Hierarchical Indian Buffet Neural Networks for Bayesian Continual Learning

Samuel Kessler[1] **Vu Nguyen**[2] **Stefan Zohren**[1] **Stephen J. Roberts**[1]

[1]University of Oxford
[2]Amazon Adelaide

Abstract

We place an Indian Buffet process (IBP) prior over the structure of a Bayesian Neural Network (BNN), thus allowing the complexity of the BNN to increase and decrease automatically. We further extend this model such that the prior on the structure of each hidden layer is shared globally across all layers, using a Hierarchical-IBP (H-IBP). We apply this model to the problem of resource allocation in Continual Learning (CL) where new tasks occur and the network requires extra resources. Our model uses online variational inference with reparameterisation of the Bernoulli and Beta distributions, which constitute the IBP and H-IBP priors. As we automatically learn the number of weights in each layer of the BNN, overfitting and underfitting problems are largely overcome. We show empirically that our approach offers a competitive edge over existing methods in CL.

1 INTRODUCTION

Humans have the ability to continually learn, consolidate their knowledge and leverage previous experiences when learning a new set of skills. In Continual Learning (CL) an agent must also learn continually, presenting several challenges including learning online, avoiding forgetting and efficiently allocating resources for learning new tasks. In CL, a neural network model is required to learn a series of tasks, one by one, and remember how to perform each. The model is given a set of M tasks sequentially $\mathscr{T}_t$ for $t = 1, \dots M$. Where each task is comprised of a dataset. The model will lose access to the training dataset for task $\mathscr{T}_t$ but will be continually evaluated on the test sets for all previous tasks $\mathscr{T}_i$ for $i \leq t$, we will introduce the problem setting more formally in the next section.

The principal challenges to CL are threefold, firstly models need to overcome *catastrophic forgetting* of old tasks; a neural network will exhibit forgetting of previous tasks after having learnt a few tasks [Goodfellow et al., 2015]. Secondly, models need to leverage knowledge transfer from previously learnt tasks for learning a new task $\mathscr{T}_t$. And finally, the model needs to have enough neural resources available to learn a new task and adapt to the complexity of the task at hand.

One of the main approaches to CL involves the use of the natural sequential learning approach embedded within Bayesian inference. The prior for task $\mathscr{T}_i$ is the posterior which is obtained from the previous task $\mathscr{T}_{i-1}$. This enables knowledge transfer and offers an approach to overcome catastrophic forgetting. Previous Bayesian CL approaches have leveraged Laplace approximations [Kirkpatrick et al., 2017] and variational inference [Nguyen et al., 2018] to aid computational tractability. Whilst Bayesian methods solve the first and second objectives above, the third objective of ensuring that the BNN has enough neural resources to adapt its complexity to the task at hand is not necessarily achieved. For instance, additional neural resources can alter performance on MNIST classification (see Table 1 in [Blundell et al., 2015]). This is a problem as the amount of neural resources required for a current task, may not be enough (or may be redundant) for a future task. Propagating a poor approximate posterior from one task will alter performance for all subsequent tasks.

Non-Bayesian neural networks use additional neurons to learn new tasks and prevent overwriting previous knowledge thus overcoming forgetting. The neural networks which have been trained on previous tasks are frozen and a new neural network is appended to the existing network for learning a new task [Rusu et al., 2016]. The problem with this approach is that of scalability: the number of neural resources increases linearly with the number of tasks. The scalability issue has been tackled with selective retraining and expansion with a group regulariser [Yoon et al., 2018]. However this solution is unable to shrink and so are vulnerable to overfitting if misspecified when starting CL. Moreover

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).

knowledge transfer and prevention of catastrophic forgetting are not solved in a principled manner, unlike approaches couched in a Bayesian framework.

As the resources required are typically unknown in advance, we propose a BNN which adds or withdraws neural resources automatically in response to the data. This is achieved by drawing on Bayesian nonparametrics to learn the structure of each hidden layer of a BNN. Thus, the model size adapts to the amount of data seen and the difficulty of the task. This is achieved by using a binary latent matrix Z, distributed according to an Indian Buffet Process (IBP) prior [Griffiths and Ghahramani, 2011]. The IBP prior on an infinite binary matrix, Z, allows inference on which and how many neurons are required for each data point in a task. The weights of the BNN are treated as draws from non-interacting Gaussians [Blundell et al., 2015]. Catastrophic forgetting is overcome by repeated application of the Bayesian update rule, embedded within variational inference [Nguyen et al., 2018]. We summarise the contributions as follows. We present a novel BNN using an IBP prior and its hierarchical extension to automatically learn the complexity of each hidden layer according to the task difficulty. The model's effective use of resources is shown to be useful in CL. We derive a variational inference algorithm for learning the posterior distribution of the proposed models. In addition, our model elegantly bridges two separate CL approaches: expansion methods and Bayesian methods (more commonly referred to as regularization based methods in CL literature).

2 INDIAN BUFFET NEURAL NETWORKS

We introduce the CL problem setting in Section 2.1, variational Bayesian approaches to CL in Section 2.2. We present the IBP prior in Section 2.3 and the IBP prior on the latent binary matrix Z is then applied to a BNN such that the complexity of each hidden layer can be learnt from the data in Section 2.4. In Section 3, the Hierarchical IBP prior (H-IBP) is introduced and applied to the BNN to encourage a more regular structure. Thus, the use of an IBP and H-IBP prior over the hidden states of the BNN can be readily used together with the Bayesian CL framework presented, and so automatically adapt its complexity according to the task.

2.1 CONTINUAL LEARNING

Continual learning (CL) is a setting whereby a model must learn a set of tasks sequentially, while maintaining performance across all tasks. In CL, the model is shown a set of M tasks sequentially $\mathscr{T}_t$ for $t = 1, \ldots M$. Each task is comprised of a dataset such that $\mathscr{T}_t : \mathscr{D}_t = \left\{(x_i, y_i)\right\}$ for $i = 1, \ldots, N_t$. The inputs $x_i \in \mathbb{R}^d$ and outputs can be $y_i \in \mathbb{R}$ in the case of regression or a categorical variable for classification. Although the model will lose access to the training dataset for task $\mathscr{T}_t$, it will be continually evaluated on all previous tasks $\mathscr{T}_i$ for $i \leq t$. t can be used as a task identifier informing the agent when to start training on a new task or what task to being tested. For a comprehensive review of CL scenarios see van de Ven and Tolias [2018], Hsu et al. [2018].

