Bayesian Flow Networks in Continual Learning

Mateusz Pyla
IDEAS NCBR
Jagiellonian University, Faculty of Mathematics and Computer Science
Jagiellonian University, Doctoral School of Exact and Natural Sciences
mateusz.pyla@ideas-ncbr.pl

Kamil Deja
IDEAS NCBR
Warsaw University of Technology

Bartłomiej Twardowski
IDEAS NCBR
Autonomous University of Barcelona

Tomasz Trzciński
IDEAS NCBR
Warsaw University of Technology
Jagiellonian University, Faculty of Mathematics and Computer Science

Abstract

Bayesian Flow Networks (BFNs) has been recently proposed as one of the most promising direction to universal generative modelling, having ability to learn any of the data type. Their power comes from the expressiveness of neural networks and Bayesian inference which make them suitable in the context of continual learning. We delve into the mechanics behind BFNs and conduct the experiments to empirically verify the generative capabilities on non-stationary data.

1 Introduction

Diffusion models [15] have been progressively advancing the state-of-the-art in generative modelling, especially in the field of image processing [2, 8, 11]. This is thanks to the usage of the diffusion processes that allows to learn complex data distributions [7, 9, 11, 16, 17].

However, diffusion models tend to struggle when in comes to non-continuous data. This is mostly because of the fact that denoising process for discrete, discretised or tabular data is not easy to define. Therefore, Alex Graves et al. recently introduced **Bayesian Flow Networks (BFNs)** [6] to efficiently train the model and iteratively update the data parameters without forward pass. The general idea behind this technique is to change the way in which we model training data where instead of modelling a single instance, authors propose to output the parameters of the distribution that best fits the training data. The main motivation behind this concept is that by doing so, authors introduce an elegant way to directly model the discrete data distribution.

In this work, we argue, that with direct modelling of parameters describing the original training data, BFNs can also be used to efficiently consolidate portions of separate data chunks. Therefore, we relate to the problem of continual learning, which tackles the ability of ML models to learn progressively as new data arrive. Bayesian update is an elegant way to manage prior belief and information from new observations. However, in Bayesian learning we often face the issue of turning theory into practical implementations, limiting the use of the Bayesian learning paradigm [3, 20, 21].

In this preliminary studies, we show the first benchmark of Bayesian Flow Networks in Continual Learning setup. We show how we can adapt several known techniques that prevent catastrophic forgetting in neural networks to continually train BFNs. We highlight their strengths and drawbacks and discuss future directions on how to employ BFNs to continually consolidate knowledge.

Preprint. Under review.

2 Related Work

Continual Learning (CL) gathers various approaches in machine learning that aim at reducing catastrophic forgetting, a phenomenon where models suffer from abrupt loss in performance when retrained with additional data. Usually there are three standard group of approaches that try to mitigate this issue: (i) architectural approaches – methods that focus on the structure of the model itself that adds task-specific submodules to the architecture; (ii) memory approaches, methods involve storing some extra information in memory which are then used to rehearse knowledge during training in subsequent task; (iii) Regularization Approaches: that identify the important weights for the learned tasks and penalise the large updates on those weights when learning a new task [18, 12].

Several of those approaches were applied in generative continual learning. In particular, [13] adapt several regularisation-based methods and introduce Variational Continual Learning where additional architectural change is added with each task. In BooVAE [5] an additive aggregated posterior expansion technique is used to continually trained Variational Autoencoders, while [1] propose to continually disentangle data representations with VAE. Several methods train GANs in CL scenarios, e.g. using memory buffer [22]. Most similarly to this work, in [23] authors benchmark the possibilities of continual-learning of diffusion models with recent CL strategies.

3 Method

Although BFNs work on different type of data, both discrete and continuous time steps, the most approachable way to understand the dynamics is through continuous data in the discrete setting and extending it as in 6.1.

