Bayesian Structural Adaptation for Continual Learning

Abhishek Kumar [*1] **Sunabha Chatterjee** [*2] **Piyush Rai** [3]

Abstract

Continual Learning is a learning paradigm where learning systems are trained on a sequence of tasks. The goal here is to perform well on the current task without suffering from a performance drop on the previous tasks. Two notable directions among the recent advances in continual learning with neural networks are (1) variational Bayes based regularization by learning priors from previous tasks, and, (2) learning the structure of deep networks to adapt to new tasks. So far, these two approaches have been largely orthogonal. We present a novel Bayesian framework based on *continually* learning the structure of deep neural networks, to unify these distinct yet complementary approaches. The proposed framework learns the deep structure for each task by learning which weights to be used, and supports inter-task transfer through the overlapping of different sparse subsets of weights learned by different tasks. An appealing aspect of our proposed continual learning framework is that it is applicable to both discriminative (supervised) and generative (unsupervised) settings. Experimental results on supervised and unsupervised benchmarks demonstrate that our approach performs comparably or better than recent advances in continual learning.

1. Introduction

Continual learning (CL) (Ring, 1997; Parisi et al., 2019) is the learning paradigm where a single model is required to learn solving a sequence of tasks. At any point of time, the model is expected to (i) make predictions for the tasks it has seen so far, (ii) if subjected to training data for a new task, adapt to the new task, leveraging the past knowledge if possible (forward transfer), and benefit the previous tasks if possible (backward transfer). While the desirable aspects of more mainstream transfer learning (sharing of bias between related tasks (Pan & Yang, 2009)) might reasonably be expected here too, the principal challenge is to retain the predictive power for the older tasks even after learning new tasks, thus avoiding the so-called *catastrophic forgetting* (Kirkpatrick et al., 2017).

Real world applications in, for example, robotics or time-series forecasting, are rife with this challenging learning scenario, the ability to adapt to dynamically changing environments or evolving data distributions being essential in these domains. Continual learning is also desirable in unsupervised learning problems as well (Smith et al., 2019; Rao et al., 2019b) where the goal is to learn the underlying structure or latent representation of the data. Also, as a skill innate to humans (Flesch et al., 2018), it is naturally an interesting scientific problem to reproduce the same capability in artificial predictive modelling systems.

Existing approaches to continual learning are mainly based on three foundational ideas. One of them is to constrain the important parameters of previous tasks to not deviate significantly from their previously learned values by using some form of regularization or trade-off between previous and new learned weights (Schwarz et al., 2018; Kirkpatrick et al., 2017; Zenke et al., 2017; Lee et al., 2017). A natural way to accomplish this is to train a model using online Bayesian inference, whereby the posterior of the parameters learned from task t serves as the prior for task $t+1$ as in Nguyen et al. (2018) and Zeno et al. (2018). This new informed prior helps in the forward transfer, and also prevents catastrophic forgetting by penalizing large deviations from itself. In particular, VCL (Nguyen et al., 2018) achieves the state of the art results by applying this simple idea to Bayesian neural networks.

The second idea is to perform an incremental model selection for every new task. For neural networks, this is done by evolving the structure as newer tasks are encountered (Golkar et al., 2019; Li et al., 2019). To this end, structural learning is a natural scheme for continual learning as a new task may require a different network (sub)structure than previous (and possibly unrelated) tasks, and even if the tasks are highly related, their lower-layer representations can be very different. Another advantage of structural learning is that while retaining a shared set of parameters (which can be

[*]Equal contribution ; work done while at IIT Kanpur. [1]Microsoft, India [2]SAP Labs, India [3]Department of Computer Science, IIT Kanpur, India. Correspondence to: Piyush Rai <piyush@cse.iitk.ac.in>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

used to model task relationships), it also allow task-specific parameters that can increase the performance of the new task while avoiding catastrophic forgetting caused due to forced sharing of parameters.

The third idea is to invoke a form of 'replay', whereby selected (Lopez-Paz et al., 2017) or generated (Hu et al., 2019) samples representative of previous tasks, are used to retrain the model after new tasks are learned.

In this work, we introduce a novel Bayesian nonparametric approach to continual learning that seeks to incorporate the ability of structure learning into the simple yet effective framework of online Bayes. In particular, our approach models each hidden layer of the neural network using the Indian Buffet Process (Griffiths & Ghahramani, 2011) prior, which enables us to learn the network structure as new tasks arrive continually. We can leverage the fact that any particular task uses a sparse subset of the connections of a neural network, and different related tasks share different subsets (albeit possibly overlapping). Thus, in the setting of continual learning, it would be more effective if the network could accommodate changes in its connections dynamically to adapt to a newly arriving task. Moreover, in our model, we perform automatic model selection where each task can select the number of nodes in each hidden layer. All of this is done under the principled framework of variational Bayes and a nonparametric Bayesian modeling paradigm.

Another appealing aspect of our approach is that in contrast to some of the recent state-of-the-art continual learning models (Yoon et al., 2018; Li et al., 2019) that are specific to supervised learning problems, our approach applies to both deep discriminative networks (supervised learning) where each task can be modeled by a Bayesian neural network (Neal, 2012; Blundell et al., 2015), as well as deep generative networks (unsupervised learning) where each task can be modeled by a variational autoencoder (VAE) (Kingma & Welling, 2013).

2. Preliminaries

Bayesian neural networks (Neal, 2012) are discriminative models where the goal is to model the relationship between inputs and outputs via a deep neural network with parameters $\boldsymbol{w}$. The network parameters are assumed to have a prior $p(\boldsymbol{w})$ and the goal is to infer the posterior given the observed data $\mathcal{D}$. Exact posterior inference is intractable in such models and posterior approximation is needed. One such approximate inference scheme is Bayes-by-Backprop (Blundell et al., 2015) which uses a mean-field variational posterior $q(\boldsymbol{w})$ over the weights. Reparameterized samples from this posterior are then used to approximate the evidence lower bound via Monte Carlo sampling. Our goal in the continual learning setting is to learn such Bayesian neural networks for a sequence of tasks by inferring the posterior $q_t(\boldsymbol{w})$ for each task t, without forgetting the information contained in the posteriors of previous tasks.

