Turing Complete Neural Network based models

by **Wojciech Zaremba**

OpenAI

Need for powerful models

- Very complicated tasks require many computational steps
- Not all tasks can be solved by feed-forward network due to limited computational power

	Feed forward network	Classical CNN	CNN for detection	RNN	Neural Turing Machine
Input Size	$O(1)$	$O(1)$	$O(n)$	$O(n)$	$O(n)$
Number of steps	$O(1)$	$O(1)$	$O(n)$	$O(n)$	$O(n)$

More computation steps with the same number of parameters

- Reuse parameters extensively
- Few architectural choices:
    - Neural GPU;
        - Developed by Keiser et al. 2015
        - Further work by Price et al. (Summer internship at OpenAI)
    - RNN with RL (large part of my PhD)
    - Grid LSTM (Kalchbrenner et. al 2015)

Neural GPU

Neural GPU [Kaiser and Sutskever, 2015]

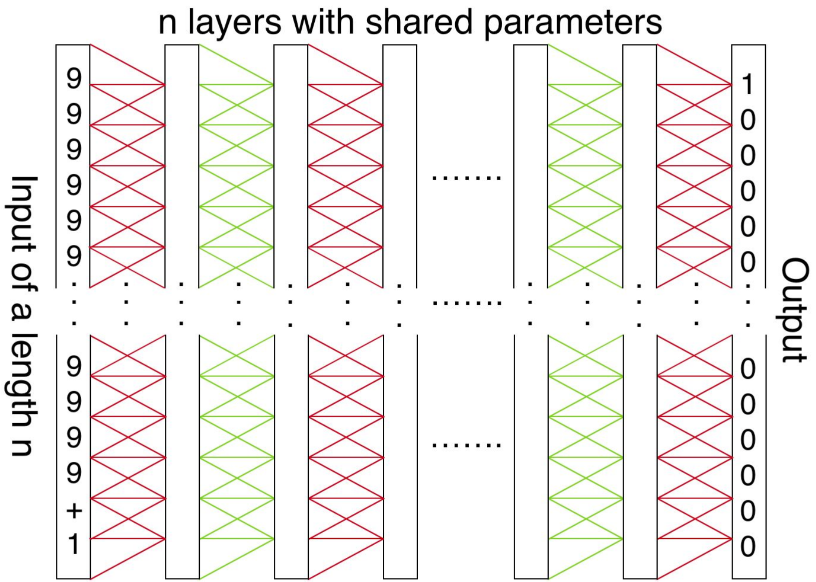
- The Neural GPU architecture learns arithmetic from examples.
- Feed in 6070124226526763509 0 + 4059459019222299864 3 get out 00000000000000000000101295832457490633733

Neural GPU [Kaiser and Sutskever, 2015]

- The Neural GPU architecture learns arithmetic from examples.
- Feed in 60701242265267635090 + 40594590192222998643 get out 00000000000000000000101295832457490633733
- Can generalize to longer examples
  - Train on up to 20-digit examples
  - Still gets > 99% of 200-digit examples right.
  - (If you get lucky on training) gets > 99% of 2000-digit examples right.

Neural GPU: architecture

- Alternates between two convolutional GRUs.
- If input has size n, does 2n total convolutions. [Need at least n to pass information from one side to the other]



Neural GPU: details

- Each digit is embedded into $1 \times 4 \times F$ space, where F is the number of "filters".
    - Input becomes $n \times 4 \times F$; convolution is 2D over the $n \times 4$.

Neural GPU: details

- Each digit is embedded into $1 \times 4 \times F$ space, where F is the number of “filters”.
  - Input becomes $n \times 4 \times F$; convolution is 2D over the $n \times 4$.
- Start with 12 different sets of weights, anneal down to only 2.

Neural GPU: details

- Each digit is embedded into $1 \times 4 \times F$ space, where F is the number of "filters".
  - Input becomes $n \times 4 \times F$; convolution is 2D over the $n \times 4$.
- Start with 12 different sets of weights, anneal down to only 2.
- Start learning single digit examples, extend length when good accuracy is achieved (< 15% errors).

Neural GPU: details

- Each digit is embedded into 1 × 4 × F space, where F is the number of “filters”.
  - Input becomes n×4×F; convolution is 2D over the n×4.
- Start with 12 different sets of weights, anneal down to only 2.
- Start learning single digit examples, extend length when good accuracy is achieved (< 15% errors).
- The sigmoid in the GRU has a cutoff, i.e. can fully saturate.

Neural GPU: details

- Each digit is embedded into $1 \times 4 \times F$ space, where F is the number of “filters”.
  - Input becomes $n \times 4 \times F$; convolution is 2D over the $n \times 4$.
- Start with 12 different sets of weights, anneal down to only 2.
- Start learning single digit examples, extend length when good accuracy is achieved (< 15% errors).
- The sigmoid in the GRU has a cutoff, i.e. can fully saturate.
- Dropout.

Neural GPU: Known Results

Problem	Base	24 filters	128 filters
Addition	2	Struggles	Works
	10	Fails	Works
Multiplication	2	Struggles	Works
	10	Fails	Fails

- Can we learn harder tasks?
  - What can we learn with bigger models?
  - What can we learn with smarter training?

Bigger models

- NeuralGPU barely fits into memory

- Bigger models require storing intermediate activations on CPU (tf.while_loop with swap memory options)

- Difficult to determine success due to huge non-determinism
  - Run large pool of experiments (once, we almost spent $0.5mln on them)

Bigger models

Problem	Base	24	128	256	512
Multiplication	2	Struggles	Works		
	10	Fails	Fails		

Bigger models

Problem	Base	24	128	256	512
Multiplication	2	Struggles	Works	Works	
	10	Fails	Fails	Fails	

Bigger models

Problem	Base	24	128	256	512
Multiplication	2	Struggles	Works	Works	Works
	10	Fails	Fails	Fails	Struggles?

