SATNet

Overview

- Background
- Framework
- Issues with SAT Net
  - Integer programming approach
  - End-to-end training results in very poor performance
- Symbol grounding
- Beyond SAT Net
- Concluding Thoughts

SAT Net

Background

The MAX SAT problem

INPUT:
Given Boolean variables $x_1, \ldots, x_m$ and causes $C_1, \ldots, C_n$ where each clause is a disjunction of literals (atoms or negations). (Conjunctive normal form)

OUTPUT:
An assignment of Boolean variables such that the number of satisfied clauses is maximized.

Notes:

- Known to be NP-hard (even when each clause has just two literals)
- The clauses can be numerically represented by matrix (M) of dimensions $n \times m$ (in the original paper, S is used instead of M)
- Often framed as an optimization problem

Partial Knowledge on Boolean Variables

An extension to the problem is to partition the m Boolean variables into two groups: input ($a^{in}_{1,\dots,k}$) and output ($a^{out}_{k+1,\dots,m}$)

Hence, we can think of MAX SAT as the following problem:

$$a^{out}_{k+1,\dots,m} = S(a^{in}_{1,\dots,k}, M)$$

Where S is an oracle and M is the matrix representing the clauses.

A **visual** variant of the problem is one in which the input is presented as images instead of text.

Problems

- Solving MAX SAT given the clauses:
  - SAT solvers, Integer programming, Semi definite programming
- Solving visual variant of MAX SAT problem (with known clauses) in an end-to-end fashion
  - CNN + differentiable architecture
  - NeurASP (in a few lessons)
- Solving the MAX SAT problem when the clauses are not known / solving visual variant is processed by a pre-trained CNN
  - This talk
- Solving the visual MAX SAT problem when the clauses are not known and no supervision on images
  - This talk

Sudoku Puzzle

Sudoku is a popular number puzzle that requires you to fill blanks in a 9X9 grid with digits so that each column, each row, and each of the nine 3×3 subgrids contains all of the digits from 1 to 9.

