A Unifying Bayesian View of Continual Learning

Sebastian Farquhar
OATML Research Group
Department of Computer Science
University of Oxford
sebastian.farquhar@cs.ox.ac.uk

Yarin Gal
OATML Research Group
Department of Computer Science
University of Oxford
yarin@cs.ox.ac.uk

1 Introduction

Some machine learning applications require continual learning—where data comes in a sequence of datasets, each is used for training and then permanently discarded. Continual learning might be needed because retaining personal data is illegal or undesireable, for example a hospital may not want to keep patient data indefinitely. Alternatively, a real-time system might have so much data that complete retraining is impractical, for example, a quadcopter reacting to changing conditions. From a Bayesian perspective, continual learning seems straightforward: Given the model posterior one would simply use this as the prior for the next task [Nguyen et al., 2018].

However, exact posterior evaluation is intractable with many models, especially with Bayesian neural networks (BNNs) [MacKay, 1992, Neal, 1995]. Instead, posterior *approximations* are often sought. In this *approximate Bayesian* continual learning set up, when training on the first dataset we use a simple distribution over neural network weights to approximates the posterior given that data. This approximate posterior is then used as a *prior* when evaluating the posterior approximation with the second dataset [Nguyen et al., 2018]. Many continual learning solutions that do not rely on BNNs are justified similarly [Kirkpatrick et al., 2017, Zenke et al., 2017, Chaudhry et al., 2018, Ritter et al., 2018]. We call these approaches *prior-focused*. Unfortunately, when posterior *approximations* are used, this prior-focused approach does not succeed in evaluations designed to capture properties of realistic continual learning use cases [Farquhar and Gal, 2018].

As an alternative to prior-focused methods, we introduce a new approximate Bayesian derivation of the continual learning loss. Our loss does not rely on the posterior from earlier tasks, and instead adapts the model itself by changing the likelihood term. We call these approaches *likelihood-focused*. We then combine prior- and likelihood-focused methods into one objective, tying the two views together under a single unifying framework of approximate Bayesian continual learning. This unifying view encompasses hybrid approaches which have previously not been explicitly studied.

Finally, we empirically evaluate the importance of the prior in approximate Bayesian continual learning by comparing prior-focused, likelihood-focused, and hybrid approaches. To support our analysis, we introduce Variational Generative Replay (VGR): a variational inference generalization of Deep Generative Replay (DGR) [Shin et al., 2017] which falls under our likelihood-focused approaches. This model can be seen as a likelihood-focused counterpart to the current state-of-the-art prior-focused approach. We show that the likelihood-focused component within a state-of-the-art hybrid approach is responsible for all of its performance on a key benchmark. We also show that the prior-focused approach does not express well-calibrated uncertainty about whether data has been seen before, while a likelihood-focused method does. This suggests that the miscalibrated uncertainty of each successive posterior may explain why successful prior-focused approaches, despite their attractive theoretical properties, fail in some continual learning settings.

Third workshop on Bayesian Deep Learning (NeurIPS 2018), Montréal, Canada.

2 Prior-focused Continual Learning

We consider variational inference with BNNs [Jordan et al., 1999, Hinton and van Camp, 1993]. In non-continual learning, one aims to approximate a parameter posterior $p(\boldsymbol{\omega}|\mathcal{D})$ given an independently and identically distributed (i.i.d.) labelled training dataset $\mathcal{D} \equiv \{(\mathbf{x}^{(i)}, y^{(i)})\}$. In the continual learning setting, members of $\mathcal{D}$ are not i.i.d. Instead we have T disjoint subsets $\mathcal{D}_t \equiv \{(\mathbf{x}_t^{(i)}, y_t^{(i)})\}$ each of which is individually i.i.d. and represent a task.

