Chapter 4

Lifelong Learning for Deep Neural Networks with Bayesian Principles

Cuong V. Nguyen[1*], Siddharth Swaroop[2*], Thang D. Bui[3], Yingzhen Li[4], Richard E. Turner[5]

[1] *Department of Mathematical Sciences, Durham University, UK*
[2] *Department of Computer Science, Harvard University, USA*
[3] *School of Computing, Australian National University, Australia*
[4] *Department of Computing, Imperial College London, UK*
[5] *Department of Engineering, University of Cambridge, UK*

Abstract

This chapter describes a general Bayesian framework for lifelong learning of artificial neural networks that can handle catastrophic forgetting in a principled way. The framework can be applied to both discriminative and generative models as well as the task-aware and task-agnostic settings. We introduce the variational continual learning algorithm, a realisation of this framework that uses online variational inference with a small memory or coreset for effective lifelong learning. We examine various practical considerations when using this algorithm and show that it performs competitively against other lifelong learning approaches on different benchmarks. We also discuss several improvements to the algorithm and outline some future research directions for Bayesian lifelong learning.

*The first two authors contributed equally to this chapter.

Preprint of a book chapter in *Towards Human Brain Inspired Lifelong Learning*, World Scientific, 2024. `https://doi.org/10.1142/9789811286711_0004`.

Correspondence to: Cuong V. Nguyen (viet.c.nguyen@durham.ac.uk)

4.1 Introduction

Deep learning provides a suite of approaches for fitting multi-layer neural networks to data. Deep neural networks have been shown to achieve remarkable prediction accuracy on several problems in important domains such as computer vision, speech processing, and natural language processing. Training these models is usually done off-line and separately for each individual task. Attempts to develop an effective lifelong learning process for neural networks, so that they can be trained in an incremental way to solve a set of different tasks, have been hindered by either catastrophic forgetting, where models quickly forget previous tasks when trained on a new task using vanilla stochastic gradient descent (French, 1999; Goodfellow *et al.*, 2014), or catastrophic intransigence (Chaudhry *et al.*, 2018) where they fail to adapt sufficiently to new data.

In this chapter, we develop a framework for performing lifelong learning in deep neural networks using a probabilistic modelling and inference. Specifically, we formalise a general Bayesian framework that can handle lifelong learning in deep models in a natural and principled way. If exact inference can be performed, our framework can automatically avoid catastrophic forgetting during the learning process. The framework is also general in the sense that it can cover both discriminative and generative deep learning models, as well as the task-aware (Kirkpatrick *et al.*, 2017; Abati *et al.*, 2020) and task-agnostic (Aljundi *et al.*, 2019; Zeno *et al.*, 2021) lifelong learning settings.

As a realisation of this framework, we introduce *Variational Continual Learning* (VCL) (Nguyen *et al.*, 2018; Swaroop *et al.*, 2018), one of the first works to use the Bayesian framework to derive a new lifelong learning approach. VCL uses online variational inference to approximate the posterior after each task is observed. Using a previously developed method for variational inference in neural networks, called Bayes by Backprop (Blundell *et al.*, 2015), VCL effectively learns the variational parameters of the approximate posterior at each iteration by maximising the variational lower bound. This maximisation process is done using a stochastic gradient-based optimiser such as Adam (Kingma and Ba, 2015). To further mitigate catastrophic forgetting due to successive posterior approximations, we enhance VCL with an episodic memory (called a coreset) and show how variational inference can be performed for lifelong learning in the presence of such coresets.

We also emphasise several practical considerations when using VCL, namely the importance of long training runs, the use of the local reparameterisation trick (Kingma *et al.*, 2015), and sensible initialisation of the variational parameters. In practice, these techniques can greatly improve model accuracy over a naive application of Bayes by Backprop. Our experiments show that VCL with these improvements performs competitively against several state-of-the-art methods on the popular Split MNIST and Permuted MNIST benchmarks for lifelong learning. We also explore the pruning effect of VCL and the role it plays in utilising model capacity for learning a sequence of tasks.

Finally, we end the chapter by briefly discussing several improvements to VCL that have been developed in recent years, such as natural-gradient variational inference methods (Chen *et al.*, 2018; Khan *et al.*, 2018; Osawa *et al.*, 2019), Generalised VCL with FiLM layers (Loo *et al.*, 2021), function-space regularisation (Titsias *et al.*, 2020; Pan *et al.*, 2020), and the Stein gradient method for choosing coresets (Chen *et al.*, 2018). We also outline future research directions for using Bayesian inference to develop a realistic lifelong learning procedure for deep neural networks.

4.2 Lifelong Learning from the Bayesian Perspective

In this section, we formalise the general Bayesian approach to lifelong learning. Consider the lifelong learning setting where there is a potentially infinite stream of *data batches* $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \ldots$ that arrive over time, and we have a model with parameters $\boldsymbol{\theta}$ that we need to continually adapt to the observed data. Under the Bayesian framework, we place a prior distribution $p(\boldsymbol{\theta})$ over $\boldsymbol{\theta}$, and every time we observe a batch $\mathcal{D}_i$, we update our posterior using the likelihood $p(\mathcal{D}_i|\boldsymbol{\theta})$. Formally, the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D}_{1:T})$ after observing T data batches $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_T$ is obtained using Bayes' rule:

$$p(\boldsymbol{\theta}|\mathcal{D}_{1:T}) \propto p(\boldsymbol{\theta}) \prod_{i=1}^{T} p(\mathcal{D}_i|\boldsymbol{\theta}) \propto p(\boldsymbol{\theta}|\mathcal{D}_{1:T-1}) p(\mathcal{D}_T|\boldsymbol{\theta}). \tag{4.1}$$

In the above equation, $p(\boldsymbol{\theta}|\mathcal{D}_{1:T-1}) \propto p(\boldsymbol{\theta}) \prod_{i=1}^{T-1} p(\mathcal{D}_i|\boldsymbol{\theta})$ is the previous posterior obtained after observing the first $T-1$ batches. This equation provides a natural way to handle lifelong learning using Bayesian principles: at every time step T, we only need to maintain the current posterior $p(\boldsymbol{\theta}|\mathcal{D}_{1:T})$, and we will continually update this posterior in light of new observations using Bayes' rule.

The formulation for performing Bayesian lifelong learning outlined above is very general. It covers both the case where the probabilistic model is discriminative and the case where it is generative. It also covers the task-aware and task-agnostic lifelong learning settings. We now outline these settings in more detail.

Discriminative model setting. For discriminative models such as classifiers, for any $t \in \{1, 2, \ldots, T\}$, each data batch $\mathcal{D}_t$ consists of N_t labelled examples $\{(\boldsymbol{x}_t^{(n)}, y_t^{(n)})\}_{n=1}^{N_t}$ and Eq. (4.1) can be rewritten as:

$$p(\boldsymbol{\theta}|\mathcal{D}_{1:T}) \propto p(\boldsymbol{\theta}|\mathcal{D}_{1:T-1}) \prod_{n=1}^{N_T} p(y_T^{(n)}|\boldsymbol{\theta}, \boldsymbol{x}_T^{(n)}), \tag{4.2}$$

where $p(y|\boldsymbol{\theta}, \boldsymbol{x})$ is the likelihood of the parameters $\boldsymbol{\theta}$ from the labelled example $(\boldsymbol{x}, y)$.

