School of Informatics

Informatics Project Proposal
Extending the Bayesian Deep Learning Method MultiSWAG

B137355
May 2021

Abstract

The aim of this project is to improve the accuracy and calibration of the Bayesian deep learning method MultiSWAG, which approximates the posterior distribution of the parameters of a deep neural network with a mixture of Gaussians. This will be achieved by replacing the mixture of Gaussians with a mixture of factor analysis models. An online version of factor analysis will be implemented to ensure that the proposed method scales to deep neural networks with millions of parameters. The new approach will be compared to MultiSWAG through a series of experiments involving linear models, Gaussian processes and deep neural networks.

Date: Sunday 9th May, 2021

Tutor: Bob Fisher
Supervisor: Michael Gutmann

1 Motivation

Over the last ten years there has been a phenomenal amount of research in deep learning - the subset of machine learning concerned with multi-layer artificial neural networks [1]. Despite this, the simple stochastic gradient descent (SGD) algorithm and its variants [2] [3] [4] [5], which use only local gradient information and the chain rule from calculus, are still the dominant optimisers used for training neural networks. More recently, it has been argued that the Bayesian deep learning method MultiSWAG leads to better generalisation than SGD [6] .

When training a neural network (NN), SGD finds a single setting of the learnable parameters, which are then used for making predictions. Given that modern NNs can easily have tens of millions of parameters, they are usually underspecified by the available training data. Therefore, many different settings of the parameter vector are able to explain the training data, often equally well. For a NN to be useful, it must also generalise to unseen test data. Unfortunately, training error is rarely a good indication of test error and good regions of the parameter space with respect to the train and tests sets are often shifted [7]. Intuitively, betting everything on a single solution found by SGD would appear a risky strategy.

In contrast, Bayesian methods embrace this uncertainty by marginalising over the parameters [8]. That is, at test time the predictions corresponding to *all* settings of the parameters are used, weighted by how likely they are *a posteriori* given the observed training data. In practical deep learning, this approach is intractable as it involves integrating over a high-dimensional solution space. Therefore, approximate methods are needed. MultiSWAG is one such approximation [6]. This method fits a Gaussian mixture model [9] to multiple instances of the parameter vector that are encountered along the SGD trajectory. At test time, multiple settings of the parameters are sampled from the Gaussian mixture model and the predictions of the resultant NNs are averaged, leading to better generalisation on many benchmark datasets [6].

1.1 Problem Statement

As modern NN architectures grow ever larger, using only a single setting of the parameters found by SGD for inference and disregarding the uncertainty in the predictions becomes less appealing. Bayesian methods, when combined with SGD, can potentially lead to much better generalisation. MultiSWAG is one way of doing this which has shown impressive results [6]. However, given its recency, it is not yet know whether modelling the posterior distribution of the parameters of a NN with a mixture of Gaussians is the best approach. As an alternative, **this project will investigate the use of a mixture of factor analysis [9] models to fit the posterior distribution of the parameters of a NN.**

1.2 Research Hypothesis and Objectives

Let $\boldsymbol{\theta}_t \in \mathbb{R}^d$ be the parameter vector of a NN after epoch t of SGD. For multiple independent runs of SGD, each of T epochs, a MultiSWAG mixture component is constructed by fitting a mean and covariance matrix to $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_T$ (in practice, the epoch count usually begins after some initial *warm-up* period, during which SGD moves toward a local minimum [10]). From [11], the mean vector, $\boldsymbol{\theta}_{\text{SWA}}$, is simply the empirical mean of the samples. That is,

$$\boldsymbol{\theta}_{\text{SWA}} = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{\theta}_t. \tag{1}$$