From https://github.com/Kyubyong/sudoku

Can be treated as an instance of MAX SAT

Above GitHub site is a good source for training and testing data

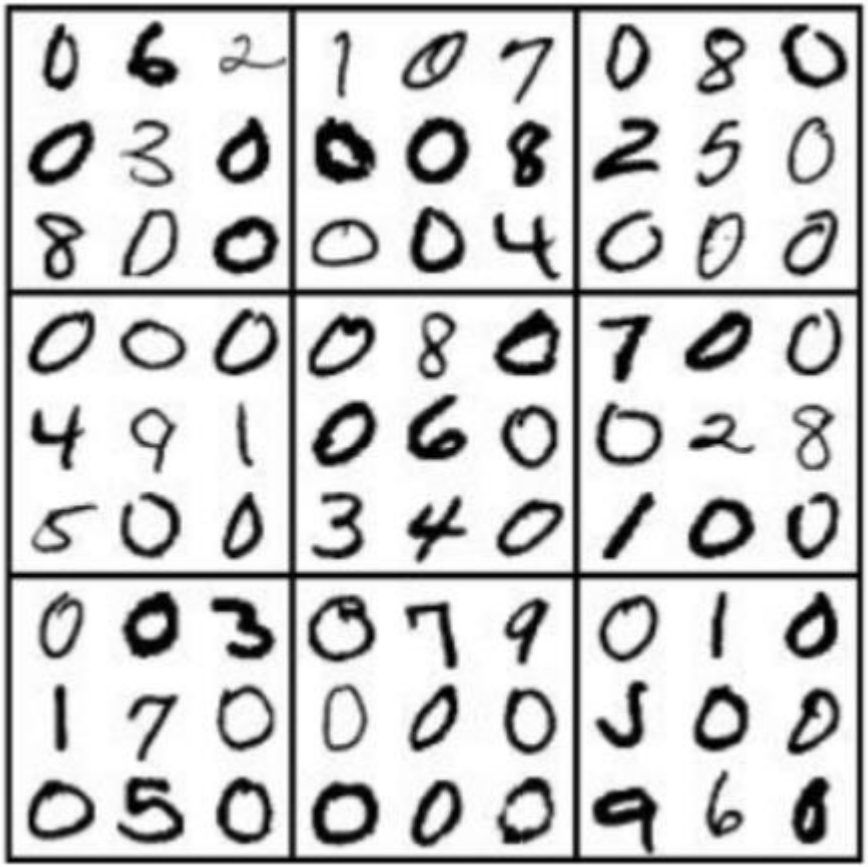
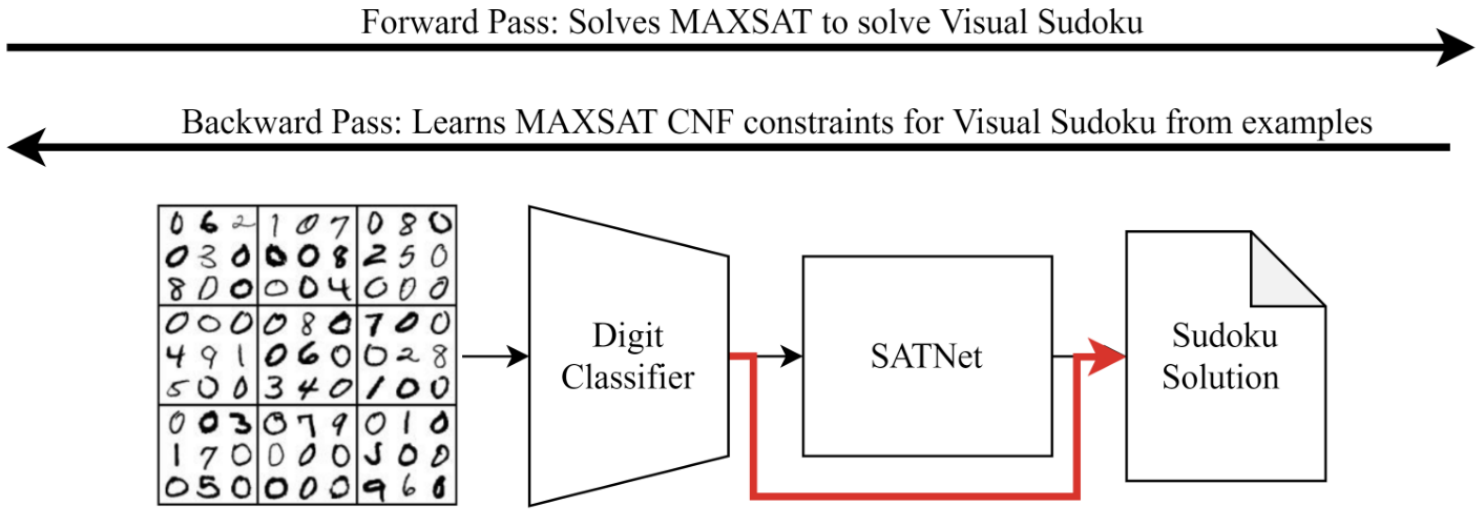


Image from Topan et al., NeurIPS 2021.

SAT Net

Framework

SAT Net Framework



Notes:

- For Sudoku, MSE loss is used; cross-entropy loss for visual variant
- A relaxation via Semi Definite programming is used to solve the MAX SAT instance
- Coordinate descent is used instead of gradient descent (both forward and backward passes) as the authors found improved performance with their semidefinite approach to MAX SAT
  - Note that this is for optimization within the forward and backward pass, SGD is used in the overall training
- For the visual Sudoku problem, LeNet is used to classify the digits (pre-trained)

Image from Topan et al., NeurIPS 2021.