How to do smarter training ?

- Extensive Curriculum
  - Curriculum through length (people used to do it)
  - Transfer from addition to multiplication doesn’t work
  - Transfer from small base to large seems to work

Bigger models and curriculum

Curriculum	128	256	512
10	Fails	Fails	Struggles?

Bigger models and curriculum

Curriculum	128	256	512
10	Fails	Fails	Struggles?
$2 \rightarrow 10$	Fails	Struggles	Struggles?

Bigger models and curriculum

Curriculum	128	256	512
10	Fails	Fails	Struggles?
$2 \to 10$	Fails	Struggles	Struggles?
$2 \to 5 \to 10$	Struggles	Works	Works?

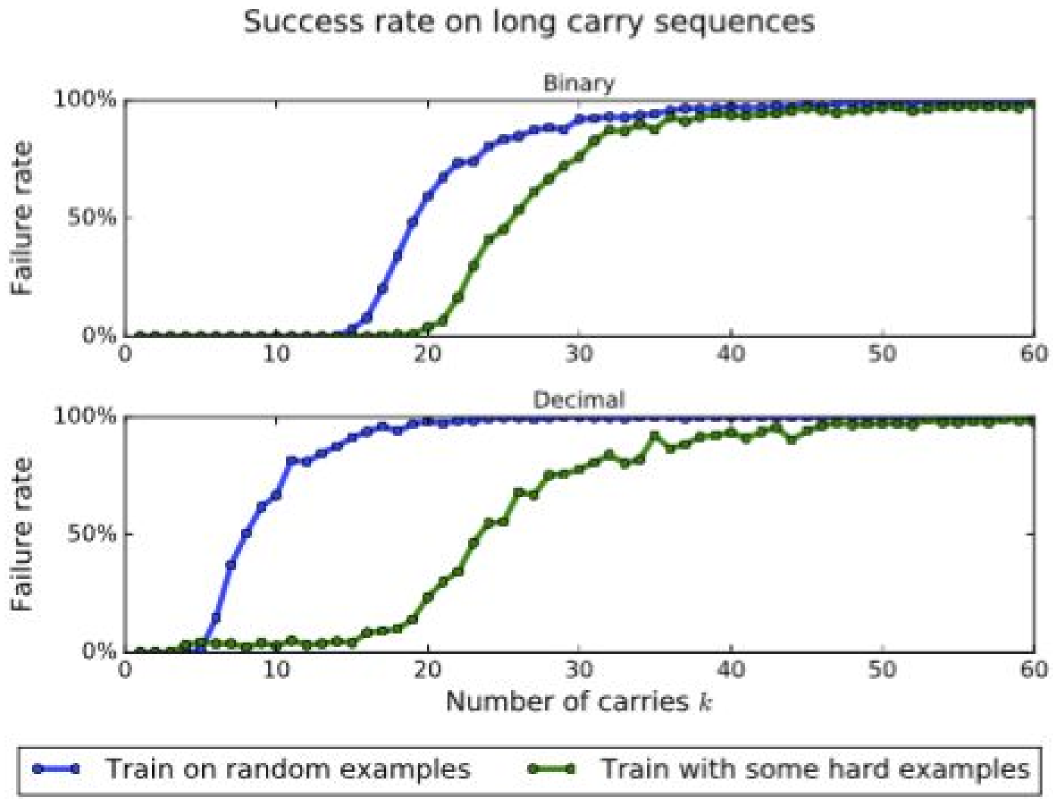
Bigger models and curriculum

Curriculum	128	256	512
10	Fails	Fails	Struggles?
$2 \to 10$	Fails	Struggles	Struggles?
$2 \to 5 \to 10$	Struggles	Works	Works?
$2 \to 4 \to 10$	Works	Works	Works?

Issues with neural GPU

- Trained on random inputs, it works reliably only on random inputs.
  - When doing addition, it cannot carry many bits.
  - Has issues with long stretches of similar digits.

Issues with carries



Issues with long similar stretches

- What is 59353073470806611971398236195285989083458222209939343360871730649133714199298764 × 71493004928584356509100241005385920385829595055047086568280792309308597157524754?

Issues with long similar stretches

- What is 593530734708066119713982361952859890834582222099393433608717306491337141992987 64 × 714930049285843565091002410053859203858295950550470865682807923093085971575247 54?
    - 424332957417500652862392857230327112302355162 72….1254256915245098421571902495277160 4056

Issues with long similar stretches

- What is 59353073470806611971398236195285989083458222209939343360871730649133714199298764 × 71493004928584356509100241005385920385829595055047086568280792309308597157524754?
    - 42433295741750065286239285723032711230235516272….12542569152450984215719024952771604056
- What is 2×1?

