PonderBayes: Rationally Pondering Neural Networks

Giulio Starace, Matteo Rosati, Victor Kyriacou, Jille van der Togt

Abstract

Conventional neural networks have their computational power equally distributed over their inputs but not over the complexity of the problem at hand. To overcome this Banino et al. [1] proposed **PonderNet**, a new algorithm that learns to **adapt the amount of computation based on the complexity of the problem**. We explore **approaches for a Bayesian treatment of PonderNet**, hypothesizing that quantifying model **uncertainty** can lead to better-informed model decisions. We implement four ensemble-based variants, one of which **achieves higher accuracy and faster convergence time on varying difficulty levels of the same task.** We also consider a more complete Bayesian treatment of the problem but ultimately run into the limitations of the current ecosystem. Our code is publicly available (see QR code).

Contributions

- We **reproduce** the findings of the original PonderNet paper, providing (not previously available) code (see QR code).
- We apply PonderNet to a new benchmarking dataset, **MNIST**.
- We propose **four "ensemble" PonderNet variants** for uncertainty quantification and exploitation.
- We explore the feasibility of a **more complete Bayesian treatmen**t of the problem through Stochastic Variational Inference (SVI).

Data and Evaluation

Evaluation: We distinguish between *interpolation* and *extrapolation* tasks. In the former, the training and evaluation data share the same difficulty. In the latter, the evaluation data is in some way more difficult.

MNIST [2] consists of 28x28 grayscale images of handwritten digits, with 60k training and 10k testing samples. The task is to classify inputs as one of the digits between 0 and 9. For interpolation, we leave the inputs untouched for both training and testing. For extrapolation, we randomly rotate the images in the testing set by degrees in the range [−112, 112].

Parity: the data consists of P-dimensional vectors of which a random number of elements $Q \geq 1$ is set to 1 or −1, and the remaining elements are set to 0. The task is to output 1 if there is an odd number of ones (including negatives) and 0 if there is an even number. For interpolation, $Q \in \{1, ..., P\}$ for both training and testing samples. For extrapolation, $Q \in \{1, ..., P/2\}$ during training and $Q \in \{P/2 + 1, ..., P\}$ during evaluation. For our work, we set $P = 16$ for interpolation and $P = 24$ for extrapolation. we generate 128,000 training samples and two pairs of 25,600 evaluation samples.

