Machine Intelligence lab
SEOUL NATIONAL UNIVERSITY

NUMBER SEQUENCE PREDICTION PROBLEMS FOR EVALUATING COMPUTATIONAL POWERS OF NEURAL NETWORKS

Hyoungwook Nam
hwnam831@snu.ac.kr

Segwang Kim
ksk5693@snu.ac.kr

Kyomin Jung
kjung@snu.ac.kr

CONTENTS

MOTIVATION

Can neural networks learn Fibonacci sequence?

A question anybody can ask but **nobody had answered**

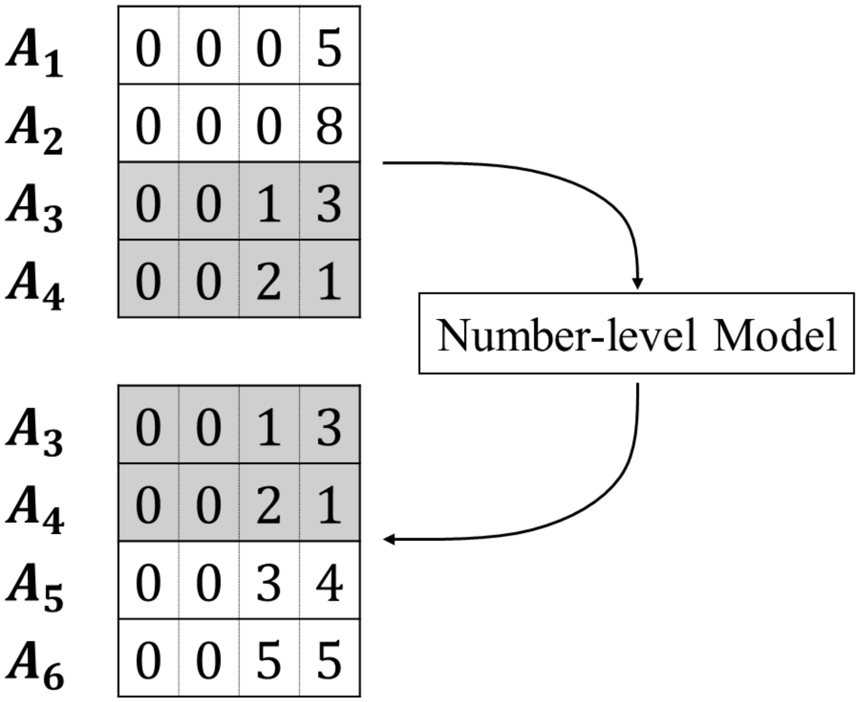
Quick test results: **CNNs** find it **easy**, but **RNNs** find it **hard**

The study is about **why** this observation happens

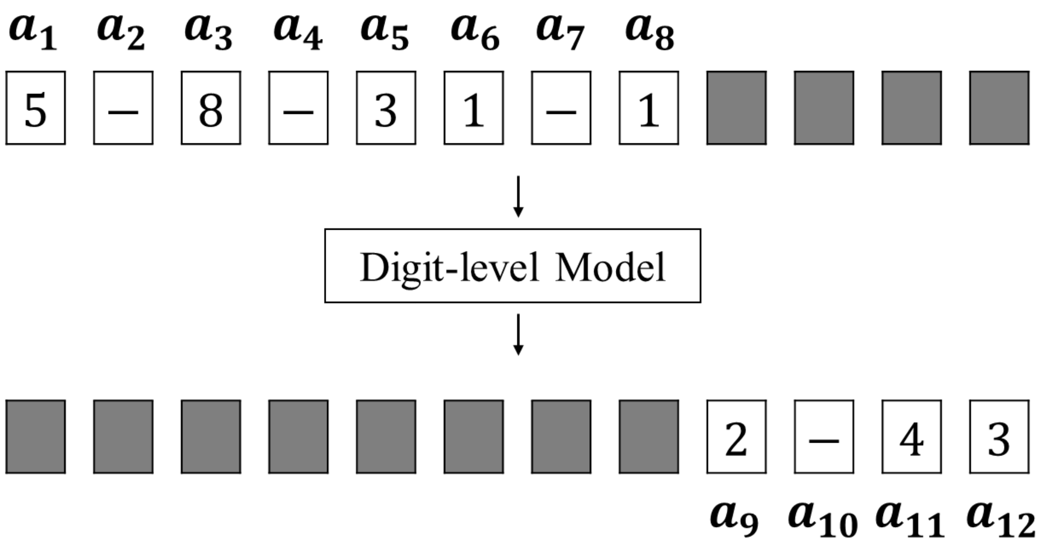
Basic idea: view CNNs as **combinatorial logic** and RNNs as **sequential state automata**

