Kronecker-factored approximation (KFAC) of the Laplace-GGN for Continual Learning

Nadhir Hassen

Abstract

One major challenge in Continual Learning is that the model forgets how to solve earlier tasks commonly known as *Catastrophic Forgetting*. Extensive work has been recently made in this direction relies of weight regularisation but may not be effective for complex deep neural network. Recently functional regularisation was developed to directly regularise the network output, although computationally expensive, is expected to perform better. However this technique limits the posterior expressiveness of fully-factorized Gaussian assumption (Cremer et al., 2018) of the inference model. Functional regularisation (Pan et al., 2021) has been developed to combine parameter and function space regularisation but does not take into account structured parameter spaces where have complicated inverse Fisher-matrix computations, which give poor predictives estimates. The method applies a full covariance Gaussian posterior approximation and in case of diagonal posterior approximation it fails to model meaningful relationships. In this work we address this issue without alternating the complexity cost. We propose a Laplace Gauss-Newton approximation to combine local parametrization and function space learning using Kronecker-factored approximation (KFAC) of the Laplace-Gauss-Newton (GGN). We use KFAC (Martens & Grosse, 2020a) posterior approximation for image classification to illustrate how we can model parameter covariances per layer without altering scalability in a Continual Learning setting.

1. Introduction

In this work we make the connection to *Continual Learning* setting by using a Gaussian Process formulation of deep networks and we make it adaptive to streaming data using variational inference. This formulation enables to resolve common underfitting problem of Laplace approximation and shows how we overcome *Catastrophic Forgetting*. We demonstrate the efficacy of our approach on different classifications problem and provide a meaningful analysis for uncertainty quantification. Finally we identify the benefits of our approach by comparing the performance to the state-of-the-art on standard benchmarks. A body of work has been done in *Continual Learning* to employ the Bayesian formalism to train deep neural network . In a general setting, a loss function defined as $N\bar{\ell}(\mathbf{w}) + \delta R(\mathbf{w})$, where $R(\mathbf{w})$ is a regularizer and δ a non-negative tempering parameter with $\mathbf{w} \in \mathbb{R}^P$. The loss function can be optimized using the GP posterior predictive construction over the function space. In the case of a supervised learning problem where the network weights $\mathbf{w}_{t-1}$ for a fixed task t are given and can be obtained through data training from task $t-1$. (Titsias, 2009) proposed a regulariser as $R(\mathbf{w}) = -\log p(\mathbf{w}|\mathcal{D}_{1:t-1}) \approx q_{t-1}(\mathbf{w}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ then used a sparse formulation of GP where a subset of vector of function outputs $\mathbf{f}_{1:t}$ over these examples denoted by $\mathbf{u}_{1:t}$, the goal is to optimise the weights $\mathbf{w}$ such that the prediction using $q_{\mathbf{w}}(\mathbf{f}_t)$ are good on current tasks while the predictive distribution $q_{\mathbf{w}}(\mathbf{u}_{1:t-1})$ is close to $q_{\mathbf{w}_{t-1}}(\mathbf{u}_{1:t-1})$. The authors regularize each task separately, that is, for all task $s < t$ we can compute $q_{\mathbf{w}}(\mathbf{u}_s)$ with $q_{\mathbf{w}_t-1}(\mathbf{u}_s)$ separately. For a trade-off parameter τ, the objective function with a KL-divergence term on a constraint family distribution $q \in \mathcal{Q}$ referred as *Mean-field-Variational Inference* take the following form

$$\max_{\mathbf{w}} - \mathbb{E}_{q_{\mathbf{w}}(\mathbf{f}_t)} \left[\bar{\ell}(\mathbf{y}, \mathbf{f}(\mathbf{x}))\right] + \tau \sum_{s=1}^{t-1} D_{KL}\left(q_{\mathbf{w}_t}(\mathbf{u}_s) \parallel q_{\mathbf{w}_{t-1}}(\mathbf{u}_s)\right). \quad (1.1)$$

However, computing the Gaussian variational approximation can be very expensive to optimize in the function space and still requires variational inference in weight space. (Pan et al., 2021) used a functional regulariser defined over memorable examples that constraints to the most informative set of data points. They consider a vector of function values $\mathbf{f}$ defined at a constraint set of input data such as inducing points and pose $q(\mathbf{w}) = \mathcal{GP}(\mathbf{m}_{\mathbf{w}}(\mathbf{x}), \boldsymbol{\kappa}_{\mathbf{w}}(\mathbf{x}, \mathbf{x}'))$. The key idea is to sample $\mathbf{w}$ from $q(\mathbf{w})$ to obtain $\tilde{q}(\mathbf{w})$ a GP posterior over $\mathbf{f}$. The approximate distribution $\tilde{q}_{\mathbf{w}_t}(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{m}_t, \mathbf{K}_t)$ called *functional prior* where $\mathbf{m}_t$ and $\mathbf{K}_t$ are the mean vector and the kernel matrix by evaluating $\mathcal{GP}(\mathbf{m}_{\mathbf{w}}(\mathbf{x}), \boldsymbol{\kappa}_{\mathbf{w}}(\mathbf{x}, \mathbf{x}'))$ at inducing locations. However,

in high dimensional spaces, conditional distributions become increasingly complicated to model the Bayesian neural network. Therefore their method does not extract enough information when tasks deviate too much from its prior distribution. To address this issue we want to exploit the local structure of the neural network feature by incorporating various structured priors across tasks. To approach this problem we build upon Pan et al. (2021) and Immer et al. (2021) works and show by allowing local linearized parameters estimates the neural network using GGN approximation which turns into a GLM model. This allows to remember old tasks by selectively slowing down learning on the weights important for those tasks using KFAC approximation to account meaningful paramter covariances, we call our approach *Stream KFAC*. We demonstrate that the GLM formulation allows to exploit a richer structure of the BNN while resolving the underfitting problems in a scalable and efficient manner. Further, we show how to formulate the optimization problem using *Variational Online Newton* (VON) algorithm to recursively optimizing the variational distribution. Finally, we show that the proposed method can be successfully used for uncertainty quantification and out-of-distribution detection.

