Sub-Ensembles for Fast Uncertainty Estimation in Neural Networks

Matias Valdenegro-Toro
Department of Artificial Intelligence, Bernoulli Institute, University of Groningen, Netherlands.
m.a.valdenegro.toro@rug.nl

Abstract

Fast estimates of model uncertainty are required for many robust robotics applications. Deep Ensembles provides state of the art uncertainty without requiring Bayesian methods, but still it is computationally expensive due to the use of large ensembles. In this paper we propose deep sub-ensembles, an approximation to deep ensembles where the core idea is to ensemble only a selection of layers close to the output, and not the whole model. This is motivated by feature hierarchy learned by convolutional networks that should allow for feature reuse across ensembles. With ResNet-20 on the CIFAR10 dataset, we obtain 1.5-2.5 speedup over a deep ensemble, with a small increase in error and loss, and similarly up to 5-15 speedup with a VGG-like network on the SVHN dataset. Our results show that this idea enables a trade-off between error and uncertainty quality versus computational performance as a sub-ensemble effectively works as an approximation of a deep ensemble.

1. Introduction

Neural networks have revolutionized many fields like object detection, behavior learning, and natural language processing. But despite these advances, most neural network models do not produce uncertainty-aware predictions, only producing a point-wise estimation of the desired output, without confidence intervals or meaningful probabilities that are usable for further decision making.

This means neural network predictions are overconfident, do not consider epistemic (model) and aleatoric (data) uncertainty, and are generally not calibrated [3]. Epistemic uncertainty is particularly of interest, as neural networks trained on limited datasets are used for high-stakes decisions [12], and this kind of uncertainty can guide further decision making and prevent wrong decision on overconfident predictions.

Many applications would benefit from well behaved probabilistic predictions, particularly in Robotics and Autonomous Driving [22]. One method that is able to produce probabilistic predictions is the Bayesian Neural Network (BNN) [14] [15], in which weights are modeled as distributions instead of point estimates, allowing uncertainty in the weights to propagate through the model, and producing calibrated output probabilities and confidence intervals. But inference on BNNs is generally untractable, so approximations are used.

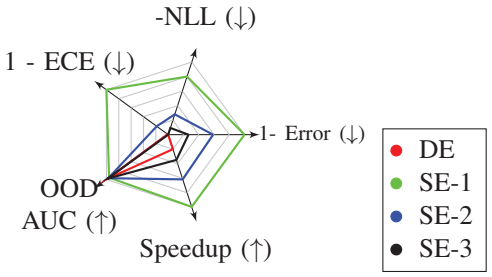
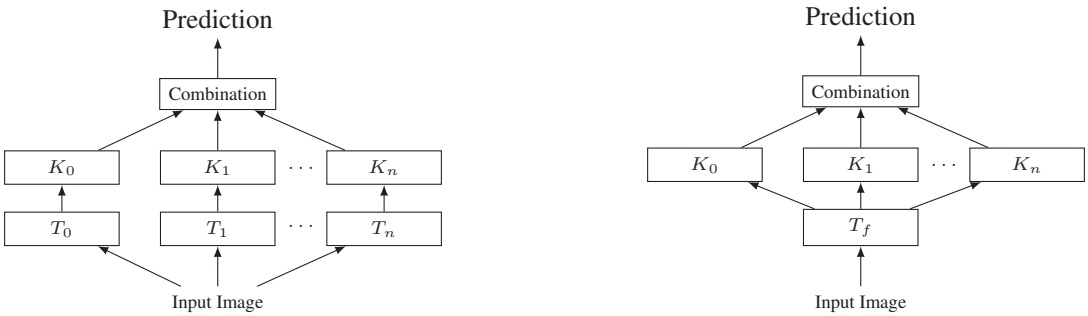


Figure 1: Radar plot comparing sub-ensembles with deep ensembles in several metrics on SVHN with $M = 5$. Values are relative to the best in each metric. Sub-ensembles (SE-) are faster than deep ensembles (DE) while slightly sacrificing other capabilities like error and calibration, depending on the number of layers participating in ensembling.

Many methods exist to augment neural networks with epistemic uncertainty that work as approximate BNN inference, for example MC-Dropout [2], MC-DropConnect [16], and Deep Ensembles [9]. In particular the latter method is a good candidate for many applications due to simplicity and uncertainty quality. For robotics applications, fast (close to real-time) estimates of uncertainty are highly desirable [22]. While Deep Ensembles are state of the art in many tasks, the use of ensembling is problematic for inference in resource constrained applications, as the computation time scales linearly with the number of ensemble members, limiting its use in practical applications.

In this paper we propose an approximation to deep ensembles. By only ensembling part of the model, while sharing a common network trunk, we show that an ensemble model still produces high quality uncertainty estimates, allowing a much faster inference time since a single



(a) Deep Ensembles

(b) Deep Sub-Ensembles

Figure 2: Conceptual comparison of Deep Ensembles and Sub-Ensembles with n ensemble members. The figure shows that in the latter, only a single trunk network T_f is shared across all ensemble members, while in the former multiple trunk networks T_i are used. K represents task networks that are ensembled with n members. In both cases the ensemble predictions are combined to produce outputs with uncertainty.

pass is required for a large trunk network, and several forward passes for smaller task networks to produce an output, which should improve overall performance. We expect that ensembling less layers than the full model should behave as an approximation to the true ensemble, enabling a trade-off between computational resources and error and uncertainty quality (see Figure 1).