Generative model setting. For generative models, such as deep generative models which are often trained using the variational auto-encoder approach (Kingma and Welling, 2014; Rezende *et al.*, 2014), for any $t \in \{1, 2, \ldots, T\}$, each data batch $\mathcal{D}_t$ consists of N_t unlabelled examples $\{\boldsymbol{x}_t^{(n)}\}_{n=1}^{N_t}$, and Eq. (4.1) can be rewritten as:

$$p(\boldsymbol{\theta}|\mathcal{D}_{1:T}) \propto p(\boldsymbol{\theta}|\mathcal{D}_{1:T-1}) \prod_{n=1}^{N_T} p(\boldsymbol{x}_T^{(n)}|\boldsymbol{\theta}), \tag{4.3}$$

where $p(\boldsymbol{x}|\boldsymbol{\theta})$ is the likelihood of $\boldsymbol{\theta}$ from the unlabelled example $\boldsymbol{x}$.

Task-aware lifelong learning setting. Task-aware lifelong learning refers to the setting where the learner is aware of the time when a task switches to a new task (Kirkpatrick *et al.*, 2017; Abati *et al.*, 2020). This setting usually requires additional knowledge of which task each data batch $\mathcal{D}_i$ belongs to, and this knowledge will allow the learner to adjust its learning procedure accordingly. For deep neural networks, a typical way to deal with task changes in the task-aware setting is to extend the network architecture with task-specific parameters whenever a new task arrives, while maintaining a set of shared network parameters for all tasks. The task-specific parameters can simply be a classifier head (Zenke *et al.*, 2017) or an entire network column (Rusu *et al.*, 2016). Fig. 4.1 illustrates multi-head networks that use a separate head for each task.

The Bayesian lifelong learning formalism encapsulated in Eq. (4.1) can easily handle these types of network extension. Specifically, whenever a

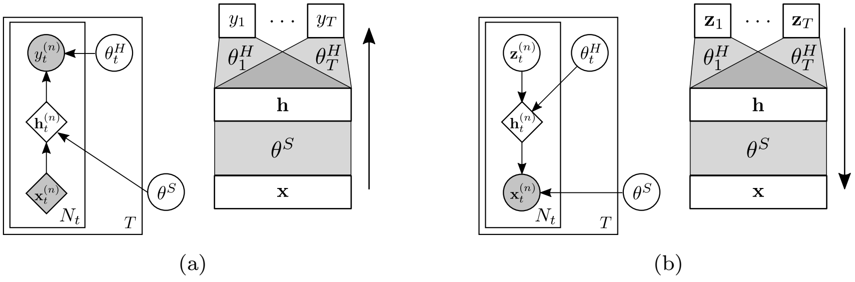

Fig. 4.1: Schematics of multi-head networks, including both the probabilistic graphical model (left) and network architecture (right), reproduced from Nguyen *et al.* (2018). (a) A multi-head discriminative model showing how network parameters might be shared. The lower-level network is parameterised by the variables $\boldsymbol{\theta}^S$ and is shared across multiple tasks. Each task $t \in \{1, 2, \ldots, T\}$ has its own "head network" $\boldsymbol{\theta}_t^H$ mapping to the outputs from a common hidden representation. The full set of parameters is therefore $\boldsymbol{\theta} = \{\boldsymbol{\theta}_{1:T}^H, \boldsymbol{\theta}^S\}$. (b) A multi-head generative model (see Section 4.3.3 for details) with shared network parameters. For each task t, the head networks $\boldsymbol{\theta}_t^H$ generate the intermediate level representations from the latent variables $\mathbf{z}_t$.

new task T arrives, we add the task-specific parameters $\boldsymbol{\theta}_T^H$ to the network, resulting in a new set of parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_{1:T}^H, \boldsymbol{\theta}^S\}$, and extend the current posterior $p(\boldsymbol{\theta}_{1:T-1}^H, \boldsymbol{\theta}^S | \mathcal{D}_{1:T-1})$ to a new posterior $p(\boldsymbol{\theta}_{1:T}^H, \boldsymbol{\theta}^S | \mathcal{D}_{1:T-1})$ with some "prior" distribution for $\boldsymbol{\theta}_T^H$. Then we can update the posterior for the whole model, including $\boldsymbol{\theta}_T^H$, using Eq. (4.1) and data $\mathcal{D}_T$ for the new task.

Task-agnostic lifelong learning setting. In contrast to the task-aware setting above, task-agnostic or task-free lifelong learning considers a more difficult scenario where the learner does not know the exact time when a task switches (Aljundi *et al.*, 2019; Jerfel *et al.*, 2019; He *et al.*, 2020; Zeno *et al.*, 2021; Jin *et al.*, 2021). In other words, the learner only receives a data batch $\mathcal{D}_i$ at each time step without knowing which task $\mathcal{D}_i$ belongs to. In this case, we typically do not use multi-head architectures and maintain only a single shared network for all tasks. Our formulisation in Eq. (4.1) can be applied directly to this setting without any modifications. Other Bayesian approaches such as natural VCL (Tseran *et al.*, 2018) or Bayesian structural adaptation (Kumar *et al.*, 2021) can also be applied to this task-agnostic setting.

Remarks. In principle, our Bayesian framework for lifelong learning presented in this section can handle continual learning naturally without suffering from catastrophic forgetting or catastrophic intransigence. Ideally, if we can always maintain the exact posterior $p(\boldsymbol{\theta}|\mathcal{D}_{1:T})$ after observing the last data batch $\mathcal{D}_T$, we would not lose any information from the prior and previous observations, and thus by design would be able to overcome catastrophic forgetting.

In practice, however, the posterior distributions are usually intractable, especially for complex models like neural networks. Hence, approximations are usually required to maintain the tractability of the (approximate) posteriors. Typically, after observing the data batch $\mathcal{D}_T$, an approximation method will replace the intractable true posterior $p(\boldsymbol{\theta}|\mathcal{D}_{1:T})$ by a tractable approximate distribution $q_T(\boldsymbol{\theta})$, which would be used as the prior to compute the next approximate posterior. Formally,

$$p(\boldsymbol{\theta}|\mathcal{D}_{1:T}) \approx q_T(\boldsymbol{\theta}) = \text{proj}\Big(q_{T-1}(\boldsymbol{\theta})p(\mathcal{D}_T|\boldsymbol{\theta})\Big), \tag{4.4}$$

where $\text{proj}(p^*(\boldsymbol{\theta}))$ is a projection operation on an intractable, un-normalised distribution $p^*(\boldsymbol{\theta})$ that returns a tractable, normalised approximate distribution. Different projection operations can be used in Eq. (4.4), such as Laplace's approximation, variational inference, moment matching, or importance sampling. These projections respectively lead to online updating methods known as Laplace propagation (Smola *et al.*, 2004), online variational inference (Ghahramani and Attias, 2000; Sato, 2001; Broderick *et al.*, 2013), assumed density filtering (Maybeck, 1982), and sequential Monte Carlo (Liu and Chen, 1998). In the next section, we will extend the variational inference approach to online learning to support continual learning of neural networks.

