Neural Sequence-to-grid Module for Learning Symbolic Rules

Segwang Kim[1], Hyoungwook Nam[2], Joonyoung Kim[1], Kyomin Jung[1]

[1] Seoul National University, Seoul, Korea

[2] University of Illinois at Urbana-Champaign, Urbana, Illinois, USA

AAAI
Association for the Advancement of Artificial Intelligence

SEOUL NATIONAL UNIVERSITY

Background

- **Symbolic reasoning problems** are testbeds for assessing logical inference abilities of deep learning models.

Program code evaluation [1]

Input:
```
j=8584
for x in range(8):
  j+=920
b=(1500+j)
print((b+7567))
```
Target: 25011.

bAbI tasks [2]

Task 2: Two Supporting Facts
John is in the playground.
John picked up the football.
Bob went to the kitchen.
Where is the football? A:playground

[1] Zaremba, et al. Learning to execute. *arXiv 2014*.
[2] Weston, et al. Towards ai-complete question answering: A set of prerequisite toy tasks. ICLR 2016.

Background

- **Symbolic reasoning problems** are testbeds for assessing logical inference abilities of deep learning models.

Program code evaluation [1]

Input:
```
j=8584
for x in range(8):
  j+=920
b=(1500+j)
print((b+7567))
```
Target: 25011.

bAbI tasks [2]

Task 2: Two Supporting Facts
John is in the playground.
John picked up the football.
Bob went to the kitchen.
Where is the football? A:playground

- The determinism of symbolic problems allows us to systematically test deep learning models with **out-of-distribution (OOD)** data.

Training examples

5	8	2	+	1	=

6	7	+	3	=

OOD Test Examples

3	0	5	3	4	+	4	2	1	=

6	9	5	2	1	+	5	0	0	2	9	=

  - Humans with algebraic mind can naturally extend learned rules.

[1] Zaremba, et al. Learning to execute. *arXiv 2014*.
[2] Weston, et al. Towards ai-complete question answering: A set of prerequisite toy tasks. ICLR 2016.

Background

- However, deep learning models cannot extend learned rules to OOD (out-of-distribution) examples.

Number sequence prediction problems [3]

Tasks	Reverse-order (training)	Geometric	Arithmetic	Fibonacci
LSTM	28.4% (1.2%)	79.4%	77.1%	80.5%
GRU	51.9% (0.9%)	69.0%	77.1%	79.3%
Attention(unidirectional)	42.0% (8.8%)	62.8%	77.0%	69.3%
Attention(bidirectional)	0.0% (0.0%)	51.0%	72.9%	60.9%
Stack-RNN	**0.0%** (0.0%)	64.1%	63.8%	69.4%
NTM	**0.0%** (0.0%)	57.1%	65.7%	68.1%

Error

Middle school level mathematics problems [4]

	Parameters	**Interpolation**	**Extrapolation**
Simple **LSTM**	18M	0.57	0.41
Simple **RMC**	38M	0.53	0.38
Attentional **LSTM**, LSTM encoder	24M	0.57	0.38
Attentional **LSTM**, bidir LSTM encoder	26M	0.58	0.42
Attentional **RMC**, bidir LSTM encoder	39M	0.54	0.43
Transformer	30M	**0.76**	**0.50**

Accuracy

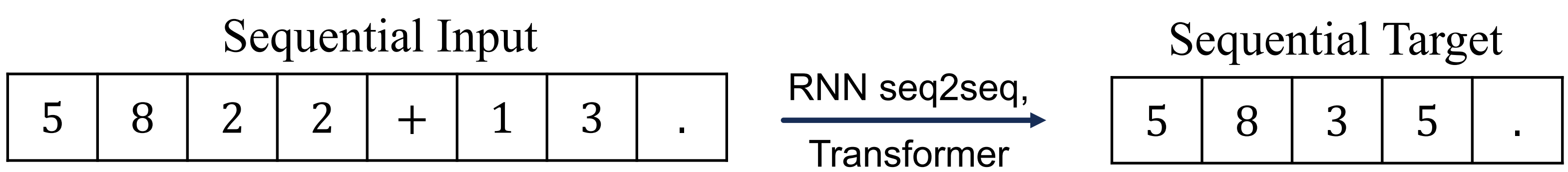
[3] Nam, et al. Number sequence prediction problems and computational power of neural networks, AAAI 2019
[4] Saxton, et al. Analysing Mathmatical Reasoning Abilities of Neural Models, ICLR 2019.

Motivation

- Idea: **if we align an input sequence into a grid, learning symbolic rules becomes easier.**

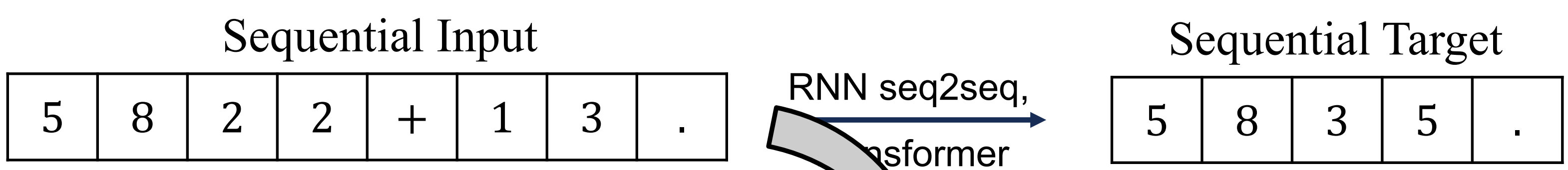
Motivation

- Idea: **if we align an input sequence into a grid, learning symbolic rules becomes easier.**
- Consider a toy decimal addition problems in two different setups:
  - Sequential setup

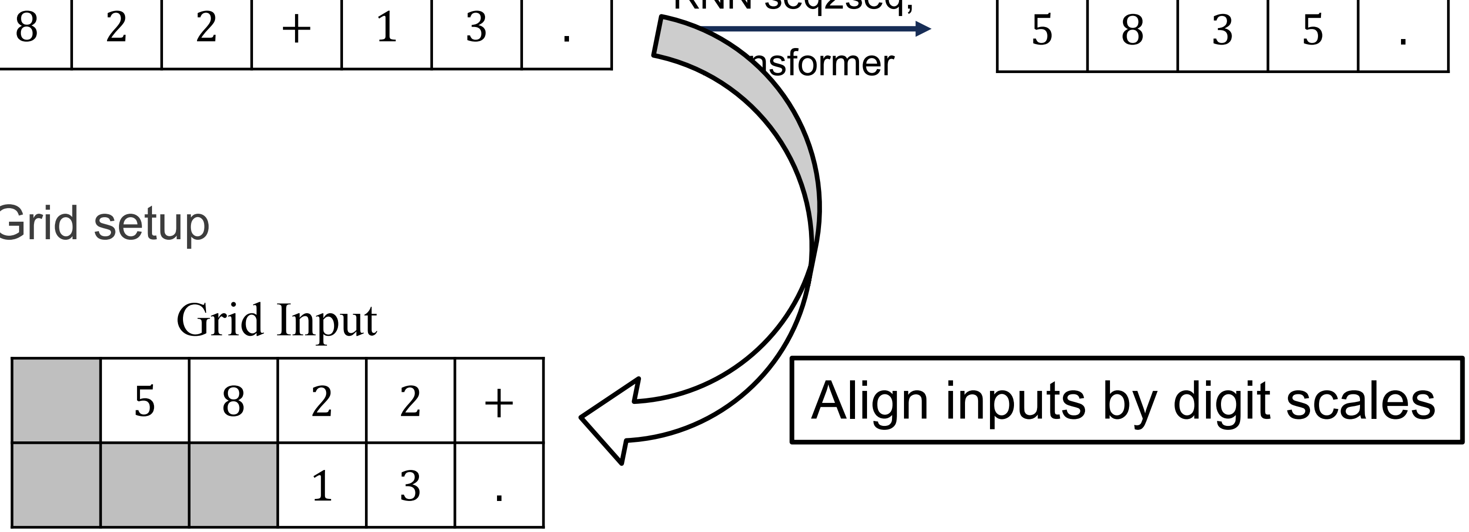


Motivation

- Idea: **if we align an input sequence into a grid, learning symbolic rules becomes easier.**
- Consider a toy decimal addition problems in two different setups:
  - Sequential setup



