A Product of Experts Approach to Early-Exit Ensembles

James Urquhart Allingham [1] **Eric Nalisnick** [2]

Abstract

Ensembles are often expensive to evaluate since they require running multiple models—each of which is costly in the case of neural networks. Using ensembles in compute-constrained applications would be much more practical if just a *subset* of the models could be evaluated. We address this issue with a novel product-of-experts-based method for *early-exit* ensembling. We rely on the fact that the product of finite-support probability distributions (e.g., the continuous uniform) has support less than or equal to that of the multiplicands. Thus, by setting a *confidence* threshold, we can stop evaluating ensemble members once the size of the support has been sufficiently reduced. We demonstrate our methodology for both real-value regression and multi-class classification.

1. Introduction

Predictive models are often subjected to dynamic constraints. For example, an autonomous vehicle must make decisions more quickly in an urban environment than in a rural one. Thus we desire our models to have *early-exit* properties: the model can be 'short circuited' and still produce a prediction that is sufficiently close to the output produced by a full execution. Ensuring and quantifying the early prediction to be 'sufficiently close' is the core research challenge, as it is hard to impose such strong constraints on today's most powerful predictive models (e.g., neural networks (NNs)). *Deep ensembles* are one such powerful predictive model: M NNs are trained in parallel, and at test time, the M predictions are aggregated to produce a final output. Unfortunately, it is difficult to use deep ensembles on low-resource devices where it is demanding to evaluate one NN, let alone M NNs. In this work, we propose a deep ensemble formulation that allows for early-exit *guarantees*. Our primary insight is to formulate the ensemble using two assumptions:

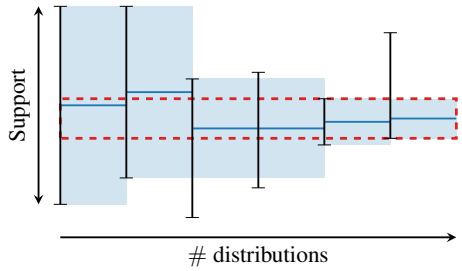


Figure 1: Illustration of successive products of uniform distributions showing the densities ( ) and means (——) of each product given the bounds of the multiplicands (⟼⟻), as well as the final density of the product ( ).

(i) each ensemble member has finite support in its predictions and (ii) the ensemble is trained as a *product of experts*. The resulting ensemble then has the property that *any subset* of members can be evaluated and the whole-ensemble prediction is guaranteed to have non-zero probability. Moreover, as additional members are evaluated, the probability of the whole-ensemble prediction is non-decreasing. We describe practical implementations for real-valued regression and multi-class classification tasks. Our experiments demonstrate that our method is able to achieve the desired early-exit properties on these tasks.

2. Background

Data We assume that our data is availabe as feature-response pairs ($\mathbf{x} \in \mathcal{X}$, $\mathbf{y} \in \mathcal{Y}$), where $\mathcal{X}$ is the feature space and $\mathcal{Y}$ is the response space (i.e., label space, in the case of classification). We then wish to build a predictive model $p : \mathcal{X} \mapsto \mathcal{Y}$. We assume access to an N-sized data set for training: $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^{N}$.

Deep Ensembles Deep ensembles (DEs; Lakshminarayanan et al., 2017) are comprised of $M \in \mathbb{Z}^+$ members, which we denote $p_m(\mathbf{y} \mid \mathbf{x})$ for $m \in [1, M]$. Each $p_m(\mathbf{y} \mid \mathbf{x})$ is parameterized by a NN, which may have parameters that are shared across members. We assume each member has support $\mathcal{Y}_m$, which may or may not be equal to the true response support $\mathcal{Y}$. From a probabilistic perspective, the ensemble can been seen as a mixture model:

$$p_\Sigma(\mathbf{y} \mid \mathbf{x}) = \sum_{m=1}^{M} w_m \cdot p_m(\mathbf{y} \mid \mathbf{x}), \qquad (1)$$

[1]University of Cambridge, UK [2]University of Amsterdam, Netherlands. Correspondence to: James Urquhart Allingham <jua23@cam.ac.uk>, Eric Nalisnick <e.t.nalisnick@uva.nl>.

DyNN workshop at the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, 2022.

where w_m are weights such that $\sum_m w_m = 1$, and $w_m \geq 0 \; \forall \; m$. In practice, the weights are usually fixed at $w_m = 1/M$. In theory, we could draw predictions from the ensemble by sampling a member and then sampling from that model, but in practice, all members are evaluated and their predictions combined (e.g., by voting, averaging).

