Reducing catastrophic forgetting in neural networks via Gaussian mixture approximation

Hoang Phan*, Anh Phan Tuan*, Son Nguyen*, Ngo Van Linh, and Khoat Than+

Hanoi University of Science and Technology
No 1, Dai Co Viet road, Hanoi, Vietnam
{phanviethoang1512, phantuananhkt2204k60, sonnguyenkstn}@gmail.com
{linhnv, khoattq}@soict.hust.edu.vn

Abstract. Our paper studies the continual learning (CL) problems in which data comes in sequence and the trained models are expected to be capable of utilizing existing knowledge to solve new tasks without losing performance on previous ones. This also poses a central difficulty in the field of CL, termed as Catastrophic Forgetting (CF). In an attempt to address this problem, Bayesian methods provide a powerful principle, focusing on the inference scheme to estimate the importance of weights. Variational inference (VI), one of the most widely used methods within this vein, approximates the intractable posterior by a factorized distribution, thus offering computational efficiency. Notwithstanding many state-of-the-art performances in practice, this simple assumption about the posterior distribution typically limits the model capacity to some extent. In this paper, we introduce a novel approach to mitigate forgetting in the Bayesian approach via enriching the posterior distribution with mixture models, which intuitively promotes neural networks to acquire knowledge from multiple tasks at a time. Moreover, in order to reduce the model's complexity growth when the number of components increases, we propose a solution that conducts low-rank decomposition on the variance of each component based on neural matrix factorization. Extensive experiments show that our method yields significant improvements compared to prior works on different benchmarks.

Keywords: Continual Learning · Catastrophic Forgetting · Gaussian Mixture.

1 Introduction

Despite the fact that artificial intelligent agents have surpassed human beings at many specified tasks, their abilities are still far behind humans in terms of performing well on wide-ranged, disjointed problems. However, this could not be completely comparable, since current systems often fail to retain the previous

* Equal contribution
+ Corresponding author

knowledge while learning new tasks [13], whilst humans have a great capability of learning in a continuous manner: accumulating knowledge and avoiding forgetting. Continual learning, therefore, has emerged in recent years as a learning regime, allowing deep learning models to learn sequential tasks efficiently.

A branch of preliminary work in CL [14, 1, 5, 11] capitalizes on the idea of injecting uncertainty into the neural network's parameters, which is known as Bayesian Neural Networks (BNNs) [2]. Unlike the original NN, BNN considers its parameters as random variables drawn from a given prior distribution. Based on this framework, Variational Continual Learning [14], or simply VCL, was one of the first proposals to formulate the continual learning as an approximation Bayesian inference problem in which the combination of online variational inference with an efficient sampling method soon achieved significant results. Uncertainty-based Continual Learning [1] then re-interpreted the KL-divergence term in VCL, defined the concept of uncertainty for hidden nodes and modified the KL-divergence term following the principle: the more uncertain a parameter is, the more likely it will be changed in subsequent tasks. Another advantage this method brings is that the number of parameters stored is less than other earlier works, which placed the importance on weights [11]. By contrast, Variational Generative Replay (VGR) [6] has shown that likelihood-focused methods - those that estimate the likelihood of the preceding tasks with synthetic data rather than directly employ the model's posteriors as prior for successive tasks - can outperform prior-focused methods. In a recent study, Uncertainty-guided Continual Bayesian Neural Networks (UCB) [5] defined a metric for measuring the weight's importance and thereby developed an appropriate training strategy.

In spite of the variational inference's popularity and the advantages it brings, the inference quality is still heavily affected by the parametric family of the posterior approximation distribution. Both recent methods VCL and UCB utilized a simple diagonal covariance Gaussian for posterior approximation, which is likely not flexible enough to match the true posterior, especially in the continual learning context. That being said, the nature of the data stream exhibits a large inconsistency. Because each of them is sampled from a different distribution, the problem of CF, that existing approaches suffer from, could be explained as the well-known mode-seeking in statistics, where the unimodal distribution is unable to capture information in multiple modes. We use an example in Fig. 1 as a simple illustration for this conclusion, a trimodal Gaussian mixture (blue curve) is approximated by a single Gaussian distribution and another mixture of two components. While both of these approximation distributions cover the middle mode, the RHS is covered by the mixture only. Since the overall range and shape of the mixture is much closer to the true distribution, it is obviously a better approximation to the target trimodal mixture.

