Dissertation Type: Research

DEPARTMENT OF COMPUTER SCIENCE

Globally Induced Variational Continual Learning

Alexander Carolan

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Engineering in the Faculty of Engineering.

Monday 23rd August, 2021

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of MEng in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Alexander Carolan, Monday 23rd August, 2021

Contents

List of Figures

List of Tables

List of Algorithms

Executive Summary

The field of artificial intelligence has undergone a transformation, as advancements in deep learning continue to expand the horizons of what it can achieve. Although significant progress has been realised, artificial intelligence still falls short of human intelligence, most notably perhaps when it comes to continual learning. This is the ability to acquire knowledge not just of a single task but to accumulate knowledge from multiple sequential tasks, whilst only ever having access to the current task data. Therefore enabling the efficient incremental development of ever more complex skills, as well as ideally the transfer of knowledge between tasks, all whilst not retaining previous task data. A variety of continual learning methods exist, aiming to solve this issue and in the process create an avenue towards general artificial intelligence. The objective of this project is to build on the work of variational continual learning, by merging it with the recent work of global inducing point variational inference. The hypothesis being that because global inducing points act as pseudo data points, they could behave as an episodic memory for the model, capturing the essential information of each task, refreshing the model's knowledge of them, thereby increasing performance. Over the course of this project the following has been achieved:

- I have developed an understanding of the fundamental principles of Bayesian probability specifically of Bayesian inference.
- I have developed an understanding of the intrinsic operation of artificial neural networks specifically of Bayesian neural networks.
- I have developed an understanding of the alternative deep learning techniques of online learning and offline learning.
- I have developed an understanding of the complimentary deep learning methods of continual learning and transfer learning.
- I have implemented both the established mean field as well as a novel globally induced variational continual learning model.
- I have performed a comprehensive hyperparameter optimisation on the globally induced variational continual learning model.
- I have obtained results exceeding those of the original and matching those of the improved variational continual learning research papers.
- I have achieved this whilst using fully online and strictly continual techniques unlike the variational continual learning research papers.

Supporting Technologies

This project was made possible with the following supporting technologies:

- The BluePebble supercomputer on which the project was executed.
- The Python 3.7.3 language in which the project was implemented.
- The Pytorch library which provided artificial neural network functionality.
- The Bayesfunc library which provided Bayesian neural network functionality.
- The Matplotlib library which provided plotting and visualisation functionality.

Acknowledgements

I would like to thank my supervisor Laurence Aitchison for his support and encouragement throughout this project. It has been an honour to work on a project as interesting as this and a pleasure to work with someone as patient as Laurence.

Chapter 1

Contextual

1.1 Introduction

Intelligence is an elusive and enigmatic capacity known intuitively to all of us yet conceptually hard to define. An essential aspect of it common to all advanced lifeforms though, is its ability to continue learning throughout its lifetime. Continual learning involves the use of data to continuously extend the model's knowledge. Specifically the ability to acquire knowledge not just of a single task, but to accumulate knowledge from multiple sequential tasks, whilst only ever having access to the current task data. Variational continual learning [1] implements this by leveraging the intrinsically continual process emerging from the recursive application of Bayesian inference, which combines what the current task data indicates about the model parameters, with what previous task data indicated about the model parameters. Naturally this can be combined with Bayesian neural networks [2], which apply Bayesian inference over the model parameters, such that the wide variety of complex tasks typically solved by artificial neural networks, due to their continuous function approximation ability, may also be solved in a continual setting.

Resulting from the intractable nature of the probability distributions occurring in Bayesian neural networks, typically exact Bayesian inference can't be performed and instead an approximate inference scheme must be chosen. This can be either a stochastic approximation such as the sampling based Markov chain Monte Carlo approximation [3] or an analytic approximation such as the variational based mean field approximation [4]. Stochastic inference schemes are asymptotically exact but computationally expensive, whereas analytic inference schemes are asymptotically inexact but computationally inexpensive, which is crucial with ever larger data sets becoming necessary. As such analytic inference schemes are gaining popularity amongst not only professionals implementing them but also amongst researchers developing them. This has led to the recent development of a global inducing point variational inference scheme [5], inspired by the local inducing point variational inference scheme used in deep Gaussian processes [6]. Creating a unifying inference scheme across both deep learning classes, which also displays properties potentially ideal for continual learning.

Consequently the focus of this project is to investigate the feasibility and effectiveness of the global inducing point variational inference scheme in a variational continual learning model. The mean field variational inference scheme used in the variational continual learning research papers, assumes that the joint distribution over the model parameters can be factorised out into independent distributions, thereby averaging over the degrees of freedom and approximating the effect of model parameters on each other by that of a single averaged effect. In contrast the global inducing point variational inference scheme, defines a set of global inducing points, which act as pseudo data points in order to induce a distribution over the model parameters. As the global inducing points act as pseudo data points, they could behave as an episodic memory for the model, capturing the essential information of each task, refreshing the model's knowledge of them, thereby increasing performance.

1.2 Motivation

The motivation for this project is simply to contribute to the development of more effective and efficient deep learning models. Taking inspiration from the operation and attributes of human intelligence, in order to assist in the advancement of continual learning methods. Naturally this raises the question *how does human intelligence operate and what are its attributes*? Human intelligence cannot be understood in isolation but only in relation to the environment within which it exists. It is through constant interaction with this environment that a continual source of information is obtained, from which knowledge and skills are extracted by interpreting it through what is already known. This interaction is highly dynamic in nature, involving multiple different tasks, changes over time in tasks, transfers of knowledge across tasks and the discarding of information from tasks. In order to be capable of all this though, it must possess the following attributes, adaptability in incorporating changes, versatility in transferring knowledge and scalability in discarding information. These are the attributes that continual learning seeks to encapsulate in its generalisation of deep learning beyond a single task.

1.2.1 Adaptability

Adaptability is integral to human intelligence, given the constant variation presented to it by the environment within which it exists. Incorporating changes involves striking an incredibly delicate balance between trusting the indications of new information, and trusting the indications of old information. This becomes only the more difficult when the information is so highly dimensional in nature, that it becomes effectively impossible to completely characterise, which is a surprisingly common occurrence. This is an immensely important problem in machine learning, since practically all real world tasks continually evolve over time, causing covariate and data set shifts which traditional methods typically struggle with [7]. The creation of models capable of overcoming these covariate and data set shifts, could significantly expand machine learning beyond the generally static tasks of present into the particularly dynamic tasks of the future.

1.2.2 Versatility

Versatility is crucial to human intelligence, provided the sheer number of tasks it must solve with only limited resources. Transferring knowledge prevents the necessity of every skill being learnt from scratch, instead allowing them to be built upon preexisting skills, and become ever more complex. The compositional nature of this knowledge allows for the efficient use of limited resources, to solve not just a single task but multiple tasks. In addition to this the tasks tend to be solved more effectively than if learnt in isolation, owing to the benefits of cross domain knowledge. This problem in machine learning is known as transfer learning, with forward transfer using knowledge from old tasks to perform more effectively on new tasks, and backward transfer using knowledge from new tasks to perform more effectively on old tasks [8]. The development of models effective at performing this would be of incredible value, enabling a modularity and composability amongst models, thereby removing the necessity for entirely new models to be developed for each task.

1.2.3 Scalability

Scalability is essential to human intelligence, considering the immense amount of information it collects it collects in comparison to the amount it stores. Discarding information constrains the model to create and maintain efficient representations, which capture only the necessary relevant aspects. The exact same problem is facing machine learning, with ever larger data sets becoming necessary in order to learn ever more nuanced tasks, the feasibility of storing this data or even communicating it is quickly diminishing [9]. The formulation of models which require less data to perform effectively obviously alleviate the problem. Bayesian neural networks achieve this by virtue of their inclusion of uncertainty in the network itself, which is greater in lower data settings and therefore imperative to effective performance. Nonetheless only continual models which don't require any previous data truly solve the problem, and provide an opportunity to revolutionise machine learning.

1.3 Proposition

The proposition for this project is to develop a globally induced variational continual learning model, and investigate the feasibility and effectiveness of this novel inference scheme in a continual learning setting. Initially following the design of the original variational continual learning research paper, except for the omission of core sets, a technique used to refresh the model's knowledge of previous tasks. Although the use of core sets improves performance, it does not allow for the use of the model in a strictly continual setting, as it retains a subset of previous task data. Thereafter following the design of the improved variational continual learning research paper, except for the omission of model reinitialisation, a technique used to escape any local optima after each task. Similarly although the use of model reinitialisation improves performance, it does not allow for the use of the model in a fully online setting, as it prevents the model from accumulating any significant amount of knowledge.

1.3.1 Challenges

The challenges involved in this project are ones well established in the field of machine learning, most prominently perhaps is the challenge of balancing plasticity in the model so that it can obtain new skills, and stability in the model so that it can retain old skills. Striking this balance in neural networks is incredibly difficult to achieve, with it otherwise leading to the infamous catastrophic interference effect, where the neural network completely and abruptly forgets previously learnt information upon learning new information [10]. Alongside this is the challenge of efficiently allocating model capacity as the number of tasks grows, with an ideal solution using as much model capacity as required until it reaches saturation, upon which it would selectively forget the least important information in order to make space for more important information. The final challenge and yet another avenue for forgetting, lies in the effectiveness of the inference scheme employed, by the necessity of its approximate nature, repeated applications of it result in the accumulation of error and so the loss of information and the forgetting of old tasks [1]. Overcoming these challenges would represent a significant step forward in the research of continual learning.

