# Indian Buffet Neural Networks for Continual Learning

**Samuel Kessler**[1,2], **Vu Nguyen**[1], **Stefan Zohren**[1,2], **Stephen Roberts**[1,2]
[1]Machine Learning Research Group,
[2]Oxford-Man Institute of Quantitative Finance,
University of Oxford
{skessler, vu, zohren, sjrob}@robots.ox.ac.uk

## Abstract

We place an Indian Buffet Process (IBP) prior over the neural structure of a Bayesian Neural Network (BNN), thus allowing the complexity of the BNN to increase and decrease automatically. We apply this methodology to the problem of resource allocation in continual learning, where new tasks occur and the network requires extra resources. Our BNN exploits online variational inference with relaxations to the Bernoulli and Beta distributions (which constitute the IBP prior), so allowing the use of the reparameterisation trick to learn variational posteriors via gradient-based methods. As we automatically learn the number of weights in the BNN, overfitting and underfitting problems are largely overcome. We show empirically that the method offers competitive results compared to Variational Continual Learning (VCL) in some settings.

## 1 Introduction

In continual learning a model is required to learn a set of tasks, one by one, and to remember solutions to each. After learning a task, the model loses access to the data [1–3]. More formally, in such continual learning problems we have a set of $M$ sequential prediction tasks $\mathcal{T}_{i=1}^{M}$ where $\mathcal{D}_1 = \left\{(x_i, y_i)_{i=1}^{N_1}\right\} \in \mathcal{T}_1$, $\mathcal{D}_2 = \left\{(x_i, y_i)_{i=N_1+1}^{N_2}\right\} \in \mathcal{T}_2$, ..., $\mathcal{D}_M = \left\{(x_i, y_i)_{i=N_{M-1}+1}^{N_M}\right\} \in \mathcal{T}_M$. When performing task $\mathcal{T}_t$ the learner typically loses access to $\mathcal{D}_{<t}$, yet must be able to continue to perform predictions for all the tasks $\mathcal{T}_{\leq t}$ [4].

The core challenges of continual learning are threefold. Firstly, models need to leverage transfer learning from previously learned tasks during the learning of a new task at time $t$ [5, 1, 6, 7]. Secondly, the model needs to have enough new neural resources available to learn the new task [6, 3, 8, 1]. Finally, the model is required to overcome *catastrophic forgetting* of old tasks. If the model, for example, is a feed-forward neural network it will exhibit forgetting of previous tasks [1, 9].

One of the popular ways to perform continual learning uses the natural sequential learning approach embedded within Bayesian inference, namely that the prior for task $\mathcal{T}_t$ is the posterior obtained from the previous task. This enables knowledge transfer and offers an approach to overcome catastrophic forgetting. Previous Bayesian approaches have involved Laplace approximations [1, 10, 7] and variational inference [2, 11, 4], to aid in computational tractability. Whilst such methods solve, in principle, the first and third objectives of continual learning, the second objective (that of ensuring adequate resources for new learning) is not necessarily achieved. For example, additional neural resources can alter performance on MNIST classification (see Table 1 in [8]). The problem is made more difficult as neural resources required for a good solution for one task might not be sufficient (or may be redundant) for a different task.

4th workshop on Bayesian Deep Learning (NeurIPS 2019), Vancouver, Canada.

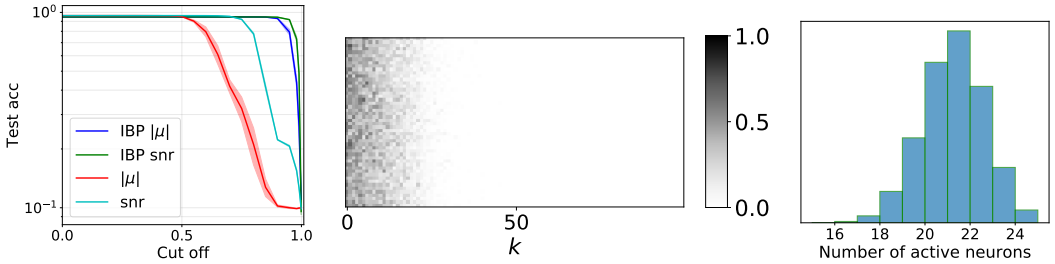



Figure 1: **Left**, Comparison of weight pruning for the IBP BNN on MNIST and a comparison with a BNN with no IBP prior. We prune weights of each network according to the absolute value of the weights and the signal to noise ratio $|\mu|/\sigma$ and apply the 'binary' $z_{nk}$ mask to activation outputs from the IBP prior BNN. These curves are an average of 5 separate optimisations $\pm$ one standard error. **Middle**, the matrix $Z$ for a batch in the test set to demonstrate which neurons are active. Notice that $Z$ is not perfectly binary as it has been relaxed with a Concrete distribution. **Right**, a histogram showing the number of active neurons for each point in the test set.

Non-Bayesian neural networks use additional neural resources to remember previous tasks and learn a new task. Neurons 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 [6]. The problem with this approach is that of scalability: the number of neural resources increases linearly with the number of tasks. The work of [3] tackles this problem with selective retraining and expansion with a suitable regulariser to ensure that the network does not expand continuously. However, these expandable networks are unable to shrink and are vulnerable to overfitting if misspecified to begin with. Moreover, knowledge transfer and the prevention of catastrophic forgetting are not solved in a principled manner, unlike approaches couched in a Bayesian framework. We summarise several solutions to the general problem in section B in the appendix.

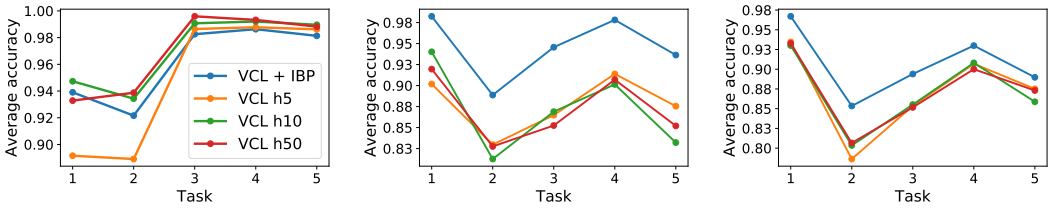
As the level of resource required is unknown in advance, we propose a Bayesian neural network which adds or withdraws neural resources automatically, in response to the data. We achieve this by using a sparse binary latent matrix $Z$, distributed according to a structured Indian Buffet Process (IBP) prior. The IBP prior on an infinite binary matrix, $Z$, allows inference on which, and how many, neurons are required for a predictive task. The weights of the BNN are treated as non-interacting draws from Gaussians [8]. Catastrophic forgetting is overcome by repeated application of the Bayesian update rule, embedded within variational inference [12, 2]. In the next section we detail the model and in Section 3 we provide representative results.