Meanwhile, choosing the more expressive (i.e. richer representation capacity) variational family could help obtain better inference. With this intuition in mind, recent studies that go beyond mean-field variational inference are: normalizing flows [16], auxiliary variables [12] and mixture models [7]. Firstly, normalizing flows is a powerful framework that allows simple probabilistic density functions

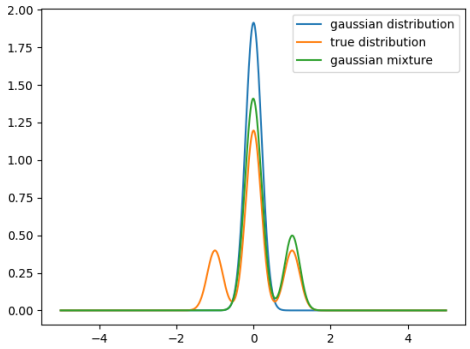


Fig. 1: The approximation abilities of Gaussian distribution and Mixture model. Given the true distribution that is a mixture of 3 Gaussian distributions $0.2\mathcal{N}(-1, 0.2) + 0.6\mathcal{N}(0, 0.2) + 0.2\mathcal{N}(1, 0.2)$, KL-divergence is used to find the approximation of the true distribution. The learned Gaussian distribution is $\mathcal{N}(0, 0.208)$ while the learned mixture of 2 Gaussian distributions is $0.754\mathcal{N}(-0.007, 0.214) + 0.246\mathcal{N}(1.003, 0.197)$.

(PDFs) to be iteratively transformed into the target PDFs via a chain of invertible mappings. Secondly, auxiliary variables are included in the posterior in order to augment itself into more expressive and structured distributions. Finally, the mixture model is deemed to have the ability to approximate any given distribution with arbitrary closeness. However, all the directions seem to be impractical to apply to BNNs in the continual learning context, since their need is to add a huge number of extra parameters. In this work, we focus on an efficient solution to can exploit Gaussian mixture approximations in CL.

The main contributions of our paper could be depicted as follows: first, we present a novel inference method exploiting the Gaussian Mixture as a posterior approximation distribution and an information-theoretic view on how this could handle the CF. Accordingly, an efficient learning algorithm is also proposed, combining the Gumbel softmax reparameterization trick and the closed-form upper bound of KL diverge between two mixture models. Moreover, we employ a parameter reduction technique using Neural matrix factorization [4], which offers computational complexity benefits required for training. Finally, our inference process is experimentally proved to be widely incorporated into existing Bayesian-based learning methods for CL.

2 Backgrounds

2.1 Bayesian Inference

Consider the Bayesian Inference in the supervised learning setting, given the dataset $\mathcal{D} = \{x_i, y_i\}_i^n$ and a BNN parameterized by θ following the prior dis-

tribution $p(\theta)$. Typically, the main goal of Bayesian inference is to derive the posterior distribution over the weights $p(\theta|\mathcal{D})$, which is in an intractable form (i.e. involving integrals). The variational inference (VI) provides an efficient solution in which the true posterior is approximated to a variational distribution $q(\theta|\lambda)$ via minimizing the Kullback–Leibler divergence between these two distributions $\mathrm{KL}(p(\theta|\mathcal{D})||q(\theta|\lambda))$. In the most popular form, both of the prior and variational distributions are assumed to be Gaussian. In short, this optimization is equivalent to maximizing the Evidence Lower Bound (ELBO) w.r.t variational parameter λ:

$$\mathcal{L}(\theta) = \underbrace{E_{q(\theta|\lambda)} \log p(\mathcal{D}|\theta)}_{\text{expected-log likelihood}} - \underbrace{\mathrm{KL}(q(\theta|\lambda)||p(\theta))}_{\text{regularizer}} \tag{1}$$

The above objective function now can be optimized with the Stochastic Gradient Variational Bayes (SGVB, [9]) estimator. More specifically, this process includes two main steps: Reparameterization Trick and Monte Carlo sampling [9].

