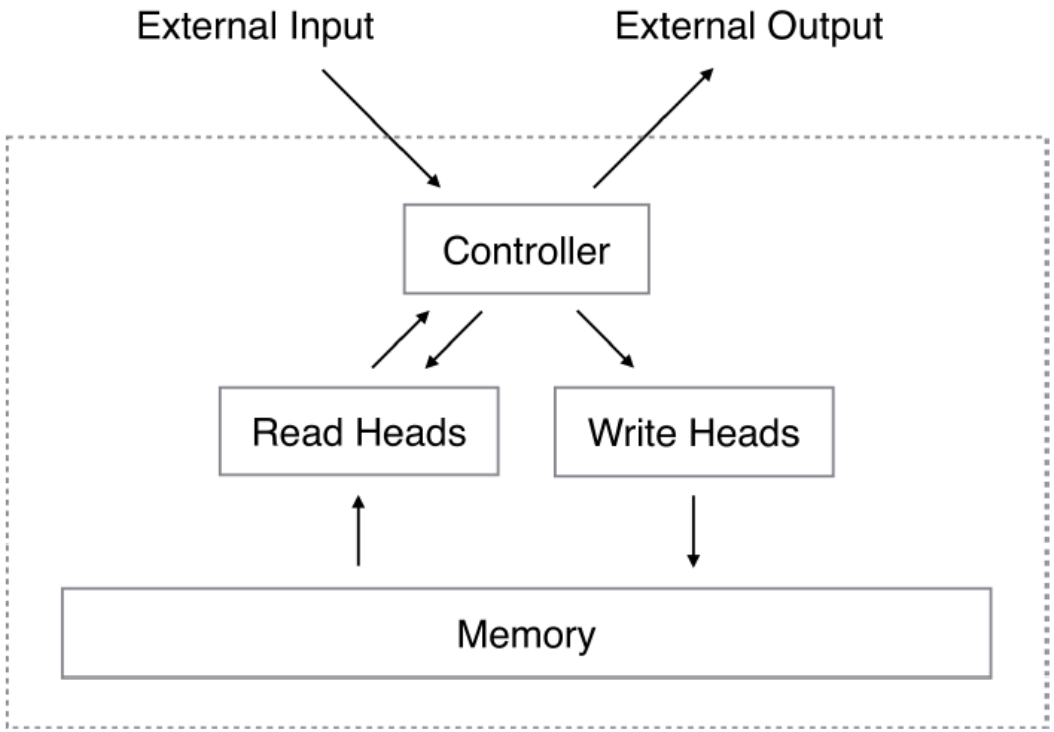
MLG Reading group, April 2, 2017
Minseop Park

Neural GPU

Neural GPUs learn algorithm (Lukasz Kaiser, Ilya Sutskever, ICLR 2016)

Can Active Memory replace attention (Lukasz Kaiser, Samy Bengio, NIPS 2017)