1.3.2 Objectives

The objectives established for this project outline the necessary stages involved in developing and demonstrating the feasibility and effectiveness of a globally induced variational continual learning model:

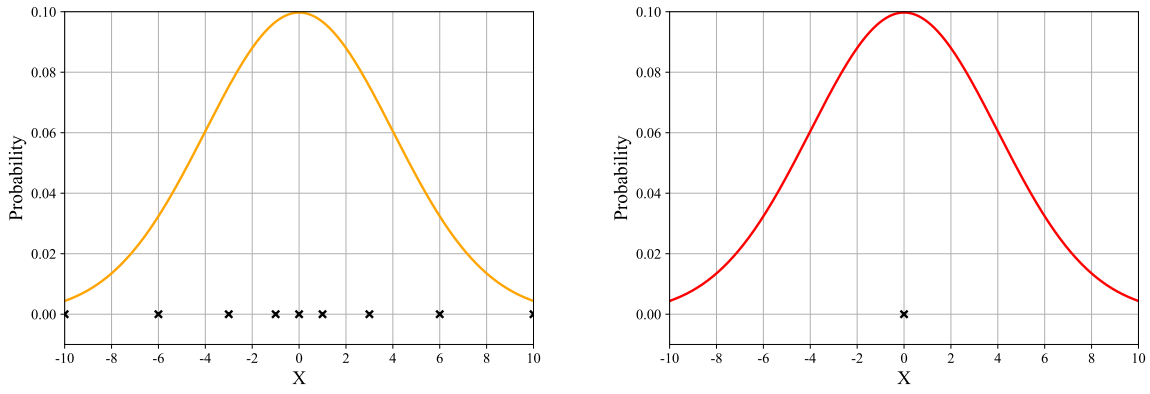
- Research Bayesian probability specifically Bayesian inference.
- Research artificial neural networks specifically Bayesian neural networks.
- Implement a mean field traditional learning model for the standard MNIST problem.
- Implement a globally induced traditional learning model for the standard MNIST problem.
- Implement a mean field variational continual learning model for the permuted MNIST problem.
- Implement a globally induced variational continual learning model for the permuted MNIST problem.
- Optimise the performance of the globally induced variational continual learning model.
- Evaluate the performance of the globally induced variational continual learning model.
- Achieve results exceeding or matching the original and improved research papers.

Chapter 2

Technical

2.1 Bayesian Probability

The study of probability theory is divided into two distinct schools, over the interpretation of the fundamental concept of probability itself. The frequentist interpretation treats probability distributions as a physical measure, associated with random processes and corresponding to the relative frequency of all possible events. As such it can be viewed as an extension of classical probability, enabling analysis of random processes lacking in natural symmetry. Whereas the Bayesian interpretation treats probability distributions as an evidential measure, associated with any statement whatsoever and corresponding to the degree of belief in a particular event. This interpretation naturally permits the use of Bayes theorem in order to perform Bayesian inference, the process of updating beliefs in accordance with new data. As a result of this it can be viewed as an extension of propositional logic, enabling reasoning about hypotheses whose propositions are uncertain. In this sense they are not competing interpretations but rather contrasting approaches to probability, with the aim of solving entirely different problems [11]. This difference being clearly illustrated in Figure 2.1, in which a Gaussian distribution models either many events or a single event depending on the interpretation.



(a) Frequentist

(b) Bayesian

Figure 2.1: Probability interpretations.

The essence of Bayesian probability can be summarised with the statement, *information should not determine beliefs but update them.* Seeing as any information collected is neither perfect nor complete, assumptions are necessary in order to establish beliefs and learn. Incorporating uncertainty into these beliefs is simply a realisation of the assumptions upon which they were built. It is becoming increasingly apparent that this is the principle upon which human intelligence operates, collecting new information about the world through its senses, assessing the uncertainty of this information, and using this to update its beliefs about the world [12].

2.1.1 Bayes Theorem

The foundation of Bayesian probability is embedded in Bayes theorem, a seemingly simple theorem with surprisingly profound theoretical consequences, allowing one to use the rules of probability theory to quantify the extent to which new data should update one's beliefs. Arising as a natural consequence of the relationship between joint and conditional probabilities, which is formalised in the product rule of probability [13]. As this contains a natural symmetry it results in two possible definitions, when considering the two variable case as follows:

$$\begin{aligned} P(D,\theta) &= P(D|\theta)P(\theta) \\ P(D,\theta) &= P(\theta|D)P(D) \end{aligned} \tag{2.1}$$

Specifically Bayes theorem is the result of equating both definitions of the product rule and performing some simple rearrangement:

$$\begin{aligned} P(\theta|D)P(D) &= P(D|\theta)P(\theta) \\ P(\theta|D) &= \frac{P(D|\theta)P(\theta)}{P(D)} \end{aligned} \tag{2.2}$$

Each component of Bayes theorem is given a specific name and serves a specific role in Bayesian inference:

$$\begin{aligned} P(\theta|D) &: Posterior \;\; \text{- } Probability\ of\ a\ model\ given\ the\ data. \\ P(D|\theta) &: Likelihood \;\; \text{- } Probability\ of\ the\ data\ given\ a\ model. \\ P(\theta) &: Prior \;\; \text{- } Probability\ of\ a\ model\ independent\ of\ the\ data. \\ P(D) &: Evidence \;\; \text{- } Probability\ of\ the\ data\ independent\ of\ a\ model. \end{aligned}$$

The posterior distribution is the component of Bayes theorem one desires to compute when performing Bayesian inference, as it allows one to find the most probable model given the data. Produced using contributions from the likelihood distribution which indicates the compatibility of the data with the model, the prior distribution which incorporates preexisting beliefs and the evidence distribution which normalises the result. In contrast frequentist inference only considers the likelihood distribution, as this is the only component directly derived from an experiment. Instead of computing an entire probability distribution, typically a point estimate such as the maximum likelihood estimation would be computed. However this undesirably results in all information regarding uncertainty being lost in the process.

2.1.2 Bayesian Inference

Exact Bayesian inference generally requires that all of the previously mentioned components be known such that it can be performed. Whilst the likelihood distribution can be simply derived from an experiment and the prior distribution can be trivially selected, in the majority of cases the evidence distribution is unknown and often intractable to compute. To understand why one must first appreciate the relationship between joint and marginal probability distributions, which is formalised as the sum rule of probability. This can also be expressed through two possible definitions, this time however arising from considering either the discrete or continuous case as follows:

$$\begin{aligned} P(D) &= \sum_{\theta} P(D,\theta) \\ P(D) &= \int P(D,\theta)\ d\theta \end{aligned} \tag{2.3}$$

Expanding the evidence distribution by application of the sum rule first and the product rule second, derives a new expression of it which more clearly explains the motivation behind its alternative name, the marginal likelihood distribution:

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{\int P(D,\theta)\ d\theta}$$
$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{\int P(D|\theta)P(\theta)\ d\theta} \tag{2.4}$$

Although the marginal likelihood distribution is tractable in low dimensional problems, computing the product of the likelihood distribution and prior distribution across all possible model parameterisations, quickly becomes intractable as the dimensionality of the problem grows. As such alternative inference techniques which avoid computing the marginal likelihood distribution become the only viable option. One such technique is the use of conjugate distributions, in which a conjugate prior distribution is selected according to the likelihood distribution, ensuring a conjugate posterior distribution in the same family as the prior distribution [14]. As a result the marginal likelihood distribution becomes unnecessary, as the posterior distribution can be normalised independently of it. This technique is illustrated in Figure 2.2 using the self-conjugate Gaussian likelihood distribution, with a known variance and an unknown mean, which requires a Gaussian prior distribution and produces a Gaussian posterior distribution.

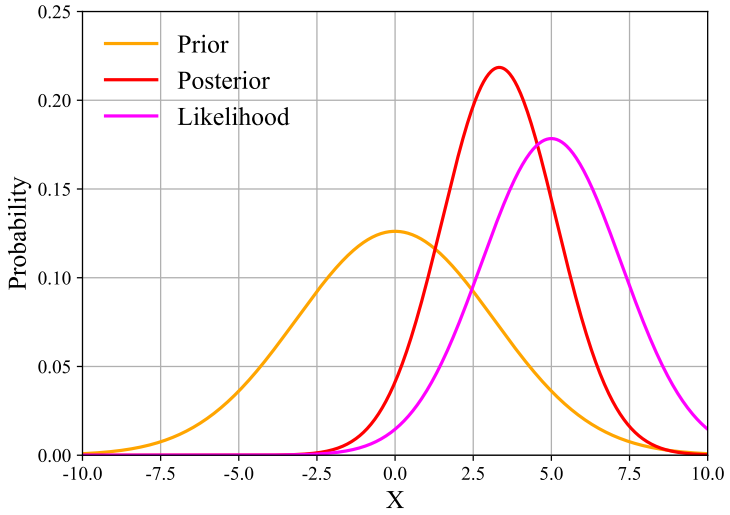


Figure 2.2: Bayesian inference.

Unfortunately however this technique is limited to simple settings, as such approximate inference schemes become necessary in generalising to more complex settings. These approximate inference schemes can be either stochastic in nature such as the sampling based techniques or analytic in nature such as the variational based techniques.

2.1.3 Variational Inference

Approximate variational inference originates from the calculus of variations and is intimately related to information theory. The founding concept of information theory being the measure of entropy, which can be interpreted as the average information content of an entire probability distribution. This information being maximised by probability distributions with the greatest uncertainty, which over an infinite range is achieved by a Gaussian distribution. Entropy forms the basis from which the remaining components of variational inference can be derived, and is defined as follows:

$$H(D) = -\int P(D) \log P(D)\ dD \tag{2.5}$$

Importantly though it were tractable this would enable one to quantify the average information content in the posterior distribution, defined over the unobserved parameter variables given the observed data variables. Through the related measure of relative entropy one would also be able to quantify the average information loss, when substituting the posterior distribution with an approximate posterior distribution, defined only over the unobserved parameter variables. This information loss could then be minimised, and in doing so a closer approximation would be achieved. Relative entropy which is more commonly known as the Kullback–Leibler divergence, is defined in the context of machine learning as follows:

$$KL(Q||P) = \int Q(\theta) \log \frac{Q(\theta)}{P(\theta|D)}\ d\theta \tag{2.6}$$

This is illustrated in Figure 2.3 below which visualises certain properties of the Kullback-Leibler divergence. The most notable of which is the property of non-negativity one common to all measures, in addition to this is the property of asymmetry making it a measure of divergence rather than distance.

