Variational Laplace in Deep Generative Network for Continual Learning

Nadhir Vincent Hassen[*]
Department of Mathematics and Industrial Engineering
Polytechnique Montreal, Quebec, Canada
`nadhir.hassen@umontreal.ca`

Abstract

One major challenge in Continual Learning is that the model forgets how to solve earlier tasks commonly known as *Catastrophe 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. In this work we address this issue without alternating the complexity cost. We propose a Laplace-Gaussian Newton approximation to combine local parametrization and function space learning. This approach finds the mode of the posterior and apply full covariance Gaussian posterior approximation centered on the mode and make the connection to GPs model. By using a Gaussian Process formulation of deep networks, we augment the model to an adaptive variational approximation to account the streaming nature of the data, our approach enables training in weight-space and a regularisation in functional space in a streaming fashion. This formulation enables to resolve common underfitting problem of Laplace approximation. We demonstrate our approach in a Continual Learning setting by combatting *Catastrophe Forgetting* due to a generalization to non- Gaussian likelihoods. 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.

1 Introduction

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 $\mathbf{w}_{t-1}$ are given and can be obtained through data training from task $t-1$. Titsias (2009) proposed $R(\mathbf{w}) = -\log p(\mathbf{w}|\mathcal{D}_{1:t-1}) \approx q_{t-1}(\mathbf{w}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ 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 of

[*]Affiliation: Mila Quebec

the *Mean-field-Variational Inference* take the following form

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

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 regularizer 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 is able to remember old tasks by selectively slowing down learning on the weights important for those tasks, we call our approach *Streaming Variational Laplace*. 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* 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

$$\mathbf{w}_{\text{MAP}} = \arg\max_{\mathbf{w}} \ell(\mathbf{w},\mathcal{D}) = \arg\max_{\mathbf{w}} \sum_{i=1}^{N} \log p(\mathbf{y}_i|\mathbf{f}(\mathbf{x}_i,\mathbf{w})) + \log p(\mathbf{w}),$$

where

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

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 and Grosse (2020) 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 and Grosse (2020) 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

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

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 $q(\mathbf{w}) = \mathcal{N}(\mathbf{w}_{\text{MAP}}, \mathbf{\Sigma}_{ggn})$, with

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

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 $\mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x}, \mathbf{y}) := \nabla^2_{\mathbf{ff}} \ell(\mathbf{y}, \mathbf{f})$ which depend on $\mathbf{x}$ and $\mathbf{y}$. Let's consider a probabilistic neural network, we define a likelihood function as a Bernoulli $p(\mathbf{x}) := \mathbf{g}_{\text{lin}}^{-1}(\mathbf{x}; \mathbf{w}) = \sigma(\mathbf{f}(\mathbf{x}; \mathbf{w}))$ and the labels $y \in \{0, 1\}$. Following Rasmussen and 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

$$\mathbf{m}_{\mathbf{w}_\star}(\mathbf{x}) := p(\mathbf{x}), \quad \boldsymbol{\kappa}_{\mathbf{w}_\star}(\mathbf{x}, \mathbf{x}') := \mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x}, y) \boldsymbol{J}_{\mathbf{w}\star}(\mathbf{x}) \mathbf{\Sigma}_{\text{ggn}\star} \boldsymbol{J}_{\mathbf{w}\star}(\mathbf{x}')^T \mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x}, y), \tag{2.2}$$

the predictive posterior is

$$\hat{p}(\mathbf{y}|\mathbf{x}, \mathcal{D}) = \mathcal{N}(\mathbf{y}|p(\mathbf{x}), \mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x}, y) \boldsymbol{J}_{\mathbf{w}\star}(\mathbf{x}) \mathbf{\Sigma}_{\text{ggn}\star} \boldsymbol{J}_{\mathbf{w}\star}(\mathbf{x}')^T \mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x}, y)) \tag{2.3}$$

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

$$\mathbf{f}_\star | \mathbf{x}_\star, \mathcal{D} \sim \mathcal{N}\left(\mathbf{f}(\mathbf{x}_\star; \mathbf{w}_\star), \sigma_\star^2\right), \quad \text{with } \sigma_\star^2 = \mathbf{K}_{\mathbf{x}_\star \mathbf{x}_\star} - \mathbf{K}_{\mathbf{x}_\star \mathbf{X}} \left(\mathbf{K}_{\mathbf{XX}} + \mathbf{\Lambda}_{\mathbf{XX}}^{-1}\right)^{-1} \mathbf{K}_{\mathbf{X}\mathbf{x}_\star},$$

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. In the Appendix.B we give the full derivation and we extend it to the case of multiclass classification. 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 *Catastrophe 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 *Catastrophe 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

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

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

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

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

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

where $\tau > 0$ is a tempering parameter. The functional regularizer in the rhs of the above equation uses an approximation to convert the weight space into the function space similar to the *functional prior* employed by Pan et al. (2021) *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 regularizer is given in the *Appendix.C*. 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 [2].