Issues with long similar stretches

- What is 59353073470806611971398236195285989083458222209939343360871730649133714199298764 × 71493004928584356509100241005385920385829595055047086568280792309308597157524754?
  - 42433295741750065286239285723032711230235516272….12542569152450984215719024952771604056
- What is 2×1?
  - 002

Issues with long similar stretches

- What is 59353073470806611971398236195285989083458222209939343360871730649133714199298764 × 71493004928584356509100241005385920385829595055047086568280792309308597157524754?
    - 42433295741750065286239285723032711230235516272….12542569152450984215719024952771604056
- What is 2×1?
    - 002
- What is 0000...0002 × 0000...0001

Issues with long similar stretches

- What is 59353073470806611971398236195285989083458222209939343360871730649133714199298764 × 71493004928584356509100241005385920385829595055047086568280792309308597157524754?
    - 42433295741750065286239285723032711230235516272….12542569152450984215719024952771604056
- What is 2×1?
    - 002
- What is 0000...0002 × 0000...0001
    - 0…..0017666666668850…..007

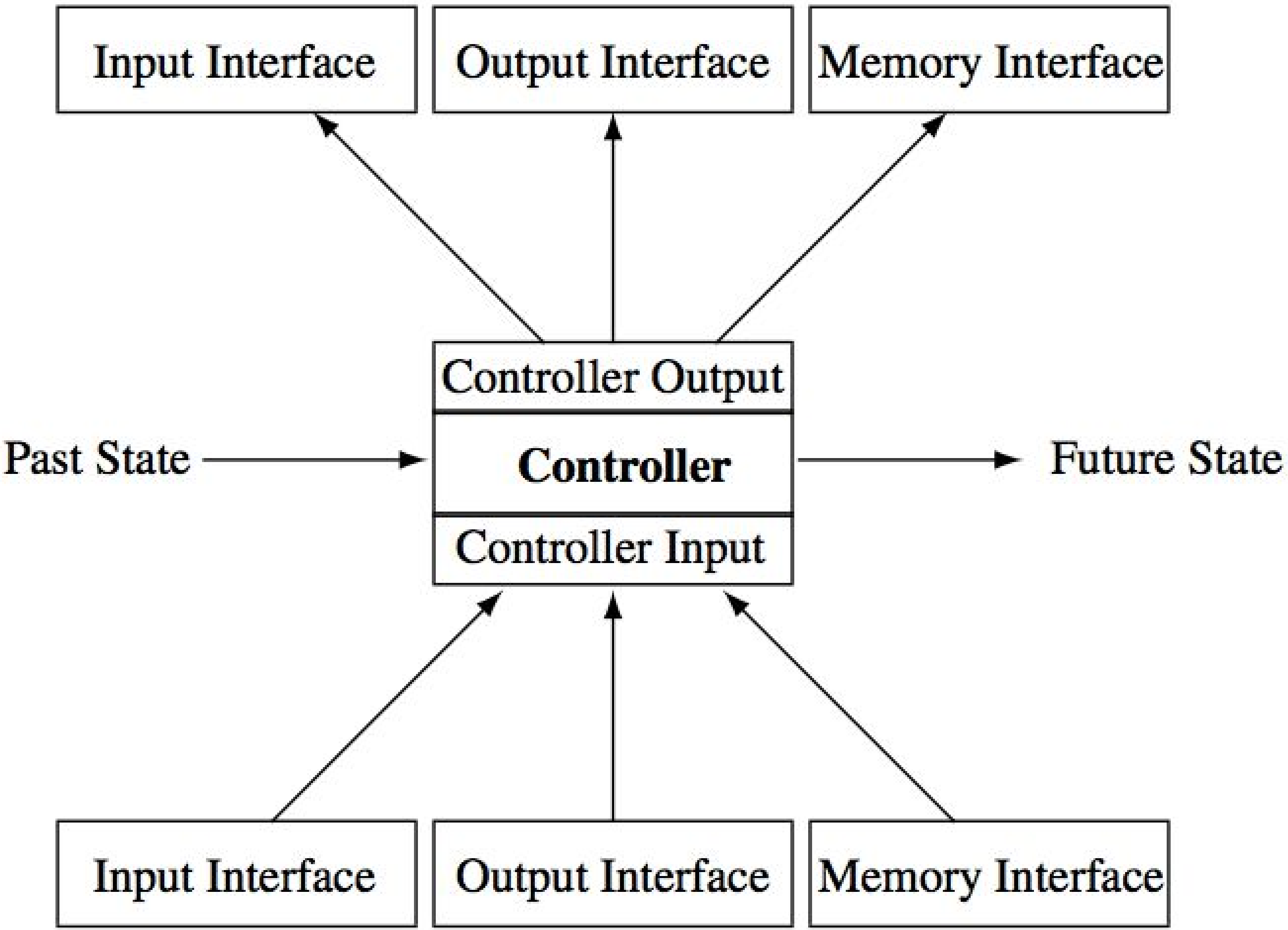
Issues with long similar stretches

- What is 59353073470806611971398236195285989083458222209939343360871730649133714199298764 × 71493004928584356509100241005385920385829595055047086568280792309308597157524754?
    - 42433295741750065286239285723032711230235516272….12542569152450984215719024952771604056
- What is 2×1?
    - 002
- What is 0000...0002 × 0000...0001
    - 0…..00176666666668850…..007
- What is 0000...0002 × 0000...0002

Issues with long similar stretches

- What is 59353073470806611971398236195285989083458222209939343360871730649133714199298764 × 71493004928584356509100241005385920385829595055047086568280792309308597157524754?
    - 42433295741750065286239285723032711230235516272….12542569152450984215719024952771604056
- What is 2×1?
    - 002
- What is 0000...0002 × 0000...0001
    - 0…..0017666666668850…..007
- What is 0000...0002 × 0000...0002
    - 0…..0017666666668850…..014

RNN with RL



Input Interface
Output Interface
Memory Interface
Controller Output
Past State
Controller
Future State
Controller Input
Input Interface
Output Interface
Memory Interface

Video

https://www.youtube.com/watch?v=GVe6kfJnRAw&feature=youtu.be

Q-learning

- Reward of 1 for every correct prediction, and 0 otherwise.
- Model trained with Q-learning
- Q(s, a) estimates sum of the future rewards for an action “a” in a state “s”.
- Q is the off-policy algorithm (remarkable)

$$Q_{t+1}(s,a) = Q_t(s,a) - \alpha\big[Q_t(s,a) - \big(R(s') + \gamma \max_a Q_n(s',a)\big)\big]$$

Q-learning as off-policy

- Policy induced by Q is the argmax_a Q(s, a)
- When we follow induced policy, we say that we are on-policy
- When we follow a different policy, we say that we are off-policy
- Q converges to Q for the optimal policy regardless of policy that we follow (as long as we can visit every state-action pair) !!!

