Neural Sequence-to-grid Module for Learning Symbolic Rules

[1]Segwang Kim, [2]Hyoungwook Nam, [1]Joonyoung Kim, [1]Kyomin Jung
[1]Seoul National University, [2]University of Illinois at Urbana-Champaign
https://github.com/SegwangKim/neural-seq2grid-module

Summary

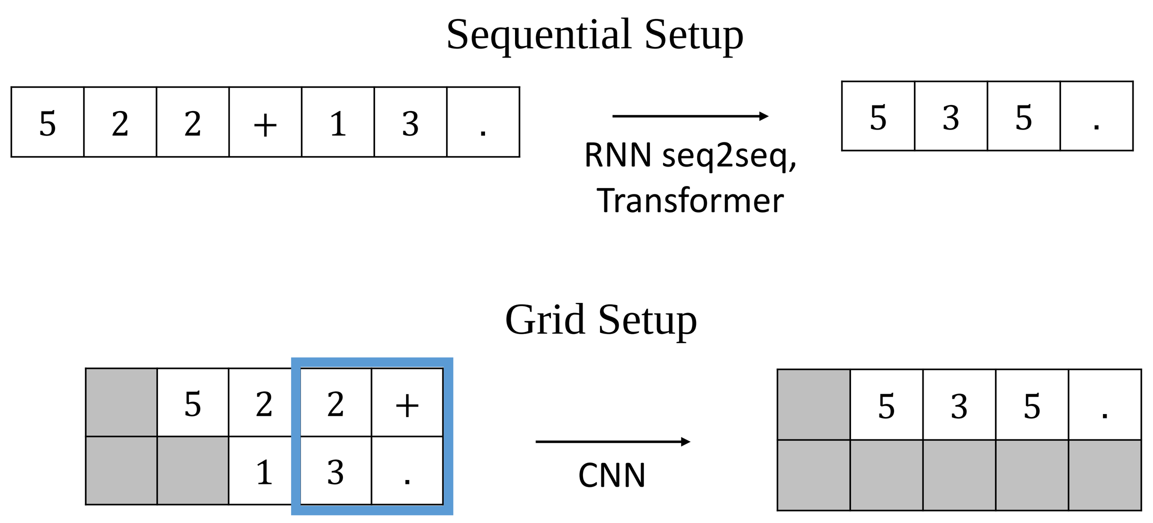
We propose a **neural sequence-to-grid module** that enhances a neural network **to learn symbolic rules.** Specifically,

- The module is an input preprocessor that **automatically aligns a sequential input into a grid**.
- The module works differentially, which ensures the **joint training** of the module and the neural network without any supervision for the alignment.

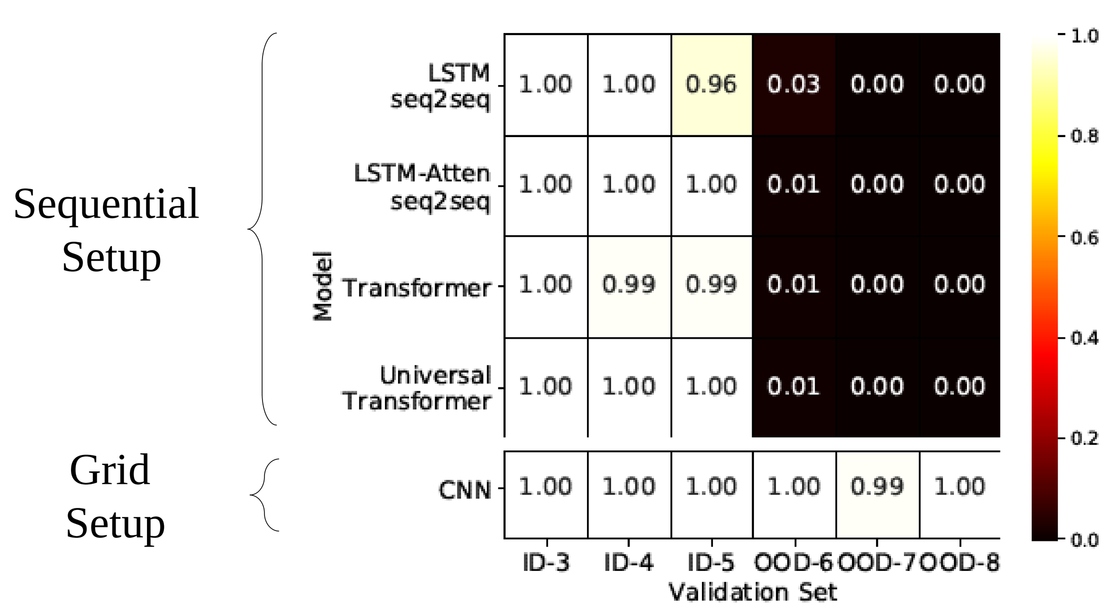
Empirically, our models outperform the standard seq2seq models at arithmetic and algorithmic problems such as program code evaluation tasks and the bAbI QA tasks.

