Masksembles for Uncertainty Estimation

Nikita Durasov[1]* Timur Bagautdinov[2] Pierre Baque[2] Pascal Fua[1]
[1]Computer Vision Laboratory, EPFL, {name.surname}@epfl.ch
[2]Neural Concept SA, {name.surname}@neuralconcept.com

Abstract

Deep neural networks have amply demonstrated their prowess but estimating the reliability of their predictions remains challenging. Deep Ensembles are widely considered as being one of the best methods for generating uncertainty estimates but are very expensive to train and evaluate.

MC-Dropout is another popular alternative, which is less expensive, but also less reliable. Our central intuition is that there is a continuous spectrum of ensemble-like models of which MC-Dropout and Deep Ensembles are extreme examples. The first one uses effectively infinite number of highly correlated models while the second one relies on a finite number of independent models.

To combine the benefits of both, we introduce Masksembles. Instead of randomly dropping parts of the network as in MC-dropout, Masksemble relies on a fixed number of binary masks, which are parameterized in a way that allows to change correlations between individual models. Namely, by controlling the overlap between the masks and their size one can choose the optimal configuration for the task at hand. This leads to a simple and easy to implement method with performance on par with Ensembles at a fraction of the cost. We experimentally validate Masksembles on two widely used datasets, CIFAR10 and ImageNet.

1. Introduction

The ability of deep neural networks to produce useful predictions is now abundantly clear but assessing the reliability of these predictions remains a challenge. Among all the methods that can be used to this end, MC-Dropout [8] and Deep Ensembles [21] have emerged as two of the most popular ones. Both of those methods exploit the concept of *ensembles* to produce uncertainty estimates. MC-Dropout does so implicitly - by training a single *stochastic* network, where randomness is achieved by dropping different subsets of weights for each observed sample. At test time, one can obtain multiple predictions by running this network multiple times, each time with a different weight configuration. Deep Ensembles, on the other hand, build an explicit ensemble of models, where each model is randomly initialized and trained independently using *stochastic* gradient descent. As importantly, both methods rely on the fact that the outputs of individual models are diverse, which is achieved by introducing stochasticity into the training or testing process.

However, simply adding randomness does not always lead to diverse predictions. In practice, MC-Dropout often performs significantly worse than Deep Ensembles on uncertainty estimation tasks [21, 28, 12]. We argue that this can be attributed to the fact that each weight is dropped randomly and independently from the others. For many configurations of hyperparameters, in particular the dropout rate, this results in similar weight configurations and, consequently, less diverse predictions [6]. Deep Ensembles seem to not share this weakness for reasons that are not yet fully understood and usually produce significantly more reliable uncertainty estimates. Unfortunately, this comes at a price, both at training and inference time: Building an ensemble requires training multiple models and using it means loading all of them simultaneously in memory, typically a very scarce resource.

In this work, we introduce *Masksembles*, an approach to uncertainty estimation that tackles these challenges and produces reliable uncertainty estimates on par with Deep Ensembles at a significantly lower computational cost. The main idea behind the method is simple - introduce a more structured way to drop model parameters than that of MC-Dropout.

Masksembles produces a *fixed number* of binary masks which specify the network parameters to be dropped. The properties of the masks effectively define the properties of the final ensemble in terms of capacity and correlations. During training, for each sample we randomly choose one of the masks and drop the corresponding parts of the model just like standard dropout. During inference, we run the model multiple times, once per mask, to obtain a set of predictions and, ultimately, an uncertainty estimate. Our method has three key hyperparameters: the total number of

*This work was supported in part by the Swiss National Science Foundation

masks, the average overlap between masks and the number of ones and zeros in each mask. Intuitively, using a large number of masks approximates MC-Dropout, while using a set of non-overlapping, completely disjoint masks, yields an Ensemble-like behavior. In other words, as illustrated in Fig. 5 Masksembles defines a spectrum of model configurations of which MC-Dropout and Deep Ensembles can be seen as extreme cases. We evaluate our method on several synthetic and real datasets and demonstrate that it outperforms MC-Dropout in terms of accuracy, calibration, and out-of-distribution (OOD) detection performance. When compared to Deep Ensembles, our method has similar performance, but is more favorable in terms of training time and memory requirements. Furthermore, Masksembles is simple to implement and can serve as a drop-in replacement for MC-Dropout in virtually any model. To summarize, the main contributions of this work are as follows:

- We propose an easy to implement, drop-in replacement method that performs better than MC-Dropout, at a similar computational cost, and that matches Ensembles at a fraction of the cost.
- We provide theoretical insight into two popular uncertainty estimation methods - MC-Dropout and Ensembles, by creating a continuum between the two.
- We provide a comprehensive evaluation of our method on several public datasets. This validates our claims and provides guidance for the choice of Masksemble parameters in practical applications.

2. Related Work

MC-Dropout [8] and Deep Ensembles [21] have emerged as two of the most prominent and practical uncertainty estimation methods for deep neural networks [1, 28, 12]. In what follows we provide a short background in uncertainty estimation, review the best-known methods, and discuss potential alternatives.