Given that modern NNs consist of millions of parameters, storing the full $d \times d$ covariance matrix is impractical. Therefore, MultiSWAG settles for estimating a low-rank plus diagonal approximation of the covariance matrix. The low-rank part is equal to

$$\frac{1}{K-1} \sum_{t=T-K+1}^{T} (\boldsymbol{\theta}_t - \overline{\boldsymbol{\theta}}_t)(\boldsymbol{\theta}_t - \overline{\boldsymbol{\theta}}_t)^\intercal, \tag{2}$$

where $\overline{\boldsymbol{\theta}}_t$ is the running average of the parameter vector after the first t epochs of SGD. Note that (2) is analogous to the standard identity for the sample covariance matrix of the observed parameter vectors, except that the sum is over the last K epochs of SGD only and the empirical mean, $\boldsymbol{\theta}_{\text{SWA}}$, which is not available during training, is replaced by $\overline{\boldsymbol{\theta}}_t$. Given that the approximate covariance in (2) involves a sum over K outer products, it necessarily has rank K. Conversely, if a rank K approximation is desired, then only the last K parameter vectors are used in the approximation, irrespective of T. In particular, if $K \ll T$ then the vast majority of $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_T$ will not take part in (2). While this has the benefit of reducing the computational overhead of MultiSWAG, it is a waste of data.

Another method which can be used to estimate a Gaussian distribution with a low-rank plus diagonal covariance matrix is factor analysis (FA) [9]. Using $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ to denote the Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, the form of a FA model fit to $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_T$ is

$$\mathcal{N}\Big(\boldsymbol{\theta}_{\text{SWA}}, \mathbf{F}\mathbf{F}^\intercal + \boldsymbol{\Psi}\Big), \tag{3}$$

where $\mathbf{F}$ is a $d \times K$ matrix and $\boldsymbol{\Psi}$ is a $d \times d$ diagonal matrix. As will be shown in Section 2.2, (3) has the exact same form as the components of the MultiSWAG mixture model. The only difference is how $\mathbf{F}$ and $\boldsymbol{\Psi}$ are estimated. In FA, they are chosen to maximise the log likelihood of the observed parameter vectors. This is commonly done by an expectation-maximisation (EM) or singular value decomposition (SVD) algorithm [9]. Crucially, both these algorithms make use of *all* the data, $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_T$, irrespective of the desired rank K. Unlike the MultiSWAG approach, the rank is de-coupled from the amount of data used for estimation and can be chosen via a more principled approach, such as Bayesian model selection [9].

Given these differences, it is hypothesised that the posterior distribution of the parameters of a NN could be better approximated by a mixture of FA models instead of the MultiSWAG mixture of Gaussians. Furthermore, it is hypothesised that this improved model for the parameter posterior would lead to better generalisation of deep neural networks (DNNs). This project will investigate whether these hypotheses are true by completing the following key objectives: **(O1)** extend the existing MultiSWAG codebase[1] to model the posterior distribution of the parameters of a DNN as a mixture of FA models; **(O2)** investigate how well the FA method approximates the posterior distribution on simple examples involving small NNs for which we are able to compute the true posterior; **(O3)** Reproduce the key experiments from [6] to establish whether the FA method leads to better generalisation of DNNs than MultiSWAG on benchmark datasets.

1.3 Timeliness, Novelty, Significance and Beneficiaries

Recently there has been increased interest in using scalable Bayesian methods for deep learning and MultiSWAG is the result of a number of works in this direction. Indeed, its name is derived from *stochastic weight averaging* (SWA) [7], which simply averages the sampled parameter vectors from the SGD trajectory and uses the mean solution, $\boldsymbol{\theta}_{\text{SWA}}$, when predicting. Extending

[1] https://github.com/izmailovpavel/understandingbdl

this, SWA-Gaussian (SWAG) [11] also estimates a low-rank approximation of the covariance matrix and uses it to model the posterior distribution of the parameters with a Gaussian. Most recently, independent SWAG runs were combined to form the MultiSWAG mixture model [6].

To the best of this author's knowledge, this is the first time that a mixture of FA models will be used to approximate the posterior distribution of the parameters of a NN. This work is significant because, as mentioned in Section 1.2, the MultiSWAG Gaussians have the same form as FA models but do not make use of all the available data when estimating the covariance matrices. Therefore, a better approximation of the posterior distribution can be expected by using a mixture of FA models, which is hypothesised to lead to better generalisation of DNNs.