The contributions of this paper are: We propose deep sub-ensembles as a simplification of deep ensembles. We evaluate sub-ensembles for regression and classification tasks, and we show that a deep sub-ensemble works as an approximation of a full deep ensemble. Sub-ensembles require significantly less computational resources (1.5x - 5.0x) to produce predictions, while still having good epistemic uncertainty quantification properties, allowing for a trade-off between computation and uncertainty quality.

2. Related Work

There are many methods to model output uncertainty in machine learning. One possible categorization is defined by scalability, where some methods can scale to large datasets and networks, while other methods have convergence issues. Three methods are considered scalable in this context [4] and are of interest for our applications.

[2] proposed that test-time Dropout is an approximation to the BNN posterior, enabling scalable uncertainty estimation in deep neural networks, with applications to computer vision [6] and others. This method is generally called MC-Dropout.

A similar approximation using DropConnect has been proposed by [16], where DropConnect is used instead of Dropout, and the authors show that it outperforms MC-Dropout in terms of error and out of distribution detection. Analogously, this method is called MC-DropConnect.

Deep Ensembles [9] is a non-Bayesian method that is the base for our proposed deep sub-ensembles method. It has been shown that an ensemble of models can produce good estimates of uncertainty, even surpassing MC Dropout, and is also applicable to complex computer vision tasks like segmentation and depth regression [4], and object detection [21].

There are some similar ideas to sub-ensembles in the literature. [18] have used multiple output heads to estimate a distribution of Q-values for exploration in reinforcement learning, which models uncertainty in the Q-values and allows for a more thorough exploration. Unlike Deep Ensembles, this method uses bootstrapping to train Deep Q-Networks.

EnsembleNet [13] is a method proposed to train a multi-head model in an end-to-end way using novel terms added to the loss function. This method has the aim of learning new heads efficiently, but the authors only evaluate 2-3 additional heads with some architectural surgery, and do not evaluate prediction uncertainty.

A network using M Heads is proposed by [11]. This is similar in concept to our proposed sub-ensembles, but there are some scientific flaws that deserve consideration. Lee et al. only evaluates 4-5 ensemble members, and only analyzes accuracy, without consideration to uncertainty quantification. We evaluate many aspects of uncertainty, including calibration, out of distribution detection, intra-ensemble correlation, and computational performance.

Overall scalable methods are able to produce good uncertainty estimates, but this comes at a high computational cost. This work explores the direction of trading uncertainty quality with computational performance gains. Our main argument is that a sub-ensemble works as a principled approximation to a full ensemble, with a user-configurable parameter that allows for trading error and uncertainty quality versus the computational load required to evaluate the sub-

ensemble.

3. Deep Sub-Ensembles as Approximation to Deep Ensembles

Training and performing inference in a Deep Ensemble is computationally expensive. We consider that a neural network architecture can be logically divided [19] into two sub-networks, the trunk network T, and the task network K. The full architecture output for an input $\mathbf{x} \in \mathcal{R}^n$ is then $K(T(\mathbf{x}))$. This concept is shown in Figure 2.

This is motivated by the fact that deep neural networks naturally learn a feature hierarchy, where feature complexity increases with depth, and the last layers perform a task (classification or regression). We hypothesize that it is not necessary to ensemble all layers, as the trunk network will learn similar features across ensemble members, with the task network having a larger contribution to uncertainty.

A Deep Sub-Ensemble conceptually corresponds to training one instance of the full network $K(T(\mathbf{x}))$ in a training set, then fixing the trunk network weights (T_f), and training additional instances of the model where only the task weights are learned. An overview of the training and inference process is shown in Algorithm 1. Each combination of fixed trunk and trainable task network is trained on the same full dataset.

We note that it is possible to conceptually define ensembles and sub-ensembles that are trained end-to-end, by using a single loss over the combines predictions, but this has disadvantages in terms of predictive power. Preliminary experiments indicate that the ensembling property of increasing performance with more ensembles does not happen with end-to-end ensembles and sub-ensembles, only producing similar results, so we did not explore this option further, as the objective of this paper is to improve the computational performance of ensembles and trade-off with predictive performance, not improve training performance per se.

The purpose of this method is to allow the construction of an ensemble that contains a common trunk network T_f, and several instances of the task networks K to build a sub-ensemble[1] $E = \{K_i \,|\, i = 0...M\}$, making the ensemble computationally less expensive to evaluate at inference time, as generally the trunk network contains more computation than the task networks, and the trunk network is evaluated once. We now describe how to train and combine ensemble members into an output predictive distribution $p_e(y \,|\, \mathbf{x})$. We denote ensemble members predictions with the $_i$ subindex (e.g. $p_i(\mathbf{x})$), and M is the number of ensemble members.