Watkins Q(lambda)[11]

- Typical policy is a combination of on-policy (95%) with a random uniform policy (5%).
- Most of the time, we are on-policy
- This allows to regress Q on the other estimate:

$$Q^*(s_t, a_t) = \sum_{i=1}^{T} \gamma^{i-1} R(s_{t+i}) + \gamma^T \max_a Q^*(s_{t+n+1}, a)$$

[11] “Reinforcement learning: An introduction” Sutton and Barto

Dynamic Discount

- In Q-learning, the model has to predict the sum of future rewards.
- However, the length of the episode might vary.
- We reparametrize Q, so it estimates the sum of future rewards divided by number of predictions left: $$\hat{Q}(s,a) := \frac{Q(s,a)}{\hat{V}(s)}$$

Curriculum[4]

- Three row addition was unsolvable in the original form
- We start with small numbers that do not require carry.

$$\boxed{\begin{matrix}1\\2\\2\end{matrix}}\;;\;\boxed{\begin{matrix}2\\0\\2\end{matrix}}\;;\;\boxed{\begin{matrix}8&3\\3&3\\3&7\end{matrix}}\;;\;\boxed{\begin{matrix}3&2&0&6&9\\1&3&1&3&1\\2&8&0&8&3\end{matrix}}\;;\;\boxed{\begin{matrix}8&0&1&8&5&2&0&2&1\\1&3&1&4&0&7&0&5&4\\3&1&3&2&7&5&0&7&1\end{matrix}}$$

[4] "Curriculum learning.", Bengio et al.

Task											
	Test length	100	100	100	100	100	100	100	100	1000	1000
	#Units	600	400	200	200	200	200	200	200	200	200
	Discount γ	1	1	1	0.99	0.95	D	D	D	D	D
	Watkins $Q(\lambda)$	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓
Task	Penalty	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓
Copying		30%	60%	90%	50%	70%	90%	100%	100%	100%	100%
Reverse		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Reverse (FF controller)		0%	0%	0%	0%	0%	0%	100%	90%	100%	90%
Walk		0%	0%	0%	0%	0%	0%	10%	90%	10%	80%
Walk (FF controller)		0%	0%	0%	0%	0%	0%	100%	100%	100%	100%
2-row Addition		10%	70%	70%	70%	80%	60%	60%	100%	40%	100%
3-row Addition		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
3-row Addition (extra curriculum)		0%	50%	80%	40%	50%	50%	80%	80%	10%	60%
Single Digit Multiplication		0%	0%	0%	0%	0%	100%	100%	100%	0%	0%

Reinforce[12]

Objective of Reinforce:

$$\sum_{a_1,\ldots,a_n} p(a_1, \ldots, a_n|\theta) \sum_i r_i$$

we access it through sampling:

$$\mathbb{E}_a \sum_i r_i$$

[12] “Simple statistical gradient-following algorithms for connectionist reinforcement learning”, Williams

Reinforce

Derivative:

$$\sum_a p'(a|\theta) \sum_i r_i + \sum_a p(a|\theta) \sum_i r_i'$$

$$p' = p(\log p)'$$

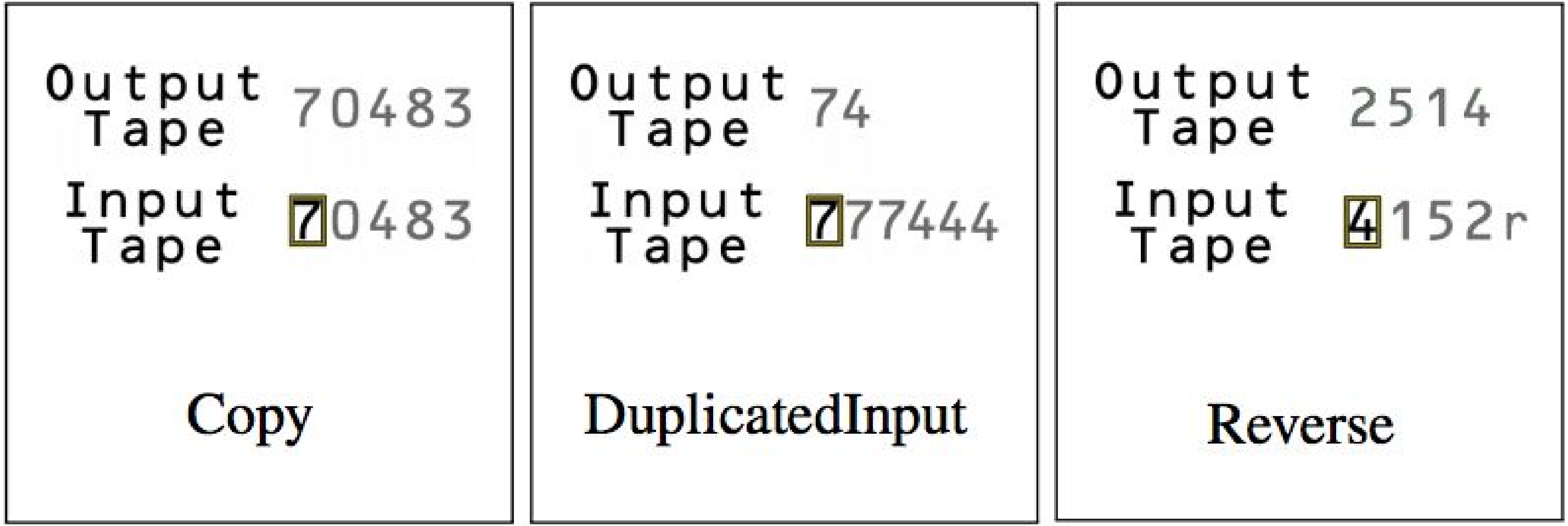
we access it through sampling:

$$\mathbb{E}_a \log p' \sum_i r_i + \sum_i r_i'$$

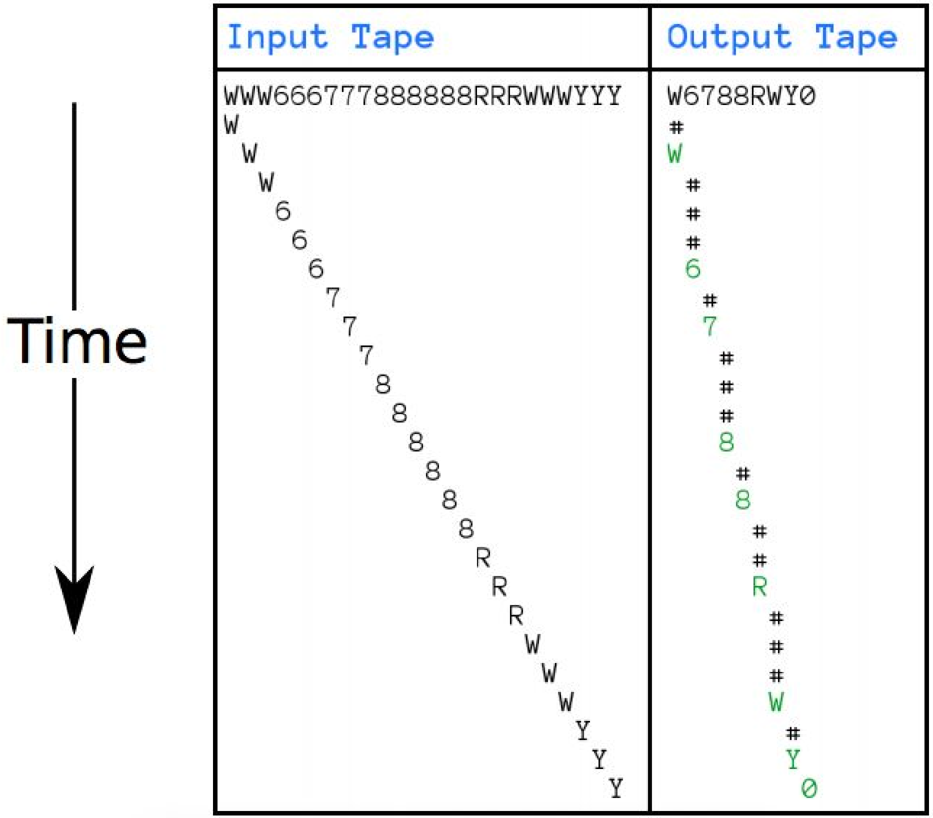
Training

- Trained with SGD
- Curriculum learning is critical
- Not easy to train (due to variance coming from sampling)
  - Various techniques to decrease variance[13]

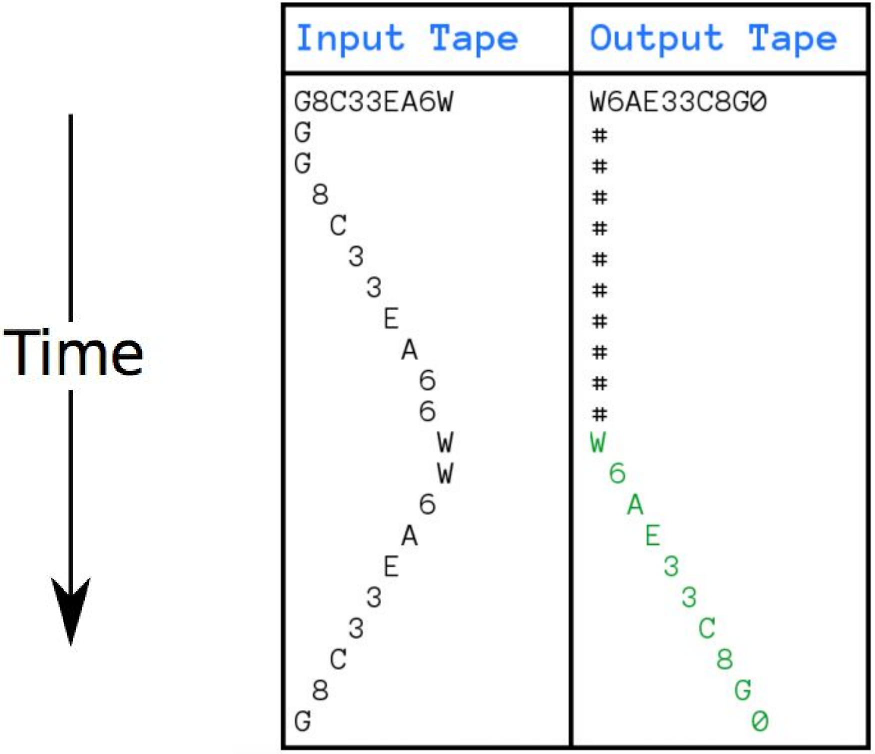
[13] “Policy Gradient Methods for Robotics” Peters and Schaal