Models

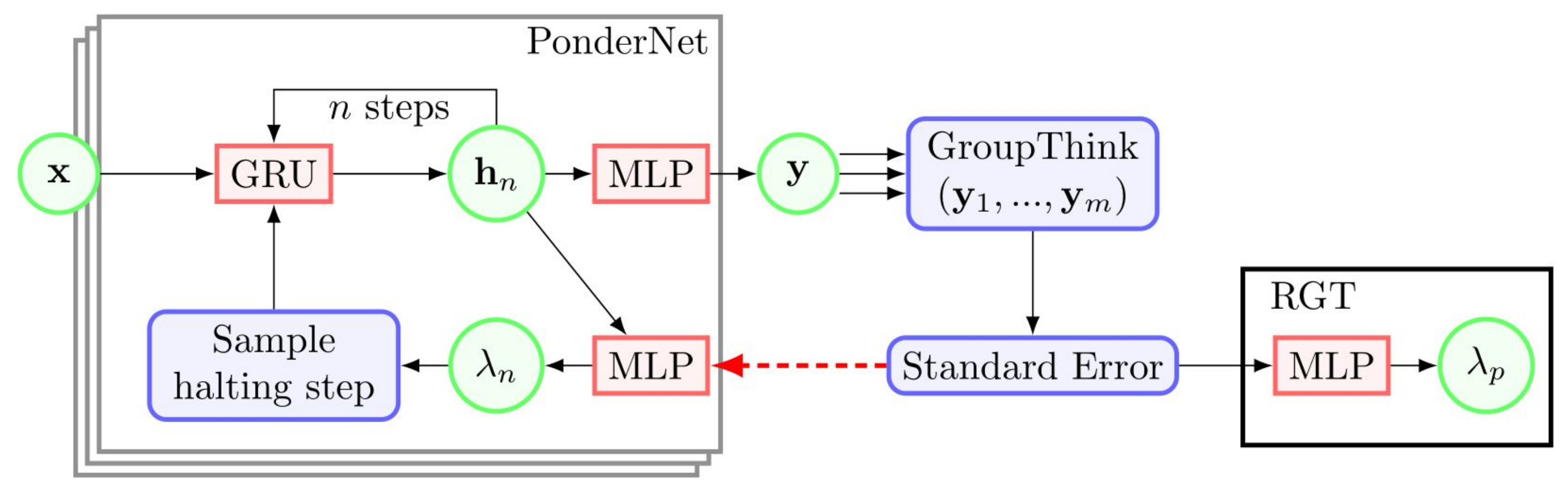

Figure 1: A graphical representation of the implemented models on the parity task modifying the original PonderNet model. The red dashed arrow represents the lambdaGT model, i.e., the standard error from GroupThink being used for the input to the λ_n layer. The additional architecture implemented in RGT and aRGT can be found in the black rectangle.

PonderNet

Requires a step function s of the form $y_n, h_{n+1}, \lambda_n = s(x, h_n)$ and an initial state h_0, where y_n and λ_n are the network outputs and scalar probability of "halting" at step n. Uses λ_n to learn the optimal value for n and uses a Bernoulli random variable Λ_n in order to represent a "continue" ($\Lambda_n = 0$) state and a terminal "halt" ($\Lambda_n = 1$) state. The transition probability is then

$$P(\Lambda_n = 1 | \Lambda_{n-1} = 0) = \lambda_n \quad \forall 1 \leq n \leq N.$$

The prediction $y \sim Y$ is sampled with $P(Y = y_n) = p_n$, where

$$p_n = \lambda_n \prod_{j=1}^{n-1} (1 - \lambda_j).$$

To train PonderNet, the sum of a reconstruction loss L_{Rec} and regularization loss L_{Reg} is optimized

$$L = \underbrace{\sum_{n=1}^{N} p_n \mathcal{L}(y, \hat{y}_n)}_{L_{Rec}} + \beta \underbrace{\mathrm{KL}(p_n || p_G(\lambda_p))}_{L_{Reg}},$$

$p_G(\lambda_p)$ is a pre-determined prior geometric distribution parameterized by the hyperparameter λ_p. By using the KL divergence between p_n and p_G, L_{Reg} not only biases the network toward the expected number of ponder steps $1/\lambda_p$ but also promotes exploration by incentivizing non-zero probability of halting for each step

A Fully Bayesian Approach

We took initial steps in the direction of learning the predictive posterior distribution $p(y^*|x^*, D)$ by learning the joint distribution $p(W, D)$ on the weights W of the layer using Stochastic Variational Inference (SVI) and Pyro [3]. Ultimately, this approach was unsuccessful due to significant variance and a lack of maturity in the development ecosystem.

Ensembles

GroupThink*:* 5 PonderNet modules, each trained independently using its own optimizer and loss instances. Predictions aggregated to quantify uncertainty.

Rational GroupThink (RGT): Extends GroupThink by learning a map from uncertainty to λ_p, so to update its priors on the expected ponder time based on how certain it is about its current outputs.

Annealed RGT (aRGT): Same as RGT, but schedules an annealing of L_{Reg} to encourage a greater exploration of λ_p values at the beginning of training.

LambdaGT: Extends GroupThink by allowing the model to incorporate its uncertainty when outputting λ_n. The uncertainty at each ponder step is concatenated with the next hidden state as input to the linear layer.

Results

Table 1: Average test set accuracy and halting step of the models on the parity task for interpolation and extrapolation. The standard error in the metrics from runs using 5 different seeds is also reported.

	Test Accuracy		Halting Step	
	Interpolation	Extrapolation	Interpolation	Extrapolation
PonderNet	0.992 ± 0.005	0.908 ± 0.076	8.5 ± 1.4	10.2 ± 1.8
PonderNet MNIST	0.992 ± 0.0002	0.630 ± 0.004	9.1 ± 0.1	10.6 ± 0.3
GroupThink	0.986 ± 0.009	0.883 ± 0.049	8.9 ± 0.3	9.3 ± 0.3
RGT	0.689 ± 0.084	0.502 ± 0.001	9.1 ± 3.1	2.4 ± 0.2
aRGT	0.745 ± 0.093	0.502 ± 0.001	6.4 ± 1.7	3.8 ± 1.2
lambdaGT	$\mathbf{0.995 \pm 0.0002}$	$\mathbf{0.919 \pm 0.022}$	8.7 ± 0.3	9.9 ± 0.3

