Can Sequential Bayesian Inference Solve Continual Learning?

Samuel Kessler SKESSLER@ROBOTS.OX.AC.UK
University of Oxford

Adam Cobb ADAM.COBB@SRI.COM
SRI International

Stefan Zohren ZOHREN@ROBOTS.OX.AC.UK
University of Oxford

Stephen J. Roberts SJROB@ROBOTS.OX.AC.UK
University of Oxford

Abstract

Previous work in Continual Learning (CL) has used sequential Bayesian inference to prevent forgetting and accumulate knowledge from previous tasks. A limiting factor to performing Bayesian CL has been exact inference in a Bayesian Neural Network (NN). We perform sequential Bayesian inference with a Bayesian NN using Hamiltonian Monte Carlo (HMC) and propagate the posterior as a prior for a new task by fitting a density estimator on HMC samples. We find that this approach fails to prevent forgetting. We propose an alternative view of the CL problem which directly models the data generating process and decomposes the CL problem in task specific and shared parameters. This method named Prototypical Bayesian CL and performs well compared to the latest Bayesian CL methods.

1. Introduction

In Continual Learning (CL) a Neural Network (NN) predictor is required to solve one task then another sequentially while maintaining performance on previously mastered tasks. One key challenge of CL is that NN weights from previously mastered tasks get overwritten when learning a new task and as such causes *catastrophic forgetting* of previous tasks French (1999). One way to prevent forgetting hinges on using recursive applications of Bayes' rule; using a Bayesian NN's posterior as a prior for a new task Huszár (2017); Kirkpatrick et al. (2017). Obtaining the posterior over NN weights is intractable and we need to resort to approximations of the posterior such as using a Laplace approximation MacKay (1992) or variational inference Graves (2011); Blundell et al. (2015) to obtain a posterior to subsequently use as a prior for learning a new task.

When performing Bayesian CL and using an approximate inference method such as a Laplace approximation Schwarz et al. (2018); Ritter et al. (2018) or variational approximation Nguyen et al. (2018); Ebrahimi et al. (2019); Kessler et al. (2019); Loo et al. (2020) most implementations will use a *multi-head* NN architecture. By multi-head we mean that there is a shared feature extractor which takes in some input $x \in \mathcal{X}$ for all tasks $g : x \to z$ which maps to some embedding space $z \in \mathcal{Z}$ and then there is a head per task $h_i : z \to \hat{y}$ where $\hat{y}$ is a prediction and $i \in \{1, \ldots, T\}$, each head could be a logistic regression in the case of binary classification. So each head is looked up when predicting each past task and each head preserves task specific information make Bayesian CL easier to perform. Indeed past works have noted this implementation "trick" and suggested using a *single head*, h as a more challenging benchmark Farquhar and Gal (2018); Van de Ven and Tolias (2019); Hsu et al. (2018). When we compare performance using variational continual learning (VCL) Nguyen et al. (2018) and see that for Split MNIST that there is a drop in performance for all tasks after having learned on a particular task: this is forgetting. What we also notice is that there is a difference in performance when using a multi-head approach versus a single-head approach, Figure 1. This throws up a set of interesting questions. If we had access to the true posterior of a single-head BNN would this be enough to prevent forgetting and would the gap in performance between single-head and multi-head BNNs be closed? Do we need to have some task specific parameters to prevent forgetting and do we need to deviate from the Bayesian CL framework by doing so?

Our contributions in this paper are to consider the above questions. We propagate the true posterior of a BNN and use this as a prior for a new task. We do this by using Hamiltonian Monte Carlo (HMC) and use a density estimation on the samples from HMC and using this approximate density or posterior as a prior for a new task within the HMC sampling process. This yields no noticeable benefits over VCL. We argue that the current formulation of Bayesian CL cannot prevent Bayesian inference from weighting the likelihood over the prior and so yielding forgetting, similarly to the Kalman filter (see Section 9). We then propose a new probabilistic model for CL and show that by explicitly modeling the generative process of the data we can achieve good performance rather then relying solely on propagating the posterior from a previous task to prevent forgetting.

2. The Continual Learning Setup

Continual learning (CL) is a setting whereby a model must learn a set of tasks sequentially, while maintaining performance across all tasks. In CL, the model is shown a set of T tasks sequentially $\mathcal{T}_t$ for $t = 1, \dots T$, (T can equal ∞). Each task is comprised of a dataset such that $\mathcal{T}_t : \mathcal{D}_t = \{(x_i, y_i)\}$ for $i = 1, \dots, N_t$. Although the model will lose access to the training dataset for task $\mathcal{T}_t$, it will be continually evaluated on all previous tasks $\mathcal{T}_i$ for $i \leq t$. The id t can be used as a task identifier informing the agent when to start training on a new task or what task to test. For a comprehensive review of CL scenarios see Hsu et al. (2018); Van de Ven and Tolias (2019).

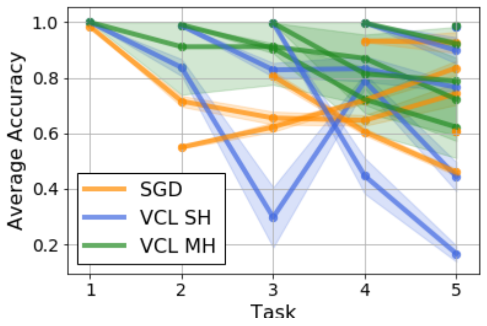
3. The Current Formulation of Bayesian Continual Learning

We consider the situation in which data arrives sequentially at times, $t = 1, 2, \dots, T$. For the first time step, $t = 1$, the model receives the dataset $\mathcal{D}_1$ and the aim is to learn a model (a neural network) with parameters θ with prior $p(\boldsymbol{\theta})$ of the conditional distribution $p(y_i|x_i, \boldsymbol{\theta})$ for all $(x_i, y_i) \in \mathcal{D}_1$. Then the predictive distribution for a test point, x_1^* is:

$$p(y_1^*|x_1^*, \mathcal{D}_1) = \int p(y_1^*|x_1^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}_1) d\boldsymbol{\theta} \tag{1}$$

We note that the above requires knowledge of the posterior over parameters $p(\boldsymbol{\theta}|\mathcal{D}_1)$. For $t = 2$, the model is required to fit $p(y_i|x_i, \boldsymbol{\theta})$ for $\mathcal{D}_1 \cup \mathcal{D}_2$. The predictive distribution for a