Intuition: Overall Approach

- Z vectors are actual inputs and outputs
- V vectors are relaxations
- S is the weight matrix
- U is a matrix derived from the gradient wrt the relaxed output and used to compute the gradient for the weights

Algorithm 1 SATNet Layer

procedure INIT()
  // rank, num aux vars, initial weights, rand vectors
  init m, n_{aux}, S
  init random unit vectors $v_\top, v_i^{\text{rand}}\ \forall i \in \{1, \ldots, n\}$
  // smallest k for which (2) recovers SDP solution
  set $k = \sqrt{2n} + 1$

procedure FORWARD($Z_\mathcal{I}$)
  compute $V_\mathcal{I}$ **from** $Z_\mathcal{I}$ via (5)
  compute $V_\mathcal{O}$ **from** $V_\mathcal{I}$ via coord. descent (Alg 2)
  compute $Z_\mathcal{O}$ **from** $V_\mathcal{O}$ via (7)
  return $Z_\mathcal{O}$

procedure BACKWARD($\partial \ell / \partial Z_\mathcal{O}$)
  compute $\partial \ell / \partial V_\mathcal{O}$ via (8)
  compute U **from** $\partial \ell / \partial V_\mathcal{O}$ via coord. descent (Alg 3)
  compute $\partial \ell / \partial Z_\mathcal{I}$, $\partial \ell / \partial S$ **from** U via (12), (11)
  return $\partial \ell / \partial Z_\mathcal{I}$

Algorithm from Wang et al., ICML 2019.

Intuition: Forward Pass

- Key thing to remember: the forward pass is itself solving *another* optimization problem (using the SDP relaxation) – this is just a subroutine in the overall optimization process
- The outputs of the forward pass are evaluated against a standard loss function

Figure from Wang et al., ICML 2019.

Intuition: Backward Pass

- Key result to derive the gradients w.r.t. weights and inputs
- U is the key item to compute and is done via coordinate descent

Theorem 1. *Define $P_o \equiv I_k - v_o v_o^T$ for each $o \in \mathcal{O}$. Then, define $U \in \mathbb{R}^{k \times n}$, where the columns $U_{\mathcal{I}} = 0$ and the columns $U_{\mathcal{O}}$ are given by*

$$\mathrm{vec}(U_{\mathcal{O}}) = (P((C+D) \otimes I_k)P)^{\dagger} \,\mathrm{vec}\left(\frac{\partial \ell}{\partial V_{\mathcal{O}}}\right), \quad (9)$$

where $P \equiv \mathrm{diag}(P_o)$, where $C \equiv S_{\mathcal{O}}^T S_{\mathcal{O}} - \mathrm{diag}(\|s_o\|^2)$, and where $D \equiv \mathrm{diag}(\|g_o\|)$. Then, the gradient of the network loss ℓ with respect to the relaxed layer inputs is

$$\frac{\partial \ell}{\partial V_{\mathcal{I}}} = -\Big(\sum_{o \in \mathcal{O}} u_o s_o^T\Big) S_{\mathcal{I}}, \quad (10)$$

where $S_{\mathcal{I}}$ is the $\mathcal{I}$-indexed column subset of S, and the gradient with respect to the layer weights is

$$\frac{\partial \ell}{\partial S} = -\Big(\sum_{o \in \mathcal{O}} u_o s_o^T\Big)^T V - (SV^T)U. \quad (11)$$

Theorem 1 is from Wang et al., ICML 2019.

Sudoku Experimental Results

SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver

Model	Train	Test
ConvNet	72.6%	0.04%
ConvNetMask	91.4%	15.1%
SATNet (ours)	99.8%	**98.3%**

(a) Original Sudoku.

Model	Train	Test
ConvNet	0%	0%
ConvNetMask	0.01%	0%
SATNet (ours)	99.7%	**98.3%**

(b) Permuted Sudoku.

Model	Train	Test
ConvNet	0.31%	0%
ConvNetMask	89%	0.1%
SATNet (ours)	93.6%	**63.2%**

(c) Visual Sudoku. (Note: the theoretical "best" test accuracy for our architecture is 74.7%.)