Variational autoencoders (Kingma & Welling, 2013) are generative models where the goal is to model a set of inputs $\{\boldsymbol{x}\}_{n=1}^N$ via stochastic latent variables $\{\boldsymbol{z}\}_{n=1}^N$. The mapping from each $\boldsymbol{z}_n$ to $\boldsymbol{x}_n$ is defined by a generator/decoder model (modeled by a deep neural network with parameters θ) and the reverse mapping is defined by a recognition/encoder model (modeled by another deep neural network with parameters ϕ). Inference in VAEs is done by maximizing the variational lower bound on the marginal likelihood. It is customary to do point estimation for decoder parameters θ and encoder parameters ϕ. However, in the continual learning setting, it would be more desirable to infer the full posterior $q_t(\boldsymbol{w})$ for each task's encoder and decoder parameters $\boldsymbol{w} = \{\theta, \phi\}$, while not forgetting the information about the previous tasks as more and more tasks are observed. Our proposed continual learning framework address this aspect as well.

Variational Continual Learning (VCL) (Nguyen et al., 2018) is a recently proposed approach to continual learning that combats catastrophic forgetting in neural networks by modeling the network parameters $\boldsymbol{w}$ in a Bayesian fashion and by setting $p_t(\boldsymbol{w}) = q_{t-1}(\boldsymbol{w})$, that is, a task reuses the previous task's posterior as its prior. VCL solves the following KL divergence minimization problem for task t with data $\mathcal{D}_t$

$$q_t(\boldsymbol{w}) = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}\left(q(\boldsymbol{w}) || \frac{1}{Z_t} q_{t-1}(\boldsymbol{w}) p(\mathcal{D}_t|\boldsymbol{w})\right) \quad (1)$$

While offering a principled way that is applicable to both supervised (discriminative) and unsupervised (generative) learning settings, VCL assumes that the model structure is held fixed throughout, which can be limiting in continual learning where the number of tasks and their complexity is usually unknown beforehand. This necessitates adaptively inferring the model structure, which can potentially evolve with each incoming task. Another limitation of VCL is that the unsupervised version, based on performing CL on VAEs, only does so for the decoder model's parameters (shared by all tasks). It uses completely task-specific encoders and, consequently, is unable to transfer information across tasks in the encoder model. Our approach addresses both these limitations in a principled manner.

3. Bayesian Structure Adaptation for Continual Learning

In this section, we present a Bayesian model for continual learning that can potentially grow and adapt its structure as more and more tasks arrive. Our model extends seamlessly for unsupervised learning as well. For brevity of exposition,

in this section, we mainly focus on the supervised setting where a task has labeled data with known task identities t, i.e., the task-incremental setting (Van de Ven & Tolias, 2019). We then briefly discuss the unsupervised extension (based on VAEs) in Sec. 3.3 where task boundaries may or may not (task-agnostic) be available and provide further details in the Supplementary Material (Sec H).

Our approach uses a basic primitive that models each hidden layer using a nonparametric Bayesian prior (Fig. 1 shows an illustration and Fig. 2 shows a schematic diagram). We can use these hidden layers to model feedforward connections in Bayesian neural networks or in the encoder/decoder of a variational autoencoder (VAE).

For simplicity, consider a single hidden layer (Fig. 1). The first task activates as many hidden nodes as required and learns the posterior over the subset of edge weights incident on each active node. Each subsequent task reuses some of the edges learned by the previous tasks and uses the posterior over the weights learned by the previous task as the prior. Additionally, it may activate some new nodes and learn the posterior over some of their incident edges. It thus learns the posterior over a subset of weights that may overlap with weights learned by previous tasks. While making predictions, a task uses only the connections it has learned. More slack for later tasks in terms of model size (allowing it to create new nodes) indirectly lets the task learn better without deviating too much from the prior (in this case, posterior of the previous tasks) and further reduces chances of *catastrophic forgetting* (Kirkpatrick et al., 2017).

3.1. Generative Story.

Omitting the task id t for brevity, consider modeling t^{th} task using a neural network having L hidden layers. We model the weights in layer l as $\boldsymbol{W}^l = \boldsymbol{B}^l \odot \boldsymbol{V}^l$, a element-wise multiplication of a real-valued matrix $\boldsymbol{V}^l$ (with a Gaussian prior $\mathcal{N}(0, \sigma_0^2)$ on each entry) and a task-specific binary matrix $\boldsymbol{B}^l$. This ensures sparse connection weights between the layers. Moreover, we model $\boldsymbol{B}^l \sim \text{IBP}(\alpha)$ using the Indian Buffet Process (IBP) (Griffiths & Ghahramani, 2011) prior, where the hyperparameter α controls the number of nonzero columns in $\boldsymbol{B}^l$ and its sparsity. The IBP prior thus enables learning the size of $\boldsymbol{B}^l$ (and consequently of $\boldsymbol{V}^l$) from data. As a result, the number of nodes in the hidden layer is learned adaptively from data. The output layer weights are denoted as $\boldsymbol{W}_{out}$ with each weight having a Gaussian prior $\mathcal{N}(0, \sigma_0^2)$. The outputs are

$$\boldsymbol{y}_n \sim \text{Lik}(\boldsymbol{W}_{out}\boldsymbol{\phi}_{NN}(\boldsymbol{x}_n)), n = 1, \ldots, N \qquad (2)$$

Here $\boldsymbol{\phi}_{NN}$ is the function computed (using parameter samples) up to the last hidden layer of the network thus formed, and Lik denotes the likelihood model for the outputs.

Similar priors on the network weights have been used in other recent works to learn sparse deep neural networks (Panousis et al., 2019; Xu et al., 2019). However, these works assume a single task to be learned. In contrast, our focus here is to leverage such priors in the continual learning setting where we need to learn a sequence of tasks while avoiding the problem of catastrophic forgetting. Henceforth, we further suppress the superscript denoting layer number from the notation for simplicity; the discussion will hold identically for all hidden layers. When adapting to a new task, the posterior of $\boldsymbol{V}$ learned from previous tasks is used as the prior. A new $\boldsymbol{B}$ is learned afresh, to ensure that a task only learns the subset of weights that are relevant for it.

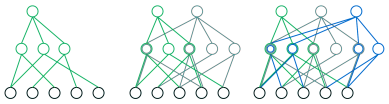


Figure 1. Illustration on single hidden layer

Stick Breaking Construction. As described before, to adaptively infer the number of nodes in each hidden layer, we use the IBP prior (Griffiths & Ghahramani, 2011), whose truncated stick-breaking process (Doshi et al., 2009) construction for each entry of B is as follows

$$\nu_k \sim \text{Beta}(\alpha, 1), \quad \pi_k = \prod_{i=1}^{k} \nu_i, \quad B_{d,k} \sim \text{Bernoulli}(\pi_k) \qquad (3)$$

for $d \in 1, ..., D$, where D denotes the number of input nodes for this hidden layer, and $k \in 1, 2, ..., K$, where K is the truncation level and α controls the effective value of K, i.e., the number of active hidden nodes. Note that the prior probability π_k of weights incident on hidden node k being nonzero decreases monotonically with k, until, say, K nodes, after which no further nodes have any incoming edges with nonzero weights from the previous layer, which amounts to them being turned off from the structure. Moreover, due to the cumulative product based construction of the π_k's, an implicit ordering is imposed on the nodes being used. This ordering is preserved across tasks, and allocation of nodes to a task follows this, facilitating reuse of weights.

