Sudoku Assistant – an AI-Powered App to Help Solve Pen-And-Paper Sudokus

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

[1]Vrije Universiteit Brussel,
[2]KU Leuven,
{emilio.gamba, maxime.mulamba}@vub.be,
{tias.guns, ignace.bleukx, senne.berden, milan.pesa}@kuleuven.be

Abstract

The Sudoku Assistant app is an AI assistant that combines 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

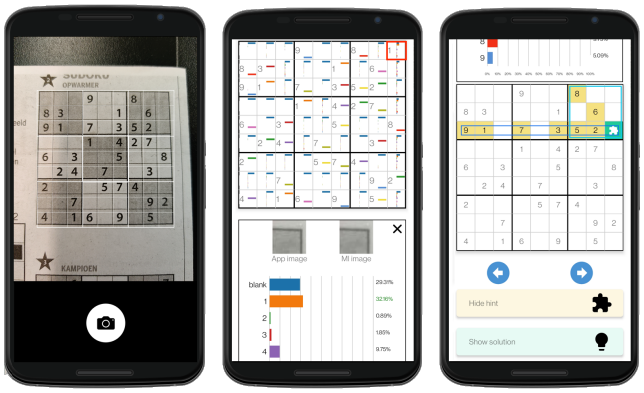
The Sudoku Assistant is a smartphone app that features two components: (1) a *hybrid predict-and-reasoning system* to recognize the digits of a Sudoku grid from a picture taken by a smartphone camera; and (2) a *hinting mechanism* to guide the user into finding the solution.

Phase 1: Hybrid of Prediction and Reasoning

The predict-and-reasoning component uses a classifier to recognize the digits in the Sudoku grid as well as a reasoning engine in order to solve the Sudoku puzzle.

Recognizing the Sudoku Digits First, the Sudoku Assistant app invites the user to align the overlay with the Sudoku grid (see Fig. 1a). After a picture of a Sudoku puzzle is taken with the app, the picture is segmented into 81 cells. A pretrained 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.



(a) (b) (c)

Figure 1: Camera picture of a sudoku (a) with predicted probabilities for the 81 cells (b).Visualisation of the hint (c) where the highlighted constraints (blue boxes) and given digits (yellow cells) derive a new value at the puzzle piece.

Better predictions with a hybrid of prediction and reasoning Therefore, we proposed an alternative approach in (Mulamba et al. 2020), 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.2. 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.

In other words, our approach performs a form of joint inference, taking into account the constraints of the Sudoku problem. 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.

Phase 2: Human-Understandable Explanations

We want the AI assistant to do more than just show the solution and ruin all the fun for the user. Instead, we want the AI to guide the user into finding the solution, to make them

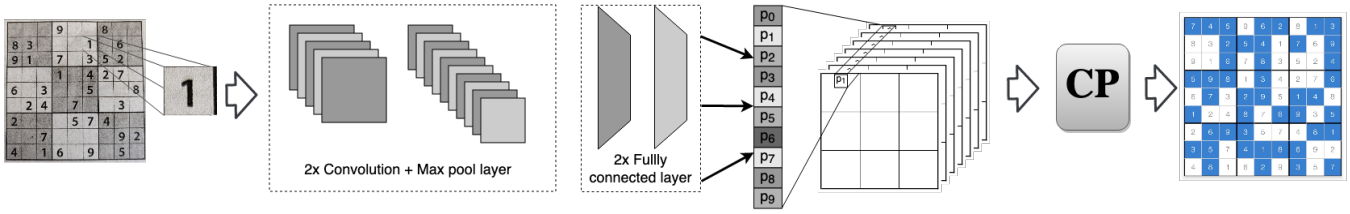


Figure 2: The hybrid prediction-and-reasoning approach.

learn some solving strategies along the way. For this, we integrated explanations from (Bogaerts, Gamba, and Guns 2021; Gamba, Bogaerts, and Guns 2021).

Providing hints The Sudoku Assistant showcases a hinting mechanism that provides the *easiest* next move among all empty cells. The provided hint, visualised in Fig. 1c, highlights which existing digits and constraints can be used to derive that cell's value. The underlying technology relies on a constraint solver to find an Optimal Constrained Unsatisfiable Subset (OCUS) of a derived unsatisfiable formula for (the negation of) each of the empty cells (Bogaerts, Gamba, and Guns 2021; Gamba, Bogaerts, and Guns 2021).

Explanation difficulty 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.

Scalability Preliminary experiments have shown that generating hints with the out-of-the-box OCUS-algorithm of Gamba, Bogaerts, and Guns (2021) is too slow for an interactive setting. Therefore, we adopt a two-stage approach which works as follows. The first stage computes an optimal subset of the problem constraints that can be used to derive a new cell value. The second stage first filters the digits of the Sudoku grid that are present in the constraints of the first stage. Next, the system computes the optimal subset of digits required to infer the new cell's value.

Implementation

The app is implemented in React-Native[1][2]. All neural network and constraint solving computations are done using API calls to a remote server with FastApi. The CNN is composed of a VGG-inspired network (Simonyan and Zisserman 2015), pre-trained on the Street View House Numbers dataset (Netzer et al. 2011). 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 (Paszke et al. 2017). All reasoning (the hybrid reasoning and explanation generation) is implemented in the CPMpy 0.9.9 constraint solving environment (Guns 2019).