Classification. Uncertainty in this task is represented through a categorical distribution over classes. Each ensemble member predicts a set of probabilities $p(y \,|\, \mathbf{x})$ which are then averaged as:

$$p_e(y \,|\, \mathbf{x}) = M^{-1} \sum_i p_i(y \,|\, \mathbf{x}).$$

For training, a standard cross-entropy loss is used.

Regression. Here each ensemble member outputs a Gaussian distribution, which is parameterized as a mean $\mu_i(x)$ and variance $\sigma_i^2(x)$, corresponding to one output head each. The combined ensemble output is then a Gaussian mixture model, with all distributions equally weighted, which can be approximately computed as:

$$\begin{aligned} fp_e(y \,|\, \mathbf{x}) &\sim \mathcal{N}(\mu_*(\mathbf{x}), \sigma_*^2(\mathbf{x})) \\ \mu_*(\mathbf{x}) &= M^{-1} \sum_i \mu_i(\mathbf{x}) \\ \sigma_*^2(\mathbf{x}) &= M^{-1} \sum_i (\sigma_i^2(\mathbf{x}) + \mu_i^2(\mathbf{x})) - \mu_*^2(\mathbf{x}) \end{aligned}$$

Regression requires a different loss, as supervision is available for $\mu_i(x)$, but not for $\sigma_i^2(x)$. A heteroscedastic Gaussian log-likelihood loss [9] [6] is used for this purpose:

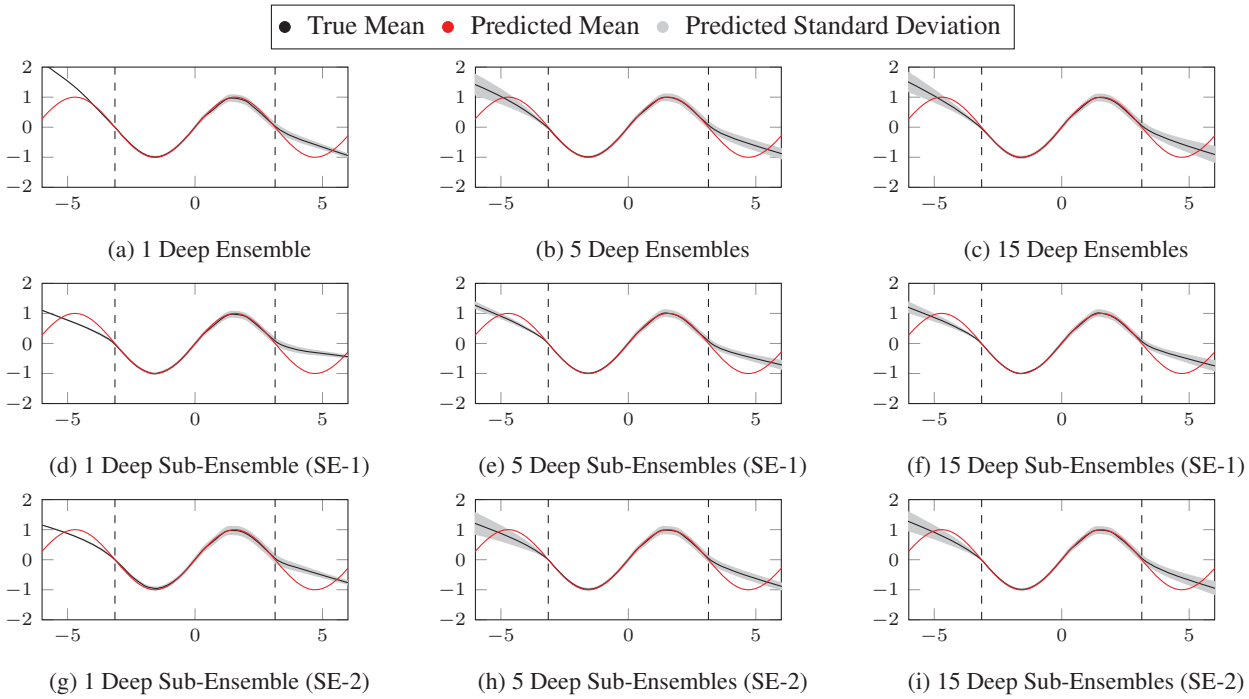
$$-\log p(y_n|\mathbf{x}_n) = \frac{\log \sigma_i^2(\mathbf{x}_n)}{2} + \frac{(\mu_i(\mathbf{x}_n) - y_n)^2}{2\sigma_i^2(\mathbf{x}_n)}$$

With this loss, the output variance $\sigma_i^2(x)$ can be interpreted as an estimate of the aleatoric uncertainty in the data, while epistemic uncertainty is obtained through ensembling, and is present in $\sigma_*^2(x)$.