## 2 Expandable Bayesian Neural Network with an Indian Buffet Process prior

We start by considering the matrix factorisation problem $X = ZA$ where $X \in \mathbb{R}^{N\times D}$, $Z \in \mathbb{Z}_2^{N\times K}$ and $A \in \mathbb{R}^{K\times D}$. Each column of $Z$, a binary matrix, corresponds to the presence of a latent feature from $A$. With $z_{nk} = 1$, the latent feature $k$ is present in observation $X_n$. In the scenario where we do not know the number of features $K$ beforehand and we desire a prior that allows the number of non-zero columns of $Z$ to be inferred then the IBP provides a suitable prior on $Z$ [12].

In our proposed model we use $Z$ distributed according to a nonparametric IBP prior, which induces a posterior to select neurons and their number. We consider a neural network with $k_l$ neurons in each of its $l = \{1, \cdots L\}$ layers. Thence, for an arbitrary activation $f$, the binary matrix is applied as follows: $h_l = f(h_{l-1}W_l) \circ Z_l$ where $h_{l-1} \in \mathbb{R}^{N\times k_{l-1}}$, $W_l \in \mathbb{R}^{k_{k-1}\times k_l}$, $Z_l \in \mathbb{Z}_2^{N\times k_l}$, and $\circ$ is the elementwise product. We have ignored biases for simplicity. The IBP has some nice properties for this application, including the number of elements sampled growing with $N$ and promoting a "richer get richer" scheme [13]. Hence the number of neurons which are selected grows with the number of points in the dataset and the same neurons will be selected by the IBP enabling learning. This neuron selection scheme is in contrast to dropout which randomly selects weights.

We use a stick-breaking IBP prior [14, 12], in which a probability $\pi_k \in [0, 1]$ is assigned to the column $Z_k$. Whether a neuron is active for data point $X_n$ is determined by $z_{nk} \sim \text{Bern}(\pi_k)$. Here $\pi_k$ is generated via the so-called stick-breaking process: $v_k \sim \text{Beta}(\alpha, 1)$ and $\pi_k = \prod_{i=1}^{k} v_i$. As a result,




(a) MNIST (b) MNIST + noise (c) MNIST + background image

Figure 2: Continual learning average accuracies on task specific test sets for different datasets and for different models. The BNN with a IBP prior is compared to baselines BNNs with no IBP prior and with fixed hidden state sizes $h \in \{5, 10, 50\}$. The accuracies reported are an average of 5 different optimisations. Break-downs of task accuracies versus the number of tasks the model has seen are available in the appendix E.

$\pi_k$ decreases exponentially with $k$. The Beta parameter $\alpha$ controls how quickly the probabilities $\pi_k$ decay. By learning the Beta parameters we can influence how many neurons are required for a particular layer and for a particular task.

Our expandable BNN has diagonal Gaussian weights, $w_{ij} \sim \mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$ for all $i$, $j$ as in [8, 2] and the binary $Z$ matrices will follow the IBP prior. For continual learning the posterior over the BNN weights and IBP parameters will form the prior for the new task. In practice, we will use a variational approximation, where the variational posterior from task $\mathcal{T}_k$ is taken as prior for $\mathcal{T}_{k+1}$. This will encourage knowledge transfer and prevent catastrophic forgetting. The parameter $\alpha$ of the IBP prior controls the number of neurons available for task $\mathcal{T}_t$, as it increases (or decreases) this should encourage the use of more (or less) neurons and hence add (or remove) new computational resources for learning the new task $\mathcal{T}_{t+1}$.

The posterior of our model given the data is approximated using structured stochastic variational inference [15]. The variational Beta parameters act globally over $Z$, thus the variational approximation we propose here retains some of the structure of the desired posterior. We make use of the reparameterisation trick [16] together with the Concrete reparamaterisation of the Bernoulli distribution [17, 18] and the Kumaraswamy reparameterisation of the Beta distribution [19, 20] to allow stochastic gradients to pass through to the Beta parameters in the hierarchical IBP posterior. The model is discussed in more detail in section C.

## 3 Results

We investigate whether the neural sparsity imposed by the IBP prior is sensible. This is done by weight pruning on the MNIST multi-class classification problem. We compare our approach with a variational BNN which has the same neural network architecture except without the IBP prior which commands the structure and number of neurons in a layer. The IBP BNN has an accuracy of 0.95 while the BNN achieved an accuracy of 0.96, however the IBP prior BNN is more robust to pruning; with pruned weights coincide with those suppressed by the IBP prior. The pruning accuracy is shown in Figure 1. There is a small improvement in the accuracies by pruning with the signal-to-noise ratio (snr), defined as $(|\mu_{ki}|/\sigma_{ki}) \circ z_{nk}$ for all $i$. This is expected as MNIST is a relatively simple problem with good accuracy even on small networks. Note that the sparsity induced by the IBP prior renders the effects of variational overpruning redundant [21]. Overall, the above results show a sensible sparsity induced by the variational IBP.

Our main experiments deal with the task of continual learning on various split MNIST datasets. For a fair comparison, we use multi-head network architectures for all experiments. The baselines are VCL networks [2] with a single hidden layer with size $h \in \{5, 10, 50\}$. The sizes are chosen to expose potential underfitting or overfitting issues. Our model also uses a single layer with a variational truncation $K = 100$. The IBP prior BNN outperforms all VCL baseline networks for the split MNIST tasks which have background noise or background images as shown Figure 2 while having less than 15 active neurons see Figures 4 and 5. The baseline models overfit on the second task and subsequently propagate a poor approximate posterior. On split MNIST the $h = 5$ baseline underfits and the $h \in \{10, 50\}$ perform well versus our model as this is a simple task and overfitting is difficult

to expose, see Figure 2a. Continual learning experiments on the not MNIST dataset shows that the baselines with $h \in \{5, 10\}$ underfit, but the $h = 50$ performs better than our model on some tasks, see Figure 6. For all the datasets considered in the continual learning experiments, the BNN with an IBP prior is able to expand as the model is required to solve more tasks, see Figure 3b to Figure 6b. Further analysis of the results is presented in the supplementary material in section E and additional experimental details are presented in section D.

