Bayesian Deep Learning for Learned Data Augmentation

Author:
Georgios K. Zefkilis

Supervisors:
Søren Hauberg
Pola Elisabeth Schwöbel

A thesis submitted in fulfillment
of the requirements for the degree of
Masters of Science in
Human Centered Artificial Intelligence

Technical University of Denmark
Copenhagen, Denmark

July 2022

Bayesian Deep Learning for Learned Data Augmentation

Master Thesis
July, 2022

By
Georgios Konstantinos Zefkilis

Published by: DTU, Department of Computer Science and Applied Mathematics, Richard Petersens Plads, Building 321, 2800 Kgs. Lyngby Denmark
www.compute.dtu.dk

Approval

This thesis has been prepared over six months for the Department of Applied Mathematics and Computer Science, at the Technical University of Denmark, DTU, in partial fulfilment for the degree Master of Science in Human Centered Artificial Intelligence, MSc.

Georgios Konstantinos Zefkilis - s151074

.....

Signature

.....

Date

Abstract

Advanced deep network architectures, robust computation, and access to big data are few of the factors that have contributed to the rapid progress of deep neural networks compare to conventional machine learning methods. However, the availability of large annotated data is scarce in specific domains. As a result the training of such models becomes insufficient.

The most popular technique to increase the size of a dataset is data augmentation. Such a process has been proven to be cumbersome and time consuming. Other methods have been developed where utilize the capabilites of generative adversarial networks (GANs) and reinforcement learning to automatically generate new samples that resemble the training data. These methods yet promising are unstable and difficult to train.

One can perform data augmentation by finding the right transformations of the input images. The development of spatial transformer network (STN) which estimates image transformations has shown remarkable results. However, STNs are also difficult to train and can be sensitive to incorrect transformations. In the study we apply the Bayesian framework using Laplace approximation to turn the STN into a probabilistic model to perform learned data augmentation while overcoming its aforementioned limitations.

The Laplace approximation method provides the flexibility to apply Bayesian inference solely on a small part of the model which reduces the computational cost significantly. The use of marginal likelihood allows for automatic parameter tuning neglecting any manual adjustments. This is a major advantage compared to other Bayesian approximation methods.

The probabilistic manner of STN gives us the capability to estimate numerous transformations rather than a deterministic one. Thus, we can evaluate every image in multiple postures, which will enhance the robustness of the training. The major benefit though is that those transformations behave as learned data augmentation scheme that improves the predictive performance of the model.

We demonstrate through a series of experiments on conventional and challenging image datasets that the probabilistic STN using Laplace approximation leads indeed to improved classification performance and better model calibration compared to the deterministic one.

Acknowledgements

Georgios Konstantinos Zefkilis, MSc Human Centered Artificial Intelligence, DTU

First and foremost I would like to thank my supervisor Pola E. Schwöbel Phd student at DTU for her guidance, support and valuable feedback she provided throughout the last six months of the study. I would also like to thank my co-supervisor Søren Hauberg, professor at DTU, for his valuable inputs and useful discussions.

Contents

1 Introduction

Deep Learning models have made significant progress in discriminative tasks. This has been enforced by the improvement of deep network architectures, robust computation, and access to massive data [1]. However, the availability of large annotated data is scarce in specific domains. For example, in the medical domain proprietary and privacy reasons limits data access and the publicly available datasets often lack size and expert annotations, rendering them insufficient for training deep neural networks [2]. One way to overcome the limited number of data available is to perform data augmentation. Data augmentation is a group of methods for creating new data points from existing data in order to artificially increase the amount of data. This includes making minor adjustments to data [3]. For image data this entails transformations such as rotation, scaling and cropping among others. The underlying assumption of data augmentation is that the target prediction of observation is invariant to small transformations of the input [4]. Generate a suitable training dataset through data augmentation is a cumbersome process that requires a significant amount of time and hand tuning.

Other than the traditional data augmentation one can use deep learning models to generate new data points. For example generative adversarial networks (GANs) [5] can learn patterns from input datasets and automatically create new examples which resemble the training data [3]. In recent studies implementation of GANs for data augmentation have shown promising results to improve network performance when data collection is prohibitively expensive [6],[7],[8]. Creating synthetic data though comes with limitations such as lack of sufficient intrinsic metrics for evaluating the quality of the generated samples. Additionally, the training of GANs is usually unstable and requires a lot of computational resources.

Other studies have used reinforcement learning to find the best policies that will perform affine transformation on images and yield the highest validation accuracy on a target dataset [9]. Such a process is however, computationally expensive and difficult to train. In their study Cubuk et al. [10] have managed to overcome those constraints to some extend yet there are open questions regarding the robustness of their solution.

Instead of generating synthetic data or trying to find the right policies one can perform data augmentation by finding the right transformations of the input images. Spatial Transformer Networks (STN) [11] apply a spatial transformation to the input image data as part of an end-to-end trained network. STN estimates the transformation parameters T_θ (see example in fig 1.1) from each input separately through a neural network.

In practice to generate such transformations is a difficult task since the number of transformations can be ambiguous (eg. there are more than one correct transformations). This is a fact that classic STN ignores thus becomes sensitive to small mis-predictions of transformations. To overcome that issue and be able to use STN

for data augmentation Schwöbel et al. [4] apply a probabilistic approach on a Spatial Transformer Network (STN) using the Bayesian framework and more specifically amortised variational inference.

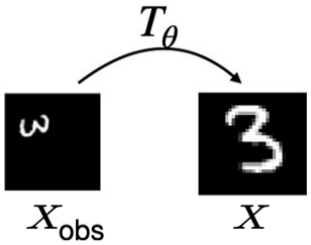


Figure 1.1: A 'true', latent image can be recovered from a natural image by applying the right transformation $X = T_\theta(X_{\text{obs}})$.

The probabilistic version of the STN learns a *probability distribution* over the transformations which allows sampling of transformations (eg. $\theta \sim p(\theta|X_{\text{obs}})$ for the example in figure 1.1). Then use those samples to create as many images as necessary to enrich the training dataset. Hence the above framework presents a clean, probabilistic way to think about data augmentation.

The probabilistic-STN has shown promising results on improving a classifier's accuracy as well as in model's robustness by using the sample transformations for continuous training.

In the present study we will implement the same idea of the probabilistic STN for learned data augmentation. We will make use of Bayesian neural networks since they provide significant advantages compare to deterministic neural networks. Though we will use a simpler method for inference namely Laplace approximation. Laplace first applied in the context of neural networks by MacKay [12], however due to prohibitive computations regarding the Hessian matrix this method hasn't been applied as much as other Bayesian ones. Recent advances in approximation of the Hessian and development of software libraries constitute Laplace a simple, cost-efficient and competitive approximation method for inference in Bayesian deep learning.

Overall we aim to prove that the probabilistic STN using Laplace approximation over a small subset of weights can firstly learn how to augment data and secondly to use the augmented images from sampling to enrich our datasets and improve the performance of our model.

2 Background

This section covers the necessary theory that the reader needs to understand the concepts, ideas, methods and network architectures used to carry out the experiments under study.

2.1 Spatial Transformer Network

Convolutional neural networks (CNN) consitute a powerful class of models which have been used for solving various tasks from classification and localisation to semantic segmentation and action recognition [13],[14],[15] [16]. A desirable property of a system when working with image reasoning is to be robust to input variations. Meaning that the system should be able to disentagle object pose and part deformation from texture and shape [11]. Introduction of local max pooling layers endow the system with a certain degree of spatial invariance to the position of features.

Remark 2.1.1: Pooling operator

The Pooling operator can be considered as a downsampling mechanism. Due to pooling operator the feature map's spatial size is progressively reduced along the depth dimension, resulting in a reduction in the number of parameters and computational cost [17].

However, pooling mechanisms come with shortcomings that makes them undesireable operators. Pooling layers discard significant amount of exact positional information. Additionally pooling mechanisms are local and predefined. With a small receptive field, the spatial invariance introduced by the pooling operator can be effective only towards the deeper layers of a CNN. Thus, the intermediate feature maps remain variant to input transformations [11], [17]. If we arbitrarily increase the receptive field, the feature map will be aggressively downsampled. An action which should be avoidable. Due to the aforementioned limitations convolutional neural networks are not invariant to the input data in a computationally and parameter manner [11].

To address the issues mentioned above Jaderberg et al. [11] introduced the spatial transformer network (STN) that provides convolutional neural networks with explicit spatial transformation capabilities.

The overall goal of the Spatial Transformer Network is to learn to perform spatial transformations on the input image to enlarge the geometric invariance of the model. For example, it can crop a region of interest, scale and correct the orientation of an image [18].

The main properties of an STN are:

- Modular: STNs can be added anywhere into existing architectures with relatively small adjustments [11], [17].

- Differentiable: STNs can be trained with back-propagation allowing for end-to-end training of the models they are inserted in [11], [17].
- Dynamic: STNs perform a spatial transformation on a feature map for each input (Contrary to the pooling layer that behaves identically across all inputs) [11], [17].

Figure 2.1 shows the architecture of spatial transformer module.

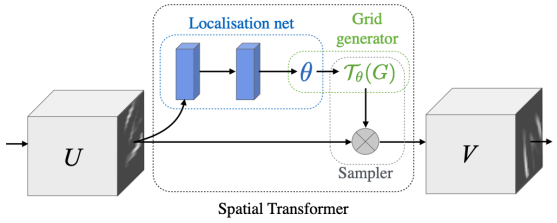


Figure 2.1: Architecture of a Spatial Transformer Module [11]

The components of the spatial transformer module are the localisation network, the grid generator and the sampler.

Localization Network:

The task of the localisation network is to identify the parameters θ of the inverse transformation $T_\theta(G)$ that will translate the input feature map into a canonical pose, making recognition in subsequent layers easier. The localisation network should have a final regression layer to generate the transformation parameters θ, regardless of the network's form (it might be a fully connected network or a convolutional network) [11],[19].

Parameterised Sampling Grid:

The grid generator iterates over the regular grid (On the left of figure 2.2) of the output/target image and uses the inverse transformation $T_\theta(G)$ to calculate the corresponding sample positions in the input/source image (Right figure in 2.2) [11].

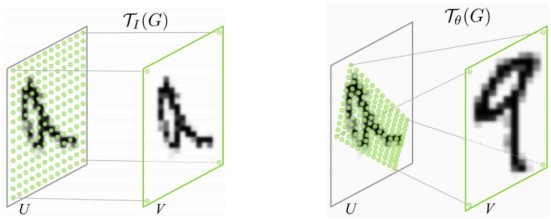


Figure 2.2: Examples of applying the parameterized sampling grid [11]

Sampler

At last the sampler iterates over the entries of the sampling grid and extracts the corresponding pixel values from the input map using bilinear interpolation [11].

The implementation of STNs has shown remarkable results. However, STN can be very sensitive. For example if the STN module gives a wrong transformation eg.

zoom in the wrong part this will affect the downstream task eg. classification. In the present study we use STN as the baseline in all experiments and we turn it into a probabilistic model by applying Bayesian inference to compare the results.

2.2 Affine Transformations

In present study we will make use of affine transformations for data augmentation. Affine transformation is a simple family of transformations that can be parametrized in a differentiable manner [4]. This is a very important requirement which allows to learn the transformation by implementing the spatial transformer network.

Affine transformation is a linear mapping technique that changes the object's orientation, size, or location while preserving lines or distance ratios from one affine space to another [20].

The affine transformation technique is typically used to correct for geometric distortions or deformations that occur with non-ideal camera angles [21]. Those methods often used in Machine Learning and Deep Learning for Image Processing and Image Augmentation.

Figure 2.3 illustrates the different affine transformations: translation, scale, shear, and rotation.

Affine Transform	Example	Transformation Matrix	
Translation		$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	t_x specifies the displacement along the x axis t_y specifies the displacement along the y axis.
Scale		$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	s_x specifies the scale factor along the x axis s_y specifies the scale factor along the y axis.
Shear		$\begin{bmatrix} 1 & sh_y & 0 \\ sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	sh_x specifies the shear factor along the x axis sh_y specifies the shear factor along the y axis.
Rotation		$\begin{bmatrix} \cos(q) & \sin(q) & 0 \\ -\sin(q) & \cos(q) & 0 \\ 0 & 0 & 1 \end{bmatrix}$	q specifies the angle of rotation.

Figure 2.3: Examples of affine transformations [21]

To apply the affine matrix to an image, the image is represented as a matrix whose entries reflects the pixel intensity at that location. The location of pixels corresponds to a vector and indicate the coordinates of the pixels in the image, $[x, y]$, where x

and y are the row and column of a pixel in the image matrix. Following that, the coordinates can be multiplied by an affine-transformation matrix to determine the point where the pixel value will be replicated in the output image [22].

However, translation is not a linear transformation. To apply this framework to translation, a third dimension is added to be able to perform transformations using matrices [20]. The third dimension is usually set to 1 (other no-zero constants can be used), so that the new coordinate is [x, y, 1]. We can then represent rotation, scaling, and translation on 2-dimensional space as multiplication by a 3-by-3 matrix (so the third dimension, which is the constant 1, allows translation) [22],[23].

Due to matrix multiplication is an associative process, it is possible to combine a series of transformations into a single transformation by multiplying the matrices of each transformation individually in the order in which the transformations are performed [22],[23].

In the present study, similarly to Schwöbel et al. [4], we will make use of a subset of affine transformations containing rotation, isotropic scaling and translation in x and y. Therefore in 2-dimensional space (including the third dimension with the homogeneous coordinates) we learn parameters $\theta = (\phi, z, t_{\mathrm{x}}, t_{\mathrm{y}})$ that parameterise the affine matrix A_θ.

$$A_\theta = \begin{bmatrix} z \cdot \cos\phi & -z \cdot \sin\phi & t_x \\ z \cdot \sin\phi & z \cdot \cos\phi & t_y \\ 0 & 0 & 1 \end{bmatrix} \in R^{3\times 3}, z > 0$$

2.3 Bayesian Machine Learning

The conventional machine learning approach treats the model's weight w as deterministic variables while, the Bayesian framework as random. Bayesian approach begins by assigning a prior distribution over the weight values which normally defined as follows:

$$w \sim p(w) = \mathcal{N}(w \mid m_0, \Sigma_0) \tag{2.1}$$

The prior distribution is updated when provided new data. Assuming we have a dataset $\mathcal{D} = \{y_i, x_i\}_{i=1}^N$, then the posterior is calculated by the Bayes theorem:

$$p(w \mid x, y) = \frac{p(y \mid x, w)p(w)}{p(y \mid x)} \tag{2.2}$$

Remark 2.3.1: Bayes theorem

- Bayes theorem can help us to convert prior probability into a posterior probability by incorporating the evidence provided by the observed data [24].
- Posterior probability allows us to evaluate the uncertainty of parameters after we have observed the data [24].
- Priors are useful can encode domain knowledge, prevent overfitting.

The likelihood term $p(y \mid x, w)$ is estimated on the training data which provide the values of the weights w. This can be done for example by using a softmax classifier (in classification) or heteroscedastic regression (in regression) [25]. The denominator in Bayes theorem which is also called the marginal probability is calculated:

$$p(y \mid x) = \int p(y \mid x, w)p(w)\mathrm{d}w \tag{2.3}$$

The marginal probability works as normalisation factor and ensures that the posterior probability will always be between [0,1].

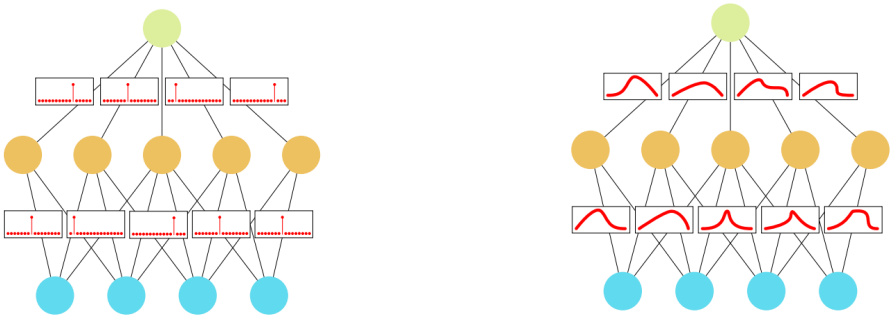
In most of the cases (unless working with Bayesian linear regression) the integral in the Equation 2.3 is intractable. As a result to calculate the posterior $p(w \mid x, y)$ in the equation 2.2 is also intractable. Consequently, the predictive distribution for a test input x^*:

$$p\left(y^* \mid x^*, x, y\right) = \int p\left(y^* \mid x^*, w\right) p(w \mid x, y)dw \tag{2.4}$$

is also intractable. To approximate the predictive distribution in equation 2.4 with the use of Monte-Carlo (MC) approximation method (a method that will be used in the study) requires knowledge of the posterior $p(w \mid x, y)$ (so that we can draw samples from it). Therefore, posterior $p(w \mid x, y)$ approximation is the major objective in Bayesian machine learning research [25]. Few of the most popular techniques to approximate the posterior will be discussed in the following section along with the one that will be used in the present study.