2.2 BAYESIAN CONTINUAL LEARNING

The CL process can be decomposed into Bayesian updates where the approximate posterior for $\mathscr{T}_{t-1}$ can be used as a prior for task $\mathscr{T}_t$. Variational CL (VCL) [Nguyen et al., 2018] uses a BNN to perform the prediction tasks where the network weights are independent Gaussians. The variational posterior from previous tasks is used as a prior for new tasks. Consider learning the first task $\mathscr{T}_1$, and $\boldsymbol{\phi}$ are the variational random variables, then the variational posterior is $q_1(\boldsymbol{\phi}|\mathscr{D}_1)$. For the subsequent task, access to $\mathscr{D}_1$ is lost and the prior will be $q_1(\boldsymbol{\phi}|\mathscr{D}_1)$, optimization of the ELBO will yield the variational posterior $q_2(\boldsymbol{\phi}|\mathscr{D}_2)$. Generalising, the negative ELBO for the t-th task is:

$$\begin{aligned}\mathscr{L}(\boldsymbol{\phi}, \mathscr{D}_t) = D_{\mathrm{KL}}\left[q_t(\boldsymbol{\phi})||q_{t-1}(\boldsymbol{\phi}|\mathscr{D}_{t-1})\right] \\ - \mathbb{E}_{q_t}[\log p(\mathscr{D}_t|\boldsymbol{\phi})].\end{aligned} \tag{1}$$

The first term acts to regularise the posterior such that it is close to previous task's posterior and the second term is the log-likelihood of the data for the current task.

2.3 INDIAN BUFFET PROCESS PRIOR

Matrix decomposition aims to represent the data X as a combination of latent features: $X = ZA + \varepsilon$ where $X \in \mathbb{R}^{N\times D}$, $Z \in \mathbb{Z}_2^{N\times K}$, $A \in \mathbb{R}^{K\times D}$ and ε is an observation noise. Each element in Z corresponds to the presence or absence of a latent feature from A. Specifically, $z_{ik} = 1$ corresponds to the presence of a latent feature A_k in observation X_i and $k \in \{1, \cdots, \infty\}$ all columns in Z with $k > K$ are assumed to be zero. In a scenario where the number of latent features K is to be inferred, then the IBP prior on Z is suitable [Doshi-Velez et al., 2009].[1]

One representation of the IBP prior is the stick-breaking formulation [Teh et al., 2007]. The probability π_k is assigned to the column z_k for $k \in \{1, \cdots, \infty\}$, whether a feature has been selected is determined by $z_{nk} \sim \mathrm{Bern}(\pi_k)$. This parameter π_k is generated according to the following stick-breaking process: $v_k \sim \mathrm{Beta}(\alpha, 1)$, and $\pi_k = \prod_{i=1}^{k} v_i$, thus π_k decreases exponentially with k. The Beta concentration parameter α controls how many features one expects to see in the data, the larger α is, the more latent features are present.

[1] We provide a notebook to demonstrate how the IBP prior can be used for the matrix factorization `https://bit.ly/3asylU5`. In particular, we illustrate how one doesn't need to specify the number of latent dictionary items to infer. This means we do not need to set the hidden state size of a BNN for our model.

2.4 ADAPTATION WITH THE IBP PRIOR

Consider a BNN with k_j neurons for each layer $j \in \{1, \dots J\}$ layers. Thence, for an arbitrary activation f, the binary matrix Z is applied elementwise $h_j = f(h_{j-1}W_j) \circ Z_j$ where $h_{j-1} \in \mathbb{R}^{N \times k_{j-1}}$, $W_j \in \mathbb{R}^{k_{j-1} \times k_j}$, $Z_j \in \mathbb{Z}_2^{N \times k_j}$, and where $\circ$ is the elementwise product and N is the number of data points per batch. We have ignored biases for simplicity. Z_j is distributed according to an IBP prior. The IBP prior has some suitable properties for this application: the number of neurons sampled grows with N and the promotion of "rich get richer" scheme for neuron selection [Griffiths and Ghahramani, 2011]. For convenience, we term the IBP BNN as IBNN for the remainder of the paper.

The number of neurons selected grow or contract according to the variational objective; which depends on the complexity of the data. This allows for efficient use of neural resources which is crucial to a successful CL model. The variational objectives for the IBP prior and BNN are introduced further down the line in Section 2.5 and Section 3.2. Additionally, the "rich get richer" scheme is useful since the common neurons are selected across tasks enabling knowledge transfer and preventing forgetting.

As a standard practice in variational inference with a Bayesian nonparametric prior, we use a truncation level K, to the maximum number of features in the variational IBP posterior. [Doshi et al., 2009] present bounds on the marginal distribution of X in a matrix factorisation setting and show that the bound decreases exponentially as K increases. A similar behaviour is expected for our application.

2.5 STRUCTURED VI

Structured stochastic VI (SSVI) has been shown to perform better inference of the IBP posterior than mean-field VI in deep latent variable models [Singh et al., 2017]. Hence, this inference method has been chosen for learning and presented next.