The proposed method is model-agnostic, meaning that it can be used in conjunction with any NN architecture. Therefore, if successful, the technique is likely to be of interest to many deep learning practitioners, both in academia and industry, working in areas such as computer vision and natural language processing. The extension to the current codebase will be made open source on GitHub, allowing other researchers to experiment with the mixture of FA models posterior distribution. If shown to be superior to MultiSWAG, there is no reason why it could not be made available as a standard training procedure in open source deep learning frameworks such as PyTorch [12]. Indeed, SWA has recently been added to the standard PyTorch library[2].

1.4 Feasibility

The code for running the MultiSWAG experiments from [6] is freely available on GitHub. Most of the implementation work will involve extending this codebase to include the mixture of FA models posterior distribution for the parameters of a DNN. Given that modern DNNs can easily contain tens of millions of parameters, it is unlikely that the classic EM and SVD batch FA algorithms will scale to such data. Therefore, an incremental, online FA algorithm will be implemented. This approach is outlined in Section 3.1.

A large proportion of the project time will be allocated to running experiments and analysing the results. It may not be possible to reproduce all the experiments from [6], but running the smaller experiments, such as those involving the MNIST dataset [13] and the seven-layer LeNet-5 model [14], should be possible. This would meet the minimum completion criteria for the project. If time allows and assuming that access to a GPU cluster is provided, some of the experiments involving the larger CIFAR-10 and CIFAR-100 datasets [15] and the ResNet models [16] will also be executed.

2 Background and Related Work

2.1 Stochastic Gradient Descent

Let $f_{\boldsymbol{\theta}} : X \to Y$ be a NN parameterised by $\boldsymbol{\theta} \in \mathbb{R}^d$ and $\{(\mathbf{x}_n, y_n)\}_{n=1}^N \subset X \times Y$ a set of training examples. Starting from a random initialisation, $\boldsymbol{\theta}_0$, vanilla SGD iteratively updates the parameter vector in the direction of the negative gradient of the training loss. Formally, the update rule is

$$\boldsymbol{\theta}_{m+1} = \boldsymbol{\theta}_m - \alpha_m \nabla_{\boldsymbol{\theta}_m} \left(\frac{1}{|\mathcal{B}|} \sum_{(\mathbf{x}_n, y_n) \in \mathcal{B}} \mathcal{L}_D(y_n, f_{\boldsymbol{\theta}_m}(\mathbf{x}_n)) + \lambda \mathcal{L}_R(\boldsymbol{\theta}_m) \right), \tag{4}$$

[2] https://pytorch.org/blog/pytorch-1.6-now-includes-stochastic-weight-averaging/

where $\mathcal{B}$ is a mini-batch of training examples and $\alpha_m > 0$ is the learning rate at update m. The term inside the large brackets is a combination of the data loss $\mathcal{L}_D$ and the regularisation loss $\mathcal{L}_R$, where the constant λ controls the trade-off between goodness of fit and model complexity. For classification problems, $\mathcal{L}_D$ is typically the cross-entropy loss and $\mathcal{L}_R$ is the Euclidean norm of $\boldsymbol{\theta}_m$. The operation $\nabla_{\boldsymbol{\theta}_m}$ indicates taking the partial derivative of the loss with respect to each element of $\boldsymbol{\theta}_m$. If the learning rate is scheduled correctly then SGD is guaranteed to converge to a local minimum [2].

2.2 From SWA to MultiSWAG

Due to the difficulty of setting the learning rate, in practice SGD does not always converge to a local minimum. Depending on the SGD trajectory, it may be beneficial to average some of the iterates $\{\boldsymbol{\theta}_m\}_{m=0}^{M}$, where M is the total number of times that the update in (4) is applied. If, after some number of updates $k < M$, the process begins to oscillate around a local minimum $\boldsymbol{\theta}^*$, then taking the average of the iterates $\{\boldsymbol{\theta}_m\}_{m=k}^{M}$ is likely to provide a better estimate of $\boldsymbol{\theta}^*$ than the final iterate $\boldsymbol{\theta}_M$. This idea was proposed in some early work in stochastic optimisation [17]. It is the motivation behind the SWA solution, $\boldsymbol{\theta}_{\text{SWA}}$, from [7], which was defined in (1).