Product of Experts An alternative to treating an ensemble as a mixture model is to treat it as a *product of experts* (PoE; Hinton, 2002):

$$p_\Pi\left(\mathbf{y} \mid \mathbf{x}\right) = \frac{1}{Z} \prod_{m=1}^{M} p_m\left(\mathbf{y} \mid \mathbf{x}\right)^{w_m}, \tag{2}$$

where w_m is a weight and Z is the *partition function* that ensures the density is normalised:

$$Z = \int_{\mathbf{y}} \prod_{m=1}^{M} p_m\left(\mathbf{y} \mid \mathbf{x}\right)^{w_m} d\mathbf{y}.$$

3. Early-Exit PoE Ensembles

We propose early-exit ensembles: an ensemble whose members can be evaluated in any subset and the combined prediction is guaranteed to place non-zero probability on the full-ensemble's prediction. The key insight is that PoE ensembles are essentially computing an *intersection* of the predictions of their members. We can write this property formally as follows. Let each ensemble component support a *restricted* set of predictions: $\mathcal{Y}_m \subset \mathcal{Y}$. Then due to the PoE being, by definition, a product, the probability of a particular $\mathbf{y} \in \mathcal{Y}$ is non-zero only when it is in the support $\mathcal{Y}_{\pi(1)} \cap \ldots \cap \mathcal{Y}_{\pi(m)} \cap \ldots \mathcal{Y}_{\pi(M)}$ where $\pi(1), \ldots, \pi(M)$ is any permutation of $[1, M]$. Moreover, let $\bar{\boldsymbol{y}}$ denote the modal prediction under the full PoE ensemble $p_\Pi\left(\mathbf{y} \mid \mathbf{x}\right)$. We then have for any subset $\pi(1), \ldots, \pi(J)$, for $J \leq M-1$, we have that $p_{1:J}\left(\bar{\boldsymbol{y}} \mid \mathbf{x}\right) \leq p_{1:J+1}\left(\bar{\boldsymbol{y}} \mid \mathbf{x}\right)$. In the following subsections we describe two practical implementations that can achieve this property.

We propose to leverage this property of the product of experts, to select the number of ensemble members to evaluate for a given input at *test time*. Specifically, we propose to specify a *confidence* threshold, which could be chosen via a held-out validation set, that represents the minimum size of the product distribution's support after which no more multiplicands will be included in the product.

3.1. Real-Valued Regression

A continuous distribution with restricted support is the uniform. The product of uniform distributions $\mathcal{U}_m\left(a_m, b_m\right)$ has a very simple form:

$$p_\Pi\left(\mathbf{y} \mid \mathbf{x}\right) = \mathcal{U}_\Pi\left(a_{\max}, b_{\min}\right), \tag{3}$$

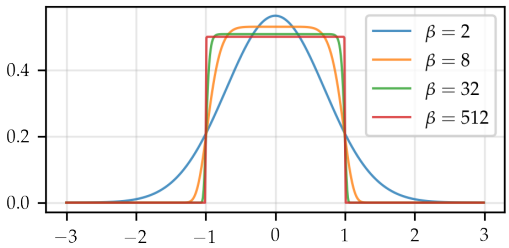


Figure 2: Comparison of GND densities while varying the shape parameter β, and fixing $\mu = 0$ and $\alpha = 1$. As $\beta \to \infty$ the GND converges to a uniform distribution.

where $a_{\max} = \texttt{max}(\{a_m\})$ and $b_{\min} = \texttt{min}(\{b_m\})$. That is, the lower and upper bounds of the product are, respectively, the maximum-lower and minimum-upper bounds of the multiplicands. As we add more ensemble members, the width of the product density can only get smaller or stay the same. This behaviour is shown in Figure 1. From a probabilistic perspective, the product model becomes more certain, as additional ensemble members are incorporated. Note that $a_{\max}$ could be larger than $b_{\min}$. This occurs when there is no overlap in the support of the multiplicands and corresponds to $Z = 0$. We discus this in Section 3.3.

