TABLE OF CONTENTS - SUPPLEMENTARY MATERIAL

A Notations **17**

B Implementation details **18**

C Discussion on the sparsity **18**

D Ablation Study **21**

E Discussion about OOD criteria **21**

F Discussion about the sources of stochasticity **22**

G Discussion about the subnetworks **23**

H Discussion about the training velocity **24**

I Distribution shift **24**

J Stabilization of the performance **25**

K On the equivalence between sequential training and Packed-Ensembles **25**

L Using groups is not sufficient to provide equivalent results to Packed-Ensembles **26**

M Efficiency of the networks trained on ImageNet **26**

N Regression **26**

A NOTATIONS

We summarize the main notations used in the paper in Table 3.

Table 3: **Summary of the main notations of the paper.**

Notations	**Meaning**
$\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{\lvert\mathcal{D}\rvert}$	The set of $\lvert\mathcal{D}\rvert$ data samples and the corresponding labels
j, m, L	The index of the current layer, the current subnetwork, and the number of layers
$\mathbf{z}^j$	The preactivation feature map and output of the layer $(j-1)$/input of layer j
ϕ	The activation function (considered constant throughout the network)
$\mathbf{h}^j$	The feature map and output of layer j, $\mathbf{h}^j = \phi(\mathbf{z}^j)$
H_j, W_j	The height and width of the feature maps and output of layer $j-1$
C_j	The number of channels of the feature maps and output of layer $j-1$
n_j	The number of parameters of layer j
B	The batch size of the training procedure
mask_m^j	The mask corresponding to the layer j of the subnetwork m
$\lfloor\cdot\rfloor$	The floor function
$\star$, $\circledast$, $\circ$	The 2D cross-correlation, the convolution, and the Hadamard product
s_j	The size of the kernel of the layer j
M	The number of subnetworks in an ensemble
$\hat{\mathbf{y}}_i^m$	The prediction of the subnetwork number m concerning the input $\mathbf{x}_i$
$\hat{\mathbf{y}}_i$	The prediction of the ensemble concerning the input $\mathbf{x}_i$
α	The width-augmentation factor of Packed-Ensembles
γ	The number of subgroups of Packed-Ensembles
$\boldsymbol{\theta}_{\alpha,m}$	The set of weights of the subnetwork m with a width factor α
$\boldsymbol{\omega}_{\alpha,\gamma}^j$	The weights of layer j with γ groups and a width factor α

Table 4: **Hyperparameters for image classification experiments.** HFlip denotes the classical horizontal flip.

Dataset	Networks	Epochs	Batch size	start lr	Momentum	Weight decay	γ-lr	Milestones	Data augmentations
C10	R18	75	128	0.05	0.9	5e-4	0.1	25, 50	HFlip
C10	R50	200	128	0.1	0.9	5e-4	0.2	60, 120, 160	HFlip
C10	WR28-10	200	128	0.1	0.9	5e-4	0.2	60, 120, 160	HFlip
C100	R18	75	128	0.05	0.9	1e-4	0.2	25, 50	HFlip
C100	R50	200	128	0.1	0.9	5e-4	0.2	60, 120, 160	HFlip
C100	WR28-10	200	128	0.1	0.9	5e-4	0.2	60, 120, 160	Medium

B Implementation details

General Considerations. Our code is implemented in PyTorch (Paszke et al., 2019) using the PyTorch Lightning framework. The code will be made publicly available after the anonymity period.

Table 4 summarizes all the hyperparameters used in the paper for CIFAR-10 and CIFAR-100. In all cases, we use SGD combined with a multistep-learning-rate scheduler multiplying the rate by γ-lr at each milestone. Note that BatchEnsemble based on ResNet-50 uses a lower learning rate of 0.08 instead of 0.1 for stability. The medium data augmentation corresponds to a combination of mixup (Zhang et al., 2018a) and cutmix (Yun et al., 2019) with 0.5 switch probability and using timm's augmentation classes (Wightman, 2019), with coefficients respectively 0.5 and 0.2. In this case, we also use RandAugment (Cubuk et al., 2020) with $m = 9$, $n = 2$, and $mstd = 1$ and a label-smoothing (Szegedy et al., 2016) of intensity 0.1.

To ensure that the layers convey sufficient information and are not weakened by groups, we have set a constant minimum number of channels per group to 64 for all experiments presented in the paper. If the number of channels per group is lower than this threshold, γ is reduced. Moreover, we do not apply subgroups (parameterized by γ) on the first layer of the network, nor on the first layer of ResNet's blocks. Experiments in which this minimum number of channels could play a significant role and bring confusion are not presented (see, for instance, PE-$(1, 4, 4)$ in Table 5).

For ImageNet, we use the A3 procedure from Wightman et al. (2021) for all models. Training with the exact A3 procedure was not always possible. Refer to the specific subsection for more details.

Please note that the hyperparameters of the training procedures have not been optimized for our method and have been taken directly from the literature (He et al., 2016; Wightman et al., 2021). We strengthened the data augmentations for WideResNet on CIFAR-100 as we were not able to replicate the results from Zagoruyko & Komodakis (2016).

Masksembles. We use the code proposed by (Durasov et al., 2021) [3]. We modified the mask generation function using binary search, as proposed by the authors since it was unable to build masks for ResNet50x4. We note that the code implies performing batch repeats at the start of the forward passes. All the results regarding this technique are therefore computed with this specification. The ResNet implementations are built using Masksemble2D layers with $M = 4$ and a scale factor of 2 after each convolution.