2. Background

2.1. GP posterior for the Neural Network with Laplace-GGN Approximation

We recall our generic loss as $\ell(\mathbf{f}, \mathbf{y})$ and denote $\mathbf{w}_\star$ a local minimum that minimise the loss $N\bar{\ell}(\mathbf{w}) + \frac{1}{2}\delta \mathbf{w}^T\mathbf{w}$ for a-K output $\mathbf{f}(\mathbf{x}; \mathbf{w})$. The minima of the loss function corresponds to the mode of the Bayesian model $p(\mathcal{D}, \mathbf{w}) := \prod_i^N \exp\{-\ell_i(\mathbf{w})\}p(\mathbf{w})$. The posterior is obtained by the Bayes rule $p(\mathbf{w}|\mathcal{D}) = p(\mathcal{D}, \mathbf{w})/p(\mathcal{D})$ but it is intractable. The Laplace approximation is based on a Gaussian approximation of the posterior. (Khan et al., 2020) derived a GP posterior based on the Generalized Gauss-Newton (GGN) approximation that can be directly built using the solutions found by deep-learning optimizers. The idea is to make the Laplace approximation $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ equal to the posterior distribution $p(\mathbf{w}|\mathcal{D})$. The Laplace approximation is based on the *Maximum a posterior* or MAP solution, we maximize the log joint distribution which leads to a more convenient objective $\arg\max_{\mathbf{w}} \sum_{i=1}^N \log p(\mathbf{y}_i|\mathbf{f}(\mathbf{x}_i, \mathbf{w})) + \log p(\mathbf{w})$ where

$$\begin{aligned}\nabla_{\mathbf{w}} \log p(\mathbf{y}|\mathbf{f}(\mathbf{x}, \mathbf{w})) &= \boldsymbol{J}(\mathbf{x})^T \boldsymbol{r}(\mathbf{y}, \mathbf{f}),\\ \nabla^2_{\mathbf{ww}} \log p(\mathbf{y}|\mathbf{f}(\mathbf{x}, \mathbf{w})) &= \boldsymbol{H}(\mathbf{x})^T \boldsymbol{r}(\mathbf{y}, \mathbf{f})\\ &\quad - \boldsymbol{J}(\mathbf{x})^T \boldsymbol{\Lambda}(\mathbf{y}, \mathbf{f}) \boldsymbol{J}(\mathbf{x}).\end{aligned}$$

We denote the residuals by $\boldsymbol{r}(\mathbf{y}, \mathbf{f}) = \nabla_{\mathbf{f}} \log p(\mathbf{y}|\mathbf{f})$, $\boldsymbol{\Lambda}(\mathbf{y}, \mathbf{f}) = -\nabla^2_{\mathbf{ff}} \log p(\mathbf{y}|\mathbf{f})$ as per-input noise and the Hessian by $\boldsymbol{H}(\mathbf{x}) = \nabla^2_{\mathbf{ww}} \mathbf{f}(\mathbf{x}, \mathbf{w})$, often this quantity is infeasible for large network. The *Generalized-Gaussian-Newton* approximates the network Hessian $\boldsymbol{H}(\mathbf{x})$ proposed by Martens & Grosse (2020b) as the following $\nabla^2_{\mathbf{ww}} \log p(\mathbf{y}|\mathbf{f}(\mathbf{x}, \mathbf{w})) \approx -\boldsymbol{J}(\mathbf{x})^T \boldsymbol{\Lambda}(\mathbf{y}, \mathbf{f}) \boldsymbol{J}(\mathbf{x})$. This approximation assumes that $\boldsymbol{H}(\mathbf{x})^T \boldsymbol{r}(\mathbf{y}, \mathbf{f}) = 0$. (Bottou et al., 2018) provided two independents sufficient conditions as a justification: i) If the neural network is a perfect predictor, the residual $\boldsymbol{r}(\mathbf{y}, \mathbf{f}(\mathbf{x}, \mathbf{w})) = 0$ vanishes for all data points $(\mathbf{x}, \mathbf{y})$, this can indicate overfitting and can be unrealistic. ii) The network model is linear envolving the Hessian to vanish. We follow the second alternative as (Martens & Grosse, 2020b) and formulate a *local linearization* of the network function $\mathbf{f}(\mathbf{x}, \mathbf{w})$ given by

$$\mathbf{f}^{\mathbf{w}_\star}_{\text{lin}}(\mathbf{x}, \mathbf{w}) = \mathbf{f}(\mathbf{x})_{\mathbf{w}\star}; \mathbf{w}_\star) + \boldsymbol{J}_{\mathbf{w}_\star}(\mathbf{x})(\mathbf{w} - \mathbf{w}_\star).$$

This linearization reduces the *Bayesian Neural Network* (BNN) to a *Bayesian Generalized Linear* (GLM) model. The coressponding log-joint distribution is given by

$$\ell_{\text{glm}}(\mathbf{w}, \mathcal{D}) = \sum_{i=1}^N \log p(\mathbf{y}_i|\mathbf{f}^{\mathbf{w}_\star}_{\text{lin}}(\mathbf{x}_i, \mathbf{w})) + \log p(\mathbf{w}),$$

where the linearization appears in the parameters and not in the input $\mathbf{x}$. The GGN-approximation brings two benefits, first the Hessian $\boldsymbol{H}$ guarantees to be positive semi-definite and second applying this approximation to the likelihood Hessian turns the underlying probabilistic model locally from a BNN into a GLM. The *Laplace-GGN-approximation* applies jointly the Laplace approximation and the GGN-approximation (Khan et al., 2018), we refer the posterior approximation $q(\mathbf{w}) = \mathcal{N}(\mathbf{w}_{\text{MAP}}, \boldsymbol{\Sigma}_{ggn})$, with

$$\boldsymbol{\Sigma}^{-1}_{ggn} = \sum_{i=1}^N \boldsymbol{J}_{\mathbf{w}_\star}(\mathbf{x}_i)^T \boldsymbol{\Lambda}(\mathbf{y}_i, \mathbf{f}_i) \boldsymbol{J}_{\mathbf{w}_\star}(\mathbf{x}) + \mathbf{S}_0^{-1}, \quad (2.1)$$

where $\mathbf{S}_0$ denotes the prior covariance such that $p(\mathbf{w}) = \mathcal{N}(\mathbf{m}_0, \mathbf{S}_0)$. From a generalised linear model point of view, we can construct a generic loss function $\ell(\mathbf{y}, \mathbf{f}) := -\log p(\mathbf{y}|\mathbf{h}(\mathbf{f}))$ where $\mathbf{h}(\mathbf{f})$ is an invertible function, $\mathbf{h}^{-1}$ is known as the link function. In the case of a Bernoulli distribution, the link function $\mathbf{h}(\mathbf{f})$ is the sigmoid function σ. Therefore to compute the predictive distribution with a given loss function the only thing that changes is $\boldsymbol{\Lambda}_{\mathbf{w}_\star}(\mathbf{x}, \mathbf{y}) := \nabla^2_{\mathbf{ff}} \ell(\mathbf{y}, \mathbf{f})$ which depend on $\mathbf{x}$ and $\mathbf{y}$. Let's consider a probabilistic neural network, we define a likelihood function as a Bernoulli $p(\mathbf{x}) := \mathbf{g}^{-1}_{\text{lin}}(\mathbf{x}; \mathbf{w}) = \sigma(\mathbf{f}(\mathbf{x}; \mathbf{w}))$ and the labels $y \in \{0, 1\}$. Following Rasmussen & Williams (2006) we can recover the GP formulation with mean $\mathbf{m}(\mathbf{x})$ and covariance function $\boldsymbol{\kappa}(\mathbf{x}, \mathbf{x}')$. The GP posterior takes the following form $\mathbf{m}_{\mathbf{w}_\star}(\mathbf{x}) := p(\mathbf{x})$ and $\boldsymbol{\kappa}_{\mathbf{w}_\star}(\mathbf{x}, \mathbf{x}') := \boldsymbol{\Lambda}_{\mathbf{w}_\star}(\mathbf{x}, y) \boldsymbol{J}_{\mathbf{w}\star}(\mathbf{x}) \boldsymbol{\Sigma}_{\text{ggn}\star} \boldsymbol{J}_{\mathbf{w}\star}(\mathbf{x}')^T \boldsymbol{\Lambda}_{\mathbf{w}_\star}(\mathbf{x}, y)$ the predictive posterior is

