Continual Learning using a Bayesian Nonparametric Dictionary of Weight Factors

Nikhil Mehta[1], **Kevin J Liang**[1,2], **Vinay K Verma**[1], **Lawrence Carin**[1]
[1]Duke University [2]Facebook AI

Abstract

Naively trained neural networks tend to experience catastrophic forgetting in sequential task settings, where data from previous tasks are unavailable. A number of methods, using various model expansion strategies, have been proposed recently as possible solutions. However, determining how much to expand the model is left to the practitioner, and often a constant schedule is chosen for simplicity, regardless of how complex the incoming task is. Instead, we propose a principled Bayesian nonparametric approach based on the Indian Buffet Process (IBP) prior, letting the data determine how much to expand the model complexity. We pair this with a factorization of the neural network's weight matrices. Such an approach allows the number of factors of each weight matrix to scale with the complexity of the task, while the IBP prior encourages sparse weight factor selection and factor reuse, promoting positive knowledge transfer between tasks. We demonstrate the effectiveness of our method on a number of continual learning benchmarks and analyze how weight factors are allocated and reused throughout the training.

1 Introduction

Deep learning, trained primarily on a single task under the assumption of independent and identically distributed (*i.i.d.*) data, has made enormous progress in recent years. However, when naively trained sequentially on multiple tasks, without revisiting previous tasks, neural networks are known to suffer catastrophic forgetting (McCloskey & Cohen, 1989; Ratcliff, 1990): the ability to perform old tasks is often lost while learning new ones. In contrast, biological life is capable of learning many tasks throughout a lifetime from decidedly non-*i.i.d.* experiences, acquiring new skills and reusing old ones to learn fresh abilities, all while retaining important previous knowledge. As we strive to make artificial systems increasingly more intelligent, natural life's ability to learn continually is an important capability to emulate.

Continual learning (Parisi et al., 2019) has attracted considerable attention recently in machine learning research, and a number of desiderata have emerged. Models should be able to learn multiple tasks sequentially, with the eventual number and complexity of tasks unknown. Importantly, new tasks should be learned without catastrophically forgetting previous ones, ideally without having to keep any data from previous tasks to re-train on. Models should also be capable of positive transfer: previously learned tasks should help with the learning of new tasks. Knowledge transfer between tasks maximizes sample efficiency, with this particularly important when data are scarce.

A number of methods (Rusu et al., 2016; Zhang et al., 2019; Lee et al., 2020) address continual learning through expansion: the model is grown with each additional task. By diverting learning to new network components for each task, these approaches mitigate catastrophic forgetting by design, as previously learned parameters are left undisturbed. A key challenge for these strategies is deciding when and how much to expand the network. While it is typically claimed that this can be tailored to the incoming task, doing so requires human estimation of how much expansion is needed, which is not a straightforward process. Instead, a preset, constant expansion is commonly employed for each new task.

Rather than relying on engineered heuristics, we choose to let the data dictate the model-expansion rate, employing a Bayesian nonparametric approach. Specifically, we couple rank-1 weight factor (WF) dictionary learning with the Indian Buffet Process (IBP) (Ghahra-

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. Copyright 2021 by the author(s).

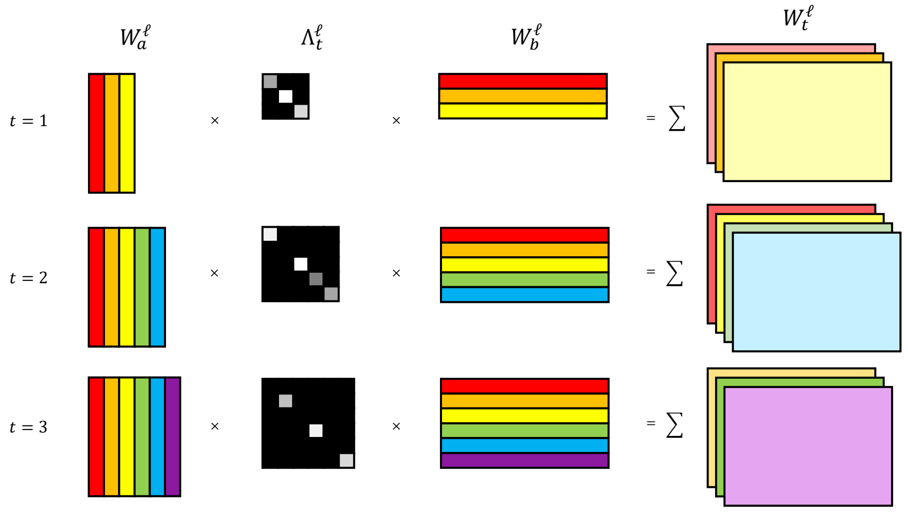


Figure 1: Layer-wise weight factors for continual learning. Dictionaries of weight factors $\mathbf{W}_a^\ell$ and $\mathbf{W}_b^\ell$ are shared across all tasks, and a task-specific sparse diagonal matrix $\mathbf{\Lambda}_t^\ell$ specifies the active factors (in this figure, elements of $\mathbf{\Lambda}_t^\ell$ that are black are zero, and brighter shades correspond to larger numbers). The weighted sum of the active weight factors yields the weight matrix for a particular task. The number of factors (columns of $\mathbf{W}_a^\ell$ and rows of $\mathbf{W}_b^\ell$) grows as needed with more tasks, with select factors being reused in future tasks. Best viewed in color.

mani & Griffiths, 2006), creating a framework we call **IBP-WF**. An IBP-based formulation allows automatic scaling of the network, but only as needed, even if the number or complexity of future tasks is unknown initially. An IBP prior also naturally encourages recycling of previously learned skills, enabling positive transfer between tasks, which other expansion methods tend to either ignore or deal with in a more *ad hoc* manner. Finally, Bayesian modeling enables model sampling, allowing for both ensembling models for increased accuracy and uncertainty estimation, which are important but rarely discussed topics in continual learning.

