Estimation efficace des incertitudes avec Packed-Ensembles

Olivier Laurent, Adrien Lafage, Enzo Tartaglione, Geoffrey Daniel, Jean-Marc Martinez, Andrei Bursuc, Gianni Franchi

Estimation efficace des incertitudes avec Packed-Ensembles

Olivier Laurent[1,2,*] Adrien Lafage[2,*] Enzo Tartaglione[3] Geoffrey Daniel[1]
Jean-Marc Martinez[1] Andrei Bursuc[4] Gianni Franchi[2]

[1] SGLS, CEA, Université Paris-Saclay
[2] U2IS, ENSTA Paris, Institut Polytechnique de Paris
[3] LTCI, Télécom Paris, Institut Polytechnique de Paris
[4] Valeo.ai

adrien.lafage@ensta-paris.fr

Résumé

Deep Ensembles (DE) est une des approches principales pour obtenir d'excellentes performances sur des tâches de classification et de détection de données hors distribution. En pratique cependant, ces ensembles et leurs membres sont généralement petits, et donc peu performants, car limités par les contraintes matérielles. Nous proposons Packed-Ensembles (PE), une méthode pour définir et entraîner des ensembles plus légers en modulant la dimension des espaces de représentation. Nous utilisons des convolutions groupées pour paralléliser l'ensemble en une unique structure et ainsi améliorer les temps d'entraînement et d'inférence. PE a la capacité de fonctionner avec une empreinte mémoire équivalente à celle d'un réseau de neurones classique. Au travers d'études approfondies, nous montrons que PE préserve les propriétés de DE, telles que la diversité, et atteint des résultats similaires sur toutes les métriques de précision et de quantification d'incertitude. Le code associé au papier est disponible sur https://github.com/ENSTA-U2IS/torch-uncertainty.

Mots Clés

Méthodes d'ensembles, Estimation des incertitudes, Détection d'OOD

Abstract

Deep Ensembles (DE) are a prominent approach to achieve excellent performance on key metrics such as accuracy, calibration, uncertainty estimation, and out-of-distribution detection. However, hardware limitations of real-world systems constrain to smaller ensembles and lower-capacity networks, significantly deteriorating their performance and properties. We introduce Packed-Ensembles (PE), a strategy to design and train lightweight structured ensembles by carefully modulating the dimension of their encoding space. We leverage grouped convolutions to parallelize the ensemble into a single common backbone and forward pass to improve training and inference speeds. PE is designed to work under the memory budget of a single standard neural network. Through extensive studies, we show that PE faithfully preserve the properties of DE, e.g., diversity, and match their performance in terms of accuracy, calibration, out-of-distribution detection, and robustness to distribution shift. We make our code available at https://github.com/ENSTA-U2IS/torch-uncertainty.

Keywords

Efficient Ensembling, Uncertainty Quantification, OOD Detection

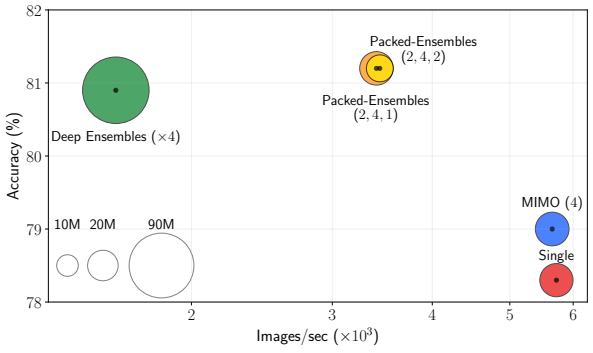


Figure 1 – **Evaluation of computation cost and performance trade-offs for multiple uncertainty quantification techniques on CIFAR-100.** The y-axis shows the accuracy, and the x-axis shows the inference time in images per second. The circle area is proportional to the number of parameters. Optimal approaches should be closer to the top-right corner. Packed-Ensembles strikes a good balance between predictive performance and speed.

[*] Equal contribution

1 Introduction

Real-world safety-critical machine learning decision systems such as autonomous driving [40; 48] impose exceptionally high reliability and performance requirements over a broad range of metrics: accuracy, calibration, robustness to distribution shifts, uncertainty estimation, and computational efficiency under limited hardware resources. Although their performance across all key dimensions has dramatically improved in the last years, vanilla Deep Neural Networks (DNNs) still have several shortcomings, notably overconfidence in correct and wrong predictions [55; 16; 21]. Deep Ensembles (DE) [36] arise as a prominent approach to address these challenges by leveraging predictions from multiple high-capacity neural networks. By averaging predictions or voting, DE achieves high accuracy and robustness since potentially unreliable predictions are exposed via the disagreement between individuals. Thanks to the simplicity and effectiveness of the ensembling strategy [8], DE have become widely used and dominate performance across various benchmarks [58; 17].

DE tick all boxes in the requirements list for real-world applications, except for computational efficiency. In fact, DE are, in practice, computationally demanding (required memory storage, number of operations, inference time) for training and testing as their costs grow linearly with the size of the individuals.

Their computational costs are, therefore, prohibitive under tight hardware constraints. This limitation of DE has inspired numerous approaches proposing computationally efficient alternatives: multi-head networks [39; 5], architectures with ensemble-imitating layers [66; 19; 61], multiple forwards on different weight subsets of the same network [14; 9], ensembles of smaller networks [33; 44], computing ensembles from a single training run [28; 15], efficient Bayesian Neural Networks [47; 12]. These approaches typically improve storage usage, train cost, or inference time at the cost of lower accuracy and lower diversity in the predictions.

An essential property of ensembles that enables improved predictive uncertainty estimation is related to the diversity in its predictions. [60] have shown that the independence of individuals is critical to the success of ensembling. [11] argue that the diversity of DE due to randomness from weight initialization, data augmentation and batching, and stochastic gradient updates, is superior to various other efficient ensembling approaches, despite their predictive performance boosts.

Few works manage to mirror this property of DE in a computationally efficient manner close to a single DNN (in terms of memory usage, number of forward passes, image throughput, etc.).