Algorithm 1 Training and Inference process for Deep Sub-Ensembles

1: **Input:** Training set D, Trunk and Task models T and K, number of ensemble members M.
2: **Output:** Trained trunk network T_f and set of ensemble members E.
3: Stack the trunk and task model T-K and train an initial instance of it on D.
4: Freeze weights of the initially trained instance of T, producing T_f
5: Set ensemble $E = \{K\}$ with the initially trained instance of K
6: **for** $i = 1$ to $i = M - 1$ **do**
7: Stack T_f and a randomly initialized instance of K and train it on D. Note that features produced by T_f might be cached for shorter training time.
8: $E = E \cup \{K\}$ (Add task network K to ensemble E)
9: **end for**
10: Ensemble predictions can now be made by evaluating T_f with an input image, then evaluating each ensemble member $K \in E$ given the output of T_f, and combining the predictions.

[1] A sub-ensemble is composed of a trunk network T_f and the set of task networks E.



(a) 1 Deep Ensemble (b) 5 Deep Ensembles (c) 15 Deep Ensembles

(d) 1 Deep Sub-Ensemble (SE-1) (e) 5 Deep Sub-Ensembles (SE-1) (f) 15 Deep Sub-Ensembles (SE-1)

(g) 1 Deep Sub-Ensemble (SE-2) (h) 5 Deep Sub-Ensembles (SE-2) (i) 15 Deep Sub-Ensembles (SE-2)

Figure 3: Comparison between ensembles and sub-ensembles in a toy regression problem. The shaded areas represent one-σ confidence intervals. Dotted lines indicate the training set limits. True mean, predicted mean, and predicted standard deviation are presented.

We now proceed to experimentally evaluate our proposed method. We use the notation SE-m to denote a specific sub-ensemble network configuration, where m denotes how many layers/blocks counted from the output layer participate in ensembling (the task layers K), while the remaining layers are shared across all ensemble members (the trunk layers T_f).

4. Results in Toy Regression

We first showcase our method with a toy regression problem, as an easily understandable example. We draw samples from $f(x) = \sin(x) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma(x))$ and $\sigma(x) = 0.15(1 + e^{-1})^{-1}$. We define a training set where $x \in [-\pi, \pi]$, and an independent test set where $x \in [-2\pi, -\pi] \cup [\pi, 2\pi]$. We use a three layer network, with 64 neurons per layers and ReLU activation. We define sub-ensembles consisting of a task networks with one and two layers, denoted SE-1 and SE-2 correspondingly. We test three different number of ensembles $M \in [1, 5, 15]$, which should produce variations in model uncertainty. A plot of the predictions is shown in Figure 3, while the training and testing set samples are shown in Figure 9, and numerical comparisons in Tables 2 (in the supplementary).

Our results in Figure 3 show that sub-ensembles produce similar uncertainty compared to a deep ensemble, with increasing out of distribution uncertainty for testing samples, and a good estimation of the aleatoric uncertainty in the training data.

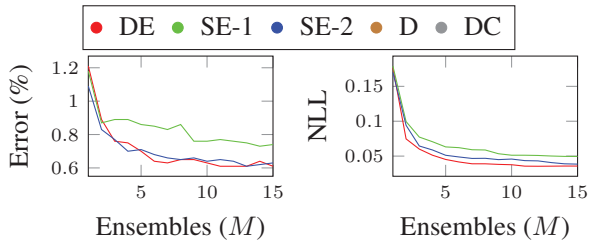
As expected, we see that uncertainty quality (as measured by the negative log-likelihood[2]) degrades as less layers are ensembled, for example when comparing SE-2 and SE-1.

5. Results in Image Classification

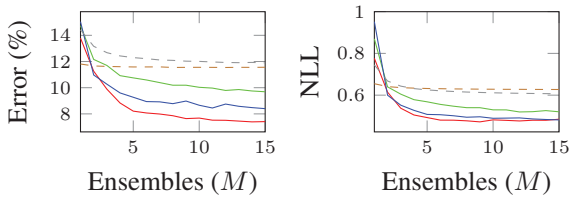
We evaluate our proposed method in three datasets for image classification: MNIST, CIFAR10, and SVHN. For MNIST [10], we use a simple batch normalized CNN consisting of a 32 3×3 convolution, followed by 64 3×3 convolution, and a fully connected layer with 128 neurons and an output fully connected layer with 10 neurons and a softmax activation. All layers use ReLU activations. We select two sets of task networks for ensembling, the first uses the last two fully connected layers (denominated SE-1), and the second task network uses the three last layers (2 FC and one Conv, denominated SE-2). These results are shown in Figure 4a.

For CIFAR10 [8], we use ResNet-20 [5] with random shifts and horizontal flips as data augmentation. We define a set of two task networks, the first containing the clas-

[2]NLL's measure fit to a probability distribution which is a proxy for error and uncertainty quality.



(a) MNIST



(b) CIFAR10

Figure 4: Results on MNIST (with a simple CNN) and CIFAR10 (with ResNet-20), showing error and negative log-likelihood as the number of ensembles is varied

sification layers and the last ResNet stack (with 64 filters and stride $S = 2$, denominated SE-1), and a second task network containing the previously defined network plus the second from last ResNet stack (with 32 filters and $S = 2$, denominated SE-2). These results are shown in Figure 4b.