Our main contributions are as follows. (*i*) We introduce learning a rank-1 weight factor dictionary for a neural network expected to perform multiple tasks. We then introduce the Indian Buffet Process as a prior for each task's weight factor selection, showing why the IBP is a natural choice given continual learning's desiderata. (*ii*) We introduce a simple-but-effective method based on feature statistics for inferring task identity (ID) in incremental class settings. (*iii*) The effectiveness of IBP-WF is demonstrated on a number of continual learning tasks, outperforming other methods. We also visualize the weight factor usage across tasks, confirming both sparsity and reuse of these factors.

2 Methods

2.1 Weight Factor Dictionary Learning

Consider a multilayer perceptron (MLP) with layers $\ell = 1, ..., L$. In a continual learning setting, we would like this neural network to learn multiple tasks. Given differences between tasks, the neural network may require a different set of weight matrices $\{\mathbf{W}_t^\ell\}_{\ell=1}^L$ for each task t. While $\{\mathbf{W}_t^\ell\}_{\ell=1}^L$ could be learned separately for each task, such a model does not incorporate knowledge reuse, and the total number of model parameters grows linearly with the number of tasks T. While immune from catastrophic forgetting, such an approach is inefficient in both computation and data.

Instead of completely independent models for each task, we propose constituting $\mathbf{W}_t^\ell$ as follows:

$$\mathbf{W}_t^\ell = \mathbf{W}_a^\ell \mathbf{\Lambda}_t^\ell \mathbf{W}_b^\ell \qquad \text{s.t.} \quad \mathbf{\Lambda}_t^\ell = \text{diag}(\boldsymbol{\lambda}_t^\ell) \tag{1}$$

where $\mathbf{W}_a^\ell \in \mathbb{R}^{J \times F}$ and $\mathbf{W}_b^\ell \in \mathbb{R}^{F \times M}$ are global parameters shared across tasks and $\boldsymbol{\lambda}_t^\ell \in \mathbb{R}^F$ is a task-specific vector. The determination of F is explained in Section 2.2, but in general F is chosen such that after T tasks, the total number of parameters of this factorized model is $(J + T + M) \cdot F$, which is significantly less than the $J \cdot M \cdot T$ parameters that would result from learning each task independently. We may equivalently express (1) as the weighted sum of rank-1 matrices formed from the outer product of the vectors corresponding to the columns $\mathbf{W}_a^\ell$ and rows of $\mathbf{W}_b^\ell$:

$$\mathbf{W}_t^\ell = \sum_{k=1}^{F} \boldsymbol{\lambda}_{t,k}^\ell \left(\mathbf{w}_{a,k}^\ell \otimes \mathbf{w}_{b,k}^\ell \right) \tag{2}$$

where $\otimes$ denotes the outer product, and the pair $\mathbf{w}_{a,k}^\ell$ and $\mathbf{w}_{b,k}^\ell$ is the k^{th} column and row of $\mathbf{W}_a^\ell$ and $\mathbf{W}_b^\ell$, respectively. Under this construction, the pairs of corresponding columns of $\mathbf{W}_a^\ell$ and rows of $\mathbf{W}_b^\ell$ can be interpreted as a dictionary of weight *factors*, while the values in $\boldsymbol{\lambda}_t^\ell$ are the factor *scores* for a particular task (see Figure 1). By sharing these global weight factors, the model can reuse features and transfer knowledge between tasks, with $\boldsymbol{\lambda}_t^\ell$ selecting and weighting the factors for a particular task t. We construct the factor scores $\boldsymbol{\lambda}_t^\ell$ as the following element-wise product:

$$\boldsymbol{\lambda}_t^\ell = \mathbf{r}_t^\ell \odot \mathbf{b}_t^\ell \tag{3}$$

where $\mathbf{b}_t^\ell \in \{0,1\}^F$ indicates the *active* factors for task t and $\mathbf{r}_t^\ell \in \mathbb{R}^F$ specifies the corresponding factor *strength*. By imposing sparsity with $\mathbf{b}_t^\ell$, we concentrate skills for each task into specific factors, leaving room for learning other tasks with other factors.

Generalizing to convolutional kernels While learning the weight factors in (1) was formulated for fully connected layers, it can be generalized to other types of layers as well, including 2D convolutional layers. Unlike the 2D weight matrices comprising the fully connected layers of a MLP, convolutional kernels are 4D: in addition to number of input and output channels (C_{in} and C_{out}), they also have two spatial dimensions denoting the height (H) and width (W) of the convolutional filter. While learning 4D tensor factors is certainly possible, in practice H and W tend to be small (*e.g.* $H = W = 3$), so we instead choose to reshape the kernel ($\mathbb{R}^{H\times W\times C_{in}\times C_{out}}$) into a 2D matrix ($\mathbb{R}^{(HWC_{in})\times C_{out}}$). We then proceed with the same weight factor learning as in (1).

2.2 Indian Buffet Process for Weight Factors

Critical to the proposed layer-wise weight factors is the number of factors F: too few and the model lacks sufficient expressivity to model every task; too many and the model consumes more memory and computation than necessary. To further complicate matters, the number of necessary factors likely increases monotonically as the model encounters more tasks. While a particular choice of F may be appropriate for T tasks, it may no longer be sufficient after T', with $T' > T$.

Rather than setting it as a constant, we let F grow naturally with the number of tasks. There are a number of expansion strategies for continual learning that have been proposed over the years (Rusu et al., 2016; Hung et al., 2019; Zhang et al., 2019; Lee et al., 2020). Many of these expand the model by a constant amount per task, or rely on the model designer to specify a schedule or heuristics for the size of the expansion. These hand-tuned strategies can be brittle, and require expert knowledge on the complexity of incoming tasks. Additionally, prior works do not use weight factor dictionaries, so expansion involves adding additional nodes to each hidden layer or learning entirely new models, which can increase test-time computation.