A separate binary matrix Z_j can be applied to each layer $j \in \{1, \dots J\}$ of a BNN. The subscript j is dropped for clarity. The structured variational approximation is: $q(\phi) = \prod_{k=1}^{K} q(v_k) q(\boldsymbol{w}_k) \prod_{i=1}^{N} q(z_{ik}|v_k)$, where the random variables are $\phi = \{v_k, \boldsymbol{w}_k, Z_k\}$ and the variational parameters $\varphi = \{\alpha_k, \beta_k, \boldsymbol{\mu}_k, \boldsymbol{\sigma}_k\}$ for all k. The variational distributions over ϕ are defined below and the variational IBP posterior is truncated to K. The constituent distributions are $q(v_k) = \text{Beta}(\alpha_k, \beta_k)$, $\pi_k = \prod_{i=1}^{k} v_i$, $q(z_{ik}) = \text{Bern}(\pi_k)$ and the BNN weights are independent draws from $q(\boldsymbol{w}_k) = \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k^2 \mathbb{1})$. Having defined the structured variational objective, the negative ELBO is:

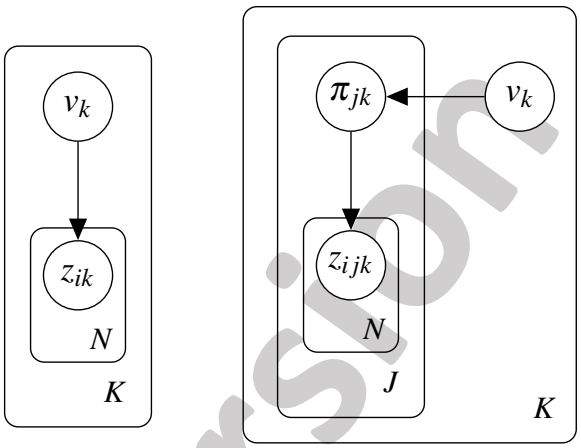

Figure 1: The graphical model for the structured variational posterior approximation for **Left**, the IBP and **Right**, the H-IBP^2. Using the language of the eponymous IBP prior metaphor, k (dishes) indicates the neurons, the number selected K adjusts flexibly. j (restaurants) is the number of layers which is fixed. i (customers) is a data point.

$$\begin{aligned} \mathcal{L}(\phi, \mathcal{D}) = & D_{\text{KL}}(q(\boldsymbol{v})||p(\boldsymbol{v})) + D_{\text{KL}}(q(\boldsymbol{w})||p(\boldsymbol{w})) \\ & - \sum_{i=1}^{N} \mathbb{E}_{q(\phi)} \left[\log p(y_i|\mathbf{x}_i, \mathbf{z}_i, \boldsymbol{w})\right] \\ & + \sum_{i=1}^{N} D_{\text{KL}}(q(\mathbf{z}_i|\boldsymbol{\pi})||p(\mathbf{z}_i|\boldsymbol{\pi})). \end{aligned} \tag{2}$$

We note that the Bernoulli random variables have an explicit dependence on the stick-breaking probabilities in the structured variational approximation. However, this explicit dependence between parameters is removed in a mean field approximation. The variational parameter Beta parameters will control the expected hidden state size for the BNN and is automatically inferred. For sequential Bayesian updates the posterior is then used as the next task's prior, thus a prior only needs to be designed for the first task.

3 HIERARCHICAL IBNN

In the previous section, we presented the IBNN model to allow a BNN to automatically select the number of neurons for each layer according to the data. For a multi-layer BNN, one can apply the IBP prior independently for each layer. However, we can add an inductive bias to ensure that the inferred number of neurons are similar for across all layers of an MLP and ensure that information is shared across layers. To do this, we propose a hierarchical IBP prior [Thibaux and Jordan, 2007] for neuron selection across multiple layers. The number of neurons from all layers are generated from the same global prior, thus will discourage irregular structure in the BNN (a BNN with adjacent wide and narrow

layers might be inferred when using independent priors on each hidden layer). Of course, this property might not be desirable for all use cases, however the majority of BNNs used in the literature have a regular structure. We term our model as HIBNN and present the graphical model which describes the hierarchical IBP prior in Figure 1. This model has the advantage of having fewer learnable parameters as neuron selection will be driven by the global prior in comparison to having a single IBP per layer.

3.1 ADAPTATION WITH THE HIERARCHICAL IBP

The global probability of selecting the neuron positioned at the k-th index across all layers is defined according to a stick-breaking process:

$$\pi_k^0 = \prod_{i=1}^{k} v_i, \quad v_k \sim \text{Beta}(\alpha, 1), \quad k = 1, \cdots \infty. \tag{3}$$

Child IBPs are defined over the structure of each individual hidden layer of a BNN which depend on π_k^0 to define the respective Bernoulli probabilities of selecting neuron k in layer j:

$$\pi_{jk} \sim \text{Beta}\left(\alpha_j \pi_k^0, \alpha_j (1 - \pi_k^0)\right), \tag{4}$$

for $j \in \{1, \cdots J\}$, $k \in \{1, \cdots \infty\}$, where J is the number of layers in a BNN, α_j are hyperparameters [Thibaux and Jordan, 2007, Gupta et al., 2012]. The selection of the k-th neuron in the j-th layer by a particular data point i in the dataset of size N is thus:

$$z_{ijk} \sim \text{Bern}(\pi_{jk}), i = 1, \cdots N. \tag{5}$$

Notice that if k is small, π_k^0 is close to 1 then the shape parameter of the child Beta distribution will be large. At the same time the scale parameter will be small. So the Bernoulli probability in Equation (5) will be close to 1, as k increases π_k^0 and π_{jk} decrease. To infer the posterior $p(\boldsymbol{v}, Z, \boldsymbol{w} | \mathscr{D})$, we perform SSVI.

3.2 STRUCTURED VI

A structured variational posterior distribution which retains properties of the true posterior is desired such that the global stick-breaking probabilities influence child stick-breaking probabilities of each layer of the BNN. Let us define the variational distributions for our hidden variables as follows, $q(v_k^0) = \text{Beta}(\alpha_k^0, \beta_k^0)$ and $q(\pi_{jk}) = \text{Beta}(\alpha_j \pi_k^0, \alpha_j(1 - \pi_k^0))$, $\pi_k^0 = \prod_{i=1}^{k} v_i^0$, $q(z_{ijk}) = \text{Bern}(\pi_{jk})$ and the weights of the BNN are drawn from $q(\boldsymbol{w}_{jk}) = \mathscr{N}(\boldsymbol{\mu}_{jk}, \boldsymbol{\sigma}_{jk}^2 \mathbb{1})$.

[2]The mean-field approximation removes all edges from these graphical models.