2.4 Bayesian Neural Networks

Neural networks (which will be the models of interest in this study) are most commonly trained in a maximum a posteriori (MAP) setting. MAP provides only a point estimate of the parameters and ignores any uncertainty they might carry. Thus, often leads to overconfident predictions, especially in areas are not close to the data region or not sufficiently covered by training data[26]. Figure 2.4 shows an example of a deterministic and Bayesian neural network.



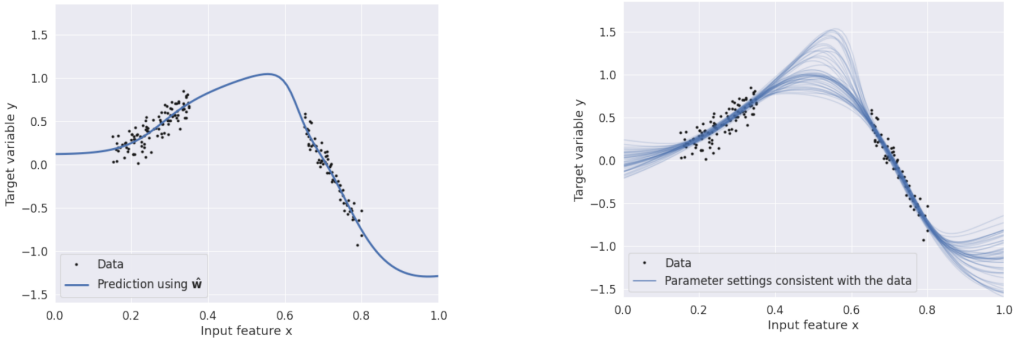
(a) Deterministic NN

(b) Bayesian NN

Figure 2.4: Left plot shows point estimate neural network. Right plot shows a Bayesian neural network with distributions over the weights [27]

Often many parameter settings are consistent with data, but each can lead to very different predictions. With Bayesian models, instead of getting a single set of these parameters we take all sets of model parameters consistent with the data into account. Thus, when marginalising over the entire spectrum of parameters Bayesian models can capture the uncertainty of both the parameter values (aleatoric) and models (epistemic). In this way the Bayesian framework can be seen as a principled way to avoid overconfidence. It is of major importance that a Machine Learning algorithm is able to say "i don't know" and pass the decision to a human being instead. Otherwise the consequences of mistakes can be fatal, for example when driving a car or diagnosing a disease [26].

Figure 2.5 illustrates the predictions made over the parameters with the conventional machine learning on the left and with the Bayesian framework on the right.



(a) Predictions using single set of parameters

(b) Predictions using all the values

Figure 2.5: Predictions from a deterministic neural network on the left and from the Bayesian one on the right

In regions with high concentration of data, the predictions of these models are highly consistent with one another, however this consistency is weak in regions where data is few [25].

For complex and deep models like those that will be using in this study the number of parameters is significantly large thus constitute the computation of Bayesian framework intractable. The log of the posterior distribution will be non convex corresponding to multiple local minima in the error function [24]. Therefore, to compute the posterior distribution, we need to use Bayesian approximation techniques.

The most common technique for approximation of the posterior distribution is Monte Carlo sampling. Though simple method suffers from few disadvantages such as slow convergence for large datasets.

Remark 2.4.1: Markov Chain Monte Carlo

A type of random walk that explores the parameter space of the target distribution in way, where regions with high density are visited more frequently than areas with low density. Draw independent samples from the probability distribution, then repeat this process many times to approximate the desired quantity.

According to Bishop [24] the most complete treatment has been based on the Laplace approximation which we will be using in this study. Though other Bayesian approximations can be considered.

Gal and Ghahramani [28] have introduced Monte Carlo dropout for Bayesian inference in deep neural networks. The key principal of Monte Carlo dropout is that in test time, after we have trained our model with regular dropout, we keep the dropout layers activated and we are sampling from the approximate posterior distribution [25]. This way, we can generate multiple different predictions for each input.

Another, approximation method is the Bayes backprop introduced by Blundell et al. [29] which is an instance of variational inference.

Remark 2.4.2: Variational Inference (VI)

Assume we are given an intractable probability distribution p. Variational techniques will try to find a distribution $q \in Q$ (a class of tractable distributions) that is most similar to p. We will then use the approximate distribution q instead of p to derive a solution.
VI in a nutshell

- Define collection of "simple" approximate probability distributions Q (the variational family)
- Define a measure of "distance" between probability distributions $\mathcal{D}[q||p]$
- Search for the distribution $q \in Q$ that resembles the exact posterior p as close as possible as measured by $\mathcal{D}[q||p]$

The core idea of the method is we want to approximate the underlying true distribution $p(w \mid \mathcal{D})$ with an approximate distribution $q(w \mid \theta)$ which is represented by parameters θ that can be learnt by using the backpropagation algorithm. Then using Monte Carlo sampling, we sample weight values at test time from the approximate distribution q and calculate the uncertainty estimates. This method requires a double amount of weight parameters compared to MC-Dropout [25].

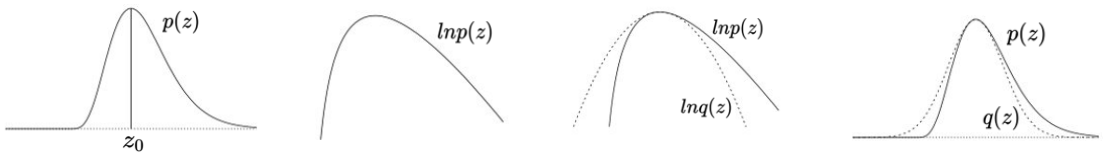
2.5 Laplace Approximation

Laplace approximation is a simple method which similarly to the rest of approximation techniques is used to approximate intractable probability densities. Its overall scope is to find a Gaussian approximation to a probability density defined over a set of continuous variables [24]. Figure 2.6 illustrates the steps involved in Laplace method. To provide better understanding of the method to the reader we will follow the explanation process described in Bishop [24] and MacKay [12]. We first explain Laplace method for a univariate continuous variable $\mathbf{z}$ (eg. one dimension) whose distribution $p(\mathbf{z})$ is defined as

$$p(\mathbf{z}) = \frac{1}{Z} f(\mathbf{z}) \tag{2.5}$$

where $Z = \int f(\mathbf{z}) dz$ is the normalisation coefficient which ensures the integral of distribution is 1 and $f(\mathbf{z})$ is a scaled version of $p(\mathbf{z})$.

The goal of Laplace is to find a Gaussian approximation $q(\mathbf{z})$ which is centered on a mode of distribution $p(\mathbf{z})$ (see plot 2.6a) [24], [12].



(a) (b) (c) (d)

Figure 2.6: The plots show the Laplace process of finding an approximate Gaussian $q(\mathbf{z})$ around the mode of a distribution $p(\mathbf{z})$ [12]. Plot (a) shows the unnormalised distribution $p(\mathbf{z})$ and its mode $\mathbf{z_0}$. Plot (b) shows the $\ln p(\mathbf{z})$ around the mode. Plot (c) visualises the $\ln p(\mathbf{z})$ along with the $\ln q(\mathbf{z})$(dotted lines). Plot (d) illustrates $q(\mathbf{z})$(dotted lines) and target distribution $p(\mathbf{z})$.

Thus, we start by finding the peak point $\mathbf{z_0}$ of the distribution $p(\mathbf{z})$ such that $p'(\mathbf{z}) = 0$. As described in Bishop [24] this is equivalent to

$$\left.\frac{df(\mathbf{z})}{d\mathbf{z}}\right|_{z=z_0} = 0 \tag{2.6}$$

A property of Gaussian distribution $f(z)$ is that its logarithm $lnf(z)$ is a quadratic function of the variables z [24]. Thus, we can make use of Taylor-expansion of $lnf(\mathbf{z})$ centered on the mode $\mathbf{z_0}$ to approximate the $lnf(\mathbf{z})$ using only the quadratic term.

$$\ln f(\mathbf{z}) \simeq \ln f(z_0) - \frac{A}{2}(z - z_0)^2 \tag{2.7}$$

where

$$A = \left.\frac{d^2 \ln f(\mathbf{z})}{dz^2}\right|_{z=z_0} \tag{2.8}$$

Remark 2.5.1: Taylor series

Taylor series represents a function with an infinite sum of terms that are calculated by the derivatives of that function at a single point.
Generally speaking in Machine Learning algorithms sometimes cost functions cannot be differentiated to find the minima. In this case we can use Taylor series to differentiate and approximate the point where the function will be at the point of interest (minima). Hence, Taylor series can be considered as tool to find the minima of the cost function of an algorithm.

By taking the exponential of 2.7 we get an approximation to our function or likelihood $f(\mathbf{z})$

$$f(\mathbf{z}) \simeq f(z_0) \exp\left\{-\frac{A}{2}(z - z_0)^2\right\} \tag{2.9}$$

To obtain a normalised distribution $q(\mathbf{z})$ we need Z to normalise $f(\mathbf{z})$. We thus have

$$Z = \int f(z)dz \simeq f(z_0) \int \exp\left\{-\frac{A}{2}(z - z_0)^2\right\} dz \simeq f(z_0)\frac{2\pi^{\frac{1}{2}}}{A^{\frac{1}{2}}} \tag{2.10}$$

From 2.5 and 2.10 we get the proper normalised approximate distribution

$$q(z) = \frac{1}{Z}f(z) = (\frac{A}{2\pi})^{\frac{1}{2}} \exp\left\{-\frac{A}{2}(z - z_0)^2\right\} = \mathcal{N}\left(\mathbf{z} \mid \mathbf{z_0}, A^{-1}\right) \tag{2.11}$$

The process above gives a detailed overview of Laplace in a single dimension. Easily Laplace can be extended to perform the same task, approximating 2.5 over M-dimensional space $\mathbf{z} \in R^M$. As before we apply Taylor expansion on the mode $\mathbf{z_0}$ to approximate $\ln f(\mathbf{z})$

$$\ln f(\mathbf{z}) \simeq \ln f(\mathbf{z}_0) - \frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^{\mathrm{T}} \mathbf{H} (\mathbf{z} - \mathbf{z}_0) \tag{2.12}$$

where H (MxM dimension) is the Hessian matrix, the matrix of second-order partial derivatives which describes the local curvature of $\ln f(\mathbf{z})$ at $\mathbf{z_0}$ [30]. The hessian is defined by

$$\mathbf{H} = -\nabla\nabla \ln f(\mathbf{z})|_{\mathbf{z}=\mathbf{z}_0} \tag{2.13}$$

The final steps are the same as before. Exponentiating 2.12 and including the normalisation constant Z for a multivariate normal we have the approximate multivariate distribution

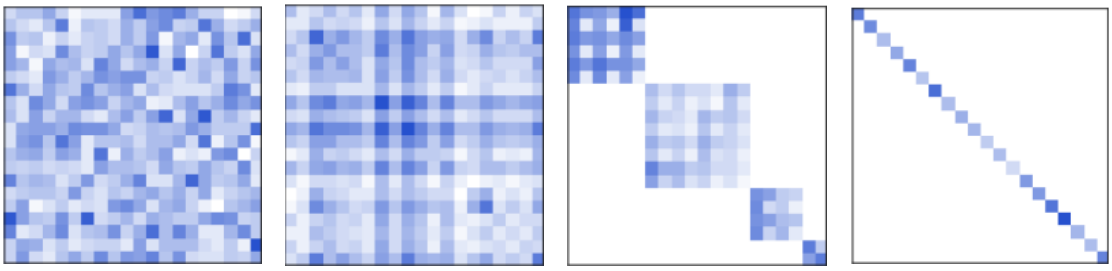
$$q(\mathbf{z}) = \frac{|\mathbf{H}|^{1/2}}{(2\pi)^{M/2}} \exp\left\{-\frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^{\mathrm{T}} \mathbf{H} (\mathbf{z} - \mathbf{z}_0)\right\} = \mathcal{N}\left(\mathbf{z} \mid \mathbf{z}_0, \mathbf{H}^{-1}\right) \tag{2.14}$$

Utlimately, the Laplace approximation method narrows down to two steps. First find the mode $\mathbf{z_0}$ (eg. using an optimisation algorithm) and second evaluate the Hessian at that mode (eg. fitting a Gaussian centered at that mode).

Though evaluating the Hessian is often infeasible due to its size which scales quadratically with the number of network parameters. Additionally, for nonlinear systems such as neural networks the Hessian is not necessarily positive definite. To overcome the later (potential) issue one can use the positive semi-definite approximations such as Fisher information matrix and generalized Gauss-Newton matrix (GGN) on the log-likelihood (Hessian depends both on the log-prior and the log-likelihood [31]. The prior is trivial to compute thus attention is put on the log-likelihood).

The aforementioned approximations are still quadratically large. We need to apply further factorisation schemes that makes the computation (and storage) of the Fisher and GGN efficient [31]. In figure 2.7 the most common schemes are shown. The full factorisation implies that there is no approximation and it can be used when the size of Hessian is relatively small. The Hessian/Fischer/GGN can be approximated using low-rank approximations by leveraging eigendecompositions of the entire matrix (see 2.7b). Kronecker-factored approximate curvature (KFAC)

(see 2.7c) block-diagonalizes the Fischer, GGN where each diagonal block is corresponding to parameters of each layer of the neural network. Then approximates each block with the Kronecker product of two matrices. Lastly the simplest of the methods is diagonal factorisation which takes into account only the diagonal elements.



(a) (b) (c) (d)

Figure 2.7: The plots show the factorisation schemes. From left to right we have Full, Low Rank, KFAC and Diagonal [31]

2.6 Related Work - Probabilistic STN

Schwöbel et al. [4] carried out a study applying amortised variational inference on a STN for learned data augmentation. For the localiser part they introduced two neural networks that accepts an observation (eg. image) as input, and outputs the mean and variance parameter for the latent variable associated with that observation. Then use those parameters to sample transformations and feed them to another network which performs the down stream task (eg. classification). Then using back propagation using the parameterisation trick [32] optimise the parameters of the neural networks instead of the individual parameters of each observation.

The benefit of this approach is that one only needs to specify the parameters in the localiser which are not dependent on the number of observations. Meaning that the number of variational parameters is constant with respect to data size. This also allows for easier approximation on a new observation. The experiments performed in the study shows that indeed using transformation samples for further training of the classifier improves its performance. Indicating that the data augmentation using Bayesian inference is useful.

However, despite the approximate posterior is Gaussian, there is an additional constraint imposed by requiring that all the variational parameters lie in the range of the network. This is known as the amortisation gap [33]. This limitation renders the localiser insensitive to out of distribution data where no optimal transformation can be found. Another limitation of this approach is that it requires to tune the KL-weights every time. For that purpose is used grid research in the study. A process that can increase the training time significantly depending of the complexity of the task.

The present study has used the same principles and uses a simpler approach to perform learned data augmentation and we believe that it can overcome the main limitations of amortised variational inference.

3 Methodology

This section provides an overview of the methods used in this study and gives a general explanation of Laplace on BNN framework. This framework will be applied on the model under study which is also described here. Then, explanation is given on how Laplace is applied on solely a subnetwork of the whole network. A short description of Laplace Redux library is given. The library will be extensively used in our study to perform Laplace approximation. At last calibration methods are also mentioned which will give an understanding on the quality of our probabilistic model compare to the deterministic one.

