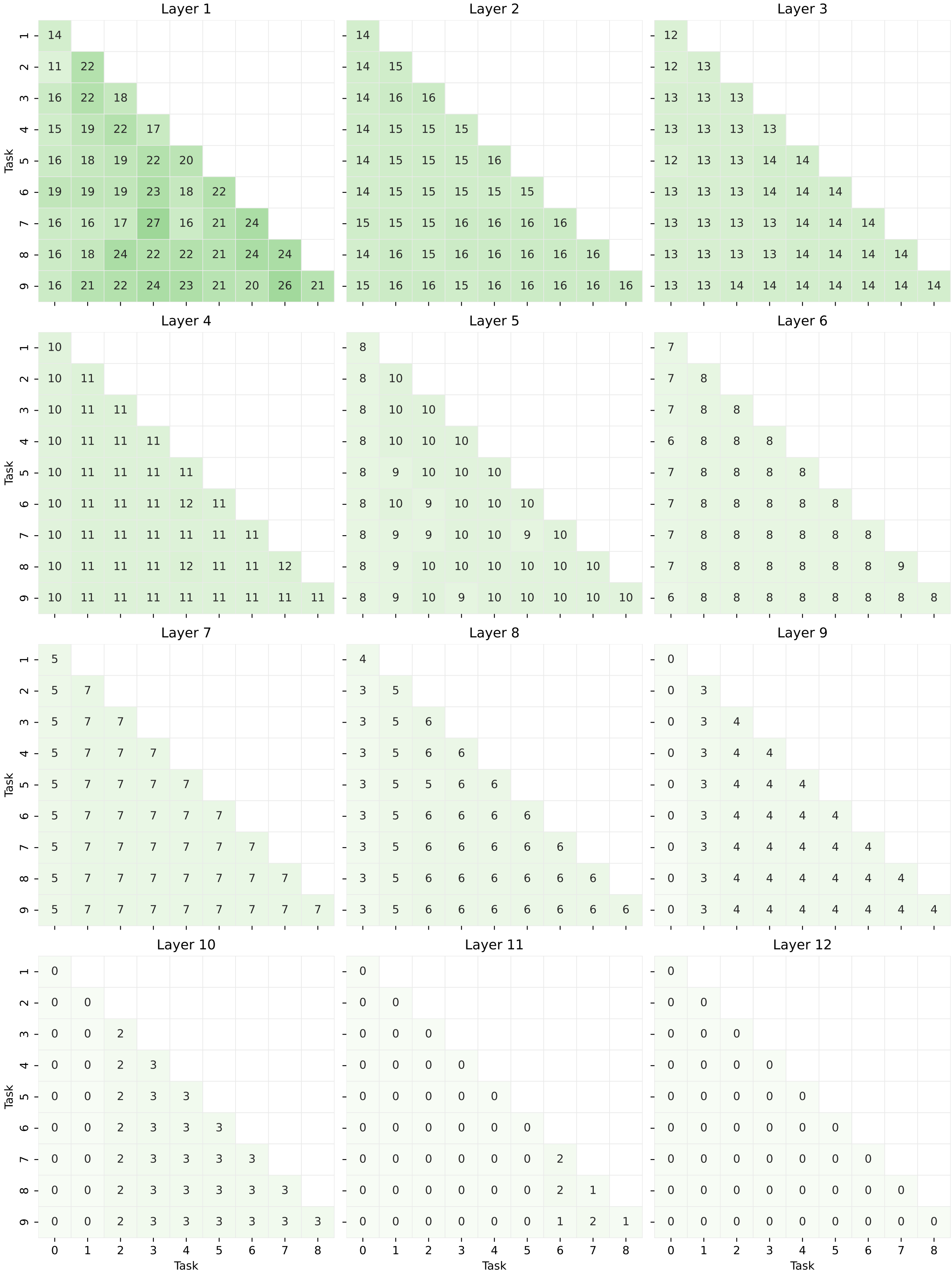


Figure 9: Intersection over Union (IoU) in percentage (rounded off to nearest integer) between task-specific masks in different layers for the CIFAR100-10 dataset.

