Variance Guided Continual Learning in a Convolutional Neural Network Gaussian Process Single Classifier Approach for Multiple Tasks in Noisy Images

Mahed Javed*, Lyudmila Mihaylova* and Nidhal Bouaynaya**
*Department of Automatic Control & Systems Engineering, The University of Sheffield, Sheffield, UK
**Department of Electrical and Computer Engineering, Rowan University, New Jersey, USA
Email: mjaved1@sheffield.ac.uk, l.s.mihaylova@sheffield.ac.uk, bouaynaya@rowan.edu

Abstract—**This work provides a continual learning solution in a single-classifier to multiple classification tasks with various data sets. A Gaussian process (GP) is combined with a Convolutional Neural Network (CNN) feature extractor architecture (CNNGP). Post softmax samples are used to estimate the variance. The variance is characterising the impact of uncertainties and is part of the update process for the learning rate parameters. Within the proposed framework two learning approaches are adopted: 1) in the first, the weights of the CNN are deterministic and only the GP learning rate is updated, 2) in the second setting, prior distributions are adopted for the CNN weights. Both the learning rates of the CNN and the GP are updated. The algorithm is trained on two variants of the MNIST dataset, split-MNIST and permuted-MNIST. Results are compared with the Uncertainty Guided Continual Bayesian Networks (UCB) multi-classifier approach [1]. The validation shows that the proposed algorithm in the Bayesian setting outperforms the UCB in tasks subject to Gaussian noise image noises and shows robustness.**

Index Terms—**deep learning, Bayesian learning, classification, artificial intelligence, machine learning, continual learning**

I. Introduction

Continual learning is a sub-field of machine learning that aims to study the performance of deep neural networks (DNNs) on non-stationary datasets [2]. The past two decades of Artificial Intelligence (AI) witnessed a rise in algorithms that can learn from stationary datasets and perform with near or beyond human-level accuracy [3], [4]. A further study on the vulnerabilities of DNNs, in the form of erroneous predictions and sensitivity to imperceptible input changes, has led to research in robustness and interpretability [5], [6]. As a solution in cases of noises and adversarial attacks, an extensive study on Bayesian learning methods and adversarial learning has contributed to advances in explainable and robust AI [7], [8]. This paper takes a step further in this direction towards achieving Bayesian methods in both continual learning and adversarial robustness.

Neural networks face challenges such as catastrophic forgetting [9], a phenomenon in which a substantial drop in performance is observed when a network is presented with a new task. This happens both when the new task resumes a portion of the current operating dataset and when it switches to a completely new data set. Continual learning [2] aims at solutions able to deal with such forgetting challenges so that the network has a reliable and robust performance when the environment changes or conditions become uncertain. Most of the available solutions have substantial memory requirements, especially in the cases of per-parameter regularisation [1], or require dynamic storage in the case of replay methods [10].

Bayesian methods provide different ways to characterise the impact of uncertainties on the final solutions, including in classification tasks. This provides confidence measures in the predictions in DNNs [11]. It also gives an insight into interpretability and explainability in AI systems [6]. Recently, different approaches for uncertainty quantification were proposed aimed at assisting continual learning. These include Variational Continual Learning (VCL) networks [12] and Uncertainty Guided Continual Bayesian (UCB) networks [1]. Both VCL and UCB are Bayesian inference approaches able to perform continual learning. The difference in their implementation is that VCL updates its posterior distribution by simply multiplying the prior by the likelihood, while UCB uses a special class of DNNs that assigns prior distribution over weights. These solutions belong to the class of Bayesian neural networks (BNNs) [7].

Each of them has its pros and cons. The VCL can be trained on small representations of the dataset and is an inference approach that is easier to implement compared with the UCB. UCB adjusts the learning rates of the means of the posterior distributions based on the variance of each of the weights. Both the VCL and UCB use a multiple classifiers approach. This means that for each task, a separate classifier is learned. A study on a single classifier will fill in this research gap. This paper investigates the performance of a single classifier in a continual learning scenario where there are multiple classification tasks, with multiple classes. The same method for updating the learning rate parameters from the variance is used as in the UCB method [1].

Particularly, there are three questions that this paper investigates: 1) can a single-classifier outperform a multi-classifier, 2) does the difference in performance change with the number of tasks or the type of weight setting used and 3) which method is more robust in the presence of Gaussian noise.

For the choice of the single-classifier, this study proposes a joint framework of a convolutional network Gaussian process (CNNGP). The classifier is a Gaussian process (GP) with a convolutional neural network (CNN) feature extractor. The CNN is used for learned feature extraction. Using CNN only for training is not sufficient since it cannot output uncertainty alone. The CNN component consists of two convolutional layers and a single fully connected layer for feature extraction. The uncertainty is characterised by the post softmax sample variance and is used to update the learning parameters of the GP only (in deterministic weight setting) and for both CNN and GP (in Bayesian weight setting). It is important to note that in the first setting, the CNN weights are deterministic scalars. In the Bayesian setting, they are probabilistic. The joint framework is trained on both split-MNIST [13], permuted-MNIST [14], CIFAR-10 and CIFAR-100 [15]. Accuracy measures are used to evaluate performance alongside comparing run times between two settings of the proposed framework (deterministic and Bayesian) and with the UCB approach. The contributions of this work are as follows.

- a framework able to characterise the impact of uncertainties on the performance of a single-classifier and multi-classifier in continual learning is proposed,
- a GP is combined with a CNN feature extractor and provides the classification result in the process of continual learning,
- the efficiency of using either a deterministic or a Bayesian weights setting is investigated,
- the performance of proposed CNNGP framework is evaluated on both split-MNIST and permuted-MNIST datasets is thoroughly evaluated.

The rest of this paper is organised as follows. Sections II-IV deal with background information regarding BNNs and GPs. Section VI provides the full description of the proposed framework while Section VIII details the experiments. A discussion of results is provided in Section IX. Finally, the paper summarises the results in conclusion and discusses future work in Section X.

