Sudoku Assistant - An AI-powered app to help solve pen-and-paper Sudokus

Tias Guns[1,2], Milan Pesa, Maxime Mulamba[1,2], Ignace Bleukx[1], Emilio Gamba[1,2], and Senne Berden[1]

[1] KU Leuven, Belgium, firstname.lastname@kuleuven.be
[2] Vrije Universiteit Brussel, Belgium, firstname.lastname@vub.be

The Sudoku Assistant app is an AI assistant that uses a combination of machine learning and constraint programming techniques, to interpret and explain a pen-and-paper Sudoku scanned with a smartphone. Although the demo is about Sudoku, the underlying techniques are equally applicable to other constraint solving problems like timetabling, scheduling, and vehicle routing.

System overview

Digit classification - prediction After a picture of a Sudoku puzzle is taken with the app, the picture is segmented into 81 cells. A pre-trained convolutional neural network (CNN) is then applied to each cell in order to make a digit prediction. For each cell, the output of the CNN is a probability distribution over the 10 possible values: digits 1-9 and the empty cell value.

Classification and Solving After the probability distributions have been predicted, a naive approach would simply assign to each cell the value with which the largest probability has been associated (i.e., the argmax). This partial Sudoku can then be given to a constrained satisfaction (CSP) solver, together with the puzzle constraints, to obtain the full solution. However, consider that with this approach, even a CNN with an accuracy of 99% would lead to a correct solution only approximately $0.99^{81} = 44\%$ of the time. This is the case because even a single misclassification almost always leads to a wrong solution or no solution at all.

Error correction - hybrid of prediction and reasoning Therefore, we proposed an alternative approach in [4], which features a deeper integration of the digit classification and the reasoning required to solve the Sudoku puzzle. This approach is schematically shown in Fig.1. The main idea is to not provide the constraint solver with merely the argmax prediction for each of the Sudoku's cells, but rather with the full distributions outputted by the CNN. It is then tasked to find the maximum likelihood solution that satisfies the Sudoku constraints, given the class probabilities. Thus, the solver now solves a constrained optimization problem (COP), rather than a CSP. Effectively, our hybrid approach allows the reasoning to correct some of the mistakes made by the CNN classifier. This allows it to solve significantly more Sudoku problems correctly compared to the naive approach.

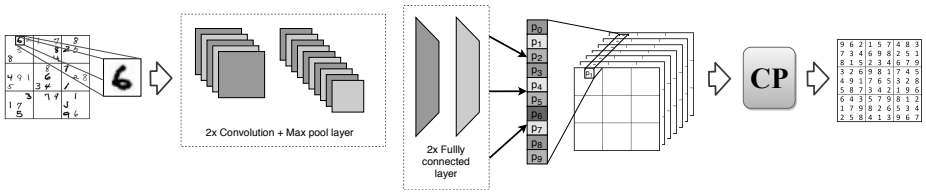
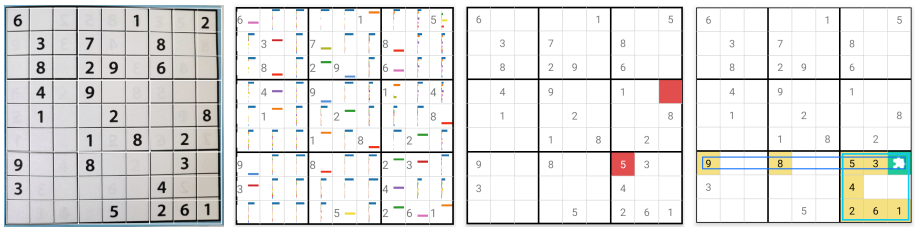


Fig. 1: The hybrid prediction-and-reasoning approach.

Providing hints - reasoning A user might be stuck during the solving when faced with a challenging Sudoku. The Sudoku Assistant showcases a hint mechanism that provides the easiest next move among all empty cells. The provided hint highlights which existing digits and constraints can be used to derive that cells value. To make sure the provided hints are easy to understand, we rely on a cost function that should approximate human understandability, e.g., taking the number of constraints and digits into account, as well as an estimate of their cognitive complexity. The underlying technology for computing the hints uses a constraint solver to find an Optimal Unsatisfiable Subset (OUS) of a derived unsatisfiable formula for (the negation of) each of the empty cells [1, 2].



(a) (b) (c) (d)

Fig. 2: Camera picture of a sudoku (a) with predicted probaiblities for the 81 cells (b). Error correction (in red) by the hybrid of prediction and reasoning (c). Visualisation of the hint (d) where the highlighted constraints (blue boxes) and given digits (yellow cells) derive a new value at the puzzle piece.

Implementation The app is implemented in React-Native. All neural network and constraint solving computations happen on a remote server. The CNN is composed of a VGG-inspired network [7], pre-trained on the Street View House Numbers dataset [5]. The last layer is replaced by a two-layers fully-connected neural network, tuned for our classification task using labeled data acquired from the app. All neural networks are implemented with PyTorch 1.10 [6]. All reasoning (the hybrid reasoning and explanation generation) is implemented in the CPMpy 0.9.9 constraint solving environment [3].

Acknowledgements

This research received partial funding from the Flemish Government (AI Research Program); and funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant No. 101002802, CHAT-Opt).

References

1. Bogaerts, B., Gamba, E., Guns, T.: A framework for step-wise explaining how to solve constraint satisfaction problems. Artificial Intelligence **300**, 103550 (2021)
2. Gamba, E., Bogaerts, B., Guns, T.: Efficiently explaining CSPs with unsatisfiable subset optimization. In: Zhou, Z.H. (ed.) Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21 (2021). https://doi.org/10.24963/ijcai.2021/191
3. Guns, T.: Increasing modeling language convenience with a universal n-dimensional array,cppy as python-embedded example. In: The 18th workshop on Constraint Modelling and Reformulation (ModRef 2019) (2019)
4. Mulamba, M., Mandi, J., Canoy, R., Guns, T.: Hybrid classification and reasoning for image-based constraint solving. In: International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research. pp. 364–380. Springer (2020)
5. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. NIPS (2011), http://ufldl.stanford.edu/housenumbers
6. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. In: NIPS Autodiff Workshop (2017)
7. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1409.1556