BatchEnsemble. For BatchEnsemble, we use two different values for weight decay. Table 4 provides the weight decay corresponding to the shared weights. However, no weight decay is applied to the vectors S and R (which generate the rank-1 matrices).

ImageNet. The batch size of Masksembles ResNet-50x4 is reduced to 1120 because of memory constraints. Concerning the BatchEnsembles based on ResNet-50 and ResNet-50x4, we clip the norm of the gradients to 0.0005 to avoid divergence.

C Discussion on the sparsity

In this section, we provide an estimation of the expected distance between a dense, fully-connected layer and a sparse one. For simplicity, we are here assuming to operate with a fully-connected layer. First, let us write our first proposition:

[3] available at github.com/nikitadurasov/masksembles

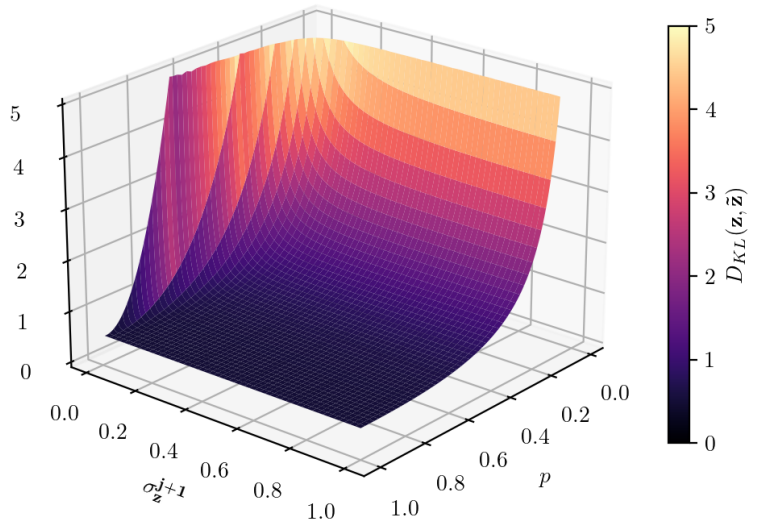

Figure 5: KL divergence for different values of p and $\boldsymbol{\sigma_z^{j+1}}$, with $\boldsymbol{\mu^j}(k) = 0.1\ \forall j, k$ and $w^j(c,k) = 0.1\ \forall j, c, k$.

Proposition C.1. *Given a fully connected layer $j+1$ defined by:*

$$\mathbf{z}^{j+1}(c) = \sum_{k=0}^{C_j - 1} \boldsymbol{\omega}^j(c,k)\mathbf{h}^j(k) \tag{6}$$

and its approximation defined by:

$$\tilde{\mathbf{z}}^{j+1}(c) = \sum_{k=0}^{C_j - 1} (\boldsymbol{\omega}^j(c,k) mask^j(k,c))\mathbf{h}^j(k) \tag{7}$$

Under the assumption that the j follows a Gaussian distribution $\boldsymbol{h}^j \sim \mathcal{N}(\boldsymbol{\mu^j}, \Sigma^j)$, where Σ^j is the covariance matrix, and $\boldsymbol{\mu}^j$ the mean vector, the Kullback–Leibler divergence between the layer and its approximation is bounded by:

$$D_{\mathrm{KL}}(\mathbf{z}, \tilde{\mathbf{z}})(c) \leq \frac{1}{2}\left\{ p + \frac{1}{p} - 2 + \frac{p \cdot (1-p) \sum_{k=0}^{C_j - 1} \boldsymbol{\omega}^j(c,k)^2 \boldsymbol{\mu^j}(k)^2}{(\boldsymbol{\sigma_z^{j+1}})^2(c)} + \frac{\left[(1-p) \times \boldsymbol{\mu_z^{j+1}}(c)\right]^2}{p(\boldsymbol{\sigma_z^{j+1}})^2(c)} \right\} \tag{8}$$

where $p \in [0;1]$ is the fraction of the parameters of $\mathbf{z}^{j+1}(c)$ included in the approximation $\tilde{\mathbf{z}}^{j+1}(c)$.

A plot for (8) is provided in Figure 5.

Proof. To prove Prop. C.1, we state first that, since $\mathbf{h}^j(k)$ follows a Gaussian distribution, and considering that $\boldsymbol{\omega}^j$ at inference time is constant and linearly-combined with a gaussian random variable, $\boldsymbol{z}^{j+1}$ will be as well gaussian-distributed.
From the property of linearity of expectations, we know that the mean for $\mathbf{z}^{j+1}(c)$ is:

$$\boldsymbol{\mu_z^{j+1}}(c) = \sum_{k=0}^{C_j - 1} \boldsymbol{\omega}^j(c,k)\boldsymbol{\mu^j}(k) \tag{9}$$

and the variance is:

$$(\boldsymbol{\sigma_z^{j+1}})^2(c) = \sum_{k=0}^{C_j - 1} \boldsymbol{\omega}^j(c,k)\left[\boldsymbol{\omega}^j(c,k)\Sigma(k,k) + 2\sum_{k'<k} \boldsymbol{\omega}^j(c,k')\Sigma(k',k)\right]. \tag{10}$$

If we assume $\Sigma(i,k) = 0 \; \forall \; i \neq k$, (10) simplifies into:

$$(\boldsymbol{\sigma}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c) = \sum_{k=0}^{C_j-1} \boldsymbol{\omega}^j(c,k)^2 \Sigma(k,k). \tag{11}$$

Let us now consider the case with the mask, similarly as presented in (3):

$$\tilde{\mathbf{z}}^{j+1}(c) = \sum_{k=0}^{C_j-1} (\boldsymbol{\omega}^j(c,k)\text{mask}^j(k,c))\mathbf{h}^j(k) \tag{12}$$

