A. Broader Impact Statement

Uncertainty estimation is important for general safe use of machine learning models. Most models used in practice do not have any kind of proper uncertainty estimation, which prevents the user from evaluating if the model believes its answers to be correct or not. One reason uncertainty estimation methods are not used in practice is computational performance, as ensembling or monte carlo sampling increases the computational cost of a prediction radically.

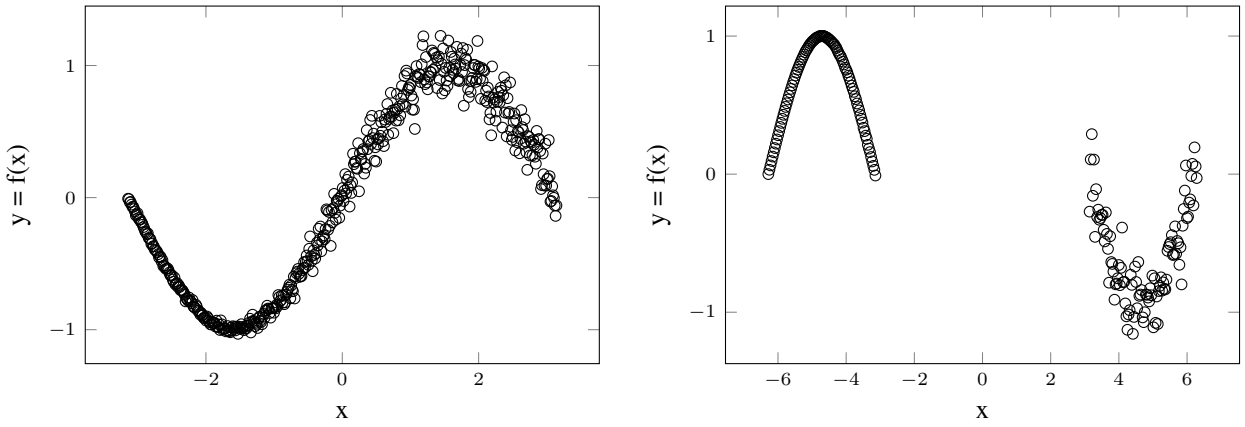
Sub-ensembles allow for an approximate uncertainty estimation at a much lower computational cost, and we expect that uncertainty estimation methods will have increased use due to less computational limitations.

Possible negative societal impact is that sub-ensembles provide only an approximation to full ensemble, and the approximation quality is unknown, so incorrect or misleading uncertainty predictions can mislead the final user into trusting or mistrusting an incorrect or correct prediction. Additional research is needed to validate and bound the uncertainty estimated produced by a sub-ensemble.

B. Additional Regression Information and Results

Figure 9 presents the samples used to train and test the toy regression examples, drawn from the distribution we mention in the paper. Table 2 presents numerical comparisons in terms of negative log-likelihood for the train and test sets.

All networks (Deep Ensembles and Deep Sub-Ensembles) used for this example were trained using Adam [7] with a learning $\alpha = 0.001$, using the heteroscedastic Gaussian log-likelihood loss, for 50 epochs and a batch size $B = 32$.



(a) Training Set

(b) Testing Set

Figure 9: Samples used as training and testing set for the Toy Regression problem

	Sub-Ensembles (SE-1)		Sub-Ensembles (SE-2)		Deep Ensembles	
# of Ensembles	Train NLL	Test NLL	Train NLL	Test NLL	Train NLL	Test NLL
1	-2.20	116.8	-2.01	195.1	-2.23	2409.7
5	-2.33	5.2	-2.22	1.46	-2.22	0.7
15	-2.34	2.5	-2.25	0.42	-2.17	0.5

Table 2: Numerical comparison between Deep Ensembles and Deep Sub-Ensembles in terms of negative log-likelihood for the Toy Regression problem

C. Detailed Neural Network Architectures

We use the following notation to describe neural network architectures. Conv(N_f, $F_w \times F_h$) is a 2D convolutional layer with N_f filters of width F_w and height F_h. MP($P_w \times P_h$) is a 2D Max-Pooling layer with sub-sampling size $P_w \times P_h$, and FC(n) is a fully connected layer with n output neurons. BN() represents a Batch Normalization layer.