3.1 Inference

We use neural network ψ parameterised by θ to learn the parameters ξ controlling the data distribution. The underlying distribution is complex, however we can sample from it. The general idea is to start with uneducated guess of normal distribution with high variance, and iteratively improve the estimation of data distribution with the help of the network as we sample more and more data. We assume that we model the data only with Gaussians.

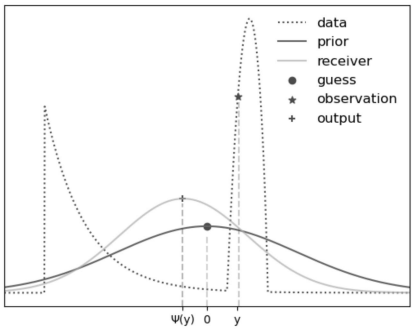

Figure 1: BFNs: Our goal is to model one dimensional data distribution controlled by unknown ξ, which we can only sample from. We start off with some initial prior belief (0 in this case) that we are very uncertain of (blue prior). We sample an observation y and add noise to obtain what we call a sender distribution – Gaussian centered at y. We pass the parameters of input to the neural network obtaining output distribution, improved version enriched by jointly processes of all the variables. We comply with the noise scheduler to obtain orange receiver. We minimise the KL divergence between the sender and receiver so that our output gets closes to the samples from true data distribution.

As in diffusion models, we set the number of steps and we establish the accuracy scheduler managing the usefulness of noised samples for various time steps. We start from the prior belief, for instance centered at 0 with huge standard deviation. We want our network to predict better data parameters from the current estimate. To update network weights, we calculate the gradient as a KL divergence between the predicted and original data in their noised forms. While explaining the mathematical formulation behind the process, we point out concrete example in 1.

More rigorously, we threat each data variable separately. Due to the independence, the input distribution can be expressed as the product of one dimensional distributions.

$$p_I(\mathbf{x} \mid \xi) = \Pi_{d=1}^{D} p_I(x^{(d)} \mid \xi^{(d)}) \tag{1}$$

Rather than data points, we receive the noisy samples forming the normal distribution centered at the true values with variance purely depended on the accuracy scheduler.

$$p_S(\mathbf{y} \mid \mathbf{x}; \alpha) = \Pi_{d=1}^{D} p_S\left(y^{(d)} \mid x^{(d)}; \alpha\right) \tag{2}$$

Since we proceed each dimension independently, we need global feedback coming from the interactions between variables. The role of neural network is to update the guess knowing the previous belief, so that we can better decode the sended sample.

$$p_O(\mathbf{x} \mid \xi, t) = \Pi_{d=1}^{D} p_O(x^{(d)} \mid \Psi^{(d)}(\xi, t)) \tag{3}$$

Since we are not aware of the true $x^{(d)}$, knowing $p_S\left(\cdot \mid x^{(d)}; \alpha\right)$ we can only marginalise over all possible values $x'^{(d)}$ weighted by the output probability, obtaining the receiver:

$$p_R(\mathbf{y} \mid \xi; t, \alpha) = \mathop{\mathbb{E}}_{p_O(\mathbf{x}'|\xi;t)} p_S(\mathbf{y} \mid \mathbf{x}'; \alpha) \tag{4}$$

Iteratively, for the next time steps, we apply Bayesian updates to improve the input distribution (that accumulates only local knowledge about a single dimension) by the acquired knowledge from receiver (that encodes global knowledge about interactions between dimensions). Both input and sender are Gaussian distributions and factorised independently, hence the update is straightforward: $\rho_{i+1} = \rho_i + \alpha$ and $\mu_{i+1} = \frac{\rho_i \mu_i + \alpha y}{\rho_{i+1}}$ Once we set the number of steps to infinity, under mild conditions for the scheduler, we are able to efficiently compute the dynamics:

$$p_F(\xi \mid \mathbf{x}; t) = p_U(\xi \mid \xi_0, \mathbf{x}; \beta(t)). \tag{5}$$

There is a freedom in choosing the underlying network, as long as it returns the new parameters of data (U-Net [14], Transformers [19], TabTransformer [10]) and inference conditions on the time step.