We assume here that $\text{mask}^j \sim Ber(p)$ where p is the probability of the Bernoulli (or 1-pruning rate). In the limit of large C_j, we know that $\tilde{\mathbf{z}}^{j+1}(c)$ follows a Gaussian distribution defined by a mean and a variance equal to:

$$\tilde{\boldsymbol{\mu}}_{\mathbf{z}}^{\boldsymbol{j+1}}(c) = \sum_{k=0}^{C_j-1} \boldsymbol{\omega}^j(c,k)\boldsymbol{\mu}^{\boldsymbol{j}}(k)p \tag{13}$$

$$(\tilde{\boldsymbol{\sigma}}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c) = \sum_{k=0}^{C_j-1} p\boldsymbol{\omega}^j(c,k)^2 \left[\boldsymbol{\mu}^{\boldsymbol{j}}(k)^2(1-p) + \Sigma(k,k)\right] \tag{14}$$

Hence, we have:

$$\tilde{\boldsymbol{\mu}}_{\mathbf{z}}^{\boldsymbol{j+1}}(c) = p \times \boldsymbol{\mu}_{\mathbf{z}}^{\boldsymbol{j+1}}(c) \tag{15}$$

$$(\tilde{\boldsymbol{\sigma}}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c) = p\left[(\boldsymbol{\sigma}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c) + (1-p)\sum_{k=0}^{C_j-1} \boldsymbol{\omega}^j(c,k)^2\boldsymbol{\mu}^{\boldsymbol{j}}(k)^2\right] \tag{16}$$

In order to assess the dissimilarity between $\mathbf{z}$ and $\tilde{\mathbf{z}}$, we can write the Kullback–Leibler divergence:

$$D_{\text{KL}}(\mathbf{z},\tilde{\mathbf{z}})(c) = \frac{1}{2}\left\{\log\left[\frac{(\tilde{\boldsymbol{\sigma}}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c)}{(\boldsymbol{\sigma}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c)}\right] + \frac{(\boldsymbol{\sigma}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c) + \left[\boldsymbol{\mu}_{\mathbf{z}}^{\boldsymbol{j+1}}(c) - \tilde{\boldsymbol{\mu}}_{\mathbf{z}}^{\boldsymbol{j+1}}(c)\right]^2}{(\tilde{\boldsymbol{\sigma}}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c)} - 1\right\} \tag{17}$$

Straightforwardly we can write the inequality:

$$D_{\text{KL}}(\mathbf{z},\tilde{\mathbf{z}})(c) \leq \frac{1}{2}\left\{\frac{(\tilde{\boldsymbol{\sigma}}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c)}{(\boldsymbol{\sigma}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c)} - 1 + \frac{(\boldsymbol{\sigma}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c) + \left[\boldsymbol{\mu}_{\mathbf{z}}^{\boldsymbol{j+1}}(c) - \tilde{\boldsymbol{\mu}}_{\mathbf{z}}^{\boldsymbol{j+1}}(c)\right]^2}{(\tilde{\boldsymbol{\sigma}}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c)} - 1\right\} \tag{18}$$

According to (16) we can write:

$$D_{\text{KL}}(\mathbf{z},\tilde{\mathbf{z}})(c) \leq \frac{1}{2}\left\{\frac{p\left[(\boldsymbol{\sigma}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c) + (1-p)\sum_{k=0}^{C_j-1} \boldsymbol{\omega}^j(c,k)^2\boldsymbol{\mu}^{\boldsymbol{j}}(k)^2\right]}{(\boldsymbol{\sigma}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c)} - 1 + \right.$$
$$\left. + \frac{(\boldsymbol{\sigma}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c) + \left[\boldsymbol{\mu}_{\mathbf{z}}^{\boldsymbol{j+1}}(c) - \tilde{\boldsymbol{\mu}}_{\mathbf{z}}^{\boldsymbol{j+1}}(c)\right]^2}{p\left[(\boldsymbol{\sigma}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c) + (1-p)\sum_{k=0}^{C_j-1} \boldsymbol{\omega}^j(c,k)^2\boldsymbol{\mu}^{\boldsymbol{j}}(k)^2\right]} - 1\right\} \tag{19}$$

Since we know that $\frac{(\boldsymbol{\sigma}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c) + \left[\boldsymbol{\mu}_{\mathbf{z}}^{\boldsymbol{j+1}}(c) - \tilde{\boldsymbol{\mu}}_{\mathbf{z}}^{\boldsymbol{j+1}}(c)\right]^2}{p\left[(\boldsymbol{\sigma}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c) + (1-p)\sum_{k=0}^{C_j-1} \boldsymbol{\omega}^j(c,k)^2\boldsymbol{\mu}^{\boldsymbol{j}}(k)^2\right]} \leq \frac{(\boldsymbol{\sigma}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c) + \left[\boldsymbol{\mu}_{\mathbf{z}}^{\boldsymbol{j+1}}(c) - \tilde{\boldsymbol{\mu}}_{\mathbf{z}}^{\boldsymbol{j+1}}(c)\right]^2}{p(\boldsymbol{\sigma}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c)}$$ we can also write:

$$D_{\text{KL}}(\mathbf{z},\tilde{\mathbf{z}})(c) \leq \frac{1}{2}\left\{p - 1 + \frac{p\cdot(1-p)\sum_{k=0}^{C_j-1} \boldsymbol{\omega}^j(c,k)^2\boldsymbol{\mu}^{\boldsymbol{j}}(k)^2}{(\boldsymbol{\sigma}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c)} + \right.$$
$$\left. + \frac{(\boldsymbol{\sigma}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c) + \left[\boldsymbol{\mu}_{\mathbf{z}}^{\boldsymbol{j+1}}(c) - \tilde{\boldsymbol{\mu}}_{\mathbf{z}}^{\boldsymbol{j+1}}(c)\right]^2}{p(\boldsymbol{\sigma}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c)} - 1\right\} \tag{20}$$