Figure 1: Accuracy on Split MNIST for simple two layer networks. The the first task, the images for $\{0, 1\}$ are mapped to the classes $\{0, 1\}$. In the second task the images for $\{2, 3\}$ are mapped $\{0, 1\}$. We compare a naive NN trained with SGD (single-headed) to VCL. VCL is evaluated on the single-headed variant (SH) and the multi-headed variant (MH).



new test point x_2^* point is:

$$p(y_2^*|x_2^*, \mathcal{D}_1, \mathcal{D}_2) = \int p(y_2^*|x_2^*, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D}_1, \mathcal{D}_2)d\boldsymbol{\theta}. \tag{2}$$

Thus we need to update our posterior to be able to learn this new conditional distribution. We can use repeated applications of Bayes' rule to calculate the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D}_1, \dots, \mathcal{D}_T)$, namely:

$$p(\boldsymbol{\theta}|\mathcal{D}_1, \dots, \mathcal{D}_{T-1}, \mathcal{D}_T) = \frac{p(\mathcal{D}_T|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D}_1, \dots, \mathcal{D}_{T-1})}{p(\mathcal{D}_T|\mathcal{D}_1, \dots, \mathcal{D}_{T-1})}. \tag{3}$$

To aid tractability one usually resorts to approximate inference e.g. a Laplace approximation or variational inference.

In CL we loose access previous training datasets, but using repeated applications of Bayes' rule, Equation (3) this limitation can be circumvented. Consider at $t = 2$ were we require a posterior $p(\boldsymbol{\theta}|\mathcal{D}_1, \mathcal{D}_2)$ to calculate the predictive distribution in Equation (2), however we have lost access to $\mathcal{D}_1$, with Bayes' rule the posterior takes the form:

$$\begin{align} \log p(\boldsymbol{\theta}|\mathcal{D}_1, \mathcal{D}_2) &= \log p(\mathcal{D}_2|\boldsymbol{\theta}, \mathcal{D}_1) + \log p(\boldsymbol{\theta}|\mathcal{D}_1) - \log p(\mathcal{D}_2|\mathcal{D}_1) \tag{4} \\ &= \log p(\mathcal{D}_2|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}|\mathcal{D}_1) - \log p(\mathcal{D}_2|\mathcal{D}_1) \tag{5} \end{align}$$

Hence we can use the posterior at t as a prior for learning a new task at $t+1$. *Observation: we require that our model with parameters $\boldsymbol{\theta}$ is a sufficient statistic of $\mathcal{D}_1$ hence the motivation for using complex BNN predictors;* a Bayesian linear regression model will be unable to model images datasets. The last term is included since without conditioning on $\boldsymbol{\theta}$, the data sets $\mathcal{D}_1$ and $\mathcal{D}_2$ are no longer independent Huszár (2017).

3.1. Variational Continual Learning

CL can be decomposed into approximate sequential Bayesian updates, Variational CL (VCL) Nguyen et al. (2018) uses a variational BNN to perform the prediction tasks where the network weights are independent Gaussians. The variational posterior from previous tasks is used as a prior for new tasks. Consider learning the first task $\mathcal{T}_1$, and $\boldsymbol{\theta}$ are the variational BNN weights then the variational posterior is $q_1(\boldsymbol{\theta}|\mathcal{D}_1)$. For the subsequent task

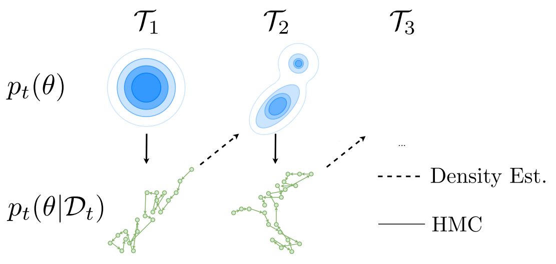


Figure 2: The propagation of posterior process, priors in blue are in the top row and posteriors are sampled in the bottom row. This is a two step process where we first perform HMC with a isotropic Gaussian prior for $\mathcal{T}_1$ then secondly we perform density estimation on samples from the posterior to obtain $\hat{p}_1(\theta|\mathcal{D}_1)$ this posterior can then be used as a prior for the new task $\mathcal{T}_2$ and so on.

the prior will be $q_1(\boldsymbol{\theta}|\mathcal{D}_1)$, optimization of the ELBO will yield the variational posterior $q_2(\boldsymbol{\theta}|\mathcal{D}_2)$. Generalizing, the negative ELBO for the t-th task is:

$$\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}_t) = \mathrm{KL}\left[q_t(\boldsymbol{\theta})||q_{t-1}(\boldsymbol{\theta}|\mathcal{D}_{t-1})\right] - \mathbb{E}_{q_t}[\log p(\mathcal{D}_t|\boldsymbol{\theta})]. \tag{6}$$

When applying VCL to the problem of Split MNIST, Figure 1 we can see that single-head (SH) VCL performs worse than multi-head VCL, despite following more closely the process of Bayesian CL described in Section 3. *Observation: using task specific parameters helps in preserving knowledge from previous tasks.* One confounder could be that since we are propagating a variational approximation of the posterior then we are actually not performing the Bayesian CL process described earlier. Next we will try to propagate the true posterior of our BNN.

3.2. Propagating the True Posterior

To perform Bayesian inference of our BNN we use HMC which produces samples Neal et al. (2011); Cobb and Jalaian (2021), we then use these samples and learn a density estimator which we can be used as a prior for a new task. This resembles sequential Bayesian inference more closely than performing approximate inference and propagating a diagonal Gaussian posterior like in VCL. Consider $\mathcal{T}_1$, to then propagate the posterior $p(\theta|\mathcal{D}_1)$ we use a density estimator to fit a probability density on the samples to obtain $\hat{p}(\theta|\mathcal{D}_t)$. For the next task $\mathcal{T}_2$ we can use our density estimate $\hat{p}(\theta|\mathcal{D}_t)$ as a prior for a new HMC sampling chain for a new task $p_{t+1}(\theta) \coloneqq \hat{p}(\theta|\mathcal{D}_t) \approx p(\theta|\mathcal{D}_t)$. The density estimators we consider are a mixture of Gaussians (GMM) (we also tried using Normalizing Flows, RealNVP Dinh et al. (2016) which did not work robustly). See Figure 2 for a diagram of this procedure. The density estimator priors have two conditions required for use within the HMC sampling procedure. 1) That they are a valid probability density function 2) that they are differentiable with respect to the inputs, i.e. samples from the BNN which we want to create a density estimator for.