Instead, we employ Bayesian nonparametrics, inferring in a principled manner the scores for the proposed rank-1 weight factors for each task and the total number of factors F needed. In particular, we impose the stick-breaking construction of the IBP (Teh et al., 2007) as a prior for factor selection:

$$v_{t,i}^\ell \sim \text{Beta}(\alpha, 1) \tag{4}$$

$$\pi_{t,k}^\ell = \prod_{i=1}^{k} v_{t,i}^\ell \tag{5}$$

$$b_{t,k}^\ell \sim \text{Bernoulli}(\pi_{t,k}^\ell) \tag{6}$$

where α is a hyperparameter controlling the expected number of nonzero factor scores, and $k = 1, 2, ...F$ indexes the factor. For the global parameters ($\mathbf{W}_a^\ell$ and $\mathbf{W}_b^\ell$) and the local factor strength ($\mathbf{r}_t^\ell$), we use point estimates. See Appendix A for an overview of the IBP and the connection of IBP-WF with IBP-based dictionary learning. Leveraging the IBP in conjunction with dictionary learning provides a number of natural advantages within the context of continual learning:

Dynamic control of F The IBP allows the number of factors F to be determined nonparametrically and dynamically, growing only as necessary given the complexity of each individual task. Simpler tasks (or ones similar to previous tasks) may require learning fewer new factors, while more complex ones lead to more, all inferred automatically. While F can theoretically grow unbounded, it does so harmonically – much slower than the requisite linear growth of the number of tasks.

Factor reuse and positive transfer Given that continual learning is often deployed when tasks are at least somewhat correlated, training independent models can lead to learning redundant features, which is inefficient both in training data and test time computation. On the other hand, the construction of $\boldsymbol{\pi}_t^\ell$ (see (5)) actively encourages reuse of existing weight factors, prioritizing recycling previously learned skills for new tasks over creating new ones, which leads to positive forward transfer. This is in contrast to other methods whose only source of transfer is initializing from the previous task's weights, whose transfer advantage may be quickly wiped away by gradient descent.

Catastrophic forgetting mitigation The newly learned weight factors in W_a and W_b are frozen at the end of a task. This mirrors the freezing of previously learned weights in existing expansion methods. By blocking the gradients to weights learned from previous tasks, we avoid forgetting the model's ability to perform older tasks. Note that while factors learned from a previous task are frozen, the factor scores may change with each incoming task allowing the model to control the usage of a previously trained factor.

Constant inference time cost At test time, IBP weight factors (outer product of each column of $\mathbf{W}_a^\ell$ and row of $\mathbf{W}_b^\ell$) can be pre-computed; given a task, the appropriate factors can be retrieved, weighted, and summed as needed to retrieve $\{\mathbf{W}_t^\ell\}_{\ell=1}^L$. Imposing the IBP prior on the usage of factors induces a prior distribution of Poisson(α) on the number of active factors. Thus, the number of nonzero factors have a prior expectation of α, regardless of the number of tasks T, so the expected forward computation of the model does not grow with T. This avoids one of the pitfalls of other expansion methods (*e.g.,* Rusu et al. (2016); Lee et al. (2020); Kumar et al. (2019)), whose inference-time computation scale with T.

2.3 Variational Inference

To determine which factors should be active for task t, we perform variational inference to infer the posterior of parameters $\theta_t = \{\mathbf{b}_t^\ell, \mathbf{v}_t^\ell\}_{\ell=1}^L$. We assume the following variational distributions:

$$q(\theta_t^\ell) = q(\mathbf{b}_t^\ell)q(\mathbf{v}_t^\ell) \tag{7}$$
$$\mathbf{b}_t^\ell \sim \text{Bernoulli}(\boldsymbol{\pi}_t^\ell) \tag{8}$$
$$\mathbf{v}_t^\ell \sim \text{Kumaraswamy}(\mathbf{c}_t^\ell, \mathbf{d}_t^\ell) \tag{9}$$

We learn the variational parameters $\{\boldsymbol{\pi}_t^\ell, \mathbf{c}_t^\ell, \mathbf{d}_t^\ell\}_{\ell=1}^L$ with Bayes by Backprop (Blundell et al., 2015). As the Beta distribution lacks a differentiable parameterization, we use the similar Kumaraswamy distribution (Kumaraswamy, 1980) as the variational distribution for $\mathbf{v}_t^\ell$. We also use a soft relaxation of the Bernoulli distribution (Maddison et al., 2017) in (6) and (8) to allow backpropagation through discrete random variables. The objective for each task is to maximize the following variational lower bound:

$$\begin{aligned}\mathcal{L}_t = &\sum_{n=1}^{N_t} \mathbb{E}_q \log p\left(y_t^{(n)} | \theta_t, x_t^{(n)}, \mathbf{W}_a, \mathbf{W}_b, \mathbf{r}_t\right) \\ &- \underbrace{\text{KL}\left(q\left(\theta_t\right) || p\left(\theta_t\right)\right)}_{\mathscr{R}}\end{aligned} \tag{10}$$

where N_t is the number of training examples in task t, $\mathbf{W}_a = \{\mathbf{W}_a^\ell\}_{\ell=1}^L$, $\mathbf{W}_b = \{\mathbf{W}_b^\ell\}_{\ell=1}^L$ and $\mathbf{r}_t = \{\mathbf{r}_t^\ell\}_{\ell=1}^L$. The variational lower bound in (10) is maximized with respect to task-specific parameters (θ_t, r_t) and global parameters ($\mathbf{W}_a, \mathbf{W}_b$). Note that in (10) the first term provides label supervision and the second term ($\mathscr{R}$) regularizes the posterior not to stray too far from the IBP prior. We use mean-field approximation and MC sampling to approximate the second term:

$$\mathscr{R} = \sum_{\ell=1}^{L} \mathbb{E}_{q(\mathbf{v}_t^\ell)} \left[\text{KL}\left(q(\mathbf{b}_t^\ell) || p(\mathbf{b}_t^\ell | \mathbf{v}_t^\ell)\right)\right] + \text{KL}\left(q(\mathbf{v}_t^\ell) || p(\mathbf{v}_t^\ell)\right) \tag{11}$$

where we take samples $\mathbf{v}_t^\ell \sim q(\mathbf{v}_t^\ell)$ to approximate $\mathbb{E}_q[\log(p(\mathbf{b}_t^\ell | \mathbf{v}_t^\ell))]$ in the first term. For the second term in (11), we derive an analytic approximation of Kullback-Leibler (KL) divergence between two Kumaraswamy distributions. The derivation and more details on doing online inference with (10) and (11) are included in Appendices B and C.