- Uses approach of https://github.com/Kyubyong/sudoku as a baseline (note that is only a GitHub site, not a paper, and was only evaluating Sudoku on CNN's compared to training data)
- The Visual Sudoku tests used LeNet architecture into the end-to-end framework – the authors claimed end-to-end training
- Permutated rows were to illustrate that SAT Net was not overfitting based on row position
- The number of clauses learned by SAT Net was limited to avoid overfitting (m=600)
- Why does the CNN fail and end up overfitting? This is a combinatorial problem, so the amount of required training data could be exponential

Sudoku Experimental Results

Chang et al., NeurIPS 2020 showed:

SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver

Model	Train	Test
ConvNet	0%	0%
ConvNetMask	0.01%	0%
SATNet (ours)	99.7%	**98.3%**

(b) Permuted Sudoku.

Model	Train	Test
ConvNet	0.31%	0%
ConvNetMask	89%	0.1%
SATNet (ours)	93.6%	**63.2%**

(c) Visual Sudoku. (Note: the theoretical "best" test accuracy for our architecture is 74.7%.)

Results of visual case highly sensitive to initial conditions (~80% reduction in accuracy)

The output labels included the symbolic representation of the input digits – so there was label leakage (which actually causes total failure)

Arbitrarily increasing number of parameters led to model failure (unlike standard DL)

- [om/Kyubyong/sudoku](https://github.com/Kyubyong/sudoku) as a baseline (note that is only a GitHub site, not a paper, and was only evaluating Sudoku on CNN's compared to training data)
- The Visual Sudoku tests used LeNet architecture into the end-to-end framework – the authors claimed end-to-end training
- rate that SAT Net was not overfitting based on row
- The number of clauses learned by SAT Net was limited to avoid overfitting (m=600)
- Why does the CNN fail and end up overfitting? This is a combinatorial problem, so the amount of required training data could be exponential

Despite its shortcomings, SAT Net has significacne

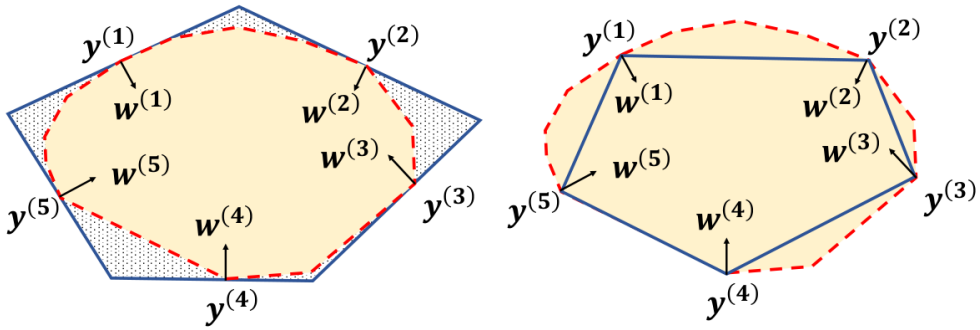
- It successfully could learn constraints in a differentiable framework
- Combinatorial forward pass and ability to derive gradients for backpropagation
- Significantly outperformed standard DL architectures
  - (this was significant at the time of publication)

SAT Net

Issues with SAT Net

Integer Programming Approach Outperforms SAT Net on Sudoku

Strategy: derive constraints from the inner and outer polytopes based on training data. Squeeze polytopes until convergence.



Outperformed SAT Net on Sudoku.

However, this approach is not differentiable.

But, if SAT Net cannot be used in end-to-end training, then why not just use an IP approach and get better results

Model	Train	Test
ConvNet (Park, 2018)	72.6%	0.04%
ConvNetMask (Park, 2018)	91.4%	15.1%
SATNet (Wang et al., 2019)	99.8%	98.3%
Outer + EQ (ours)	100%	100%

(b) Permuted Sudoku

Model	Train	Test
ConvNet (Park, 2018)	0%	0%
ConvNetMask (Park, 2018)	0.01%	0%
SATNet (Wang et al., 2019)	99.7%	98.3%
Outer + EQ (ours)	100%	100%

Figure and table from Meng and Chang, ICML 2021.