## 4 Summary

We introduce a structured IBP prior over a Bayesian Neural Network (BNN), with application to continual learning. The IBP prior effectively induces sparsity in the network, allowing it to add neural resources for new tasks yet still overcome the overfitting problems which plague VCL networks. Our goal is continual learning and not to induce sparsity for a parsimonious model or for the sake of compression, however it would be interesting to compare to BNNs designed for these goals by introduction of sparsity inducing priors [22, 23]. Natural extensions to this work include the application of the IBP prior directly to the BNN weights as well as more extensive testing with a broader range of data-sets, with larger numbers of tasks.

## References


[1] 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.

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

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

[4] Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. Task Agnostic Continual Learning Using Online Variational Bayes. In *Bayesian Deep Learning workshop, Neural Information Processing Systems*, 2018.

[5] Sebastian Thrun. Lifelong Learning: A Case Study. Technical report, 1995.

[6] 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.

[7] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming Catastrophic Forgetting by Incremental Moment Matching. In *Neural Information Processing Systems*, 2017.

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

[9] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. In *arXiv:1312.6211v3*, 2015.

[10] Hippolyt Ritter, Aleksandar Botev, and David Barber. Online Structured Laplace Approximations for Overcoming Catastrophic Forgetting. In *Neural Information Processing Systems*, 2018.

[11] Siddharth Swaroop, Cuong V Nguyen, Thang D Bui, and Richard E Turner. Improving and Understanding Variational Continual Learning. In *Continual Learning workshop, Neural Information Processing Systems*, 2018.

[12] Finale Doshi-Velez, Kurt T Miller, Jurgen Van Gael, and Yee Whye Teh. Variational Inference for the Indian Buffet Process. In *International Conference on Artificial Intelligence and Statistics*, 2009.

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

[14] 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.

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

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

[17] 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.

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

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

[20] 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.

[21] Brian L Trippe and Richard E Turner. Overpruning in Variational Bayesian Neural Networks. In *Advances in Approximate Bayesian Inference workshop, Neural Information Processing Systems*, 2017.

[22] Christos Louizos, Karen Ullrich, and Max Welling. Bayesian Compression for Deep Learning. In *Neural Information Processing Systems*, 2017.

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

[24] Zoubin Ghahramani and Thomas L Griffiths. Infinite latent feature models and the Indian Buffet Process. In *Advances in Neural Information Processing Systems*, pages 475–482, 2006.

[25] Lancelot F James et al. Bayesian Poisson calculus for latent feature modeling via generalized Indian Buffet Process priors. *The Annals of Statistics*, 45(5):2016–2045, 2017.

[26] Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia C Wilson, and Michael I Jordan. Streaming Variational Bayes. In *Neural Information Processing Systems*, 2013.

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

[28] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip H S Torr. Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence. In *European Conference on Computer Vision (ECCV)*, 2018.

[29] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim Sk T-Brain. Continual Learning with Deep Generative Replay. In *Neural Information Processing Systems*, 2017.

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

[31] Sebastian Farquhar and Yarin Gal. A Unifying Bayesian View of Continual Learning. In *Bayesian Deep Learning workshop, Neural Information Processing Systems*, 2018.

[32] 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.

[33] Finale Doshi-Velez and Zoubin Ghahramani. Accelerated sampling for the Indian Buffet Process. In *International Conference on Machine Learning*. ACM, 2009.

[34] Frank Wood and Thomas L Griffiths. Particle filtering for nonparametric Bayesian matrix factorization. In *Neural Information Processing Systems*, 2007.

[35] Yee Whye Teh, Dilan Grür, and Zoubin Ghahramani. Stick-breaking construction for the Indian Buffet Process. In *Artificial Intelligence and Statistics*, pages 556–563, 2007.

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

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

[38] Diederik P Kingma and Jimmy Lei Ba. ADAM: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

# A Preliminaries

In this section, we describe the Indian Buffet Process prior used in our model and the Variational Continual Learning (VCL) framework which we use for continual learning.

## A.1 Indian Buffet Process Prior

The Indian Buffet Process (IBP) [24, 13, 25] is a stochastic process defining a probability distribution over sparse binary matrices with a finite number of rows and an infinite number of columns. This distribution is suitable to use as a prior for models with a potentially infinite number of features. The form of the prior ensures that only a finite number of features will be present in any finite set of observations, but allows for extra features to appear as more data points are observed. The IBP probability density is defined as follows:

$$p(Z) = \frac{\alpha^K}{\prod_{i=1}^{N} K_i!} \exp\{-\alpha H_N\} \prod_{k=1}^{K} \frac{(N-m_k)!(m_k-1)!}{N!} \tag{1}$$

where $K$ is the number of non-zero columns in $Z$, $m_k$ is the number of ones in column $k$ of $Z$, $H_N = \sum_{n=1}^{N} 1/n$ is the $N$-th harmonic number, and $K_h$ is the number of occurrences of the non-zero binary vector $h$ among the columns in $Z$. The parameter $\alpha$ controls the expected number of features present in each observation.

The name of the Indian Buffet Process originates from the metaphor, where the rows of $Z$ correspond to customers and the columns correspond to dishes in an infinitely long buffet. The first customer samples the first $\text{Poisson}(\alpha)$ dishes. The $i$-th customer then samples dishes with probability $m_k/i$, where $m_k$ is the number of people who have already sampled dish $k$. The $i$-th customer also samples $\text{Poisson}(\alpha/i)$ new dishes. Therefore, $z_{nk}$ is one if customer $n$ tried the $k$-th dish and zero otherwise.