Variational continual learning (VCL) (Nguyen et al., 2018) addresses catastrophic forgetting using online inference, *i.e.*, the posterior inferred from the most recent task is used as a prior for the incoming task. However, recent work (Farquhar & Gal, 2018, 2019) suggests that approximate online inference often does not succeed in mitigating catastrophic forgetting in realistic continual learning settings, as methods based solely on approximate inference rely on a simple prior to capture everything learned on all previous tasks. Thus, instead of performing online inference for all parameters $\{\mathbf{r}_t^\ell, \mathbf{b}_t^\ell, \mathbf{v}_t^\ell\}$, we only apply online inference for $\mathbf{v}_t^\ell$ and learn task-specific parameters $\{\mathbf{r}_t^\ell, \mathbf{b}_t^\ell\}$. Note that online inference over $\mathbf{v}_t^\ell$ encourages the reuse of factors from previous tasks while having task-specific parameters allows the model to easily adapt to a new task by using new factors.

Preserving knowledge If all of $\mathbf{W}_a^\ell$ and $\mathbf{W}_b^\ell$ were free to move without constraint, then catastrophic forgetting may still occur. Indeed, the model could "reuse" a factor from a previous task and then repurpose it entirely, undermining the ability to do the former task. To prevent this, the weight factors (*i.e.,* the columns of $\mathbf{W}_a^\ell$ and rows of $\mathbf{W}_b^\ell$) with factor probability $\pi_{t,k}^\ell > \kappa$ are locked (*e.g.*, with a stop gradient operator) at the conclusion of a task. Weight factors below the threshold κ are left free to be modified by future tasks. Throughout our experiments, we set the threshold as $\kappa = 0.5$, but this can be adjusted based on tolerance for forgetting. We include an ablation study on selecting κ in the Appendix H.2. Alternatively, other regularization methods (*e.g.,* Kirkpatrick et al. (2017)) can be used to prevent important factors from drifting too far, but we leave this combination to future work.

2.4 Task Inference at Test Time

IBP-WF addresses catastrophic forgetting and allows for positive knowledge transfer. However, as with many continual learning methods, IBP-WF requires the task identity associated with each input at test time in order to select the proper $\{\boldsymbol{\Lambda}_t^\ell\}_{\ell=1}^L$. The validity of this assumption has occasionally been questioned (Farquhar & Gal, 2018; Aljundi et al., 2019; Lee et al., 2020). We outline here a mechanism for enabling IBP-WF to operate in an incremental class setting, inferring the task identity at test time. Given a data point x, we can infer the task identity by defining the probability of x belonging to a particular task t as follows:

$$P(t|x) \propto P(x|t)P(t) \tag{12}$$

However, using (12) requires learning a generative model $P(x|t)\ \forall t \in \{1, 2, ..., T\}$, which can be expensive in both computation and the number of parameters. To alleviate this issue, we propose a simple yet effective alternative: we define an approximation to $P(x|t)$ by using the feature distribution induced by an intermediate hidden layer of the trained neural network. In particular, we approximate (12) by using $P(t|\phi(x))$ as a surrogate for $P(t|x)$:

$$P(t|x) \approx P(t|\phi(x)) = \frac{P(\phi(x)|t)P(t)}{\sum_{t'} P(\phi(x)|t')P(t')} \tag{13}$$

where ϕ is an intermediate layer defined using the proposed weight factorization as shown in (1) with task-specific weights of the first task. We work with the feature space induced by the parameters of the first task as they are accessible by the training data of all tasks that follow. Next, we assume $P(\phi(x)|t)$ to be a Gaussian distribution: $P(\phi(x)|t) = \mathcal{N}(\phi(x)|\mu_t, \Sigma_t)$, where the parameters are the empirical estimates using the training data.

$$\begin{aligned}\hat{\mu}_t &= \frac{1}{N_t}\sum_{n=1}^{N_t} \phi(x_t^{(n)}), \\ \hat{\Sigma}_t &= \frac{1}{N_t}\sum_{n=1}^{N_t} (\phi(x_t^{(n)}) - \hat{\mu}_t)(\phi(x_t^{(n)}) - \hat{\mu}_t)^T\end{aligned} \tag{14}$$

where $x_t^{(n)}$ is a training sample from task t. When we train our model on task $t > 1$, we use the task-specific weights learned for the first task to compute $\{\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t\}$. The parameters $\{\mu_t, \Sigma_t\}$ are stored to infer test-time task identity. While the features may not be exactly Gaussian distributed, this assumption has been shown to work well in deep learning (Heusel et al., 2017; Lee et al., 2018), and we find it effective in practice; see Appendix D.2 for task inference accuracy experiments and t-SNE visualizations of ϕ. Notably, we achieve similar accuracy to generative model task inference (Lee et al., 2020), with a far cheaper method.

Considering the marginal task distribution $P(t) \propto N_t$, the task identity can be inferred as follows:

$$\begin{aligned}\hat{t} = \arg\min_t \Bigg[&\frac{\log|\hat{\Sigma}_t|}{2} - \log(N_t) \\ &+ \frac{1}{2}(\phi(x) - \hat{\mu}_t)^T \hat{\Sigma}_t^{-1} (\phi(x) - \hat{\mu}_t) \Bigg]\end{aligned} \tag{15}$$

where $\hat{t}$ is the inferred task and I is the identity matrix. While such a strategy does require storing statistics $\hat{\mu}_t$ and $\hat{\Sigma}_t$, the total size of these is still considerably smaller than parameter statistics required by certain regularization methods (*e.g.,* EWC (Kirkpatrick et al., 2017)), as well as the coresets or generative models used by replay methods (Nguyen et al., 2018; Shin et al., 2017; van de Ven & Tolias, 2018).

Remark: (Informal) The approximation in (13) is exact if ϕ is an invertible map since $P(s|t) = M \times P(x|t)$ with $M = \left|\det \frac{\partial \phi^{-1}}{\partial s}\right|$ when $s = \phi(x)$. See Appendix D.1 for a formal proof.

3 Related Works

There have been a number of diverse continual learning methods that have been proposed in recent years, most of which can be roughly grouped by strategy into a few categories, with some overlap. Regularization-based approaches (Kirkpatrick et al., 2017; Zenke et al., 2017; Li & Hoiem, 2017; Nguyen et al., 2018; Aljundi et al., 2018; Schwarz et al., 2018; Ritter et al., 2018) add a loss term constraining the network parameters to remain close to solutions of previously learned tasks. Others use replay (Kirkpatrick et al., 2017; Lopez-Paz & Ranzato, 2017; Shin et al., 2017; Nguyen et al., 2018; Rolnick et al., 2019), which retrains the model on samples from earlier tasks, either from a saved core set or with a generative model that must be learned.