The Case Against SAT Net

At NeurIPS 2020, Chang et al., reviewed SAT Net in a paper and found major issues with the results around Visual Sudoku

- Label Leakage
- Initialization / hyperparameters / training decisions
- Symbol grounding

Label Leakage

- Wang et al. claimed SAT Net for Visual Sudoku was trained in an end-to-end fashion
- However, while training samples were visual, the ground truth labels included the symbolic representation of the visual input

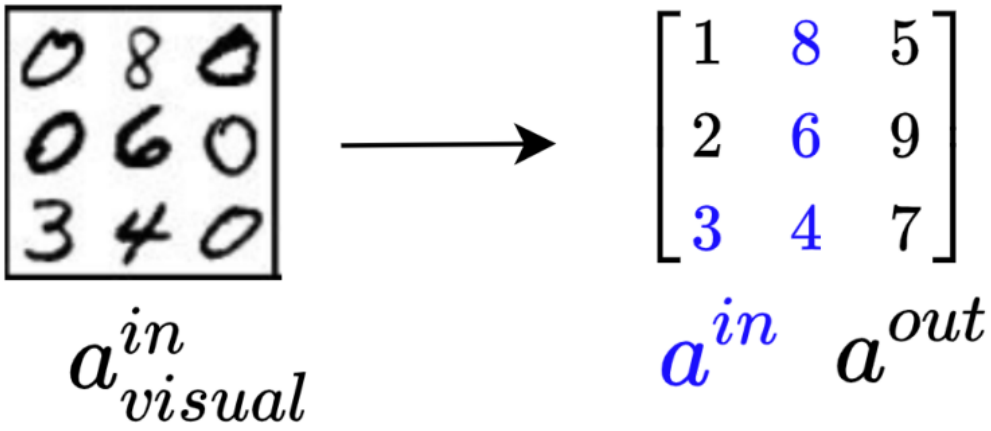


Image from Topan et al., NeurIPS 2021.

Sensitivity to Initial Conditions and the Use of Leaked Labels

Successful Run (20%)

Unsuccessful Run (80%)