3.1 Bayesian Inference with Laplace Approximation

In section 2.5 we showed the main principles behind Laplace approximation. In this section we will perform inference using Laplace which will be used in the model of interest.

We assume we have observed data $\mathcal{D} = \{y_i, X_i\}_{i=1}^N$ where y is the target variable (e.g.class label), and X are observations. In the present study, we will consider the latter to be images. The learning process produces a model $p(y \mid X, w)$ that maps the example's feature vector X to an output variable y, and the mapping comes from a family of models, parametrised by w, a d-dimensional vector, $w \in \mathcal{R}^d$. To turn our model into a Bayesian one we impose an isotropic prior (as discussed in section 2.3) on $p(w) = \mathcal{N}(w|0, \lambda I)$ on parameter's vector, which acts as a regulariser. The training procedure finds the optimal parameter vector $\hat{w}$ as an optimum of the log-likelihood of the unnormalised posterior, $\mathcal{L}(w)$ (MAP estimate):

$$\hat{w}_{\text{MAP}} = \text{argmax}_{\text{w}}\, \mathcal{L}(w) = \text{argmax}_{\text{w}}\, p(\mathcal{D}|w)p(w) = \text{argmax}_{\text{w}}[\log p(y|X, w) + \log p(w)] \tag{3.1}$$

The unnormalised posterior $\mathcal{L}(w)$ can be normalised as follows:

$$p(w \mid \mathcal{D}) = \frac{1}{Z} p(y \mid X, w)p(w) \quad Z := \int p(y \mid X, w)p(w)dw \tag{3.2}$$

Z is the marginal likelihood which is intractable.

After finding the MAP estimate $\hat{w}_{MAP}$, we follow the process described in section 2.5 and approximate the posterior by finding the Hessian using second order Taylor expansion (calculating the second derivative of the log posterior around $\hat{w}_{MAP}$). Then we exponentiate and normalise using the normalisation constant Z similar to 2.14.

Remark 3.1.1: Marginal Likelihood

The marginal likelihood can be approximated to

$$Z \approx \exp\left(-\mathcal{L}\left(\mathcal{D}; w_{\text{MAP}}\right)\right)(2\pi)^{D/2}(\det \Sigma)^{1/2} \tag{3.3}$$

The marginal likelihood is useful for model selection and hyper parameter tuning and will be used in the study for tuning the prior precision.

Eventually, the approximate posterior takes the desired form of a full covariance Gaussian with a covariance matrix H^{-1} [34].

$$p(w \mid \mathcal{D}) = q(w) = \mathcal{N}(w|\hat{w}_{\text{MAP}}, \mathrm{H}^{-1}) \tag{3.4}$$

However, the goal is to be able to make predictions for new data points. The predictive distribution derives from marginalisation of the posterior $p(w|\mathcal{D})$ which is equal

$$p(y^*|X^*, \mathcal{D}) = \int p(y^* \mid X^*, w)p(w \mid \mathcal{D})dw \tag{3.5}$$

The posterior predictive distribution translates uncertainty in weights to uncertainty in predictions [34]. However, this distribution is intractable and further approximation needed. The simplest approximation method is Monte Carlo integration using S samples. As a result the 3.5 approxiamted as follows:

$$p(y^* \mid X^*, \mathcal{D}) = \frac{1}{S}\sum_{i=1}^{S} P(y^* \mid X^*, w^i) \tag{3.6}$$

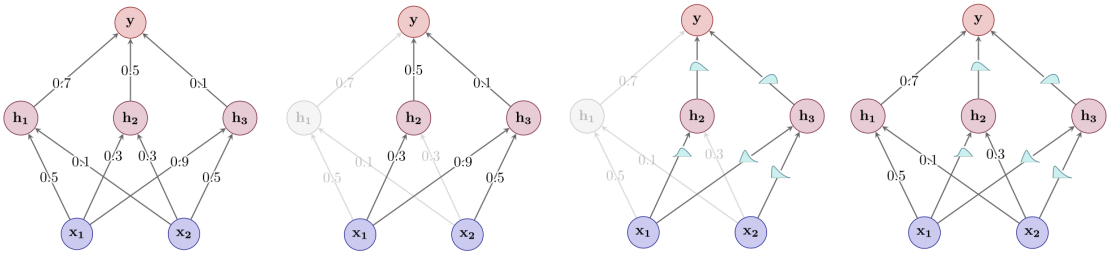
where

$$\mathbf{w}^{(i)} \sim q(\mathbf{w})$$

3.2 Subnetwork Inference

One of the main advantages of subnetwork inference is that overcomes the computational constraints rising from calculating the Hessian over all parameters of the network. According to Daxberger et al. [34] the posterior predictive distribution of a full network can be represented by that of a subnetwork. Also, in their research Izmailov et al. [35] shown that effective inference can be performed on a low dimensional subspace of the parameters space. Subnetwork inference has further been motivated by findings that neural networks can be prunned without affecting test accuracy [36] as well as that around the space of a local optimum, there are many dimensions that leave the predictions unchanged [37]. Suggesting that a small subnetwork can contain the predictive power of a neural network.

Figure 3.1 shows the general steps behind the subnetwork inference process. It starts by training the model on MAP to obtain point estimates on weights $\widehat{\mathbf{w}}$, then select



(a) Point estimation (b) Subnetwork (c) Inference (d) Prediction

Figure 3.1: The plots show the process of applying Bayesian inference on a subset of weights [34].

the subnetwork ($\mathbf{w}_S \in \mathrm{R}^S$, S « D) and apply inference. At last we make predictions using the full network with a mix of Bayesian and deterministic weights [34].

As described in Daxberger et al. [34] the inference over sub weights uses the following approximation over the posterior

$$\begin{aligned} p(\mathbf{w} \mid \mathcal{D}) &\approx p\left(\mathbf{w}_S \mid \mathcal{D}\right) \prod_d \delta\left(\mathbf{w}_d - \widehat{\mathbf{w}}_d\right) \\ &\approx q\left(\mathbf{w}_S\right) \prod_d \delta\left(\mathbf{w}_d - \widehat{\mathbf{w}}_d\right) = q_S(\mathbf{w}) \end{aligned} \tag{3.7}$$

The first approximation breaks down the full posterior $p(\mathbf{w} \mid \mathcal{D})$ into a posterior $p\left(\mathbf{w}_S \mid \mathcal{D}\right)$ over the subnetwork $\mathbf{w}_S \in \mathrm{R}^S$ and fixed, deterministic values $\widehat{\mathbf{w}}_d$ to the $D - S$ remaining weights $\mathbf{w}_d$ [34],[31]. In other words, the remaining weights w_d are simply set to their MAP estimates (eg. $\widehat{\mathbf{w}}_d = \mathbf{w}_d^{MAP}$), requiring no additional computation [31].

Since posterior inference over the subnetwork is still intractable, 3.7 further approximates $p\left(\mathbf{w}_S \mid \mathcal{D}\right)$ by $q\left(\mathbf{w}_S\right)$. Hence, the resulting approximate subnetwork posterior $q_S(\mathbf{w})$ with Laplace takes the form [34]:

$$q_S(\mathbf{w}) = \mathcal{N}\left(\mathbf{w}_S \mid \widehat{\mathbf{w}}_S, \mathbf{H}_S^{-1}\right) \prod_r \delta\left(\mathbf{w}_d - \widehat{\mathbf{w}}_d\right) \tag{3.8}$$

Subnetwork's size S will be chosen so that it will be significantly smaller than the entire network i.e. $S \ll D$. For sufficient small subnetwork S, it is feasible to compute and store the full $(S \times S)$ subnetwork's covariance matrix ($\mathbf{H}_S^{-1}$). As a result, we can discover informative dependencies across the weights in the subnetwork [31].

However, it can be challenging to define the proper subnetwork directly (which ideally should produce a predictive posterior closest to full network's predictive distribution [34]). To overcome this burden Daxberger et al. [34] proposes a strategy that minimises the Wassertein distance between the sub and full network's weights posteriors. Though in our case we don't apply that approach and explicitly choose the layer of interest that we will perform inference over. The subnetwork has been

chosen as part of the localisation network under the assumption that will allow us to draw samples to perform a type of data augmentation.

3.3 Probabilistic STN

Assuming we have the same dataset as in section 3.1 $\mathcal{D} = \{y_i, X_i\}_{i=1}^N$ where y is the target variable (e.g.class label), and X are observations of the covariates. Our model consists of two parts the classifier and the STN module.

The STN localiser network estimates a transformation $\theta(X)$ that transforms a coordinate grid and interpolates an image accordingly (see sections 2.2, 2.1). The classifier then receives those transformations and computes $p(y|T_\theta(X))$. In this study we replace the deterministic transformations $\theta(X)$ with the posterior over transformations $p(\theta|X)$.

Thus, our model becomes

$$p(y|X) = \int p(y, T_\theta|X)d\theta = \int \underbrace{p_{w_o}(y|T_\theta(X))}_{\text{Classifier}} \underbrace{p_{w_s}(T_\theta|X)}_{\text{Localiser}} d\theta \tag{3.9}$$

w_o and w_s refers to the deterministic weights of the model and those that we apply Laplace respectively. Thus we will apply Laplace approximation on the posterior $p_{w_s}(T_\theta|X)$. The weights of the classifier maintain their deterministic nature. In the STN we will apply inference only on a subnetwork of the module thus the rest of the weights will remain deterministic as well.

The classifier part yields $p(y \mid T_\theta(X)) = y$ and the localiser part $T_\theta \sim p_{w_s}(T_\theta \mid X)$.

Figure 3.2 illustrates the model under study. We have the two networks localiser and classifier. The idea is to apply Laplace approximation on a subset of weights in the localiser network. We choose a subset instead of the whole localisation network for computational purposes. Subnetwork inference applied based on the process described in section 3.2.

We aim to use the approximate posterior over the weights of the subnetwork to generate various transformations which then will be used for continuous training of the classifier on the test time evaluating the equation 3.6.

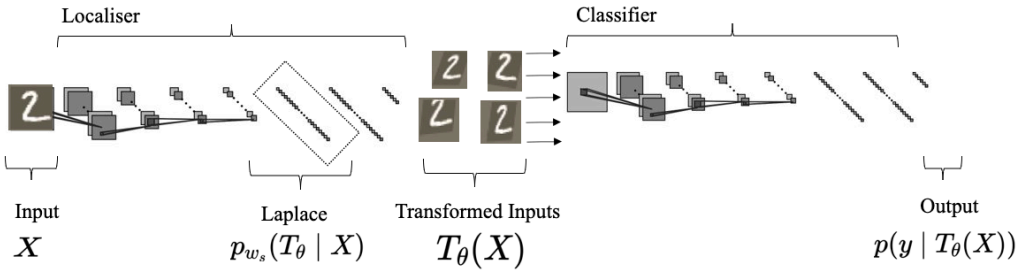


Figure 3.2: Probabilistic STN

Specifically, we continue the training by generating one sample from the transformations (S = 1) and pass it to the classifier. In the implementation of VAE, Kingma

et.al [32] suggest sample size 1 along with relatively large minibatch size. The minibatch was between 128 and 256 depending on the experiment. Throughout the training process the weights of the localiazer network are fixed meaning that they are not updating to learn new transformations. Then on the testing we generate 10 samples from the transformation (S = 10) which are used to increase the batch size each time and they pass through the classifier to give the predictions.

3.4 Laplace Redux Library

Though the process behind Laplace is intuitive, in practise it can be quite difficult to apply. It requires a lot of experimentation and efficient computation and storage of the Hessian. Thus, it can take significant amount of time to prototype with Laplace before one can end up with a well functioning solution.

In the present study will make use of Laplace Redux library. A newly developed library made by Daxberger et al. [31]. The library provides all the necessary functionalities to apply Laplace approximation. One of those is the subnetwork API which allows to apply inference on selected modules (eg. layers) of the network. The subnetwork API follows the principles mentioned in section 3.2. This specific API will be used extensively throughout the study. Also the approximations and factorisation schemes mentioned in section 3.1 are included in the package (Though for subnetwork are limited to full and diagonal factorisation only). Thus, it allows an efficient computation of the Hessian.

The inference it will be performed post-hoc. Meaning that the we will train the network first and will apply Laplace afterwards. Experiments made by the authors showed that post-hoc inference can improve calibration of the model without affecting the accuracy. The library doesn't provide an intuitive way to perform inference at training time.

3.5 Calibration Measures

Its calibration is an important factor for a model's interpretability. Given how well or not a model is calibrated we can make conclusions on whether model's predictions are trustworthy. In other words calibration of a model help us to assess whether a model is over or under confident over it's predictions. Modern deep neural networks are often overconfident in their predictions while the implementation of Bayesian framework can overcome this issue.

Throughout the experiments in the study will provide information about models calibration and evaluate whether or not the Bayesian inference improved their calibration. We will use reliability plots and calibration curves to visually show and compare the calibration between the deterministic and Bayesian neural networks. Additionally, expected calibration error (ECE) and negative log likelihood (NLL) will provide scalar indications on models' quality.

Remark 3.5.1: Entropy

Occasionally in the study the entropy will be used to assess the uncertainty of a model. Entropy in general is given by the equation below [24].

$$\mathrm{H}[x] = -\sum_{x} p(x) \log p(x) \tag{3.10}$$

The higher the entropy the less information we can derive from a model meaning high uncertainty.

ECE partitions the predicted probabilities into a fixed number of bins and takes the weighted average of the difference between the fraction of predictions in the bin that are correct (accuracy) and the mean of the probabilities in the bin (confidence)[38] [39]. More precisely,

$$ECE = \sum_{i=1}^{K} \frac{|B_i|}{N} \left|\mathrm{acc}_i - \mathrm{conf}_i\right| \tag{3.11}$$

where acc_i and conf_i denote the accuracy and average confidence in the i-th bin and $|\mathrm{B}_i|$ denote the number of samples in bin B_i.

Larger values of ECE error indicate a larger difference between output confidence and actual model accuracy of the prediction meaning larger miscalibration. Smaller values of ECE indicate less miscalibration. The network model is perfectly calibrated if all output confidence values equal the actual accuracies ie. ECE = 0.

To derive the reliability diagram we plot accuracy as a function of confidence. The difference between the two for a given bin is called the gap and indicates how well the probabilities ensemble the accuracy. Accuracy and confidence for a given bin are calculated from equations 3.12, 3.13 respectively:

$$\mathrm{acc}_i = \frac{1}{|B_i|} \sum_{j \in B_i} \mathbf{1}\left(\hat{y}_j = y_j\right) \tag{3.12}$$

$$\mathrm{conf}_i = \frac{1}{|B_i|} \sum_{j \in B_i} \hat{p}_j \tag{3.13}$$

The negative log likelihood (NLL) also known as the cross entropy loss typically used as an objective function for training and measures the inequality between predicted and actual confidence for the true label. When it's equal to zero, all the data have been predicted correctly and we have 100% confidence. Thus the lower the score the better the calibration. For a probabilistic model $p\left(\mathbf{Y} \mid \mathbf{X}\right)$ and n samples the negative log likelihood is given by [40] :

$$\mathcal{L} = -\sum_{i=1}^{n} \log\left(p\left(y_i \mid \mathbf{x}_i\right)\right) \tag{3.14}$$

Although, ECE has been used extensively throughout the literature as a validation measure of models' calibration [41],[31],[34],[42],[43]. There are studies that have raised concerns with regards to that methodology[44],[39].

Nixon et al. [39] addresses several issues related to ECE metric. They state that ECE cannot sufficiently compute calibration across all predictions when extended to multi-classification problem since it was structured for binary classification problems.

Additionally, they define as major weakness the dispersion of data across fixed ranges. The ECE metric is weighted based on the relative number of samples in each bin. Because the network predictions are typically very confident, a few bins on the right side contribute most to the ECE [39]. The ECE focuses only on making sure that the more confident samples are calibrated.

Another issue related with the number of bins and the data points fall into those is the bias-variance trade-off. A larger number of bins will lead to low biased measures but as fewer samples are allocated into each bin, they will have high variance. At last they mention that some times might be possible to get 0 calibration error because overconfident and under confident predictions fall in the same bin. This effect renders cumbersome to judge whether a model has good calibration due to model improvements or due to cancellation of predictions.