$$\begin{aligned}&\hat{p}(\mathbf{y}|\mathbf{x}, \mathcal{D}) = \mathcal{N}(\mathbf{y}|p(\mathbf{x}),\\ &\boldsymbol{\Lambda}_{\mathbf{w}_\star}(\mathbf{x}, y) \boldsymbol{J}_{\mathbf{w}\star}(\mathbf{x}) \boldsymbol{\Sigma}_{\text{ggn}\star} \boldsymbol{J}_{\mathbf{w}\star}(\mathbf{x}')^T \boldsymbol{\Lambda}_{\mathbf{w}_\star}(\mathbf{x}, y)) \quad (2.2)\end{aligned}$$

where $\mathbf{\Sigma}_{\text{ggn}\star}^{-1}$ denote the Hessian matrix of the loss function evaluated at all $\mathbf{w} = \mathbf{w}_\star$ with dimension $K \times K$ and $\boldsymbol{J}_{\mathbf{w}_\star}(\mathbf{x})$ is the $P-$ Jacobian vector. The approximate inference in this GP model is obtained in closed-form. For a new location $\mathbf{x}_\star$ the Lapalce-GGN-approximation evaluated at $\mathbf{w}_\star = \mathbf{w}_{\text{MAP}}$ we have $\mathbf{f}_\star|\mathbf{x}_\star, \mathcal{D} \sim \mathcal{N}\left(\mathbf{f}(\mathbf{x}_\star; \mathbf{w}_\star), \sigma_\star^2\right)$ with $\sigma_\star^2 = \mathbf{K}_{\mathbf{x}_\star\mathbf{x}_\star} - \mathbf{K}_{\mathbf{x}_\star\mathbf{X}}\left(\mathbf{K}_{\mathbf{XX}} + \mathbf{\Lambda}_{\mathbf{XX}}^{-1}\right)^{-1}\mathbf{K}_{\mathbf{X}\mathbf{x}_\star}$, where $\mathbf{K}_{\mathbf{x}_\star\mathbf{X}}$ denotes the kernel evaluated between $\mathbf{x}_\star$ and the N training points, and $\mathbf{\Lambda}_{\mathbf{XX}}$ is a diagonal matrix with entries $\mathbf{\Lambda}(\mathbf{y}_n; \mathbf{f}_n)$ for $n \in \mathbb{N}$ data points. This can be generalised to other loss functions such as regression and multi-class classification. In addition, this GP posterior is referred as a functional prior that we will discuss later. Bishop (2006) provided a useful intepretation for the predective variance in equation (2.1). That is the first term can be interpreted as the epistemic uncertainty (model noise), while the second term takes a form that resembles the aleatoric uncertainty (label noise).

3. Methods

3.1. Streaming Deep Neural Network with Laplace-GNN approximation

The ability to formulate a probabilistic adaptive methods that handle time evolving behavior can improve substantially the model fit and its applicability on real-world problems. The difficulties in such models are on one hand the adaptation of the non-linear latent functions based on the former posterior discoveries as new prior beliefs. The main goal of Continual Learning is to avoid *Catastrophic Frogetting*. One approach is to include new data as they arrive and retrain the Gaussian process model every time, however this scale poorly on large dataset. We build our approach upon the work of Bui et al. (2017) which is to find a way how the covariances matrices could incrementally be adapted to new incoming samples to avoid *Catastrophic Frogetting* and reduce the computational cost using a sparse version of it. This method is based on optimizing the variational distribution in a streaming fashion. The variational distribution $q_t(\mathbf{u}_t, \mathbf{f})$ is recursively updated as the batch data $\mathcal{D}_t$ is coming, where $\mathbf{u}$ are the function values evaluated at induced point location $\mathbf{z}_t$. The true posterior is given by

$$p(\mathbf{u}_t, \mathbf{f}_t|\mathcal{D}_{1:t}) = \frac{p(\mathcal{D}_{1:(t-1)}|\mathbf{f}_t)p(\mathcal{D}_t|\mathbf{f}_t)p(\mathbf{u}_t, \mathbf{f}_t)}{p(\mathcal{D}_{1:t})}. \quad (3.1)$$

The data is approximated using the previous variational approximation $q_{t-1}(\mathbf{u}_{t-1})$, the variational distribution q_t is obtained by

$$q(\mathbf{u}_t, \mathbf{f}_t|\mathcal{D}_{1:t}) \approx \frac{p(\mathcal{D}_{1:(t-1)})q_{t-1}(\mathbf{u}_{t-1})}{p(\mathbf{u}_{t-1})}\frac{p(\mathcal{D}_t|\mathbf{f}_t)p(\mathbf{u}_t, \mathbf{f}_t)}{p(\mathcal{D}_{1:t})} \quad (3.2)$$

where

$$p(\mathcal{D}_{1:(t-1)}|\mathbf{f}_t) \approx \frac{p(\mathcal{D}_{1:(t-1)})q_{t-1}(\mathbf{u}_{t-1})}{p(\mathbf{u}_{t-1})}.$$