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

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 the parameters updates take the following form

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

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

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

For large neural network, optimizing $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ can be very expensive, we can make extra assumptions to reduce the computational cost. First, we assume that the Jacobians do not change significantly, so we can ignore the derivative with respect to $\mathbf{K}_t(\mathbf{w})$ and only use $\mathbf{m}_t(\mathbf{w})$ for all inducing points (memorable examples) $\mathbf{z}_t$. Second, the full $\mathbf{K}_{t-1}$ can be factorised across tasks and reduced to a block-diagonal matrix $\mathbf{K}_{t-1,s}$ for all s tasks. Second, we use a diagonal covaraince matrix $\boldsymbol{\Sigma}$ which correspond to a mean-field approximation reducing the cost of inversion. Then the loss function over tasks has the following form

$$\max_{\mathbf{w}} -N\mathbb{E}_{q(\mathbf{w})}\left[\bar{\ell}_t(\mathbf{y}, \mathbf{f}_{\mathbf{w}}(\mathbf{x}))\right] - \tau \sum_{s=1}^{t-1} \frac{1}{2}\Bigg(\mathbf{m}_{t,s}^T \mathbf{K}_{t,s}^{-1}\mathbf{m}_{t,s} + \mathbf{m}_{t-1,s}^T \mathbf{K}_{t-1,s}^{-1}\left(\mathbf{m}_{t-1,s} - 2\mathbf{m}_{t,s}\right)\mathbf{D}_{t-1,s}^{-1}\mathbf{m}_{t,s}\Bigg) + cst, \tag{3.5}$$

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 memorable examples. The gradient of the predective loss in equation (3.5) corresponds to

$$N\nabla_{\mathbf{w}}\bar{\ell}_t(\mathbf{w}) - \tau\sum_{s=1}^{t-1}\left(\nabla_{\mathbf{w}}\mathbf{m}_{t,s}(\mathbf{w})\mathbf{K}_{t,s}^{-1} + \nabla_{\mathbf{w}}\mathbf{m}_{t,s}(\mathbf{w})\mathbf{K}_{t-1,s}^{-1}\left(\mathbf{m}_{t-1,s} - 2\mathbf{m}_{t,s}\right)\mathbf{D}_{t-1,s}^{-1}\right),$$

where $\nabla_{\mathbf{w}}\mathbf{m}_{t,s}(\mathbf{w}) = \nabla_{\mathbf{w}}\left[\sigma\left(\mathbf{f}_{\mathbf{w}}(\mathbf{x})\right]\right. = \boldsymbol{\Lambda}_{\mathbf{w}}(\mathbf{x})\boldsymbol{J}_{\mathbf{w}}(\mathbf{x})^T$. We see clearly if we had to use the full training data it can be very costly since the computation of $\boldsymbol{J}_{\mathbf{w}}(\mathbf{x})$ require an evaluation over all data points for all tasks $s < t$. Applying the sparse variational GP reduce dramatically the cost of computation. More specifically, for a binary classification task with a cross-entropy loss and a sigmoid function $\sigma(\mathbf{f}_{\mathbf{w}}(\mathbf{x}))$, the $P-$jacobian vector corresponds to $\boldsymbol{J}_{\mathbf{w}}(\mathbf{x}) = \nabla_{\mathbf{w}}\mathbf{f}_{\mathbf{w}}(\mathbf{x})^T$ and the scalar noise precision is $\boldsymbol{\Lambda}_{\mathbf{w}}(\mathbf{x}) = \nabla^2_{\mathbf{ff}}\ell(\mathbf{y}, \mathbf{f}) = \sigma\left(\mathbf{f}_{\mathbf{w}}(\mathbf{x})\right)\left(1 - \sigma(\mathbf{f}_{\mathbf{w}}(\mathbf{x})\right)$. The final form of the predictive mean and covariance are

$$\mathbf{m}_{t,s} = \sigma(\mathbf{f}_{\mathbf{w}}(\mathbf{x})), \quad \mathbf{K}_{t,s}(\mathbf{w}) = \boldsymbol{\Lambda}_{\mathbf{w}}(\mathbf{x})\left[\boldsymbol{J}_{\mathbf{w}}(\mathbf{x})\boldsymbol{Diag}(\boldsymbol{\Sigma}_t)\boldsymbol{J}_{\mathbf{w}}(\mathbf{x})^T\right]\boldsymbol{\Lambda}_w(\mathbf{x}), \tag{3.6}$$

where $\boldsymbol{Diag}$ denote the diagonal matrix, In the multi-class classification case $\mathbf{f}$ is $K-1$ length output of the neural network and $\boldsymbol{J}_{\mathbf{w}_\star}$ becomes a $(K-1)\times P$ Jacobian matrix and $\boldsymbol{\Lambda}_{\mathbf{w}}(\mathbf{x})$ becomes a $(K-1)(K-1)$ matrix.

