DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Beyond Deep Learning: Selected Topics on Novel Challenges
Seminar Report

Uncertainty via Ensemble Learning

İsmail Pamir 03724650

Contents

1 Introduction

Machine learning models have been moved to very advanced levels in terms of accuracy with the machine learning studies in recent years. Nevertheless, they can easily be fooled by intentionally derived samples or some natural samples. The lack of representing uncertainty in machine learning models raises this problem, and it also causes overconfident and miscalibrated predictions [1]. Although these problems do not seem to be very important because they are rarely encountered, it can cause very bad results in some cases. For instance, this may lead to death in autonomous driving [2, 3]. Misdiagnosis may cause very costly consequences in terms of human life and finance in medical applications [2, 4], and mispredictions may lead to huge financial losses for companies in trading and finance [2, 5].

One of the most common ways to solve this kind of uncertainty problem is to use a Bayesian approach. Because Bayesian approaches introduce a natural probabilistic way to estimate uncertainty in deep learning [1]. Bayesian approaches, that define a distribution on model parameters rather than fixed model parameters, aim to form posterior predictive distribution by marginalizing the distribution of $\tilde{x}$ given model paramaters θ over the posterior distribution of θ given observed data X. You can see the whole formula in Equation 1.1.

$$P(\tilde{x} \mid \mathbf{X}) = \int_{\Theta} p(\tilde{x} \mid \theta, \mathbf{X}) p(\theta \mid \mathbf{X}) \mathrm{d}\theta \tag{1.1}$$

where

$$P(\theta \mid X) = \frac{P(X \mid \theta) P(\theta)}{P(X)} = \frac{P(X \mid \theta) P(\theta)}{\int_{\theta} P\left(X \mid \theta'\right) P\left(\theta'\right) d\theta'} = \frac{P(X, \theta)}{\int_{\theta} P\left(X, \theta'\right) d\theta'} \tag{1.2}$$

Posterior predictive distribution takes uncertainty about θ into account that is why it is desired to take advantage of it while estimating uncertainty about the decisions of the neural networks. However, it has also some serious drawbacks. The neural networks implemented with the Bayesian approach need a lot of modifications compared to standard neural networks [6]. Additionally, they are highly sensitive to hyperparameter choice and initialization of weight [1]. Even different initialization of weight can lead to convergence to different local minima. As a result of this, researchers have started to investigate a simpler and scalable way for uncertainty estimation without Bayesian inference. Nowadays, there are a lot of ways to estimate uncertainty in deep learning models such as ensemble learning based methods, post-processing methods [7] and test time data augmentation [8]. In this report, ensemble learning based methods will be investigated and they will be compared to a

classical approach which is Bayesian model averaging, and dropout-based methods which can be also counted as ensemble learning.

2 Ensemble Learning

Ensemble learning is an approach to combining individually trained several models in different ways to form a single model in order to make a decision collectively. It aims to make the model more accurate and decrease the likelihood of rare cases in the model. The main motivation of ensemble learning is that combining many simple models can lead to finding a solution to a complex problem.

Ensemble methods commonly have two parts. The first part has multiple models that make a prediction about output with a given data and the second part has one model or approach which aggregates the outputs of the several decision-maker models. Ensemble learning methods differ from each other in terms of the learning procedure and the aggregation method. The most common methods are bagging, boosting, and stacking. In some sense, also dropout can be classified as a kind of ensemble learning.

Bagging is also called bootstrap aggregating. It has two parts which are bootstrapping and aggregating. Bootstrapping provides that each classifier is trained by randomly sampled training data from a fixed set. Training data are sampled with replacement and this allows to use of the same data for different classifier training. Each classifier focuses on a specific aspect of the probability distribution thanks to the bootstrapping procedure. Aggregating combines weak learners which are obtained by training after bootstrapping to provide a more advanced combined classifier.

Unlike the parallel structure of the bagging procedure, the boosting procedure consists of a series of classifiers. Training samples for the classifier are chosen based on the previous classifier performance. It is determined in which data is predicted wrong before that classifier and the next classifier focuses more on these samples for training. Thus, boosting tries to find a new classifier that makes a better prediction on the samples which is not covered well by previous classifiers [9].

There are two features that distinguish stacking from bagging and boosting. First, stacking uses different learning algorithms to obtain model rather than combining models which is obtained using same algorithm. And second, stacking uses a meta model to combine weak learners instead of using deterministic functions. In other words, it tries to achieve better results by using the power of machine learning in the ensemble part. Graczyk et al. showed that stacking algorithms give better accuracy in most cases compared to bagging and boosting, on the other hand it is the most unstable one that can cause very low accuracy in some cases

compared to other methods [10].

In the simplest sense, dropout is to deactivate some nodes randomly using Bernoulli distribution during the training phase. Although it is generally used as an overfitting prevention mechanism in the training phase, it can also be used in uncertainty estimation by enabling it at test time. It can be used as a Bayesian approximation in neural networks [11]. Some of the researchers claim dropout is a kind of ensemble learning. For example, Hara, Saitoh, and Shouno reported that they regarded dropout learning as ensemble learning although it differentiates in some aspects from ensemble learning. The researchers said that dropout learning is performed with a different set of hidden units of the same model despite the fact that the original ensemble learning models use an aggregation over the fixed set of hidden units but with many models or architectures. In terms of aggregating results of different models, they indicated that both methods use the same methodology [12].