Finally, according to: (15)

$$D_{\text{KL}}(\mathbf{z},\tilde{\mathbf{z}})(c) \leq \frac{1}{2}\left\{p + \frac{1}{p} - 2 + \frac{p\cdot(1-p)\sum_{k=0}^{C_j-1} \boldsymbol{\omega}^j(c,k)^2\boldsymbol{\mu}^{\boldsymbol{j}}(k)^2}{(\boldsymbol{\sigma}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c)} + \frac{\left[(1-p)\times\boldsymbol{\mu}_{\mathbf{z}}^{\boldsymbol{j+1}}(c)\right]^2}{p(\boldsymbol{\sigma}_{\mathbf{z}}^{\boldsymbol{j+1}})^2(c)}\right\}$$

finding back (8).

□

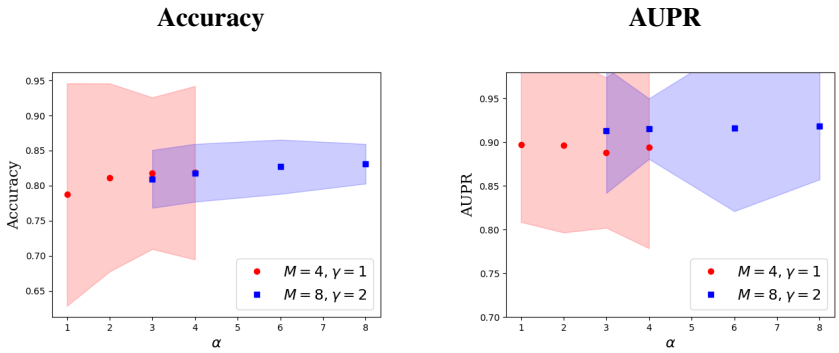

Figure 6: Accuracy and AUPR of Packed-Ensembles with ResNet-50 on CIFAR-100 depending on α.

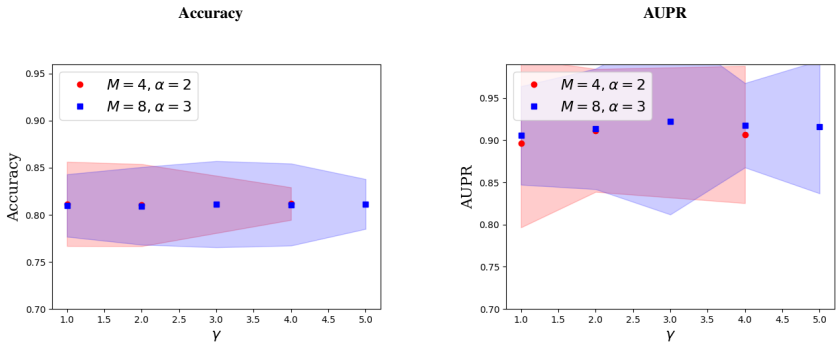

Figure 7: Accuracy and AUPR of Packed-Ensembles with ResNet-50 on CIFAR-100 depending on γ.

D Ablation Study

Our algorithm mainly depends on three hyperparameters. M represents the number of subnetworks in the ensemble, α controls the power of representation of the DNN, and γ is an extra parameter that controls the sparsity degree of the DNN. To evaluate the sensitivity of Packed-Ensembles to these parameters, we train 5 ResNet-50 on CIFAR-10 similarly to the protocol explained in section 4.1. Figures 6 and 7 show that the more we add subnetworks increasing M, the better the performance, in terms of accuracy and AUPR. We also note that the results are stable with γ. Moreover, the resulting accuracy tends to increase with α until it reaches a plateau. These statements are confirmed by the results in Table 5.

E Discussion about OOD criteria

Deep Ensembles (Lakshminarayanan et al., 2017) and Packed-Ensembles are ensembles of DNNs that can be used to quantify the uncertainty of the DNNs prediction. Similarly to Bayesian Neural Network, one can take the softmax outputs of posterior predictive distribution, which define the **MSP** $= \max_{y_i}\{P(\mathbf{y}_i|\mathbf{x}, \mathcal{D})\}$. The MSP can also be used for classical DNN, yet we use the conditional likelihood instead of the posterior distribution in this case.

One can also use the Maximum Logit (ML) as an uncertainty criterion and the entropy of the posterior predictive distribution as an uncertainty criterion, which is defined by **Ent.** $= \mathcal{H}(P(\mathbf{y}_i|\mathbf{x}, \mathcal{D}))$ with $\mathcal{H}$ being the entropy function. Another metric is the mutual information between two random variables, which is defined by: **MI** $= \mathcal{H}(P(\mathbf{y}_i|\mathbf{x}, \mathcal{D})) - \frac{1}{M}\sum_{m=0}^{M-1}\mathcal{H}(P(\mathbf{y}|\boldsymbol{\theta}_{\alpha,m}, \mathbf{x}))$. It represents a

Table 5: **Performance (Acc / ECE / AUPR) of Packed-Ensembles for various α and γ** with ResNet-50 on CIFAR-100 and $M = 4$.