## A.2 Variational Continual Learning

The continual learning process can be decomposed into a Bayesian update and approximate inference of the task $\mathcal{T}_{t-1}$ posterior can be used as a prior for the new task $\mathcal{T}_t$. Variational Continual Learning (VCL) [2] uses a BNN to perform a prediction problem where the weights are independent Gaussians and uses the variational posterior from the previous task as the prior for the next. Consider learning the first task $\mathcal{T}_1$ and let $\boldsymbol{\phi}$ denote a vector of parameters, then the variational posterior will be: $q_1(\boldsymbol{\phi}|\mathcal{D}_1)$. For the next task $\mathcal{T}_2$, we lose access to $\mathcal{D}_1$ and the prior will be $q_1(\boldsymbol{\phi}|\mathcal{D}_1)$. The optimisation of the ELBO yields $q_2(\boldsymbol{\phi}|\mathcal{D}_2)$. Generalising, the negative ELBO for the $t$-th task is:

$$\mathcal{L}_t(\boldsymbol{\phi}, \mathcal{D}_t) = \text{KL}\left[q_t(\boldsymbol{\phi})||q_{t-1}(\boldsymbol{\phi}|\mathcal{D}_{t-1})\right] - \mathbb{E}_{\boldsymbol{\phi}\sim q_t(\boldsymbol{\phi})}\left[\log p(\mathcal{D}_t|\boldsymbol{\phi})\right]. \tag{2}$$

The first term acts to regularise the posterior over $\mathcal{T}_t$, ensuring continuity with $\mathcal{T}_{t-1}$ and second term is the log likelihood of the data.

# B Related work

In this section we discuss the literature on Continual Learning and that associated with the use of the IBP prior in Deep Learning.

## B.1 Continual Learning

Continual Learning can be viewed as a sequential learning problem and an approach to learning in this setting is through online Bayesian inference [26]. Elastic Weight Consolidation (EWC) is a seminal piece of work in continual learning which performs online Bayesian inference with a diagonal Laplace approximation to make Bayesian inference tractable [1]. This reduces to an $L^2$ regularisation ensuring that the new weights for task $t$ are close to all previous task weights in terms of Euclidean distance. Synaptic Intelligence (SI) [27] creates an importance measure that is determined by the loss over the optimisation trajectory and by the Euclidean distance it has moved from the previous task's local minimum. SI uses this importance measure to weight an $L^2$ regularisation ensuring that the optimal weights for $\mathcal{T}_t$ are similar to those for $\mathcal{T}_{t-1}$. Another regularisation based approach, one

learns the conditional distribution regularised for task $\mathcal{T}_{t-1}$ so that it is close to that of $\mathcal{T}_t$ in terms of KL-divergence this can also be approximated as an $L^2$ regularisation similarly to EWC and SI [28]. The work of [7] also uses a diagonal approximation to the Fisher information used for the Laplace approximation for Bayesian approximate inference together with techniques from transfer learning literature. Instead of approximating the Fisher information as diagonal and ignoring correlations in parameter space, [10] uses a block-diagonal Kronecker factored approximation which accounts for covariances between weights of the same layer and assumes independence between the weights of different layers. Recent work has also been made on variational approximations to sequential Bayesian inference in continual learning and proposed for discriminative and generative models [2, 11].

Another approach to continual learning is to expand the neural network model to ensure the predictive performance on previous tasks is retained and allowing for new neural capacity for learning of new tasks. One approach is Progressive Networks [6] which freezes weights learnt from previous tasks and connections are made from the frozen networks to a new network which is trained on the current task. This allows the Progressive Network to leverage previous knowledge to remember old tasks and also allows new neural capacity for learning a new task. This solution is linear in the number of networks needed for $T$ tasks. A more efficient expansion approach is to selectively retrain neurons and if required, expand the network with a group sparsity regulariser to ensure sparsity at the neuron level [3].

Several other solutions to continual learning have been proposed, involve replaying data from previous tasks with a generative model trained to reconstruct $\mathcal{D}_i$ for $i < t$ [29], storing summaries of data with coresets [2] or storing random samples from each task [30] and ensuring that loss incurred on this memory dataset is smaller for $\mathcal{T}_t$ than for $\mathcal{T}_{t-1}$. Combining methods also yield good results, these include using VCL and generative replay approaches [31] and using Progressive Networks and EWC to ensure that the number of parameters in the network does not increase with the number of tasks [32].

### B.2 The Indian Buffet Process prior in Deep Learning

The IBP prior has been used for sparse matrix factorisation. The inference for IBP has been performed in several ways, including Gibbs sampling [33, 13], particle filtering [34], slice sampling [35], and using variational inference [12]. For generative models in deep learning the IBP has been used to model the latent state for VAEs, inference has been performed with mean-field variational inference, using black-box variational inference [36] by [37]. The stick-breaking VAE by [19] introduces a suitable reparameterisation to handle gradient based learning with the reparameterisation trick [16]. Because the mean-field approximation removes much of the structure of a hierarchical model like the IBP, [20] uses structured stochastic variational inference [15] to allow dependencies between global and local parameters and achieve better results in VAEs over the mean-field approximation.

## C Inference

In this section we develop a variational approach for performing inference on the posterior distribution of the BNN weights and the IBP parameters. We use a structured variational model where dependencies are established between global Beta parameters over local parameters which comprise the BNN hidden layers [15], similarly to [20]. Once we have obtained our variational posterior, placing our inference procedure within the VCL framework set out in section A.2 is straightforward. The following set of equations govern the hierarchical IBP prior BNN model for an arbitrary layer $l \in \{1, ..., L\}$ of a BNN:

$$v_k \sim \text{Beta}\left(\alpha, 1\right), \qquad \text{for } k \in \{1, \cdots \infty\}, \tag{3}$$

$$\pi_k = \prod_{i=1}^{k} v_i, \qquad \text{for } k \in \{1, \cdots \infty\}, \tag{4}$$

$$z_{nk} \sim \text{Bern}(\pi_k), \qquad \text{for } k \in \{1, \cdots \infty\}, n \in \{1, \cdots N\}, \tag{5}$$

$$W^l_{k.} \sim \mathcal{N}(\mu^l_{k.}, (\sigma^l_{k.})^2), \qquad \text{for } k \in \{1, \cdots \infty\}, \tag{6}$$

$$h^l_k = f(h^{l-1} W^l_{k.}) \circ z^l_{nk} \qquad \text{for } k \in \{1, \cdots \infty\}, n \in \{1, \cdots N\}. \tag{7}$$

$k$ denotes a neuron in layer $l$, denotes a row from the weight matrix $W^l_{k.}$ and identifies a column of our binary matrix. $\circ$ is the elementwise multiplication operation. The binary matrix $Z$ controls the inclusion of a particular neuron $k$, $W^l_{k.} \in \mathbb{R}^{k_{l-1}}$, $h^{l-1} \in \mathbb{R}^{k_{l-1}}$ and $z_{nk} \in \mathbb{Z}_2 = \{0, 1\}$.