2.1 Why Should We Consider to Use Ensemble Learning?

There are many aspects where the use of ensemble learning can be advantageous. It can be said that ensemble learning is easier to implement and understand. It also reduces complexity of machine learning and provides more accurate models. In some case, the parallel training is possible as well. However, the most relevant advantage for this report is that ensemble learning allows uncertainty representation and estimation taking advantage of Monte Carlo sampling.

3 Uncertainty

Generally, classification models are forced to yield results within a predetermined set for each input. Since it is usual to make point estimates for parameters in machine learning, there is not a metric to measure whether the answers given by the model for a particular input are random answers or logical answers [13]. However, as the studies in the field of machine learning increase, the necessity of the capability of machine learning models to give the answer "I do not know" has started to emerge for some specific implementation areas. For example, the confidence level of the machine learning model that diagnoses cancer is very important. In this case, the lack of representation of confidence may cause life-changing consequences [13]. There are many aspects of machine learning that can create uncertainty in predictions [13]. For example, these are noisy data that can lead to the uncertainty of estimation, model parameter uncertainty that represents the uncertainty of the answer of the question of which parameters should be chosen to predict well [13], and structure uncertainty that represents the uncertainty of the answer of the question of which model should be used to interpolate/extrapolate well [13].

There are two types of uncertainty which are epistemic uncertainty and aleatoric uncertainty. The epistemic uncertainty is concerned with the uncertainty in the model's parameters and this is because there is not enough information about data to have good predictions [14]. Aleatoric uncertainty is concerned with the noise inherent in the observations [14]. So it can be said that epistemic uncertainty can be associated with model uncertainty and aleatoric uncertainty can be associated with the data uncertainty.

Yarin Gal noted that epistemic uncertainty can be reduced with an increasing number of data points [13]. Unlike the epistemic uncertainty, Yarin Gal also showed that aleatoric uncertainty can not be decreased by increasing the number of data points because the increasing number of data points does not lead to decrease the noise inherent of the data [13].

3.1 Measuring Quality of the Predictive Uncertainty

The existence of all these uncertainty concerns which are explained above arises the necessity to obtain predictive uncertainty. Models should not only make predictions about incoming input but also tell the confidence level of this prediction. However, the problem is not only to make models able to give us a predictive uncertainty but also to evaluate of quality of the predictive uncertainty. Because measuring the quality of predictive uncertainties has always been a difficult task since there is not any ground truth information on uncertainty [6]. Lakshminarayanan, Pritzel, and Blundell proposed to use some evaluation measures that are obtained by considering practical applications of neural networks. They measured calibration by proper scoring rules which are log predictive probabilities and the Brier score. Furthermore, they visualized it using reliability diagrams that are a good way to show underconfident or overconfident predictions. And also, they noted that the network should have high predictive uncertainty when inferring on a dataset where it is not trained. That's why researchers tried to measure "if the network knows what it knows" with out of distribution examples [6].

4 Uncertainty via Ensemble Learning

The model M* which is obtained for a specific problem may seem that it fits the data sensibly well and the parameter estimation is quite rational. It can be quite convenient to use M* to predict results in standard statistical practice [15]. However, you can imagine that there is another model M** that also provides reasonable estimates but it provides different predictions for the same cases [15]. In such a case, it would be risky if the inference is made based on only one model. Ensemble learning algorithms can be used as a solution to capture the uncertainty due to model selection in such a case. In this chapter, some ensemble methods to estimate uncertainty will be explained, and these methods will be compared to each other

in terms of accuracy, computational power, and memory usage.

4.1 Bayesian Deep Learning and Bayesian Model Averaging

In the traditional deep learning methods, a single solution is obtained, which optimizes the error in the training phase, for parameters. It causes the loss of the notion of uncertainty in the deep neural networks. As it is mentioned in Chapter 1, Bayesian approaches introduce a natural probabilistic way to estimate uncertainty in deep learning [1]. It defines a distribution on model parameters rather than fixed model parameters. Thereby, it captures the uncertainty arising from the data or the model.

Bayesian model averaging is one of the most traditional ways to represent uncertainty in the ensemble learning sense based on Bayesian deep learning. Assume that the quantity that is tried to inference is Δ. So marginalization can be used and it can be said that its posterior distribution given data D is

$$\text{pr}(\Delta \mid D) = \sum_{k=1}^{K} \text{pr}\left(\Delta \mid M_k, D\right) \text{pr}\left(M_k \mid D\right) \tag{4.1}$$

This is the weighted average of the posterior distributions. They have weights which represents their posterior model probability where $M_1, \ldots, M_K$ are the models considered. And the posterior probability for model M_K can be written as follows

$$\text{pr}\left(M_k \mid D\right) = \frac{\text{pr}\left(D \mid M_k\right) \text{pr}\left(M_k\right)}{\sum_{l=1}^{K} \text{pr}\left(D \mid M_l\right) \text{pr}\left(M_l\right)} \tag{4.2}$$

where

$$\text{pr}\left(D \mid M_k\right) = \int \text{pr}\left(D \mid \theta_k, M_k\right) \text{pr}\left(\theta_k \mid M_k\right) d\theta_k \tag{4.3}$$