TWO TYPES OF THE PROBLEMS

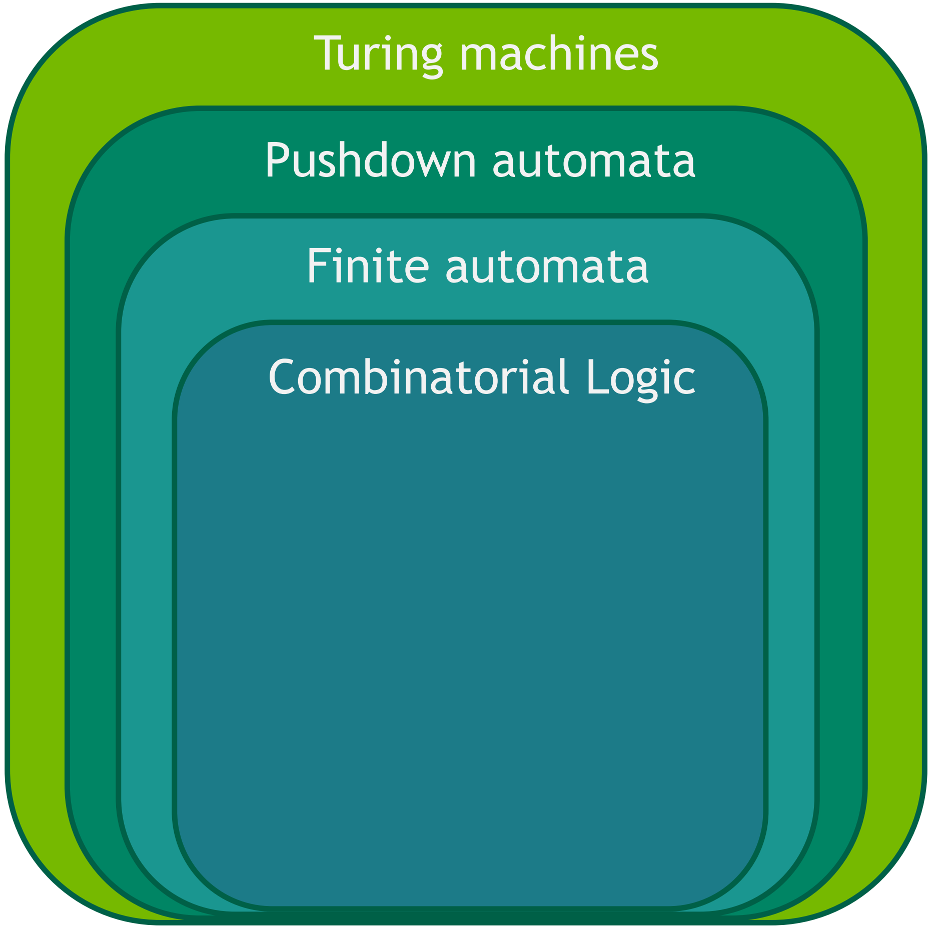
Number-level (CNN)



Digit-level (RNN)