2.1. Background

The goal of *Uncertainty Estimation* (UE) is to produce a measure of confidence for model predictions. There are two major types of uncertainty that can be modeled within the Bayesian framework [5]. *Aleatoric uncertainty*, also known as *Data uncertainty* [7], captures noise that arises from data and therefore is irreducible, because it is directly caused by the natural complexity of the data. Sensor noise, labels noise and classes overlapping are among the most common sources of aleatoric uncertainty. *Epistemic uncertainty* or *model uncertainty* [7] accounts for uncertainty in the parameters of the model that we learned, that is, it represents our ignorance about parameters of the data generation process. This type of uncertainty is reducible and therefore, given enough data, could be completely eliminated. In addition to aforementioned types of uncertainty, there is also *Distributional uncertainty*. Also known as *dataset shift* [31], this type of uncertainty is useful when there is a mismatch between the training and testing data distributions, and a model is confronted with unfamiliar samples.

For all the aforementioned types of uncertainty ideally we want our model to output an additional signal indicating how reliable is its prediction for a particular sample. More formally, given an input, we want our model to output the actual prediction target as well as the corresponding uncertainty measure. For regression tasks, the uncertainty measure can be the variance or confidence intervals around the prediction made by the network [29]. For classification tasks, popular uncertainty measures are the entropy and max-probability [25, 22]. Ideally, the measure of uncertainty should be high when the model is incapable of producing accurate prediction for the given input and low otherwise. Generally speaking, uncertainty can be either learned from the data directly [22, 19], in particular for big data regimes and aleatoric uncertainty, or could be acquired from a diverse set of predictions [8, 22], which are obtained from stochastic reguralization techniques or ensembles. In this work, we focus on the latter family of methods.

2.2. Ensembles

Deep Ensembles [21] involve training an ensemble of deep neural networks on the same data, by initializing each network randomly. This approach is originally inspired by a classical ensembling method, bagging [14]. An uncertainty estimate is obtained by running all the elements in the ensemble and aggregating their predictions. The major drawback of Deep Ensembles is the computational overhead: it requires training multiple independent networks, and during inference it is desirable to keep all these networks in memory. More generally, ensembling has been a popular way to boost the performance of individual neural network models [13, 20, 22, 30], with the quality improvements often being associated with diverse predictions. In [6] authors provide a comprehensive comparison of several popular ensemble learning methods as well as several approximate Bayesian inference techniques. Extensive experiments on vision tasks suggest that simple ensembles have the lowest correlation between individual models. This is an important advantage of these methods, as ultimately this leads to highly diversified predictions on the samples which are very distinct from those observed in training.

Snapshot-based methods take a slightly different approach towards creating an ensemble, and try to collect a diverse set of weight configurations over the course of a single training. Snapshot ensembles [18] build upon the idea of using cyclical learning rates to ensure that the optimization process can efficiently explore multiple local optima.

Similarly, Fast Geometric Ensembling [9], uses a method to identify low-error paths between local optima to obtain a collection of diverse representations. Although these approaches partially tackle training time issues, they typically perform worse than the standard ensembles [1] and consume the same amount of memory. Furthermore, because of the specifics of their training procedure, in particular shared initialization with a pre-trained network, individual snapshots are not guaranteed to be decorrelated, which might hurt the performance [1].

2.3. Dropout

Dropout [33] is an extremely popular and simple approach to stochastically regularizing deep neural networks. During training, neurons are randomly dropped with fixed probability, which effectively produces an approximation of several slightly different model architectures and allows a single model to mimic ensemble behavior. Originally Dropout was used during training as a stochastic regularization technique, but [8, 7] proposed using it during inference, providing sound mathematical justification for ensembling and Bayesian model averaging analogies. MC-Dropout is extremely easy to use, and has zero memory overhead compared to a single model, but shows consistently worse performance compared to Deep Ensembles [28, 12]. In particular, different predictions made by several forward passes with randomly generated masks seem to be overly correlated and strongly underestimate the variance. In this paper we postulate that the training procedure, which makes any combination of the weights possible, will tend to create uniformity between predictions. As opposed to Ensemble techniques, MC-Dropout techniques do not let multiple versions of the network actually be created, and the weights are forced to converge jointly.

2.4. Other Methods

Although Bayesian inference provides a natural way to assess uncertainty of model predictions, it is prohibitively expensive to directly apply these techniques to deep neural networks. A number of approximate inference techniques were developed for *Bayesian Neural Networks* (BNN) [23, 26, 24] that try to alleviate these problems, to name a few: *Bayes by Backprop* [2] uses Variational Inference to learn a parametric distribution over model weights that approximates the true posterior, *Laplace Approximation* [32] tries to find a Gaussian approximation to the posterior modes, MC-Dropout [8] could be seen as an approximate Bayesian procedure as well. Although these techniques are theoretically grounded, in practice Deep Ensembles often show significantly better performance [28, 1], in terms of both accuracy and quality of the resulting uncertainty estimates.

3. Method

In this section, we introduce Masksembles, an approach to uncertainty estimation that creates a continuum between single networks, deep ensembles, and MC-Dropout. It builds on the intuition that dropout-based methods can be seen as stochastic variants of ensembles [22], albeit ensembles with an infinite cardinality. We will argue that the main reason for MC-Dropout's poor performance is the high correlation between the ensemble elements that makes the overall predictions insufficiently diverse. Masksembles is designed to give control over the correlation between the elements of an ensemble and, hence, achieve a satisfactory compromise between reliable uncertainty estimation and acceptable computational cost.