is the marginalized likelihood of model M_k respect to θ_k where θ_k is the model parameters of M_k. Hoeting et al. noted that Bayesian model averaging performs more accurate predictions compared to single model [15]. Furthermore, Dong, Xiong, and Yu showed that Bayesian model averaging will decrease uncertainty in the prediction besides the accuracy compared to a single model in the domain of hydrological models [16]. BMA gives us a lot of advantageous aspects compared to the single-model selection, especially when the model uncertainty is involved in [17]. For instance, BMA tries to prevent the overconfidence that emerges due to ignorance of model uncertainty [17] and it tries to avoid that model is forced to make a prediction in the all-or-nothing scheme that is used for classical deep learning methods [17]. Besides the several advantages, BMA has some disadvantages in terms of implementation. The number of terms in the posterior distribution which is marginalized can be enormous and it can not be tractable in some cases, and also, integrals in Equation 4.3 can be hard to compute [15]. And even, determining of a prior distribution over the models, which is pr(M_k), is also a very hard problem to overcome [15]. BMA supposes that true predictive distribution is the same kind of distribution as the hypothesis class of the prior and it tries to find a soft

model by averaging models and obtains a single model within the hypothesis class [6]. On the contrary, ensemble methods combine different models to have a powerful model and it is expected that ensemble methods work better when predictive distribution is out of the hypothesis class [6]. It is why simple and scalable methods grabbing attention in the area of uncertainty estimation. And this leads us to ensemble learning without Bayesian approaches.

4.2 Dropout-Based Methods for Uncertainty Estimation

Although dropout is commonly used as a mechanism for preventing over-fitting in many models in deep learning, Gal and Ghahramani showed that dropout usage in deep learning can be also explicated as a Bayesian approximation of the Deep Gaussian process [11]. With this justification, a basic method is obtained to have predictive uncertainty using dropout. If the dropout is used not only in the training phase but also in the testing phase, it can quantify the neural network uncertainty. In this chapter, some dropout-based methods will be explained and how they can help us to attain a predictive uncertainty will be showed.

4.2.1 MC Dropout

In the normal dropout, the dropout mechanism is applied only during training time. This situation causes the deterministic results for inputs in test time. But Gal and Ghahramani proposed a framework for applying dropout during both training and test time so that in test time, predictions are not deterministic anymore [11]. With this mechanism, they aimed to achieve some information about approximate predictive distribution which is given by

$$q\left(\mathbf{y}^* \mid \mathbf{x}^*\right) = \int p\left(\mathbf{y}^* \mid \mathbf{x}^*, \boldsymbol{\omega}\right) q(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} \tag{4.4}$$

where $\boldsymbol{\omega} = \{\mathbf{W}_i\}_{i=1}^{L}$ is set of random variables for a model with L layers and $q(\boldsymbol{\omega})$ is

$$\begin{aligned} &\mathbf{W}_i = \mathbf{M}_i \cdot \operatorname{diag}\left(\left[\mathbf{z}_{i,j}\right]_{j=1}^{K_i}\right) \\ &\mathbf{z}_{i,j} \sim \text{ Bernoulli } \left(p_i\right) \text{ for } i = 1, \ldots, L, j = 1, \ldots, K_{i-1} \end{aligned} \tag{4.5}$$

and matrices $\mathbf{M}_i$ as variational parameters. They proposed that to use moment-matching to obtain the first and second moments of the predictive distribution. Because the prediction of the input can be determined by calculating the first moment of outputs and predictive uncertainty can be achieved by calculating the second moment of outputs. High variance indicates the high uncertainty of the output and vice versa. They sampled T set of vectors from the Bernoulli distribution $\left\{\mathbf{z}_1^t, \ldots, \mathbf{z}_L^t\right\}_{t=1}^{T}$ with $\mathbf{z}_i^t = \left[\mathbf{z}_{i,j}^t\right]_{j=1}^{K_i}$, giving $\left\{\mathbf{W}_1^t, \ldots, \mathbf{W}_L^t\right\}_{t=1}^{T}$ and tried the estimate first moment of predictive distribution which is

$$\mathbb{E}_{q(\mathbf{y}^* \mid \mathbf{x}^*)}\left(\mathbf{y}^*\right) \approx \frac{1}{T} \sum_{t=1}^{T} \widehat{\mathbf{y}}^*\left(\mathbf{x}^*, \mathbf{W}_1^t, \ldots, \mathbf{W}_L^t\right) \tag{4.6}$$

They called it MC-Dropout because it has many common points with Monte Carlo sampling, it does the same thing in a the different way. It is also some kind of ensemble learning because it takes different vectors to obtain different networks and just aggregates their results. And it also obtains the second moment in the same fashion [11].

Table 4.1: Average test performances in RMSE and predictive log likelihood [11]

Datasets	Avg. Test RMSE and Std. Errors			Avg. Test LL and Std. Errors		
	VI	PBP	Dropout	VI	PBP	D
Boston Housing	4.32 ±0.29	3.01 ±0.18	2.97 ±0.19	-2.90 ±0.07	-2.57 ±0.09	-2.46 ±0.06
Concrete Strength	7.19 ±0.12	5.67 ±0.09	5.23 ±0.12	-3.39 ±0.02	-3.16 ±0.02	-3.04 ±0.02
Energy Efficiency	2.65 ±0.08	1.80 ±0.05	1.66 ±0.04	-2.39 ±0.03	-2.04 ±0.02	-1.99 ±0.02
Kin8nm	0.10 ±0.00	0.10 ±0.00	0.10 ±0.00	0.90 ±0.01	0.90 ±0.01	0.95 ±0.01

In Table 4.1, VI stands for variational inference and PBP stands for probabilistic back-propagation. In this table, you can see the comparison of the uncertainty quality of the methods in terms of the average log likelihood. Researchers noted that uncertainty quality can be determined from the predictive log-likelihood because it shows how well the model captures the data. As you can see in Table 4.1, dropout methods outperform Bayesian approaches [11].