2.2 Bayesian approach in Continual Learning

Variational Continual Learning [14] used the online Bayesian update following the Bayes rule, and the posterior after observing previous tasks would be the prior of the next one. The sequential tasks are denoted as $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_T\}$. Then, the posterior distributions are recursively computed as:

$$p(\theta|\mathcal{D}_{1:t}) = \frac{p(\theta|\mathcal{D}_{1:t-1})\, p(\mathcal{D}_t)}{p(\mathcal{D}_t|\mathcal{D}_{1:t-1})} \quad (0 < t \leq T) \tag{2}$$

With $p(\theta)$ as the prior distribution placed on θ. Due to the intractable property, $q_t(\theta) \approx p(\theta|\mathcal{D}_{1:t})\ \forall\, t$ is the t^{th} task's posterior approximation. Specifically, both the prior q_0 and posterior $q_i\,(i > 0)$ are chosen to be multivariate diagonal Gaussian distributions to simplify the computation. The VCL's objective function on t^{th} step is:

$$\mathcal{L}_{\text{VCL}}(\theta) = E_{q_t(\theta)} \log(p(\mathcal{D}_t|\theta)) - \mathrm{KL}(q_t(\theta)|q_{t-1}(\theta))$$

In a major advance in regularizing the change of the parameters, UCB [5] estimates the importance of each weight by the multiplicative inverse of its standard deviation $\Omega = \frac{1}{\sigma}$. According to this notion of importance, they controlled the parameter-wise learning rate update at each step as:

$$\Omega_\mu \leftarrow \frac{1}{\sigma} \Rightarrow \alpha_\mu \leftarrow \frac{\alpha_\mu}{\Omega_\mu}$$
$$\Omega_\sigma \leftarrow 1 \Rightarrow \alpha_\sigma \leftarrow \frac{\alpha_\sigma}{\Omega_\sigma}$$

The learning rate scheduler in UCB aimed to lessen the substantial shifting in important parameters, placed on their inherent uncertainty. This also provided a memory-beneficial consequence, since it neither accesses the past data nor stores the quantities associated with previous tasks.

2.3 Gumbel softmax and categorical reparameterization

Normally, training a neural network often involves backpropagation through a chain of continuous-valued and differentiable functions. Even so, the discrete random variables used in stochastic neural networks to represent distributions sometimes yield a more meaningful and interpretable representation. In this section, we briefly summarize some concepts behind the idea of smoothly relaxing discrete distributions with Gumbel-Softmax and the way of training these models with reparameterization trick (path-derivative) gradients.

Gumbel-Softmax trick [8]: Let α be an n-dimensional vector on simplex $\Delta^{n-1} = \{(x_1, x_2, ...x_n) |\ \ x_i \in (0,1),\ \sum_{i=1}^{n} x_i = 1\}$ and $g = \{g_1, g_2, ...g_n\}$ with g_i are i.i.d drawn from the Gumbel distribution $G(0,1)$. Clearly, sampling from the multinomial distribution with probability vector given by α can be written as $y = \text{soft\_max}(\frac{\log \alpha + g}{\tau})$ where $\tau > 0$ is the temperature parameter.

3 Gaussian Mixture Approximation in Bayesian Inference for Continual Learning

In this section, we first present our proposal that exploits Gaussian mixture approximation in continual learning. Then we introduce a solution to reduce the number of parameters of the Gaussian mixture.

3.1 Proposed Method

A proper choice of the posterior approximation distribution theoretically expands the searching space for the true posterior. Especially in continual learning settings, neural networks must be learned on data from several tasks, a typically used unimodal Gaussian distribution is not rich and expressive enough to approximate the true posterior of weights. Accordingly, a Gaussian mixture approximation is more suitable to capture the multi-modality in data modeling. We conducted a simple experiment to show how GM can be a better approximation to the true distribution than a single Gaussian distribution (Fig. 1). This result provides convincing evidence to apply Gaussian mixture approximation in learning multiple tasks (i.e. scenario of continual learning).

For continual learning, for each task, we approximate the true posterior of weights to a Gaussian mixture instead of using unimodal Gaussian distribution as in existing studies [14, 1, 5]. The approximate distribution is presented as follows:

$$q(\theta|\lambda) = \sum_{i=1}^{K} \pi_i \mathcal{N}(\mu_i, \sigma_i) \ \ (\pi \in \Delta^{K-1}) \tag{3}$$

Note that the posterior learned from the previous task is often used as the prior in the current task [14, 1]. Therefore, both the prior and variational distributions are Gaussian mixtures. They are plugged into Equation 1. However, optimizing

the evidence lower bound (Equation 1) poses two main challenges. On the one hand, KL-divergence between two Gaussian mixtures (the regularization term) does not have a closed-form formula. On the other hand, sampling from such distributions typically involves categorical variables (the expected-log likelihood). The lower bound in this case is thus difficult to optimize.