Generalised Normal Approximation Unfortunately, training NNs with uniform likelihoods is not possible due to the lack of gradients from the loss function—the true label either is or isn't inside the bounds of the uniform. Thus, we must resort to approximation. We replace the uniform distribution with a generalised Normal distribution (GND). The GND is a continuous distribution whose density

$$\mathcal{G}(\mathrm{x} \mid \mu, \alpha, \beta) = \frac{\beta}{2\alpha\Gamma(\beta^{-1})} \exp\left[-\left(\frac{|\mathrm{x} - \mu|}{\alpha}\right)^{\beta}\right], \tag{4}$$

is parametrised by a location μ, a scale α, and a shape β. As shown in Figure 2, by varying the shape parameter, we can interpolate between a Normal distribution (with scale $\alpha/\sqrt{2}$) for $\beta = 2$, and a uniform distribution (with bounds $\mu \pm \alpha$) as $\beta \to \infty$. By gradually increasing the shape parameter β from a low value (e.g., 2) to a high value (e.g., 32) during the course training, we are able to train an ensemble of approximately uniform distributions. It is important to gradually increase the shape parameter since early in training the ensemble members make random predictions which do not match the data distribution. This is problematic for two reasons. Firstly, the higher the value of β, the smaller the gradients are, which makes training very slow. Secondly, for training examples which are even slightly outside of the range $[\mu - \alpha, \mu + \alpha]$, the log-likelihood will be increasingly negative with larger βs, resulting in numerical instability.

Replacing uniform distributions with GNDs poses a new challenge—the product of GNDs does not have a closed form solution. For high β values we could approximate the

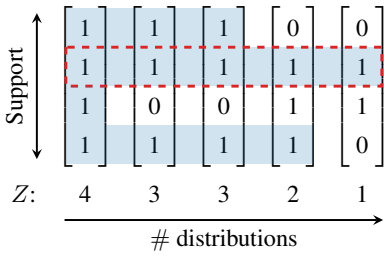


Figure 3: Four-class illustration of successive product distributions with hard one-vs-rest classifiers showing the support for the classes ( ) and the normalizing constant (Z) for each product, as well as the support for the final product ( ).

product of GNDs as a product of uniforms, however, early in training when β is small this will be a poor approximation. Luckily, for most regression problems the dimensionality of $\mathbf{y}$ is small, thus we can use numerical integration to approximate Z. We find that the using trapezoid rule rule

$$Z = \int \prod_{m=1}^{M} p_m\left(\mathrm{y} \,|\, \mathbf{x}\right)^{w_m} d\mathrm{y} \tag{5}$$

$$\approx \sum_{k=1}^{K} \frac{\prod_m p_m\left(\mathrm{y}_{k-1}\right)^{w_m} + \prod_m p_m\left(\mathrm{y}_k\right)^{w_m}}{2} \Delta_{\mathrm{y}} \tag{6}$$

tends to work well.

3.2. Multi-Class Classification

To extend our method to classification, we require a discrete distribution whose product (i) has support smaller than or equal to the multiplicands, (ii) for which (i) holds for any subset of the multiplicands. One such distribution that meets these desiderata is the hard one-vs-rest categorical:

$$p_m\left(\mathrm{y} \,|\, \mathbf{x}\right) = \prod_{k=1}^{K} [\sigma(f_k) > 0]^{[\mathrm{y}=k]} \cdot [\sigma(f_k) \leq 0]^{[\mathrm{y}\neq k]}, \tag{7}$$

where k indexes the classes, $\sigma(\cdot)$ is the logistic function, f_k is the kth output of a NN $f(\mathbf{x})$, and $[\mathtt{cond}]$ is the Iverson bracket which takes the value 1 if $\mathtt{cond}$ is true and 0 otherwise. 0^0 is defined to be 1. We illustrate this distribution in Figure 3. Here the normalizing constant of $p_\Pi\left(\mathrm{y} \,|\, \mathbf{x}\right)$ is

$$Z = \sum_{\mathrm{y}'} \prod_{k=1}^{K} [\sigma(f_k) > 0]^{[\mathrm{y}'=k]} \cdot [\sigma(f_k) \leq 0]^{[\mathrm{y}'\neq k]}, \tag{8}$$

and takes a value in $[0, K]$. That is, Z is equal to the number of classes in the support of the product distribution.

Tempered Approximation Unfortunately, as with the uniform distribution, our hard one-vs-rest categorical does not provide gradients for training. Thus, we once again have to resort to approximation. In this case, we start with a standard one-vs-rest categorical and increase a temperature T to make the distribution harder throughout training:

$$p_m\left(\mathrm{y} \,|\, \mathbf{x}\right) = \prod_{k=1}^{K} \sigma(T f_k)^{[\mathrm{y}=k]} \cdot (1 - \sigma(T f_k))^{[\mathrm{y}\neq k]}. \tag{9}$$

Note that as $T \to \infty$, (9) better approximates (7). Furthermore, since NNs tend to be overconfident, this is a reasonable approximation even for smaller temperatures.

