Learning to Execute

by **Wojciech Zaremba** and Ilya Sutskever

ref: http://arxiv.org/abs/1410.4615

NYU

Google

Examples

Input:

```
i=8827
c=(i-5347)
print((c+8704) if 2641<8500 else
    5308)
```

Target: 12184.

Input:

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

Target: 25011.

Sequence of character on the input and on the output.

Why is it important ?

It’s a very hard task that requires:

- modelling long-distance dependencies
- memory (e.g. variable assignment)
- branching (if-statement)
- multiple tasks within one

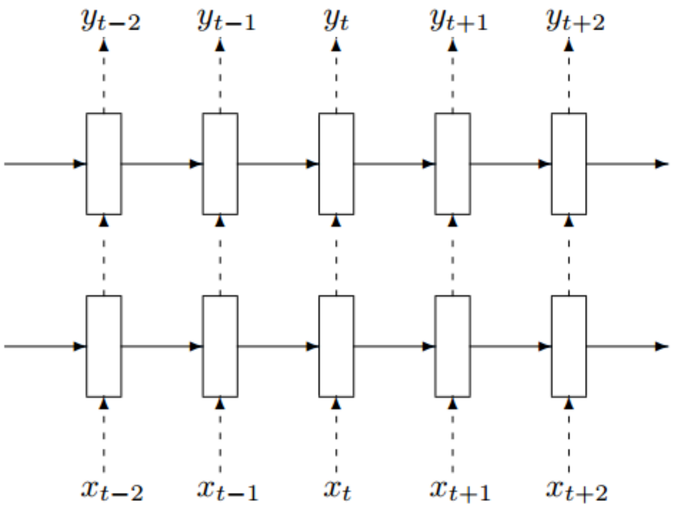
Data consumption

Model reads programs character by character, and tries to predict execution output.

It doesn’t need to predict the next character in every step.

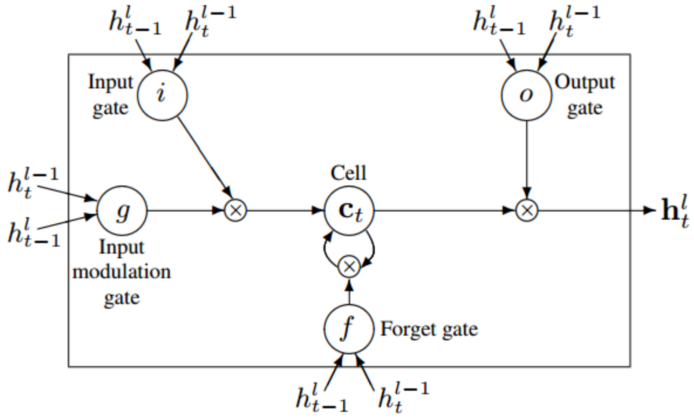
Our model - RNN

- 2 layers
- 400 units each
- trained with SGD
- cross-entropy loss
- Input vocabulary size 42
- Output vocabulary size 11



Our model - RNN with LSTM* cells

- LSTM presumably can model long range dependencies
- Train until there is no improvement on a validation set.



* Graves, Generating Sequences With Recurrent Neural Networks

Subclass of programs

- can be evaluated with a single left-to-right pass
- operations: addition, subtraction, multiplication, variable assignment, if-statement, and for-loops
- Problem complexity is defined in terms of the length of numbers and depth of nesting

Why is it difficult ?

RNN’s point of view:

Input:
vqppkn
sqdvfljmnc
y2vxdddsepnimcbvubkomhrpliibtwztbljipcc
Target: hkhpg

Qualitative results. Exact prediction.

Input:

```
f=(8794 if 8887<9713 else (3*8334))
print((f+574))
```

Target: 9368.

Model prediction: 9368.

Properly deals with if statement and addition.

Qualitative results. 1 digit mistake.

Input:

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

Target: 25011.