The truncated stick-breaking approximation is a practically plausible and intuitive solution for continual learning since a fundamental tenet of continual learning is that the model complexity should not increase in an unbounded manner as more tasks are encountered. Suppose we fix a budget on the maximum allowed size of the network (number of hidden nodes in a layer) after it has seen, say, T tasks, which essentially corresponds to the truncation level for each layer. Then, for each task, nodes are allocated conservatively from this total budget, in a fixed order, conveniently controlled by the α hyperparameter. In (Sec. 3.4), we also discuss a dy-

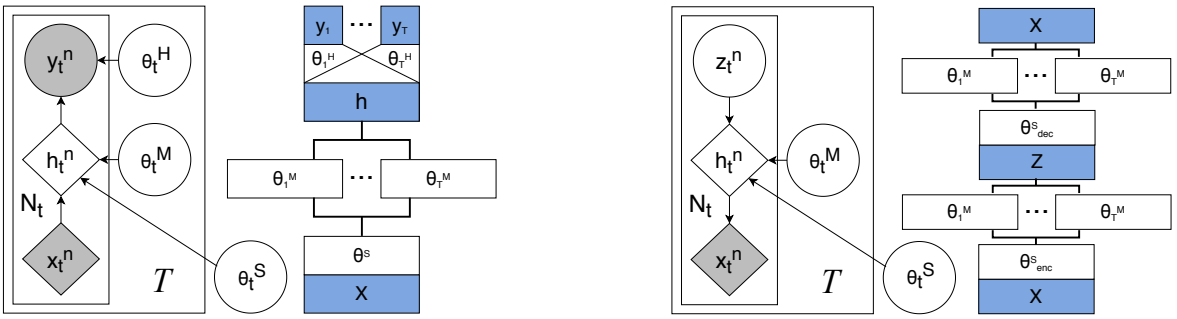


Figure 2. Schematics representing our models. In both (left) Discriminative model and (right) Generative model (VAE), θ_S are parameters shared across all task, θ^M are the task specific mask parameters, and θ^H are last layer separate head parameters. In our exposition, we collectively denote these parameters by $\boldsymbol{W} = \boldsymbol{B} \odot \boldsymbol{V}$ with the masks being $\boldsymbol{B}$ and other parameters being $\boldsymbol{V}$.

namic expansion scheme that avoids specifying a truncation level (and provide experimental results).

3.2. Inference

Exact inference is intractable in this model due to non-conjugacy. Therefore, we employ variational inference (Blei et al., 2017) to approximate the posterior. We use the structured mean-field approximation (Hoffman & Blei, 2015), which performs better than the standard mean-field approximation, as the former captures the dependencies in the approximate posterior distributions of $\boldsymbol{B}$ and $\boldsymbol{\nu}$. In particular, we use $q(\boldsymbol{V}, \boldsymbol{B}, \boldsymbol{v}) = q(\boldsymbol{V})q(\boldsymbol{B}|\boldsymbol{v})q(\boldsymbol{v})$, where, $q(\boldsymbol{V}) = \prod_{d=1}^{D}\prod_{k=1}^{K}\mathcal{N}(V_{d,k}|\mu_{d,k}, \sigma_{d,k}^2)$ is mean field Gaussian approximation for network weights. Corresponding to the Beta-Bernoulli hierarchy of (3), we use the conditionally factorized variational posterior family, that is, $q(\boldsymbol{B}|v) = \prod_{d=1}^{D}\prod_{k=1}^{K}\text{Bern}(B_{d,k}|\theta_{d,k})$, where $\theta_{d,k} = \sigma(\rho_{d,k} + logit(\pi_k))$ and $q(\boldsymbol{v}) = \prod_{k=1}^{K}\text{Beta}(v_k|\nu_{k,1}, \nu_{k,2})$. Thus we have $\Theta = \{\nu_{k,1}, \nu_{k,2}, \{\mu_{d,k}, \sigma_{d,k}, \rho_{d,k}\}_{d=1}^{D}\}_{k=1}^{K}$ as set of learnable variational parameters.

Each column of $\boldsymbol{B}$ represents the binary mask for the weights incident to a particular node. Note that although these binary variables (in a single column of $\boldsymbol{B}$) share a common prior, the posterior for each of these variables are different, thereby allowing a task to selectively choose a subset of the weights, with the common prior controlling the degree of sparsity.

$$\begin{aligned}\mathcal{L} =& \mathbb{E}_{q(\boldsymbol{V},\boldsymbol{B},\boldsymbol{v})}[\ln p(\boldsymbol{Y}|\boldsymbol{V}, \boldsymbol{B}, \boldsymbol{v})] \\ &- KL(q(\boldsymbol{V}, \boldsymbol{B}, \boldsymbol{v})||p(\boldsymbol{V}, \boldsymbol{B}, \boldsymbol{v}))\end{aligned} \tag{4}$$

Bayes-by-backprop (Blundell et al., 2015) is a common choice for performing variational inference in such models. Eq. 4 defines the Evidence Lower Bound (ELBO) in terms of data-dependent likelihood and data-independent KL terms which further gets decomposed using mean-field factorization.

$$\begin{aligned}\mathcal{L} =& \frac{1}{S}\sum_{i=1}^{S}[f(\boldsymbol{V}^i, \boldsymbol{B}^i, \boldsymbol{v}^i) - KL[q(\boldsymbol{B}|\boldsymbol{v}^i)||p(\boldsymbol{B}|\boldsymbol{v}^i)]] \\ &- KL[q(\boldsymbol{V})||p(\boldsymbol{V})] - KL[q(\boldsymbol{v})||p(\boldsymbol{v})]\end{aligned} \tag{5}$$

The expectation terms are optimized by unbiased gradients from the respective posteriors. All the KL divergence terms in (Eq. 5) have closed form expressions; hence using them directly rather than estimating them from Monte Carlo samples alleviates the approximation error as well as the computational overhead, to some extent. The log-likelihood term can be decomposed as

$$\begin{aligned}f(\boldsymbol{V}, \boldsymbol{B}, \boldsymbol{v}) &= \log \text{Lik}(\boldsymbol{Y}|\boldsymbol{V}, \boldsymbol{B}, \boldsymbol{v}) \\ &= \log \text{Lik}(\boldsymbol{Y}|\boldsymbol{W}_{out}\boldsymbol{\phi}_{NN}(\boldsymbol{X}; \boldsymbol{V}, \boldsymbol{B}, \boldsymbol{v}))\end{aligned} \tag{6}$$

where $(\boldsymbol{X}, \boldsymbol{Y})$ is the training data. For regression, Lik can be Gaussian with some noise variance, while for classification it can be Bernoulli with a probit or logistic link. Details of sampling gradient computation for terms involving beta and Bernoulli r.v.'s is provided in the Supplementary Material. (Sec. E).

