Deep Sub-Ensembles for Fast Uncertainty Estimation in Image Classification

Matias Valdenegro-Toro
German Research Center for Artificial Intelligence
28359 Bremen, Germany
matias.valdenegro@dfki.de

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. In this paper we propose deep sub-ensembles, an approximation to deep ensembles where the core idea is to ensemble only the layers close to the output, and not the whole model. 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 NLL, 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.

1 Introduction

Neural networks have revolutionized many fields like object detection, behavior learning, and natural language processing. But most neural network models are overconfident, producing predictions that do not consider epistemic uncertainty, and are generally not calibrated.

Many methods exist to augment neural networks with epistemic uncertainty, for example MC-Dropout [1] and Deep Ensembles [4]. In particular the latter method is a good candidate for many applications due to simplicity and quality of uncertainty. For robotics applications, fast (close to real-time) estimates of uncertainty are highly desirable [10].

In this paper we propose a simplification of the Deep Ensembles method. 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 in image classification tasks. This allows for a much faster inference time as a single pass is required for a large trunk network, and several forward passes for the sub-models connected to the output.

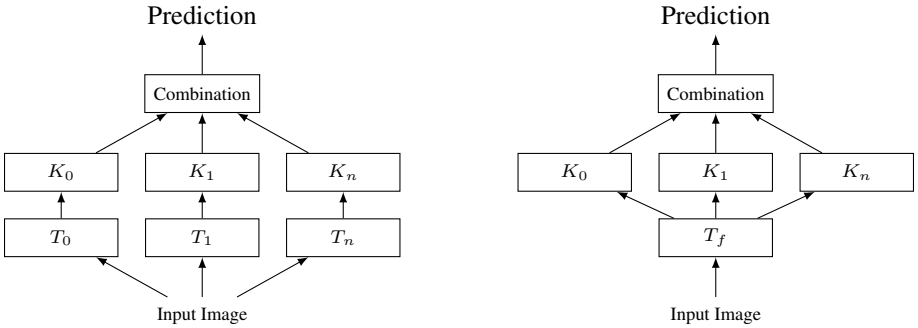
2 Deep Sub-Ensembles

Deep Ensembles [4] is a non-Bayesian method for uncertainty quantification of machine learning models. It has been shown that an ensemble of models can produce good estimates of uncertainty, even surpassing methods like MC-Dropout.

Training and performing inference in a Deep Ensemble is computationally expensive. We consider that a neural network architecture can be logically divided [8] into two sub-networks, the trunk network T, and the task network K. The full architecture output for an input x is then $K(T(x))$.

A Deep Sub-Ensemble conceptually corresponds to training one instance of the full network $K(T(x))$ in a training set, and then fixing the trunk network weights (T_f), and training additional instances

4th workshop on Bayesian Deep Learning (NeurIPS 2019), Vancouver, Canada.



(a) Deep Ensembles

(b) Deep Sub-Ensembles

Figure 1: Conceptual comparison of Deep Ensembles and Deep 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. In both cases the ensemble predictions are combined to produce outputs with uncertainty.

Input: Training set D, Trunk and Task models T and K, number of ensemble members n.
Output: Trained trunk network T_f and ensemble members E.

1: Stack the trunk and task model T-K and train an initial instance of it on D.
2: Freeze weights of the initially trained instance of T, producing T_f
3: Set ensemble $E = \{K\}$ with the initially trained instance of K
4: **for** $i = 1$ to $i = n - 1$ **do**
5: Stack T_f and a randomly initialized instance of K and train it on D.
6: $E = E \cup K$
7: **end for**
8: Ensemble predictions can now be made by evaluating T_f with an input image, then evaluating each ensemble member in E given the output of T_f, and combining the predictions.

Figure 2: Training and Inference process for Deep Sub-Ensembles

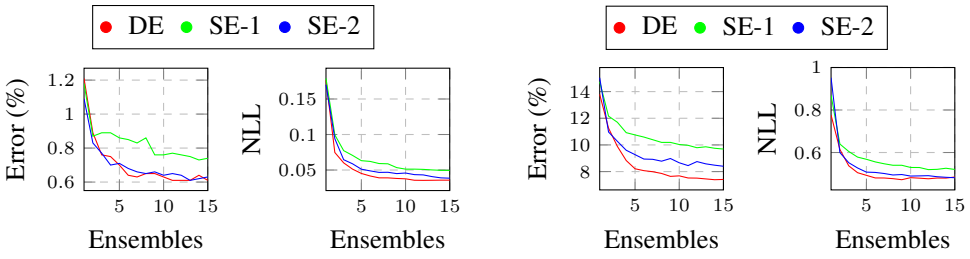
of the model where only the task weights are learned. An overview of the training and inference process is shown in Figure 2. Our concept of a Deep Sub-Ensemble is similar to Bootstrapped DQN [7], where a shared network and multiple output heads are used to produce high quality uncertainty estimates, but in a Deep Sub-Ensemble no bootstrap estimates are used, each combination of fixed trunk and trainable task network is trained on the same full dataset.