COMPUTATIONAL POWERS



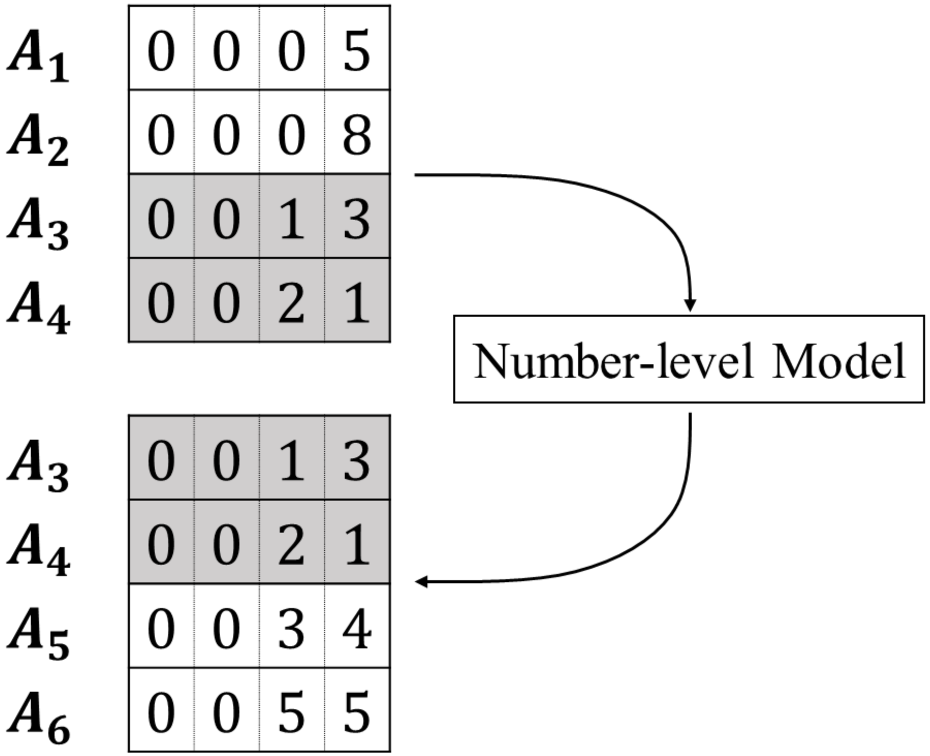
→ Digit-level Fibonacci prediction

→ Number-level Fibonacci prediction

CONTENTS

NUMBER-LEVEL DATA LAYOUT

2-dimensional grid of digits



- Digit: a b-dimension **one-hot vector**
- Number: a d-digit **row** A_i
- Input: a **grid** of n numbers $A_{1 \dots n}$
- Target: a **shifted sequence** $A_{s \dots s+n}$

NUMBER-LEVEL SEQUENCES

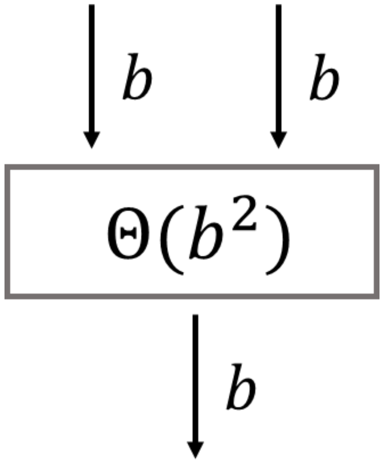
Order-k linear homogeneous recurrence

- Order-2 relations: $A_{n+2} = pA_{n+1} + qA_n$
  - Fibonacci: $(p, q) = (1,1)$ / Arithmetic: $(p, q) = (2, -1)$
- Order-3 relations: $A_{n+3} = pA_{n+2} + qA_{n+1} + rA_n$
  - Progression: $(p, q, r) = (3, -3, 1)$ / Jumping Fibonacci: $(p, q, r) = (1,0,1)$

- Number-level prediction is learning a combinatorial function of $(A_{n-k}, \dots A_n) \rightarrow A_{n+1}$

DIFFICULTY AND COMPLEXITY

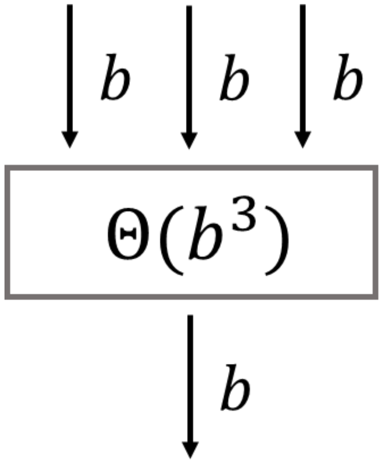
The number of logical gates and the depth of the circuit



