Neural Abstract Reasoner

Victor Kolev[*]
victor.kolev@yahoo.com

Bogdan Georgiev[†]
bogdan.m.georgiev@gmail.com

Svetlin Penkov [‡]
svet@sciro.ai

Abstract

Abstract reasoning and logic inference are difficult problems for neural networks, yet essential to their applicability in highly structured domains. In this work we demonstrate that a well known technique such as spectral regularization can significantly boost the capabilities of a neural learner. We introduce the Neural Abstract Reasoner (NAR), a memory augmented architecture capable of learning and using abstract rules. We show that, when trained with spectral regularization, NAR achieves 78.8% accuracy on the Abstraction and Reasoning Corpus, improving performance 4 times over the best known human hand-crafted symbolic solvers. We provide some intuition for the effects of spectral regularization in the domain of abstract reasoning based on theoretical generalization bounds and Solomonoff's theory of inductive inference.

1 Introduction

Extracting and reasoning with abstract concepts is a crucial ability for any learner that is to operate in combinatorially complex open worlds or domains with limited or structured data. It is well known that neural learners struggle to operate in such conditions due to their poor generalization capabilities in structured domains [5, 15]. In this work, we demonstrate that spectral regularization provides neural networks with a strong inductive bias towards learning and utilizing abstract concepts akin to a symbolic learner.

For that purpose, we employ the Abstraction and Reasoning Corpus (ARC) [5] which contains tasks related to manipulating colored patterns in a grid. In order to successfully solve the tasks in the corpus an agent needs to be able to count, manipulate numbers, work with topological and geometric concepts as well as recognise the notion of objects. There are 400 training tasks and 400 (distinct) evaluation tasks. Each task has a small set of input-output example pairs (between 1 and 5) and a query input pattern. This is quite a challenging dataset due to the small amount of example data, large number of different tasks and their abstract nature.

So far, the best known solution, with a success rate of 20%, is the winner of the ARC Kaggle challenge, and it is a carefully hand-crafted symbolic solver written in approx. 7k lines of C++ code. In this paper, we introduce the Neural Abstract Reasoner (NAR) which achieves an accuracy rate of 79% and so outperforming even the best symbolic solver created by a human. The NAR architecture contains a Differentiable Neural Computer (DNC) that learns general problem solving skills and a Transformer network responsible for solving particular task instances (see. Fig 2 in Chollet [5]).

Importantly, spectral regularization plays a fundamental role in the successful training of NAR. From a purely machine learning perspective, spectral regularization is known to reduce the effective number of parameters in the network, however we provide some additional theoretical intuition and demonstrate that spectral regularization also pushes the network towards finding algorithmically simpler solutions as recommended by Solomonoff's theory of inductive inference [13].

[*]Sofia High School of Mathematics, Bulgaria
[†]Fraunhofer IAIS, Research Center for ML (FZML) and Competence Center ML2R, Germany
[‡]Sciro Research, Bulgaria

4th Knowledge Representation and Reasoning Meets Machine Learning Workshop (KR2ML 2020), at NeurIPS.

2 Related Work

Neuro-symbolic architectures Hybrid neuro-symbolic approaches enable agents to solve structured tasks from raw data, while learning faster and being more robust to noise [8, 25, 17, 14]. However, the majority of methods proposed so far are designed with specific domains in mind, making them inapplicable to a broader range of tasks. A notable exception is the architecture proposed by Ellis et al. [6], which is capable of learning rules from geometry, vector algebra, and physics and solve tasks such as drawing pictures or building complete scenes. Importantly, these methods often require lots of data, which is in stark contrast with human capabilities.

The ARC dataset [5] is specially designed to push research towards data efficient learners, as there are hunderds of tasks, each of which is represented by no more than 5 input/output examples. To the best of our knowledge, the Neural Abstract Reasoner, presented in this paper, is the first architecture that achieves a performance rate of 79% on the ARC dataset, outperforming state-of-the-art hand-coded symoblic systems by a factor of 4. The NAR architecture is a composition of a slowly learning Differentiable Neural Computer and a fast adapting Transformer network creating an outer learning and inner executing loops, as suggested in [5].