II. Bayesian Neural Networks

A simple DNN architecture can be represented as a function $\mathbf{f}$ that takes the input vector $\mathbf{X}$ and outputs the vector $\mathbf{Y}$. The input $\mathbf{X} = [x_1, x_2, \cdots, x_N]$ is a vector that contains a series of samples $x_n, n = 1,...,N$, where N denotes the sample size/ Each sample is drawn independently from the data distribution say $\mathscr{D} = (\mathbf{X}, \mathbf{Y})$. The term data 'set' and data 'vector' are often used interchangeably in literature. In the case of a set based representation, the input dataset takes the form $\mathbf{X} = \{x_n\}_{n=1}^{N}$. The hidden layers consist of a series of linear and non-linear activation performed with respect to the input $\mathbf{X}$, the weight $\mathbf{w}$ and the bias $\mathbf{b}$ vectors of the DNN, where

$$\mathbf{Y} = \mathbf{f}\ (\mathbf{X}, \mathbf{w},\ \mathbf{b}). \tag{1}$$

Given a sequence of layers, the outputs from the activation from one layer l become the inputs of the activation of the consecutive layer $l+1$, such that the expression of activation is given by

$$\mathbf{a}_i^{(l+1)} = b_i^{(l+1)}\big(\mathbf{1} + \sum_{j=1}^{N_j} w_{i,j}^{(l+1)}\mathbf{x}_j^{(l)}\big), \quad \text{where} \quad x_j^{(l)} = \phi(\mathbf{a}_j^{(l)}). \tag{2}$$

Here, $\mathbf{a}_i^{l+1}$ denotes the activation for the i^{th} node in layer $l+1$ connecting node j in the previous layer l for which the associated weights and biases are $w_{i,j}^{(l+1)}$ and $b_i^{(l+1)}$, respectively. The $\mathbf{1}$ denotes the unit vector. The activation $a_j^{(l)}$ of the j^{th} node in the past layer l undergoes a non-linear transformation through the function ϕ to obtain the output $x_j^{(l)}$. These are weighted and summed for all N_j nodes in the layer l. Common choices for ϕ include Rectified Linear Unit (ReLU) [16] in the form of (3) or the tanh function [3]. The approaches of this paper make common use of the ReLU activation, where

$$\phi(\mathbf{a}_j^{(l)}) \to x_j^{(l)} = \begin{cases} \mathbf{a}_j^{(l)} & \text{if } \mathbf{a}_j^{(l)} \geq 0 \\ 0 & \text{if } \mathbf{a}_j^{(l)} \leq 0. \end{cases} \tag{3}$$

In classic DNN architectures, the weights and biases are deterministic. This means, regardless of the number of runs, the output would always be a point estimate. This is different in the case of BNNs that assume a prior distribution of the weights [7]. A common choice of prior distribution is the Gaussian one and the weights $w_{i,j}^{(l+1)}$ of BNNs [7] can be calculated according to

$$w_{i,j}^{(l+1)} = \mu_j^{(l+1)} + \sigma_j^{(l+1)}\ \varepsilon \tag{4}$$

and $\varepsilon \sim \mathscr{N}(0,1)$ is a zero mean, unit variance random Gaussian variable. The terms $\mu_j^{(l+1)}$ and $\sigma_j^{(l+1)}$ refer to the mean and standard deviation of the BNN weights.

Training of DNNs involves simple backpropagation of gradients with respect to a loss function. There are different ways to train BNNs. Specifically, this work considers the training method Bayes by Backprop (BBB) [17]. BBB training is a variant of variational inference (VI) [18]. This forms the subject of the next subsection.

III. Variational Inference

The fundamental idea of VI is to convert the problem of Bayesian inference to an optimisation task. In the Bayesian framework, the objective is to compute the posterior distribution $P(\mathbf{w}|\mathscr{D})$ where

$$P(\mathbf{w}|\mathscr{D}) = \frac{P(\mathbf{w})P(\mathscr{D}|\mathbf{w})}{\int P(\mathscr{D}|\mathbf{w})P(\mathbf{w})}, \tag{5}$$

based on the prior $P(\mathbf{w})$ and likelihood function $P(\mathscr{D}|\mathbf{w})$. For brevity, the term $\mathbf{w}$ denotes the weights of the model (e.g. BNN) and $\mathscr{D}$ is the data. The immediate observable drawback of this approach is that the denominator represents the sum of all possible prior values along with the weights. For large architectures with billion of weight parameters, the denominator is intractable and computationally expensive. The evidence term in the denominator of (5) is often neglected.

VI applies a proxy distribution $q(\mathbf{w}|\theta)$, parameterised by θ, to best approximate the posterior distribution. This is performed by minimising the Kullback-Leibler divergence (KLD) between the variational posterior and the real posterior

$P(\mathbf{w}|\theta)$, [18], [19], where the optimal parameter vector is given by

$$\begin{aligned}\theta^* &= \underset{\theta}{\text{argmin}}\ \text{KL}\ [q(\mathbf{w}|\theta)\|P(\mathbf{w}|\theta)],\\ &= \underset{\theta}{\text{argmin}}\ \text{KL}\ [q(\mathbf{w}|\theta)\|P(\mathbf{w})] - \mathbb{E}_{q(\mathbf{w}|\theta)}[\log P(\mathscr{D}|\mathbf{w})].\\ &= \underset{\theta}{\text{argmin}}\ \mathscr{L}_{VI} \approx \sum_{i=1}^{Q} \log q(\mathbf{w}^{(i)}|\theta) - \log P(\mathbf{w}^{(i)}) - \log P(\mathscr{D}|\mathbf{w}^{(i)}).\end{aligned} \tag{6}$$

The BBB approach [17] is briefly described in the next subsection.