γ \ α	1	2	3	4
1	0.7872 / 0.0165 / 0.8969	0.8116 / 0.0203 / 0.8966	0.8187 / 0.0201 / 0.8825	0.8183 / 0.0230 / 0.8939
2	0.7857 / 0.0185 / 0.9024	0.8103 / 0.0295 / 0.9115	0.8186 / 0.0197 / 0.9127	0.8242 / 0.0190 / 0.9088
4	/	0.8119 / 0.0180 / 0.9066	0.8182 / 0.0236 / 0.9140	0.8225 / 0.0226 / 0.9229

Table 6: Comparison of the effect of the different uncertainty criteria for OOD on CIFAR-100 with different sets of parameters for Packed-Ensembles.

Criterion	OOD eval	$\alpha = 2, \gamma = 1$ $M = 4$	$\alpha = 3, \gamma = 1$ $M = 8$	$\alpha = 4, \gamma = 2$ $M = 8$	$\alpha = 6, \gamma = 4$ $M = 8$	$\alpha = 8, \gamma = 1$ $M = 16$
MSP	AUPR (↑)	0.8952 ± 0.0132	0.9055 ± 0.0034	0.9153 ± 0.0012	0.9149 ± 0.0071	0.9141 ± 0.0057
ML	AUPR (↑)	$\mathbf{0.9183 \pm 0.0098}$	$\mathbf{0.9175 \pm 0.0044}$	$\mathbf{0.9285 \pm 0.0012}$	$\mathbf{0.9265 \pm 0.0070}$	$\mathbf{0.9268 \pm 0.0068}$
Ent.	AUPR (↑)	$\mathbf{0.9105 \pm 0.0138}$	$\mathbf{0.9152 \pm 0.0035}$	$\mathbf{0.9260 \pm 0.0016}$	$\mathbf{0.9237 \pm 0.0066}$	$\mathbf{0.9252 \pm 0.0060}$
MI	AUPR (↑)	$\mathbf{0.8649 \pm 0.0061}$	$\mathbf{0.9139 \pm 0.0077}$	$\mathbf{0.9157 \pm 0.0072}$	$\mathbf{0.9196 \pm 0.0109}$	$\mathbf{0.9245 \pm 0.0091}$
v	AUPR (↑)	0.8404 ± 0.0071	0.8746 ± 0.0056	0.8827 ± 0.0033	0.8842 ± 0.0102	0.8931 ± 0.0072
MSP	AUC (↑)	0.8056 ± 0.0260	0.8204 ± 0.0101	0.8408 ± 0.0033	0.8432 ± 0.0134	0.8387 ± 0.0094
ML	AUC (↑)	$\mathbf{0.8562 \pm 0.0194}$	$\mathbf{0.8421 \pm 0.0115}$	$\mathbf{0.8665 \pm 0.0027}$	$\mathbf{0.8621 \pm 0.0144}$	$\mathbf{0.8607 \pm 0.0114}$
Ent.	AUC (↑)	$\mathbf{0.8361 \pm 0.0271}$	$\mathbf{0.8427 \pm 0.0095}$	$\mathbf{0.8662 \pm 0.0027}$	$\mathbf{0.8617 \pm 0.0136}$	$\mathbf{0.8614 \pm 0.0096}$
MI	AUC (↑)	$\mathbf{0.7711 \pm 0.0064}$	$\mathbf{0.8312 \pm 0.0135}$	$\mathbf{0.8402 \pm 0.0116}$	$\mathbf{0.8468 \pm 0.0163}$	$\mathbf{0.8513 \pm 0.0120}$
v	AUC (↑)	0.7305 ± 0.0153	0.7799 ± 0.0129	0.7943 ± 0.0082	0.7999 ± 0.0166	0.8092 ± 0.0113

measure of the ensemble entropy, which is the entropy of the posterior minus the average entropy over predictions.

The last metric, used in active learning, is the variation ratio (Beluch et al., 2018), which measures the dispersion of a nominal variable and is calculated as the proportion of predicted class labels that are not the modal class prediction. It is defined by: $\mathbf{v} = 1 - \frac{f_i}{M}$, where f_i is the number of predictions falling into the modal class category.

In Table 6, the results for the different metrics are reported. We note that **ML** seems to be the best metric to detect OOD. This metric is followed by **Ent.** and then **MI**. Note that **v**, widely used in active learning, does not seem effective in detecting OOD samples. This shows us that it is essential to use a good criterion in addition to good ensembling.

F Discussion about the sources of stochasticity

As written in the introduction of the paper, diversity is essential to the success of ensembling, be it for its accuracy but also for calibration and OOD detection. Three primary sources can induce weight diversity, and therefore diversity in the function space, during the training. These sources are the initialization of the weights, the composition of the batches, and the use of non-deterministic backpropagation algorithms [4]. On Table 7, we measure the performance and diversity of Packed-Ensembles trained on CIFAR-100. This diversity is measured by the mutual information and is twofold: we compute the in-distribution mutual information (ID**MI**) on the test set of CIFAR-100 and the OOD mutual information (OOD**MI**) on SVHN. of the mean performance of Packed-Ensembles over five experiments in accuracy, NLL, calibration, and OOD detection on CIFAR-100. Concerning the performance, we compute the accuracy, ECE, and AUPR, which are proxies of the quality of this diversity. Results of Table 7 lead to several takeaways. First, they hint that there is no clear best set of trivial sources of stochasticity. Except for the first (and greyed) line, which corresponds to ensembling completely identical networks (the training being totally deterministic, which the null MI confirms), the results seem equivalent in diversity (via mutual information) and ID/OOD performance. Secondly, it shows that the use of non-deterministic algorithms can be sufficient to generate diversity. It was noted that this effect does not always happen depending on the selected architecture and the precision used (`float16`, or `float32`).