The proposed scheduler is of form $\beta(t) \doteq \sigma_1^{-2t} - 1$ for $t \in [0, 1]$ yielding accuracy rate $\alpha(t) = \frac{2 \log \sigma_1}{\sigma_1^{2t}}$. When α is 0, the model is uninformative of samples and confidence increases with the higher values. σ_1 is the hyperparameter standing for standard deviation at the final time.

3.2 Training

Loss function can be intuitively understood as costs of revealing the underlying data distribution with the least possible effort or information. Our objective is to match output distribution to the data distribution indirectly by optimizing KL divergence between their noisy versions. Specifically, we minimise KL divergence between sender and receiver distributions: $D_{KL}\left(p_S \parallel p_R\right)$ in order to bring output predictions closer and closer to the true data values.

$$L^n(\mathbf{x}) \doteq \mathop{\mathbb{E}}_{p(\xi_1, \ldots, \xi_{n-1})} \sum_{i=1}^{n} D_{KL}\left(p_S\left(\cdot \mid \mathbf{x}; \alpha_i\right) \parallel p_R(\cdot \mid \xi_{i-1}; t_{i-1}, \alpha_i)\right), \tag{6}$$

This loss indirectly optimises our true goal:

$$L^r(\mathbf{x}) = - \mathop{\mathbb{E}}_{p_F(\xi|\mathbf{x},1)} \ln p_O(\mathbf{x} \mid \xi; 1). \tag{7}$$

Let us note that this kind of form follows information-theory interpretation: we minimise the number of nats required to transmits a sample between two distributions.

3.3 BFN in Continual Learning

We propose to extend the basic idea of BFNs in order to benchmark it with several known continual-learning strategies. In particular, we start with a simple regularisation strategy, where we prevent model in subsequent tasks to diverge from the previous by penalising $\mathcal{L}_1$ or $\mathcal{L}_2$ norm.

We compare the regularisation approach with two rehearsal-based methods. In the first one we employ a simple buffer-based rehearsal where we store a subset of previous data examples in a buffer and use them together with new data samples when retraining a model on new tasks. In the second one, taking advantage of the generative model we continually train, we propose to generate examples from previous tasks in order to use them as rehearsal samples in a generative replay approach.

4 Experiments

We evaluate the performance of BFNs in Continual Learning using the standard MNIST dataset and our new scenario with tabular data on US flights connections in 2013 [4]. One of the most common setting in which we are able to assess the continual learning capabilities of the proposed model is

to split the training dataset into disjoint chunks and perform the training in a sequential way. In Class-Incremental Learning set up, each task often contains the same . Each task τ_i is associated with a dataset $\mathcal{D}_i$, and the objective is to model the distribution of $\mathcal{D}_i$.

In particular, we split the MNIST in CIL setting 5×2 by dividing it into 5 tasks, each binary classification of two consecutive digits. Following [6], we also binarise the images. In flight dataset we divide group flights by month of the journey obtaining 12 tasks.

4.1 Image dataset

In Figure 2, we present the results of our experiments with MNIST dataset. To measure the catastrophic forgetting, after each task, we generate 1000 examples and report the share of each class as measured by the externally trained classifier. As visible, in finetuning (without any CL strategy), we can observe catastrophic forgetting as with each new task model abruptly forgets how to generate examples from the previous classes. On the other hand, both: buffer-based and generative-based replay prevent catastrophic forgetting, as even after the last task, we can observe some generations of classes from the first task. For qualitative analysis we provide some samples in Figure 4.

