CLR-DRNets: Curriculum Learning with Restarts to Solve Visual Combinatorial Games

Yiwei Bai
Cornell University, Ithaca, NY, USA

Di Chen
Cornell University, Ithaca, NY, USA

Carla P. Gomes
Cornell University, Ithaca, NY, USA

Abstract

We introduce a curriculum learning framework for challenging tasks that require a combination of pattern recognition and combinatorial reasoning, such as single-player visual combinatorial games. Our work harnesses Deep Reasoning Nets (DRNets) [4], a framework that combines deep learning with constraint reasoning for unsupervised pattern demixing. We propose CLR-DRNets (pronounced Clear-DRNets), a curriculum-learning-with-restarts framework to boost the performance of DRNets. CLR-DRNets incrementally increase the difficulty of the training instances and use restarts, a new model selection method that selects multiple models from the same training trajectory to learn a set of diverse heuristics and apply them at inference time. An enhanced reasoning module is also proposed for CLR-DRNets to improve the ability of reasoning and generalize to unseen instances. We consider Visual Sudoku, i.e., Sudoku with hand-written digits or letters, and Visual Mixed Sudoku, a substantially more challenging task that requires the demixing and completion of two overlapping Visual Sudokus. We propose an enhanced reasoning module for the DRNets framework for encoding these visual games We show how CLR-DRNets considerably outperform DRNets and other approaches on these visual combinatorial games.

2012 ACM Subject Classification Computing methodologies → Machine learning

Keywords and phrases Unsupervised Learning, Combinatorial Optimization

Digital Object Identifier 10.4230/LIPIcs.CP.2021.17

Funding This research was supported by NSF awards CCF-1522054 (Expeditions in computing), CNS-1059284 (Infrastructure), and an ARO award (DURIP, W911NF-17-1-0187, AIDA compute cluster).

Acknowledgements We want to thank Wenting Zhao and anonymous reviewers for their valuable feedback.

1 Introduction

Deep learning has surpassed human-level performance on many perception tasks, ranging from object recognition to language translation. However, these successes heavily rely on the availability of large datasets and corresponding labels. In contrast, humans often only have access to a few examples, and therefore they resort to meticulous reasoning about prior knowledge to compensate for the lack of labeled data and fill in the data information gaps. Herein we consider unsupervised or weakly supervised single-player visual combinatorial games. Humans tackle such challenging tasks by combining pattern recognition with reasoning about prior knowledge (the games' rules). Visual combinatorial games capture various real-world applications, particularly scientific data interpretation tasks, which are in general unsupervised or weakly supervised but for which rich prior knowledge is often available [4]. Consider the case of Visual Sudoku [22], a variant of the standard Sudoku with hand-written

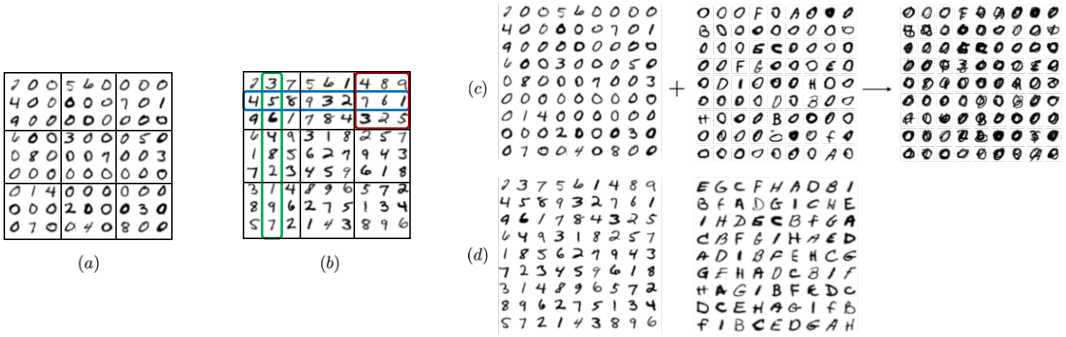

Figure 1 (a) Visual Sudoku: 0 denotes the empty cell, other digits are hints. Goal: replace the empty cells with digits from 1 to 9 to obtain a valid Sudoku, i.e., the cells in each row (blue rectangle), column (green rectangle), and any of the nine marked 3x3 boxes (red square) have to have all-different digits. (b) The solution to (a). (c) Visual Mixed Sudoku: 0 and O denote empty cells. The goal is to replace the empty cells (denoted by overlapping 0 and O characters) in the mixed Sudoku on the right with digits from 1 to 9 and letters from A to I, and obtain two valid demixed Sudokus. Note overlapping here is the max operator. (d) The solution to (c).