Although Nixon et al. [39] have identified those issues that comes together with ECE their proposed solutions don't provide a robust process of computing the calibration. Therefore, non of these fixes described in their research will be considered in the study. Nonetheless it is important to take into account the shortcomings of ECE method when we refer to improved calibration.

4 Experiments

Under this section we aim to evaluate whether the assumptions for learned data augmentation are valid. First, apply Laplace approximation on the MNIST dataset and then continue with more challenging tasks such as random placement MNIST and Mapiliary traffic signs [45]. All three datasets evaluated on the same way with minor though critical changes on the architecture of network.

For MNIST dataset a small convolutional network was used (2x Conv2d,Maxpool2d, Batchnorm, ReLU) followed by two fully connected layers for both localiser and classifier. For the random placement MNIST an extra convolutional layer added in the classier and an extra fully connected layer on the localiser. For the most challenging Mapiliary dataset the same structure as in random placement MNIST used for the classifier while a deeper and wider network used in localiser.

The additional fully connected layer was added mainly to reduce the parameters during the inference and increase computational efficiency. Unless stated otherwise, throughout this section STN and MAP neural network (NN) will be used interchangeably referring to the model prior the inference (Laplace approximation). For the probabilistic model the terms Laplace and Bayesian Neural Network (BNN) will be used.

4.1 MNIST

The MNIST dataset contains handwritten digits. It has a training set of 60,000 examples, and a test set of 10,000 examples. The digits have been size-normalized and centered in a fixed-size image [46]. MNIST is a a popular dataset which constitute a good candidate for developing prototypes before moving to more complicated datasets.

4.1.1 Sanity Check & Subnetwork Selection

Prior of carrying out the experiment we tested whether the Laplace Redux library functions as described in Daxberger et al. [31]. Additionally, we wanted to test whether our assumption to apply Laplace approximation on the last layer of the localiser was correct. Meaning that could provide good enough transformations to feed to the classifier. To test the library and our assumption we applied the Subnetwork API over a couple of different modules.

To evaluate the behaviour of subnetwork we observed the distribution of samples as well as NLL and ECE values for small and large prior precision values (eg. 0.01 and 1000). The key assumption is that when we have small prior precision we expect to have more variance on the weights and large calibration error while when the prior precision is large the outcome is expected to match the MAP solution. After a series of experiments (see appendix A) we got an abnormal behaviour of last layer. Hence, we ended up using the second last layer of the localiser for Laplace approximation.

We can see the distribution of weights (samples) when inference applied on the 2nd

last layer of the STN module in figure 4.1. We see that for small prior precision the values for samples varies in larger range compare to values for large prior precision.

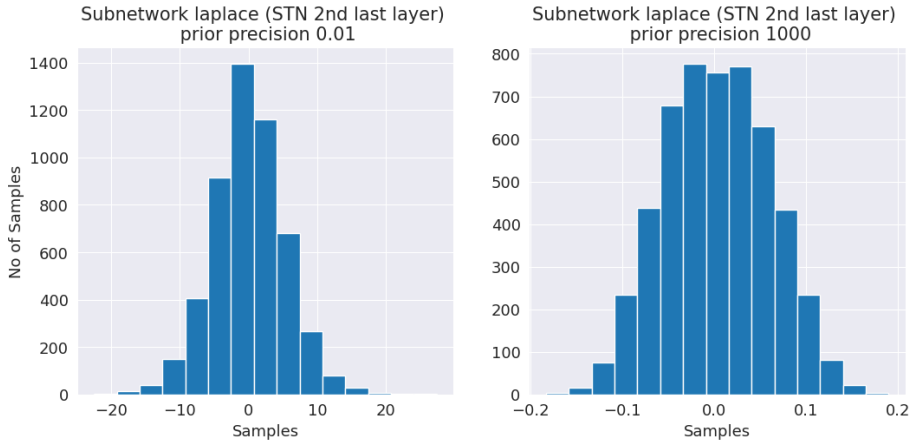


Figure 4.1: Visualization of Subnetwork inference on 2nd last layer of localiser.

Table 4.2 shows the metric values for MAP and Subnetwork inference. The yielded values verify our assumptions about how ECE and NLL should behave when we adjust the prior precision.

	MAP (STN)	Subnetwork STN 2nd Last Layer	
		Small prior prec.	Large prior prec.
Acc ↑	99.4%	98.8%	99.4%
ECE ↓	0.3%	62.6%	0.3%
NLL ↓	0.0202	1.12	0.02

Table 4.1: Acc, ECE and NLL values for MAP (STN),2nd Last Layer STN Subnetwork.

In figures below we see that when we use the second last layer of the STN we have larger variety on the transformed images (see figure 4.2b). On the other hand when the last layer (initial assumption) is used the transformed images are more homogeneous almost identical in the majority of samples (see figure 4.2a).

Hence, the resulting samples from the 2nd last layer looks more promising candidates for data augmentation. In any case further investigation needed to explain the abnormal behaviour of the Laplace approximation when applied on the last layer of the STN module. Based on the findings from the experiments the cause could be either the initialisation of weights and biases on the last layer or the low amount of parameters on that layer.

(a) Samples from last layer (b) Samples from 2nd last layer

Figure 4.2: Samples of the transformation generated from the last layer (a) of the localiser and the second last layer of the localiser (b)

4.1.2 Uncertainty estimation of Subnetwork inference

One of the advantages of Bayesian neural network (BNN) is that it averages all the weights of posterior distribution thus it can capture the uncertainties in data and models. After applying inference on the selected subnetwork we want to investigate how certain is the BNN network compare to the deterministic one on making predictions. We have tried four approaches. We will show how both networks perform on the MNIST data, when the MAP misclassifies the predictions of MNIST data, when we have distribution drift for in-distribution data and on an out-of-distribution dataset (OOD) (KMNIST [47] in this case).

In figure 4.3 we see inputs of the MNIST dataset along with the posterior samples,posterior predictive probabilities and MAP prediction probabilities.

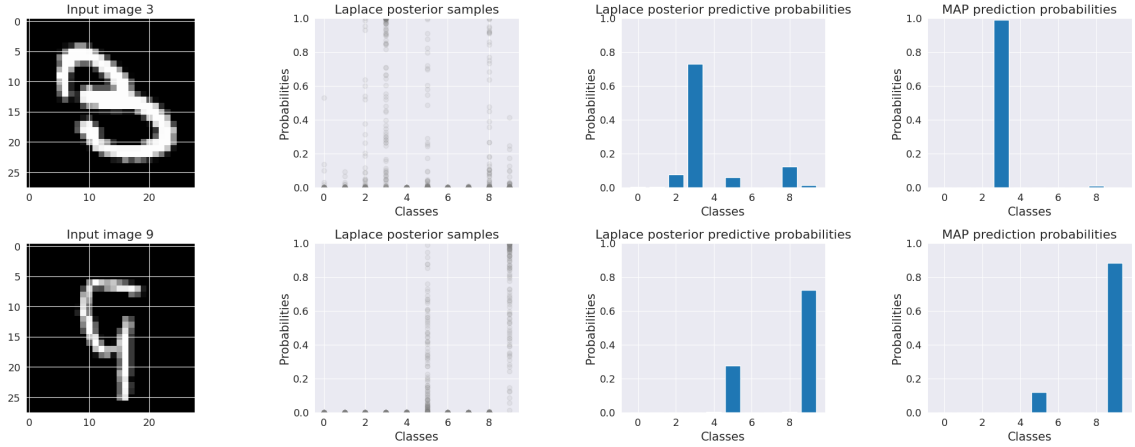


Figure 4.3: Visualization of uncertainties of the network

The figures shows that the BNN is less confident on its predictions compare to the MAP. In almost all cases the MAP is around 100% certain of the outcome although it might not be straight forward in few cases. For example in the first row in figure 4.3 number 3 could easily be mistaken for an 8. Next we show the confidence of the networks when MAP has wrongly classified the given inputs.

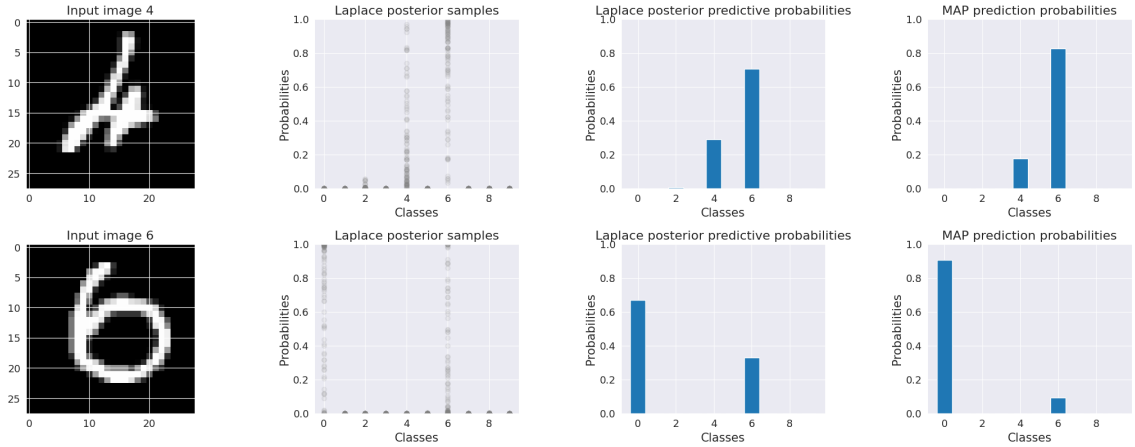


Figure 4.4: Visualization of uncertainties of the network on mis-classified images

In figure 4.4 we see that when MAP mis-classifies an input the Laplace network mis-classifies those inputs too. We also observe both networks to be uncertain on their decisions. Though the MAP assigns higher probability on the wrong classes in all cases. Whereas the Laplace assigns higher probability on the correct classes compare to MAP.

We used rotated MNIST to test the uncertainty of both models when there is distribution drift on in distribution data. We tried the experiment for rotations of 30,60,90,120,150 and 180 degrees. Figure 4.5 shows the performance and calibration of both models for each of the rotation cases.

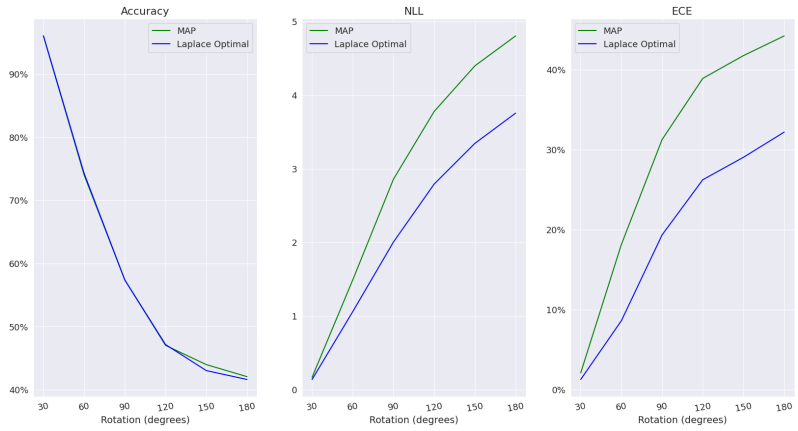


Figure 4.5: Comparison between Laplace and MAP on In-Distribution shift (Laplace optimal indicates that optimal prior precision has been used)

We see that the accuracy gradually decreases as the rotation degrees increases. Especially after 30° the performance in terms of accuracy drops significantly. Following

the drop in test accuracy, NLL and ECE are increasing (meaning worst calibration). Though, the values for Laplace model remain constantly lower than the MAP with the gap between the two to get larger as the shift in data changes. The lower values in NLL and ECE indicate that the Bayesian model maintains better calibration.

Figure 4.6 illustrates two examples of 120° (top row) and 150°(bottom row)

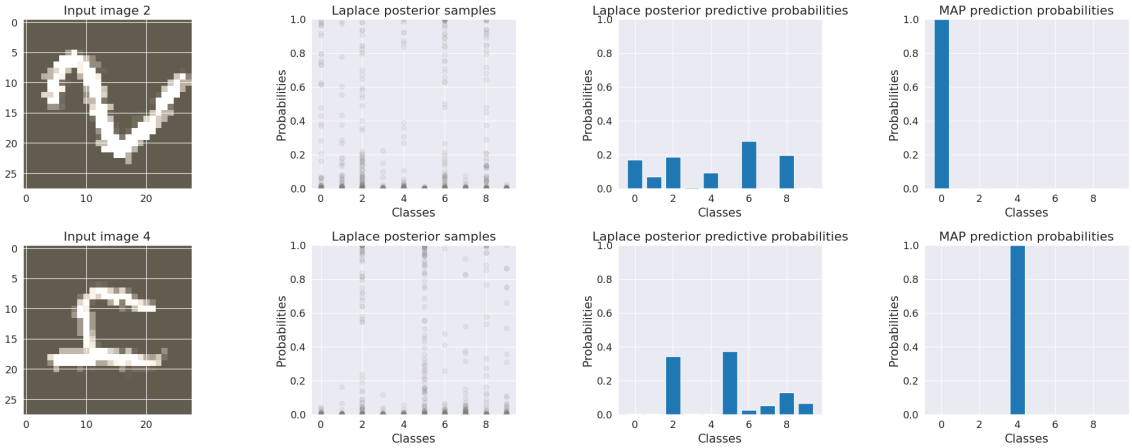


Figure 4.6: Visualization of uncertainties of the network on In-Distribution shift

The 120° example on top shows that the Laplace model cannot correctly classify the given image yet its uncertainty is quite high as predictive probabilities for all classes is below 40%. On the contrary the MAP model predicts the wrong class with 100% confidence. In the 150° example at the bottom we observe that the Laplace model, again, cannot predict the correct class, however the predictive probabilities for all classes remain again below 40% indicating that the model is very uncertain on its predictions. In this example the MAP model correctly classifies the given image with 100% confidence.

Lastly we tested the uncertainty of the networks on out of distribution dataset. Specifically, we used the models trained on MNIST to classify images from KMNIST.

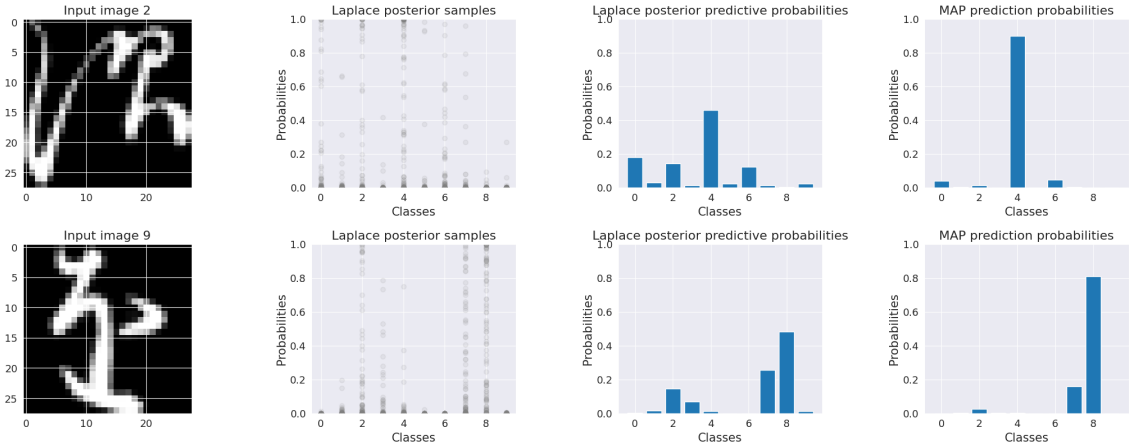


Figure 4.7: Visualization of uncertainties of the network on OOD images

Figure 4.7 shows clearly that Laplace isn't able to classify with certainty any of the images. The majority of probabilities are below 50% . If we had imposed a rejection

rate even on 80% Laplace wouldn't be able to classify almost any of the images. On the contrary MAP shows again high confidence on its decision yet all the predictions are wrong.

Figures 4.6, 4.7 are good examples that Bayesian inference with Laplace (at least in this case) is more trustworthy than MAP when we want to make predictions. Also shows that we could easier detect corrupted data compare to the deterministic network.