We construct a toy dataset of 10 classes where each class is generated according to a Gaussian distribution arranged in a circle about the origin, $x \in \mathcal{X} \in \mathbb{R}^2$ and each class visibly separable. Each task will require that we perform a binary classification of each of

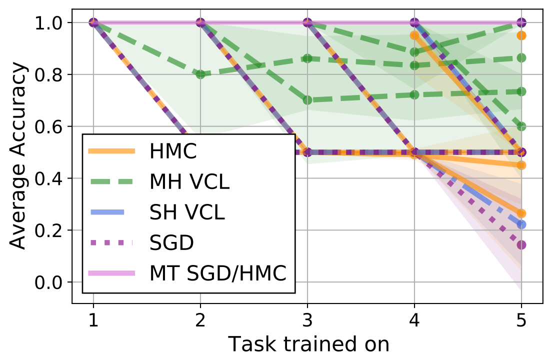


Figure 3: Continual learning binary classification results from 5 toy Gaussians tasks over 5 random seeds. The pink solid line is a multi-task (MT) baseline using an SGD MLP with the same model as the BNNs using for CL.

two adjacent classes for 5 tasks, see Figure 4 for a visualization of the data. We train a 2 layer BNN with hidden state size of 10, (training an MLP on the same multi-task dataset gets an accuracy of 1.0). For $\mathcal{T}_1$ we set the prior $p_1(\theta) = \mathcal{N}(0, \tau^{-1}\mathbb{1})$ with $\tau = 10$. We burn-in the HMC chain for 1000 steps and sample for 5000 more steps, we run 20 different chains to obtain samples from our posterior which we then pass to our density estimators. We use a step size of 0.001 and trajectory length of $L = 20$. We then assess convergence and effective sample size; all chains converge and the effective sample size of the samples from all chains is similar to other BNNs in Cobb and Jalaian (2021).

When using a GMM to provide a density estimate for the HMC samples we optimize for the number of components by using a holdout set of samples from the HMC sampling. We also ensure that samples from our density estimator $\hat{p}(\theta|\mathcal{D}_t)$ provide accurate estimates for the current task $\mathcal{T}_t$, see Section 8 for details on the HMC sampling diagnostics. We obtain the following results in Figure 3 which demonstrate that propagating the posterior in the form of a GMM density estimator fails to prevent forgetting. Moreover, comparing multi-head VCL with single-headed VCL or HMC (also single-headed) we see the multi-head VCL clearly outperform both methods indicating that the source of the knowledge retention is not through the propagation of the posterior but through the task specific parameters in each head. *Observation: we can conclude that just relying on sequential Bayesian inference is not sufficient in preventing forgetting in this setting.* We stress that we are assuming that the density estimator is able to encapsulate the HMC samples with a high degree of fidelity: indeed we are able to sample from the density and recover our BNN performance, Figure 5. This is backed up by current state of the art probabilistic methods in CL which use inducing points from previous tasks to prevent forgetting Titsias et al. (2019); Pan et al. (2020).

4. Prototypical Bayesian Continual Learning

We have shown that solely relying on sequential Bayes cannot ensure that forgetting is prevented and we have shown that modeling the CL problem in terms task specific and global parameters like in meta-learning (see Section 10) helps prevent forgetting as well. Hence we model the generative process of CL datasets where new tasks are comprised of new classes of images. We use a NN to embed images of each class and maintain this embedding as we learn continually. Each class embedding is modeled as a Gaussian; this is a probabilistic version of Prototypical Networks Snell et al. (2017) or iCarl Rebuffi et al. (2017) which both create embedding functions for different classes which are simply arithmetic means over embeddings and perform classification through nearest neighbors. Our approach borrows from the meta-learning model, probabilistic clustering for online classification Harrison et al. (2019) and we re-cast the general generative model for probabilistic clustering in CL.

Model. The generative model, ProtoCL, the classes are generated from a categorical distribution with a Dirichlet prior:

$$y_{i,t} \sim \text{Cat}(p_{1:J}), \quad p_{1:J} \sim \text{Dir}(\alpha_t). \tag{7}$$

We learn a joint embedding space for our data with a NN, $z = f(x; w)$ with parameters w. The embedding space for each class is Gaussian whose mean has a prior which is also Gaussian:

$$z_{it}|y_{it} \sim \mathcal{N}(\bar{z}_{yt}, \Sigma_\epsilon), \quad \bar{z}_{yt} \sim \mathcal{N}(\mu_{yt}, \Lambda_{yt}^{-1}). \tag{8}$$

By modeling a mean embedding for each class we ensure that we are modeling each new task with a specific set of parameters. To ensure that the mean embeddings do not drift we need make use of a memory of past task data to prevent forgetting. In practice we use 1000 data points per task, see Fig 8 for a sensitivity analysis. This is similar to the most recent approaches in probabilistic CL Titsias et al. (2019); Kapoor et al. (2021); Pan et al. (2020). We can flexibly scale to large vision benchmarks as $f(\cdot\,; w)$ does the representation learning and can be e.g. a CNN encoder. For full details with regards to obtaining the log posterior of the embeddings and class labels needed for optimization, the recursive updates for Dirichlet and Gaussian priors and for making predictions see Section 11 for details. Code to reproduce all experiments is located here https://github.com/skezle/bayes_cl_posterior. The purpose of this model is not to say that we can get better results than other baselines, but rather to demonstrate that by modeling the generative process and not blindly relying on sequential Bayesian inference we can obtain strong initial results (Table 4) in comparison to other Bayesian CL methods.

Table 1: Mean accuracies across all tasks over several CL vision benchmarks (for *class incremental learning*, Section 11.2 Van de Ven and Tolias (2019)). All results are averages and standard errors over 10 seeds. *Uses the predictive entropy to make a decision about which head for class incremental learning.