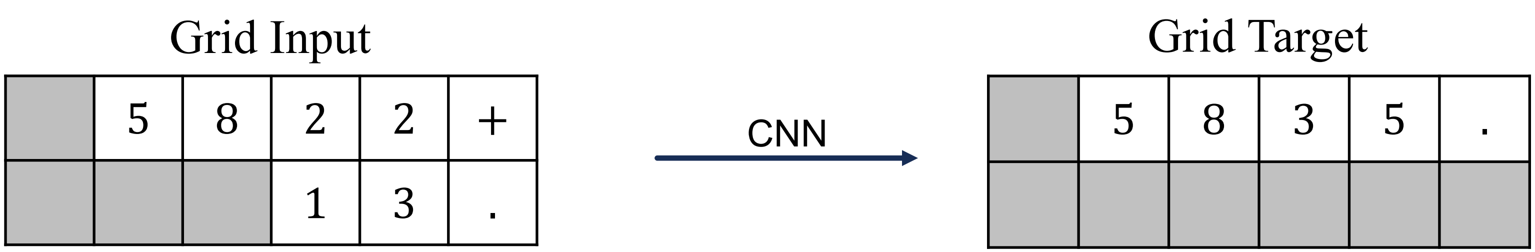
  - Grid setup



Motivation

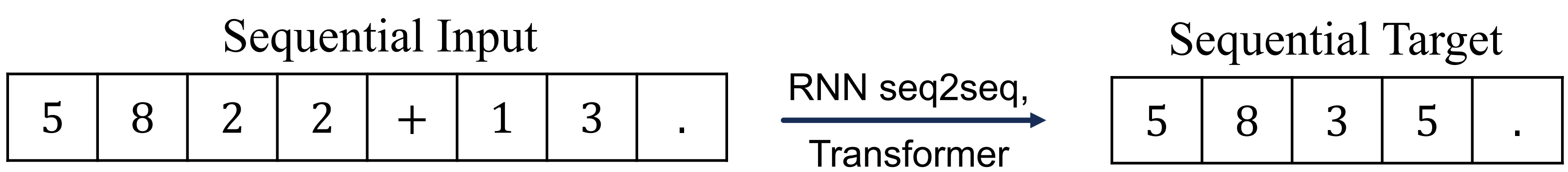
- Idea: **if we align an input sequence into a grid, learning symbolic rules becomes easier.**
- Consider a toy decimal addition problems in two different setups:
  - Sequential setup

  - Grid setup

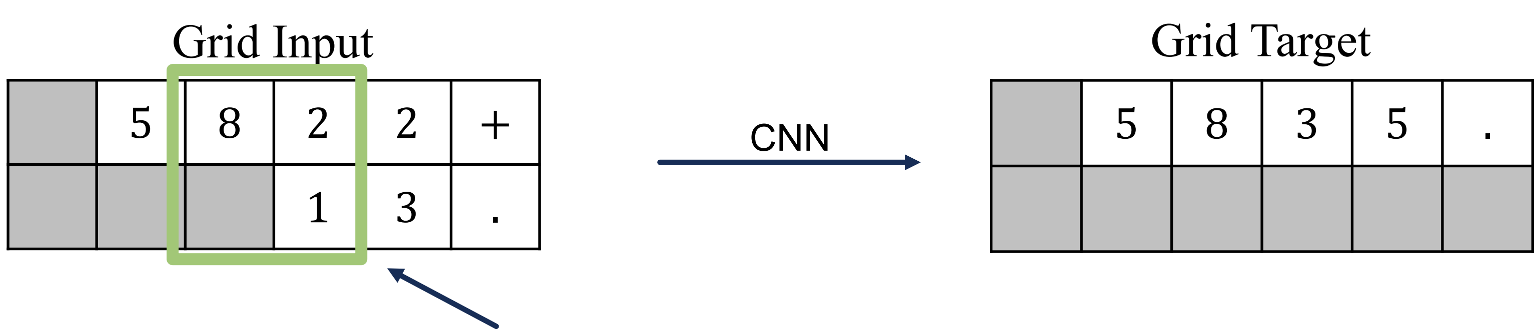


Motivation

- Idea: **if we align an input sequence into a grid, learning symbolic rules becomes easier.**
- Consider a toy decimal addition problems in two different setups:
  - Sequential setup



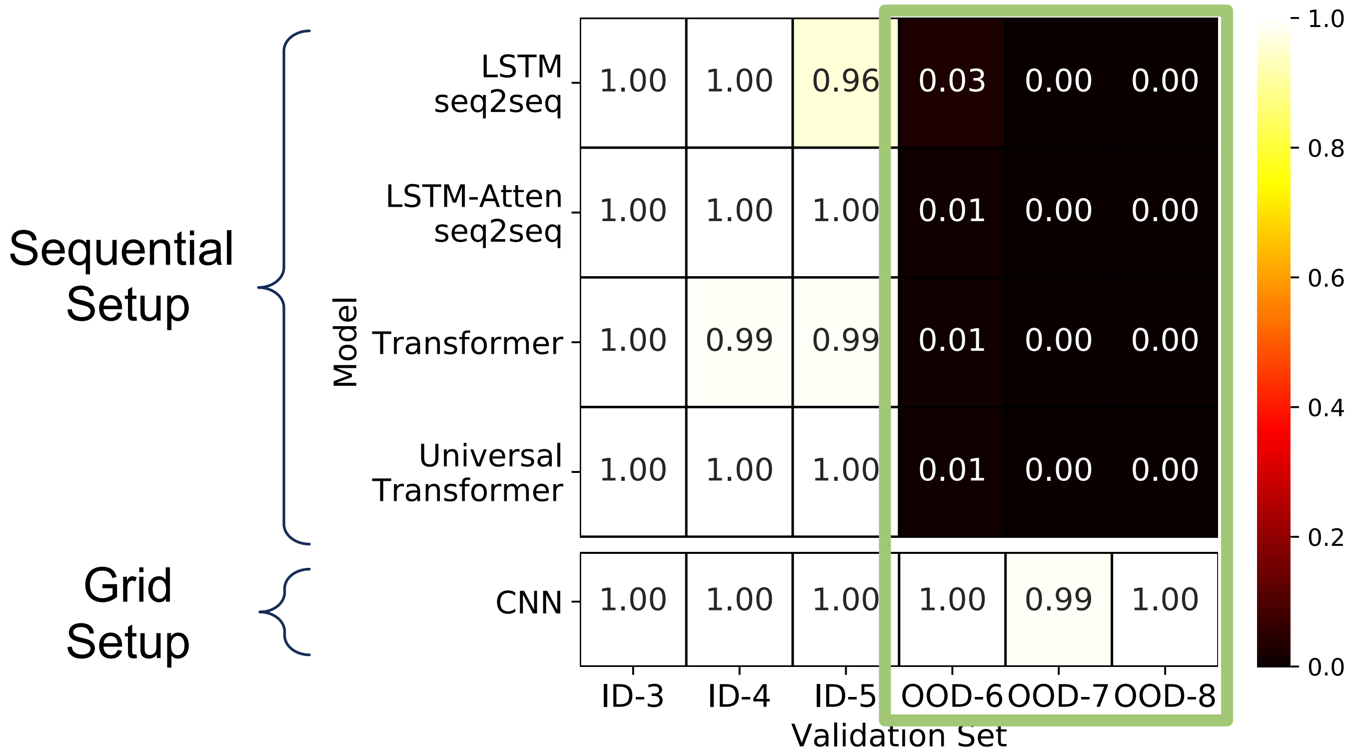
  - Grid setup



The convolution kernel can learn the addition rule, i.e., **inductive bias**.