We further assessed additional functionalities of Laplace library to ensure whether or not the successfull uncertainty estimation of the Laplace is a result of the subnetwork selection. The functionalities used were the Last Layer Laplace and Diagonal Laplace over all the weights of the network. All performed equally well on capture the uncertainties. Thus, by using any other inference would have led to the same outcome. The advantage of using subnetwork is that we can sample transformations that we will later use for continual training of the classifier.

For the OOD experiment we tested our assumption that Laplace can capture the uncertainty better than the amortised variational inference technique used by Schwöbel et al. [4]. We run the same experiment with the same network architecture using amortised variational inference and we measure the entropy of the yielded distribution and compare it with Laplace and MAP.

	MAP (STN)	Laplace	pSTN [4]
Entropy ↓	0.54	1.07	1.05

Table 4.2: Entropy score for MAP, Laplace and pSTN on the OOD.

We see that Laplace and amrotised variational inference are almost equally uncertain when tested on out of distribution data since both have quite high entropy score.

4.1.3 Subset Evaluation

In this section we aim to evaluate the performance of Laplace approximation on subsets of MNIST dataset compare to MAP. We will use subsets of 100,500, 2000, 5000 and 10000 inputs. In figure 4.8 below we see the accuracy, negative log likelihood and expected calibration error for all subsets.

We demonstrate how well the uncertainty of the posterior predictive distribution matches the quality of the predictions.

As we have seen already the implementation of Laplace approximation doesn't affect the accuracy of the model. Thus we observe identical performance on the accuracy metric as shown on the corresponding plot on the left in figure 4.8. On the other hand we observe that the Laplace approximation has improved the negative log likelihood and calibration of the model in the entire spectrum of MNIST subsets.

Obviously, the biggest improvements are observed on the small datasets and tend to get reduced significantly as we increase the amount of data.

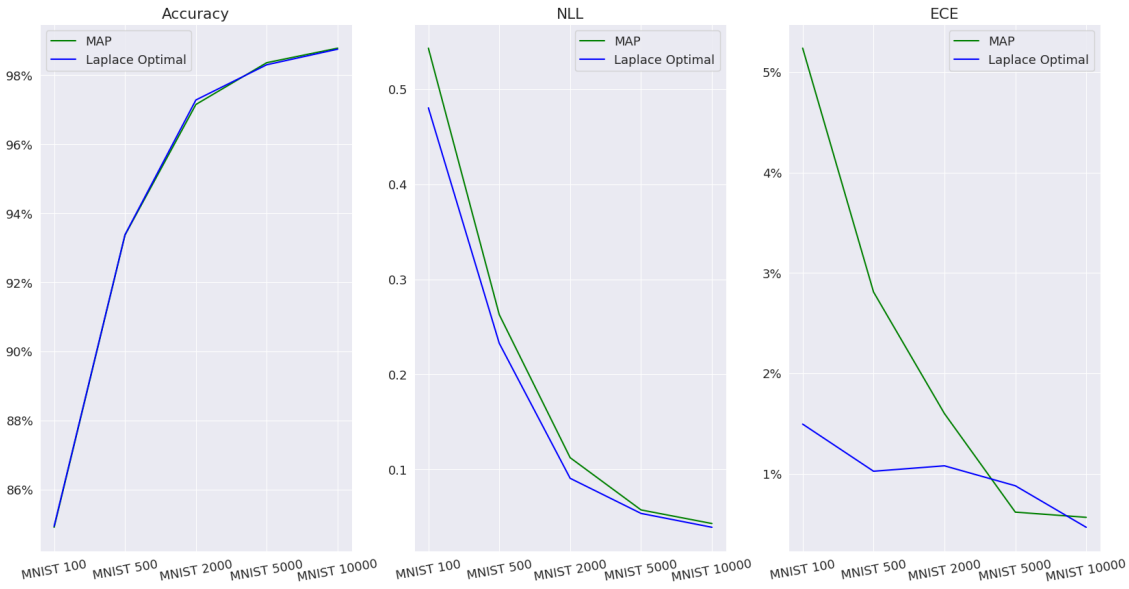


Figure 4.8: Comparison between Laplace and MAP on MNIST subsets

In figure 4.9 we see how the models are calibrated when we look on the MNIST 100 subset as well as on the full dataset.

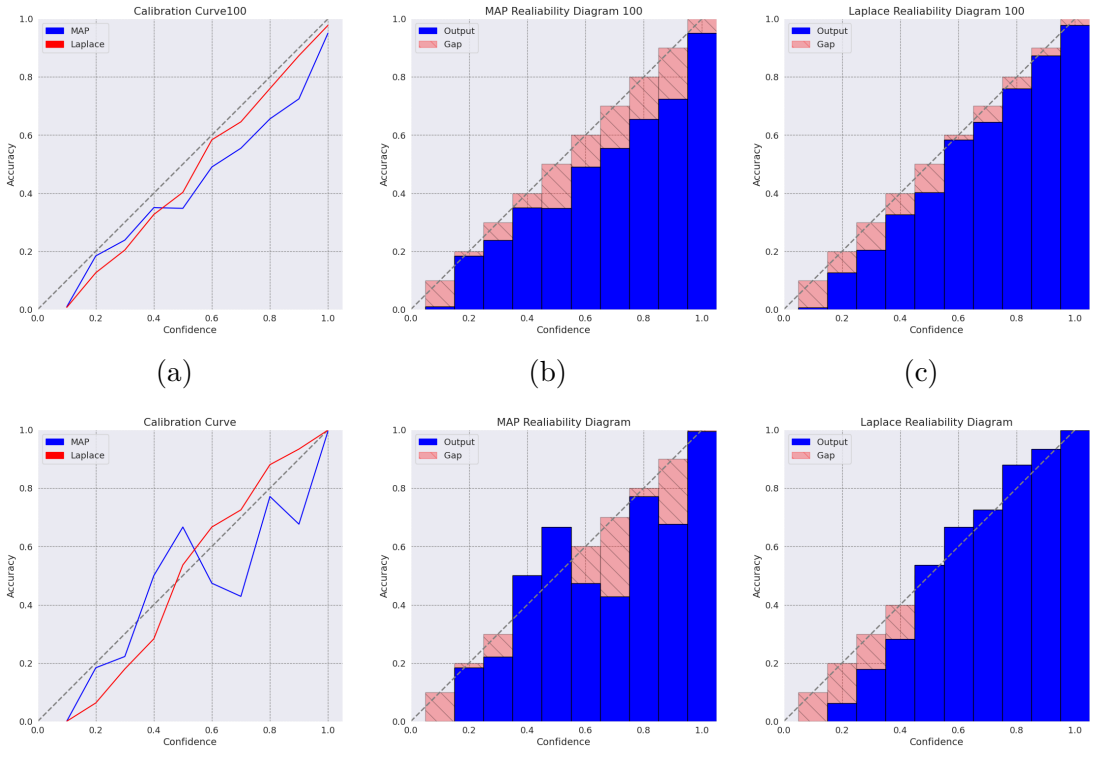


Figure 4.9: Calibration curves and Reliability diagrams for MNIST 100 (top row) and full dataset (bottom row). On the plots (a,d) we see the calibration curves for both Laplace and MAP. On (b,e) and (c,f) we see the reliability diagrams for MAP and Laplace respectively

The plots on the left of figure 4.9 shown the calibration curves between the two models while the plots in the middle and right shown the reliability diagrams for MAP and Laplace models respectively. According to Daxberger et al. [31] one advantage of applying the Laplace approximation is to improve calibration at some extend and when is possible (This was also observed on the in-distribution drift data in previous section).

For the MNIST100 is quite clear that MAP is overconfident. This shown in 4.9a where the calibration curve for MAP (blue line) is below calibration line (dotted line) meaning that the confidence is higher than the actual accuracy. On the other hand Laplace is very close to the calibration line meaning that confidence and accuracy are almost equal in some areas. In reliability diagrams 4.9b and 4.9c we can see better the difference between confidence and accuracy for MAP and Laplace respectively (The difference is called gap and indicated with red colour on the plots. For more details refer to section 3.5). As a result we can see how the Laplace implementation has improved the model's calibration.

For the full dataset the MAP solution is very well calibrated thus we can observe only a slight improvement in calibration when we implement inference. This is especially seen on the areas where the true predictions is on high values while on the low values we see that Laplace doesn't impact the calibration (these can be seen on both datasets).

4.1.4 Continual training

In previous sections we showed that the implementation of Laplace approximation improves the calibration of the model(mainly on the small datasets) as well as allows to sample transformations from the STN module. Those transformation samples produce different transformed versions of the input image. This corresponds to a type of data augmentation. Therefore those images could be used by the classifier for further training aiming to improve it's accuracy (which hasn't changed with inference so far).

In this section, we will continue training the model on test phase for each one of the subsets using the sample transformations as inputs.

On table 4.3 we see the test accuracy scores of the continual training process for each one of the MNIST subsets.

	MNIST 100	MNIST 500	MNIST 2000	MNIST 5000	MNIST 10000
MAP	84.9% ±0.05	93.4% ±0.11	97.1% ±0.12	98.3 % ±0.06	98.8% ±0.02
Laplace	**89.95%** ±0.65	**95.56%** ±0.38	**97.74%** ±0.21	**98.74%** ±0.13	**99.07%** ±0.03
Optimal prior precision	1.61	4.52	8.36	13.41	18.74

Table 4.3: Test accuracy scores for each subset after applying post hoc training.

We observe that the performance of the model has been increased across the subsets. The improvement is more visible for the smaller subsets(eg. MNIST 100) and is decreasing as we increase the number of data. Figure 4.10 gives a visual representation of the aforementioned observation.

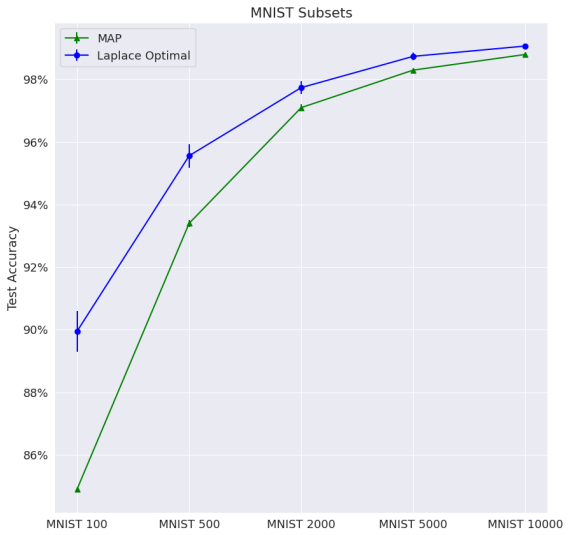


Figure 4.10: Accuracy scores for each subset after continous training.

The optimal prior precision is increasing with the number of data inputs. To get the optimal prior precision we maximized the marginal likelihood (equation 3.3) on the subnetwork. However, since we find the marginal likelihood only for a subset of network might not be good representative of the entire model. Thus, we carried out cross validation using the subnetwork approximation directly [34]. The values for prior precision λ were in the range of $\lambda = [0.01 - 30]$. Both ways yielded the approximately the same outcome.

As described on a similar study by Schwöbel et al. [4], from the entire process we can conclude that the improvement on classifier's performance can be viewed as an indicator for having learned a useful data augmentation scheme. This supports also the hypothesis that data augmentation is especially useful when data is a limited resource.

4.2 Random Placement MNIST

In this section we tried the Laplace approximation in a more challenging task namely random placement MNIST. We randomly sample an x and y coordinate and place the MNIST accordingly on a black background, after downscaling it by 50%. In this experiment we have the scale parameter fixed and no rotation is applied, i.e.$\theta_{\text{true}} = [0, 0.5, t^x_{\text{true}}, t^y_{\text{true}}]$

Figure 4.11a illustrates the random placement MNIST on the left panel which used as input to the model. On the right panel 4.11b are the trasformations derived from the localisation network of the STN module. We see that the localisation network scale in and learns parameters x and y to get the correct direction. The test accuracy of the model is 98.56%.

(a) The random placement MNIST inputs (b) The transformations from STN

Figure 4.11: Visualization of (a) the random placement MNIST and (b) the transformations as generated by the STN module

We carried out the same experiments as in MNIST. Thus, firstly verified the assumptions about the subnetwork selections. The conclusion is the same as in the MNIST and the second last layer of localisation network selected for inference (see further results in appendix A).

Figure 4.12 illustrates the samples transformation from the STN module after inference.

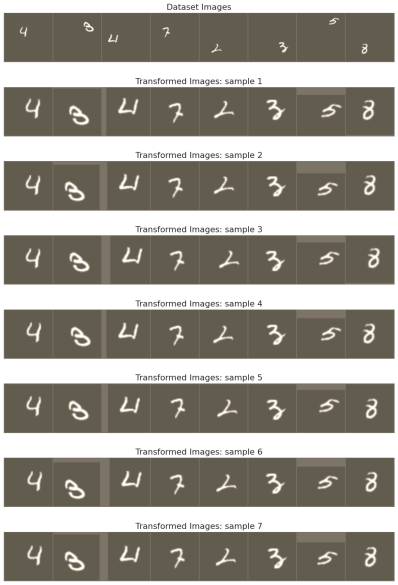


Figure 4.12: Results of subnetwork Laplace on second last layer of STN module for random placement MNIST.

In this case we see that the transformations are more homogeneous due to the fixed scale parameters in the STN modules. The localisation network learns the x,y parameters allows solely small transformations. Though the difference among the samples are not significant yet there is a degree of variation.

Then we evaluate visually how uncertain is the deterministic model compare to probabilistic one after we apply the subnetwork inference with Laplace approximation. From left to right figure 4.13 shows the input: randomly placed digits. Then shows the samples generated from the posterior distribution via Monte Carlo sampling. Then follows the posterior predictive probabilities and the MAP predictions.

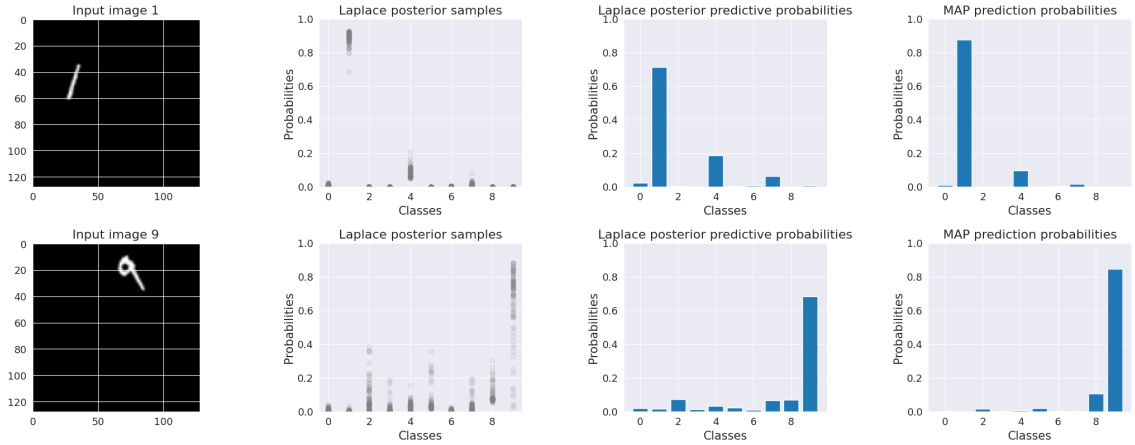


Figure 4.13: Visualization of uncertainties of the network

From the figure 4.13 we see again that the probabilistic model is slightly more conservative on its predictions than the MAP which shows higher confidence in all samples.

Looking into images that have been mis-classified (figure 4.14) from MAP we see that overall both models have high degree of uncertainty.