Algorithm 1: Reparameterization trick for Gaussian mixture

Input: Prior distribution: $p(\theta)$ and posterior: $q(\theta|\lambda) = \sum_{i=1}^{K} \pi_i \mathcal{N}(\mu_i, \sigma_i)$
Number of samples: N; temperature : τ

Output: Estimation of the log-likelihood $E_{q(\theta|\lambda)} \log p(\mathcal{D}, \theta)$

Function `Log-likelihoodEstimation`($p(\theta)$, $q(\theta|\lambda)$, N, τ)**:**

```
for n ← 1 to N do
    for k ← 1 to K do
        u_k ← μ_k + σ_k ⊙ ε_k where ε_k ~ N(0,1)
        g_k ~ Gumbel(0,1)
    end
    y = (y_1, y_2, ... y_K) ← soft_max((log(π)+g)/τ)   //   Gumbel softmax trick
    h_n ← Σ_{k=1}^{K} u_k y_k
end
return (1/N) Σ_{n=1}^{N} log p(h_n);
```

In terms of the first difficulty, the Kullback–Leibler divergence, fortunately, has a closed-form upper bound [3]. Rather than directly maximizing the ELBO, we substitute the KL divergence in Equation 1 with its upper bound. Theorem 1 presents the `Upperbound_KL`.

Theorem 1. *Consider 2 mixtures* $f(x) = \sum_{i=1}^{K} \pi_i^a \mathcal{N}(\mu_i^a, \sigma_i^a)$ *and* $g(x) = \sum_{i=1}^{K} \pi_i^b \mathcal{N}(\mu_i^b, \sigma_i^b)(\pi_i^a, \pi_i^b \in \Delta^{K-1})$*, we have the below inequality:*

$$\mathrm{KL}(f|g) \leq \mathrm{KL}(\pi^a|\pi^b) + \sum_{i=1}^{K} \pi_i^\alpha \, \mathrm{KL}(\mathcal{N}(\mu_i^a, \sigma_i^a)|\mathcal{N}(\mu_i^b, \sigma_i^b)) \tag{4}$$

Regarding the latter challenge, Gaussian mixture is regarded as a consolidation of the categorical and Gaussian distributions. We propose a strategy to approximate the expected-log likelihood (Equation 1) with Monte Carlo and reparameterization trick. Algorithm 1 presents the approximation of the expected-log likelihood. The discrete mixing coefficients $y = \{y_1, y_2, .., y_K\}$ are reparameterized based on the Gumbel-Softmax trick. Then samples $u = \{u_1, u_2, .., u_K\}$) generated by each Gaussian component are linearly combined: $h = \sum_{k=1}^{K} u_k y_k$ to calcu-

late the expected-log likelihood. Moreover, the mixture coefficient π is reparameterized by a 1-1 transformation via softmax function: $\pi = \text{soft\_max}(0, \hat{\pi}_1, \hat{\pi}_2, \ldots \hat{\pi}_{K-1})$. For brevity, this turns the constrained optimization problem into an unconstrained one that could be optimized by back-propagation gradient.

We emphasize that the proposed technique is model-agnostic, which means that it is compatible with any Bayesian-based approaches. Without loss of generality, we analyze the application of our scheme on VCL [14] and UCB [5]. In Algorithm 2, we derived the training algorithm for-almost-all continual learning methods following the Bayesian principle. For those methods which adapt the learning in each iteration (e.g. UCB, Sect. 2.2), a `LearningRateUpdate` procedure, which takes the current learning rate α and parameter of interest λ and returns the updated learning rate, would be retained as in the original study.

Algorithm 2: Training of proposed method in continual learning scenario

Input: A sequence of T datasets $\mathcal{D}_{t=1}^{T} = \{x_t^{(n)}, y_t^{(n)}\}_{n=1}^{N_T}$
Prior distribution $p_t(\theta)$, number of samples : N
Learning rate α_λ, temperature: τ

Output: Update the variational parameter λ for t^{th} task

```
for t ← 1 to T do
    repeat
        L1 = Log-likelihoodEstimation(p_t(θ), q_t(θ|λ), N, τ)
        L2 = Upperbound_KL(q_t(θ|λ)|p_t(θ))
        L  = L1 − L2
        λ = λ − α_λ ∇_λ L
        α_λ = LearningRateUpdate(α_λ, λ)          // Optional
    until convergence;
end
return Updated variational parameter λ
```

3.2 Dimension reduction via neural matrix factorization

Using Gaussian mixture approximation leads to the number of parameters to being multiplied (along with the size of components) in comparison with unimodal Gaussian approximation. Consequently, optimization via a typical algorithm is expensive, and a dimensional reduction technique is necessary in this case. Recently, [17] proposed an idea of decomposing variance of each Gaussian component $\mathcal{N}(\,.\,|\mu, \sigma)$ in (3) by: $\sigma = \text{diag}(H) = \text{diag}(UV^T)$ for some matrix $H \in \mathbb{R}^{M\times N}, U \in \mathbb{R}^{M\times K}, V \in \mathbb{R}^{N\times K}$ and K is often set equal to a small positive integer. As a result, the number of parameters decreases from $M \times N$ to $(M + N) \times K$ in each component. Experimental results showed that this matrix factorization not only compresses the neural network, but also provides competitive performance.