In this work, we aim to design a DNN architecture that closely mimics properties of ensembles, in particular, having a set of independent networks, in a computationally efficient manner. Previous works propose ensembles composed of small models [33; 44] and achieve performances comparable to a single large model. We build upon this idea and devise a strategy based on small networks trying to match the performance of an ensemble of large networks. To this end, we leverage *grouped convolutions* [35] to delineate multiple subnetworks within the same network. The parameters of each subnetwork are not shared across subnetworks, leading to independent smaller models. This method enables fast training and inference times while predictive uncertainty quantification is close to DE (Figure 1). In summary, our contributions are the following:

— We propose *Packed-Ensembles* (PE), an efficient ensembling architecture relying on grouped convolutions, as a formalization of structured sparsity for Deep Ensembles;

— We extensively evaluate PE regarding accuracy, calibration, OOD detection, and dataset shift on classification and regression tasks. We demonstrate that PE achieves state-of-the-art predictive uncertainty quantification.

— We thoroughly study and discuss the properties of PE (diversity, sparsity, stability, behavior of subnetworks) and release our PyTorch implementation.

2 Background

In this section, we present the formalism for this work and offer a brief background on grouped convolutions and ensembles of DNNs.

2.1 Background on convolutions

The convolutional layer [37] consists of a series of cross-correlations between feature maps $\boldsymbol{h}^j \in \mathbb{R}^{C_j \times H_j \times W_j}$ regrouped in batches of size B and a weight tensor $\boldsymbol{\omega}^j \in \mathbb{R}^{C_{j+1} \times C_j \times s_j^2}$ with C_j, H_j, W_j three integers that represent the number of channels, the height and the width of $\boldsymbol{h}^j$ respectively. C_{j+1} and s_j are also two integers corresponding respectively to the number of channels of $\boldsymbol{h}^{j+1}$ (the output of the layer) and the kernel size. Finally, j is the layer's index and will be fixed in the following formula. The supplementary material details the main notations in Table 3. The bias of convolution layers will be omitted in the following for simplicity. Hence the output value of the convolution layer, denoted $\circledast$, is:

$$\begin{aligned} \boldsymbol{z}^{j+1}(c,:,:) &= (\boldsymbol{h}^j \circledast \boldsymbol{\omega}^j)(c,:,:) \\ &= \sum_{k=0}^{C_j-1} \boldsymbol{\omega}^j(c,k,:,:) \star \boldsymbol{h}^j(k,:,:) \end{aligned} \tag{1}$$

where $c \in [\![0, C_{j+1}-1]\!]$ is the index of the considered channel of the output feature map, $\star$ is the classical 2D cross-correlation operator, and $\boldsymbol{z}^j$ is the pre-activation feature map such that $\boldsymbol{h}^j = \phi(\boldsymbol{z}^j)$ with ϕ an activation function.

To embed an ensemble of subnetworks, we leverage grouped convolutions, already used in ResNext [70] to train several DNN branches in parallel. The grouped

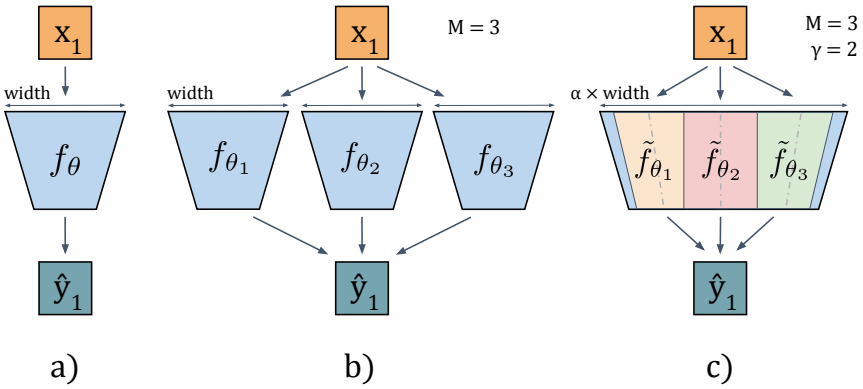


Figure 2 – **Overview of the considered architectures:** (*left*) baseline vanilla network; (*center*) Deep Ensembles; (*right*) Packed-Ensembles-$(\alpha, M = 3, \gamma = 2)$.

convolution operation with γ groups and weights $\boldsymbol{\omega}_\gamma^i \in \mathbb{R}^{C_{j+1} \times \frac{C_j}{\gamma} \times s_j^2}$ is given in equation 2, γ dividing C_j for all layers. Any output channel c is produced by a specific group (set of filters), identified by the integer $\left\lfloor \frac{\gamma c}{C_{j+1}} \right\rfloor$, which only uses $\frac{1}{\gamma}$ of the input channels.

$$\begin{aligned} \boldsymbol{z}^{j+1}(c,:,:) &= (\boldsymbol{h}^j \circledast \boldsymbol{\omega}_\gamma^j)(c,:,:) \\ &= \sum_{k=0}^{\frac{C_j}{\gamma}-1} \boldsymbol{\omega}_\gamma^j(c,k,:,:) \star \boldsymbol{h}^j\left(k + \left\lfloor \frac{\gamma c}{C_{j+1}} \right\rfloor \frac{C_j}{\gamma}, :, :\right) \end{aligned} \tag{2}$$

The grouped convolution layer is mathematically equivalent to a classical convolution where the weights are multiplied element-wise by the binary tensor $\text{mask}_m \in \{0,1\}^{C_{j+1} \times C_j \times s_j^2}$ such that $\text{mask}_m^j(k,l,:,:) = 1$ if $\left\lfloor \frac{\gamma l}{C_j} \right\rfloor = \left\lfloor \frac{\gamma k}{C_{j+1}} \right\rfloor = m$ for each group $m \in [\![0, \gamma - 1]\!]$. The complete layer mask is finally defined as $\text{mask}^j = \sum_{m=0}^{\gamma-1} \text{mask}_m^j$ and the grouped convolution can therefore be rewritten as:

$$\boldsymbol{z}^{j+1} = \boldsymbol{h}^j \circledast \left(\boldsymbol{\omega}^j \circ \text{mask}^j\right) \tag{3}$$

where $\circ$ is the Hadamard product.