3.1. Masksembles as Structured Dropout

Both Ensembles and MC-Dropout are ensemble methods, that is, they produce multiple predictions for a given input, and then use an aggregated measure such as variance or entropy as an uncertainty estimator. Formally, consider a dataset $\{\boldsymbol{x}_i, y_i\}_i$, where $\boldsymbol{x}_i \in \mathbb{R}^F$ are input features, and y_i are scalars or labels. For classification problems, we model conditional distribution $p(y|\boldsymbol{x};\boldsymbol{\theta})$ as a mixture

$$p(y|\boldsymbol{x};\boldsymbol{\theta}) = \frac{1}{N}\sum_{k=1}^{N} p(y|\boldsymbol{x};\boldsymbol{\theta}_k) \ ,$$

where $\boldsymbol{\theta}_k$ are the weights of element k of the ensemble, and N is the size of the ensemble.

For Deep Ensembles, individual models are independent and do not share any weights, and hence each $\boldsymbol{\theta}_k$ is a separate set of weights, trained fully independently. By contrast, for MC-Dropout there is a single shared $\boldsymbol{\theta}$, and individual models are obtained as $\boldsymbol{\theta}_k = \tilde{\boldsymbol{b}}_k^t \cdot \boldsymbol{\theta}$, where $\tilde{\boldsymbol{b}}_k^t \in \{0,1\}^{|\boldsymbol{\theta}|}$ are random binary masks which are re-sampled for each iteration t. The elements of $\tilde{\boldsymbol{b}}_k^t$ follow Bernoulli distribution parameterized by dropout probability p, and the parameters corresponding to the same network activations are dropped simultaneously. At inference time, a small number of predictions is produced by randomly sampling the masks. In practice, for any reasonable choice of p, the masks $\tilde{\boldsymbol{b}}_k^t$ will overlap significantly, which will tend to make predictions $p(y|\boldsymbol{x};\boldsymbol{\theta}_k)$ highly correlated, and thus could lead to underestimated uncertainty. Furthermore, because masks are resampled at each training iteration, each network unit needs to be able to form a coherent response with any other unit, which creates a mixing effect and leads to uniformity between the predictions from different masks.

To tackle these issues, we propose using a finite set of pre-defined binary masks, which are generated so that their overlap can be controlled. They are then used to drop corresponding network activations so that the resulting models

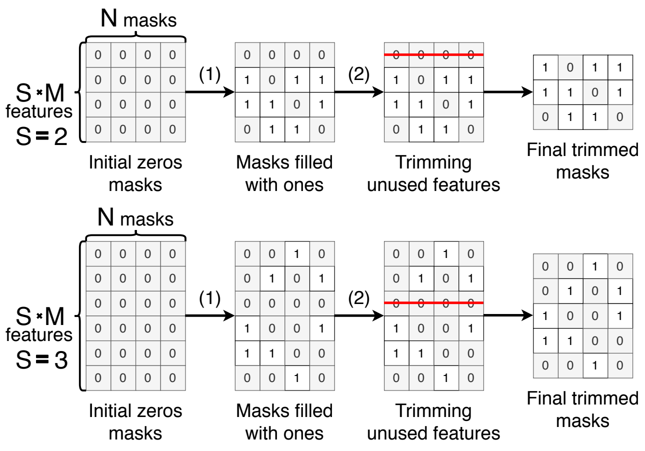

Figure 1. **Masks generation algorithm.** Here we use $\mathbf{N} = 4$, $\mathbf{M} = 2$, $\mathbf{S} = 2.0$ and 3.0. In the first case $\mathbf{M} \cdot \mathbf{S} = 4$ and in the second $\mathbf{M}{\cdot}\mathbf{S} = 6$. First, we generate 4 masks with $\mathbf{M}{\cdot}\mathbf{S}$ zeros each. Second, we randomly choose 2 positions for every of 4 masks and fill them with ones. Finally, we drop features that are not used in any mask. In the first case, the IoU is 0.44 and in the second 0.16.

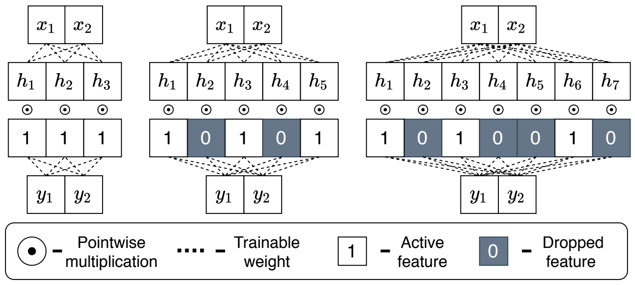

Figure 2. **Masksembles in a simple case.** Example of Masksembles MLP behavior for different values of $\mathbf{S} \in [1.0, 1.7, 2.3]$ ($\mathbf{N}$ and $\mathbf{M}$ are fixed): x_k - inputs, h_k - activations, y_k - outputs.

are sufficiently decorrelated and do not suffer from the mixing effect described above. In what follows, we describe the algorithm for mask generation, explain the way we apply generated masks to models and discuss the trade-offs it incurs.

3.2. Generating Masks

Generating masks with a controllable amount of overlap is central to Masksembles. Let us first introduce the key parameters of our mask generation process:

- **N** - number of masks.
- **M** - number of ones in each mask.
- **S** - scale that controls the amount of overlap given **N** and **M**.

The masks generation algorithm is summarized below. It starts by generating **N** vectors of all zeros of size $\mathbf{M} \times \mathbf{S}$. Then, in each of those vectors, it randomly sets **M** of these elements to be 1 in each vector. As depicted by Fig. 1, it then looks for positions that are all zero in the resulting vectors. These correspond to features that will be dropped.

