ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

Enhancing Computer Vision with Knowledge: a Rummikub Case Study

Simon Vandevelde[1,2,3], Laurent Mertens[1,2], Sverre Lauwers[1] and Joost Vennekens[1,2,3] *

1- KU Leuven - Dept. Of Computer Science
J.-P. De Nayerlaan 5, 2860 Sint-Katelijne-Waver - Belgium

2- Leuven.AI - KU Leuven institute for AI
B-3000, Leuven - Belgium

3- Flanders Make - DTAI-FET

Abstract. Artificial Neural Networks excel at identifying individual components in an image. However, out-of-the-box, they do not manage to correctly integrate and interpret these components as a whole. One way to alleviate this weakness is to expand the network with explicit knowledge and a separate reasoning component. In this paper, we evaluate an approach to this end, applied to the solving of the popular board game Rummikub. We demonstrate that, for this particular example, the added background knowledge is equally valuable as two-thirds of the data set, and allows to bring down the training time to half the original time.

1 Introduction

Artificial Neural Networks (ANNs) are considered a tried and tested method to identify objects in an image. However, correctly interpreting these objects in relation to each other to form a complete picture remains difficult, as demonstrated in the literature [1, 2]. One way to overcome this issue is by adding explicit *background knowledge* about the identified items and their relations. In this work, we apply this approach to the popular boardgame Rummikub. Concretely, we consider the task of correctly detecting all the tiles in a photo of a Rummikub game state. We compare the performance of a "vanilla" ANN setup to one that extends this setup with explicit knowledge and reasoning by means of the IDP-Z3 [3] system. In the following paragraphs, we introduce Rummikub and IDP-Z3.

Rummikub[1] is a popular board game in which players are given tiles defined by a number $n \in [1, 2, \ldots, 13]$ and a color $c \in [\text{red}, \text{blue}, \text{black}, \text{yellow}]$. To win the game, players need to be the first to place all their tiles in the center of the game field. Tiles may only be played when they correctly form a *set*. Two types of sets exist: a *group*, in which 3 or 4 tiles share the same n but have different c, and a *run*, which is a series of 3 to 13 tiles of same c with subsequent n. The game also contains two joker tiles which may be used as "wildcards" to form sets, as indicated by a smiling face. All these concepts are illustrated in Fig. 1.

*This research received funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme

[1] `https://rummikub.com`

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

(a) Example of a group. (b) Example of a run, containing a joker.

Fig. 1: Rummikub set examples

IDP-Z3 is a logical reasoning engine for first order logic. It adheres to the Knowledge Base Paradigm [4], which states that knowledge should be modeled *declaratively* in a Knowledge Base (KB) regardless of its use. As such, the KB is merely a "bag of knowledge".To put this knowledge to use, it can be given to a reasoning engine such as IDP-Z3, which supports several kinds of inference tasks. This approach supports reusability: once formalized, the KB can be re-used to solve different types of problems in the same problem domain without modifications. I.e., the same KB could be used to check satisfiability, find (optimal) solutions, derive consequences, explain incorrectness, etc.

The specific modeling language used for IDP-Z3 is FO($\cdot$), which extends classical first order logic by adding concepts (e.g., types and aggregates) to make modeling more user-friendly. As an example, consider the following logical formula which formalizes the rule of a correct Rummikub group.

$$\forall t_1, t_2 \in \mathit{Tile} : t_1 \neq t_2 \wedge \mathit{set}(t_1) = \mathit{set}(t_2) \wedge \mathit{group}(\mathit{set}(t_1)) \Rightarrow \\ \mathit{number}(t_1) = \mathit{number}(t_2) \wedge \mathit{color}(t_1) \neq \mathit{color}(t_2).$$

This can be read as: "for all different tiles t_1, t_2 within a group must hold that their numbers are the same and their colors are different".[2]

2 Methodology

We use a custom image dataset consisting of 285 manually captured images of Rummikub playing fields, employing three different zoom levels, four different lighting levels and two different backgrounds. Special attention was paid to ensure the images are realistic: they all contain a varying number of only valid sets, at various positions and angles. Each set ranges from 3 to 13 tiles, and is diverse in terms of colors and numbers. The tiles are also often not perfectly aligned, much like they would be in a real game. Images were annotated for tile bounding boxes and tile number/color, with a total of 4336 tiles annotated. Our dataset is publicly available through Kaggle [3].

