# Adaptive Computation Time for Recurrent Neural Network

Alex Graves

# Outline

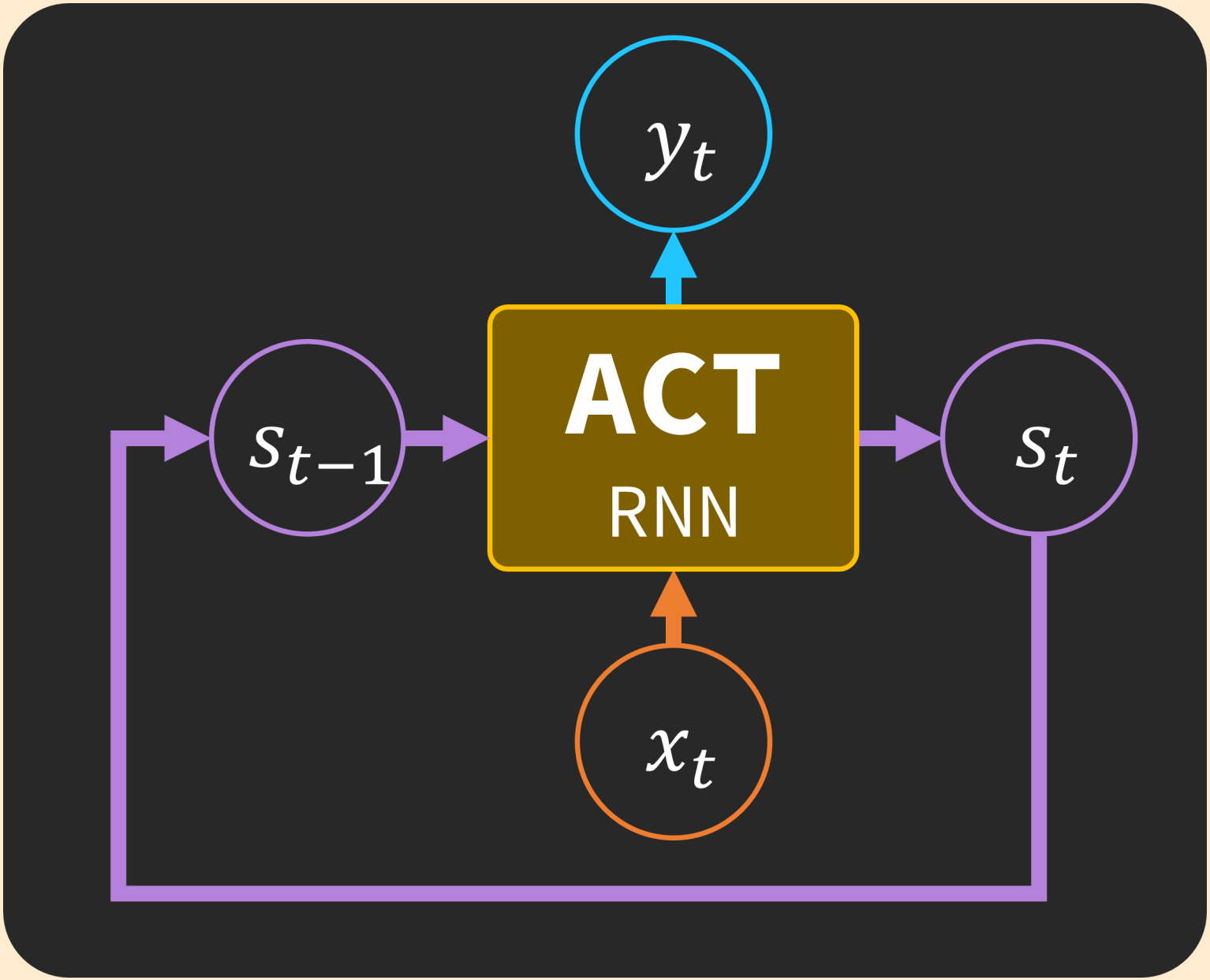
- Introduction
- Methodology
- Experiments
- Conclusion

# Introduction

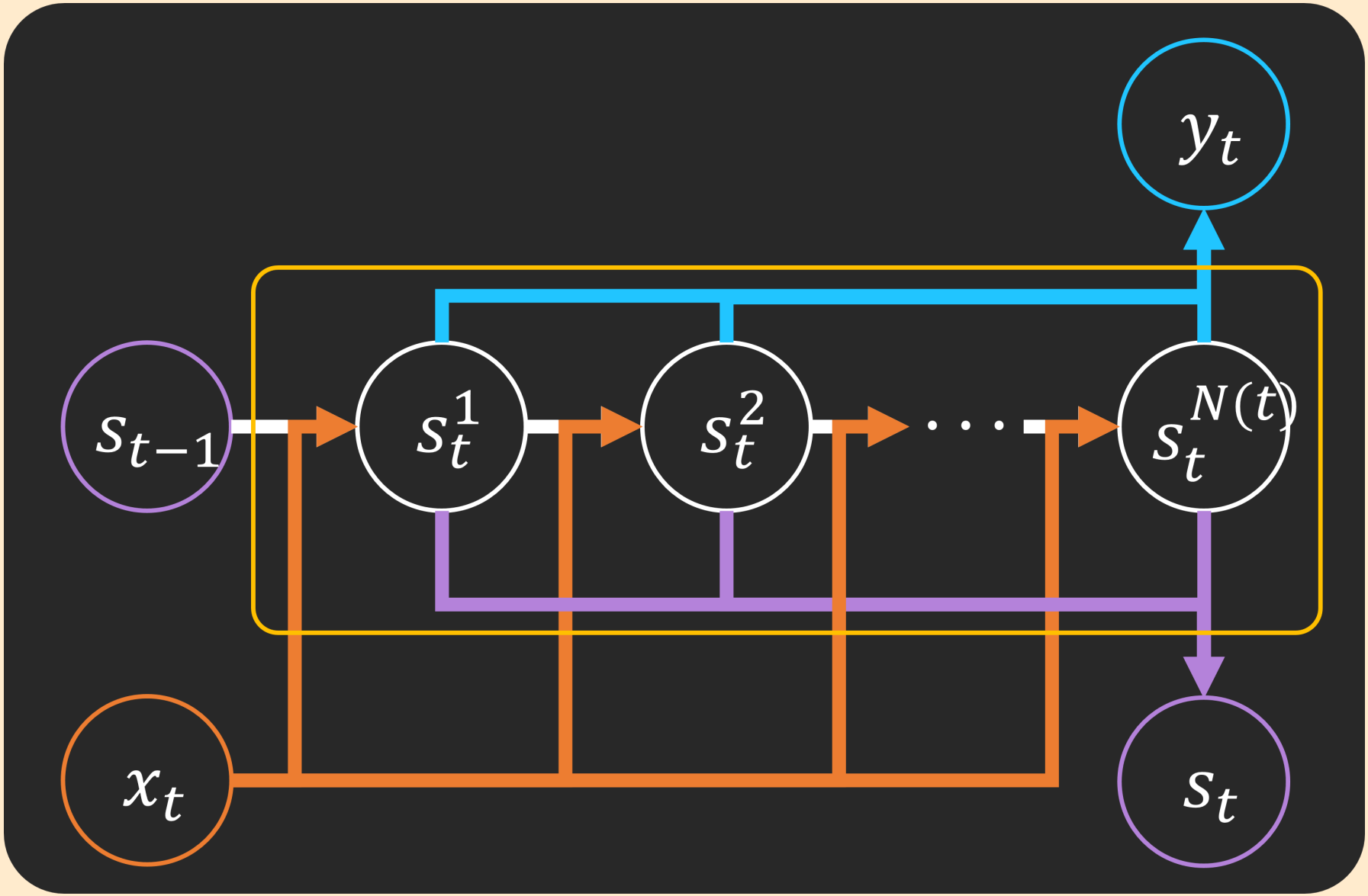
Even in the same type of task, different cases have different complexity.
However, traditional deep learning methods use the same amount of calculation to process, which is obviously unreasonable.

Therefore, this paper proposes a method to make neural network adaptive computation times.
Using this method, you can spend appropriate calculation costs on cases of different complexity and achieve good results.