Introduction

Symbolic reasoning problems, e.g., program code evaluation tasks or the bAbI tasks, are tasks to learn deterministic symbolic rules. Hence, they are useful testbeds for assessing the logical inference abilities of deep learning models. Whereas humans can extend the learned rules on out-of-distribution (OOD) data, it has been found that the standard seq2seq models cannot systematically generalize on OOD data. To address this difficulty, **our idea is to align an input sequence into a grid.** Our idea can be described by the following toy addition problem in two different setups.



We trained neural networks on additions of short-digit numbers and test them on additions of long-digit numbers. As a result, only the grid setup leads to OOD generalization.

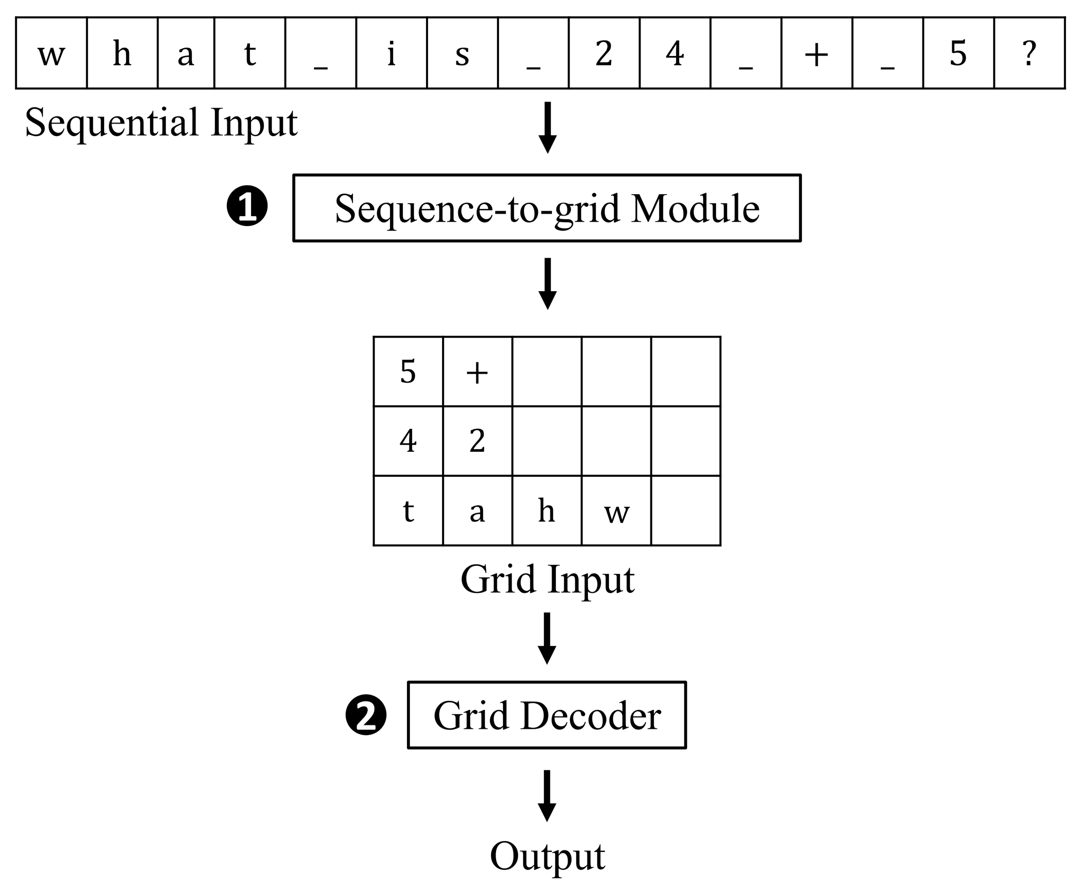


However, **we cannot formulate most of the symbolic problems in such a grid setup.**

Method

- To implement our **sequence-to-grid** method, we suggest the sequence-input grid-output architecture that decouples symbolic reasoning into two steps:

❶ automatically aligning an input sequence into a grid.
❷ doing semantic computations over the grid.