4.3 Variational Inference for Lifelong Learning

In this section, we describe Variational Continual Learning (VCL), a method for continual/lifelong learning with deep neural networks using variational inference. VCL can be used together with an episodic memory to reduce catastrophic forgetting in these networks. We illustrate how VCL can be applied to discriminative and generative models, and note some practical considerations when using the method. Finally, we show some experimental evaluations and discuss the pruning effects of this method.

4.3.1 *Variational Continual Learning*

VCL applies Eq. (4.4) above with a projection operation that minimises the Kullback–Leibler (KL) divergence between the input distribution and the output approximate distribution. More specifically, the approximate posterior $q_T(\boldsymbol{\theta})$ in Eq. (4.4) is computed by:

$$q_T(\boldsymbol{\theta}) = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}\Big(q(\boldsymbol{\theta}) \parallel \frac{1}{Z_T} q_{T-1}(\boldsymbol{\theta}) p(\mathcal{D}_T|\boldsymbol{\theta})\Big), \tag{4.5}$$

where the minimisation is considered over a family $\mathcal{Q}$ of distributions, and Z_T is the normalising constant of $q_{T-1}(\boldsymbol{\theta})p(\mathcal{D}_T|\boldsymbol{\theta})$ that is usually not required to solve this optimisation problem. During the learning process for VCL, the zeroth approximate posterior is set to be the prior, $q_0(\boldsymbol{\theta}) = p(\boldsymbol{\theta})$, and subsequent approximate posteriors are computed using Eq. (4.5) every time a new batch of data $\mathcal{D}_T$ is received.

As discussed in Section 4.2, if we can maintain the true posterior at every iteration, we will be able to avoid catastrophic forgetting. For instance, if the family $\mathcal{Q}$ contains all of the true posteriors and assuming we can always find the true posterior when solving the minimisation problem (4.5), then catastrophic forgetting can be overcome. In practice, however, we usually choose a simpler distribution family $\mathcal{Q}$, such as Gaussian distributions, leading to information loss that can cause catastrophic forgetting to occur. To mitigate this potential forgetting problem, VCL can be extended to include a small episodic memory (Lopez-Paz and Ranzato, 2017), also called a coreset, which contains representative examples from previous batches of data to help the model refresh important information before making predictions.

This *Coreset VCL* algorithm can be described as follows. At every iteration T, we observe a new batch of data $\mathcal{D}_T$ and construct a coreset C_T from $\mathcal{D}_T \cup C_{T-1}$, where C_{T-1} is the previous coreset. Instead of maintaining the approximate posterior $q_T(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\mathcal{D}_{1:T})$, Coreset VCL maintains an approximate posterior $\tilde{q}_T(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\mathcal{D}_{1:T} \setminus C_T)$ that approximates the true posterior after observing all data except those in the current coreset. In the lifelong learning setting, $\tilde{q}_T(\boldsymbol{\theta})$ can be updated using the following recursion:

$$\begin{aligned}\tilde{q}_T(\boldsymbol{\theta}) &= \mathrm{proj}\Big(\tilde{q}_{T-1}(\boldsymbol{\theta})\ p(\mathcal{D}_T \cup C_{T-1} \setminus C_T|\boldsymbol{\theta})\Big)\\ &= \arg\min_{q \in \mathcal{Q}} \mathrm{KL}\Big(q(\boldsymbol{\theta}) \parallel \frac{1}{\tilde{Z}}\ \tilde{q}_{T-1}(\boldsymbol{\theta})\ p(\mathcal{D}_T \cup C_{T-1} \setminus C_T|\boldsymbol{\theta})\Big), \end{aligned}\tag{4.6}$$

where $p(\mathcal{D}_T \cup C_{T-1} \setminus C_T|\boldsymbol{\theta})$ is the likelihood contribution from data points that are either not selected for the current coreset or removed from the previous coreset, and $\tilde{Z}$ is the normalisation constant. The recursion (4.6) ensures the approximate posterior $\tilde{q}_T(\boldsymbol{\theta})$ incorporates all non-coreset observations in the first T iterations.

When making predictions (e.g. in a classification problem) on a validation or test set, Coreset VCL first refreshes the important information stored in the current coreset C_T by making the variational update:

$$q_T(\boldsymbol{\theta}) = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}\Big(q(\boldsymbol{\theta}) \parallel \frac{1}{Z}\tilde{q}_T(\boldsymbol{\theta})p(C_T|\boldsymbol{\theta})\Big), \tag{4.7}$$

and then uses $q_T(\boldsymbol{\theta})$ to make prediction on a test input $\boldsymbol{x}^*$:

$$p(y^*|\boldsymbol{x}^*, \mathcal{D}_{1:T}) \approx \int q_T(\boldsymbol{\theta})p(y^*|\boldsymbol{\theta}, \boldsymbol{x}^*)\mathrm{d}\boldsymbol{\theta}. \tag{4.8}$$

By using Eq. (4.7), $q_T(\boldsymbol{\theta})$ incorporates all observations $\mathcal{D}_{1:T}$ and thus can be thought of as an approximation of $p(\boldsymbol{\theta}|\mathcal{D}_{1:T})$. Furthermore, applying this equation immediately before making predictions enables the trained model to be less forgetful of the important information in the coreset C_T.

In principle, the coreset C_T can be constructed from $\mathcal{D}_T \cup C_{T-1}$ using various methods. The simplest methods use random sampling or K-center clustering to construct the coreset (Nguyen *et al.*, 2018). More sophisticated methods include using Stein gradients (Chen *et al.*, 2018), bilevel optimisation (Borsos *et al.*, 2020), or diverse samples selection (or rainbow memory) (Bang *et al.*, 2021).

4.3.2 *VCL for Discriminative Models*

In this section, we shall illustrate how VCL can be used with discriminative probabilistic models. Note that the schematics for discriminative multi-head networks are shown in Figure 4.1(a). We will focus on vanilla VCL without the coreset here, but the formulation below can be extended easily to Coreset VCL.