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 network K_i, 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. For classification, the sub-ensemble output is the average of task network probability predictions, namely $f(x) = N^{-1} \sum_i K_i(T_f(x))$.

3 Experimental Results in Image Classification

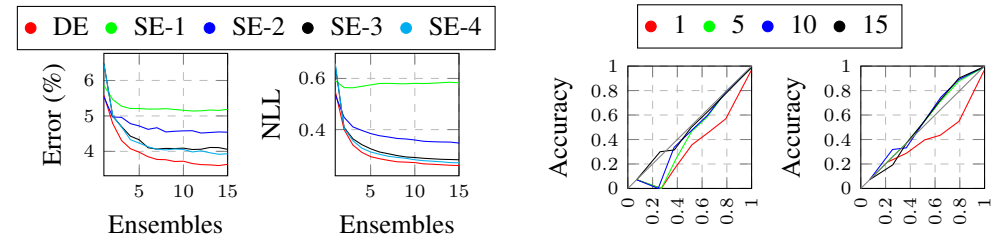
We evaluate our proposed method in three datasets for image classification: MNIST, CIFAR10, and SVHN. For MNIST [5], we use a simple batch normalized CNN consisting of a 32 3×3 convolution, followed by 64 3×3 convolution, and a fully connected layer with 128 neurons and an output fully connected layer 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 3a.

For CIFAR10 [3], we use ResNet-20 [2] with random shifts and horizontal flips as data augmentation. We define a set of two task networks, the first containing the classification layers and the last ResNet stack (with 64 filters and stride $S = 2$, denominated SE-1), and a second task network containing



(a) MNIST (b) CIFAR10

Figure 3: 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



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

(b) Calibration Curves, SE-2 (left) vs DE (right)

Figure 4: Results on SVHN using a batch normalized VGG-like network

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 3b.

Finally, for SVHN [6] we use a batch normalized VGG-like network [9], 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 4.

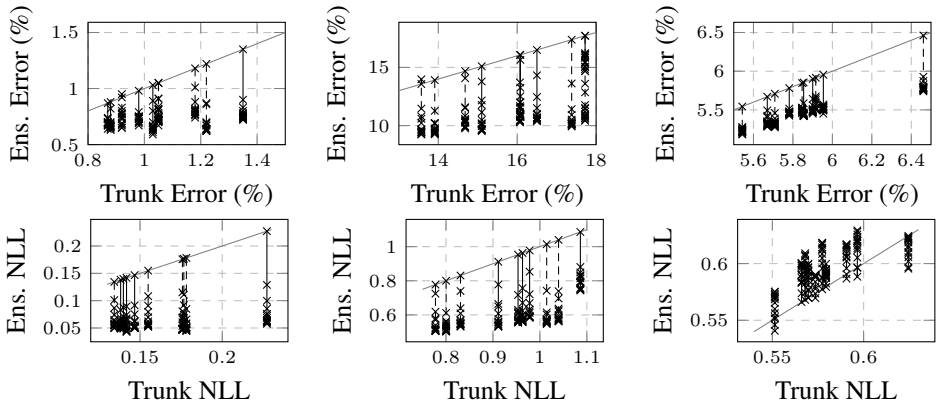
3.1 Error and Uncertainty Quality

For all datasets we evaluate both the classification error, and the negative log-likelihood. For SVHN we additionally evaluate the calibration curve. We compare our proposed method (called Deep Sub-Ensembles, SE) with Deep Ensembles [4] (DE), as the number of ensemble members is varied, from 1 to 15 ensemble members and task networks.

On MNIST as shown in Figure 3a, 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 preliminar viability of our idea.

On CIFAR-10 (Figure 3b), error increases by around 2% with a sub-ensemble when compared to Deep Ensembles, but the increase of NLL is minor, specially when ensembling two sets of layers (SE-2). Finally on SVHN (Figure 4a), 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 since SE-3.

Calibration curves available in Figure 4b show that both methods are calibrated, starting from being underconfident with the base model (single ensemble member), and with increasing confidence as ensemble members are added.



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

Figure 5: Relationship between Sub-Ensemble and trunk network performance, in terms of error and negative log-likelihood. Here we only evaluate SE-1.

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

3.2 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. We evaluated this by training 10 runs of Deep Sub-Ensembles, and evaluating the trunk model error and the ensemble error, these results are shown in Figure 5. It can be seen that there is a strong correlation between trunk and ensemble error, with the same effect happening for negative log-likelihood. This indicates that a more thorough design and training of the trunk model might be necessary for good ensemble performance.