Complexity and generalization The analysis of complexity and generalization metrics applied to neural networks has formed a central line of theoretical ML research with a variety of recent breakthroughs in terms of PAC-based and compression methods (cf. [1, 2, 10, 21, 26] and the references therein). In particular, many generalization approaches based on spectral norm analysis have been so far proposed and investigated [16, 19]. However, to our knowledge the present work is the first to address the relationship between spectral norms' behaviour and abstract reasoning tasks, whereby a strong relationship between a classical spectral regularisation and the ability of a neural model towards learning abstract reasoning (concepts and rules) is demonstrated. As touched upon in Section 4, one could draw motivation from well-known algorithmic information theory concepts such as Solomonoff inference and program generation based on least complexity [13, 3, 20].

3 Methods

Description The ARC dataset consists of a train and evaluation portions D and D_v, respectively. Each portion consists of 400 tasks (train tasks are augmented to ≈15000 through color permutations and rotations). The individual tasks T are grouped in tags τ based on the skills needed to solve them [4]. A task $T_j = \left\{[i_1^e, o_1^e]; \ldots; [i_5^e, o_5^e]; [i^q, o^q]\right\}$ consists of up to five example input-output pairs and one query pair. A neural learner has to infer a logical rule $\pi : i \rightarrow o$. All inputs and outputs are grids of variable sizes with 10 colors. A solution is correct only when all the pixels on the grid match.

First, we derive a latent representation of the grids with an InceptionNet-style [22] deterministic auto-encoder. Let $G \in \{0, \ldots, 9\}^{10\times 10}$ be a grid, and $\mathcal{E} : G \rightarrow \mathbb{R}^n, \mathcal{D} : \mathbb{R}^n \rightarrow G$ be the encoder and decoder, parametrized by θ. We train an embedding network by minimizing the standard autoencoder cross-entropy loss $\min_\theta : L_e = \sum_{G\in D} H\big(G, \mathcal{D}(\mathcal{E}(G))|\theta\big)$.

Next, we consider all latent grid embeddings, and we train a Differentiable Neural Computer [9] $\mathcal{M}_\mu$ with parameters μ to infer an instruction set ψ. All inputs are processed by a Transformer Decoder Stack $\mathcal{T}$ [24] with parameters η, which self-attends to all inputs and cross-attends to ψ:

$$\psi = \mathcal{M}_\mu\Big(\big\{[i_1^e, o_1^e]; \ldots; [i_5^e, o_5^e]\big\}\Big), \qquad \widehat{o}_j = \mathcal{T}_\eta\big(i_j|\psi; i_{k,k\neq j}\big).$$

The whole model is then trained end-to-end via ADAM [11] to minimize the cross-entropy loss between the query target and the decoded test output prediction.

$$\min_{\mu,\eta} : L = \sum_{T\in D} H\Big(\mathcal{D}(o^q), \mathcal{D}(\widehat{o}^q|\mu, \eta)\Big)$$

We employ a two-stage curriculum during training, first training only a on a single tag τ, and then expanding to the whole train dataset D. Additionally, during the first stage of training, spectral regularization [27] with a larger λ value is applied, which is then annealed in the second stage. When evaluating the model, we apply additional optimization steps (similar to [7, 12]), as described in Algorithm 1. See Appendix C for additional details.

Motivation We build on methods from Santoro et al. [18] and use a memory-augmented neural network (the Differentiable Neural Computer [9]) to derive context for the current task. The multiple read heads and attention mechanisms allow the DNC to relate the input and the output of a pair and compare them to the other input/output pairs that it has already processed. Unlike Santoro et al. [18], we leverage a Transformer to carry out the task execution based on the DNC context. This decouples the learning of the instruction set from the program execution itself, and allows us to use the input/output relations directly, rather than the more standard $[i_t, o_{t-1}]$. Lastly, the Transformer network relates the inputs to each other, thereby exploiting similarities within them.

Performance As this work is still in progress, these are preliminary results evaluated on grids up to 10×10. Nonetheless, we outperform all currently known solutions, including a hand-crafted symbolic solution (see Fig. 1). Spectral regularization proved instrumental for this, and other regularization methods did not yield any significant results (see Fig. 2).

Without any additional adaptation steps [7], evaluation performance remains low at 1%, while only after 3 steps, that number climbs up to 78.8%. Analyzing more closely the network changes made by the adaptation steps, the gradient norm of $\mathcal{M}$ is $< 1 \times 10^{-7}$, which implies that the DNC is acting as a true meta-learner, and only the Transformer requires a small change $[3 \times 10^{-4}, 7 \times 10^{-3}]$ to execute the instructions flawlessly. We again attribute this generalization to spectral regularization.