4.2.2 Concrete Dropout

In the MC dropout method, dropout probability is a hyperparameter and it is necessary to have a good parameter to achieve well-calibrated uncertainty estimates. Therefore, fine-tuning should be done for dropout parameter. Gal, Hron an Kendall proposed a new dropout alternative which is concrete(CONtinuos Relaxations of disCRETE random variables) dropout and it achieves better performance and better calibrated models. They noted that using a continuous relaxation for dropouts discrete probability brings us to the ability to make automatic tuning of dropout probability in large models [18]. Continuous relaxation means categorical reparameterization trick and it solves the problem that sampling from a categorical distribution is not differentiable with respect to dropout parameter. This method proposes that the random variable z, which is dropout random variable, can be sampled with a reparametrization trick, which is explained below, instead of sampling directly from a discrete distribution. Let say g is a random variable from Gumbel distribution

$$g = -\log(-\log(u)) \quad \text{with} \quad u \sim \text{Uniform}[0,1] \tag{4.7}$$

then the discrete random variable can be sampled with function

$$z = \arg\max_k [g_k + \log \boldsymbol{\pi}_k] \tag{4.8}$$

But there is still a problem. Argmax function is still not differentiable with respect to z. It can be said that argmax function can be approximated with softmax and temperature parameter

τ.

$$\tilde{\mathbf{z}} = \sigma\left(\frac{\log \pi + g}{\tau}\right) \tag{4.9}$$

They proposed the sampling from the Concrete distribution with some temperature and this sampling results in a range of [0, 1] instead of sampling from the discrete Bernoulli distribution which generates 0 or 1. This provides multiplicative noises to add to weights [18]. In binary case, g can be calculated as log(u) and Equation 4.9 becomes a form such as

$$\tilde{\mathbf{z}} = \text{sigmoid}\left(\frac{1}{\tau} \cdot (\log p - \log(1-p) + \log u - \log(1-u))\right) \tag{4.10}$$

where $u \sim \text{Unif}(0, 1)$.

Hereby, the gradient of the objective function can be calculated with respect to $\tilde{\mathbf{z}}$ and a proper dropout probability can be found to obtain a better result and calibration. Gal, Hron, and Kendall compared concrete dropout with MC dropout using an image segmentation example. Results showed that concrete dropout outperforms MC dropout in terms of accuracy and calibration. But they also reported that it requires more computational power compared to MC dropout. You can see the accuracy result in Table 4.2 and calibration result in Figure 4.1 [18].

Table 4.2: Intersection over union accuracy of the methods with MC sampling and without MC sampling [18]

DenseNet Model Variant	MC Sampling	IoU
No Dropout	-	65.8
Dropout (manually-tuned p = 0.2)	NO	67.1
Dropout (manually-tuned p = 0.2)	YES	67.2
Concrete Dropout	NO	67.2
Concrete Dropout	YES	67.4

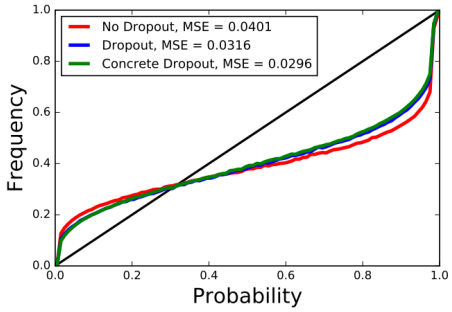
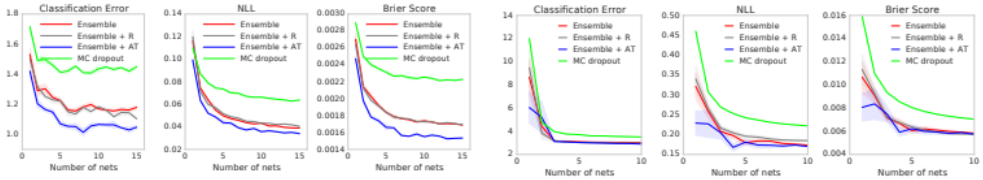


Figure 4.1: Reliability diagrams for methods [18]

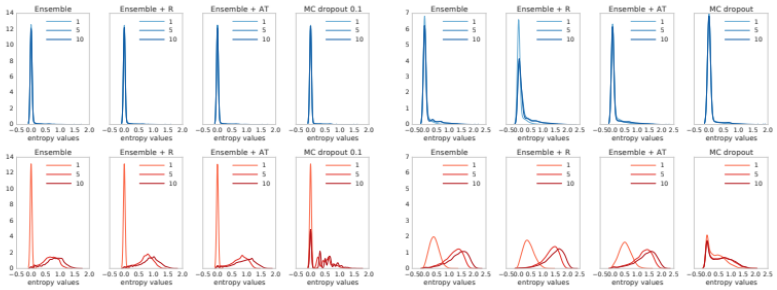
4.3 Ensemble in Model Domain for Uncertainty Estimation (Deep Ensembles)

