Can Expressive Posterior Approximations Improve Variational Continual Learning?

Sayantan Auddy[1] Jakob Hollenstein Matteo Saveriano Antonio Rodríguez-Sánchez Justus Piater

***Abstract*— Mean field variational inference (MFVI) has been successfully used in the past for continual learning. However, Gaussian mean field approximation has been shown to be inferior to more expressive forms of posterior approximation for training latent variable models and single task Bayesian neural networks (BNNs). In this paper, we examine whether expressive posterior approximations obtained with normalizing flows (NF) can result in improved continual learning compared to the mean field approach. Results from our preliminary experiments on the Permuted MNIST benchmark indicate that with longer training durations, over all tasks, BNNs with NF perform marginally better than BNNs with MFVI. Additionally, BNNs with NF are superior to BNNs with MFVI at remembering more recent tasks, while the performance on older tasks is similar between the two methods.**

I. Introduction

Neural networks are trained to extract knowledge from a single dataset at a time to solve a given task. When a network is trained using a different dataset, it can no longer perform the task associated with the previous dataset. This is known as catastrophic forgetting [1]. Methods for continual learning for neural networks aim to minimize the effect of catastrophic forgetting in order to allow networks to learn in an open-ended manner without requiring explicit access to all the datasets that it was trained on in the past. Additional objectives of continual learning are also to use previous knowledge to improve the learning of new tasks, and to enable learning a large number of sequential tasks [2].

The current literature on continual learning using neural networks comprises of a large number of different approaches. While some methods rely on generative modeling of previously seen data and pseudo-rehearsal techniques [3], others rely on regularization schemes to protect the network parameters that are essential for performing well on previous tasks [2], [4].

The problem of continual learning has also been tackled from the perspective of approximate Bayesian inference over the parameters of a network. In Variational Continual Learning (VCL) [5], [6], the posterior distribution over the model parameters that is learned in task i is employed as the prior distribution over the parameters in the next task $i+1$, thereby exploiting the general framework of Bayesian inference for continual learning. VCL [5] uses Mean-Field Variational Inference (MFVI), in which the posterior distribution of the network parameters is approximated by a Gaussian distribution with a diagonal covariance matrix. The errors due to the approximation of the posterior by a diagonal-covariance Gaussian will propagate and accumulate in this setup. Given the very high dimensionality of the space of parameters of a neural network, an approximate Gaussian distribution presumably does not capture all the complexities of the true posterior.

Recent contributions in VI have shown that more expressive posterior distributions obtained using Implicit/Semi-Implicit VI [7]–[11] or Normalizing Flows [12]–[14] can result in better performance. In the context of classification or regression tasks with neural networks, this results in more accurate predictions as well as better uncertainty estimates on out-of-distribution data [14].

In this paper, our main contributions are: 1) extending BNNs with normalizing flows [14] for use in continual learning, and 2) investigating whether more expressive posterior approximations for continual learning have any tangible benefits over MFVI. We present results of the preliminary comparison of the continual learning performance of a Bayesian Neural Network (BNN) using MFVI [15] to the performance of a BNN using normalizing flows [14] on the Permuted MNIST continual learning benchmark. These results show that when the performance over all tasks is considered, a BNN with normalizing flows is marginally better than a BNN with MFVI for longer training durations. However, a BNN with normalizing flows is especially better than a BNN with MFVI at remembering more recent tasks, while a similar level of forgetting occurs for older tasks.

The remainder of the paper is organized as follows: in section II, we briefly describe how MFVI and VI with Normalizing Flows can be used for training BNNs. This is followed by section III, in which we describe how we adapt a BNN with Normalizing Flows [14] for continual learning. The results from our preliminary experiments are presented in section IV. Finally, a discussion of the observed results, concluding remarks, and possible directions for future work are presented in section V.

