

JOHANNES KILIAN LANGER

---

MASTER THESIS DISPUTATION

## TABLE OF CONTENTS

---

**ORIENTATION**

**RECURRENT NEURAL NETWORKS**

**OBJECTIVE**

**BAND MATRIX RESTRICTION**

**IMPLEMENTATION**

**EXPERIMENTS**

**DISCUSSION**

# Machine Learning

Supervised

Reinforcement

Unsupervised

Deep Learning

Recurrent Neural Networks

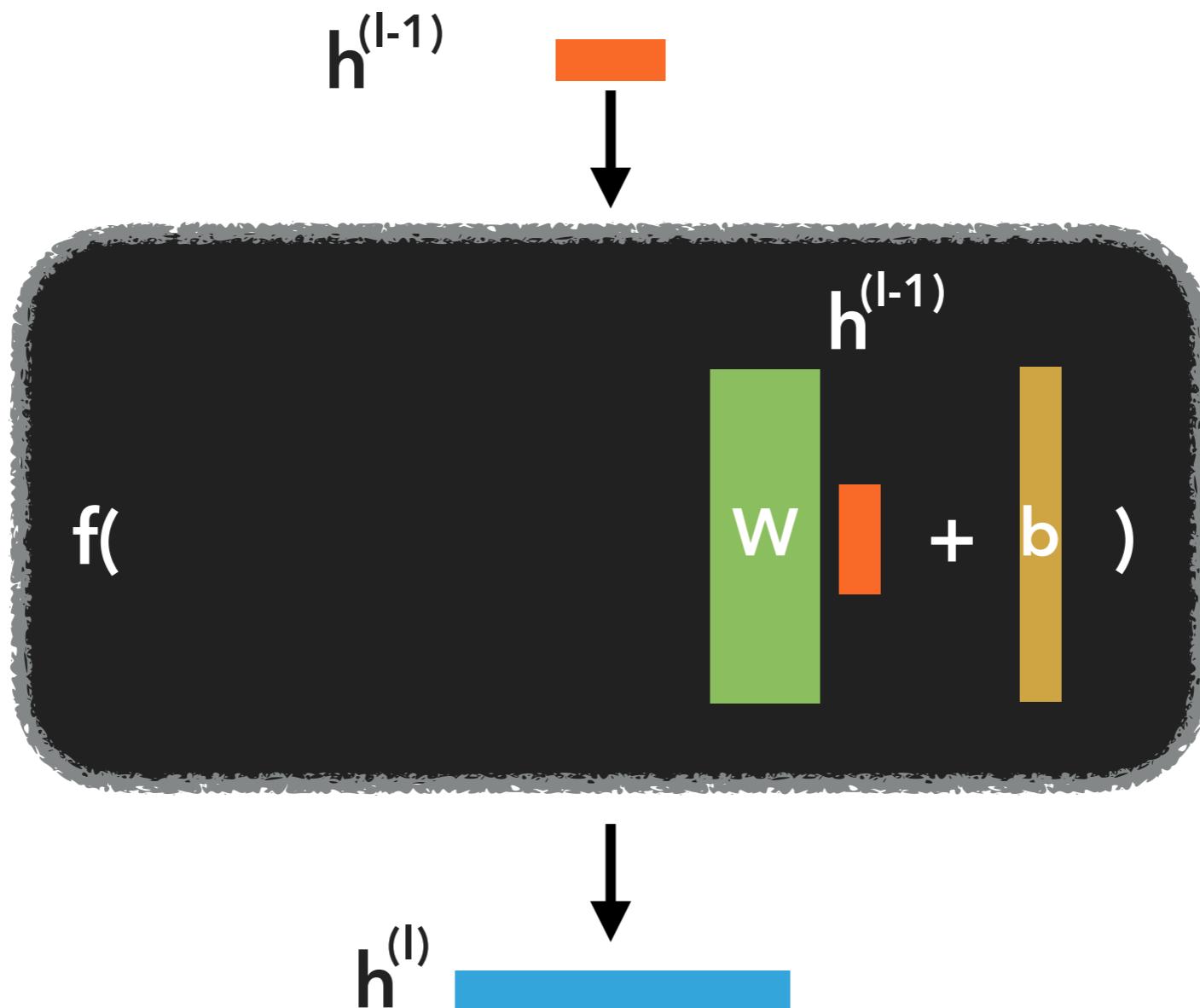


## RECURRENT NEURAL NETWORKS

---

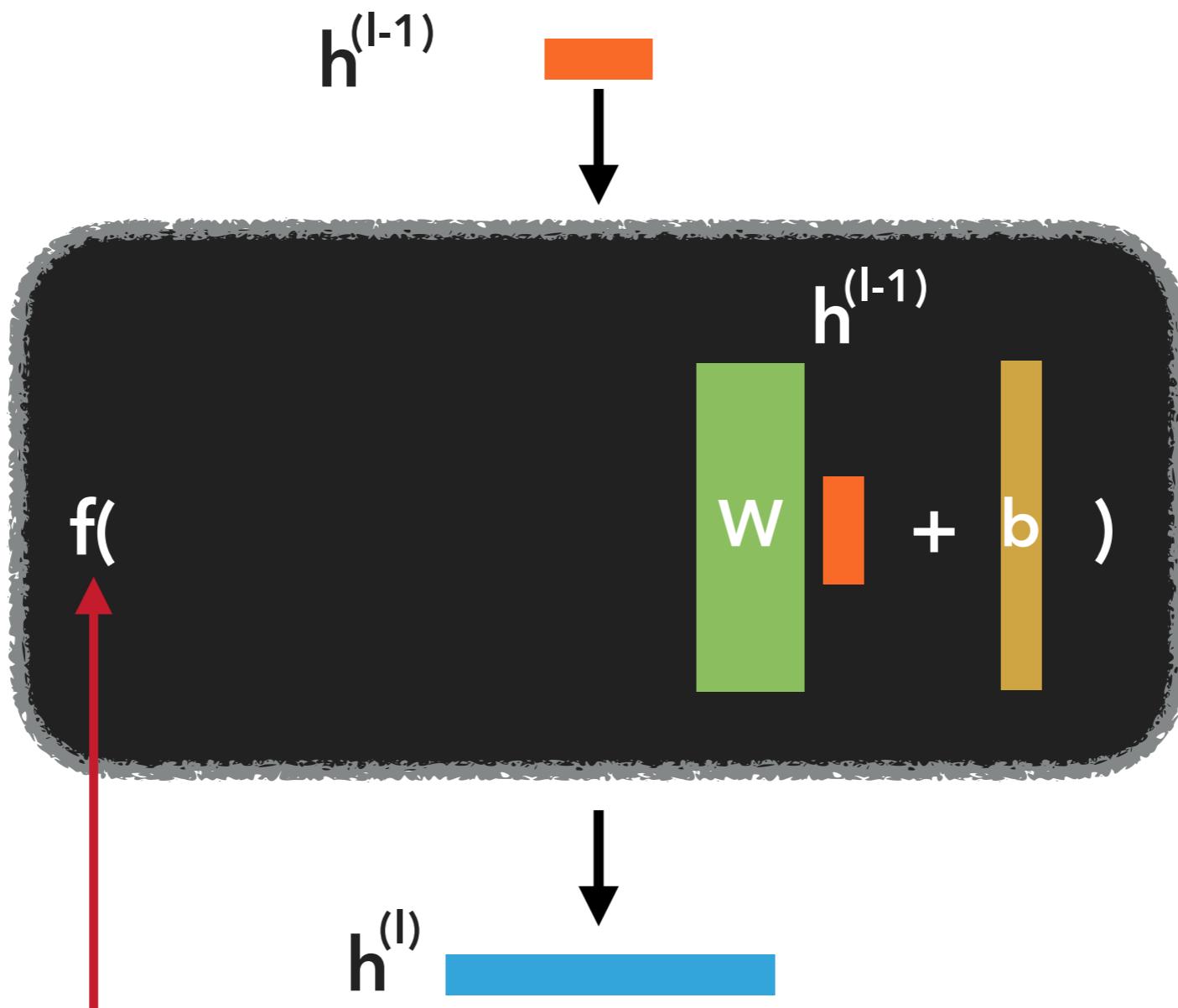
### FF-LAYER

$$h^{(l)} = f(W h^{(l-1)} + b)$$



### FF-LAYER

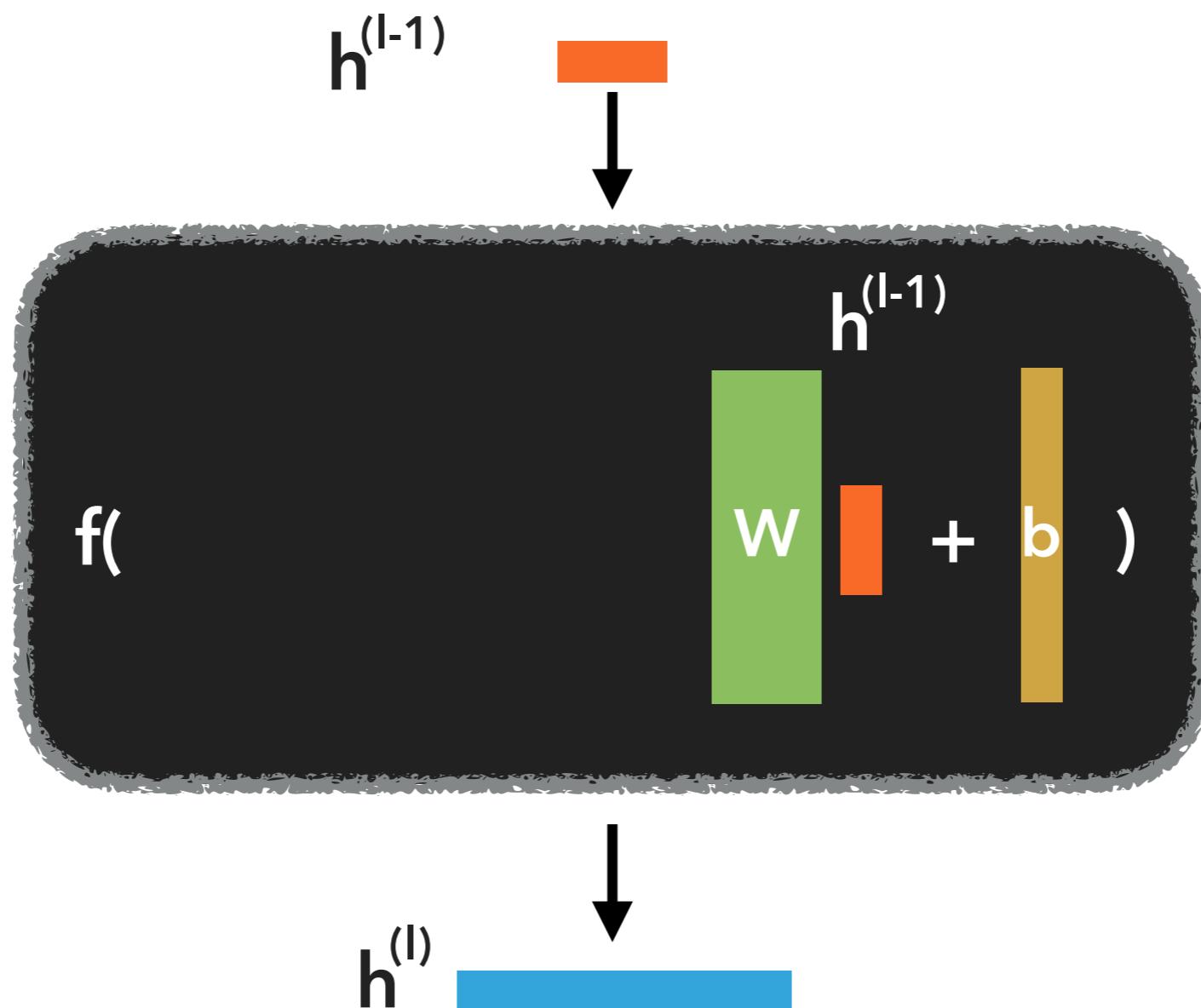
$$h^{(l)} = f(Wh^{(l-1)} + b)$$



non-linear function required to achieve universal approximator property

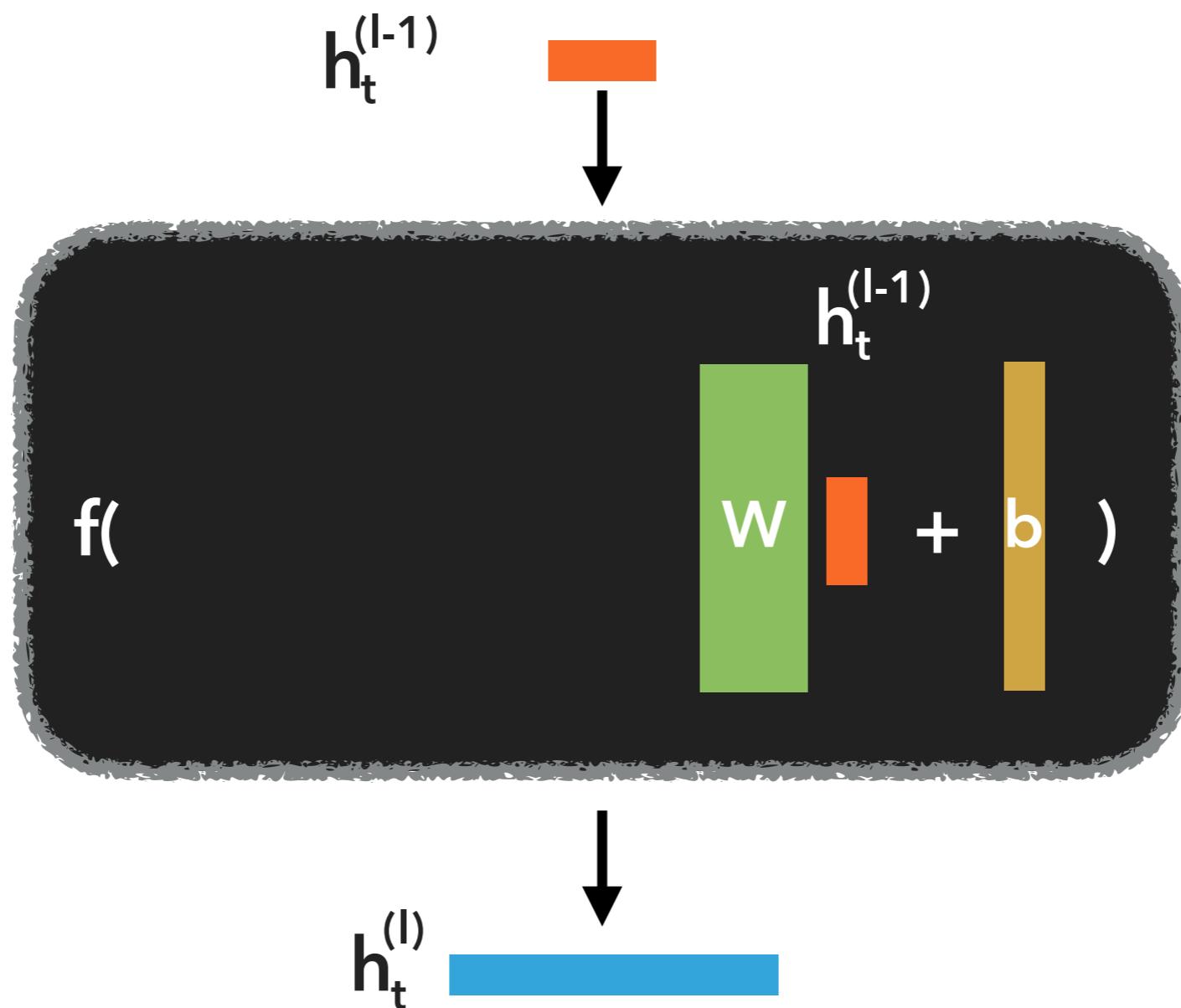
# RECURRENT NEURAL NETWORKS

---