SWA was later extended to also approximate the covariance matrix of the SGD iterates [11]. Together, the mean vector $\boldsymbol{\theta}_{\text{SWA}}$ and the approximate covariance matrix are used to define a Gaussian posterior distribution over the parameters of a NN. At test time, multiple settings of the parameter vector are sampled from the Gaussian posterior distribution and the predictions of the resultant NNs are averaged. This method, called SWA-Gaussian (SWAG), can be seen to approximate the true Bayesian model average, $p(y|\mathbf{x}, \mathcal{D}) = \int_{\boldsymbol{\theta}} p(y|\mathbf{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D})$, where all settings of $\boldsymbol{\theta}$ are used, weighted by their posterior probabilities in light of the observed data $\mathcal{D}$.

SWAG estimates a low-rank plus diagonal covariance matrix of the SGD iterates. The low-rank part is given in (2). The diagonal part is $\boldsymbol{\Sigma}_{\text{diag}} = \text{diag}\big(\overline{\boldsymbol{\theta}^2} - \boldsymbol{\theta}_{\text{SWA}}^2\big)$, where $\overline{\boldsymbol{\theta}^2}$ is the empirical mean of the squared parameter vectors collected at the end of each epoch of SGD. That is, $\overline{\boldsymbol{\theta}^2} = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{\theta}_t^2$, where the square is applied element-wise. The approximation in (2) can be written in matrix form as $\frac{1}{K-1}\hat{\mathbf{D}}\hat{\mathbf{D}}^\intercal$, where $\hat{\mathbf{D}}$ is formed by concatenating the vectors $(\boldsymbol{\theta}_t - \overline{\boldsymbol{\theta}}_t)$ corresponding to the last K epochs of SGD. It follows that each component of the MultiSWAG mixture model is a Gaussian distribution of the form

$$\mathcal{N}\Big(\boldsymbol{\theta}_{\text{SWA}}, \frac{1}{2(K-1)}\hat{\mathbf{D}}\hat{\mathbf{D}}^\intercal + \frac{1}{2}\boldsymbol{\Sigma}_{\text{diag}}\Big). \tag{5}$$

By absorbing the constants into the matrices in (5), we can rewrite the Gaussian distribution in the exact same form as the FA model in (3), with $\mathbf{F} = \frac{1}{\sqrt{2(K-1)}}\hat{\mathbf{D}}$ and $\boldsymbol{\Psi} = \frac{1}{2}\boldsymbol{\Sigma}_{\text{diag}}$.

The samples from a single SWAG model are all centred on the same local minimum. Given the highly non-convex nature of DNN loss surfaces [18], there are often many local minima in different regions of the parameter space which give rise to equally good solutions. It is argued that these are exactly the circumstances in which an approximate Bayesian model average can have the biggest impact on accuracy and calibration [8], and more diversity can be achieved by combining the samples from multiple independently trained SWAG models [6]. Starting from a different random initialisation, each independent run ends up in a different *basin of attraction* of the loss surface. At test time, the parameter vectors which contribute to the Bayesian model average are sampled from each SWAG model with equal probability. Thus, the final posterior distribution over the parameters is a mixture of Gaussians. This is the MultiSWAG method which this project will extend by replacing the mixture of Gaussians by a mixture of FA models.

2.3 Factor Analysis

A FA model for the parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ of a NN is of the form $p(\boldsymbol{\theta}) = \mathcal{N}\big(\mathbf{c}, \mathbf{F}\mathbf{F}^\intercal + \boldsymbol{\Psi}\big)$, where $\mathbf{c} \in \mathbb{R}^d$, $\mathbf{F} \in \mathbb{R}^{d \times K}$ and $\boldsymbol{\Psi}$ is a $d \times d$ diagonal matrix [9]. The value of $\mathbf{c}$ which maximises the likelihood of the observed data is the empirical mean of the observations [9], which in this case is $\boldsymbol{\theta}_{\text{SWA}}$. By substituting $\mathbf{c} = \boldsymbol{\theta}_{\text{SWA}}$ into $p(\boldsymbol{\theta})$, the distribution in (3) is obtained. Having set the mean, an EM or SVD algorithm can in theory find the maximum likelihood estimates of $\mathbf{F}$ and $\boldsymbol{\Psi}$ [9]. However, both methods require storing all the observations in memory, making them impractical for fittings the parameters of large DNNs.