digits (see Fig. 1). We solve Visual Sudokus **without any Sudoku labels** by combining our perception skills, for digit recognition, with logical reasoning about Sudoku rules, to disambiguate noisy digits and fill in the missing digits. The missing digits simulate real-world settings in which there are missing data. Visual Mixed Sudoku (Fig. 1) is even more challenging than Visual Sudoku as it requires identifying the missing digits or letters of two partially filled overlapping Sudokus. Visual Mixed Sudoku involves demixing the partially filled Sudokus and inferring the missing digits. As we show in the experimental section, a straightforward approach for Visual Sudoku that first uses a well-trained state-of-the-art deep-learning digit classifier and passes the digit information as input to a powerful combinatorial solver, such as a SAT solver, does not lead to satisfactory results. Even though the classifier has high accuracy (99.4%), it still makes mistakes and therefore, when there are many hints in the Visual Sudoku, it is likely that the classifier will make a few mistakes, leading to noisy data that SAT solvers cannot handle. This approach performs even more poorly for the Visual Mixed Sudoku, given the higher probability of making digit/letter classification mistakes due to a combination of factors (additional challenging demixing task, lower accuracy of letter classifiers, and the fact that we double the number of hints corresponding to the two overlapping Sudokus). Therefore an approach integrating digit/letter recognition with reasoning about the Sudoku rules in an end-to-end fashion is required. Deep Reasoning Networks (DRNets) [4] is a framework proposed recently that seamlessly integrates deep learning with constraint reasoning via an interpretable latent space, to incorporate prior knowledge (such as Sudoku rules). DRNets were shown to be effective for unsupervised demixing tasks, such as the demixing of two solved (i.e., all the digits filled in) overlapping Visual Sudokus. Nevertheless, DRNets have limited reasoning generalization capabilities, in particular for completion tasks. As we will show, Visual Sudoku and Visual Mixed Sudoku are substantially challenging for DRNets since that in addition to the demixing task, they involve the demanding completion task that requires inferring missing digits. **Our contributions: (1)** We propose **CLR-DRNets** (pronounced clear DRNets), a **curriculum learning** framework to boost the performance of DRNets. Our approach is inspired by how humans learn to solve complex problems, starting with easy instances and gradually increasing the instances' difficulty. Another intuition behind our

approach is that the perception task is relatively easier than the reasoning task in many visual combinatorial games. Therefore, it is crucial to carefully select a sequence of training examples to balance the reasoning and perception tasks to prevent the model from focusing only on the perception task. **(2)** We propose an **enhanced reasoning module** to improve the power of reasoning and generalization to unseen data. **(3)** We propose **restarts,** a new model selection strategy for improving the performance at test time. We note that it is usually hard to solve a complex visual combinatorial problem in a single inference step of a neural network. Since CLR-DRNets are trained without label supervision, only with prior knowledge supervision (e.g., domain rules), we can still improve and customize the learned model with respect to the specific test data by further optimizing the loss function. We also find that the reasoning module of CLR-DRNets can learn different heuristics during training. Thus, saving several different models from the training phase and applying each of them during test time sequentially is really helpful. **(4)** We propose encodings for **(4.1) Visual Sudoku** and **(4.2) Visual Mixed Sudoku** using the CLR-DRNets framework. and **(5)** show how **CLR-DRNets substantially outperform DRNets and other approaches on these visual combinatorial games**.

2 Related Work

Combining perception and reasoning. CLR-DRNets leverage Deep Reasoning Networks (DRNets) [4]. DRNets were shown to be effective for unsupervised demixing tasks. **Here we use the DRNets framework for unsupervised visual combinatorial games, which are more challenging reasoning tasks than demixing,** and therefore we develop **a strengthened reasoning module for DRNets.** Furthermore, **CLR-DRNets further boost the power of DRNets with its curriculum-learning-with-restarts approach and generalize to unseen instances, in contrast to DRNets, designed to solve instances.** For Visual Sudoku, other approaches focus on learning the Sudoku rules from the labeled data [1, 24, 16, 9]. For example, SATNET [22] introduces a differentiable maximum satisfiability (MAXSAT) solver that can be incorporated with deep learning models to capture the reasoning rules efficiently. [3] learns problem related cost functions for Constraint Networks, which are solved with a specialized constraint solver. [18] integrates the perception and reasoning through exposing the predicted probability to the constraint solver. However, except for DRNets and [18], previous approaches require labeled training Sudoku data.

Curriculum learning (CL) for solving hard tasks. CL is widely used in deep reinforcement learning [19, 7, 21]. For example, Feng et al. [7] let the model firstly learn from the easy sub-task created from the hard original task. Then gradually increase the hardness of the sub-task until the agent can solve the original task. Our CL method shares a similar idea but we do not use it in a RL setting and we do not require that the easy instances are generated by the hard instances. Some works also employ CL for deep learning [23, 12]. For instance, Hacohen et al. [12] introduce a CL framework for image classification.

Restarts in deep learning. *Restarts* are widely used in the CP/SAT community (e.g., [10, 2]) and also in stochastic optimization to solve non-convex problems (e.g., [6, 8]). In the deep learning training phase, learning rate restart [11] is proven to increase the performance. DRNets use simple restarts, basically to randomly group different instances into one mini-batch to compute various gradients, CLR-DRNets in addition leverage different models acquired from one training trajectory and apply them sequentially to test (unseen) data.

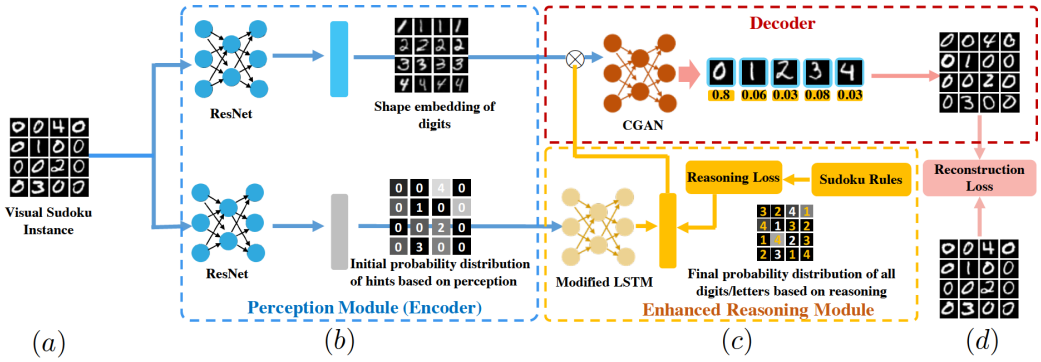

Figure 2 (a) The digit visual Sudoku instance. (b) The perception model (encoder) of CLR-DRNets. The top blue rectangle is the latent space capturing the shape information of all possible digits per cell. The bottom gray rectangle is the initial estimation of input digit visual Sudoku instance only based on perception. Here we employ the heat map to represent the predicted confidence of each cell. (c) The enhanced reasoning module of CLR-DRNets (not in the original DRNets) consists of a modified LSTM model. It takes the initial estimation as the input, constrained by the relaxed Sudoku rules' loss function, computing the completion results. (d) The decoder part leverages a pre-trained cGAN to generate all the possible digit images w.r.t the shape embeddings of (b) per cell. We can reconstruct each cell guided by the completion results of (c). (The figure uses 4×4 Sudokus for easier visualization. All our experiments are with 9×9 Sudokus.)

