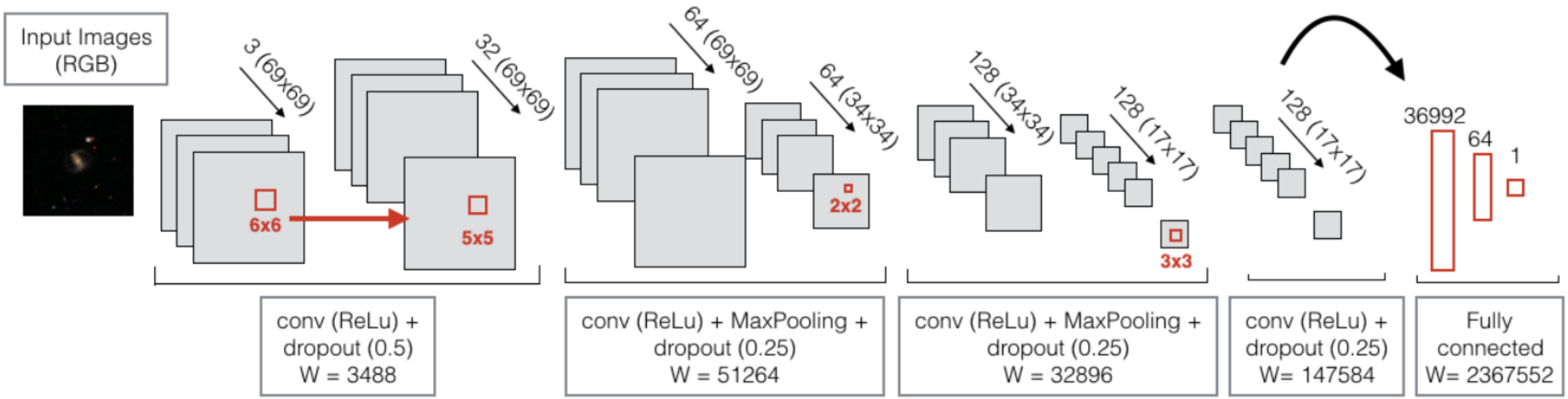
BEYOND CLASSIFICATION: IMAGE2IMAGE NETWORKS

UP TO NOW CNNs MAP IMAGES (SIGNALS) INTO FLOATS



Dominguez-Sanchez+18

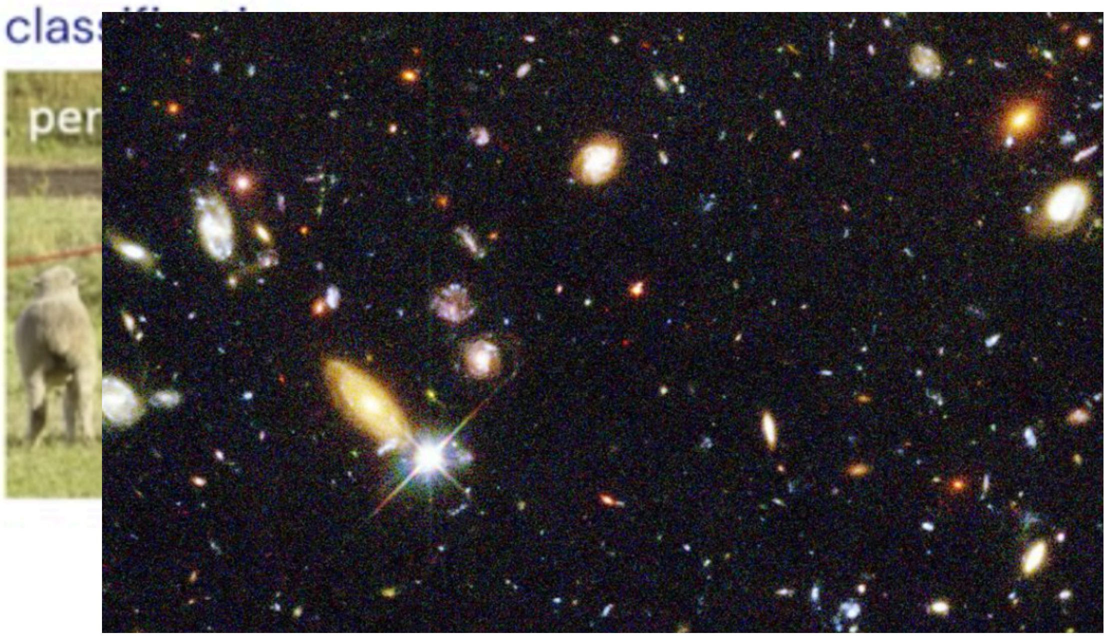
Classification has its limits ….