Lakshminarayanan et al. proposed a method called Deep Ensembles which is an alternative to Bayesian neural networks and does not contain a Bayesian approach for quantifying predictive uncertainty [6]. Besides this method is simple to implement, it is able to parallel computation, and it requires very little hyperparameter tuning. It gives superior quality predictive uncertainty estimations [6]. In this method, different m neural networks are trained in a parallel way and the results of them are aggregated using the bagging methodology. However, the entire dataset is used to train each network unlike classical bagging because neural networks perform better with more data. Also Lakshminarayanan et al. realized that having random initialization of the network parameters and random shuffling of the data points is enough to have a good performance in practice [6]. The quality of the predictive uncertainty estimation is examined in terms of out of distribution entropy and calibration in this research. Lakshminarayanan et al. tested their proposed method on MNIST dataset using 3-layer MLP and SVHN dataset using VGG-style convnet and results showed that the deep ensemble outperforms MC dropout. These results can be seen in Figure 4.2. It also showed that adversarial training is very helpful for small-sized deep ensembles but its effect decreases as the number of networks increases[6].



(a) MNIST dataset using 3-layer MLP (b) SVHN using VGG-style convnet

Figure 4.2: Evaluating predictive uncertainty as a function of ensemble size M. [6]



(a) MNIST-NotMNIST (b) SVHN-CIFAR10

Figure 4.3: Histogram of the predictive entropy on test example from known classes (top row) and unknown classes (bottom row), while varying ensemble size M [6]

Lakshminarayanan et al. also tested their method on the out-of-distribution data and

evaluated the entropy of the results. It is expected high entropy in out-of-distribution data because it signals that the model is giving low confidence in unfamiliar data, and this is the behavior that is desired [6]. The deep ensemble method gave higher entropy on unknown data than MC dropout as a result of these experiments [6]. You can see the entropy histograms in Figure 4.3

4.4 Ensemble in Weight Domain for Uncertainty Estimation

In ensemble methods in the weight domain for uncertainty representation, some approximation is usually used to fit a distribution and try to form an approximate posterior distribution over weight vectors. Then weights are sampled from this distribution and outputs obtained with these weights can be considered as Monte Carlo sampling from the output distribution to represent uncertainty [1]. In this section, the Laplace approximation, one of the simplest methods for an approximation to distribution, will be introduced firstly, and the snapshot ensemble, which is the step into a simple and scalable approach, will be explained. Lastly, the SWA family which is a simple and scalable approach for uncertainty representation will be explained.

4.4.1 Laplace Approximation

Laplace approximation, in the simplest sense, tries to approximate posterior distribution with Taylor expansion. Let say θ^* is a maximum a priori estimate of $h(\theta)$. And $p(\theta) = logh(\theta)$ then the equation below can be written using Taylor expansion

$$p(\theta) \approx p(\theta^*) + (\theta - \theta^*)\dot{p}(\theta^*) + \frac{1}{2}(\theta - \theta^*)^2\ddot{p}(\theta^*) \tag{4.11}$$

and it can be said that $\dot{p}(\theta^*)$ is zero because it θ^* is MAP estimation. So the equation below will be obtained if the exponential function is applied on both sides and the function is expanded to M-dimensional distribution.

$$\begin{aligned} h(\theta) &\approx \exp\left(p\left(\theta^*\right) + \frac{1}{2}\left(\theta - \theta^*\right)^T \mathbf{A}\left(\theta - \theta^*\right)\right) \textit{where } \mathbf{A} = -\ddot{p}(\theta^*) \\ &\approx \exp\left(p\left(\theta^*\right)\right) \exp\left(\frac{1}{2}\left(\theta - \theta^*\right)^T \mathbf{A}\left(\theta - \theta^*\right)\right) \\ &\approx h\left(\theta^*\right) \exp\left(\frac{1}{2}\left(\theta - \theta^*\right)^T \mathbf{A}\left(\theta - \theta^*\right)\right) \end{aligned} \tag{4.12}$$

With this equation, the posterior distribution can be approximated and uncertainty can be expressed. One can see that like a normal distribution where $h(\theta) \approx \mathcal{N}(\theta^*, \mathbf{A}^{-1})$.

4.4.2 Snapshot Ensemble

The snapshot ensembles need to be learned before going deep into the SWA family. It is often not possible to find the global minimum with deep learning networks since there

are more than one minimum point in objective function [19]. And also, some researches showed that less sharp local minimums give better generalization results [20]. This motivation leads us to snapshot ensembles. Huang et al. proposed a method that does not require any additional training cost unlike traditional ensembles and train a single traditional network to have ensembles. They referred to it as implicit ensemble learning [19]. Their approach is developed using the non-convex nature of the neural network and escaping capability from the local minimum of SGD. They allowed the SGD to approximate local minimums M times instead of training M different networks. In each local minimum, they took snapshot the weight of the network so that they obtained M networks from a single training phase. After that, they took averages of them. They used a cyclic learning schedule to escape from the current minima and approximate one other minima. They also noted that if there are parallel resources, still snapshot ensemble is available during training time. In this case, K x M networks can be obtained. K is the number of parallel resources and M is the snapshot number. They also showed that snapshot ensembles generalize posterior distribution better than classical ensemble methods and dropout so that it provides more accuracy and better uncertainty representation. [19]. You can see the visualization of the snapshot ensemble process in Figure 4.4.