Usefulness of Aligned Grid Inputs

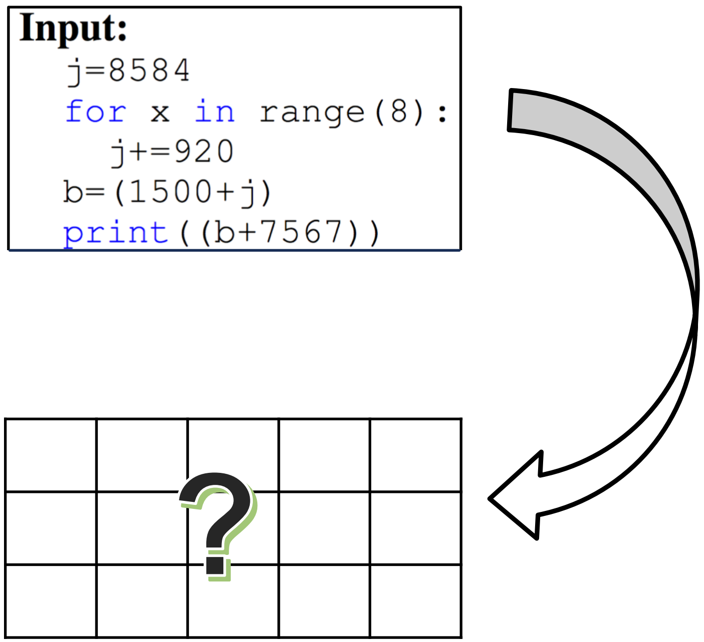
- Depending on setups, OOD generalization is achieved or not.

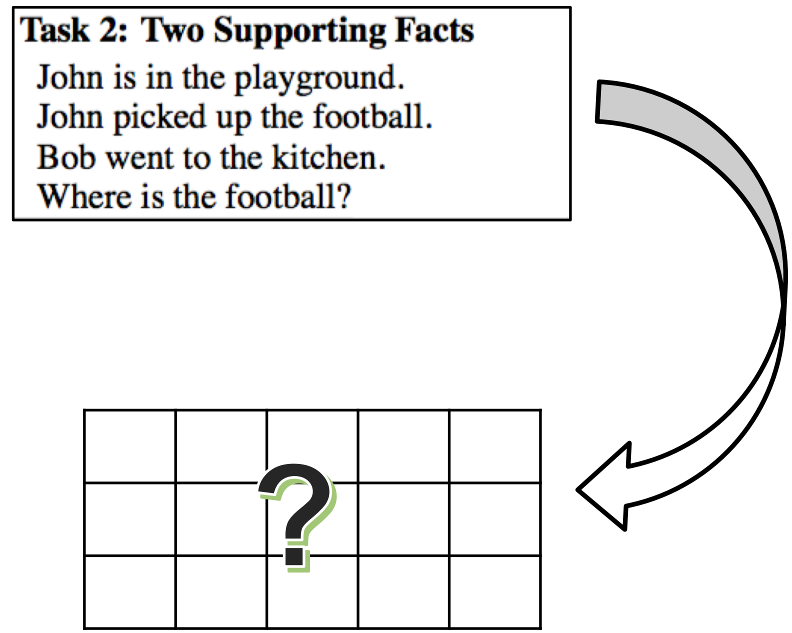


- **Providing aligned grid inputs for CNN can be key to extend symbolic rules.**

Motivation

- However, most of symbolic problems cannot be formulated in such grid setup.

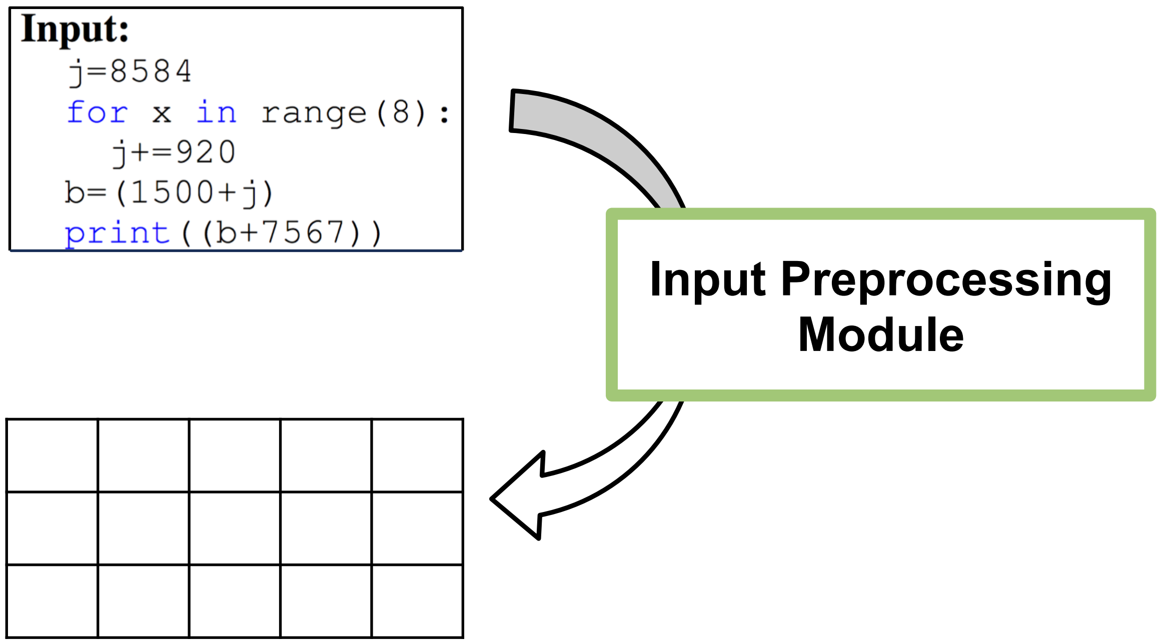




- How to align programming instructions?
- How to align words?

Research Goal

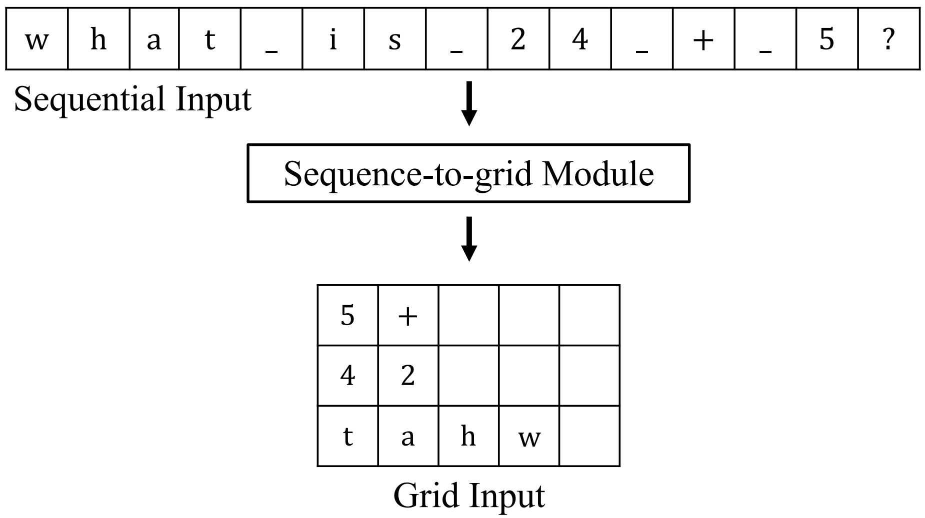
- Therefore, we need a new **input preprocessing module**.