Table 1: Average accuracy on final task

Method \ Dataset	**permuted MNIST**	**split MNIST**	**fashion MNIST**	**notMNIST**
VCL-Gauss	73.77	96.9	95.7	92.1
VCL-GMM	75.52	97.77	97.78	93.9

Unfortunately, it would be tough to tune the target rank for factorization in the continual learning scenario, since a large K causes the over-parameterization in some tasks whereas a small K might be underfitting in the others. For a better solution, we alternatively use neural matrix factorization [4] to overcome this shortcoming and describe the amelioration gained in Sect. 4.4. With $H \approx \text{MLP}(U, V)$, this dimension reduction is no longer a low-rank approximation. The density function of each Gaussian component in the posterior is: $\mathcal{N}(\,.\,|\mu, \sigma) = \mathcal{N}(\,.\,|\mu, f(U, V))$ where f is defined as the multilayer perceptron parameterized by θ.

4 Experiments

To study the contributions of the above methodologies in the continual learning scenario, we conducted extensive experiments in comparison with earlier baselines, which are also governed by the Bayesian regime. Additionally, in the last subsection, we analyze the effect of the matrix factorized used in dimension reduction. All the results are averaged on five random seeds.

Datasets: The datasets for evaluation are MNIST, fashion MNIST and notMNIST. The detailed settings are described in each of the following sections.

Evaluation metrics: At the point our model has been trained on i consecutive tasks so far, let $R_{i,j}$ be the accuracy of the achieved on j^{th} task. We use two different benchmark protocols (higher is better) to evaluate the performance at T^{th} task: $\text{ACC}_\text{T} = \frac{1}{T}\sum_{i=1}^{T} R_{T,i}$ and $\text{BWT}_\text{T} = \frac{1}{T-1}\sum_{i=1}^{T-1} R_{T,i} - R_{i,i}$. Conceptually, the **Average accuracy** (ACC_T) score is to measure model's overall performance, whereas **Backward Transfer** (BWT_T) indicates its knowledge transfer ability on preceding tasks.

4.1 Task-incremental with multi-head architecture

First, we incorporate the introduced techniques into VCL on the four below benchmark datasets:

- **Permuted MNIST:** [10] is a variation of the original MNIST, in which the image's pixels in are randomly shuffled via a random (and fixed) permutation at each task.

– **Split MNIST:** [18] MNIST is partitioned into five subsets. In particular, each of them comprises images from two different classes, namely 0/1, 2/3, 4/5, 6/7, 8/9.
– **FashionMNIST, NotMNIST:** Similar to the split MNIST, our model would be incrementally trained in five separated binary classification tasks.

We replicate the model architectures used in [14], which is composed of two fully connected layers (100 units each for permuted MNIST and 256 for split MNIST). The given results in Table 1 show that VCL with mixture posterior significantly outperforms the one with single diagonal Gaussian in terms of Average accuracy ($1 - 2\%$ each dataset).

Ordinarily, the architecture of a discriminative model can be divided into two parts: classifier (final layer) and extractor (earlier layers), the catastrophic forgetting might occur on two components with different degrees, depending on the data. Various methods [14, 15] relied on the additional information about task identification at inference time, as such, to avoid CF in the classifier.