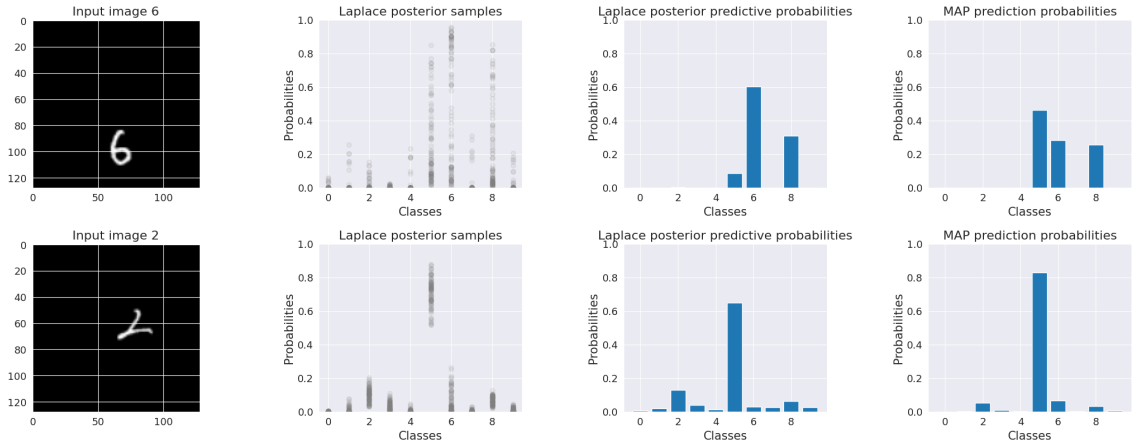


Figure 4.14: Visualization of uncertainties of the network for mis-classified data

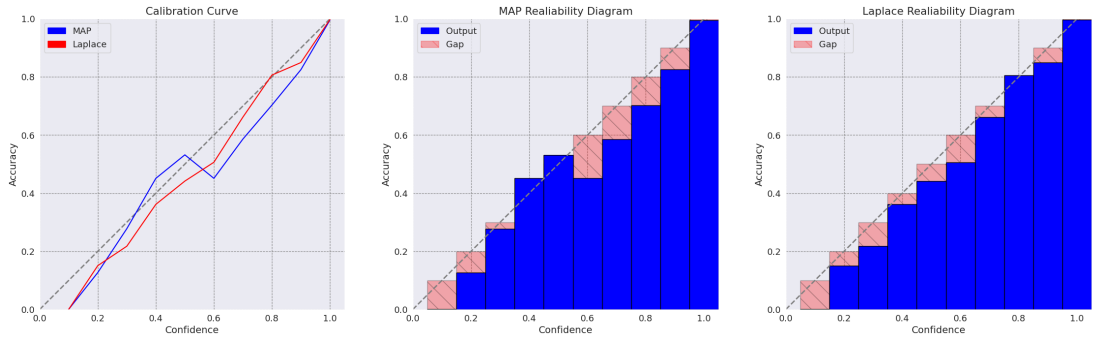
However, we observe in the fist row the MAP model assign higher probability on the wrong class compare to Laplace which classifies the input correctly though with

high uncertainty. On the last image both models mis-classify the image yet Laplace maintains higher uncertainty.

The calibration curve and reliability plots for the Laplace and MAP in figure 4.15 indicates that both models are very well calibrated with Laplace to be slightly better in areas with high confidence. The table 4.4 show the accuracy (Acc), ECE and NLL scores for both models. The accuracy isn't affected by the inference while the expected calibration error is slightly improved after Laplace. The negative log likelihood is approximately equal in both models. The results, for Laplace are after optimizing the prior precision by maximizing the marginal likelihood which yields 35.09.

	MAP (STN)	Laplace
Acc ↑	98.6%	98.6%
ECE ↓	0.4%	0.2%
NLL ↓	0.043	0.046

Table 4.4: Acc, ECE and NLL values for MAP (STN), and second Last Layer STN Subnetwork for random placement MNIST.



(a) (b) (c)

Figure 4.15: Calibration curves and Reliability diagrams for the full dataset of random placement MNIST. On then (a) we see the calibration curves for both Laplace and MAP (STN). On (b) and (c) we see the reliability diagrams for MAP and Laplace respectively

To conclude the experiment we performed the continual training as we did in section 4.1.4. This time though we didn't use subsets and used the entire dataset instead. In table 4.5 we see the accuracy before and after inference with Laplace. Similar to MNIST experiment we see that for large amount of data the usage of Bayesian inference for data augmentation has small impact on classifier's performance.

	Random placement MNIST
MAP	98.6% ±0.16
Laplace	**99.02%** ±0.11
Optimal prior precision	35.09

Table 4.5: Test accuracy scores after applying post hoc training on the entire dataset of random placement MNIST.

The experiment on the random placement MNIST agrees at a great extend with the results found on the MNIST dataset. The main difference we observe was the transformations $T_\theta(X)$ were quite homogenous, which might have overfitted the classifier though not such signs observed.

4.3 Mapiliary Traffic Sign Dataset

Mapiliary traffic sign dataset consists of traffic signs from countries around the globe. It contains in total 100.000 images and 313 classes. From 100.000 images 47.547 are partially annotated while the rest 52.453 are fully annotated [45]. In the study only the fully annotated images were used. Originally the fully annotated images were divided into training, validation and test sets, consisting of 36.589, 5.320, and 10.544 images, respectively.

Each of the original images contains more than one traffic signs. In this task we perform a comparison among a simple convolutional neural network (CNN), a deterministic STN (MAP STN) and the probabilistic STN (Laplace). To keep comparison fair among the models we further processed the images to contain only one traffic sign per image. This accomplished by selecting the bounding boxes that are not collide with other bounding boxes (in the same image). For example in figure 4.16 we see two examples of images in the dataset. On 4.16a we see that the signs are very close to each other while on the 4.16b there is space in between. By expanding the bounding boxes by a margin of 150 px we could capture whether the images are overlapping with each other or not. In case of overlapping the images discarded from the final dataset. Solely the top 10 remaining classes used in the study. The final training dataset was limited to 4698 images while 500 was used as test data. The 20% of training images used for validation.

The implementation of the experiment carried out as follows. Initially, we trained a classifier (3x Conv2d,Maxpool2d,Batchnorm, ReLU plus 2 fully connected layers) on the full image (see figure 4.17a). This classifier used as baseline.

For the STN we pretrained a classifier (the same structure as the baseline one) on the ground truth images (meaning the traffic signs only). Then we set up an STN module on top of that. The STN module has 4 x Conv2d,Maxpool2d,Batchnorm, ReLU plus 3 fully connected layers. During the training of the STN model the pretrained weights of the classifier used thus solely the localiser's network (STN part) weights were updated.

(a) Traffic signs with no enough space in between

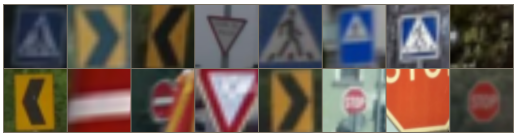
(b) Traffic signs with space in between

Figure 4.16: Example of raw data on Mapiliary dataset

After training, Laplace approximation performed on the second last layer of the localiser. However, this time a wider fully connected layer applied consisting of approximately 32000 parameters due to the complexity of the data. For this amount of parameters diagonal approximation applied to handle the size of the Hessian matrix while for the previous cases full approximations provided satisfactory results.

Figure 4.17a shows the input images on the network while the 4.17b shows the outputs of the STN. We can see that the STN has overall captured the correct images (traffic signs). There are though few cases where the transformed image doesn't ensembles the ground truth image correctly. Consequently the downstream task to classify those images properly becomes more difficult.

(a) Full traffic sign image inputs



(b) The transformations from STN

Figure 4.17: Visualization of (a) input data and (b) the transformations as generated by the STN module for the Mapiliary dataset

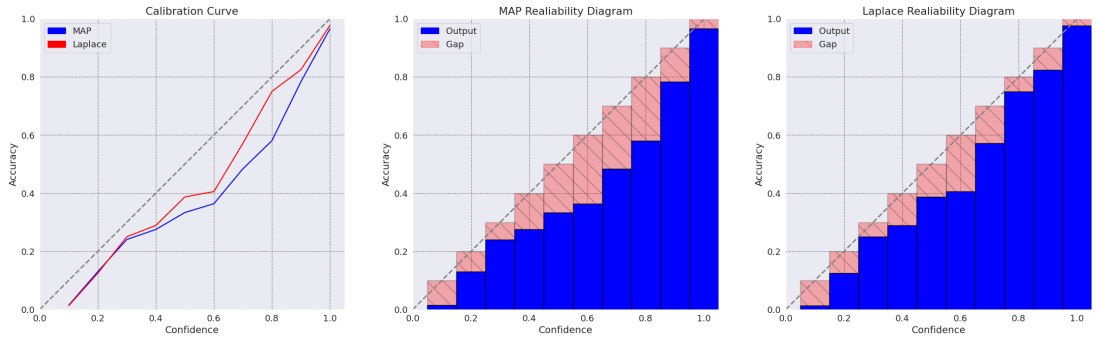
The results on accuracy, expected calibration error and negative log likelihood for each model are shown on table 4.6.

	CNN	MAP (STN)	Laplace
Acc ↑	69.8%	79.4%	79.4%
ECE ↓	6.4%	6.1%	3.1%
NLL ↓	0.752	0.680	0.657

Table 4.6: Acc, ECE and NLL values for CNN, MAP (STN) and Laplace on Mapiliary dataset.

From the table 4.6 is obvious that the localiser network has improved the performance of the model at a great extend by increasing the test accuracy from 69.8 % to 79.4 %. Additionally, has also improved its calibration. The ECE calibration has slightly been reduced compare to CNN while the NLL has shown greater decrease. After applying the Laplace approximation we see that the test accuracy remains unchanged compare to MAP STN (as observed in the other experiments) while the calibration records a significant improvement by dropping the ECE from 6.1% to 3.1% and NLL from 0.680 to 0.657.

The calibration between the MAP STN and Laplace can be further seen on the plot 4.18.



(a) (b) (c)

Figure 4.18: Calibration curves and Reliability diagrams for the full dataset. On (a) we see the calibration curves for both Laplace and MAP (STN). On (b) and (c) we see the reliability diagrams for MAP and Laplace respectively

The plots in figure 4.18 show the calibration of both MAP STN and Laplace. We see that overall, Laplace improves the calibration of the model. Mainly, between the (confidence) values 0.4-0.8 the improvement is of larger scale than it is for lower or higher values. The reliability diagrams provide a more closer insight in calibration of MAP and Laplace. There we see the same that for the mid values confidence and accuracy have larger gap for the MAP model than the Bayesian one.

It has been shown already that Bayesian inference can capture uncertainties on the predictions. Figure 4.19 shows two examples where there is a difference on the predictive probabilities between the two models.

We see the Bayesian model to be more uncertain on its predictions compare to MAP model. The Laplace model even though classifies the images correctly seems to assign lower probabilities to the correct class and gives also scores to other classes as well. The MAP model on the other hand even in cases such as the bottom image in figure 4.19 where the target image is not very clear it gives a prediction with high degree of certainty.

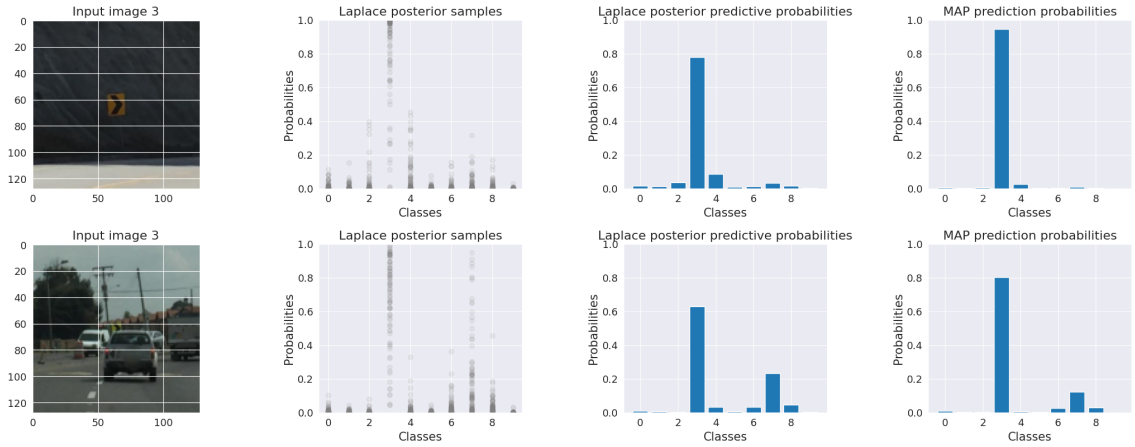


Figure 4.19: Visualization of uncertainties of the network for Mapiliary dataset

In cases where inputs mis-classified by the MAP model (see figure 4.20) we see similar results as the other two experiments. Even though Laplace mis-classifies the input samples too it does it with higher uncertainty compare to MAP.

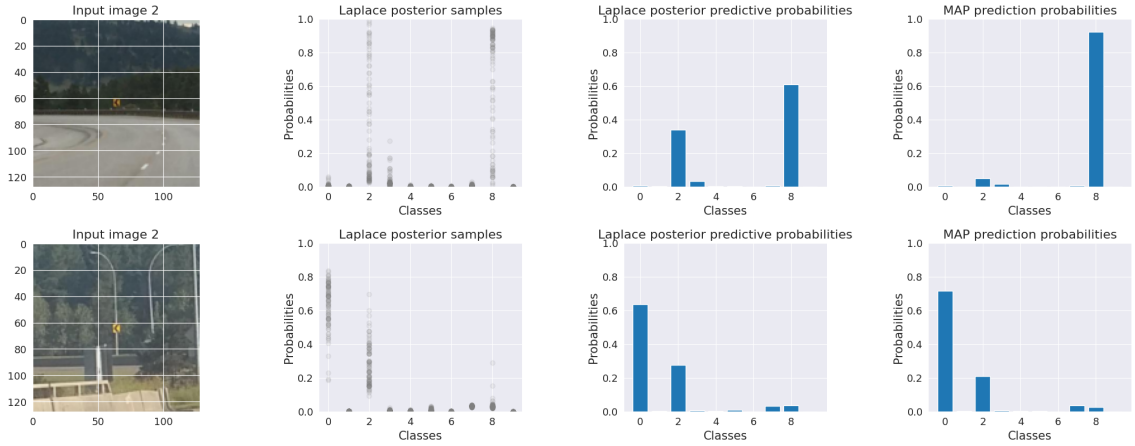


Figure 4.20: Visualization of uncertainties of the network for mis-classified data for Mapiliary dataset

Moving forward to the experiment we will use the properties of Bayesian inference to generate data from the STN module and use them to feed the classifier for continuous training. Figure 4.21 shows how transformations look like for few of the input images. We see a variety of transformations per input image. Though, in few cases the transformations have not resemble the input data successfully.

During the continuous training for this experiment the training time was significant longer compare to previous experiments. Major reasons it is believed that are due to the low amount of test data and the complexity of the data the model needed longer time to converge. More specifically for Mapiliary data the model run for 200 epochs more than the other two cases presented before.

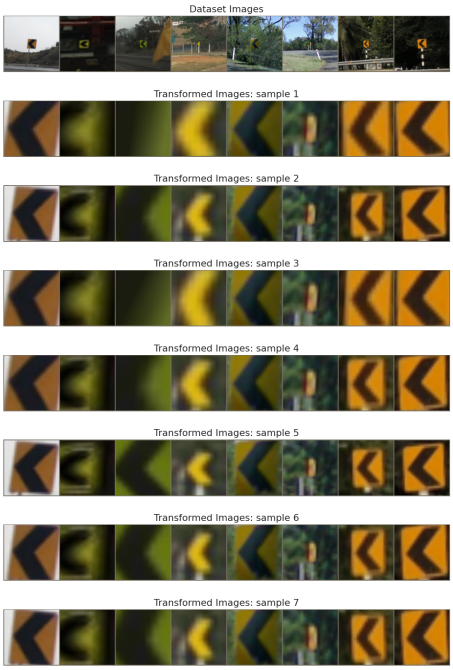


Figure 4.21: Results of subnetwork Laplace on second last layer of STN module for the Mapiliary data

On the table 4.7 we see that also in this case we have improvement on the test accuracy.

	Mapiliary
MAP	79.4% ±0.35
Laplace	**80.88%** ±1.31
Optimal prior precision	280.36

Table 4.7: Test accuracy scores for after applying post hoc training on the entire Mapiliary dataset.

Similarly to random placement MNIST continuous training applied on the whole training dataset and not to a subset of it. Thus, we see solely small increase in the accuracy. This observation also verifies the assumption that when we have large amount of data inference has minor impact on the performance of the model. The values for prior precision evaluated through marginal likelihood as well as CV grid search. Again both values yielded similar results. In this case, the amount of parameters was significantly larger thus the prior precision was larger too.