Output Tape
70483
Input Tape
70483
Copy
Output Tape
74
Input Tape
777444
DuplicatedInput
Output Tape
2514
Input Tape
4152r
Reverse

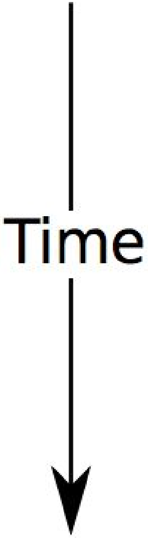
Task - DuplicatedInput

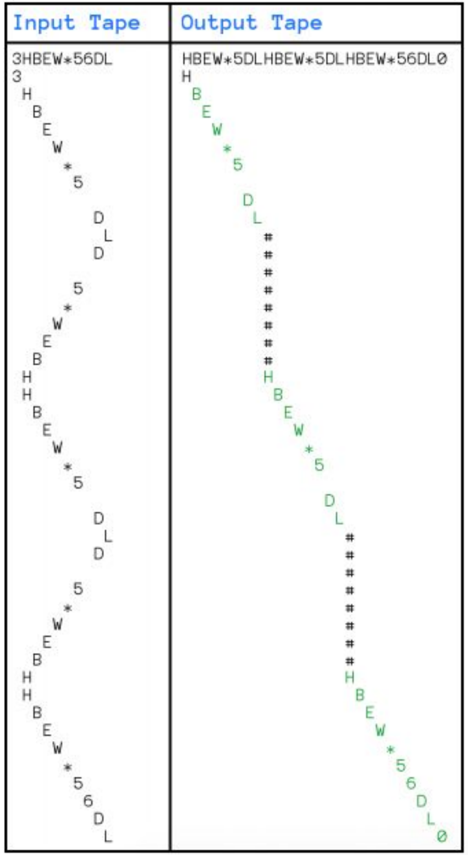


Task - Reverse



Task - RepeatCopy

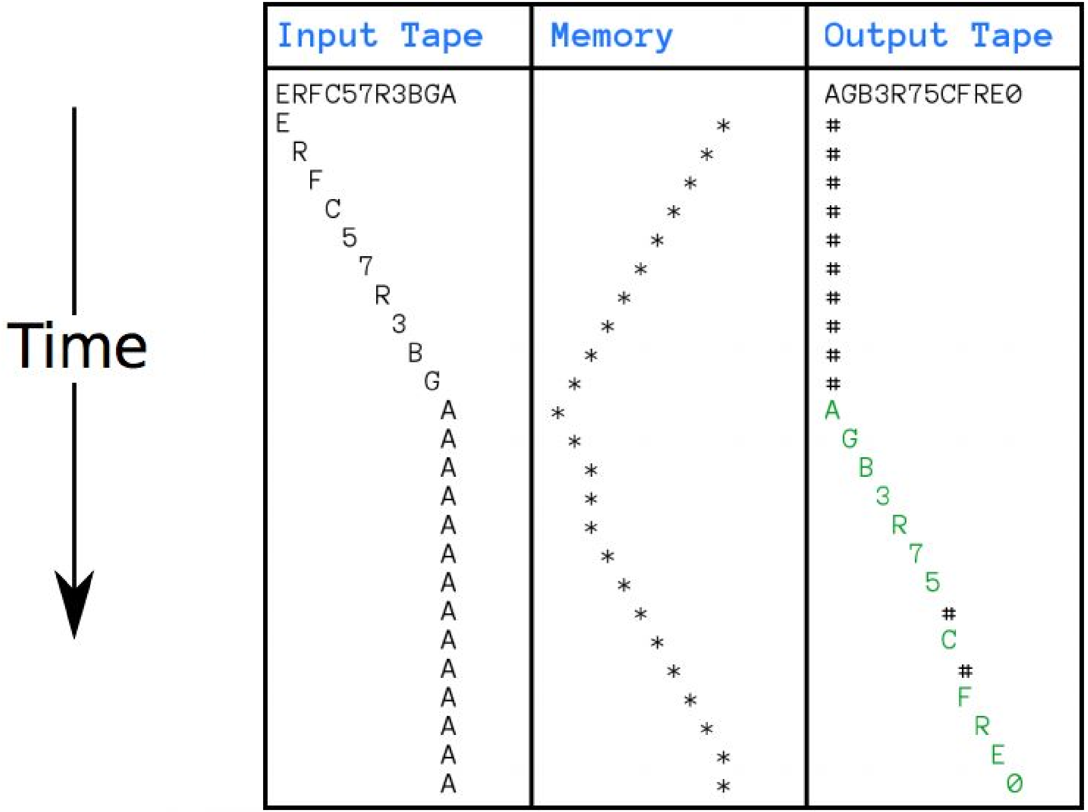




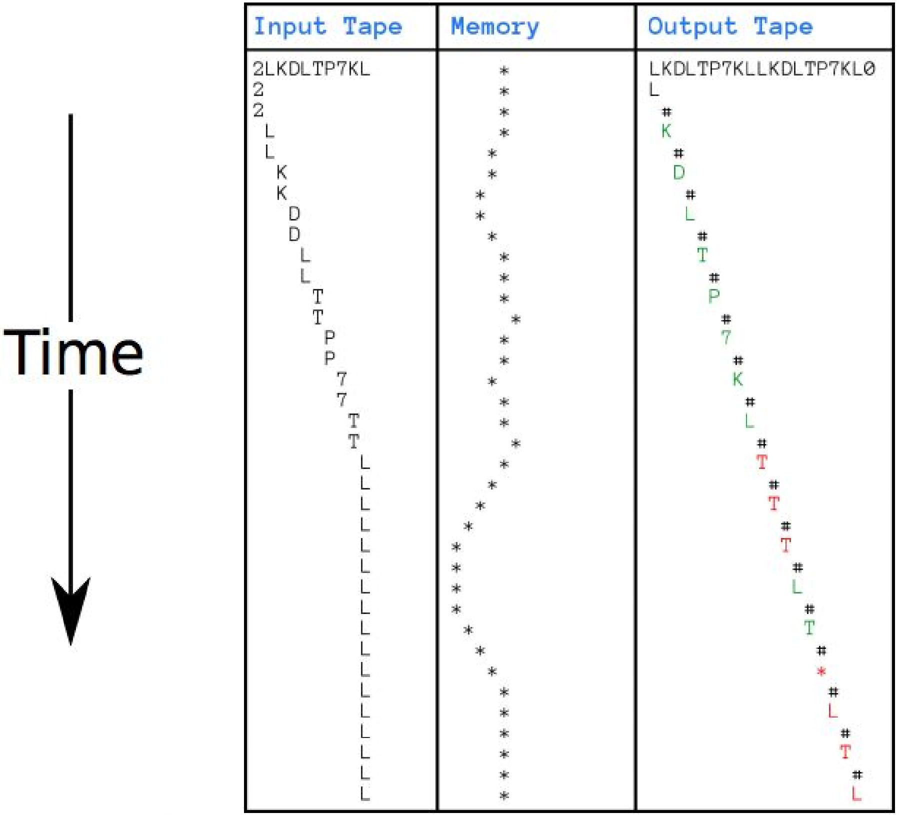
Memory interface

- Memory is a tape with 3 actions, go to the left, stay, go to the right