For $\mathbf{S} = 1.0$, the algorithm then generates **N** masks of size **M** that will contain only ones and therefore fully overlapping. Conversely, setting **S** to large positive values will generate **N** masks that will have mostly zero values and therefore very few overlapping one values.

3.3. Inserting Masksembles Layer into Networks

We start with the very simple case depicted by Fig. 2. It shows a MLP with one hidden layer of size 3 and 2-dimensional inputs and outputs. We insert Masksembles layer right after the hidden layer and increase its size to $\mathbf{M} \times \mathbf{S} - D$ where D is the number of dropped features so that the sizes of the masks and the size of the layer match. Then at run-time we apply the mask values and ignore all components of the hidden layer for which they are zero.

Fig. 2 features 3 different configurations obtained by fixing both **M** and **N** — here, **N** is sufficiently large so that $D = 0$ — and changing the value of **S** to 1.0, 1.7, and 2.3, therefore increasing masks sizes and hidden size in range [3, 5, 7].

Applying such modification to the model creates a larger one that allows for multiple relatively uncorrelated predictions. This process extends naturally to MLPs of any dimension. It also extends to convolutional architectures without any significant modification. As in MC-dropout, we can drop entire channels [34] instead of individual activations. In other words, treating the whole channel as a single feature is equivalent to the fully-connected case discussed above.

In general, this means that we may have to adapt the size of network layers each time we change **M**, **N**, or **S**, which can be cumbersome. Fortunately, keeping $\mathbf{S} \times \mathbf{M}$ equal to the original channels number, 3 in the MLP example, makes it possible to apply Masksembles without any such change of the network configutration.

3.4. Parameters and Model Transitions

The chosen parameters **N**, **M** and **S** define several key model properties:

- *Total model size*: The total number of model parameters. In the above MLP example, this quantity is equal to $4 \cdot (\mathbf{M} \times \mathbf{S} - D)$ where D is the number of trimmed features during the mask generation procedure. This functional relationship is illustrated in Fig. 3.
- *Model capacity*: The number of activations that are used during one forward pass of the model, which is equal to **M**.
- *Masks IoU* average *Intersection over Union (IoU)* between generated masks. We derive approximation

for this quantity that appears to depend only on **S** : $\text{IoU}(\mathbf{S}) = 1/(2\mathbf{S}-1)$ and therefore **N**, **M** do not influence it, as show in Fig. 3.

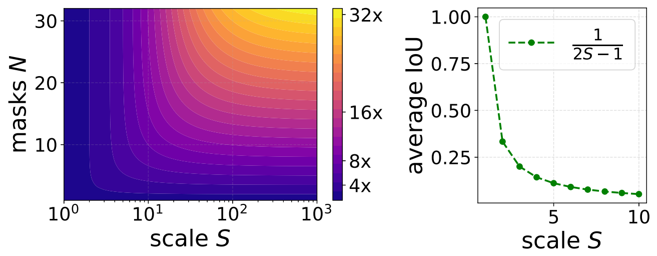

Figure 3. **Influence of the parameters. (Left)** Relative *Total model size* as a function of **S** and **N**, with **M** being fixed. We refer to size of single model ($\mathbf{S}=1, \mathbf{N}=1$) as 1x. **(Right)** IoU of the generated masks as a function of **S**. Note that it does not depend on neither **N** nor **M**.

In fact, Deep Ensembles and MC-Dropout can be both seen as extreme cases and Masksembles can transition smoothly from one to the other (Fig. 5).

Ensembles: When **S** goes to infinity, *Masks IoU* goes to zero and Masksembles start behaving like Ensembles that uses non-overlapping masks.

Dropout: As **N** increases, each individual mask configuration becomes less and less likely to be picked during training. In the limit where **N** goes to infinity, we reproduce the situation where each mask is seen only once, which is equivalent to MC-Dropout. In this case, the dropout rate would be $1 - \frac{1}{\mathbf{S}}$.

Transition: In Fig. 4, we use a simple classification problem to illustrate the span of behaviors Masksembles can cover. Specifically, we show that by reducing mask overlap, that is, decreasing the correlation between binary masks, yields progressively more diverse predictions on out-of-training examples and generates more ensemble-like behavior.

3.5. Implementation details

The only extra computation performed in Masksembles over single network is cheap tensor-to-mask multiplication and therefore comparing to convolutional and fully-connected layers this product induces negligible computational overhead. Furthermore, we rely on similar optimizations as in [35] to make our implementation more efficient.

4. Experiments

In this section, we first introduce the evaluation metrics which we use to quantitatively compare Masksembles against MC-Dropout and Ensembles. We then compare our method with the baselines on two broadly used benchmark datasets CIFAR10 [20] and ImageNet [4].

Note that, we did not use any complex data augmentation schemes, such as AugMix [17], Mixup [37] or adversarial training [10], in order to avoid entangling their effects with those of Masksembles. Ultimately, these techniques are complementary to ours and could easily be used together.