The sparse variational form of this functional regulariser has two advantages. First it provides a scalable computation on large neural network and second provide an analytical solution. After training on a specific task, we select a set of few memorable examples 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 *Catastrophe Forgetting*. 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.

4 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. (2017a) 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 *Catastrophe 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.

5 Results

5.1 Adaptative posterior to reduce *Catastrophe forgetting*

The figure 1 demonstrates how memorable examples improve accuracy accross 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 *catastrophe Forgetting*. In contrast, the last figure in the bottom right uses a diagonal kernel GP, we see when the number of tasks is high, the neural network struggles 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.

5.2 Weight regularisation versus functional regularization for uncertainity quantification

In this section, we visualize the GP kernel and predictive distribution for DNNs trained on CIFAR and MNIST. 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 75.6% and 74.3%), respectively. EWC is 71.6% and VCL is 67.4%. We see our results outperforms the weight regularization techniques but lower than 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, while backward transfer means the accuracy on the previous tasks increases as more tasks are observed. The higher is better, for Split-CIFAR and Split-MNIST in *Appendix.E* we do better than weight regularization technique such as VCL

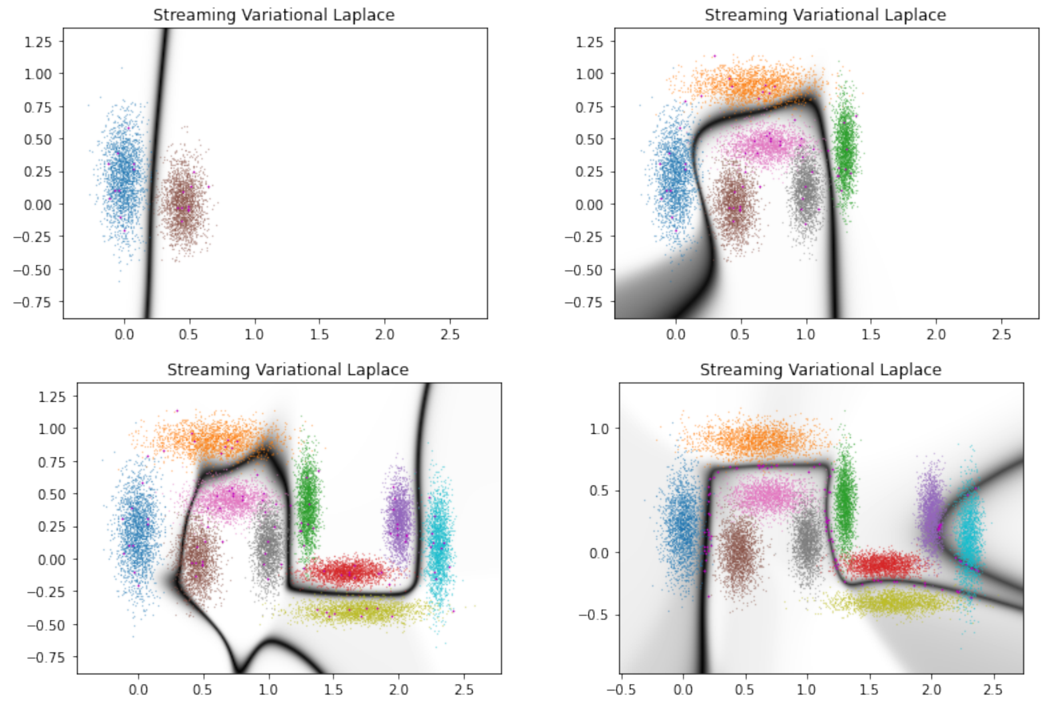

Figure 1: Decision boundary adapts the neural network output after task 1, task 3 and task 5 with average train accuracy of 99.34% and average train log-lileihood of 0.0220. The last figure (bottom right) illustrates the Laplace-GGN method with a diagoal kernel.