- The module must **automatically align an sequence into a grid without supervision for the alignment**.

Our Method

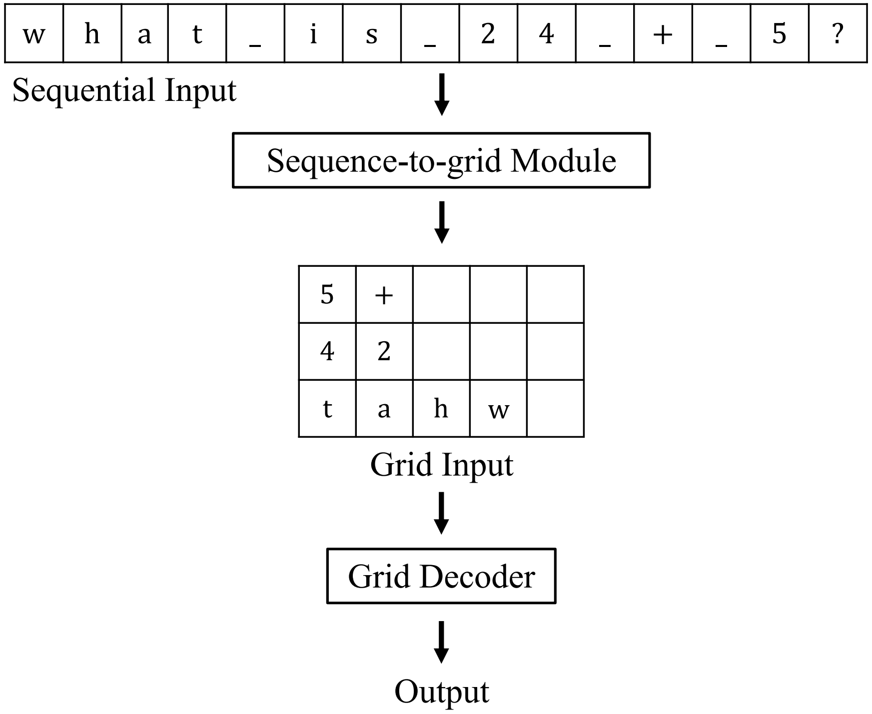
- We propose a **neural sequence-to-grid (seq2grid) module**.
  - an input **preprocessor.**
  - It **learns how to segment and align** an input sequence into a grid.



- The preprocessing is done via our novel **differentiable mapping.**
  - It ensures a **joint training** of our module and the neural network **in an end-to-end fashion** via a backpropagation.

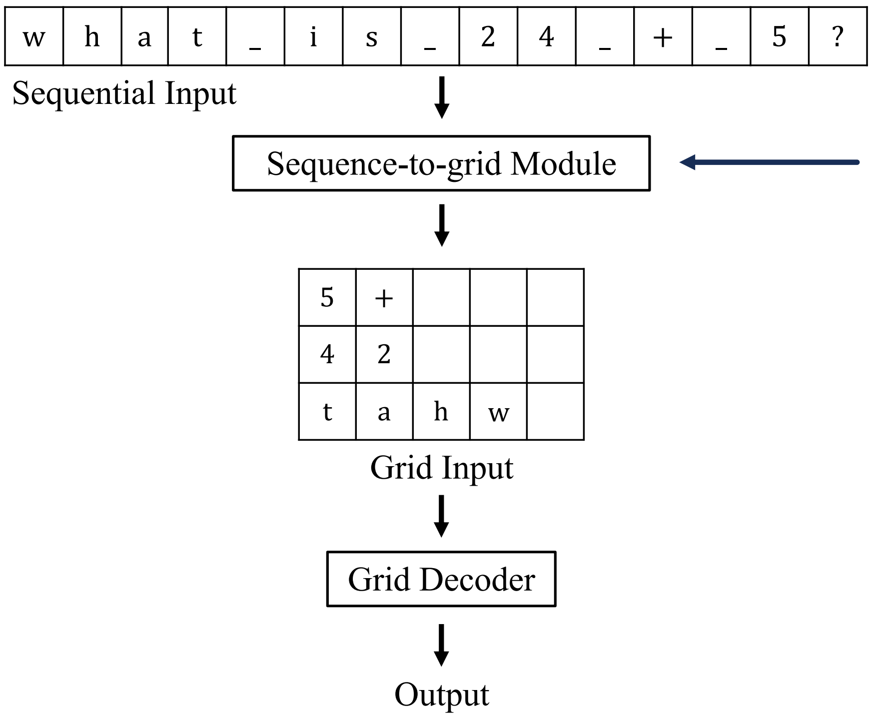
Method: Sequence-input grid-output Architecture

- First, we propose the **sequence-input grid-output architecture.**



Method: Sequence-input grid-output Architecture

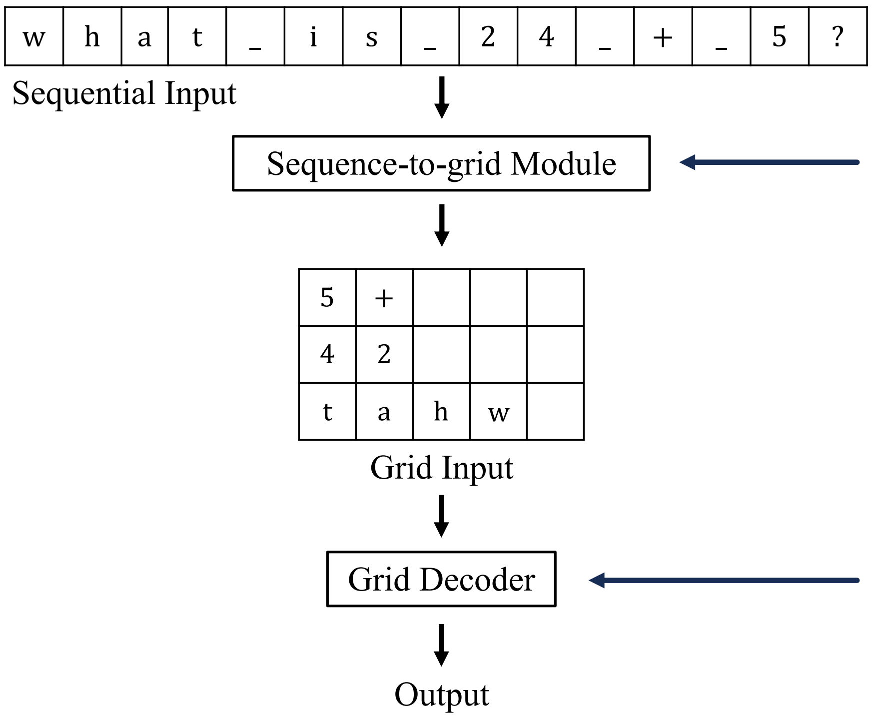
- First, we propose the **sequence-input grid-output architecture.**