Model prediction: 23011.

Often leading digits and the last digits are correct.

Qualitative results. Exact prediction.

Input:

```
c=445
d=(c-4223)
for x in range(1):
  d+=5272
print((8942 if d<3749 else 2951))
```

Target: 8942.

Model prediction: 8942.

Some very nested examples might be very simple.

Qualitative results. 2 digit mistake.

Input:

```
a=1027
for x in range(2):
  a+=(402 if 6358>8211 else 2158)
print(a)
```

Target: 5343.

Model prediction: 5293.

Again, leading digits and the last digits are correct.

Scheduling strategies

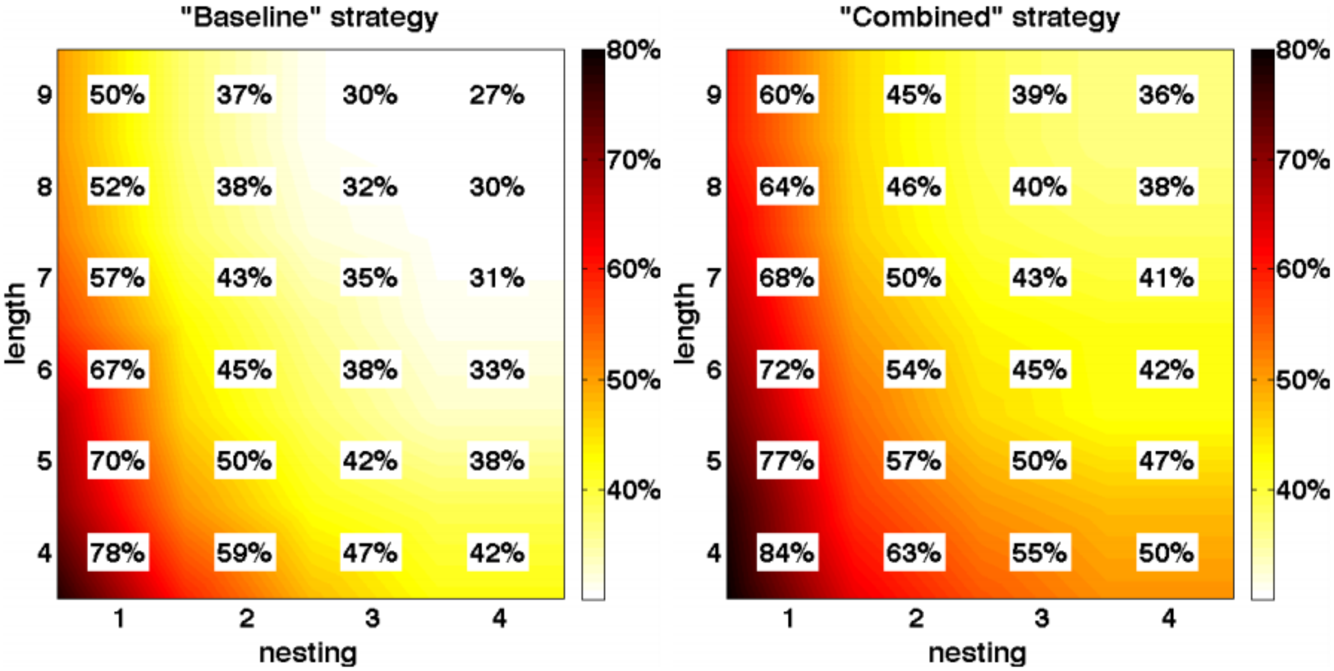
- No curriculum learning (baseline)
  - Learning with target distribution
- Naive curriculum strategy (naive)
  - Making task gradually more difficult