Given that there is no emerging best set of stochasticity, we use the faster non-deterministic backpropagation algorithms and different initializations to ensure enough stochasticity and for programming convenience.

[4] see https://docs.nvidia.com/deeplearning/cudnn/api/index.html

Table 7: **Comparison of the diversities and the performance wrt. the different sources of stochasticity on CIFAR-100. ND** corresponds to the use of Non-deterministic backpropagation algorithms, **DI** to different initializations, and **DB** to different compositions of the batches. A standard error (over five runs) is included in small font.

Stochasticity			**ResNet-18**				
ND	**DI**	**DB**	**Acc** (↑)	**ECE** (↓)	**AUPR** (↑)	IDMI	OODMI
-	-	-	71.70±0.06	0.0497±0.0013	87.32±0.91	0±0	0±0
✓	-	-	75.79±0.22	0.0365±0.0044	89.53±0.47	0.1945	0.4001
-	✓	-	76.20±0.04	0.0419±0.0006	89.54±0.39	0.2011	0.4391
-	-	✓	76.06±0.02	0.0434±0.0011	88.70±0.27	0.1987	0.4079
✓	✓	-	76.10±0.05	0.0424±0.0004	88.65±0.42	0.1995	0.4360
✓	-	✓	76.19±0.11	0.0433±0.0010	88.87±0.15	0.2032	0.4090
-	✓	✓	76.14±0.07	0.0437±0.0008	89.21±0.38	0.1943	0.4195
✓	✓	✓	76.29±0.07	0.0445±0.0006	89.00±0.54	0.1954	0.4060
Stochasticity			**ResNet-50**				
-	-	-	77.63±0.23	0.0825±0.0018	89.19±0.65	0±0	0±0
✓	-	-	80.94±0.10	0.0179±0.0010	90.23±0.62	0.1513	0.4022
-	✓	-	81.01±0.06	0.0202±0.0011	91.10±0.39	0.1524	0.4088
-	-	✓	80.87±0.10	0.0178±0.0010	90.80±0.30	0.1505	0.4115
✓	✓	-	81.16±0.10	0.0210±0.0008	91.69±0.56	0.1584	0.4135
✓	-	✓	81.14±0.07	0.0200±0.0007	90.41±0.39	0.1503	0.3897
-	✓	✓	81.10±0.05	0.0186±0.0016	90.85±0.29	0.1521	0.4034
✓	✓	✓	81.08±0.08	0.0198±0.0013	90.68±0.25	0.1534	0.4031

G Discussion about the subnetworks

Discussing the width and depth of deep neural networks is an essential research topic. Researchers have focused on the correct approaches to increase the depth of DNN, increasing the accuracy. Nguyen et al. (2020) demonstrate that the width and depth are connected to the model capacity and enable the DNN to learn block structures, leading to good accuracy. Hence, by splitting the width, chances are that we could decrease the model capacity.

Deep neural networks are heavily over-parameterized DNNs as suggested by lottery ticket hypothesis (Frankle & Carbin, 2018). Thus it is possible to kill up to 80% of the neurons without losing too much performance. MIMO builds on this assumption to ensure that each network drives several networks simultaneously. However, unlike in our work, the boundaries between the networks are not clearly defined. Thus, the same neuron can be used for multiple different subnetworks (DNNs) in the same ensemble. In our case, we clearly assign each neuron to a DNN from the ensemble. This way, the DNNs are not mixed up and can learn an independent representation. However, as in MIMO, we rely on the fact that not all neurons are helpful, so we split the width of the initial DNNs into a set of DNNs. This decomposition may seem crude. However, it allows us to better parallelize Packed-Ensembles for training and inference. To overcome the fact that our networks are not wide enough if M is too large, we have added an alpha hyperparameter that can increase the width of the subnetworks. In Figure 8, we study the influence of the width of the subnetworks.

First, we notice that the accuracy increases with the width, while the AUPR looks relatively constant. This seems to confirm the importance of the alpha parameter in balancing the width of the DNN. Furthermore, reducing the width does not appear to reduce the accuracy significantly. Thus, this justifies our choice to split the width of the DNN to produce several subnetworks since the uncertainty quantification will be constant, and the accuracy will not drop drastically. Moreover, the addition of the alpha parameter allows us to add a new degree of freedom to our ensemble and enables it to increase its accuracy.

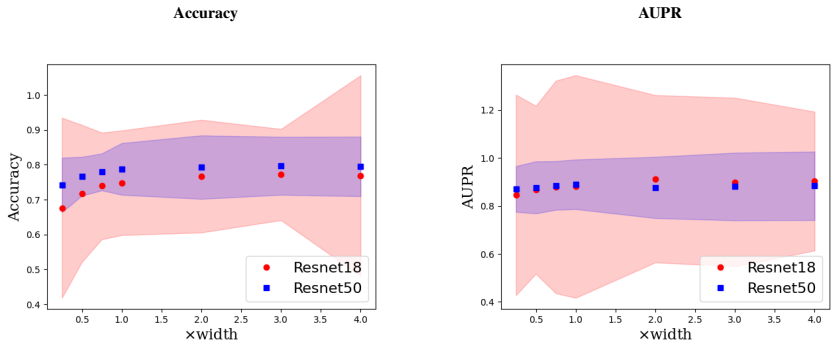

Figure 8: Accuracy and AUPR curves of ResNet-18 in red and ResNet-50 in blue on CIFAR-100 with different widths. When the width is equal to 1, it corresponds to the original ResNet; when the width is equal to x, the width of every layer is multiplied by x.