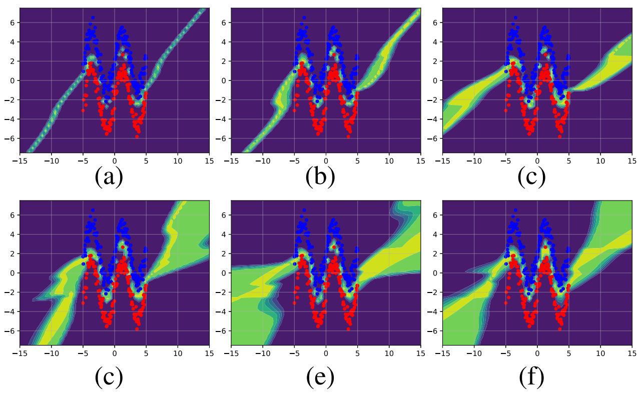

Figure 4. **Single Model to Ensembles transition via Masksembles**. The task is to classify the red vs blue data points drawn in the range $x \in [-5, 5]$ from two sinusoidal functions with added Gaussian noise. The background color depicts the entropy assigned by different models to individual points on the grid, low in blue and high in yellow. **(a)** Single model, **(b) - (e)** Masksembles models with $\mathbf{N} = 4$, $\mathbf{M} = 100$, $\mathbf{S} = [1.1, 2.0, 3.0, 10.0]$, **(f)** - Ensembles model. For a fair comparison with ensembles, we used a fixed value for $\mathbf{M} = 100$ when training Masksembles with different **S**, so that each Masksembles-model has the same 100 hidden-units capacity and the correlation between models decreases from (b) to (e). For high mask-overlap values, Masksembles behaves almost like a Single Model but starts behaving more and more like Ensembles as the overlap decreases.

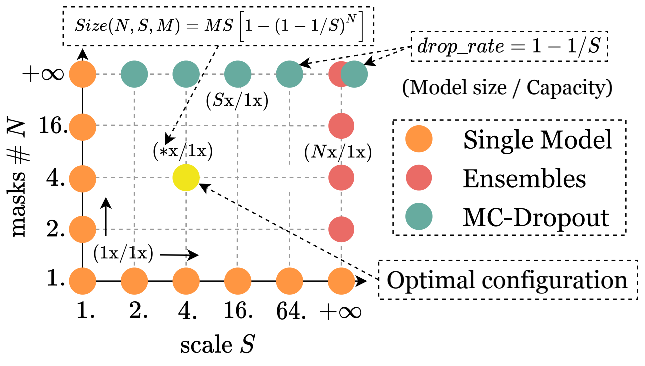

Figure 5. **Ensembles to MC-Dropout transition**. **Green** / **Red** points denote extreme cases that correspond to these models, whereas the **Yellow** point represents an optimal parameter choice one could choose for a specific computational cost vs performance trade-off. Here **M** is fixed while **N** and **S** vary.

4.1. Metrics

Given a classifier with parameters $\boldsymbol{\theta}$, let $p(y|\boldsymbol{x};\boldsymbol{\theta})$ be the predicted probability for a sample represented by features $\boldsymbol{x}$ to have a label y, and let $p_{gt}(y|\boldsymbol{x})$ denote the true conditional distribution over the labels given the features. We use several different metrics to analyze how close $p(y|\boldsymbol{x};\boldsymbol{\theta})$ is to $p_{gt}(y|\boldsymbol{x})$ and to compare our method to others, in terms of both accuracy and quality of uncertainty estimation.

Accuracy. For classification tasks, we use the standard classification accuracy, i.e. the percentage of correctly classified samples on the test set.

Entropy (ENT). It is one of the most popular measures used to quantify uncertainty [22, 25]. For classification tasks it is defined as

$$-\sum_{c=1}^{N_c} p(y=c|\boldsymbol{x};\boldsymbol{\theta}) \log p(y=c|\boldsymbol{x};\boldsymbol{\theta})$$

where N_c is the number of classes.

Expected Calibration Error (ECE). ECE quantifies the quality of uncertainty estimates specifically for in-domain uncertainty. A model is considered well-calibrated if its predicted distribution over labels $p_(y|\boldsymbol{x};\boldsymbol{\theta})$ is close to the true distribution $p_{gt}(y|\boldsymbol{x})$. As in [11], we define ECE to be the average discrepancy between the predicted and real data distribution:

$$\mathbb{E}_{\boldsymbol{x},y\sim p_{gt}} \left|p(y|\boldsymbol{x};\boldsymbol{\theta}) - p_{gt}(y|\boldsymbol{x})\right| \ ,$$

which is a common metric for calibration evaluation.

Out-of-Distribution Detection (OOD ROC / PR). Domain shift occurs when the features in the training data do not follow the same probability distribution as those in the test distribution. Detecting out-of-distribution samples is an important task and uncertainty estimation can be used for this purpose under the assumption that our model returns higher uncertainty estimates for such samples. To this end, we use the uncertainty measure produced by our models as a classification score that determines whether a sample is in-domain or not. We then apply standard detection metrics, ROC and PR AUCs, to quantify the performance of our model.

Model Size This corresponds to the *Total model size* defined in Section 3.4. A major motivation for Masksembles is to match the performance of Deep Ensembles while using a smaller model that requires significantly less memory. We use total number of weights that parameterize our models as a proxy for that. In addition to the model size, we also report the total GPU time used to train any particular model.