Variational Continual Learning (VCL) explicitly performs variational inference treating the approximate posterior distribution $q(\boldsymbol{\omega}_{t-1}|\mathcal{D}_{1:t-1})$ at the end of the $t-1$'th task as the prior when learning the t'th task. This entails the variational evidence lower bound loss function for the t'th task:

$$\mathcal{L}^t_{VCL}(q_t(\boldsymbol{\omega})) = \sum_{n=1}^{N_t} \mathbb{E}_{\boldsymbol{\omega}\sim q_t(\boldsymbol{\omega})} \Big[\log p(y_t^{(n)}|\boldsymbol{\omega}, \mathbf{x}_t^{(n)})\Big] - KL\big(q_t(\boldsymbol{\omega}) \parallel q_{t-1}(\boldsymbol{\omega})\big). \tag{1}$$

The first term in equation (1) is the expected log-likelihood of the current model over the data in the current dataset. The second term penalizes difference between the current model and the approximate posterior of the previous task. Although they are not formulated using BNNs, Elastic Weight Consolidation (EWC) [Kirkpatrick et al., 2017], Synaptic Intelligence (SI) [Zenke et al., 2017], Riemanian Walk [Chaudhry et al., 2018], and the extension of EWC offered by Ritter et al. [2018] all rely on a related Bayesian justification and can be expressed with similar functional form. We call these *prior-focused* methods because they treat old posteriors as the prior for a new model. These broadly correspond to what Parisi et al. [2018] call regularization approaches.

3 Likelihood-focused Continual Learning

Prior-focused approaches assume that $q_{t-1}(\boldsymbol{\omega})$ approximates the posterior well enough to be used for continual learning. However, we do not need to make this assumption. For example, Deep Generative Replay (DGR) [Shin et al., 2017] uses successive datasets to train a Generative Adversarial Network (GAN) to simulate old datasets. These approaches build on rehearsal by Ratcliff [1990] and the extension by Robins [1995] using pseudo-rehearsal—randomly generated synthetic data.

All of these can be seen as attempts to estimate the log-likelihood of old data according to the current model, as we show below. As a result, we can call these *likelihood-based* approaches. In order to capture this behaviour in a Bayesian setting, we consider the standard ELBO loss, expanded to account for the multiple datasets:

$$\mathcal{L}^t_{ELBO}(q_t(\boldsymbol{\omega})) = \sum_{t'=1}^{t} \left[\sum_{n=1}^{N_t} \mathbb{E}_{\boldsymbol{\omega}\sim q_{t'}(\boldsymbol{\omega})} \Big[\log p(y_{t'}^{(n)}|\boldsymbol{\omega}, \mathbf{x}_{t'}^{(n)})\Big]\right] - KL\big(q_t(\boldsymbol{\omega}) \parallel p(\boldsymbol{\omega})\big) \tag{2}$$

in which the KL-divergence is always computed to the original prior and the log-likelihood is summed over all previously seen tasks. In continual learning, however, we do not have $\mathcal{D}_{t'}$ for $t' < t$ so we cannot calculate the log-likelihood directly. Instead, we estimate the expectation of the log-likelihood for each $t' < t$ separately as:

$$\sum_{n=1}^{N_t} \mathbb{E}_{\boldsymbol{\omega}\sim q_{t'}(\boldsymbol{\omega})} \Big[\log p(y_{t'}^{(n)}|\boldsymbol{\omega}, \mathbf{x}_{t'}^{(n)})\Big]\Big] \approx \int \log\big[p(y|\boldsymbol{\omega}, \mathbf{x})\big] p_{t'}(\mathbf{x}, y) q(\boldsymbol{\omega}) d\mathbf{x} dy d\boldsymbol{\omega}. \tag{3}$$

This relies on a separately learned generative model $p_{t'}(\mathbf{x}, y)$. We sample $\hat{\mathbf{x}}, \hat{y} \sim p_{t'}(\mathbf{x}, y)$ before training each task. We then create $\tilde{\mathcal{D}}_t \equiv (\tilde{\mathbf{x}}, \tilde{y}) = (\hat{\mathbf{x}} \cup \mathbf{x}, \hat{y} \cup y)$ and train on $\tilde{\mathcal{D}}_t$. Samples from a standard ELBO loss on minibatches drawn from this dataset approximate the true loss.

The joint probability distribution $p(\mathbf{x}, y)$ can be estimated in a number of ways. Early dual-memory methods are very simple approximations. Storing a small coreset or sample of the old data, as Ratcliff [1990], offers a Monte Carlo approximation of the distribution. Pseudorehearsal effectively approximates $p(\mathbf{x}, y)$ by learning $p(y|\mathbf{x})$ and approximating $p(\mathbf{x}) \sim Bern(0.5)$.