3.3. Unsupervised Continual Learning

Our discussion thus far has primarily focused on continual learning where each task is a supervised learning problem. Our framework however readily extends to unsupervised continual learning (Nguyen et al., 2018; Smith et al., 2019; Rao et al., 2019b) where we assume that each task involves learning a deep generative model, commonly a VAE. In this case, each input observation $\boldsymbol{x}_n$ has an associated latent variable $\boldsymbol{z}_n$. Collectively denoting all inputs as $\boldsymbol{X}$ and latent variables as $\boldsymbol{Z}$, we can define ELBO similar to Eq. 4 as

$$\begin{aligned}\mathcal{L} =& \mathbb{E}_{q(\boldsymbol{Z},\boldsymbol{V},\boldsymbol{B},\boldsymbol{v})}[\ln p(\boldsymbol{X}|\boldsymbol{Z}, \boldsymbol{V}, \boldsymbol{B}, \boldsymbol{v})] \\ &- KL(q(\boldsymbol{Z}, \boldsymbol{V}, \boldsymbol{B}, \boldsymbol{v})||p(\boldsymbol{Z}, \boldsymbol{V}, \boldsymbol{B}, \boldsymbol{v}))\end{aligned} \tag{7}$$

Note that, unlike the supervised case, the above ELBO also involves an expectation over $\boldsymbol{Z}$. Similar to Eq. 5 this can be

approximated using Monte Carlo samples, where each $\boldsymbol{z}_n$ is sampled from the amortized posterior $q(\boldsymbol{z}_n|\boldsymbol{V}, \boldsymbol{B}, \boldsymbol{v}, \boldsymbol{x}_n)$. In addition to learning the model size adaptively, as shown in the schematic diagram (Fig. 2 (ii)), our model learns shared weights and task-specific masks for the encoder and decoder models. In contrast, VCL uses fixed-sized model with entirely task-specific encoders, which prevents knowledge transfer across the different encoders.

3.4. Other Key Considerations

Task Agnostic Setting Our framework extends to task-agnostic continual learning as well where the task boundaries are unknown. Based on (Lee et al., 2020), we use a gating mechanism (Eq. 8 with t_n represents the task identity of n^{th} sample x_n) and define marginal log likelihood as

$$p(\boldsymbol{t}_n = k|\boldsymbol{x}_n) = \frac{p(\boldsymbol{x}_n|\boldsymbol{t}_n = k)p(\boldsymbol{t}_n = k)}{\sum_{k=1}^{K} p(\boldsymbol{x}_n|\boldsymbol{t}_n = k)p(\boldsymbol{t}_n = k)} \tag{8}$$

$$\begin{aligned}\log p(\boldsymbol{X}) =& \mathbb{E}_{q(t=k)}\left[p(\boldsymbol{X}, \boldsymbol{t} = k|\theta)\right] \\ &+ KL\left(q(\boldsymbol{t} = k)||p(\boldsymbol{t} = k|\boldsymbol{X}, \theta)\right)\end{aligned} \tag{9}$$

where, $q(t = k)$ is the variational posterior over task identity. Similar to E-step in Expectation Maximization (Moon, 1996), we can reduce the KL-Divergence term to zero and maximize marginal log likelihood in M-step as

$$\arg\max_{\theta} \mathbb{E}_{p(\boldsymbol{t}=k|\boldsymbol{X}, \theta_{\text{old}})} \log p(\boldsymbol{X}|\boldsymbol{t} = k) \tag{10}$$

Here, $\log p(\boldsymbol{X}|\boldsymbol{t} = k)$ is intractable but can be replaced with its variational lower bound (Eq. 7). We use Monte Carlo sampling for approximating $p(\boldsymbol{x}_n|\boldsymbol{t}_n = k)$. The prior distribution over tasks $p(\boldsymbol{t} = k)$ can be assumed to be a uniform distribution but it fails to consider the degree upto which each mixture is being used. Therefore, we keep a count over the number of instances belonging to each task and use that as prior (i.e $p(\boldsymbol{t} = k) = \frac{N_k}{N}$, with N_k being effective number of instances belonging to task k and $N = \sum_k N_k$)

Inspired from (Rao et al., 2019a), we rely on a threshold to determine if the data point is an instance from a new task or not. During training, any instance with $\mathbb{E}_{p(\boldsymbol{t}_n|\boldsymbol{x}_n)}(\text{ELBO}_{\boldsymbol{t}_n})$ less than threshold T_{new} is added to a buffer $\mathcal{D}_{\text{new}}$. Once the buffer $\mathcal{D}_{\text{new}}$ reaches a fixed size limit M, we extend our network with new task parameters and train our network on $\mathcal{D}_{\text{new}}$, with known task labels (i.e $p(y = T + 1) = 1$ where T is total number of tasks learned). Note that training this mixture model will require us to have all task specific variational parameters to be present at every time step unlike the case in earlier settings where we only need to store the masks and can discard the variational parameters of previously seen tasks. This will result in storage problems since the number of parameters will grow linearly with the number of tasks. To overcome this issue we fix the task specific mask parameters and prior parameters before the network is trained on new task instances. After the task specific parameters have been fixed, the arrival of data belonging to a previously seen task t_{prev} is handled by training the network parameters with task.

Masked Priors Using the previous task's posterior as the prior for current task (Nguyen et al., 2018) may introduce undesired regularization in case of partially trained parameters that do not contribute to previous tasks and may promote catastrophic forgetting. Also, the choice of the initial prior as Gaussian leads to creation of more nodes than required due to regularization. To address this, we mask the new prior for the next task t with the initial prior p_t defined as

$$p_t(\boldsymbol{V}_{d,k}) = B^o_{d,k} q_{t-1}(\boldsymbol{V}_{d,k}) + (1 - B^o_{d,k}) p_0(\boldsymbol{V}_{d,k}) \tag{11}$$

where $\boldsymbol{B}^o$ is the overall combined mask from all previously learned tasks i.e., $(\boldsymbol{B}^1 \cup \boldsymbol{B}^2 ... \cup \boldsymbol{B}^{t-1})$, q_{t-1}, p_t are the previous posterior and current prior, respectively, and p_0 is the prior used for the first task. The standard choice of initial prior p_0 can be a uniform distribution.

