DeepMind

PonderNet: Learning to Ponder

Andrea Banino*, Jan Balaguer*, Charles Blundell

* Equal Contribution

Introduction

1. Standard neural networks have fixed computational complexity.
2. Problems have an inherent complexity that is independent of input size
3. Adjusting the computational budget ("ponder time") of an algorithm can improve its accuracy, as well as its generalization capabilities to examples not seen during training.
4. Prior methods to learn to ponder suffer from unstable learning, require brittle fine-tuning of parameters and/or do not scale up.
5. We present a novel approach to learning to ponder in deep neural networks named PonderNet.

Contributions

1. **Architecture:** in PonderNet, the halting node predicts the probability of halting conditional on not having halted before. We compute the exact probability of halting at each step as a geometric distribution.

2. **Loss:** we don't regularize PonderNet to explicitly minimize the number of computing steps, but incentivize exploration instead. The pressure of using computation efficiently happens naturally as a form of Occam's razor.

3. **Inference:** PonderNet is probabilistic both in terms of number of computational steps and the prediction produced by the network.

Method

PonderNet assumes a trainable step function **s** of the form:

$$\overset{\text{prediction}}{\hat{y}_n}, \overset{\text{updated hidden state}}{h_{n+1}}, \overset{\text{halting node}}{\lambda_n} = s(\overset{\text{input}}{x}, \overset{\text{hidden state}}{h_n})$$

PonderNet unrolls the step function for up to **N** computation steps, and derives the exact probability of stopping after **n** computation steps:

$$\overset{\text{prob of n computation steps}}{p_n} = \overset{\text{prob of halting at step n}}{\lambda_n} \prod_{j=1}^{n-1} \overbrace{(1-\lambda_j)}^{\text{prob of not halting in steps 1... n-1}}$$

The loss function of PonderNet is:

$$L = \underbrace{\sum_{n=1}^{N} p_n \overset{\text{pre-defined loss}}{\mathcal{L}}(y,\hat{y}_n)}_{L_{Rec} = \text{expected loss over ponder time}} + \overset{\text{hyper parameters}}{\boxed{\beta}} \underbrace{KL(p_n || \overset{\text{geometric prior}}{p_G}(\boxed{\lambda_p}))}_{L_{Reg} \text{ promotes exploration of ponder time}}$$

Experiment 1: Parity task

In this task, the input is an array of fixed sized filled with ±1 and 0. The task is to learn whether the number of non-zero items is odd or even. When evaluated on a held-out test set, we found that PonderNet was more accurate (**Fig 1a top**) and required less compute steps (**Fig 1a bottom**) than SotA baseline methods (ACT; Graves, 2016). We also found that PonderNet's accuracy extrapolated better outside of the training distribution, and devoted more computation during extrapolation trials (**Fig 1b**; training with inputs including 1-48 ones, evaluating with 49-96 ones).

We also explored the effect of the regularization hyper-parameters, and found that both accuracy and compute steps were more robust to hyper-parameter choice in PonderNet (**Fig 2b** and **2c**) than in ACT (**Fig 2a**).