Table 8: **Comparison of training and inference times of different ensemble techniques** using torch1.12.1+cu113 on an RTX 3090. All ensembles have four subnetworks.

models *ResNet-50 CIFAR-100*	**f32 Precision** **Training** ↓ *s/epoch*	**Inference** ↑ *im/s*	**f16 Precision** **Training** ↓ *s/epoch*	**Inference** ↑ *im/s*
Single Model	37.06	3709	22.42	5718
Packed-Ensembles-(2,4,1)	179.50	1381	51.20	3406
Packed-Ensembles-(2,4,2)	175.10	1501	52.11	3440
Deep Ensembles	145.30	1001	84.86	1609
MIMO	37.90	3574	24.44	5649
BatchEnsemble	58.78	1809	53.97	1916

H Discussion about the training velocity

Our experiments show that grouped convolutions are not as fast as they could theoretically be, and confirm the statements made by many PyTorch and TensorFlow users [5]. Following the idea that grouped convolutions are bandwidth-bound, we advise readers to leverage Native Automatic Mixed Precision (AMP) and cuDNN benchmark flags when training a Packed-Ensembles to reduce the bandwidth bottleneck compared to the baseline. AMP also divides the VRAM usage by two while yielding equally good results. Future improvements of PyTorch grouped convolutions should help Packed-Ensembles develop its full potential, increasing its current assets. We note in Table 8 that using `float16`, Packed-Ensembles is only $1.6\times$ slower than the single model during inference. Furthermore, Packed-Ensembles is only $2.3\times$ slower during training than the single model, making it an efficient model capable of training four models in half the time of a Deep Ensembles.

I Distribution shift

In this section, we evaluate the robustness of Packed-Ensembles under dataset shift. We use models trained on CIFAR-100 (Krizhevsky, 2009) and shift the data using corruptions and perturbations proposed by (Hendrycks & Dietterich, 2019) to produce CIFAR-100-C. There are five levels of perturbations called "severity," from one, the weakest, to five, the strongest. In real-world scenarios, distributional shift is crucial, as explained by (Ovadia et al., 2019), and it is critical to study how much a model prediction shifts from the original training data distribution. Thanks to Figure 9, we notice that Packed-Ensembles achieves the highest accuracy and lowest ECE under distributional shift, leading to a method robust against this uncertainty.

[5] For instance https://github.com/pytorch/pytorch/issues/75747

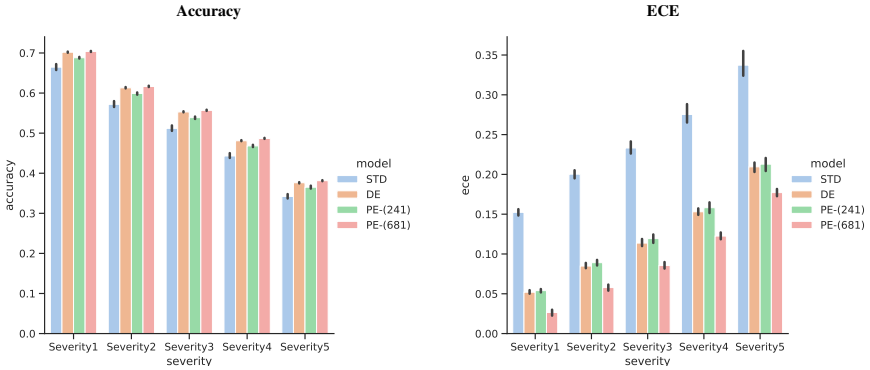

Figure 9: Accuracy and Calibration under distributional shift. Comparison of the accuracy and ECE under all types of corruptions on (a) CIFAR-100-C (Hendrycks & Dietterich, 2019) with different levels of severity.

Table 9: **Comparison between the results obtained with Packed-Ensembles and a similar ResNeXt-50**. The dataset is CIFAR-10.

Network	Acc	NLL	ECE	AUPR	AUC	FPR95	Params (M)
PE ResNet-50	**96.0**	**0.1367**	**0.0087**	**97.1**	**94.9**	**14.5**	23.6
ResNeXt-50	90.4	0.4604	0.0709	90.4	82.5	63.4	**23.0**

J STABILIZATION OF THE PERFORMANCE

We perform five times each training task on CIFAR-10 and CIFAR-100 to estimate a better value and be able to compute the variance. Let us first note that the standard deviation for the single DNN on CIFAR-100 with a ResNet-50 architecture amounts to 0.68%. Ensemble strategies shrink the standard variation to 0.43% for Deep Ensembles and 0.19% for Packed-Ensembles. Thus it seems that Packed-Ensembles makes DNN predictions more stable in addition to improving accuracy and uncertainty quantification. This result is interesting as it appears to contradict Neal et al. (2018), who claim that wider DNNs have a smaller variance. This stability might come from the ensembling.

K ON THE EQUIVALENCE BETWEEN SEQUENTIAL TRAINING AND PACKED-ENSEMBLES

The sequential training of Deep Ensembles differs significantly from the training procedure of Packed-Ensembles. The main differences lie in the subnetworks' batch composition and the best models' selection.

Concerning Packed-Ensembles, the batches are strictly the same for all subnetworks, thus removing one source of stochasticity compared to sequential learning. Yet, in practice, we show empirically that random initialization and stochastic algorithms are sufficient to get diverse subnetworks (see Appendix F for more details).