Each Deep Sub-Ensemble architecture configuration was designed to test the trade-off between computation and error/uncertainty quality, so we divided a given neural network architecture into blocks of layers, and then selected blocks to make sub-ensembles, starting from the block that produced the network output and moving backwards. We leave a more fine-grained analysis of this trade-off for future work.

Most architectures are trained using the Adam optimizer [7], using a learning rate $\alpha = 0.001$. No data augmentation was used, except for CIFAR10 training. The standard cross-entropy loss is used to train all classification models.

C.1. Simple CNN for MNIST

We used a simple CNN architecture for the MNIST dataset, namely configuration:

$$\text{Conv(32, } 3 \times 3\text{)-BN()-Conv(64, } 3 \times 3\text{)-BN()-MP(}2 \times 2\text{)-FC(128)-FC(10).}$$

All learnable layers use the ReLU activation, except for the last FC layer which uses softmax for classification. This network is trained with the Adam optimizer for 30 epochs with a batch size $B = 32$.

Given the neural network architecture described above, we defined two sub-ensemble configurations for task networks:

SE-1 MP(2×2)-FC(128)-FC(10)

SE-2 Conv(64, 3×3)-BN()-MP(2×2)-FC(128)-FC(10)

The trunk network consists of the layers that are left in the best network when removing layers to form the task network.

C.2. Batch Normalized VGG-like for SVHN

For evaluation on the SVHN dataset, we use a simple architecture that is similar to the VGG network with Batch Normalization. We first define a VGG module/block as a stack of two 3×3 convolutions and one 2×2 Max-Pooling layer, with Batch Normalization in between:

$$\text{VGG(n) = Conv(n, } 3 \times 3\text{)-BN()-Conv(n, } 3 \times 3\text{)-BN()-MP(}2 \times 2\text{)}$$

Then we used the following configuration as base network: VGG(32)-VGG(64)-VGG(128)-Conv(128, 3×3)-BN()-VGG(128)-FC(128)-BN()-FC(10). All learnable layers use ReLU activations, except for the last FC layer which uses a softmax activation.

Given this architecture we defined four sub-ensemble task network configurations:

SE-1 FC(128)-BN()-FC(10)

SE-2 Conv(128, 3×3)-BN()-VGG(128)-FC(128)-BN()-FC(10)

SE-3 VGG(128)-Conv(128, 3×3)-BN()-VGG(128)-FC(128)-BN()-FC(10)

SE-4 VGG(64)-VGG(128)-Conv(128, 3×3)-BN()-VGG(128)-FC(128)-BN()-FC(10)

In each configuration, the trunk network consists of the layers that are left in the base network when removing the layers to form the task network. We train these architectures for 30 epochs with Adam and using a batch size $B = 32$.

C.3. ResNet-20 for CIFAR10

For CIFAR10 we used ResNet-20. We prefer this network because of its faster training due to its depth and it obtains good classification performance.

We use a residual connection module ResLayer(n, s), as defined in [5], where n is the number of filters, and s is the convolution stride. This module contains two 3×3 convolutional layers and an additive residual connection between the input and the last convolutional layer's output. If strides are larger than one, then a 1×1 strided convolution is applied to the input, to make dimensions compatible for the addition operation in the residual connection. ReLU activations are used in the convolutional layers, and after the additive residual connection. Batch Normalization layers are added after each convolutional layer.