Algorithm 1: NAR evaluation cycle

Given: $\mathcal{E}, \mathcal{D}, \mathcal{M}_\mu, \mathcal{T}_\eta$; D_v evaluation dataset;
Hyperparameters: α step size; k number of steps
for *Task* $T \in D_v$ **do**
  $\mu_0 \leftarrow \mu$, $\eta_0 \leftarrow \eta$
  for *step in 1:k* **do**
    $\psi = \mathcal{M}_{\mu_{\text{step}-1}}\left(\{[i_1^e, o_1^e]; \ldots; [i_5^e, o_5^e]\}\right)$
    $[\widehat{o}_1^e, \ldots, \widehat{o}_e^5] = \mathcal{T}_{\eta_{step-1}}\left([i_1^e, \ldots, i_5^e, i^q]|\psi\right)$
    Evaluate $\mathcal{L} = \sum_{i=1}^{5} H\left(\mathcal{D}(o_i^e), \mathcal{D}(\widehat{o}_i^e)\right)$
    Adjust parameters:
    $\mu_{step} \leftarrow \mu_{step-1} - \alpha \nabla_{\mu_{step-1}} \mathcal{L}$
    $\eta_{step} \leftarrow \eta_{step-1} - \alpha \nabla_{\eta_{step-1}} \mathcal{L}$
  end
  $\psi = \mathcal{M}_{\mu_k}\left(\{[i_1^e, o_1^e]; \ldots; [i_5^e, o_5^e]\}\right)$
  $[\widehat{o}_1^e, \ldots, \widehat{o}_e^5, \widehat{o}^q] = \mathcal{T}_{\eta_k}\left([i_1^e, \ldots, i_5^e, i^q]|\psi\right)$
  Compute accuracy of $\widehat{o}^q$
end

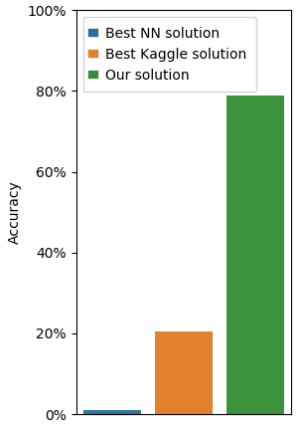

Figure 1: We achieve 78.8% evaluation accuracy on the ARC dataset. The reported results are calculated for 100 unseen tasks from the corpus.

4 Effects of spectral regularization: stable ranks and complexity

The surprisingly significant effect of a simple spectral regularization strategy in the reasoning tasks suggests strong connections with generalization and the underlying model complexity estimates. On one hand, this motivates the analysis of spectral regularization in terms of some well-known generalization bounds (e.g. based on stable rank and spectral norms), however, we first discuss a perspective inspired by algorithmic information theory. Intuitively, abstract reasoning tasks are induced by a concise set of logic rules and combinatorial patterns, and, hence, it is natural to search for *short* programs producing these rules - in this regard, we give motivation as to why spectral regularization naturally shrinks the search space towards *shorter* programs.

Spectral regularization and polynomials as algorithmically simple programs. Classical methods from program inference and algorithmic information theory, such as Solomonoff inference and Occam's razor [13], suggest that "simpler" program models are preferable in terms of forming abstract concepts and generalization - a formal approach towards such issues is given, e.g., by Kolmogorov complexity theory [13, 20]. Although the evaluation of algorithmic complexity is a demanding task (Kolmogorov complexity is theoretically uncomputable), one could attempt to devise various proxy metrics that capture the algorithmic complexity of a given function/program.

Here, in an attempt to evaluate and explain the algorithmic complexity of our models from a spectral-regularization perspective, we consider approximations in terms of a simple but flexible class π_n of programs computing rational-coefficient polynomials of maximal degree $n \in \mathbb{N}$. Intuitively, approximating a model f in terms of π_n for lower values of n corresponds to discovering programs of decreased algorithmic length (in terms of operations) that effectively compute f. In this direction, we bring forward some classical approximation theory results implying that lower spectral norms yield lower degrees of the approximation polynomial. To ease notation, here we discuss the 1-dimensional case, however, similar results hold for higher dimensions as well [23]:

Proposition 1. *Let $f : [0, 1] \to \mathbb{R}$ represent a model with Lipschitz constant L. Then, there exists a polynomial $B_n \in \pi_n$ of degree n, so that $\|f - B_n\|_{L^\infty[0,1]} \leq \frac{3L}{2\sqrt{n}}$, where $\|.\|_{L^\infty[0,1]}$ denotes the usual* sup*-norm over the interval $[0, 1]$.*

Since the Lipschitz constant of a neural model is bounded above by the spectral norms of the layers W_i, spectral regularization gives control over L; moreover, a lower value of L implies that one can decrease the polynomial degree n and retain similar approximation qualities. These observations support our empirical results - introducing spectral regularization steers the model search space towards algorithmically simpler and more robust functions.