For discriminative models, for any $t \in \{1, 2, \ldots, T\}$, each data batch $\mathcal{D}_t$ contains N_t labelled examples $\{(\boldsymbol{x}_t^{(n)}, y_t^{(n)})\}_{n=1}^{N_t}$. The update rule for VCL in Eq. (4.5) can be rewritten as maximising the variational lower bound to the online marginal log-likelihood:

$$q_T(\boldsymbol{\theta}) = \arg\max_{q \in \mathcal{Q}} \mathcal{L}^T_{\mathrm{VCL}}(q(\boldsymbol{\theta})), \tag{4.9}$$

where

$$\mathcal{L}^T_{\mathrm{VCL}}(q(\boldsymbol{\theta})) = \sum_{(\mathbf{x},y)\in\mathcal{D}_T} \mathbb{E}_{q(\boldsymbol{\theta})}\Big[\log p(y|\boldsymbol{\theta}, \mathbf{x})\Big] - \mathrm{KL}(q(\boldsymbol{\theta})||q_{T-1}(\boldsymbol{\theta})). \tag{4.10}$$

If the family $\mathcal{Q}$ comprises Gaussian mean-field approximate posteriors, then $q(\boldsymbol{\theta}) = \prod_{d=1}^{D} \mathcal{N}(\theta_d; \mu_d, \sigma_d^2)$, where D is the number of dimensions of $\boldsymbol{\theta}$ and $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_D)$. In this case, we can find $q(\boldsymbol{\theta})$ by maximising Eq. (4.9) with respective to the variational parameters $\{\mu_d, \sigma_d\}_{d=1}^{D}$ which are the mean and the diagonal covariance of the Gaussian. This optimisation problem can be done using Bayes by Backprop (Blundell *et al.*, 2015) and requires Monte Carlo approximation of the expected log-likelihood term in the approximate marginal likelihood.

4.3.3 *VCL for Generative Models*

This section discusses how VCL can be applied to Bayesian generative models, specifically to deep generative models as often learned using the variational auto-encoder (VAE) framework (Kingma and Welling, 2014; Rezende *et al.*, 2014). In a standard VAE setup, we model the likelihood of each example $\boldsymbol{x}$ as $p(\boldsymbol{x}|\boldsymbol{\theta}) = \int p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) p(\mathbf{z}) d\mathbf{z}$, where $\mathbf{z}$ are the latent variables and $p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})$ is defined by a neural network with parameters $\boldsymbol{\theta}$. Given an unlabelled dataset $\mathcal{D}$, the model is trained by maximising the following variational lower bound:

$$\mathcal{L}_{\text{VAE}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{\mathbf{x} \in \mathcal{D}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) p(\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \right], \tag{4.11}$$

where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is the approximate posterior of the latent variables (or the encoder), and $\boldsymbol{\phi}$ denotes the variational parameters of this approximate posterior.

In the lifelong learning setting, we sequentially receive batches of data, where for any $t \in \{1, 2, \ldots, T\}$, each data batch $\mathcal{D}_t$ contains N_t unlabelled examples $\{\boldsymbol{x}_t^{(n)}\}_{n=1}^{N_t}$. The schematics for multi-head generative models used in this setting are shown in Figure 4.1(b). In VCL, we use a Bayesian VAE model where we approximate the posterior $p(\boldsymbol{\theta}|\mathcal{D}_{1:T})$ with a variational distribution $q_T(\boldsymbol{\theta})$. We can then rewrite the VCL update rule (4.5) as maximising the full variational lower bound:

$$q_T(\boldsymbol{\theta}), \boldsymbol{\phi} = \arg\max_{q, \boldsymbol{\phi}} \mathcal{L}_{\text{VCL}}^{T}(q(\boldsymbol{\theta}), \boldsymbol{\phi}), \tag{4.12}$$

where

$$\mathcal{L}_{\text{VCL}}^{T}(q(\boldsymbol{\theta}), \boldsymbol{\phi}) = \sum_{\mathbf{x} \in \mathcal{D}_T} \mathbb{E}_{q(\boldsymbol{\theta})} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) p(\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \right] - \text{KL}(q(\boldsymbol{\theta}) || q_{T-1}(\boldsymbol{\theta})). \tag{4.13}$$

In this formulation, the encoder network $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is parameterised by $\boldsymbol{\phi}$ and can be either task-specific or partly shared among different tasks. Similarly to discriminative models, we can solve the optimisation problem (4.12) by using backpropagation.

4.3.4 *Relationship to Other Work*

VCL can sometimes be regarded as a regularisation-based approach to lifelong learning. Specifically, in the variational lower bound of Eq. (4.10) or Eq. (4.13), the first term on the right-hand side (usually called the expected log-likelihood term) favours approximate posteriors that maximise the expected log-likelihood of the training data. The second term in these equations, usually called the KL-to-prior term, serves as a regulariser that pulls the approximate posterior towards the prior (i.e. the previous approximate posterior).

Several lifelong learning methods for discriminative deep neural networks also employ a regularisation-based approach that maximises the following general objective at each iteration T:

$$\mathcal{L}^T(\boldsymbol{\theta}) = \sum_{(\mathbf{x},y)\in\mathcal{D}_T} \log p(y|\boldsymbol{\theta},\mathbf{x}) - \frac{1}{2}\lambda_T(\boldsymbol{\theta}-\boldsymbol{\theta}_{T-1})^\intercal \Sigma_{T-1}^{-1}(\boldsymbol{\theta}-\boldsymbol{\theta}_{T-1}), \quad (4.14)$$

where the matrix Σ_{T-1} controls the relative regularisation strength of each element of $\boldsymbol{\theta}$ and λ_T controls the overall strength of the regulariser.

There are several instances of this regularisation-based approach. For example, Laplace Propagation (Smola *et al.*, 2004) applies Laplace's approximation at each iteration, leading to the recursion $\Sigma_T^{-1} = \Phi_T + \Sigma_{T-1}^{-1}$, with $\Phi_T = -\nabla\nabla_{\boldsymbol{\theta}} \sum_{(\mathbf{x},y)\in\mathcal{D}_T} \log p(y|\boldsymbol{\theta},\mathbf{x})\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_T}$ and $\lambda_T = 1$. Another popular regularisation-based lifelong learning algorithm that also takes inspiration from Bayesian principles is Elastic Weight Consolidation (EWC) (Kirkpatrick *et al.*, 2017). This algorithm approximates $\Phi_T \approx \mathrm{diag}\big(\sum_{(\mathbf{x},y)\in\mathcal{D}_T} (\nabla_{\boldsymbol{\theta}} \log p(y|\boldsymbol{\theta},\mathbf{x}))^2 \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_T}\big)$ and modifies the regularisation term to $\frac{1}{2}\lambda_T \sum_{t=1}^{T-1}(\boldsymbol{\theta}-\boldsymbol{\theta}_{t-1})^\intercal \Phi_t(\boldsymbol{\theta}-\boldsymbol{\theta}_{t-1})$. The regularisation strengths in Σ_T^{-1} can also be computed using path integrals, as in the Synaptic Intelligence (SI) algorithm (Zenke *et al.*, 2017). The performance of SI and VCL has been analysed and shown to be correlated with the complexity of the observed tasks by Nguyen *et al.* (2019).