II. Bayesian Neural Networks

When a conventional neural network is trained using Maximum Likelihood Estimation (MLE) or Maximum a Posteriori (MAP) estimation, we end up with point estimates for each learnable parameter of the network. In contrast, each parameter of a Bayesian Neural Network (BNN) is defined to be a distribution instead of a point estimate. A single trained BNN model thus behaves as an infinite ensemble of networks (in which each sample from the learned distribution of parameters acts as an individual network) [15]. This results

All authors are with the Intelligent and Interactive Systems Group, Department of Computer Science, University of Innsbruck, Technikerstrasse 21A, 6020 Innsbruck, Austria. [1]sayantan.auddy@uibk.ac.at

in additional capabilities such as the avoidance of overfitting by inexpensive model averaging and also the computation of predictive uncertainties.

The goal of Bayesian inference is to find the posterior distribution of the network parameters $\mathbf{w}$ given the observed data $\mathcal{D}$: $p(\mathbf{w}|\mathcal{D})$. However, exact Bayesian inference on the parameters of a neural network is intractable and for practical purposes, we need to rely on approximate inference techniques, among which Variational Inference (VI) is a widely adopted choice [16]. Mean Field VI, which assumes that the approximate posterior is a Gaussian distribution with a diagonal covariance matrix, has been successfully used for training generative latent variable models such as Variational Auto Encoders (VAEs) [17], as well as BNNs [15].

VI approximates the true posterior by $q_\phi(\mathbf{w})$ and aims to minimize the Kullback-Leibler (KL) divergence between the approximate posterior distribution of the weights $q_\phi(\mathbf{w})$ and the true posterior $p(\mathbf{w}|\mathcal{D})$. In MFVI, $q_\phi(\mathbf{w})$ is assumed to be a diagonal Gaussian distribution (parameterized by $\phi = (\mu, \sigma)$) and thus a sample of $\mathbf{w}$ can be generated by sampling a unit Gaussian $\epsilon \sim \mathcal{N}(0, I)$, and shifting it by a mean μ and scaling it by a standard deviation σ to obtain $\mathbf{w} = \mu + \sigma \odot \epsilon$, where $\odot$ denotes the Hadamard product. This is the reparameterization trick [15] that allows us to use standard backpropagation and gradient ascent for optimizing (1) .

Since we do not have access to the true posterior, instead of minimizing the variational parameters ϕ with respect to $\mathrm{KL}(q_\phi(\mathbf{w})||p(\mathbf{w}|\mathcal{D}))$, we maximize the Evidence Lower Bound (ELBO) given by

$$\mathcal{L}(\phi) = \mathbb{E}_{q_\phi}\big[\log p(\mathcal{D}|\mathbf{w})\big] - \mathrm{KL}\big(q_\phi(\mathbf{w})||p(\mathbf{w})\big) \tag{1}$$

$$= \mathbb{E}_{q_\phi}\big[\log p(\mathcal{D}|\mathbf{w}) - \log q_\phi(\mathbf{w}) + \log p(\mathbf{w})\big] \tag{2}$$

where $p(\mathbf{w})$ is the prior over the parameters. The first part of (1) is data dependent and encourages maximum likelihood estimation. The second term encourages the posterior $q_\phi(\mathbf{w})$ to be as similar as possible to the prior $p(\mathbf{w})$, and for continual learning, this regularization term is also exploited for retaining information from the previous prior.

Although, using MFVI gives us a way of training BNNs in an efficient way, the diagonal Gaussian approximation for the approximate posterior results in an underestimation of the variance of the posterior distribution which can, in turn, result in poor predictive uncertainties [13]. More complex posterior approximations can be created using mixture models, but this would significantly increase the number of learnable parameters. One way of creating richer posterior approximations is to use Normalizing Flows (NF) [12], which transforms a simple probability distribution (such as a Gaussian) into a complex distribution through a sequence of bijective mappings. Although it is relatively straightforward to use NF with latent models such as VAEs [13], using them for BNNs is more complicated because the dimensionality of the latent variable in a VAE (output of the encoder) is much smaller than in BNNs where the latent variables are the network parameters.