That is at $\mathcal{D}_{1:(t-1)}$ the data is not accessible anymore, therefore the likelihood at $t-1$ is approximated by $q_{t-1}(\mathbf{u}_{t-1})$. The information of $\mathcal{D}_{1:(t-1)}$ acquired so far will be preserved via the optimization of the functional regularizer between the two distributions for each new batch. Following Bissiri et al. (2016) and Blundell et al. (2015) the bayesian formulation of a loss-based approach can be interpreted as a maximization problem with respect to a distribution $q(\mathbf{w}) \in Q$ where Q belongs to a variational mean-field family. The objective function referred as *Laplace-Functional-Prior* takes the form of a weight regularizer through the Neural Network and a functional regularizer

$$\max_{q(\mathbf{w})\in Q} -N\mathbb{E}_{q_{(\mathbf{w})}}\left[\bar{\ell}(\mathbf{y}, f_{\mathbf{w}}(\mathbf{x})) + \log\frac{q(\mathbf{w})}{p_(\mathbf{w})}\right] + \tau\mathbb{E}_{\tilde{q}_{\mathbf{w}}(\mathbf{f})}\left[\log\frac{\tilde{q}_{\mathbf{w}}(\mathbf{f})}{p_{\mathbf{w}}(\mathbf{f})}\right], \quad (3.3)$$

where $\tau > 0$ is a tempering parameter. The functional regulariser in the rhs of the above equation uses an approximation to convert the weight space into the function space *see Appendix.D*. The assumption made here is that $q(\mathbf{w}) = \mathcal{N}(\mathbf{w}_{\text{MAP}}, \mathbf{\Sigma}_{ggn}) \approx p(\mathbf{w}|\mathcal{D})$. Interestingly, the functional regularizer can be interpreted as a KL-divergence. The weights are sampled according to $\mathbf{w} \sim q(\mathbf{w})$ and $\tilde{q}_t(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{m}_t(\mathbf{w}), \mathbf{K}_t(\mathbf{w}))$ with $\mathbf{m}_t(\mathbf{w})$ and $\mathbf{K}_t(\mathbf{w})$ denote the vector and the kernel matrix obtained by evaluating $\mathcal{GP}(\mathbf{m}_{\mathbf{w}}(\mathbf{z}), \boldsymbol{\kappa}_{\mathbf{w}}(\mathbf{z}, \mathbf{z'}))$ at memorable points selection. The memorable examples or inducing point selection defined by Pan et al. (2021) corresponds to the noise precision $\mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x}_i, \mathbf{y}_i)$ and can therefore be interpreted as the relevance of the data example i. We simply pick the top influential for all i. The full derivation of the regulariser is given in the *Supplementary material*. When memorables examples do not change within tasks, then $\boldsymbol{\kappa}_{\mathbf{z}_{t-1}\mathbf{z}_t} = \mathbf{I}$ and $\mathbf{K}_{\mathbf{z}_{t-1}} = \mathbf{K}_{\mathbf{z}_t}$. The functional regulariser simplifies [1].

$$\frac{1}{2}\Bigg(Tr\left(\mathbf{K}_{\mathbf{z}_t\mathbf{z}_t}^{-1}\right) + \boldsymbol{\mu}_t^T\mathbf{K}_{\mathbf{z}_t\mathbf{z}_t}^{-1}\boldsymbol{\mu}_t + \boldsymbol{\mu}_{t-1}^T\mathbf{\Sigma}_{t-1}^{-1}\left(\boldsymbol{\mu}_{t-1} - 2\boldsymbol{\mu}_t^T\right)\mathbf{D}_{t-1}^{-1}\boldsymbol{\mu}_t\Bigg) + cst. \quad (3.4)$$

To optimise the objective function in equation (3.3) we use a variational inference algorithm referred as Variational Online Newton (VON) method. The algorithm proceeds by first sampling $\tilde{q}_t(\mathbf{w})$ at iteration t and then updating the variational distribution. In the case of a binary classification

[1] For clarity of notation, for a given distribution q we have $q_{\mathbf{w}_t}(\mathbf{u}) := q_t(\mathbf{u})$

the parameters updates take the following form

$$\boldsymbol{\Sigma}_{t+1}^{-1} \leftarrow (1-\beta_t)\,\boldsymbol{\Sigma}_t^{-1} + \beta_t \Bigg[\sum_i \boldsymbol{J}_{\mathbf{w}_t}(\mathbf{x}_i)^T \boldsymbol{\Lambda}_{\mathbf{w}_t}(\mathbf{x}_i) \boldsymbol{J}_{\mathbf{w}_t}(\mathbf{x}_i) - 2\nabla_{\boldsymbol{\Sigma}} D_{KL}\left(\tilde{q}_{\mathbf{w}_t}(\mathbf{u}) || p_{\mathbf{w}}(\mathbf{u})\right)\Bigg],$$

and

$$\mu_{t+1} \leftarrow \boldsymbol{\mu}_t - \beta_t \boldsymbol{\Sigma}_{t+1}\left[\frac{N}{\tau}\nabla_{\mathbf{w}}\bar{\ell}_t(\mathbf{w}_t) - \nabla_{\boldsymbol{\mu}} D_{KL}\left(\tilde{q}\mathbf{w}_t(\mathbf{u}) || p_{\mathbf{w}}(\mathbf{u})\right)\right].$$

For large neural network, optimizing $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ can be very expensive. Using natural gradient we clearly see that $\boldsymbol{\mu}$ depends on $\boldsymbol{\Sigma}$. Therefore we apply the KFAC approximation to the expected GGN-Hessian approximation and for the Fisher information matrix under the distribution $q(\mathbf{w})$ for a fixed likelihood. In particular, we construct a block-diagonal matrix and factorizes across the neural network layers. For a single pair data point $(\mathbf{x}_n, \mathbf{y}_n)$ we can write the KFAC approximation based on that observation evaluated at the GGN-approximation to the Hessian for a particular likelihood as follow

$$\left[\boldsymbol{J}(\mathbf{x}_n)^T \boldsymbol{\Lambda}(\mathbf{y}_n; \mathbf{f}_n) \boldsymbol{J}(\mathbf{x}_n)\right]_l = \boldsymbol{Q}_l^{(n)} \otimes \boldsymbol{W}_l^{(n)},$$

where $\boldsymbol{Q}_l$ denotes the covariance of the activation and is quadratic in the number of neurons of the l-th layer, $\boldsymbol{W}_l$ denotes the output of the layer l and is quadratic in the number of neurons of the previous layer. For each layer, we assume $p_l(\mathbf{w}) = \mathcal{N}(0, \delta^{-1}\mathbf{I})$ is an isotropic Gaussian prior where $\mathbf{I}_l$ denotes the identity matrix which store the number of parameters of the l-th layer. Both $\boldsymbol{Q}_l$ and $\boldsymbol{W}_l$ are positive semi-definite which enable fast inversion since the inversion is done individually. Zhang et al. (2018) and Botev et al. (2017) made a further approximation, commonly referred to as *dampening* in the context of second-order optimization methods. The approximation take the following form

$$\begin{aligned}&\left(\boldsymbol{Q}_l + \sqrt{\delta}\mathbf{I}\right) \otimes \left(\boldsymbol{W}_l + \sqrt{\delta}\mathbf{I}\right) \\ &= \boldsymbol{Q}_l \otimes \boldsymbol{W}_l + \boldsymbol{Q}_l \otimes \sqrt{\delta}\mathbf{I} + \sqrt{\delta}\mathbf{I} \otimes \boldsymbol{W}_l + \delta \mathbf{I}_l.\end{aligned} \tag{3.5}$$