Figure 1

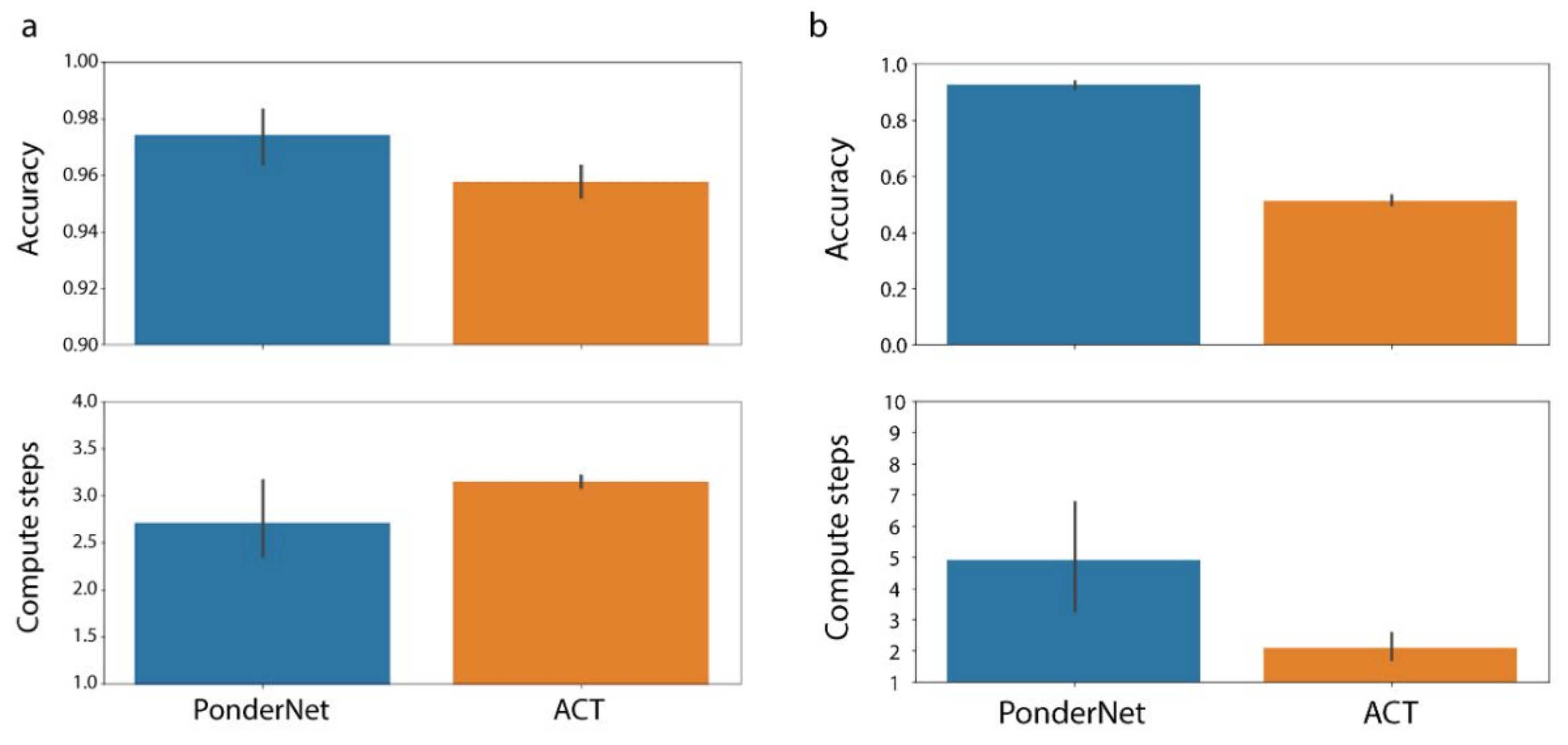
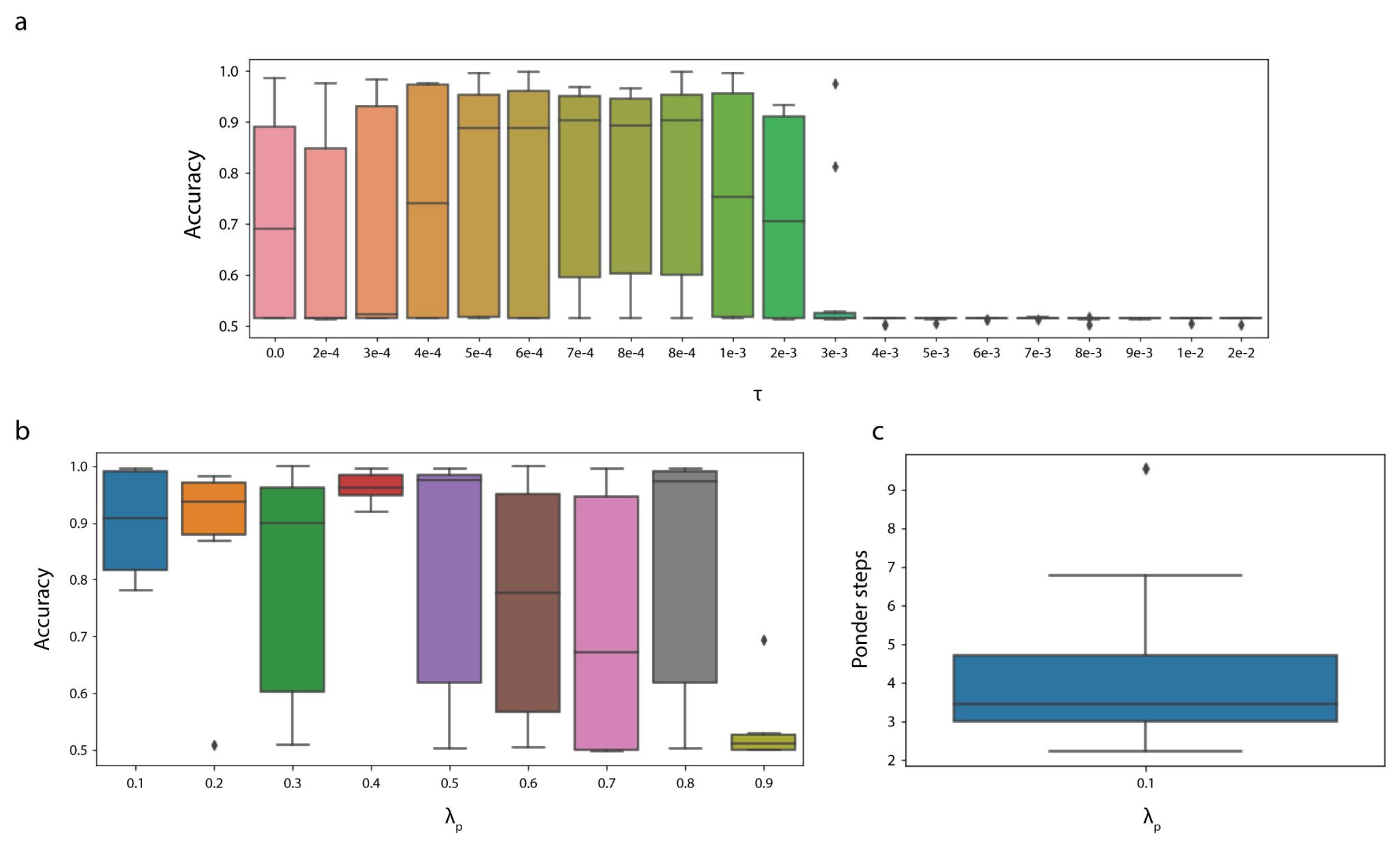