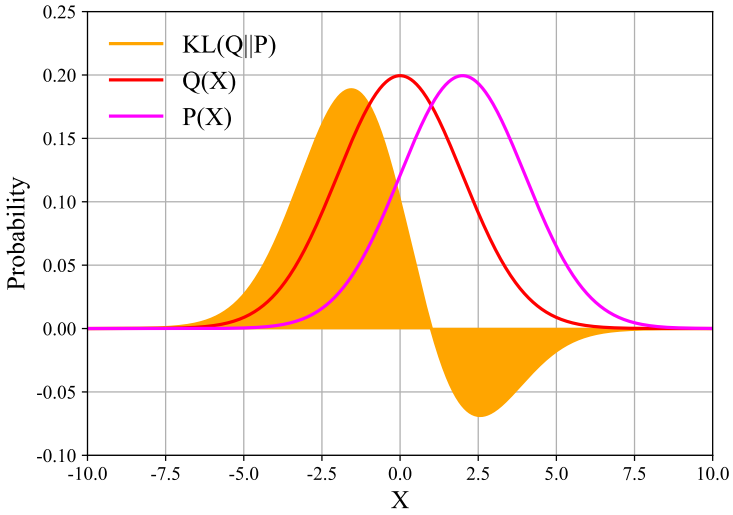


Figure 2.3: Kullback–Leibler divergence.

Crucially although the Kullback-Leibler divergence is intractable to directly minimise, it can be tractable to indirectly minimise. This is achieved by instead maximising a related measure known as the evidence lower bound, which consists of a probability weighted integral across the approximate posterior distribution over two distinct terms [15]. The first term being the log likelihood distribution otherwise known as the measure of cross entropy, which enables one to quantify the average information between the empirical distribution and the predicted distribution. Cross entropy is also known as the logistic loss owing to its use as a common loss function in logistic regression problems. The last term being simply the difference between the log prior distribution and the log approximate posterior distribution. Collectively these terms form the evidence lower bound, defined in the context of machine learning as follows:

$$L(Q) = \int Q(\theta)[\log P(D|\theta) + \log P(\theta) - \log Q(\theta)]\ d\theta \tag{2.7}$$

Alternatively the probability weighted integral can by definition be expressed as the expectation across the approximation posterior distribution, which can be estimated through the sampling:

$$L(Q) = E_Q[\log P(D|\theta) + \log P(\theta) - \log Q(\theta)] \tag{2.8}$$

Expanding the definition of the Kullback-Leibler divergence reveals the nature of its relationship to, and the reasoning behind name of the evidence lower bound:

$$\begin{aligned} KL(Q||P) &= \int Q(\theta) \log \frac{Q(\theta)}{P(\theta|D)} \, d\theta \\ KL(Q||P) &= \int Q(\theta) \log \frac{Q(\theta)P(D)}{P(\theta, D)} \, d\theta \\ KL(Q||P) &= -\int Q(\theta)[\log P(\theta, D) - \log Q(\theta)] \, d\theta + \log P(D) \\ KL(Q||P) &= -\int Q(\theta)[\log P(D|\theta) + \log P(\theta) - \log Q(\theta)] \, d\theta + \log P(D) \end{aligned} \tag{2.9}$$

As a result of the Kullback-Leibler divergence exhibiting the property of non-negativity and the log evidence distribution not depending on the approximate posterior distribution, the following equation shows that the evidence lower bound is in fact a lower bound on the log evidence distribution, and that maximising it does indeed minimise the Kullback-Leibler divergence:

$$\log P(D) = KL(Q||P) + L(Q) \tag{2.10}$$

This relationship is illustrated in Figure 2.4 below which demonstrates how this interdependent maximisation and minimisation process, always cancels out to equal a constant value defined by the log evidence distribution.

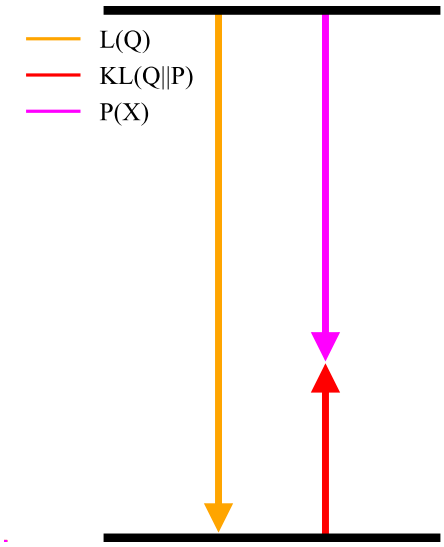


Figure 2.4: Evidence lower bound.

Critically though as one can chose the family of approximate posterior distributions over which the variational inference is performed, one can convert the intractable minimisation of the Kullback-Leibler divergence which requires access to the posterior distribution, into a tractable maximisation of the evidence lower bound which doesn't require access to it. It is important to note that the Kullback-Leibler divergence is just one possible, albeit the most prevalent divergence measure that can be minimised in variational inference. Furthermore variational inference is one among many projection techniques that convert an intractable unnormalised posterior distribution, into a tractable normalised approximate posterior distribution [1].

2.2 Artificial Neural Networks

The essence of artificial neural networks is captured in their ability to approximate any continuous function arbitrarily well. This ability is formalised in the universal approximation theorems, which state that artificial neural networks of various architectures, with either an arbitrary width and fixed depth [16] or an arbitrary depth and fixed width [17], given the appropriate weights can approximate any continuous function. They do not however reveal insights into which particular architecture is ideal, explore whether greater width or depth is preferable, nor provide a construction for the weights but merely state that a construction is possible. Theirin lies the difficulty of developing effective artificial neural networks, exploring a vast expanse of infinite possible configurations.

2.2.1 Architecture

The architecture of artificial neural networks at the most fundamental level, consists of a collection of neurons connected together into layers of three distinct types. The neurons themselves receive weighted incoming connections and produce weighted outgoing connections, with the weights encoding the relationship between the inputs and outputs. The input layer simply receives the empirical input, whilst the hidden layers apply transformations and the output layer simply produces the predicted output. This architecture is illustrated in Figure 2.5 as a graphical structure, whose nodes correspond to the neurons and whose edges correspond to the connections along with their associated weights.

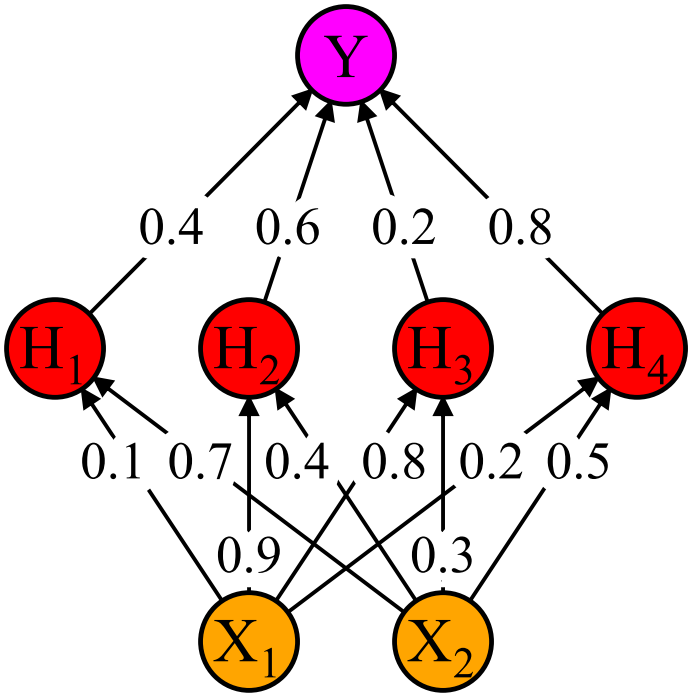


Figure 2.5: Artificial neural network.

Various architectures exist the most general of which being the fully connected neural network, which requires that every neuron in a layer be connected to every neuron in the succeeding layer [18]. Although capable of solving a diverse set of problems, often there exists specialised architectures more effective at solving particular problems. Convolutional neural networks utilise sparse connections and weight sharing in order to gain translational equivariance [19], making them ideal for spatial problems such as image recognition. Recurrent neural networks instead utilise cyclic connections in order to gain internal memory [20], making them ideal for temporal problems such as speech recognition. Generally any of these architectures can function in either of the two fundamental machine learning paradigms. Supervised learning requires empirical input and output pairs in order to learn a functional relationship between the two, whereas unsupervised learning requires only an empirical input in order to learn a compact representation of it. Within these paradigms both generative and discriminative models are possible, using either parametric or nonparametric techniques to solve classification or regression problems.

2.2.2 Operation

The operation of artificial neural networks at the most elementary level, is comprised of two alternating phases in which information flows through the network in opposite directions. The forward propagation phase used for prediction is initiated at the input layer, where the empirical input is propagated to each succeeding layer in the network, in order to produce the predicted output [21]. This propagation involves each neuron receiving activations from neurons in the preceding layer, applying a transformation to these activations, and thereby producing an activation for neurons in the succeeding layer. The transformation applied at each neuron in every layer consists of a linear transformation, in the form of a summation over the activations multiplied by the corresponding weights and the addition of a bias:

$$z_j = \sum_{i=1}^{n} w_{ij} a_i + b_j \tag{2.11}$$

The following transformation is also applied at each neuron in a hidden layer and consists of a nonlinear transformation, in the form of an activation function applied to the summation. A variety of activation functions exist including the rectified linear unit, sigmoid function and hyperbolic tangent:

$$a_j = \phi(z_j) \tag{2.12}$$

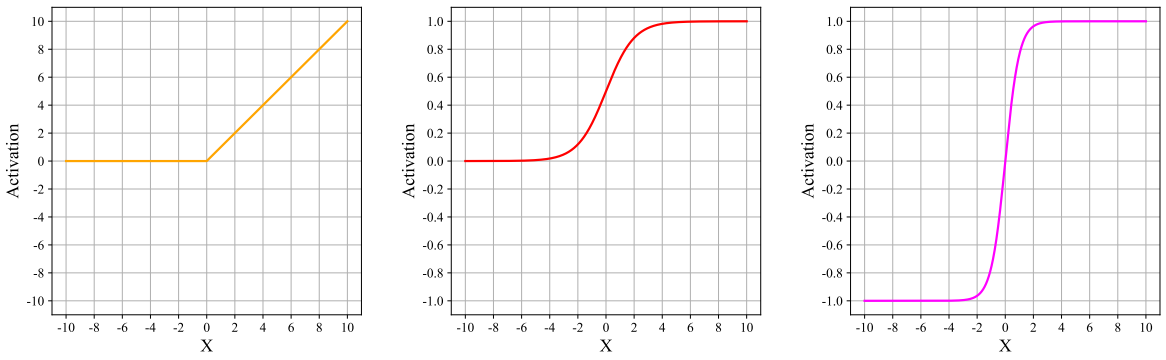
$$\phi(z_j) = \begin{Bmatrix} 0 & z_j \leq 0 \\ z_j & z_j > 0 \end{Bmatrix}, \quad \phi(z_j) = \frac{1}{1 + e^{-z_j}}, \quad \phi(z_j) = \frac{e^{z_j} - e^{-z_j}}{e^{z_j} + e^{-z_j}} \tag{2.13}$$