Order-2 relation

Width = $\theta(b^2)$

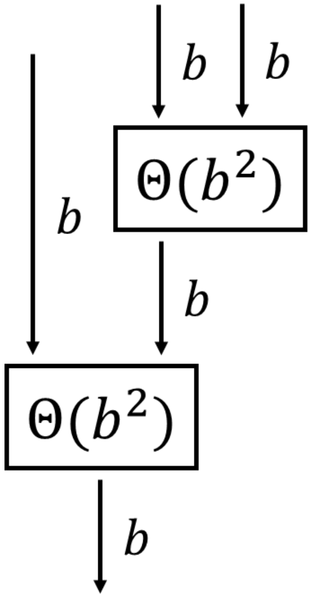
Depth = 1



Order-3 relation

Width = $\theta(b^3)$

Depth = 1



Order-3 relation

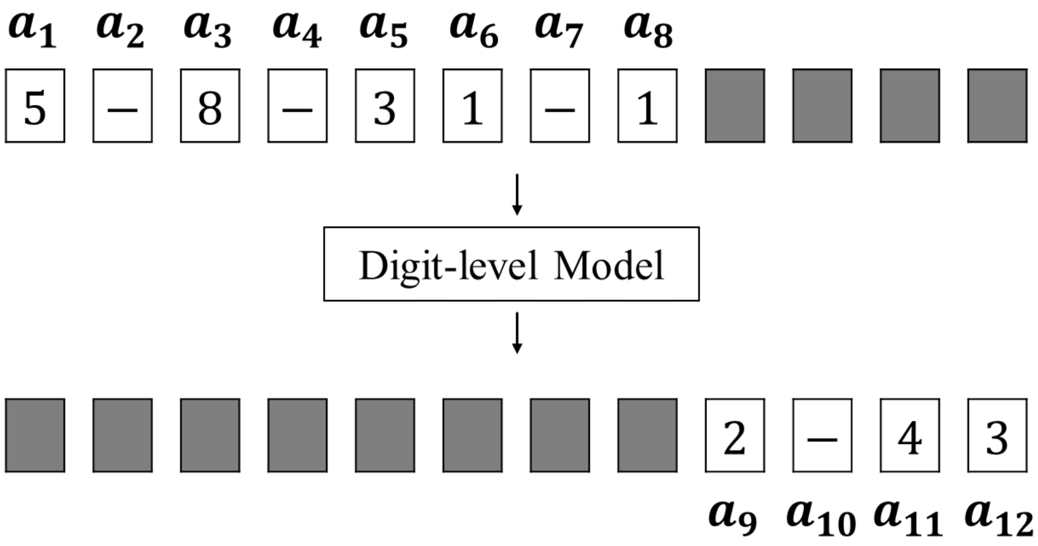
Width = $\theta(b^2)$

Depth = 2

CONTENTS

DIGIT-LEVEL DATA LAYOUT

Sequence of digits



- **Single-digit** data per a time step
- **Little-endian** order (smaller digits first)
- Input: n digits $a_{1\ldots n}$ followed by s blanks
- Target: n blanks followed by $a_{n\ldots n+s}$