A. Bayes by Backpropagation

The optimal weights of the variational parameters θ^* are found by standard gradient descent until the variational loss $\mathscr{L}_{VI}$ is optimised. In the BBB approach, Monte Carlo samples are drawn from the variational posterior to approximate the exact cost, where $\mathbf{w}^{(i)}$ is the i^{th} sample and Q is the total runs. Gaussian processes, on the other hand, can be trained directly on the negative marginal log-likelihood term, $-\mathbb{E}_{q(\mathbf{w}|\theta)}[\log P(\mathscr{D}|\mathbf{w})]$ (known as the reconstruction term [20]). Hence, there is no need to compute the KLD term for Gaussian processes training unless one uses non-Gaussian likelihoods (see Section IV). For BNNs, this is unavoidable. The problem is that the KLD term and its gradients do not have closed-form solutions. BBB circumvents the problem by exploiting a reparameterisation process [21]. The objective is to reparameterizes $q(\mathbf{w}|\theta)$ with a parameter-free distribution $q(\varepsilon)$ [21]. BBB chooses zero-mean Gaussian distribution.

The reparameterisation works two-folds: 1) sample from the parameter-free distribution $q(\varepsilon)$ and 2) transform the ε samples and θ into a sample from $q(\mathbf{w}|\theta)$. The function $g(.)$ transforms ε and θ to a sample from θ, it takes the form

$$\begin{aligned}\theta &= (\mu, \sigma^2),\\ \varepsilon &\sim q(\varepsilon) = \mathscr{N}(0,1),\\ g(\theta,\varepsilon) &= \mathbf{w} = \mu + \sigma\varepsilon.\end{aligned} \tag{7}$$

Here, μ and σ represent the mean and the standard deviation used to parameterise $q(\mathbf{w}|\theta)$. It is common [1], to adopt a soft-normalisation on σ, such that $\sigma^2 = \log(1+\exp(\sigma))$. This yields positive values. Based on (7) the gradients $\Delta\mu$ and $\Delta\sigma$ can be obtained in the following way

$$\begin{aligned}\Delta\mu &= \frac{\partial g}{\partial \mathbf{w}} + \frac{\partial g}{\partial \mu},\\ \Delta\sigma &= \frac{\partial g}{\partial \mathbf{w}}\frac{\varepsilon}{\sigma} + \frac{\partial g}{\partial \sigma}.\end{aligned} \tag{8}$$

However, this reparameterisation has been shown to yield biased gradients with a high variance. Studies of uncertainty propagation have assisted in achieving improved calibration approaches to uncertainties [22]. These overcome the shortcomings of VI based methods.

IV. Gaussian Process

A GP is a stochastic process defining a distribution over possible functions that fit a set of points [20]. A GP is a non-parametric method that in the considered type of classification problems can define a distribution over the weights of a network for classification. Given the input data $\mathbf{X}$ of size N, a function $\mathbf{f} = [f_1, f_2, \cdots, f_N]$ is used to best approximate the real function $\mathbf{Y} = f(\mathbf{X})$. This is a typical regression case. If these functions can be represented by the sufficient statistics μ and $\sigma_{GP} = diag(\mathbf{K}(\mathbf{X}\prime, \mathbf{X}'))$, the Gaussian process can be denoted as $p(\mathbf{f}|\mathbf{X}) = GP(\mu, \mathbf{K}(\mathbf{X}\prime, \mathbf{X}'))$. The $diag(.)$ refers to the operator that returns the diagonal elements of the input matrix.

In theory, a GP has an infinite hypothesis space, as a non-parametric process. In practice, this space is defined by the covariance function $\mathbf{K}(\mathbf{X}\prime, \mathbf{X}')$. Also referred to as the covariance matrix or the kernel matrix [20], the $\mathbf{K}$ function determines the discrepancy between the real data point $\mathbf{X}\prime$ and the consecutive data point $\mathbf{X}'$ from the training data $\mathbf{X}$. This paper adopts the squared exponential covariance kernel (SQE) [20], where

$$\mathbf{K}(\mathbf{X}\prime, \mathbf{X}') = \sigma_0^2 \exp\left[-\frac{1}{2}\left(\frac{\mathbf{X}\prime - \mathbf{X}'}{\lambda}\right)^2\right]. \tag{9}$$

In the next section $\mathbf{K}_{XX}$ will denote the kernel matrix based on the training inputs X. Then, σ_0^2 and λ represent the associated amplitude and lengthscale parameters, respectively. These parameters determine the distance from the mean and the length of extrapolation respectively. Usually, for the regression case, computing the posterior mean and covariance function is straightforward. This is because Gaussian likelihoods have closed-form expressions for the sufficient statistics of the posterior $p(\mathbf{f}|\mathbf{y})$ based on a multi-class label set $\mathbf{y} = \{y_n\}_{n=1}^{N}$. For classification, a Gaussian approximation to the posterior is required since non-Gaussian likelihoods make the integrating over a functional space $\mathbf{f}$ intractable.

Although there are different VI approaches, this paper specifically is inspired by ideas from the scalable VI approximation from [23] and [24]. There are two reasons behind this selection: firstly, scalable approximations are ideal for complex inputs such as image data and secondly, that VI is simple to implement and allows gradient-based optimisation which is part of machine learning libraries like Tensorflow and Torch. This is considered in the next section.

V. Scalable Variational Inference For Sparse Gaussian Process Classification

The marginal likelihood and the KLD term in ELBO do not have a closed form solution, especially for non-Gaussian observation likelihoods which are commonly used in classification applications. To tackle this, scalable and computationally efficient GP approaches can be implemented in different ways [25], including by introducing sparsity. In particular, the approach used in this work is inspired by the sparse methods in [23] and [24].