The authors of [14] get around this problem by defining the approximate posterior as a mixture of simpler distributions: first, a vector of random variables $\mathbf{z}_0$ is drawn from a fully factored Gaussian distribution (diagonal covariance matrix). $\mathbf{z}_0$ is then transformed to a sample $\mathbf{z}$ from a more complex distribution by passing it through an NF. The approximate posterior can then be parameterized by

$$\mathbf{z} \sim q_\phi(\mathbf{z}); \qquad \mathbf{w} \sim q_\phi(\mathbf{w}|\mathbf{z}), \tag{3}$$

where $q_\phi(\mathbf{z})$ represents the mixing density [14]. The approximate posterior distribution thus becomes $q_\phi(\mathbf{w}) = \int q_\phi(\mathbf{w}|\mathbf{z}) q_\phi(\mathbf{z}) \mathrm{d}\mathbf{z}$. For a fully connected layer of a BNN, the conditional distribution for $\mathbf{w}$ in (3) is a fully factored Gaussian defined by

$$q_\phi(\mathbf{w}|\mathbf{z}) = \prod_{i=1}^{D_{in}} \prod_{j=1}^{D_{out}} \mathcal{N}(z_i \mu_{ij}, \sigma_{ij}^2) \tag{4}$$

where D_{in} and D_{out} are the dimensions of the input and output of the layer respectively, z_i denotes the i^{th} component of the NF output, and μ_{ij} and σ_{ij} are the trainable mean and standard deviation parameters of the BNN. Due to the form of $q_\phi(\mathbf{w})$, the calculation of the entropy term $-\mathbb{E}_{q_\phi}\big[\log q_\phi(\mathbf{w})\big]$ in (2) cannot be obtained in a closed form. This problem is circumvented by further lower-bounding the entropy using an auxiliary distribution $r_\theta(\mathbf{z}|\mathbf{w})$ (which approximates $q_\phi(\mathbf{z}|\mathbf{w}) = \frac{q_\phi(\mathbf{w}|\mathbf{z}) q_\phi(\mathbf{z})}{q_\phi(\mathbf{w})}$). This results in rewriting the ELBO from (2) as

$$\begin{aligned}
\mathcal{L}(\phi, \theta) &= \mathbb{E}_{q_\phi}\big[\log p(\mathcal{D}|\mathbf{w}, \mathbf{z}) - \log q_\phi(\mathbf{w}) + \log p(\mathbf{w})\big] \\
&= \mathbb{E}_{q_\phi}\big[\log p(\mathcal{D}|\mathbf{w}, \mathbf{z}) - \log\big(\frac{q_\phi(\mathbf{w}|\mathbf{z}) q_\phi(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{w})}\big) + \log p(\mathbf{w})\big] \\
&\approx \mathbb{E}_{q_\phi}\big[\log p(\mathcal{D}|\mathbf{w}, \mathbf{z}) - \log\big(\frac{q_\phi(\mathbf{w}|\mathbf{z}) q_\phi(\mathbf{z})}{r_\theta(\mathbf{z}|\mathbf{w})}\big) + \log p(\mathbf{w})\big] \\
&= \mathbb{E}_{q_\phi}\big[\log p(\mathcal{D}|\mathbf{w}, \mathbf{z}) - \log q_\phi(\mathbf{w}|\mathbf{z}) - \log q_\phi(\mathbf{z}) \\
&\quad + \log r_\theta(\mathbf{z}|\mathbf{w}) + \log p(\mathbf{w})\big] \\
&= \mathbb{E}_{q_\phi}\big[\log p(\mathcal{D}|\mathbf{w}, \mathbf{z}) - \log q_\phi(\mathbf{w}|\mathbf{z}) + \log p(\mathbf{w}) \\
&\quad + \log r_\theta(\mathbf{z}|\mathbf{w}) - \log q_\phi(\mathbf{z})\big] \\
&= \mathbb{E}_{q_\phi}\Big[\log p(\mathcal{D}|\mathbf{w}, \mathbf{z}) - \mathbb{E}_{q_\phi}\big[\log \frac{q_\phi(\mathbf{w}|\mathbf{z})}{p(\mathbf{w})}\big] \\
&\quad + \log r_\theta(\mathbf{z}|\mathbf{w}) - \log q_\phi(\mathbf{z})\Big] \\
&= \mathbb{E}_{q_\phi}\Big[\log p(\mathcal{D}|\mathbf{w}, \mathbf{z}) - \mathrm{KL}\big(q_\phi(\mathbf{w}|\mathbf{z})||p(\mathbf{w})\big) \\
&\quad + \log r_\theta(\mathbf{z}|\mathbf{w}) - \log q_\phi(\mathbf{z})\Big]
\end{aligned} \tag{5}$$