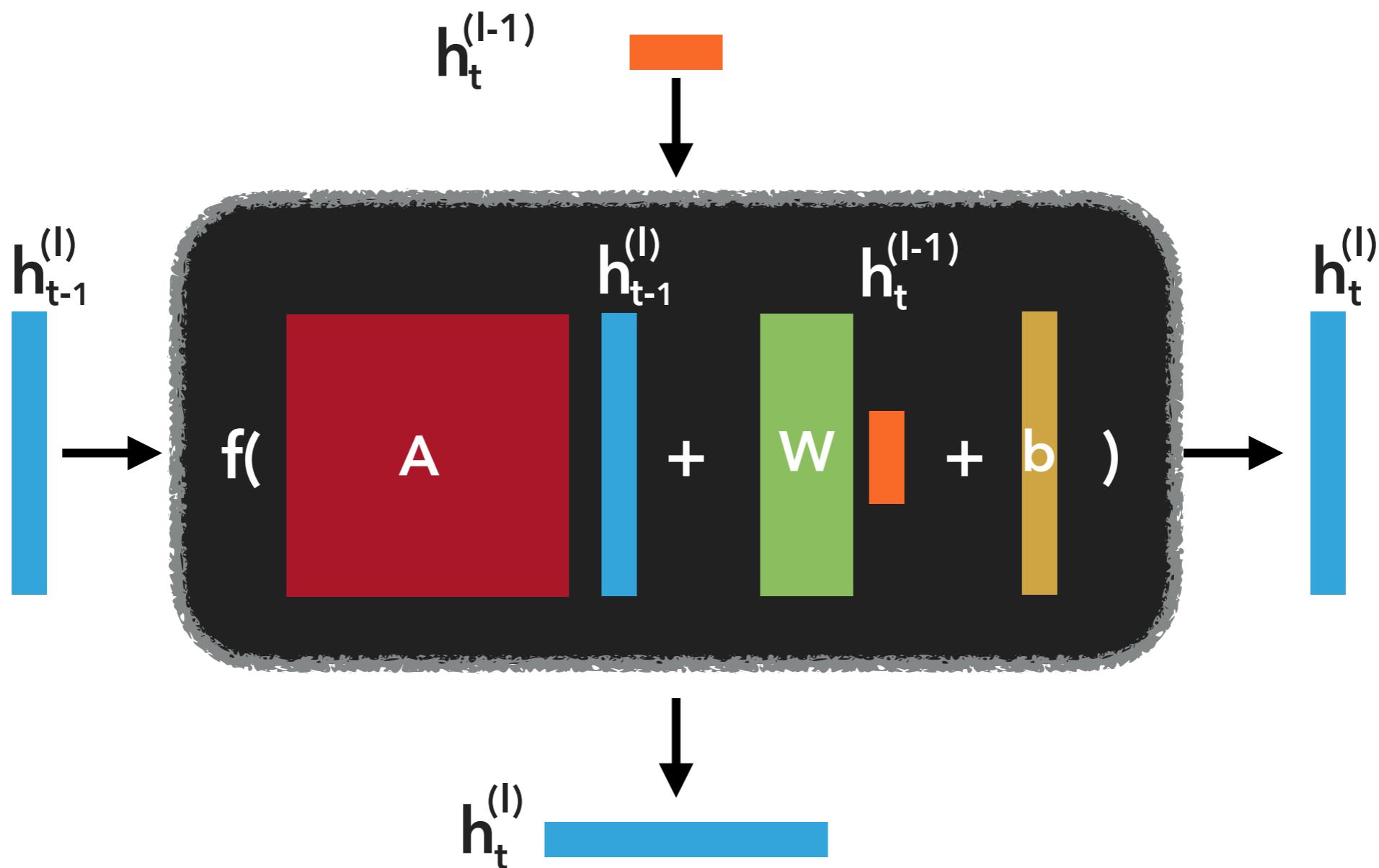
# RECURRENT NEURAL NETWORKS

---



# RECURRENT NEURAL NETWORKS

---

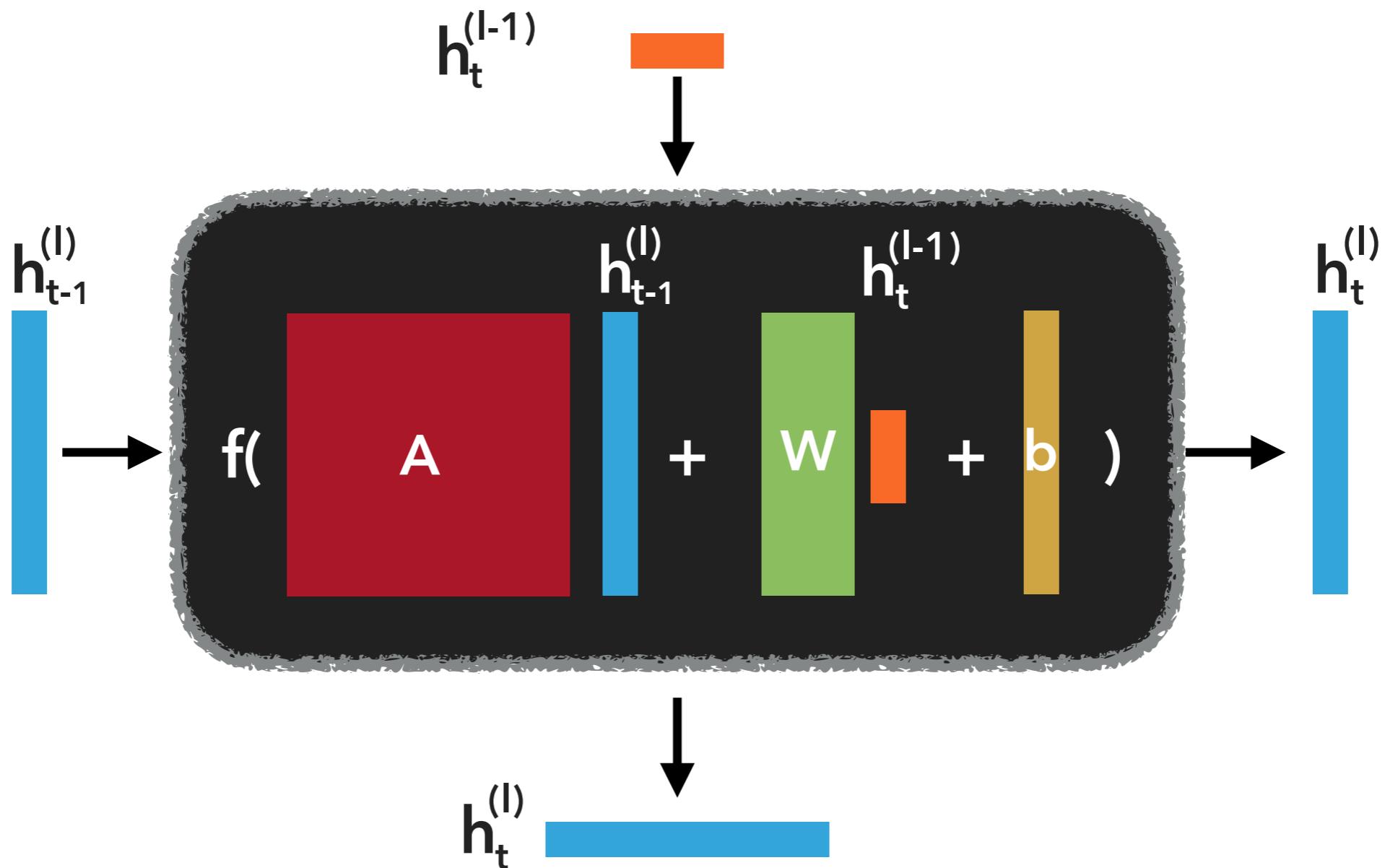


## RECURRENT NEURAL NETWORKS

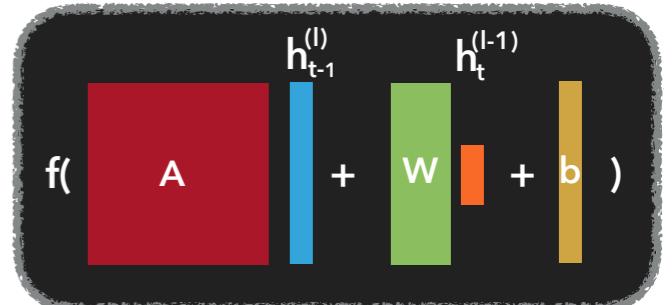
---

### RNN-CELL

$$h_t^{(l)} = f(Ah_{t-1}^{(l)} + Wh_t^{(l-1)} + b)$$



## VANISHING AND EXPLODING GRADIENT



$$h_t^{(l)} = f(Ah_{t-1}^{(l)} + Wh_t^{(l-1)} + b)$$

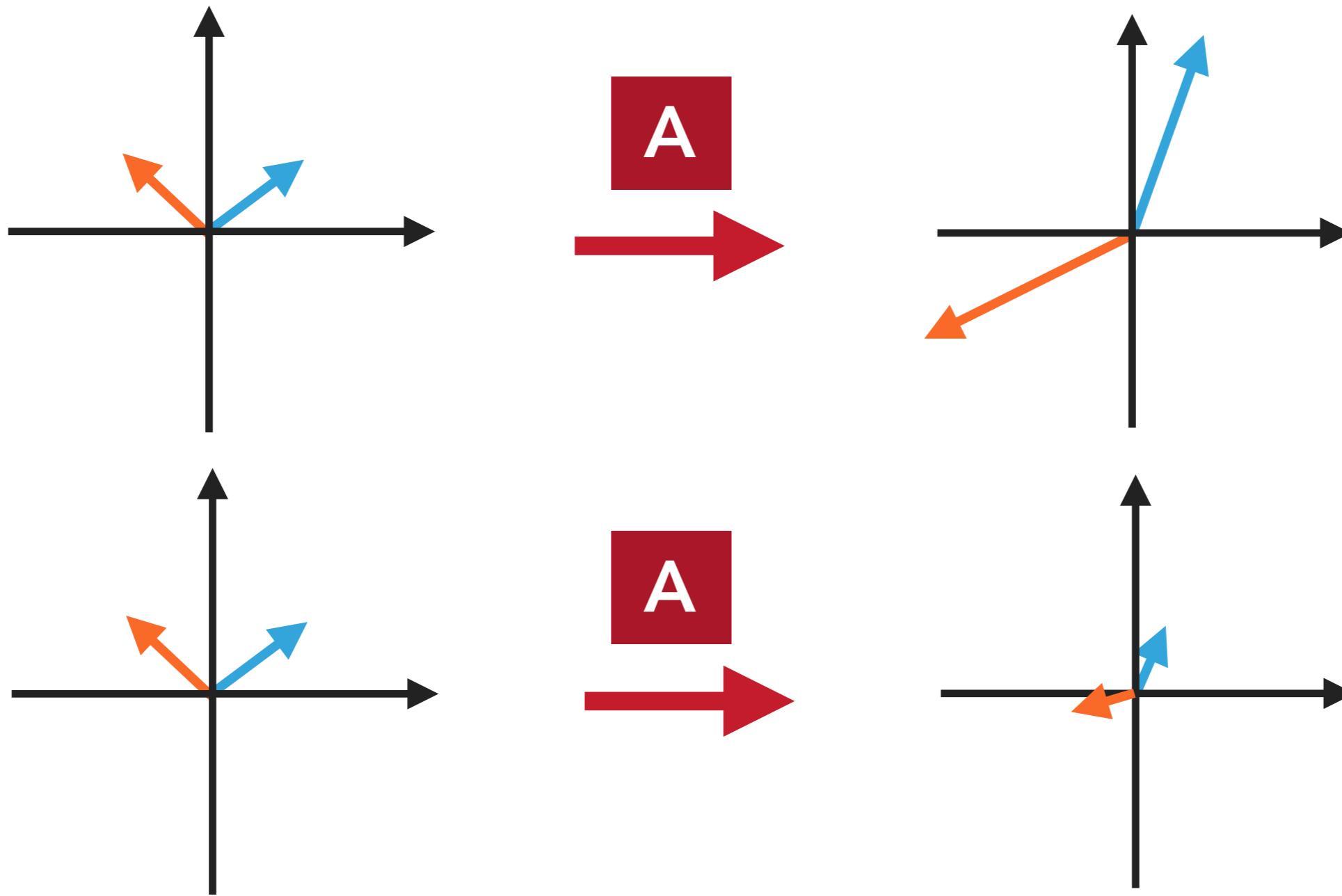
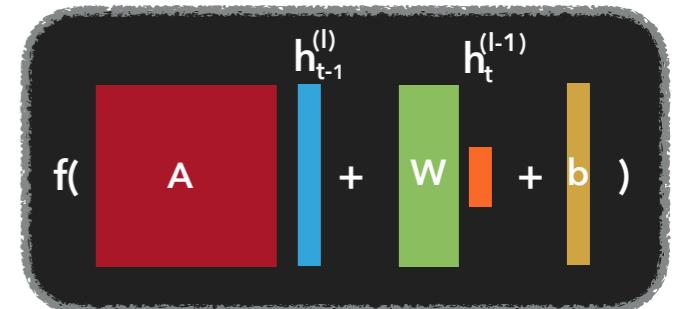


backpropagation

$$\frac{\partial h_T^{(l)}}{\partial A_{a,b}} = \dots + \dots A^t \frac{\partial h_{T-t}^{(l)}}{\partial A_{a,b}}$$

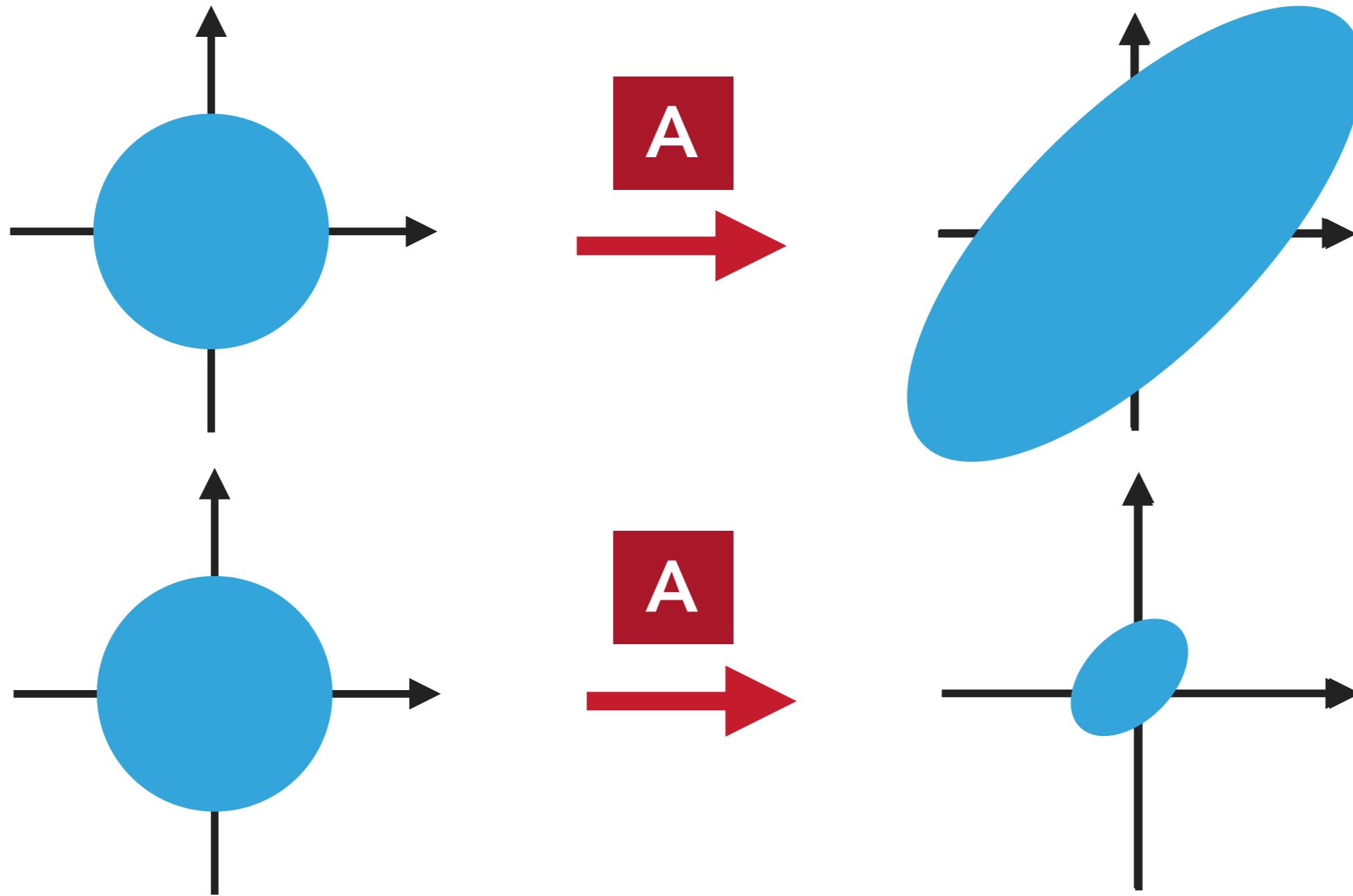
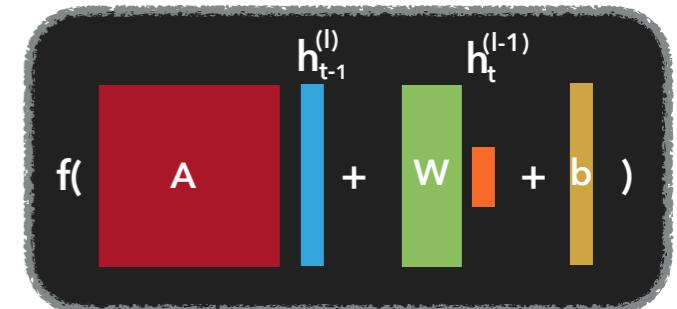
## VANISHING AND EXPLODING GRADIENT