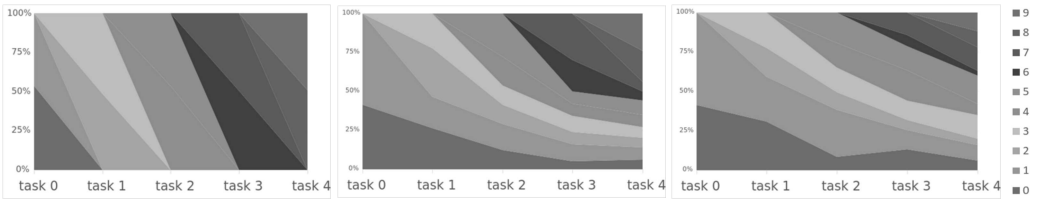

Figure 2: Results of classification of the generated images (a) left: using finetuning (b) middle: memory-based (c) right: generative-based method. (d) The colours corresponds to percentage share of consecutive digits across the sequential training.

5 Tabular data

To evaluate the performance of BFNs in modelling categorical data in the continual learning scenario, we refer to the problem of tabular data modelling. The results are presented in Figure 3. We inspect the test loss metric as a proxy for model surprise of provided data.

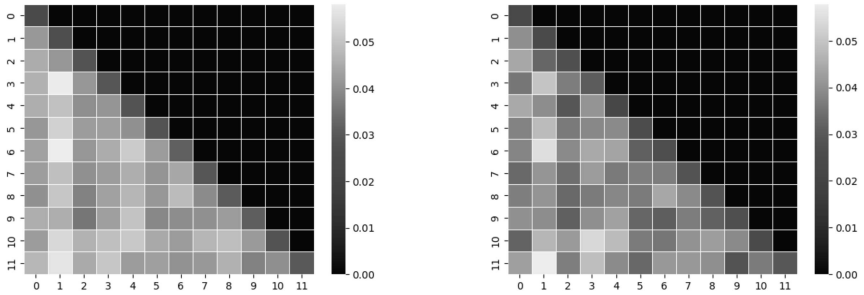

Figure 3: Results of loss applied on test images (a) left: using finetuning (b) right: generative-replay method. Loss induced by BFNs measures bits by dimension and offers (negative log-) likelihood estimation interpretation. On Y axis incrementally we are proceed with tasks, whereas values on X axis corresponds to data batches from indicated tasks.

6 Conclusion, Limitations and Future work

Bayesian Flow Networks (BFNs) are exciting family of generative models that are able to deal with various type of data. In this work we highlight that modelling data distribution parameters does not prevent those models from catastrophic forgetting. However, BFNs can benefit from known CL strategies such as rehearsal and generative replay. In our future works, we plan to explore further how we can benefit from the sweet combination of Bayesian update, with neural modelling in order to continually adjust parameters of the data distrubution.

References

[1] Alessandro Achille, Tom Eccles, Loic Matthey, Chris Burgess, Nicholas Watters, Alexander Lerchner, and Irina Higgins. Life-long disentangled representation learning with cross-domain latent homologies. *Advances in Neural Information Processing Systems*, 31, 2018.

[2] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis.*Advances in Neural Information Processing Systems*, 34, 2021.

[3] Nan Ding, Xi Chen, Tomer Levinboim, Sebastian Goodman, and Radu Soricut. Bridging the gap between practice and pac-bayes theory in few-shot meta-learning, 2021.

[4] CC0: Public Domain. Flights dataset in 2013. `https://www.kaggle.com/datasets/mahoora00135/flights/data`, 2013.

[5] Evgenii Egorov, Anna Kuzina, and Evgeny Burnaev. Boovae: Boosting approach for continual learning of vae. *Advances in Neural Information Processing Systems*, 34, 2021.

[6] Alex Graves, Rupesh Kumar Srivastava, Timothy Atkinson, and Faustino Gomez. Bayesian flow networks, 2023.

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[8] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation.*Journal of Machine Learning Research*, 23(47):1–33, 2022.

[9] Chin-Wei Huang, Jae Hyun Lim, and Aaron C Courville. A variational perspective on diffusion-based generative models and score matching. *Advances in Neural Information Processing Systems*, 34, 2021.

[10] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings, 2020.

[11] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 21696–21707. Curran Associates, Inc., 2021.

[12] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

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

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[15] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[16] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

[17] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2020.