The structured variational distribution is defined as follows

$$q(\phi) = \prod_{k=1}^{K} q(v_k^0) \prod_{j=1}^{J} q(\pi_{jk} | v_k^0) q(\boldsymbol{w}_{jk}) \prod_{i=1}^{N} q(z_{ijk} | \pi_{jk}) \tag{6}$$

where $\phi = \{\alpha_k^0, \beta_k^0, \boldsymbol{\mu}_{jk}, \boldsymbol{\sigma}_{jk}\}$ for all j and k, up to the variational truncation, K. Having defined the structured variational distribution, the negative ELBO is:

$$\begin{aligned}
\mathscr{L}(\boldsymbol{\phi}, \mathscr{D}) = & \text{KL}(q(\boldsymbol{v}^0) || p(\boldsymbol{v}^0)) + \sum_{j=1}^{J} \text{KL}(q(\boldsymbol{\pi}_j | \boldsymbol{v}^0) || p(\boldsymbol{\pi}_j | \boldsymbol{v}^0)) \\
& + \sum_{j=1}^{J} \text{KL}(q(\boldsymbol{w}_j) || p(\boldsymbol{w}_j)) \\
& - \sum_{j=1}^{J} \sum_{i=1}^{N} \mathbb{E}_{q(\phi)} [\log p(y_i | \mathbf{x}_i, \mathbf{z}_{ij}, \boldsymbol{w}_j)] \\
& + \sum_{j=1}^{J} \sum_{i=1}^{N} \text{KL}(q(\mathbf{z}_{ij} | \boldsymbol{\pi}_j) || p(\mathbf{z}_{ij} | \boldsymbol{\pi}_j)).
\end{aligned} \tag{7}$$

The child stick-breaking variational parameters for each layer are conditioned on the global stick-breaking parameters and the binary masks z_{ijk} for each neuron k in each layer j are conditioned on the child stick-breaking variational parameters. Thus, the variational structured posterior is able to capture dependencies of the prior. The learnable parameters are α_k^0, β_k^0, $\boldsymbol{\mu}_{jk}$ and $\boldsymbol{\sigma}_{jk}$ for all k neurons and for all layers j.

3.2.1 Inference

The variational posterior is obtained by optimising Equation (7) using structured stochastic VI. For inference to be tractable, we utilise three reparameterisations. The first is for the Gaussian weights [Kingma and Welling, 2014]. The second is an implicit reparameterisation of the Beta distribution [Figurnov et al., 2018]. The third reparameterisation uses a Concrete relaxation to the Bernoulli distribution [Maddison et al., 2017, Jang et al., 2017]. Details of these are in the Supplementary material, Section 1.

4 RELATED WORKS

IBP priors and model selection in deep learning. An IBP prior has been used in VAEs to automatically learn the number of latent features. Stick-breaking probabilities have been placed directly as the VAE latent state [Nalisnick and Smyth, 2017]. The IBP prior has been used to learn the number of features in a VAE hidden state using mean-field VI [Chatzis, 2018] with black-box VI [Ranganath et al., 2014] and structured VI [Hoffman and Blei, 2015, Singh et al., 2017]. As an alternative to truncation, Xu et al. [2019] use a Russian roulette sampling scheme to sample from the infinite sum in the stick-breaking process for the IBP. Model

selection for BNNs has been performed with the Horseshoe prior over weights [Ghosh et al., 2019]. The IBP prior has been employed in BNNs to induce sparcity [Panousis et al., 2019] and simultaneously to our work for CL [Kumar et al., 2019]. Both of these approaches apply the IBP prior differently to our work and previous work applying the IBP to VAEs. Also Kumar et al. [2019] deviates from sequential Bayes by storing masks over weights for each task and uses design choices which mean the IBP is not the sole means of selecting weights, we expand on this in Section 7. Instead of using an IBP prior, Bernoulli distributions (or its Concrete relaxation) is used to select the width and number of layers in a BNN [Dikov et al., 2019]. This approach is not non-parametric and so not as desirable for CL. Recently and subsequently to our work, Mehta et al. [2020] have proposed a CL approach where weights matrices are factorised and the IBP prior applied to the diagonal in the factorisation. This work does not use sequential Bayes for CL but rather different neural network weights are used for different tasks and thus alleviates forgetting.

Bayesian continual learning. Repeated application of Bayes' rule can be used to update a model given the arrival of a new task. Previous work has used Laplace approximations [Kirkpatrick et al., 2017] and variational inference [Nguyen et al., 2018]. Bayesian methods can also be intuitively thought of as a weight space regularisation. Explicit regularisation in weight space have also proved successful in CL [Zenke et al., 2017, Schwarz et al., 2018]. Our method builds upon Nguyen et al. [2018] as the framework for learning continually. None of these works deal with the issue of resource allocation to alleviate potential overfitting or underfitting problems in CL. The model we present adapts its size for CL, the scenario where BNNs need adapt to a changing data distribution or to *concept drift* in CL has been studied too [Kurle et al., 2020].

Adaptive models in continual learning. Non-Bayesian CL approaches use additional neural resources to learn new tasks and remember previous tasks. One approach boils down to learning individual networks for each task [Rusu et al., 2016]. More efficient use of resources can be done by selective retraining of neurons and expansion with a group sparsity regulariser [Yoon et al., 2018]. However this approach is unable to shrink and continues to expand if it overfits on the first task. Another approach uses reinforcement learning by adding neural resources by penalising the complexity of the task network in the reward function [Xu and Zhu, 2018]. Recently Rao et al. [2019] propose an unsupervised CL model (CURL) in scenarios that lack a task identifier. CURL is adaptive, insofar that if a new task is detected then a new component is added to the mixture of Gaussians in the model. Task learning in CL can be modeled as a mixture of expert models. The experts are distributed according to a Dirichlet prior [Lee et al., 2020]; new experts can be added to the mixture automatically.

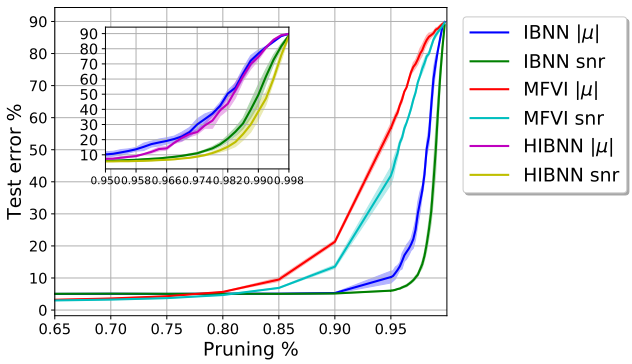

Figure 2: The weight pruning curves show test error versus the percentage of weights which have been zeroed out according to the magnitude of the variational mean and snr ($|\mu|/\sigma$). The HIBNN and IBNN are much sparser than a MFVI BNN and the HIBNN is more robust to pruning and therefore sparser than the IBNN.