Matrix Multiplication

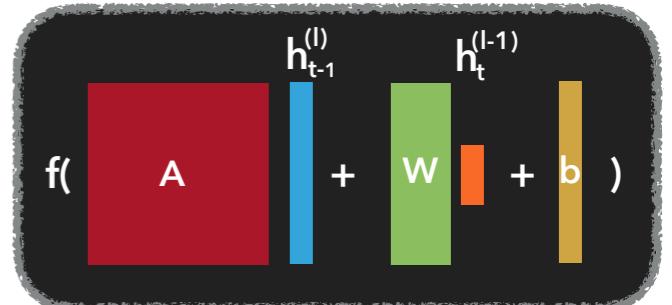


## VANISHING AND EXPLODING GRADIENT

Matrix Multiplication



## VANISHING AND EXPLODING GRADIENT



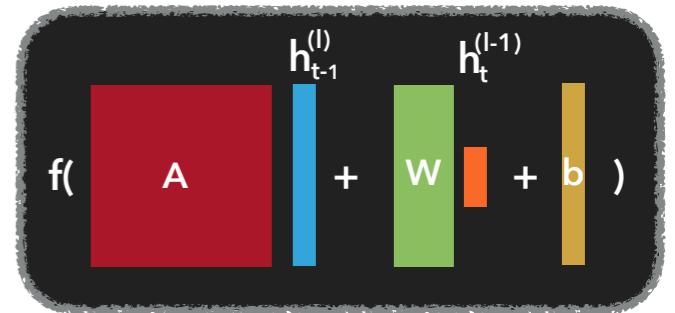
$$h_t^{(l)} = f(Ah_{t-1}^{(l)} + Wh_t^{(l-1)} + b)$$



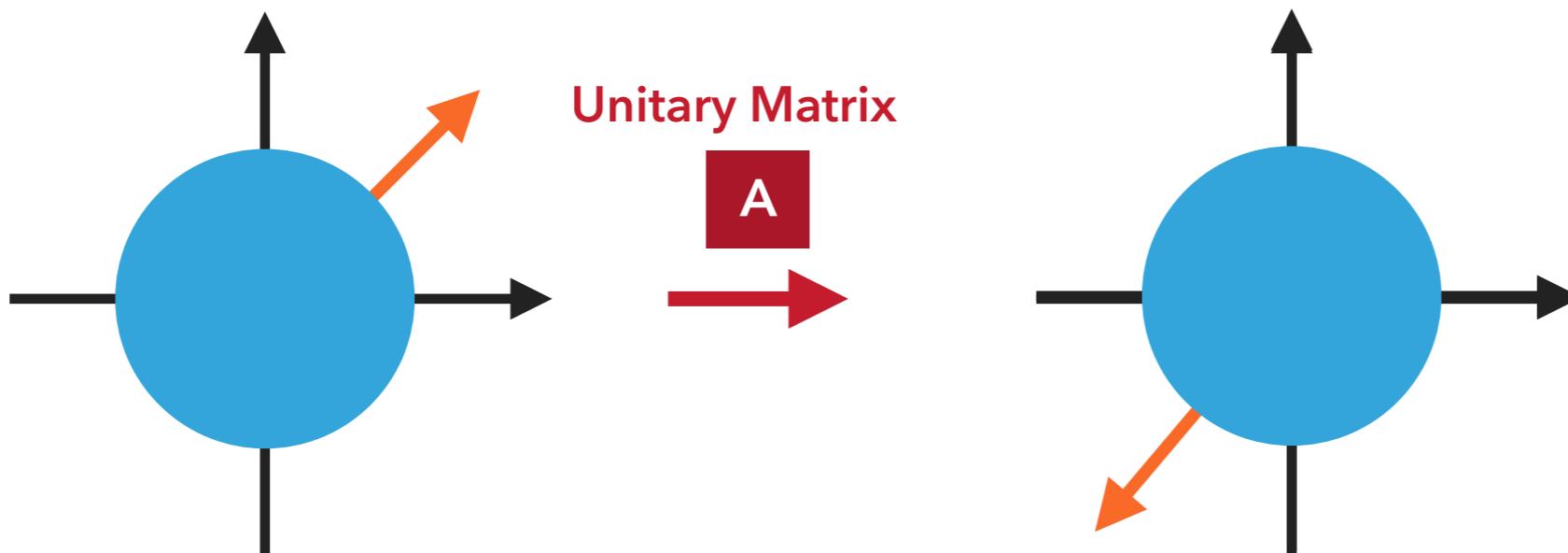
backpropagation

$$\frac{\partial h_T^{(l)}}{\partial A_{a,b}} = \dots + \dots A^t \frac{\partial h_{T-t}^{(l)}}{\partial A_{a,b}}$$

## UNITARY RNN

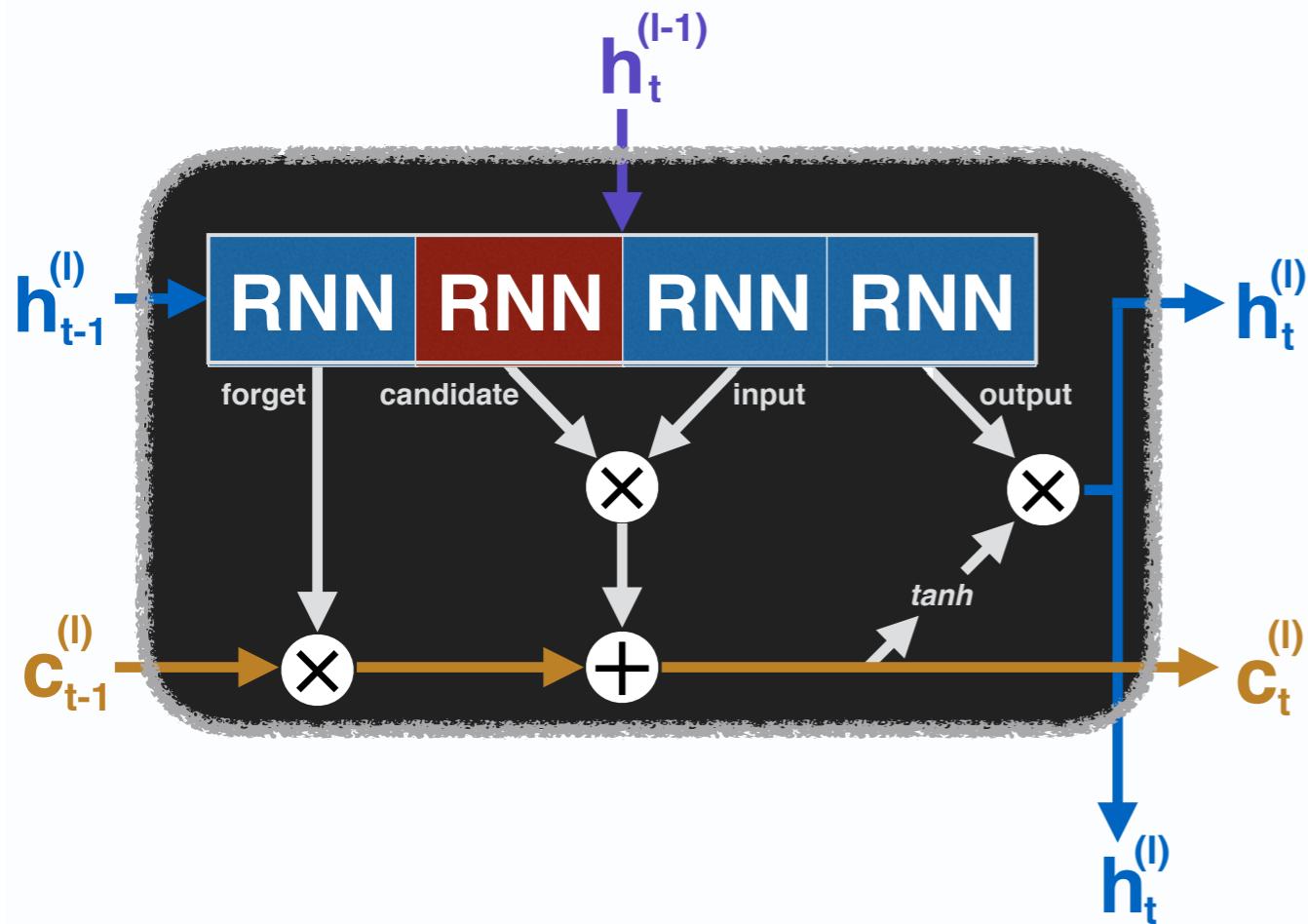


$$\frac{\partial h_T^{(l)}}{\partial A_{a,b}} = \dots + \dots A^t \frac{\partial h_{T-t}^{(l)}}{\partial A_{a,b}}$$

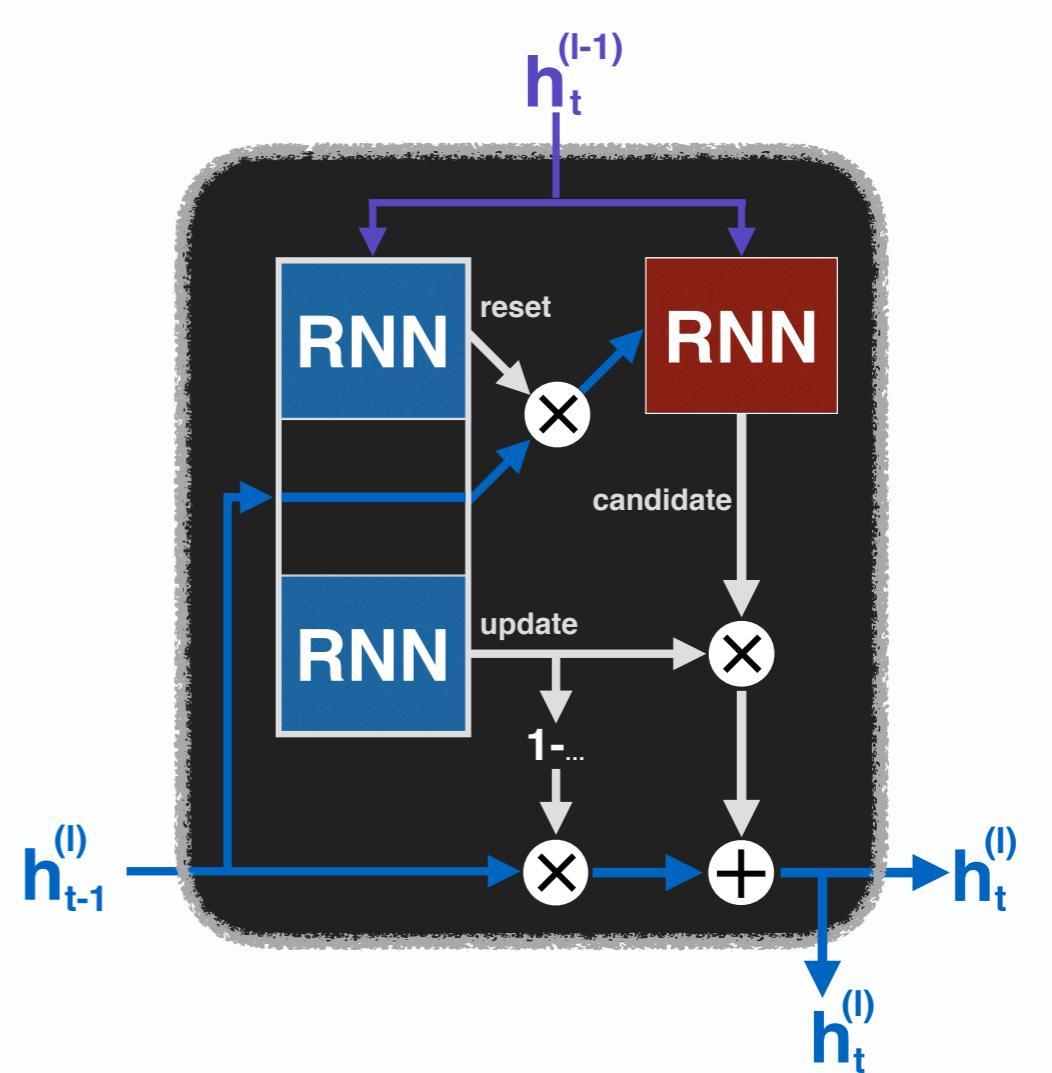


## GATED SOLUTIONS

### LSTM

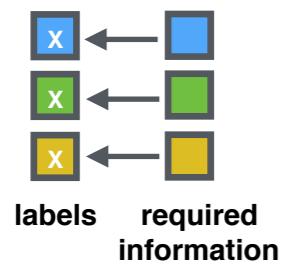


### GRU



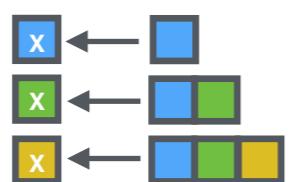
# TASKS REQUIRING HIGH INPUT HISTORY STORAGE CAPACITY

**non sequential**



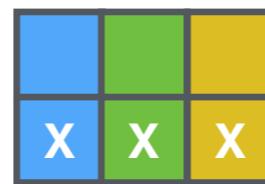
labels      required information

**sequential**

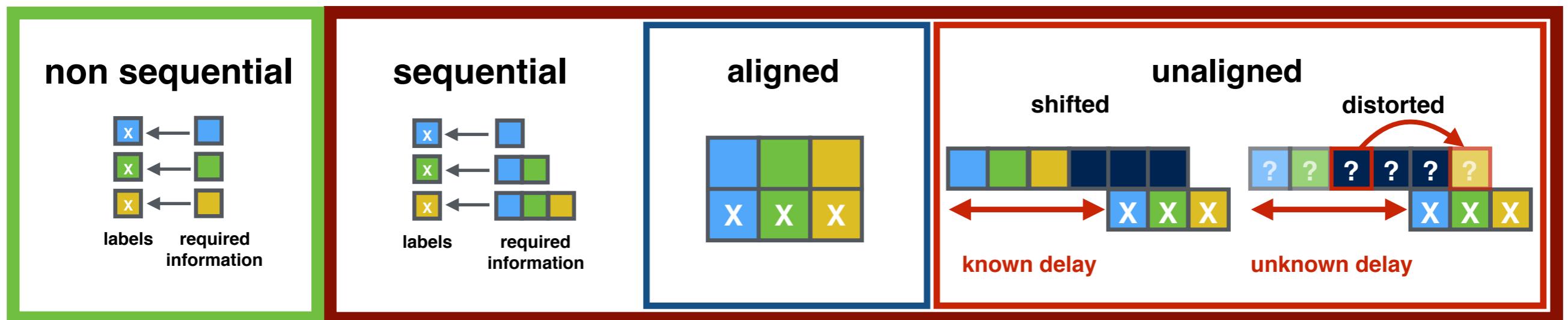


labels      required information

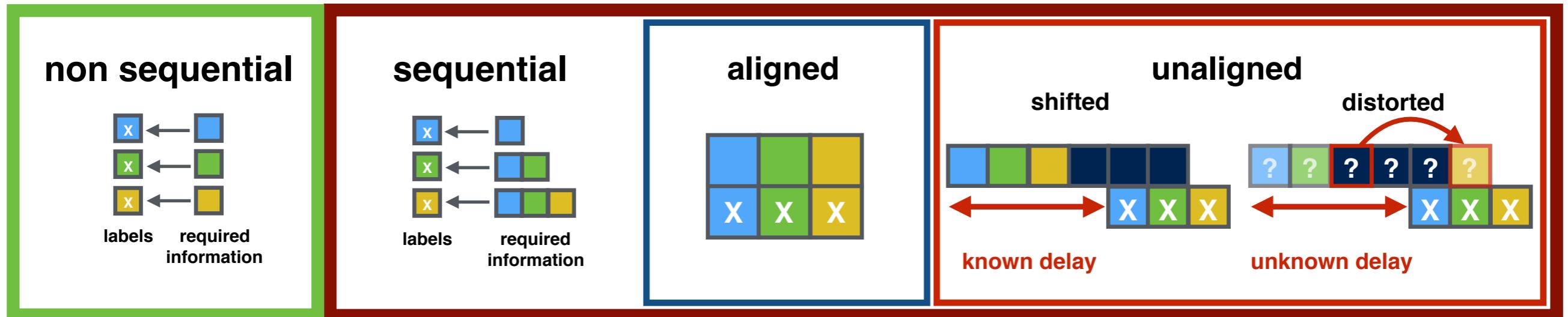
**aligned**



# TASKS REQUIRING HIGH INPUT HISTORY STORAGE CAPACITY



# TASKS REQUIRING HIGH INPUT HISTORY STORAGE CAPACITY



## high numbers of hidden units

- **intractability**
- **overcomplexity**

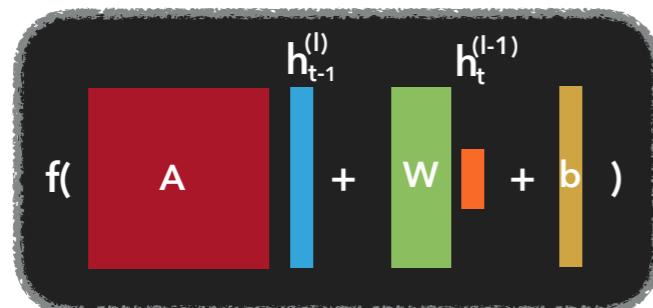
$$f(A h_{t-1}^{(l)} + W h_t^{(l-1)} + b)$$