To correctly detect all tiles in a depicted Rummikub game state, we propose the pipeline shown in Fig. 2. It consists of four steps:

[2]For a more detailed explanation on FO($\cdot$), we refer to [3].
[3]`https://www.kaggle.com/datasets/sverrela/rummikub`

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

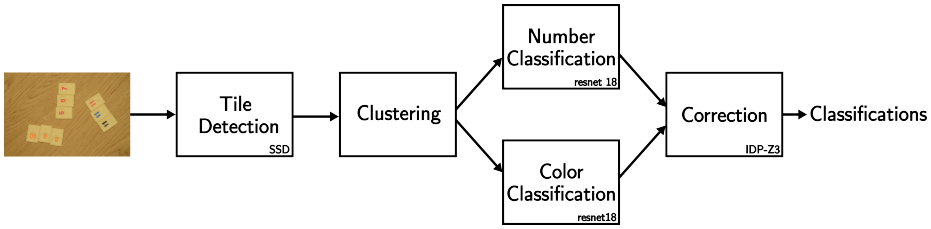


Fig. 2: Overview of the tile detection/classification pipeline

Step 1 - Tile detection: Bounding boxes are generated for each individual tile in the image by means of a standard single shot multi-box detector (SSD) [5] trained on our dataset.

Step 2 - Clustering: A hand-crafted algorithm clusters the individual bounding boxes into sets. This clustering algorithm is designed to be robust to size and orientation, so that it can handle the random nature of the sets.

Step 3 - Number/Color classification: Each detected tile is classified for n and c by means of two ResNet18 [6] networks trained on our dataset. The output of each network represents the confidence levels assigned to each possible n and c for each tile. A pure ANN setup would at this point assign the class with the highest confidence to each tile. Instead, we pass all confidence values on to the next step for further processing.

Step 4 - Correction: In this final step, the confidence levels obtained in Step 3 are used to model an optimization problem at the level of an entire set. Concretely, instead of assuming the most likely class for each tile individually, we try to find the most likely class for *all* tiles in a set, given that the set must be correct (i.e., we explicitely assume the shown game state to be valid). This way, we are able to enrich our detections with background knowledge.

The optimization problem in Step 4 is straightforward. We model the confidences in IDP-Z3 using binary functions of the form $attr\_conf : Tile \times Attr \to Number$, and add them to our Rummikub KB. To compute the overall confidence score of a possible solution, we use the following straightforward sum:

$$\begin{aligned} acc() = & \sum_{t \in Tile, c \in Color} color\_conf(t, c) | color(t) = c \\ + & \sum_{t \in Tile, n \in Number} number\_conf(t, n) | number(t) = n. \end{aligned}$$

In other words, we sum the confidences of the colors/numbers that have been assigned to the tiles via the *color* and *number* functions. We then let IDP-Z3 find the optimal assignments of these functions, giving us the classifications with the highest confidences that are feasible.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

As a concrete example, consider the following color confidences :

$$\begin{aligned}
&\text{Tile 1: } (1, red) \to 0.8, (1, blue) \to 0.1, (1, black) \to 0.05, (1, orange) \to 0.05,\\
&\text{Tile 2: } (2, red) \to 0.2, (2, blue) \to 0.7, (2, black) \to 0.09, (2, orange) \to 0.01,\\
&\text{Tile 3: } (3, red) \to 0.5, (2, blue) \to 0.15, (2, black) \to 0.05, (2, orange) \to 0.3.
\end{aligned}$$

Ignoring the numbers for the sake of the example, and assuming the set is a group, all tiles must therefore have a different color. As such, instead of the highest likely *individual* colors *(red, blue, red)*, IDP-Z3 will correct these classifications to *(red, blue, orange)* as the most confident *feasible* classification.