Table 1: Method Evaluation on Split-CIFAR

Method	Accuracy	Forward transfer	Backward transfer
Stream-Laplace-Full	$75.6 \pm 0.9\%$	$2.8 \pm 0.3\%$	$-3.9 \pm 0.4\%$
Stream-Laplace-Diag	$74.3 \pm 2.3\%$	$2.5 \pm 0.9\%$	$-5.4 \pm 0.3\%$
FROMP Pan et al. (2021)	$76.2 \pm 0.4\%$	$6.1 \pm 0.7\%$	$-2.6 \pm 0.9\%$
VCL-improved Swaroop et al. (2019b)	$67.4 \pm 1.4\%$	$1.8 \pm 3.1\%$	$-9.2 \pm 1.8\%$
EWC Kirkpatrick et al. (2017b)	$71.6 \pm 0.9\%$	$0.17 \pm 0.9\%$	$-2.3 \pm 1.4\%$

and EWC and slighly lower perforamces than FROMP, probably due to a the compexity of the prior underlying in our model.

To evaluate the *evidence uncertainty*, 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 $\boldsymbol{\kappa}_\star(\mathbf{x}, \mathbf{x}') := \boldsymbol{J}_\star(\mathbf{x})\boldsymbol{\Sigma}_\star \boldsymbol{J}_\star^T(\mathbf{x}')$ 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, as in equation (7.2), 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 (3) the decomposition of the two source of uncertainity on the predictive variance. After training on 5 tasks, this shows that the uncertainty of the model is low (left) and the label noise is high (right). Meaning that the model is unable to flexibly model the data and instead it can just explains it with high label noise. The estimated aleatoric uncertainty is much higher than the epistemic uncertainty, implying that the model is flexible enough and make accurate predictions.

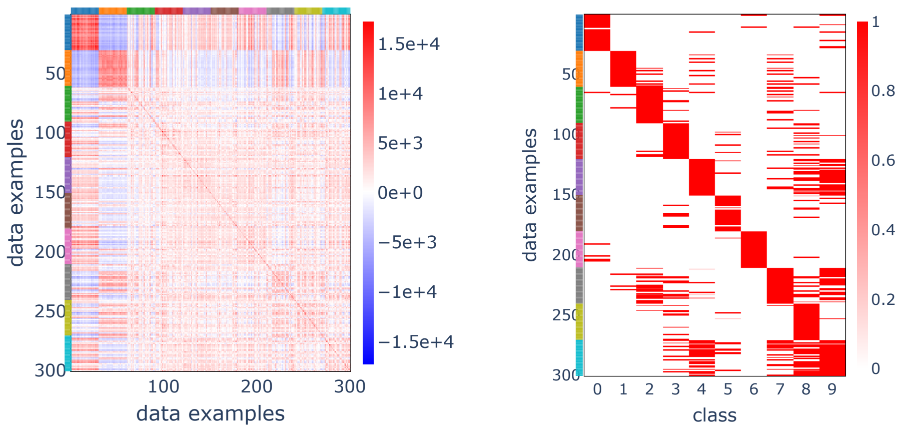

Figure 2: Deep-Neural Network kernel with a *mean-field-variational-inference* method as in equation (1.1) on binary-MNIST dataset, (left) The kernel illustarte the uncertain behaviour of out-of-class predictive while mainting a high accuracy of class mean-prediction (right)

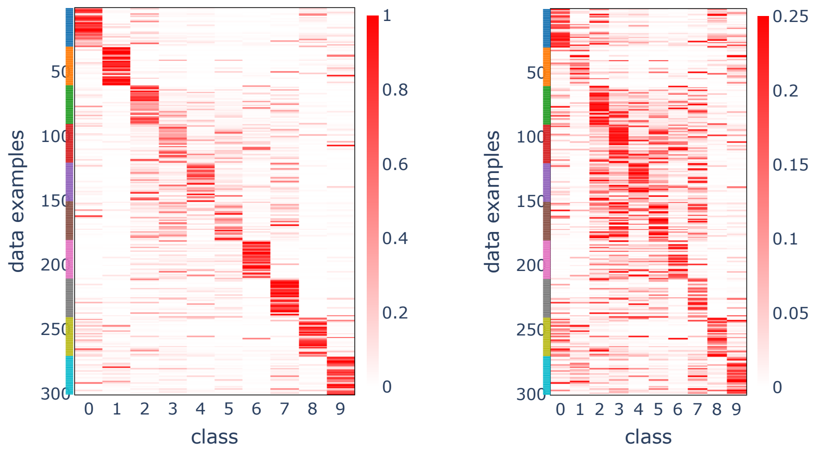

Figure 3: Variational Lapalce Kernel after 5 tasks on CIFAR dataset: Epistemic uncertainty (left) and aleatoric uncertainty (right)

6 Conclusion

In this paper we proposed a functional regularisation technique in a Continual Learning setting where the main goal is to combat *Catastrophe 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.