## high delays

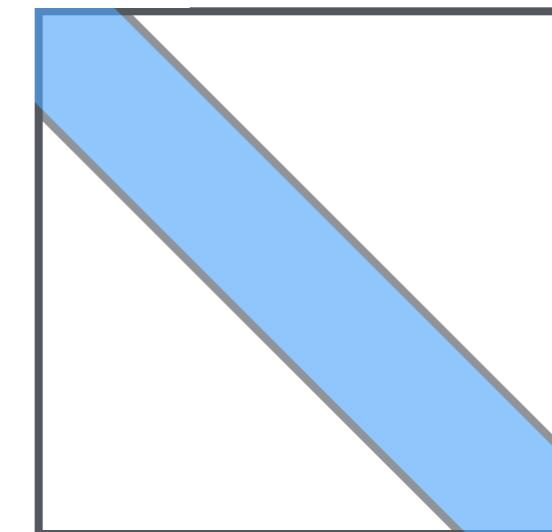
- **vanishing and exploding gradient problem**

## BAND MATRIX RESTRICTION

# BAND RNN



Band Matrix Restriction



intractability → linear growth in necessary storage and operations:  $O(n)$

overcomplexity → linear growth in number of parameters\task storage capacity:  $O(n)$

vanishing and exploding gradient problem NOT solved BUT

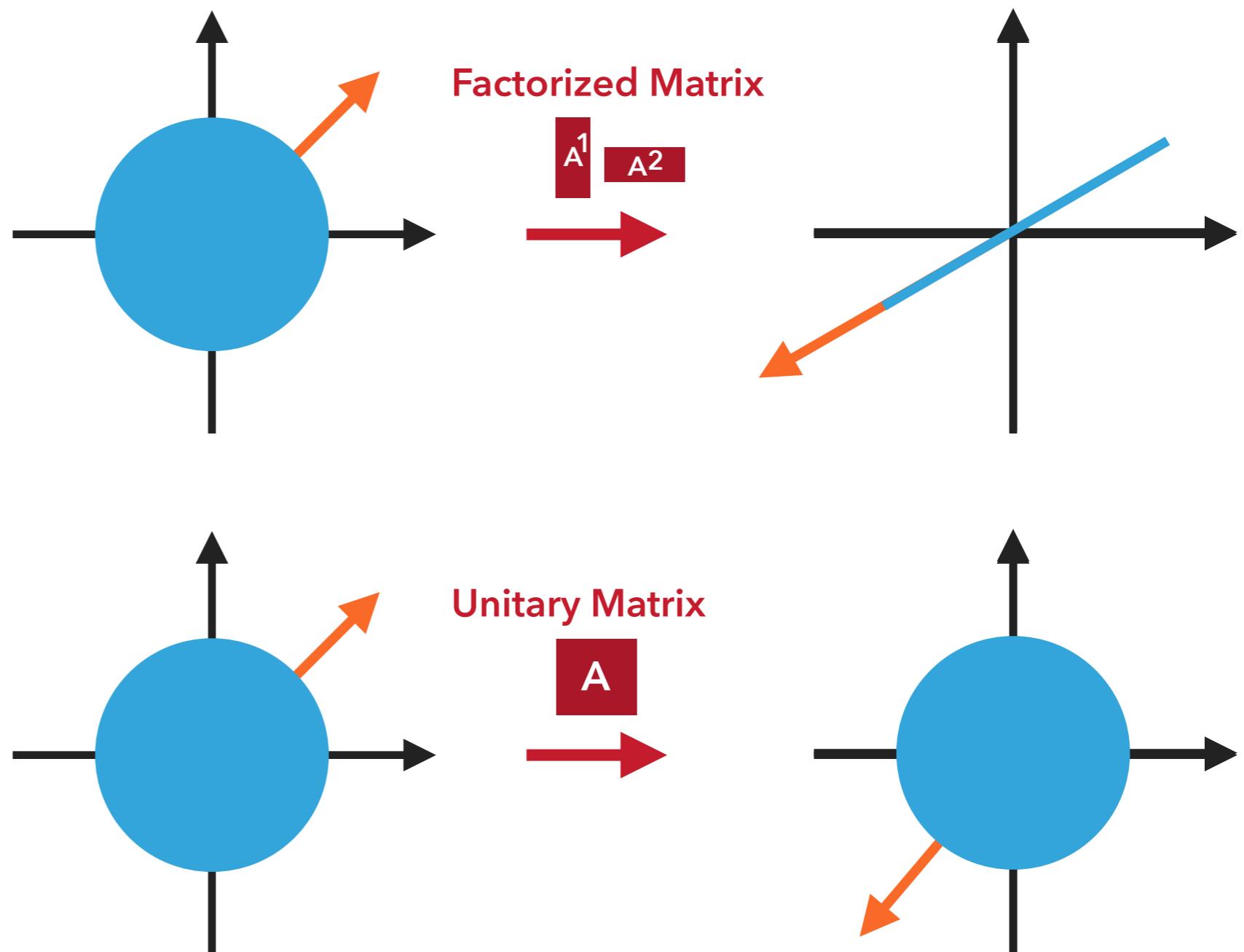
- close to but lower than full rank
- less chaotic
- shift initialization

# MEMORY CAPACITY OF OTHER MATRIX RESTRICTIONS

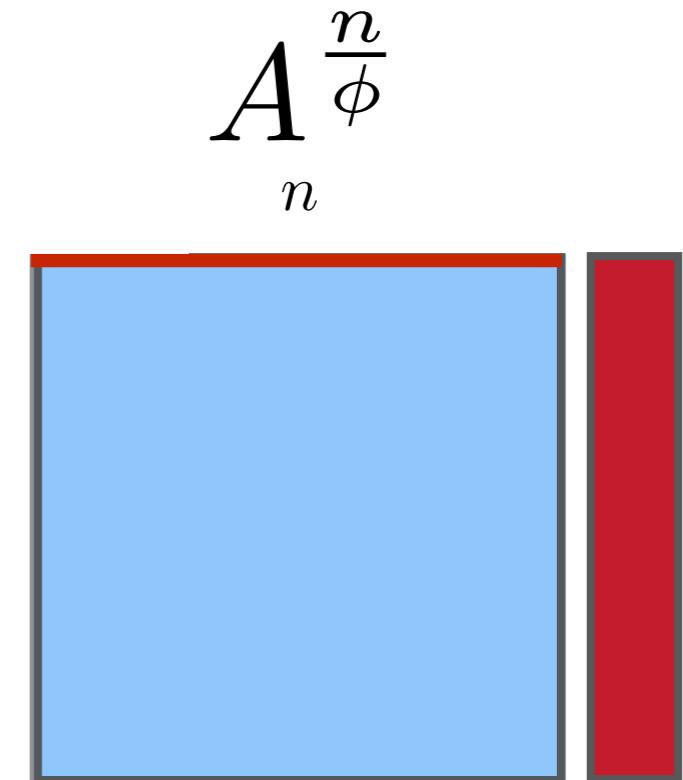
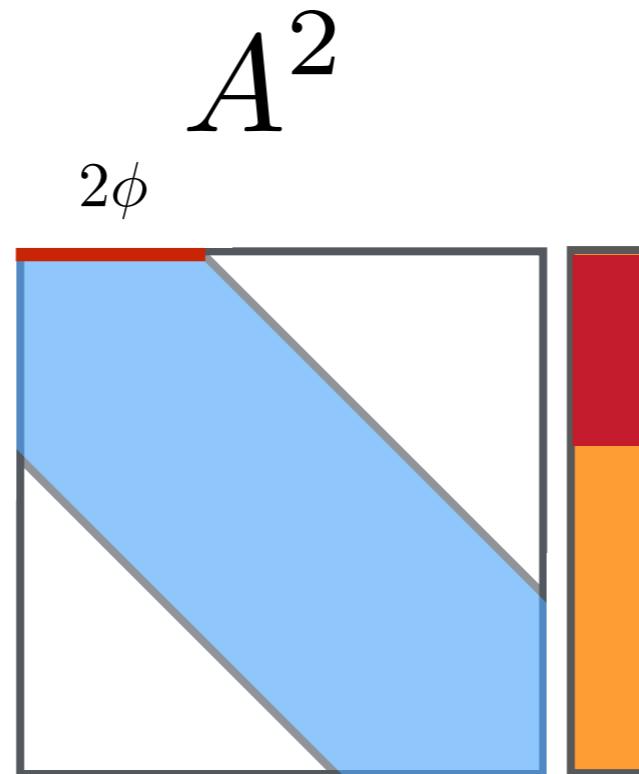
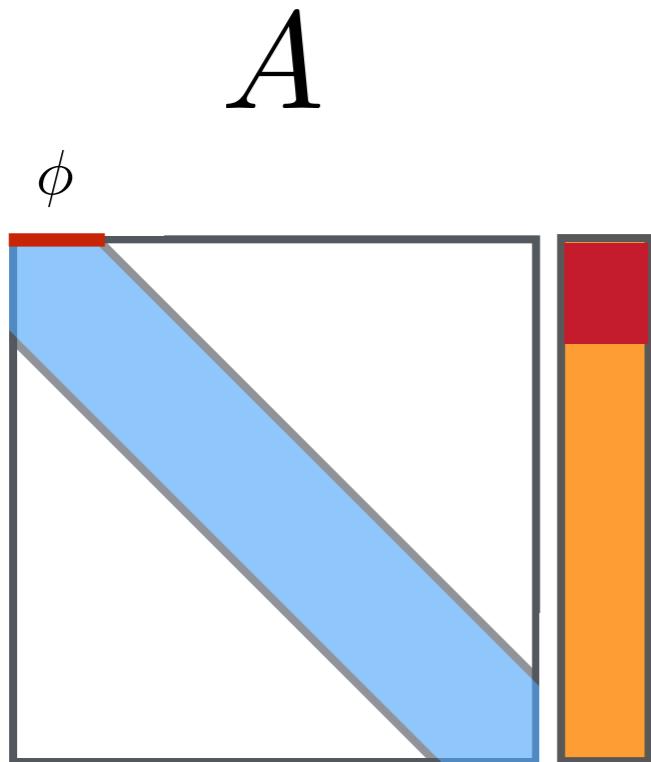
**Factorized**  
forgets much information

$$f(A h_{t-1}^{(l)} + W h_t^{(l-1)} + b)$$

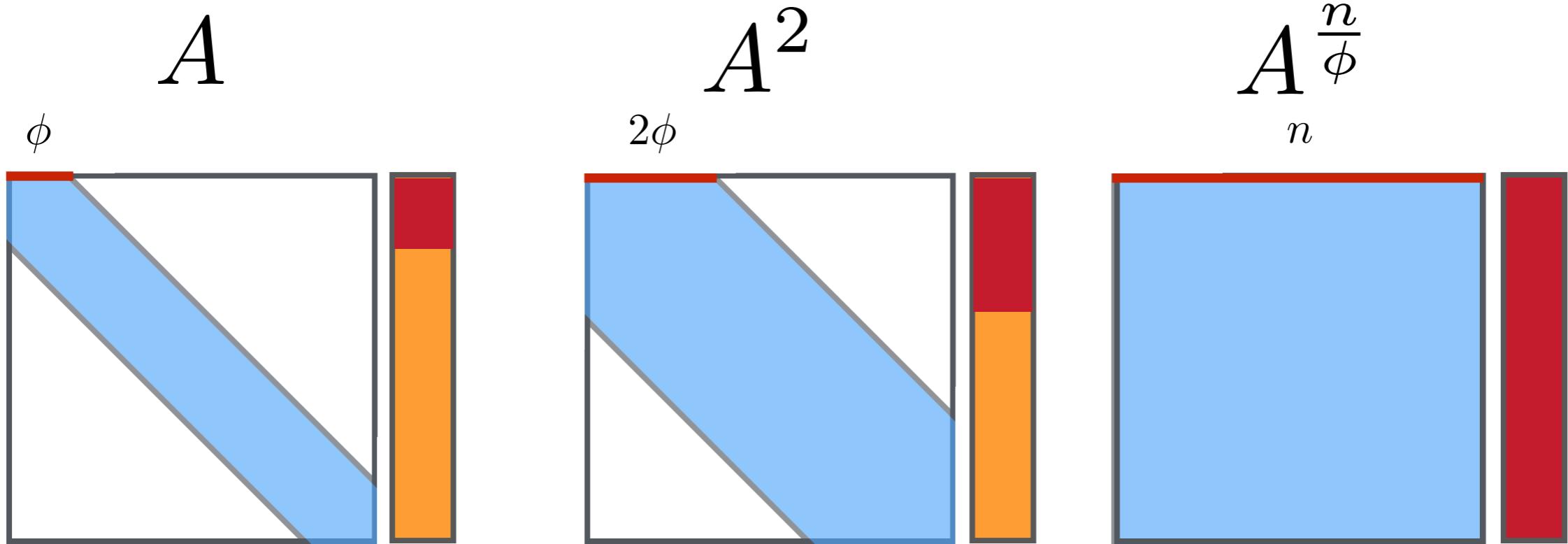
**URNN**  
needs input to forget



# MESSAGE PASSING

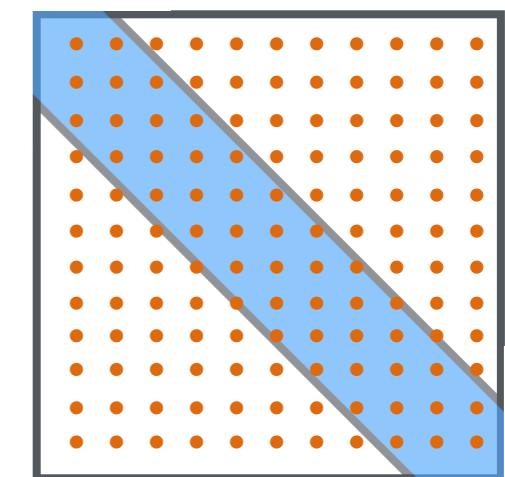


## MESSAGE PASSING



idea: adding a sparse grid on top of the band

- speeds up the message passing
- adds complexity
- complex system need long-range interactions



# SHIFT INITIALIZATION

Shift initialization

$$\mathbf{f}(W_h) = h_{i-1} + W_i + \begin{pmatrix} 4 \\ 5 \end{pmatrix} + b = h_i$$

The diagram illustrates the computation of a band matrix restriction. On the left, a large matrix  $W_h$  is shown with a red diagonal line indicating the band structure. To its right is the equation for shift initialization:

- A vertical vector  $h_{i-1}$  with entries 1, 1, 1, 1, 1, 1, 1.
- A plus sign.
- A vertical vector  $W_i$  with a red diagonal line.
- A plus sign.
- A vertical vector  $x_i$  with entries 4, 5.
- A plus sign.
- A vertical vector  $b$  (empty).
- An equals sign followed by a vertical vector  $h_i$  with entries 4, 5, 1, 1, 1, 1, 1.