1. **aligning** an input sequence into a grid automatically

Method: Sequence-input grid-output Architecture

- First, we propose the **sequence-input grid-output architecture.**



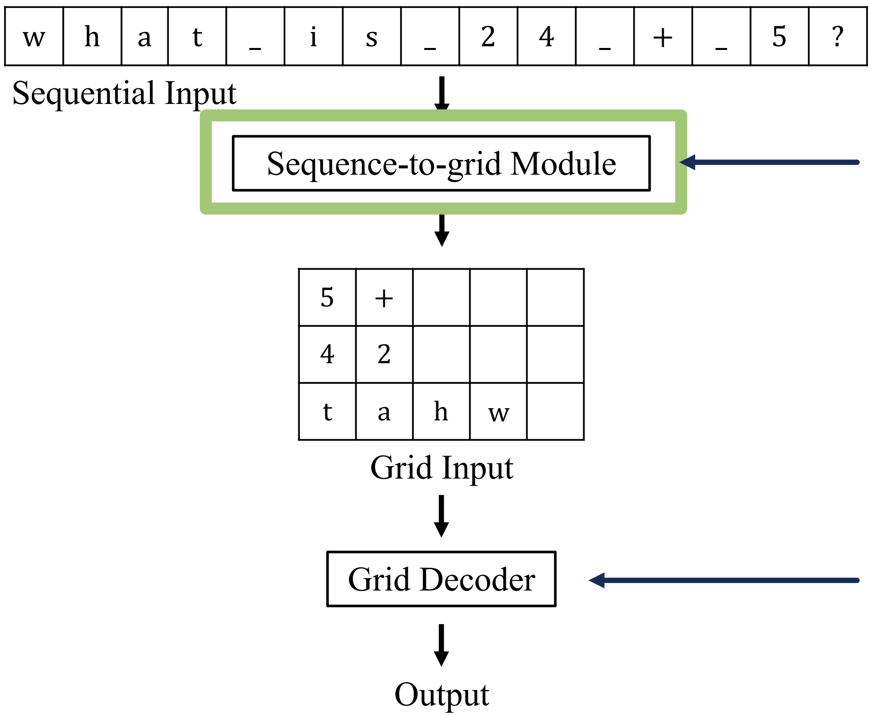
1. **aligning** an input sequence into a grid automatically

2. **semantic computations** over the grid.

Any neural network that can get the grid is possible
e.g., ResNet, TextCNN,

Method: Sequence-input grid-output Architecture

- First, we propose the **sequence-input grid-output architecture.**



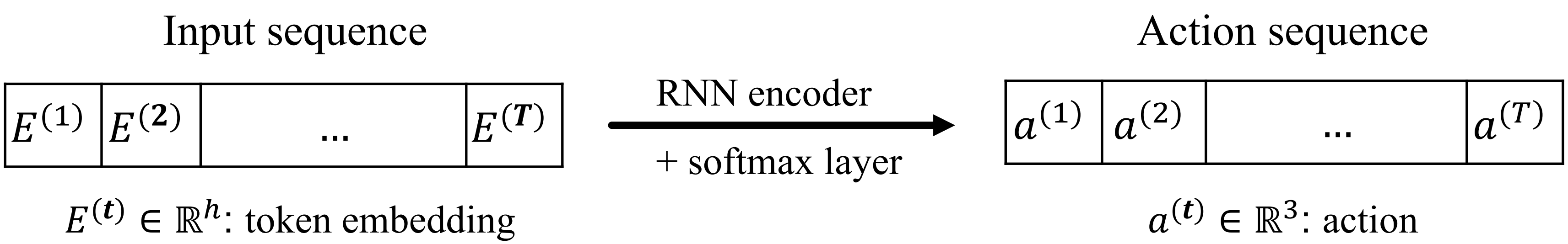
1. aligning an input sequence into a grid automatically

2. semantic computations over the grid.

Any neural network that can get the grid is possible
e.g., ResNet, TextCNN,

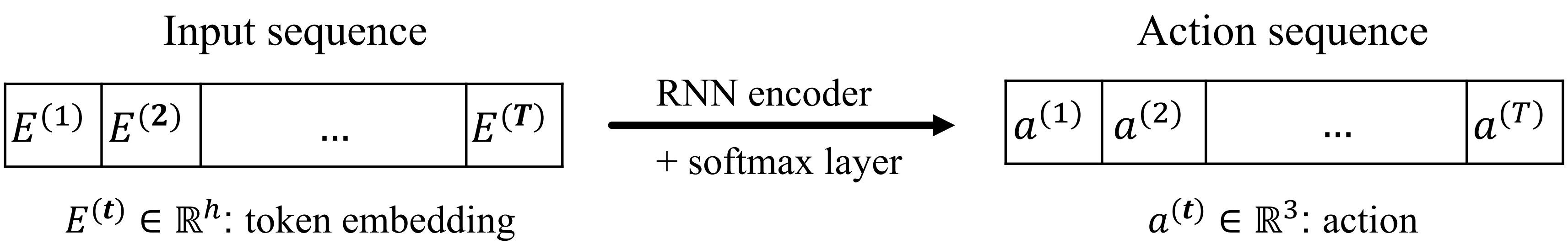
Method: Neural Sequence-to-grid Module

- The sequence-to-grid module does following:
  - 1. Encodes input sequence to the **action** sequence.

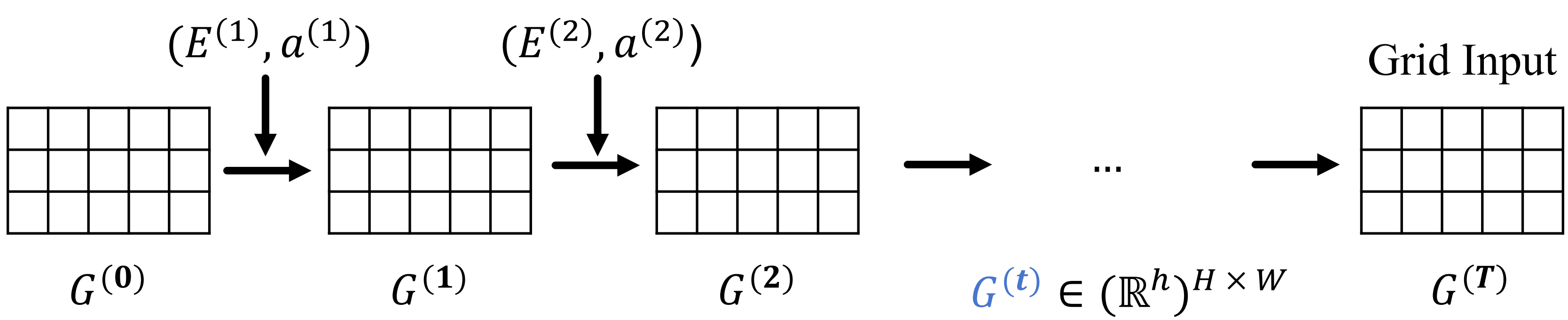


Method: Neural Sequence-to-grid Module

- The sequence-to-grid module does following:
  - 1. Encodes input sequence to the **action** sequence.



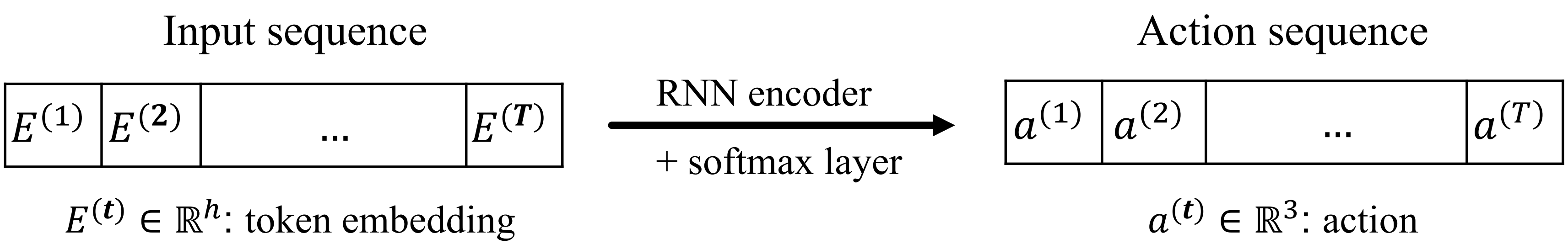
  - 2. Outputs a grid input from the zero-initialized **grid** $G^{(0)}$ via **nest list evolutions.**