In sparse GP frameworks, inducing points $\mathbf{Z}$ and inducing output variable $\mathbf{u}$ approximate the inputs $\mathbf{X}$ and the functional value $\mathbf{f}$ respectively. This can reduce the complexity of the covariance matrix inversion which is $\mathscr{O}(N^3)$ in general. The difference is that the approach from [23] reduces inversion cost from $\mathscr{O}(N^3)$ to $\mathscr{O}(NM^2)$ while [24] reduces it to $\mathscr{O}(m^{1+\frac{1}{O}})$ where m is the number of inducing points and O is the dimension of the kernel matrix, N and M are total training and testing points. Using the sparse methods, the likelihood of the GP can be simplified in the form

$$p(\mathbf{f}|\mathbf{u}) = \mathscr{N}(\mathbf{f}|K_{X,Z}K_{Z,Z}\mathbf{u}, \tilde{K}), \\ \text{where} \quad \tilde{K} = K_{X,X} - K_{X,Z}K_{Z,Z}^{-1}K_{Z,X}, \tag{10}$$

and the ELBO takes the form

$$\mathscr{L}_{VI} \approx -\mathbb{E}_{q(\mathbf{u})p(\mathbf{f}|\mathbf{u})}[\log p(y|\mathbf{u})] - KL[q(\mathbf{u}\|p(\mathbf{u}))]. \tag{11}$$

Here, the K_{ZZ} represents the kernel matrix based on inducing points Z. The choice of inducing points is a large field of research on its own. Scalable VI handles this issue by using a variational Bayes approach that automatically selects the inducing points by optimising the KLD term (see Section III). This involves closely matching the posterior to the variation distribution $q(\mathbf{u})$. The variational bound on the logarithm of the marginal likelihood $\log p(\mathbf{y})$ can be obtained from (12) given below.

$$\log(p(\mathbf{y}|\mathbf{u})) \geq \mathbb{E}_{p(\mathbf{f}|\mathbf{u})}\left[p(y|\mathbf{f})\right], \tag{12}$$

From here, the optimisation process follows the same procedure mentioned in Section III. The difference is that scalable VI is used to find the optimal inducing points. The gradients for the sufficient statistics of the variational distribution $q(\mathbf{u})$ with respect to the bound are computed. Standard optimisation schemes e.g. stochastic gradient descent [26] can then be used to update the learning parameters of the covariance kernel of the GP. The next section will discuss the proposed framework in detail.

VI. Methodology

A. Convolutional Network Gaussian Process Framework

The proposed framework consists of two components: a CNN model feature extractor and a GP placed after that takes these features as inputs. The CNN has two convolution layers of 32 and 64 filters of 3x3 kernel size and one fully connected layer. Convolution of images results in the decomposition of features. These features are learned hierarchically, starting from simple features (e.g. edges) in earlier layers and more complex at the end [27]. A max-pooling layer is introduced between the second layer and two dropout layers each with probability 0.5 and 0.25, respectively. Pooling layers downsample the features and dropout is used as a regularizer. The fully connected layer, on the other hand, converts the flattened outputs to a 16x128 feature vector. The CNN architecture can be seen in Figure 1. Post softmax function outputs a vector of size 16x10 containing a prediction for each of the 10 classes in permute-MNIST and CIFAR-10 dataset, 16x2 for split-MNIST and 16x100 for CIFAR-100.

B. Training in CNN-GP

The training in CNN-GP is as follows: firstly, the predicted class labels are forward propagated, secondly, the weights of the CNN-GP are updated based on the maximum likelihood loss between the predicted labels and the ground truth labels. This is then followed by the backpropagation of the maximum likelihood function of the GP regularised with the KLD loss. The forward step before regularisation involves replacing the input sample from the mini batch with close neighbouring images in the mini batch. These samples are then passed as inputs to the CNN-GP to get the new predictive labels. The regularised loss compares the new predictive labels with ground truth ones. The neighbour replacement step is inspired from [28]. The only difference is that their approach focuses on noisy labels, the approach in this work focuses on input images. A full description of the algorithm is provided in Algorithm 1 from Section VII along with notation definitions in Table VIII.

C. Variance Guided Learning Rate Update

There are two versions of the network: deterministic and Bayesian settings. In a deterministic setting, the weights of hidden layers are scalar. The learning rate γ_{GP} of the only GP is updated. In the Bayesian setting, it is assumed that the weights are sampled from the prior distribution. This is specified as follows: a scaled mixture of two Gaussian distributions with variances 0.0 and 6., respectively. The scaling mixing coefficient is set to 0.25. This is inspired by UCB's approach [1].

Both the learning rates γ_{CNN} of the CNN and the GP γ_{GP} components are updated. The GP variance characterises the uncertainty, e.g. high variance corresponds to high uncertainty, low variance means accurate results. Using the variance of the post softmax samples, the learning rate is updated depending on the variable patience (ρ). The scaling is performed by simply multiplying the learning rates with the mean of the variance array divided by the standard deviation of the variance array. At each run, if the validation accuracy obtained is less than the one from the previous episode, then ρ is dropped by -1. If ρ reaches 0, a single-step of learning rate update is performed. Otherwise, the default value of $\rho = 5$ is retrieved.

D. Datasets: Split-MNIST, Permuted-MNIST and CIFAR-10

The proposed framework is tested on the datasets split-MNIST and permuted-MNIST. First, the original MNIST [27] is split into five tasks. For example, task 1 will contain images labelled 0 and 1, task 2 will have images labelled 2 and 3. The sequence ends with task 5 which contains images labelled 8 and 9. In permuted-MNIST, a sequence of ten tasks are used, each task being has images labelled 0 till 9. In each task, the pixels of every image sample is randomly perturbed. This dataset is more challenging than split-MNIST.

In this work, the configurations for both datasets follow similarly to [1]. The default setting consists of 10 permutations with a seed number of 100. The averaged accuracies are computed by dividing the correctly classified samples by the