Scheduling strategies

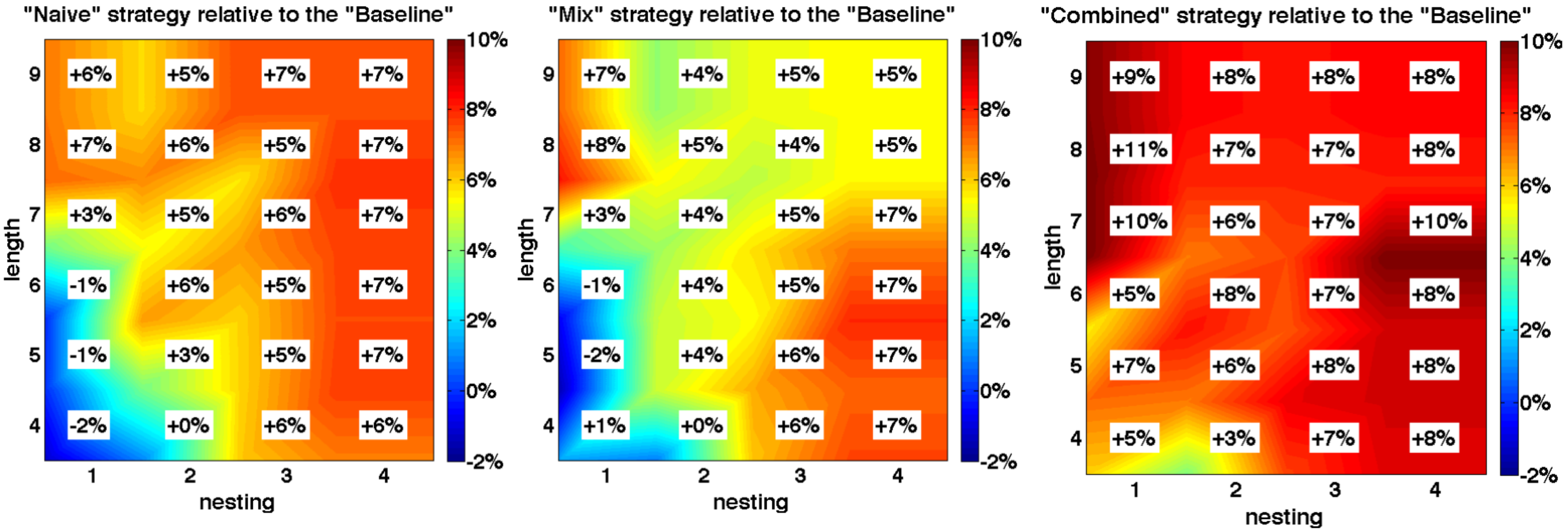
- Mixed strategy (mix)
  - Mix of all levels of hardness. Simplest programs occur as often as hardest one. Distribution rand (10^rand(length)) vs rand(10^length).

- Combined strategy (combined)
  - Combination of mix with naive curriculum learning (so far the best).

Quantitative results. Absolute performance.



Quantitative results. Relative performance.



Understanding vs. memorizing

- We don’t know how much our networks “understand” the meaning of programs vs how much they memorize.
- Test dataset, validation dataset, and training datasets have no common samples, but are very similar.

Copying task

- Consume string of numbers and reproduce the same string.
- Finite number of epochs.
- How good is LSTM memory ?
- How to prime LSTM memory toward memorization ?

Input: 123456789$, Target: 123456789.