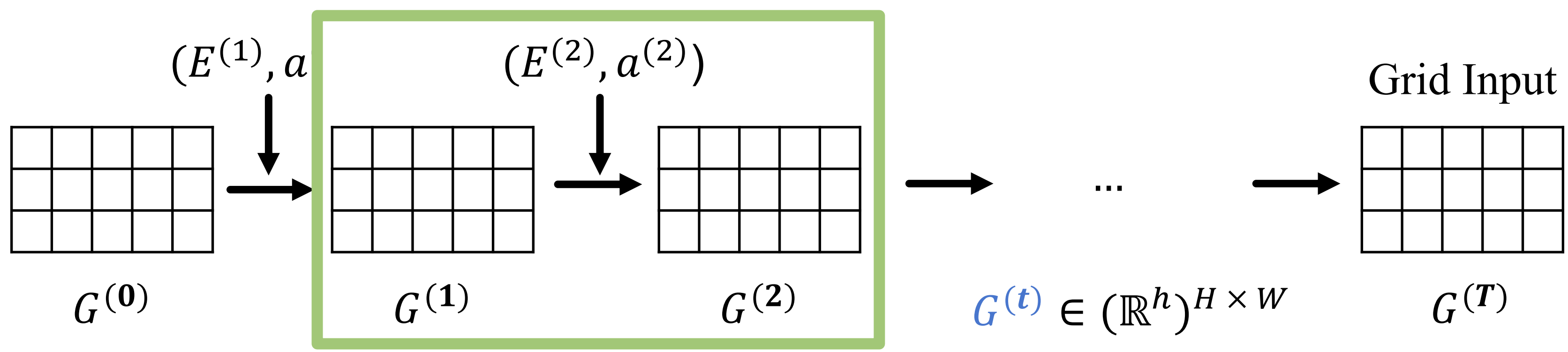
Grid of the size H-by-W

Method: Neural Sequence-to-grid Module

- The sequence-to-grid module does following:
  - 1. Encodes input sequence to the **action** sequence.



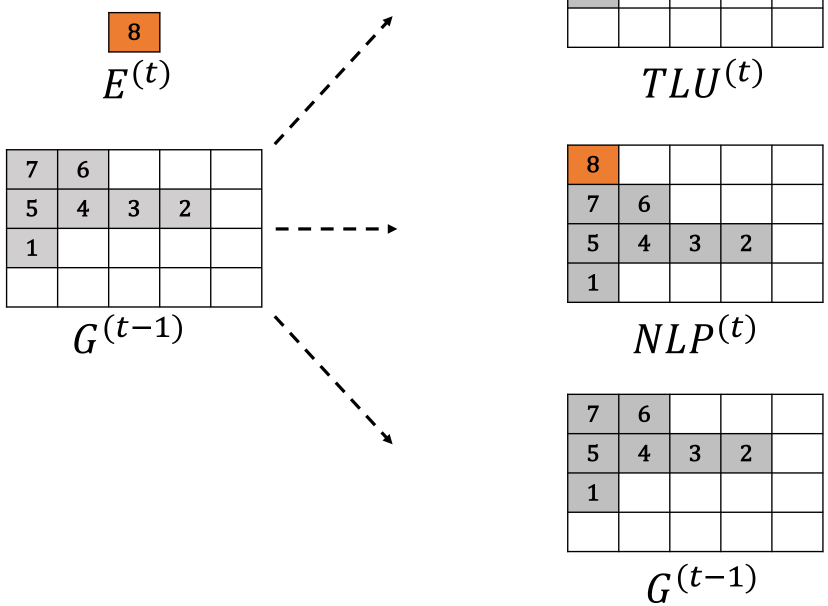
  - 2. Outputs a grid input from the zero-initialized **grid** $G^{(0)}$ via **nest list evolutions.**



Method: Nest List Evolution

Candidates

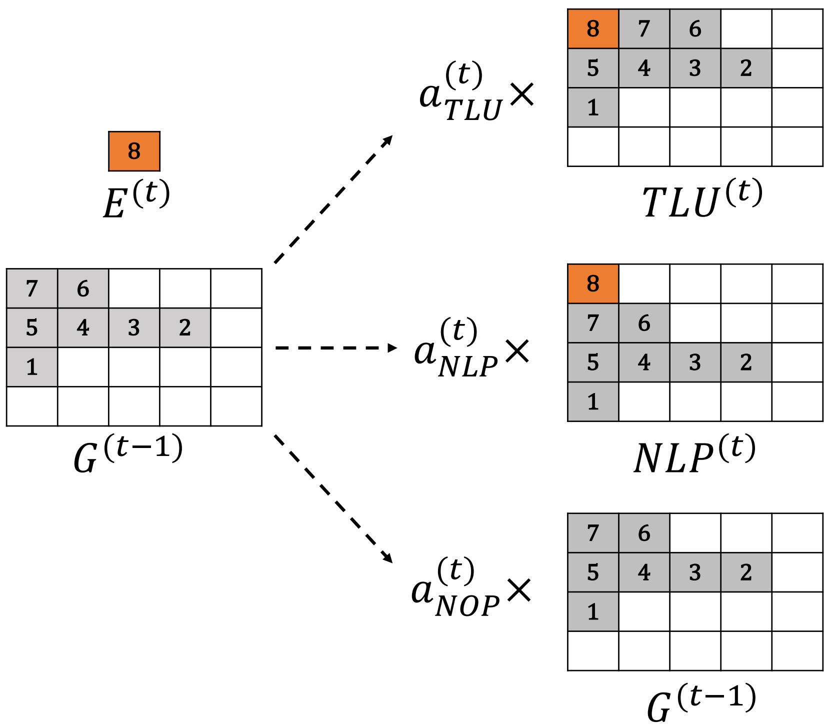
nested list operations



- Top-list-update (TLU)
- New-list-push (NLP)
- No-op (NOP)

Method: Nest List Evolution

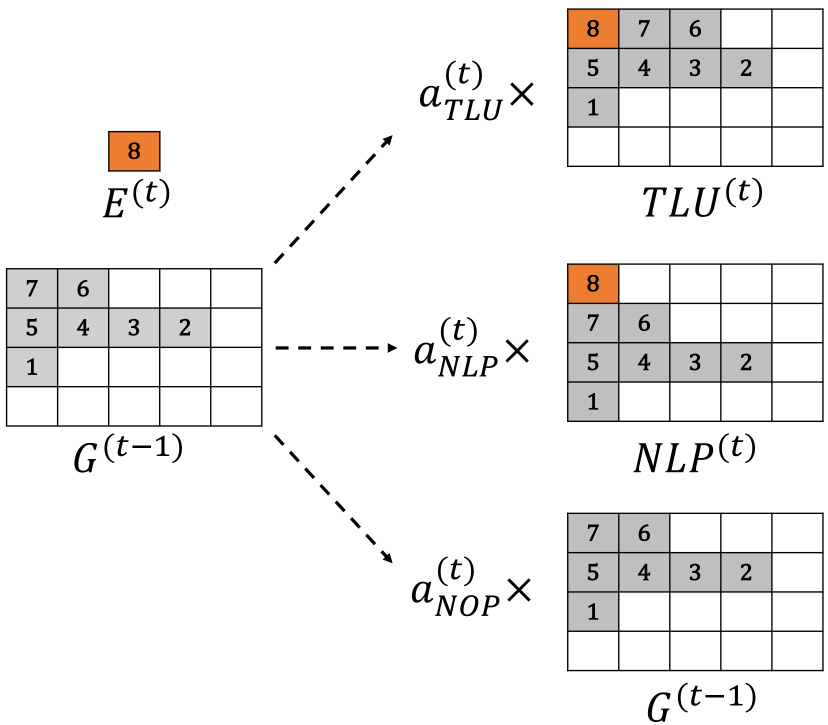
- Each component of an action $a^{(t)} = (a^{(t)}_{TLU}, a^{(t)}_{NLP}, a^{(t)}_{NOP})$ is the **probability** of performing one of three nested list operations:



- Top-list-update (TLU)
- New-list-push (NLP)
- No-op (NOP)

Method: Nest List Evolution

- Each component of an action $a^{(t)} = (a^{(t)}_{TLU}, a^{(t)}_{NLP}, a^{(t)}_{NOP})$ is the **probability** of performing one of three nested list operations:



- Top-list-update (TLU)
- New-list-push (NLP)
- No-op (NOP)

- In each evolution step, $G^{(t-1)}$ with $\left(E^{(t)}, a^{(t)}\right)$ grows to $G^{(t)}$

$$G^{(t)} = a^{(t)}_{TLU} \cdot TLU^{(t)} + a^{(t)}_{NLP} \cdot TLU^{(t)} + a^{(t)}_{NOP} \cdot G^{(t-1)}$$

Arithmetic and Algorithmic Problems

- We test our module on three arithmetic and algorithmic problems.

Number sequence prediction problem

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

Algebraic word problem

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

Computer program evaluation problem

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

Target	`368.`

  - 1M training examples.
  - Tokenize all examples by characters and decimal digits.

Arithmetic and Algorithmic Problems

- We test our module on three arithmetic and algorithmic problems.

Number sequence prediction problem

Input	7008 -205 4 7221.
Target	14233.

Algebraic word problem

Input	Sum -3240245475 and 11.
Target	368.

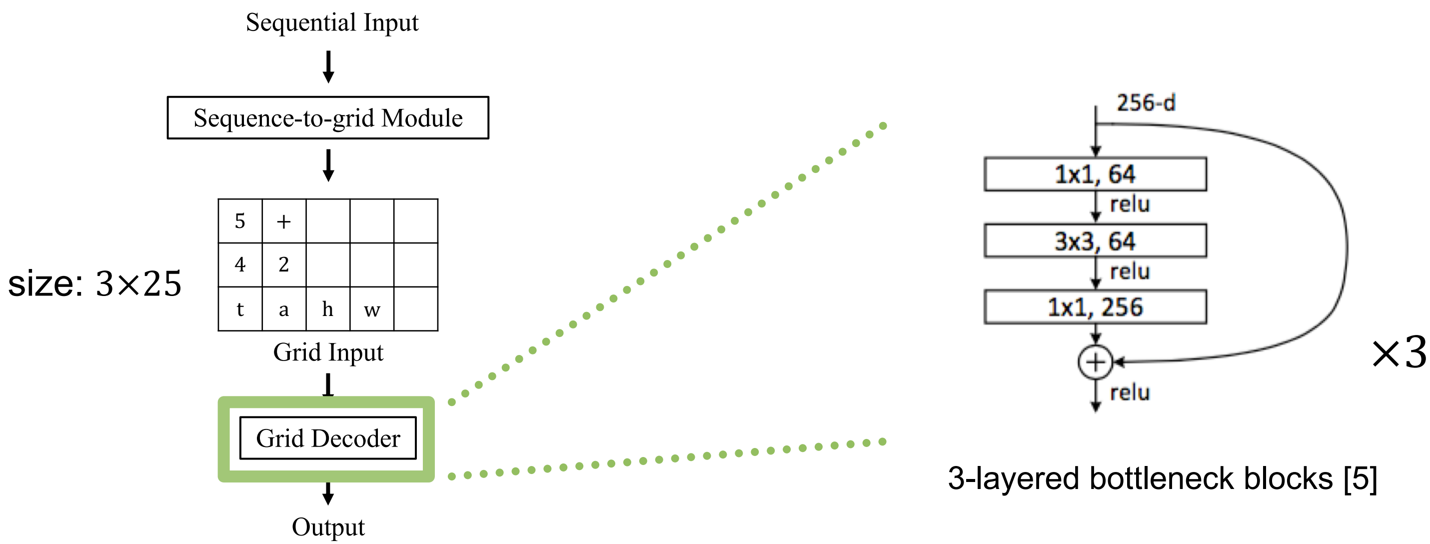
Computer program evaluation problem

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

  - 1M training examples.
  - Tokenize all examples by characters and decimal digits.
- Two test sets (10k test examples each).
  - In-distribution (ID): examples sampled from the training distribution.
  - Out-of-distribution (OOD): examples with unprecedented longer digits.

Arithmetic and Algorithmic Problems

- Grid decoder: three stacks of 3-layered bottleneck blocks of ResNet.

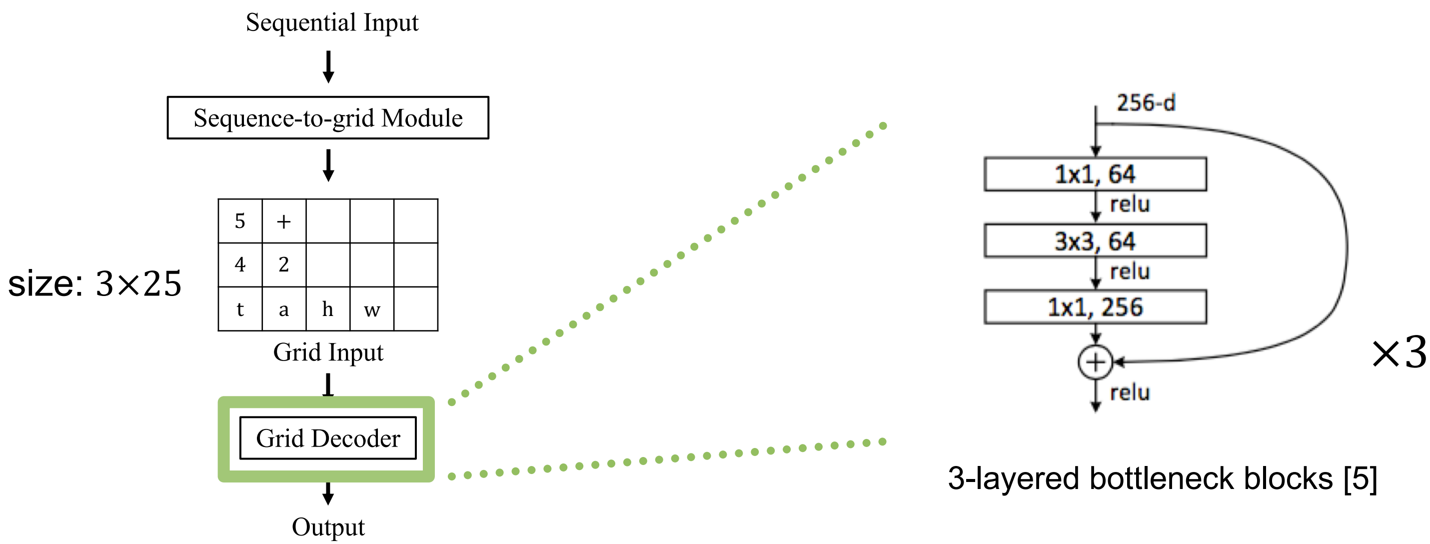


3-layered bottleneck blocks [5]