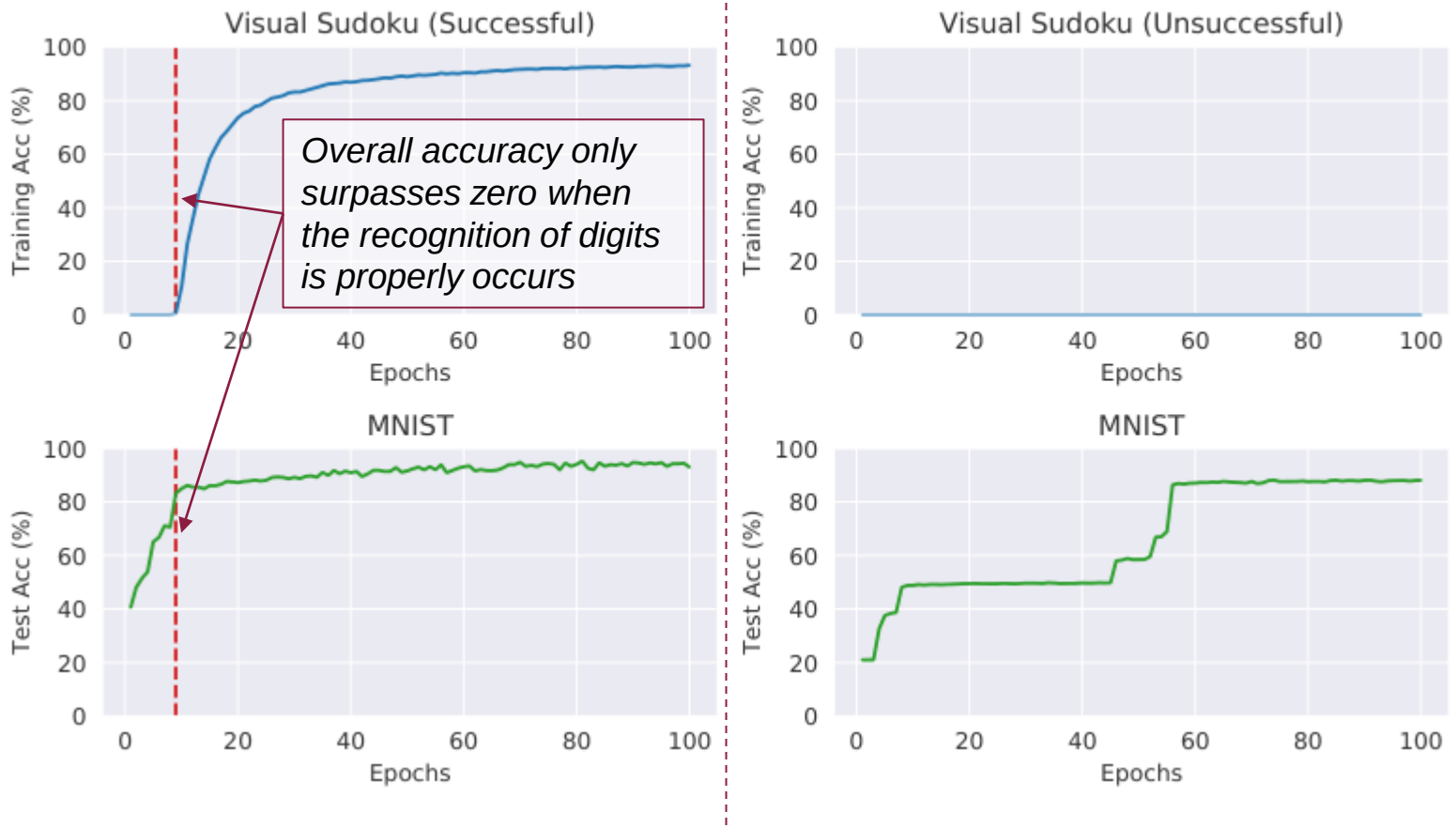


Figure Chang et al., NeurIPS 2021.

The relationship between symbols and perception

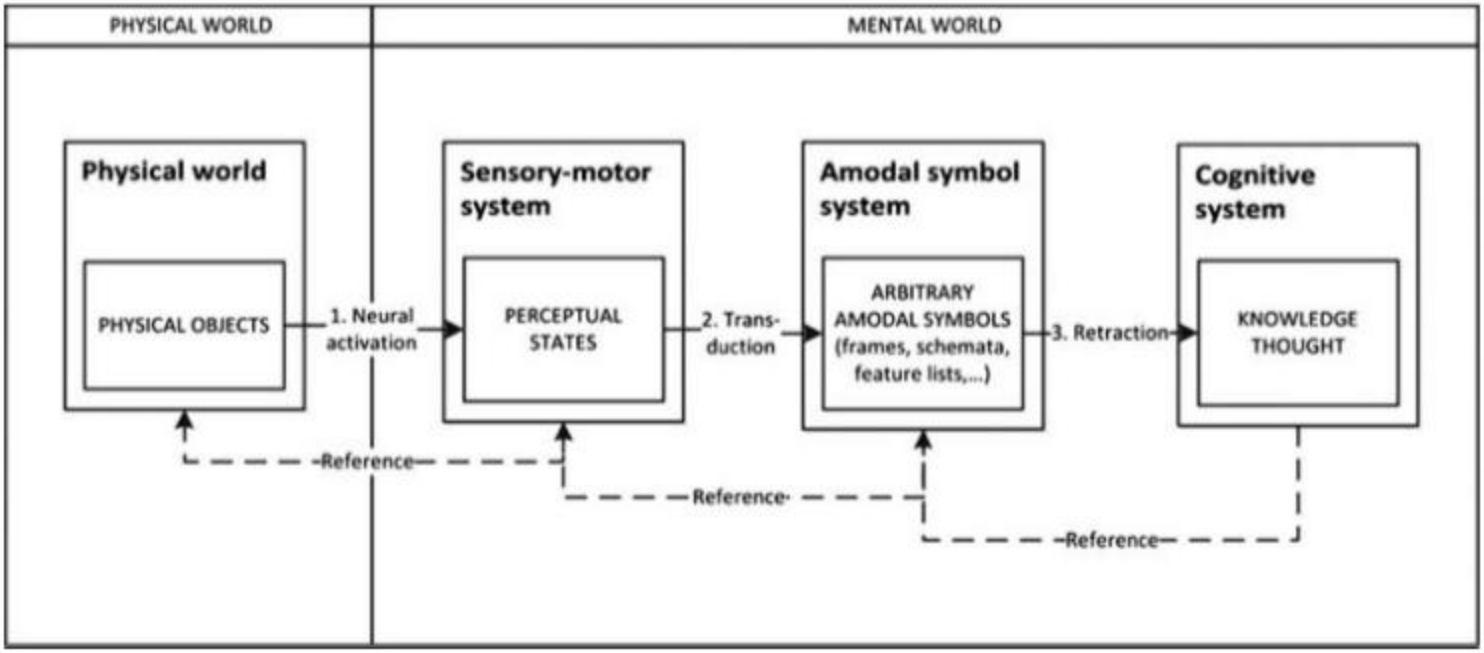
- Transduction problem
  - If they exist, how then, are the perceptual states mapped into amodal symbols? (Barsalou, 1999)
- Symbol grounding problem
  - The reverse of the transduction problem
  - How are amodal symbols grounded in perception? (Barsalou, 1999)
- Chang et al. argue that SAT Net did not adequately solve the symbol grounding problem.
  - They probably really mean the transduction problem – as the issue was the transduction of perception into symbols
  - Note: Symbol grounding does come up in ML verification – ensuring that a symbol maps back properly to percpetion