## SHIFT INITIALIZATION

Shift initialization

$$f(W_h) = h_{i-1} + W_i + \begin{pmatrix} 4 \\ 5 \end{pmatrix} + b = h_i$$

Closed Shift initialization

$$f(W_h) = h_{i-1} + W_i + \begin{pmatrix} 4 \\ 5 \end{pmatrix} + b = h_i$$

## SHIFT INITIALIZATION

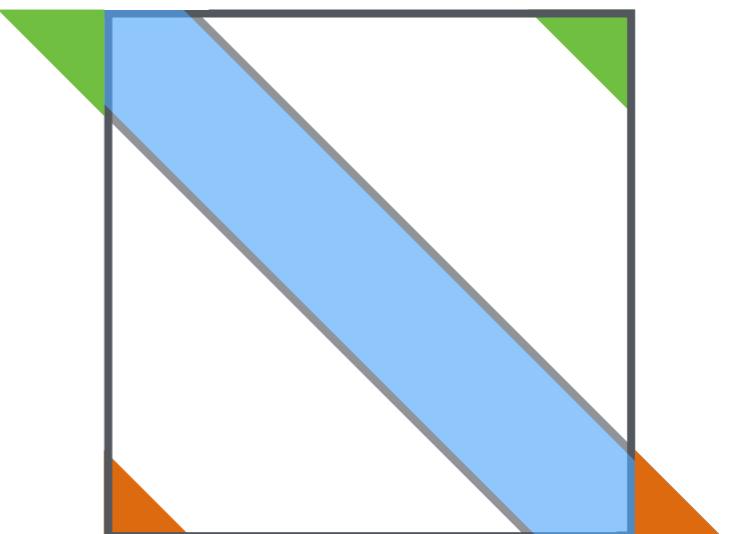
Shift initialization

$$f(W_h) = h_{i-1} + W_i \begin{pmatrix} 4 \\ 5 \end{pmatrix} + b = h_i$$

**idea: adding values to the corners of the band matrix**

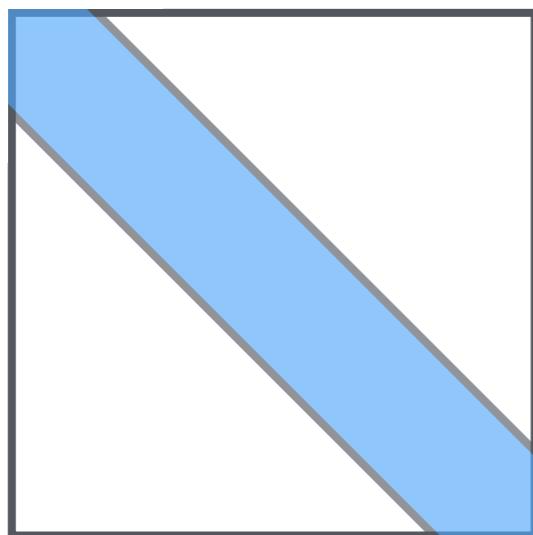
Closed Shift initialization

$$f(W_h) = h_{i-1} + W_i \begin{pmatrix} 4 \\ 5 \end{pmatrix} + b = h_i$$

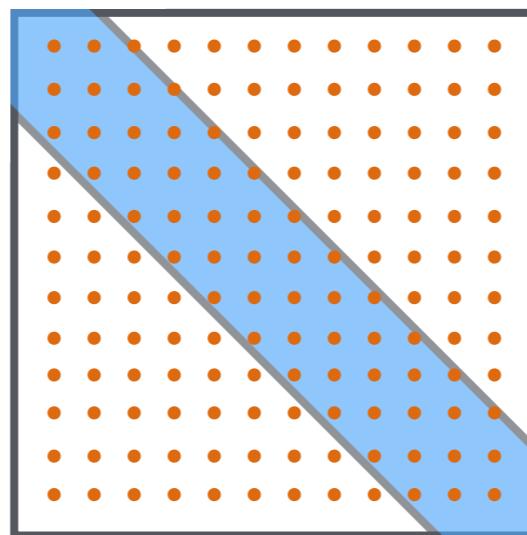


## FINAL MODELS

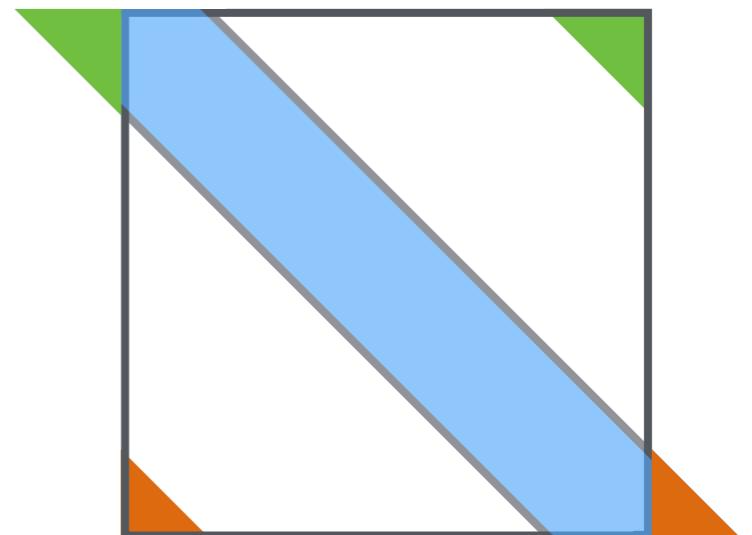
**band**



**band grid combination**

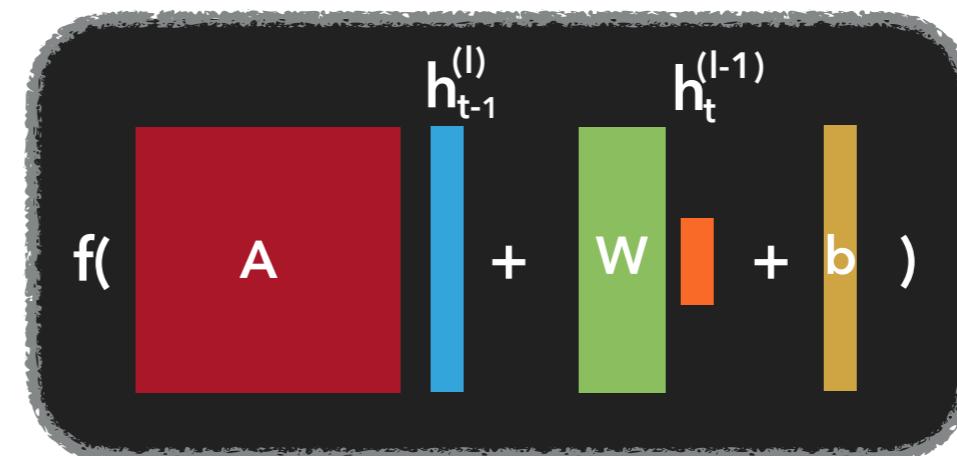


**closed band**



**Band Matrix Restrictions**

A



# SPARSE AND BLOCK FORMAT

## standard matrix

128

33584718	00000000	00000000	00000000
85561452	20000000	00000000	00000000
34473723	11000000	00000000	00000000
35121141	17500000	00000000	00000000
31828215	77430000	00000000	00000000
71235718	81168000	00000000	00000000
86272487	27127708	00000000	00000000
53512217	2728511000000000	00000000	00000000
04265157	24572252	00000000	00000000
0082555886	38374210000000	00000000	00000000
0002352658	16412132000000	00000000	00000000
0000222758	11175671300000	00000000	00000000
0000045124	52812856710000	00000000	00000000
0000001647	31831246225000	00000000	00000000
000000013772	162457477200	00000000	00000000
000000002263	237222826110	00000000	00000000
000000000113	268452222471	00000000	00000000
000000000075	1372572528327	00000000	00000000
000000000026	2411217712851000000	00000000	00000000
00000000000242	81223421841600000	00000000	00000000
00000000000064	17713626252420000	00000000	00000000
00000000000062	3416486113258000	00000000	00000000
00000000000007	1827155788138700	00000000	00000000
00000000000008	1537135151771830	00000000	00000000
00000000000008	0213756637758352	00000000	00000000
00000000000008	0024813112474161	00000000	00000000
00000000000008	0006312235513813	00000000	00000000
00000000000008	00000122871121818	00000000	00000000
00000000000008	0000015132418775	00000000	00000000
00000000000008	0000007175857867	00000000	00000000
00000000000008	0000000116822111	00000000	00000000
00000000000008	0000000037555142	00000000	00000000

128

# SPARSE AND BLOCK FORMAT

## standard matrix

33584718	000000000000000000000000	00000000
85561452	200000000000000000000000	00000000
34473723	110000000000000000000000	00000000
35121141	175000000000000000000000	00000000
31828215	774300000000000000000000	00000000
71235718	811680000000000000000000	00000000
86272487	271277000000000000000000	00000000
53512217	272851100000000000000000	00000000
04265157	245722520000000000000000	00000000
0082555886	383742100000000000000000	00000000
0002352658	164121320000000000000000	00000000
0000222758	117567130000000000000000	00000000
0000004512	452812856710000000000000	00000000
000000164731	813124622500000000000000	00000000
0000000137721624	574772000000000000000000	00000000
0000000022632372	228261100000000000000000	00000000
0000000001132684	522224710000000000000000	00000000
00000000007513726	725283270000000000000000	00000000
00000000000262411217712851	000000000000000000000000	00000000
00000000000024281223421841600000	000000000000000000000000	00000000
000000000000006417713626252420000	000000000000000000000000	00000000
00000000000000623416486113258000	000000000000000000000000	00000000
00000000000000071827155788138700	000000000000000000000000	00000000
00000000000000001537135151771830	000000000000000000000000	00000000
00000000000000000213756637758352	000000000000000000000000	00000000
00000000000000000024813112474161	000000000000000000000000	00000000
0000000000000000006312285513813	000000000000000000000000	00000000
00000000000000000000122871121818	000000000000000000000000	00000000
0000000000000000000015182418775	000000000000000000000000	00000000
0000000000000000000007175857867	000000000000000000000000	00000000
000000000000000000000116822111	000000000000000000000000	00000000
00000000000000000000000037555142	000000000000000000000000	00000000

## sparse format

# SPARSE AND BLOCK FORMAT

standard matrix

33584718	000000000000000000000000
85561452	200000000000000000000000
34473723	110000000000000000000000
35121141	175000000000000000000000
31828215	774300000000000000000000
71235718	811680000000000000000000
86272487	271277000000000000000000
53512217	272851100000000000000000
04265157	245722520000000000000000
00825558	638374210000000000000000
00023526	581641213200000000000000
00002227	581117567130000000000000
00000451	245281285671000000000000
00000016	473183124622500000000000
00000001	377216245747720000000000
00000000	226323722282611000000000
00000000	011326845222247100000000
00000000	007513726725283270000000
00000000	002624112177128510000000
00000000	000242812234218416000000
00000000	000064177136262524200000
00000000	000062341648611325800000
00000000	00000718271557881387000000
00000000	0000001537135151771830000000
00000000	0000000000021375668775835200000000
00000000	00000000000202481311247416100000000
00000000	000000000000006312285513813000000000
00000000	00000000000000001228711218180000000000
00000000	00000000000000000000151824187750000000000
00000000	00000000000000000000071758578670000000000
00000000	000000000000000000000116822111000000000000
00000000	00000000000000000000003755514200000000000000

sparse format

00	08					
09	01	10				
	11	02	12			
		13	03	14		
			15	04	16	
				17	05	18
					19	06
						20
						21
						07

block format

00000000	33584718	00000000
00000000	85561452	20000000
00000000	34473723	11000000
00000000	35121141	17500000
00000000	31828215	77430000
00000000	71235718	81168000
00000000	86272487	27127700
00000000	53512217	27285110
04265157	24572252	00000000
00825558	63837421	00000000
00023526	58164121	32000000
00002227	58111756	71300000
00000451	24528128	56710000
00000016	47318312	46225000
00000001	37721624	57477200
00000000	22632372	22826110
01132684	52222471	00000000
00751372	67252832	70000000
00026241	12177128	51000000
00002428	12234218	41600000
00000641	77136262	52420000
00000062	34164861	13258000
00000007	1827155788	13870000
00000000	1537135151	77183000
02137566	87758352	00000000
00248131	12474161	00000000
00063122	85513813	00000000
00001228	71121818	00000000
00000151	82418775	00000000
00000071	75857867	00000000
00000001	16822111	00000000
00000000	37555142	00000000

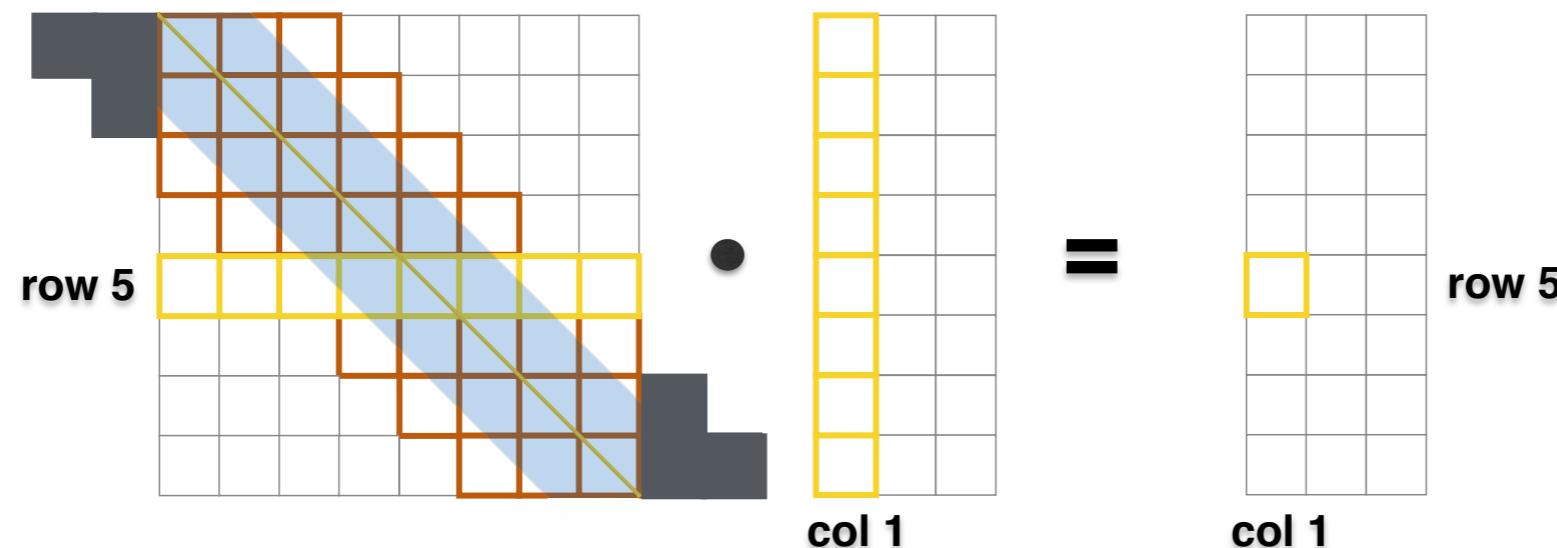
128

128

96

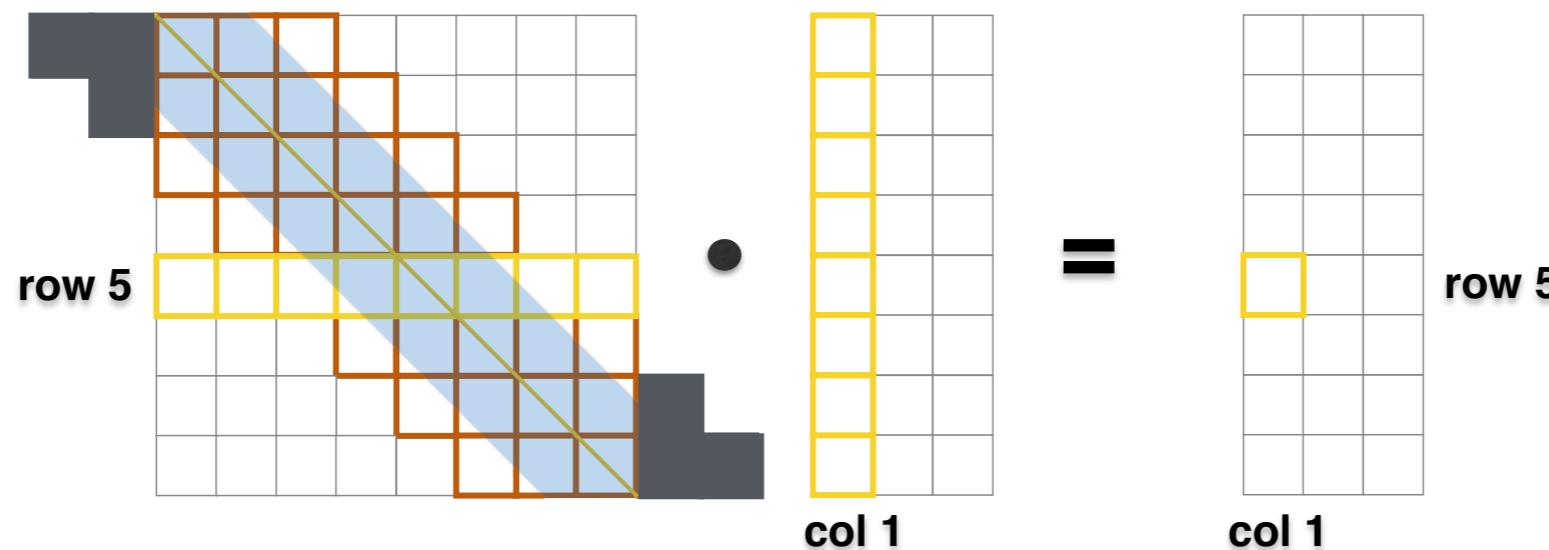
# GENERAL BAND-MATRIX MATRIX MULTIPLICATION IDEA