3 Programme and Methodology

3.1 Online Factor Analysis

An alternative to batch FA algorithms is to use gradient methods to maximise the log likelihood, $L(\mathbf{F}, \boldsymbol{\Psi}) = \frac{1}{T}\sum_{t=1}^{T} \log p(\boldsymbol{\theta}_t|\mathbf{F}, \boldsymbol{\Psi})$. By adapting a general result for latent variable models from [9] to FA, the partial derivatives of $\log p(\boldsymbol{\theta}|\mathbf{F}, \boldsymbol{\Psi})$ with respect to $\mathbf{F}$ and $\boldsymbol{\Psi}$ can be written as

$$\nabla_{\mathbf{F},\boldsymbol{\Psi}} \log p(\boldsymbol{\theta}|\mathbf{F}, \boldsymbol{\Psi}) = \int_{\mathbf{h}} p(\mathbf{h}|\boldsymbol{\theta}, \mathbf{F}, \boldsymbol{\Psi}) \nabla_{\mathbf{F},\boldsymbol{\Psi}} \log p(\boldsymbol{\theta}, \mathbf{h}|\mathbf{F}, \boldsymbol{\Psi}), \tag{6}$$

where $\mathbf{h}$ represents the latent variables. Since $\mathbf{h}$ is independent from $\mathbf{F}$ and $\boldsymbol{\Psi}$ in the FA model,

$$\nabla_{\mathbf{F},\boldsymbol{\Psi}} \log p(\boldsymbol{\theta}, \mathbf{h}|\mathbf{F}, \boldsymbol{\Psi}) = \nabla_{\mathbf{F},\boldsymbol{\Psi}} \log \Big(p(\boldsymbol{\theta}|\mathbf{h}, \mathbf{F}, \boldsymbol{\Psi}) p(\mathbf{h}) \Big) = \nabla_{\mathbf{F},\boldsymbol{\Psi}} \log p(\boldsymbol{\theta}|\mathbf{h}, \mathbf{F}, \boldsymbol{\Psi}). \tag{7}$$

It follows that (6) is just the expected value of $\nabla_{\mathbf{F},\boldsymbol{\Psi}} \log p(\boldsymbol{\theta}|\mathbf{h}, \mathbf{F}, \boldsymbol{\Psi})$ over the distribution $p(\mathbf{h}|\boldsymbol{\theta}, \mathbf{F}, \boldsymbol{\Psi})$. It turns out that this expectation can be computed in closed-form, since both $p(\boldsymbol{\theta}|\mathbf{h}, \mathbf{F}, \boldsymbol{\Psi})$ and $p(\mathbf{h}|\boldsymbol{\theta}, \mathbf{F}, \boldsymbol{\Psi})$ are tractable Gaussian distributions whose mean and covariance matrices are given in [9].

This interpretation gives rise to an online stochastic gradient algorithm for optimising $\mathbf{F}$ and $\boldsymbol{\Psi}$. Given a single observation, $\boldsymbol{\theta}_t$, the partial derivatives $\nabla_{\mathbf{F},\boldsymbol{\Psi}} \log p(\boldsymbol{\theta}_t|\mathbf{F}, \boldsymbol{\Psi})$ can be computed in closed-form. By definition of the log likelihood, $L(\mathbf{F}, \boldsymbol{\Psi})$, the expectation of these sample derivatives is proportional to $\nabla_{\mathbf{F},\boldsymbol{\Psi}} L(\mathbf{F}, \boldsymbol{\Psi})$. This is exactly what is required for a stochastic gradient algorithm. Hence, the sample derivatives can be used with SGD or any of its variants to update $\mathbf{F}$ and $\boldsymbol{\Psi}$. It is worth noting that the calculation of these derivatives uses $\boldsymbol{\theta}_{\text{SWA}}$, which is not known during training. However, it can be replaced by the running average $\overline{\boldsymbol{\theta}}_t$, as done in the SWAG covariance approximation in (2). One limitation of this approach is that many sample derivatives may be required in order to converge to the optimal $\mathbf{F}$ and $\boldsymbol{\Psi}$. However, a sufficiently large number of samples can be obtained by using the NN parameter vectors observed after each mini-batch update instead of waiting until the end of each epoch, at the expense of extra computation.