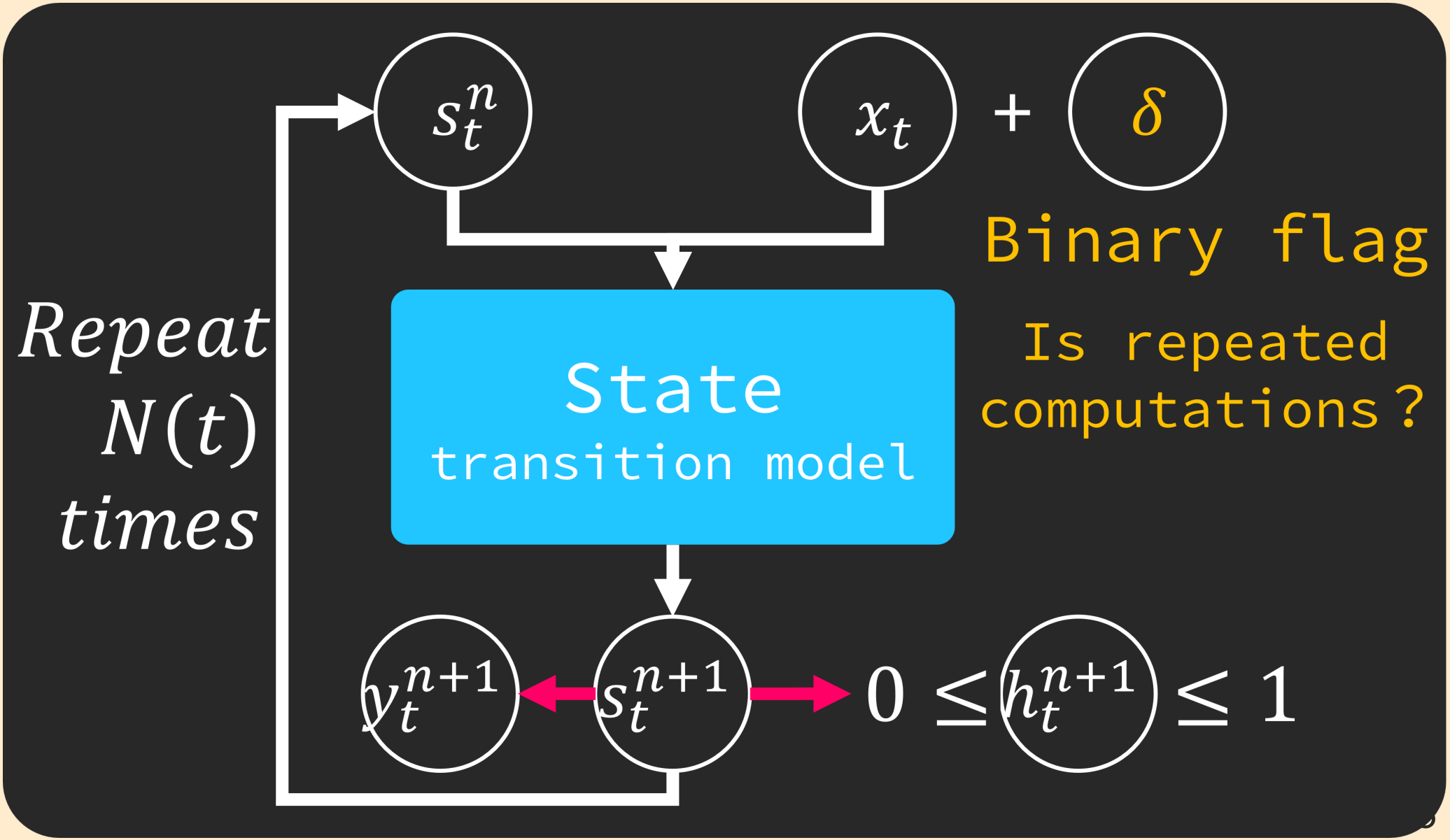
# Methodology



# ACT



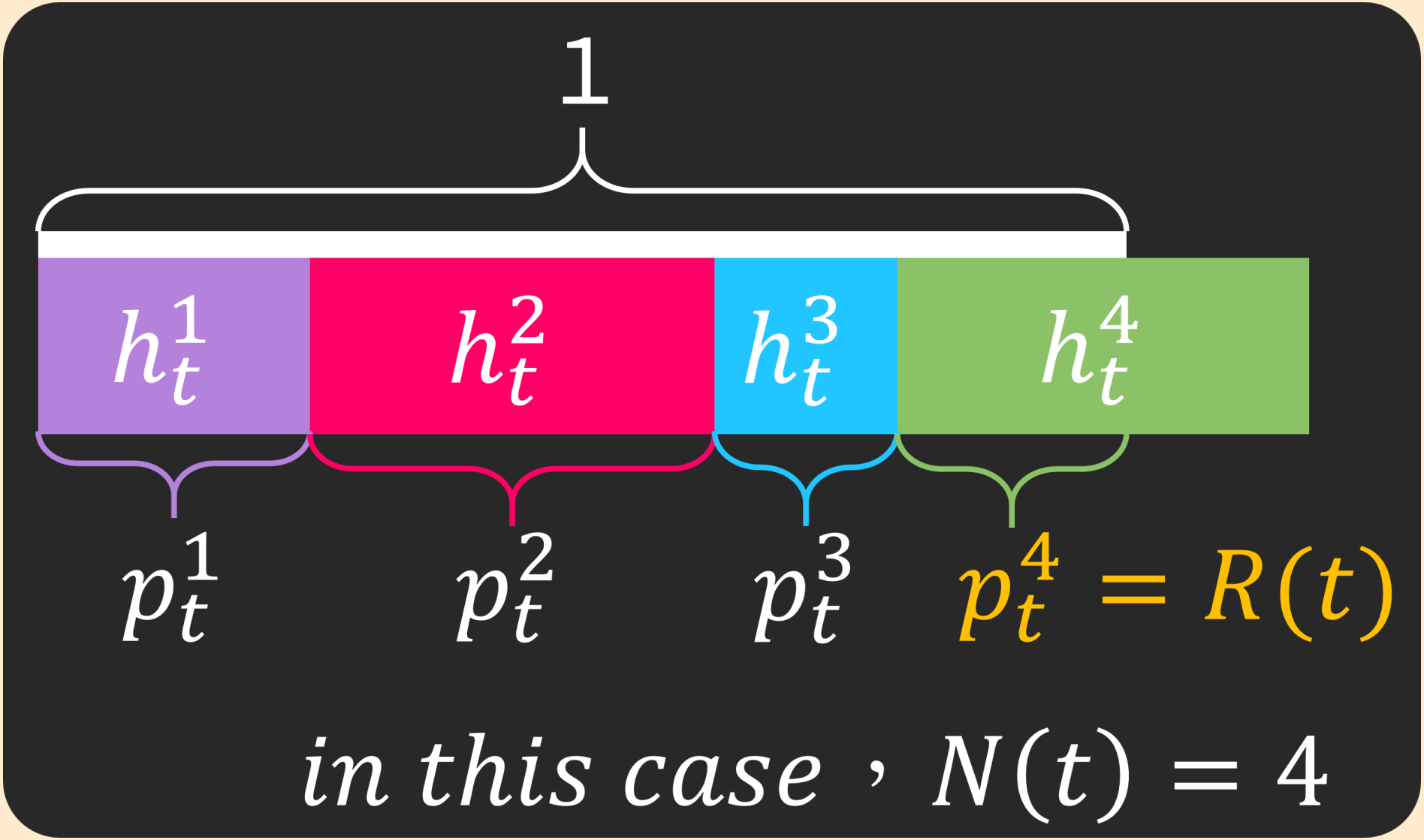
# ACT


$s_t^n$
$x_t$
+
$\delta$
Binary flag
Is repeated computations?
Repeat $N(t)$ times
State
transition model
$y_t^{n+1}$
$s_t^{n+1}$
$0 \leq h_t^{n+1} \leq 1$