Method	**Split MNIST**	**Split f-MNIST**	**Split CIFAR10**
VCL Nguyen et al. (2018)	33.01 ± 0.08	32.77 ± 1.25	-
HIBNN* Kessler et al. (2019)	$\mathbf{85.50 \pm 3.20}$	43.70 ± 20.21	-
FROMP Pan et al. (2020)	$\mathbf{84.40 \pm 0.00}$	68.54 ± 0.00	38.54 ± 17.50
ProtoCL (ours)	$\mathbf{85.69 \pm 2.92}$	$\mathbf{75.69 \pm 3.34}$	$\mathbf{51.65 \pm 3.57}$

5. Discussion

We revisit sequential Bayesian inference which underpins many methods in Bayesian CL. We show that if we were to follow Eq (5) strictly using HMC samples and assuming our density estimator captures these samples then we are unable to prevent forgetting. This is not entirely surprising when we revisit the Bayesian filtering literature where sequential

Bayesian inference can be used in close form to track data, but is unable to recall past tasks (see Section 9). Sequential Bayesian updating forgets and hence why previous probabilistic approaches to CL have also relied on multi-head architectures and coresets for instance. This is why we should alternatively focus on modeling the CL problem which is what ProtoCL does in a very simple fashion. Bayes is agnostic to the model, if one's model is poor then Bayes will still infer the parameters of this model.

Why do some methods which rely on sequential Bayesian inference perform well? Most work which performs sequential Bayesian inference uses variational inference. One explanation could be that variational BNNs are known to be sparse due to variational overpruning Trippe and Turner (2018) and so the variational BNNs have more capacity to learn more tasks through Swaroop et al. (2019); Mallya and Lazebnik (2018). **Is sequential Bayes useless for CL?** No, it provides a form of regularization of the likelihood. In the case of VI the regularization is explicit in the form of a KL divergence. However, sequential Bayes on its own is not enough to prevent forgetting and ensuring past task information is not forgotten.

6. Acknowledgments

We would like to thank Sebastian Farquhar and Siddharth Swaroop for interesting discussions. This material is based upon work supported by the United States Air Force and DARPA under Contract No. FA8750-20-C-0002. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force and DARPA.

References

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

Adam D Cobb and Brian Jalaian. Scaling hamiltonian monte carlo inference for bayesian neural networks with symmetric splitting. *Uncertainty in Artificial Intelligence*, 2021.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10(3):197–208, 2000.

Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

Sayna Ebrahimi, Mohamed Elhoseiny, Trevor Darrell, and Marcus Rohrbach. Uncertainty-guided continual learning with bayesian neural networks. *arXiv preprint arXiv:1906.02425*, 2019.

Sebastian Farquhar and Yarin Gal. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733*, 2018.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. *arXiv preprint arXiv:1806.02817*, 2018.

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

Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E Turner. Meta-learning probabilistic inference for prediction. *arXiv preprint arXiv:1805.09921*, 2018.

Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F-radar and signal processing*, volume 140, pages 107–113. IET, 1993.

Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018.

Alex Graves. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24, 2011.

James Harrison, Apoorva Sharma, Chelsea Finn, and Marco Pavone. Continuous meta-learning without tasks. *arXiv preprint arXiv:1912.08866*, 2019.

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.

Ferenc Huszár. On Quadratic Penalties in Elastic Weight Consolidation. Technical report, 2017.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

Sanyam Kapoor, Theofanis Karaletsos, and Thang D Bui. Variational auto-regressive gaussian processes for continual learning. In *International Conference on Machine Learning*, pages 5290–5300. PMLR, 2021.

Samuel Kessler, Vu Nguyen, Stefan Zohren, and Stephen Roberts. Hierarchical indian buffet neural networks for bayesian continual learning. *arXiv preprint arXiv:1912.02290*, 2019.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell.

Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. ISSN 0027-8424. doi: 10.1073/pnas.1611835114.

Noel Loo, Siddharth Swaroop, and Richard E Turner. Generalized variational continual learning. *arXiv preprint arXiv:2011.12328*, 2020.

David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

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

Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard E Turner, and Mohammad Emtiyaz Khan. Continual deep learning by functional regularisation of memorable past. *arXiv preprint arXiv:2004.14070*, 2020.

Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.

Michael K Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American statistical association*, 94(446):590–599, 1999.

Sachin Ravi and Alex Beatson. Amortized bayesian meta-learning. In *International Conference on Learning Representations*, 2018.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured laplace approximations for overcoming catastrophic forgetting. *arXiv preprint arXiv:1805.07810*, 2018.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4528–4537. PMLR, 2018.

Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017.

Siddharth Swaroop, Cuong V Nguyen, Thang D Bui, and Richard E Turner. Improving and understanding variational continual learning. *arXiv preprint arXiv:1905.02099*, 2019.

Michalis K Titsias, Jonathan Schwarz, Alexander G de G Matthews, Razvan Pascanu, and Yee Whye Teh. Functional regularisation for continual learning with gaussian processes. *arXiv preprint arXiv:1901.11356*, 2019.

Brian Trippe and Richard Turner. Overpruning in variational bayesian neural networks. *arXiv preprint arXiv:1801.06230*, 2018.

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

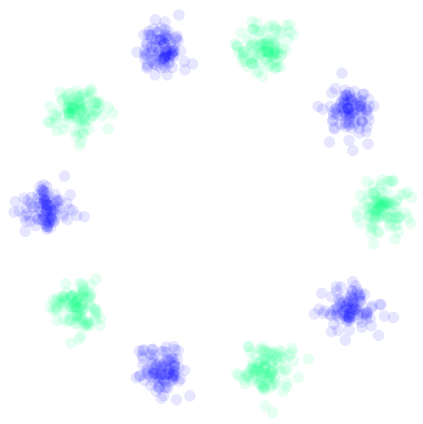


Figure 4: The toy Gaussian dataset used as a simple dataset to perform binary classification in a CL setting. Color indicates a different class. There are 5 binary classification tasks of adjacent Gaussians which we are required to learn continually. Each task uses a new Gaussian dataset.

7. The Toy Gaussians Dataset

See Figure 4 for a visualization of the toy Gaussians dataset which we use to demonstrate using a density estimator on HMC samples as a surrogate of the posterior and use this as a prior for a new task. We construct a simple CL problem of 5 tasks of binary classification of data arranged in a ring. Each task is binary classification of two adjacent Gaussian datasets, then the next task is the next two adjacent Gaussians (not including dataset see in $\mathcal{D}_1$). This should enable us to use small BNNs and efficiently perform HMC sampling and density estimation over HMC samples. With just a 2 layer network with 10 neurons as a network can perform binary classification over this entire 5 task binary classification problem in a multi-task fashion. Hence we postulate that a BNN with the same size should be able to learn all 5 binary classification tasks continually and that the functional approximation capabilities of the network are not a constraint fro learning.