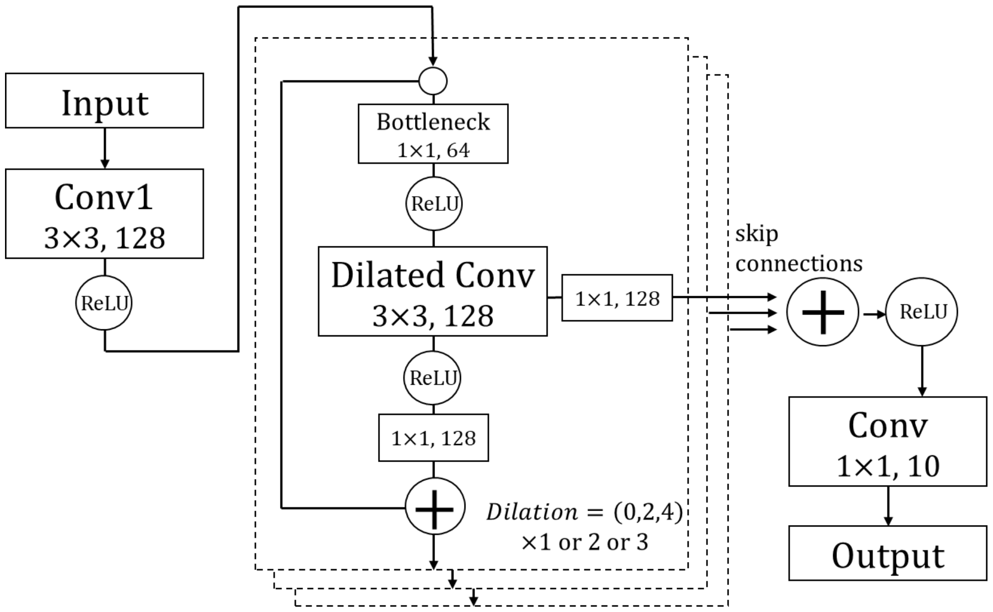
DIGIT-LEVEL SEQUENCES

Their complexities correspond to sequential state machines

- Counting sequences: **Finite** automata
    - $A_{n+1} = A_n + c$ (fixed c)
- Palindromes: **Pushdown** automata
    - Finite length palindromes are solvable by finite automata
    - Training with length 1~12 / Validation with length 16
- Fibonacci/Arithmetic/Geometric: **Queue** automata
    - Cannot be solved by stack calculator in this setup
    - Queue automata are equivalent to **Turing machines**

CONTENTS

NUMBER-LEVEL CNN MODEL

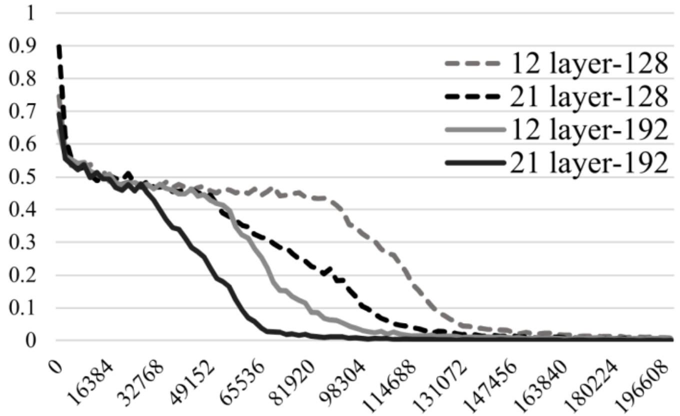


- Residual CNNs with dilated convolutions
- 12 (1 block) / 21 (2 blocks) / 30 (3 blocks) layer configurations
- 64 / 128 / 192 internal channels
- Input and output have same dimensions