Inspired by Neural Turing Machine



Key features

End-to-end learnable memory network

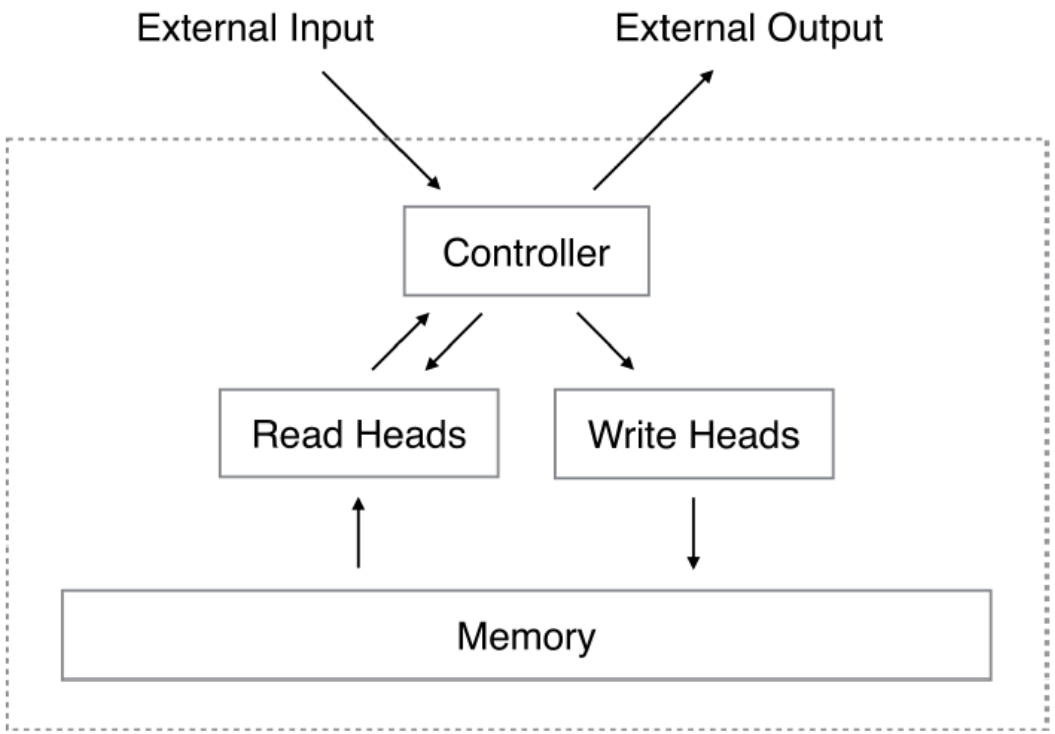
Algorithmic tasks : copy, repeat copy, sorting
-> to be Generalized well!!

Shortcomings

Soft attention on entire memory at each step

Sequential input

Inspired by Neural Turing Machine



Key features

End-to-end learnable memory network

Algorithmic tasks : copy, repeat copy, sorting
-> to be Generalized well!!

Shortcomings

Soft attention on entire memory at each step

Sequential input

Parallel Structure

Active Memory Model

Soft Attention to Active Memory

Soft Attention :