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

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[20] Hao Wang and Dit-Yan Yeung. Towards bayesian deep learning: A framework and some existing methods. *IEEE Transactions on Knowledge and Data Engineering*, 28(12):3395–3408, 2016.

[21] Hao Wang and Dit-Yan Yeung. A survey on bayesian deep learning. *ACM computing surveys (csur)*, 53(5):1–37, 2020.

[22] Chenshen Wu, Luis Herranz, Xialei Liu, Yaxing Wang, Joost van de Weijer, and Bogdan Raducanu. Memory replay gans: Learning to generate new categories without forgetting. In *NeurIPS*, 2018.

[23] Michał Zając, Kamil Deja, Anna Kuzina, Jakub M. Tomczak, Tomasz Trzciński, Florian Shkurti, and Piotr Miłoś. Exploring continual learning of diffusion models, 2023.

Appendix

6.1 Extension

For discrete or discretised case, we assume that we model the distributions by α parameters, which is the vector of length equal to the number of possible values, i.e. α_i is the probability of generating i^{th} value. For more technicalities, we refer to chapter 5 in the original paper.

In order to obtain closed-form update formulas, under mind conditions we set the accuracy schedule β which is just compound sum of accuracy rates, i.e. $\beta(t) \doteq \int_{t'=0}^{t} \alpha_{t'} dt'$ as in diffusion models. Then, through the series of mathematical derivations, we are able to calculate the overall Bayesian Flow update given the time step as in equation 205 in the original paper.

6.2 Additional results - image generations

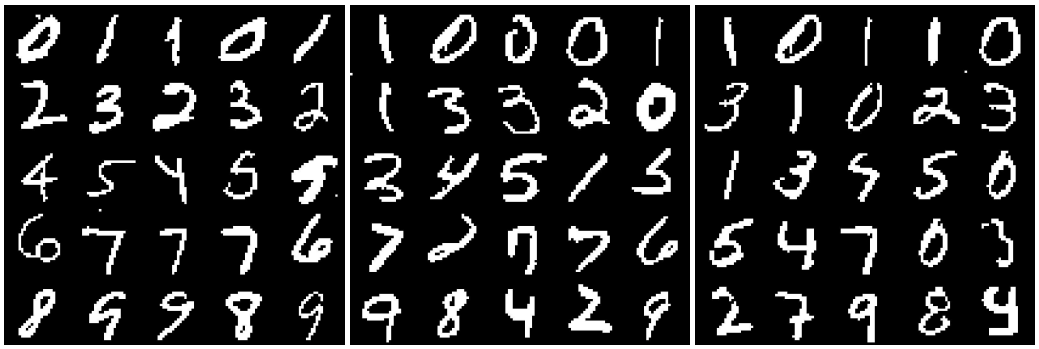

Figure 4: Results of image generations for (a) finetuning; (b) memory-based approach; (c) generative-based approach. Each row corresponds to the consecutive task.

6.2.1 Additional results – regularisation-based approach.

We also notice the quality degradation for regularisation-based approaches. We observed two cases, either the λ coefficient corresponding to the weight of regularisation component is too small and therefore not significant, or there is quality degradation for image generation.

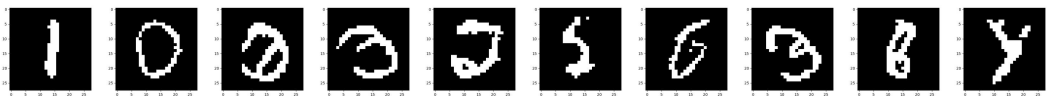

Figure 5: Results of image generation for regularisation-based approach. For each task, two samples were generated.

Experiment details For work on MNIST, we apply standard U-Net, widely applied for diffusion models with 64 model channels, 2 resnet blocks and attention resolution 32,16,8. The size of the model is 6.1M parameters.

For work on tabular data (5 categorical, 9 numerical columns), we use TabTransformer with dimension 32, depth 6 and 8 heads. Overall models has around 600k learnable parameters. There are roughly 320k data points.