GEMM loops through all blocks of the full matrices A and B

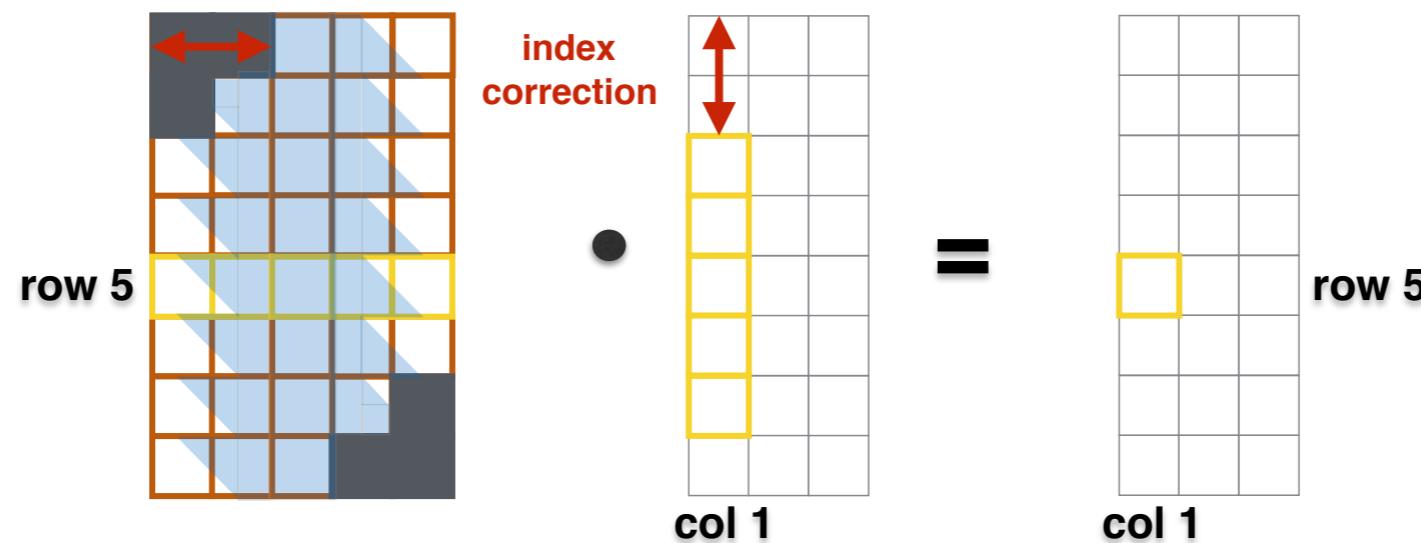


# GENERAL BAND-MATRIX MATRIX MULTIPLICATION IDEA

GEMM loops through all blocks of the full matrices A and B

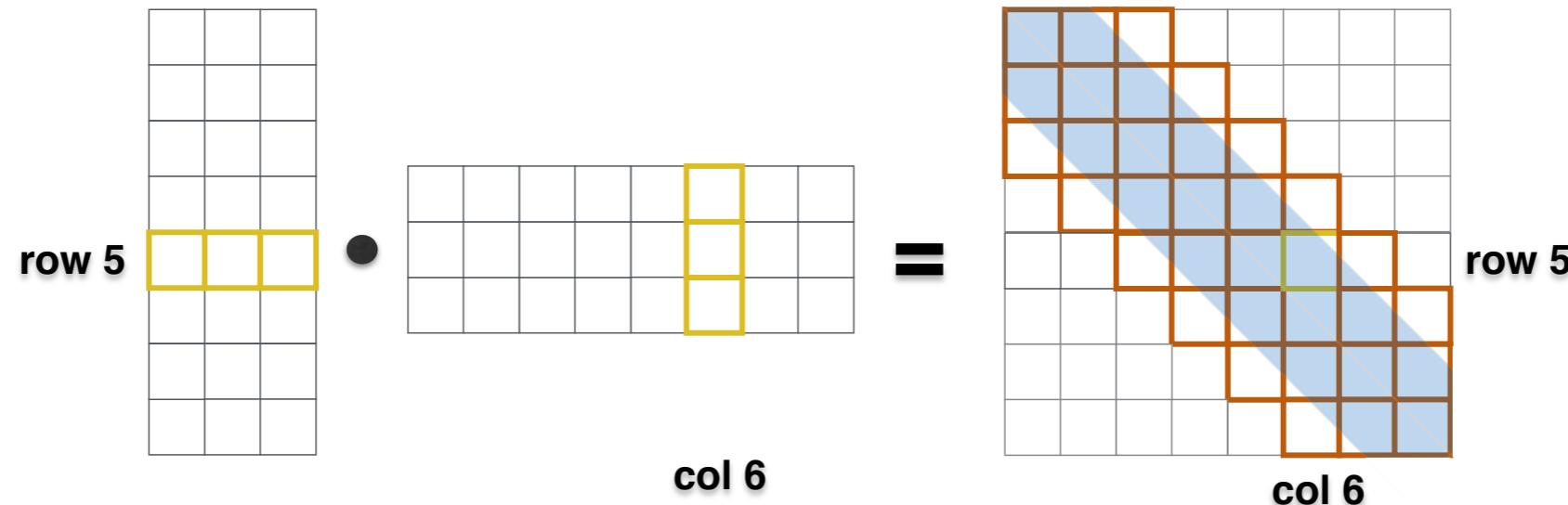


GBMM cuts the loop using a sparse blocked band matrix for A and corrects in block indices in B



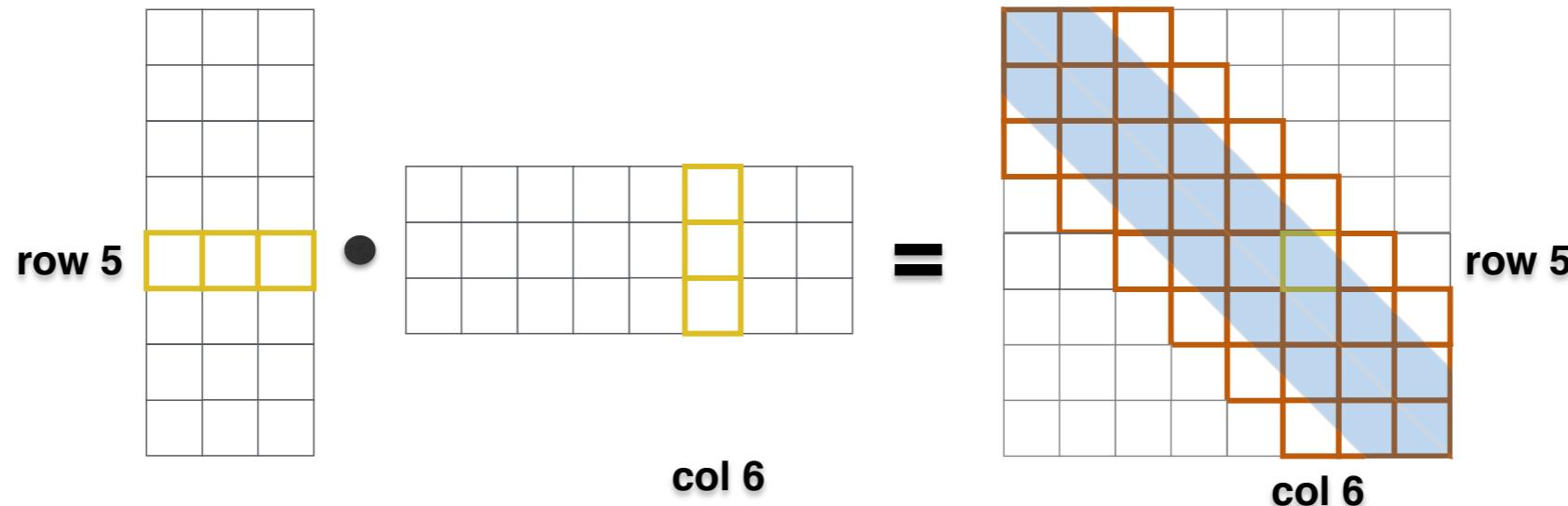
# GENERAL TO BAND-MATRIX MATRIX MULTIPLICATION IDEA

GEMM calculates all blocks of the resulting matrix C

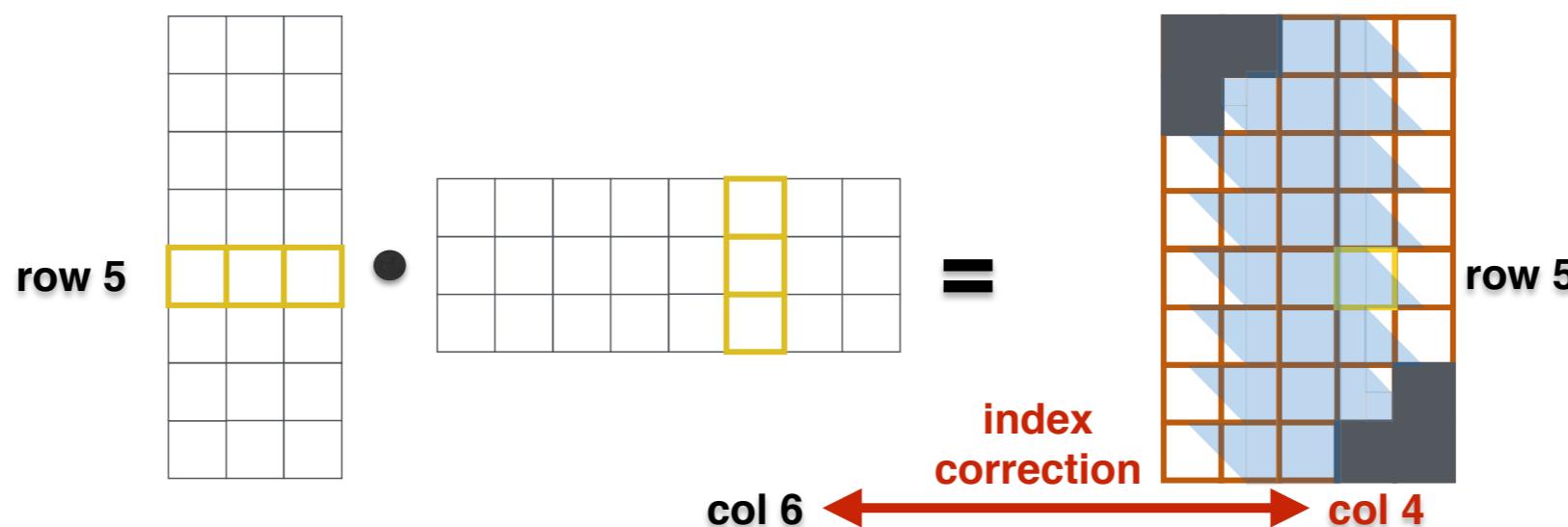


# GENERAL TO BAND-MATRIX MATRIX MULTIPLICATION IDEA

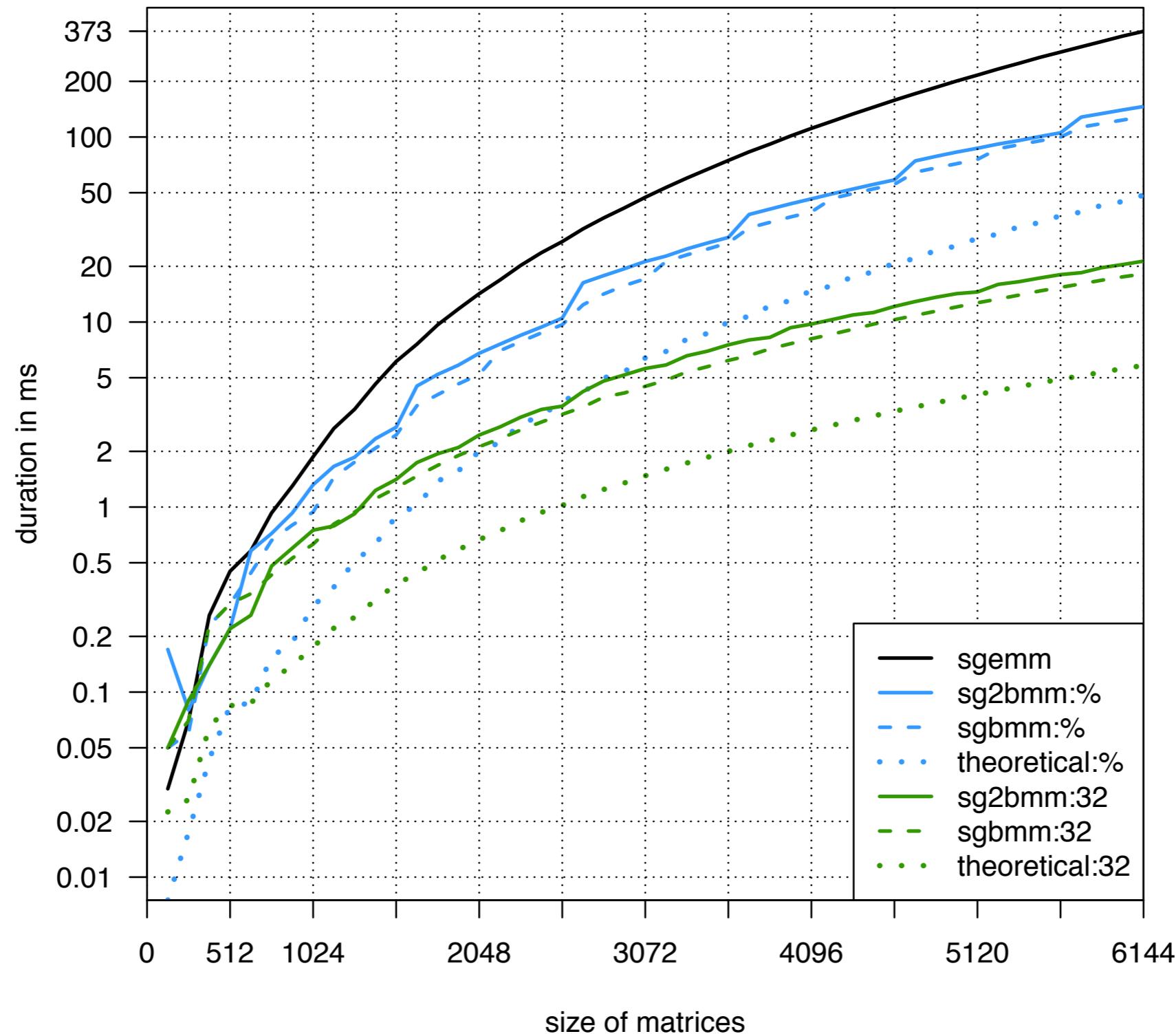
GEMM calculates all blocks of the resulting matrix C



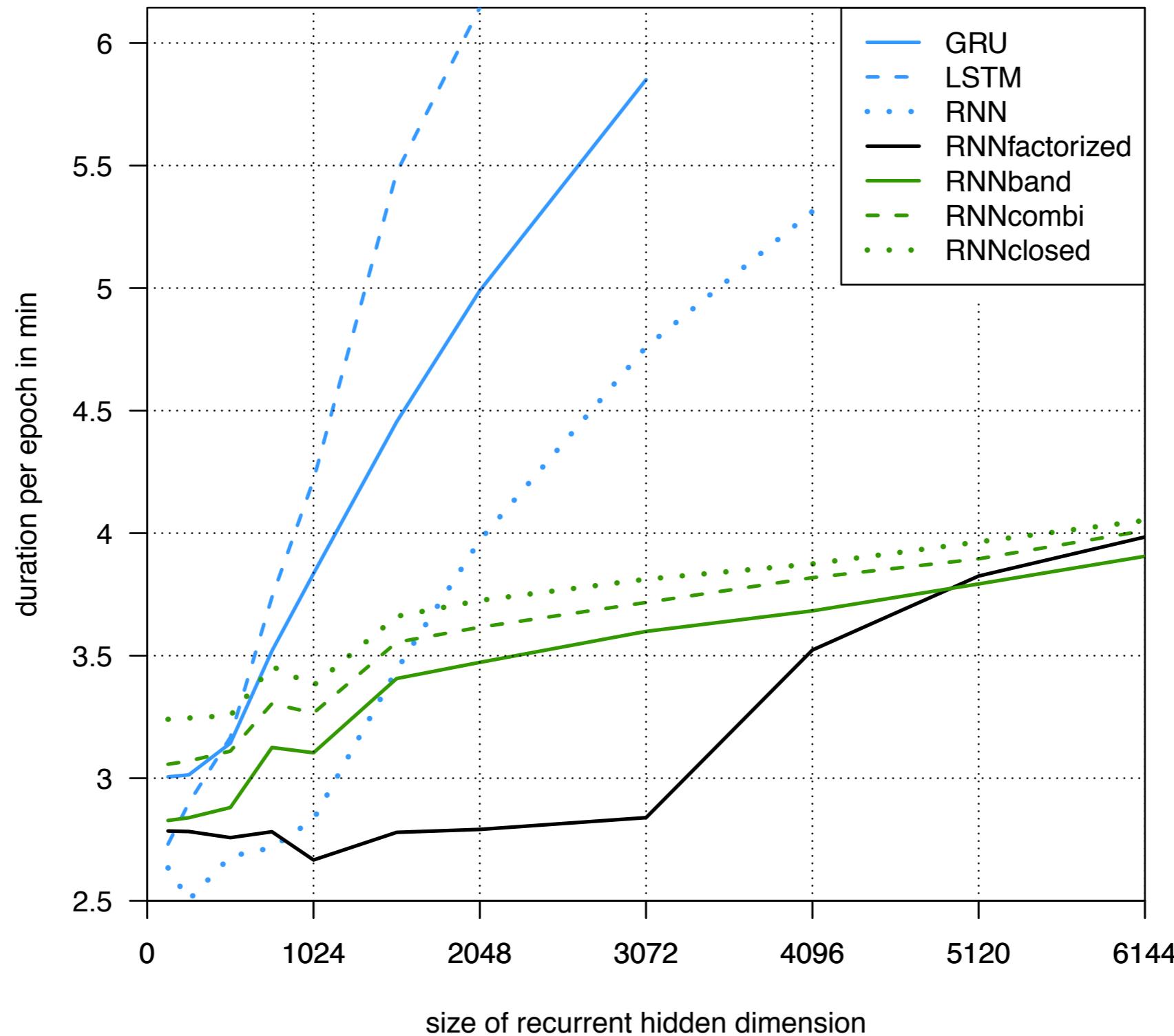
SG2BMM only calculated the blocks of C stored in blocked band format