In the last line of (5), the first term is the log likelihood of the observed data. For continual learning, we are interested in the KL term which involves the prior $p(\mathbf{w})$. The use of this term will be elaborated on in the next section. For an expanded form of the remaining terms, the reader is referred to [14].

III. Method

Consider a sequence of datasets $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_N$ which need to be used for training a BNN in a continual manner. Thus, dataset $\mathcal{D}_1$ corresponds to task 1, $\mathcal{D}_2$ corresponds to task 2, and so on. For task 1, we assume that the prior

over the network parameters in (5) is a standard normal distribution $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. Once we have trained the BNN following the process of [14], for task 1 we obtain the approximate conditional posterior $q_\phi(\mathbf{w}|\mathbf{z})$. For learning task 2, this posterior can be used as the prior in (5). In general the old posterior can be used as the prior for the next task for learning each of the successive tasks.

Since both the posterior and prior in the KL term of (5) can be expressed as Normal distributions, it is convenient to obtain a simplified expression for the KL term. It can be shown that the the KL divergence between two Normal distributions $\mathcal{N}(\mu_1, \sigma_1)$ and $\mathcal{N}(\mu_2, \sigma_2)$ is given by

$$\mathrm{KL}\big(\mathcal{N}(\mu_1, \sigma_1)||\mathcal{N}(\mu_2, \sigma_2)\big) = \frac{1}{2}\Big(-\log\frac{\sigma_1^2}{\sigma_2^2} + \big(\frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{\sigma_2^2}\big) - 1\Big) \tag{6}$$

For task 1, we can consider the first Normal distribution in (6) to be as given in (4), and the second normal distribution is $\mathcal{N}(0, 1)$. Since we are using fully factored distributions, it can be shown that for a layer of a BNN, (6) becomes

$$\begin{aligned}\mathrm{KL}\big(q_\phi(\mathbf{w}|\mathbf{z})||p(\mathbf{w})\big) &= \sum_{i,j} \mathrm{KL}\big(\mathcal{N}(z_i\mu_{i,j}, \sigma_{i,j})||\mathcal{N}(0, 1)\big) \\ &= \sum_{i,j} \frac{1}{2}\Big(-\log\sigma_{i,j}^2 + \sigma_{i,j}^2 + z_i\mu_{i,j}^2 - 1\Big)\end{aligned} \tag{7}$$

where i, j iterate over the input and output dimensions of the layer. Equation (7) is the form of the KL term in (5) used in [14] for learning a single task.

For continual learning, for task 1 we also use (7). For the successive tasks we rewrite (7) as

$$\begin{aligned}&\mathrm{KL}\big(q_\phi(\mathbf{w}|\mathbf{z})||q_\phi^{old}(\mathbf{w}|\mathbf{z})\big) \\ &= \sum_{i,j} \mathrm{KL}\big(\mathcal{N}(z_i\mu_{i,j}, \sigma_{i,j})||\mathcal{N}(z_i^{old}\mu_{i,j}^{old}, \sigma_{i,j}^{old})\big) \\ &= \sum_{i,j} \frac{1}{2}\Big(-\log\frac{\sigma_{i,j}^2}{\sigma_{i,j}^{old2}} + \big(\frac{\sigma_{i,j}^2 + (z_i\mu_{i,j} - z_i^{old}\mu_{i,j}^{old})^2}{\sigma_{i,j}^{old2}}\big) - 1\Big)\end{aligned} \tag{8}$$

and use this form as the KL term in (5). Here, the superscript old denotes the frozen parameters belonging to the posterior distribution learned during the previous task. The remaining terms in (5), which do not involve the prior, are used as is.