Generalization via spectral norms and stable ranks. We recall that the stable rank of a matrix A, $\mathrm{rank_s}(A)$, is defined as the ratio $\|A\|_F^2/\|A\|_2^2$ and note that $\mathrm{rank_s}(A)$ is at most the rank of A. The stable rank is intuitively understood as a continuous proxy to $\mathrm{rank}(A)$ and as a measure for the true parameter count of A. Now, let f be a deep neural model consisting of d layers whose corresponding weight matrices are denoted by $W_i, i = 1, \ldots, d$. Recent works (e.g. [16, 1]) obtain generalization bounds on f, roughly speaking, in terms of the expression $\mathcal{O}\left(\prod_{i=1}^d \|W_i\|_2^2 \sum_{i=1}^d \mathrm{rank_s}(W_i)\right)$, where $\|W_i\|_2$ denotes the spectral norm of the matrix W_i. A related stronger compression-based estimate in terms of so-called noise-cushions is obtained in [1]. In other words, the generalization error is influenced by the spectral norms as well as stable ranks of the layers.

In this direction, we evaluated our model and the effect of stable ranks. Interpreting Fig. 3, one observes that initially while the model adopts to the single task of `pattern_expansion` it increases and stabilizes a true parameter count ~ 8000; afterwards, the model is introduced to the full task bundle where a significant decrease of the stable ranks is observed - according to the last expression this leads to better generalization, and further implies that at the end of training one actually deals with simpler models with better compression properties.

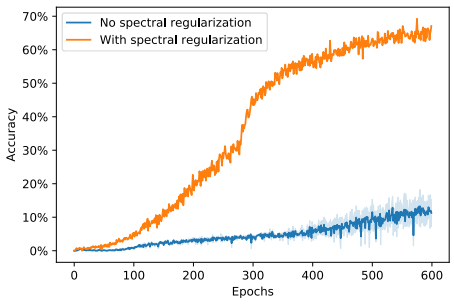

Figure 2: Performance on the `pattern expansion` task from the ARC dataset. The exact same models (DNC Transformer) were trained with and without spectral regularization.

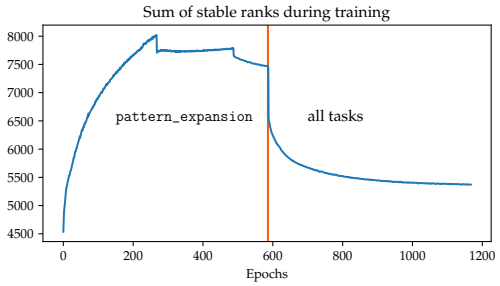

Figure 3: Stable ranks vary drastically depending on the task distribution. With a larger pool of tasks, NN's stable ranks decrease rapidly, as it is optimized for greater generalization, as opposed to specialization to a particular task.

5 Conclusion and Acknowledgements

We have demonstrated that spectral regularization provides neural learners with a significant boost in performance on abstract reasoning tasks. We believe that studying the complexity of the underlying models in the context of powerful frameworks such as Kolmogorov complexity or Solomonoff's theory of inductive inference is a promising step towards closing the neuro-symbolic gap. We would like to thank Dimitar Vasilev (Microsoft Inc.) for the computational resources used in this work.

References

[1] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *35th International Conference on Machine Learning, ICML 2018*, volume 1, pages 390–418, 2018. ISBN 9781510867963.

[2] Peter L. Bartlett, Dylan J. Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, 2017.

[3] Léonard Blier and Yann Ollivier. The description length of deep learning models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2216–2226. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7490-the-description-length-of-deep-learning-models.pdf.

[4] Davide Bonin. Task tagging. *Kaggle public notebook*. URL https://www.kaggle.com/davidbnn92/task-tagging. Accessed on 08.10.2020.

[5] Francois Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

[6] Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sable-Meyer, Luc Cary, Lucas Morales, Luke Hewitt, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: Growing generalizable, interpretable knowledge with wake-sleep bayesian program learning. *arXiv preprint arXiv:2006.08381*, 2020.