# Repeat Times

$$N(t): 1 < \sum_{n=1}^{N(t)} h_t^n \ and \ 1 > \sum_{n=1}^{N(t)-1} h_t^n$$

# Halting Probability



# Remainder

$$R(t) = 1 - \sum_{n=1}^{N(t)-1} h_t^n$$

# Output & State

$$y_t = \sum_{n=1}^{N(t)} p_t^n \, y_t^n$$

$$s_t = \sum_{n=1}^{N(t)} p_t^n \, s_t^n$$

# Loss

$$\mathcal{L} = \mathcal{L}_{Task} + \mathcal{L}_{Ponder}$$

# Ponder Loss



$$\mathcal{L}_{\mathcal{P}} = \tau \sum_{t=1}^{T} \rho_t$$

$$= \tau \sum_{t=1}^{T} [N(t) + R(t)]$$

# Ponder Loss



$$\mathcal{L}_{\mathcal{P}} = \tau \sum_{t=1}^{T} [N(t) + R(t)]$$

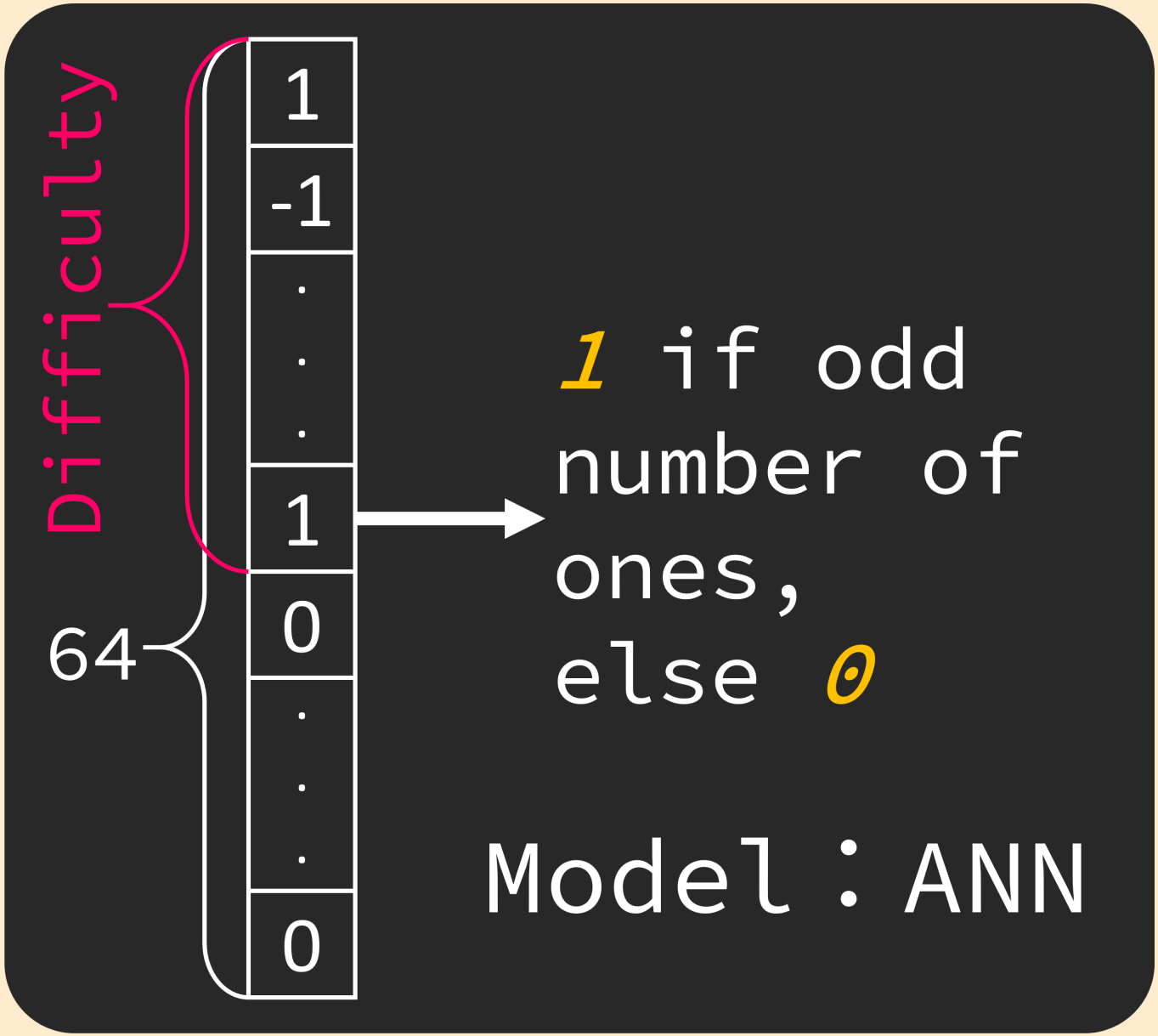
# Ponder Loss



$$\mathcal{L}_{\mathcal{P}} = \tau \sum_{t=1}^{T} \left(1 - \sum_{n=1}^{N(t)-1} h_t^n\right)$$