It is also easy to see that if we ignore the mixing density specified in (3) (by considering $\mathbf{z}$ to be always equal to $\mathbb{1}$), we can ignore the $\log r_\theta(\mathbf{z}|\mathbf{w})$ and $\log q_\phi(\mathbf{z})$ terms in (5), which reduces (5) to (1). Thus, by doing so, we get the ELBO for an MFVI BNN. This allows us to use the same code base for both types of BNNs (MFVI and NF).

IV. Experiments

In this section, we present the results of our preliminary experiments for comparing the continual learning performance of an MFVI BNN against that of an NF BNN. We focus on the core aspect of variational continual learning and so we do not use any data summarization techniques such as the coreset method [18] used in [5] since such methods can be used to augment any kind of continual learning method. Additionally, we do not employ any kind of model expansion during our experiments.

A. Experimental Setup

We use the Permuted MNIST (PMNIST) continual learning benchmark, which like MNIST is a ten-digit classification problem. A task in PMNIST consists of different permutations of the pixels in the images. The length of the task sequence is determined by the number of different permutations that are used. We use 10 tasks, in which the first task is the same as the original MNIST and for the rest of the 9 tasks, different permutations are used. For each task, we use a training set of 50000 images, a validation set of 10000 images, and a test set of 10000 images. We use a batch size of 100 during training. Although PMNIST is not considered a very challenging continual learning problem [19], its wide use and simplicity make it a suitable choice for initial experiments.

For our continual learning experiments, we extended the code[1] provided by the authors of [14]. For both experiments, NF- and MFVI-BNNs, we used the same code base in which only the variational distributions (NF, MFVI) are changed, by the switch of a single flag.

For our experiments we use a fully connected BNN with 3 hidden layers, each containing 200 units. We use ReLU activations and the Adam optimizer with a learning rate of 10^{-3}. We compare the performance of 2 types of BNNs (MFVI vs NF) and train each network for 10, 50, and 100 epochs for each PMNIST task. We repeat every experiment for 3 different seeds thereby resulting in a total of 18 different sets of results. We group the results by epochs and network types in the plots presented in this section. For evaluation, we compare the test accuracy of the network on the task it has just been trained on, as well as all the test accuracy for all the tasks seen in the past. We not only compare the average of these test accuracies but also look at how the accuracy of each task evolves over time as newer tasks are learned. Finally, we also compute some overall continual learning metrics based on [20].

B. Results

Fig. 1 shows the average test accuracies for all the past tasks and the current task after the network is trained for the current task. This means for example that when the network is trained for task 5, we compute its average test accuracy for tasks 0 to 5. The results shown are averaged for the 3 different seeds used. Based on these results, we can say that both the MFVI and NF BNNs perform best when they are trained for 50 epochs per task, and here both types of BNNs perform similarly well. However, it is interesting to note that for 10 epochs/task, MFVI performs better than NF, but for 100 epochs/task, the NF BNN performs better. This switch in performance may be due to the additional parameters in