Not easy to learn procedures

Inefficient : Designing attention mechanism, parameterizing it

Only one activated memory

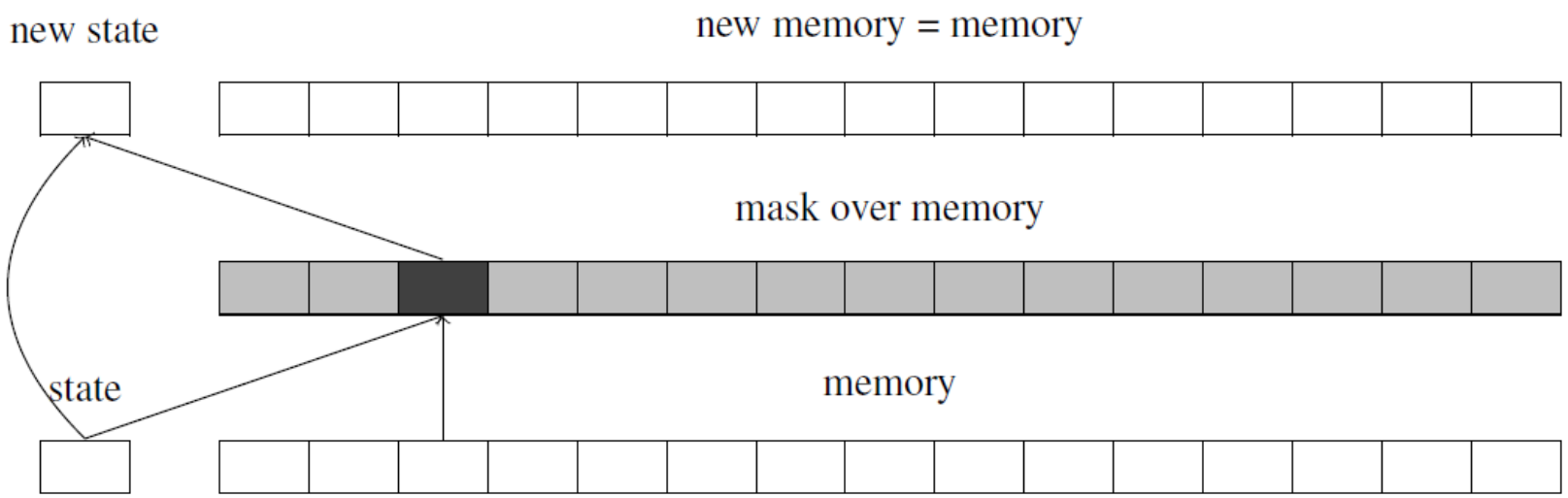


Figure 1: Attention model. The state vector is used to compute a probability distribution over memory. Weighted average of memory elements, with focus on one of them, is used to compute the new state.

Soft Attention to Active Memory

Active Memory Model :

1-D convolution
on memory

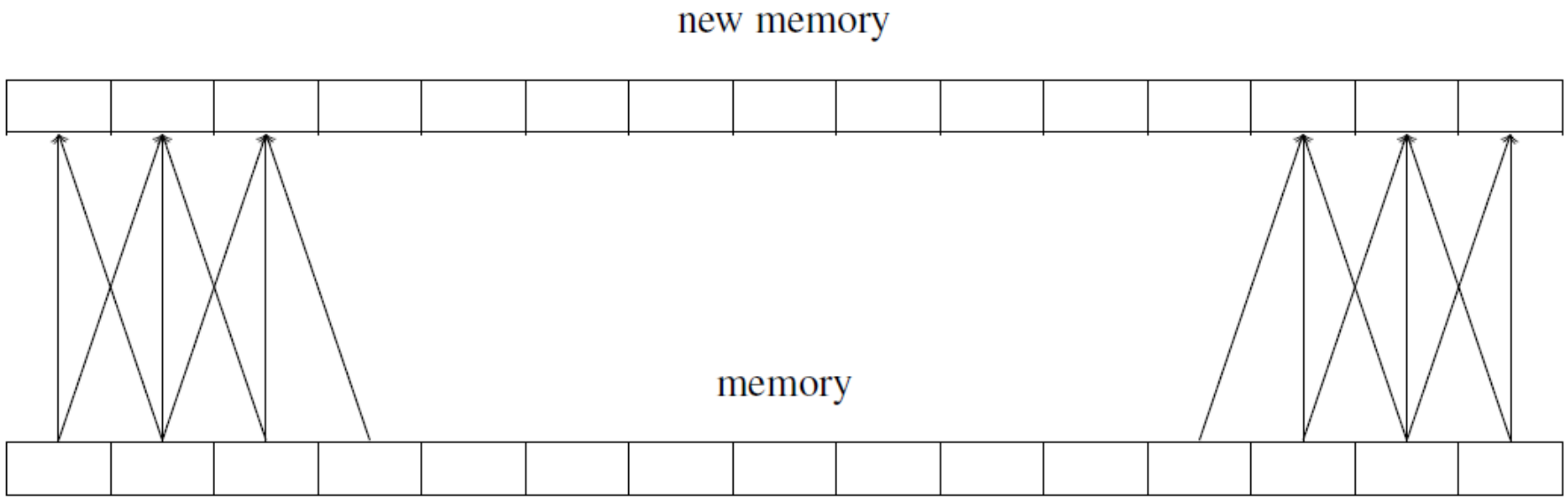


Figure 2: Active memory model. The whole memory takes part in the computation at every step. Each element of memory is active and changes in a uniform way, e.g., using a convolution.