This approximation consists of increasing the posterior concentration by the cross-products. Consequently, since the cross-products are semi-definite matrices then the eigenvalues of the posterior precision matrix increases which reduce the posterior covariance in order of magnitude. The cheap computation of the Kronecker-factored Laplace approximations is even more efficient in updating per-layer hyperparameters since it represents a per-layer block-diagonal approximation by construction. Therefore we are able to sample $\mathbf{w}$ from $q(\mathbf{w})$ to obtain $\tilde{q}(\mathbf{w})$ a GP posterior over $\mathbf{f}$ and build the approximate distribution $\tilde{q}_{\mathbf{w}_t}(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{m}_t, \mathbf{K}_t)$ by evaluating $\mathcal{GP}(\mathbf{m}_{\mathbf{w}}(\mathbf{x}), \boldsymbol{\kappa}_{\mathbf{w}}(\mathbf{x}, \mathbf{x}'))$. Finally, the loss function over tasks has the following form

$$\begin{aligned}&\max_{\mathbf{w}} -N\mathbb{E}_{q_{(\mathbf{w})}}\left[\bar{\ell}_t(\mathbf{y}, \mathbf{f}_{\mathbf{w}}(\mathbf{x}))\right] + \\ &\tau \sum_{s=1}^{t-1} \frac{1}{2}\Bigg(\mathbf{m}_{t,s}^T \mathbf{K}_{t,s}^{-1} \mathbf{m}_{t,s} + \mathbf{m}_{t-1,s}^T \mathbf{K}_{t-1,s}^{-1} \\ &(\mathbf{m}_{t-1,s} - 2\mathbf{m}_{t,s})\,\mathbf{D}_{t-1,s}^{-1}\mathbf{m}_{t,s}\Bigg) + cst,\end{aligned} \tag{3.6}$$

where $\mathbf{m}_{t,s}$ is the subvector of $\boldsymbol{\mu}_t$ corresponding to the task s, similarly $\mathbf{K}_{t,s}$ is a kernel matrix that includes the uncertainty and the weighting of the past tasks at inducing points. The sparse variational form of this functional regulariser has two advantages. First it provides a scalable computation on large neural network and second provides an analytical solution. After training on a specific task, we select a set of few memorable examples (Pan et al., 2021) which play the role of inducing points from the function space in $\mathcal{D}_t$, denoted by $\mathbf{z}_t$. This regulariser encourages the approximate variational covariance posterior $\mathbf{K}$ to remain close to the approximate second moment $\mathbf{m}_t^T \mathbf{K}_{t-1}^{-1} \mathbf{m}_{t-1}$ on one hand and on the other hand it forces the approximate mean $\mathbf{m}_t$ to be close to $\mathbf{m}_{t-1}$, which is beneficial since it encourages the predictions of the past to remain the same and therefore avoid *Catastrophic Forgetting*.

4. Computational Consideration

Optimizing the corresponding mean and covariance obtained from the evidence lower bound make the functional regulariser adaptive from task to task since it exhibits a clear dependence on past distribution of memorable examples. The covariance matrix $\mathbf{K}_s$ for all $s < t$ increases also linearly with the number of tasks with a cost $\mathcal{O}(M^3 t)$ which is much cheaper than full covariance matrix when M is not too large. Due to the Kronecker-factored Laplace approximations the complexity only depends on the number of factors (we make use of the backpack package by Dangel et al. (2020)). Let C be the number of class, D the number of pair data points and H_l the size of the l-th hidden layer, therfore we compute the eigendecomposition of all Kronecker factors in $\mathcal{O}(D^3C^3 + \sum_l H_l^3)$ for each task but only keep the eigenvalues necessary to compute equation 3.6.

5. Related Work

Continual Learning problem is referred as a model forgets how to solve earlier tasks. Extensive work has been recently made in this direction relies of regularization of the parameters using the previous posterior. Elastic Weight Consolidation (EWC) Kirkpatrick et al. (2017) proposes Laplace

and Fisher Matrix approximations while Variational Continual Learning (VCL) makes the recursive approximate posterior explicit proposed by Nguyen et al. (2018) and Swaroop et al. (2019a). Both of which make model parameters adaptive to new tasks while regularising weights by prior knowledge from the earlier tasks. Similar methods to Variational Continual Learning have been proposed and referred to as functional regularisation for Continual Learning, which leverages the Gaussian Process to construct an approximate posterior belief over the underlying task-specific function. A posterior belief is then constructed in the optimization step as a regulariser to prevent the model from deviating from the previous tasks. Recently this phenomenon has been investigated in the context of neural networks as catastrophic forgetting. Titsias et al. (2020) proposed to regularize neural networks using Gaussian Processes properties in the functional space. The idea is based on a regularisation in a functional space using sparse GP's. However this method scales very poorly and for two major reasons. First, in a Continual Learning setting the data is not all available at once, it is impossible to obtain unbiased stochastic gradient. Second, when task boundary is unknown, new classes may appear during training and old classes may never be seen again. This could lead to *Catastrophic Forgetting*. To address this issue, Pan et al. (2021) presented a regularization technique with memorable past examples. This method uses the GP formulation of *Deep Neural Network (DNN)* to obtain a weight-training method that exploits correlations among memorables examples in the function space. The key difference to Titsias et al. (2020) method is first, the kernel uses all the network weights to *help* the learning phase specially in the early stages. Second, they introduce an adaptive prior that forces the mean to be close to the past mean. Third, *inducing points* are replaced by *memorable past example* which are more informative and are much cheaper to compute. This method may improve the scalability issue since it does not require a separate optimisation problem. That is the computation procedure only require a forward-pass through the network followed by selecting the top examples.Bayesian Neural Network based on *Laplace-GGN* approximation presents a good alternative, because it uses a linearized model for posterior inference that can effectively resolves common underfitting problems of the Laplace approximation and enables alternative inference schemes for BNNs in function space using Gaussian Processes.

6. Results

6.1. Adaptative posterior to reduce *Catastrophic forgetting*

The figure (1) demonstrates how memorable examples improve accuracy across tasks on a toy dataset. The red dots represent the most informative subset of data examples lying in the decision boundary. That is, we see after training on the second task, the new network outputs are forced to maintain the same prediction on the memorable past examples as the previous network, as new tasks arrive, we see a clear separation, this can be interpret as a reduction of *catastrophic Forgetting*. In contrast, the bottom representation uses a damped KFAC kernel GP, we see when the number of tasks is high, the neural network adapts quite easily to make a clear separation across classes. To study the effect of *out-of-distribution* we use of the kernel obtained with deep Neural Network to GP to interpret and visualize such correlations. We compute the kernel for classes outside of the training dataset using Laplace-GGN. In the figure (2) we train on the Binary MNIST-dataset and visualize the predictive mean and kernel on all 10 classes denoted by differently colored regions on the y-axis. We illustrate the kernel and the predictive mean for the Laplace-GGN approximation. We see in the kernel that examples with same class labels are correlated. Since its only trained on binary MNIST (digits 0 and 1), the correlations are low for the out-of-class samples, the model shows that it cannot make overconfident predictions.