5 EXPERIMENTAL RESULTS

To demonstrate the effectiveness of the IBP and H-IBP priors on determining the size of the BNN, we perform weight pruning to see whether the pruned weights coincide with the weights dropped by the IBP and H-IBP priors in Section 5.1. Furthermore, we then use the IBNN and HIBNN in a CL setting in Section 5.2. Also in the supplementary material, further continual learning results and all experimental details are outlined. Unless explicitly stated, all curves are an average of 5 independent runs $\pm$ one standard error. By test error, we refer to $(1 - \text{accuracy}) \times 100$.[3]

5.1 IBP INDUCES SPARSITY

We perform weight pruning to see whether the IBP posterior sensibly selects neurons through the binary matrix Z. Weights are pruned in two ways. The first is pruning according to $|\mu|$: zeroing out weights according to the magnitude of their mean. Important weights will be large in absolute value and so pruned last. Secondly, according to signal to noise ratio: $|\mu|/\sigma$ (snr). Weights with high uncertainty will also be zeroed out first. Weight pruning is performed on MNIST and compared to a mean-field BNN [Blundell et al., 2015] (denoted MFVI in plots). The pruning accuracies in Figure 2 demonstrate that the HIBNN is indeed much sparser than a BNN and that pruning according to snr is more robust, as expected. The HIBNN is more robust to pruning than the IBNN due to its inductive bias leading to the more regular structure Figure 3. The baseline BNN

[3]Our code is available at `https://github.com/skezle/IBP_BNN`.

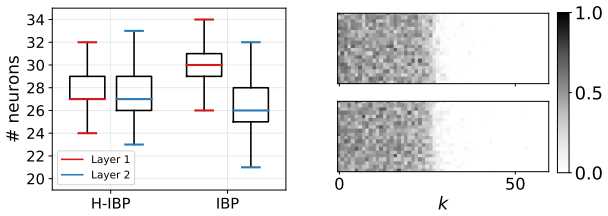

Figure 3: The number of active neurons in each layer of the IBNN and HIBNN. The HIBNN introduces an inductive bias which encourages and enables a regular structure. The Z matrices for the HIBNN show the neurons which are being learnt on the left.[4]

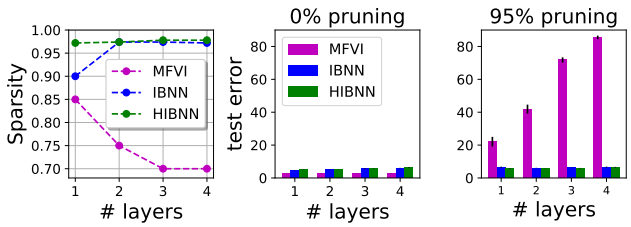

Figure 4: **Left**, as the depth of the IBNN and HIBNN increases the networks tend to remain very sparse while the MFVI BNN is becomes less sparse. **Middle & Right**, the HIBNN and IBNN remain robust to pruning with increasing depth.

has two layers with hidden state sizes of 200, the HIBNN and IBNN use a variational truncation of $K = 200$ for fair comparison. The HIBNN and IBNN achieves an accuracy of 0.95 before pruning while MFVI achieves 0.98 before pruning. This gap in performance is due to the approximate inference of the H-IBP and IBP posteriors and the various reparameterisations used, in particular the Concrete reparameterisation which is applying a 'soft' mask on the hidden layers of the HIBNN. The IBNN and HIBNN are slightly less sparse compared to Sparse Variational Dropout which is specifically designed to be sparse [Molchanov et al., 2017], see Section 2 in the supplement.

Varying depth. As we increase the depth we see that the HIBNN and IBNN remain sparse while the MFVI's sparsity decreases, Figure 4. We measure the sparsity as the pruning percentage at which the accuracy drops over 10% (the kinks in Figure 4). There is little variation of the accuracy with depth for all models before pruning, however after pruning 95% of the weights with the snr our models retain their performance while the MFVI BNN performs worse with depth since it becomes less sparse with depth Figure 4. See the supplement for the same analysis on fashion-MNIST, Section 3.2.

5.2 CONTINUAL LEARNING EXPERIMENTS

Adaptive complexity. Approximate inference of the IBP and H-IBP posteriors is challenging in a stationary setting making the performance attenuated in comparison to a BNN with only independent Gaussian weights. Despite this, the approximate IBP and H-IBP posteriors are useful in non-stationary CL setting, where the amount of resources are unknown beforehand. In Figure 5, one can see that the average accuracies across all CL tasks for permuted MNIST vary considerably with the hidden state size for VCL hence the benefit of our model which automatically infers the hidden state size for each task, see Figure 6. The details of the experiment will be introduced below.

Continual learning scenarios. Three different CL scenarios are used for evaluation [van de Ven and Tolias, 2018]. The first is *task incremental learning* (`CL1`) where the task identifier is given during evaluation. The second is *domain incremental learning* (`CL2`), the task identifier needs to be inferred at test time. The domain increases with each new task and the models are required to perform binary classification. The third is *incremental class learning* (`CL3`), the task identifier and specific class need to be inferred at test time. The models are required to do multi-class classification for each task. During training, the task identifiers are given for all scenarios.[5] Using a multi-head architecture and the predictive entropy we can infer the task; the head with the lowest predictive entropy is chosen for `CL2` and `CL3` [Von Oswald et al., 2020]. For Permuted MNIST `CL2` a single-head architecture is used [van de Ven and Tolias, 2018].