3 CLR-DRNets

We start by providing a high-level description of DRNets.[4].

DRNets. DRNets perform end-to-end unsupervised deep reasoning using a perception module (*encoder*) to produce the *initial estimation of the visual inputs*, which are constrained to adhere to prior knowledge via a *reasoning module*. The reasoning module encodes the constraint loss function using the *initial estimation*. A *generative decoder* uses the initial estimation of the visual inputs to generate the reconstruction of the input. DRNets solve the problem by jointly optimizing the reconstruction loss, encouraging the reconstruction to be similar to the input, and the constraint loss function, to enforce the domain rules.

CLR-DRNets borrow the general framework from DRNets and further strengthen it with *an enhanced reasoning module*. We propose *a curriculum learning framework* to tackle the difficult visual combinatorial completion tasks, which is beyond the capability of the original DRNets. Moreover, *restarts*, a new model selection strategy is proposed to boost the performance at test time. The enhanced reasoning module in CLR-DRNets is much more powerful and allows generalization to unseen data, in contrast to DRNets. Adapting from the DRNets [4] framework, CLR-DRNets formulate the entire process as a **data-driven optimization problem**:

$$\min_{\theta_p,\theta_r} \frac{1}{N}\sum_{i=1}^{N} \underbrace{\lambda^p \psi^p(f_{\theta_p}(x_i))}_{\substack{\text{regularize the}\\ \text{initial estimation}}} + \underbrace{\psi^r(f_{\theta_r}(f_{\theta_p}(x_i)))}_{\substack{\text{regularize the}\\ \text{reasoning output}\\ \text{new in CLR-DRNets}}} + \underbrace{\lambda^l \mathcal{L}(cGAN(f_{\theta_r}(f_{\theta_p}(x_i))), x_i)}_{\substack{\text{regularize both the}\\ \text{initial estimation and the reasoning output}\\ \text{modified from DRNets}}} \quad (1)$$

where f_{θ_r} and f_{θ_p} are the reasoning module and the perception module respectively, $cGAN$ is a pre-trained conditional generative adversarial network (cGAN)[17], x_i is the i-th data point of the input, ψ^p, ψ^r are the penalty functions of continuously relaxed constraints

related to the perception module and the reasoning module, λ^p, λ^l are weight scalars and $\mathcal{L}$ is any distance metric. In equation 1, the term labeled by "*regularize both the initial estimation and the reasoning output*" is **a modified version of DRNets' original loss function** and the term labeled by "*regularize the reasoning output*" is **completely new in CLR-DRNets.** These modified terms are associated with CLR-DRNets' enhanced reasoning module. Below we describe CLR-DRNets highlighting the main differences with respect to the general DRNets framework.

CLR-DRNets. We illustrate the CLR-DRNets framework with our proposed models (see Fig. 2) for the visual Sudoku games (Fig. 1 (a)). Similarly to DRNets, CLR-DRNets employ two ResNets for the perception module to predict the distribution of all digits in each cell along with the shape embedding (*initial estimation*). The constraints for the initial estimation (ψ^p) are that the predicted digits probability distribution of each cell should converge to only one digit.

CLR-DRNets' enhanced reasoning module is a modified long short term memory (LSTM) model [14] that can compute all the possible digits' distribution for each cell, capturing the Sudoku structure, and with sufficient power to reason about missing digits (more details below). The constraints for the reasoning module (ψ^r) are the relaxed Sudoku constraint losses, which can regularize the reasoning outputs to satisfy the Sudoku rule (more details below).

The decoder is similar to DRNets' and consists of a conditional Generative Adversarial Networks (cGAN) [17] pre-trained on the prototypes, which are images of all possible single digits. The decoder generates all possible digit images w.r.t the shape embeddings of the initial estimation. Then per each cell, all possible digit images are remixed based on the distribution of digits from the reasoning and the perception module. Cells that are predicted as hints are encouraged to have a reconstructed image similar to the input image, which can further regularize the initial estimation and the reasoning outputs.

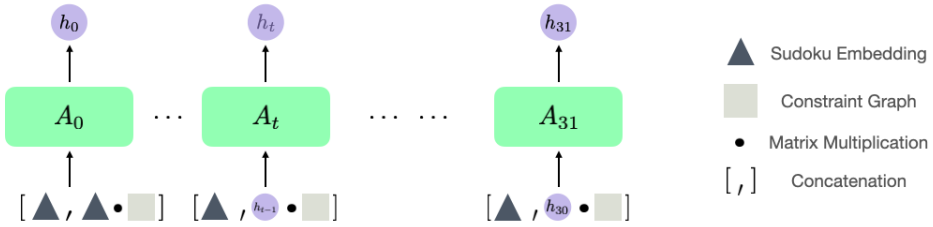

Figure 3 Modified LSTM model. The Sudoku embedding replaces each digit of a Sudoku with a learnable embedding. The constraint graph is a 81×81 matrix capturing Sudoku constraints between every pair of cells (i.e., there is a constraint between any pair of cells in the same row, same column, and same block). h_{31} goes through a fully connected layer to get the final reasoning results.