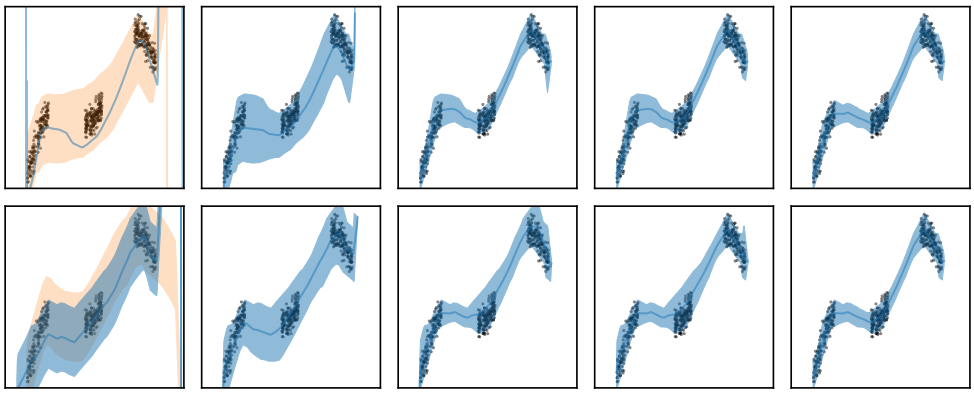
3.3. Support Collapse to the Empty Set

When none of the experts agree on the prediction for a particular input and the product's support has collapsed to the empty set, the normalizing constant Z is 0. Conceptually, each of the experts has vetoed some of the support for the prediction until there no support anywhere. We consider such inputs to be out of distribution (OOD).

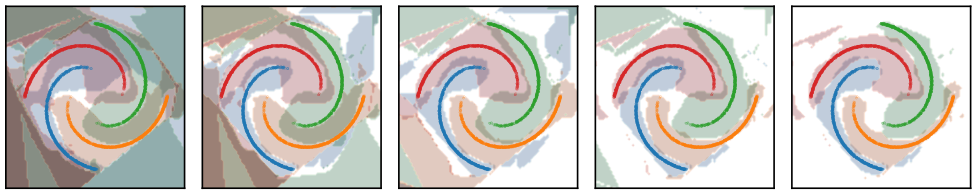
4. Results

In Figure 4a, we examine the the fit for a 5-member early-exit ensemble trained using the GND approximation. We show the fit as progressively more members are included in the prediction, with two different orderings. We see that the order of evaluation is unimportant. Firstly, as expected, the final result is the same, and in both cases the the support size always decreases or is unchanged. Secondly, within a few evaluations the fit has almost converged and the last few multiplicands tend to have smaller effects. Note that even for the poorer early fits, the prediction bounds cover the data well. Finally, note that there is no support outside of the training data range—this data is considered OOD. Figure 4b, shows that early-exit ensembles can be applied to multi-class classification. As expected, the support for each input shrinks as we add more multiplicands to the product.

Figure 6 compares our early-exit ensemble to a standard deep ensemble on the MNIST dataset. The entropy decays smoothly as more members are evaluated for the early-exit ensemble, but remains constant for the DE, suggesting that entropy could be used as a confidence threshold to trigger early exiting at test time. However, we also see that the test error and Brier score of the early-exit ensemble are worse than the DE. Noting that the product NLL encourages the ensemble but not necessarily the individual ensemble members to fit the data, we tried to improve performance by adding an additional loss term which encourages each of the members to fit the data. While this reduced the performance gap, it did not remove it. NLL is more favourable for early-exit ensembles, however, these results are not directly comparable because we have allowed the early-exit ensemble to reject 'OOD' inputs in the test set, which the other models are unable to do. Since the test set is sampled from the same distribution as the training set, we should not



(a) Comparison of two different orderings (top vs bottom) on the *simple 1d* dataset (Antorán et al., 2020). In each column the product support (■) and mean (——), as well as the support (■) of the multiplicand to be included in the next column's product are shown.



(b) Support for each of 4-classes (■, ■, ■, ■) in a simple spirals classification task.