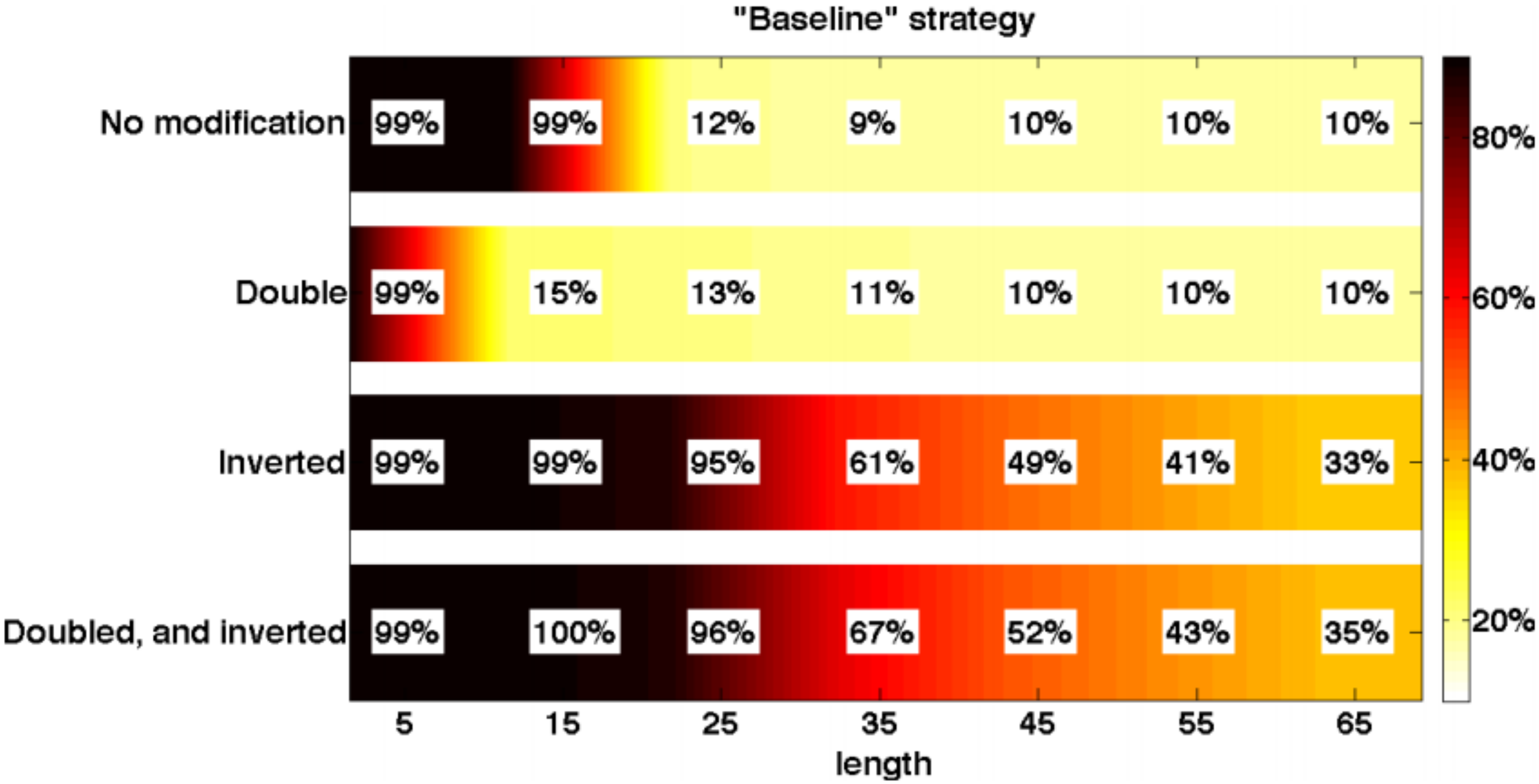
Priming strategies

- Inverting input*
  - Much easier to learn identity than suppress intermediate results (e.g. 98765432**1 -> 1**…. vs **1**23456789 -> **1**….).