To the best of this author's knowledge, this online gradient algorithm is a novel approach to fitting FA models to high-dimensional data. To match the existing MultiSWAG codebase, the algorithm will be implemented in PyTorch [12]. The code will be thoroughly tested by comparing it to the scikit-learn [19] implementation of FA, which will act as a *test oracle*. In addition, an online version of the EM algorithm from [9] will also be implemented and compared to the gradient-based approach. This will involve modifying the M-step to update $\mathbf{F}$ and $\boldsymbol{\Psi}$ incrementally as each $\boldsymbol{\theta}_t$ is sampled, in contrast to the batch algorithm which uses all samples, $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_T$, to construct new estimates of $\mathbf{F}$ and $\boldsymbol{\Psi}$ on each iteration of EM. If at least one of these online FA algorithms is implemented correctly, objective **(O1)** will be satisfied.

3.2 Experiments

Linear Models. The online FA algorithms will be used to approximate the posterior distribution of the parameters of linear regression models [20]. Given the linear regression Gaussian likelihood function and a Gaussian prior over the model parameters $\boldsymbol{\theta}$, it turns out that the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ is also Gaussian, with mean and covariance which can be computed in closed-form using the observed data $\mathcal{D}$ [20]. The similarity between these closed-form solutions and the parameters of the Gaussian posteriors estimated by both SWAG and FA will be computed. This will complete objective **(O2)** for the case of linear NNs. In addition, these experiments will provide evidence of which online FA algorithm works better, the gradient-based approach or online EM. Whichever method comes out on top will be used in the remaining experiments.

Gaussian Processes. For a class of flexible non-parametric models called Gaussian processes (GPs) [21], it turns out that the predictive posterior $p(y|\mathbf{x}, \mathcal{D})$ is a Gaussian distribution whose mean and covariance matrix can be computed in closed-form. Moreover, under certain conditions a GP is equivalent to a NN with a single fully-connected hidden layer of infinite width [22]. These properties make for an interesting comparison. Given data $\mathcal{D}$ generated by a GP, $p(y|\mathbf{x}, \mathcal{D})$ can be approximated by applying both SWAG and the proposed FA method to a NN with a single, very wide, fully-connected hidden layer. Since in this case the ground truth $p(y|\mathbf{x}, \mathcal{D})$ is known, the quality of the approximations can be evaluated. This work will act as an intermediate step between the experiments involving linear models and DNNs.

Deep Neural Networks. Both the posterior distribution of the parameters of a DNN and its posterior predictive distribution are intractable [23]. However, objective **(O3)** can still be completed by reproducing a subset of the experiments from [6], to assess both the accuracy and calibration of the proposed FA method on test data in comparison to MultiSWAG. Calibration measures how well the accuracy and confidence of a model's predictions are aligned, which is important for uncertainty estimates of probabilistic classifiers [11].

3.3 Risk Assessment

The main project risk is that the online FA algorithms do not work as expected. Even if this is the case, the project's minimum completion criteria will still be achievable. The key experiments outlined in Section 3.2 could be executed using the scikit-learn implementation of FA instead of the online version. This would just limit the size of the NNs involved in the experiments. The experiments involving linear models will require no more than a standard laptop or desktop computer, while experiments involving GPs and NNs with a few hidden layers will require access to a GPU. Given that this project will be completed part-time over a 12-month period, it is not anticipated that this will be a problem. However, if GPU access is limited, then many comparisons can be made with SWAG instead of the much more costly MultiSWAG mixture model. This would not detract too much from the results since the focus of the project is on the different approaches to estimating the covariance matrix in (3). As the project will only use synthetic or publicly available datasets, there are no risks associated with data acquisition.

3.4 Ethics

There are no ethical issues associated with this project. Only synthetic and publicly available benchmark datasets will be used, as described in Section 4. Moreover, the proposed method simply builds on recent research in Bayesian deep learning and at most is expected to provide a small improvement compared to algorithms which already exists and are publicly available.