Neural GPU architecture

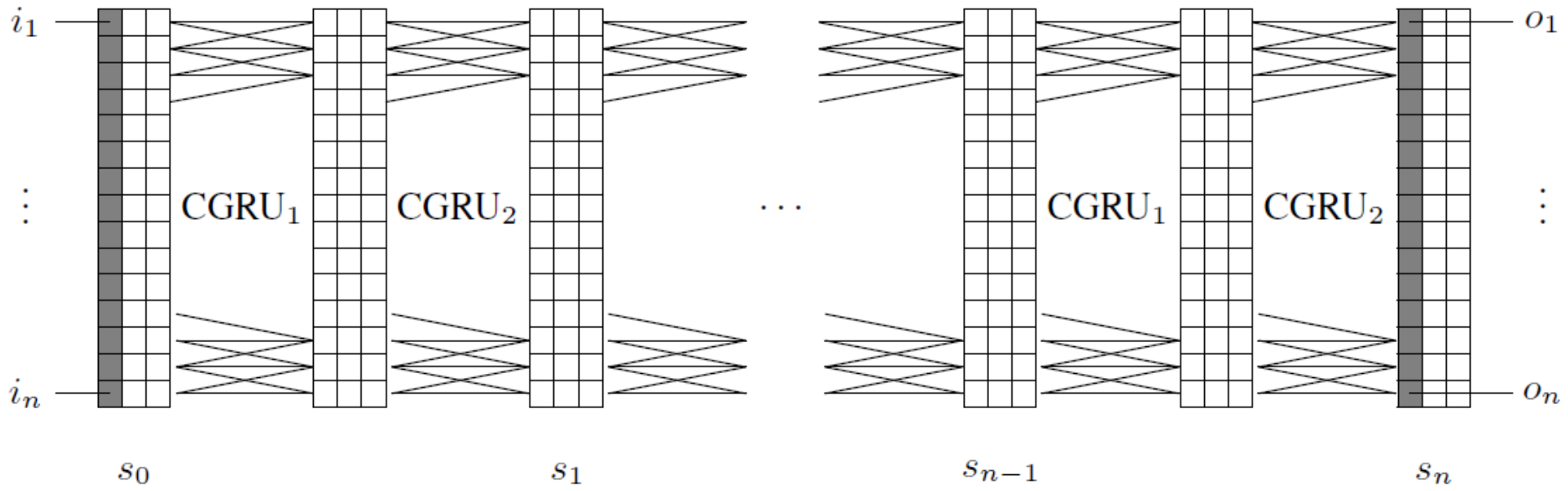


Figure 3: Neural GPU with 2 layers and width $w = 3$ unfolded in time.

$$s_{t+1} = \mathrm{CGRU}_l(\mathrm{CGRU}_{l-1} \ldots \mathrm{CGRU}_1(s_t) \ldots) \quad \text{and} \quad s_{\text{fin}} = s_n.$$

Convolutional GRU

$$\mathrm{GRU}(x, s) \quad = \quad u \odot s + (1 - u) \odot \tanh(Wx + U(r \odot s) + B), \quad \text{where}$$
$$u = \sigma(W'x + U's + B') \quad \text{and} \quad r = \sigma(W''x + U''s + B'').$$

u : update gate, r : reset gate, x: input, s: state (memory)

GRU has 1 less gate than LSTM

$$\mathrm{CGRU}(s) \quad = \quad u \odot s + (1 - u) \odot \tanh(U * (r \odot s) + B), \quad \text{where}$$
$$u = \sigma(U' * s + B') \quad \text{and} \quad r = \sigma(U'' * s + B'').$$