- Controller always reads from the previous memory location, and always saves to the next memory location
- It stores a high dimensional vector through which we backpropagate

Task - Reverse with memory



Task. RepeatCopy with memory. Failure



Gradient Checking - motivation

- Very simple to make a mistake in the implementation

- How to verify a stochastic algorithm?

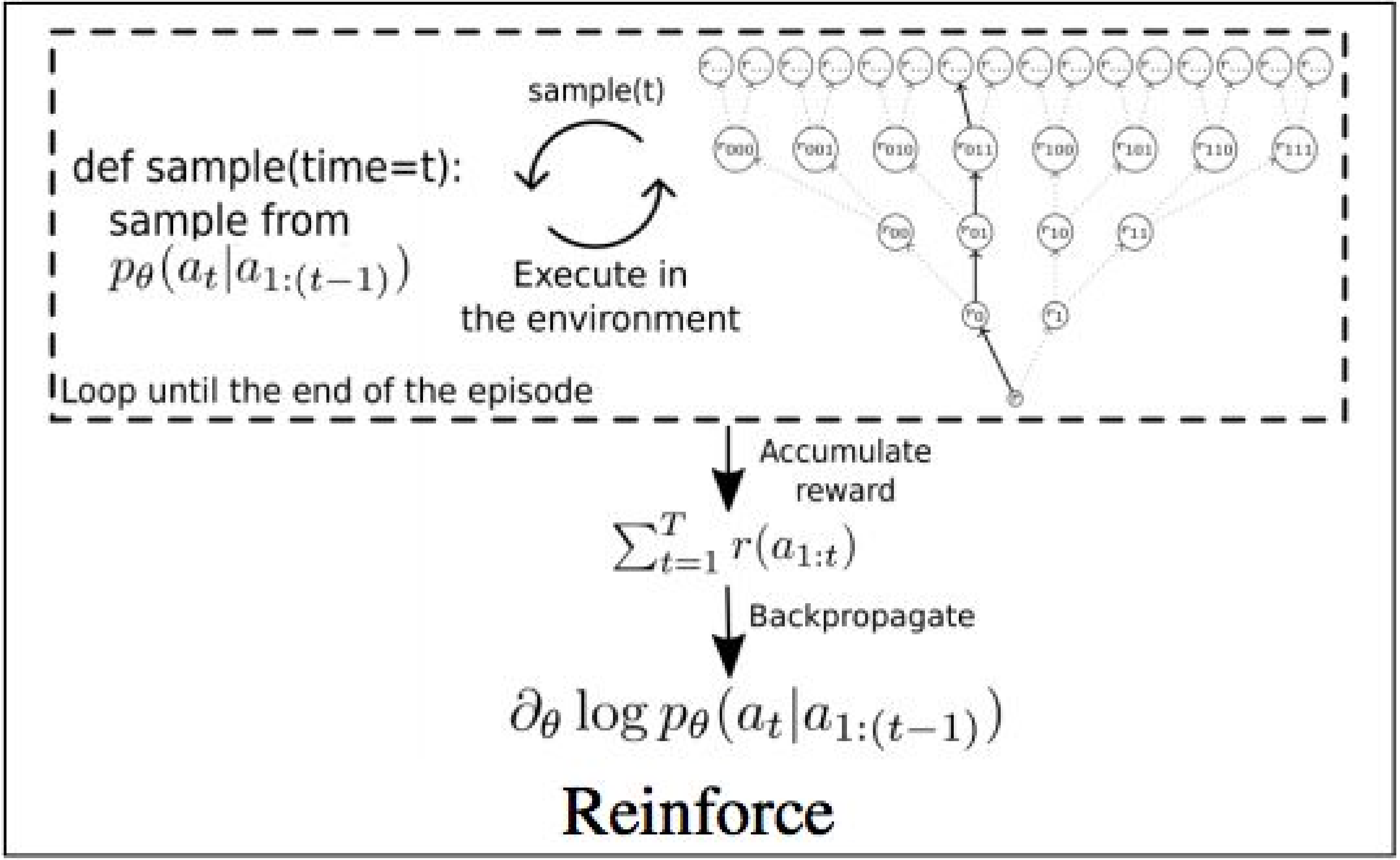
Gradient Checking for Reinforce

- We could sample actions many times and compare the average gradient to average of the numerical gradient.