6.2. Weight regularisation versus functional regularization for uncertainty quantification

In this section, we visualize the GP kernel and predictive distribution for deep neural network trained on CIFAR. We consider Split CIFAR because it is a more challenging benchmark than MNIST, it consists of 6 tasks. The first task is the full CIFAR-10 dataset, followed by 5 tasks, each consisting of 10 consecutive classes from CIFAR-100. We run each method 5 times. We use multi-head CNN with 4 convolutional layers, then 2 dense layers with dropout. The number of inducing points are set in to 200 points, and we run each method 5 times. We consider two baselines, the first baseline consists of networks trained on each task separately. Doing so the training cannot profit from forward transfer from other tasks, and sets a lower limit which we must outperform. The second baseline consists of training all tasks jointly and may reveal better accuracy results. The results are summarised in table 1. We run all methods 5 times and report the mean and standard error. For baselines, we train from scratch on each task and jointly on all tasks, we achieve an accuracy of our Stream KFAC without and with damping with 75.6% and 75.9% respectively. The EWC method Kirkpatrick et al. (2017) achieve 71.6% and VCL Swaroop et al. (2019b) is 67.4%. We see our results outperforms the weight regularization techniques but lower than Pan et al. (2021) FROMP. Finally, we analyse the forward and backward transfer for our method and compared to benchmarks. Forward transfer means the accuracy on the current tasks increases as number of past tasks increases,

Table 1. Method Evaluation on Split-CIFAR

METHOD	ACCURACY	FORWARD-T	BACKWARD-T
S-KFAC	$75.6 \pm 0.9\%$	$2.8 \pm 0.3\%$	$-3.9 \pm 0.4\%$
S-KFAC-D	$75.9 \pm 3.1\%$	$2.9 \pm 0.9\%$	$-5.2 \pm 0.1\%$
FROMP	$76.2 \pm 0.4\%$	$6.1 \pm 0.7\%$	$-2.6 \pm 0.9\%$
VCL-I	$67.4 \pm 1.4\%$	$1.8 \pm 3.1\%$	$-9.2 \pm 1.8\%$
EWC	$71.6 \pm 0.9\%$	$0.17 \pm 0.9\%$	$-2.3 \pm 1.4\%$

while backward transfer means the accuracy on the previous tasks increases as more tasks are observed. The higher is better, for Split-CIFAR we do better than weight regularization technique such as VCL and EWC and slightly lower perforamces than FROMP, probably due to a the complexity of the prior underlying in our model.

To evaluate the *out-of-distribution*, we visualize the GP kernel and predictive distribution of the deep neural network. Our goal is to strengthen our understanding of the deep neural network performance on classification tasks applied on different tasks. The kernel evaluated at the *map* estimate $\boldsymbol{\kappa}_\star(\mathbf{x}, \mathbf{x}') := \boldsymbol{J}_\star(\mathbf{x})\boldsymbol{\Sigma}_\star\boldsymbol{J}_\star^T(\mathbf{x}')$ is an $NK \times NK$ dimensional matrix which is difficult to visualize. To be able to visualize it we compute the sum of the diagonal entries to get $N \times N$ matrix. The output corresponds to the sum of the kernel GPs for each class. The posterior mean is computed according to $\mathbf{f}_\star(\mathbf{x}; \mathbf{w}) \in \mathbb{R}^K$.

For multiclass-classification task, the covariance matrix corresponds to

$$\boldsymbol{\Lambda}_{\mathbf{w}_\star}(\mathbf{x})\boldsymbol{J}_{\mathbf{w}_\star}(\mathbf{x})\boldsymbol{\Sigma}_{\mathbf{w}_\star}\boldsymbol{J}_{\mathbf{w}_\star}(\mathbf{x})^T\boldsymbol{\Lambda}_{\mathbf{w}_\star}(\mathbf{x}) + \boldsymbol{\Lambda}_{\mathbf{w}_\star}(\mathbf{x}),$$

the first term can be interpreted as the *aleatoric uncertainty* (label noise) and the second term is interpreted as the *epistemic uncertainty* (model noise) Bishop (2006) . We illustrate in the figure (2) the predictive kernel on MNIST dataset, we clearly see the kernel is able to detect meaningful correlation in *out-of-distribution* example where is nearly absent in the full covariance case.

7. Conclusion

In this paper we proposed a functional regularisation technique in a Continual Learning setting where the main goal is to combat *Catastrophic Forgetting*. We showed adding a continual variational distribution to approximate the true distribution of the weight can help to identify a richer structure of the latent functions. Our method uses the Gaussian Process formulation to combine the weight space and the function space of the neural network. We assured that our method can borrow the optimization techniques of deep learning while learning the hidden structure in the function space and achieved the state-of-the-art performance in comparison with classical methods. We showed using the KFAC approximation *helps* to infer a richer structure of param-

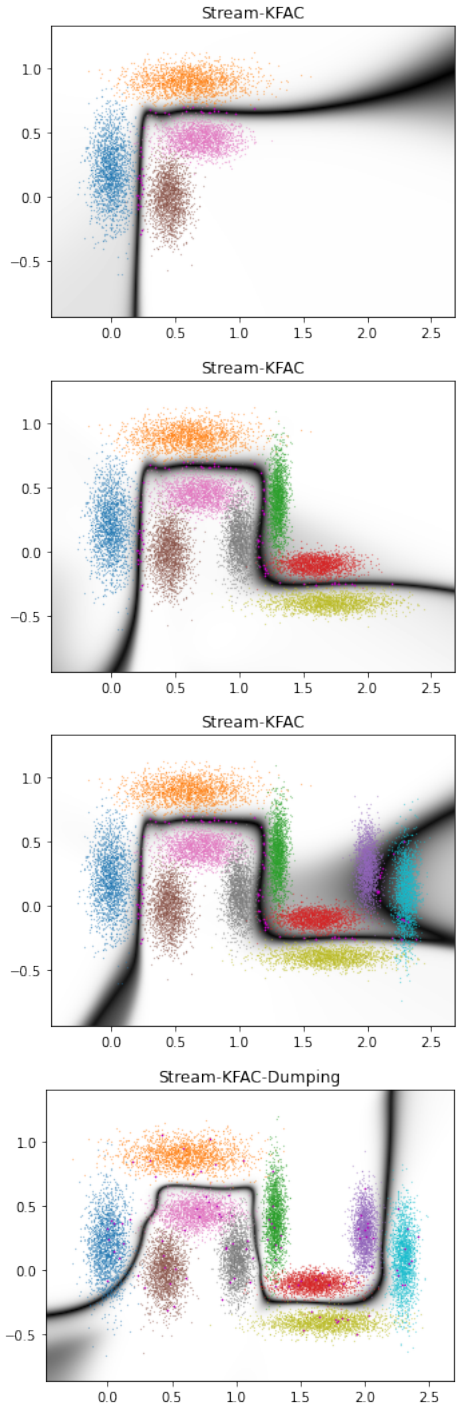

Figure 1. Decision boundary adapts the neural network output after task 1, task 3 and task 5 with average train accuracy of 99.34% and average train log-lileihood of 0.0220. The last figure (bottom) illustrates the Steram-KFAC method with *dumping*.