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[8] Alexander L Gaunt, Marc Brockschmidt, Nate Kushman, and Daniel Tarlow. Differentiable programs with neural libraries. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1213–1222, 2017.

[9] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[10] Yiding Jiang*, Behnam Neyshabur*, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgIPJBFvH.

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. Dynamic evaluation of neural sequence models. In *International Conference on Machine Learning*, pages 2766–2775, 2018.

[13] Ming Li and Paul M.B. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Publishing Company, Incorporated, 3 edition, 2008. ISBN 0387339981.

[14] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *International Conference on Learning Representations*, 2018.

[15] Gary Marcus. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*, 2018.

[16] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, 2017.

[17] Svetlin Penkov and Subramanian Ramamoorthy. Learning programmatically structured representations with perceptor gradients. In *International Conference on Learning Representations*, 2018.

[18] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850, 2016.

[19] Amartya Sanyal, Philip H. Torr, and Puneet K. Dokania. Stable rank normalization for improved generalization in neural networks and gans. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1enKkrFDB.

[20] Jürgen Schmidhuber. Discovering neural nets with low kolmogorov complexity and high generalization capability. *Neural Netw.*, 10(5):857–873, July 1997. ISSN 0893-6080. doi: 10.1016/S0893-6080(96)00127-X. URL https://doi.org/10.1016/S0893-6080(96)00127-X.

[21] Taiji Suzuki, Hiroshi Abe, and Tomoaki Nishimura. Compression based bound for non-compressed network: unified generalization error analysis of large compressible deep neural network. *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ByeGzlrKwH.

[22] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[23] Lloyd N. Trefethen. *Approximation Theory and Approximation Practice*. SIAM, 2013.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[25] Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning*, pages 5045–5054, 2018.

[26] Colin Wei and Tengyu Ma. Data-dependent sample complexity of deep neural networks via lipschitz augmentation. In *Advances in Neural Information Processing Systems 32*, pages 9725–9736. Curran Associates, Inc., 2019.

[27] Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

Supplementary Material to Neural Abstract Reasoner

Victor Kolev[*] victor.kolev@yahoo.com
Bogdan Georgiev[†] bogdan.m.georgiev@gmail.com
Svetlin Penkov[‡] svet@sciro.ai

Appendix A Abstraction and Reasoning Corpus

The Abstraction and Reasoning Corpus (ARC) [2] is a dataset of grid-based pattern recognition and pattern manipulation tasks. A decision-making agent sees a small number of examples of input and output grids that illustrate the underlying logical relationship between them. It then has to infer this logical rule and apply it correctly on a test query.

In many ways, the benchmark is similar to the Bongard problems (view [3]) – relations are highly abstract and geometric. Moreover, only 3-5 examples are presented for each task, therefore, the benchmark tests the ability of an decision-making agent to (i) grasp abstract logic and (ii) adapt quickly to new tasks.

There are 400 training and 400 evaluation task examples, structured as follows:

- each task consists of a train and a test set;
- the train set includes 3-5 input/output pairs;
- the test set includes 1 input/output pair;
- an input/output pair is comprised of an input grid and an output grid, the relation between which follows a consistent logic throughout the task;
- grids are rectangular and are divided into 1×1 squares;
- grid patterns are drawn with 10 colors;
- grid sizes vary between 1 and 30 in length and width; input and output grid sizes are not necessarily equal.

No set of rules exists that can solve all tasks, and while some skills are useful for multiple of them, each task has its own unique principle. This makes trivial approaches like brute-force computation impractical.

If a human was approaching those tasks, they would easily be able to spot logical relations – we have developed the necessary priors to find similarities and infer logic. Therefore, ideally the neural network would derive this prior during training, and that would allow it to generalize well to the evaluation dataset.

For the current scope of our research we use all tasks with grids of size not larger than 10×10. For the train set, we augment the tasks to 15000 by permuting colors and by exploiting that the tasks are invariant to rotation and symmetry.

[*]Sofia High School of Mathematics, Bulgaria
[†]Fraunhofer IAIS, Research Center for ML (FZML) and Competence Center ML2R, Germany
[‡]Sciro Research, Bulgaria

4th Knowledge Representation and Reasoning Meets Machine Learning Workshop (KR2ML 2020), at NeurIPS.