Figure 4: Evolution of a 5-member early-exit ensemble as 1 to 5 (left to right) members are included in the prediction .

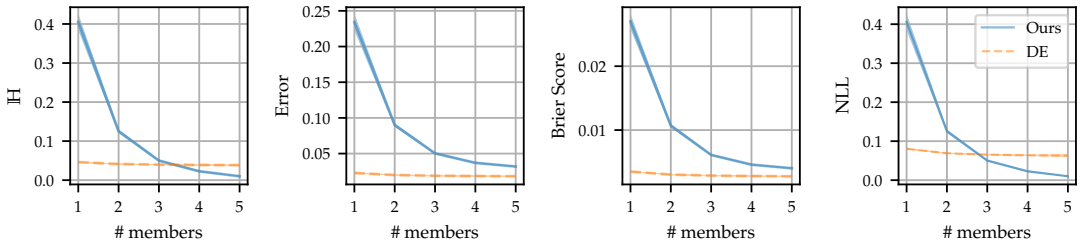


Figure 5: Predictive entropy ($\mathbb{H}$), classification error (Error), Brier score, and negative log-likelihood (NLL) for a product of hard one-vs-rest classifiers (Ours), and a standard deep ensemble (DE), evaluated on the MNIST test set.

expect any inputs to be rejected. Yet they sometimes are, indicating overconfidence of the early-exit ensemble.

5. Related Work

Endowing neural networks with early-exit properties has received much attention of late (Scardapane et al., 2020b; Laskaridis et al., 2021). Many of these approaches rely on parameterized gates that determine when to exit and/or how much weight to place on an exit (Graves, 2016; Teerapittayanon et al., 2016; Scardapane et al., 2020a). The only work we are aware of that considers ensembling and early exiting is that of Qendro et al. (2021). However, this work re-interprets early-exit architectures as an ensemble. We, on the other hand, are interested in endowing traditional ensembling with an early termination property. In this regard, our work is more related to work that constructs ensembles progressively, such as *boosting* (Schapire, 1999; Grubb & Bagnell, 2012). Yet our work allows for ensemble members to be trained in parallel, not sequentially as in boosting.

6. Conclusion & Future Work

We have described a practical algorithm for early-exit ensembles that uses a novel product-of-experts formulation to provide guarantees. We have demonstrated that our algorithm can be applied to real-valued regression and multi-class classification tasks. However, we have also seen that, when early exiting, predictive performance of our method is poor versus standard ensembles. A first step in future work is to remedy this short coming. It is possible that a more rigorous hyperparameter search could be the solution– the standard hyperparameters were based on the DE and early-exit specific hyperparameters were not exhaustively searched. Our approach must also be scaled up and validated on more realistic settings. We believe that investigating the OOD detection capabilities of our method would be fruitful.

Acknowledgements

The authors would like to thank José Miguel Hernández-Lobato, Christos Louizos, and Javier Antorán for helpful discussions. JUA acknowledges funding from the EPSRC, the Michael E. Fisher Studentship in Machine Learning, and the Qualcomm Innovation Fellowship.

References

Antorán, J., Allingham, J. U., and Hernández-Lobato, J. M. Depth uncertainty in neural networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/781877bda0783aac5f1cf765c128b437-Abstract.html.

Graves, A. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Grubb, A. and Bagnell, D. Speedboost: Anytime prediction with uniform near-optimality. In Lawrence, N. D. and Girolami, M. (eds.), *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pp. 458–466, La Palma, Canary Islands, 21–23 Apr 2012. PMLR. URL https://proceedings.mlr.press/v22/grubb12.html.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6402–6413, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/9ef2ed4b7fd2c810847ffa5fa85bce38-Abstract.html.

Laskaridis, S., Kouris, A., and Lane, N. D. Adaptive inference through early-exit networks: Design, challenges and directions. In *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning*, pp. 1–6, 2021.

Qendro, L., Campbell, A., Lio, P., and Mascolo, C. Early exit ensembles for uncertainty quantification. In Roy, S., Pfohl, S., Rocheteau, E., Tadesse, G. A., Oala, L., Falck, F., Zhou, Y., Shen, L., Zamzmi, G., Mugambi, P., Zirikly, A., McDermott, M. B. A., and Alsentzer, E. (eds.), *Proceedings of Machine Learning for Health*, volume 158 of *Proceedings of Machine Learning Research*, pp. 181–195. PMLR, 04 Dec 2021. URL https://proceedings.mlr.press/v158/qendro21a.html.