[5] He, et al. Deep Residual Learning for Image Recognition, CVPR 2015

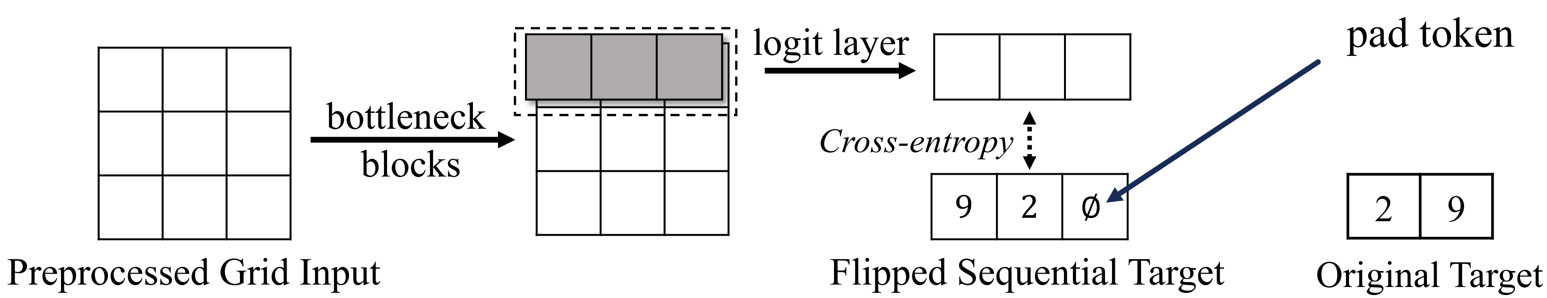
Arithmetic and Algorithmic Problems

- Grid decoder: three stacks of 3-layered bottleneck blocks of ResNet.



3-layered bottleneck blocks [5]

- The seq2grid module and the grid decoder are **simultaneously** trained by reducing cross-entropy loss.



[5] He, et al. Deep Residual Learning for Image Recognition, CVPR 2015

Results: Arithmetic and Algorithmic Problems

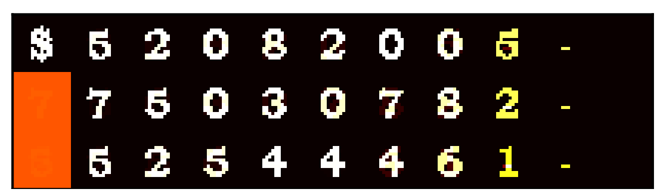
- On OOD test set, our models outperform baselines by large margin.

	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**

Table 1: Best sequence-level accuracy (out of 5 runs) on number sequence prediction problems (sequence), algebraic word problems (Add-or-sub), and computer program evaluation problems (Program)

- In number sequence prediction problem, our module **automatically aligns** numbers by digit scales.

… -16444525 -28703057 -50028025$ $\xrightarrow{\text{seq2grid module}}$



An Input example

Visualization of the grid input

Results: Arithmetic and Algorithmic Problems

- In computer program evaluation problem,
  - We investigate accuracy by instructions.

	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

```
print((11*7288719))
```

```
print(((6110039 if 7327755<3501784 else
1005398)*11))
```

```
b=6367476
for x in range(19):b-=9082877
print((3569363 if 7448172<9420320 else b))
```

```
e=(450693 if 4556818<2999168 else 3618338)
for x in range(10):e-=4489485
print(e)
```

OOD snippet examples

Results: Arithmetic and Algorithmic Problems

- In computer program evaluation problem,
  - We investigate accuracy by instructions.

	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	$\boxed{0.14}$

```
print((11*7288719))
```

```
print(((6110039 if 7327755<3501784 else
1005398)*11))
```

```
b=6367476
for x in range(19):b-=9082877
print((3569363 if 7448172<9420320 else b))
```

```
e=(450693 if 4556818<2999168 else 3618338)
for x in range(10):e-=4489485
print(e)
```

OOD snippet examples

Accuracy over snippets containing * instructions

Results: Arithmetic and Algorithmic Problems

- In computer program evaluation problem,
  - We investigate accuracy by instructions.

	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

```
print((11*7288719))
```

```
print(((6110039 if 7327755<3501784 else
1005398)*11))
```

```
b=6367476
for x in range(19):b-=9082877
print((3569363 if 7448172<9420320 else b))
```

```
e=(450693 if 4556818<2999168 else 3618338)
for x in range(10):e-=4489485
print(e)
```

OOD snippet examples

- 57% accuracy in snippets containing `if-else` is surprising.

Results: Arithmetic and Algorithmic Problems

- In computer program evaluation problem,
  - We investigate accuracy by instructions.

	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

```
print((11*7288719))
```

```
print(((6110039 if 7327755<3501784 else
1005398 * 1))
```

```
b=6367476
for x in range(19):b-=9082877
print((3569363 if 7448172<9420320 else b))
```

```
e=(450693 if 4556818<2999168 else 3618338)
for x in range(10):e-=4489485
print(e)
```

OOD snippet examples

- 57% accuracy in snippets containing `if-else` is surprising.
  - Since those snippets can contain other instructions as well.

bAbI QA Tasks

- We further test our module on bAbI QA tasks.

Task 2. two-supporting-facts

> ⟨CLS⟩ Where is the apple ? ⟨SEP⟩ Mary journeyed to the garden . Sandra got the football there . Mary picked up the apple there . Mary dropped the apple .

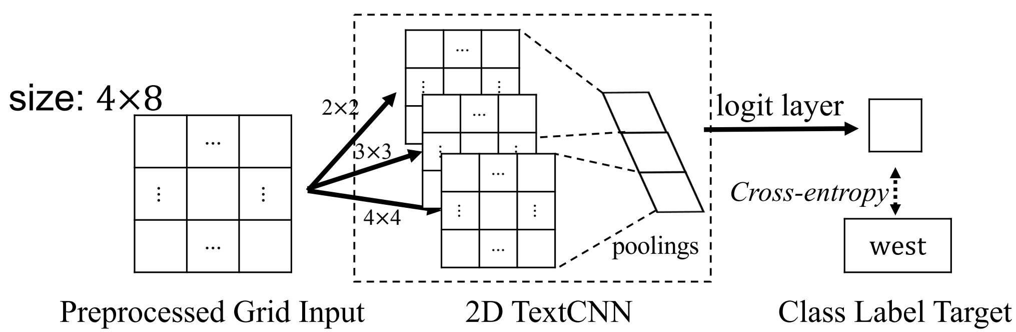
Task 17. basic-deduction

> ⟨CLS⟩ What is gertrude afraid of ? ⟨SEP⟩ Wolves are afraid of sheep . Gertrude is a wolf . Winona is a wolf . Sheep are afraid of mice . Mice are afraid of cats . Cats are afraid of sheep . Emily is a cat . Jessica is a wolf .

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 .

  - Training models on all tasks at once (10k joint tasks).
  - Tokenize the input (question + story) by words.
- Grid decoder: a 2D version of TextCNN.



Results: bAbI QA Tasks

- Our sequence-to-grid method **makes bAbI tasks easier.**

	#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

Table 3: Error and #Failed tasks ($> 5\%$ error) on the bAbI QA 10k joint tasks (for 10 runs).

- TextCNN fail at almost all tasks.

Results: bAbI QA Tasks

- Our sequence-to-grid method **makes bAbI tasks easier.**

	#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

Table 3: Error and #Failed tasks (> 5% error) on the bAbI QA 10k joint tasks (for 10 runs).

  - TextCNN fail at almost all tasks.
- Our module can **compress** long inputs into grid inputs.
  - 79 (average # of input tokens) > 32 (# of the grid slots)
  - Only necessary words along story arcs are selected.
- Our model **does not need** a complex and expensive **memory.**

Closing Remarks

- Our seq2grid module:
  - **Input preprocessor.**
  - It **automatically aligns** an sequential input into a grid.
  - During training, it requires **no supervision for the alignment.**
  - Its nest list operations ensure the **joint training** of the module and the grid decoder.
  - It **enhances neural networks** in various symbolic reasoning tasks.

- Code: https://github.com/segwangkim/neural-seq2grid-module

- Contact: ksk5693@snu.ac.kr