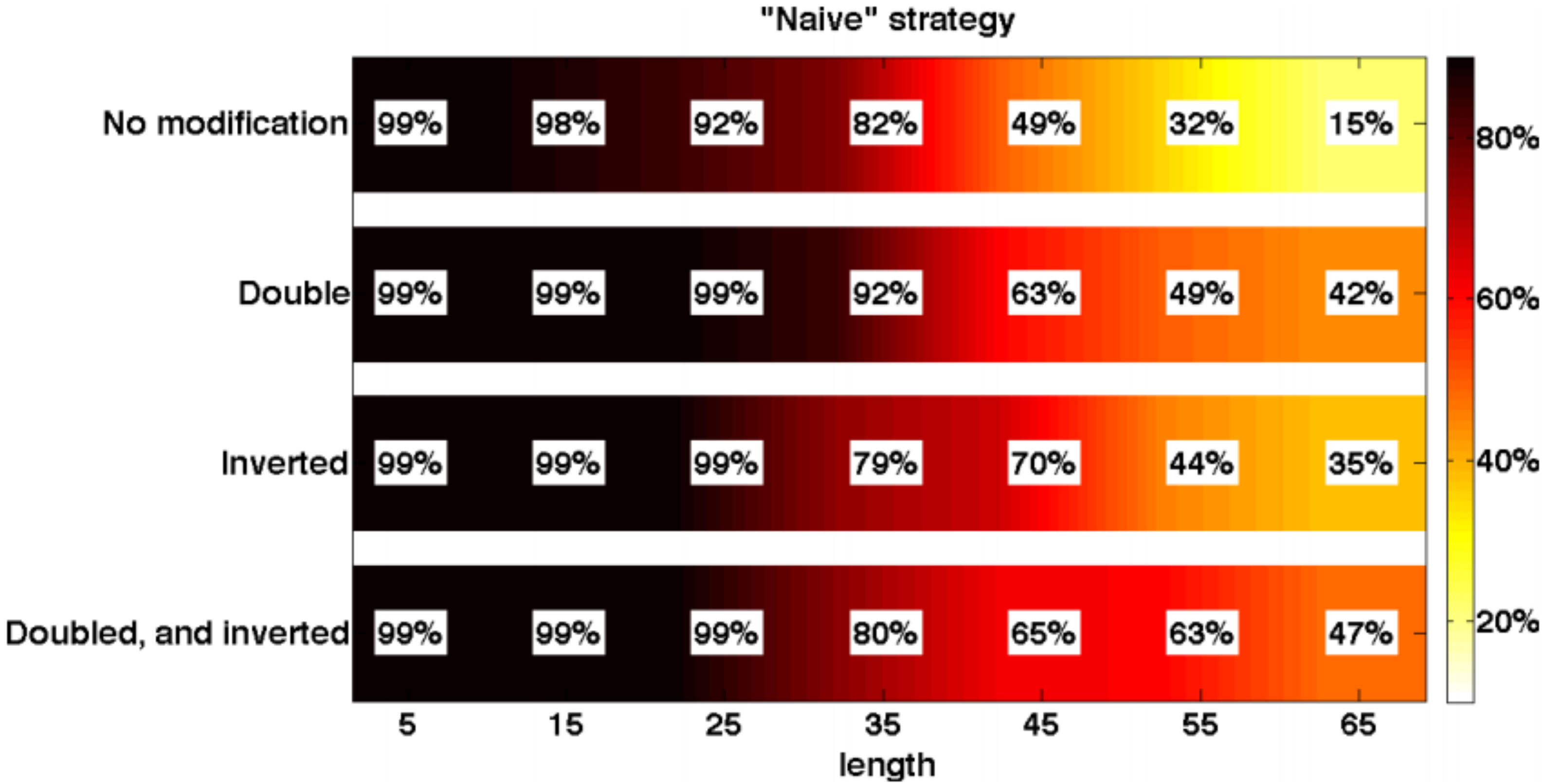
- Doubling input
  - Allows to refine memories.

*Sutskever et al., Sequence to sequence learning.