CLR-DRNets' enhanced reasoning module. DRNets' reasoning ability entirely relies on the continuous relaxation of the combinatorial constraints, i.e., it converts the discrete constraints into a differentiable loss function and solves the problem by optimizing it. However, this reasoning method is *stateless*, i.e., it is difficult to generalize to unseen data. CLR-DRNets' enhanced reasoning module employs a constraint graph and a modified LSTM (see Fig. 3) to learn from the training examples and generalize to unseen data. The constraint graph is a square matrix where each entry represents the constraints between these two elements.

Algorithm 1 Curriculum Learning framework for solving data-driven optimization problems.

Input: Training problem instances D, Target problem instances T, CLR-DRNets model M, Difficulty Gap G, Hard set of constraints ψ^h.
Select or generate a set of instances of proper difficulty level D_{train} based on ψ^h;
while D_{train} *is not as hard as* T **do**
  Train M using D_{train}via optimizing equation 1 ;
  do
    Select/generate a harder set of instances $D^{\dagger}_{\text{train}}$ based on D_{train}, ψ^h;
    $D_{\text{train}} \leftarrow D^{\dagger}_{\text{train}}$;
  while $D^{\dagger}_{train}$ *meets* G;
end

For example, the constraint graph for a 4×4 Sudoku is a 16×16 matrix. If entry (x, y) is one, it means the cell x and cell y are in the same row, column, or block otherwise zero. Then the input of each LSTM block is the concatenation of the Sudoku embedding and the multiplication of the hidden state and the constraints graph, where the Sudoku embedding replaces each digit of a Sudoku with a learnable embedding.

Now we explain how the unsupervised differentiable Sudoku loss function is derived, using DRNets' continuous relaxation. The reasoning module computes the digits' distribution for each Sudoku's cells and we denote the distribution of cell (i, j) as $P_{i,j}$ (row i, column j). The loss for encoding the Sudoku's row constraint is: $L_r = -\sum_{i=1}^{9} H(\frac{1}{9}\sum_{j=1}^{9} P_{i,j})$, where H is the entropy function. Similarly, we can define the column and block constraint loss L_c and L_b respectively. Since each cell contains one digit, the semantic constraint loss for cells is: $L_{\text{cell}} = \sum_{i=1}^{9}\sum_{j=1}^{9} H(P_{i,j})$. These loss functions minimize the entropy of each cell's digits distribution while maximizing the entropy of the average distribution of digits in each row, column, and box, forcing the distribution of each cell to converge to one digit while each row, column and box has different digits. Formally, the unsupervised **Sudoku constraint loss** is defined as: $L_{\text{Sudoku}} = \lambda_1(L_c + L_r + L_b) + \lambda_2 L_{\text{cell}}$, where λ_1 and λ_2 are weight scalars. Moreover, we show how the reconstruction loss for cell (i, j) is derived. Denote the embeddings of cell (i, j) as $z_{i,j}[t]$, where t can be any possible object, e.g., t can be digit 1 to 9 in visual Sudoku games. The predicted digit distribution from the perception module for the cell (i, j) is $P'(i, j)$. Note $P_{i,j}$ is defined above as the digits' distribution predicted by the reasoning module. These two distributions are mixed to form the $P_{\text{mix}}(i, j)$ for reconstruction: $P_{\text{mix}}(i, j)[t] = (1 - P'(i, j)[0]) * P'(i, j)[t] + P'(i, j)[0] * P(i, j)[t]$, where digit 0 represents the missing digits of the Sudoku.The reconstructed input for cell (i, j) is: $X_{\text{recon}}(i, j) = \sum_{t=1}^{9} P_{\text{mix}}(i, j)[t] * cGAN(z_{i,j}[t])$. The reconstruction loss is then defined as $L_{\text{recon}} = \sum_{i=1}^{9}\sum_{j=1}^{9} \mathcal{L}(X(i, j), X_{\text{recon}}(i, j))$, where X is the input image and $\mathcal{L}$ is any distance metric. This mixed distribution contributes to the reasoning ability of CLR-DRNets in which the wrong perception estimation can be corrected by the reasoning outputs while the initial estimation of images also help the reasoning module make decisions.

Note that we relax the hard discrete constraints into continuous constraint loss functions. The different difficulty of the relaxed constraint loss functions ψ^r and ψ^p is a key challenge for training, which could cause the optimization to focus on the easy part and converge to some local minimum quickly. Thus, we proposed the curriculum learning to tackle the difficulty imbalance among different losses.

Curriculum learning for CLR-DRNets. We introduce a curriculum learning framework (see Alg. 1) to manage the difficulty imbalance of the perception and reasoning tasks in the data-driven optimization problem. Based on the prior knowledge and problem structure, we can identify a set of constraints ψ^h that are quite hard to optimize. We can generate/find some problems that are easier enough in terms of the hard constraints set ψ^h. For example, in Visual Sudoku, completing the Sudoku is much harder than classifying digit images, so we set ψ^h as the Sudoku rules. Visual Sudoku instances with many hints are easier than those with fewer hints. A model that perfectly solves the easier instances can be easily trained. Based on the problem structure, we define a difficulty gap G to guide the generation/selection of instances that are slightly harder. For example, in Visual Sudoku, the difficulty gap G is set to be 5 hints: we remove 5 hints to gradually increase the instance difficulty. We then use these harder instances to continue training our model. This process can be repeated several times to finally solve the instances of desired difficulty level. More details in the appendix.