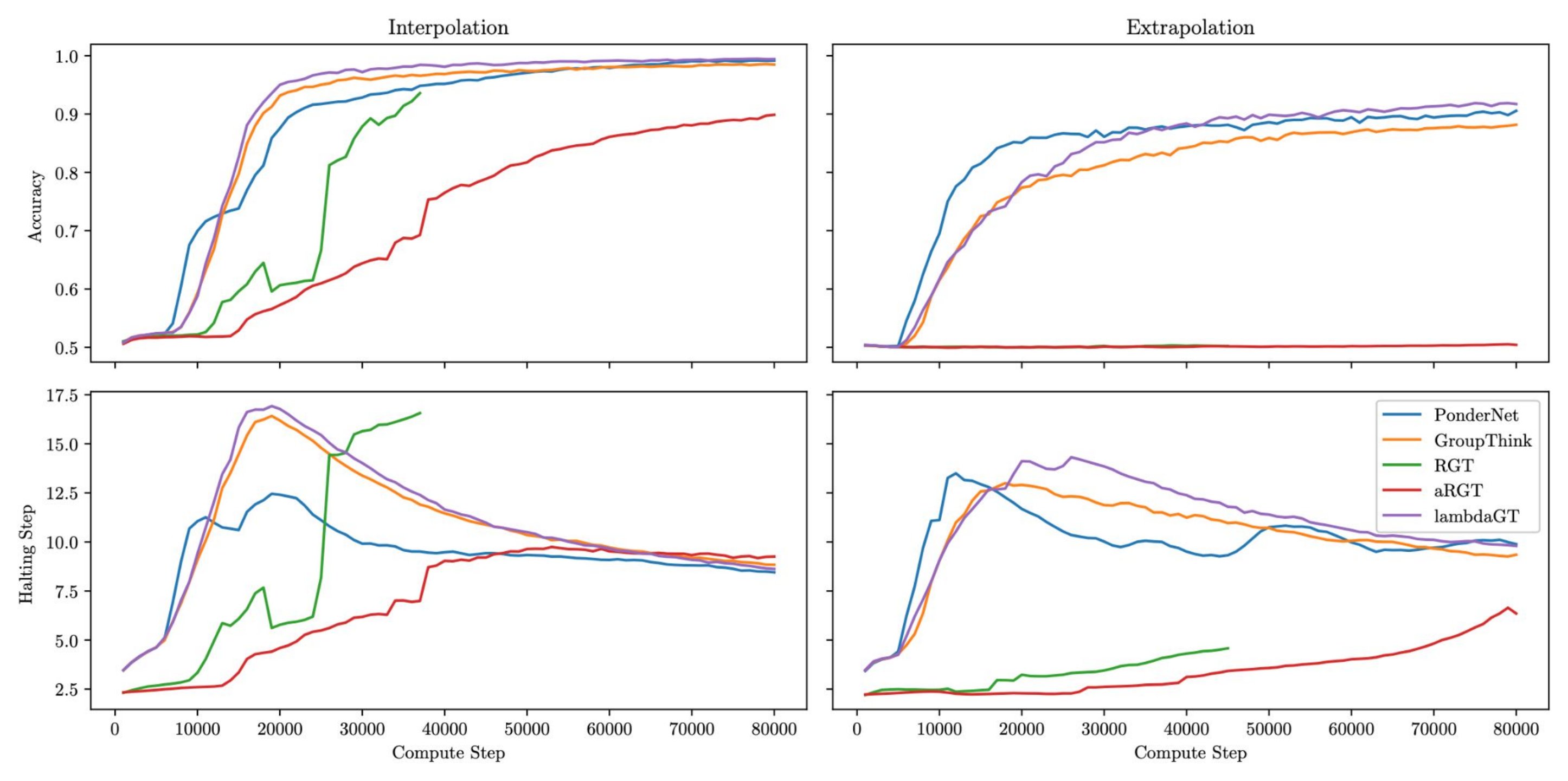

Figure 2: Average validation accuracy and halting step over compute steps for the four models. Average was computed over 5 random seeds.

References

[1] A. Banino, J. Balaguer, and C. Blundell, 'PonderNet: Learning to Ponder', presented at the 8th ICML Workshop on Automated Machine Learning (AutoML), May 2021

[2] L. Deng, 'The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]', IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 141–142, Nov. 2012, doi: 10.1109/MSP.2012.2211477.

[3] E. Bingham et al., 'Pyro: Deep Universal Probabilistic Programming', Journal of Machine Learning Research, vol. 20, no. 28, pp. 1–6, 2019.