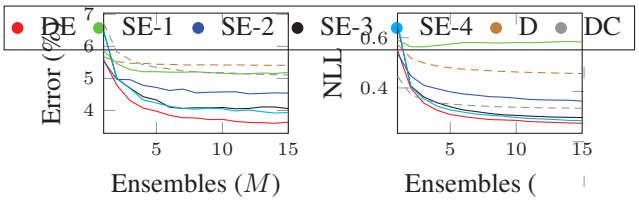
Finally, for SVHN [17] we use a batch normalized VGG-like network [20], with modules defined as two convolutional layers with the same number of filters and ReLU activation, and one 2×2 max pooling layer. The network is composed of modules with 32, 64, 128, and 128 filters, and followed by a fully connected layer of 128 neurons, and a final output layer with 10 neurons and softmax activation. We define a set of four task networks that we evaluate, namely taking the classification layers, and going from these layers backwards through the network modules, denominated as SE-1 to SE-4. These results are shown in Figure 5.

Overall we used Adam [7] with a learning rate $\alpha = 0.01$ and trained models until convergence. Detailed architectures and training details are available in the supplementary material.

5.1. Error and Uncertainty Quality

For all datasets we evaluate both the classification error, and the negative log-likelihood (NLL). For SVHN we additionally evaluate the calibration curve in the Section 5.3. We compare our proposed method (called Deep Sub-Ensembles, SE) with Deep Ensembles [9] (DE) as a baseline, as the number of ensemble members is varied, from 1 to 15 ensemble members and task networks.

We also consider two additional baselines for error and negative log-likelihood on SVHN and CIFAR10: MC-



(a) Error and Negative Log-Likelihood (NLL) as the number of ensembles is varied

Figure 5: Results on SVHN using a batch normalized VGG-like network

Dropout [2] (Denoted by D) and MC-DropConnect [16] (Denoted by DC). These baselines are selected to check if deep sub-ensembles is worse than other scalable uncertainty methods. We found there is a considerable variation in error and NLL in terms of the drop probability p with these methods, so we tune p for each method independently by minimizing error. Tuning results are available in the supplementary material.

On MNIST as shown in Figure 4a, ensembling two layers of the model (SE-2) has error comparable with Deep Ensembles, but only ensembling the fully connected layers (SE-1) produces a higher error. The uncertainty as measured by the negative log-likelihood is comparable in all three scenarios, indicating the preliminary viability of our idea.

On CIFAR-10 (Figure 4b), error increases by around 2% with a sub-ensemble when compared to Deep Ensembles, and the increase of NLL is minor, specially when ensembling two sets of layers (SE-2). Sub-ensembles also outperforms MC-Dropout and MC-DropConnect in both error and negative log-likelihood, indicating that it is a high quality approximation of a deep ensemble.

Finally on SVHN (Figure 5), there is a more marked difference in increasing error as less layers are ensembled. From SE-2 there is a clear improvement on negative log-likelihood, being very similar to the deep ensembles baseline from the SE-3 configuration. In terms of error, sub-ensembles outperforms both MC-Dropout and MC-DropConnect, while configuration SE-2 outperforms MC-Dropout but not DropConnect, while configurations with more ensembled layers (SE-3 and SE-4) do outperform MC-DropConnect by a small margin.

Overall our results show that Deep Sub-Ensembles is in all cases an approximation to Deep Ensembles, with always having higher error, but negative log-likelihood can be similar, depending on how many layers are ensembled. This is expected as ensembling less layers than the full model should behave as an approximation to the true ensemble,

	Sub-Ensembles (SE-1)			Deep Ensembles		
M	AUC	Mean Entropy		AUC	Mean Entropy	
		ID	OOD		ID	OOD
1	0.655	0.051	0.195	0.787	0.054	0.281
5	0.957	0.043	0.867	0.986	0.112	1.325
10	0.971	0.042	0.986	0.994	0.125	1.589
15	0.973	0.040	1.009	0.996	0.130	1.665

Table 1: Numerical results for OOD detection on SVHN-CIFAR10. ID/OOD Entropy corresponds to sample means.

enabling a trade-off between computational resources and error and uncertainty quality.

5.2. Out of Distribution Detection - SVHN vs CIFAR10

We have also evaluated the out of distribution detection (OOD) capabilities of Deep Sub-Ensembles. For this we used the ensemble model trained on SVHN, and evaluated on the CIFAR10 test set for ODD examples, and in the SVHN test set for in-distribution (ID) examples, as both are color image datasets, the image sizes are compatible (both are 32×32), and there are no classes in common.

To decide if an example is out of distribution, we use the entropy $H(\mathbf{x})$ of the ensemble probabilities $p_e(y \mid \mathbf{x})$:

$$H(\mathbf{x}) = -\sum_{c \in C} p_e(y_c \mid \mathbf{x}) \log p_e(y_c \mid \mathbf{x})$$

Where the subindex $_c$ indicates probability of class c. Then we use a threshold in the entropy to decide if an example is in-distribution or out-of-distribution. The idea is that in-distribution examples will have a low entropy, as certain class probabilities dominate the prediction, while out-of-distribution examples will have a uniform class probability distribution, which increases entropy.