There have been several work that take Bayesian approaches to lifelong learning. Ritter *et al.* (2018) used Bayesian online learning with the block-diagonal Kronecker factored approximation of the Hessian to update their quadratic penalty for each new task. Ebrahimi *et al.* (2020) trained variational Bayesian neural networks using learning rates adapted from the uncertainty induced by the probability distribution of the network's parameters. Kessler *et al.* (2021) employed hierarchical Indian buffet process priors for Bayesian neural networks to allow the model to use resources more effectively in lifelong learning. Wang *et al.* (2021) developed a method for dynamically expanding and combining model parameters to actively forget knowledge that interferes with learning new tasks. Farquhar and Gal (2018) proposed using generative adversarial networks to generate data for old tasks that can be used together with VCL. Zeno *et al.* (2021) considered the task-agnostic lifelong learning setting and derived fixed-point updates for variational Bayesian neural networks. Chen *et al.* (2021) developed a generative regularisation approach to prevent catastrophic forgetting, and finally, Bayesian lifelong learning with non-stationary data was considered by Kurle *et al.* (2020).

4.3.5 *Practical Considerations for VCL*

We now discuss applications of VCL to discriminative lifelong learning benchmarks. As discussed in Section 4.3.2, we use Bayes by Backprop (Blundell *et al.*, 2015) to optimise the means and variances $\{\mu_d, \sigma_d\}_{d=1}^{D}$ of our mean-field Gaussian approximating family, optimising the objective function (4.10). This involves Monte Carlo sampling of the expected log-likelihood term, and we use Adam (Kingma and Ba, 2015) to optimise the function.

There are several details that, although seemingly technically trivial, are practically important and will greatly improve results over naively applying Bayes by Backprop for VCL. Crucially, we optimise for a much longer time than previous works, and do not early-stop. Although progress can sometimes appear to stall during optimisation, in reality progress is just extremely slow. We can also speed up the algorithm's convergence rate using two simple techniques: using the local reparameterisation trick and improving the initialisation of the variational parameters.

Local reparameterisation trick. We can employ the local reparameterisation trick (Kingma *et al.*, 2015) during Monte Carlo sampling of the likelihood term. Specifically, instead of sampling each (Gaussian) weight independently, we sample the pre-activation latent variables just before each neuron's non-linearity (this latent variable is a linear combination of the neuron's input weights). This leads to two improvements: (i) it reduces the variance of stochastic gradients during sampling, and (ii) it is marginally quicker as we sample fewer random variables. The first improvement is particularly important as it speeds up convergence drastically, substantially reducing the number of epochs required for convergence.

Initialisation. Another important trick is to improve the initialisation of the weights of our neural network when running the optimiser. We experimented with (i) initialising the mean-field Gaussian weights by setting the means at the maximum likelihood solution of a deterministic neural network and setting the variances to be small, (ii) initialising the means to be small and random and setting the variances to be small, and (iii) initialising at the means and variances of the prior. We find that using (ii) improves the convergence speed and reduces the standard deviation in final accuracy across many runs. Intuitively, this is because initialising randomly allows the network to quickly learn the best trade-off between the new task's data (the expected log-likelihood term) and information from previous tasks (the KL-to-prior term). Specifically, the means are randomly initialised to be of the order 10^{-1} and the variances are initialised of the order 10^{-3}.

4.3.6 *Experimental Evaluations of VCL*

In this section, we show some experimental evaluations of VCL after incorporating the practical tips in Section 4.3.5. We consider the following two lifelong learning benchmarks, both of which use the popular MNIST handwritten digit recognition dataset. Code to run all experiments is available at: `https://github.com/nvcuong/variational-continual-learning`.

Split MNIST. In this benchmark, we have to sequentially solve five binary classification tasks from the MNIST dataset: {0v1}, {2v3}, {4v5}, {6v7}, {8v9}. The challenge in Split MNIST is to obtain good performance on new tasks while retaining performance on old ones. We assume a *task-aware* setting and use the previously discussed multi-head setup. We train a one-hidden-layer neural network with 200 units and ReLU activation functions, and use a standard Normal prior for the first task. We train for 600 epochs (with batch size 256), using Adam learning rate 5×10^{-3}, and report mean and standard deviation over 10 runs. Results are summarised in Table 4.1. Without coresets, we achieve a final test accuracy of 98.5±0.4%, and with a coreset of 40 randomly chosen data points per

Table 4.1: Final average test accuracy on Split MNIST for various methods. Methods with an asterisk (*) use some sort of episodic memory. Results marked with a double asterisk (**) were taken or read from a graph in the respective papers.

Method	Hidden layer size	Final average test accuracy
VCL	{200}	98.5±0.4%
*VCL + 40 random coreset	{200}	98.2±0.4%
EWC (Kirkpatrick *et al.*, 2017)	{200}	63.1%
Laplace Propagation (Smola *et al.*, 2004)	{200}	61.2%
SI (Zenke *et al.*, 2017)	{200}	98.9%
Vadam VCL (Tseran *et al.*, 2018)	{256, 256}	99.2%**
UCL (Ahn *et al.*, 2019)	{256, 256}	99.7%**

task, we achieve 98.2±0.4% (coresets are not required to improve results on this benchmark). These are improvements compared to previous lifelong learning methods such as Elastic Weight Consolidation (Kirkpatrick *et al.*, 2017), which achieves 63.1%, and Laplace Propagation (Smola *et al.*, 2004), which achieves 61.2%.

Permuted MNIST. This benchmark consists of tasks received sequentially, each of which is the standard (10-way) MNIST classification task, with the pixels having undergone a fixed permutation randomly selected for each task. Ideally, a network with two or more hidden layers would use lower layer(s) to 'de-permute' the images and higher layer(s) to solve MNIST, which is then constant between tasks (we use a single-head setup as this is a task-agnostic setting). We train a two-hidden-layer model with 100 units in each layer and ReLU activation functions, again using a standard Normal prior for the first task. We train for 800 epochs (with batch size 1024), using Adam learning rate 5×10^{-3}, and report the mean and standard deviation over 5 runs. Results are summarised in Table 4.2. Without coresets, VCL achieves a final average test accuracy of 93±1%, and with coresets, VCL achieves 94.6±0.3%. For reference, training the same network but seeing all the data together (batch mode) has an accuracy of 97% (this is an upper bound on the performance possible with this model and inference scheme).

Table 4.2: Final average test accuracy on Permuted MNIST for various methods (results taken from respective papers). A hidden layer size of $\{n_1, n_2\}$ indicates two hidden layers, the lower hidden layer having n_1 hidden units and the upper hidden layer having n_2 units (followed by a softmax over the 10 MNIST classes). Methods with an asterisk (*) use some sort of episodic memory. Results marked with a double asterisk (**) were read from a graph in the source paper. Results with a dagger (†) are from Chaudhry *et al.* (2019).