HOW DO I CLASSIFY THIS IMAGE?

Classification has its limits ….

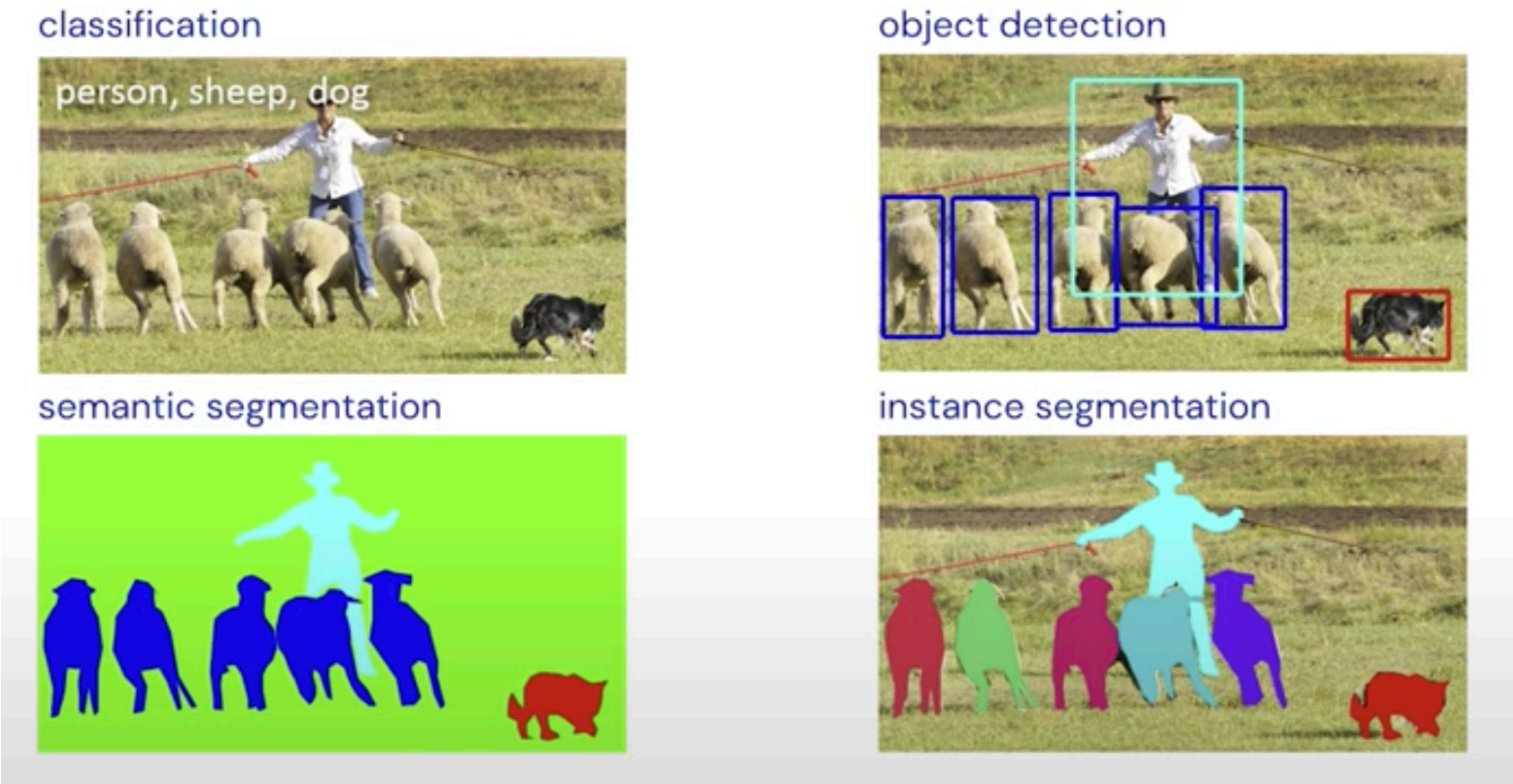
HOW DO I CLASSIFY THIS IMAGE?