5 Results

Throughout the study we applied the Bayesian framework via Laplace on a spatial transformer network aiming to learn a successful data augmentation scheme to improve classifier's performance. We evaluated three datasets on the same tasks. For all the experiments we observed similar results. We saw that when the sample transformations used for continuous training of the classifier the performance increased in all cases. The performance significantly improved when smaller subsets of the main dataset used. However, as the data size increased the performance of the classifier during continuous training recorded small improvement. This observation verifies our assumption that data augmentation is mainly useful when the data size is small (see table 4.6). The findings about the improvement of the downstream task during continuous training align with findings by Schwöbel et al. [4] where they used amortised inference for the same purpose.

As discussed previously in section 2.3 deterministic neural networks suffer by poor calibration and overconfidence[40].

In all three experiments we observed that the MAP NN was overconfident. Even in cases where the input was not very clear MAP NN was very confident to its prediction (see figures 4.19, 4.3). We saw (for all three experiments) that when the MAP NN mis-classifies an input sample the BNN most likely will mis-classify as well but with much less confidence compare to MAP NN.

When testing on in-distribution shifted data the deterministic network was confident in all cases though sometimes the prediction was completely wrong. On the other hand the network with Laplace showed that it mis-classified the inputs yet it couldn't confidently give any prediction.

The results were outstanding when OOD data were considered. In that case we saw the MAP NN assign high probabilities to wrong classes while the BNN model recorded high degree of uncertainty in all cases. The fact that BNN can capture uncertainty is a major advantage over deterministic NN and is a especially important in professional areas such as healthcare. For example in automated health care, control should be passed on to human doctors when the confidence of a disease diagnosis network is low [40], [48]. For OOD experiment with MNIST and KMNIST data, we also saw the assumption that Laplace can capture better the uncertainties compare to amortised variational inference used by Schwöbel et al. [4] doesn't hold. Both techniques seemed to be equally uncertain. Though, more experiments required to ensure that this is the case on more complicated datasets as well.

Regarding calibration the experiments showed that the models had better calibration after inference. For the MNIST dataset great reduction of the ECE is observed for the small subsets. For MNIST 100 the model records ECE for Laplace of the magnitude 1.5% compare to 5% recorded on MAP NN (see figure 4.18). We see that the difference for both models for ECE is getting smaller as the data size increases. Though there is the case for MNIST 5000 where the ECE is higher for

Laplace compare to MAP NN. The NLL on the other hand is consistent across the range of the MNIST subsets. The difference between the models for that metric isn't significantly large. Though the behaviour is as expected meaning that NLL for Laplace should be lower than the one for MAP NN. Additionally, we saw that when we made predictions using shifted data the calibration of Laplace was constantly better than MAP NN.

For the random placement MNIST we saw that the ECE slightly decreased for the Laplace model though the NLL remained approximately the same. In this experiment the MAP NN model was calibrated well enough since it recorded ECE of 0.4. Thus, no much better calibration was expected.

For the Mapiliary dataset the calibration improved significantly due to Laplace. ECE dropped from 6.1% to 3.1% and NLL from 0.680 to 0.657. The improvement in calibration agrees with similar studies that have concluded the same outcome when Bayesian inference applied [41],[31],[34].

One of the major elements of the study that played vital role in the formulation of the results was the selection of subnetwork. The initial assumption to apply inference on the last layer of localiser rejected. Thus we had to perform further exploration to get the subnetwork that could generate proper transformation on the input images for the scope of the study. As discussed in section 3.2 to identify the subnetwork that represents the posterior of the full network is a challenging process. Across the experiments the structure of the model changed to be able to capture the correct subset of weights. For example for the random placement MNIST an additional forward linear layer added. The reason for it was to reduce the amount of parameters for computational reasons for the second last layer of the localiser (which was the chosen subnetwork). The same applied for Mapiliary dataset however the second last layer was significatly wider compare to random placemenent MNIST and MNIST. Specifically the parameters that we performed inference were approximatelly 5000, 4000 and 32000 for MNIST, Random MNIST and Mapiliary respectively. During experimentation we could observe that the performance of the model in terms of calibration and quality of sample transformations was changing depending on the amount of parameters that would be selected as well as the number of layers. Thus, this process was similar to a hyper parameter tuning until to get the right subset of weights.

Another major element of the Laplace approximation process was the implementation of marginal likelihood. We saw that for the determination of prior precision (crucial parameter for the performance of the Laplace model) grid research and marginal likelihood yielded the same result. Meaning that we can completely eliminate the computational inefficient grid search method and use directly the marginal likelihood for hyper-parameter tuning instead. Hence, we can perform the learned data augmentation process without manual adjustments. This is a major advantage over amortised variational inference used by Schwöbel et al. [4] for data augmentation which requires grid research each time to define the optimal KL-weights.

6 Discussion

Laplace approximation is an intuitive process to approximate intractable probability distributions. Laplace fits a Gaussian distribution on the mode of the posterior distribution. In practice however the posterior distribution will be multi modal thus there will be different Laplace approximation for each mode. Though, a mixture of Laplace approximations might be able to capture the multi-modality of the posterior [49]. Additionally, the posterior distribution it is expected to be close to Gaussian as the number of data or parameters increases. Thus Laplace will be most useful for large amount of data or large amount of parameters (as it is in our case).

Laplace approximation suffers from two major limitations. Firstly Laplace may fail to capture important global properties because it is based on the true distribution at a specific value [24]. Therefore, variational inference might be more appropriate approach because it adopts a more global perspective. Second is the computation and storage of the Hessian. We saw in section 3.1 that there are factorisation techniques that provide efficient computations yet they heavily depend on computational budget available. For example, in their experiment Daxberger et al. [34] claim that for subnetwork inference the maximum amount of weights that they can perform full approximation of the Hessian is approximately 40000. In our experiments this amount was significantly less approximately 6000. Hence it might not be straightforward to apply those approximations of the Hessian especially for large networks with complicated data. Also for some of the approximations such as the diagonal approximation there are contradictory perspectives in literature on whether should be used or not as it can affect the predictive performance of the model [50],[51].

The subnetwork API in Laplace Redux library used extensively in the study. The main idea behind subnetwork inference is that a subset of weights can preserve the uncertainty of a full network. This approach has two major benefits in our case. First it helps to reduce the required computational cost and second allows to apply inference on the layer of preference so we can draw samples and use them for data augmentation. However, this functionality comes with its limitations.

According to Daxberger et al. [34] there is not fixed way to sufficiently select the proper subnetwork. In their study they propose to use the Wassertein rule as the best subnetwork selection strategy. In our case though we explicitly define the subnetwork to be a small part of the localiser. Though the results showed that the selection was correct (for the purpose of the study) there was tuning involved to find the right number of parameters and the correct layer that gives the best results. For example the network architecture had to change several times for the challenging data such as random placement MNIST and Mapiliary. Also due to limited computational resources we couldn't select more layers as part of the subnetwork. Meaning that might have missed subset of weights that give better results.

Upon that more experimentation is needed to ensure whether our reasoning for the selection of the specific subnetwork provides indeed the best result. For example

we could experiment more with the Wassertein rule or try to modify the network architecture and include more layers.

When established the subnetwork we used the Bayesian properties to draw sample transformations from it which used to continue training the classifier. The results showed that there was improvement on the classifier performance for all the experiments. This, indicates that we have learnt a successful data augmentation scheme. The reasoning behind this approach was that we wanted the localiser not to learn new transformations (hence keeping its weights frozen) but to use inference to derive sample transformations. Then we wanted the classifier to keep updating its weights so it will learn to classify the samples derived from the localiser. On the other hand if we had performed inference in the classifier and kept updating the localiser then we wouldn't be able to generate sample transformations thus we couldn't perform any data augmentation.

In principal we could have applied inference in both classifier and localiser. However, we wanted to test the assumption that by applying inference on a specific part of the localiser is sufficient for data augmentation. Since the assumption proved valid it is not believed that the outcome would have changed if more layers from classifier were involved. Additionally, this would have added more computational complexity.

Lastly, we need to consider when is better to perform data augmentation during training or test time. In the study we performed data augmentation on test time where solely the calibration of the model improved and not the accuracy. Then we performed continual training which is closer to performing data augmentation on training time. Here we proved that the performance of the downstream task can improve. However, if we had performed data augmentation during training (at real training time) the model might be able to learn more transformations and generate more heterogeneous samples at each iteration which might lead to even better performance of the classifier. The Laplace library which used in this study allowed to prototype relatively quickly. However, since it is an under development library there were many limitations that didn't allow to do further exploration and create customised solutions for our case such as not able to perform data augmentation during training

Overall, applying Bayesian inference using Laplace on a Spatial Transformer Network we were able to create a successful data augmentation scheme. This scheme had better performance when applied on small datasets which was the main assumption of the study. Since data augmentation is necessary when data is scarce it was important to show that by applying Laplace on a small dataset we will have significant improvement on models performance. Also with Laplace approximation we were able to derive a model with better calibration in all experiments under study. Thus verifying that the Bayesian framework is better at capturing uncertainties than a deterministic model. Especially in out-of-distribution and shifted in-distribution data the uncertainty estimation of Laplace was dominant compared to MAP.

On last note, the application of marginal likelihood provided a robust, quick and easy way to estimate prior precision which significantly influenced the results. Therefore, this gives an advantage of Laplace over other Bayesian approximation methods such as the amortised variational inference used by Schwöbel et al. [4].

Bibliography

[1] C. Shorten and T.M Khoshgoftaar. *A survey on Image Data Augmentation for Deep Learning*. 2019. URL: https://doi.org/10.1186/s40537-019-0197-0.

[2] John T. Guibas, Tejpal S. Virdi, and Peter S. Li. "Synthetic Medical Images from Dual Generative Adversarial Networks". In: *CoRR* abs/1709.01872 (2017). arXiv: 1709.01872. URL: http://arxiv.org/abs/1709.01872.

[3] Aysegul Takimoglu. *What is Data Augmentation? Techniques Examples in 2022*. 2022. URL: https://research.aimultiple.com/data-augmentation/.

[4] Pola Schwöbel et al. "Probabilistic Spatial Transformers for Bayesian Data Augmentation". In: *CoRR* abs/2004.03637 (2020). arXiv: 2004.03637. URL: https://arxiv.org/abs/2004.03637.

[5] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. DOI: 10.48550/ARXIV.1406.2661. URL: https://arxiv.org/abs/1406.2661.

[6] Veit Sandfort et al. "Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks". In: *Scientific Reports* 9.1 (Nov. 2019), p. 16884. ISSN: 2045-2322. DOI: 10.1038/s41598-019-52737-x. URL: https://doi.org/10.1038/s41598-019-52737-x.

[7] Shobhita Sundaram and Neha Hulkund. *GAN-based Data Augmentation for Chest X-ray Classification*. 2021. DOI: 10.48550/ARXIV.2107.02970. URL: https://arxiv.org/abs/2107.02970.

[8] Saman Motamed, Patrik Rogalla, and Farzad Khalvati. "Data augmentation using Generative Adversarial Networks (GANs) for GAN-based detection of Pneumonia and COVID-19 in chest X-ray images". In: *Informatics in Medicine Unlocked* 27 (2021), p. 100779. ISSN: 2352-9148. DOI: https://doi.org/10.1016/j.imu.2021.100779. URL: https://www.sciencedirect.com/science/article/pii/S2352914821002501.

[9] Ekin Dogus Cubuk et al. "AutoAugment: Learning Augmentation Policies from Data". In: *CoRR* abs/1805.09501 (2018). arXiv: 1805.09501. URL: http://arxiv.org/abs/1805.09501.

[10] Ekin D. Cubuk et al. "RandAugment: Practical data augmentation with no separate search". In: *CoRR* abs/1909.13719 (2019). arXiv: 1909.13719. URL: http://arxiv.org/abs/1909.13719.

[11] Max Jaderberg et al. "Spatial Transformer Networks". In: *CoRR* abs/1506.02025 (2015). arXiv: 1506.02025. URL: http://arxiv.org/abs/1506.02025.

[12] David J. C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge: Cambridge University Press, 2003. ISBN: 0521642981 9780521642989. URL: https://www.worldcat.org/title/information-theory-inference-and-learning-algorithms/oclc/856654329?referer=br&ht=edition.

[13] Georgia Gkioxari, Ross B. Girshick, and Jitendra Malik. "Contextual Action Recognition with R*CNN". In: *CoRR* abs/1505.01197 (2015). arXiv: 1505.01197. URL: http://arxiv.org/abs/1505.01197.

[14] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A Unified Embedding for Face Recognition and Clustering". In: *CoRR* abs/1503.03832 (2015). arXiv: 1503.03832. URL: http://arxiv.org/abs/1503.03832.

[15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *CoRR* abs/1411.4038 (2014). arXiv: 1411.4038. URL: http://arxiv.org/abs/1411.4038.

[16] Christian Szegedy et al. "Going Deeper with Convolutions". In: *CoRR* abs/1409.4842 (2014). arXiv: 1409.4842. URL: http://arxiv.org/abs/1409.4842.

[17] Kevin Zakka. *Deep Learning Paper Implementations: Spatial Transformer Networks - Part II*. https://kevinzakka.github.io/2017/01/18/stn-part2/. Assessed: 2022-07-05.

[18] Ghassen Hamrouni. *Spatial Transformer Network Tutorial*. https://pytorch.org/tutorials/intermediate/spatial_transformer_tutorial.html. Assessed: 2022-07-05.

[19] Thomas Kurbiel. *Spatial Transformer Networks*. https://towardsdatascience.com/spatial-transformer-networks-b743c0d112be. Assessed: 2022-07-05.

[20] Arthur Coste. "Image Processing : Affine Transformation, Landmarks registration, Non linear Warping". In: (Oct. 2012). DOI: 10.13140/RG.2.2.13653.27360.

[21] *Linear mapping method using affine transformation*. URL: https://www.mathworks.com/discovery/affine-transformation.

[22] *Digital image processing*. URL: https://en.wikipedia.org/wiki/Digital_image_processing.

[23] D.J. Eck. *Introduction to Computer Graphics* \. Open textbook library. David J. Eck, 2016. URL: https://books.google.dk/books?id=zc2AzQEACAAJ.

[24] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. URL: /bib/bishop/Bishop2006/Pattern-Recognition-and-Machine-Learning-Christophe-M-Bishop.pdf,/bib/bishop/Bishop2006/978-0-387-31073-2_sm.pdf, https://www.microsoft.com/en-us/research/people/cmbishop/#!prml-book.

[25] Buu Truong Phan. "Bayesian Deep Learning and Uncertainty in Computer Vision". Masters. Waterloo: University of Waterloo, Sept. 2019. URL: http://hdl.handle.net/10012/15056.

[26] Hippolyt Ritter, Aleksandar Botev, and David Barber. "A Scalable Laplace Approximation for Neural Networks". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=Skdvd2xAZ.

[27] Laurent Valentin Jospin et al. "Hands-on Bayesian Neural Networks - a Tutorial for Deep Learning Users". In: *CoRR* abs/2007.06823 (2020). arXiv: 2007.06823. URL: https://arxiv.org/abs/2007.06823.

[28] Yarin Gal and Zoubin Ghahramani. *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*. 2015. DOI: 10.48550/ARXIV.1506.02142. URL: https://arxiv.org/abs/1506.02142.

[29] Charles Blundell et al. "Weight uncertainty in neural network". In: *International conference on machine learning*. PMLR. 2015, pp. 1613–1622.

[30] James Brennan. *The Laplace Approximation*. https://james-brennan.github.io/posts/laplace_approximation/. Assessed: 2022-07-05.

[31] Erik Daxberger et al. "Laplace Redux - Effortless Bayesian Deep Learning". In: *CoRR* abs/2106.14806 (2021). arXiv: 2106.14806. URL: https://arxiv.org/abs/2106.14806.

[32] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2013. DOI: 10.48550/ARXIV.1312.6114. URL: https://arxiv.org/abs/1312.6114.

[33] Chris Cremer, Xuechen Li, and David Duvenaud. "Inference Suboptimality in Variational Autoencoders". In: *CoRR* abs/1801.03558 (2018). arXiv: 1801.03558. URL: http://arxiv.org/abs/1801.03558.