The output layer instead typically applies either the identity function in regression problems or the softmax function in classification problems. The softmax function being a multidimensional generalisation of the logistic function, used to normalise the outputs into forming a probability distribution:

$$\hat{y}_j = \sigma(z_j) \tag{2.14}$$

$$\sigma(z_j) = \frac{z_j}{\sum_{i=1}^{n} z_i} \tag{2.15}$$

Ultimately the result of this process is a series of function compositions, the nonlinear component of which is crucial to the continuous function approximation ability. The choice of activation function plays a pivotal role in the networks ability to overcome the vanishing gradient problem, with the rectified linear unit typically suffering the least from it, owing to its constant gradient [22]. The activation functions above are illustrated in Figure 2.6 displaying the distinctive quality of the rectified liner unit.



(a) Rectified linear unit (b) Sigmoid function (c) Hyperbolic tangent

Figure 2.6: Activation functions.

The backward propagation phase used for training is initiated at the output layer, where the partial derivative of the loss with respect to either the weights or biases, is propagated to each preceding layer in the network, in order to optimise these parameters. The loss function evaluates the performance of the network by comparing the empirical output with the predicted output. Typically quadratic loss is used in regression problems, whilst logistic loss is used in classification problems. The logistic loss function is used to ignore the contribution of all outputs except the predicted output:

$$L = -\sum_{j=1}^{n} y_j \log \hat{y}_j \tag{2.16}$$

This propagation involves each parameter receiving its partial derivative from parameters in the succeeding layer, applying the chain rule to these partial derivatives, and thereby producing a partial derivative for the parameters in the preceding layer. The chain rule applied at each parameter decomposes the partial derivative of a composite function, in terms of the partial derivatives of its components:

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial \hat{y}_j}\frac{\partial \hat{y}_j}{\partial z_j}\frac{\partial z_j}{\partial w_{ij}}, \quad \frac{\partial L}{\partial b_j} = \frac{\partial L}{\partial \hat{y}_j}\frac{\partial \hat{y}_j}{\partial z_j}\frac{\partial z_j}{\partial b_j} \tag{2.17}$$

Applying this process of auto-differentiation to the logistic loss function above, derives two remarkably similar partial derivatives for the weight and bias parameters:

$$\frac{\partial L}{\partial w_{ij}} = (\hat{y}_j - y_j)a_i, \quad \frac{\partial L}{\partial b_j} = (\hat{y}_j - y_j) \tag{2.18}$$

Combining auto-differentiation with gradient descent enables the optimisation, with the magnitude of adjustment being controlled by the learning rate hyperparameter:

$$w_{ij} = w_{ij} - \eta\frac{\partial L}{\partial w_{ij}}, \quad b_j = b_j - \eta\frac{\partial L}{\partial b_j} \tag{2.19}$$

Accelerating the convergence of gradient descent which uses the entirety of the data set in each epoch, is stochastic gradient descent which instead uses subsets of the data set in each epoch. Further improvement is achieved through the use of more advanced optimisers such as Adam [23], which more effectively explore the loss function space illustrated in Figure 2.7 visualising both local and global optima.

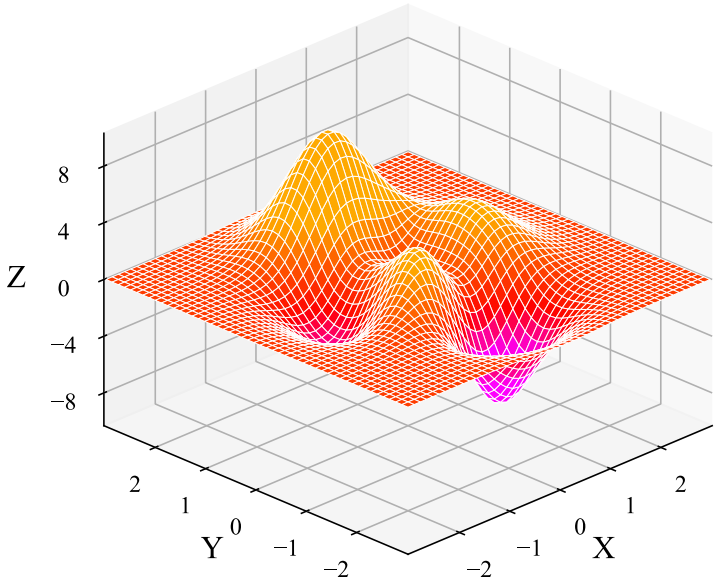


Figure 2.7: Loss function space.

2.3 Bayesian Neural Networks

The extension of artificial neural networks from a frequentist scheme to that of a Bayesian scheme, with the inclusion of uncertainty in the parameters and predictions, produces a Bayesian neural network. Constructed through the fusion of Bayesian networks and artificial neural networks, with the intention being to combine the probabilistic guarantees and continuous function approximation of each, into a single model [24]. Whilst artificial neural networks effectively perform maximum likelihood estimation, Bayesian neural networks naturally perform Bayesian inference, thereby representing the parameters of the network not as point estimates but rather as probability distributions. In doing so this not only provides probabilistic guarantees in high risk settings, it also improves performance in low data settings. Ultimately the ambition is to enable machines to know, what they don't know.

2.3.1 Architecture

The architecture of Bayesian neural networks incorporates all the traits of artificial neural networks, whilst expanding upon these to also include those of Bayesian models. A Bayesian neural network is defined by a posterior distribution, produces a likelihood distribution and is constrained by a prior distribution. The posterior distribution is in practice an approximate posterior distribution, defined over the joint distribution of the weight variables given the joint distribution of the parameter variables. This approximate posterior distribution is sampled in order to create an instance of an artificial neural network with a particular set of weights. This architecture is illustrated in Figure 2.8 as a graphical structure, whose parameters capture vastly more information about the weights, whilst typically only doubling the amount of variables.

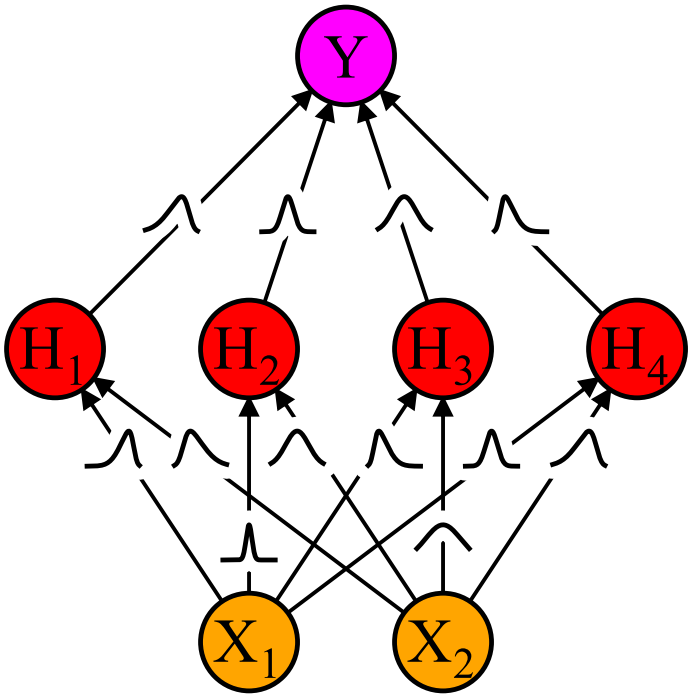


Figure 2.8: Bayesian neural network.

These parameters in fact represent an uncountably infinite ensemble of artificial neural networks, where each instance has a particular set of weights and therefore encodes a different relationship between the empirical input and predicted output. In doing so the probability distribution over the parameters, induces a probability distribution over the predicted outputs. From the process of training not a single but an ensemble of artificial neural networks, emerges a principled technique of Bayesian regularisation. In contrast the frequentist techniques of L1 and L2 regularisation, are equivalent to using a Laplace or Gaussian prior distribution respectively and performing maximum a posteriori estimation. One could consider this an implementation of a Bayesian Occam's razor, which penalises both overly simple and overly complex artificial neural networks [25].