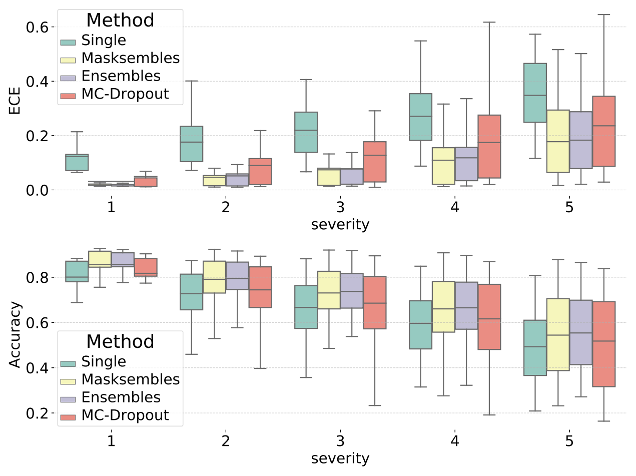

Figure 6. **CIFAR-10 results on corrupted images. Masksembles** and **MC-Dropout** models with tuned (on validation set) dropout rate parameter values, **Ensemble** of independently trained models and **Single** model. All approaches, except Single, use 4 models whose predictions are averaged.

4.2. CIFAR10

CIFAR10 is among the most popular benchmarks for uncertainty estimation. The dataset is relatively small-scale, and very large models tend to easily overfit the data. We therefore picked the Wide-ResNet-16-4 [36] for our experiments and used training procedures similar to the original paper. Following [8], we place dropout layers before all the convolutional and fully connected layers in the MC-Dropout model. Masksembles layers are added in exactly the same way.

Accuracy and ECE. One of the standard ways to assess the robustness of a model is to evaluate the dependency of its performance with respect to inputs perturbations [16, 28]. In our experiments we investigate the behaviour of all the considered models under the influence of domain-shifts induced by typical sources of noise.

Following [35, 28], we evaluate model accuracy and ECE on a corrupted version of CIFAR10 [16]. Namely, we consider 19 different ways to artificially corrupted the images and 5 different levels of severity for each of those corruptions.

We train all the models on the original uncorrupted CIFAR10 training set and test them both on the original images and their corrupted versions. In Table 1, we report our results on the original uncorrupted images while and in Fig. 6 the results on the noisy ones. Each box represents the first and third quantiles of accuracy and ECE while the error bars denote the minimum and maximum values. We tested four different approaches, that is, a single network, MC-Dropout, Ensembles, and our Masksembles. Unsurprisingly, as the severity of the perturbations increases, using

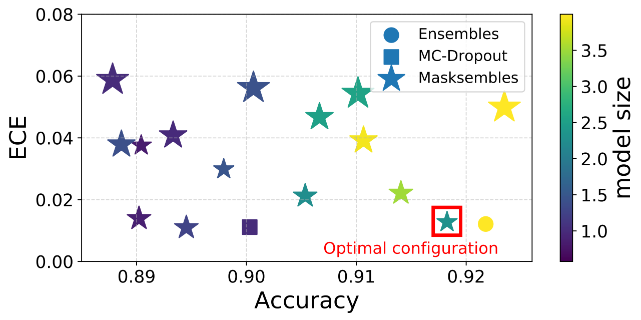

Figure 7. **Spanning the space of behaviors**. Models in the bottom right corner are better. The color represents the relative size of the model compared to a single model. The size of Masksembles markers denote mask overlap between 1.0 and 0.2.

a single network is the least robust approach and degrades fastest. Masksembles performs on par with Ensembles and consistently outperforms MC-dropout, even though we extensively tuned MC-Dropout for the best possible performance by doing a grid search on the dropout rate (which turned out to be $p \approx 0.1$).

In these experiments, Masksembles were trained with a fixed number of $\mathbf{N} = 4$ masks. We varied both $\mathbf{S}$ in the range $[1.0, 5.0]$ and $\mathbf{M}$ in the range $[0.3C, 2.0C]$, where C is the number of channels in the original model. Fig. 7 depicts the resulting range of behaviors. The 2D coordinates of the markers depict their accuracy and ECE, their colors the corresponding model size, and the size of the star markers denotes the value of the mask-overlap. For comparison purposes, we also display MC-dropout and Ensembles results in a similar manner, simply replacing the star by a square and a circle respectively. As can be seen, the optimal Masksemble configuration depicted by the green star in the lower right corner delivers performances that are similar to those of Ensembles for a model size that is almost half the size. This is the one we also used to the experiments depicted by Fig. 6.

In Fig. 8, we chose the Masksembles parameters so that the model size always remains the same: We fix number of masks $\mathbf{N} = 4$ and vary $\mathbf{S}$ and $\mathbf{M}$ so that the *total model size* remains constant, which means that a lower mask overlap results in a lower *capacity* for each individual model. This procedure is important in practice because it enables us to take large network, such as ResNet, and add Masksembles layers as we would add Dropout layer *without* having to change the rest of the architecture.

For this set of experiments we obtain a consistent improvement in calibration quality by reducing mask overlap and achieve ensembles-like performance with $\mathbf{S} = 5$.

OOD ROC and PR. For OOD task, we followed the evaluation protocol of [3]. Namely, we trained our models on CIFAR10, and then used CIFAR10 test images as

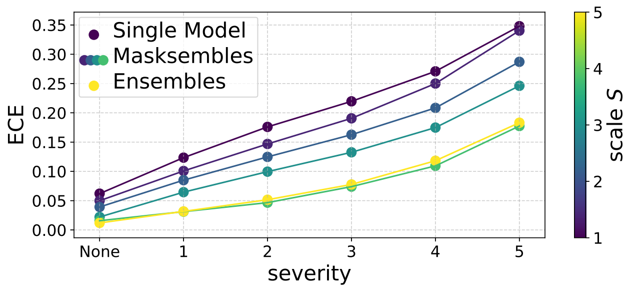

Figure 8. **CIFAR10 ECE**. ECE as a function of severity of image corruptions. Each Masksembles curve corresponds to a different $\mathbf{S}$. In this experiment we force $\mathbf{M} \times \mathbf{S}$ to remain constant and equal to the original Wide-ResNet channels number and we vary $\mathbf{S}$ in the range $[1., 1.1, 1.4, 2.0, 3.0]$. Larger values of $\mathbf{S}$ correspond to more ensembles-like behavior.