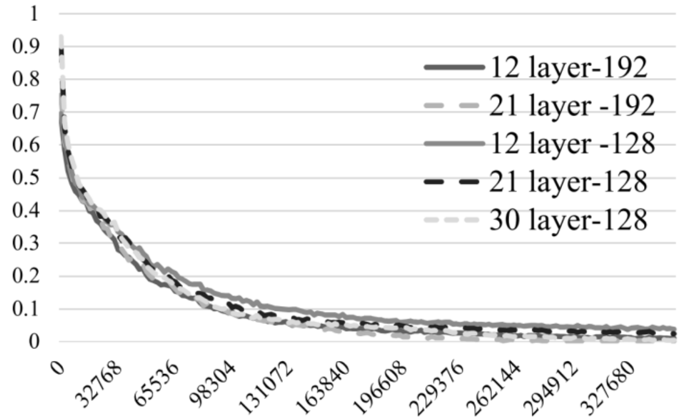
NUMBER-LEVEL RESULTS

Deep / Shallow & Wide / Deeper & narrow

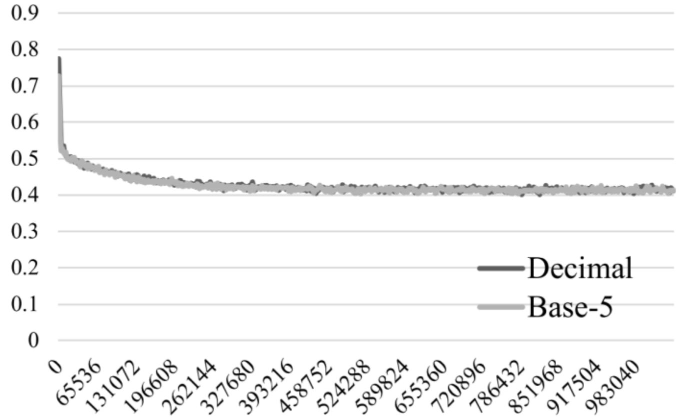
$A_n = 2A_{n-1} - A_{n-2} + A_{n-3}$



$(p,q) \in \{(1,1),(2,-1),(3,-2),(1,2)\}$



$A_n = 4A_{n-1} - 6A_{n-2} + 4A_{n-3} - A_{n-4}$

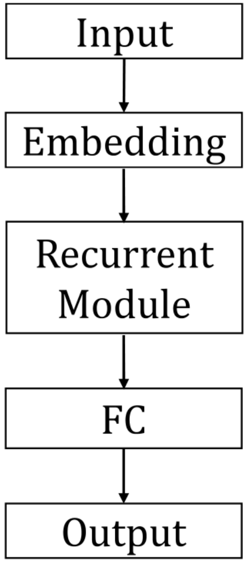


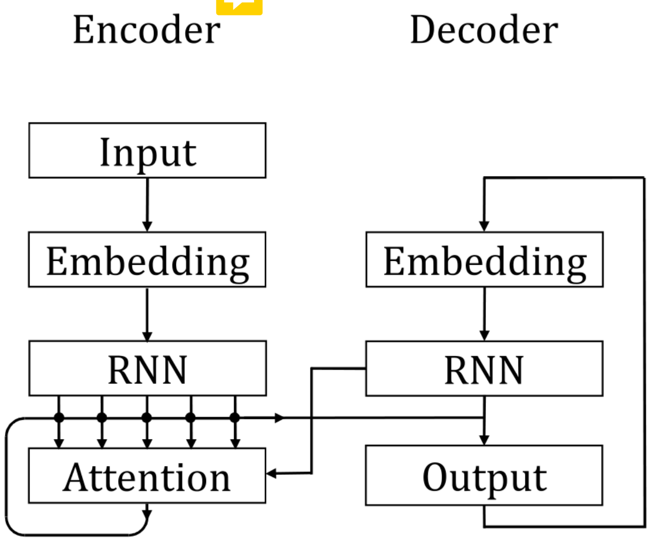
Depth of a problem is a better indicator for the complexity

CNNs tend to learn **deep but narrow** rules

Could not solve 3+ deep problems

DIGIT-LEVEL MODELS





Recurrent Module: LSTM, GRU, Stack-RNN, or Neural Turing Machine

Encoder-decoder model with **Attention**

DIGIT-LEVEL RESULTS

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%

Palindrome training errors suggest that all of them can simulate finite automata

Memory-augmented models could simulate up to pushdown automata

None of them could solve problems with complexity of queue automata

CONTENTS

CONTRIBUTIONS

- Suggested an algorithmic task suite for machine learning
  - Well-defined and possible to generate arbitrary number of examples
- Defined the complexities of the sequence generation rules
  - Effective ways to predict the difficulties of the problems
- Showed that computational powers of current deep learning models are limited
  - Even complex memory augmented models are not Turing-capable yet

DISCUSSIONS & FUTURE WORKS

Possible ways to overcome the computational limits

- Architecture-level
  - Turing-capable memory architectures
  - CNN achitecture for deeper combinatorial logic
- Training-level
  - Decouple number of inputs and computation steps
  - Reinforcement learning, Incremental training with transfer learning, etc.

Machine Intelligence lab
SEOUL NATIONAL UNIVERSITY

THANK YOU

Hyoungwook Nam
hwnam831@snu.ac.kr

Segwang Kim
ksk5693@snu.ac.kr

Kyomin Jung
kjung@snu.ac.kr