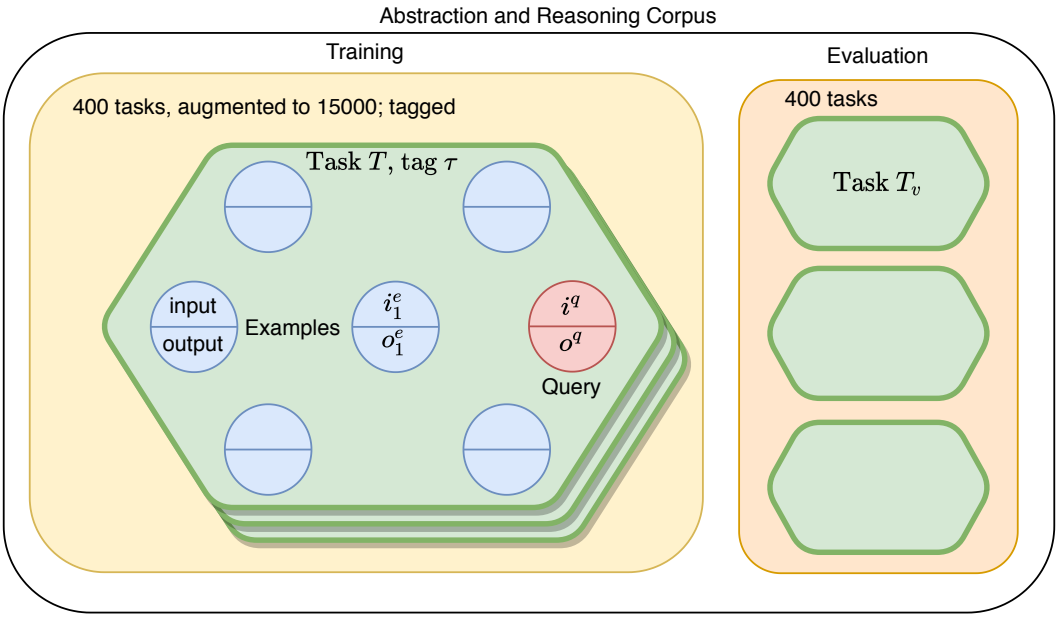

Figure 1: Structure of the Abstraction and Reasoning Corpus dataset.

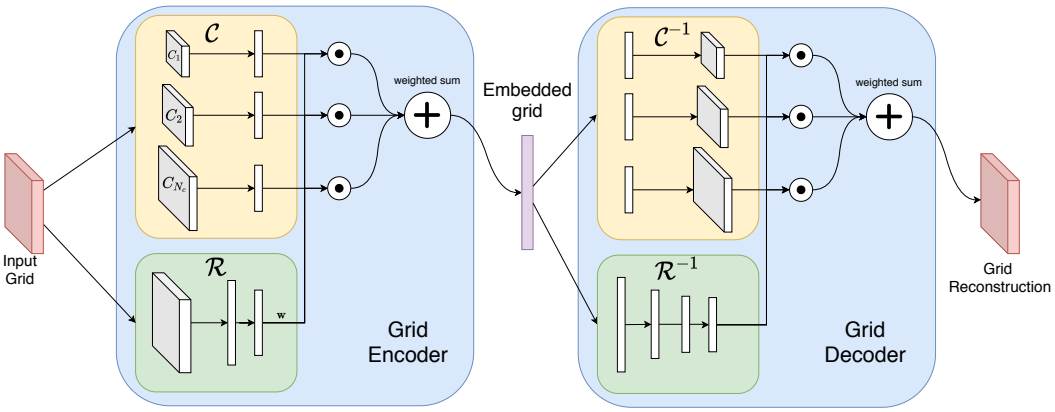

Figure 2: Structure of the Embedding network.

Appendix B Grid Embedding

Prior to embedding, we zero-pad all grids to be 10×10, with the original grid in the center of the image. Additionally, colors in the grids are represented as one-hot vectors, making the final dimensionality of the grids $10 \times 10 \times 10$ (10 colors).

The embedding is done with a convolutional neural network, comprised of an encoder and a decoder. The encoder consists of a basket of 10 convolutions of filter sizes equal to $1, \ldots 10$ (the $\mathcal{C}$ module, Fig. 2). Different filter sizes enable the network to capture both local and global patterns. The convolution outputs are flattened and passed to linear layers with hyperbolic tangent activation functions, which transform them into the desired dimensionality ($n = 256$). A second neural network $\mathcal{R}$ computes weights for summing the 10 resulting vectors. The decoder shares the same architecture with the encoder, but in reverse order – first linear layers, after that convolutions and then a weighted sum; finally a softmax over the color dimension.