Baseline models. We compare our models to VCL [Nguyen et al., 2018], since the IBNN and HIBNN models build on top of it. We also compare to EWC [Kirkpatrick et al., 2017] and SI [Zenke et al., 2017]. We compare to DEN [Yoon et al., 2018] which is an expansion method which expands a neural network by a fixed number of weights for each new task, uses regularisation to mitigate overfitting and freezes weights from previous tasks. For DEN the predictive entropy is used for inferring the correct head for `CL2` and `CL3`. Another baseline used is GEM [Lopez-Paz and Ranzato, 2017], which uses replay as its primary mechanism for alleviating forgetting. We hypothesise that GEM will be insensitive to model size. It also achieves strong results [Hsu et al., 2018]. VCL is the fairest comparison for our model and will also utilise uncertainties for `CL2` and `CL3`. Our objective is to demonstrate that limited or excessive neural resources can cause problems in CL in comparison

[4] Z is usually shown in *left-ordered form*, however since the inference procedure is based on the stick breaking construction, order is meaningful and sorted according to k [Xu et al., 2019].

[5] *Task-free CL* is a more challenging scenario and requires the model the infer new tasks during training.

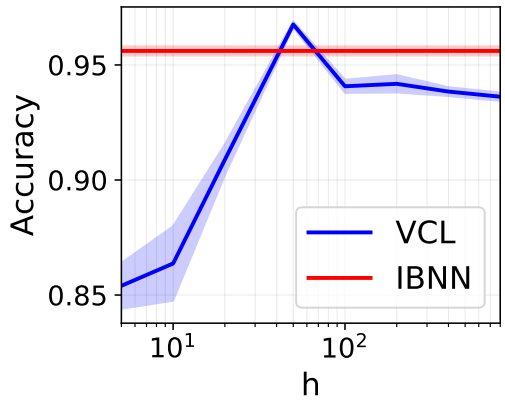

Figure 5: Test accuracies for Permuted MNIST `CL1`, for different VCL widths. Our model yields good results without having to specify a width, only a small range of values outperforms our model.

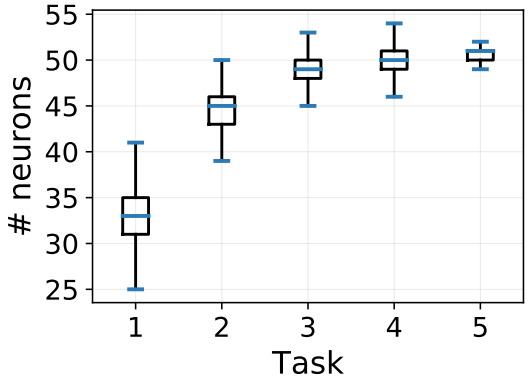

Figure 6: Number of active neurons[7] selected by the IBP variational posterior for each task, for permuted MNIST `CL1`. Our model is able adapt and manage resources effectively.

to our adaptive model.

MNIST Experimental details. All models use a single layer with varying hidden state sizes. The use of a single layer is enough as MNIST is a simple task. The results for EWC in Table 1 on Split MNIST outperform those presented in Hsu et al. [2018], van de Ven and Tolias [2018] which use larger models. We report accuracies for our non adaptive baselines (EWC, SI, GEM and VCL) over a set of hidden state sizes $\mathscr{H} = \{10, 50, 100, 400\}$. A hidden state of 10 might seem small but we also set the IBP prior parameter $\alpha = 5$ for task 1. This corresponds to only 5 neurons being selected by each data point in expectation. The initial hidden state size for DEN is set to 50.

Permuted MNIST benchmarks. The permuted MNIST benchmark involves performing multiclass classification on MNIST where in each task, the pixels have been shuffled by a fixed permutation. Our model is able to overcome overfitting and underfitting which result in increased forgetting which affect VCL. See Figure 8 in the supplement, for a per task accuracy breakdown. In contrast, the IBNN expands continuously Figure 6. There is a small gap in performance between our model and VCL for $h = 50$ due to the approximations used for inference of the variational IBP posterior. Our method outperforms all regularisation based methods and DEN[6] on all CL scenarios and provides comparable results to GEM, see Table 1.

Split MNIST benchmarks. The split MNIST benchmark for CL involves a sequence of classification tasks of MNIST and more difficult variants with background noise and background images denoted S+ε and S+img. For EWC, SI and VCL notice a considerable difference in performance with hidden state size in Table 1. GEM is sensitive for `CL3` only. EWC and SI perform well for `CL1` only. The IBNN outperforms VCL as it is not susceptible to overfitting or underfitting and thus propagating a subsequent poor posterior for a new task resulting in forgetting.

Split MNIST is a simple task which doesn't show overfitting, hence the use of the MNIST variant datasets where the IBNN outperforms all VCL models of different sizes as it is not susceptible to overfitting or underfitting. Indeed our method outperforms not only VCL but EWC and SI and performs comparably to GEM. When analysing the performance of the VCL baselines, we notice they have a tendency to overfit on the second task and propagate a poor approximate posterior and hence underperform in comparison to the IBNN model, Figures 6 and 9 in the supplement. The IBNN increases its capacity over the course of CL, Figure 7 in the supplement. The standard errors for VCL and our method on `CL3` are large due to the severity of making mistakes for multiclass classification.

DEN performs well on all Split MNIST tasks and variants due to its "time-stamped inference" which segregates parts of the network per task and so uncertainties over seen tasks are well defined thus the good results for `CL2` and `CL3`. Indeed removing it renders performance comparable to IBNN, see Section 4 in the supplement. Statistical processes which mimic this could be an interesting direction for Bayesian expansion methods in CL.

Increasing task complexity. To test the expansion capabilities of our models we devise a set tasks of increasing difficulty: two tasks from MNIST, followed by two from fashion MNIST, followed by two from CIFAR10. We compare the HIBNN and IBNN models to VCL with two lay-

[6] It is not clear how to reconcile single-head networks and DEN, thus `CL2` for DEN is omitted.

[7] We define a neuron as active by aggregating all neurons where $z_{ik} > 0.1$ for data point $\mathbf{x_i}$ and neuron k.

Table 1: Average test accuracies on MNIST and variants over 5 runs. For EWC, SI, GEM and VCL the median accuracy is taken from hidden state sizes, $\mathscr{H} = \{10, 50, 100, 400\}$. We also show the range between max and min average accuracies in $\mathscr{H}$. The models with the best median or mean accuracy are highlighted. If the IBNN mean accuracy lies within the min-max range then our model is also highlighted. Our IBNN achieves good performance overall compared to the baselines which can underfit/overfit. DEN performs very poorly on permuted MNIST.