A residual stack ResStack(n, s) consists of three residual modules, the first with a given stride, and the final two with a stride $S = 1$. The full ResNet-20 network can be now defined as:

$$\text{ResLayer(16, 1)-ResStack(16, 1)-ResStack(32, 2)-ResStack(64, 2)-GAP()-FC(10)}$$

Where GAP is the Global Average Pooling layer and the last FC layer uses a softmax activation. Give the ResNet-20 architecture above, we define two sub-ensemble task network configurations:

SE-1 ResStack(64, 2)-GAP()-FC(10)

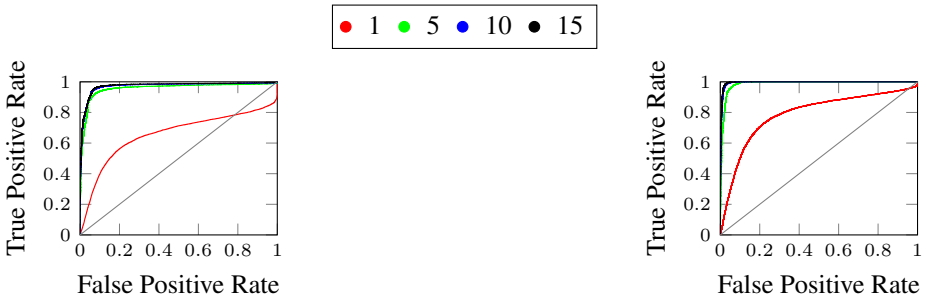
SE-2 ResStack(32, 2)-ResStack(64, 2)-GAP()-FC(10)

As before, the trunk network is formed by the remaining layers of the architecture after removing the task layers.

All CIFAR10 architectures are trained using Adam, for 100 epochs, and a batch size $B = 128$. We use data augmentation by performing random horizontal flips, and randomly shifting the image horizontally and vertically by up to 4 pixels.

D. Additional Out of Distribution Detection Results

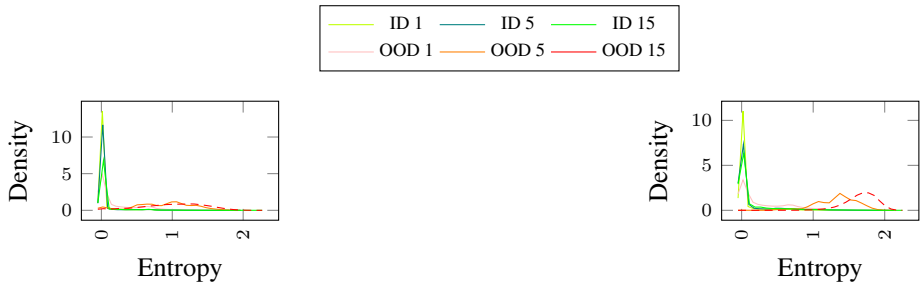
This section presents the ROC curves and entropy distribution for the out of distribution detection results in the main paper.



(a) Deep Sub-Ensembles (SE-1)

(b) Deep Ensembles

Figure 10: ROC Curves for out-of-distribution detection using entropy on SVHN vs CIFAR10. Note how the AUC greatly increases when $M > 1$.



(a) Deep Sub-Ensembles (SE-1)

(b) Deep Ensembles

Figure 11: Entropy distributions for in-distribution (ID, SVHN) and out of distribution (OOD, CIFAR10) samples

E. Intra-Ensemble Correlation Analysis

We wish to study the behavior of predictions of a sub-ensemble, and how it compares to a full ensemble. For this purpose we compute the intra-ensemble correlation of the predictions on the SVHN and CIFAR10 datasets. This is estimated by computing the correlation between predictions made by an ensemble of M - 1 members, and the predictions made by a newly trained ensemble member (completing M members).

These results are presented in Figure 12. In both datasets it can be seen that an SE-1 sub-ensemble has the highest correlation among ensemble members, which indicates that there is reduced variety in the predictions. A ensemble (in red) has the lowest correlation in both cases. Note that in all models and ensemble configurations, the correlation decreases as ensemble members are added, which is the expected behavior.