Training technique

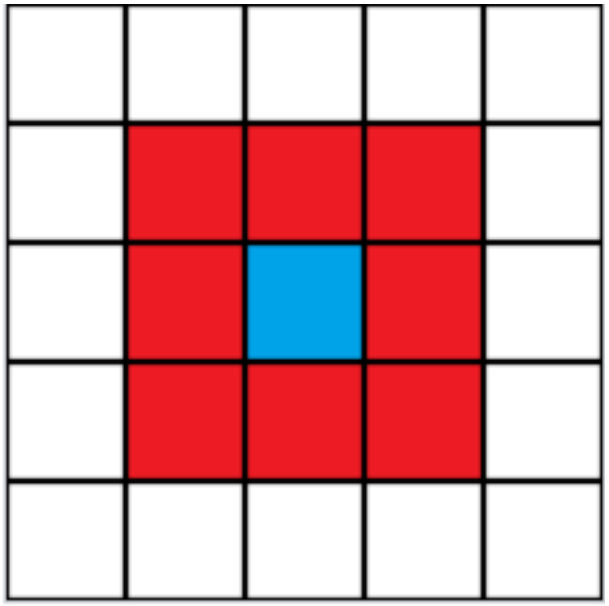
Curriculum Learning : train 7-digit number only after crossing a curriculum progress threshold on 6 digit numbers

Gradient Noise : add noise to gradient drawn from the normal distribution with mean 0 and variance inversely proportional to the square root of step number

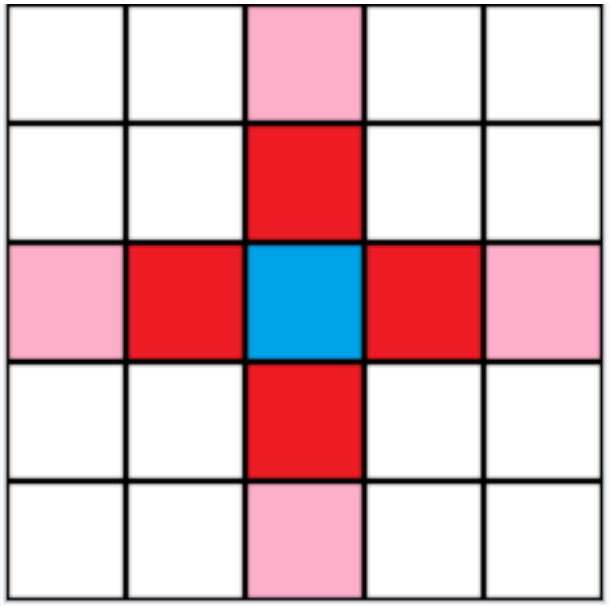
Gate Cutoff : use $\sigma'(x) = \max(0, \min(1, 1.2\sigma(x) - 0.1))$

Dropout : small dropout rate (6%~13%) helps

Connection to Cellular Automata



The red cells are the Moore neighborhood for the blue cell.



The red cells are the von Neumann neighborhood for the blue cell. The extended neighborhood includes the pink cells as well.

2-D cellular automaton

- Grid of Cells
- Cell : (state, neighborhood)
- Cell state at t = $f(C_t, N_{t-1})$

Results : Neural GPU

Task@Bits	Neural GPU	stackRNN	LSTM+A
badd@20	100%	100%	100%
badd@25	100%	100%	73%
badd@100	100%	88%	0%
badd@200	100%	0%	0%
badd@2000	100%	0%	0%
bmul@20	100%	N/A	0%
bmul@25	100%	N/A	0%
bmul@200	100%	N/A	0%
bmul@2000	100%	N/A	0%

Table 1: Neural GPU, stackRNN, and LSTM+A results on addition and multiplication. The table shows the fraction of test cases for which every single bit of the model's output is correct.