2.2 Background on Deep Ensembles

For a classification problem, let us define a dataset $\mathcal{D} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{|\mathcal{D}|}$ containing $|\mathcal{D}|$ pairs of samples $\boldsymbol{x}_i \in \mathbb{R}^{H_j \times W_j}$ and one-hot-encoded labels $\boldsymbol{y}_i \in \mathbb{R}^{N_C}$ modeled as the realisation of a joint distribution $\mathcal{P}_{(X,Y)}$ where N_C is the number of classes in the dataset. The input data $\boldsymbol{x}_i$ is processed via a neural network $f_{\boldsymbol{\theta}}$ which is a parametric probabilistic model such that $\hat{\boldsymbol{y}}_i = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = P(Y = \boldsymbol{y}_i | X = \boldsymbol{x}_i; \boldsymbol{\theta})$. This approach consists in considering the prediction $\hat{\boldsymbol{y}}$ as parameters of a Multinoulli distribution.

To improve the accuracy of predictions, the accuracy of the predicted uncertainties, and the detection of OOD samples, [36] have proposed to ensemble M randomly initialized DNNs as a large predictor called Deep Ensembles. These ensembles can be seen as a discrete approximation of the intractable bayesian marginalization on the weights, according to [69]. If we note $\{\boldsymbol{\theta}_m\}_{m=0}^{M-1}$ the set of trained weights for the M DNNs, Deep Ensembles consists in averaging the predictions of these M DNNs as in equation equation 4.

$$P(\boldsymbol{y}_i | \boldsymbol{x}_i, \mathcal{D}) = \frac{1}{M} \sum_{m=0}^{M-1} P(\boldsymbol{y}_i | \boldsymbol{x}_i, \boldsymbol{\theta}_m) \tag{4}$$

3 Packed-Ensembles

This section presents how to train multiple subnetworks using grouped convolution efficiently. Then, we explain how our new architectures are equivalent to training several networks in parallel.

3.1 Revisiting Deep Ensembles

Although Deep Ensembles provide undisputed benefits, they also come with the significant drawback that the training time and the memory usage in inference increase linearly with the number of networks. To alleviate these problems, we propose assembling small subnetworks, which are DNNs with fewer parameters. Moreover, while ensembles to this day have been trained sequentially, we suggest leveraging grouped convolutions to massively accelerate their training and inference computations thanks to their smaller size. The propagation of grouped convolutions with M groups, M being the number of subnetworks in the ensemble, ensures that the subnetworks are trained independently while dividing their encoding dimension by a factor M.

More details on the usefulness of grouped convolutions to train ensembles can be found in subsection 3.3.

To create Packed-Ensembles (illustrated in Figure 2), we build on small subnetworks but compensate for the dramatic decrease of the model capacity by multiplying the width by the hyperparameter α, which can be seen as an expansion factor. Hence, we propose *Packed-Ensembles*-$(\alpha, M, 1)$ as a flexible formalization of ensembles of small

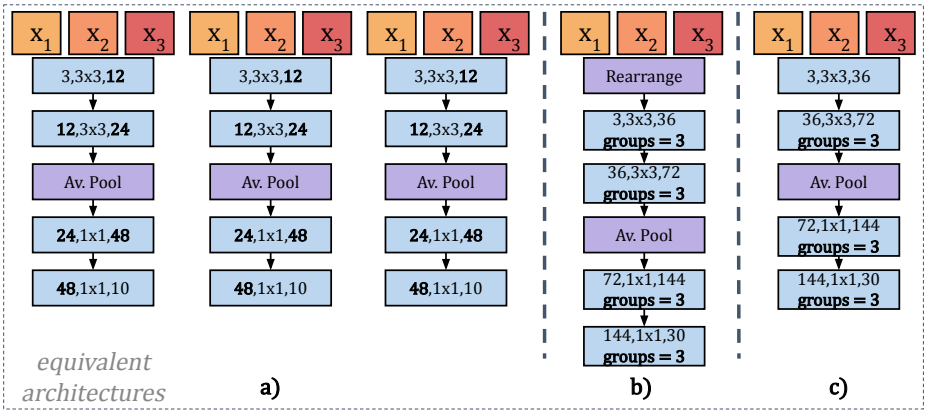


Figure 3 – **Equivalent architectures for Packed-Ensembles.** **(a)** corresponds to the first sequential version, **(b)** to the version with the rearrange operation and grouped convolutions and **(c)** to the final version beginning with a full convolution.

subnetworks. For an ensemble of M subnetworks, Packed-Ensembles-$(\alpha, M, 1)$ therefore modifies the encoding dimension by a factor $\frac{\alpha}{M}$ and the inference of our ensemble is computed with the following formula:

$$\hat{\boldsymbol{y}} = \frac{1}{M} \sum_{m=0}^{M-1} P(\boldsymbol{y}|\boldsymbol{\theta}_{\alpha,m}, \boldsymbol{x}) \text{ with } \boldsymbol{\theta}_{\alpha,m} = \{\boldsymbol{\omega}_{\alpha}^{j} \circ \text{mask}_{m}^{j}\}_{j}, \tag{5}$$

where $\boldsymbol{\omega}^{j,\alpha}$ is the weight of the layer j of dimension $(\alpha C_{j+1}) \times (\alpha C_j) \times s_j^2$.
In the following, we add another hyperparameter γ corresponding to the number of groups of each subnetwork of the Packed-Ensembles, creating another level of sparsity. These groups are also called "subgroups" and are applied to the different subnetworks. Formally, we denote our technique *Packed-Ensembles*-(α, M, γ), where the hyperparameters are in the parentheses. In this work, we consider the case of a constant number of subgroups across the layers; therefore, γ divides αC_j for all j.