[1] https://github.com/AMLab-Amsterdam/MNF_VBNN

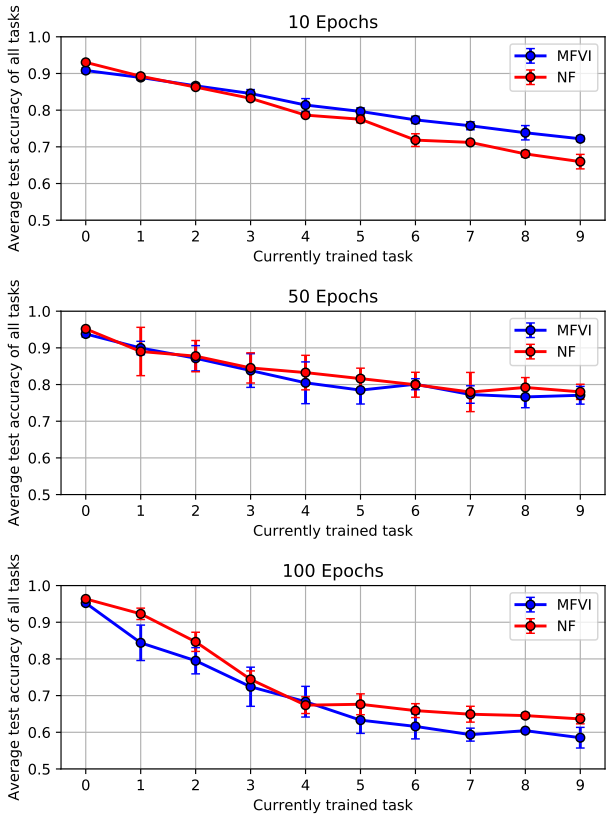

Fig. 1. Mean test accuracy for all Permuted MNIST tasks seen till the current task. For each task the network is trained for (top) 10 epochs, (middle) 50 epochs, (bottom) 100 epochs.

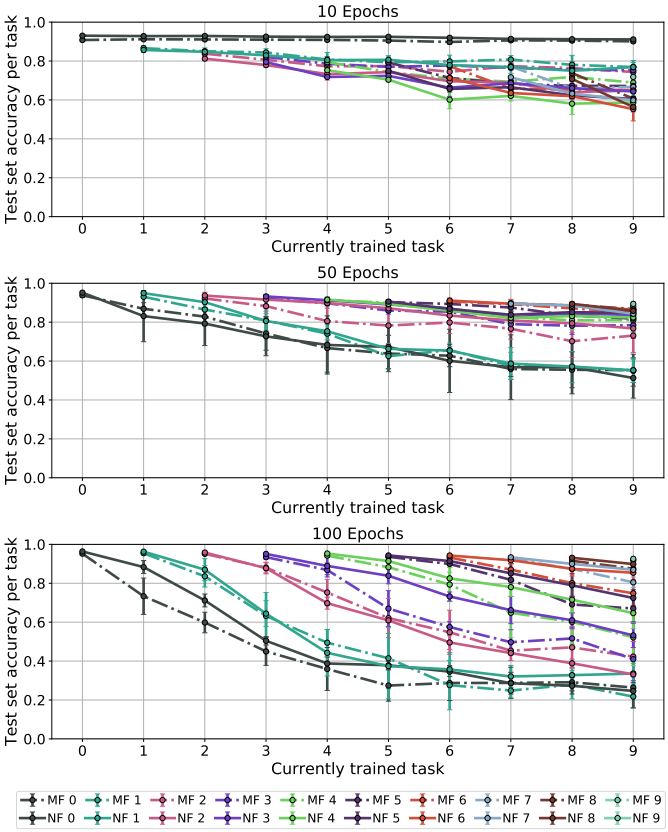

Fig. 2. Test accuracy for individual PMNIST tasks. Solid lines and dashed lines represent NF and MFVI (abbreviated as MF) networks respectively. Each task from 0 to 9 has a distinct color. Average results for 3 different seeds are shown. For each task the network is trained for (top) 10 epochs, (middle) 50 epochs, (bottom) 100 epochs.

the NF BNN, which may require more training iterations to be optimized. With an increase in training epochs, the variability of the results for the different seeds also increases, as indicated by the error bars (1 standard deviation) in the plots. However, Fig. 1 does not give us a complete picture of the continual learning performance of a network. From a network's average test accuracy we cannot get any information about the relative performance of the network on the older and current tasks.

To get a better understanding of the relative performance on old and new tasks, in Fig. 2 we also track the performance on each task individually as the newer tasks are presented to the network. As before, average results for 3 different seeds are shown.

Fig 2 (top) shows the results for BNNs trained for 10 epochs for each PMNIST task. Here it is evident that for both kinds of networks, task 0 is remembered exceedingly well. The performance of both the networks for task 0 remains constant around 0.96 (MF 0 and NF 0 in the plot) even after learning all 10 tasks. The performance for the remaining tasks is considerably worse, but in general, the performance for older tasks is better than newer ones. This means that for a relatively small number of training epochs per task, after learning the first task (for which a unit normal prior is used), the data-dependent part of (1) (for MFVI) or (5) (for NF) has a lesser influence than the KL part. There is, however, no clear distinction between MFVI and NF.