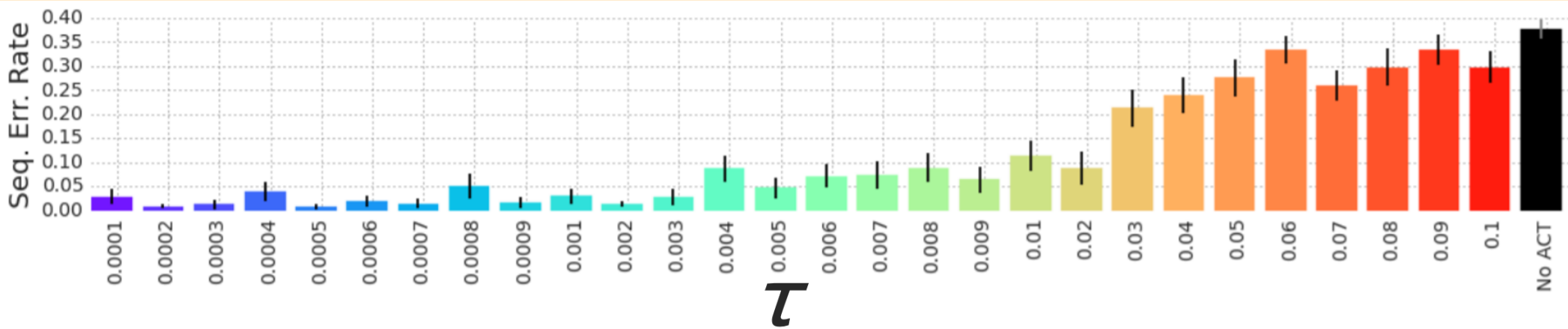
目標：$h_t^{n<N(t)}$ 越大越好

# Parity



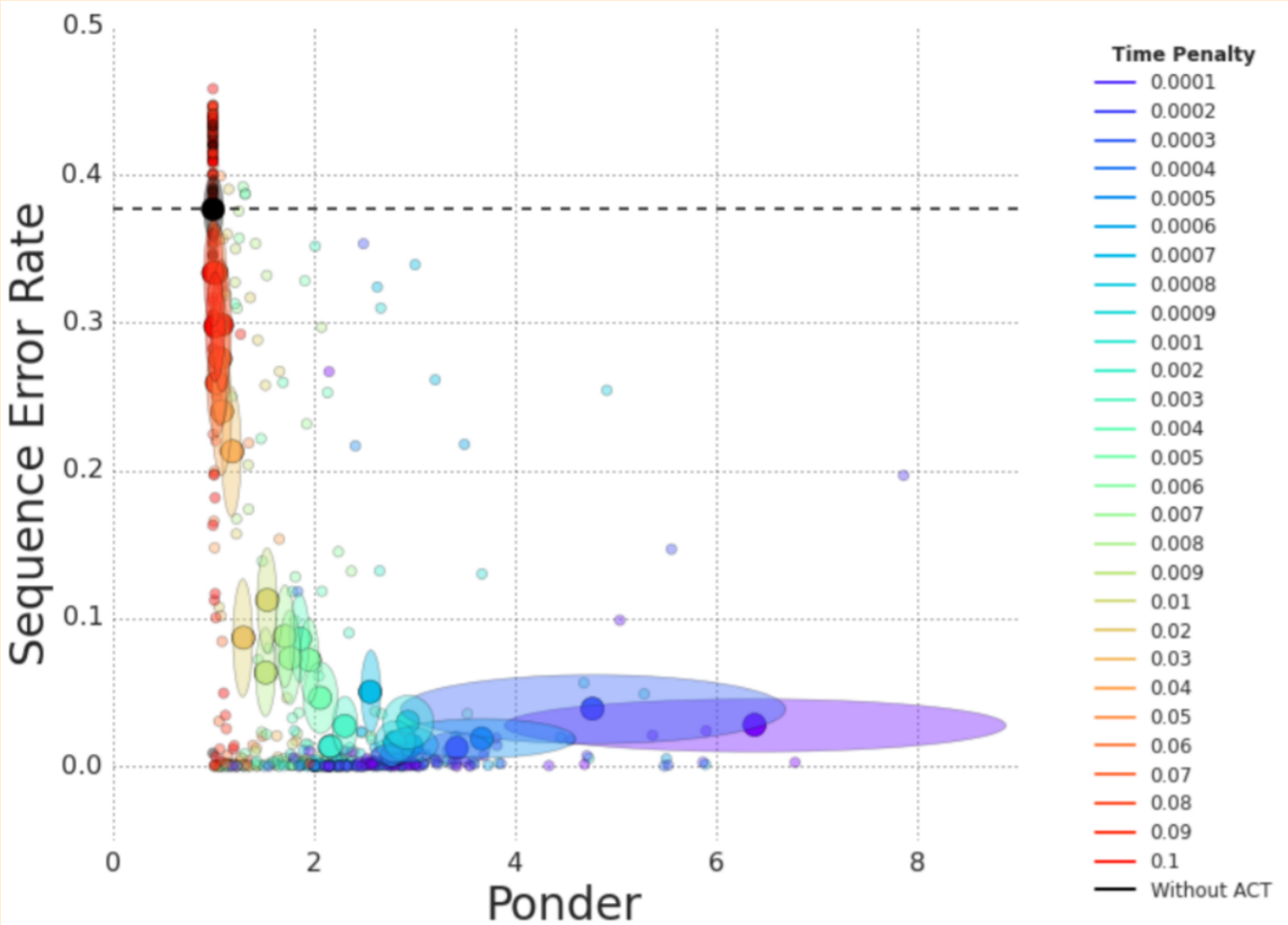
# Time Penalty





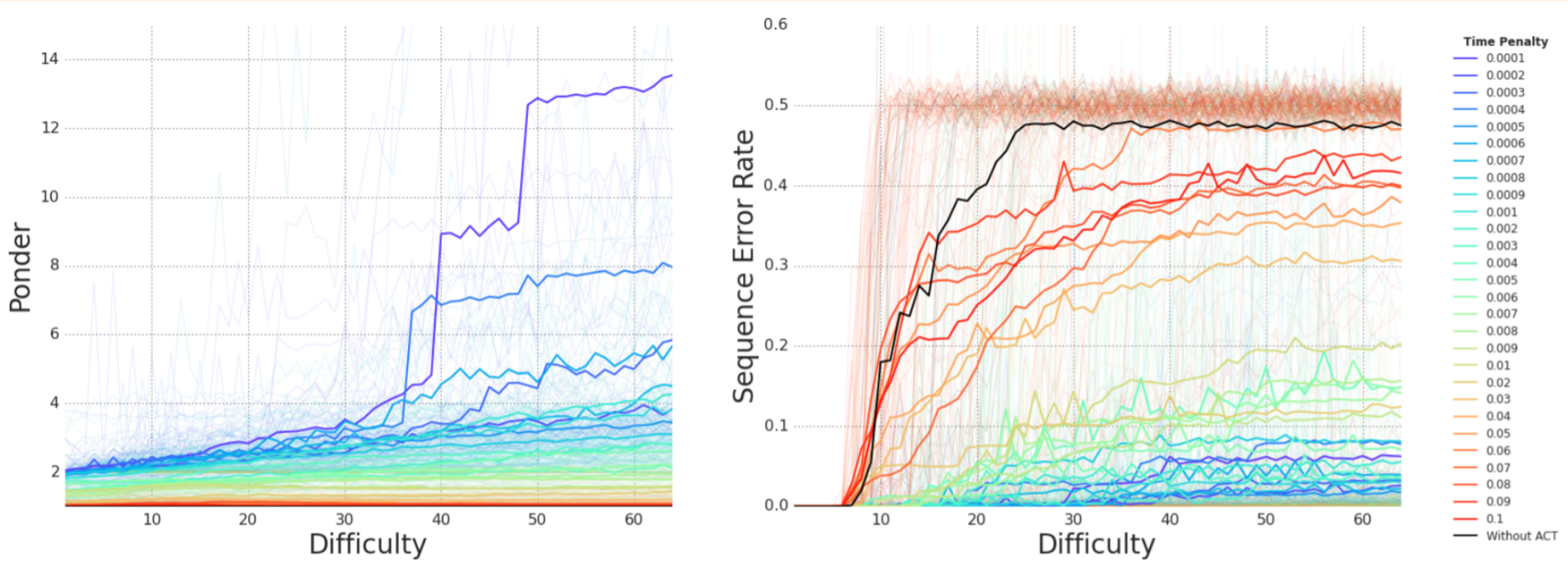
# Ponder



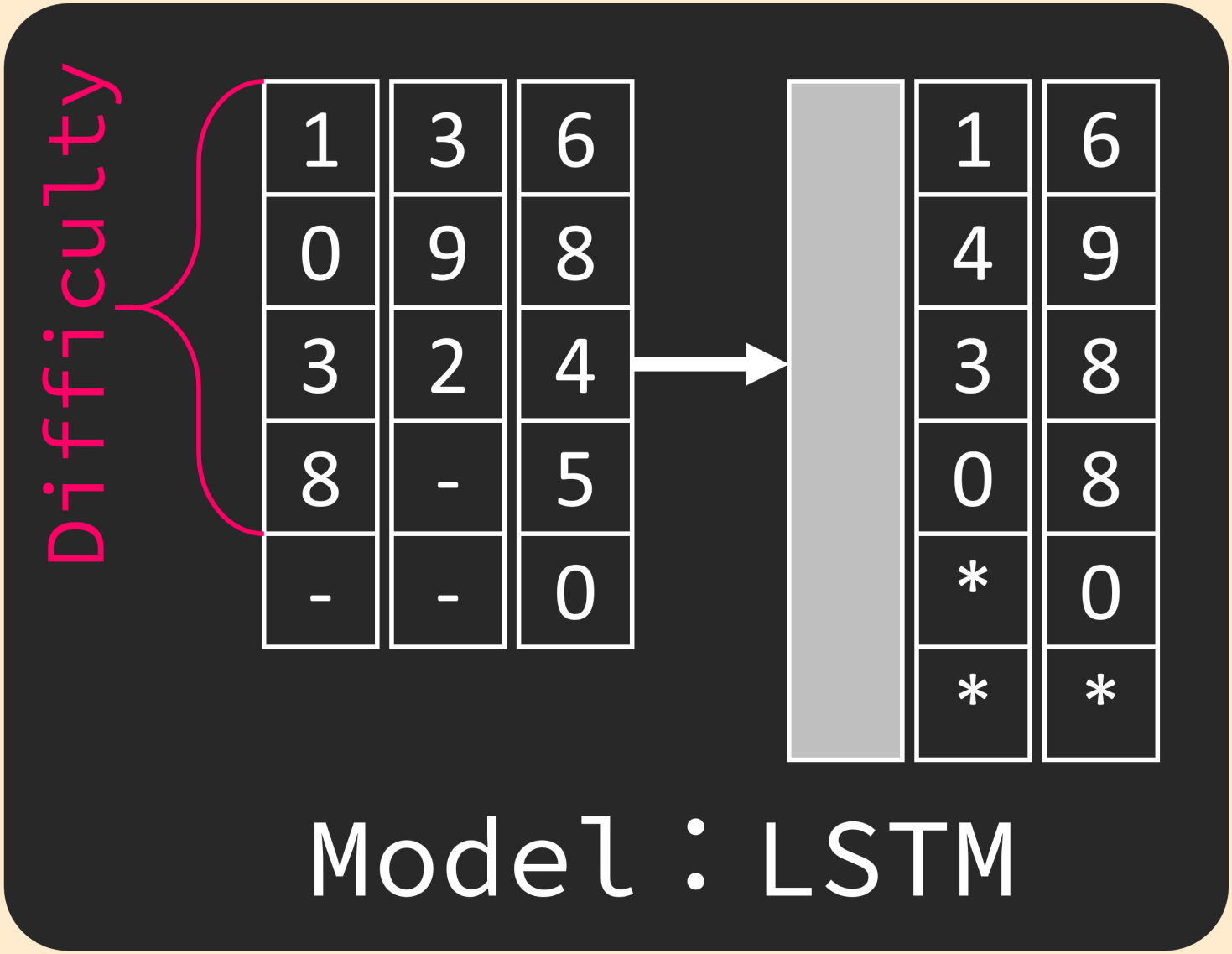

Sequence Error Rate
0.5
0.4
0.3
0.2
0.1
0.0
0
2
4
6
8
Ponder
Time Penalty
0.0001
0.0002
0.0003
0.0004
0.0005
0.0006
0.0007
0.0008
0.0009
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
Without ACT

# Ponder



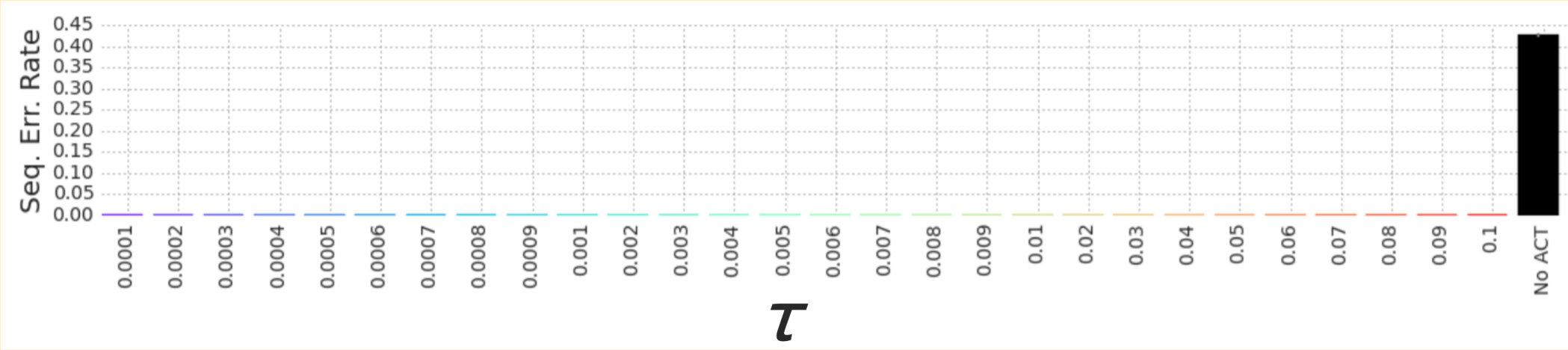

Ponder
Difficulty
Sequence Error Rate
Difficulty
Time Penalty
0.0001
0.0002
0.0003
0.0004
0.0005
0.0006
0.0007
0.0008
0.0009
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
Without ACT