Figure 2



Experiment 2: bAbI

We trained PonderNet on the bAbI question answering dataset (Weston et al, 2015) on the joint 10k training set. We found that PonderNet matched SotA results, faster and with a lower average error.

Architecture	Average Error	Tasks Solved
Memory Networks (Sukhbaatar et al., 2015)	4.2± 0.2	17
DNC (Graves, 2016)	3.8± 0.6	18
Universal Transformer (Dehghani et al., 2018)	0.29± 1.4	20
Transformer+PonderNet	0.15± 0.9	20

Exp 3: Paired Associative Inference

This task is thought to capture the essence of reasoning – the appreciation of distant relationships among elements distributed across multiple facts or memories. It has been shown to benefit from the addition of adaptive computation. PonderNet matched the performance of SotA methods (Banino et al, 2020) even though it was not designed specifically for this task.

Length	UT	MEMO	PonderNet
3 items (trained on: A-B-C - accuracy on A-C)	85.60	98.26(0.67)	97.86(3.78)

Discussion

We introduced PonderNet, a new algorithm for learning to adapt the computational complexity of neural networks. It optimizes a novel objective function that combines prediction accuracy with a regularization term that incentivizes exploration over the pondering time.

We demonstrated on the parity task that a neural network equipped with PonderNet can increase its computation to extrapolate beyond the data seen during training. Also, we showed that our methods achieved the highest accuracy in complex domains such as question answering and multi-step reasoning.

Adapting existing recurrent architectures to work with PonderNet is very easy: it simply requires to augment the step function with an additional halting unit, and to add an extra term to the loss. Critically, we showed that this extra loss term is robust to the choice of the hyper-parameter that defines a prior on how likely is that the network will halt, which is an important advancement over ACT.