[1]Picture of the smartphone is taken from https://toppng.com/android-phone-frame-hd-PNG-free-PNG-Images_78481

[2]Companion website https://visualsudoku.cs.kuleuven.be

Related Works

In recent years, the combination of deep learning for image perception and symbolic reasoning to visual sudoku has become increasingly popular. Some systems jointly learn the constraints to classify cell values (Brouard, de Givry, and Schiex 2020; Wang et al. 2019; Yang, Ishay, and Lee 2020). However this requires solver-specific mechanisms whereas our approach works with any constraint solver. Another limitation, also present in (Bai, Chen, and Gomes 2021; Mulamba et al. 2020) is that instances are built by sampling images from MNIST (Deng 2012), on which recent machine learning approaches achieve near-perfect accuracy (An et al. 2020). Images from a phone camera are in RGB space, can contain artifacts such as grid borders and are overall noisier depending on the angle of the camera or lighting conditions. They are a bigger challenge.

A non-exhaustive list of works (Reeson et al. 2007; Caine and Cohen 2007), showcase (interactive) tutoring systems using the Sudoku puzzle, for instance, to teach consistency propagation algorithms in Constraint Programming (Howell et al. 2018). Other works rely on Minimal Unsatisfiable Subsets (MUS) to compute a minimal subset of constraints and facts that can be used to derive new information (Bogaerts, Gamba, and Guns 2021; Espasa et al. 2021). Few works have considered optimizing MUSs with respect to a given cost-function. The only criterion considered is cardinality-minimality (Lynce and Silva 2004; Ignatiev et al. 2015).

Conclusion and Future Work

The Sudoku Assistant App combines machine learning and constraint reasoning to interpret, solve and explain pen-and-paper sudokus. The system has room for further improvements. As illustrated in Fig. 1a, the user must align the grid overlay with the target sudoku. A potential improvement in the image acquisition is to train a CNN to extract a sudoku from an image, as done in Object Detection (Paliwal et al. 2020).

The hint mechanism of the Sudoku Assistant relies on a simplistic cost-function to approximate the cognitive complexity of an explanation. An open question there is how to design a cost function that better reflects the human-interpretability of explanations.

Finally, we can assume that printed numbers usually are initial clues of the puzzle, and that hand-written numbers usually are from human input. The constraint solver could focus more on printed numbers while searching the feasible space.

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

An, S.; Lee, M. J.; Park, S.; Yang, H. S.; and So, J. 2020. An Ensemble of Simple Convolutional Neural Network Models for MNIST Digit Recognition. *ArXiv*, abs/2008.10400.

Bai, Y.; Chen, D.; and Gomes, C. P. 2021. CLR-DRNets: Curriculum Learning with Restarts to Solve Visual Combinatorial Games. In Michel, L. D., ed., *27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25-29, 2021*, volume 210 of *LIPIcs*, 17:1–17:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Bogaerts, B.; Gamba, E.; and Guns, T. 2021. A framework for step-wise explaining how to solve constraint satisfaction problems. *Artificial Intelligence*, 300: 103550.

Brouard, C.; de Givry, S.; and Schiex, T. 2020. Pushing Data into CP Models Using Graphical Model Learning and Solving. In Simonis, H., ed., *Principles and Practice of Constraint Programming - 26th International Conference, CP 2020, Louvain-la-Neuve, Belgium, September 7-11, 2020, Proceedings*, volume 12333 of *Lecture Notes in Computer Science*, 811–827. Springer.

Caine, A.; and Cohen, R. 2007. Tutoring an Entire Game with Dynamic Strategy Graphs: The Mixed-Initiative Sudoku Tutor. *Journal of Computers*, 2(1): 20–32.

Deng, L. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6): 141–142.

Espasa, J.; Gent, I. P.; Hoffmann, R.; Jefferson, C.; and Lynch, A. M. 2021. Using small muses to explain how to solve pen and paper puzzles. *arXiv preprint arXiv:2104.15040*.

Gamba, E.; Bogaerts, B.; and Guns, T. 2021. Efficiently Explaining CSPs with Unsatisfiable Subset Optimization. In Zhou, Z.-H., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*.

Guns, T. 2019. 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)*.

Howell, I.; Woodward, R. J.; Choueiry, B. Y.; and Bessiere, C. 2018. Solving Sudoku with Consistency: A Visual and Interactive Approach. In *IJCAI*, volume 2018, 5829–5831.

Ignatiev, A.; Previti, A.; Liffiton, M.; and Marques-Silva, J. 2015. Smallest MUS extraction with minimal hitting set dualization. In *Proceedings of CP*.

Lynce, I.; and Silva, J. P. M. 2004. On Computing Minimum Unsatisfiable Cores. In *Proceedings of SAT*.

Mulamba, M.; Mandi, J.; Canoy, R.; and Guns, T. 2020. Hybrid classification and reasoning for image-based constraint solving. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, 364–380. Springer.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning. *NIPS*.

Paliwal, S.; D, V.; Rahul, R.; Sharma, M.; and Vig, L. 2020. TableNet: Deep Learning model for end-to-end Table detection and Tabular data extraction from Scanned Document Images. *CoRR*, abs/2001.01469.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Reeson, C. G.; Huang, K.-C.; Bayer, K. M.; and Choueiry, B. Y. 2007. An interactive constraint-based approach to Sudoku. In *AAAI*, 1976–1977.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Wang, P.; Donti, P. L.; Wilder, B.; and Kolter, J. Z. 2019. SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, 6545–6554. PMLR.

Yang, Z.; Ishay, A.; and Lee, J. 2020. NeurASP: Embracing Neural Networks into Answer Set Programming. In Bessiere, C., ed., *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, 1755–1762. ijcai.org.