Another class of continual learning methods rely on expansion, the approach taken by IBP-WF. Progressive Neural Networks (Rusu et al., 2016) learn a new neural network column for each new task, with previous columns' features as additional inputs. While avoiding catastrophic forgetting by design, both memory and computation grow linearly with the number of tasks T, just as if one were to learn independent models per task. Side-tuning (Zhang et al., 2019) learns a separate "side" network for each task, adding the output to a shared base model; while this experiences linear growth T of the model size, it reduces the cost by keeping each side network small. As an alternative to constant growth, Reinforced Continual Learning (Xu & Zhu, 2018) uses an LSTM (Hochreiter & Schmidhuber, 1997) controller and REINFORCE (Williams, 1992) to determine the expansion rate, while Dynamically Expandable Networks (Yoon et al., 2018) expand by a constant amount before using sparsity regularization and loss-based heuristics to prune away unused units. Pruning between tasks is also utilized by Hung et al. (2019), where the pruning and re-training is used to prevent excessive growth of the model. MNDPT (Veniat et al., 2021) adds new modules to the model with new tasks, reusing modules of older similar tasks.

Table 1: The average accuracy of seen tasks after learning on a sequence of tasks using a MLP.

Method	Replay	Split MNIST		Permuted MNIST	
		Incremental Task	Incremental Class	Incremental Task	Incremental Class
Adagrad		98.24 ± 0.59	19.73 ± 0.12	90.78 ± 0.18	27.59 ± 1.07
EWC		98.64 ± 0.87	19.89 ± 0.04	92.49 ± 0.34	23.97 ± 3.21
SI		99.16 ± 0.52	19.71 ± 0.10	95.45 ± 0.59	56.88 ± 4.93
MAS		99.23 ± 0.18	19.58 ± 0.11	96.76 ± 0.26	49.95 ± 2.53
LwF		99.61 ± 0.05	22.31 ± 0.51	81.47 ± 0.38	30.63 ± 0.76
VCL		96.79 ± 0.35	19.43 ± 0.02	91.33 ± 0.93	16.21 ± 0.59
Naive rehearsal	✓	99.39 ± 0.11	85.97 ± 0.75	96.75 ± 0.19	96.53 ± 0.11
VCL-Coreset	✓	98.75 ± 0.06	85.15 ± 0.61	93.46 ± 0.49	66.96 ± 4.10
GEM	✓	98.56 ± 0.08	88.28 ± 0.26	97.14 ± 0.09	96.88 ± 0.05
DGR	✓	99.54 ± 0.05	91.61 ± 0.26	93.74 ± 0.24	92.96 ± 0.53
RtF	✓	99.66 ± 0.03	**92.56** ± 0.21	97.31 ± 0.01	96.23 ± 0.04
IBP-WF (Ours)		**99.69** ± 0.05	92.40 ± 0.15	**97.52** ± 0.06	**97.50** ± 0.06

A few works have also explored continual learning from a Bayesian nonparametric perspective. Lee et al. (2020) combine the Dirichlet process with a mixture of experts, where each expert is a neural network responsible for a subset of the data. While this approach does allow the data to dictate model expansion, mixing only occurs at the prediction representation, as opposed to throughout the model as in IBP-WF. This mixture of experts thus can lead to redundant feature learning and unnecessary extra computation. Recently, there have been other attempts to apply IBP to learn the structure of a neural network for continual learning. Kumar et al. (2019) proposed Bayesian Structure Adaptation for Continual Learning (BSCL), which expands the hidden units in each layer using a binary mask for the weight filters, with an IBP as the prior of the mask. Since BSCL uses an IBP over the entire weight matrix, the inference parameters grow quadratically with the layer size requiring more memory and making it hard to scale to large networks, whereas we use the IBP to model the factor scores where the inference parameters only grow linearly with the layer size. H-IBP Bayesian Neural Networks (HIBNN) (Kessler et al., 2020) uses sequential Bayes to apply hierarchical IBP to the hidden layer activations of a fully connected neural network. In contrast, we expand the number of factors of the weight matrix in each layer, allowing us to scale our method to deeper networks with convolutional layers.

4 Experiments

We evaluate our method in two settings, which we call incremental *task* learning and incremental *class* learning. In incremental task learning, the task identity (ID) of each sample is revealed at test time. In this case, we can simply use the Λ_t from the task ID given. On the other hand, in incremental class learning, we are not given task IDs during testing. This is the more difficult case, with many earlier continual learning methods tending to do poorly. We address this challenge by using the approach described in Section 2.4, inferring the task identity by using the training statistics at an intermediate layer. For task inference in our incremental-class experiments, we consider ϕ in (13) to be the representation after the first layer, as it performed best. We purposely have chosen to *not* supplement IBP-WF with replay, to isolate the advantages of using IBP and weight factors. This puts IBP-WF at a disadvantage compared to replay-based methods. Nevertheless, IBP-WF outperforms or is comparable to replay-based methods. Note that *with* replay (where we no longer freeze the IBP "dishes" after they are learned by a given task), the IBP-WF performance is likely to improve further (via backward transfer); we reserve that for future work.

We additionally perform an ablation study over the effect of the IBP, and then visualize some IBP-WF weight factors to verify some of its behavior. The description of baselines and the training details are in Appendices F and G, respectively. Ablation studies for IBP-WF's α and κ are also included in Appendices H.1 and H.2. All experiments are run on a NVIDIA Titan X GPU.