The closed form solution to the true posterior of our IBP parameters and BNN weights involves integrating over the joint distribution of the data and our hidden variables, $\boldsymbol{\varphi} = \{\boldsymbol{Z}, \boldsymbol{\pi}, \boldsymbol{W}\}$. Since it is not possible to obtain a closed form solution to this integration we will make use of variational inference and the reparameterisation trick [16]. We use a structured variational approximation [15], this approach has been shown to perform better than the mean-field approximation in VAEs [20]. The variational approximation used is

$$q(\boldsymbol{\phi}) = \prod_{k=1}^{K} q(v_k; \tau_{k_1}, \tau_{k_2}) q(w_{k.}; \omega_{k_1.}, \omega_{k_2.}) \prod_{n=1}^{N} q(z_{nk}; \pi_k | v_k), \tag{8}$$

where the variational posterior is truncated up to $K$, the prior is still infinite [19]. $\boldsymbol{\phi} = \{\boldsymbol{\tau}, \boldsymbol{v}, \boldsymbol{\omega}\}$ denotes the set of variational parameters which we optimise over. Each term in Equation (8) is specified as follows

$$q(v_k; \tau_{k_1}, \tau_{k_2}) = \text{Beta}(v_k; \tau_{k_1}, \tau_{k_2}), \tag{9}$$

$$\pi_k = \prod_{j=1}^{k} v_j, \tag{10}$$

$$q(z_{nk}; \pi_k) = \text{Bern}(z_{nk}; \pi_k), \tag{11}$$

$$q(w_{k.}; \omega_{k_1.}, \omega_{k_2.}) = \mathcal{N}(w_{k.}; \omega_{k_1.}, \omega_{k_2.}). \tag{12}$$

Now that we have defined our structured variational approximation in Equation (8) we can write down the objective for task $\mathcal{T}_t$ as

$$\arg\min_{\boldsymbol{\phi}} \text{KL}\left(q_t(\boldsymbol{\phi}) || p_t(\boldsymbol{\phi} | \mathcal{D}_t)\right) \tag{13}$$

$$= \arg\min_{\boldsymbol{\phi}} \text{KL}\left(q_t(\boldsymbol{\phi}) || q_{t-1}(\boldsymbol{\phi} | \mathcal{D}_{t-1})\right) - \mathbb{E}_{q_t(\boldsymbol{\phi})}[\log p(\mathcal{D}_t | \boldsymbol{\phi})]. \tag{14}$$

In the above formula, $q_t(\boldsymbol{\phi})$ is the approximate posterior for $\mathcal{T}_t$ and $q_{t-1}(\boldsymbol{\phi}|\mathcal{D}_{t-1})$ is the approximate posterior for task $\mathcal{T}_{t-1}$ and prior for $\mathcal{T}_t$. By substituting Equation (8), we obtain the negative ELBO objective for each task $\mathcal{T}_t$ as:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\phi}, \mathcal{D}_t) = &\text{KL}(q_t(\boldsymbol{v}) || q_{t-1}(\boldsymbol{v} | \mathcal{D}_{t-1})) + \text{KL}(q_t(\boldsymbol{w}) || q_{t-1}(\boldsymbol{w} | \mathcal{D}_{t-1})) \\ &- \sum_{n \in \mathcal{D}_t} \mathbb{E}_{q_t(\boldsymbol{\phi})}[\log p(y_n | \boldsymbol{x_n}, \boldsymbol{z_n.})] + \text{KL}(q_t(\boldsymbol{z_n.} | \boldsymbol{v}) || q_{t-1}(\boldsymbol{z_n.} | \boldsymbol{v}, \mathcal{D}_{t-1})).\end{aligned} \tag{15}$$

To estimate the gradient of the Bernoulli and Beta variational parameters requires a suitable reparameterisation. Samples from the Bernoulli distribution in Equation (11) arise after taking an `argmax` over the Bernoulli parameters. The `argmax` is discontinuous and a gradient is not possible to calculate. We reparameterise the Bernoulli as a Concrete distribution [17, 18]. Additionally we reparameterise the Beta as a Kumaraswamy distribution for the same reasons [19]; to separate sampling nodes and parameter nodes in the computation graph (Figure 2 in [18] for clarification) and allow the use of stochastic gradient methods to learn the variational parameters $\boldsymbol{\phi}$ in the approximate IBP posterior. Variational inference on the Gaussian weights of the BNN, $\boldsymbol{\omega}$ in Equation (12) is performed with a mean-field approximation and identical to [8, 2]. In the next sections we detail the reparameterisations of the Bernoulli and Beta distributions and show how to calculate the KL-divergence terms in Equation (15).

### C.1 The variational Gaussian weight distribution reparameterisation

The variational posterior over the weights of the BNN are diagonal Gaussian $w_{k.} \sim \mathcal{N}(w_{k.} | \omega_{k_1.}, \omega_{k_2.} \mathbb{1})$. By using a reparameterisation, one can represent the BNN weights using a deterministic function $w_{k.} = g_\phi(\epsilon)$, where $\epsilon \sim \mathcal{N}(0, \mathbb{1})$ is an auxiliary variable and $g_\phi(.)$ a deterministic function parameterised by $\phi = (\omega_{k_1.}, \omega_{k_2.})$. The BNN weights can be sampled directly through the reparameterisation: $w_{k.} = \omega_{k_1.} + \omega_{k_2.} \epsilon$. By using this simple reparameterisation the weight samples are now deterministic functions of the variational parameters $\omega_{k_1.}$ and $\omega_{k_2.}$ and the

noise comes from the independent auxiliary variable $\epsilon$ [16]. Taking a gradient of our ELBO objective in Equation (15) the expectation of the log-likelihood may be rewritten by integrating over $\epsilon$ so that the gradient with respect to $\omega_{k_1.}$ and $\omega_{k_2.}$ can move into the expectation allowing for gradients to be calculated using the chain rule [8].