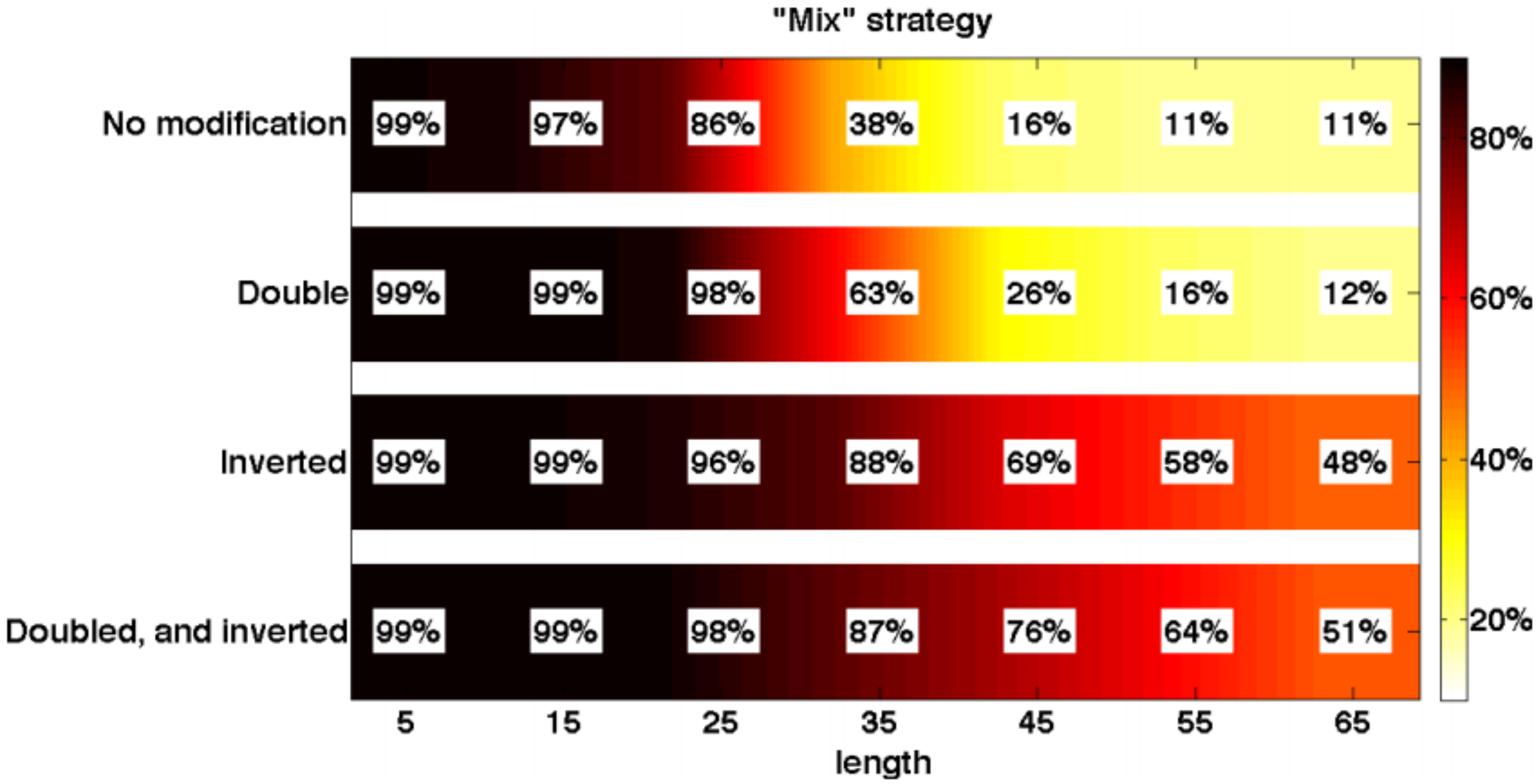
Copying results. baseline strategy.