# KERNEL BENCHMARK



# MODEL BENCHMARK



## COMPARABILITY

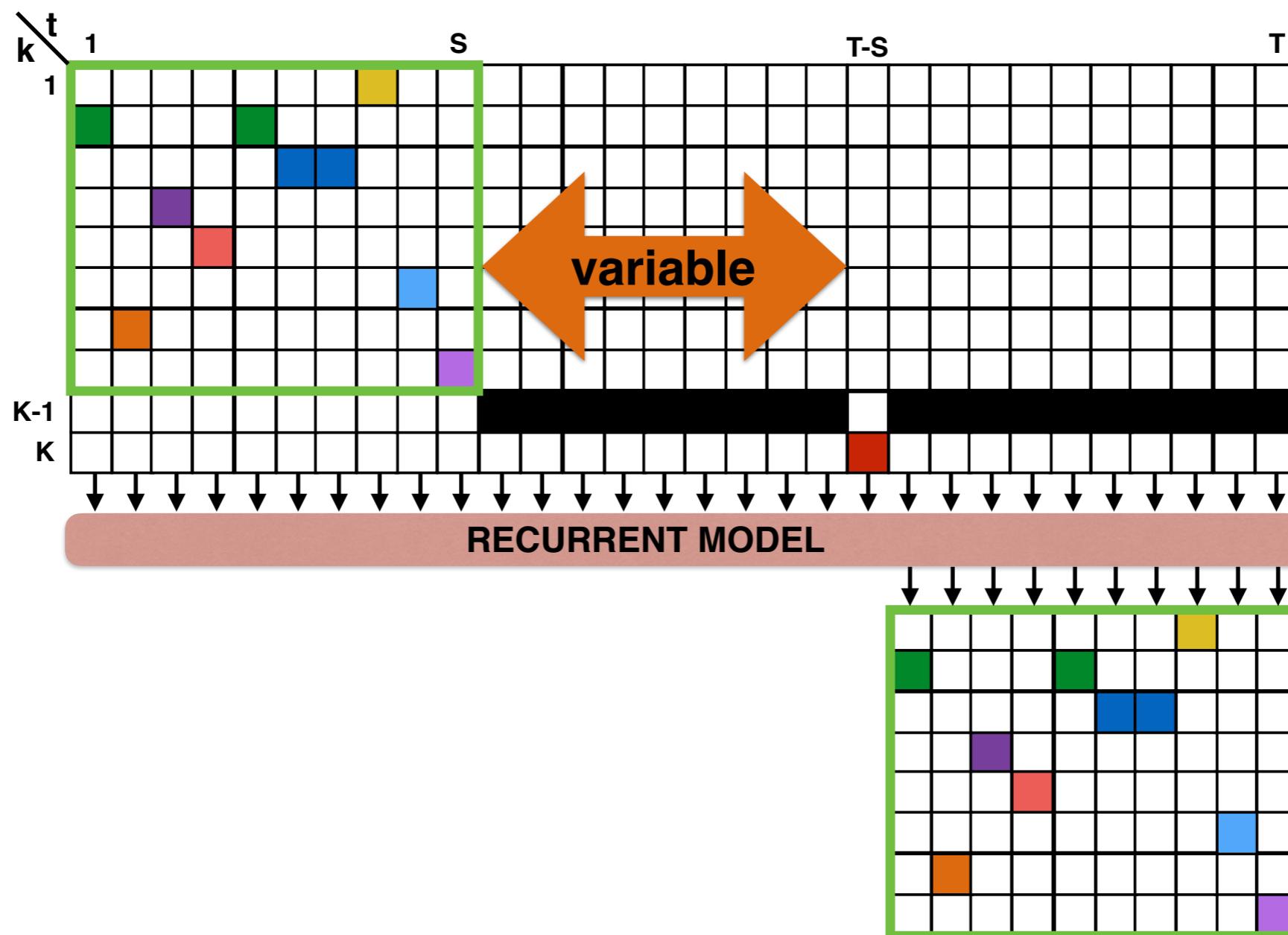
- fixed number of trainable parameters: 80k (540k for big model)
- band-half-width and factorizing dimension: 32

Example for Braille Task

model	hidden units
GRU	160
LSTM	128
RNN	256
Factorized	864
Band	896
Combi	864
Closed	864
Closed Fixed	864
Big Closed Fixed	5888

Note: an LSTM with 1000 hidden units has as much parameters as a  
band model (band-half-width of 32) with 55000 hidden units

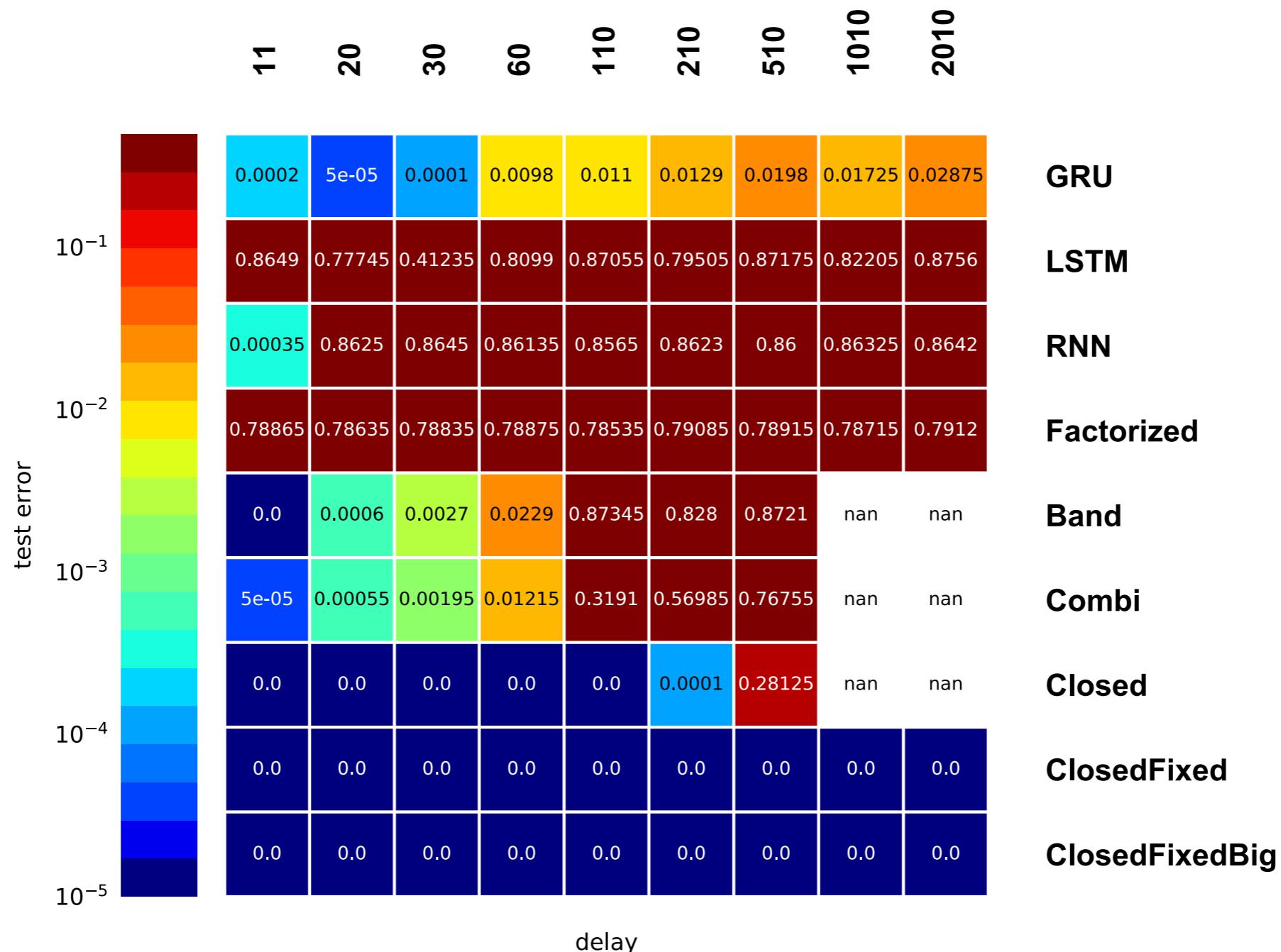
## COPY TASK - INPUT



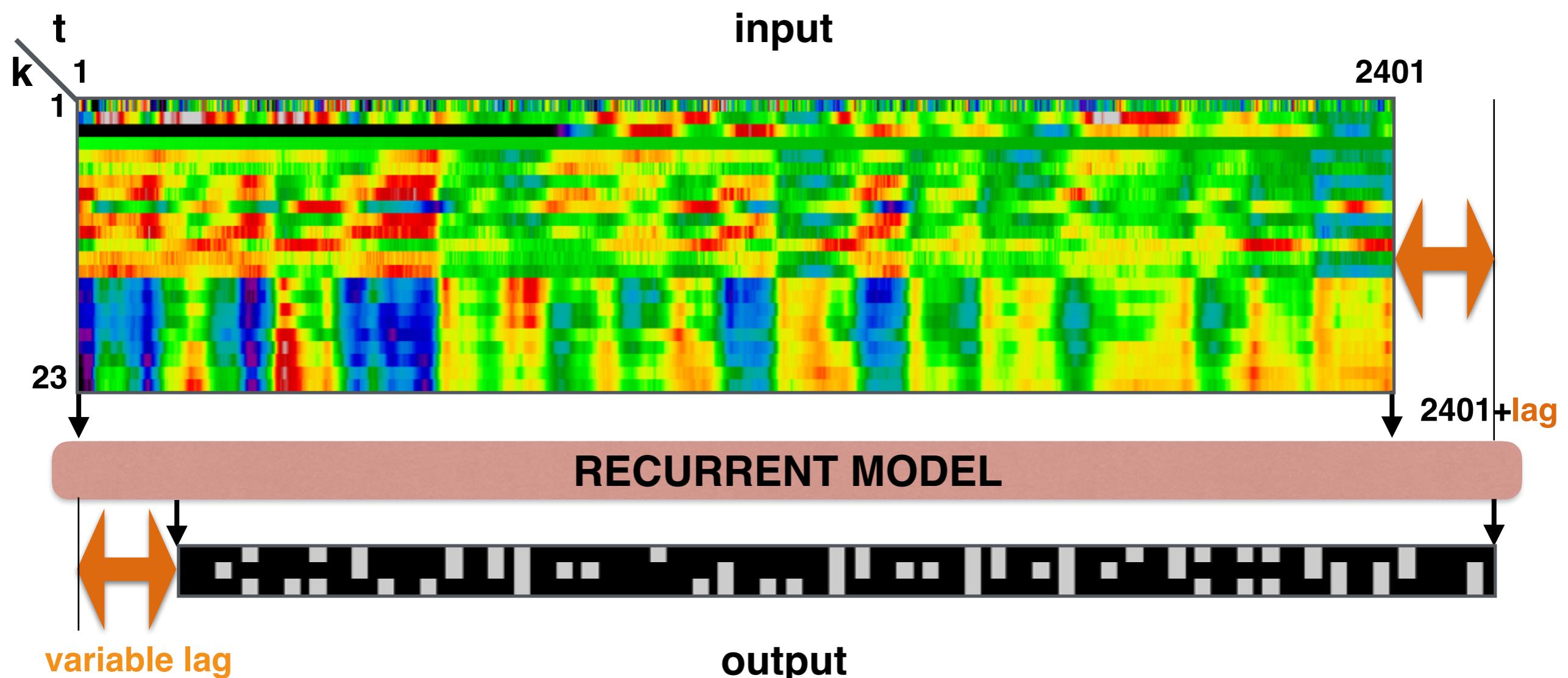
## EXPERIMENTS

---

# COPY TASK - RESULTS

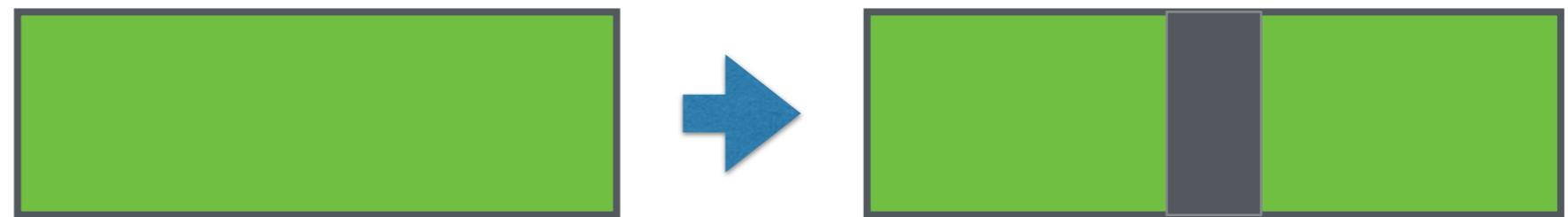


### BRAILLE TASK - INPUT



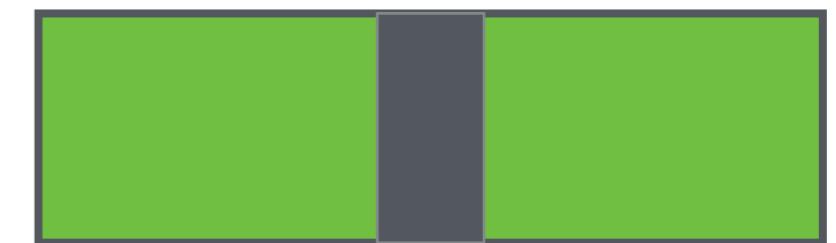
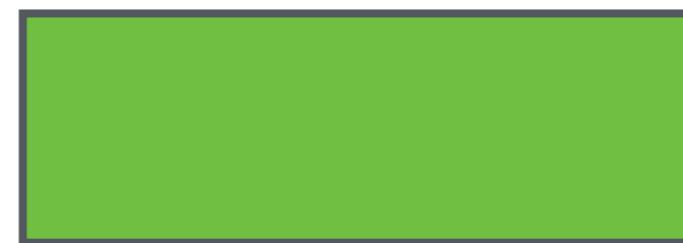
## BRAILLE TASK - DISTORTIONS

**paused**

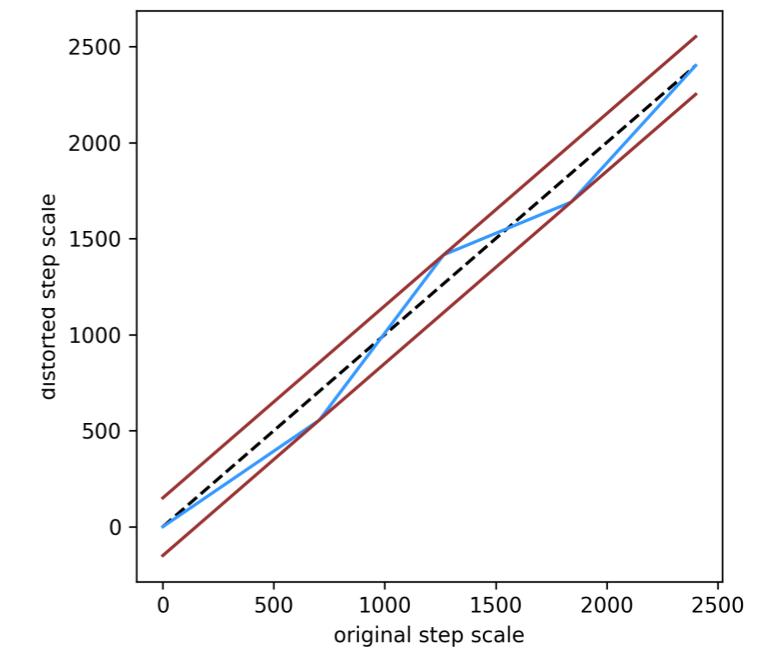
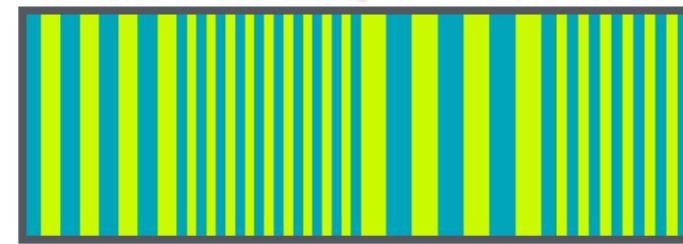
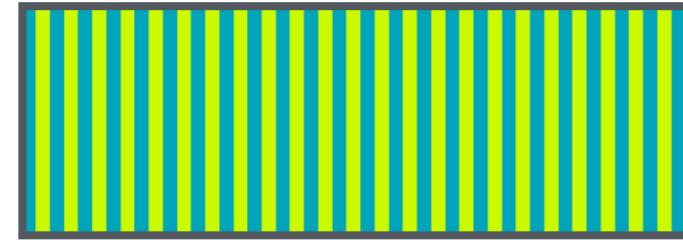