4.2 Task-incremental with single-head architecture

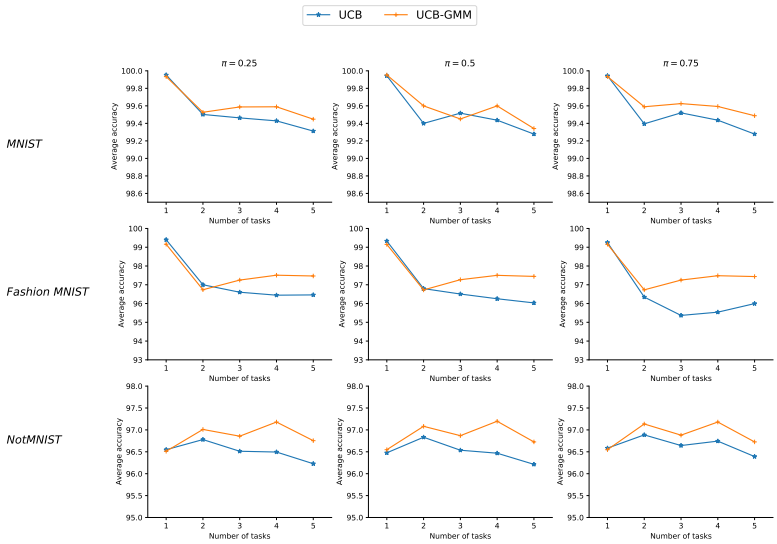


Fig. 2: Gaussian and Gaussian Mixture on UCB

The VCL's prerequisite about task boundaries, however, is rarely feasible in the use cases. In contrast, UCB implemented a single head network for all tasks, thus becoming an effective baseline. Before digging deeper into the experiment result, we briefly recall a minor difference between UCB and VCL, which lies in

the chosen prior. UCB uses a Gaussian mixture of two components (as mentioned in [2]) with the weight factor π.

To ensure a fair comparison, the model architecture again remains as the original UCB implementation [+]. Moreover, we carefully select π on different values $(0.25, 0.5, 0.75)$. We then plot the overall performance on split MNIST, fashion MNIST and notMNIST in Fig. 2.

4.3 Task-incremental with data overlapping

Up to now, many prior works focused on datasets with isolation characteristics, which means there is no class overlapping in two separated tasks. To the best of our knowledge, this is the first experimental setup to simulate the repetition of data at different times. For example, the MNIST dataset now is allocated for nine classification problems: $0/1, 1/2, 2/3, \ldots, 8/9$. This larger correlation allows the learners to selectively transfer information between tasks in a soft way. We

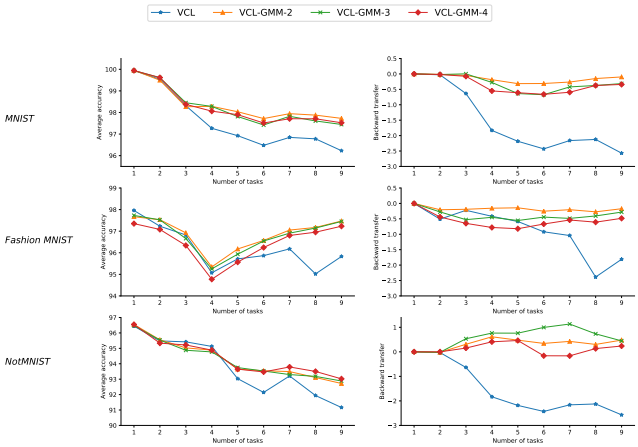


Fig. 3: ACC (left) and BWT (right)

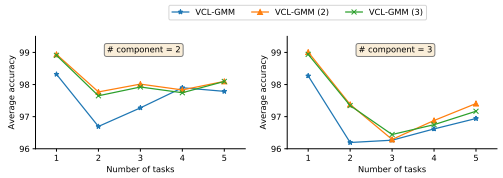
observe that our proposed method in this setting produces stable outputs: the performances on a range of hyper-parameters (ACC curves stay closed to each other for all values of num_component $\in \{2, 3, 4\}$) and the knowledge transfer abilities (BWTs almost keep unchanged).

4.4 Additional experiments about dimension reduction

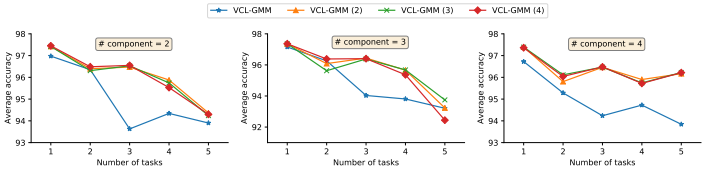
In Sect. 3.2, we suggested that neural matrix factorization can compact the variational approximation for the covariance since it reduces the number of parameters without diminishing the capacity. In the implementation, we employ a two-hidden-layer MLP with the size of $(2K, K, K, 1)$ which decreases the number of

[+] https://github.com/SaynaEbrahimi/UCB

trainable parameters for each component from $2MN$ to $MN+K(M+N+3K+1)$ and offers comparatively less resource in training when $M, N >> K$. With similar strategies, Fig. 4 quantify the impact of this technique in Fashion MNIST and NotMNIST datasets. We observe ~1% increase in average accuracy on almost all experiments, especially $2,4\%$ (from $93,8 \rightarrow 96,4$) in the case of 4 components GMM on NotMNIST.