[34] Erik A. Daxberger et al. "Expressive yet Tractable Bayesian Deep Learning via Subnetwork Inference". In: *CoRR* abs/2010.14689 (2020). arXiv: 2010.14689. URL: https://arxiv.org/abs/2010.14689.

[35] Pavel Izmailov et al. "Subspace Inference for Bayesian Deep Learning". In: *CoRR* abs/1907.07504 (2019). arXiv: 1907.07504. URL: http://arxiv.org/abs/1907.07504.

[36] Jonathan Frankle and Michael Carbin. "The Lottery Ticket Hypothesis: Training Pruned Neural Networks". In: *CoRR* abs/1803.03635 (2018). arXiv: 1803.03635. URL: http://arxiv.org/abs/1803.03635.

[37] Wesley J. Maddox, Gregory W. Benton, and Andrew Gordon Wilson. "Rethinking Parameter Counting in Deep Models: Effective Dimensionality Revisited". In: *CoRR* abs/2003.02139 (2020). arXiv: 2003.02139. URL: https://arxiv.org/abs/2003.02139.

[38] Mahdi Pakdaman Naeini, Gregory F Cooper, and Milos Hauskrecht. "Obtaining Well Calibrated Probabilities Using Bayesian Binning." In: *AAAI*. 2015, 2901–2907.

[39] Jeremy Nixon et al. "Measuring Calibration in Deep Learning". In: *CoRR* abs/1904.01685 (2019). arXiv: 1904.01685. URL: http://arxiv.org/abs/1904.01685.

[40] Chuan Guo et al. "On Calibration of Modern Neural Networks". In: *CoRR* abs/1706.04599 (2017). arXiv: 1706.04599. URL: http://arxiv.org/abs/1706.04599.

[41] Alexander Immer, Maciej Korzepa, and Matthias Bauer. *Improving predictions of Bayesian neural nets via local linearization*. 2020. DOI: 10.48550/ARXIV.2008.08400. URL: https://arxiv.org/abs/2008.08400.

[42] Juozas Vaicenavicius et al. "Evaluating model calibration in classification". In: *CoRR* abs/1902.06977 (2019). arXiv: 1902.06977. URL: http://arxiv.org/abs/1902.06977.

[43] Seonguk Seo, Paul Hongsuck Seo, and Bohyung Han. "Confidence Calibration in Deep Neural Networks through Stochastic Inferences". In: *CoRR* abs/1809.10877 (2018). arXiv: 1809.10877. URL: http://arxiv.org/abs/1809.10877.

[44] Nicolas Posocco and Antoine Bonnefoy. "Estimating Expected Calibration Errors". In: *CoRR* abs/2109.03480 (2021). arXiv: 2109.03480. URL: https://arxiv.org/abs/2109.03480.

[45] Christian Ertler et al. "Traffic Sign Detection and Classification around the World". In: *CoRR* abs/1909.04422 (2019). arXiv: 1909.04422. URL: http://arxiv.org/abs/1909.04422.

[46] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[47] Tarin Clanuwat et al. "Deep Learning for Classical Japanese Literature". In: *CoRR* abs/1812.01718 (2018). arXiv: 1812.01718. URL: http://arxiv.org/abs/1812.01718.

[48] Xiaoqian Jiang et al. “Calibrating predictive model estimates to support personalized medicine”. en. In: *J Am Med Inform Assoc* 19.2 (Oct. 2011), pp. 263–274.

[49] Runa Eschenhagen et al. “Mixtures of Laplace Approximations for Improved Post-Hoc Uncertainty in Deep Learning”. In: *CoRR* abs/2111.03577 (2021). arXiv: 2111.03577. URL: https://arxiv.org/abs/2111.03577.

[50] David J. C. MacKay. “A Practical Bayesian Framework for Backpropagation Networks.” In: *Neural Comput.* 4.3 (1992), pp. 448–472. URL: http://dblp.uni-trier.de/db/journals/neco/neco4.html#MacKay92a.

[51] Sebastian Farquhar, Lewis Smith, and Yarin Gal. “Try Depth Instead of Weight Correlations: Mean-field is a Less Restrictive Assumption for Deeper Networks”. In: *CoRR* abs/2002.03704 (2020). arXiv: 2002.03704. URL: https://arxiv.org/abs/2002.03704.

A Sanity Checks

A.0.1 MNIST

First we evaluated the Last Layer API and Subnetwork API on a basic classifier without the STN module. We applied the Subnetwork API on the last layer of the classifier to ensure that it gives the same results as the Last Layer API (this API applies on the last layer of the neural network by default).

For the evaluation we observed the distribution of samples for small and large prior precision values. The key assumption is that when we have small prior precision we expect to have more variance and large calibration error while when the prior precision is large it is expected the outcome to match the MAP solution. In figure A.1 we see the distribution of samples for both APIs.

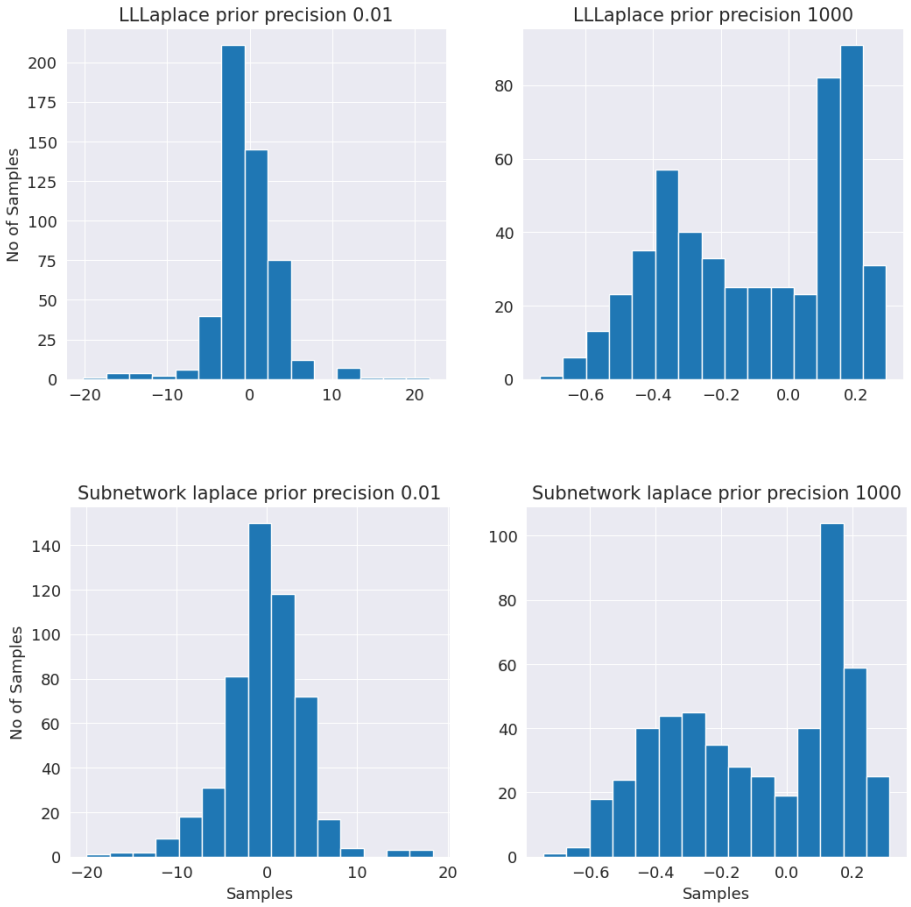


Figure A.1: On the top row the results of Last Layer Laplace. On the bottom the results of Subnetwork Laplace using the Subnetwork functionality

We see that both APIs gives approximately the same distributions and perform as intended. For small prior precision (eg. 0.01. See figures on the left) there is higher variance while the variance decreases as we increase the prior precision. Also the

metrics indicate that with high prior precision the Laplace model behaves as the MAP due to the flat prior. The metric values for comparison are shown in table A.1 below.

	MAP	LLLaplace Small Prior	LLLaplace Large Prior	Sub Laplace Small Prior	Sub Laplace Large Prior
Acc	99.2 %	99.2 %	99.2 %	99.2 %	99.2 %
ECE	0.5 %	82.3 %	0.5 %	83.4 %	0.5 %
NLL	0.034	1.78	0.0339	1.85	0.0338

Table A.1: Metric values for small/large value precision for Last Layer Laplace (LLLaplace) and Subnetwork Laplace (Sub Laplace)

For the **second sanity check** we applied the subnetwork functionality on the last layer of the localization network of the STN module. The results are shown in figure A.2 below.

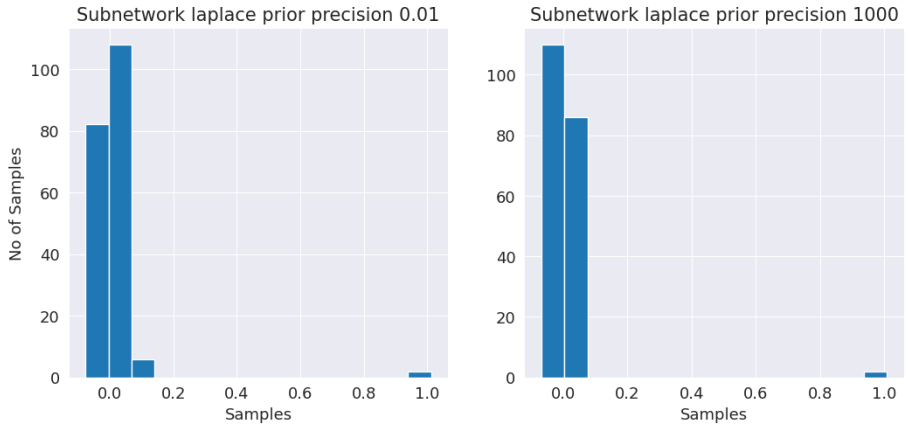


Figure A.2: Applied Laplace APIs on the P-STN. Results of subnetwork Laplace using the Subnetwork functionality on the last layer of the localization network

When apply the sub-network API on the last layer of the STN module we observe an abnormal behaviour of the Laplace approximation. We see that the prior precision doesn't affect the performance of the model and model performs identical to MAP in both cases. Table A.2 shows the metric values of MAP solution (before inferece) and Subnetwork STN (applied on the last layer of the localisation).

	MAP (STN)	Subnetwork STN
		Small/Large prior precision
Acc	99.4%	99.4%
ECE	0.3%	0.2%
NLL	0.0202	0.0195

Table A.2: Acc, ECE and NLL values for MAP (STN), Last Layer STN Subnetwork.

The reason for such behaviour might be that the amount of parameters on last layer of STN is too low (eg. between 68-200 depending the transformation parameters defined). Thus, the approximate posterior of the chosen subnetwork doesn't minimise the exact posterior of the full network [31]. Another potential reason could be the initialisation of zero-weights we impose on the last layer of STN. Though, by changing the initialisation to non-zero weights we see the distribution of weights changes but the impact of prior precision seems the same as before for this case by checking on the metric values as well as the ´spread of samples (see figure A.3).

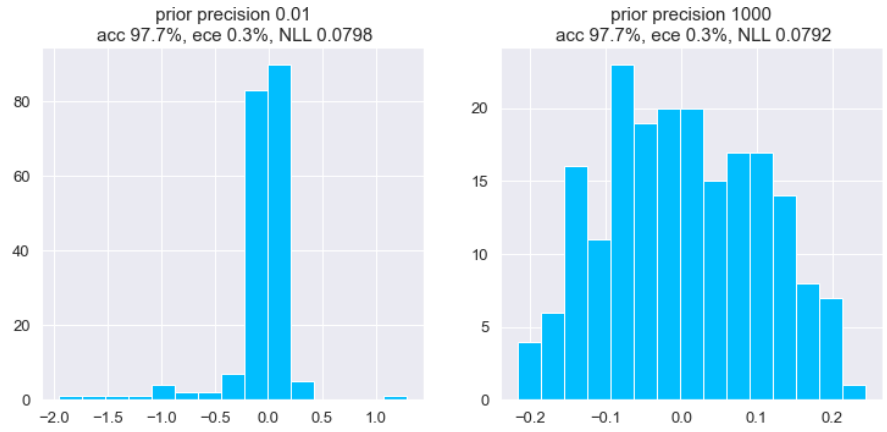


Figure A.3: Results of subnetwork Laplace on last layer of STN with non-zero initialization

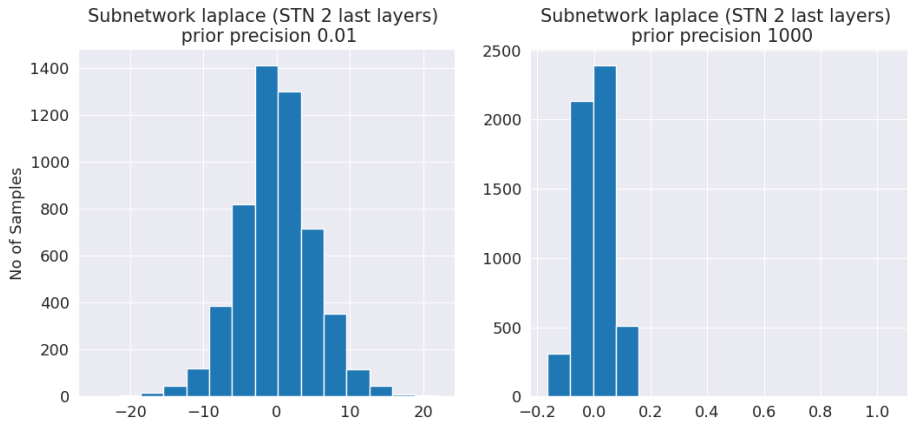


Figure A.4: Visualization of Subnetwork inference on two last layers

We see that by including the second last layer (A.4) the outcome of subnetwork inference performs as intended. Indeed we have higher variance for small prior precision and also higher ECE (see table A.3). The distribution when the large prior precision applied looks more concentrated around zero due to initialisation of weights (on 0) and biases (on 1) on the last layer. Though the metric values verifies our assumption as shown in table A.3.

	MAP (STN)	Subnetwork STN 2 Last Layers	
		Small prior prec.	Large prior prec.
Acc	99.4%	98.8%	99.4%
ECE	0.3%	63.2%	0.2%
NLL	0.0202	1.14	0.0197

Table A.3: Acc, ECE and NLL values for MAP (STN), 2 Last Layers STN Subnetwork.

A.0.2 Random Placement MNIST

The plot A.5 show how the weights are distributed when we change the prior precision values for random placement MNIST. The outcome is the same as in the MNIST which also verifies that the selected subnetwork (second last layer of localisation network) work as intended. The ECE might be higher on the left plot compare to the right one but still relatively small compare to the corresponding plot for MNIST dataset. This is due to the fact that larger value used (eg. 0.1) as small prior precision compare to MNIST. For smaller values the Cholesky factorization was not positive semi definite and therefore couldn't proceed with the computation.

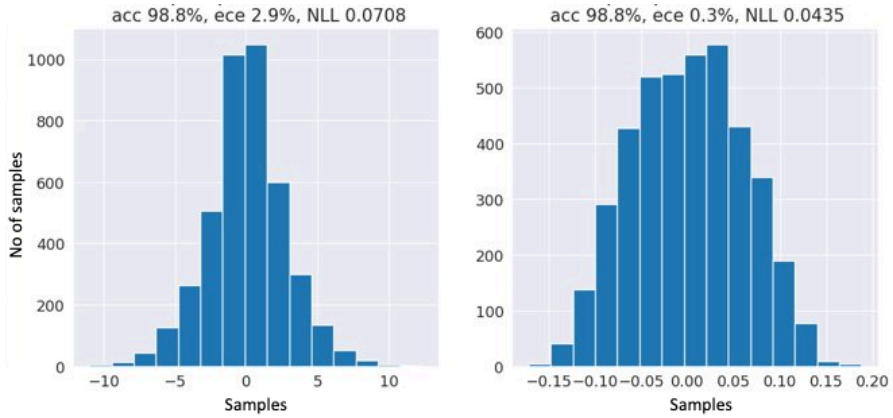


Figure A.5: Results of subnetwork Laplace on second last layer of STN module