Selective Finetuning While training with reparameterization (Gumbel-softmax), the sampled masks are close to binary but not completely binary which reduces performance a bit with complete binary mask. So we fine-tune the network with fixed masks to restore performance.

Dynamic Expansion Although our inference scheme uses a truncation-based approach for the IBP posterior, it is possible to do inference in a truncation-free manner. One possibility is to greedily grow the layer width until performance saturates. However we found that this leads to a bad optima (low peaks of likelihood). We can leverage the fact that, given a sufficiently large number of columns, the last few columns of the IBP matrix tend to be all zeros. So we can increase the number of hidden nodes after every iteration to keep the number of such empty columns equal to a constant value $\mathcal{T}^l$ in following manner.

$$C^l_j = C^l_{j+1} \prod_{i}^{D^l} \mathbb{I}(B^l_{ij} = 0), \quad G^l = \mathcal{T}^l - \sum_{j=1}^{K^l} C^l_j \tag{12}$$

where l represents current layer index, B^l is the sampled IBP mask for current task, and C^l_j indicates if all columns from j^{th} column onward are empty. G^l is the number of hidden units to expand in the current network layer.

4. Related Work

One of the key challenges in continual learning is to prevent catastrophic forgetting, typically addressed through regularization of the important parameter updates, preventing

them from drastically changing from the value learnt from the previous task(s). Notable methods based on this strategy include EwC (Kirkpatrick et al., 2017), SI (Zenke et al., 2017), LP (Smola et al., 2003), etc. As an alternative to regularizing in the weight space, functional regularization has also been proposed where the idea is to directly regularize the function's outputs when moving from one task to the next task (Benjamin et al., 2018; Titsias et al., 2019; Pan et al., 2020).

Bayesian approaches offer a natural remedy for catastrophic forgetting in that, for any task, the posterior of the model learnt from the previous task serves as the prior for the current task, which is the canonical online Bayes. This approach is used in recent works like VCL (Nguyen et al., 2018) and task agnostic variational Bayes (Zeno et al., 2018) for learning Bayesian neural networks in the CL setting. Our work is most similar in spirit to and builds upon this body of work.

Another key aspect in CL methods is *replay*, where some samples from previous tasks are used to fine-tune the model after learning a new task (thus refreshing its memory in some sense and avoiding catastrophic forgetting). Some of the works using this idea include (Lopez-Paz et al., 2017), which solves a constrained optimization problem at each task, the constraint being that the loss should decrease monotonically on a heuristically selected replay buffer; (Hu et al., 2019), which uses a partially shared parameter space for inter-task transfer and *generates* the replay samples through a data-generative module; and (Titsias et al., 2020), which learns a Gaussian process for each task, with a shared mean function in the form a feedforward neural network, the replay buffer being the set of inducing points typically used to speed up GP inference. For VCL and our work, the coreset serves as a replay buffer (Appx. C); but we emphasize that it is not the primary mechanism to overcome catastrophic forgetting in these cases, but rather an additional mechanism to preventing it.

Recent work in CL has investigated allowing the structure of the model to dynamically change with newly arriving tasks. Among these, strong evidence in support of our assumptions can be found in (Golkar et al., 2019), which also learns different sparse subsets of the weights of each layer of the network for different tasks. The sparsity is enforced by a combination of weighted L_1 regularization and threshold-based pruning. There are also methods that do not learn subset of weights but rather learn the subset of hidden layer nodes to be used for each task; such a strategy is adopted by either using Evolutionary Algorithms to select the node subsets (Fernando et al., 2017) or by training the network with task embedding based attention masks (Serrà et al., 2018). One recent approach (Adel et al., 2020), instead of using binary masks, tries to adapt network weights at different scales for different tasks; it is also designed only for discriminative tasks.

Among other related work, (Li et al., 2019; Yoon et al., 2018; Xu & Zhu, 2018) either reuse the parameters of a layer, dynamically grows the size of the hidden layer, *or* spawn a new set of parameters (the model complexity being bounded through regularization terms or reward based reinforcements). Most of these approaches however tend to be rather expensive and rely on techniques, such as neural architecture search. In another recent work (simultaneous development with our work), (Kessler et al., 2020) did a preliminary investigation on using IBP for continual learning. They however use IBP on hidden layer activations instead of weights (which they mention is worth considering), do not consider issues such as the ones we discussed in Sec. 3.4, and only applies to supervised setting. Modelling number active nodes for a given task has also been explored by (Serrà et al., 2018; Fernando et al., 2017; Ahn et al., 2019), but modelling posterior over connections weights between these nodes achieves more sparsity and flexibility in terms of structural learning at the cost of increased number of parameters, (von Oswald et al., 2020) tries to amortize the network parameters directly from input samples which is a promising direction for future work.

For non-stationary data, online variational Bayes is not directly applicable as it assumes independently and identically distributed (i.i.d.) data. As a result of which the variance in Gaussian posterior approximation will shrink with an increase in the size of training data, (Kurle et al., 2020) proposed use of Bayesian forgetting, which can be naturally applied to our approach enabling it to work with non-stationary data but it requires some modifications for task-agnostic setup. In this work, we have not explored this extension keeping it as future work.

5. Experiments

We perform experiments on both supervised and unsupervised continual learning scenarios. We also evaluate our model on task-agnostic setup for unsupervised CL and compare our method with relevant state-of-the-art methods. In addition to the quantitative (accuracy/log-likelihood comparisons) and qualitative (generation) results, we also examine the network structures learned by our model. Some of the details (e.g., experimental settings) have been moved to the Supplementary Material.