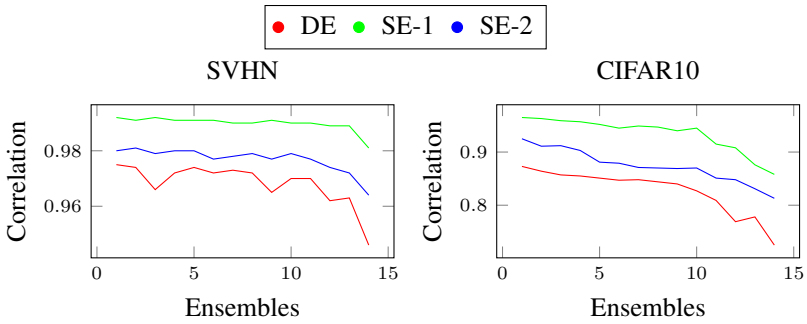


Figure 12: Intra-ensemble Correlation on SVHN and CIFAR10 as the number of ensembles is varied, compared across multiple sub-ensemble configurations.

Results using the SE-2 sub-ensemble configuration indicate a decreased correlation in comparison to SE-1, with values being closer to a full ensemble. These results add to the evidence that a sub-ensemble behaves as an approximation to a full ensemble.

F. Accuracy as Function of Confidence for SVHN-CIFAR10 OOD Detection

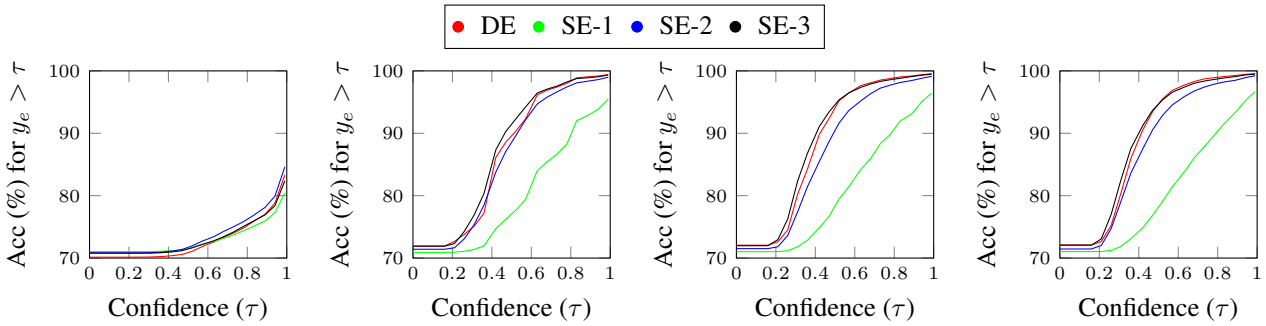
For the out of distribution detection task in SVHN-CIFAR10 datasets, we also evaluate the quality of predicted probabilities as an additional result and validation. In the paper we evaluated the OOD detection capabilities in terms of the output distribution entropy, and we also expect that the probability confidence values can also be individually used for out of distribution detection. For OOD samples, low confidence values should be produced, while for ID samples, high confidence values should be outputted.

To test this hypothesis, we analyze the probability predictions in the OOD and ID datasets. For each prediction, we use the maximum probability $\max_c p_e(y_c \,|\, \mathbf{x})$ as confidence score, and the predicted class based on the maximum probability. We compute the accuracy for samples ID-OOD where $y_e = \max_c p_e(y_c \,|\, \mathbf{x}) > \tau$ as a function of the confidence score threshold τ. We expect that as τ increases, the set of samples $\max_c p_e(y_c \,|\, \mathbf{x}) > \tau$ should converge to the ID dataset, as the most confidence samples should belong to the ID dataset and be correctly classified. Any prediction made in the OOD dataset will generally be wrong, as the model is just guessing these samples, and this will decrease overall accuracy.