8. Density Estimation Diagnostics

We provide plots to show that the HMC chains which we run to sample from the posterior have converged in Figure 6. The tasks are the toy Gaussians dataset described in Section 7. We run 20 HMC sampling chains and randomly select one chain to plot for each seed (of 10). We run HMC over 10 seeds and aggregate the results. The posterior $p(\theta|\mathcal{D}_1)$ is approximated with a GMM and used as a prior for the second task and so forth for tasks $\mathcal{T}_{2:5}$. Our BNN is of size of 2 layers with Gaussian weights and a hidden state size of 10.

We provide empirical evidence to show that the density estimators have fit to HMC samples of the posterior in Figure 5. Where we show the number of components of the GMM density estimator which we use as a prior for a new task, the BNN accuracy when sampling weights from our GMM and the effective sample size (ESS) of the 20 chains which is a measure of how correlated the samples are (higher is better). These ESS values are in line with previous work which uses HMC for BNN inference Cobb and Jalaian (2021).

9. Bayesian Continual Learning as Filtering

In the Bayesian filtering literature the parameters $\{\boldsymbol{\theta}_t\}$ are denoted as a hidden state all relevant information about $\{\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_k\}$ given observations up to time k is encapsulated

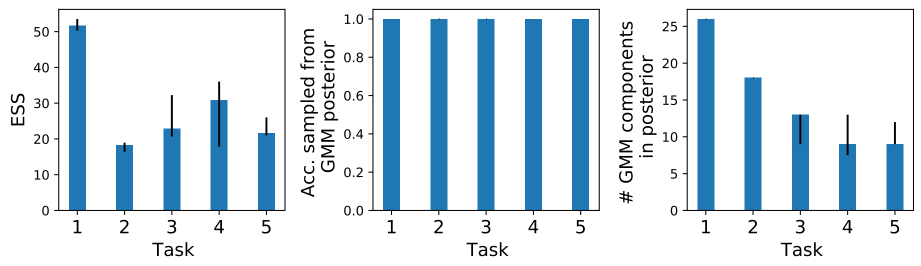


Figure 5: Diagnostics from using a GMM prior fit to samples of the posterior generated from HMC, all results are for 10 random seeds. **Left,** effective sample sizes (ESS) of the resulting HMC chains of the posterior, all are greater than those reported in other works using HMC for BNNs Cobb and Jalaian (2021). **Middle,** the accuracy of the BNN when using samples from the GMM density estimator instead from the samples from HMC. **Right,** The optimal number of components of each GMM posterior fitted with a holdout set by maximizing the likelihood.

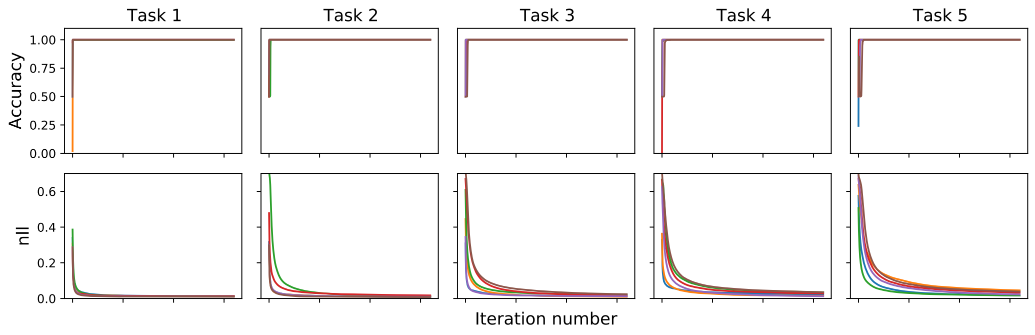


Figure 6: Convergence plots from a randomly sampled HMC chain (of 20) for each task over 10 different runs (seeds) for 5 tasks from the toy Gaussians dataset using a GMM density estimator as a prior conditioned on previous task data.

by the posterior distribution $p(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_k | y_0, y_1, \dots, y_k)$. We estimate this posterior distribution recursively in time; in particular the marginal distribution $p(\boldsymbol{\theta}_k | y_0, y_1, \dots, y_k)$ denoted as the *filtering distribution* Doucet et al. (2000, 2001). Bayesian filtering has been used for applications such as target tracking Gordon et al. (1993) or estimating the volatility of a security Pitt and Shephard (1999). In all but the simplest case of a Gaussian state-space model: Gaussian likelihood and Gaussian parameters (or hidden state), exact inference is intractable; the seminal *Kalman filter* relies on sequential Bayesian updates Kalman (1960).

9.1. Sequential Bayesian Estimation of a Single Parameter

Let's consider an even simpler setting than CL and consider data arriving online from a non-stationary distribution one point after another: $y_0, y_1, \dots$. The likelihood is $p(y_t|\theta_t) = \mathcal{N}(y_t; f(\cdot\,; \theta_t), \sigma^2)$ such that $y_t = f(\cdot\,; \theta_t) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and where the function $f(\cdot\,; \theta_t) = \theta_t$ are simply the parameters. We consider a Gaussian prior over the parameters θ such that $p(\theta_0) = \mathcal{N}(\theta_0; 0, \sigma_0^2)$. Considering the calculation of our posterior at time t since our prior and likelihood are Gaussian then so is the posterior hence the prior and posterior are $\mathcal{N}(\theta_{t-1}; \hat{\theta}_{t-1}, \hat{\sigma}^2_{t-1})$ and $\mathcal{N}(\theta_t; \hat{\theta}_t, \hat{\sigma}^2_t)$ respectively. By using sequential Bayesian inference we can have closed form update equations for our posterior parameters:

$$\hat{\sigma}_t^2 = \left(\frac{1}{\hat{\sigma}^2_{t-1}} + \frac{1}{\hat{\sigma}^2} \right)^{-1} \tag{9}$$

$$\hat{\theta}_t = \frac{\hat{\sigma}^2}{\sigma^2} y_t + \frac{\hat{\sigma}^2}{\hat{\sigma}^2_{t-1}} \hat{\theta}_{t-1}. \tag{10}$$

We can see from our sequential estimate of the mean that the posterior is a linear combination of the observation the prior. If the observations are non-stationary like in CL then the mean parameters will start to deviate from the prior and the posterior will eventually *track* the new distribution of observations. We show this in the following notebook.