5.1. Supervised Continual Learning

We first evaluate our model on standard supervised CL benchmarks. We experiment with several existing approaches such as, Pure Rehearsal (Robins, 1995), EwC (Kirkpatrick et al., 2017), IMM (Lee et al., 2017), DEN

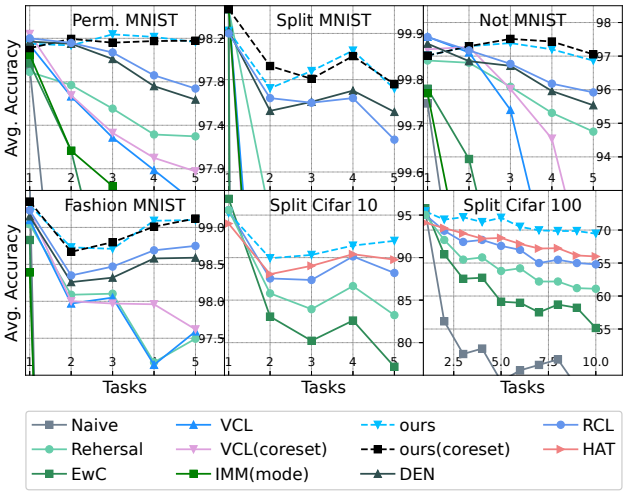


Figure 3. Mean test accuracies of tasks seen so far as newer tasks are observed on multiple benchmarks. Mean value is calculated as average of accuracies of all tasks after observing the last task

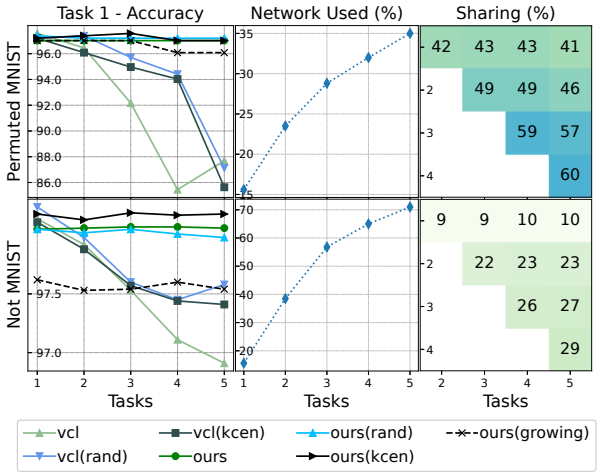


Figure 4. Variation in accuracy of first task (left), percentage of network capacity in use (middle) as new tasks are observed and percentage of shared network connections (right) among different task pairs in first hidden layer

(Yoon et al., 2018), RCL (Xu & Zhu, 2018), HAT (Serrà et al., 2018), VCL (Nguyen et al., 2018), and "Naïve" which learns a shared model for all the tasks. We perform our evaluations on five supervised CL benchmarks: SplitMNIST, Split notMNIST(small), Permuted MNIST, Split fashionMNIST and Split Cifar100. The last layer heads (Supp. Mat. D.1) were kept separate for each task for a fair comparison. For Split Cifar10, Split MNIST, Split notMNIST and Split fashionMNIST each dataset is split into 5 binary classification tasks. For Split Cifar100, the dataset was split into 10 multiclass classification tasks. For Permuted MNIST, each task is a multiclass classification problem with a fixed random permutation applied to the pixels of every image. We generated 5 such tasks for our experiments[1].

Performance evaluation Suppose we have a sequence of T tasks. To gauge the effectiveness of our model towards preventing catastrophic forgetting, we report (i) the test accuracy of first task after learning each of the subsequent tasks; and (ii) the average test accuracy over all previous tasks $1, 2, \ldots t$ after learning each task t. For fair comparison, we use the same architecture for each of the baselines (details in Supp. Mat.), except for DEN, RCL which learn the structure of the network dynamically.

Fig. 3 shows the mean test accuracies on all supervised benchmarks as new tasks are observed. As shown, the average test accuracy obtained by our method (without as well as with coresets) is better than the compared baselines (here, we have used random point selection method for coresets). Moreover, the accuracy drops much more slowly than other baselines, showing the efficacy of our model in preventing catastrophic forgetting. In Fig. 4, we show that deviations in accuracy of first task as new tasks arrive and compare specifically with VCL. In this case, we observe that our method yields a relatively stable first task accuracy as compared to VCL, which is a result of adaptive structural learning. We also observe that, for permuted MNIST, the accuracy of the first task initially increases with training of new tasks which shows the presence of backward transfer, which is another desideratum of CL. We also report the performance of our dynamically growing network variant (for more details refer Supp. Mat. Sec. 3.4).

Network hidden layer sizes	Avg accuracy (5 tasks)
$[200]$	98.180 ± 0.187
$[100, 50]$	98.188 ± 0.163
$[250, 100, 50]$	98.096 ± 0.152

Table 1. Comparing performance on Permuted MNIST under different network configurations

To justify the choice of single hidden layer with 200 units in MNIST like experiments, we compare our model on Permuted MNIST experiment with multiple network depths and with separate heads. As shown in Table 1, a single hidden layer is sufficient for obtaining good enough results. Table 2 shows the mean test accuracies obtained on permuted MNIST and split Cifar-100 experiments. Here, we can observe that the use of uniform prior performs better in comparison to Gaussian prior in retaining the highest mean accuracy. On the other hand, when masks are not used, the performance drops but the results are still better as

[1]The code for our models can be found at this link: https://github.com/npbcl/icml21

Masked Prior	Permuted MNIST	Split Cifar-100
Uniform prior	98.180	73.745
Gaussian prior	97.994	69.921
No Masking	97.489	65.399

Table 2. Avg accuracy obtained under different masking priors

Method	Epochs	Time/Task (sec)	Acc (10 tasks)
Ours	10	142	0.9794
VCL	100	380	0.9487
EwC	10	51	0.9173

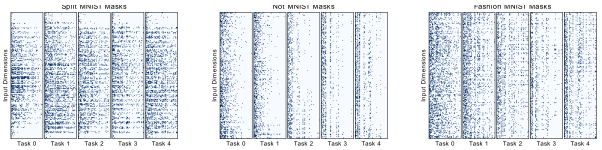
Table 3. Performance statistics on Permuted MNIST for 10 tasks. Number of epochs and time taken for training is on per task basis

compared to VCL, which can be attributed to the IBP prior based structural learning.

We have performed our experiments with separate heads for each task of permuted MNIST. Some methods use a single head and do not require task labels at test-time. Table 3 shows a comparison with some of the baselines (that supports single head) with our model (single head) on Permuted MNIST for 10 tasks. We also report number of epochs and average time to run for a rough comparison of time complexity taken by each model.

Some Structural Observations An appealing aspect of our work is that, the results reported above, which are competitive with the state-of-the-art, are achieved with very sparse neural network structures learned by the model, which we analyze qualitatively here. Fig. 4 shows some examples of network structures learnt by our model. As shown in Fig. 4 (Network Used), the IBP prior concentrates weights on very few nodes, and learns sparse structures. Also, most newer tasks tend to allocate fewer weights and yet perform well, implying effective forward transfer. Another important observation as shown in Fig. 4 is that the weight sharing between similar tasks in notMNIST is higher than that between non-similar tasks in permuted MNIST. Also note that new tasks show higher weight sharing, irrespective of similarity. This is an artifact induced by IBP (Sec. 3.1) which tends to allocate more active weights on upper side of matrix. We therefore conclude that although a new task tends to share weights learned by old tasks, the new connections that it creates are indispensable for its performance. Intuitively, the more unrelated a task is to previously seen ones, the more new connections it will make, thus reducing *negative transfer* (an unrelated task adversely affecting other tasks) between tasks.