### C.2 The variational Beta distribution reparameterisation

The Beta distribution can be reparameterised using the Kumaraswamy distribution [19] with parameters $a$ and $b$. The Kumaraswamy distribution has a density

$$p(x; a, b) = abx^{a-1}(1 - x^a)^{b-1}. \tag{16}$$

When $a = 1$ or $b = 1$ the Kumaraswamy and Beta are identical. This reparameterisation has been used successfully to learn a discrete latent hidden representation in a VAE where the parameters $a$ and $b$ are learnt using stochastic gradient descent [19, 20]. The Kumaraswamy distribution can be reparameterised as

$$p(x; a, b) \sim (1 - u^{1/b})^{1/a}, \tag{17}$$

where $u \sim \mathrm{U}[0, 1]$ from the Uniform distribution.

The KL divergence between our variational Kumaraswamy posterior and Beta prior has a closed form:

$$\mathrm{KL}\left[q(v_k; a, b)||p(v_k; \alpha, \beta)\right] = \frac{a - \alpha}{a}\left(-\gamma - \psi(b) - \frac{1}{b}\right) + \log ab + \log B(\alpha, \beta) \tag{18}$$

$$- \frac{b - 1}{b} + (\beta - 1)b \sum_{m=1}^{\infty} \frac{1}{m + ab} B\left(\frac{m}{a}, b\right), \tag{19}$$

where $\gamma$ is the Euler constant, $\psi$ is the digamma function, $B$ is the beta function and the infinite sum can be approximated by a finite sum.

### C.3 The variational Bernoulli distribution reparameterisation

The Bernoulli distribution can be reparameterised using a continuous approximation to the discrete distribution. If we have a discrete distribution $(\alpha_1, \cdots \alpha_K)$ where $\alpha_j \in \{0, \infty\}$ and $D \sim \text{Discrete}(\alpha) \in \{0, 1\}$, then $P(D_j = 1) = \frac{\alpha_j}{\sum_k \alpha_k}$. Sampling from this distribution requires performing an `argmax` operation, the crux of the problem is that the `argmax` operation doesn't have a well defined derivative.

To address the derivative issue above, we use the Concrete distribution [17] or Gumbel-Softmax distribution [18] as an approximation to the Bernoulli distribution. The idea is that instead of returning a state on the vertex of the probability simplex like `argmax` does, these relaxations return states inside the inside the probability simplex (see Figure 2 in [17]). We follow the Concrete formulation and notation from [17] to sample from the probability simplex as

$$X_j = \frac{\exp((\log \alpha_j + G_k)/\lambda)}{\sum_{i=1}^{n} \exp((\log \alpha_i + G_i)/\lambda)} \tag{20}$$

with temperature hyperparameter $\lambda \in (0, \infty)$, parameters $\alpha_j \in (0, \infty)$ and i.i.d. Gumbel noise $G_j \sim \text{Gumbel}(0, 1)$. This equation resembles a `softmax` with a Gumbel perturbation. As $\lambda \to 0$ the `softmax` computation approaches the `argmax` computation. This can be used as a relaxation of the variational Bernoulli distribution and can be used to reparameterise Bernoulli random variables to allow gradient based learning of the variational Beta parameters downstream in our model.

When performing variational inference using the Concrete reparameterisation for the posterior, a Concrete reparameterisation of the Bernoulli prior is required to properly lower bound the ELBO 15. If $q(z_{nk}; \pi_k|v_k)$ is the Bernoulli variational posterior over sparse binary masks $z_{nk}$ for weights $w_{k.}$ and all data points $n \in \{1, \cdots N\}$ and $p(z_{nk}; \pi_k|v_k)$ is the Bernoulli prior. To guarantee a lower bound on the ELBO both Bernoulli distributions require replacing with Concrete densities, i.e.,

$$\mathrm{KL}\left[q(z_{nk}; \pi_k|v_k)||p(z_{nk}; \pi_k|v_k)\right] \geq \mathrm{KL}\left[q(z_{nk}; \pi_k, \lambda_1|v_k)||p(z_{nk}; \pi_k, \lambda_2|v_k)\right], \tag{21}$$

| Dataset | Training set size | Test set size |
|---|---|---|
| Split MNIST | $50,000$ | $10,000$ |
| Split MNIST + noise | $50,000$ | $12,000$ |
| Split MNIST + background images | $50,000$ | $12,000$ |
| not MNIST | $200,000$ | $10,000$ |

Table 1: Sizes of the training and test sets for the datasets used.

where $q(z_{nk}; \pi_k, \lambda_1 | v_k)$ is a Concrete density for the variational posterior with parameter $\pi_k$, temperature parameter $\lambda_1$ given global parameters $v_k$. $p(z_{nk}; \pi_k, \lambda_2 | v_k)$ is the Concrete prior. Equation (21) is evaluated numerically by sampling from the variational posterior (we will take a single Monte Carlo sample [16]). At test time we can sample from a Bernoulli using the learnt variational parameters of the Concrete distribution [17].

In practice, we use log transformation to alleviate underflow problems when working with Concrete probabilities. One can instead work with $\exp(Y_{nk}) \sim \text{BinConcrete}(\pi_k, \lambda_1 | v_k)$, as the KL divergence is invariant under this invertible transformation and Equation (21) is valid for optimising our Concrete parameters [17]. For binary Concrete variables we can sample from $y_{nk} = (\log \pi_k + \log u - \log(1-u))/\lambda_1$ where $u \sim \text{U}[0,1]$ and the log-density (before applying the sigmoid activation) is $\log q(y_{nk}; \pi_k, \lambda_1 | v_k) = \log \lambda_1 - \lambda_1 y_{nk} + \log \pi_k - 2 \log(1 + \exp(-\lambda_1 y_{nk} + \log \pi_k))$ [17].