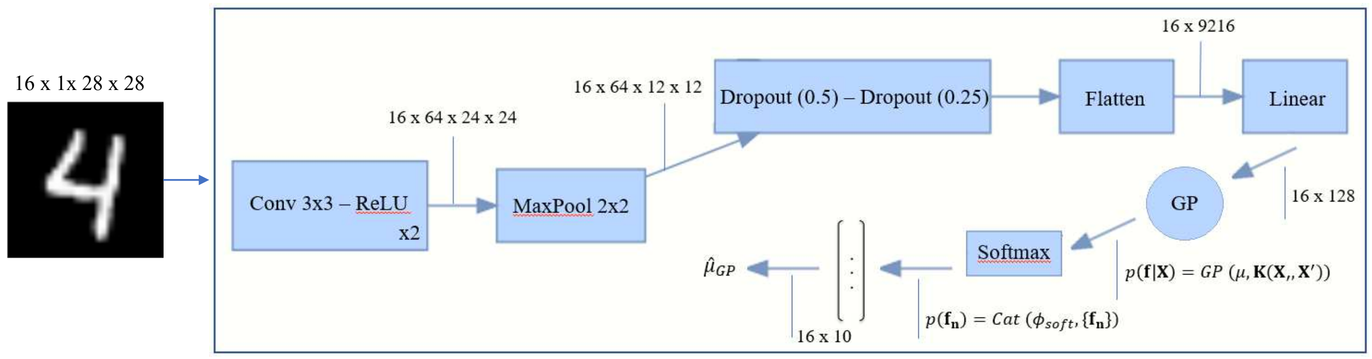
16 x 1x 28 x 28
16 x 64 x 24 x 24
16 x 64 x 12 x 12
16 x 9216
Conv 3x3 – ReLU x2
MaxPool 2x2
Dropout (0.5) – Dropout (0.25)
Flatten
Linear
GP
16 x 128
Softmax
$p(\mathbf{f}|\mathbf{X}) = GP\ (\mu, \mathbf{K}(\mathbf{X},, \mathbf{X}'))$
$p(\mathbf{f_n}) = Cat\ (\phi_{soft}, \{\mathbf{f_n}\})$
16 x 10
$\hat{\mu}_{GP}$

Fig. 1. The CNNGP framework at test time. It consists of a CNN feature extractor and a GP component placed after that takes these features as inputs. This diagram represents the deterministic version i.e. scalar weights. In the Bayesian setting, the weights of the convolutional layers are replaced with prior distributions. The input has size 16x1x28x28 for its batch size, channel depth, height and width respectively. The output dimensions are shown for each output.

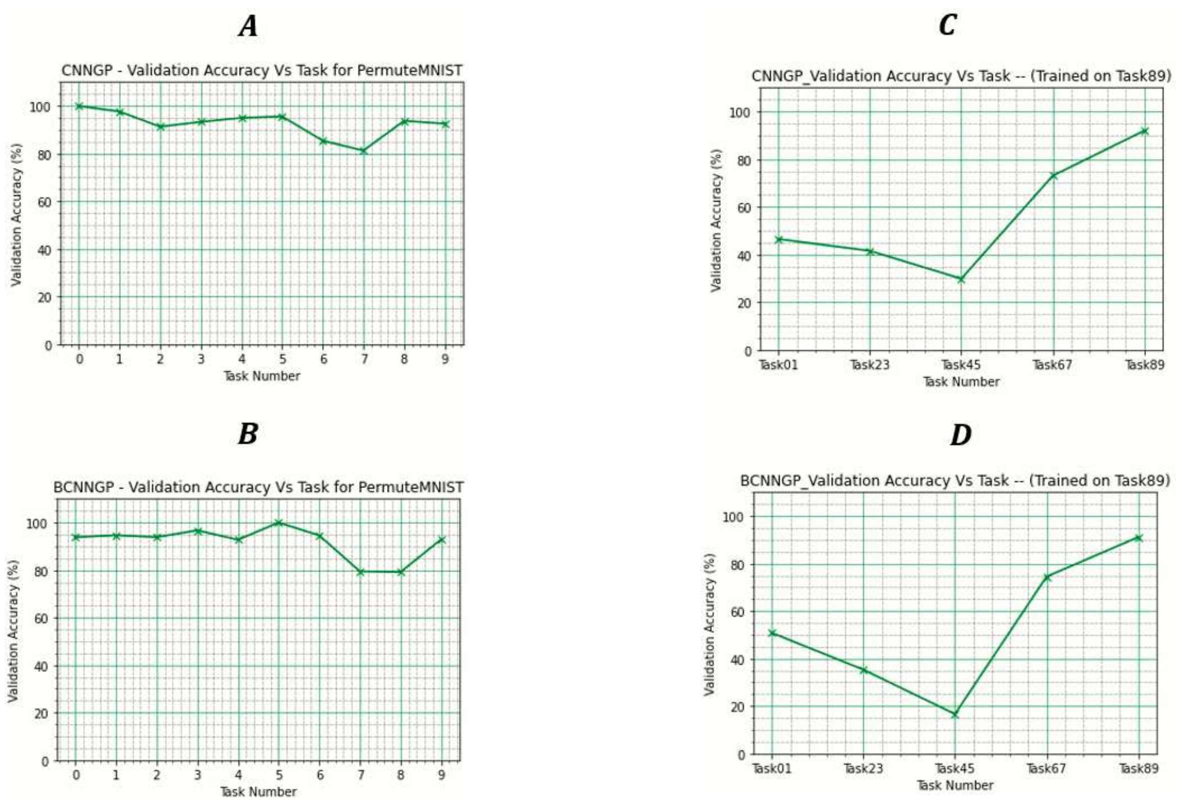

Fig. 2. Comparison of accuracy for both the deterministic (A, C) and Bayesian setting of the proposed framework(B, D) on both split (top row) and permuted-MNIST (bottom row).

total number of samples. This is repeated for both split-MNIST and permuted-MNIST. The CIFAR-10 [15] dataset consists of 60000 coloured images of size 32 x 32 of 10 different categories. The dataset is divided into 50000 training and 10000 testing images. The majority of the categories include vehicles and animals.

E. Robustness Analysis with Gaussian Noise

In network training, the parameters of the noise impacting input images are unknown. Gaussian white noise is used widely across DNN literature to test robustness [22], [29]. This is also used for performance validation in this paper. Commonly, the standard deviation of the additive Gaussian noise can be added incrementally and used to pollute the inputs. This procedure is usually followed by measuring the signal-to-noise ratio (SNR) respectively for each batch of the testing set. The calculation is performed as shown in (13) taken from [30], where

$$SNR = 10 \cdot \log_{10} \left[\frac{\sum_0^{n_x-1} \sum_0^{n_y-1} [r(x,\ y)]^2}{\sum_0^{n_x-1} \sum_0^{n_y-1} [r(x,\ y) - t(x,\ y)]^2} \right]. \quad (13)$$

Here, $r(x,\ y)$ and $t(x,\ y)$ refer to the training and test images and n_x and n_y their respective sizes. The terms $\sum_0^{n_x-1}, \sum_0^{n_y-1}$ refer to the summation of both numerator and denominator elements w.r.t the sizes of both training and testing datsaset.