However there is some directions we can consider to improve the scalability issue. The Bayesian neural network is computational intensive due to the computation of the covariance matrix function of the GP. As future work we can consider using the Kronecker factorization approximations to speed up computation and reduce substantially the computational complexity without altering the quality of the approximation. In addition, there is some questions to explore, do we have better techniques to detect task boundaries and uncertainty quantification in the Bayesian neural network framework? These questions can be answered by considering additional relaxation of some assumptions done in this work and could be considered as future work.

References

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

Bissiri, P. G., Holmes, C. C., and Walker, S. G. (2016). A general framework for updating belief distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):1103–1130.

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

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

Bui, T. D., Nguyen, C., and Turner, R. E. (2017). Streaming sparse gaussian process approximations. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30, pages 3299–3307. Curran Associates, Inc.

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

Immer, A., Korzepa, M., and Bauer, M. (2021). Improving predictions of bayesian neural nets via local linearization. In Banerjee, A. and Fukumizu, K., editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 703–711. PMLR.

Jacot, A., Gabriel, F., and Hongler, C. (2020). Neural tangent kernel: Convergence and generalization in neural networks.

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

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

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. (2017a). Overcoming catastrophic forgetting in neural networks.

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. (2017b). Overcoming catastrophic forgetting in neural networks.

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

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

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

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

Shi, J., Khan, M. E., and Zhu, J. (2019). Scalable training of inference networks for gaussian-process models.

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

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

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

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

7 Appendix

7.1 Appendix.A1: From GLM to Gaussian Processes

That is we can convert the Bayesian GLM in weight space into the function space with a particular kernel. Following Rasmussen and Williams (2006), the corresponding log-joint of equation (**??**). For a prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{m}_0, \mathbf{S}_0)$, the mean and the covariance function is given by

$$\begin{aligned}\mathbf{m}(\mathbf{x}) &= \mathbb{E}_{p(\mathbf{w})}[\mathbf{f}_{\text{lin}}^{\mathbf{w}_\star}(\mathbf{x}, \mathbf{w}_\star)] = \mathbf{f}_{\text{lin}}^{\mathbf{w}_\star}(\mathbf{x}; \mathbf{m}_0)\\ \boldsymbol{\kappa}(\mathbf{x}, \mathbf{x}') &= Cov_{p(\mathbf{w})}\left[\mathbf{f}_{\text{lin}}^{\mathbf{w}_\star}(\mathbf{x}; \mathbf{w}), \mathbf{f}_{\text{lin}}^{\mathbf{w}_\star}(\mathbf{x}'; \mathbf{w})\right]\\ &= \boldsymbol{J}(\mathbf{x}; \mathbf{w}_\star)\mathbf{S}_0\boldsymbol{J}(\mathbf{x}^\star; \mathbf{w}_\star)^T.\end{aligned}$$

The approximate inference in this GP model is obtained in closed-form. For the regression case, we denote the approximate posterior as $q(\mathbf{f})$. For a new location $\mathbf{x}_\star$ the Lapalce-GGN-approximation evaluated at $\mathbf{w}_\star = \mathbf{w}_{\text{MAP}}$ is given by

$$\mathbf{f}_\star|\mathbf{x}_\star, \mathcal{D} \sim \mathcal{N}\left(\mathbf{f}(\mathbf{x}_\star; \mathbf{w}_\star), \sigma_\star^2\right), \quad \text{with } \sigma_\star^2 = \mathbf{K}_{\mathbf{x}_\star\mathbf{x}_\star} - \mathbf{K}_{\mathbf{x}_\star\mathbf{x}_N}\left(\mathbf{K}_{\mathbf{x}_N\mathbf{x}_N} + \boldsymbol{\Lambda}_{NN}^{-1}\right)^{-1}\mathbf{K}_{\mathbf{x}_N\mathbf{x}_\star},$$

where $\mathbf{K}_{\mathbf{x}_\star\mathbf{x}_n}$ denotes the kernel evaluated between $\mathbf{x}_\star$ and the N training points, and $\boldsymbol{\Lambda}_{NN}$ is a diagonal matrix with entries $\boldsymbol{\Lambda}(\mathbf{y}_n; \mathbf{f}_n)$. This formulation can be extended to the multi-output setting. Finally, the predictive GP is given by

$$p_{\mathcal{GP}}(\mathbf{y}|\mathbf{x}, \mathcal{D}) = \mathbb{E}_{q(\mathbf{f})}[p(\mathbf{y}|\mathbf{f})],$$

this can be evaluated by Monte-Carlo sampling with $\mathbf{f} \sim q(\mathbf{f})$.