10. Multi-head Continual Learning and Meta-Learning

We perform sequential Bayesian inference with a BNN by using a GMM density estimator to fit the samples of the posterior. By fitting a mixture of Gaussians as our posterior for task $\mathcal{T}_t$ and using it as a prior before performing HMC for task $\mathcal{T}_{t+1}$ we approach exact sequential Bayes on small BNNs for a toy dataset. This is still not exact sequential Bayesian inference, however we show that by sampling from the GMM we can obtain 100% accuracy, hence the GMM is a high fidelity posterior. By using this procedure show that we are unable to prevent forgetting and multi-headed VCL performs best. This leads us to conclude that the main mechanism to prevent forgetting is the use of different heads for different tasks. The different heads are acting to retain task specific information. More generally, it is the use of task specific parameters to enable to preserve previous task predictors. This is similar to the problem of *meta-learning* which specifically decomposes the meta-learning solutions into global shared parameters and task specific parameters. In this section we will elaborate on this by exploring gradient-based meta-learning Finn et al. (2017, 2018); Grant et al. (2018) and meta-learning probabilistic inference methods Gordon et al. (2018); Ravi and Beatson (2018). The graphical model for the meta-learning is shown in Figure 7, each

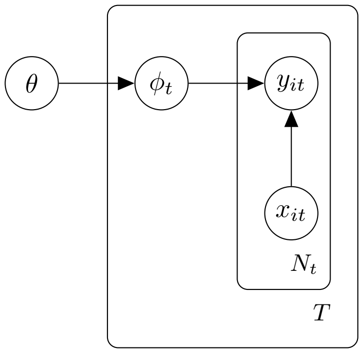


Figure 7: Graphical model of meta-learning models where θ are the global shared parameters and ϕ_t are task specific parameters for $t \in \{1, \dots, T\}$ and there are $i = \{1, \dots, N_t\}$ points in the dataset for task t.

task specific parameter ϕ_t is distinct and influences the prediction of y_{it} given x_{it}, a meta level global parameter θ captures global information about all tasks and influences ϕ_t. The goal of meta-learning is to learn θ which generalizes to new unseen datasets.

Gradient based meta-learning. We learn θ in a meta-learning stage by maximum likelihood. Consider a task inputs $X^t = \{x_i^t\}_{i=1}^{N_t}$, and the set of all datasets from all tasks is $\{X^t\}_{t=1}^T$ which we denote as $\{X\}$ in short hand, then without splitting our dataset into support and query sets for ease of notation and employing the factorization in Figure 7 then:

$$p(\{Y\}|\{X\}, \theta) = \int p(\{Y, \phi\}|\{X\}, \theta) d\phi, \tag{11}$$

$$= \int \prod_t p(\phi_t|\theta) \prod_{i=1}^{N_t} p(y_{it}|x_{it}, \phi_t) d\phi_t. \tag{12}$$

In MAML Finn et al. (2017); Grant et al. (2018) the distribution of task specific parameters $p(\phi_t|\theta)$ is approximated by point estimates around $\hat{\phi}_t = \theta + \alpha \nabla_\theta p(Y^t|X^t, \theta)$ and the global parameters θ act as an initialization of the NN parameters for few shot adaptation to obtain $\hat{\phi}_t$ and α is the SGD step-size. Hence the we can rewrite the marginal likelihood in Eq. (12) as:

$$p(\{Y\}|\{X\}, \theta) \approx \prod_t \prod_{i=1}^{N_t} p(y_i^{(t)}|x_i^{(t)}, \hat{\phi}_t), \tag{13}$$

which is the MAML objective.

Meta-learning probabilistic inference. Instead of a gradient based approach to meta-learning we can meta-learn an amortization network which produces task specific parameters Gordon et al. (2018); Ravi and Beatson (2018). For a test point x, a prediction is governed by $p(y|x, \phi_t, \theta)$ where the task specific parameters are generated by an amortized variational inference $\phi_t \sim q_\psi(\phi|\mathcal{D}_t, \theta)$: a feed forward network with parameters ψ is learned to output task specific parameters given training $\mathcal{D}_t = \{(x_n^{(t)}, y_n^{(t)})\}_{n=1}^{N_t}$. In practice a feature extractor $h_\theta(x)$ is shared across all tasks and feeds into a task specific linear classifier head, ϕ_t, this is learned in an end-to-end manner backpropagating back to θ and ψ via the objective:

$$\mathcal{L}(\theta, \psi) = \frac{1}{MT} \sum_{m,t} \log \int p(y_m^{(t)} | x_m^{(t)}, \phi^{(t)}, \theta) \cdot q_\psi(\phi^{(t)} | D^{(t)}, \theta) d\phi, \tag{14}$$

in practice this can be optimized via Monte-Carlo integration Gordon et al. (2018).

Meta-learning and multi-head CL. The meta-learning methods detailed above have a clear demarcation between shared parameters and task specific parameters. In MAML-type gradient based meta-learning the global parameters θ is an NN initialization which generalizes in a few steps of gradient descent to obtain the task specific parameter ϕ_t. For amortized inference based meta-learning approaches the shared parameters are the feature extractor and amortization network which output task specific parameter heads. In practice this amortization approach is similar to multi-head CL models which also has a shared feature extractor and task specific heads, although none of the Bayesian CL literature explicitly models the problem in terms of shared parameters and task specific parameters. This is the key idea we wish to highlight in this work. It is clear that sequential Bayesian inference is not enough to solve CL problems and instead they need to model this task specific and shared parameters which they implicitly implement with task specific heads. Simply using a single BNN to model all the tasks in the CL loop is too general of a solution and is bound to suffer from catastrophic forgetting. With this in mind we tackle the CL problem with a probabilistic model which explicitly models each class and is decomposed into task specific parameters and shared parameters in Section 4.