Fig. 5(a) shows the masks are captured on the pixel values where the digits in MNIST datasets have high value and



(a) split MNIST (b) not MNIST (c) fashion

Figure 5. Masks learned for first hidden layer after training on each task of split MNIST (**left**), not MNIST (**middle**) and fashion MNIST (**right**) experiments. Active nodes are represented as dark colored cells in the matrix.

zeros elsewhere which represents that our models adapts with respect to data complexity and only uses those weights that are required for the task. Due to the use of the IBP prior, the number of active weights tends to shrink towards the first few nodes of the first hidden layer. This observation supports our idea of using the IBP prior to learn the model structure based on data complexity. Similar behaviour can be seen in notMNIST and fashionMNIST in Fig. 5(b and c).

5.2. Unsupervised Continual Learning

We next evaluate our model on generative tasks under the CL setting. For that, we compare our model with existing approaches such as Naïve, EwC (Kirkpatrick et al., 2017) and VCL (Nguyen et al., 2018). We do not include other methods mentioned in supervised setup as their implementation does not incorporate generative modeling. We perform continual learning experiments for deep generative models using a VAE style network. We consider two datasets, MNIST and notMNIST. For MNIST, the tasks are sequence of single digit generation from 0 to 9. Similarily, for notMNIST each task is single character generation from A to J. Note that, unlike VCL and other baselines where all tasks have separate encoder and a shared decoder, as we discuss in Sec. 3.3, our model uses a shared encoder for all tasks, but with task-specific masks for each encoder (cf., Fig. 2 (ii)). This enables transfer of knowledge across tasks while the task-specific mask retains knowledge specific to the task and effectively prevents catastrophic forgetting.

Sequential Generation: As shown in Fig 6 (left), the modeling changes we introduce for the unsupervised setting, results in much improved log-likelihood on held-out sets. In each individual figure in Fig 7, each row represents the set of generated samples from all the previously seen tasks and the current task. We observe that the quality of generated samples from our method does not deteriorate much in comparison to other baselines as more tasks are encountered. This shows that our model can efficiently perform sequential generative modeling while reusing subsets of the network and activating minimal number of nodes for each task.

Task-Agnostic Learning: Fig 6 (right) shows a particular case on MNIST data where nine tasks were inferred out

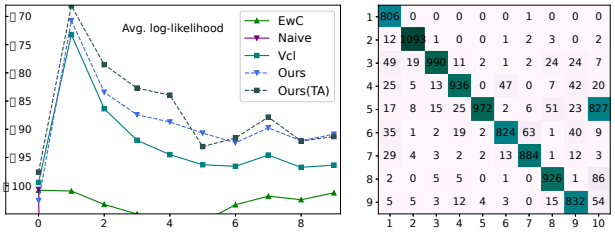


Figure 6. Mean log-likelihoods (left) for MNIST sequential generation and confusion matrix (right) representing test samples mapped to each generative task learned in TA (task-agnostic) setting

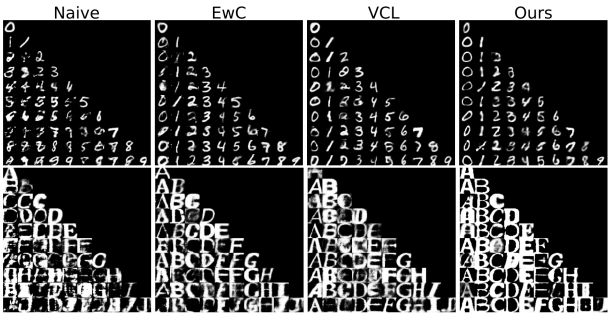


Figure 7. Sequential generation for MNIST (**top**) and notMNIST (**bottom**) datasets (Supp. Material contains more illustrations and zoomed-in versions)

of 10 classes, with high correlation among classes 4 and 9 due to visual similarity between them. Each task uses a subset of the network connections. This result illustrates our model's ability to learn task relations based on network sharing. Further, the log-likelihood obtained for the task-agnostic setting is comparable to our model with known task boundaries, suggesting that our approach can be used effectively in task-agnostic settings as well.

Representation Learning: Table 4 represents the quality of the *unsupervisedly* learned representation by our unsupervised continual learning approach. For this experiment, we use the learned representations in a KNN classification model with different values of K. We note that, despite having task-specific encoders, VCL and other baselines fail to learn good latent representations, while the proposed model learns good representations when task boundaries are known and is also comparable to state-of-the-art baseline CURL (Rao et al., 2019a) that are specifically designed for task-agnostic unsupervised representation learning.

Fig 8 shows the t-SNE representations learned by our model in both scenarios with known and unknown task boundaries and shows a comparison with VCL. We observe that when task boundaries are known, the model learns very good separate latent representation, but the task boundaries start to somewhat overlap in the task-agnostic setting as the task inference is not perfect. VCL, on the other hand does not learn good latent representation as it does not have a task-inference mechanism.

Method	**3-NN**	**5-NN**	**10-NN**
Naive	30.1%	33.1%	36.0%
EwC	16.6%	19.5%	22.3%
VCL	16.0%	19.1%	30.2%
Ours	**0.37%**	**0.40%**	**0.08%**
Ours (Task Agnostic)	5.79%	5.32%	5.62%
CURL (Task Agnostic)	**4.58%**	**4.35%**	**4.50%**

Table 4. MNIST K-NN test error rates obtained in latent space for both task-agnostic and know task setting.

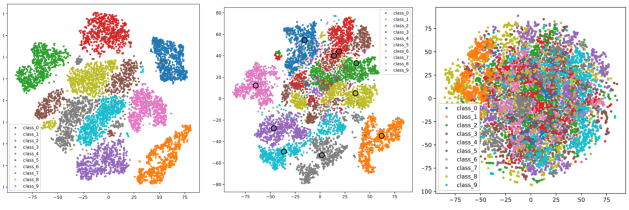


Figure 8. Comparison of t-SNE plots learned between our task-known (**left**), task-agnostic (**middle**) models and VCL(**right**)

6. Conclusion

We have successfully unified structure learning in Bayesian neural networks with the variational Bayes approach of doing continual learning, demonstrating competitive performance with state-of-the-art models on both discriminative (supervised) and generative (unsupervised) learning problems. In this work, we have experimented with task-incremental continual learning for supervised setup and sequential generation task for unsupervised setting. we believe that our task-agnostic setup can be extended to class-incremental learning scenario where inputs from a set of classes arrives sequentially and model is expected to perform classification over all observed classes. It would also be interesting to generalize this idea to more sophisticated network architectures, such as recurrent or residual neural networks, possibly by also exploring improved approximate inference methods. A few other interesting extensions would be in semi-supervised continual learning and continual learning with non-stationary data. Adapting other sparse Bayesian structure learning methods, e.g. (Ghosh et al., 2018) to the continual learning setting is also a promising avenue. Adapting the *depth* of the network is a more challenging endeavour that might also be undertaken. We leave these extensions for future work.