Summing the convolutional outputs enables the embedding network to be agnostic to the order in which it receives them (as would be with an RNN for instance). The weighs reflect the fact that grid sizes vary and therefore not all filter sizes would be equally applicable or useful.

While training the Embedding network, we found that spectral regularization again proved to be instrumental for achieving 95% perfect reconstruction accuracy. In contrast, networks regularized by weight decay failed to climb above 6%.

What is more, tanh functions yielded a network that is 3 orders of magnitude more stable to Gaussian input noise than the same network, trained under the same conditions, with ReLU activations. We postulate that this is due to the fact that ReLU is unbounded for $x > 0$, therefore random perturbations would have a greater impact.

Appendix C Architecture Details

Figure 3 and 4 give greater detail about the architecture we devise, as well as the procedure for training it end-to-end.

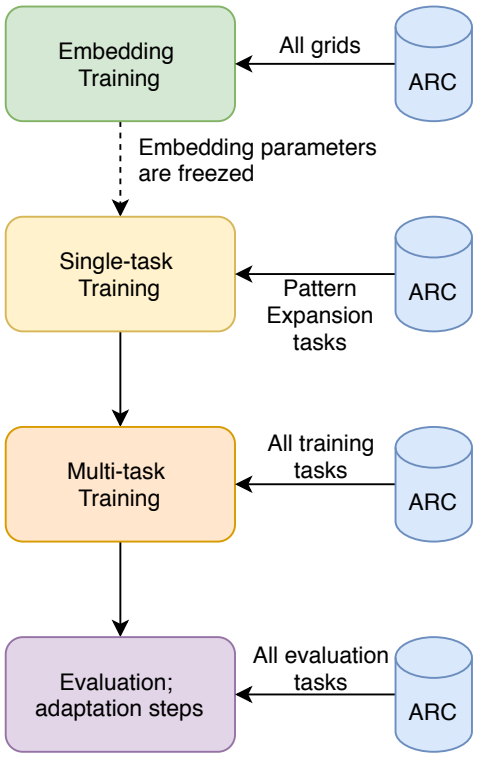

Figure 3: Order of procedures in training the model. Total runtime is ~ 50 hours on a single K80 GPU.

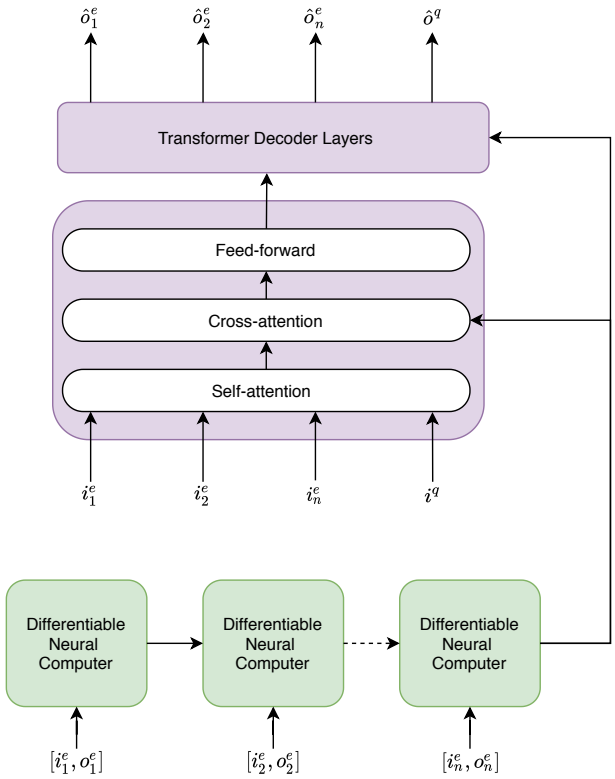

Figure 4: We use a memory-augmented neural network (DNC) to extract the context of the current task from example input-output pairs $[i_1^e, o_1^e], \ldots, [i_n^e, o_n^e]$. The final DNC output is passed to the cross-attention layers of a Transformer Decoder, which processes all inputs (examples and query) and produces output predictions. Loss is calculated as by-pixel cross-entropy of the output predictions and the targets, and the model is trained end-to-end.