VII. Algorithm Description

The developed is presented in Algorithm 1. All experiments in our paper use the following default arguments; batch size=16, episodes=200, learning rate of GP=0.1, neighbors sampling no.=10, KLD scaling factor = 1, ρ = 5

```
Require: $L$: episodes, $\gamma$: learning rate (GP), $K$: number of
  nearest neighbours for choice of new sample, $B$: batch size,
  $\beta$: KLD scaling factor, $\rho$ : patience
Do initialization of weights: $\theta^l_{CNN}$, $\theta^l_{GP}$
for $l = 0, \cdots, L$ do
    Sample mini batch ($x_i$,$y_i$), of length $N$ from dataset
      $D = X, Y$ where $X$ and $Y$ are 4-D tensors holding
      images and labels from the entire dataset, where
      $x_i \in \mathbb{R}^{hxwxc}$ (image height, width and channel) and
      $y_i \in \mathbb{R}^{1xC}$ ($C$ is total number of classes).
    BEGIN Updating weights of CNN-GP based on the
      maximum likelihood $\mathscr{L}_{MLE}$
    do → forward pass of CNN component $f^{CNN} : x_i \rightarrow z_i$,
      where $z_i \in \mathbb{R}^{UxC}$ and $U$ is the number of hidden units'
      feature outputs passed from final fully connected layer
      of CNN feature extractor
    do → forward pass of GP $f^{GP} : z_i$ to obtain the posterior
      likelihood $p\big(y_i|f^{GP}(z_i);\big(\mu_i, K_i\big)\big)$ where $\mu_i$ represents
      the mean of the GP and $K_i$ is the squared exponential
      kernel and $\mathscr{N}$ represents the Gaussian distribution
    do → Compute the expected log likelihood to obtain
      max likelihood loss:
      $\mathscr{L}_{max} \approx \sum_{i=1}^{N} \mathbb{E}_q \Big[ \log\big(p(y_i \| f^{GP}(x_i); \mu_i, \sigma_i^2\big) -$
      $\beta D_{KL}\big(q(u_i)\|p(u_i)\big)\big)\Big]$
    do → Compute gradients of loss with respect to the
      weights of CNN feature extractor and GP :
      $\frac{\partial \mathscr{L}_{max}}{\partial \theta_{GP}}, \frac{\partial \mathscr{L}_{max}}{\partial \theta_{CNN}}$
    do → Update the parameters of GP and the weights
      $\theta^{l+1}_{CNN}$ CNN feature extractor for the $l^{th}$ episode:
      $\theta^{l+1}_{CNN} \leftarrow \theta^l_{CNN} - \gamma \frac{\partial \mathscr{L}_{max}}{\partial \theta^l_{CNN}} . \mathscr{L}_{max}$,
      $\theta^{l+1}_{GP} \leftarrow \theta^l_{GP} - \gamma \frac{\partial \mathscr{L}_{max}}{\partial \theta^l_{GP}} . \mathscr{L}_{max}$
    BEGIN backpropagation of maximum likelihood loss
      regularised with KLD $\mathscr{L}_{max} + \mathscr{L}^{KLD}$
    do → Replace input samples with top-10 neighbours $\hat{x}_i$
    do → forward pass of the CNN feature extractor
      $f^{CNN} : \hat{x}_i \rightarrow \hat{z}_i$
    do → forward pass of the GP
      $f^{GP} : \hat{z}_i \rightarrow p\big(\hat{y}_i|f^{GP}(\hat{z}_i); \hat{\mu}_i, K_{\hat{x}_i}\big)$
    do → Calculate $\mathscr{L}_{max} + \mathscr{L}^{KLD}$
    do → Update the new parameters of
      $\theta^l_{GP}, GP : \theta^l_{GP} \leftarrow \theta^l_{GP} - \gamma \frac{\partial \mathscr{L}^{GP}}{\partial \theta^l_{GP}} \mathscr{L}^{GP}$
    do → Update the weights of the CNN feature extractor :
      $\theta^l_{CNN} \leftarrow \theta^l_{CNN} - \gamma \frac{\partial \mathscr{L}^{GP}}{\partial \theta^l_{CNN}} . \mathscr{L}^{GP}$
end
```

VIII. Experiments

A. Accuracy Comparison Per Task - Deterministic vs Bayesian Setting

This experiment aims to compare the performance of the CNNGP framework on both split and permuted-MNIST datasets for each of the two settings. The experiment is carried out by training the framework on each of the tasks for 200 episodes and then validating with the respective task. This is performed 100 times for each episode and then averaged. These averaged accuracy is plotted for the deterministic setting (CNNGP) in Figure 2A and 2C, for the Bayesian setting (BCNNGP) in 2B and 2D. The x-axis represents the task number ranging from 0-5 for split-MNIST (top row) and 0-9 for permuted-MNIST (bottom row).

B. Comparison of Episodes Required to Train on Permuted-MNIST - Deterministic Vs Bayesian

The purpose of this experiment is to compare the number of episodes required for both model configurations to converge on permuted-MNIST. This is carried out by training each task repeatedly 80 times. Then, the number of episodes required for each is averaged and tabulated for both the configurations in Table II. Comparison for split-MNIST was not required since the results were closely similar ($\pm$ 2 episodes).

C. Comparison of Computational Time Taken on Single Episode on Permuted-MNIST - Deterministic Vs Bayesian

The focus of this experiment is to compare the computational time taken to train on a single episode of the permuted-MNIST testing set. Once again both the deterministic and Bayesian setting is compared. The testing time is measured in minutes and averaged across 50 repeats for each of the 10 tasks. The results are presented in Table III.