We evaluate performance using area under the ROC curve (AUC) as the number of ensemble members is varied, as this summarizes performance across thresholds. Results are presented in Table 1 and Figure 10 (supp) , while we also make a kernel density estimate of the ID and OOD entropy distributions, shown in Figure 11 (in supplementary).

Our results indicate that probabilities produced by Deep Ensembles have an excellent capability for out-of-distribution detection, starting from 5 ensemble members. Deep Sub-Ensembles also produces good separation between ID and OOD examples, but requires more ensemble members to reach performance that is slightly worse than a Deep Ensemble, at 15 ensemble members. Entropy distributions show that it clearly divides ID and OOD examples, but Deep Sub-Ensembles produce lower entropy values for OOD samples across all number of ensemble members. We interpret this as evidence of Deep Sub-Ensembles being approximations of Deep Ensembles.

We provide additional results in the supplementary material on accuracy as a function of confidence threshold in this task. These results show that sub-ensembles closely approximates the confidence properties of a deep ensemble, with a small loss of less than 2% accuracy with $M = 15$ and the SE-2 and SE-3 configurations. SE-1 has a larger gap of around 10% accuracy.

5.3. Calibration

As mentioned before, calibration is an important feature of uncertainty-aware machine learning models. We produce reliability diagrams [1] (also known as confidence-accuracy plots or calibration curves), and estimate the expected calibration error [3] (ECE), both on the SVHN dataset. To produce reliability diagrams, we use $N = 7$ histogram bins.

Reliability diagrams are available in Figure 6, and they show that both methods are calibrated, starting from being underconfident with the base model (single ensemble member), and with increasing calibration as ensemble members are added.

A plot of expected calibration error versus number of ensemble members is shown in Figure 6e. These plots show how ECE decreases with increasing number of ensemble members, except for the case of SE-1 which has a level of miscalibration across all number of ensembles. SE-2 has an acceptable degree of calibration, while SE-3 has ECE that is comparable with a full Deep Ensemble.

5.4. Trunk vs Ensemble Network Performance

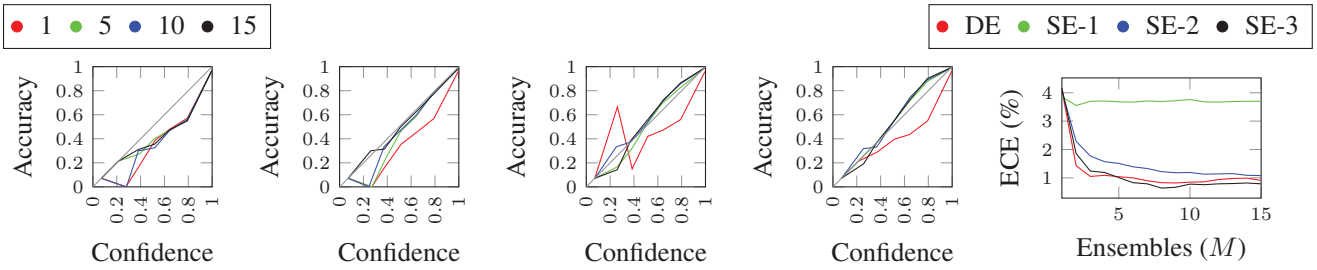
One additional property of a Deep Sub-Ensemble is that since a trunk network is trained once, due to random weight initialization and randomness in the training process, the model might not produce the best features given the data, and the ensemble performance could be limited by the trunk network.

We evaluated this hypothesis by performing 10 runs of training Deep Sub-Ensembles with the SE-1 configuration on MNIST, CIFAR10, and SVHN with ensemble members varying from 1 to 15, and evaluating the trunk model error and the ensemble error. These results are shown in Figure 7.

It can be seen that there is a strong correlation between trunk and ensemble error, with the same effect happening for negative log-likelihood, except for SVHN which does not perform well with the SE-1 configuration in terms of NLL. Overall these results indicate that a more thorough design and training of the trunk model might be necessary for good ensemble performance.