Method	Hidden layer size	Number of tasks	Final average test accuracy
VCL	{100, 100}	10	93±1%
*VCL + 200 random coreset	{100, 100}	10	94.6±0.3%
Kronecker-factored Laplace	{100, 100}	50	90%
(Ritter *et al.*, 2018)	{100, 100}	10	96%**
EWC (Kirkpatrick *et al.*, 2017)	{2000, 2000}	10	97%
	{100, 100}	10	84%
SI (Zenke *et al.*, 2017)	{2000, 2000}	10	97%
	{100, 100}	10	86%
CLNP (Golkar *et al.*, 2019)	{2000, 2000}	10	98.42±0.04%
	{100, 100}	10	95.8%
*A-GEM (Chaudhry *et al.*, 2019)	{256, 256}	20	89.1±0.14%
	{256, 256}	10	92.3%**
BLLL-REG (Ebrahimi *et al.*, 2020)	{100, 100}	10	92.2%
*GEM	{256, 256}	20	89.5±0.48%†
(Lopez-Paz and Ranzato, 2017b)	{256, 256}	10	93.1%**†
	{100, 100}	20	80%
Riemannian Walk	{256, 256}	20	85.7±0.56%†
(Chaudhry *et al.*, 2018)	{256, 256}	10	91.6%**†
Progressive NNs	{256, 256}	20	93.5±0.07%†
(Rusu *et al.*, 2016)	{256, 256}	10	94.6%**†

4.3.7 *Discussion*

We now look into how VCL continually learns tasks, exploring how it uses its model capacity, primarily by looking at the learned weights in the model.

Split MNIST. We first consider the model trained without coresets (although exactly the same effects happen with coresets). When we look at

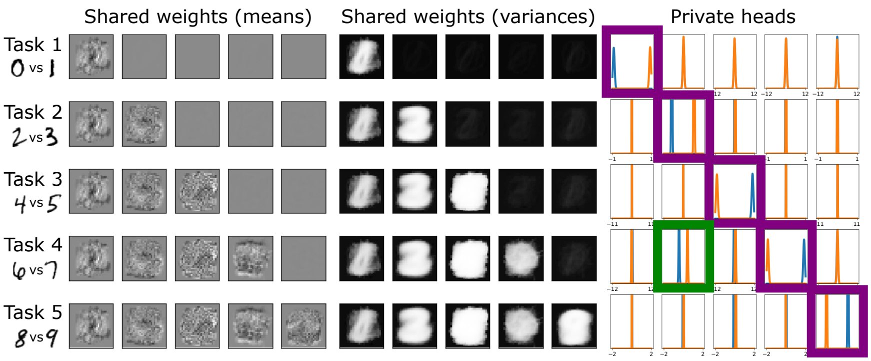

Fig. 4.2: The active units learned for Split MNIST without coresets. Rows correspond to stages of lifelong learning. Left: means, Centre: variances, Right: output weights for each task's two classes. Exactly the same effect is observed when incorporating coresets.

weight values into and out of each unit after training, we find that only one unit is being used in each of the five tasks. This is despite having 200 units in the single layer; the remaining units are pruned out as part of the optimisation process. The five active (un-pruned) units are plotted in Figure 4.2. This effect is similar to that observed in Trippe and Turner (2017), with entire units pruned out, as opposed to just individual weights. The pruned units appear to have input weights at or near the prior (standard normal Gaussian), with output weights near a delta function (zero mean, small variance), therefore minimising their effect on the output prediction. Removing all pruned units from the network does not change the network's accuracy.

This pruning effect seems to be due to the choice of the inference scheme. Intuitively, the pruning effect can be explained by looking at the optimisation function (4.10). By reducing the effect of a unit on the output prediction (e.g. setting output weights to have zero mean and small variance), the input weights to the unit can be set to their prior. The increase in the KL-to-prior term due to the small variance of the output weights is offset by the reduction in the KL-to-prior term from the numerically more input weights. Provided the expected log-likelihood term does not change too much, a pruned solution is therefore more optimal. We now focus on what our pruned solutions reveal about how our model approaches lifelong learning tasks, and debate whether the pruning effect is a feature or a bug for lifelong learning.

As the model only uses one hidden unit per task, it has learned to use a fraction of its total capacity to successfully learn binary classifiers in the Split MNIST experiment. The high accuracies indicate this is an efficient use of model capacity. The remaining unused units can be used for other tasks that we may see in the future. This implies that pruning is a beneficial feature for lifelong learning, forcing the model to efficiently use its capacity.

Additionally, the pruning effect allows us to see some forward and backward transfer (see Figure 4.2), which are both important qualities in a good lifelong learning solution. Forward transfer is visible when previous tasks' active units have non-zero weights for subsequent tasks. For example, unit 2, which was learned after task 2 (classifying digits {2v3}), has non-zero output weights after task 4 (classifying {6v7}). The model therefore uses some information about the task {2v3} in solving the task {6v7}. This is highlighted in green in the right of Figure 4.2. Although less visible in the plots, there is also backward transfer in the same units: unit 2's input weights change slightly after training on task 4 ({6v7}), potentially changing accuracy on task 2. In this case however, any backward transfer does not result in different accuracies; this could be because there is no potential for improvement given the high accuracies involved.

The exact same effects are seen when we train the model with coresets: the same pruning effect occurs (Figure 4.2 is similar when trained with coresets), and similar accuracies are obtained (within one standard deviation). This shows that for a simple task such as multi-head Split MNIST, there is no need for incorporating coresets.

This pruning effect is also similar to that considered in Golkar *et al.* (2019), where they prune out entire units. However, they have hyperparameters that control the degree of pruning (and corresponding accuracy loss). Previous work on variational inference for Bayesian neural networks has found that variational inference methods can be used to prune large parts of the network (Louizos *et al.*, 2017), and we show here how this pruning is done over units as opposed to weights (see also Trippe and Turner (2017)). In comparison to Golkar *et al.* (2019), our method automatically prunes out entire units and is able to re-use the units for both forward and backward transfer.

Permuted MNIST. We now look into how the two-hidden-layer model learns the Permuted MNIST task (results in Table 4.2). We first consider the model trained without coresets. We find that there is still pruning within the network. The numbers of active (un-pruned) units after training

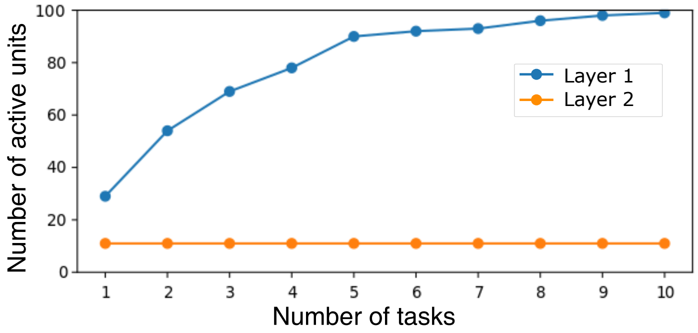

Fig. 4.3: Number of active units per hidden layer after each task in Permuted MNIST, without coresets. Exactly the same effect is observed when incorporating coresets.

on each task are summarised in Figure 4.3.

There are more active units in Permuted MNIST than were in Split MNIST, perhaps due to the more difficult nature of Permuted MNIST (classifying between 10 digits, as opposed to between 2). However, only 11 units are used in the second hidden layer, with the remaining 89 units pruned out. Additionally, the output weights on these 11 units do not change between tasks. This confirms that the hidden layers effectively depermute the images, allowing the output weights to just classify between the 10 digits.