CLR-DRNets' training strategy. The success of CLR-DRNets relies on the seamless cooperation of the perception module, the reasoning module, and the generative decoder. We propose two training strategies for achieving this goal, different from DRNets', since DRNets do not have parameters for the reasoning module. Both strategies employ a pre-trained cGAN and a pre-trained classifier. The first strategy is *joint training*: we train the entire CLR-DRNets at the same time, i.e., optimizing the loss function (see equation 1) and the gradients affect both the perception module and the reasoning module. This joint strategy can handle the case where only noisy, e.g., handwritten, input data are available. However, for some challenging reasoning problems, the noisy input may harm the ability of the reasoning module. Thus, we propose a second strategy, *separate training*: we separately train the reasoning module with non-noisy input (i.e., input values are known), i.e., optimizing only the second term of the equation 1. This *separate training* strategy can tackle more intricate problems, but it requires the non-noisy input training data (no labels required though).

It is challenging to generalize a *single* data-driven optimization model to unseen problem instances, capturing all the logical relations across instances, given the combinatorial search space. So we introduce *restarts* to remedy this issue.

Restarts, a new model selection strategy. The reasoning effort required for solving visual combinatorial games can be huge. A single inference step of the deep learning model is unlikely to be able to solve all the instances with different levels of difficulty. We derive our objective function **without label supervision**, where the only supervision is based on the prior knowledge, so we can still optimize our CLR-DRNets model for a few steps in the test phase to further improve the model and customize it with respect to the test data. Also, we observe that the accuracy metric is increasing smoothly during the training process, but the set of training instances that can be solved varies quite a bit. Thus, we postulate that our model is doing local search to solve the problem and the model parameters can be loosely interpreted as heuristic of the search algorithm. Inspired by the *restart* scheme broadly used by the combinatorial optimization community [10, 2], we propose a new model selection method based on restarts (see Alg. 2). Since we have observed that we can get very different heuristics during the training process, so the restart scheme starts by collecting several models with top validation performance to form a model pool M. Then we start from one model and for each unsolved test case the loss function is optimized until the instance is solved or for a maximum of restart gap g steps, switching to the next model when all the

Algorithm 2 Restart scheme for CLR−DRNets.

Input: Test instances T, CLR-DRNets model pools M, Restart Gap g, metric ϕ to evaluate whether the instance is solved

```
idx ← 0;
while T = ∅ or idx < len(M) do
    m ← M[idx];
    for i ← 1 to g do
        R ← m(T); R: the solution of T.
        S ← φ(R); S: correctly solved instances of T.
        T ← T\S
        update parameter of m w.r.t equation 1;
    end
    idx ← idx + 1;
end
```

instances are processed. The scheme is supposed to fine-tune the model for the underlying test data, so the restart gap g is typically small. Thus the restart procedure takes much fewer time compared with the training. Note that this scheme does not require labels.

4 Experiments

4.1 Visual-Sudoku

As an example of Visual Sudoku, using digits, see Fig. 1(a). We also considered Visual Sudoku, using letters. The CLR-DRNets model for the Visual Sudoku is illustrated in Section 3 and Fig. 2. We prepare two training sets for the digit Visual Sudoku: one contains only noisy training data (denoted as noisy dataset), i.e., the digits of the Sudoku are images, another consists of non-noisy training data (denoted as non-noisy dataset), i.e., the value of digit/letter images are known. Note the training data for CLR-DRNets **do not include** the solution of the Sudoku. The noisy dataset consists of seven difficulty levels: 51, 46, 42, 36, 31, 25 and 20 hints. We generate $10,000$ Visual-Sudoku instances for each difficulty level. For the non-noisy dataset, we generate $100,000$ standard Sudoku instances with a uniform distribution of 18 hints to 25 hints. There is no difficulty imbalance issue for non-noisy dataset, so we do not separate the dataset based on its difficulty.

We explain our training and test settings here. For the *restarts* scheme, the size of the model pool M is 100, i.e. we select and save the top 100 models in terms of the validation performance. And the restart gap g is set to 10 steps, i.e., we move to the next model after optimizing one model 10 steps. The metric ϕ is whether the input Sudoku is solved. Note that we do not assign partial credits for the Sudoku solution. The loss we optimize in the training phase and the test phase is $L_{\text{Sudoku}} + \lambda_3 L_{\text{recon}}$. The distance metric we used for L_{recon} is $\mathcal{L}_1$ loss. We use Adam as the optimizer, and the learning rate is $3e-4$ (including for restarts (test)). The weight scalars $\lambda_1 = 1.0$, $\lambda_2 = 0.01$ (for training and testing) and $\lambda_3 = 0.001$.

We compare our model with DRNets [4], SATNET [22], a higher-order constraint optimization based approach [18] (referred to as HOCOP/HOCOP (C) where C refers to do the model calibration on the validation set) and a Cost Function Network based approach [3] (referred to as CFN). The learning settings of these methods are different. SATNET and

CFN both require the solutions (labels) of the data. HOCOP and CLR-DRNets do not require the solutions, however, HOCOP leverages a high-efficient Sudoku solver and this kind of solver may not exist for other problems. **CLR-DRNets use the fewest supervision (no solutions (labels)) to learn to solve the problem and generalize to unseen data.** We use two test datasets. The first dataset (denoted as $D_{\text{avg-36hints}}$) is SATNET's dataset (CNF and HOCOP also use this dataset) consisting of Sudokus with a mean of 36 hints, and the second dataset (denoted as D_{17hints}) is formed of 1000 Sudoku with 17 hints. The CLR-DRNets model is trained on the noisy dataset (*joint training*) to solve the $D_{\text{avg-36hints}}$ and on the non-noisy dataset (*separate training*) to solve the D_{17hints}.

Table 1 The test set Sudoku accuracy on the Digit Visual Sudoku task for different approaches.