Acknowledgment: PR acknowledges support from Visvesvaraya Young Faculty Fellowship and from Qualcomm Innvotation Fellowship.

References

Adel, T., Zhao, H., and Turner, R. E. Continual learning with adaptive weights (claw). In *International Conference on Learning Representations*, 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*, volume 32, pp. 4392–4402. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/2c3ddf4bf13852db711dd1901fb517fa-Paper.pdf.

Benjamin, A. S., Rolnick, D., and Kording, K. Measuring and regularizing networks in function space. *arXiv preprint arXiv:1805.08289*, 2018.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. *ICML*, 2015.

Doshi, F., Miller, K., Van Gael, J., and Teh, Y. W. Variational inference for the indian buffet process. In *AISTATS*, pp. 137–144, 2009.

Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A. A., Pritzel, A., and Wierstra, D. Pathnet: Evolution channels gradient descent in super neural networks. *CoRR*, abs/1701.08734, 2017. URL http://arxiv.org/abs/1701.08734.

Flesch, T., Balaguer, J., Dekker, R., Nili, H., and Summerfield, C. Comparing continual task learning in minds and machines. *Proceedings of the National Academy of Sciences*, 115(44):E10313–E10322, 2018.

Ghosh, S., Yao, J., and Doshi-Velez, F. Structured variational learning of bayesian neural networks with horseshoe priors. *ICML*, 2018.

Golkar, S., Kagan, M., and Cho, K. Continual learning via neural pruning. *CoRR*, abs/1903.04476, 2019. URL http://arxiv.org/abs/1903.04476.

Griffiths, T. L. and Ghahramani, Z. The indian buffet process: An introduction and review. *JMLR*, 12(Apr):1185–1224, 2011.

Hoffman, M. and Blei, D. Stochastic Structured Variational Inference. *AISTATS*, 38:361–369, 2015.

Hu, W., Lin, Z., Liu, B., Tao, C., Tao, Z., Ma, J., Zhao, D., and Yan, R. Overcoming catastrophic forgetting via model adaptation. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGvcoA5YX.

Kessler, S., Nguyen, V., Zohren, S., and Roberts, S. Hierarchical indian buffet neural networks for bayesian continual learning, 2020. URL https://arxiv.org/abs/1912.02290.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *ICLR*, 2013.

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

Kurle, R., Cseke, B., Klushyn, A., van der Smagt, P., and Günnemann, S. Continual learning with bayesian neural networks for non-stationary data. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJlsFpVtDB.

Lee, S., Ha, J., Zhang, D., and Kim, G. A neural dirichlet process mixture model for task-free continual learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJxSOJStPr.

Lee, S.-W., Kim, J.-H., Jun, J., Ha, J.-W., and Zhang, B.-T. Overcoming Catastrophic Forgetting by Incremental Moment Matching. *NIPS*, Mar 2017.

Li, X., Zhou, Y., Wu, T., Socher, R., and Xiong, C. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. *ICML*, 2019.

Lopez-Paz, D. et al. Gradient episodic memory for continual learning. In *NIPS*, pp. 6467–6476, 2017.

Moon, T. K. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60, 1996.

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. *ICLR*, 2018.

Pan, P., Swaroop, S., Immer, A., Eschenhagen, R., Turner, R. E., and Khan, M. E. Continual deep learning by functional regularisation of memorable past. In *NeurIPS*, 2020.

Pan, S. J. and Yang, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10): 1345–1359, 2009.

Panousis, K., Chatzis, S., and Theodoridis, S. Nonparametric bayesian deep networks with local competition. In *International Conference on Machine Learning*, pp. 4980–4988, 2019.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

Rao, D., Visin, F., Rusu, A., Pascanu, R., Teh, Y. W., and Hadsell, R. Continual unsupervised representation learning. In *Advances in Neural Information Processing Systems 32*, pp. 7647–7657. Curran Associates, Inc., 2019a.

Rao, D., Visin, F., Rusu, A., Pascanu, R., Teh, Y. W., and Hadsell, R. Continual unsupervised representation learning. In *Advances in Neural Information Processing Systems*, pp. 7645–7655, 2019b.

Ring, M. B. Child: A first step towards continual learning. *Machine Learning*, 28(1):77–104, 1997.

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

Schwarz, J., Luketina, J., Czarnecki, W. M., Grabska-Barwinska, A., Whye Teh, Y., Pascanu, R., and Hadsell, R. Progress & Compress: A scalable framework for continual learning. *ICML*, May 2018.

Serrà, J., Surís, D., Miron, M., and Karatzoglou, A. Overcoming catastrophic forgetting with hard attention to the task. *CoRR*, abs/1801.01423, 2018.

Smith, J., Baer, S., Kira, Z., and Dovrolis, C. Unsupervised continual learning and self-taught associative memory hierarchies. *ICLR*, 2019.

Smola, A. J., Vishwanathan, V., and Eskin, E. Laplace propagation. In *NIPS*, pp. 441–448, 2003.

Titsias, M. K., Schwarz, J., Matthews, A. G. d. G., Pascanu, R., and Teh, Y. W. Functional regularisation for continual learning with gaussian processes. In *International Conference on Learning Representations*, 2019.

Titsias, M. K., Schwarz, J., Matthews, A. G. d. G., Pascanu, R., and Teh, Y. W. Functional regularisation for continual learning using gaussian processes. *ICLR*, 2020.

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

von Oswald, J., Henning, C., Sacramento, J., and Grewe, B. F. Continual learning with hypernetworks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgwNerKvB.

Xu, J. and Zhu, Z. Reinforced Continual Learning. *NIPS*, art. arXiv:1805.12369, May 2018.

Xu, K., Srivastava, A., and Sutton, C. Variational russian roulette for deep bayesian nonparametrics. In *International Conference on Machine Learning*, pp. 6963–6972, 2019.

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 *ICML*, pp. 3987–3995. JMLR. org, 2017.

Zeno, C., Golan, I., Hoffer, E., and Soudry, D. Task agnostic continual learning using online variational bayes. *arXiv preprint arXiv:1803.10123*, 2018.