For the selection of models, Packed-Ensembles considers subnetworks as a whole (i.e., maximize the ensemble accuracy on the validation set) and therefore selects the best ensemble at a given epoch. On the other hand, sequential training selects the best networks individually, possibly on different epochs, which does not guarantee that the best ensemble is selected but ensures the optimality of subnetworks over the epochs.

Table 10: **Comparison of the efficiency of the networks trained on ImageNet (Deng et al., 2009).** All ensembles have $M = 4$ subnetworks and $\gamma = 1$. *Mult-Adds* corresponds to the inference cost, i.e., the number of Giga multiply-add operations for a forward pass which is estimated with Torchinfo.

Method	**Net**	**Params** (M) ↓	**Mult-Adds** (G) ↓
Single Model	R50	25.6	4.09
BatchEnsemble	R50	25.7	16.36
MIMO	R50	31.7	4.45
Masksembles	R50	25.7	16.36
Packed-Ensembles ($\alpha = 3$)	R50	59.1	9.29
Deep Ensembles	R50	102.4	16.36
Single Model	R50x4	383.6	70.0
BatchEnsemble	R50x4	384.4	256.0
MIMO	R50x4	408.3	65.4
Masksembles	R50x4	384.0	256.0
Packed-Ensembles ($\alpha = 2$)	R50x4	392.0	64.47
Deep Ensembles	R50x4	1534.4	280.0

L USING GROUPS IS NOT SUFFICIENT TO PROVIDE EQUIVALENT RESULTS TO PACKED-ENSEMBLES

To make sure that the use of groups cannot simply explain our results, we compare Packed-Ensembles to a single ResNeXt-50 (32×4d) (Xie et al., 2017) in Table 9. ResNeXt-50 is fairly equivalent to our method but does not propagate groups, only used in the middle layer of each block, which are therefore not independent. We keep the same training optimization procedures and data-augmentation strategies detailed in Appendix B.

M EFFICIENCY OF THE NETWORKS TRAINED ON IMAGENET

Table 10 provides the efficiency of the networks trained on ImageNet-1k (see section 4.1.3), in number of parameters and multiply-additions. PE-(3, 4, 1) was preferred to PE-(3, 4, 2) for ResNet50 to improve the representation capacity of the subnetworks.

N REGRESSION

To generalize our work, we propose to study regression tasks. We replicate the setting developed by Hernández-Lobato & Adams (2015), Gal & Ghahramani (2016), and Lakshminarayanan et al. (2017).

For the training in the one-dimensional regression setting, we minimize the gaussian NLL (21) using networks with two outputs neurons which estimate the parameters of a heteroscedastic gaussian distribution (Nix & Weigend, 1994; Kendall & Gal, 2017). One output corresponds to the mean of the predicted gaussian distribution, and the softplus applied on the second is its variance. The ensemble's mean $\bar{\mu}_\theta(\mathbf{x}_i)$ is computed using the empirical mean over the estimators and the variance using the formula of a mixture $\bar{\sigma}_\theta(\mathbf{x}_i)^2 = M^{-1} \sum_m \left(\sigma_{\theta_m}(\mathbf{x}_i)^2 + \mu_{\theta_m}(\mathbf{x}_i)^2\right) - \bar{\mu}_\theta(\mathbf{x}_i)$ (Lakshminarayanan et al., 2017).

$$\mathcal{L}\left(\mu_{\theta_m}(\mathbf{x}_i), \sigma_{\theta_m}(\mathbf{x}_i)^2, y_i\right) = \frac{(y_i - \mu_{\theta_m}(\mathbf{x}_i))^2}{2\sigma_{\theta_m}(\mathbf{x}_i)^2} + \frac{1}{2}\log \sigma_{\theta_m}(\mathbf{x}_i)^2 + \frac{1}{2}\log 2\pi \tag{21}$$

We compare Packed-Ensembles-$(2, 3, 1)$ and Deep Ensembles on the UCI datasets in Table 11. The subnetworks of these methods are based on multi-layer perceptrons with a single hidden layer, containing 400 neurons for the more extensive Protein dataset and 200 for the others, and a ReLU non-linearity. The results show that Packed-Ensembles and Deep Ensembles provide equivalent results on most datasets.

Table 11: Comparison between the results obtained with Packed-Ensembles and Deep Ensembles on regression tasks

Datasets	RMSE		NLL	
	Packed-Ensembles	Deep Ensembles	Packed-Ensembles	Deep Ensembles
Boston housing	**2.218 ± 0.099**	**2.219 ± 0.098**	**2.028 ± 0.034**	2.047 ± 0.028
Concrete	**5.092 ± 0.225**	5.167 ± 0.234	**2.854 ± 0.028**	2.885 ± 0.032
Energy	**1.675 ± 0.085**	1.712 ± 0.067	**1.543 ± 0.072**	**1.553 ± 0.060**
Kin8nm	**0.058 ± 0.003**	**0.058 ± 0.003**	-1.442 ± 0.010	**-1.452 ± 0.010**
Naval Propulsion Plant	**0.002 ± 0.000**	**0.002 ± 0.000**	**-4.835 ± 0.066**	-4.833 ± 0.097
Power Plant	3.127 ± 0.018	**3.097 ± 0.020**	2.607 ± 0.007	**2.600 ± 0.007**
Protein	3.476 ± 0.030	**3.412 ± 0.017**	2.472 ± 0.033	**2.442 ± 0.015**
Wine	**0.482 ± 0.006**	**0.483 ± 0.006**	0.622 ± 0.014	**0.611 ± 0.013**
Yacht	**1.949 ± 0.215**	2.511 ± 0.283	**2.023 ± 0.075**	**2.023 ± 0.074**