3.2 Computational cost

In a convolutional setting, the number of parameters in a layer involving C_j input channels, C_{j+1} output channels, kernels of size s_j and γ subgroups is equal to $M \times \left[\frac{\alpha C_j}{M} \frac{\alpha C_{j+1}}{M} s_j^2 \gamma^{-1}\right]$.
The same formula applies to dense layers as 1×1 convolutions. Two specific cases emerge whenever the architectures of the subnetworks are fully convolutional or dense. If $\alpha = \sqrt{M}$, the number of parameters in the ensemble is equal to the number of parameters in a single model. With $\alpha = M$, each subnetwork corresponds to a single model (and their ensemble is therefore equivalent in size to Deep Ensembles).

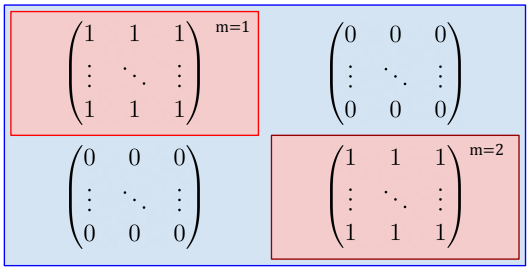
3.3 Implementation details

In this paper, we propose a simple way of designing efficient ensemble convolutional layers using grouped convolutions. To take advantage of the parallelization capabilities of GPUs in training and inference, we replace the sequential training architecture, **(a)** in Figure 3, with the parallel implementations **(b)** and **(c)**. Figure 3 summarizes different equivalent architectures for a simple ensemble of $M = 3$ DNNs with three convolutional layers and a final dense layer (equivalent to a 1×1 convolution) with $\alpha = \gamma = 1$.
In **(b)**, we propose to stack the feature maps on the channel dimension (denoted as the "rearrange"[1] operation). This yields a feature map $\boldsymbol{h}^j$, of size $M \times C_i \times H_j \times W_j$ regrouped by batches of size only $\frac{B}{M}$, with B the batch size of the ensemble. One solution to keep the same batch size is to repeat the batch M times so that its size equals B after the rearrangement. Using convolutions with M groups and γ subgroups per subnetwork, each feature map is convoluted separately by each subnetwork and yields its own independent output. Grouped convolutions are propagated until the end to ensure that gradients stay independent between subnetworks. Other operations, such as Batch Normalization [30], can be applied directly as long as they can be grouped or have independent actions on each channel. Figure 4a illustrates the mask used to code Packed-Ensembles in the case where $M = 2$. Similarly, Figure 4b shows the mask with $M = 2$ and $\gamma = 2$.
Finally, **(b)** and **(c)** are also equivalent. It is indeed possible to replace the "rearrange" operation and the first grouped convolution with a standard convolution if the same images are to be provided simultaneously to all the subnetworks. We confirm in Appendix F that this procedure is not detrimental to the ensemble's performance, and we take advantage of this property to provide this final optimization and simplification.

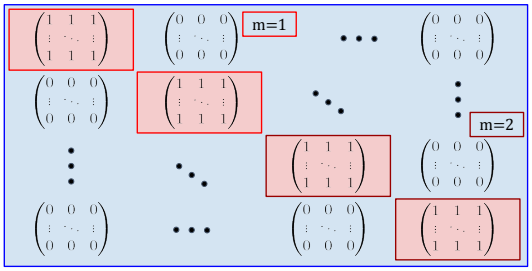
4 Experiments

To validate the performance of our method, we conduct experiments on classification tasks and measure the influence

1. See https://einops.rocks/api/rearrange/



(a) $M = 2, \gamma = 1$



(b) $M = 2, \gamma = 2$

Figure 4 – Diagram representation of a subnetwork mask: $\texttt{mask}^j$, with $M = 2$, j an integer corresponding to a fully connected layer

of the parameters α and γ. Regression tasks are detailed in Appendix N.

4.1 Datasets and architectures

First, we demonstrate the efficiency of Packed-Ensembles on CIFAR-10 and CIFAR-100 [34], showing how the method adapts to different data complexities. As we propose to replace a single model architecture with several subnetworks, we study the behavior of Packed-Ensembles on architectures with various sizes: ResNet-18, ResNet-50 [20], and Wide ResNet28-10 [72]. We compare it against Deep Ensembles [36] and three other approximated ensembles from the literature: BatchEnsemble [66], MIMO [19], and Masksembles [9].

Secondly, we report our results for Packed-Ensembles on ImageNet, which we compare against all baselines. We run experiments with ResNet-50 and ResNet-50x4. All training runs are started from scratch.

Metrics. We evaluate the overall performance of the models in classification tasks using the accuracy (Acc) in % and Negative Log-Likelihood (NLL). We choose the classical Expected Calibration Error (ECE) [51] for the calibration of uncertainties [2] and measure the quality of the OOD detection using the Areas Under the Precision/Recall curve (AUPR) and Under the operating Curve (AUC), as well as the False Positive Rate at 95% recall (FPR95), all expressed in %, similarly to [23].

We use accuracy as the validation criterion (i.e., the final trained model is the one with the highest accuracy). During inference, we average the softmax probabilities of all subnetworks and consider the index of the maximum of the output vector to be the predicted class of the ensemble. We define the prediction confidence as the value of this maximum (also called maximum softmax probability).

For the out-of-distribution detection tasks on CIFAR-10 and CIFAR-100, we use the SVHN dataset [54] as an out-of-distribution dataset and transform the initial classification problem into a binary classification between in-distribution (ID) data and OOD data using the maximum softmax probability as OOD criterion. We discuss the different OOD criteria in appendix E. For ImageNet, we use two out-of-distribution datasets: ImageNet-O (IO) [25] and Texture (T) [65], and use the Mutual Information (**MI**) as a criterion for the ensembles techniques (see Appendix E for details on **MI**) and the maximum softmax probability for the single model and MIMO. To measure the robustness under distribution shift, we use ImageNet-R [24] and evaluate the Accuracy, ECE, and NLL, denoted rAcc, rECE, and rNLL on this dataset respectively.