Table 1: Comparison of methods evaluated in this abstract

	Prior	**Likelihood given...**
VGR	$p(\boldsymbol{\omega})$	Generative models of old tasks
VCL	$q_{t-1}(\boldsymbol{\omega})$	Current task
VCL w/ coresets	$q_{t-1}(\boldsymbol{\omega})$	Current task & Coresets
Coreset only	$p(\boldsymbol{\omega})$	Current task & Coresets

3.1 Variational Generative Replay

To study the contributions of the different components of Bayesian continual learning we suggest a new continual learning approach complementary to VCL. At the end of training the t'th task, we train a GAN using the task data to estimate $p_t(\mathbf{x}, y)$ and store this GAN. The collection of stored GANs are sampled to create $\tilde{\mathcal{D}}_{t+1}$. We call this Variational Generative Replay (VGR). DGR can be seen as a maximum likelihood estimation approximation of our approach, with some other implementation differences.

4 A Unifying View of Continual Learning

We can combine prior- and likelihood-focused approaches, drawing on the strengths of each. We use the approximation of (3) alongside a fresh prior for each task to form a hybrid loss function:

$$\mathcal{L}^t_{Hybrid}(q_t(\boldsymbol{\omega})) = \sum_{t'=1}^{t} \left[\int \log\big[p(y|\boldsymbol{\omega}, \mathbf{x})\big] p_{t'}(\mathbf{x}, y) q(\boldsymbol{\omega}) d\mathbf{x} dy d\boldsymbol{\omega} \right] - KL\big(q_t(\boldsymbol{\omega}) \parallel q_{t-1}(\boldsymbol{\omega})\big). \quad (4)$$

For a hybrid approach, part of each dataset is used to estimate a posterior for future tasks, while some is reserved to estimate a generative model of old data. Although they do not explicitly interpret it as such, two hybrid approaches have already been presented. Nguyen et al. [2018] extend VCL with coresets—small held-out samples from earlier datasets—which are used to fine-tune the model at the end of each task. This approximates a special case of (4), though the generative and current components of the likelihood are trained separately, rather than using a single loss function as we suggest. Chaudhry et al. [2018] also implicitly approximate (4) when they supplement training with a sample of data from old tasks.

5 Experimental Analysis

We empirically assess the importance of the prior in prior-focused and hybrid continual learning by isolating the different components of VCL and comparing their performance with VGR.[1] We compare the four variant models shown in Table 1, using implementations by Nguyen et al. [2018] for VCL. Experimental details can be found in Appendix A.

VCL (with and without coresets) and VGR perform well on simple tasks like Permuted MNIST (as described in Goodfellow et al. [2013], see fig 1) and multi-headed Split MNIST (as described by Zenke et al. [2017], see fig 2). We present these as standard baseline experiments, despite the significant shortcomings shown in Farquhar and Gal [2018]. These show that likelihood- and prior-focused approaches as well as hybrid ones have good performance on simple experiments.

However, when Split MNIST is performed without multi-heading (see Chaudhry et al. [2018], Farquhar and Gal [2018]), we see that VGR significantly outperforms VCL (figure 3). Moreover, we show by ablation that coresets (the likelihood-focused component) contribute all of the performance of the hybrid approach, since the coresets only model performs exactly as well as VCL with coresets. This indicates that the posterior approximations used for continual learning are insufficient.

When we specifically examine the posteriors used by VCL, we see that the uncertainty of the prior model does not distinguish seen from unseen datasets after the first task (figure 4). We assess

[1]We do not compare VCL with DGR directly, as the use of mean-field variational inference in the former but not the latter makes it hard to isolate the difference between likelihood- and prior-focused continual learning.

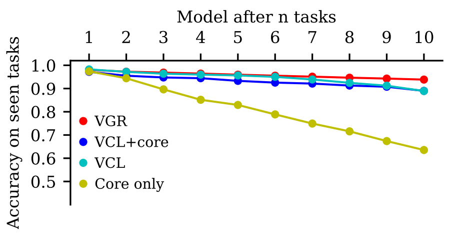

Figure 1: Permuted MNIST. All main methods perform well, although coresets on their own do not.