Dataset	CLR-DRNets	DRNets	SATNET	CFN	HOCOP	HOCOP (C)	ResNet+SAT
$D_{\text{avg-36hints}}$	**0.996**	0.81	0.632	0.763	0.929	**0.996**	0.821
D_{17hints}	0.88	0	0	NA	NA	NA	**0.918**

CLR-DRNets outperform SATNET, DRNets, HOCOP and CFN on both datasets (see Table. 1) and CLR-DRNets do not require the labels. ResNet+SAT denotes a sequential coupling of ResNet with a SAT Solver, i.e., passing the digit classification of the input hand-written Sudoku, using ResNet, as input to a modern SAT-Solver to get the final results. For the dataset with an average of 36 hints, CLR-DRNets significantly surpass the performance of ResNet+SAT since CLR-DRNets can correct some perception mistakes guided by the Sudoku rules. Note though that CLR-DRNets do not outperform ResNet+SAT on 17 hint instances, given the high digit accuracy of ResNet for few number of hints, which results in high probability of perfect recognition of the input Sudoku. Nevertheless, often in many tasks we may not get an almost-perfect perception model, which is the case of e.g., the letter Visual Sudoku (the classifier accuracy is only 97.5%). In fact, when tested on the letter Visual Sudokus (trained using *separate training* on the non-noisy dataset), CLR-DRNets consistently outperform ResNet+SAT and DRNets for all the instances (see Fig. 4, left panel).

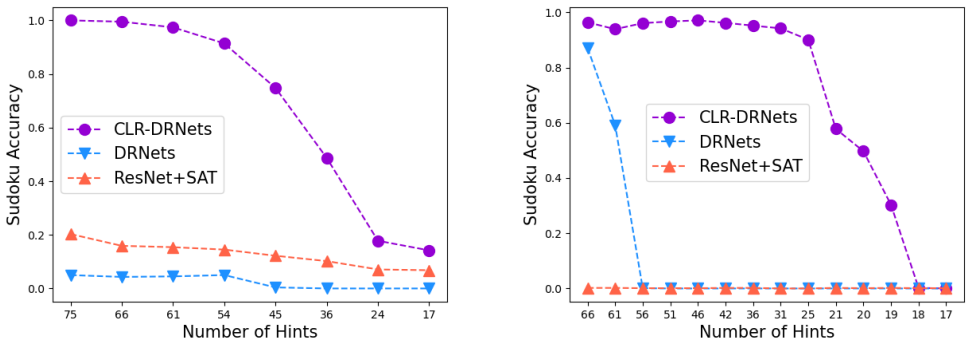

Figure 4 Accuracy of CLR DNRNets, DRNets and ResNet+SAT on Letter Visual Sudoku (left) and Visual Mixed Sudoku (right).

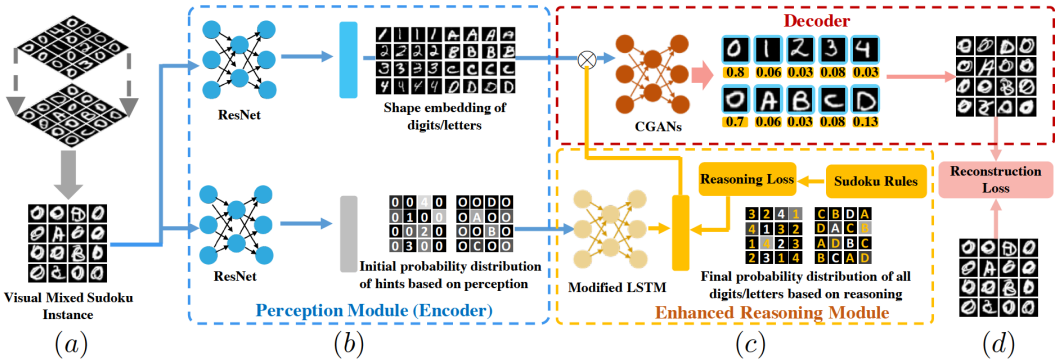

Figure 5 (a) A visual mixed Sudoku instance. We use a max operator to mix two visual Sudokus. (b) The perception module (encoder) of CLR-DRNets. The top blue rectangle is the latent space capturing the shape information of all possible digits and letters per cell. The bottom gray rectangle is the initial estimation of the input visual mixed Sudoku instance, only based on perception. Here we employ a heat map to represent the predicted confidence of each cell. (c) The enhanced reasoning module consists of a modified LSTM model. It takes the initial estimation as the input, constrained by the relaxed Sudoku rules loss function, computing the completion results. (d) The decoder leverages two pre-trained cGANs to generate all the possible digit and letter images w.r.t the shape embeddings of (b) per cell. Then we can reconstruct each cell guided by the completion results of (c). (We use 4×4 Sudokus for easier visualization. All our experiments are with 9×9 Sudokus.)

4.2 Visual Mixed Sudoku

The Visual Mixed Sudoku task is computationally more challenging than Visual Sudoku, requiring a tighter combination between perception and reasoning. We mix (using max operator) two Visual Sudokus (see Fig. 1(c)) to form a Visual Mixed Sudoku instance, offsetting the digits and letters to the top left and bottom right direction by two pixels. One Visual Sudoku consists of digits 0 to 9 (0 denotes the empty cell). The other one is formed of letters A to J (J denotes the empty cell). All the digit images are sampled from MNIST [15] and all the letter images are sampled from EMNIST [5]. The sizes of the training set, with different difficulty, and test set are all $10,000$. We employ the *joint training* strategy to train the CLR-DRNets model since the pre-trained classifier can only achieve an accuracy of around 90% (due to the complexity of classifying mixed digits and images). The CLR-DRNets model for Visual Mixed Sudoku is similar to that for Visual Sudoku and its framework is illustrated in the Fig. 5. The perception module (see Fig. 2(b)) consists of two ResNet-18 [13] models. One model is used to generate the shape embeddings for all possible letters and digits per cell. The other model is employed to generate the initial probability distribution of each cell. The reasoning module (see Fig. 2(c)) consists of a modified LSTM, the same as for Visual Sudoku. The initial probability distributions are fed into the modified LSTM to compute the final probability distribution. The decoder (see Fig. 2(d)) consists of two cGANs (G_d for digits and G_l for letters). The shape embeddings are fed into two cGANs to generate all the possible letter and digit images of each cell. We formally define how we reconstruct the input images. For each cell, denote the shape embedding as $z_{d,0} \cdots z_{d,9}$ for digits, $z_{l,A} \cdots z_{l,J}$ for letters and denote the final probability distribution as $P_{d,0} \cdots P_{d,9}$ for digits, $P_{l,A} \cdots P_{l,J}$ for letters. We reconstruct the input image as: $\max\left(\sum_{i=0}^{9} P_{d,i} G_d(z_{d,i}), \sum_{i=A}^{J} P_{l,i} G_l(z_{l,i})\right)$. We use L_1 loss as our reconstruction loss (L_{recon}).