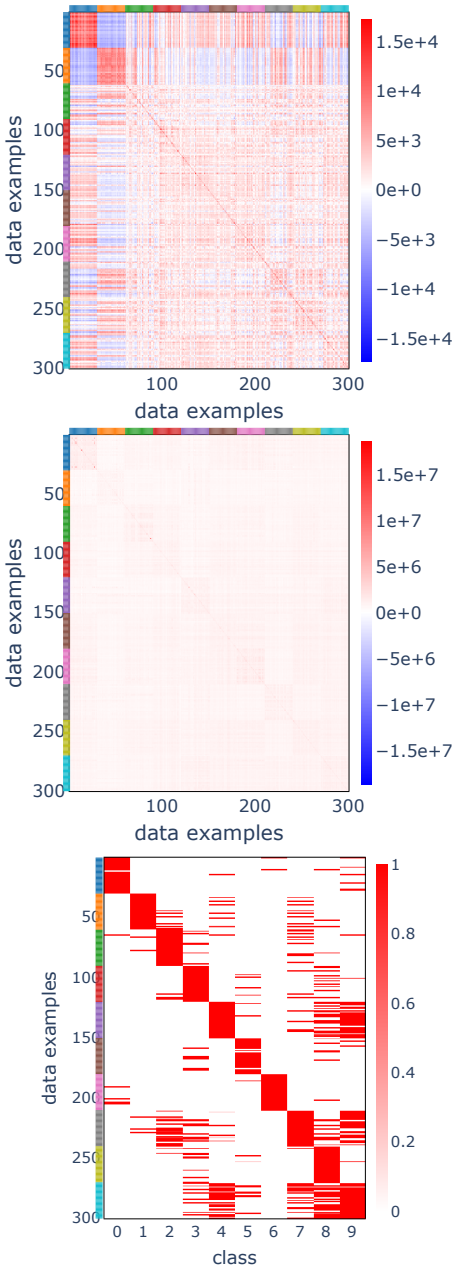

Figure 2. Predictive uncertainty with the Streaming KFAC with dumping (top) versus full covariance matrix method (middle) on binary-MNIST dataset, the KFAC-dumped kernel shows meaningful correlation in *out-of-distribution* while maintaining a high accuracy of class mean-prediction (bottom)

eter covariances which reduces the gap between the true and the corresponding variational posterior without altering the complexity cost. This formulation proved that we can outperforms classical weight space techniques and compete with functional space method even in large scale continual learning problem such as image classification.

References

Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006.

Bissiri, P. G., Holmes, C. C., and Walker, S. G. A general framework for updating belief distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):1103–1130, Feb 2016. ISSN 1369-7412. doi: 10.1111/rssb.12158. URL http://dx.doi.org/10.1111/rssb.12158.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks, 2015.

Botev, A., Ritter, H., and Barber, D. Practical Gauss-Newton optimisation for deep learning. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 557–565. PMLR, 06–11 Aug 2017. URL http://proceedings.mlr.press/v70/botev17a.html.

Bottou, L., Curtis, F. E., and Nocedal, J. Optimization methods for large-scale machine learning, 2018.

Bui, T. D., Nguyen, C., and Turner, R. E. Streaming sparse gaussian process approximations. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30, pp. 3299–3307. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/f31b20466ae89669f9741e047487eb37-Paper.pdf.

Cremer, C., Li, X., and Duvenaud, D. Inference suboptimality in variational autoencoders, 2018.

Dangel, F., Kunstner, F., and Hennig, P. Backpack: Packing more into backprop, 2020.

Immer, A., Korzepa, M., and Bauer, M. Improving predictions of bayesian neural nets via local linearization. In Banerjee, A. and Fukumizu, K. (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 703–711. PMLR, 13–15 Apr 2021. URL http://proceedings.mlr.press/v130/immer21a.html.

Khan, M. E., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., and Srivastava, A. Fast and scalable bayesian deep learning by weight-perturbation in adam, 2018.

Khan, M. E., Immer, A., Abedi, E., and Korzepa, M. Approximate inference turns deep networks into gaussian processes, 2020.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks, 2017.

Martens, J. and Grosse, R. Optimizing neural networks with kronecker-factored approximate curvature, 2020a.

Martens, J. and Grosse, R. Optimizing neural networks with kronecker-factored approximate curvature, 2020b.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. Variational continual learning, 2018.

Pan, P., Swaroop, S., Immer, A., Eschenhagen, R., Turner, R. E., and Khan, M. E. Continual deep learning by functional regularisation of memorable past, 2021.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

Swaroop, S., Nguyen, C. V., Bui, T. D., and Turner, R. E. Improving and understanding variational continual learning, 2019a.

Swaroop, S., Nguyen, C. V., Bui, T. D., and Turner, R. E. Improving and understanding variational continual learning, 2019b.

Titsias, M. Variational learning of inducing variables in sparse gaussian processes. *AISTATS*, 2009.

Titsias, M. K., Schwarz, J., de G. Matthews, A. G., Pascanu, R., and Teh, Y. W. Functional regularisation for continual learning with gaussian processes, 2020.

Zhang, G., Sun, S., Duvenaud, D., and Grosse, R. Noisy natural gradient as variational inference. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5852–5861. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/zhang18l.html.