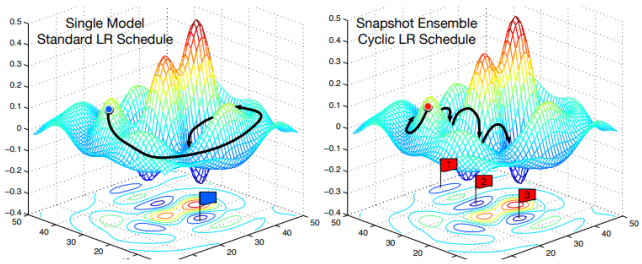


Figure 4.4: Left: Visualization of the single model with standard LR schedule. Right: Visualization snapshot ensemble with cyclic learning rate schedule [19]

4.4.3 Stochastic Weight Averaging (SWA)

SWA is snapshot-based ensemble learning but it has an important different side from snapshot learning. It suggests that it is more promising to take the average of weights and to have one averaged weight vector instead of averaging the result of the different models [21]. So weighted average of the points traversed by SGD during training is referred to as Stochastic Weight Averaging. The Cyclic or constant learning rate can be used in this methodology. Izmailov et al. showed that there are a lot of advantages aspects of SWA compared to traditional snapshot ensemble [21]. These advantages can be listed as follows.

- Izmailov et al. showed that it is not possible to reach the central point of the optimal sets using SGD, it always traverses around the central point. So if snapshots are averaged, it would be a better approximation for the central point of the optimal sets [21].

- Although forward passes as many as the number of the models should be done during the test with snapshot ensemble, it will be sufficient to do a single forward pass in SWA and therefore, computational time will decrease during the test time [21].
- It reduces memory consumption compared to snapshot ensemble. [21].

SWA stores two different weight vectors. One of them is for conventional training of the network, and another one is for storing the running average of the weights. Its memory consumption is negligible considering traditional training. It only maintains one copy of the running average of the DNNs during training. So considering the memory consumed by other ensemble methods, twice as much memory as traditional training is not bad at all. Also, after the training is complete, only weighted averages are needed to be kept [21]. The formula for running average can be seen in the Equation 4.13.

$$w_{\text{SWA}} \leftarrow \frac{w_{\text{SWA}} \cdot n_{\text{models}} + w}{n_{\text{models}} + 1} \tag{4.13}$$

However, having only fixed weights for testing is a disadvantage for uncertainty representation and SWA should be expanded to have the ability to estimate uncertainty. This problem is solved with the SWA-Gaussian method which is explained below.

4.4.4 SWA-Gaussian

It is aimed to approximate posterior distribution with a Gaussian that has the first moment as SWA solution and second moment calculated using SGD iterates [1]. So that uncertainty estimation will be able to be made by sampling from this Gaussian distribution to perform Bayesian model averaging [1].

SWAG-Diagonal

Maddox et al. firstly considered that it would be better to start with the simple diagonal format for covariance matrix [1]. For this purpose, they maintained the second moment for each weight besides the first moment and they obtained a diagonal covariance matrix using the equations below.

$$\overline{\theta^2} = \frac{1}{T}\sum_{i=1}^{T}\theta_i^2 \; and \; \Sigma_{\text{diag}} = \text{diag}\left(\overline{\theta^2} - \theta_{\text{SWA}}^2\right) \tag{4.14}$$

This formula gives an approximation formed by $\mathcal{N}\left(\theta_{\text{SWA}}, \Sigma_{\text{Diag}}\right)$ to the posterior distribution. Maddox et al. noted that it is not required to store these models on the GPU and SWAG-Diagonal only requires a single update of the running averages of the weights per epoch. Thereby, SWAG-Diagonal brings negotiable computational and memory complexity compared to standard training. [1]

SWAG-LR

Although the diagonal approach is generally used in the Bayesian approaches, it is actually a very restrictive approach that ignores the correlations within weights. Maddox et al. expanded the idea to take advantage of the flexibility of the low-rank plus diagonal covariance matrix [1]. They noted that covariance matrix of the SGD iterates can be written as

$$\Sigma = \frac{1}{T-1} \sum_{i=1}^{T} \left(\theta_i - \theta_{\text{SWA}}\right) \left(\theta_i - \theta_{\text{SWA}}\right)^\top \tag{4.15}$$

but they do not have exact value of θ_{SWA} during training, they tried to approximate it with

$$\Sigma \approx \frac{1}{T-1} \sum_{i=1}^{T} \left(\theta_i - \bar{\theta}_i\right) \left(\theta_i - \bar{\theta}_i\right)^\top = \frac{1}{T-1} DD^\top \tag{4.16}$$

where D is the deviation matrix of columns $D_i = \left(\theta_i - \bar{\theta}_i\right)$ and θ_i is the running estimate of the parameters from the first i epochs. They only used the last K of vectors of D_i to have a low rank for the covariance matrix. These vectors correspond to the last K epoch of the training. Then they combined this low-rank covariance with the diagonal covariance to obtain an approximation like $\mathcal{N}\left(\theta_{\text{SWA}}, \frac{1}{2} \cdot \left(\Sigma_{\text{diag}} + \Sigma_{\text{low-rank}}\right)\right)$ for the posterior distribution [1].