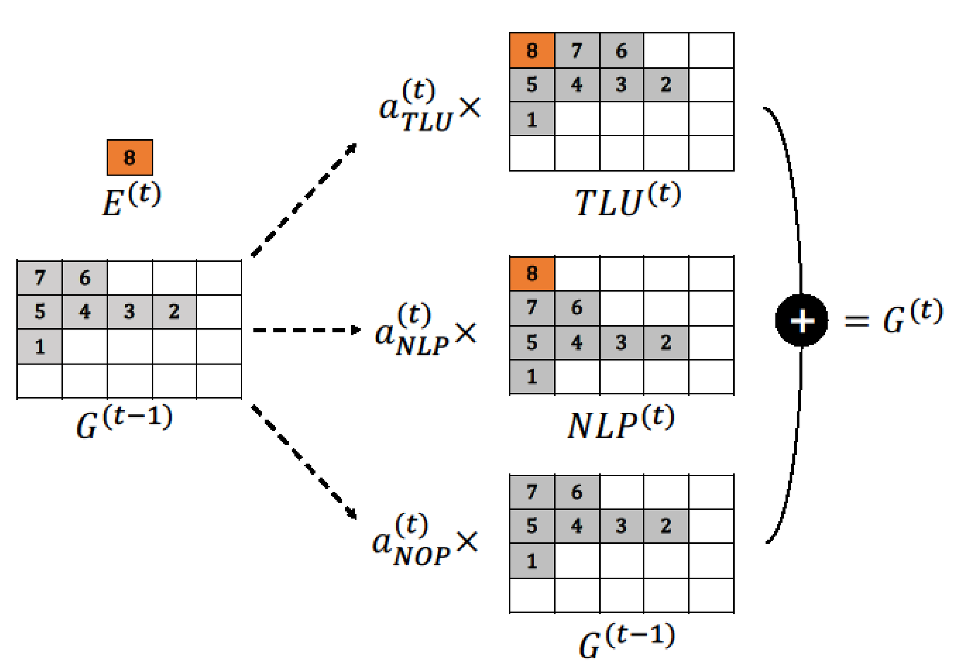


❶ **The sequence-to-grid module** encodes input sequence to the action sequence

$$(E^{(1)}, \dots, E^{(T)}) \xrightarrow[\text{+ softmax layer}]{\text{RNN encoder}} (a^{(1)}, \dots, a^{(T)})$$

token embeddings $E^{(t)} \in \mathbb{R}^h$ — action sequence $a^{(t)} = (a^{(t)}_{TLU}, a^{(t)}_{NLP}, a^{(t)}_{NOP})$

and outputs a grid input from the zero-initialized grid $G^{(0)} \in (\mathbb{R}^h)^{H \times W}$ via differentiable nested list evolution.



❷ Depending on tasks, we choose **the grid decoder** that can process the grid, e.g., ResNet, TextCNN.

- Our module and the grid decoder can **be jointly trained** in an end-to-end fashion via a backpropagation.
- Here, **no supervision for the alignment is required.**

Applications

Arithmetic and Algorithmic Problems

Number sequence prediction problem (Sequence)

Input	`7008 -205 4 7221.`
Target	`14233.`

Algebraic word problem (Add-or-sub)

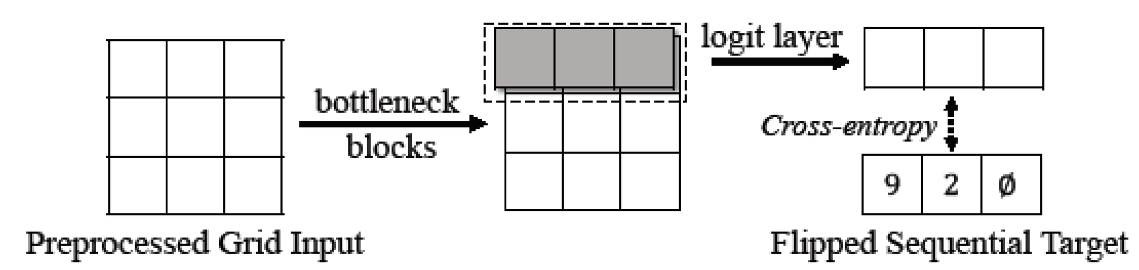
Input	`Sum -3240245475 and 11.`
Target	`368.`

Computer program evaluation problem (Program)

Input	`j=891`
	`for x in range(11):j-=878`
	`print((368 if 821<874 else j)).`

Target	`368.`

- The above problems ask to answer digit numbers.
- We train and test models on each task separately.
- The ID test set follows the training distribution.
- The OOD test set has examples containing unseen long-digit numbers.
- Tokenize the example by characters and decimal digits.
- Grid decoder: bottleneck blocks of ResNet.