The baseline accuracy is $\frac{26032}{26032+10000} \sim 72.2\%$, corresponding to always predicting the ID dataset (SVHN) correctly. Results are shown in Figure 13 as the number of ensemble members is varied for $M \in \{1, 5, 10, 15\}$.

Our results show how the predicted probabilities improve in quality as more ensemble members are used. We see that the plots are ordered exactly by the number of ensembles layers, configuration SE-1 being the lowest quality, and configuration SE-3 being comparable to the full deep ensemble.

We believe these results add confirmatory evidence that sub-ensembles behave as an approximation to deep ensembles.



(a) $M = 1$ ensemble members (b) $M = 5$ ensemble members (c) $M = 10$ ensemble members (d) $M = 15$ ensemble members

Figure 13: Accuracy vs confidence plot for SVHN-CIFAR10 out of distribution detection. Accuracy is computed for $y_e = \max_c p_e(y_c \,|\, \mathbf{x})$ larger than a confidence threshold τ.

G. Computational Performance Results

In this section we provide numerical details of the results that were presented in Figures 10 and 11 of the paper. Computational complexity results for CIFAR10 are presented in Table 3, while results for SVHN are presented in Table 4 and 5. In all tables we present summarized results for $M \in \{1, 5, 10, 15\}$.

	DE			SE-1				SE-2			
M	Error (%)	NLL	FLOPs	Error (%)	NLL	FLOPs	Speedup	Error (%)	NLL	FLOPs	Speedup
1	13.79 %	0.78	82 M	14.74 %	0.87	82 M	1.00	15.04 %	0.95	82 M	1.00
5	8.21 %	0.49	410 M	10.75 %	0.57	187 M	2.19	9.27 %	0.51	292 M	1.40
10	7.68 %	0.48	820 M	10.03 %	0.53	319 M	2.58	8.66 %	0.49	555 M	1.48
15	7.41 %	0.48	1.2 G	9.69 %	0.52	450 M	2.74	8.40 %	0.48	819 M	1.50

Table 3: Numerical computational performance results on CIFAR10 with ResNet-20

	DE			SE-1				SE-2			
M	Error (%)	NLL	FLOPs	Error (%)	NLL	FLOPs	Speedup	Error (%)	NLL	FLOPs	Speedup
1	5.58 %	0.55	92 M	5.86 %	0.59	92 M	1.00	5.54 %	0.54	92 M	1.00
5	3.98 %	0.29	459 M	5.21 %	0.57	92 M	4.97	4.72 %	0.38	149 M	3.08
10	3.72 %	0.27	918 M	5.16 %	0.58	93 M	9.87	4.58 %	0.36	221 M	4.16
15	3.63 %	0.26	1.4 G	5.19 %	0.58	94 M	14.70	4.54 %	0.35	292 M	4.72

Table 4: Numerical computational performance results on SVHN with a VGG-like network, for DE vs SE-1/SE-2

	DE			SE-3				SE-4			
M	Error (%)	NLL	FLOPs	Error (%)	NLL	FLOPs	Speedup	Error (%)	NLL	FLOPs	Speedup
1	5.58 %	0.55	92 M	6.48 %	0.65	92 M	1.00	6.47 %	0.66	92 M	1.00
5	3.98 %	0.29	459 M	4.33 %	0.32	262 M	1.75	4.23 %	0.31	376 M	1.22
10	3.72 %	0.27	918 M	4.09 %	0.29	476 M	1.93	4.05 %	0.28	731 M	1.26
15	3.63 %	0.26	1.4 G	4.06 %	0.28	689 M	2.00	3.93 %	0.27	1.1 G	1.27