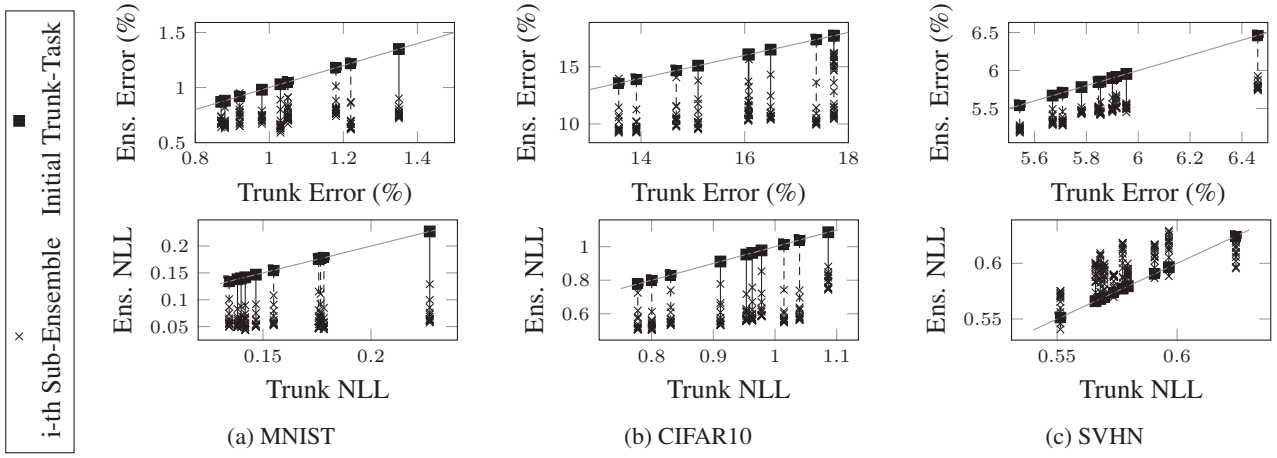
5.5. Computational Performance Analysis

Ensembling less layers has a theoretical computation cost advantage over ensembling the full model. In this section we aim to evaluate this hypothesis and measure how



(a) SE-1 (b) SE-2 (c) SE-3 (d) DE (e) ECE vs M

Figure 6: Reliability diagrams with different number of ensembles M (a-d) and Expected Calibration Error as the number of ensembles is varied (e) on SVHN. Sub-ensembles are competitive when ensembling at least two layers.



(a) MNIST (b) CIFAR10 (c) SVHN

Figure 7: Relationship between Sub-Ensemble (Y axis) and Trunk network performance (X axis), in terms of error and negative log-likelihood, for configuration SE-1. There is a strong correlation between the trunk and sub-ensemble performance.

much speedup can be obtained by using a sub-ensemble instead of a full deep ensemble.

For this purpose we estimate the number of floating point operations (FLOPs) [3] for each architecture, as we vary the number of ensembles. We evaluate models for SVHN and CIFAR10, as they are the most complex ones. We plot the error and negative log-likelihood as function of FLOPs for different number of ensemble members, as a way to show the trade-off between error and uncertainty quality with computational requirements, and we also compute the speedup of a sub-ensemble over a full ensemble with number of FLOPs as proxy for computational performance. This is adequate as FLOPs are an implementation independent metric, and we are comparing similar architectures where FLOPs should not deviate considerably from real performance.

Results are shown in Figure 8 for CIFAR10 and SVHN. For CIFAR10, there is a clear trade-off between error and computation, but it is possible to trade small NLL amounts for big gains in computational performance (up to 1.5-2.5 times).

For SVHN, similar patterns in error trade-offs are seen, and the NLL decreases considerably with small compute, for example SE-1 almost has no gains in NLL with very small increases in FLOPs. SE-2 and SE-3 allow to trade small variations in NLL for large computational gains (up to 2-5 times).

Looking at speedups it can be seen that a sub-ensemble obtains decent speed improvements over a full ensemble, but there are large variations in speedup depending on model complexity and number of ensemble members. For example a maximum speedup of 15 can be reached with SE-1 on a VGG-like network, but smaller speedups are obtained on ResNet, maximum 2.7 with SE-2. It is clear that speedups heavily depend on the granularity and selection of layers to be ensembled.

5.6. Discussion

In this section we discuss our overall results. We believe our experimental results show that sub-ensembles work as