(a) Fashion MNIST



(b) NotMNIST

Fig. 4: Neural matrix factorization on multi-head VCL

5 Conclusion and future work

In this paper, we propose a framework, which is applicable to any Bayesian-based approach for continual learning. Our methodology unifies several recent proposals in variational inference and latent feature models and entails significant gains in the model's performance. We found that it is beneficial for incorporating these enhancements into VCL and UCB via distinct setups. The future work should take the initialization of mixtures into account (e.g. Iterated Laplace Approximations). In addition, we plan to properly find the number of components of the Gaussian mixture according to each task or even cast it into a trainable parameter while ensuring computation time and memory.

Acknowledgements

This work was funded by Gia Lam Urban Development and Investment Company Limited, Vingroup and supported by Vingroup Innovation Foundation (VINIF) under project code VINIF.2019.DA18

References

1. Ahn, H., Cha, S., Lee, D., Moon, T.: Uncertainty-based continual learning with adaptive regularization. arXiv preprint arXiv:1905.11614 (2019)
2. Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D.: Weight uncertainty in neural network. In: International Conference on Machine Learning. pp. 1613–1622. PMLR (2015)
3. Do, M.N.: Fast approximation of kullback-leibler distance for dependence trees and hidden markov models. IEEE signal processing letters **10**(4), 115–118 (2003)
4. Dziugaite, G.K., Roy, D.M.: Neural network matrix factorization. arXiv preprint arXiv:1511.06443 (2015)
5. Ebrahimi, S., Elhoseiny, M., Darrell, T., Rohrbach, M.: Uncertainty-guided continual learning with bayesian neural networks. In: International Conference on Learning Representations (ICLR) (2020)
6. Farquhar, S., Gal, Y.: A unifying bayesian view of continual learning. arXiv preprint arXiv:1902.06494 (2019)
7. Jaakkola, T.S., Jordan, M.I.: Improving the mean field approximation via the use of mixture distributions. In: Learning in graphical models, pp. 163–173. Springer (1998)
8. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax. In: International Conference on Learning Representations, ICLR (2017)
9. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: International Conference on Learning Representations, ICLR (2014)
10. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences **114**(13), 3521–3526 (2017)
11. Linh Ngo Van, Nam Le Hai, H.P., Than, K.: Auxiliary local variables for improving regularization/prior approach in continual learning. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD) (2022)
12. Maaløe, L., Sønderby, C.K., Sønderby, S.K., Winther, O.: Auxiliary deep generative models. In: International conference on machine learning. pp. 1445–1453. PMLR (2016)
13. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: The sequential learning problem. In: Psychology of learning and motivation, vol. 24, pp. 109–165. Elsevier (1989)
14. Nguyen, C.V., Li, Y., Bui, T.D., Turner, R.E.: Variational continual learning. In: International Conference on Learning Representations (ICLR) (2018)
15. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 2001–2010 (2017)
16. Rezende, D., Mohamed, S.: Variational inference with normalizing flows. In: International conference on machine learning. pp. 1530–1538. PMLR (2015)
17. Swiatkowski, J., Roth, K., Veeling, B., Tran, L., Dillon, J., Snoek, J., Mandt, S., Salimans, T., Jenatton, R., Nowozin, S.: The k-tied normal distribution: A compact parameterization of gaussian mean field posteriors in bayesian neural networks. In: International Conference on Machine Learning. pp. 9289–9299. PMLR (2020)
18. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: International Conference on Machine Learning. pp. 3987–3995. PMLR (2017)

A Appendix

In this section, we present the proof of KL upper bound in theorem 1:

Theorem 2. *Consider 2 mixtures* $f(x) = \sum_{i=1}^{K} \pi_i^a \mathcal{N}(\mu_i^a, \sigma_i^a)$ *and* $g(x) = \sum_{i=1}^{K} \pi_i^b \mathcal{N}(\mu_i^b, \sigma_i^b)(\pi_i^a, \pi_i^b \in \Delta^{K-1})$*, we have the below inequality:*

$$\mathrm{KL}(f|g) \leq \mathrm{KL}(\pi^a|\pi^b) + \sum_{i=1}^{K} \pi_i^\alpha \, \mathrm{KL}(\mathcal{N}(\mu_i^a, \sigma_i^a)|\mathcal{N}(\mu_i^b, \sigma_i^b)) \tag{5}$$