We implement our models using the PyTorch-Lightning framework built on top of PyTorch. Both are open-source Python frameworks. Appendix B and Table 4 detail the hyper-parameters used in our experiments across architectures and datasets. Most training instances are completed on a single Nvidia RTX 3090 except on ImageNet, for which we use 2 to 8 Nvidia A100-80GB.

Results. Table 1 presents the average performance over five runs on the classification task using the hyper-parameters in Table 4. We showcase that in the particular setting of $\alpha = 2$ and $\gamma = 2$, Packed-Ensembles yields similar results to Deep Ensembles while having a lower memory cost than the single model. On CIFAR-10 [34], we notice that its relative performance compared to Deep Ensembles seems to increase as the original architecture gets bigger. For ResNet-18, the method matches Deep Ensembles on OOD detection metrics but shows slightly worse performance on the others. Using ResNet-50, both models seem to perform equivalently, while Packed-Ensembles slightly outperforms Deep Ensembles in terms of classification performance with Wide ResNet28-10.

On CIFAR-100, Deep Ensembles outperform Packed-Ensembles on ResNet-18. However, we argue that ResNet-18 architecture does not have enough representation capacity to be divided into subnetworks for CIFAR-100. Indeed, when we look at the results of ResNet-50, we can see that Packed-Ensembles has better results than Deep Ensembles. This analysis demonstrates that, given a sufficiently large network, Packed-Ensembles is able to match Deep Ensembles with only 16% of its parameters. In appendix D, we discuss the influence of the representation capacity.

Based on the results in Table 2, we can conclude that Packed-Ensembles improves uncertainty quantification for OOD and distribution shift on ImageNet compared to Deep Ensembles and Single model and that it improves the accu-

2. Note that the benchmark conducted at https://github.com/google/uncertainty-baselines uses only ECE to measure the calibration quality of the models

Table 1 – **Performance comparison (averaged over five runs) on CIFAR-10/100 using ResNet-18 (R18), ResNet-50 (R50), and Wide ResNet28-10 (WR) architectures.** All ensembles have $M = 4$ subnetworks, we highlight the best performances in bold. For our method, we consider $\alpha = \gamma = 2$, except for WR, where $\gamma = 1$. *Mult-Adds* corresponds to the inference cost, i.e., the number of Giga multiply-add operations for a forward pass which is estimated with [64].

Method	Data	Net	Acc ↑	NLL ↓	ECE ↓	AUPR ↑	AUC ↑	FPR95 ↓	Params (M) ↓	Mult-Adds ↓
Single Model	C10	R18	94.0	0.238	0.035	94.0	89.7	33.8	11.17	0.56
BatchEnsemble	C10	R18	92.9	0.257	0.031	92.4	87.8	32.1	11.21	2.22
MIMO ($\rho = 1$)	C10	R18	94.0	0.228	0.033	94.4	90.2	28.6	11.19	2.24
Masksembles	C10	R18	94.0	0.188	0.009	93.6	89.5	27.8	11.24	2.22
Packed-Ensembles	C10	R18	94.3	0.178	**0.007**	**94.7**	**91.3**	23.2	**8.18**	**0.48**
Deep Ensembles	C10	R18	**95.1**	**0.156**	0.008	**94.7**	**91.3**	**18.0**	44.70	2.22
Single Model	C10	R50	95.1	0.211	0.031	95.2	91.9	23.6	23.52	1.30
BatchEnsemble	C10	R50	93.9	0.255	0.033	94.7	91.3	20.1	23.63	5.19
MIMO ($\rho = 1$)	C10	R50	95.4	0.197	0.030	95.1	90.8	26.0	23.59	5.22
Masksembles	C10	R50	95.3	0.175	0.019	95.7	92.2	22.1	23.81	5.19
Packed-Ensembles	C10	R50	95.9	0.137	**0.008**	**97.3**	**95.2**	**14.4**	**14.55**	**1.00**
Deep Ensembles	C10	R50	**96.0**	**0.136**	**0.008**	97.0	94.7	15.5	94.08	5.19
Single Model	C10	WR	95.4	0.200	0.029	96.1	93.2	20.4	36.49	5.95
BatchEnsemble	C10	WR	95.6	0.206	0.027	95.5	92.5	22.1	36.59	23.81
MIMO ($\rho = 1$)	C10	WR	94.7	0.234	0.034	94.9	90.6	30.9	36.51	23.82
Masksembles	C10	WR	94.0	0.186	0.016	97.2	95.0	14.5	36.53	23.82
Packed-Ensembles	C10	WR	**96.2**	**0.133**	**0.009**	**98.1**	**96.5**	**11.1**	**19.35**	**4.06**
Deep Ensembles	C10	WR	95.8	0.143	0.013	97.8	96.0	12.5	145.96	23.82
Single Model	C100	R18	75.1	1.016	0.093	88.6	79.5	55.0	11.22	0.56
BatchEnsemble	C100	R18	71.2	1.236	0.116	86.0	75.4	60.2	11.25	2.22
MIMO ($\rho = 1$)	C100	R18	75.3	0.962	0.069	89.2	80.7	52.9	11.36	2.24
Masksembles	C100	R18	74.2	1.054	0.061	86.7	76.3	59.8	11.24	2.22
Packed-Ensembles	C100	R18	76.4	0.858	0.041	88.7	79.8	57.1	**8.18**	**0.48**
Deep Ensembles	C100	R18	**78.2**	**0.800**	**0.018**	**90.2**	**82.4**	**50.5**	44.88	2.22
Single Model	C100	R50	78.3	0.905	0.089	87.4	77.9	57.6	23.70	1.30
BatchEnsemble	C100	R50	66.6	1.788	0.182	85.2	74.6	60.6	23.81	5.19
MIMO ($\rho = 1$)	C100	R50	79.0	0.876	0.079	87.5	76.9	64.7	24.33	5.22
Masksembles	C100	R50	78.5	0.832	0.046	90.3	81.9	52.3	23.81	5.19
Packed-Ensembles	C100	R50	**81.2**	**0.703**	**0.020**	**90.0**	**81.7**	56.5	**15.55**	**1.00**
Deep Ensembles	C100	R50	80.9	0.713	0.026	89.2	80.8	**52.5**	94.82	5.19
Single Model	C100	WR	80.3	0.963	0.156	81.0	64.2	80.1	**36.49**	**5.95**
BatchEnsemble	C100	WR	82.3	0.835	0.130	**88.1**	**78.2**	**69.8**	36.59	23.81
MIMO ($\rho = 1$)	C100	WR	80.2	0.822	**0.028**	84.9	72.0	72.8	36.51	23.82
Masksembles	C100	WR	74.4	0.937	0.063	76.1	60.0	75.1	36.53	23.82
Packed-Ensembles	C100	WR	**83.9**	**0.678**	0.089	86.2	73.2	80.7	36.62	**5.96**
Deep Ensembles	C100	WR	82.5	0.903	0.229	81.6	67.9	71.3	145.96	23.82