# D Experimental details

For all experiments, the BNN architecture used for incorporating the IBP prior has a single layer with ReLU activation functions, the variational truncation parameter for the IBP variational posterior is set to $K = 100$: the maximum number of nodes in our network is 100. At the start of the optimisation the parameters of the Beta distribution are initialised with $\alpha_k = 5.0$ and $\beta_k = 1.0$ for all $k$. The temperature parameters of the Concrete distributions for the variational posterior and prior are set to $\lambda_1 = 1.0$ and $\lambda_2 = 1.0$ respectively (the prior distribution is also chosen as Concrete in order to find a proper lower bound of the ELBO for reasons discussed in section C.3). Our implementation of the IBP is adapted from the code by [20].

Our implementation of the BNN and the continual learning framework is based off of code from [2]. The BNNs use a multi-head network. The Gaussian weights of the BNN have their means initialised with the maximum likelihood estimators and variances equal to 1. We use an Adam optimiser [38] and train for $1000$ epochs with a learning rate of $0.0001$.

For the weight pruning experiment the baseline BNN has a hidden layer of size 100. The only difference to the details above is that both the BNN and the BNN with IBP prior are trained for 600 epochs.

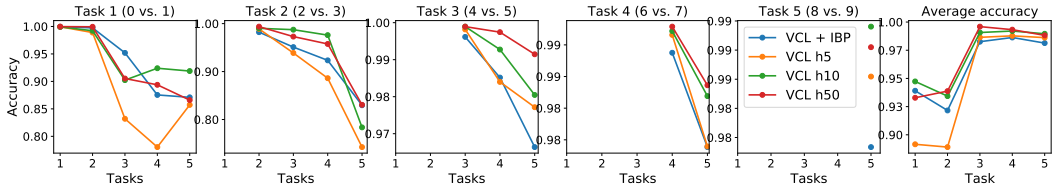
We summarise the sizes of the datasets used for experiments in table 1.

# E Further results on MNIST variants

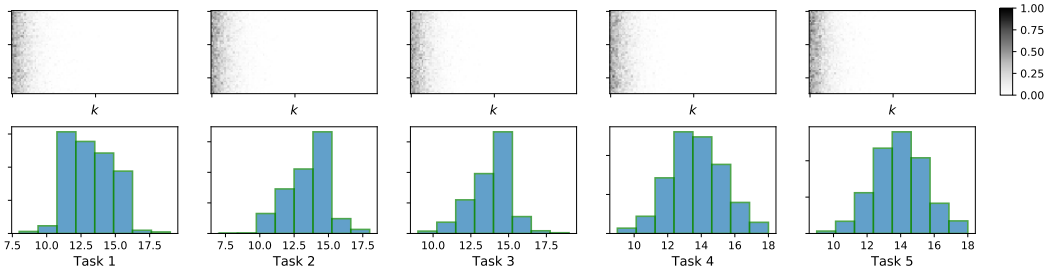
We elaborate on the results which are presented in section 3. The results are shown for various MNIST variants and the not MNIST dataset. The accuracies of each task after successive continual learning steps are show in Figures 3 - 6.

For the split MNIST dataset Figure 3 we note that the IBP prior BNN is able to outperform the 5 neuron VCL, this network is underfitting. One the other hand the 10 and 50 neuron VCL networks outperform the IBP prior network in particular for tasks 3 to 5. The IBP prior BNN is able to expand a small amount as the number of tasks increases. Regarding MNIST with background noise[1], it is clear that the IBP prior model outperforms all VCL baselines for all tasks. The IBP prior model also expands slightly, however it doesn't extend its capacity past the largest VCL baseline model considered, $h = 50$. Similarly the results using an MNIST + random background[1] images in general

[1]The data is obtained from `https://sites.google.com/a/lisa.iro.umontreal.ca/public_static_twiki/variations-on-the-mnist-digits`




(a) Split MNIST task accuracies versus the number of tasks the model has seen and performed the Bayesian update for. The final plot is a per task average accuracy as shown previously.



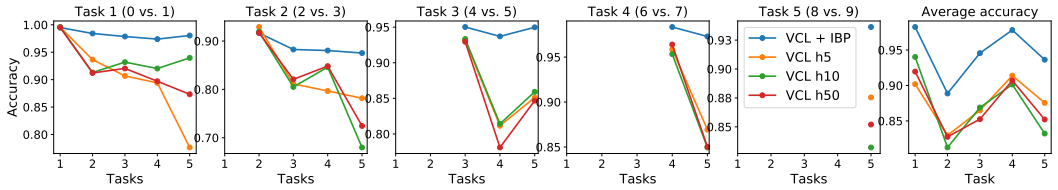

(b) Per task sparse $Z$ matrix for a batch in test set and histograms of the number of active neurons per point in the test set.

Figure 3: Split MNIST task accuracies versus the number of tasks the model has seen and performed the Bayesian update for. Our model is compared to VCL benchmarks with different numbers of hidden states denoted $hx, x \in \{5, 10, 50\}$ in the plot legend. Accuracies are an average of 5 optimisations. We also show the sparse $Z$ matrix of the IBP prior model after each Bayesian approximate update together the a histogram of the number of neurons which are active for each data point in the test set. The average number of neurons which are active per point in the test set is increases steadily from 13.2 to 13.9

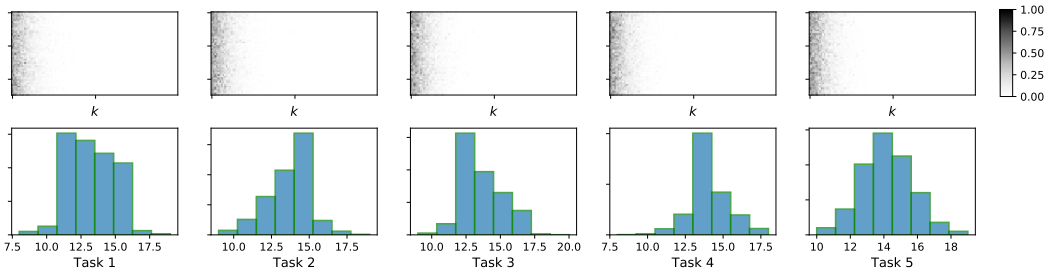
shows the IBP prior BNN outperforming all VCL baselines except when the model first sees the data in tasks 2, 3, and 4. Despite this the IBP prior network forgets these tasks less throughout the continual learning process. The results for not MNIST [2] show that IBP prior BNN outperforms the $h \in \{5, 10\}$ BNNs for task 1, 2 and 3. The $h = 50$ BNN outperforms the IBP prior BNN for all tasks apart from the first.