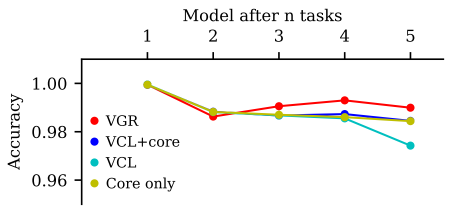

Figure 2: Multi-headed Split MNIST. All methods perform well.

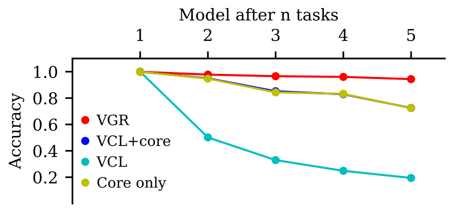

Figure 3: VGR performs well on single-headed Split MNIST. VCL performance is determined by coreset.

uncertainty using the mutual information between the model parameters and data labels given the test dataset for each of the five datasets for each of five models (shown scaled by the largest mutual information for each model). However, the uncertainty measured for the posteriors from VGR (which includes training on generated data) performs much better (see fig 5). The posteriors from VGR clearly distinguish seen from unseen data.

6 Discussion

Although prior-focused methods have a Bayesian justification, they assume that model posteriors are good enough to be used as priors for learning. We show that this assumption can be too strong, evidenced by poor performance on single-headed Split MNIST and the fact that VCL models do not have well calibrated uncertainty about unseen data. Instead, we show that existing dual-memory approaches can be interpreted from a Bayesian perspective. We show that simple likelihood-focused methods can outperform prior-focused methods and can offer better uncertainty estimates.

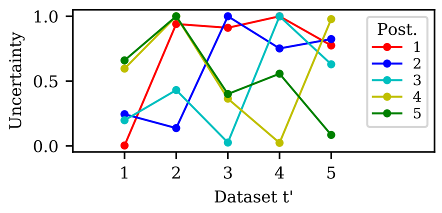

Figure 4: Each posterior's scaled uncertainty is shown for VCL. It should be low for only datasets with $t' \leq t$ but is not.

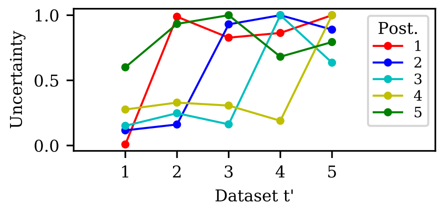

Figure 5: For VGR uncertainty (rescaled mutual information) is low on seen tasks and high on unseen ones as it should be.

References

Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence. *Proceedings of the European Conference on Computer Vision*, 11215:556–572, 2018. 1, 2, 4, 5

Sebastian Farquhar and Yarin Gal. Towards Robust Evaluations of Continual Learning. In *Workshop on Lifelong Learning: A Reinforcement Learning Approach at ICML*, Lifelong Learning: A Reinforcement Learning Approach Workshop at ICML 2018, 2018. 1, 5

Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. *arXiv*, 2013. 5

Alex Graves. Practical Variational Inference for Neural Networks. *Neural Information Processing Systems*, 2011. A

Geoffrey E. Hinton and Drew van Camp. Keeping the neural networks simple by minimizing the description length of the weights. *Proceedings of the sixth annual conference on Computational learning theory - COLT '93*, pages 5–13, 1993. 2

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. Introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999. 2

Diederik P. Kingma, Tim Salimans, and Max Welling. Variational Dropout and the Local Reparameterization Trick. *Advances In Neural Information Processing Systems*, pages 2575–2583, 2015. A

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. *Neural Information Processing Systems*, 114(13):3521–3526, 2017. 1, 2

David J. C. MacKay. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3):448–472, 1992. 1

Radford M. Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995. 1

Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational Continual Learning. *International Conference on Learning Representations*, 2018. 1, 4, 5, A

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual Lifelong Learning with Neural Networks: A Review. *arXiv*, 2018. 2