4 Evaluation

Online Factor Analysis. Synthetic data will be used to verify the correctness and robustness of the proposed online FA algorithms. First, data will be generated using actual FA models for various settings of the parameters $\mathbf{c}, \mathbf{F}$ and $\mathbf{\Psi}$. It will then be verified that in each case the online algorithms are able to recover the true covariance matrix from the observed data. Euclidean distance will be used as a measure of how dissimilar the online estimates of the covariance are from both the true value and the value returned by the scikit-learn batch FA implementation.

Linear Models. Experiments involving the posterior distribution of the parameters of linear models will use the same regression datasets as [11] - the Boston Housing, Concrete Compression Strength, Energy Efficiency, Naval Propulsion, Yacht Hydrodynamics and Combined Cycle Power Plant datasets from the UCI Machine Learning Repository [24]. For each dataset, the mean and covariance matrix of the posterior distribution of the linear regression parameters will be computed in closed-form, by each online FA algorithm and by SWAG. Euclidean distance will be used as a measure of dissimilarity between the corresponding estimates.

Gaussian Processes. Synthetic datasets of varying dimensionality will be generated by GPs with squared exponential covariance functions. In each case, the mean and covariance of the Gaussian predictive posterior $p(y|\mathbf{x}, \mathcal{D})$ will be estimated by both SWAG and the proposed FA method for different test points $(\mathbf{x}, y)$. The estimates will be compared to the ground truth mean and covariance via Euclidean distance. Visual comparisons of the ground truth and estimated predictive posteriors will also be generated by fitting the models to 1 and 2-dimensional data.

Deep Neural Networks. For DNN experiments, MNIST [13], CIFAR-10 and CIFAR-100 [15] datasets will be used. Starting with LeNet-5 [14], as many experiments as possible from [6] will be executed using the mixture of FA models instead of the MultiSWAG mixture of Gaussians. The exact number of experiments will depend on the efficiency of the online FA algorithm. However, it is hoped that running at least some of the experiments involving the 18 and 20-layer ResNet models [16] will be possible. The aim is to verify whether the proposed method results in better accuracy and calibration than MultiSWAG. Calibration will be measured by the negative log-likelihood metric and by plotting *reliability diagrams*, as in [11].

5 Expected Outcomes

It is expected that an online FA algorithm which scales to large datasets will be implemented. While there exist online algorithms for learning latent variable models [23] [25], they are not specific to FA. Thus, the proposed solution may be of interest to researchers in its own right.

It is hoped that the proposed method will result in better generalisation of NNs than MultiSWAG. However, given the difficulty of tuning these models, it may be difficult to beat the results reported in [6]. Even so, there is a significant advantage of using the FA method, as pointed out in Section 1.2. That is, unlike MultiSWAG, the rank of the approximate covariance matrices estimated by online FA is not tied to the amount of data used for estimation. This opens up the possibility of choosing the rank in a more principled way, such as via Bayesian model selection. As long as online FA can be implemented efficiently, it would appear a fundamentally better way of modelling the posterior distribution of the parameters of a NN.