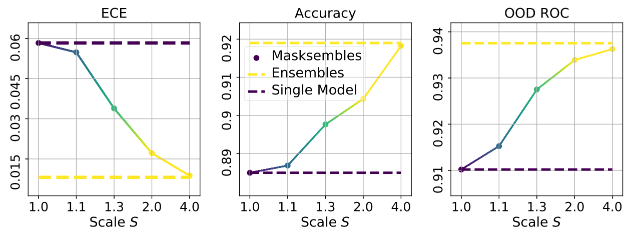

Figure 9. **Fixed capacity for CIFAR10**. ECE, Accuracy and OOD as functions of the scale parameter $\mathbf{S}$ with fixed $\mathbf{M}$ and $\mathbf{N}$. The colors represent different masks overlap values.

	Single	MC-D	MaskE	NaiveE
Accuracy	0.89	0.90	**0.92**	**0.92**
ECE	0.06	**0.01**	**0.01**	**0.01**
Size	1x	1x	2.3x	4x
Time	10m	12m	16m	40m
OOD ROC	0.91	0.92	**0.94**	**0.94**
OOD PR	0.94	0.95	**0.96**	**0.96**

Table 1. **CIFAR10 results.** Each model uses 4 samples during inference and we used the uncorrupted versions of the images.

our in-distribution samples and images from the SVHN dataset [27] (which belong to classes that are not in CIFAR10), as our out-of-distribution samples. We report our OOD results in the two bottom rows of Table 1. Again, performance of our method is similar to that of Ensembles at a fraction of the computational cost, and significantly better than that of a single network and MC-dropout.

Fixed Capacity Model. In order to provide clear evidence for Single Model $\leftrightarrow$ Ensembles transition of Masksembles we perform fixed capacity experiments and report the results in Fig. 9. Similarly to our previous experiments we set $\mathbf{N} = 4$ and vary the scale parameter in range $\mathbf{S} \in [1.0, 4.0]$, while keeping parameter $\mathbf{M}$ fixed. These results confirm that Masksembles are able to span the entire spectrum of behaviours between different models: the

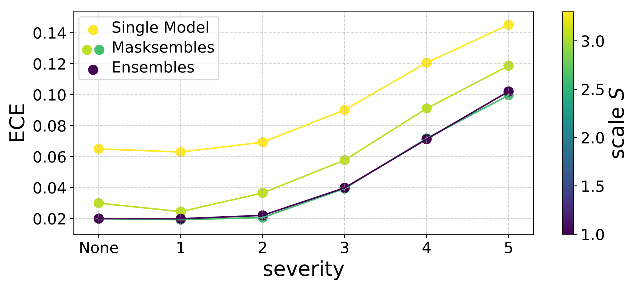

Figure 10. **ImageNet ECE**. ECE as a function of severity of image corruptions. Each Masksembles curve corresponds to a different **S**. In this experiment we force **M** × **S** to be constant and equal to original ResNet-50 channels number.

key metrics gradually improve as **S** is increased, and the Masksembles's behavior (solid line) clearly transitions from Single Model to Ensembles behavior (dashed lines).

4.3. ImageNet

The ImageNet [4] is another dataset that is widely used to assess the ability of neural networks to produce uncertainty estimates. For this dataset, we rely on the well-known ResNet-50 [15] architecture as a base model, using the setup similar to that in [1]. Similarly to our CIFAR10 experiments, for the MC-Dropout and Masksembles models, we introduce dropout (masking) layers before every convolutional layer starting from `stage3` layers.

We followed exactly the same evaluation protocol as in Section 4.2 and report our results on the original images in Table 2 and on the corrupted ones in Fig. 11. In Table 2 and Fig. 10 we use the same procedure as for CIFAR10 to select Masksembles parameters so that the *total model size* remains fixed.

For this dataset, Masksembles demonstrate their best performance for a choice of parameters correponding to a *Masks IoU* of 0.5. Similarly to our results on CIFAR10, the performance of Masksembles both in terms of accuracy and ECE are very close to those of Ensembles and significantly better than those of MC-Dropout. Note that, our method achieves this results with a training time and memory consumption that is 4 (four) times smaller than that of Ensembles, and nearly the same as that of a single network. In terms of calibration quality on corrupted images, we achieve Ensemble-like performance starting from $\mathbf{S} = 2$ and up.

5. Conclusion

In this work, we introduced Masksembles, a new approach to uncertainty estimation in deep neural networks. Instead of using a fixed number of independently trained models as in Ensembles, or drawing a new set of random binary masks at each training step as in MC-Dropout, Masksembles predefines a pool of binary masks with a controllable overlap and stochastically iterates through them. By changing the parameters that control the mask generatio process, we can span a range of behaviors between those of MC-Dropout and Ensembles. This allows us to identify model configurations that provide a useful trade-off between the high-quality uncertainty estimates of Deep Ensembles at a high computational cost, and the lower performance of MC-Dropout at a lower computational cost. In fact, our experiments demonstrate that we can achieve the performance on par with that of Deep Ensembles at a fraction of the cost. Because Masksembles is easy to implement, we believe that it can serve as a drop-in replacement for MC-Dropout. In future work, we will explore ways to accelerate Masksembles by exploiting the structure of the masks, and apply our method to tasks that require a scalable uncertainty estimation method, in particular reinforcement learning and Bayesian optimization.