SWA Low Rank Asymptotic Covariance Approximation (SWAG-Hessian)

Maddox et al. proposed the idea that asymptotic covariance approximation using low-rank covariance can be done also using Hessian approximation [22]. They used a diagonal approximation of the Hessian and then they multiplied it by low-rank covariance. This brings the approximation

$$\left(\mathbb{H}\left(\theta'\right)^{-1} \Sigma\right)^{1/2} \approx \text{diag}\left(\frac{1}{\tau + \mathcal{I}_{ii}}\right) X \tag{4.17}$$

The purpose of this method is to combine the benefits of SWAG (trajectory dependence), and Laplace approximation (curvature information) [22].

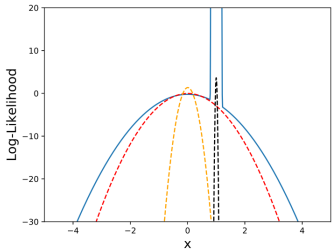


Figure 4.5: SWAG and Laplace approximations. Blue is true likelihood. Red is SWAG, Orange is SWAG-Hessian, Black is Laplace [22]

As you can see in Figure 4.5, Laplace approximation is trapped by a single-mode. SWAG captures the distribution better compared to others. And SWAG-Hessian gives a value in

the middle [22]. Maddox et al. showed that SWA family methods outperform dropout-based, conventional methods and temperature scaling in terms of not only accuracy but also the quality of the uncertainty representation. Low negative log-likelihood indicates good calibration and good predictive uncertainty, as negative log-likelihood shows how well the model captured distribution. Maddox et al. showed that SWAG gives the lowest negative log likelihood in different dataset and architecture compared to conventional methods [1]. You can see this result in Figure 4.6

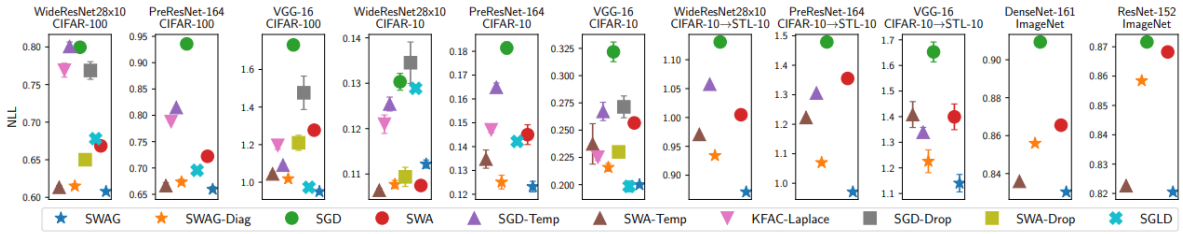


Figure 4.6: Negative log likelihoods for methods. [1]

Also, Maddox et al. showed that SWAG methods are better in terms of calibration. You can see the reliability diagrams in Figure 4.7

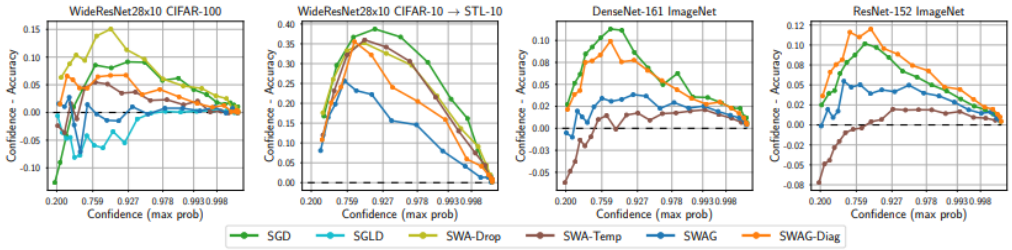


Figure 4.7: Reliability diagrams for methods [1]

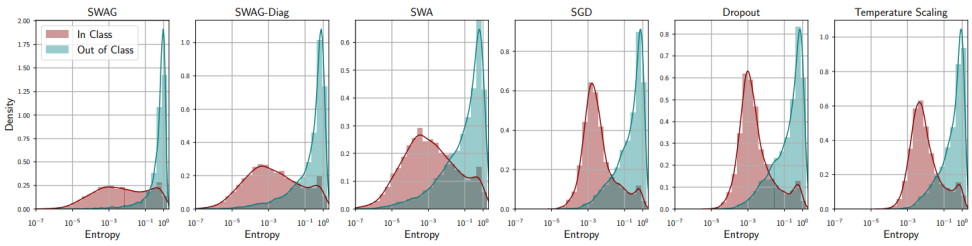


Figure 4.8: In and out of sample entropy distributions for methods [1]

As it is mentioned earlier, the network, which gives a good predictive uncertainty, should give high entropy in out-of-distribution data. Maddox et al. also examined methods on the out-of-distribution datasets and results showed that SWA-based methods give lower entropy

on in-class examples and higher entropy on out-of-class examples. This implies a better quality uncertainty estimation. You can see the entropy results in Figure 4.8

5 Conclusion

To sum up, all ensemble methods which are explained in this report provide us a way to achieve simple and scalable predictive uncertainty. But of course, each of these methods has advantages and disadvantages over the others. These advantages and disadvantages can be evaluated from 4 different aspects. These are memory consumption, the requirement of computational power, the accuracy of the model, and the quality of uncertainty estimation.