Failure of symbol grounding

Challenges in symbol grounding are somewhat different than the problem of SAT Net – i.e., how do we determine (without human inspection) that symbols are properly grounded in perception

Source: Zhang et al., 2019: https://openreview.net/pdf?id=SJgEl3A5tm

Perception-Cognition:
A view from cognitive science



Transduction deals with mapping perception to symbolic representations

Figure Coëgnarts, 2017.

Testing SAT Net's ability to address transduction

Chang et al. re-evaluate SAT Net by masking the output

	Non-Visual Sudoku		Visual Sudoku	
Accuracy	Original	Masked Outputs	Original	Masked Outputs
Train	99.7±0.0%	99.7±0.0%	18.5±12.3%	0.0±0.0%
Test	97.6±0.1%	97.6±0.1%	11.9±7.9%	0.0±0.0%

The argument presented by Chang et al. is that SAT Net does not solve the transduction* but only appeared to due to a combination of label leakage and tuned training/hyperparameter settings.

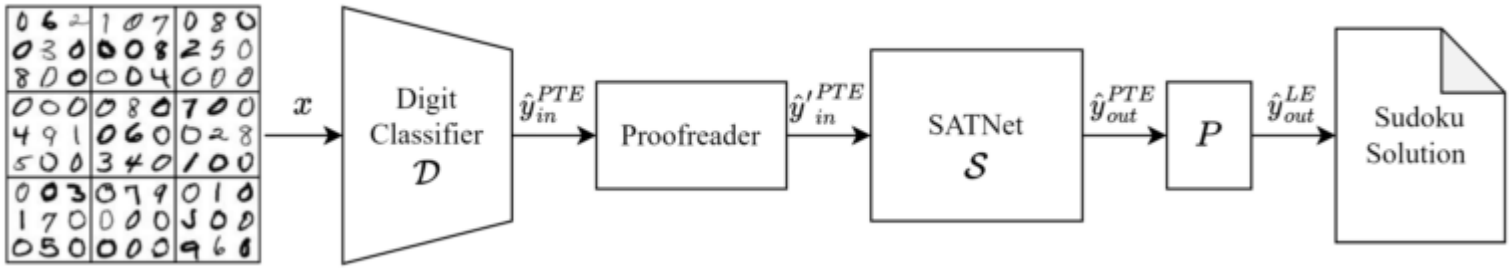
They further explore the limits of SAT Net in their paper for visual problems

However, the non-visual results largely held

*The authors use the term "symbol grounding"

Table from Chang et al., NeurIPS 2021.