Supplementary material

The derivation of the functional prior and its optimization

When the loss corresponds to the log of a probability distribution $\ell(\mathbf{y}, \mathbf{f}) := -\log p(\mathbf{y}|\mathbf{f}_w(\mathbf{x}))$ (e.g. Binary classification, multi-class classification) then the the solution is equal to the posterior distribution of a probabilistic model with the likelihood $p(\mathbf{y}|\mathbf{f}_\mathbf{w}(\mathbf{x}))$ and prior $p(\mathbf{w})$. The GPs can be interpreted in the form of canonical Gaussian measures **?**. The variational distribution $q_t(\mathbf{u}_t, \mathbf{f}_t)$ is optimized recursively for each new batch of data $\mathcal{D}_t$ given the previous variational distribution $q_{t-1}(\mathbf{u}_{t-1}, \mathbf{f}_{t-1})$[1]. The KL divergence has the following form

$$
\begin{aligned}
D_{KL}(\tilde{q}_t(\mathbf{u}_t|\mathcal{D}_{1:t})||p_t(\mathbf{u}_t|\mathcal{D}_{1:t})) &= -D_{KL}(\tilde{q}_t(\mathbf{u}_t)||p(\mathbf{u}_t)) - D_{KL}(\tilde{q}_t(\mathbf{u}_{t-1})||\tilde{q}_{t-1}(\mathbf{u}_{t-1})) \\
&+ D_{KL}(\tilde{q}_t(\mathbf{u}_{t-1})||p(\mathbf{u}_{t-1})) \\
&= \frac{1}{2}\left(\log|\mathbf{K}_{\mathbf{z}_t}| - \log|\mathbf{\Sigma}_t| - M_t + Tr\left(\mathbf{K}_{\mathbf{z}_t}^{-1}\mathbf{\Sigma}_t\right) + \boldsymbol{\mu}_t^T\mathbf{K}_{\mathbf{z}_t}^{-1}\boldsymbol{\mu}_t\right) \\
&+ \frac{1}{2}\Bigg(\log|\bar{\mathbf{K}}_{\mathbf{z}_{t-1}}| - \log|\mathbf{\Sigma}_{t-1}| - Tr\left(\mathbf{D}_{t-1}^{-1}\mathbf{K}_{t-1}\right) \\
&- \boldsymbol{\mu}_{t-1}^T\mathbf{\Sigma}_{t-1}^{-1}\boldsymbol{\mu}_{t-1} + 2\boldsymbol{\mu}_{t-1}\mathbf{\Sigma}_{t-1}^{-1}\boldsymbol{\kappa}_{\mathbf{z}_{t-1}\mathbf{z}_t}\boldsymbol{\mu}_t \\
&- \left(\boldsymbol{\kappa}_{\mathbf{z}_{t-1}\mathbf{z}_t}\boldsymbol{\mu}_t\right)^T\mathbf{D}_{t-1}^{-1}\left(\boldsymbol{\kappa}_{\mathbf{z}_{t-1}\mathbf{z}_t}\boldsymbol{\mu}_t\right)\Bigg),
\end{aligned}
$$

where $\mathbf{D}_t = \left(\mathbf{\Sigma}_t^{-1} - \mathbf{K}_{\mathbf{z}_t}^{-1}\right)^{-1}$. The weights are sampled according to $\mathbf{w} \sim q(\mathbf{w})$ and $\tilde{q}_t(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{m}_t(\mathbf{w}), \mathbf{K}_t(\mathbf{w}))$ with $\mathbf{m}_t(\mathbf{w})$ and $\mathbf{K}_t(\mathbf{w})$ denote the vector and the kernel matrix obtained by evaluating $\mathcal{GP}(\mathbf{m}_\mathbf{w}(\mathbf{z}), \boldsymbol{\kappa}_\mathbf{w}(\mathbf{z}, \mathbf{z'}))$ at memorable points selection. In the optimization step, we follow the proposed algorithm of **?**, where $\mathbf{w}_t$ is not sampled from $q_t(\mathbf{w})$ but set equal to $\boldsymbol{\mu}$ such that $\mathbf{w}_t = \boldsymbol{\mu}_t$. When memorables examples do not change within tasks, then $\boldsymbol{\kappa}_{\mathbf{z}_{t-1}\mathbf{z}_t} = \mathbf{I}$ and $\mathbf{K}_{\mathbf{z}_{t-1}} = \mathbf{K}_{\mathbf{z}_t}$. The functional regulariser simplifies

$$
\begin{aligned}
-\frac{1}{2}\Bigg(&Tr\left(\mathbf{K}_{\mathbf{z}_t\mathbf{z}_t}^{-1}\right) + \boldsymbol{\mu}_t^T\mathbf{K}_{\mathbf{z}_t\mathbf{z}_t}^{-1}\boldsymbol{\mu}_t \\
&+ \boldsymbol{\mu}_{t-1}^T\mathbf{\Sigma}_{t-1}^{-1}\left(\boldsymbol{\mu}_{t-1} - 2\boldsymbol{\mu}_t^T\right)\mathbf{D}_{t-1}^{-1}\boldsymbol{\mu}_t\Bigg) + cst.
\end{aligned}
$$

The distributions are defined as

$$
\begin{aligned}
\tilde{q}_t(\mathbf{u}_t) &= \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) \\
p(\mathbf{u}_t) &= \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{z}_t\mathbf{z}_t}) \\
\tilde{q}_t(\mathbf{u}_{t-1}) &= \mathcal{N}\left(\boldsymbol{\kappa}_{\mathbf{z}_{t-1}\mathbf{z}_t}\boldsymbol{\mu}_t, \mathbf{K}_{\mathbf{z}_{t-1}\mathbf{z}_{t-1}}\right) \\
\tilde{q}_{t-1}(\mathbf{u}_{t-1}) &= \mathcal{N}(\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}) \\
p(\mathbf{u}_{t-1}) &= \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{Z}_{t-1}\mathbf{Z}_{t-1}}),
\end{aligned}
$$

where $\tilde{q}_t(\mathbf{u}_{t-1}) = \int p(\mathbf{u}_{t-1}|\mathbf{u}_t)\tilde{q}_t(\mathbf{u}_t)d\mathbf{u}_t$.

Approximation of the Functional Regularizer

We convert the weight space integral by a function space integral using a change of variable $\mathbf{f} = \mathbf{X}\mathbf{w}$ with a matrix $\mathbf{X}$ and a given function $\mathbf{h}(\mathbf{f})$, we can therefore replace the weight space integral as the function space integral as follow

$$
\int h(\mathbf{X}\mathbf{w})\mathcal{N}(\mathbf{w}|\boldsymbol{\mu}, \mathbf{\Sigma})d\mathbf{W} = \int \mathbf{h}(\mathbf{f})\mathcal{N}(\mathbf{f}|\mathbf{X}\boldsymbol{\mu}, \mathbf{X}\mathbf{\Sigma}\mathbf{X}^T)d\mathbf{f}.
$$

Since $\log q_{t-1}(\mathbf{w})$ cannot be always expressed as a function of $\mathbf{f} = \boldsymbol{J}_{w_t}\mathbf{w}$, this gives

$$
\mathbb{E}_{q(\mathbf{w})}\left[\log\frac{q_{t-1}(\mathbf{w})}{p(\mathbf{w})}\right] \approx \mathbb{E}_{\tilde{q}_\mathbf{w}(\mathbf{f})}\left[\log\frac{\tilde{q}_{\mathbf{w}_{t-1}}(\mathbf{f})}{p(\mathbf{f})}\right].
$$

[1]For clarity of notation $\tilde{q}_{\mathbf{w}_t}(\mathbf{u}) := \tilde{q}_t(\mathbf{u})$