Lemma 1: Jensen inequality

Suppose $a_1, a_2, \ldots, a_K$ are non negative numbers whose sum equal to 1 and $x_1, x_2, \ldots, x_K$ are K arbitrary real numbers . Given a real-value convex function $f(x) : \mathbb{R} \to \mathbb{R}$. Jensen inequality shows that:

$$f(\sum_{i=1}^{K} a_i x_i) \leq \sum_{i=1}^{K} a_i f(x_i)$$

The equality holds if and only if $x_i = x_j \; \forall i \neq j$

Lemma 2: Log sum inequality

Let $a_1, a_2, \ldots, a_K$ and $b_1, b_2, \ldots, b_K$ be non-negative numbers. We have:

$$\sum_{i=1}^{K} a_i \log \frac{a_i}{b_i} \geq (\sum_{i=1}^{K} a_i) \log \frac{\sum a_i}{\sum b_i}$$

Proof: Consider $f(x) = x \log(x)$ $(x \in \mathbb{R}^+)$. We have:

$$\Rightarrow f'(x) = \log(x) + 1$$

$$\Rightarrow f^{"}(x) = \frac{1}{x} > 0 \text{ or } f(x) \text{ is a strictly convex function.}$$

Using Jensen inequality:

$$\sum_{i=1}^{K} \alpha_i f(t_i) \geq f(\sum_{i=1}^{K} \alpha_i t_i)$$

keeps for any $\alpha \geq 0$ has the property $\sum_i \alpha_i = 1$.
Choose $\alpha_i = \frac{b_i}{\sum b_j}$ and $t_i = \frac{a_i}{b_i}$:

$$\sum_{i=1}^{K} \frac{b_i}{\sum b_j} \frac{a_i}{b_i} \log\left(\frac{a_i}{b_i}\right) \geq \left(\sum_{i=1}^{K} \frac{b_i}{\sum b_j} \frac{a_i}{b_i}\right) \log\left(\sum_{i=1}^{K} \frac{b_i}{\sum b_j} \frac{a_i}{b_i}\right)$$

$$\Leftrightarrow \frac{\sum a_i \log\left(\frac{a_i}{b_i}\right)}{\sum b_j} \geq \frac{(\sum a_i) \log\left(\frac{\sum a_i}{\sum b_i}\right)}{\sum b_j}$$

$$\Leftrightarrow \sum_{i=1}^{K} a_i \log\left(\frac{a_i}{b_i}\right) \geq \left(\sum_{i=1}^{K} a_i\right) \log\left(\frac{\sum_i a_i}{\sum b_i}\right) \text{(q.e.d)}$$

Proof for Theorem 1

$$\begin{aligned}
\mathrm{KL}(f||g) &= \mathrm{KL}(\sum_{i=1}^{K} \pi_i^a f_i || \sum_{i=1}^{K} \pi_i^b g_i) \\
&= \int_x \sum_{i=1}^{K} \pi_i^a f_i(x) \log\left(\frac{\sum \pi_i^a f_i(x)}{\sum \pi_i^b g_i(x)}\right) dx \\
&\leq \int_x \sum_{i=1}^{K} \pi_i^a f_i(x) \log\left(\frac{\pi_i^a f_i(x)}{\pi_i^b g_i(x)}\right) dx \\
&= \int_x \sum_{i=1}^{K} \pi_i^a f_i(x) log\left(\frac{\pi_i^a}{\pi_i^b}\right) dx - \int_x \sum_i \pi_i^a f_i(x) \log\left(\frac{f_i(x)}{g_i(x)}\right) dx \\
&= \sum_{i=1}^{K} \pi_i^a \log\left(\frac{\pi_i^a}{\pi_i^b}\right) - \sum_{i=1}^{K} \pi_i^a \int_x f_i(x) \log\left(\frac{f_i(x)}{g_i(x)}\right) dx \\
&= \mathrm{KL}(\pi^a||\pi^b) + \sum_{i=1}^{K} \pi_i^a \mathrm{KL}(f_i||g_i) \\
&= \mathrm{KL}(\pi^a|\pi^b) + \sum_{i=1}^{K} \pi_i^\alpha \ \mathrm{KL}(\mathcal{N}(\mu_i^a, \sigma_i^a)|\mathcal{N}(\mu_i^b, \sigma_i^b))
\end{aligned}$$

Hence, this gives us the desired inequality in Equation2.