4.1 Datasets and Architectures

We evaluate IBP-WF on a number of common continual learning benchmarks. For each, the model is trained on a series of classification tasks arriving in sequence. This is done *without* revisiting the data from previous

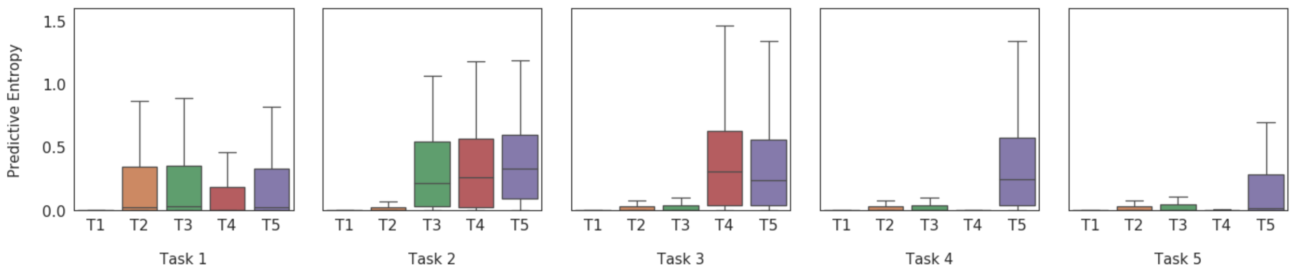
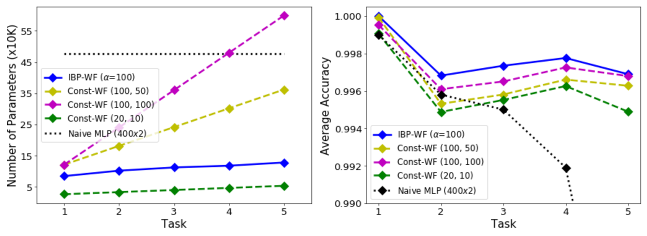
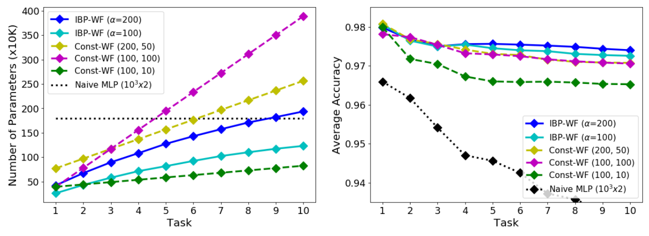


Figure 2: Uncertainty in the incremental class setting for Split MNIST dataset. Each plot depicts the uncertainty on the test sets after training on each task sequentially. The y-axis denotes the uncertainty (as the predictive entropy in nats), and x-axis denotes the test sets ($\mathcal{T}_1$ through $\mathcal{T}_5$) for each task.



(a) Split MNIST (5 tasks)



(b) Permuted MNIST (10 tasks)

Figure 3: Parameter usage (left) and Average accuracy (right) for IBP-WF and Const-WF. We see that IBP-WF performs favorably by learning the required number of factors for each task in contrast to Const-WF.

tasks, unless otherwise stated (*e.g.,* some baselines use a memory buffer for replay to relax this constraint). The standard train/validation/test splits were used.

Split MNIST Following Zenke et al. (2017), the 10 digit classes of the MNIST (LeCun et al., 1998) dataset are split into a series of 5 binary classification tasks: 0 vs 1, 2 vs 3, 4 vs 5, 6 vs 7, and 8 vs 9. In the incremental task setting, where the task ID is given, this reduces to a binary classification problem during testing. In the incremental class setting, without task labels, each model must predict one of $2t$ classes, up to a maximum of 10 once all tasks have been seen.

Permuted MNIST First used to characterize catastrophic forgetting in neural networks by Goodfellow et al. (2013), Permuted MNIST has remained a common continual learning benchmark. The first classification task is typically chosen to be the MNIST dataset, unchanged. Each subsequent task consists of the same 10-way digit classification, but with the pixels of the entire MNIST dataset randomly permuted in a consistent manner. An arbitrary number of tasks can be generated in this manner; for our experiments, we use 10 tasks. In the incremental-task setting, test-time evaluation is a 10-way classification problem, while in incremental class learning we have up to 100 classes.

Results are shown in Table 1. In addition to these results, we also compare against additional baselines in Appendix F. IBP-WF outperforms other methods in most cases and is inferior only to replay-based methods in one setting. Unlike replay-based methods though, IBP-WF does not require saving data examples or separately learning bulky generative models. Compared with non-replay methods, we see significant improvement, especially in the incremental class setting. Additionally, due to the Bayesian nature of IBP-WF, one can quantify the predictive uncertainty, which is a desirable property of a model, especially in decision making. Uncertainty estimates can also be used to detect out-of-distribution samples (Gal, 2016) and adversarial attacks (Smith & Gal, 2018). We demonstrate the former in Figure 2, where data from unseen tasks can be identified by the model's significantly higher uncertainty. See Appendix E for additional details on uncertainty estimation methodology.

Split CIFAR10 We split the CIFAR10 (Krizhevsky, 2009) dataset into a sequence of 5 binary classification tasks (see Figure 5 for the class pairings). Similar to Split MNIST, this is a binary classification problem at test time in the incremental task setting, and $2t$-wise classification in the incremental class setting.

State-of-the-art classification performance on CIFAR10 is typically achieved with convolutional neural networks. We demonstrate that IBP-WF can scale by using ResNet-20 (He et al., 2016) as our architecture, with separate batch normalization layers for each task. For task inference at test time, we take the average

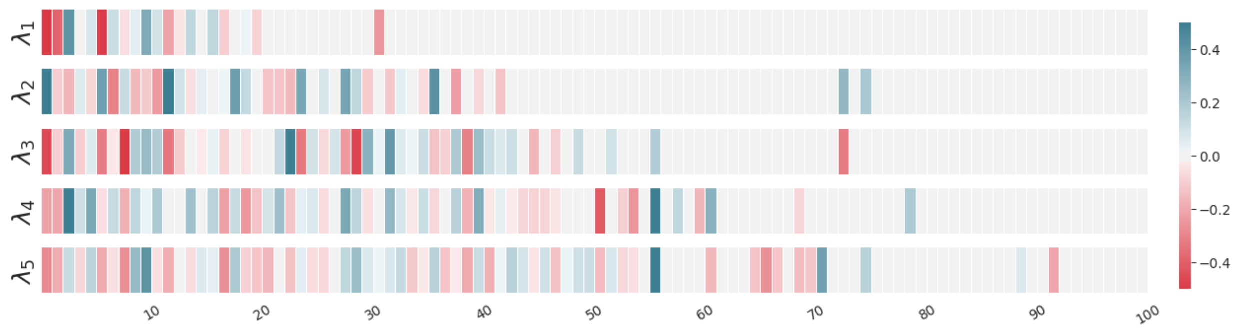


Figure 4: (Horizontal axis: Weight factors in the dictionary, vertical axis: Factor scores for 5 tasks in Split CIFAR10). High overlap in *earlier* factor scores indicate that tasks reuse factors that were learned by previously seen tasks. The IBP prior encourages the reuse of previously learned factors, while inducing sparsity on the total number of active factors.