7.2 Appendix.A2: The detailed of the Laplace-GGN-approximation

We can obtain a distribution $q(\mathbf{w})$ that is equal to the posterior of the following linear model

$$\tilde{\mathbf{y}} = \boldsymbol{J}(\mathbf{x})\mathbf{w} + \boldsymbol{\varepsilon}, \quad \text{with} \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}(\mathbf{x}, \mathbf{y})^{-1}) \quad \text{and} \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{S}_0 = \delta^{-1}\mathbf{I}_P), \tag{7.1}$$

where the observations are defined $\tilde{\mathbf{y}}_i := \boldsymbol{J}(\mathbf{x}_i)\mathbf{w}_\star - \boldsymbol{\Lambda}(\mathbf{x}_i, \mathbf{y}_i)^{-1}\boldsymbol{r}(\mathbf{x}_i, \mathbf{y}_i)$, the residuals are defined $\boldsymbol{r}(\mathbf{x}, \mathbf{y}) := \nabla_{\mathbf{f}}\ell(\mathbf{y}, \mathbf{f})$. Using the properties of the multivariate Gaussian distribution, Jacot et al. (2020) obtained a GP regression model defined with $K \times K$ neural tangent kernel (NTK) such as

$$\tilde{\mathbf{y}} = \mathbf{f}_{\mathcal{GP}}(\mathbf{x}) + \boldsymbol{\varepsilon} \quad \text{with} \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}(\mathbf{x}, \mathbf{y})^{-1}) \quad \text{and } \mathbf{f}_{\mathcal{GP}}(\mathbf{x}) \sim \mathcal{GP}(\mathbf{0}, \delta^{-1}\boldsymbol{J}(\mathbf{x})\boldsymbol{J}(\mathbf{x}')).$$

This formulation enables to convert the neural network posterior from the weight space to the function space. Therefore we can benefit the computational efficiency and interpretabilities properties of a GP model.

In the case of a probabilistic neural network, where the likelihood function is a Bernoulli defined as $p(\mathbf{x}) := \sigma(\mathbf{f}_{\mathbf{w}}(\mathbf{x}))$ where σ denotes the sigmoid function and the labels $\mathbf{y} \in \{0, 1\}$. For outputs $\mathbf{y}$, the residual and the Hessian are defined as follow

$$\boldsymbol{r}(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}) - \mathbf{y} \quad \text{and } \boldsymbol{\Lambda}(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})\left(1 - p(\mathbf{x})\right).$$

Since the residuals are linear in $\mathbf{y}$ and the Hessian is independent of $\mathbf{y}$, we can rewrite $\mathbf{y}$ from the definition of $\tilde{\mathbf{y}}$ and $\boldsymbol{r}(\mathbf{x}, \mathbf{y})$ from equation (7.1), therefore we obtain

$$\mathbf{y} = \underbrace{p(\mathbf{x}) + \boldsymbol{\Lambda}(\mathbf{x}, \mathbf{y})\boldsymbol{J}(\mathbf{x})\left(\mathbf{w} - \mathbf{w}_\star\right)}_{\mathbf{f}_{\text{lin}}(\mathbf{x})} + \boldsymbol{\varepsilon} \quad \text{with } \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}(\mathbf{x}, \mathbf{y})) \quad \text{and } \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{S}_0 = \delta^{-1}\mathbf{I}_P).$$

Following Rasmussen and Williams (2006) we can recover the GP formulation with mean $\mathbf{m}(\mathbf{x})$ and covariance function $\boldsymbol{\kappa}(\mathbf{x}, \mathbf{x}')$:

$$\begin{aligned}\mathbf{m}(\mathbf{x}) :&= \mathbb{E}_{q(\mathbf{w})}\left[\mathbf{f}_{\text{lin}}(\mathbf{x})\right] = p(\mathbf{x}) + \boldsymbol{\Lambda}(\mathbf{x}, \mathbf{y})\boldsymbol{J}(\mathbf{x})\left(\boldsymbol{\mu} - \mathbf{w}_\star\right) = p(\mathbf{x}),\\ \boldsymbol{\kappa}(\mathbf{x}, \mathbf{x}') &:= \mathbb{E}_{q(\mathbf{w})}\left[\left(\mathbf{f}_{\text{lin}}(\mathbf{x}) - \mathbf{m}(\mathbf{x})\right)\left(\mathbf{f}_{\text{lin}}(\mathbf{x}') - \mathbf{m}(\mathbf{x}')\right)\right]\\ &= \boldsymbol{\Lambda}(\mathbf{x}, \mathbf{y})\boldsymbol{J}(\mathbf{x})\mathbb{E}_{q(\mathbf{w})}\left[(\mathbf{w} - \mathbf{w}_\star)(\mathbf{w} - \mathbf{w}_\star)^T\right]\boldsymbol{J}(\mathbf{x}')^T\boldsymbol{\Lambda}(\mathbf{x}', \mathbf{y}')\\ &= \boldsymbol{\Lambda}(\mathbf{x}, \mathbf{y})\boldsymbol{J}(\mathbf{x})\boldsymbol{\Sigma}\boldsymbol{J}(\mathbf{x}')^T\boldsymbol{\Lambda}(\mathbf{x}', \mathbf{y}').\end{aligned}$$