2.3.2 Operation

The operation of Bayesian neural networks incorporates all the principles of artificial neural networks, whilst expanding upon these to also include those of Bayesian inference. The forward propagation phase differs in that it first reparameterises the weights in terms of a deterministic and differentiable function, which itself depends upon an unbiased random variable. This reparameterisation trick transforms samples of the parameter free unbiased random variable, along with the parameters of the approximate posterior distribution, into samples of the approximate posterior distribution [26]. In doing so this creates instances of artificial neural networks which together estimate the Bayesian neural network, with the accuracy of this estimation increasing with the number of samples:

$$\begin{aligned} p(\mathcal{E}) \\ g(\mathcal{E}, \theta) = W \\ f(W, \theta) = \log P(D|W) + \log P(W) - \log Q(W|\theta) \end{aligned} \tag{2.20}$$

As is standard the empirical input is propagated through the network, in order to produce a predicted output and so derive the logistic loss. However this alone is no longer sufficient as a loss function, instead the evidence lower bound is used as the loss function. The log likelihood distribution is provided by the networks as the logistic loss, whilst the difference between the log prior distribution and the log approximate posterior distribution is provided by evaluating the samples under those probability distributions. All of these terms from the ensemble of networks are collected together, in order to provide an estimation of the evidence lower bound:

$$L(Q) = E_Q[\log P(D|W) + \log P(W) - \log Q(W|\theta)] \tag{2.21}$$

$$L(Q) \approx \sum_{i=1}^{n} \log P(D|W_i) + \log P(W_i) - \log Q(W_i|\theta) \tag{2.22}$$

The backward propagation phase also differs in that it first converts the partial derivative of an expectation into an expectation of partial derivatives, for unbiased estimates of the partial derivative of the evidence lower bound with respect to the parameters. It achieves this by redefining the expectation across the approximate posterior distribution to be an expectation across the unbiased random variable:

$$\begin{aligned} \frac{\partial L(Q)}{\partial \theta} &= \frac{\partial}{\partial \theta} E_Q[f(W, \theta)] \\ \frac{\partial L(Q)}{\partial \theta} &= \frac{\partial}{\partial \theta} \int Q(W|\theta) f(W, \theta) \; dW \\ \frac{\partial L(Q)}{\partial \theta} &= \frac{\partial}{\partial \theta} \int p(\mathcal{E}) f(W, \theta) \; d\mathcal{E} \\ \frac{\partial L(Q)}{\partial \theta} &= E_p \left[\frac{\partial f(W, \theta)}{\partial W} \frac{\partial W}{\partial \theta} + \frac{\partial f(W, \theta)}{\partial \theta} \right] \end{aligned} \tag{2.23}$$

In the same way as usual the partial derivatives are propagated through the network, in order to optimise the parameters and adjust the network. However rather than merely adjust point estimates this adjusts probability distributions, therefore an uncountably infinite ensemble of artificial neural networks is being trained simultaneously. This formulation also does not depend on a closed form expression for the evidence lower bound, allowing for a variety of prior distribution and approximate posterior distribution families to be used. This allows for great flexibility in the variational inference scheme that is applied, enabling the use of novel schemes with unique properties.

2.4 Variational Inference Schemes

The variational inference scheme specifies the family of approximate posterior distributions over which variational inference is performed. It is within this family of approximate posterior distributions that the optimisation occurs, if the posterior distribution happens to be a member of this family then it will be inferred, otherwise the member of the family most closely approximating it will be inferred [27]. Therefore it is commonly believed that the type of variational inference scheme applied, is a significant factor in the performance of the model being developed. This has led to the development of many different schemes, with a compromise occurring between the accuracy of the approximation and the simplicity in its computation.

2.4.1 Mean Field Variational Inference

Mean field variational inference has its origins in spin glass theory and its use of mean field theory to study spin glass models [28]. Mean field variational inference assumes that the approximate posterior distribution, can be factorised into independent probability distributions over the parameters [29]. The approximate posterior distribution can alternatively be viewed as having a diagonal covariance matrix. This converts a many-body problem into a one-body problem, by averaging over the degrees of freedom and approximating the effect of parameters on each other by that of a single averaged effect:

$$Q(W|\theta) = \prod_{i=1}^{I} Q(W_i|\theta_i) \tag{2.24}$$

The major disadvantage of this inference scheme, is in fact its assumption of independence amongst the parameters of the network. As such this explicitly ignores correlations amongst the parameters, which is not ideal as strong correlations do typically exist. However not only are the correlations not estimated, as a consequence of this the variances tend to be underestimated too.

2.4.2 Global Inducing Point Variational Inference

Global inducing point variational inference has its origins in deep Gaussian process literature and its use of local inducing point variational inference to train deep Gaussian process models [6]. Global inducing point variational inference defines a set of global inducing points, which act as pseudo data points in order to induce an approximate posterior distribution [5]. The global inducing points are defined at the input layer and propagated through the network in an alternating process, which produces the weights of the current layer from the the inducing points of the preceding layer, and produces the inducing points of the succeeding layer from the weights of the current layer. This structure enables the optimal approximate posterior distribution over the top layer weights to be derived, exhibiting strong dependencies on lower layers weights and from which a correlated approximate posterior distribution over all layer weights is developed:

$$Q(\{W_l\}_{l=1}^{L+1}) = \prod_{l=1}^{L+1} Q(W_l|\{W_l\}_{l=1}^{l-1}) \tag{2.25}$$

A number of advantages arise from this inference scheme, the most significant of which is that it models cross layer correlations. In contrast to local inducing points which are defined separately at each layer, global inducing points are defined jointly for all layers, and therefore model not only the input to output transformation of individual layers but the input to output transformation of the whole network. Furthermore as a consequence of using inducing points, the approximate posterior distribution is defined over function space rather than parameter space, which provides a more intuitive interpretation as it is the function rather than the parameters that is usually of interest. Also since Bayesian neural networks are a special case of deep Gaussian processes with a particular choice of kernel, this creates a unifying inference scheme across both deep learning classes, allowing for consistent comparisons of the two.

2.5 Continual Learning

The continual learning method seeks the ability to acquire knowledge not just of a single task but to accumulate knowledge from multiple sequential tasks, whilst only ever having access to the current task data set. It is a generalised form of the online learning technique in which learning is performed on individual data points sequentially, in contrast to the offline learning technique in which learning is performed on all data points simultaneously [30]. Continual learning expands the constraints of online learning, from learning from individual data points, to learning from individual task data sets. Although a fully online approach may still be retrieved, if the task data set contains only a single data point. Continual learning methods generally belong to one of three categories, model based methods which operate by modifying the model architecture, rehearsal based methods which operate by rehearsing pseudo data and inference based methods which operate by performing inference over parameters.

2.5.1 Synaptic Intelligence

Synaptic intelligence uses a regularised maximum likelihood estimation of the parameters at each task. This uses a lambda term to control the overall regularisation of the current task parameters towards the previous task parameters, as well as a sigma term to control the relative regularisation of each parameter:

$$L_t(\theta) = \log P(D_t|\theta) - \tfrac{1}{2}\lambda_t(\theta - \theta_{t-1})^T\Sigma_{t-1}^{-1}(\theta - \theta_{t-1}) \tag{2.26}$$

The sigma term is estimated using a measure of importance of each parameter to each task [31]. This is dictated by a comparison between the rate of change in the gradient of the regularised likelihood and the gradient of the parameters.

2.5.2 Elastic Weight Consolidation

Elastic weight consolidation similarly uses a regularised maximum likelihood estimation of the parameters at each task. This differs though in its use of a projection technique known as Laplace's approximation, which results in a recursion for the sigma term:

$$L_t(\theta) = \log P(D_t|\theta) - \tfrac{1}{2}\lambda_t(\theta - \theta_{t-1})^T\Sigma_{t-1}^{-1}(\theta - \theta_{t-1}), \quad \Sigma_t^{-1} = \Phi + \Sigma_{t-1}^{-1} \tag{2.27}$$

The sigma term is estimated using the average Hessian of the likelihood distribution rather than the full Hessian of the likelihood distribution [32]. This is achieved using a measure of the Fisher information, derived from the previous task parameters.

2.5.3 Variational Continual Learning

Variational continual learning uses variational inference over the approximate posterior distribution of the parameters at each task. This leverages the intrinsically continual process emerging from the recursive application of Bayesian inference, which combines what the current task data indicates about the parameters, with what previous task data indicated about the parameters. Recursive application of Bayesian inference is as simple as using the approximate posterior distribution of the previous task, as the prior distribution when inferring the approximate posterior distribution of the current task. Thereby constraining the inference process, by limiting the adjustment of parameters most influential in prediction, whilst still allowing the adjustment of other parameters:

$$P(\theta|D_t) \propto P(D_t|\theta)P(\theta|D_{t-1}) \approx Q_t(\theta) \tag{2.28}$$

This method can be extended with the inclusion of a subset of previous task data known as a core set, although this does not allow for the use of the model in a strictly continual setting. The core set is analogous to an episodic memory for the model, capturing the essential information of each task, refreshing the model's knowledge of them, thereby increasing performance:

$$P(\theta|D_t) \propto P(\theta|D_t \setminus C_t)P(C_t|\theta) \approx Q_t(\theta)P(C_t|\theta) \tag{2.29}$$

In practice the use of Bayesian neural networks makes exact Bayesian inference intractable, so instead approximate variational inference is used. In the original variational continual learning research paper [1], the mean field variational inference scheme is used, in conjunction with the reparameterisation trick. However in the improved variational continual learning research paper [33], the same inference scheme is used but in conjunction with the local reparameterisation trick. Alongside this it also adjusts certain hyperparameters and performs model reinitialisation after each task, although this does not allow for the use of the model in a fully online setting. In either case the objective is to maximise the evidence lower bound, in which after the initial task the prior distribution, is replaced by the approximate posterior distribution of the previous task:

$$L_t(Q_t) = E_{Q_t}[\log P(D_t|\theta) + \log Q_{t-1}(\theta) - \log Q_t(\theta)] \tag{2.30}$$

Uncertainty is extremely useful in determining the importance of information from previous tasks, therefore the use of an inference scheme which more accurately estimates this may improve performance. Substituting the mean field variational inference scheme which ignores all correlations in uncertainty, for the global inducing point variational inference scheme which captures cross layer correlations in uncertainty, should provide more accurate estimations. Furthermore as global inducing points act as pseudo data points, they could also behave as an episodic memory serving the same purpose as core sets, whilst simply being optimised parameters of the model and remaining strictly continual.

Chapter 3

Execution

3.1 Standard MNIST

The execution of this project began by developing a traditional instead of continual learning model, consisting of a Bayesian neural network operating with either a mean field or global inducing point variational inference scheme, capable of solving a single task in the form of the standard MNIST problem. The standard MNIST problem involves predicting the value of handwritten digits ranging from 0 to 9, using a data set which contains a total of 60,000 training images and 10,000 testing images. The images are 28 x 28 pixels in size and as illustrated in Figure 3.1, use a greyscale colour palette in conjunction with anti-aliasing.