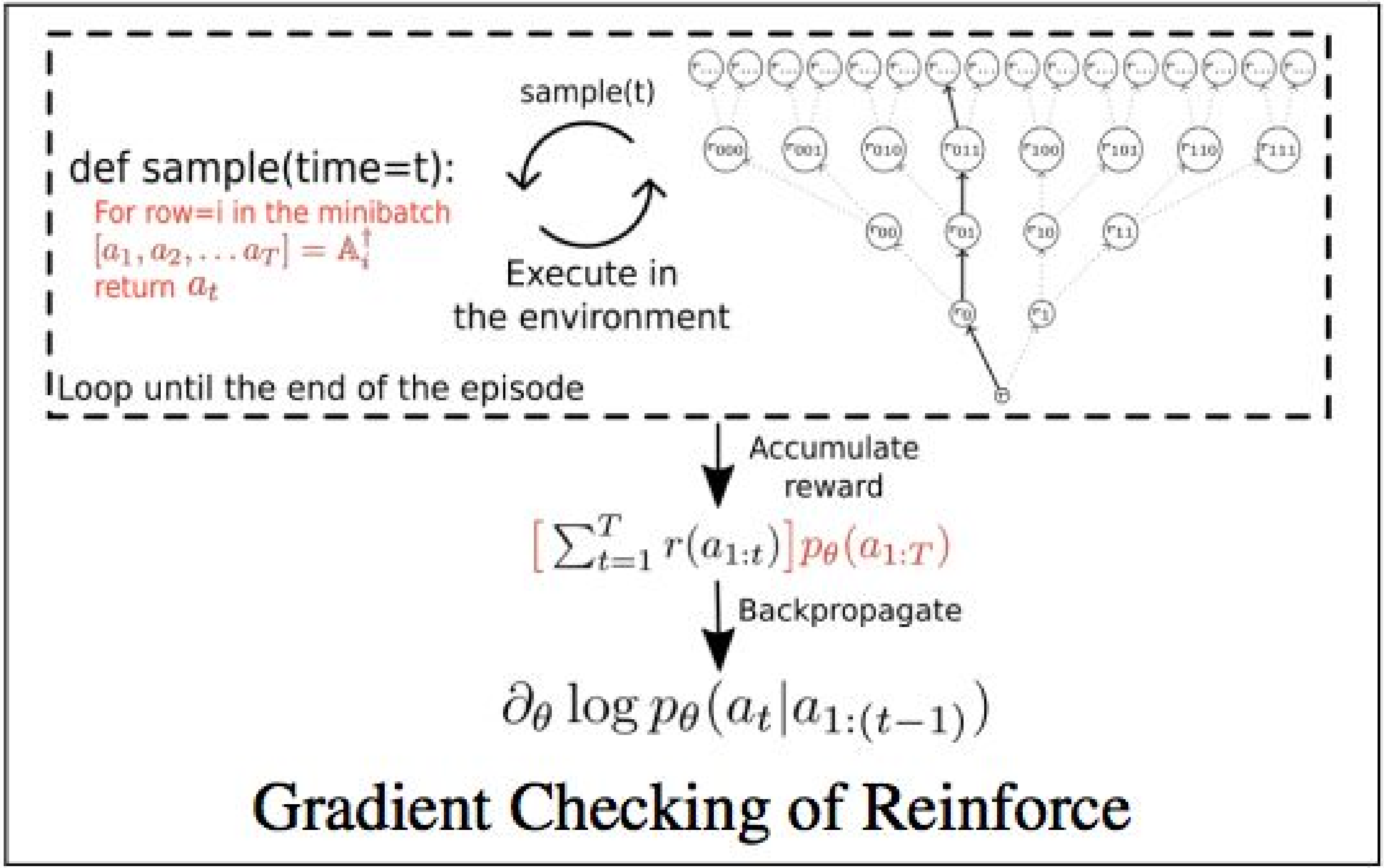
Gradient Checking for Reinforce

- We could sample actions many times and compare the average gradient to average of the numerical gradient.

- Impractical. To get good precision we would need millions of samples.



def sample(time=t):
sample from
$p_\theta(a_t|a_{1:(t-1)})$
sample(t)
Execute in
the environment
Loop until the end of the episode
Accumulate
reward
$\sum_{t=1}^{T} r(a_{1:t})$
Backpropagate
$\partial_\theta \log p_\theta(a_t|a_{1:(t-1)})$
Reinforce



def sample(time=t):
For row=i in the minibatch
$[a_1, a_2, \ldots a_T] = \mathbb{A}_i^{\dagger}$
return a_t
sample(t)
Execute in the environment
Loop until the end of the episode
Accumulate reward
$\left[\sum_{t=1}^{T} r(a_{1:t})\right] p_\theta(a_{1:T})$
Backpropagate
$\partial_\theta \log p_\theta(a_t | a_{1:(t-1)})$
Gradient Checking of Reinforce

Gradient Checking for Reinforce

- It was critical to make the model work.
- We can limit the size of the action space during gradient checking
- Gradient checking takes seconds

Q&A

- NeuralGPU
- Bigger -> better
- Curriculum
- Adversarial examples for NeuralGPU
- Q-learning
  - Dynamic discount
  - Watkins Q(lambda)
- Reinforce
- Memory
- Gradient checking

Thanks to Eric Price, Ilya Sutskever and Rob Fergus