racy with a moderate training and inference cost.

Study on the parameters α and γ. Table 1 reports results only for $\alpha = 2$ and $\gamma = 2$; however, depending on the task, the architecture used, and the available memory, one might want to tune those parameters to fit more precisely its needs. Figures 6 and 7 showcase the performance evolution of the Packed-Ensembles as the α values vary.

5 Discussions

We have shown that Packed-Ensembles has attractive properties, mainly by providing a similar quality of Uncertainty Quantification as Deep Ensembles while using a reduced architecture and computing cost. Several questions can be raised, and we conducted some studies - detailed in the Appendix sections - to provide possible answers.

Discussion on the sparsity As described in section 3, one could interpret Packed-Ensembles as leveraging group convolutions to approximate Deep Ensembles with a mask operation applied to some components. In Appendix C, by using a simplified model, we propose a bound of the approximation error based on the Kullback-Leibler divergence between the Deep Ensemble and its pruned version. This bound depends on the density of ones in the mask p, and more specifically, depends on the terms $(p-1)$, $p(p-1)$, and $(p-1)^2/p^2$. By manipulating these terms, corresponding to modifying the number of subnetworks M, the number of groups γ, and the dilatation faction α, we could theoretically be able to control the approximation error.

On the sources of stochasticity Diversity is essential in ensembles and is usually obtained by exploiting two primary sources of stochasticity: the random initialization of the model's parameters and the shuffling of the batches. A last source of stochasticity is introduced during the training phase by the non-deterministic behavior of the backpropagation algorithms.

In Appendix F, we study the function space diversities which arise from every possible combination of these

Table 2 – **Performance comparison on ImageNet [7] using ResNet-50 (R50) and ResNet-50x4 (R50x4).** All ensembles have $M = 4$ subnetworks and $\gamma = 1$. We highlight the best performances in bold. For OOD tasks, we use ImageNet-O (IO) [25] and Texture (T) [65], and for distribution shift we use ImageNet-R [24]. The number of parameters and operations are available in Appendix M.

Method	Net	Acc	ECE	AUPR (T)	AUC (T)	FPR95 (T)	AUPR (IO)	AUC (IO)	FPR95 (IO)	rAcc	rNLL	rECE
Single Model	R50	77.8	0.1206	18.0	80.9	68.6	3.6	50.8	90.8	23.5	5.187	0.0822
BatchEnsemble	R50	75.9	0.0348	20.2	81.6	66.5	4.0	55.2	82.3	21.0	6.148	0.1649
MIMO	R50	77.6	0.1465	18.4	81.6	66.8	3.7	52.2	90.6	23.4	5.115	0.0585
Masksembles	R50	73.6	0.2093	13.6	79.7	68.3	3.3	47.7	87.7	21.2	5.139	0.0107
Packed-Ensembles $\alpha = 3$	R50	77.9	0.1796	**35.1**	**88.2**	**43.7**	**9.9**	**68.4**	**80.9**	23.8	4.978	0.0221
Deep Ensembles	R50	**79.2**	0.2326	19.6	83.4	62.1	3.7	52.5	85.5	**24.9**	**4.879**	**0.0182**
Single Model	R50×4	80.2	0.0221	20.5	82.6	63.9	4.9	60.2	87.4	26.0	5.190	0.1721
BatchEnsemble	R50×4	77.7	0.0237	23.8	82.8	63.8	4.4	58.4	80.5	23.4	6.079	0.2029
MIMO	R50×4	80.3	0.0150	19.3	82.5	66.1	4.9	60.7	86.4	25.8	5.278	0.1886
Masksembles	R50×4	/	/	/	/	/	/	/	/	/	/	/
Packed-Ensembles $\alpha = 2$	R50×4	81.3	0.1034	**34.6**	**88.1**	**50.3**	**9.6**	**69.9**	**79.2**	26.6	4.848	**0.0750**
Deep Ensembles	R50×4	**82.1**	0.0534	23.0	85.6	58.1	5.0	62.7	81.9	**28.2**	**4.789**	0.1048

sources. It follows that only one of these sources is often sufficient to generate diversity, and no peculiar pattern seems to emerge to predict the best combination. Specifically, we highlight that even the only use of non-deterministic algorithms introduces enough diversity between each subnetwork of the ensemble.

Ablation study We perform ablations studies to assess the impact of the parameters M, α, and γ on the performances of Packed-Ensembles. Appendix D provides in-depth details of this study. No explicit behavior appears from the results we obtained. A trend shows that a higher number of subnetworks helps get better OOD detection, but the improvement in terms of AUPR is not significant.

Training speed Depending on the chosen hyperparameters α, M, and γ, PE may have fewer parameters than the single model, as shown in Table 1. This translates into an expected lower number of operations. However, a study of the training and inference speeds, developed in Appendix H, shows that using PE-(2,4,1) does not significantly increase the training and testing times compared to the single model while improving accuracy and uncertainty quantification performances.
However, this also hints that the actual speedup is not optimal despite the significant acceleration offered by 16 bits floating points.