11. Prototypical Bayesian Continual Learning

The generative model for CL problems where new tasks are comprised of new classes can be modeled as by using a categorical distribution with a Dirichlet prior:

$$y_{i,t} \sim \mathrm{Cat}(p_{1:J}), \quad p_{1:J} \sim \mathrm{Dir}(\alpha_t). \tag{15}$$

We learn a joint embedding space for our data with a NN, $z = f(x; w)$ with parameters w. The embedding space for each class is Gaussian whose mean has a prior which is also Gaussian:

$$z_{it} | y_{it} \sim \mathcal{N}(\bar{z}_{yt}, \Sigma_\epsilon), \quad \bar{z}_{yt} \sim \mathcal{N}(\mu_{yt}, \Lambda_{yt}^{-1}). \tag{16}$$

By ensuring that we have an embedding per class and using a memory of past data to ensure that the embedding does not drift we can solve the incremental class learning problem in CL. The posterior parameters are $\eta_t = \{\alpha_t, \mu_{1:J,t}, \Lambda_{1:J,t}^{-1}\}$.

11.1. Inference

As the Dirichlet prior is conjugate with the Categorical distribution and so is the Normal distribution with a Normal prior over the mean, then we can calculate posteriors in closed

form and update our parameters as we see new data without using gradient based updates. We optimize the model by maximizing the posterior of the data and use the softmax over class probabilities to perform predictions, we perform gradient based learning of the NN embedding function $f(\cdot)$. The posterior parameters are $\eta_t = \{\alpha_t, \mu_{1:J,t}, \Lambda^{-1}_{1:J,t}\}$. As the Dirichlet prior is conjugate with the Categorical distribution and so the Normal distribution with a Normal prior over the mean, we can update these distributions in closed form.
Sequential updates. We can obtain our posterior:

$$\begin{aligned} p(\eta_t|\mathcal{D}_1) &\propto p(\mathcal{D}_t|\eta_t)p(\eta_t) && (17)\\ &= \prod_{i=1}^{N_t} p(z_t^i|y_t^i;\bar{z}_{y_t},\Sigma_{\epsilon,y_t})p(y_t^i;p_{1:J})p(p_{i:J}|\alpha_t)p(z_{y_t};\mu_{y_t,t},\Lambda^{-1}_{y_t,t}) && (18)\\ &= \mathcal{N}(\mu_{t+1},\Sigma_{t+1})\mathrm{Dir}(\alpha_{t+1}). && (19)\end{aligned}$$

Concentrating on the Categorical-Dirchlet conjugacy:

$$\begin{aligned} \mathrm{Dir}(\alpha_{t+1}) &\propto p(p_{1:J}|\alpha_t)\prod_{i=1}^{N_t} p(y_t^i;p_{i:J}) && (20)\\ &\propto \prod_{j=1}^{J} p_j^{\alpha_j-1}\prod_{i=1}^{N_t}\prod_{j=1}^{J} p_j^{\mathbb{1}(y_t^i=j)} && (21)\\ &= \prod_{j=1}^{J} p_j^{\alpha_j-1+\sum_{i=1}^{N_t}\mathbb{1}(y_t^i=j)}. && (22)\end{aligned}$$

Thus:

$$\alpha_{t+1,j} = \alpha_{t,j} + \sum_{i=1}^{N_t}\mathbb{1}(y_t^i=j). \qquad (23)$$

Also, due to Normal-Normal conjugacy, then the posterior for the Gaussian prototype of the embedding for each class is:

$$\begin{aligned} \mathcal{N}(\mu_{t+1},\Sigma_{t+1}) &\propto \prod_{i=1}^{N_t}\mathcal{N}(z_t^i|y_t^i;\bar{z}_{y_t},\Sigma_\epsilon)\mathcal{N}(\bar{z}_{y_t};\mu_{y_t,t},\Lambda^{-1}_{y_t}) && (24)\\ &= \prod_{y_t\in C_t}\mathcal{N}(z_{y_t}|y_t;\bar{z}_{y_t},\frac{1}{N_{y_t}}\Sigma_\epsilon)\mathcal{N}(\bar{z}_{y_t};\mu_{y_{t+1}},\Lambda^{-1}_{y_t}) && (25)\\ &= \prod_{y_t\in C_t}\mathcal{N}(\bar{z}_{y_t};\mu_{t+1},\Lambda^{-1}_{y+1}). && (26)\end{aligned}$$

The update equations for the mean and variance of the posterior are:

$$\begin{aligned} \Lambda^{-1}_{y_{t+1}} &= \Lambda^{-1}_{y_t} + N_{y_t}\Sigma_\epsilon^{-1} && (27)\\ \Lambda_{y_{t+1}}\mu_{y_{t+1}} &= N_{y_{t+1}}\Sigma_\epsilon^{-1}\bar{z}_{y_t} + \Lambda_{y_t}\mu_{y_t}, \quad \forall y_t\in C_t. && (28)\end{aligned}$$

Objective. The likelihood we want to optimize is:

$$p(z,y) = \int p(z,y|\theta)p(\theta)d\theta, \tag{29}$$

where $\theta = \{p_{1:J}, \bar{z}_{y_t}\}$,

$$p(z,y) = \int \prod_{i=1}^{N_t} p(z_{it}|y_{it};\bar{z}_{y_t},\Sigma_\epsilon)p(y_{it}|p_{1:J})p(p_{1:J}|\alpha_t)p(\bar{z}_{y_t};\mu_{y_t,t},\Lambda_{y_t,t}^{-1})dp_{1:J}d\bar{z}_{y_t} \tag{30}$$

$$= \int \prod_{i=1}^{N_t} p(z_{it}|y_{it};z_{y_t},\Sigma_\epsilon)d\bar{z}_{y_t} \underbrace{\int \prod_{i=1}^{N_t} p(y_{it}|p_{1:J})p(p_{1:J}|\alpha_t)dp_{1:J}}_{\prod_i p(y_i)=p(y)} \tag{31}$$

$$= p(y)\prod_{i=1}^{N_t} Z_i^{-1}\int \mathcal{N}(\bar{z}_{y_{it}};c,C)d\bar{z}_{y_t} \tag{32}$$

$$= p(y)\prod_{i=1}^{N_t} \mathcal{N}(z_{it}|y_{it};\mu_{y_t,t},\Sigma_\epsilon+\Lambda_{y_t,t}^{-1}). \tag{33}$$