# BRAILLE TASK - DISTORTIONS

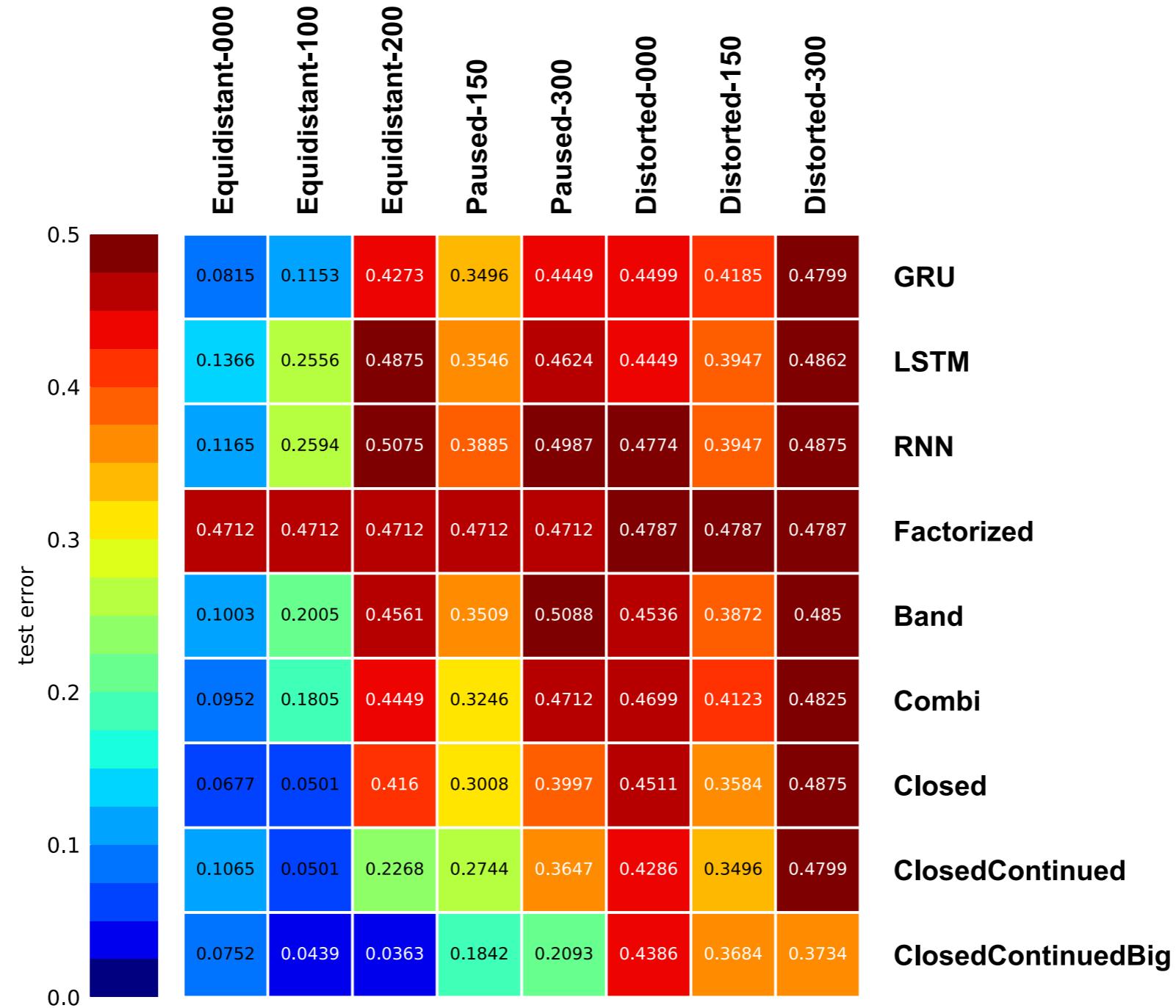
**paused**



**distorted**



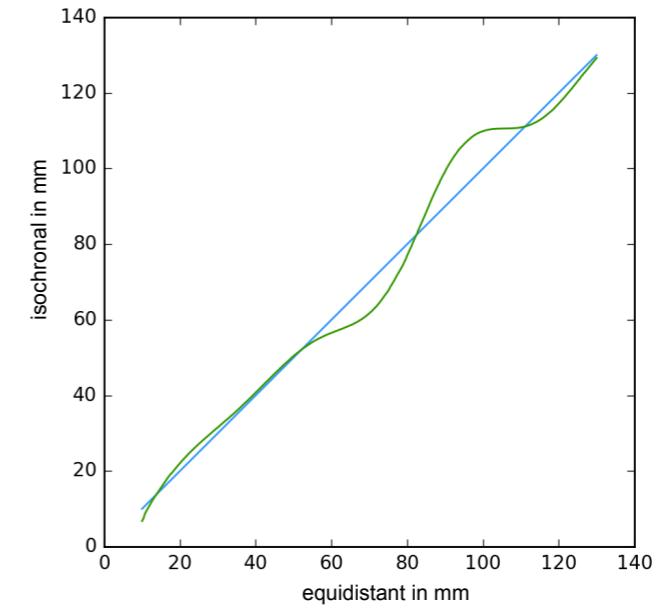
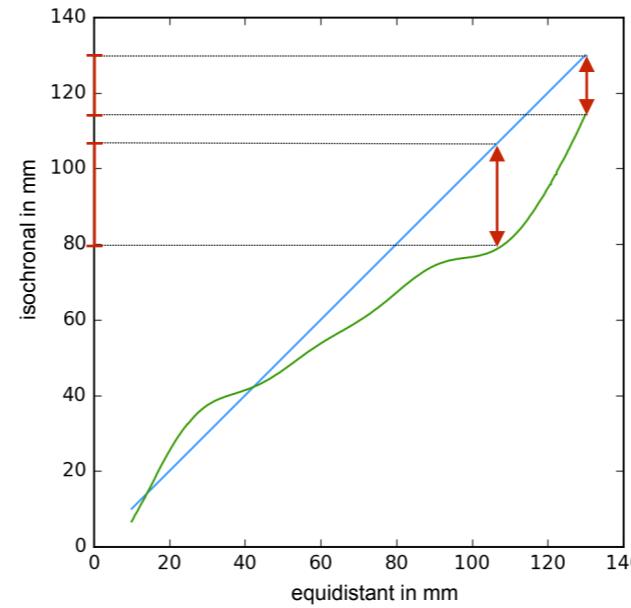
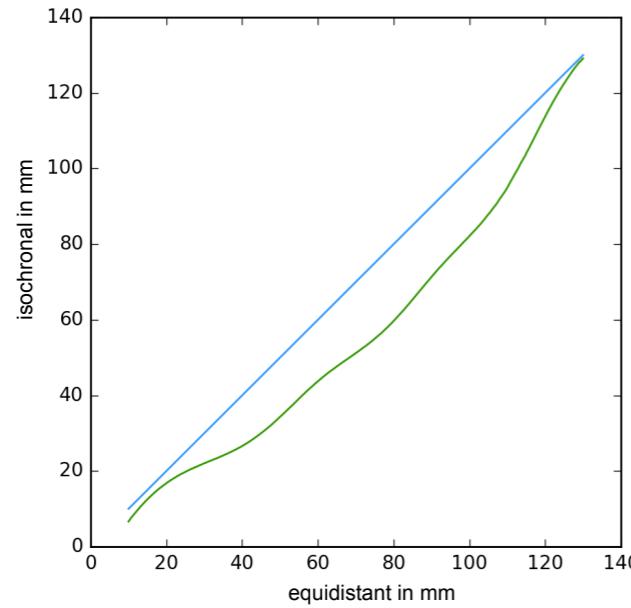
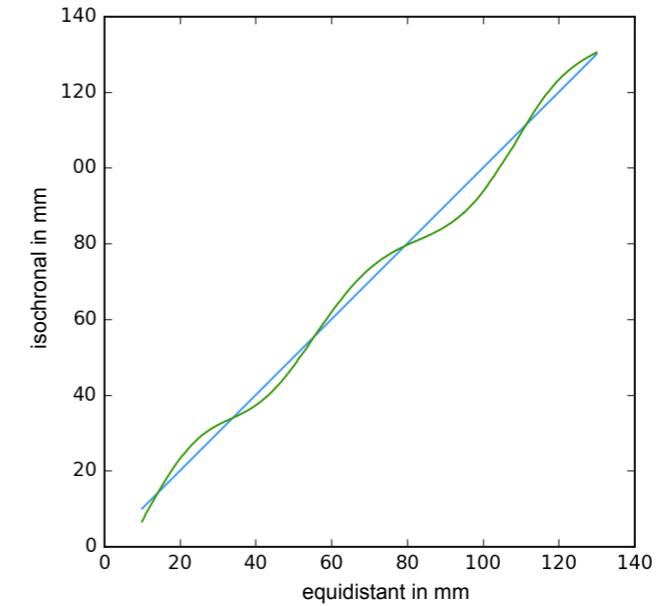
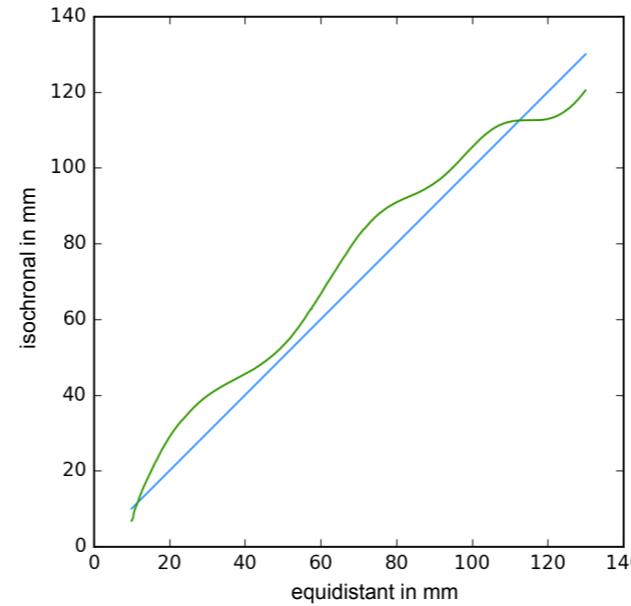
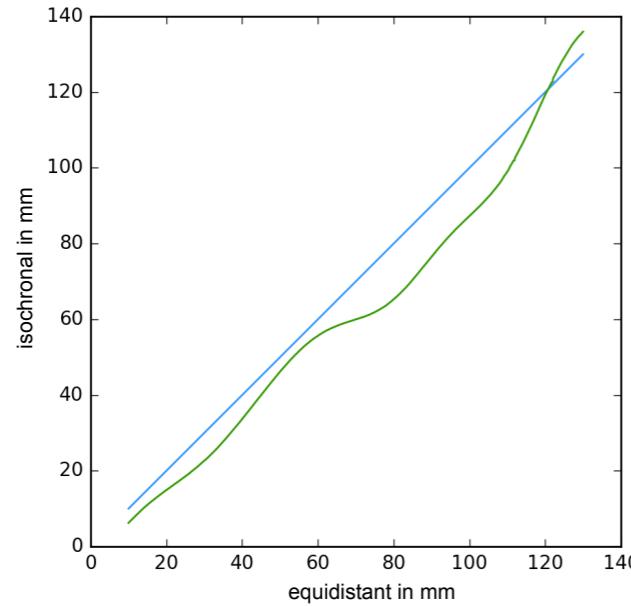
# BRAILLE TASK - CONSTANT SPEED RESULTS



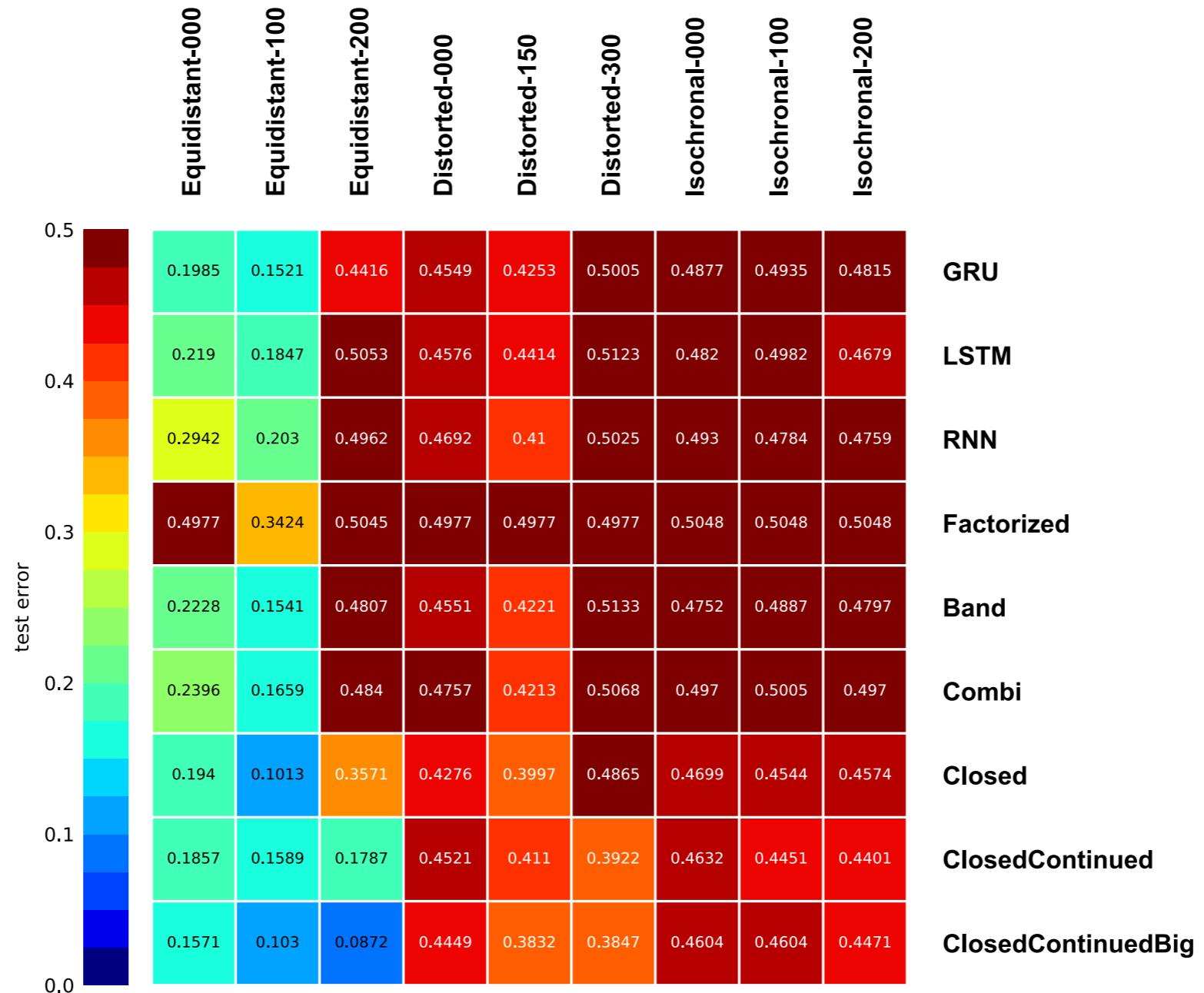
## EXPERIMENTS

---

# BRAILLE TASK - VARIABLE SPEED



# BRAILLE TASK - VARIABLE SPEED RESULTS



# EVALUATION

- classical models suffered overfitting for high delays  
→ regularization (dropout)

# EVALUATION

- classical models suffered overfitting for high delays  
→ regularization (dropout)
- closed band RNNs showed best performance (increased with hidden units and delay)  
→ comparison to the unitary RNN

# EVALUATION

- classical models suffered overfitting for high delays  
→ regularization (dropout)
- closed band RNNs showed best performance (increased with hidden units and delay)  
→ comparison to the unitary RNN
- fixed shift initialization required for high delays  
→ epoch wise increasing band-half-with

# EVALUATION

- classical models suffered overfitting for high delays
  - regularization (dropout)
- closed band RNNs showed best performance (increased with hidden units and delay)
  - comparison to the unitary RNN
- fixed shift initialization required for high delays
  - epoch wise increasing band-half-with
- band RNNs shift capacity from task to input history storage
  - combinations of band RNNs and classical RNNs

# EVALUATION

- classical models suffered overfitting for high delays
  - regularization (dropout)
- closed band RNNs showed best performance (increased with hidden units and delay)
  - comparison to the unitary RNN
- fixed shift initialization required for high delays
  - epoch wise increasing band-half-with
- band RNNs shift capacity from task to input history storage
  - combinations of band RNNs and classical RNNs
- braille task required high delays and input history storage capacity
  - more tests on natural data sets

# THANK YOU