# Addition



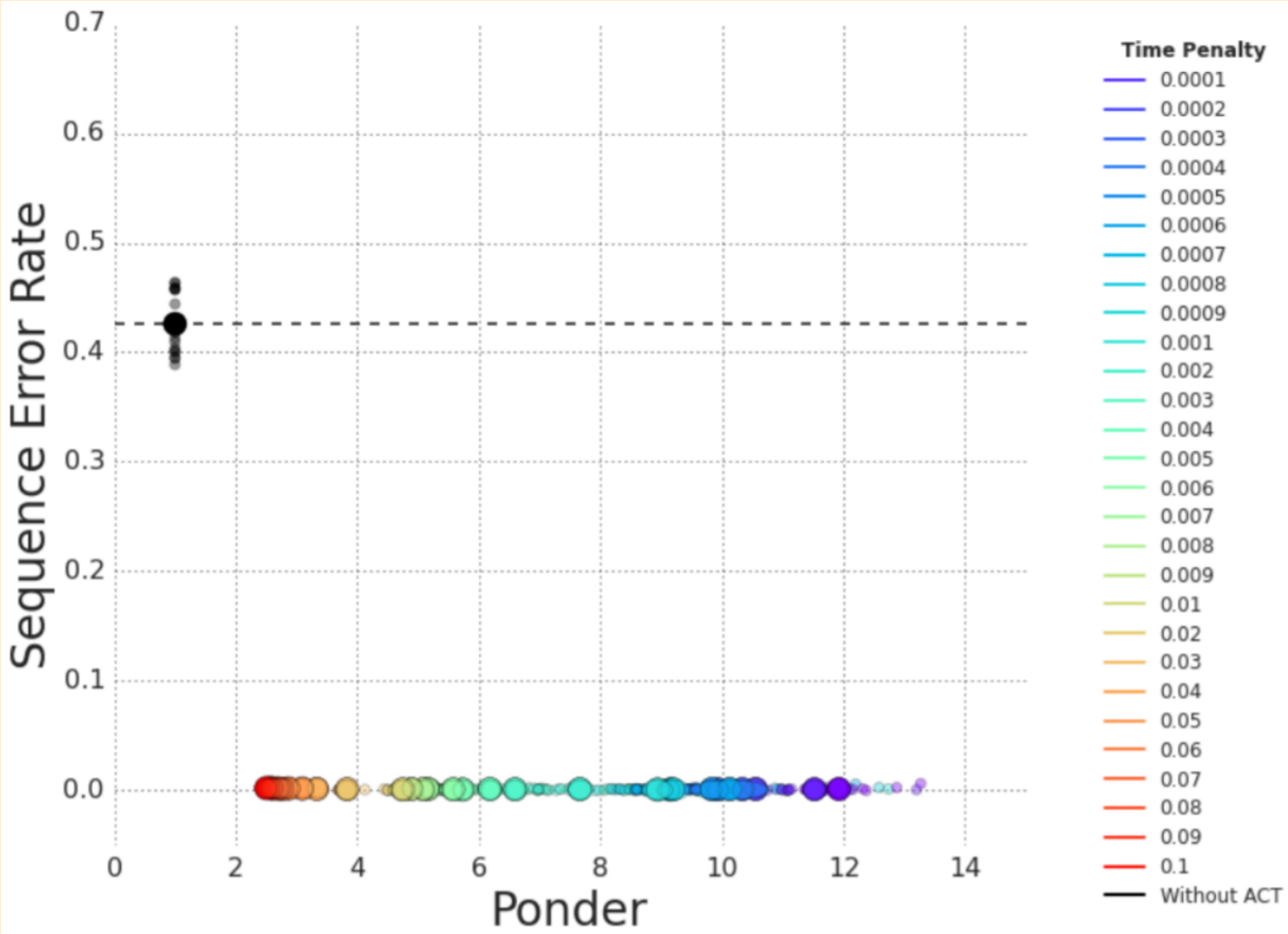
# Time Penalty





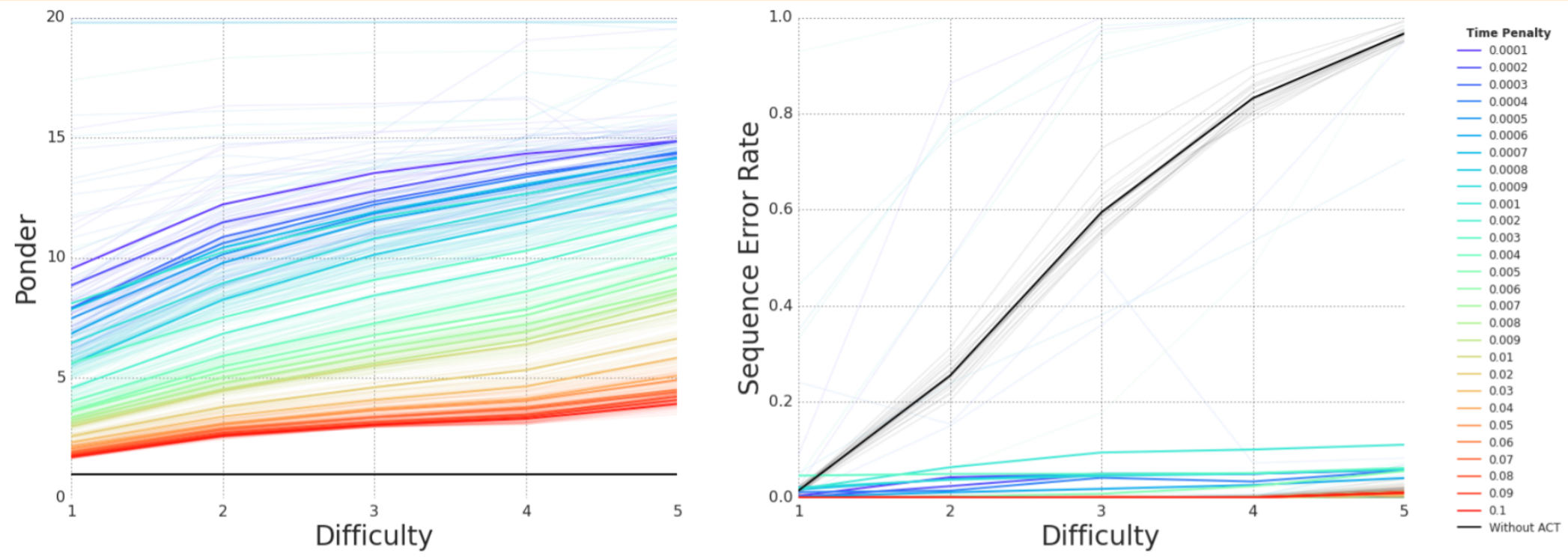
# Ponder




Sequence Error Rate
Ponder
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
2
4
6
8
10
12
14
Time Penalty
0.0001
0.0002
0.0003
0.0004
0.0005
0.0006
0.0007
0.0008
0.0009
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
Without ACT