The trend of remembering older tasks better seems to change when the number of training epochs per task is increased to 50 and 100 (Fig. 2 middle and bottom respectively). In Fig. 2 (middle), the first 2 tasks are the worst remembered ones in the end and generally newer tasks are remembered better. For the 50 epochs experiment, there is no significant difference between MFVI (dashed lines) and NF (solid lines), apart from task 2 for which NF performs slightly better. The results for the different seeds show more variability than in the previous case.

As the number of training epochs is further increased to 100 in Fig. 2 (bottom), the difference between older and newer tasks opens up much more for both MFVI and NF. For tasks 0-3, there is a significant drop in performance. However, it is noticeable that for tasks 3-9, NF performs significantly better than MFVI, which contributes to the overall better performance of NF shown in Fig. 1 (bottom).

Continual Learning Metrics: To enable easier comparison of the overall performance of the two kinds of networks, we also compute some continual learning metrics suggested in [20]. Specifically, we calculate (i) accuracy (ACC) – the average test accuracy for the current task and the previously

seen tasks (computed for all current tasks), (ii) remembrance (REM) – measures how well previous tasks are remembered as a new current task is learned, (iii) positive backward transfer (BWT^+) – measures how the performance on previous tasks is improved after learning the current task, (iv) forward transfer (FWT) – measures the performance on future unseen tasks after learning a current task, (v) model size efficiency (MS) – measures how the model size grows with the number of tasks, and (vi) sample storage efficiency (SSS) – measures how effectively data samples from different tasks are cached. Finally, a continual learning score (CL_{score}), the average of (i)-(vi) is computed. Each metric lies in the range [0.0, 1.0] (1.0 is the best). The reader is referred to [20] for a detailed discussion of these metrics, the motivation for their use, and their mathematical expressions.

Table I shows the values of the continual learning metrics described above, where each metric (cols 3-8) is the average value obtained for 3 different seeds. Since neither MFVI nor NF uses any kind of model expansion or data caching mechanism, the values of MS and SSS are always 1.0. Overall, MFVI achieves the best result for CL_{score}, but the difference with NF is not significant. For a benchmark such as PMNIST, in which the pixels are permuted for the different tasks, the performance of a network on individual tasks is expected to be decoupled (learning a current task 5 does not improve the performance on task 4 or task 6). This is reflected in the values for BWT^+ and FWT in Table I. For the metrics which are comparatively more relevant for PMNIST (ACC and REM), it can be seen that NF performs marginally better than MFVI, for 50 and 100 epochs. Fig. 3 shows a graphical representation of the same results. Similar to Fig. 1, the results in Table I and Fig. 3 show that there is not much difference in the performance of the two VI methods. Fig. 2 provides a clearer view of the difference in the performance of the two methods, especially how the individual performances on the current and previous tasks affect the overall performance.

TABLE I

CONTINUAL LEARNING METRICS FOR MFVI (ABBREVIATED AS MF) AND NF NETWORKS FOR PMNIST FOR 10, 50 AND 100 EPOCHS (EP). THE VALUE OF EACH METRIC IS AN AVERAGE OVER 3 RUNS.

EP	VI	ACC	REM	BWT^+	FWT	MS	SSS	CL_{score}
10	MF	0.779	**0.937**	0.0	**0.113**	1.0	1.0	**0.638**
10	NF	0.739	0.915	0.0	0.108	1.0	1.0	0.627
50	MF	0.797	0.851	0.0	0.101	1.0	1.0	0.625
50	NF	**0.810**	0.855	0.0	0.100	1.0	1.0	0.628
100	MF	0.645	0.636	0.0	0.105	1.0	1.0	0.565
100	NF	0.677	0.663	0.0	0.105	1.0	1.0	0.574

Note that the results we obtained on PMNIST for MFVI do not reach the state-of-the-art [5], presumably because we do not use the same architecture and hyperparameters but rather keep the parameterization and architecture of [14] which is not evaluated with PMNIST.