Beyond re-using the upper level weights, there is not much evidence of forward or backward transfer. We should expect this from Permuted MNIST because the network trains on all MNIST digits on the first task itself, hence already learning the 'best' way to classify between MNIST digits. Any subsequent permuted images have little room to improve upon this. Instead, the remaining focus of Permuted MNIST seems to be on ensuring we use available model capacity as efficiently as possible. Increasing the model capacity improves results: training a network with 250 units in the lower hidden layer (instead of 100) improves final average test accuracy to 95.5% (10 tasks).

Incorporating coresets also improves results. However, the number of active units (plotted in Figure 4.3) looks the same. Instead, training VCL with coresets (which can be viewed as changing the order in which the model trains on data, or, changing the schedule with which we visit training data) appears to reinforce previous tasks' images: it lowers forgetting in the network.

Table 4.2 summarises some recent works' results on Permuted MNIST. As can be seen, different papers use different numbers of hidden units in their hidden layers, and test over different numbers of tasks. However, as the findings in this paper indicate, Permuted MNIST primarily tests for model capacity, highlighting how we cannot faithfully compare results when model capacity and number of tasks differ. We propose that when using Permuted MNIST as a benchmark in lifelong learning, model capacity is kept fairly limited (for example, two hidden layers with either 100 or 256 hidden units each), and number of tasks is kept high (minimum 10, possibly more). Of all methods, Ritter *et al.* (2018) and Golkar *et al.* (2019) achieve significantly superior results, as they use a relatively small network and test over many tasks. Note that 2000 units in two hidden layers is an extremely large model for 10 tasks: as Golkar *et al.* (2019) note, they can achieve high (as good as single-task) performance having pruned out a large proportion of their network. With such a large model capacity, this benchmark no longer tests any desiderata in lifelong learning aside from avoiding catastrophic forgetting, which many other benchmarks can also do.

Many papers often use Permuted MNIST to demonstrate how their method (and other baseline methods) exhibit some of lifelong learning's desiderata. For example, they show some form of resistance to forgetting or efficient use of model capacity via metrics or plots. In this case, comparing the final average test accuracy is no longer so important, but we believe far better comparisons would be achieved if a more challenging hidden layer size and number of tasks was used.

4.4 Improvements to VCL

There has been follow-up work that uses core ideas that we have discussed above and looks at generalising and improving them. We shall briefly discuss them in this section.

Natural-gradient variational inference. One strand of work optimises the variational objective function (4.5) with natural-gradients instead of using Bayes by Backprop to optimise for the variational parameters (Chen *et al.*, 2018; Khan *et al.*, 2018; Osawa *et al.*, 2019). Natural-gradient update steps are a principled way of incorporating the information geometry of the distribution being optimised (Amari, 1998). By incorporating the geometry of the distribution, we expect to take gradient steps in much better directions, speeding up gradient based optimisation. By using such natural-gradient variational inference, the objective (4.5) is therefore optimised quicker with the same performance on MNIST benchmarks (Osawa *et al.*, 2019; Eschenhagen, 2019). The faster optimisation has also allowed the VCL objective function to be scaled to larger benchmarks (Eschenhagen, 2019) and architectures.

Generalised VCL and FiLM layers. It is possible to temper the KL-to-prior term in the variational objective function (4.10). Loo *et al.* (2021) showed that this modification to VCL (called *Generalised VCL*) recovers Online EWC (Schwarz *et al.*, 2018) as a limiting case, allowing for interpolation between the two approaches. This leads to a theoretical generalisation and increased performance on several benchmarks.

Loo *et al.* (2021) also introduced task-specific FiLM layers (Perez *et al.*, 2018) to take advantage of and reduce pruning in variational Bayesian neural networks, finding that this also leads to improved performance. These FiLM layers linearly modulate features in a neural network, and making these additional parameters task-specific allows useful features for a task to be amplified and inappropriate ones ignored. This improves the pruning effect by allowing FiLM layers to prune units, instead of relying on the output weights of a unit to become zero-mean low-variance distributions. Loo *et al.* (2021) find that FiLM layers reduce pruning and improves performance of VCL (and Generalised VCL) on many benchmarks.

Function-space regularisation. Over a sequence of many tasks, independent regularisation of weights can still lead to forgetting. Instead, we are ultimately interested in neural network outputs or predictions, and we want to maintain these predictions over the course of many tasks. This has led to ideas to regularise the function space of neural networks directly, instead of only regularising in the weight space (Titsias *et al.*, 2020; Pan *et al.*, 2020). These methods store input-output pairs (the memorable examples), similar to coresets in VCL, but regularise them directly in the loss function. They are derived from the variational objective function, and use the same core idea of variational updates for continual learning. They show significantly better performance than VCL on larger scale problems, and are a promising direction of future research.

Improving coreset selection with Stein gradients. In the original version of VCL, coresets are chosen without knowledge of the approximate posterior using e.g. random sampling or K-center clustering. Chen *et al.* (2018) proposed an improvement to VCL that uses Stein gradients to construct the coresets. This method iteratively updates a coreset and moves it closer to samples from the posterior distribution, thus allowing the coreset to be constructed using the approximate posteriors without changing the inference procedure. Coresets constructed using Stein gradients are shown to outperform random and K-center coresets on the Permuted MNIST benchmark.

4.5 Conclusions and Future Directions

This chapter has introduced the Bayesian approach to lifelong learning for deep neural networks. This Bayesian framework provides a principled way to tackle lifelong learning and has a great potential for future research. One important question is how to scale Bayesian inference in general, and for lifelong learning in particular, to a large dataset such as ImageNet (Deng *et al.*, 2009) and for very deep neural networks. A good solution to this question would enable applications of Bayesian methods to several state-of-the-art deep models in computer vision or natural language processing that contain several million parameters. Another important direction for future explorations is to improve function-space regularisation approaches, potentially by using better posterior approximations and better memorable example selection mechanisms, and scale these approaches to larger datasets.

Bibliography

Abati, D., Tomczak, J., Blankevoort, T., Calderara, S., Cucchiara, R., and Bejnordi, B. E. (2020). Conditional channel gated networks for task-aware continual learning, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3931–3940.

Ahn, H., Cha, S., Lee, D., and Moon, T. (2019). Uncertainty-based continual learning with adaptive regularization, in *Advances in Neural Information Processing Systems*, Vol. 32.

Aljundi, R., Kelchtermans, K., and Tuytelaars, T. (2019). Task-free continual learning, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11254–11263.

Amari, S.-I. (1998). Natural gradient works efficiently in learning, *Neural Computation* **10**, 2, pp. 251–276.

Bang, J., Kim, H., Yoo, Y., Ha, J.-W., and Choi, J. (2021). Rainbow memory: Continual learning with a memory of diverse samples, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8218–8227.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural network, in *International Conference on Machine Learning*.

Borsos, Z., Mutny, M., and Krause, A. (2020). Coresets via bilevel optimization for continual learning and streaming, in *Advances in Neural Information Processing Systems*.

Broderick, T., Boyd, N., Wibisono, A., Wilson, A. C., and Jordan, M. I. (2013). Streaming variational Bayes, in *Advances in Neural Information Processing Systems*.