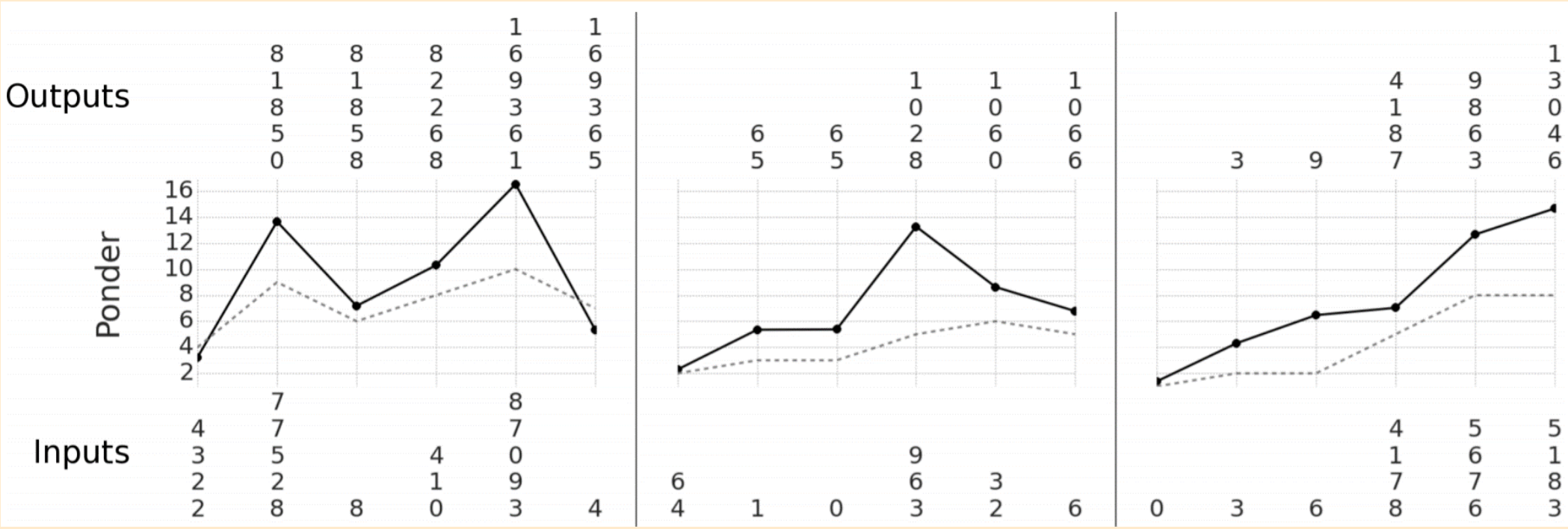
# Ponder



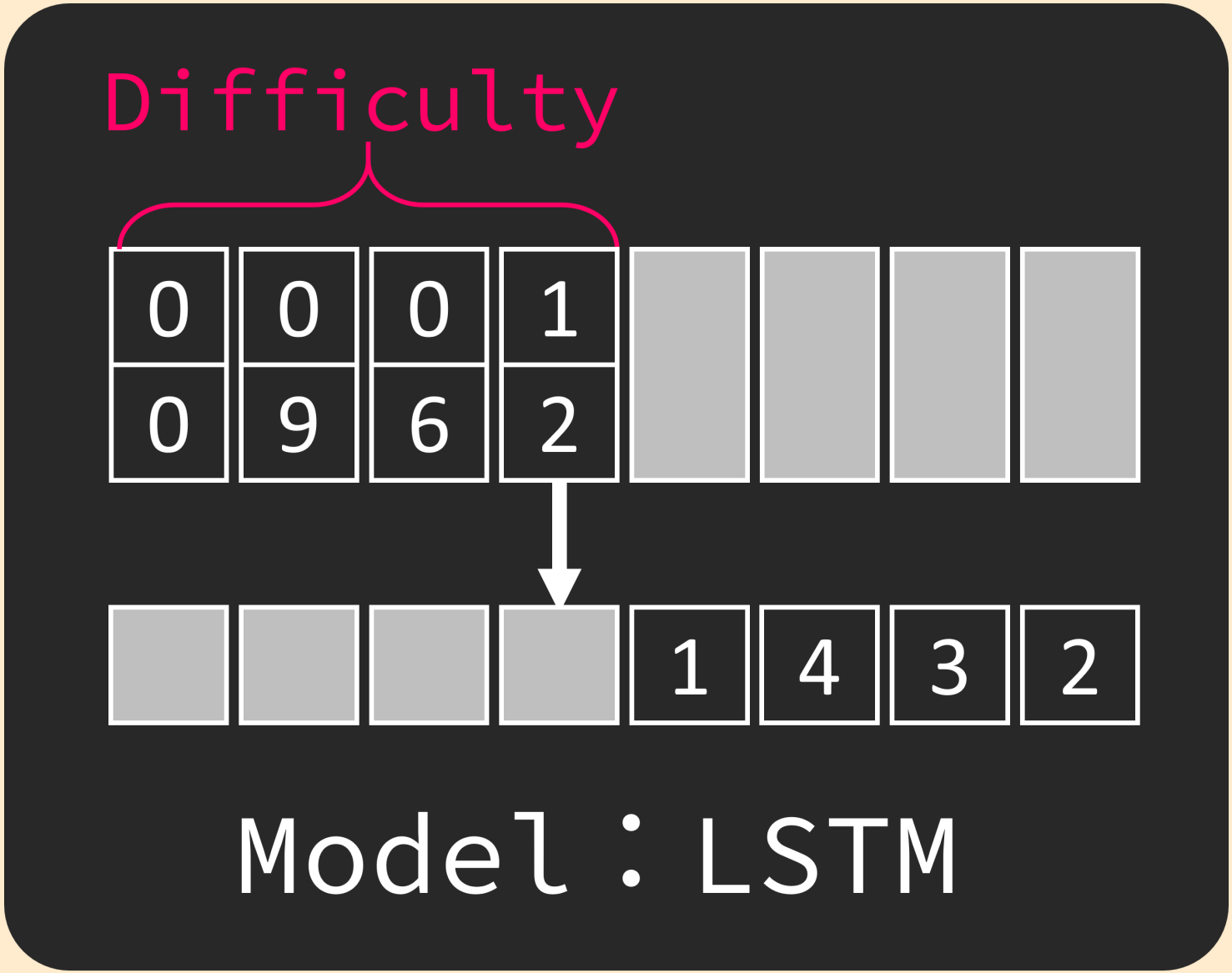

Ponder
Difficulty
Sequence Error Rate
Difficulty
Time Penalty
0.0001
0.0002
0.0003
0.0004
0.0005
0.0006
0.0007
0.0008
0.0009
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
Without ACT

# Ponder



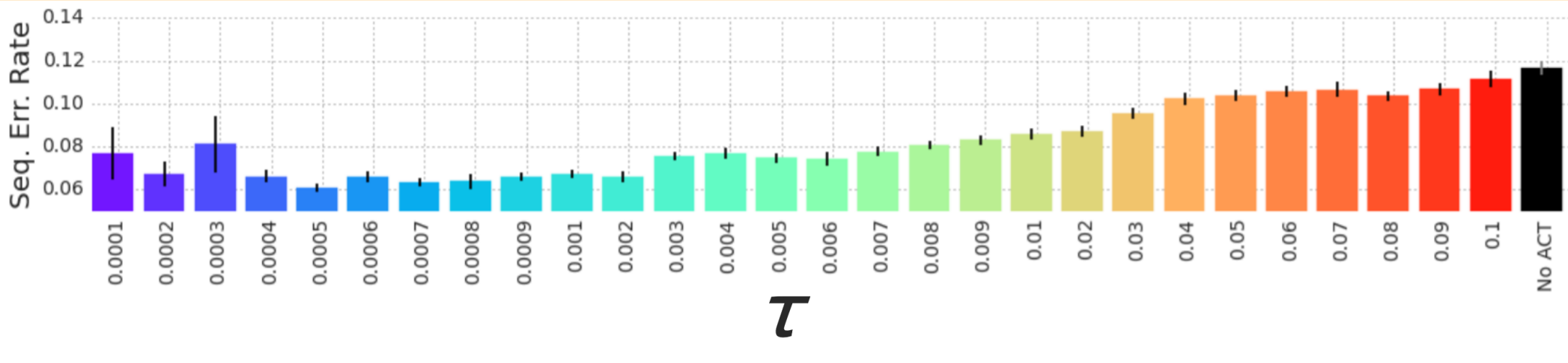

Outputs
8 8 8 16 16
1 1 2 9 9
8 8 2 3 3
5 5 6 6 6
0 8 8 1 5
Ponder
16
14
12
10
8
6
4
2
Inputs
7 8
4 7 7 5 2 8 8 4 1 0 8 7 0 9 3 4 2 2
6 5 6 5 10 10 10 2 6 0 6 6
6 4 1 0 9 6 3 2 6
3 9 4 1 8 7 13 0 4 9 8 6 4 6
4 5 5 1 6 7 1 8 7 8 6 3
0 3 6