D. Robustness Analysis - UCB Vs Bayesian Setting

The objective of this experiment is two-fold: a) to compare per-class accuracy on split-MNIST, permuted-MNIST and CIFAR-10 b) test robustness against Gaussian noise attacks, all for cases of Bayesian setting and UCB. The experiments are carried out by first training UCB on both split, permuted-MNIST and CIFAR-10 using the hyperparameters similar to those provided in [1] and also in Algorithm VII. In all experiments where CNN-GP is compared with UCB, the CNN component of CNN-GP is reduced to two fully connected layers. This is done since UCB architecture also uses only two fully connected layers. Furthermore, in the per-task accuracy experiments on permuted-MNIST, two versions of CNN-GP are tested. One where the input features are set to 128 and grid size to 16, termed BCNNGP. The other version uses 256 input features and a grid size of 64, termed BCNNGP-256.

Once trained, the system is perturbed with Gaussian noise. This noise is added by varying the value of the standard deviation from 0-2 in steps of 0.25. The images are perturbed with the noise and the respective SNR values are calculated using equation (13) for each batch per task. This is averaged across all batches and tasks to obtain the final SNR value.

TABLE I
PER-TASK ACCURACY ON PERMUTED-MNIST FOR BOTH UCB AND BAYESIAN SETTING

Model Type	**Task 0**	**Task 1**	**Task 2**	**Task 3**	**Task 4**	**Task 5**	**Task 6**	**Task 7**	**Task 8**	**Task 9**
UCB	76.29	60.91	61.30	61.55	60.60	60.41	60.25	61.32	60.60	61.30
BCNNGP	40.78	43.65	34.25	35.77	37.20	34.89	32.31	33.74	36.50	39.11
BCNNGP-256	86.33	83.08	67.89	67.55	89.23	68.42	63.36	84.35	85.41	87.62

TABLE II
COMPARISON OF EPISODES REQUIRED FOR TRAINING ON PERMUTED-MNIST FOR BOTH DETERMINISTIC AND BAYESIAN SETTING

Tasks	**Deterministic**	**Bayesian**
0	26	45
1	33	8
2	112	22
5	74	9
6	40	10
7	54	14
8	95	29
9	18	4

TABLE III
COMPARISON FOR AVERAGE TIME TAKEN FOR RUNNING A SINGLE EPISODE FOR BOTH DETERMINISTIC AND BAYESIAN SETTING

	Deterministic	**Bayesian**
Average time for a single episode (mins)	5.02	11.65

As the results are obtained, the respective per-task and per-SNR accuracies are calculated. Per-task is obtained by averaging the accuracies across all SNR values for each task. Per-SNR accuracies are calculated by averaging across all tasks for each SNR value. The results are presented for split-MNIST in Tables IV and V, for permuted-MNIST in Table I and for CIFAR-10 and CIFAR-100 in Table VI and VII

IX. DISCUSSION

The results show that the performance of the deterministic and Bayesian settings is similar. The difference is approximately 5% in terms of accuracy. However, on the majority of the tasks, the deterministic setting takes more iterations to converge than the Bayesian setting. However, the computational time for the deterministic setting (5.02mins) is nearly half that of the Bayesian setting (11.65mins). When comparing with the UCB approach, the CNNGP can perform better than UCB in some tasks (Task 1 and 3) in split-MNIST. However, it is unable to perform in permuted-MNIST. In terms of robustness, for high SNR values (0-20) UCB is more robust to Gaussian attacks than CNNGP, for low SNR values (0 to -6) CNNGP is more robust. However, UCB is more robust than CNNGP in the permuted-MNIST dataset. Therefore, the main takeaway message of this study is that for relatively simple datasets and few tasks, a single-classifier can perform as good as a multi-classifier for continual learning, while being robust to noise. However, if the dataset is difficult to train and has more than, e.g. ten tasks, then a multi-classifier setting is better suited.

TABLE IV
PER-TASK ACCURACY ON SPLIT-MNIST FOR BOTH UCB AND BAYESIAN SETTING

Model Type	**Task 0**	**Task 1**	**Task 2**	**Task 3**	**Task 4**
UCB	99.52	93.23	93.23	95.59	97.87
BCNNGP	96.17	95.22	94.69	97.36	92.86

TABLE V
PER-SNR ACCURACY ON SPLIT-MNIST FOR BOTH UCB AND BAYESIAN SETTING

	Signal-to-Noise Ratio								
Model Type	20	12	6	2	0	-2	-4	-5	-6
UCB	99.67	99.50	99.39	98.32	97.28	95.50	93.35	91.28	88.70
BCNNGP	94.95	93.35	95.59	95.22	96.40	96.67	95.91	91.33	97.91

For the interest of the readers, this work can be improved further with multiple GP classifiers or by testing with other forms of learning rate update methods such as momentum-based learning rate schedules [26].

X. CONCLUSIONS AND FUTURE WORK

This study investigates the performance of a single-classifier in continual learning compared to the state-of-the-art multi-classifier methods such as UCB. In particular, a GP classifier placed after a CNN architecture of two convolutional layers and one fully connected layers is used. The learning rate updates are performed by scaling according to the post softmax sample variance. Experiments are carried out on both split and permuted-MNIST. Two settings are adopted: deterministic and Bayesian. The results show that the performance of both settings is on par. However, the Bayesian setting takes a lower number of episodes to converge. On split-MNIST, the performance of CNNGP and UCB is on par but UCB outperforms CNNGP in permuted-MNIST. In the future, it is possible to further experiment with various CNN architectures for feature extraction as well as exploring other ways to update learning parameters.

Acknowledgements. This work was supported by the UK EPSRC via Grant No. EP/ T013265/1, NSF-EPSRC: “ShiRAS. Towards Safe and Reliable Autonomy in Sensor Driven” jointly with the USA National Science Foundation, Grant No. NSF ECCS 1903466, and the UK Research and Innovation (UKRI) Trustworthy Autonomous Systems (TAS) programme (EPSRC Ref: EP/ V026747/1).