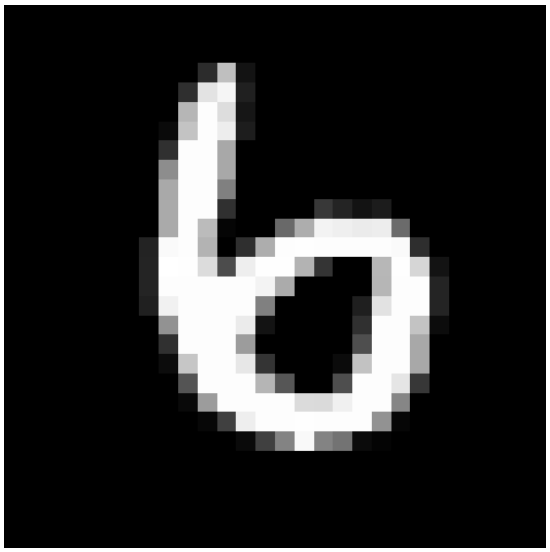


Figure 3.1: Standard data.

3.1.1 Implementation

Implementing a Bayesian neural network presents a series of nuanced differences when compared to an artificial neural network. The most prominent of which is the addition of a sampling dimension which dictates the number of samples of the approximate posterior, alongside the conventional batch dimension which determines the size of the batches in stochastic gradient descent, and the feature dimension which describes the number of elements in the empirical input. Not only this but the evidence lower bound must be computed rather than simply the log likelihood distribution, which requires that the difference between the log prior distribution and the log approximate posterior distribution also be computed.

3.1.2 Control Flow

The control flow is dictated by the main function which begins by configuring a summary writer, enabling a number of metrics to be logged including the evidence lower bound, logistic loss and predictive accuracy. After this it proceeds to load the data, construct the network and then initiate its training and testing.

```
def main(args):
    log_directory = get_log_directory(args)
    summary_writer = SummaryWriter(str(log_directory), flush_secs=5)

    train_loader, test_loader, data_size = get_data(args.batch_size)
    net = get_net(args.type, args.inducing_size).to(device)

    train(net, args.epochs, args.learning_rate, args.train_samples,
          args.test_samples, train_loader, test_loader, data_size,
          args.log_frequency, args.val_frequency, summary_writer)
    summary_writer.close()
```

Algorithm 3.1: Traditional main function.

3.1.3 Data Augmentation

The data augmentation is performed by the get data function which first transforms the data to a tensor, and then flattens it such that it can be propagated through the network. It then sets the batch size and shuffles the training data but not the testing data, creating a consistent testing scheme across executions.

```
def get_data(batch_size):
    transforms = Compose([ToTensor(), Flatten()])

    train_dataset = MNIST("inputs", train=True, download=True, transform=transform)
    test_dataset = MNIST("inputs", train=False, download=True, transform=transform)

    train_loader = DataLoader(train_dataset, shuffle=True, batch_size=batch_size,
                              pin_memory=True, num_workers=cpu_count())
    test_loader = DataLoader(test_dataset, shuffle=False, batch_size=batch_size,
                             pin_memory=True, num_workers=cpu_count())

    data_size = len(train_dataset)
    return train_loader, test_loader, data_size
```

Algorithm 3.2: Traditional data function.

3.1.4 Network Architecture

The network architecture is constructed by the get net function which can implement either a mean field or global inducing point variational inference scheme. The architecture consists of a 784 wide input layer, two 100 wide hidden layers, a 10 wide output layer and rectified linear units in between. Additionally the global inducing point variety sets the inducing size, which is the number of global inducing points used.

```
def get_net(type, inducing_size):
    if (type == "global"):
        fc1 = bf.GILinear(in_features=784, out_features=100,
                          inducing_batch=inducing_size, bias=True, full_prec=True)
        fc2 = bf.GILinear(in_features=100, out_features=100,
                          inducing_batch=inducing_size, bias=True, full_prec=True)
        fc3 = bf.GILinear(in_features=100, out_features=10,
                          inducing_batch=inducing_size, bias=True, full_prec=True)
        net = nn.Sequential(fc1, nn.ReLU(), fc2, nn.ReLU(), fc3)
        net = bf.InducingWrapper(net, inducing_batch=inducing_size,
                                 inducing_shape=(inducing_size, 784))
        return net

    elif (type == "factorised"):
        fc1 = bf.FactorisedLinear(in_features=784, out_features=100, bias=True)
        fc2 = bf.FactorisedLinear(in_features=100, out_features=100, bias=True)
        fc3 = bf.FactorisedLinear(in_features=100, out_features=10, bias=True)
        net = nn.Sequential(fc1, nn.ReLU(), fc2, nn.ReLU(), fc3)
        return net
```

Algorithm 3.3: Traditional network function.

3.1.5 Training Stage

The training stage is executed by the train function which cycles through each epoch, and through each batch of the training data, performing both forward and backward propagation each time. The Adam optimiser is used for the backward propagation and sets its learning rate. At regular intervals this training stage is interrupted to initiate a testing phase in order to assess the performance on the testing data.

```
def train(net, epochs, learning_rate, train_samples, test_samples, train_loader,
          test_loader, data_size, log_frequency, val_frequency, summary_writer):
    step = 0
    optimizer = optim.Adam(net.parameters(), lr=learning_rate)
    for epoch in range(epochs):
        net.train()
        for batch, targets in train_loader:
            batch, targets = batch.to(device), targets.to(device)
            logits, logpq, _ = bf.propagate(net, batch.expand(train_samples, -1, -1))

            probs = Categorical(logits=logits)
            logloss = probs.log_prob(targets).mean(1)

            elbo = -(logloss.mean() + logpq.mean() / data_size)
            elbo.backward()

            optimizer.step()
            optimizer.zero_grad()

            if ((step + 1) % log_frequency == 0):
                summary_writer.add_scalar("Elbo", elbo.item(), step)
            step += 1

        if ((epoch + 1) % val_frequency == 0):
            test(net, epoch, step, test_samples, test_loader, summary_writer)
```

Algorithm 3.4: Traditional train function.

3.1.6 Testing Stage

The testing stage is executed by the test function which cycles through only one epoch and performs only forward propagation. As such the tracking of gradients is disabled to prevent unnecessary computation. Furthermore rather than compute the evidence lower bound the predictive accuracy is computed instead.

```
def test(net, epochs, steps, test_samples, test_loader, summary_writer):
    total_logloss = []
    total_accuracy = []
    with torch.no_grad():
        net.eval()
        for batch, targets in test_loader:
            batch, targets = batch.to(device), targets.to(device)
            logits, _, _ = bf.propagate(net, batch.expand(test_samples, -1, -1))
            logits = logits.log_softmax(-1).logsumexp(0)

            probs = Categorical(logits=logits)
            logloss = -(probs.log_prob(targets).mean(0))

            preds = logits.argmax(1)
            accuracy = (preds == targets).float().mean()

            total_logloss.append(logloss.item())
            total_accuracy.append(accuracy.item())

        average_logloss = torch.tensor(total_logloss).mean().item()
        average_accuracy = torch.tensor(total_accuracy).mean().item()

        summary_writer.add_scalar("Loss", average_logloss, steps)
        summary_writer.add_scalar("Accuracy", average_accuracy, steps)

        print(f"Epoch: [{epochs + 1}] Loss: {average_logloss:.5f}
              Accuracy: {average_accuracy:.5f}")
```

Algorithm 3.5: Traditional test function.

3.2 Permuted MNIST

The execution of this project transitioned to expanding from a traditional to continual learning model, capable of solving multiple sequential tasks in the form of the permuted MNIST problem. The permuted MNIST problem also involves predicting the value of handwritten digits ranging from 0 to 9, instead though it uses ten data sets constructed from the standard MNIST data set, by applying ten separate random permutations to the pixels of the training images and testing images, creating ten separate tasks. The images are still 28 x 28 pixels in size and as illustrated in Figure 3.2, appear indistinguishable from noise but actually contain the same distribution of greyscale values as their standard counterpart, and as such can still be distinguished from one another.

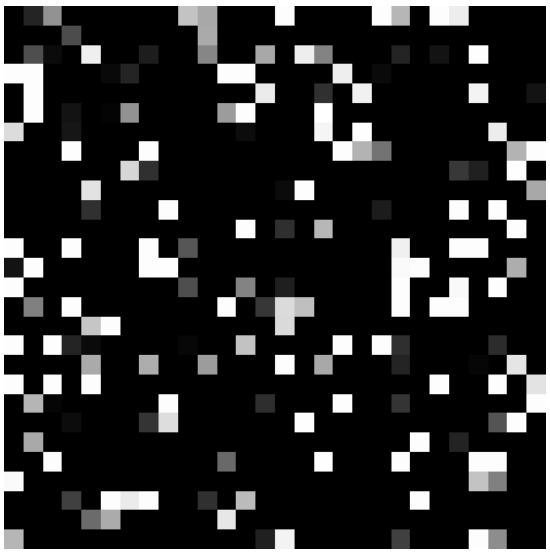


Figure 3.2: Permuted data.

3.2.1 Implementation

Implementing a continual learning model also presents a series of nuanced differences when compared to a traditional learning model. The most essential of which is that the training stage has to cycle through each task in addition to each epoch and each batch, whilst the testing stage has to cycle through all previous tasks. This is so that it can compute the predictive accuracy over all the previous tasks instead of just the current task, as the model must be capable of solving all tasks simultaneously whilst only encountering them sequentially. In addition to this the inference step must be altered to use the approximate posterior distribution of the previous task instead of the prior distribution.

3.2.2 Control Flow

The control flow now executes functions which have been modified to handle multiple sequential tasks.

```
1  def main(args):
2      log_directory = get_log_directory(args)
3      summary_writer = SummaryWriter(str(log_directory), flush_secs=5)
4
5      train_loaders, test_loaders, data_size = get_data(args.tasks, args.batch_size)
6      net = get_net(args.type, args.inducing_size).to(device)
7
8      train(net, args.tasks, args.epochs, args.learning_rate, args.train_samples,
9            args.test_samples, train_loaders, test_loaders, data_size,
10           args.log_frequency, args.val_frequency, summary_writer)
11     summary_writer.close()
```

Algorithm 3.6: Continual main function.