Appendix D Additional Results

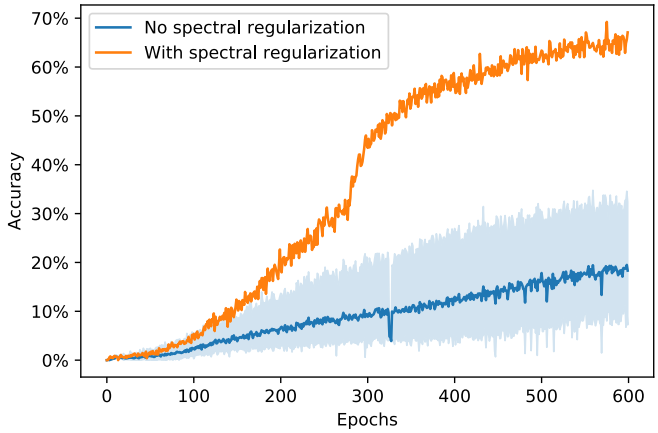

Figure 5: When we observed that stable ranks, a proxy of the number of parameters, were negatively correlated with generalization performance, we analyzed if an explicitly smaller model would improve performance without spectral regularization. While better performance and greater variance are observed, the model with spectral regularization remains unmatched.

Appendix E Approximation via *polynomial* programs

In this Section we briefly elaborate on the polynomial approximation mentioned in the main text. In general, classical results in this direction are based, e.g. on Chebyshev, Legendre and Bersntein polynomial approximations. Here we discuss a 1-dimensional approximation via Bernstein polynomials, but similar estimates hold in higher dimensions as well as other polynomial schemes (e.g. Chebyshev). We refer to [1, 5] for a thorough collection of results as well as further references.

A well-known method of approximating one-dimensional continuous functions is by means of Bernstein polynomials. Let $f : [0, 1] \to \mathbf{R}$ be continuous. The Bernstein polynomial $B_n(f; t)$ of order n corresponding to f is defined as:

$$B_n(f;t) := \sum_{k=0}^{n} f\left(\frac{k}{n}\right)\binom{n}{k} t^k (1-t)^{n-k}. \tag{1}$$

Theorem 1. *The following estimate holds:*

$$\|f - B_n(f;t)\|_{L^\infty} \leq \frac{3}{2}\omega\left(f; \frac{1}{\sqrt{n}}\right), \tag{2}$$

where ω denotes the modulus of continuity defined as

$$\omega(f;\delta) := \sup\left\{|f(x) - f(y)| : x, y \in [a, b], |x - y| \leq \delta\right\}. \tag{3}$$

In other words, ω measures the maximal jump $|f(x) - f(y)|$ over points x, y which are no more than a distance δ apart.

Note that whenever f has a Lipschitz constant L, the modulus of continuity $\omega(f, \delta)$ is controlled above via $L\delta$. Applying this bound in the estimate of Theorem 1 yields the result mentioned in the text. As already mentioned, more technical high dimensional analogues of the estimates are also available [5, 1] - e.g. an analogues result for Chebyshev polynomials holds where the modulus of continuity ω is appropriately replaced by extrema over complex ellipsoids [4].

Appendix F Hyperparamters

Hyperparameter	Value	Description
learning rate	0.001	Learning rate of the models
regularization λ	6e-4	Coefficient in front of the spectral regularization penalty in the loss
annealing factor	10	Factor by which λ is divided to weaken regularization
		Embedding network hyperparameters
latent dimension	256	The dimension of the embedded grids
kernel sizes	$1, \ldots, 10$	Sizes of the used kernels in the convolutional modules
# of kernels	128	The number of convolutional kernels
		DNC hyperparameters (temporal linkage is disabled)
# of read heads	6	Number of attention heads in the reading mechanism
# of LSTM layers	3	Layers of the LSTM controller
LSTM hidden size	512	Hidden size of the LSTM
memory dimesions	32×64	Word length of 64, 32 memory locations
		Transformer hyperparameters
# of decoders	4	Number of networks in the decoder stack
# of attention heads	16	Number of heads in the multi-head attention mechanism
Transformer hidden size	256	Internal hidden size of the linear layers in the Transformer

References

[1] *Polynomial Approximation Theory*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-30726-6. doi: 10.1007/978-3-540-30726-6_5. URL https://doi.org/10.1007/978-3-540-30726-6_5.

[2] Francois Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

[3] Alexandre Linhares. A glimpse at the metaphysics of bongard problems. *Artificial Intelligence*, 121(1-2): 251–270, 2000.

[4] L Trefethen. Multivariate polynomial approximation in the hypercube. *Proceedings of the American Mathematical Society*, 145:4837–4844, 2017.

[5] Lloyd N. Trefethen. *Approximation Theory and Approximation Practice*. SIAM, 2013.