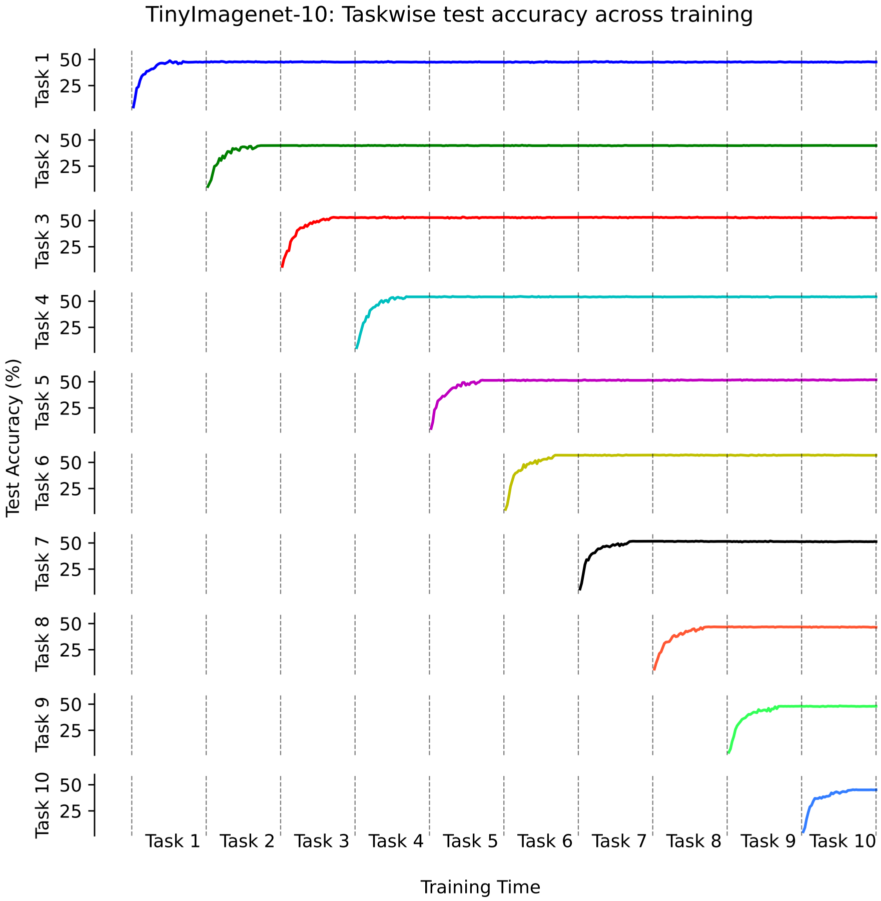
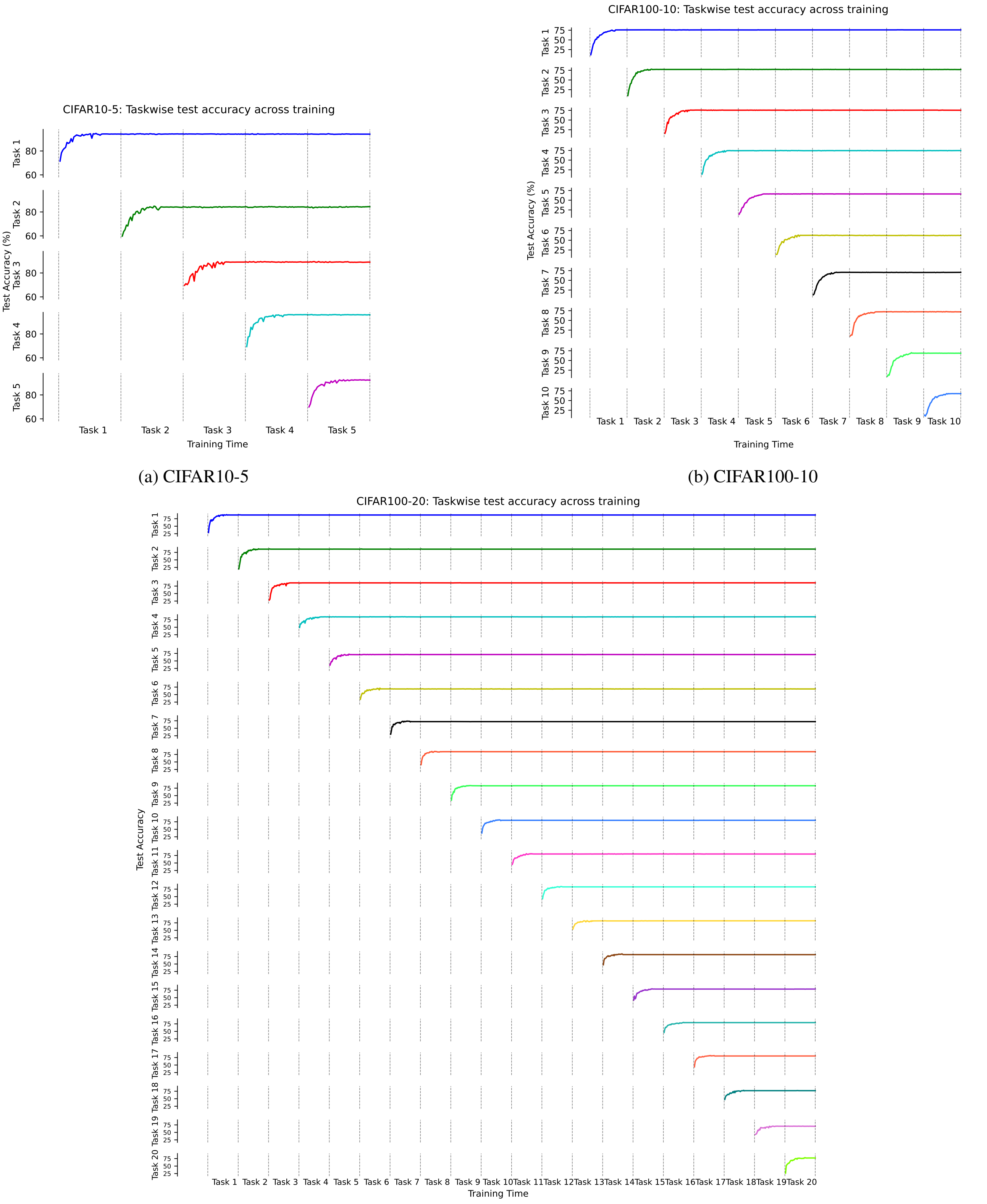


Figure 10: Our model's taskwise test accuracies across training in Tiny Imagenet-10 dataset- our model can preserve previously gained knowledge throughout the sequence of tasks.



(a) CIFAR10-5

(b) CIFAR100-10

(c) CIFAR100-20

Figure 11: Our model's taskwise test accuracies across training in CIFAR variants- our model can retain performance in seen tasks throughout the sequence of tasks.