	EWC (max, min)	SI (max, min)	GEM (max, min)	DEN	VCL (max, min)	IBNN
P CL1	90.0 (94.0, 79.1)	91.8 (95.1, 82.9)	94.8 (95.9, 88.4)	91.4±0.5	93.9 (96.8, 86.9)	**95.6±0.2**
P CL2	88.2 (93.0, 78.2)	89.2 (93.3, 84.1)	**95.6** (**96.7**, **88.1**)	-	88.7 (95.1, 75.2)	**93.7±0.6**
P CL3	59.3 (66.9, 24.0)	47.5 (55.4, 18.1)	**94.6** (**95.8**, **87.3**)	63.9±19.2	84.0 (94.1, 40.5)	**93.8±0.3**
S CL1	98.9 (99.0, 94.8)	98.1 (99.2, 95.5)	98.1 (98.2, 98.0)	**99.1±0.1**	96.6 (98.0, 93.7)	95.3±2.0
S CL2	63.7 (74.8, 63.3)	78.0 (80.1, 72.9)	94.0 (94.8, 92.2)	**98.9±0.1**	87.4 (94.9, 81.9)	91.0±2.2
S CL3	19.9 (21.8, 19.8)	18.8 (19.6, 15.4)	89.2 (89.6, 87.8)	**99.1±0.1**	69.0 (78.9, 66.3)	85.5±3.2
S+ε CL1	94.6 (96.5, 81.8)	88.4 (91.1, 72.7)	95.6 (95.7, 95.1)	**97.2±0.2**	89.2 (90.3, 86.3)	95.1±1.1
S+ε CL2	72.7 (74.6, 68.7)	66.1 (72.1, 61.8)	78.2 (78.7, 77.2)	84.8±16.7	69.1 (70.2, 61.7)	**89.7±3.8**
S+ε CL3	18.8 (19.0, 13.7)	13.7 (16.7, 10.4)	74.1 (74.4, 70.7)	**90.9±12.8**	32.5 (37.8, 30.0)	78.7±11.7
S+img CL1	88.2 (90.0, 78.1)	80.8 (87.4, 67.8)	91.7 (91.8, 91.1)	**93.8±0.8**	87.1 (87.9, 85.7)	91.6±1.2
S+img CL2	67.7 (73.9, 62.9)	63.3 (67.0, 58.5)	**74.1** (**76.4**, **74.0**)	**79.1±13.1**	70.4 (75.2, 65.3)	**80.5±7.8**
S+img CL3	17.0 (18.0, 12.0)	13.4 (16.4, 10.3)	63.9 (64.5, 56.3)	**91.6±5.1**	54.3 (66.1, 39.9)	66.2±13.4

Table 2: Accuracies for tasks of increasing difficulty. Accuracies are an average over 5 runs. VCL use different hidden state sizes, $\mathscr{H} = \{50, 100, 400\}$, we show the range between the max and min average accuracies centre at the median. Our models are able to overcome underfitting.

	VCL (max, min)	IBNN	HIBNN
CL1	**82.3** (**86.1**, **81.2**)	80.8±2.0	**81.3±1.8**
CL2	79.9 (83.1, 76.7)	78.7±1.5	**81.5±1.4**

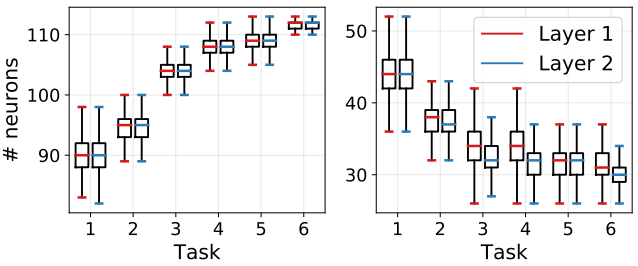

Figure 7: Inferred structure for the HIBNN and IBNN respectively. Our models can infer varied structures, expand and contract.

ers and widths in $\mathscr{H} = \{50, 100, 400\}$. Our models have two layers with $K = 200$. The larger width VCL networks perform well but smaller ones exhibit forgetting due to underfitting. For the HIBNN we allow the hyperparameter α_j to increase for each new dataset seen i.e. every two tasks. We perform random search over the IBP and H-IBP parameters for the IBNN and HIBNN models, Section 6.1. The HIBNN performs better than the IBNN, additionally we can see Figure 7 that both of our models can have very different structure after learning on a task. Notice that since there is a sharing of parameters at a global level that the widths of the HIBNN match across different layers unlike the IBNN.

6 CONCLUSION AND DISCUSSION

Model size is an important contributing factor for CL performance. Most CL methods assume a perfectly selected model. Our novel Bayesian CL framework nonparametrically adapts the complexity of a BNN to the task difficulty. Our model is based on the IBP prior for selecting the number of neurons for each task and uses stochastic variational inference. The models presented reconcile two different approaches to CL: Bayesian or regularization based approaches and dynamic architecture approaches through the use of a IBP and H-IBP prior.

We have demonstrated our model on MNIST like datasets. Future work will focus on showing that our method is able to scale to larger vision datasets which are used in CL. This will involve applying the IBP and H-IBP priors to Convolutional Neural Networks (CNN). Also ensuring that inference can be performed efficiently will be another challenge since inference in the IBNN and HIBNN is more expensive than for VCL (see Section 6.3) and VCL has been shown to produce poor results when scaling to these larger models [Pan et al., 2020]. This could be performed with other inference methods such as using natural gradients [Osawa et al., 2019] or only training certain parts of the convolutional block of the CNN [Ovadia et al., 2019]. This is left for future work.

Acknowledgements

The authors would like to than the anonymous reviewers for the helpful comments. The authors would also like to thank Sebastian Farquhar for helpful discussions and Kyriakos Polymenakos for proof reading a draft of the paper. Thanks for to all the great open source software which made this research possible `Numpy` and `scipy` [Virtanen et al., 2020], `matplotlib` [Hunter, 2007] `Tensorflow` [Abadi et al., 2016] and `Tensorflow Distributions` [Dillon et al., 2017].