[3] Not to be confused with FLOPS, which is floating point operations per second

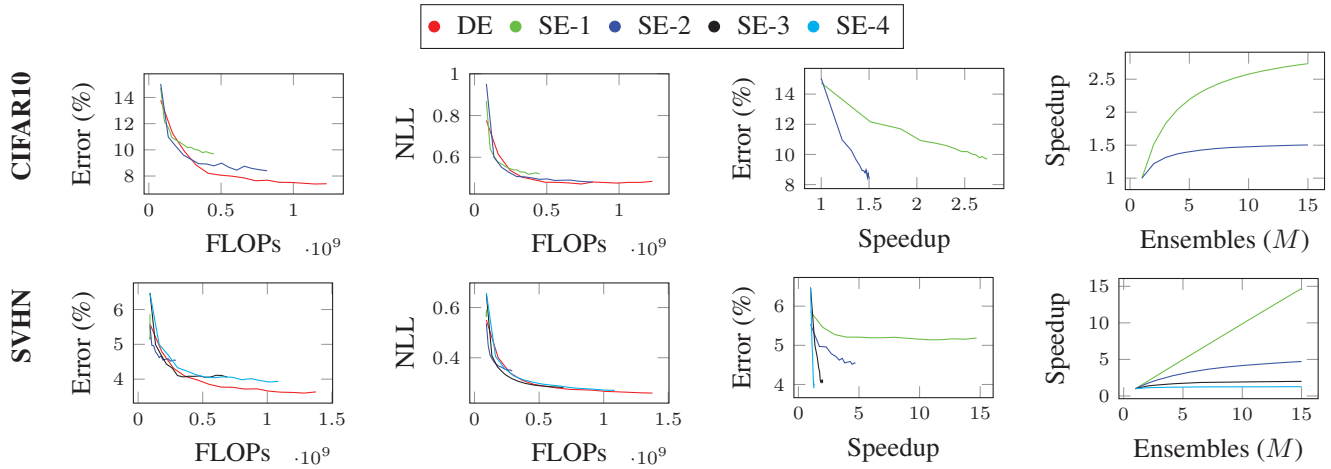


Figure 8: Computational performance measured as number of floating point operations (FLOPs) and Speedup (versus a deep ensemble) compared with Error and NLL on CIFAR10 using ResNet-20 (top) and SVHN using VGG-like network (bottom). Note how increasing FLOPS decreases error and NLL, while Error vs Speedup plateaus at different error levels depending on sub-ensemble configuration (SE-1, SE-2, etc), showing the trade-off between error and sub-ensemble computational complexity. NLL is very similar in all configurations indicating similar uncertainty estimation capabilities.

an approximation to deep ensembles, as in most cases performance is lower (in terms of error and NLL) than the full deep ensemble, and this gap increases when less layers are ensembled (e.g. with SE-1 on all datasets we evaluated). The gap reduces as more layers are ensembled, and in some cases it is indistinguishable from the deep ensemble performance (e.g. with SE-2 error on MNIST, or SE-2 NLL on CIFAR10).

This pattern is also reflected in other metrics, such as out of distribution performance (AUC), expected calibration error, and intra-ensemble correlations. This indicates that the number of layers to be ensembled works as a parameter that tunes between full and partial ensemble performance. This means that a sub-ensemble works as an approximation to a deep ensemble.

Overall the performance gap between sub-ensembles and full ensembles depends on many factors, including dataset, model architecture, and which layers will be ensembled. We believe this latter factor is the most important, as it directly controls the ensemble and its potential for correctly estimating uncertainty. The selection of which layer the architecture is divided into task and trunk network can be considered a hyper-parameter.

6. Conclusions

In this paper we have presented deep sub-ensembles for neural networks, a simplification of deep ensembles with the purpose of reducing computation time at inference.

Our results show that it might not be necessary to ensemble all the layers in a model, and that a trade-off between computation time and uncertainty quality is be possible, depending on the task and dataset being learned. Sub-ensembles require the network to be divided into trunk and task networks, and the layer where this division is made can be considered as a hyper-parameter.

We evaluated our method in three image classification datasets: MNIST, CIFAR10, and SVHN. On CIFAR10, sub-ensembles with the SE-2 configuration produce a 1.5-2% gap in error, and a negligible gap in negative log-likelihood. On SVHN, the error gap is less then 0.5% with the SE-3 and SE-4 configurations and same negligible gap in negative log-likelihood.

Our results also show that sub-ensembles have similar calibration properties as a deep ensemble, with a 0.5% gap in expected calibration error on SVHN. For out of distribution detection between SVHN and CIFAR10, we also find that there is a small gap (around 2-3%), as measured in area under the ROC curve.

Computational performance in terms of FLOPs, we measured speedups up to 1.5-2.5 for ResNet-20 on the CIFAR10 dataset, and speedups of 5-15 for a VGG-like network on the SVHN dataset, with small increase in error and NLL.

We expect that sub-ensembles will increase the use of neural networks with uncertainty in their outputs, as the reduced computational cost will make these methods more practical for real-world applications, all while being better calibrated than standard neural networks, and will small gaps in performance with a full ensemble.

References

[1] Morris H DeGroot and Stephen E Fienberg. The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 32(1-2):12–22, 1983. 6

[2] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016. 1, 2, 5

[3] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org, 2017. 1, 6

[4] Fredrik K Gustafsson, Martin Danelljan, and Thomas B Schön. Evaluating scalable bayesian deep learning methods for robust computer vision. *arXiv preprint arXiv:1906.01620*, 2019. 2

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4, 11

[6] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017. 2, 3

[7] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5, 10, 11

[8] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 4

[9] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017. 1, 2, 3, 5

[10] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998. 4

[11] Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra. Why m heads are better than one: Training a diverse ensemble of deep networks. *arXiv preprint arXiv:1511.06314*, 2015. 2

[12] Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific reports*, 7(1):1–14, 2017. 1

[13] Hanhan Li, Joe Yue-Hei Ng, and Paul Natsev. Ensemblenet: End-to-end optimization of multi-headed models. *arXiv preprint arXiv:1905.09979*, 2019. 2

[14] David JC MacKay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992. 1

[15] David JC MacKay. The evidence framework applied to classification networks. *Neural computation*, 4(5):720–736, 1992. 1

[16] Aryan Mobiny, Hien V Nguyen, Supratik Moulik, Naveen Garg, and Carol C Wu. Dropconnect is effective in modeling uncertainty of bayesian deep networks. *arXiv preprint arXiv:1906.04569*, 2019. 1, 2, 5

[17] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 5

[18] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pages 4026–4034, 2016. 2

[19] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014. 3

[20] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5

[21] John Skinner, David Hall, Haoyang Zhang, Feras Dayoub, and Niko Sünderhauf. The probabilistic object detection challenge. *arXiv preprint arXiv:1903.07840*, 2019. 2

[22] Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, et al. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420, 2018. 1