# Sort



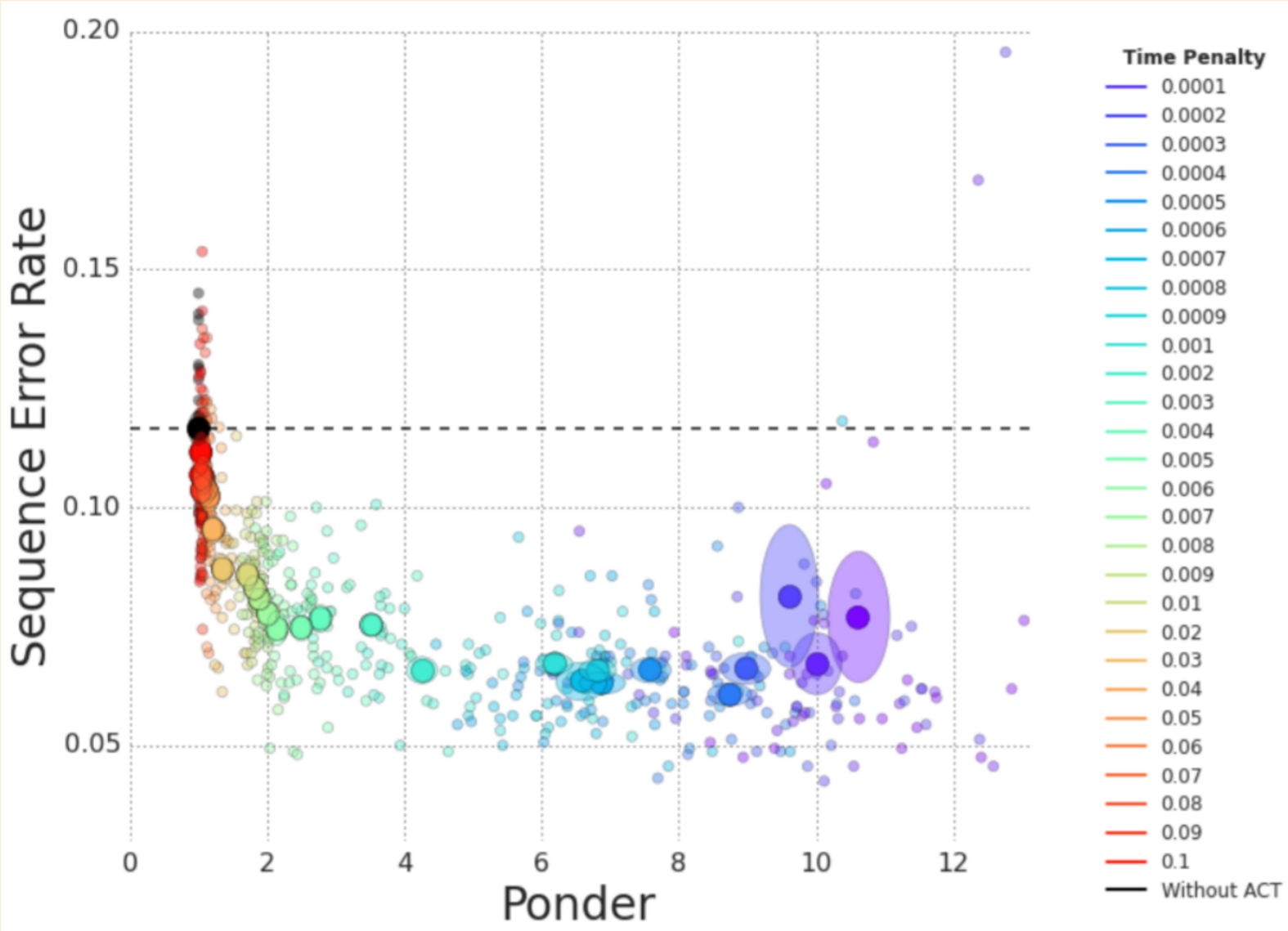
# Time Penalty





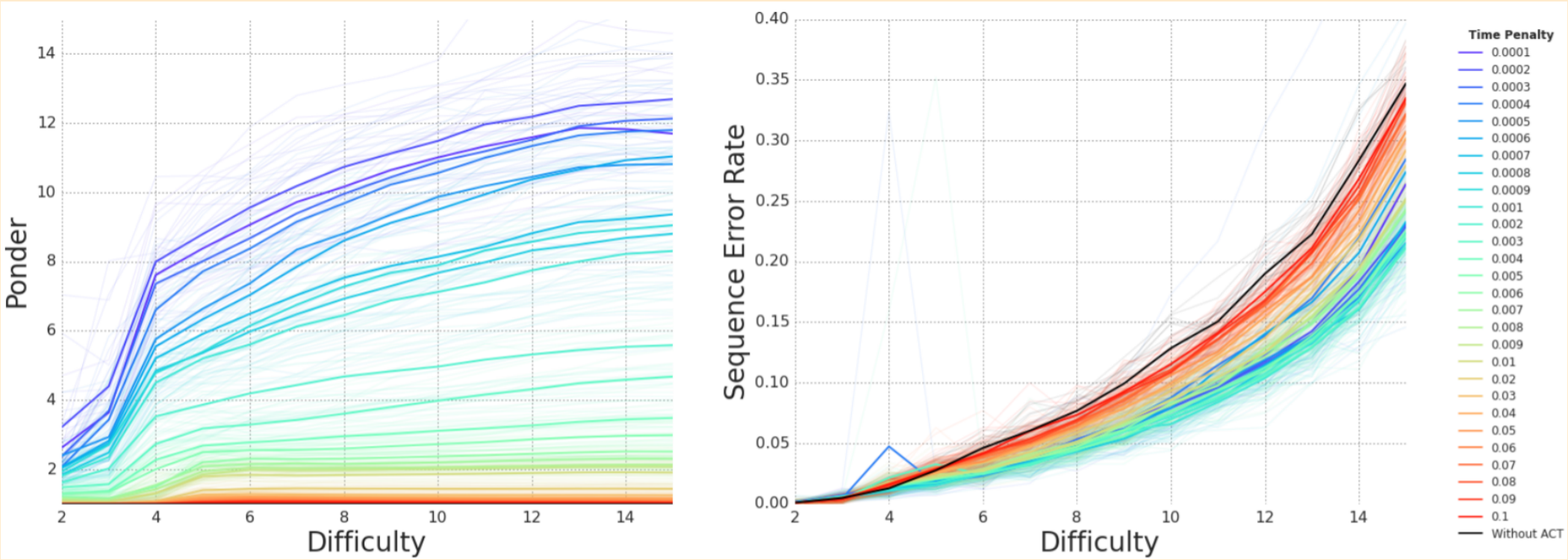
# Ponder




Sequence Error Rate
0.20
0.15
0.10
0.05
0
2
4
6
8
10
12
Ponder
Time Penalty
0.0001
0.0002
0.0003
0.0004
0.0005
0.0006
0.0007
0.0008
0.0009
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
Without ACT

# Ponder



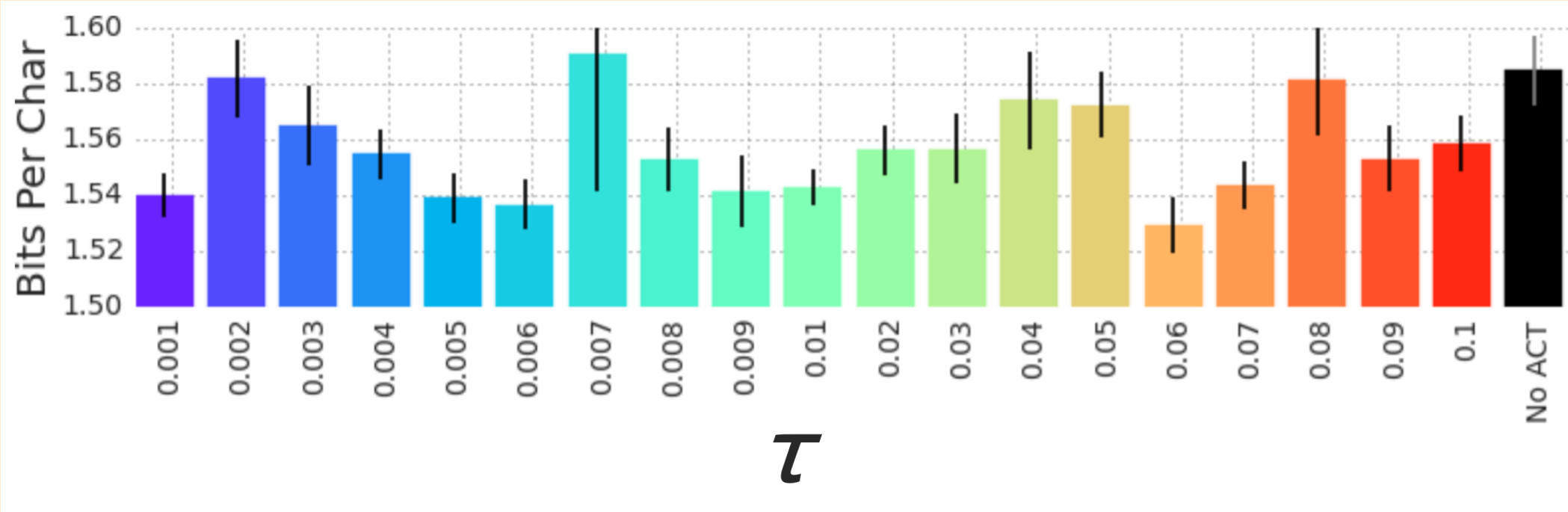

Ponder
Difficulty
Sequence Error Rate
Difficulty
Time Penalty
0.0001
0.0002
0.0003
0.0004
0.0005
0.0006
0.0007
0.0008
0.0009
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
Without ACT