7.3 Appendix.B: Loss functions

- **The Binary Classification Loss** We use Lapalce Approximation to derive the Logistic loss. First, we need to assume that $y_i \in \{0,1\}$ follows a Bernouilli distribution and the loss function coresponds to a log probability distribution such as $\ell(\mathbf{y}, \mathbf{f_w}(\mathbf{x})) := -\log p(\mathbf{y}|\mathbf{h}(\mathbf{f_w}(\mathbf{x})))$ where $\mathbf{h}(.)$ is a link function. Therefore we can write the loss function as $\sigma(\mathbf{f}) - (1-\mathbf{y})\log(1-\sigma(\mathbf{f})) = -\mathbf{yf} + \log(1+e^{\mathbf{f}})$ where $\sigma(\mathbf{f}) := 1/\left(1+e^{-\mathbf{f}}\right)$ is the sigmoid function. The predictive distribution is given by

$$\hat{p}(\mathbf{y}|\mathbf{x}, \mathcal{D}) := \mathcal{N}\big(\mathbf{y}|\sigma(\mathbf{f_w}(\mathbf{x})), \mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x})\boldsymbol{J}_{\mathbf{w}_\star}(\mathbf{x})\mathbf{\Sigma}_{\mathbf{w}_\star}\boldsymbol{J}_{\mathbf{w}_\star}(\mathbf{x})^T\mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x}) + \mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x})\big),$$

with

$$\mathbf{\Sigma}_{\mathbf{w}_\star}(\mathbf{x})^{-1} := \boldsymbol{J}_{\mathbf{w}_\star}(\mathbf{x})^T\mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x})\boldsymbol{J}_{\mathbf{w}_\star}(\mathbf{x}) + \mathbf{S}_0$$

where $\mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x}) := \sigma(\mathbf{f}_{\mathbf{w}_\star}(\mathbf{x}))\left(1-\sigma\left(\mathbf{f}_{\mathbf{w}_\star}(\mathbf{x})\right)\right)$ is a scalar. When the model $\mathbf{y} = \mathbf{f}(\mathbf{x}) + \boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x}))$ we can formulate a GP-posterior as follow

$$\mathbf{m}_{\mathbf{w}_\star(\mathbf{x})} := \sigma(\mathbf{f}_{\mathbf{w}_\star}(\mathbf{x})), \quad \boldsymbol{\kappa}_{\mathbf{w}_\star}(\mathbf{x}, \mathbf{x}') := \mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x})\boldsymbol{J}_{\mathbf{w}_\star}(\mathbf{x})\mathbf{\Sigma}_{\mathbf{w}_\star}\boldsymbol{J}_{\mathbf{w}_\star}(\mathbf{x})^T\mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x}'),$$

we note here that we have an additional term $\mathbf{\Lambda}_{\mathbf{w}_\star}$ on both sides of the covariance. This is becaue of the nonlinearity in the loss function caused by the sigmoid link function.

- **The Multi-Class Classification Loss** When we consider more than two classes per task, we are in a multiclass setting, we need a different link function, usually we use a multinomial logit likelihood or commonly known as a softmax function. The loss is given by

$$\ell(\mathbf{y}, \mathbf{f}) = -\mathbf{y}^T\Phi(\mathbf{f}) + \log\left(1 + \sum_{k}^{K-1} e^{f_k}\right),$$

the $k-$th element of $\Phi(\mathbf{f})$ corresponds to $\frac{e^{f_j}}{1+\sum_k^{K-1} e^{f_k}}$, where K denote the number of category and $\mathbf{y}$ is a one-hot-encoding vector of size $K-1$. The length of the vector $\mathbf{f}$ is $K-1$ and represents the output of the neural network. The link function $\mathbf{h}(.)$ is the softmax mapping from $K-1$ length vector to a $K-1$ dimensional vector which its entries are in the interval $(0,1)$. The predective distribution can be written as follow