First of all, all ensemble methods which are explained in this report outperform BMA in terms of all aspects. And therefore, they will be compared among themselves. Since it is required to train only one model in MC dropout, concrete dropout, and SWA-based methods, they require less memory compared to BMA and Deep ensemble. But in SWA-based methods, two copies of the weights are needed to store the first and second moment. So that SWA-based methods require much more memory than Dropout-based methods. Furthermore, MC dropout, concrete dropout and SWA-based methods requires less computational power compared to others because they only need to train one model. But when dropout is used, converging to minimum points needs more epochs. That is why SWA-based methods are more efficient compared to dropout-based methods in terms of computational power. In terms of the quality of uncertainty estimation researches showed that SWA-based methods gives best results [1]. In the researches scanned while preparing this report, no comparison was found between the deep ensemble and the concrete dropout in this respect since both method's research came out in the same period. But it is known that both outperform the MC dropout method in terms of quality of uncertainty estimation [6, 18]. SWA-based methods and deep ensembles outperform MC dropout and concrete dropout in terms of accuracy [1, 6]. Although SWA-based methods give better calibration results, deep ensemble methods can generalize better and reach higher accuracy [1]. Also, Maddox et al. noted that if training time and computational resources are limited and inference time is not limited, it is more valuable to use the SWA-based method [1]. In this case, determining the needs and choosing a method accordingly would be the most reasonable way.

Bibliography

[1] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson. "A simple baseline for bayesian uncertainty in deep learning". In: *Advances in Neural Information Processing Systems*. 2019, pp. 13153–13164.

[2] L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, and M. Bennamoun. "Hands-on Bayesian Neural Networks–a Tutorial for Deep Learning Users". In: *arXiv preprint arXiv:2007.06823* (2020).

[3] Q. Rao and J. Frtunikj. "Deep learning for self-driving cars: chances and challenges". In: *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*. 2018, pp. 35–38.

[4] J. Ker, L. Wang, J. Rao, and T. Lim. "Deep learning applications in medical image analysis". In: *Ieee Access* 6 (2017), pp. 9375–9389.

[5] R. C. Cavalcante, R. C. Brasileiro, V. L. Souza, J. P. Nobrega, and A. L. Oliveira. "Computational intelligence and financial markets: A survey and future directions". In: *Expert Systems with Applications* 55 (2016), pp. 194–211.

[6] B. Lakshminarayanan, A. Pritzel, and C. Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles". In: *Advances in neural information processing systems* 30 (2017), pp. 6402–6413.

[7] J. K. Yamamoto, K. Koike, A. T. Kikuda, G. A. da Cruz Campanha, and A. Endlen. "Post-processing for uncertainty reduction in computed 3D geological models". In: *Tectonophysics* 633 (2014), pp. 232–245.

[8] M. S. Ayhan and P. Berens. "Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks". In: (2018).

[9] D. Opitz and R. Maclin. "Popular ensemble methods: An empirical study". In: *Journal of artificial intelligence research* 11 (1999), pp. 169–198.

[10] M. Graczyk, T. Lasota, B. Trawiński, and K. Trawiński. "Comparison of bagging, boosting and stacking ensembles applied to real estate appraisal". In: *Asian conference on intelligent information and database systems*. Springer. 2010, pp. 340–350.

[11] Y. Gal and Z. Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning". In: *international conference on machine learning*. 2016, pp. 1050–1059.

[12] K. Hara, D. Saitoh, and H. Shouno. "Analysis of dropout learning regarded as ensemble learning". In: *International Conference on Artificial Neural Networks*. Springer. 2016, pp. 72–79.

[13] Y. Gal. "Uncertainty in deep learning". In: *University of Cambridge* 1.3 (2016).

[14] A. Kendall and Y. Gal. "What uncertainties do we need in bayesian deep learning for computer vision?" In: *Advances in neural information processing systems*. 2017, pp. 5574–5584.

[15] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky. "Bayesian model averaging: a tutorial". In: *Statistical science* (1999), pp. 382–401.

[16] L. Dong, L. Xiong, and K.-x. Yu. "Uncertainty analysis of multiple hydrologic models using the Bayesian model averaging method". In: *Journal of Applied Mathematics* 2013 (2013).

[17] M. Hinne, Q. F. Gronau, D. van den Bergh, and E.-J. Wagenmakers. "A conceptual introduction to Bayesian model averaging". In: *Advances in Methods and Practices in Psychological Science* 3.2 (2020), pp. 200–215.

[18] Y. Gal, J. Hron, and A. Kendall. "Concrete dropout". In: *Advances in neural information processing systems*. 2017, pp. 3581–3590.

[19] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger. "Snapshot ensembles: Train 1, get m for free". In: *arXiv preprint arXiv:1704.00109* (2017).

[20] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. "On large-batch training for deep learning: Generalization gap and sharp minima". In: *arXiv preprint arXiv:1609.04836* (2016).

[21] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson. "Averaging weights leads to wider optima and better generalization". In: *arXiv preprint arXiv:1803.05407* (2018).

[22] W. Maddox, T. Garipov, P. Izmailov, D. Vetrov, and A. G. Wilson. "Fast Uncertainty Estimates and Bayesian Model Averaging of DNNs". In: *Uncertainty in Deep Learning Workshop at UAI*. 2018.