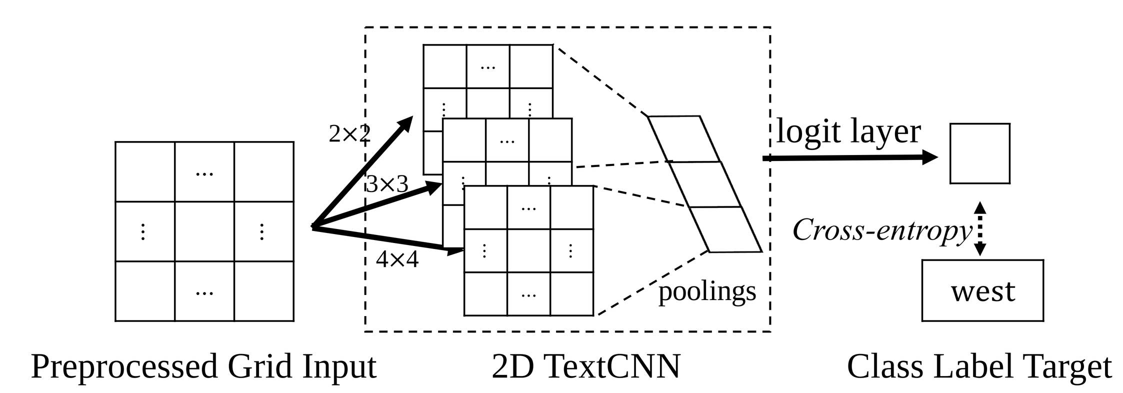
- To flip predictions in inference, the cross-entropy loss for the paddings must be considered during training.

The bAbI Tasks

Task 19. path-finding

⟨CLS⟩ How do you go from the garden to the office ? ⟨SEP⟩ The kitchen is west of the office . The office is north of the hallway . The garden is east of the bathroom . The garden is south of the hallway . The bedroom is east of the hallway .

- The 10k joint bAbI task is to learn 20 tasks at once, where each task asks to learn specific logic, such as induction or path-finding, given as natural language.
- Tokenize the input (question + story) by words.
- Grid decoder: a 2D version of TextCNN.



Results

Arithmetic and Algorithmic Problems

	Sequence		Add-or-sub		Program	
	ID	OOD	ID	OOD	ID	OOD
Baselines						
LSTM	0.21	0.00	0.99	0.00	0.25	0.07
LSTM-Atten	0.68	0.00	**1.00**	0.00	0.37	0.01
RMC	0.01	0.00	0.99	0.00	0.33	0.01
Transformer	0.97	0.00	0.97	0.00	0.37	0.00
UT	**1.00**	0.00	**1.00**	0.00	**0.62**	0.00
Ours						
S2G-CNN	0.96	**0.99**	0.98	0.53	0.51	0.33
S2G-ACNN	0.90	0.92	0.96	**0.55**	0.44	**0.35**

- Report the best sequence-level accuracy.
- In the number sequence prediction problem, **our module aligns numbers by digit scales**:

- In the computer program evaluation problem, our model successfully extends rules of IF-ELSE instructions on OOD examples:

	instruction	ID	OOD
LSTM-Atten	`IF-ELSE`	0.46	0.26
	`FOR`	0.06	0.03
	⋆	0.07	0.04
UT	`IF-ELSE`	**0.81**	0.01
	`FOR`	**0.38**	0.00
	⋆	**0.52**	0.00
S2G-CNN	`IF-ELSE`	0.73	**0.57**
	`FOR`	0.20	0.09
	⋆	0.25	0.14

The bAbI Tasks

	#params	Error	#Failed tasks
Baselines[5]			
LSTM	25.6M	24.9 ± 5.8	12.1 ± 3.7
Transformer	0.5M	33.1 ± 1.7	18.9 ± 0.3
UT	0.5M	26.8 ± 6.0	15.0 ± 4.0
TextCNN	0.2M	37.8 ± 0.4	19.0 ± 0.0
Ours			
S2G-TextCNN	0.8M	10.8 ± 0.8	6.0 ± 0.0

- Report errors and #Failed tasks (>5% error).
- Without a complex and expensive memory, our module can compress long inputs into the fixed-size grid.

This work was supported by Samsung Research Funding & Incubation Caenter of Samsung Electronics.