OOD criteria The maximum softmax probability is often used as criterion for discriminating OOD elements. However, this criterion is not unique, and others can be used, such as the Mutual Information, the maximum logit, or the Shannon entropy of the mean prediction. Although no relationship is expected between this criterion and the method of Packed-Ensembles, we obtained different performances in OOD detection according to the selected criterion. The results are detailed in Appendix E and show that an approach based on the maximum logit seems to give the best results in detecting OOD.
It should be noted that this discussion focuses primarily on CIFAR-100 and that the notion of OOD depends on the training distribution. Such a discussion does not necessarily generalize to all datasets. Indeed, preliminary results have shown that Mutual information outperforms the other criteria for our method applied to the ImageNet dataset.

6 Related Work

Ensembles and uncertainty quantification. Bayesian Neural Networks (BNNs) [46; 53] are the cornerstone and primary source of inspiration for uncertainty quantification in deep learning. Despite the progress enabled by variational inference [31; 4], BNNs remain challenging to scale and train for large DNN architectures [10]. DE [36] arise as a practical and efficient instance of BNNs, coarsely but effectively approximating the posterior distribution of weights [69]. DE are currently the best-performing approach for both predictive performance and uncertainty estimation [58; 17].

Efficient ensembles. The appealing properties in performance and diversity of DE [11], but also their major downside related to computational cost, have inspired a large cohort of approaches aiming to mitigate it. BatchEnsemble [66] spawns an ensemble at each layer thanks to an efficient parameterization of subnetwork-specific parameters trained in parallel. MIMO [19] shows that a large network can encapsulate multiple subnetworks using a multi-input multi-output configuration. A single network can be used in ensemble mode by disabling different sub-sets of weights at each forward pass [14; 9]. [43] leverage the sparse networks training algorithm of [49] to produce ensembles of sparse networks. Ensembles can be computed from a single training run by collecting intermediate model checkpoints [28; 15], by computing the posterior distribution of the weights by tracking their trajectory during training [47; 12], and by ensembling predictions over multiple augmentations of the input sample [1]. However, most of these approaches require multiple forward passes.

Neural network compression. The most intuitive approach for reducing the size of a model is to employ DNNs

that are memory-efficient by design, relying on, e.g., channel shuffling [74], point-wise convolutional filters [41], weight sharing [3], or a combination of them. Some of the most popular architectures that leverage such models are SqueezeNet [29], ShuffleNet [75], and MobileNet-v3 [27]. Some approaches conduct automatic model size reduction, e.g., network sparsification [50; 45; 13; 63]. These approaches aim at removing as many parameters as possible from the model to improve memory and computation efficiency. Similarly, quantization approaches [18; 42] avoid or minimize the computation cost of floating point operation and optimize the use of the much more efficient integer computation.

Grouped convolutions. To the best of our knowledge, grouped convolutions (group of convolutions) were introduced by [35]. Enabling the computation of several independent convolutions in parallel, they developed the idea of running a single model on multiple GPU devices. [70] demonstrate that using grouped convolutions leads to accuracy improvements and model complexity reduction. So far, grouped convolutions have been used primarily for computational efficiency but also to compute multiple output branches in parallel [5]. Here, we re-purpose them to delineate multiple subnetworks within a network and efficiently train an ensemble of such subnetworks.

7 Conclusions

We propose a new ensemble framework: Packed-Ensembles, which is able to approximate Deep Ensembles in terms of uncertainty quantification and accuracy. Our work provides several new findings: first, we show that ensembling independent small neural networks can be equivalent to ensembling independent deep neural networks. Secondly, we demonstrate that not all sources of diversity are needed to improve the diversity of the ensemble. Thirdly, we show that Packed-Ensembles is more stable than a single DNN. Fourthly, we highlight that there is a trade-off between the accuracy and the size of the parameters, and Packed-Ensembles allows us to have flexible and efficient ensembling. In the future, we intend to explore Packed-Ensembles for more complex tasks.

8 Reproducibility

Alongside this paper, we provide the source code of Packed-Ensembles layers. Moreover, two notebooks enable the training of Packed-Ensembles based on the ResNet-50 architecture, both on CIFAR-10 and CIFAR-100 (public datasets). To ensure reproducibility, we report the performance given a specific random seed with a deterministic training process. Furthermore, it should be noted that the source code contains two PyTorch Module classes to produce Packed-Ensembles efficiently. The idea would be to release a Python package to provide easier access to Packed-Ensembles layers. A readme file at the root of the project details how to install and run experiments. In addition, we showcase how to get Packed-Ensembles from LeNet [38].

9 Ethics

The purpose of this paper is to provide a method for a better estimation of Deep Learning model uncertainty. Nevertheless, we acknowledge their limitation, which could become critical when applied to safety systems. While this work aims to improve the reliability of Deep Neural Networks, this approach is not ready for deployment in safety-critical systems. We show the limitations of our approach in several experiments. Many more validation and verification steps would be needed before real-world deployment to ensure robustness to various unknown situations, corner cases, adversarial attacks, and biases.

References

[1] Arsenii Ashukha, Alexander Lyzhov, Dmitry Molchanov, and Dmitry Vetrov. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. In *ICLR*, 2020. 7

[2] William H Beluch, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. The power of ensembles for active learning in image classification. In *CVPR*, 2018. 16

[3] Gabriel Bender, Hanxiao Liu, Bo Chen, Grace Chu, Shuyang Cheng, Pieter-Jan Kindermans, and Quoc V. Le. Can weight sharing outperform random architecture search? An investigation with TuNAS. In *CVPR*, 2020. 8

[4] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *ICML*, 2015. 7

[5] Hao Chen and Abhinav Shrivastava. Group ensemble: Learning an ensemble of convnets in a single convnet. *arXiv preprint arXiv:2007.00649*, 2020. 2, 8

[6] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPR*, 2020. 14

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 7, 22

[8] Thomas G Dietterich. Ensemble methods in machine learning. In *IWMCS*, 2000. 2

[9] Nikita Durasov, Timur Bagautdinov, Pierre Baque, and Pascal Fua. Masksembles for uncertainty estimation. In *CVPR*, 2021. 2, 5, 7, 14