Unsupervised Learning to address transduction



- Topan et al., seek to directly address the shortcomings of SAT Net:
  - Use of unsupervised learning for digit recognition
  - Additional loss term to account for in accurate digits
  - Addition of proofreader layer improved performance (an extra boost, but not directly related to the problem of transduction)

Image from Topan et al., NeurIPS 2021.

Use of permutation matrix

$$\hat{y}_{in}^{PTE} P = \hat{y}_{in}^{LE}$$

- The pre-trained encodings (PTE) of input digits is directly related to the (correct) label encodings (LE) by some permutation matrix P.

$$\hat{y}_{out}^{PTE} P \approx y^{LE}$$

- We can assume that results from the supervised training is related to label encoding via the same matrix P.

Image from Topan et al., NeurIPS 2021.

Two-phased training

1. Freeze digit classifier (the unsupervised method) and train SAT Net weights with a special loss function that allows us to also learn permutation matrix P. Stop training upon convergence of P.
2. Freeze P and switch to traditional loss (cross-entropy) and learn weights; the digit classifier is also unfrozen at this point as well

Results

Model Configuration	Grounded vs. Ungrounded Data	Total Board Accuracy (%)	Per-Cell Accuracy (%)	Visual Accuracy (%)
Original SATNet	grounded	66.5 ± 1.0	98.8 ± 0.1	99.0 ± 0.0
Original SATNet	ungrounded	0 ± 0.0	11.2 ± 0.1	11.6 ± 0.0
Our Method	**ungrounded**	$\mathbf{64.8 \pm 3.0}$	$\mathbf{98.4 \pm 0.2}$	$\mathbf{98.9 \pm 0.1}$

Table from Topan et al., NeurIPS 2021.

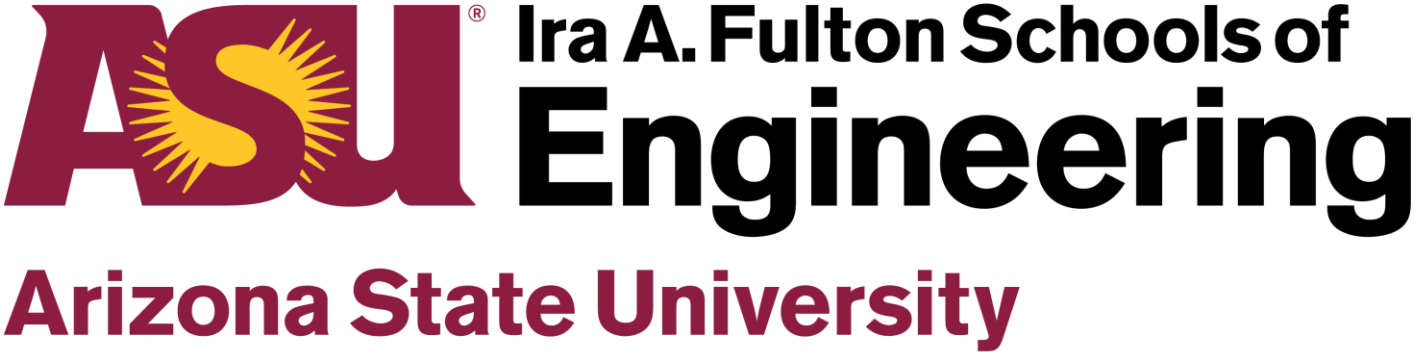
Concluding Thoughts

- The original SAT Net paper studied how to learn logical constraints, do back propagation where the forward function is a combinatorial problem and address transduction – was this too much of a leap for a single study?

- Practicalities of training significantly altered results – how often does this happen in DL papers?

- Topan et al. do not fully solve transduction – they just do it for Soduku. They essentially leverage the fact that the solution still has numbers (even when unmasked) – will this be the case in other, more real-world problems?

- There are other approaches to learning of constraints (papers posted on today's lesson)


Ira A. Fulton Schools of
Engineering
Arizona State University