3 Results

All code required to reproduce the results discussed in this section is available from our dedicated GitLab repository[4]. All evaluations are performed using an Intel Xeon E5-2630 v3 with an NVIDIA Quadro P2000 and 32 GB of memory, using our full dataset as test data.[5] We ran each experiment 10 times, and report averages and standard deviations.

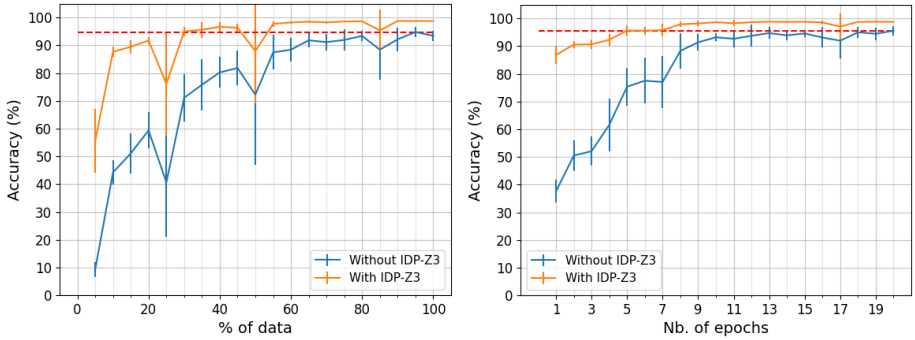
We evaluated the pipeline in terms of the size of the training data, by limiting the available data to a % subset, as shown in Fig 3a. For every run, ResNet18 color and number models were trained for 5 and 20 epochs respectively. A few observations are noteworthy. First, when using only 5% of the data, the context-based correction step greatly increases the total image accuracy from 9.31±2.63% to 55.56±11.50%. Second, the full pipeline outperforms the pure ANN approach, even at its highest reached accuracy (98.76±0.15% @ 90% vs. 94.79±1.85% @ 95% resp., or a 3.97% increase). Third, it seems that, in general, adding the reasoning step seems to lower the standard deviations, acting like a "stabiliser" of sorts. For example, between 30 and 45% of the data, the ANNs had a standard deviation between 5.57 and 8.5, while the full pipeline's standard deviation remained between 1.37 and 2.8. Fourth, and most interesting, when taking the detections into account, the highest accuracy reached by the classifiers is already reached when using only about 30% of the data (with IDP-Z3 @ 30%: 94.98±1.79%, without IDP-Z3 @ 95%: 94.79±1.85%), as indicated by the dashed red line.

To evaluate the effect of training time, Fig. 3b shows an analogous experiment conditioning on the number of epochs instead of training data. Here, both the ResNet18 color and number models are trained for x epochs. The same tendencies are apparent, though the initial jump in accuracy seems to be notably higher (37.61±4.14% to 86.83±3.32% after one epoch). Again, the highest

[4] `https://gitlab.com/EAVISE/sva/knowledge-enhanced-rummikub-detector`

[5] During training, the ResNet18 networks are being fed the annotated bounding boxes, while during evaluation, they receive their input from the SSD network, which is slightly different. Note that, while typically using the same data for train and testing purposes is considered bad practice, we argue that in this particular case it allows to evaluate the pure ANN approach in "ideal" circumstances, further highlighting the added benefit of adding a logical reasoning step.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.



(a) Accuracy when training on % of data (b) Accuracy when training for x epochs

Fig. 3: Ablation experiment results. The dashed red lines indicate the highest accuracy reached by the ANN-only approach.

accuracy reached by the classifiers is achieved much earlier when adding logical reasoning: 5 epochs are sufficient instead of 20 (with IDP-Z3 @ 5 epochs: 95.64±1.84%, without IDP-Z3 @ 20 epochs: 95.56±1.66%). Similarly, the full pipeline outperforms the pure ANN approach (98.84±0.00% @ 13 epochs vs. 95.56±1.66% @ 20 epochs resp.), and the standard deviations have decreased across the board.