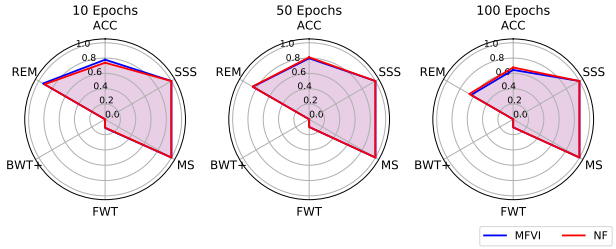

Fig. 3. Continual learning metrics [20] for MFVI and NF.

V. CONCLUSIONS

Although there is no clear advantage between the performance of MFVI and NF in the results presented in Section IV-B, overall, we note that with more training epochs, NF performs better than MFVI at remembering more recent tasks. However, with more epochs at some point, there is also a degradation of the overall continual learning performance with an increase of catastrophic forgetting of older tasks. This brings us to some critical questions in the context of continual learning: what causes this behavior? For future continual learning applications outside of research experiments, how should we choose the ideal training duration (epochs) for each task? Should this duration be constant or dynamic?

For a model with limited capacity (without any rehearsal or data caching mechanisms), such as the ones we used in our experiments, catastrophic forgetting is inevitable at some point with the continued increase in tasks. In such a scenario it is perhaps desirable to gracefully forget the older tasks while remembering the newer ones. From the results in Fig. 2, BNNs with NF give a slight indication of this property, but only a more thorough investigation can shed more light on this behavior. We leave this investigation for future work. It will also be interesting to compare the performance of BNNs trained with other alternative VI techniques such as Implicit and Semi-Implicit VI [7]–[10] and to carry out this comparison using more challenging continual learning benchmarks such as CORe50 [21].

REFERENCES

[1] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual Lifelong Learning with Neural Networks: A Review. *Neural Networks*, 113:54–71, May 2019.

[2] Jonathan Schwarz, Jelena Luketina, Wojciech M. Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & Compress: A scalable framework for continual learning. *arXiv:1805.06370 [cs, stat]*, July 2018.

[3] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual Learning with Deep Generative Replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017.

[4] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

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

[6] Siddharth Swaroop, Cuong V. Nguyen, Thang D. Bui, and Richard E. Turner. Improving and Understanding Variational Continual Learning. *arXiv:1905.02099 [cs, stat]*, 2019.

[7] Ferenc Huszár. Variational Inference using Implicit Distributions. *arXiv:1702.08235 [cs, stat]*, February 2017.

[8] Jiaxin Shi, Shengyang Sun, and Jun Zhu. Kernel Implicit Variational Inference. In *International Conference on Learning Representations*, 2018.

[9] Nick Pawlowski, Andrew Brock, Matthew C. H. Lee, Martin Rajchl, and Ben Glocker. Implicit Weight Uncertainty in Neural Networks. *arXiv:1711.01297 [cs, stat]*, May 2018.

[10] Mingzhang Yin and Mingyuan Zhou. Semi-Implicit Variational Inference. In *International Conference on Machine Learning*, pages 5660–5669, 2018.

[11] Michalis K Titsias and Francisco Ruiz. Unbiased Implicit Variational Inference. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 167–176, 2019.

[12] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *arXiv:1605.08803 [cs, stat]*, February 2017.

[13] Danilo Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015.

[14] Christos Louizos and Max Welling. Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2218–2227, 2017.

[15] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pages 1613–1622, 2015.

[16] Ethan Goan and Clinton Fookes. *Bayesian Neural Networks: An Introduction and Survey*, pages 45–87. Springer International Publishing, Cham, 2020.

[17] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[18] Jonathan H. Huggins, Trevor Campbell, and Tamara Broderick. Coresets for scalable bayesian logistic regression, 2016.

[19] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating Continual Learning Scenarios: A Categorization and Case for Strong Baselines. *arXiv:1810.12488 [cs]*, January 2019.

[20] Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget, there is more than forgetting: New metrics for Continual Learning. *arXiv:1810.13166 [cs]*, October 2018.

[21] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 17–26. PMLR, 13–15 Nov 2017.