Where in Eq. (32) we use §8.1.8 in Petersen et al. (2008). The term $p(y)$ is:

$$p(y) = \int p(y|p_{1:J})p(p_{1:J}|\alpha_t)dp_{1:J} \tag{34}$$

$$= \int p_y \frac{\Gamma(\sum_{j=1}^J \alpha_j)}{\prod_{j=1}^J \Gamma(\alpha_j)} \prod_{j=1}^J p_j^{\alpha_j-1}dp_{1:J} \tag{35}$$

$$= \frac{\Gamma(\sum_{j=1}^J \alpha_j)}{\prod_{j=1}^J \Gamma(\alpha_j)} \int \prod_{j=1}^J p_j^{\mathbb{1}(y=j)+\alpha_j-1}dp_{1:J} \tag{36}$$

$$= \frac{\Gamma(\sum_{j=1}^J \alpha_j)}{\prod_{j=1}^J \Gamma(\alpha_j)} \frac{\prod_{j=1}^J \Gamma(\mathbb{1}(y=j)+\alpha_j)}{\Gamma(1+\sum_{j=1}^J \alpha_j)} \tag{37}$$

$$= \frac{\cancel{\Gamma(\sum_{j=1}^J \alpha_j)}}{\prod_{j=1}^J \Gamma(\alpha_j)} \frac{\prod_{j=1}^J \Gamma(\mathbb{1}(y=j)+\alpha_j)}{\sum_{j=1}^J \alpha_j \cancel{\Gamma(\sum_{j=1}^J \alpha_j)}} \tag{38}$$

$$= \frac{\prod_{j=1,j\neq y}^J \Gamma(\alpha_j)}{\prod_{j=1}^J \Gamma(\alpha_j)} \frac{\Gamma(1+\alpha_y)}{\sum_{j=1}^J \alpha_j} \tag{39}$$

$$= \frac{\prod_{j=1,j\neq y}^J \Gamma(\alpha_j)}{\prod_{j=1}^J \Gamma(\alpha_j)} \frac{\alpha_y\Gamma(\alpha_y)}{\sum_{j=1}^J \alpha_j} \tag{40}$$

$$= \frac{\alpha_y}{\sum_{j=1}^J \alpha_j}, \tag{41}$$

where we use the identity $\Gamma(n+1) = n\Gamma(n)$.

Predictions. To make a prediction for a test point $\hat{x}$:

$$p(\hat{y}|\hat{x},x_{1:t},y_{1:t}) = p(\hat{y}|\hat{x},\eta_t) = \frac{p(\hat{y},\hat{x}|\eta_t)}{\sum_{y'} p(y',\hat{x}|\eta_t)} = \frac{p(\hat{y},\hat{z}|\eta_t)}{\sum_{y'} p(y',z|\eta_t)}, \tag{42}$$

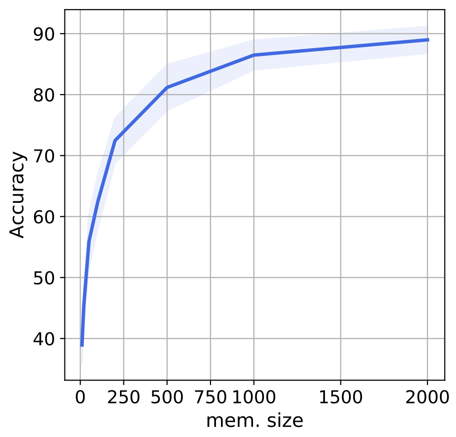


Figure 8: Split MNIST average accuracy over 5 tasks for different memory sizes for previous tasks. Accuracies are over 10 seeds. The use of memories ensures that previous task class embeddings remain the same and distinct from new classes.

where η_t is a sufficient statistic for (x, y).

Preventing forgetting. As we wish to retain the task specific embeddings and probabilistic model parameters as if we were performing multi-task learning: as if we had access to the entire dataset. Then at the end of learning a task $\mathcal{T}_t$ we take a small subset of the data $\mathcal{M}_t \subset \mathcal{D}_t$ as a memory to ensure that posterior parameters and prototypes do not drift.

11.2. Experimental Setup

We perform the evaluation of our model using *class-incremental learning*: for each new task our model needs to predict the actual class of the data in the task. For instance for the Split-MNIST example explored in Figure 1, for the first task images of $\{0, 1\}$ are shown and the class labels are $\{0, 1\} \in \mathbb{N}^{10}$, then for the second task $\{2, 3\}$ and the class labels are $\{2, 3\} \in \mathbb{N}^{10}$ and so on.

11.3. Prototypical Bayesian Continual Learning: Additional Results

We show the variation of in performance of ProtoCL with the size of the memory of each task in Figure 8. We also show the variation of the Dirichlet parameters in Figure 9, the class probability priors all increase as the corresponding task datasets are seen.

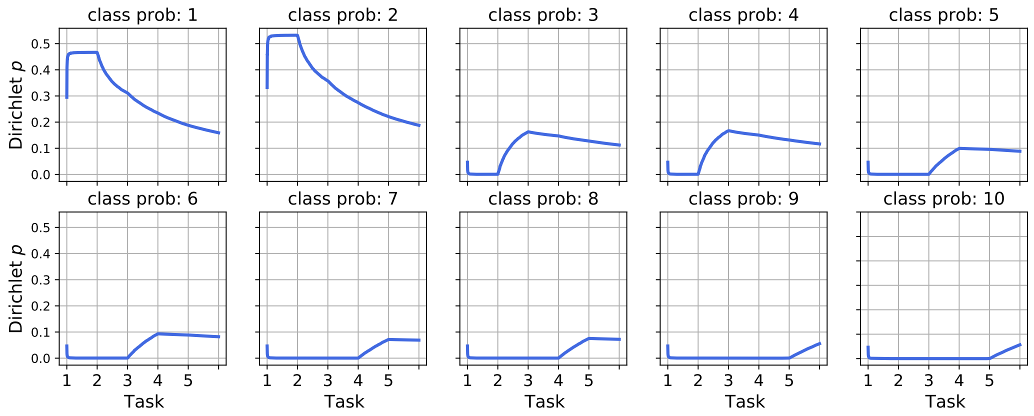


Figure 9: The evolution of the Dirichlet parameters α_t for each class over the course of 5 Split MNIST tasks for ProtoCL. All α_{tj} are shown over 10 seeds with ± 1 standard error. By the end of training all classes are roughly equally likely.