Table 2: The average accuracy of all seen tasks after learning the task sequence.

Method	Replay	Split CIFAR10	
		Incremental Task	Incremental Class
Adagrad		71.56 ± 1.73	19.59 ± 0.02
L_2		74.36 ± 0.83	16.86 ± 0.08
EWC		75.91 ± 1.64	18.84 ± 0.06
Online EWC		88.34 ± 1.06	17.54 ± 0.34
SI		87.19 ± 2.06	19.06 ± 0.09
MAS		85.68 ± 1.36	16.29 ± 0.14
Naive rehearsal	✓	87.79 ± 0.88	34.24 ± 1.38
IBP-WF (Ours)		**90.94** ± 2.65	**40.40** ± 0.21

across spatial dimensions H and W to get the feature statistics ϕ and then proceed with the parameter estimation procedure introduced in (14). We keep a buffer of 400 images from previous tasks for the naive rehearsal baseline. Table 2 shows the results on Split CIFAR10. We again see that IBP-WF performs well relative to the baseline methods.

4.2 IBP Ablation Study

To demonstrate the benefits of the IBP prior, we perform an ablation study comparing IBP-WF with a variant without the IBP for weight factor selection and expansion, in which factor usage and model growth must be manually set. As the expansion rate is constant, we call this Const-WF(ν, ω), parameterized by the starting number of factors (ν) in the first task and number of new factors (ω) added per task. We compare IBP-WF with Const-WF for several corresponding settings in Figure 3. Importantly, the IBP allows for automatic expansion as needed, with sublinear harmonic parameter growth, without human intervention. IBP-WF's factor reuse also results in positive transfer, yielding better accuracy, despite having fewer parameters than Const-WF.

4.3 Visualizations

Weight factor utilization Central to our method is the IBP prior that controls the growth of the number of factors and encourages the model to reuse factors. This controlled growth makes IBP-WF more efficient than independent models, while the reuse allows for positive knowledge transfer between tasks. The factor usage is visualized by plotting the expected factor scores $\mathbb{E}[\boldsymbol{\lambda}_t]$ for the first layer of a model trained on the Split CIFAR10 in Figure 4. One can clearly see the impact of using the IBP as regularization: early factors are prioritized in earlier tasks, and new factors are used with later tasks. We emphasize that the number of new parameters is not defined directly by some preset schedule, but rather is inferred from the data.

The sparsity induced by the IBP can also be seen. With each new task, an increasing number of factor scores have nonzero entries, as the model adapts the number of factors F based on the task objective. However, even for a later task, the probability of a factor being active remains high for only a few. As a result, each *draw* from the posterior tends to be sparse, regularized by the IBP to have α active factors in expectation. Another appealing aspect of using an IBP is that the *rate* of allocating a new factor decreases with tasks. Finally, following the “rich-get-richer” principle (here for “rich”, or widely utilized, factors), the IBP encourages that factors are reused based on the total number of prior tasks using it.

CIFAR10 filters We also visualize the first layer convolutional representations for a model trained with IBP-WF on Split CIFAR10 (Figure 5). We observe an interesting property of the model: the feature maps in earlier tasks are similar compared to the diverse feature maps for later tasks. This can be attributed to early tasks using few factors due to the regularizing effect induced by the IBP on the rank of the weight filter.

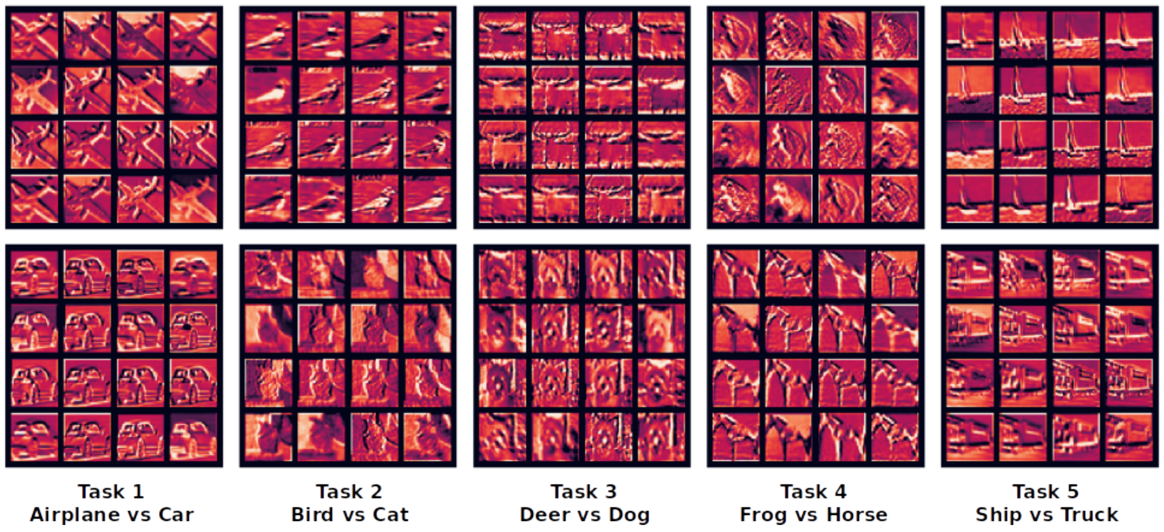


Figure 5: The first layer representations for each class in a trained IBP-WF ResNet-20. Each 4×4 grid shows the feature representations after convolution using 16 kernels in the first layer.

However, as the model observes more data, the filters become more diverse since the number of active factors increase, resulting in varied features maps. This shows that as the model sees more tasks, the complexity of the layer increases through the newly invoked filters.