3.2.3 Data Augmentation

The data augmentation now performs random permutations to the data to create ten separate data sets.

```
def get_data(tasks, batch_size):
    transforms = [Compose([ToTensor(), Flatten(), Permute(torch.randperm(784))])
                  for _ in range(tasks)]

    train_datasets = [MNIST("inputs", train=True, download=True, transform=transform)
                      for transform in transforms]
    test_datasets = [MNIST("inputs", train=False, download=True, transform=transform)
                     for transform in transforms]

    train_loaders = [DataLoader(train_datasets[task], shuffle=True, batch_size=batch_size,
                     pin_memory=True, num_workers=cpu_count()) for task in range(tasks)]
    test_loaders = [DataLoader(test_datasets[task], shuffle=False, batch_size=batch_size,
                    pin_memory=True, num_workers=cpu_count()) for task in range(tasks)]

    data_size = [len(train_datasets[task]) for task in range(tasks)]
    return train_loaders, test_loaders, data_size
```

Algorithm 3.7: Continual data function.

3.2.4 Network Architecture

The network architecture now enables previous networks to be copied for use in the inference step.

```
def get_old_net(net):
    old_net = deepcopy(net)
    for param in old_net.parameters():
        param.requires_grad = False
    return old_net
```

Algorithm 3.8: Continual network function.

3.2.5 Training Stage

The training stage now trains the network on multiple tasks using the previous network.

```
def train(net, tasks, epochs, learning_rate, train_samples, test_samples, train_loaders,
          test_loaders, data_size, log_frequency, val_frequency, summary_writer):
    step = 0
    optimizer = optim.Adam(net.parameters(), lr=learning_rate)
    for task in range(tasks):
        old_net = get_old_net(net).to(device)
        for epoch in range(epochs):
            net.train()
            for batch, targets in train_loaders[task]:
                batch, targets = batch.to(device), targets.to(device)
                logits, new_logpq, new_weights = bf.propagate(net, batch.expand(
                                                              train_samples, -1, -1))
                _, old_logpq, _ = bf.propagate(old_net, batch.expand(
                                               train_samples, -1, -1),
                                               sample_dict=new_weights, detach=False)
                logpq = new_logpq if (task == 0) else new_logpq - old_logpq

                probs = Categorical(logits=logits)
                logloss = probs.log_prob(targets).mean(1)

                elbo = -(logloss.mean() + logpq.mean() / data_size[task])
                elbo.backward()

                optimizer.step()
                optimizer.zero_grad()

                if ((step + 1) % log_frequency == 0):
                    summary_writer.add_scalar("Elbo", elbo.item(), step)
                step += 1

            if ((epoch + 1) % val_frequency == 0):
                test(net, task, epoch, step, test_samples, test_loaders, summary_writer)
```

Algorithm 3.9: Continual train function.

3.2.6 Testing Stage

The testing stage now tests the network on multiple tasks without using the previous network.

```
def test(net, tasks, epochs, steps, test_samples, test_loaders, summary_writer):
    total_logloss = []
    total_accuracy = []
    with torch.no_grad():
        for task in range(tasks + 1):
            net.eval()
            for batch, targets in test_loaders[task]:
                batch, targets = batch.to(device), targets.to(device)
                logits, _, _ = bf.propagate(net, batch.expand(test_samples, -1, -1))
                logits = logits.log_softmax(-1).logsumexp(0)

                probs = Categorical(logits=logits)
                logloss = -(probs.log_prob(targets).mean(0))

                preds = logits.argmax(1)
                accuracy = (preds == targets).float().mean()

                total_logloss.append(logloss.item())
                total_accuracy.append(accuracy.item())

        average_logloss = torch.tensor(total_logloss).mean().item()
        average_accuracy = torch.tensor(total_accuracy).mean().item()

        summary_writer.add_scalar("Loss", average_logloss, steps)
        summary_writer.add_scalar("Accuracy", average_accuracy, steps)

        print(f"Task: [{tasks + 1}] Epoch: [{epochs + 1}] Loss: {average_logloss:.5f}
              Accuracy: {average_accuracy:.5f}")
```

Algorithm 3.10: Continual test function.

Chapter 4

Evaluation

4.1 Original Variational Continual Learning

The original variational continual learning research paper introduces a model intentionally restrictive in model capacity, using only two 100 wide hidden layers, which they propose should become standard for the permuted MNIST problem. The reasoning being that almost any sufficiently large model has enough model capacity to trivially learn multiple sequential tasks, as any single task typically consumes only so much of that model capacity. Therefore the use of sufficiently large models doesn't encourage the application of transfer learning nor the efficient allocation of model capacity, which is what is of interest when evaluating continual learning models. The original mean field variational continual learning model sets the hyperparameters at 100 epochs, a 256 batch size and a 0.001 learning rate. The original globally induced variational continual learning model additionally sets hyperparameters of 3 training samples, 10 testing samples and a variable inducing size which is the focus of evaluation.

4.1.1 Results

The following results were computed as an average of five separate executions for a total of seven different inducing sizes. The results displayed in Figure 4.1 constitute the predictive accuracy of the model after it encounters each new task, revealing a general downward trend as one would expect. Interestingly though increasing the inducing size somewhat negates this process, with a significant divergence amongst the different inducing sizes emerging after the sixth task. Another notable aspect is that the predictive accuracy doesn't decrease until after the second task, suggesting that even under this model the model capacity had yet to be saturated.

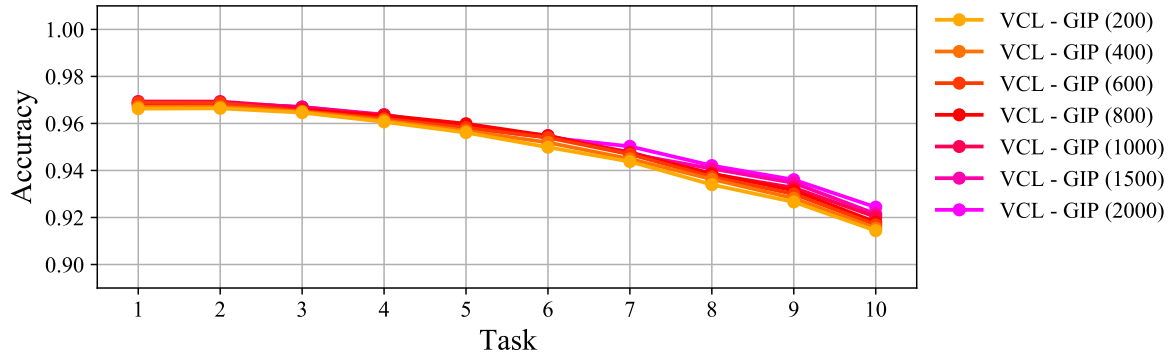


Figure 4.1: Original results.

The results displayed in Table 4.1 express the predictive accuracy of the model after encountering the final task, approaching the theoretical maximum of 97% achieved on a single task. The results reinforce the notion that increasing the inducing size increases the predictive accuracy of the model, reaching 92% when using an inducing size of 600. Furthermore the globally induced model significantly outperforms

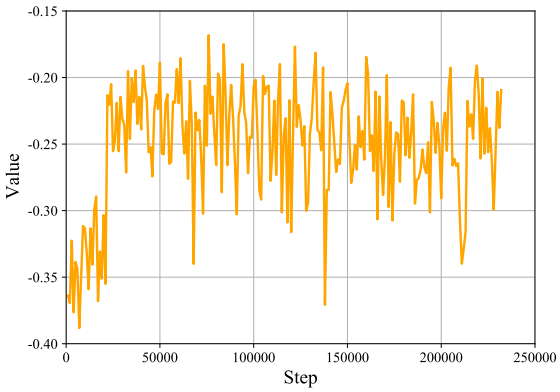
the mean field model by 2% and is only 1% shy of the mean field model using a core set of 200, which doesn't even operate in a strictly continual learning setting.

Model	Predictive Accuracy
VCL - MF (0)	90%
VCL - MF (200)	93%
VCL - GIP (200)	91%
VCL - GIP (400)	91%
VCL - GIP (600)	92%
VCL - GIP (800)	92%
VCL - GIP (1000)	92%
VCL - GIP (1500)	92%
VCL - GIP (2000)	92%

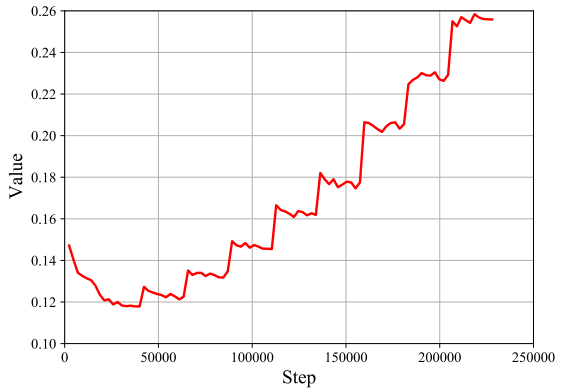
Table 4.1: Original results.

4.1.2 Logs

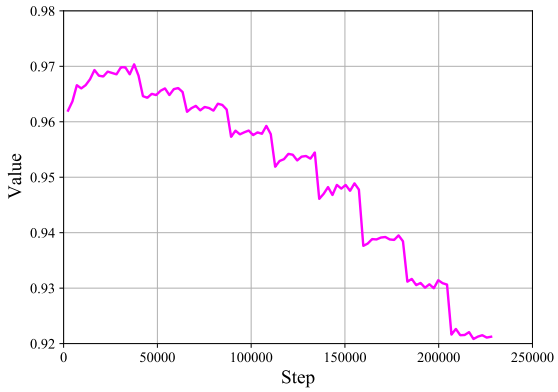
The following logs were computed using an inducing size of 1000, although similar characteristics were observed across all inducing sizes. The evidence lower bound in Figure 4.2 soon converges to a rough value of -0.25, upon which it exhibits highly chaotic behaviour, suggesting the model is struggling to produce an approximate posterior distribution which covers the entire input space. The logistic loss and predictive accuracy initially improve, however after encountering each new task rapidly diminish before slightly improving again, ultimately finishing at values of 0.26 and 0.92 respectively.



(a) Evidence lower bound



(b) Logistic loss



(c) Predictive accuracy

Figure 4.2: Original logs.