Chaudhry, A., Dokania, P. K., Ajanthan, T., and Torr, P. H. (2018). Riemannian walk for incremental learning: Understanding forgetting and intransigence, in *European Conference on Computer Vision*, pp. 532–547.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. (2019). Efficient lifelong learning with A-GEM, in *International Conference on Learning Representations*.

Chen, P.-H., Wei, W., Hsieh, C.-J., and Dai, B. (2021). Overcoming catastrophic forgetting by Bayesian generative regularization, in *International Confer-*

ence on Machine Learning.

Chen, Y., Diethe, T., and Lawrence, N. (2018). Facilitating Bayesian continual learning by natural gradients and Stein gradients, in *Continual Learning Workshop @ NeurIPS.*

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database, in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255.

Ebrahimi, S., Elhoseiny, M., Darrell, T., and Rohrbach, M. (2020). Uncertainty-guided lifelong learning in Bayesian networks, in *International Conference on Learning Representations.*

Eschenhagen, R. (2019). Natural gradient variational inference for continual learning in deep neural networks, Tech. rep., University of Osnabruck.

Farquhar, S. and Gal, Y. (2018). A unifying Bayesian view of continual learning, in *Bayesian Deep Learning Workshop @ NeurIPS.*

French, R. M. (1999). Catastrophic forgetting in connectionist networks, *Trends in cognitive sciences* **3**, 4, pp. 128–135.

Ghahramani, Z. and Attias, H. (2000). Online variational Bayesian learning, in *NIPS Workshop on Online Learning.*

Golkar, S., Kagan, M., and Cho, K. (2019). Continual learning via neural pruning, in *Neuro-AI Workshop @ NeurIPS.*

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. (2014). An empirical investigation of catastrophic forgetting in gradient-based neural networks, in *International Conference on Learning Representations.*

He, X., Sygnowski, J., Galashov, A., Rusu, A. A., Teh, Y. W., and Pascanu, R. (2020). Task agnostic continual learning via meta learning, in *Lifelong Machine Learning Workshop @ ICML.*

Jerfel, G., Grant, E., Griffiths, T., and Heller, K. A. (2019). Reconciling meta-learning and continual learning with online mixtures of tasks, in *Advances in Neural Information Processing Systems*, Vol. 32.

Jin, X., Sadhu, A., Du, J., and Ren, X. (2021). Gradient-based editing of memory examples for online task-free continual learning, in *Advances in Neural Information Processing Systems*, Vol. 34.

Kessler, S., Nguyen, V., Zohren, S., and Roberts, S. J. (2021). Hierarchical Indian buffet neural networks for Bayesian continual learning, in *Uncertainty in Artificial Intelligence*, pp. 749–759.

Khan, M., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., and Srivastava, A. (2018). Fast and scalable Bayesian deep learning by weight-perturbation in Adam, in *International Conference on Machine Learning.*

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization, in *International Conference on Learning Representations.*

Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick, in *Advances in Neural Information Processing Systems.*

Kingma, D. P. and Welling, M. (2014). Stochastic gradient VB and the variational auto-encoder, in *International Conference on Learning Representations.*

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu,

A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., *et al.* (2017). Overcoming catastrophic forgetting in neural networks, *Proceedings of the National Academy of Sciences* **114**, 13, pp. 3521–3526.

Kumar, A., Chatterjee, S., and Rai, P. (2021). Bayesian structural adaptation for continual learning, in *International Conference on Machine Learning*.

Kurle, R., Cseke, B., Klushyn, A., van der Smagt, P., and Günnemann, S. (2020). Continual learning with Bayesian neural networks for non-stationary data, in *International Conference on Learning Representations*.

Liu, J. S. and Chen, R. (1998). Sequential Monte Carlo methods for dynamic systems, *Journal of the American Statistical Association* **93**, 443, pp. 1032–1044.

Loo, N., Swaroop, S., and Turner, R. E. (2021). Generalized variational continual learning, in *International Conference on Learning Representations*.

Lopez-Paz, D. and Ranzato, M. (2017). Gradient episodic memory for continual learning, in *Advances in Neural Information Processing Systems*.

Louizos, C., Ullrich, K., and Welling, M. (2017). Bayesian compression for deep learning, in *Advances in Neural Information Processing Systems*.

Maybeck, P. S. (1982). *Stochastic models, estimation, and control* (Academic Press).

Nguyen, C. V., Achille, A., Lam, M., Hassner, T., Mahadevan, V., and Soatto, S. (2019). Toward understanding catastrophic forgetting in continual learning, *arXiv preprint arXiv:1908.01091* .

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. (2018). Variational continual learning, in *International Conference on Learning Representations*.

Osawa, K., Swaroop, S., Jain, A., Eschenhagen, R., Turner, R. E., Yokota, R., and Khan, M. E. (2019). Practical deep learning with Bayesian principles, in *Advances in Neural Information Processing Systems*.

Pan, P., Swaroop, S., Immer, A., Eschenhagen, R., Turner, R., and Khan, M. E. E. (2020). Continual deep learning by functional regularisation of memorable past, in *Advances in Neural Information Processing Systems*.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. (2018). FiLM: Visual reasoning with a general conditioning layer, in *AAAI Conference on Artificial Intelligence*.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models, in *International Conference on Machine Learning*.

Ritter, H., Botev, A., and Barber, D. (2018). Online structured Laplace approximations for overcoming catastrophic forgetting, in *Advances in Neural Information Processing Systems*.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks, *arXiv:1606.04671* .

Sato, M.-A. (2001). Online model selection based on the variational Bayes, *Neural Computation* **13**, 7, pp. 1649–1681.

Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. (2018). Progress & compress: A scalable framework for continual learning, in *International Conference on Machine Learning*.

Smola, A. J., Vishwanathan, S., and Eskin, E. (2004). Laplace propagation, in *Advances in Neural Information Processing Systems*.

Swaroop, S., Nguyen, C. V., Bui, T. D., and Turner, R. E. (2018). Improving and understanding variational continual learning, in *Continual Learning Workshop @ NeurIPS*.

Titsias, M. K., Schwarz, J., de G. Matthews, A. G., Pascanu, R., and Teh, Y. W. (2020). Functional regularisation for continual learning with Gaussian processes, in *International Conference on Machine Learning*.

Trippe, B. and Turner, R. (2017). Overpruning in variational Bayesian neural networks, in *Advances in Approximate Bayesian Inference Workshop @ NIPS*.

Tseran, H., Khan, M. E., Harada, T., and Bui, T. D. (2018). Natural variational continual learning, in *Continual Learning Workshop @ NeurIPS*.

Wang, L., Zhang, M., Jia, Z., Li, Q., Bao, C., Ma, K., Zhu, J., and Zhong, Y. (2021). AFEC: Active forgetting of negative transfer in continual learning, in *Advances in Neural Information Processing Systems*.

Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence, in *International Conference on Machine Learning*.

Zeno, C., Golan, I., Hoffer, E., and Soudry, D. (2021). Task-agnostic continual learning using online variational Bayes with fixed-point updates, *Neural Computation* **33**, 11, pp. 3139–3177.