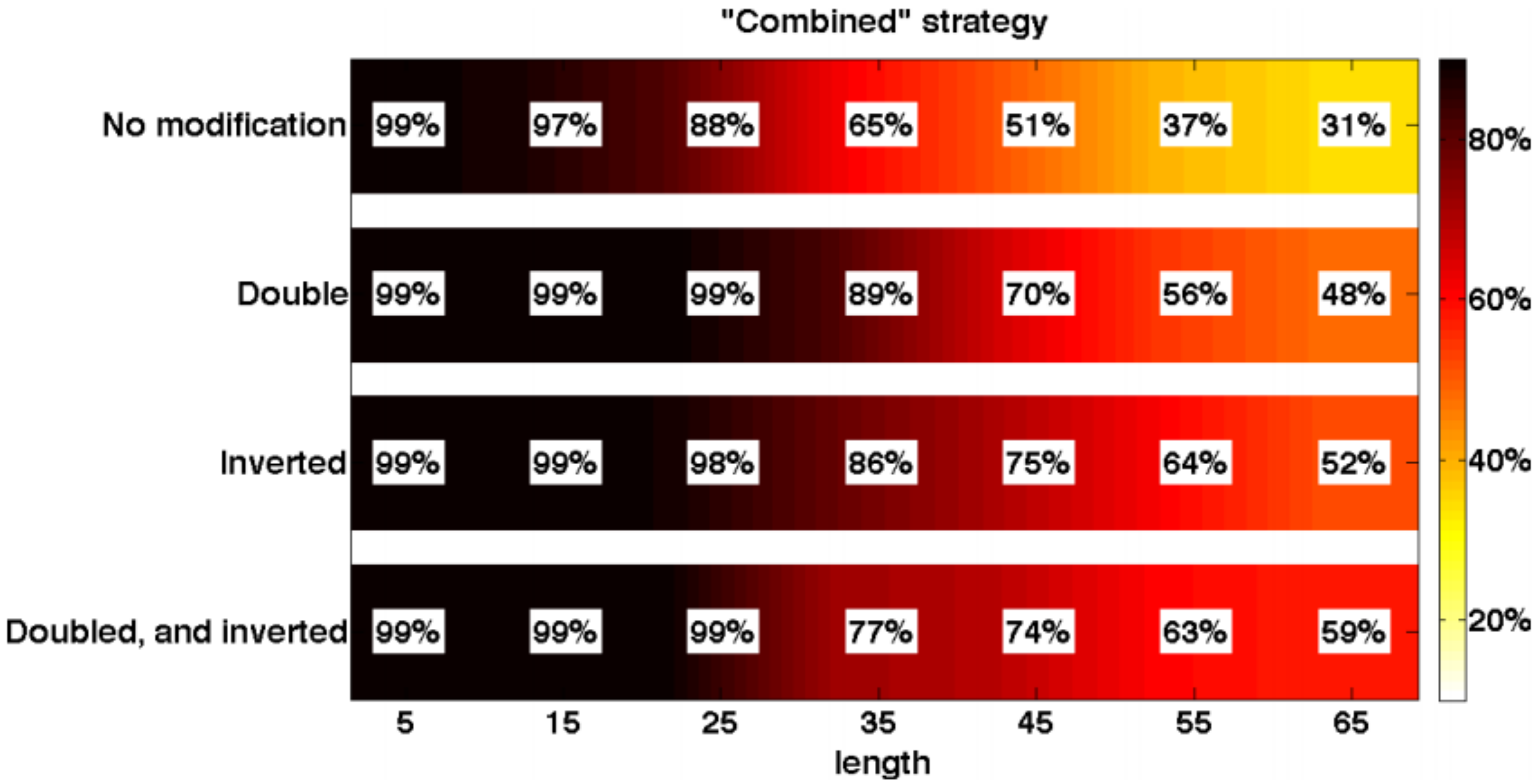
Copying results. naive strategy.



Copying results. mixed strategy.



Copying results. combined strategy.



Q&A

- Predicting program execution results
- RNN with LSTMs
- Scheduling strategies (baseline, naive, mix, combined)
- Copying task, and priming (inverting, doubling input).

I am happy to answer any questions.