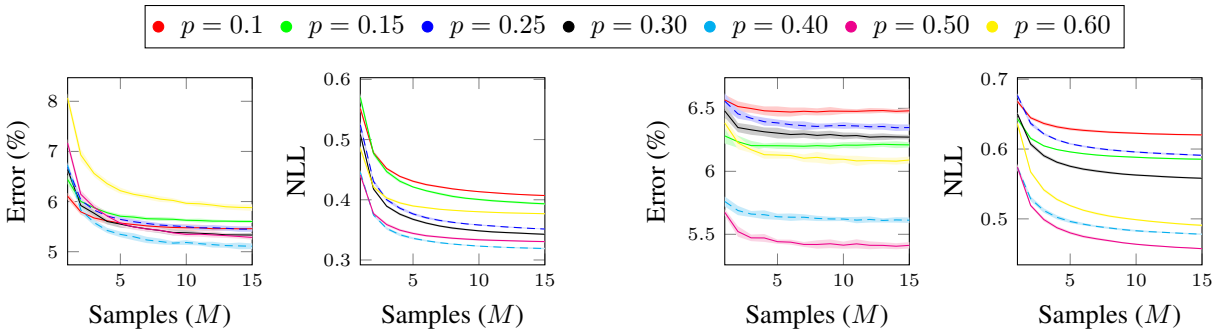
Table 5: Numerical computational performance results on SVHN with a VGG-like network, for DE vs SE-3/SE-4

H. Drop Probability Tuning for MC-Dropout and MC-DropConnect

While producing baselines that use Dropout and DropConnect, we found that the drop probability p can have a large impact in the resulting error and negative log-likelihood, so we decided to tune the value of p independently in each dataset (SVHN or CIFAR10), and for MC-Dropout and MC-DropConnect.

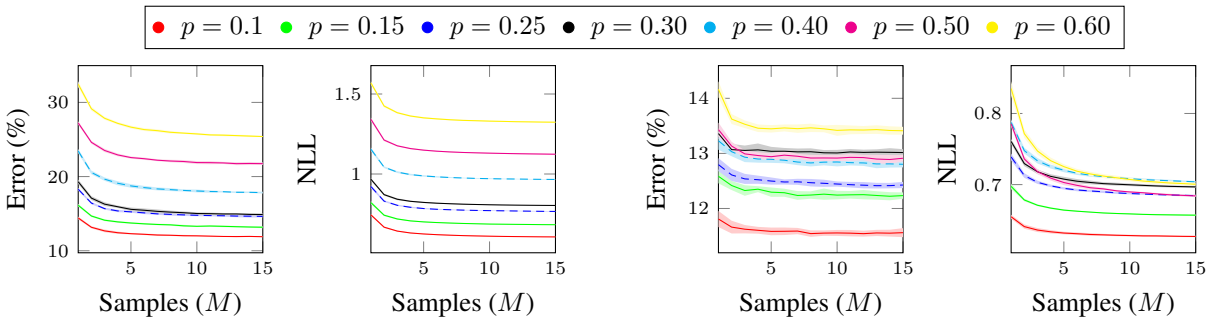
For this purpose, for each model architecture we trained 10 instances of that model, and we varied the Dropout/DropConnect probability in $p \in \{0.1, 0.15, 0.25,$
$0.30, 0.40, 0.50, 0.60\}$, which contains many values common in the literature, and others used to cover a wide spectrum. For each model, we draw samples in range $M \in \{1 - 15\}$ and compute classification error and negative log-likelihood for each set of samples. We estimate the mean and standard deviation of each metric as the number of samples M is varied.

Results are presented in Figure 14 for SVHN, and Figure 15 and for CIFAR10.



(a) DropConnect (b) Dropout

Figure 14: Error and Negative log-likelihood as # of Samples is varied for MC-DropConnect and MC-Dropout on SVHN



(a) DropConnect (b) Dropout

Figure 15: Error and Negative log-likelihood as # of Samples is varied for MC-DropConnect and MC-Dropout on CIFAR10

We use a criteria of minimum error to select the best drop probability, as indicated by our tuning results. For SVHN, we use $p = 0.4$ for DropConnect, and $p = 0.5$ for Dropout. In CIFAR10, we use $p = 0.1$ for both DropConnect and Dropout.