5 Conclusions

An expansion-based approach combining a dictionary of weight factors with the IBP has been introduced, which we call IBP-WF. This synergy provides important characteristics within the context of continual learning, including knowledge reuse across tasks, data-driven model expansion, and catastrophic-forgetting mitigation. We also propose a simple and efficient task-inference scheme, utilizing feature statistics for each task and enabling operation in incremental class settings. A number of experiments on common continual-learning benchmarks show the effectiveness of IBP-WF. Ablation studies demonstrate the effectiveness of the IBP over linear expansion, and visualizations of the inferred factor scores and weights illustrate the regularization effects of our method. Notably, the motivation of IBP-WF is orthogonal to a number of other continual learning strategies, and combining some of these with IBP-WF is a promising direction for future work. For example, IBP-WF can readily be augmented with replay, and the Dirichlet process mixture model (as in Lee et al. (2020)) may be a natural Bayesian nonparametric alternative to our feature statistic method for inferring tasks.

Acknowledgements: This work was supported by the DARPA L2M Program.

References

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory Aware Synapses: Learning What (not) to Forget. *European Conference on Computer Vision*, 2018.

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-Free Continual Learning. *Computer Vision and Pattern Recognition*, 2019.

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

John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12 (Jul):2121–2159, 2011.

Sebastian Farquhar and Yarin Gal. Towards Robust Evaluations of Continual Learning. *arXiv preprint arXiv:1805.09733*, 2018.

Sebastian Farquhar and Yarin Gal. A Unifying Bayesian View of Continual Learning. *arXiv preprint arXiv:1902.06494*, 2019.

Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

Zoubin Ghahramani and Thomas L Griffiths. Infinite Latent Feature Models and the Indian Buffet Process. *Neural Information Processing Systems*, 2006.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. *Neural Information Processing Systems*, 2014.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An Empirical Investigation of Catastrophic Forgetting in Gradient-based Neural Networks. *arXiv preprint arXiv:1312.6211*, 2013.

I. S. Gradshteyn and I. M. Ryzhik. *Table of integrals, series, and products*. Elsevier/Academic Press, Amsterdam, seventh edition, 2007.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *Computer Vision and Pattern Recognition*, 2016.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *Neural Information Processing Systems*, 2017.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 1997.

Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating Continual Learning Scenarios: A Categorization and Case for Strong Baselines. *arXiv preprint arXiv:1810.12488*, 2018.

Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, Picking and Growing for Unforgetting Continual Learning. *Neural Information Processing Systems*, 2019.

Samuel Kessler, Vu Nguyen, Stefan Zohren, and Stephen Roberts. Hierarchical indian buffet neural networks for bayesian continual learning. *arXiv preprint arXiv:1912.02290*, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

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*, 2017.

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. 2009.

Abhishek Kumar, Sunabha Chatterjee, and Piyush Rai. Nonparametric Bayesian Structure Adaptation for Continual Learning. *arXiv preprint arXiv:1912.03624*, 2019.

Ponnambalam Kumaraswamy. A Generalized Probability Density Function for Double-Bounded Random Processes. *Journal of Hydrology*, 1980.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks. *Neural Information Processing Systems*, 2018.

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

Zhizhong Li and Derek Hoiem. Learning Without Forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017.

Kevin J Liang, Geert Heilmann, Christopher Gregory, Souleymane O Diallo, David Carlson, Gregory P Spell, John B Sigman, Kris Roe, and Lawrence Carin. Automatic Threat Recognition of Prohibited Items at Aviation Checkpoint with X-ray Imaging: A Deep Learning Approach. *SPIE Anomaly Detection and Imaging with X-Rays (ADIX) III*, 2018a.

Kevin J Liang, Chunyuan Li, Guoyin Wang, and Lawrence Carin. Generative Adversarial Network Training is a Continual Learning Problem. *arXiv preprint arXiv:1811.11083*, 2018b.

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

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

Michael McCloskey and Neal J Cohen. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. *The Psychology of Learning and Motivation*, 1989.

Joseph Victor Michalowicz, Jonathan M. Nichols, and Frank Bucholtz. *Handbook of Differential Entropy*. Chapman and Hall/CRC, 2013. ISBN 1466583169.

Eric Nalisnick and Padhraic Smyth. Stick-breaking variational autoencoders, 2016.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? *arXiv preprint arXiv:1810.09136*, 2018.

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

German Ignacio Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual Lifelong Learning with Neural Networks: A Review. *Neural Networks*, 2019.

Roger Ratcliff. Connectionist Models of Recognition Memory: Constraints Imposed by Learning and Forgetting Functions. *Psychology Review*, 1990.

Dezső Ribli, Anna Horváth, Zsuzsa Unger, Péter Pollner, and István Csabai. Detecting and Classifying Lesions in Mammograms with Deep Learning. *Scientific reports*, 8 (1):1–7, 2018.

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

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience Replay for Continual Learning. *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. *arXiv preprint arXiv:1606.04671*, 2016.

Jonathan Schwarz, Jelena Luketina, Wojciech M Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & Compress: A Scalable Framework for Continual Learning. *International Conference on Machine Learning*, 2018.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual Learning with Deep Generative Replay. *Neural Information Processing Systems*, 2017.

Lewis Smith and Yarin Gal. Understanding measures of uncertainty for adversarial example detection. *arXiv preprint arXiv:1803.08533*, 2018.

Yee Whye Teh, Dilan Grür, and Zoubin Ghahramani. Stick-breaking construction for the indian buffet process. *Artificial Intelligence and Statistics*, 2007.

Gido M van de Ven and Andreas S Tolias. Generative Replay with Feedback Connections as a General Strategy for Continual Learning. *arXiv preprint arXiv:1809.10635*, 2018.

Gido M van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

Tom Veniat, Ludovic Denoyer, and Marc'Aurelio Ranzato. Efficient continual learning with modular networks and task-driven priors. *International Conference on Learning Representations*, 2021.

Ronald J Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine learning*, 1992.

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

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

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

Jeffrey O Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. Side-Tuning: Network Adaptation via Additive Side Networks. *arXiv preprint arXiv:1912.13503*, 2019.