# Wiki Char Pred

## Model：LSTM

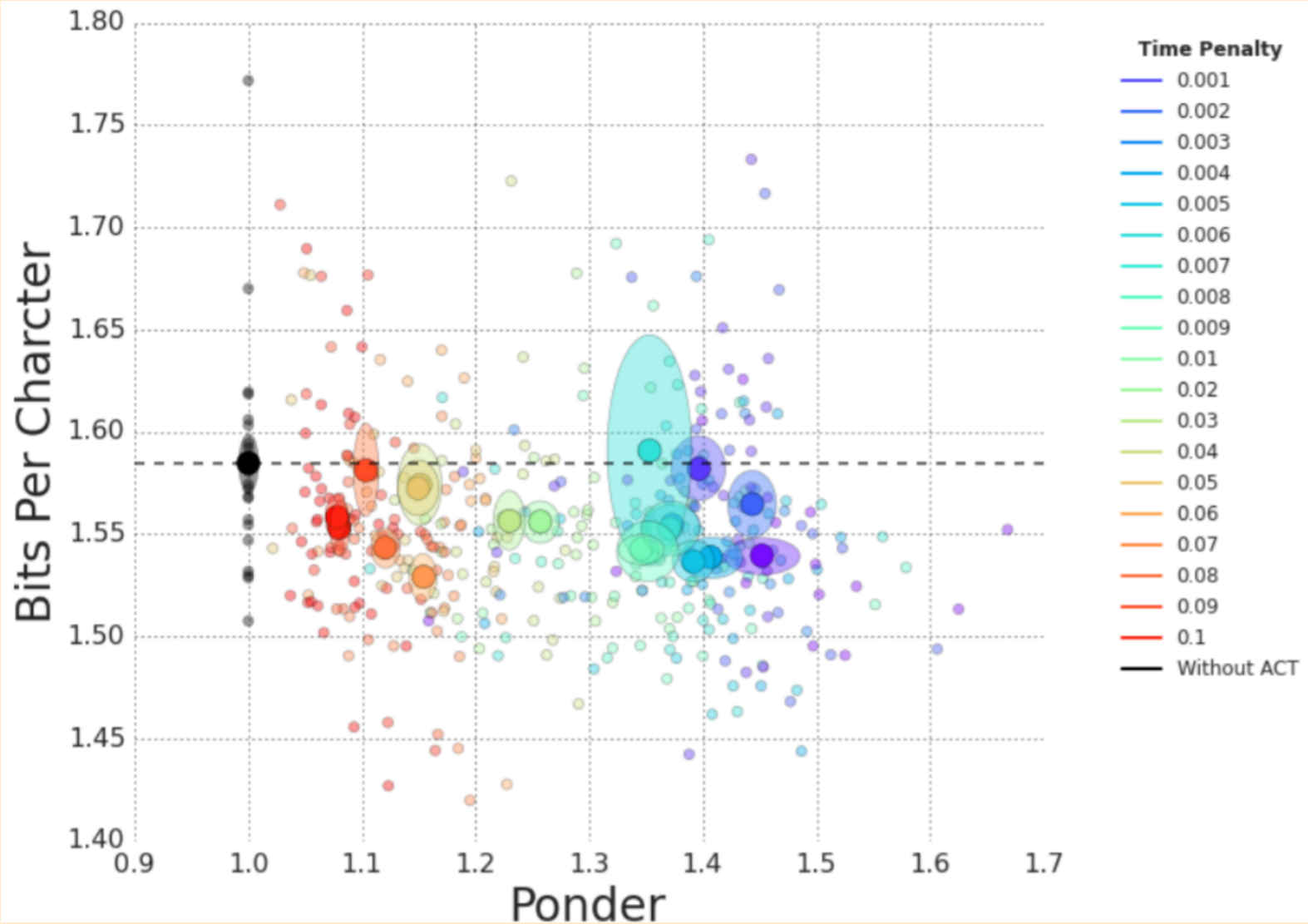
# Time Penalty





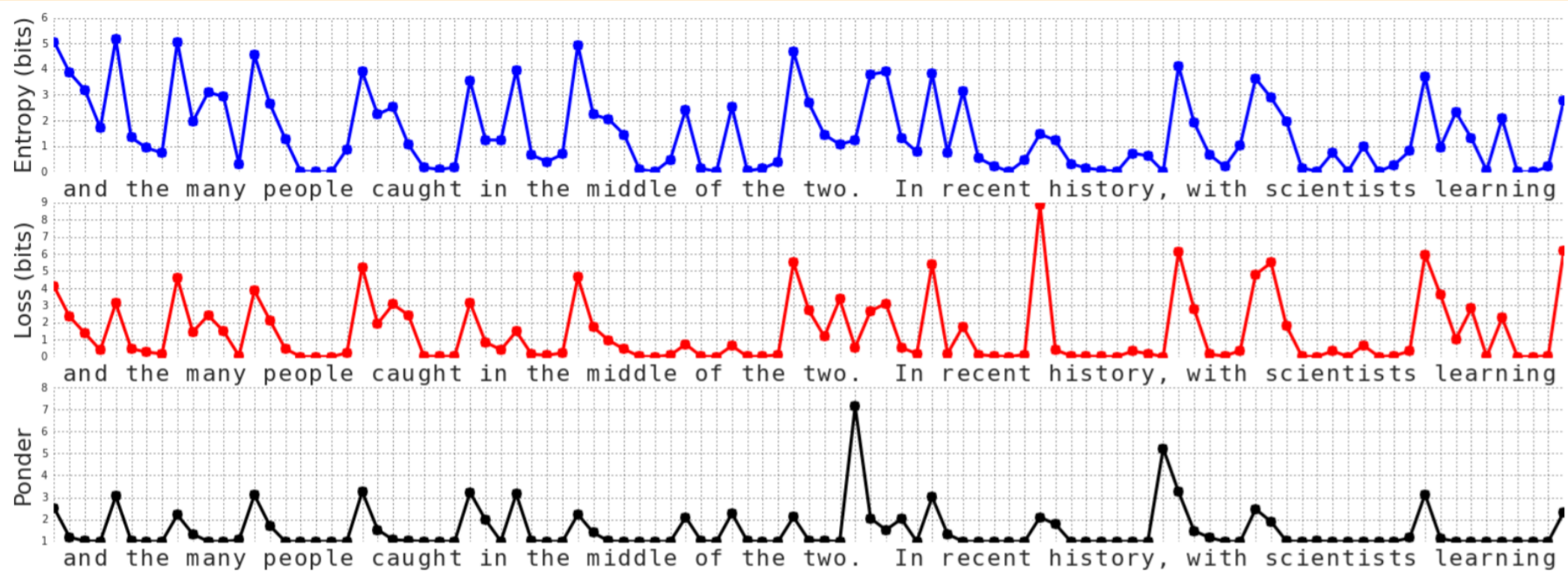
# Ponder




Bits Per Charcter
Ponder
1.80
1.75
1.70
1.65
1.60
1.55
1.50
1.45
1.40
0.9
1.0
1.1
1.2
1.3
1.4
1.5
1.6
1.7
Time Penalty
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
Without ACT

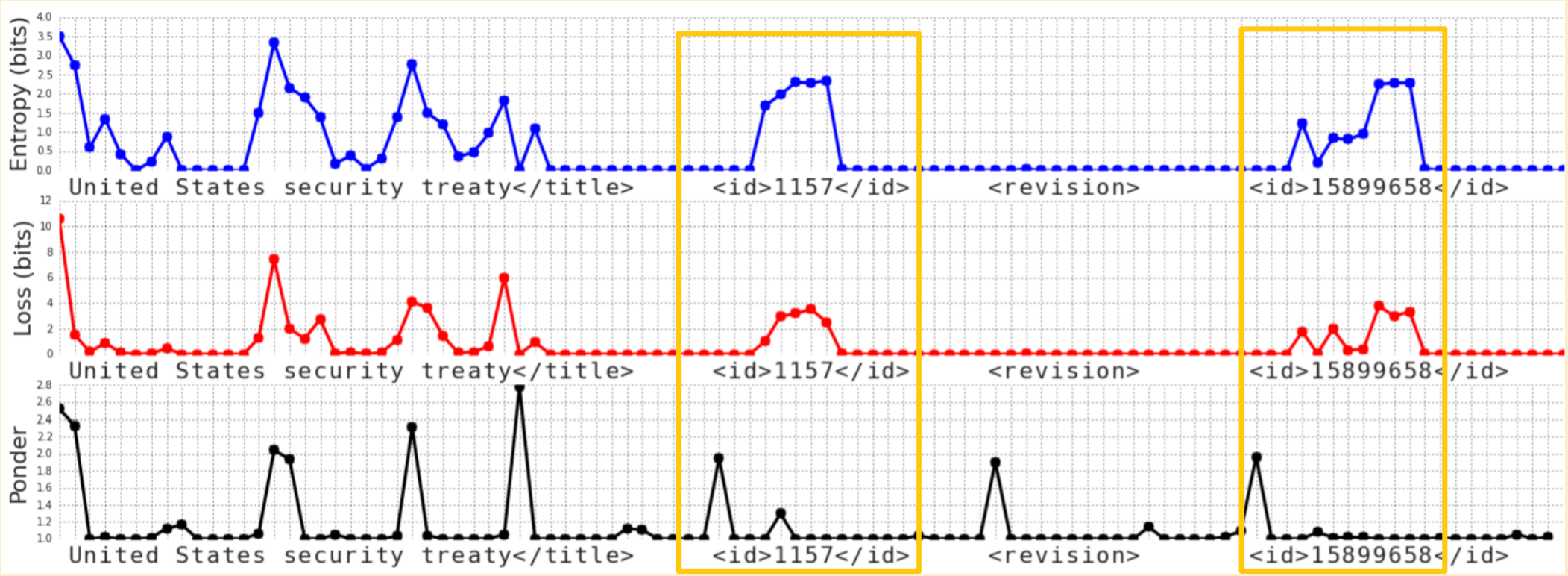
# Ponder




Entropy (bits)
and the many people caught in the middle of the two. In recent history, with scientists learning
Loss (bits)
and the many people caught in the middle of the two. In recent history, with scientists learning
Ponder
and the many people caught in the middle of the two. In recent history, with scientists learning

# Ponder




Entropy (bits)
Loss (bits)
Ponder
United States security treaty</title>
<id>1157</id>
<revision>
<id>15899658</id>

# Conclusion

- Allowing the model to increase or decrease the number of thoughts based on the complexity of the problem can indeed improve the effect.
- The current algorithm is very sensitive to time penalty($\tau$), and a better method is needed to automatically adjust time penalty.
- Ponder times may be used as a standard to distinguish structure and noise.