	Single	Masksembles			NaiveE	MC-DP
Acc.	0.71	0.72	0.71	0.70	0.74	0.69
ECE	0.07	0.03	0.02	0.02	0.02	0.03
IoU	1.0	0.7	0.3	0.2	-	-
Time	50h	55h	60h	70h	200h	80h

Table 2. **ImageNet results.** Accuracy and ECE results for **Single**, **Masksembles** models using masks overlapping values (0.7, 0.3, 0.2), **Ensembles**, and **MC-Dropout**. All the models have the same size as the **Single model**, except for **Ensembles** which is 4x times larger.

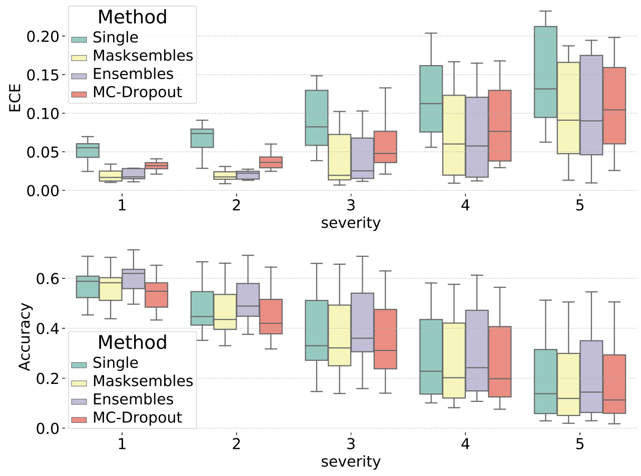

Figure 11. **ImageNet results on corrupted images. Masksembles** and **MC-Dropout** with tuned dropout rate parameter values, **Ensemble** of independently trained models and **Single** model. All approaches, except **Single**, include 4 models, their predictions are averaged.

References

[1] Arsenii Ashukha, Alexander Lyzhov, Dmitry Molchanov, and Dmitry Vetrov. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. arXiv preprint arXiv:2002.06470, 2020.

[2] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. arXiv preprint arXiv:1505.05424, 2015.

[3] Kamil Ciosek, Vincent Fortuin, Ryota Tomioka, Katja Hofmann, and Richard Turner. Conservative uncertainty estimation by fitting prior networks. In International Conference on Learning Representations, 2019.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009.

[5] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? Structural safety, 31(2):105–112, 2009.

[6] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. arXiv preprint arXiv:1912.02757, 2019.

[7] Yarin Gal. Uncertainty in deep learning. University of Cambridge, 1(3), 2016.

[8] Y. Gal and Z. Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In International Conference on Machine Learning, pages 1050–1059, 2016.

[9] T. Garipov, P. Izmailov, D. Podoprikhin, D. Vetrov, and A. Wilson. Loss Surfaces, Mode Connectivity and Fast Ensembling of DNNs. In Advances in Neural Information Processing Systems, 2018.

[10] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572, 2014.

[11] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. arXiv preprint arXiv:1706.04599, 2017.

[12] Fredrik K Gustafsson, Martin Danelljan, and Thomas B Schon. Evaluating scalable bayesian deep learning methods for robust computer vision. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pages 318–319, 2020.

[13] L. K. Hansen and P. Salamon. Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(10):993–1001, 1990.

[14] T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning: Data Mining, Inference and Prediction. Springer, 2009.

[15] 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.

[16] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. arXiv preprint arXiv:1903.12261, 2019.

[17] Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. arXiv preprint arXiv:1912.02781, 2019.

[18] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. arXiv preprint arXiv:1704.00109, 2017.

[19] 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.

[20] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[21] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles. In Advances in Neural Information Processing Systems, 2017.

[22] 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.

[23] David JC MacKay. Bayesian methods for adaptive models. PhD thesis, California Institute of Technology, 1992.

[24] David JC MacKay. A practical bayesian framework for backpropagation networks. Neural computation, 4(3):448–472, 1992.

[25] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. In Advances in Neural Information Processing Systems, pages 7047–7058, 2018.

[26] Radford M Neal. Bayesian learning for neural networks, volume 118. Springer Science & Business Media, 2012.

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

[28] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In Advances in Neural Information Processing Systems, pages 13991–14002, 2019.

[29] Tim Pearce, Alexandra Brintrup, Mohamed Zaki, and Andy Neely. High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. In International Conference on Machine Learning, pages 4075–4084. PMLR, 2018.

[30] Michael P Perrone and Leon N Cooper. When networks disagree: Ensemble methods for hybrid neural networks. Technical report, BROWN UNIV PROVIDENCE RI INST FOR BRAIN AND NEURAL SYSTEMS, 1992.

[31] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. Dataset shift in machine learning. The MIT Press, 2009.

[32] Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In 6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings, volume 6. International Conference on Representation Learning, 2018.

[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15:1929–1958, 2014.

[34] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient Object Localization Using Convolutional Networks. In Conference on Computer Vision and Pattern Recognition, pages 648–656, 2015.

[35] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. arXiv preprint arXiv:2002.06715, 2020.

[36] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. arXiv preprint arXiv:1605.07146, 2016.

[37] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412, 2017.