Scardapane, S., Comminiello, D., Scarpiniti, M., Baccarelli, E., and Uncini, A. Differentiable branching in deep networks for fast inference. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4167–4171. IEEE, 2020a.

Scardapane, S., Scarpiniti, M., Baccarelli, E., and Uncini, A. Why should we add early exits to neural networks? *Cognitive Computation*, 12(5):954–966, 2020b.

Schapire, R. E. A brief introduction to boosting. In *Ijcai*, volume 99, pp. 1401–1406. Citeseer, 1999.

Teerapittayanon, S., McDanel, B., and Kung, H.-T. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2464–2469. IEEE, 2016.

A. Implementation Details

Our models are implemented using `jax`. We use `flax` to build neural networks, `optax` for optimisation, and `difrax` for constructing probability distributions.

All ensembles are constructed from 5 members. The members are multi-layer perceptrons (MLPs) with residual connections. The MLPs each consist of a `Dense` input layer, followed by D residual blocks and an output layer. The residual blocks consist of a `Dense` hidden layer followed by a `ReLU` activation and `BatchNorm`. For regression, we used a hidden width of 50 and $D = 2$ residual blocks. For classification, we used a hidden width of 100 and $D = 5$ residual blocks.

We initialise the weights using the same fan-in uniform variance scaling scheme as `PyTorch`. Similarly, for our `BatchNorm` layers, we set the `epsilon` to 1×10^{-6} and `momentum` to 0.9 to match the `PyTorch` defaults.

We train our models using SGD with learning rates of 1×10^{-4} and 3×10^{-3} for the toy regression/classification and MNIST respectively, a momentum of 0.9, and a weight decay of 1×10^{-4}. We train for 200 and 50 epochs, for the toy settings and MNIST, respectively. We used a held out validations set to choose the best epoch. For the early-exit ensembles, we only performed validation after β/T was within 95% of the final value. We increased β/T from 2 to 16 using a linear schedule updated per-batch.

For all regression models, we predicted the location parameter μ using an NN, but learnt a homoscedastic scale parameter σ.

For the product of GNDs, we calculate Z in (6) using `jax.numpy.trapz`, which is fully differentiable. We use $\Delta_y = 0.001$, $y_{min} = -10$ and $y_{max} = 10$.

In Figures 5 and 6, the results are averaged over 3 random seeds and all possible subsets of each ensemble size.

The loss used for training the early-exit ensembles is:

$$\mathcal{L} = -\alpha \cdot \log p_{\Pi}(\mathbf{y} \mid \mathbf{x}) - (1 - \alpha) \cdot \sum_m \log p_m(\mathbf{y} \mid \mathbf{x}), \tag{10}$$

where $\alpha = 0.5$. We found that this improved the performance of the individual ensemble members without reducing the performance of the ensemble itself. Note that the first term–the NLL of the product–says nothing about the fit of the individual ensemble members, only the fit of the product. We also tried pre-training the early-exit ensemble as a standard ensemble, but found that this did not help.

B. Additional Results

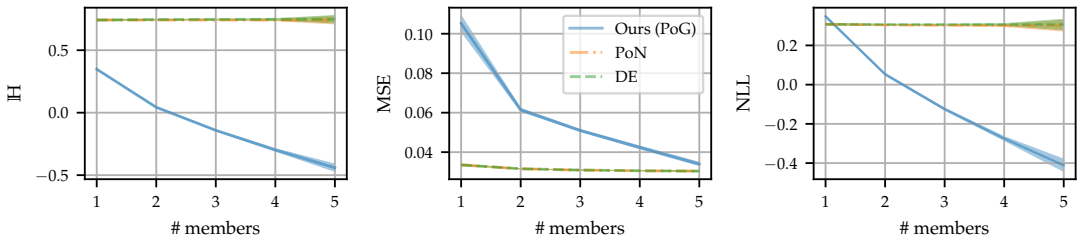


Figure 6: Predictive entropy ($\mathbb{H}$), mean squared error (MSE) and negative log-likelihood (NLL) for a product of GNDs (PoG), a product of normals (PoN), and a standard deep ensemble (DE), evaluated on the test set of *simple 1d*. The results here match those of Figure 5. Additionally, we see that the PoN behaves very similarly to the DE, suggesting that a product formulation is not sufficient for early exiting and that *finite support* is also required.