3.3 Out of Distribution Detection - SVHN vs CIFAR10

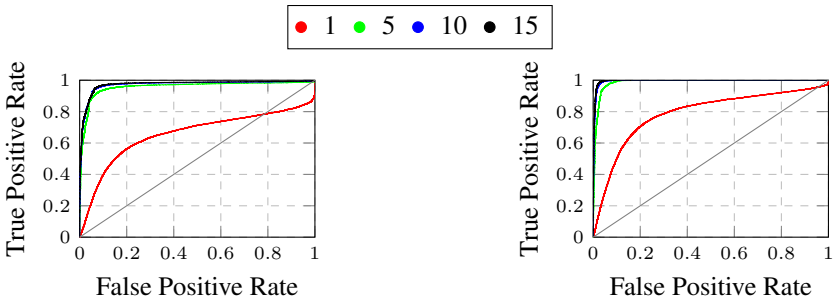
We have also evaluated the out of distribution detection (ODD) capabilities of Deep Sub-Ensembles. For this we used the ensemble model trained on SVHN, and evaluated on the CIFAR10 test set for ODD examples, and in the SVHN test set for in-distribution (ID) examples, as the image sizes are compatible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

To decide if an example is out of distribution, we use the entropy of the ensemble probabilities:

$$H(x) = -\sum_{c \in C} f(x)_c \log f(x)_c$$

Then we put 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 of this method using the area under the ROC curve as the number of ensemble members is varied. Results are presented in Table 1 and Figure 6. 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. The mean ID and OOD entropy show that it clearly divides ID and OOD examples.



(a) Deep Sub-Ensembles

(b) Deep Ensembles

Figure 6: ROC Curves for out-of-distribution detection using entropy on SVHN vs CIFAR10.

	Sub-Ensembles			Deep Ensembles		
# of Ensembles	AUC	ID Entropy	OOD Entropy	AUC	ID Entropy	OOD Entropy
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.

3.4 Computational Performance Analysis

Ensembling less layers has a theoretical advantage over ensembling the full model. In this section we aim to evaluate this hypothesis and measure how 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) [1] 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, computed as the number of FLOPs of a full ensemble divided by FLOPs for the sub-ensemble.

Results are shown in Figure 7 for CIFAR10 with ResNet-20, and 8 for SVHN with a batch normalized VGG-like network. 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.

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

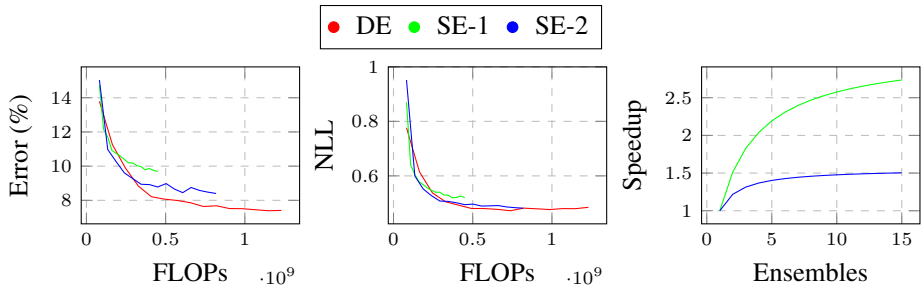


Figure 7: Computational performance measured as FLOPs and Speedup on CIFAR10 using ResNet-20

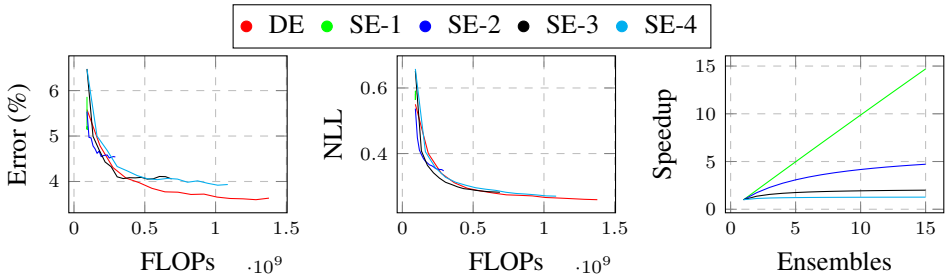


Figure 8: Computational performance measured as FLOPs and Speedup on SVHN using a batch normalized VGG-like network

4 Conclusions and Future Work

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

Our preliminary 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 might be possible, depending on the task and dataset being learned. 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.

As future work, we wish to evaluate on the ImageNet dataset, and explore ways to train sub-ensembles in an end-to-end fashion, which could reduce training time. We will also extend this evaluation to regression and estimate computation times on CPU and GPU.

Acknowledgements

This work has been partially supported by the Autonomous Harbour Cleaning project funded by EIT Digital (Ref 18181).

References

[1] 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.

[2] 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.

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

[4] 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.

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

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

[7] 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.

[8] 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.

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

[10] 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.