Note that the sparse $Z$ matrices shown in Figures 3 to 6 are not binary like in the original IBP formulation as we use a Concrete relaxation to the Bernoulli distribution. A neuron is defined as active in the IBP BNN when $z_{nk} > 0.1$ for the Figures 1 and 3b to 6b.

[2]Data obtained from `http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html`

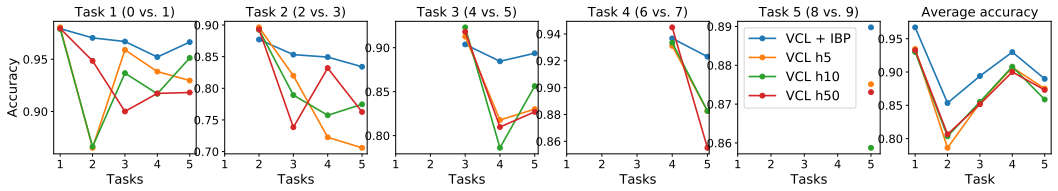



(a) Split MNIST + noise dataset continual learning task accuracies versus the number of tasks the model has seen and performed the Bayesian update for. The final plot is a per task average accuracy as shown previously.

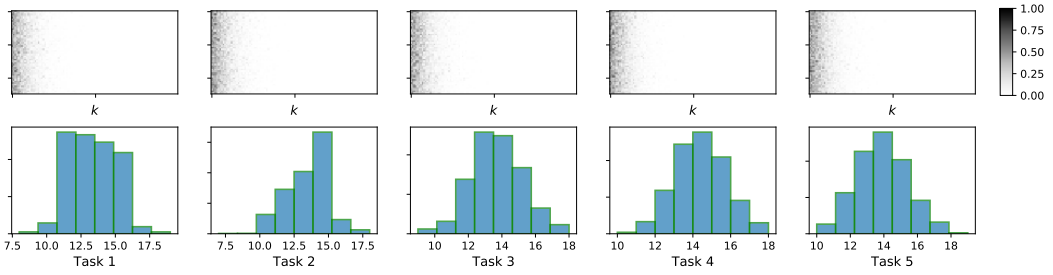



(b) Per task sparse $Z$ matrix for a batch in test set and histograms of the number of active neurons per point in the test set.

Figure 4: Split MNIST with random noise task accuracies versus the number of tasks the model has seen and performed the Bayesian update for. Our model is compared to VCL benchmarks with different numbers of hidden states. Accuracies are an average of 5 optimisations. We also show the sparse $Z$ matrix of the model after each Bayesian approximate update together the a histogram of the number of neurons which are active for each point in the test set. The average number of neurons which are active per point in the test set is increases steadily from 13.3 to 14.2

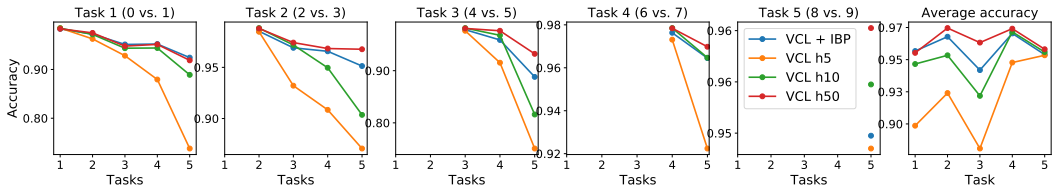



(a) Split MNIST + background image dataset continual learning task accuracies versus the number of tasks the model has seen and performed the Bayesian update for. The final plot is a per task average accuracy as shown previously.

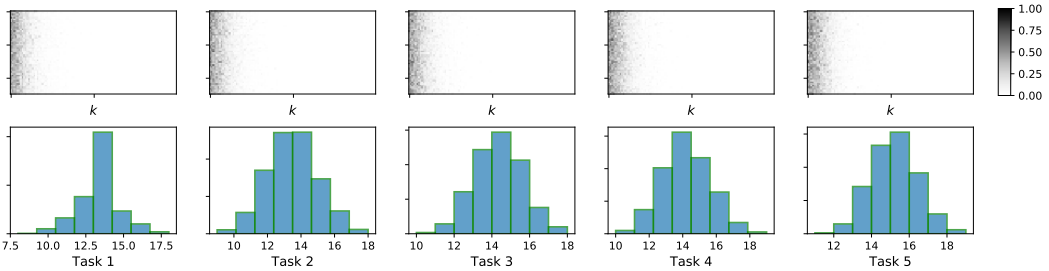



(b) Per task sparse $Z$ matrix for a batch in test set and histograms of the number of active neurons per point in the test set.

Figure 5: Split MNIST with random background images task accuracies versus the number of tasks the model has seen and performed the Bayesian update for. Our model is compared to VCL benchmarks with different numbers of hidden states. Accuracies are an average of 5 optimisations. We also show the sparse $Z$ matrix of the model after each Bayesian approximate update together the a histogram of the number of neurons which are active for each point in the test set. The average number of neurons which are active per point in the test set is increases steadily from 13.4 to 14.0.




(a) Split Not MNIST image dataset continual learning task accuracies versus the number of tasks the model has seen and performed the Bayesian update for. The final plot is a per task average accuracy.




(b) Per task sparse $Z$ matrix for a batch in test set and histograms of the number of active neurons per point in the test set.

Figure 6: Split Not MNIST task accuracies versus the number of tasks the model has seen and performed the Bayesian update for. Our model is compared to VCL benchmarks with different numbers of hidden states. Accuracies are an average of 5 optimisations. We also show the sparse $Z$ matrix of the model after each Bayesian approximate update together the a histogram of the number of neurons which are active for each point in the test set. The average number of neurons which are active per point in the test set increases steadily from 13.2 to 14.7