References

Mart\'\in Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A System for Large-scale Machine Learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Berkeley, CA, USA, 2016. USENIX Association. ISBN 978-1-931971-33-1.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Networks. In *International Conference on Machine Learning*, 2015.

Sotirios P. Chatzis. Indian Buffet Process deep generative models for semi-supervised classification. Technical report, 2018.

Georgi Dikov, Patrick van der Smagt, and Justin Bayer. Bayesian Learning of Neural Network Architectures. In *AISTATS*, 2019.

Joshua V Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A Saurous. TensorFlow Distributions. 2017.

Finale Doshi, Kurt Miller, Jurgen Van Gael, and Yee Whye Teh. Variational inference for the Indian Buffet Process. In *Artificial Intelligence and Statistics*, pages 137–144, 2009.

Finale Doshi-Velez, Kurt T Miller, Jurgen Van Gael, Yee Whye Teh, Jurgen Van, Gael * Yee, and Whye Teh. Variational Inference for the Indian Buffet Process. Technical report, 2009.

Michael Figurnov, Shakir Mohamed, and Andriy Mnih. Implicit Reparameterization Gradients. In *Neural Information Processing Systems*, 2018.

Soumya Ghosh, Jiayu Yao, and Finale Doshi-Velez. Model Selection in Bayesian Neural Networks via Horseshoe Priors. *Journal of Machine Learning Research*, 20(182): 1–46, 2019.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. 2015.

Thomas L Griffiths and Zoubin Ghahramani. The Indian Buffet Process: An Introduction and Review. *Journal of Machine Learning Research*, 12:1185–1224, 2011.

Sunil Kumar Gupta, Dinh Phung, and Svetha Venkatesh. A Slice Sampler for Restricted Hierarchical Beta Process with Applications to Shared Subspace Learning. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pages 316–325, 2012.

Matthew D Hoffman and David M Blei. Structured Stochastic Variational Inference. In *International Conference on Artificial Intelligence and Statistics*, 2015.

Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating Continual Learning Scenarios: A Categorization and Case for Strong Baselines. In *Continual Learning Workshop, 32nd Conference on Neural Information Processing Systems*, 2018.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparametrization with Gumbel-Softmax. In *International Conference on Learning Representations*, 2017.

Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014.

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

Abhishek Kumar, Sunabha Chatterjee, and Piyush Rai. Bayesian structure adaptation for continual learning. *arXiv preprint arXiv:1912.03624*, 2019.

Richard Kurle, Botond Cseke, Alexej Klushyn, Patrick Van Der Smagt, and Stephan Günnemann. Continual Learning with Bayesian Neural Networks for Non-Stationary Data. In *ICLR*, 2020.

Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A Neural Dirichlet Process Mixture Model for Task-Free Continual Learning. In *International Conference on Learning Representations*, 2020.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient Episodic Memory for Continual Learning. In *Neural Information Processing Systems*, 2017.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: a Continual Relaxation of Discrete Random Variables. In *International Conference on Learning Representations*, 2017.

Nikhil Mehta, Kevin J Liang, and Lawrence Carin. Bayesian Nonparametric Weight Factorization for Continual Learning. Technical report, 2020.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational Dropout Sparsifies Deep Neural Networks. In *International Conference on Machine Learning*, 2017.

Eric Nalisnick and Padhraic Smyth. Stick-Breaking Variational Autoencoders. In *International Conference on Learning Representations*, 2017.

Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational Continual Learning. In *International Conference on Learning Representations*, 2018.

Kazuki Osawa, Siddharth Swaroop, Anirudh Jain, Runa Eschenhagen, Richard E Turner, Rio Yokota, and Mohammad Emtiyaz Khan. Practical Deep Learning with Bayesian Principles. In *Neural Information Processing Systems*, 2019.

Yaniv Ovadia, Emily Fertig, Google Research, Jie Ren, Zachary Nado Google Research, Sebastian Nowozin Google Research, Joshua V Dillon Google Research, Balaji Lakshminarayanan, and Jasper Snoek. Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift. In *Neurips*, 2019.

Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard E Turner, and Mohammad Emtiyaz Khan. Continual Deep Learning by Functional Regularisation of Memorable Past. In *Neurips*, 2020. ISBN 2004.14070v4.

Konstantinos P. Panousis, Sotirios Chatzis, and Sergios Theodoridis. Nonparametric Bayesian deep networks with local competition. *36th International Conference on Machine Learning, ICML 2019*, 2019-June(May):8772–8783, 2019.

Rajesh Ranganath, Sean Gerrish, and David M Blei. Black Box Variational Inference. In *Artificial Intelligence and Statistics*, 2014.

Dushyant Rao, Francesco Visin, Andrei A. Rusu, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Continual Unsupervised Representation Learning. In *Neural Information Processing Systems*, 2019.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks. In *arXiv:1606.04671v3*, 2016. ISBN 1606.04671v3.

Jonathan Schwarz, Jelena Luketina, Wojciech M Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress and Compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, 2018.

Rachit Singh, Jeffrey Ling, and Finale Doshi-Velez. Structured Variational Autoencoders for the Beta-Bernoulli Process. In *Workshop on Advances in Approximate Bayesian Inference, Neural Information Processing Systems*, 2017.

Yee Whye Teh, Dilan Grür, and Zoubin Ghahramani. Stick-breaking Construction for the Indian Buffet Process. In *International Conference on Artificial Intelligence and Statistics*, 2007.

Romain Thibaux and Michael I Jordan. Hierarchical Beta processes and the Indian buffet process. In *Artificial Intelligence and Statistics*, pages 564–571, 2007.

Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning. In *NeurIPS Continual Learning workshop*, 2018.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Johannes Von Oswald, Christian Henning, João Sacramento, and Benjamin F Grewe. Continual Learning with Hypernetworks. In *International Conference on Learning Representations*, 2020.

Ju Xu and Zhanxing Zhu. Reinforced Continual Learning. In *Neural Information Processing Systems*, 2018.

Kai Xu, Akash Srivastava, and Charles Sutton. Variational Russian Roulette for Deep Bayesian Nonparametrics. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong Learning with Dynamically Expandable Networks. In *International Conference on Learning Representations*, 2018.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual Learning Through Synaptic Intelligence. In *International Conference on Machine Learning*, 2017.