We explain our training and test settings here. For training, we separately train two cGANs with part of the MNIST/EMNIST dataset, i.e. the digit and letter images used to pre-train the cGAN have no intersection with the visual Sudokus' images. The other parts

are jointly trained through optimizing the loss function $L = L_{\text{Sudoku}} + \lambda_3 L_{\text{recon}}$. To tackle the different difficulty of the perception and reasoning part, we start with easy Visual Mixed Sudoku instances, i.e., with 75 (out of 81) hints. And the following curriculum tasks are designed as 66, 61, 56, 51, 46, 42, 36, 31 and 25 hints. For each difficulty level, we generate $10,000$ Mixed-Visual-Sudoku instances. From 25 hints to 24 hints, we observe a phase transition property of the combinatorial search problem so we can only train our model using instances with 25 hints. But surprisingly, the model can still generalize to harder cases (less than 25 hints). In our experiments, we always mix two Visual Sudokus with the same number of hints. For the *restart* scheme, the size of the model pool M is 20. The restart gap is 100. The metric ϕ is to check the validity of the Sudokus' solution. The loss function we optimized for the *restart* scheme (test) is the same as for training, i.e. $L_{\text{Sudoku}} + \lambda_3 L_{\text{recon}}$. Now we have 4 weight scalars for the Sudoku loss equation, denoted as $\lambda_{1,d}, \lambda_{2,d}, \lambda_{1,l}$ and $\lambda_{2,l}$, where d and l refer to digit and letter. We set them as $\lambda_{1,d} = \lambda_{1,l} = 1.0, \lambda_{2,d} = 0.01, \lambda_{2,l} = 0.02$. We up-weight the λ_2 for letters Sudoku in that recognizing letters is usually harder than digits. λ_3 is set as 0.005. These parameters are the same for training and testing. The optimizer we selected is Adam and the learning rate is $3e-4$ in the training and $1e-4$ in the *restarting* (test) phase.

CLR-DRNets' Sudoku accuracy is significantly higher than DRNets' and also better than ResNet+SAT (see Fig. 4). De-mixing is very challenging for standard deep learning methods. DRNets can only solve some easy instances (e.g., 66 hints). When the number of hints decreases, the difficulty of the reasoning task increases comparatively to the perception task, making it more challenging (or infeasible) for DRNets to learn the task.

4.3 Ablation Studies

The results above show that CLR-DRNets significantly outperform the baselines, due to (1) the curriculum learning (see Alg. 1) and (2) the *restart* scheme (see Alg. 2). We conducted ablation studies to analyze the contribution of the two factors and the results are showed in the table 2. In both tasks, curriculum learning contributes the most to the improvement. *Restarts* also play an important role, especially for challenging instances (17 hint Visual Sudoku and 20 hints Visual Mixed Sudoku). We postulate that the 25 hint case is not very hard, therefore a single model suffices.

Table 2 The test set Sudoku accuracy performance on different tasks, we report the proportion to the CLR-DRNets' results. r refers to the *restart* scheme and c refers to the curriculum learning.

Task	CLR-DRNets	w/o c	w/o r	w/o r+c
Visual Sudoku (17-hints)	1.0	0.237	0.450	0.007
Visual Mixed Sudoku (20-hints)	1.0	0.005	0.955	0.004
Visual Mixed Sudoku (25-hints)	1.0	0.009	0.472	0

4.4 Standard 17-hints Sudoku

We also evaluate CLR-DRNets on learning to solve standard Sudokus (i.e., hint values known), supervised only by the Sudoku rules (no labeled data). We unsupervised train it on standard Sudoku task by simply optimizing the loss function, L_{Sudoku}. The model architecture is exactly the same as the reasoning module of Visual Sudoku, i.e. a modified LSTM. We train our model on $100,000$ Sudoku with a uniform distribution of 18 to 25 hints.

And test on 1,000 Sudoku with 17 hints. We compared CLR-DRNets against RRN [20], which is a totally supervised method also leveraging the Sudoku rules to design the model architecture.

Table 3 Sudoku accuracy for solving 17-hint standard Sudokus.

	CLR-DRNets	RRN
Sudoku Accuracy	0.912	0.64

From Table 3, we can see that CLR-DRNets outperform RRN largely. Here RRN means that we train RRN model on our training set (18–25 hints Sudoku). The reason RRN does not perform as well as their paper is that we never let RRN see the 17 hints Sudoku in the training phase, so RRN cannot generalize as CLR-DRNets does to the 17-hints case. The optimizer is Adam with learning rate $1e-3$. We use a learnable embedding for each digit 0 to 9 with dimension 10. The model pool M for *restart* scheme is 100 and the restart gap g is 10. And the learning rate during the *restart* scheme is also $1e-3$.