6 Research Plan, Milestones and Deliverables

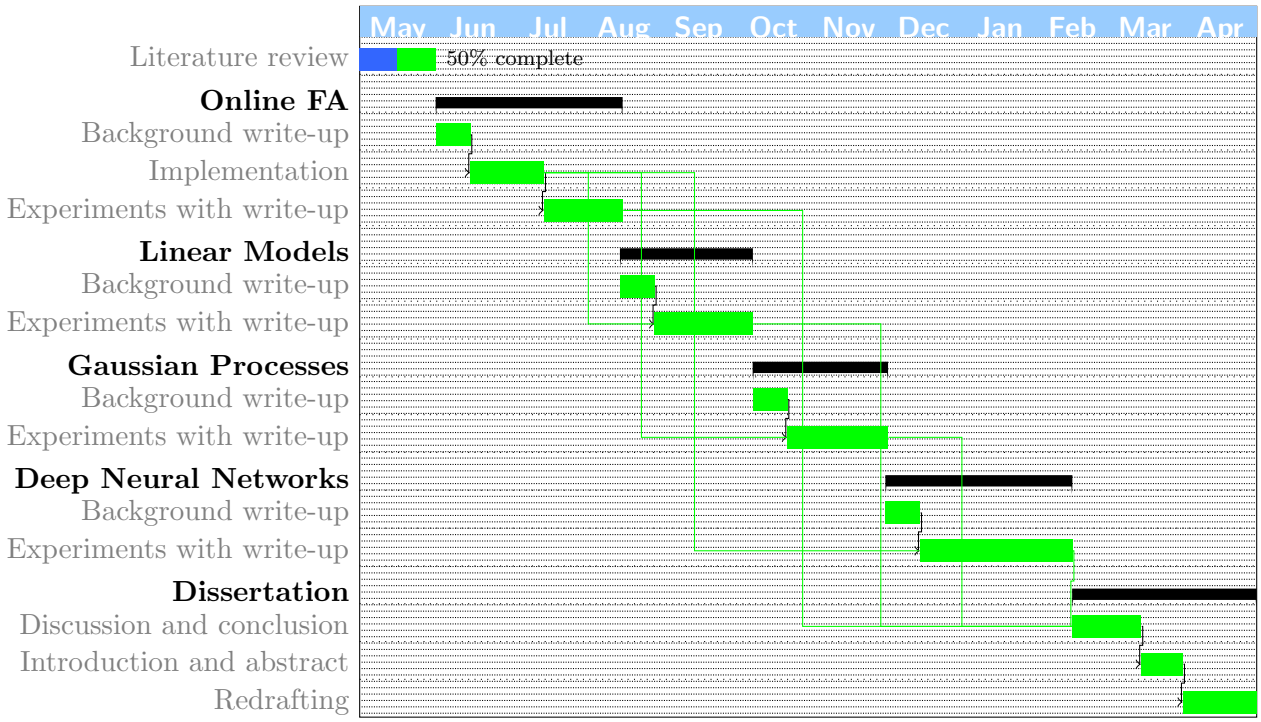

Figure 1: Gantt Chart of the activities defined for this project.

Milestone	Month	Description
M_1	July	Implementation of online FA algorithms
M_2	August	Experiments with online FA complete
M_3	October	Experiments with linear models complete
M_4	November	Experiments with Gaussian processes complete
M_5	February	Experiments with deep neural networks complete
M_6	April	Submission of dissertation

Table 1: Milestones defined in this project.

Deliverable	Month	Description
D_1	July	Working code and unit tests for online FA algorithms
D_2	August	Experimental results for online FA
D_3	October	Experimental results for linear models
D_4	November	Experimental results for Gaussian processes
D_5	February	Experimental results for deep neural networks
D_6	April	Complete dissertation

Table 2: List of deliverables defined in this project.

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[2] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[3] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[4] Matthew D Zeiler. ADADELTA: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

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

[6] Andrew G Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *arXiv preprint arXiv:2002.08791*, 2020.

[7] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew G Wilson. Averaging weights leads to wider optima and better generalization. *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence*, 2:876–885, 2018.

[8] Andrew G Wilson. The case for Bayesian deep learning. *arXiv preprint arXiv:2001.10995*, 2020.

[9] David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[10] Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate Bayesian inference. *Journal of Machine Learning Research*, 18:4873–4907, 2017.

[11] Wesley Maddox, Timur Garipov, Pavel Izmailov, Dmitry Vetrov, and Andrew G Wilson. A simple baseline for bayesian uncertainty in deep learning. *Proceedings of Advances in Neural Information Processing Systems*, 32, 2019.

[12] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *Proceedings of Advances in Neural Information Processing Systems*, 32:8024–8035, 2019.

[13] Li Deng. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29:141–142, 2012.

[14] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEE*, 86:2278–2324, 1998.

[15] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 2012.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[17] James C Spall. *Introduction to Stochastic Search and Optimization*. Wiley, 2003.

[18] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Proceedings of Advances in Neural Information Processing Systems*, 31, 2018.

[19] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[20] Kevin P Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[21] Carl E Rasmussen and Christopher K I Williams. *Gaussian processes for machine learning*. MIT Press, 2006.

[22] Christopher K I Williams. Computing with infinite networks. *Proceedings of the 9th International Conference on Neural Information Processing Systems*, pages 295–301, 1996.

[23] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[24] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[25] Olivier Cappé and Eric Moulines. Online EM algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71:593–613, 2017.