[10] Michael Dusenberry, Ghassen Jerfel, Yeming Wen, Yian Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable bayesian neural nets with rank-1 factors. In *ICML*, 2020. 7

[11] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019. 2, 7

[12] Gianni Franchi, Andrei Bursuc, Emanuel Aldea, Séverine Dubuisson, and Isabelle Bloch. Tradi: Tracking deep neural network weight distributions. In *ECCV*, 2020. 2, 7

[13] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*, 2018. 8, 18

[14] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016. 2, 7, 21

[15] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *NeurIPS*, 2018. 2, 7

[16] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *ICML*, 2017. 2

[17] Fredrik K Gustafsson, Martin Danelljan, and Thomas B Schon. Evaluating scalable bayesian deep learning methods for robust computer vision. In *CVPR Workshops*, 2020. 2, 7

[18] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *ICLR*, 2016. 8

[19] Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew Mingbo Dai, and Dustin Tran. Training independent subnetworks for robust prediction. In *ICLR*, 2020. 2, 5, 7

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5, 14

[21] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *CVPR*, 2019. 2

[22] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019. 19, 20

[23] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*, 2017. 5

[24] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *CVPR*, 2021. 5, 7

[25] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *CVPR*, 2021. 5, 7

[26] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *ICML*, 2015. 21

[27] Andrew G. Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for MobileNetV3. In *ICCV*, 2019. 8

[28] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get M for free. In *ICLR*, 2017. 2, 7

[29] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and $<$ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 8

[30] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 4

[31] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 1999. 7

[32] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *NeurIPS*, 2017. 21

[33] Dan Kondratyuk, Mingxing Tan, Matthew Brown, and Boqing Gong. When ensembling smaller models is more efficient than single large models. *arXiv preprint arXiv:2005.00570*, 2020. 2

[34] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, MIT, 2009. 5, 19

[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 2, 8

[36] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, 2017. 2, 3, 5, 7, 16, 21

[37] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989. 2

[38] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998. 8

[39] 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

[40] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J. Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, Michael Sokolsky, Ganymed Stanek, David Stavens, Alex Teichman, Moritz Werling, and Sebastian Thrun. Towards fully autonomous driving: Systems and algorithms. In *IV*, 2011. 2

[41] Feng Liang, Zhichao Tian, M. Dong, Shuting Cheng, Li Sun, Hai Helen Li, Yiran Chen, and Guohe Zhang. Efficient neural network using pointwise convolution kernels with linear phase constraint. *Neurocomputing*, 2021. 8

[42] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In *NeurIPS*, 2017. 8

[43] Shiwei Liu, Tianlong Chen, Zahra Atashgahi, Xiaohan Chen, Ghada Sokar, Elena Mocanu, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. Deep ensembling with no overhead for either training or testing: The all-round blessings of dynamic sparsity. In *ICLR*, 2022. 7

[44] Ekaterina Lobacheva, Nadezhda Chirkova, Maxim Kodryan, and Dmitry Vetrov. On power laws in deep ensembles. In *NeurIPS*, 2020. 2

[45] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. In *ICLR*, 2018. 8

[46] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 1992. 7

[47] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In *NeurIPS*, 2019. 2, 7

[48] Rowan McAllister, Yarin Gal, Alex Kendall, Mark Van Der Wilk, Amar Shah, Roberto Cipolla, and Adrian Weller. Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning. In *IJCAI*, 2017. 2

[49] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 2018. 7

[50] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *ICML*, 2017. 8

[51] Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *AAAI*, 2015. 5

[52] Brady Neal, Sarthak Mittal, Aristide Baratin, Vinayak Tantia, Matthew Scicluna, Simon Lacoste-Julien, and Ioannis Mitliagkas. A modern take on the bias-variance tradeoff in neural networks. *arXiv preprint arXiv:1810.08591*, 2018. 19

[53] Radford M Neal. Bayesian learning for neural networks. *PhD thesis, University of Toronto*, 1995. 7

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

[55] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, 2015. 2

[56] Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? Uncovering how neural network representations vary with width and depth. In *ICLR*, 2020. 18

[57] D.A. Nix and A.S. Weigend. Estimating the mean and variance of the target probability distribution. In *ICNN*, 1994. 21

[58] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua V. Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *NeurIPS*, 2019. 2, 7, 19

[59] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 14

[60] Michael P Perrone and Leon N Cooper. When networks disagree: Ensemble methods for hybrid neural networks. Technical report, Brown University, 1992. 2

[61] Alexandre Ramé, Rémy Sun, and Matthieu Cord. Mixmo: Mixing multiple inputs for multiple outputs via deep subnetworks. In *ICCV*, 2021. 2

[62] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 14

[63] Enzo Tartaglione, Andrea Bragagnolo, Attilio Fiandrotti, and Marco Grangetto. Loss-based sensitivity regularization: towards deep sparse neural networks. *Neural Networks*, 2022. 8

[64] Torchinfo. Torchinfo. https://github.com/TylerYep/torchinfo. Version: 1.7.1. 6, 22

[65] Haoqi Wang, Zhizhong Li, Litong Feng, and Wayne Zhang. ViM: Out-of-distribution with virtual-logit matching. In *CVPR*, 2022. 5, 7

[66] Yeming Wen, Dustin Tran, and Jimmy Ba. BatchEnsemble: an alternative approach to efficient ensemble and lifelong learning. In *ICLR*, 2019. 2, 5, 7

[67] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019. 14

[68] Ross Wightman, Hugo Touvron, and Herve Jegou. Resnet strikes back: An improved training procedure in timm. In *NeurIPS 2021 - Workshop ImageNet PPF*, 2021. 14

[69] Andrew G Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. In *NeurIPS*, 2020. 3, 7

[70] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 2, 8, 21

[71] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *CVPR*, 2019. 14

[72] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 5, 14

[73] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 14

[74] Qing-Long Zhang and Yubin Yang. SA-Net: Shuffle attention for deep convolutional neural networks. In *ICASSP*, 2021. 8

[75] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018. 8