Extended Neural GPU

- Neural GPU was good at algorithmic task
- But not good at real world problem : Machine translation
- Reason ? Output y only depends on the hidden state
- Make **Output decoder** similar to Seq2seq model

Extended Neural GPU architecture

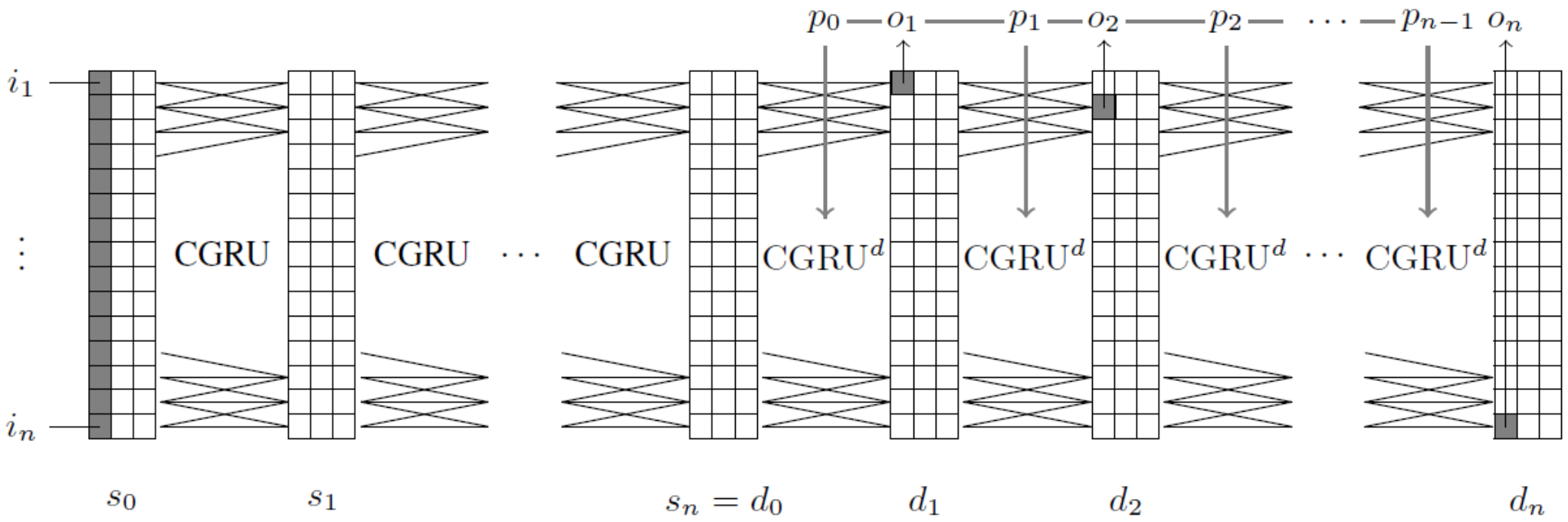


Figure 5: Extended Neural GPU with active memory decoder. See the text below for definition.

Extended Neural GPU

Skip connection : $s' = s + U * s$

Decoding step : $d_{t+1} = \mathrm{CGRU}_l^d(\mathrm{CGRU}_{l-1}^d(\ldots \mathrm{CGRU}_1^d(d_t, p_t)\ldots, p_t), p_t)$

$$\mathrm{CGRU}^d(s,p) = u \odot s + (1-u) \odot \tanh(U * (r \odot s) + \boldsymbol{W} * \boldsymbol{p} + B), \text{ where}$$
$$u = \sigma(U' * s + \boldsymbol{W'} * \boldsymbol{p} + B') \quad \text{and} \quad r = \sigma(U'' * s + \boldsymbol{W''} * \boldsymbol{p} + B'').$$

where $p_{k+1} = p_k$ with $p_k[0, k, :] \leftarrow E' o_k$.

Result : Extended NGPU

Model	Perplexity (log)	BLEU
Neural GPU	30.1 (3.5)	< 5
Markovian Neural GPU	11.8 (2.5)	< 5
Extended Neural GPU	3.3 (1.19)	**29.6**
GRU+Attention	3.4 (1.22)	26.4

Table 1: Results on the WMT English->French translation task. We provide the average per-word perplexity (and its logarithm in parenthesis) and the BLEU score. Perplexity is computed on the test set with the ground truth provided, so it do not depend on the decoder.

Thank you