$$\hat{p}(\mathbf{y}|\mathbf{x}, \mathcal{D}) := \mathcal{N}\big(\mathbf{y}|\Phi(\mathbf{f_w}(\mathbf{x})), \mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x})\boldsymbol{J}_{\mathbf{w}_\star}(\mathbf{x})\mathbf{\Sigma}_{\mathbf{w}_\star}\boldsymbol{J}_{\mathbf{w}_\star}(\mathbf{x})^T\mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x}) + \mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x})\big), \tag{7.2}$$

with

$$\mathbf{\Sigma}_{\mathbf{w}_\star}(\mathbf{x})^{-1} := \boldsymbol{J}_{\mathbf{w}_\star}(\mathbf{x})^T\mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x})\boldsymbol{J}_{\mathbf{w}_\star}(\mathbf{x}) + \mathbf{S}_0,$$

where $\mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x}) := \Phi(\mathbf{f}_{\mathbf{w}_\star})\left(1-\Phi(\mathbf{f}_{\mathbf{w}_\star})\right)^T$ is a $(K-1)\times(K-1)$ matrix and $\boldsymbol{J}_{\mathbf{w}_\star}(\mathbf{x})$ is a $(K-1)\times P$ Jacobian matrix. In this case the mean function is a $K-1$ length matrix and the covariance function is a square matrix of size $K-1$. The predective distribution can be written as a GP-posterior as follow

$$\mathbf{m}_{\mathbf{w}_\star(\mathbf{x})} := \Phi(\mathbf{f}_{\mathbf{w}_\star}(\mathbf{x})), \quad \boldsymbol{\kappa}_{\mathbf{w}_\star}(\mathbf{x}, \mathbf{x}') := \mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x})\boldsymbol{J}_{\mathbf{w}_\star}(\mathbf{x})\mathbf{\Sigma}_{\mathbf{w}_\star}\boldsymbol{J}_{\mathbf{w}_\star}(\mathbf{x})^T\mathbf{\Lambda}_{\mathbf{w}_\star}(\mathbf{x}'), \tag{7.3}$$

7.3.1 Appendix.C: 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 $p(\mathbf{y}|\mathbf{f_w}(\mathbf{x}))$ and prior $p(\mathbf{w})$. The GPs can be interpreted in the form of canonical Gaussian measures Shi et al. (2019). 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 $q_{t-1}(\mathbf{u}_{t-1}, \mathbf{f}_{t-1})$[3]. The KL divergence has the following form

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

where $\mathbf{D}_t = \left(\boldsymbol{\Sigma}_t^{-1} - \mathbf{K}_{\mathbf{z}_t}^{-1}\right)^{-1}$. 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 Khan et al. (2020), 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

$$
\begin{aligned}
-\frac{1}{2}\Bigg(&Tr\left(\mathbf{K}_{\mathbf{z}_t\mathbf{z}_t}^{-1}\right) + \boldsymbol{\mu}_t^T\mathbf{K}_{\mathbf{z}_t\mathbf{z}_t}^{-1}\boldsymbol{\mu}_t \\
&+ \boldsymbol{\mu}_{t-1}^T\boldsymbol{\Sigma}_{t-1}^{-1}\left(\boldsymbol{\mu}_{t-1} - 2\boldsymbol{\mu}_t^T\right)\mathbf{D}_{t-1}^{-1}\boldsymbol{\mu}_t\Bigg) + cst. \qquad (7.4)
\end{aligned}
$$

The distributions are defined as

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

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

7.4 Appendix.D: 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

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

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

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

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

7.5 Appendix.E: Empirical results

Table 2: Method Evaluation on MNIST-Split

Method	Accuracy	Forward transfer	Backward transfer
Stream-Laplace-Full	$98.81 \pm 2.3\%$	$-2.1 \pm 0.1\%$	$-1.8 \pm 0.7\%$
Stream-Laplace-Diag	$97.3 \pm 1.4\%$	$-1.3 \pm 0.4\%$	$-0.4 \pm 0.5\%$
FROMP Pan et al. (2021)	$99 \pm 0.1\%$	$-0.07 \pm 0.05\%$	$-0.5 \pm 0.2\%$
VCL-improved Swaroop et al. (2019b)	$94.6 \pm 0.3\%$	$-2.4 \pm 3.5\%$	$-2.1 \pm 0.3\%$
EWC Kirkpatrick et al. (2017a)	$71.6 \pm 0.9\%$	$-0.14 \pm 0.8\%$	$-1.3 \pm 2.4\%$