TABLE VI
PER-TASK ACCURACY ON CIFAR-10 FOR BOTH UCB AND BAYESIAN SETTING

Model Type	**Task 0**	**Task 1**	**Task 2**	**Task 3**	**Task 4**
UCB	80.63	86.41	69.92	70.66	70.68
BCNNGP	61.20	93.55	67.24	61.29	67.14

TABLE VII
PER-TASK ACCURACY ON CIFAR-100 FOR BOTH UCB AND BAYESIAN SETTING

Model Type	**Task 0**	**Task 1**	**Task 2**	**Task 3**	**Task 4**
UCB	44.51	38.30	39.61	38.02	37.61
BCNNGP	45.87	54.44	11.29	13.71	8.14

REFERENCES

[1] S. Ebrahimi, M. Elhoseiny, T. Darrell, and M. Rohrbach, "Uncertainty-Guided Continual Learning with Bayesian Neural Networks," *arXiv preprint arXiv:1906.02425*, 2019.

[2] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A Continual Learning Survey: Defying Forgetting in Classification Tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. Feburary, 2021.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[5] G. Marcus, "The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence," *arXiv preprint arXiv:2002.06177*, 2020.

[6] T. G. Dietterich, "Steps Toward Robust Artificial Intelligence," *AI Magazine*, vol. 38, no. 3, pp. 3–24, 2017.

[7] R. M. Neal, *Bayesian Learning for Neural Networks*, vol. 118. Springer Science & Business Media, 2012.

[8] M. Javed and L. Mihaylova, "Leveraging uncertainty in adversarial learning to improve deep learning based segmentation," in *Proc. of the Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pp. 1–8, IEEE, 2019.

[9] R. M. French, "Catastrophic Forgetting in Connectionist Networks," *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[10] D. L. Silver and R. E. Mercer, "The Task Rehearsal Method of Life-Long Learning: Overcoming Impoverished Data," in *Conf. of the Canadian Society for Computational Studies of Intelligence*, pp. 90–101, Springer, 2002.

[11] A. Kendall and Y. Gal, "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?," in *Advances in Neural Information Processing Systems*, pp. 5574–5584, 2017.

[12] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, "Variational Continual Learning," *arXiv preprint arXiv:1710.10628*, 2017.

[13] F. Zenke, B. Poole, and S. Ganguli, "Continual Learning Through Synaptic Intelligence," in *International Conf. on Machine Learning*, pp. 3987–3995, PMLR, 2017.

[14] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks," *arXiv preprint arXiv:1312.6211*, 2013.

[15] A. Krizhevsky, G. Hinton, *et al.*, "Learning Multiple Layers of Features From Tiny Images," 2009.

[16] V. Nair and G. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proc. of 27th International Conf. on Machine Learning*, ICML'10, (USA), pp. 807–814, Omnipress, 2010.

[17] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight Uncertainty in Neural Networks," *arXiv preprint arXiv:1505.05424*, 2015.

TABLE VIII
NOTATIONS AND DEFINITIONS

Notation	**Meaning**
L	Total number of episodes
γ_{CNN}/γ_{GP}	Learning rate of the CNN and the GP component respectively
ρ	Patience
K	Number of nearest neighbours to pick a new sample from
B	Batch size
β	Kullback-Leibler divergence scaling factor
l	Episode number
λ	Lengthscale parameters of the GP classifier
A	The amplitude for the squared exponential kernel
u_i	Variational free parameters for the i^{th} batch of data
$q(u_i)$	Variational likelihood based on the i^{th} batch of data
$p(u_i)$	Expected real likelihood based on the i^{th} batch of data
θ^l_{CNN}	Weights of the CNN feature extractor for the l^{th} episode
θ^l_{GP}	Weights of the GP classifier for the l^{th} episode
x_i	Data sample from the i^{th} batch of data
y_i	Label sample from the i^{th} batch of data
X	4D-data tensor holding the data samples
Y	4D-data tensor holding the labels samples
$\mathscr{D}$	Dataset ordered pair holding X and Y
U	Number of units passed from CNN
δx_i	The difference between the i^{th} data point and the CNNGP
f^{GP}	The Gaussian process function
f^{CNN}	The CNN function
$\hat{y}_i^{CNN}$	Softmax prediction from the CNN feature extractor
$\hat{y}_z^{CNN}$	Prediction from the z^{th} node from the CNN feature extractor
$\mathscr{L}_{max}$	Maximum likelihood loss
$\mathscr{L}^{KLD}$	Kullback-Leibler Divergence

[18] G. Hinton and D. Van Camp, "Keeping Neural Networks Simple by Minimising the Description Length of Weights. 1993," in *Proc. of COLT-93*, pp. 5–13.

[19] A. Graves, "Practical Variational Inference for Neural Networks," in *Advances in Neural Information Processing Systems*, pp. 2348–2356, 2011.

[20] C. E. Rasmussen, "Gaussian Processes In Machine Learning," Springer, 2003.

[21] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[22] D. Dera, G. Rasool, and N. Bouaynaya, "Extended Variational Inference for Propagating Uncertainty in Convolutional Neural Networks," in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, IEEE, 2019.

[23] J. Hensman, A. Matthews, and Z. Ghahramani, "Scalable Variational Gaussian Process Classification," in *Artificial Intelligence and Statistics*, pp. 351–360, PMLR, 2015.

[24] A. G. Wilson, Z. Hu, R. R. Salakhutdinov, and E. P. Xing, "Stochastic variational Deep Kernel Learning," *Advances in Neural Information Processing Systems*, vol. 29, pp. 2586–2594, 2016.

[25] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When Gaussian Process Meets Big Data: A Review of Scalable GPs," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4405–4423, 2020.

[26] H. Robbins, "A Stochastic Approximation Method," *Annals of Mathematical Statistics*, vol. 22, pp. 400–407, 2007.

[27] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied To Document Recognition," *Proc. of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[28] J. Li, Y. Wong, Q. Zhao, and M. S. Kankanhalli, "Learning to Learn from Noisy Labeled Data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5051–5059, 2019.

[29] G. Carannante, D. Dera, G. Rasool, N. C. Bouaynaya, and L. Mihaylova, "Robust Learning via Ensemble Density Propagation in Deep Neural Networks," in *Proc. of the IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, IEEE, 2020.

[30] R. C. Gonzales and R. E. Woods, *Digital Image Processing*. Prentice Hall New Jersey, 2002.