4.2 Improved Variational Continual Learning

The improved variational continual learning research paper advances the original model by replacing the reparameterisation trick with the local reparameterisation trick, which greatly reduces variance in the gradients by translating uncertainty about global parameters, into local noise that is independent across data points in a batch. It is currently unknown whether this technique can be applied to the global inducing point inference scheme, due to the inference occurring in function space rather than parameters space. The improved variational continual learning model changes the following hyperparameters to 800 epochs and a 1024 batch size but maintains a 0.001 learning rate. The improved globally induced variational continual learning model adopts these and maintains the additional hyperparameters from the original model.

4.2.1 Results

The following results were likewise computed as an average of five separate executions for a total of seven different inducing sizes. The results displayed in Figure 4.3 reveal a similar downward trend, however one which shows much more consistency across inducing sizes, with a significant divergence amongst the different inducing sizes only emerging after the eighth task, compared to the sixth task before. As such the variance in predictive accuracy after the final task is greatly reduced, suggesting this model would scale better to problems with a larger numbers of tasks.

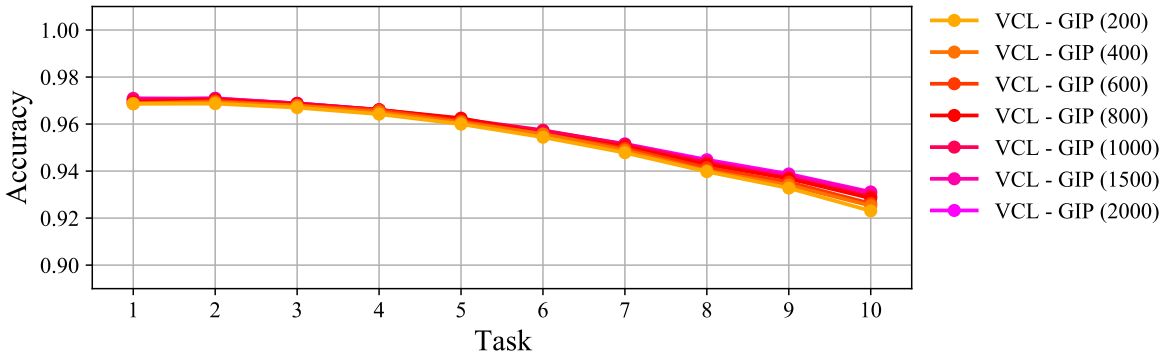


Figure 4.3: Improved results.

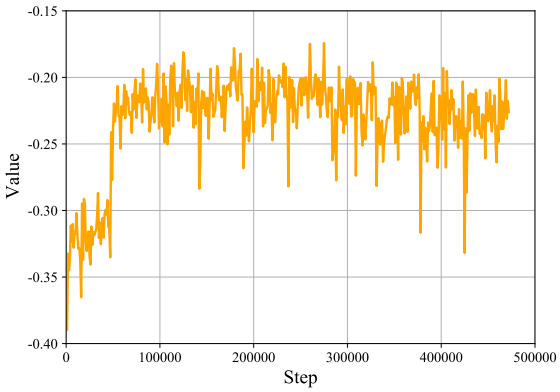
The results displayed in Table 4.2 confirm that longer and larger executions increase the predictive accuracy of the model, now reaching 93% accuracy when using an inducing size of just 400. At the same time though it has enabled the mean field model to also achieve this accuracy, and a 2% gain for the mean field model using a core set of 200, although it is difficult to distinguish how much of this is a consequence of not operating in a fully online setting.

Model	Predictive Accuracy
VCL - MF (0)	93%
VCL - MF (200)	95%
VCL - GIP (200)	92%
VCL - GIP (400)	93%
VCL - GIP (600)	93%
VCL - GIP (800)	93%
VCL - GIP (1000)	93%
VCL - GIP (1500)	93%
VCL - GIP (2000)	93%

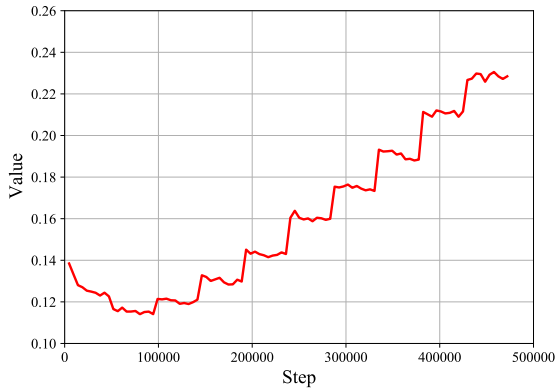
Table 4.2: Improved results.

4.2.2 Logs

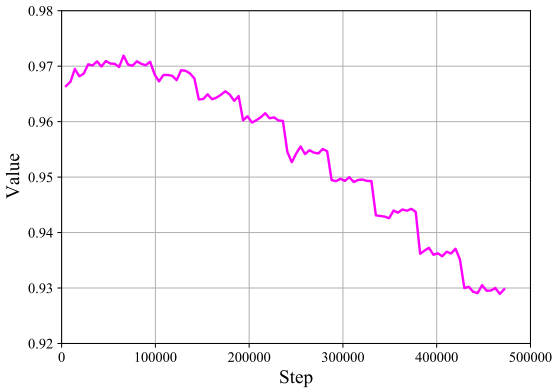
The evidence lower bound in Figure 4.4 converges to a similarly rough value of -0.20, however it exhibits significantly more stable behaviour, suggesting that the more accurate estimations of it provided by the larger batches, produces an approximate posterior distribution which better covers the input space. The logistic loss and predictive accuracy follow the exact same pattern as before, except that they now finish at values of 0.23 and 0.93 respectively.



(a) Evidence lower bound



(b) Logistic loss



(c) Predictive accuracy

Figure 4.4: Improved logs.

Chapter 5

Conclusion

5.1 Current Achievements

This project has demonstrated the feasibility as well as the effectiveness of a globally induced variational continual learning model, with it significantly exceeding the performance of the original mean field model and matching the performance of the improved mean field model. Achieving all this whilst operating in a very principled manner, using only techniques which can be applied in a fully online and strictly continual setting. Furthermore the more realistic execution time present in the original implementations, makes the additional performance of the original globally induced model highly appealing for practical use, in which one would be regularly training the model on new tasks, and thus couldn't incur the drastic increase in execution time presented by the improved implementations. Unfortunately the globally induced models weren't able to match or exceed the performance of the mean field models using a core set, this is to be expected however as the core set consists of actual data, whereas the global inducing points consist of pseudo data, attempting to imitate actual data. In the process of executing this project I have developed a comprehensive understanding of the theories that lies at the intersection of Bayesian modeling and artificial neural networks, unifying these theories into a single source with a cohesive notation and definitive purpose, something I personally found difficult to find during my research.

5.2 Future Developments

This project has great scope for development, one potential avenue for exploration would be the use of alternative divergence measures. Specifically of interest would be symmetric divergence measures, which uniformly compare the approximate posterior distributions of the current task and previous task. Another extension although complex would be to investigate whether the local reparameterisation trick, could be adapted to the global inducing point inference scheme. A final suggestion would be to implement the model in a deep Gaussian process with a variety of kernels, and evaluate whether any of these posses any specific advantages over Bayesian neural networks for continual learning.

Bibliography

[1] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

[2] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 1992.

[3] Radford M Neal. Bayesian learning via stochastic dynamics. *Advances in neural information processing systems*, 1993.

[4] Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. *Computational learning theory*, 1993.

[5] Sebastian W Ober and Laurence Aitchison. Global inducing point variational posteriors for bayesian neural networks and deep gaussian processes. *arXiv preprint arXiv:2005.08140*, 2020.

[6] Andreas Damianou and Neil D Lawrence. Deep gaussian processes. *Artificial intelligence and statistics*, 2013.

[7] Jose G Moreno-Torres, Troy Raeder, Rocio Alaiz-Rodriguez, Nitesh V Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. *Pattern recognition*, 2012.

[8] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? *Advances in Neural Information Processing Systems*, 1996.

[9] Masa-Aki Sato. Online model selection based on the variational bayes. *Neural computation*, 2001.

[10] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*, 1989.

[11] Eric-Jan Wagenmakers, Michael Lee, Tom Lodewyckx, and Geoffrey J Iverson. Bayesian versus frequentist inference. *Bayesian evaluation of informative hypotheses*, 2008.

[12] David C Knill and Alexandre Pouget. The bayesian brain: the role of uncertainty in neural coding and computation. *Trends in Neurosciences*, 2004.

[13] Bruno A Olshausen. Bayesian probability theory. *The Redwood Center for Theoretical Neuroscience*, 2004.

[14] Daniel Fink. A compendium of conjugate priors. *Citeseer*, 1997.

[15] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 2017.

[16] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 1989.

[17] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

[18] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 1958.

[19] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[20] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. *Institute for Cognitive Science*, 1985.

[21] Ben Krose and Patrick Van Der Smagt. An introduction to neural networks. *Citeseer*, 1993.

[22] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] Ethan Goan and Clinton Fookes. Bayesian neural networks: An introduction and survey. *Case Studies in Applied Bayesian Data Science*, 2020.

[25] Yarin Gal. Uncertainty in deep learning. *PhD thesis, University of Cambridge*, 2016.

[26] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. *International Conference on Machine Learning*, 2015.

[27] Sebastian Farquhar, Lewis Smith, and Yarin Gal. Liberty or depth: Deep bayesian neural nets do not need complex weight posterior approximations. *arXiv preprint arXiv:2002.03704*, 2020.

[28] Marc Mezard, Giorgio Parisi, and Miguel Angel Virasoro. Spin glass theory and beyond: An introduction to the replica method and its applications. *World Scientific*, 1987.

[29] Manfred Opper and Ole Winther. A mean field algorithm for bayes learning in large feed-forward neural networks. *Advances in Neural Information Processing Systems*, 1997.

[30] Richard S Sutton, Steven D Whitehead, et al. Online learning with random representations. *Proceedings of the Tenth International Conference on Machine Learning*, 2014.

[31] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. *International Conference on Machine Learning*, 2017.

[32] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 2017.

[33] Siddharth Swaroop, Cuong V Nguyen, Thang D Bui, and Richard E Turner. Improving and understanding variational continual learning. *arXiv preprint arXiv:1905.02099*, 2019.