Roger Ratcliff. Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological Review*, 97(2):285–308, 1990. 3, 3

Hippolyt Ritter, Aleksandar Botev, and David Barber. Online Structured Laplace Approximations For Overcoming Catastrophic Forgetting. *arXiv*, 2018. 1, 2

Anthony Robins. Catastrophic Forgetting, Rehearsal and Pseudorehearsal. *Connection Science*, 7(2): 123–146, 1995. ISSN 13600494. 3

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual Learning with Deep Generative Replay. *Advances In Neural Information Processing Systems*, pages 2994–3003, 2017. 1, 3

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual Learning Through Synaptic Intelligence. *Proceedings of the 34th International Conference on Machine Learning*, pages 3987–3995, 2017. 1, 2, 5

A Experimental Details

For permuted MNIST, we follow Nguyen et al. [2018] where possible using an implementation made public by those authors. We use a Bayesian neural network [Graves, 2011] with two hidden layers of 100 units with ReLu activations. The priors are initialized as a unit Gaussian and the parameters are initialized with the mean of a pre-trained maximum likelihood model and a small initial variance (10^{-6}). We use the Adam optimizer [Kingma et al., 2015] with learning rate 10^{-3}. For both VGR and VCL we use a single head, again following Nguyen et al. [2018]. For all results we present the average over 10 runs, with a different permutation each time. Standard errors are not shown as they are under a tenth of a percent.

For VCL we train for 100 epochs using a batch size of 256 on all the data except the with-held coresets. We train the whole single head on coresets for 100 epochs and use 200 digits of each permutation as a coreset chosen using the same k-center coresets algorithm used by Nguyen et al. [2018]. VCL without coresets is exactly the same, but without a final training step on coresets. The coresets only algorithm is exactly the same as VCL, except that the prior is always initialized as though it were the first task.

We train VGR for 120 epochs using a GAN trained for 200 epochs on each MNIST digit, using the same data which is used for the main training process. We use 6000 generated digits per class, sampled fresh for each task, and initialize network weights using the previous task. We use a batch size of 256 times the number of seen tasks, ensuring that the number of batches is held constant.

The GAN is trained with an Adam optimizer with learning rate $2 * 10^{-4}$ and β_1 of 0.5. The network has four fully-connected hidden layers with 256, 512, 1024 and 784 weights respectively. It uses Leaky ReLu with α of 0.2.

The settings for Split MNIST follow Nguyen et al. [2018] where possible. We use exactly the same architecture as for Permuted MNIST, including the single head, except that each hidden layer has 256 weights, similarly to Nguyen et al. [2018] on their multi-headed Split MNIST. Results are shown averaged over 10 runs, with a different coreset selection each time. Standard errors are not shown as they are of the order of a tenth of a percent.

For all architectures we train for 120 epochs. For VCL we use batch sizes equal to the training set size. We use coresets of 40 digits per task selected using the same k-center coreset algorithm as Nguyen et al. [2018], which are withheld from the training. We train for 120 epochs on the concatenated coreset across all the heads together.

For VGR, we use 6000 digits per class generated by a convolutional GAN. Unlike VCL, we cap batch sizes at 30,000 rather than having the batch size equal the training set size. This is because the examples generated by VGR result in larger training sets which exceeded the memory available on our GPU.The GAN is trained for 50 epochs on each MNIST class using the same optimizer as the non-convolutional GAN used for Permuted MNIST. It has a fully connected layer followed by two convolutional layers with 64 and 1 channel(s) and 5x5 convolutions. Each convolutional layer is preceded by a 2x2 up-sampling layer. The activations are Leaky ReLu's with α of 0.2.

Multi-headed Split MNIST has almost precisely the same settings as the Single-headed Split MNIST above. Following Nguyen et al. [2018], for VCL we have five heads in the final layer, each of which is trained separately. Coresets are used to train each head in turn. Batch size equal is to training set size. Results are shown averaged over 10 runs with fresh coreset selection each time. Standard errors are not shown as they are of the order of a tenth of a percent.

For VGR, unlike VCL, we only perform multi-heading as a way of selecting predictions, rather than using multiple heads during training as well, because each batch contains data from multiple tasks.