It is, however, important to note that the correction step does add additional inference time, albeit the effect appears to be rather limited. In our experiments, tile detection, clustering, and classifying required about 0.04s per tile, while correcting an entire set requires another 0.095s. E.g., for a set consisting of 5 tiles the pipeline would require 0.2s to generate the classifications, and another 0.1s to correct them if necessary.

4 Related Work

In recent years, there has been a rise in approaches aimed at extending ANNs with explicit reasoning. The most notable approach, as shown in works such as DeepProbLog [7] and NeurASP [8], combine the two on a true neurosymbolic level, tightly coupling the neural networks with logic. In this approach, the logic program is evaluated throughout the training process itself, resulting in increasingly higher accuracies. However, in general, such systems have difficulties with scaling to complex domains, and are slow to train. For a more complete overview of neurosymbolic approaches, we refer to [9].

The work that comes closest to ours is that of Mulamba et al. [10], who present a similar pipeline approach, but for the detection of digits in a (partial) Sudoku. However, they did not evaluate the effect of data or training time, and instead only report on the final accuracy. Furthermore, the unknown tile positions in Rummikub, as opposed to fixed grid positions for Sudoku, make it a harder problem to solve.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

5 Conclusion

We have demonstrated an approach to improve the accuracy of a computer vision system for the detection and validation of Rummikub game states, by adding explicit reasoning on background knowledge. Through our evaluations, we have shown that, for this problem, background knowledge is worth as much as two-thirds of the data set, or slightly more than half of the training time. In this sense, our approach is most useful in situations where data is scarce or difficult to gather, or when the ANNs are constraint by hardware limitations, such as in edge devices. However, our approach can only succeed when there exists a clear relationship between all output classes, which is not always the case. Among others, examples of real-life applications satisfying this constraint include sensor fusion and input detection in forms (e.g., tax forms).

As part of future work, we intend to compare our approach to a more neurosymbolic approach by implementing the Rummikub example in, e.g., NeurASP. We also plan to extend the work to allow the pipeline to suggest corrections when errors (i.e., invalid sets) are present in the image. Finally, we expect to further evaluate our pipeline on some of the real-life problem domains mentioned above.

References

[1] Laurent Mertens, Elahe' Yargholi, Hans Op de Beeck, Jan Van den Stock, and Joost Vennekens. FindingEmo: An image dataset for emotion recognition in the wild (accepted at NeurIPS 2024). 2024.

[2] Oge Marques, Elan Barenholtz, and Vincent Charvillat. Context modeling in computer vision: Techniques, implications, and applications. *Multimedia Tools and Applications*, 51(1):303–339, January 2011.

[3] Pierre Carbonnelle, Simon Vandevelde, Joost Vennekens, and Marc Denecker. IDP-Z3: A reasoning engine for FO(.). *arXiv preprint arXiv:2202.00343*, 2022.

[4] Marc Denecker and Joost Vennekens. Building a Knowledge Base System for an Integration of Logic Programming and Classical Logic. In *Logic Programming*, volume 5366, pages 71–76. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[5] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[7] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. *Advances in neural information processing systems*, 31, 2018.

[8] Zhun Yang, Adam Ishay, and Joohyung Lee. Neurasp: Embracing neural networks into answer set programming. In *Proceedings of IJCAI-20*, pages 1755–1762. International Joint Conferences on Artificial Intelligence Organization, July 2020.

[9] Giuseppe Marra, Sebastijan Dumančić, Robin Manhaeve, and Luc De Raedt. From statistical relational to neurosymbolic artificial intelligence: A survey. *Artificial Intelligence*, 328:104062, March 2024.

[10] Maxime Mulamba, Jayanta Mandi, Rocsildes Canoy, and Tias Guns. Hybrid Classification and Reasoning for Image-Based Constraint Solving. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 364–380, 2020.