5 Conclusions

We introduce CLR-DRNets, a curriculum-learning-with-restarts framework for DRNets, along with an enhanced reasoning module. We demonstrate the CLR-DRNets' effectiveness on challenging single-player visual combinatorial games, achieving state-of-the-art performance with weak supervision from prior knowledge (domain rules).

References

1 Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. *arXiv preprint*, 2017. arXiv:1703.00443.

2 Armin Biere and Andreas Fröhlich. Evaluating CDCL restart schemes. In Daniel Le Berre and Matti Järvisalo, editors, *Proceedings of Pragmatics of SAT 2015, Austin, Texas, USA, September 23, 2015 / Pragmatics of SAT 2018, Oxford, UK, July 7, 2018*, volume 59 of *EPiC Series in Computing*, pages 1–17. EasyChair, 2018.

3 Céline Brouard, Simon de Givry, and Thomas Schiex. Pushing data into cp models using graphical model learning and solving. In *International Conference on Principles and Practice of Constraint Programming*, pages 811–827. Springer, 2020.

4 Di Chen, Yiwei Bai, Wenting Zhao, Sebastian Ament, John Gregoire, and Carla Gomes. Deep reasoning networks for unsupervised pattern de-mixing with constraint reasoning. In *International Conference on Machine Learning*, pages 1500–1509. PMLR, 2020.

5 Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

6 Travis Dick, Eric Wong, and Christoph Dann. How many random restarts are enough. Technical report, Technical report, 2014.

7 Dieqiao Feng, Carla P. Gomes, and Bart Selman. Solving hard AI planning instances using curriculum-driven deep reinforcement learning. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 2198–2205. ijcai.org, 2020. doi:10.24963/ijcai.2020/304.

8 Matteo Gagliolo and Jürgen Schmidhuber. Learning restart strategies. In *IJCAI*, pages 792–797, 2007.

9 Artur Garcez, Tarek R Besold, L d Raedt, Peter Fo ldiak, Pascal Hitzler, Thomas Icard, Kai-Uwe Ku hnberger, Luis C Lamb, Risto Miikkulainen, and Daniel L Silver. Neural-symbolic learning and reasoning: contributions and challenges, 2015.

10 Carla P Gomes, Bart Selman, Henry Kautz, et al. Boosting combinatorial search through randomization. *AAAI/IAAI*, 98:431–437, 1998.

11 Akhilesh Gotmare, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation. *arXiv preprint*, 2018. `arXiv:1810.13243`.

12 Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. *arXiv preprint*, 2019. `arXiv:1904.03626`.

13 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*, pages 770–778, 2016.

14 Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

15 Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

16 Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems*, pages 3749–3759, 2018.

17 Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint*, 2014. `arXiv:1411.1784`.

18 Maxime Mulamba, Jayanta Mandi, Rocsildes Canoy, and Tias Guns. Hybrid classification and reasoning for image-based constraint solving. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 364–380. Springer, 2020.

19 Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *arXiv preprint*, 2020. `arXiv:2003.04960`.

20 Rasmus Palm, Ulrich Paquet, and Ole Winther. Recurrent relational networks. In *Advances in Neural Information Processing Systems*, pages 3368–3378, 2018.

21 Zhipeng Ren, Daoyi Dong, Huaxiong Li, and Chunlin Chen. Self-paced prioritized curriculum learning with coverage penalty in deep reinforcement learning. *IEEE transactions on neural networks and learning systems*, 29(6):2216–2226, 2018.

22 Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*, pages 6545–6554, 2019.

23 Daphna Weinshall, Gad Cohen, and Dan Amir. Curriculum learning by transfer learning: Theory and experiments with deep networks. *arXiv preprint*, 2018. `arXiv:1802.03796`.

24 Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems*, pages 2319–2328, 2017.

A Appendix

DRNets' continuous relaxation for discrete constraints

Here we introduce more formally how DRNets apply continuous relaxation to the discrete constraints. The basic idea is to employ entropy to model the discrete constraints. For example, in Sudoku, we require that each row, col, and block, is filled with different digits (denoted as AllDiff constraints). Then for each cell, we use a probability distribution over all the possible digits to represent the estimation of one cell. We add all the cells' probability

within one row, col or block, consider the best case, the summation vector should be an all-one vector, which also means it has the highest entropy. Thus, to encourage the model's output to satisfy the AllDiff constraints, we maximize the entropy defined above. More formally, use $e_i, i = 1 \dots n$ to represent the discrete variable and $p_i, i = 1 \dots n$ for the corresponding probability distribution. Then to represent $e_i \neq e_j \ \forall i \neq j$, we can maximize the function, $H(\sum_{i=1}^{n} p_i)$,

where H refers to the entropy. We also want to force the probability distribution to converge to one point since each distribution actually refers to one single point (denoted as Cardinality constraint). We can minimize the relaxed loss function to achieve this, i.e., $\sum_{i=1}^{n} H(p_i)$

The last discrete constraint we usually use is to select k items from n candidates. This is a little bit tricky, but we can use a hinge style loss to do that. If we want to select exactly k items from $e_i, i = 1 \dots n$, we can optimize this relaxed loss function, $\max(H(\sum_{i=1}^{n} p_i) - \log(k), 0)$. The idea here is simple, if the entropy probability distribution summation is $\log(k)$, supported by the Cardinality constraint, we actually selected k items from n candidates.

Once we have these powerful relaxations, we can convert a constrained optimization problem to an unconstrained optimization problem. Consider we want to optimize the objective function L under the constraint ϕ. Then we can firstly continuously relax the constraint ϕ to ψ, and optimize the loss function L_{relax} ($L_{\text{relax}} = L + \lambda\psi$), which is approximately equal to solving the original constraint optimization problem.