Classification has its limits ….



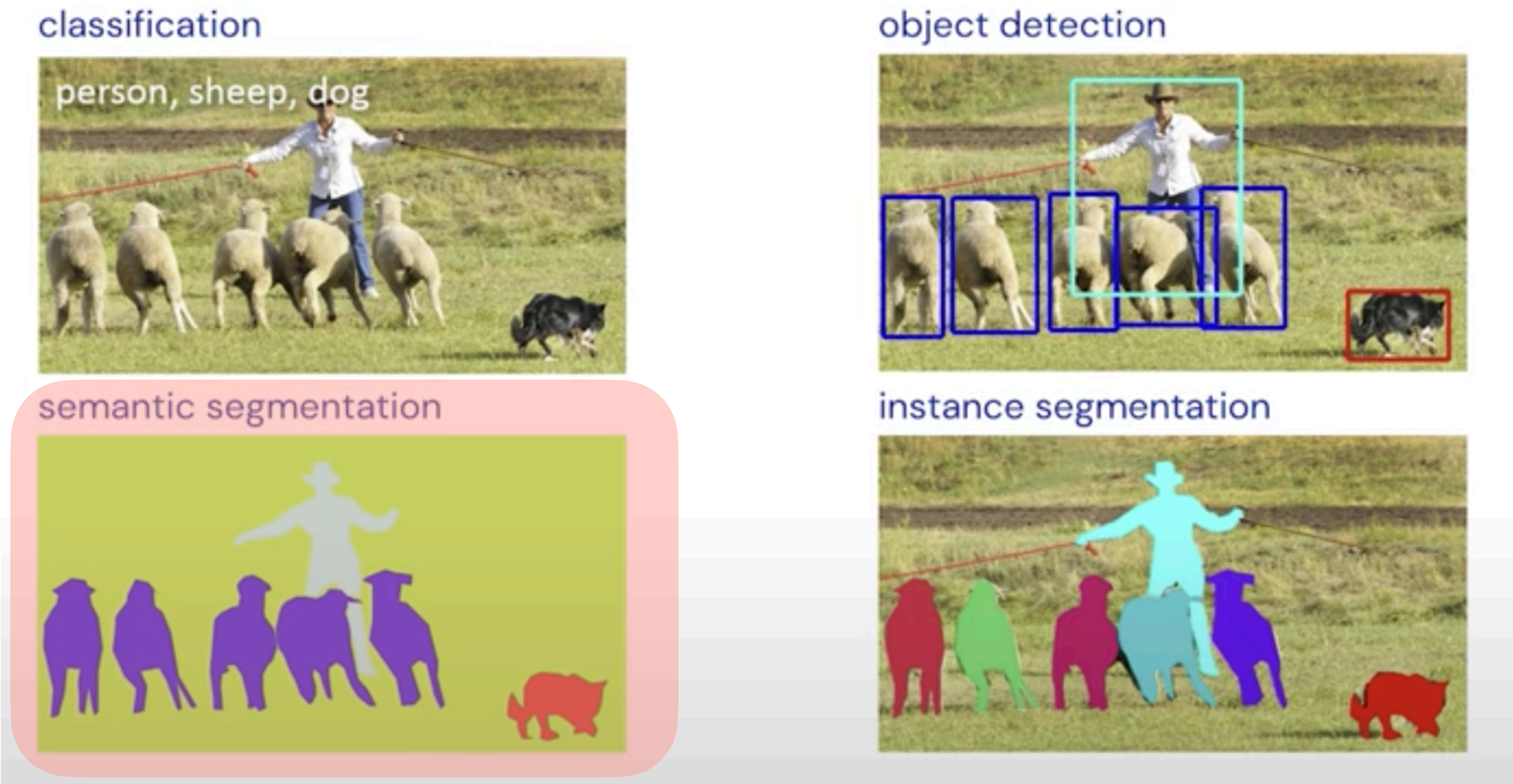
HOW DO I CLASSIFY THIS IMAGE?

Going beyond classification: increasing complexity



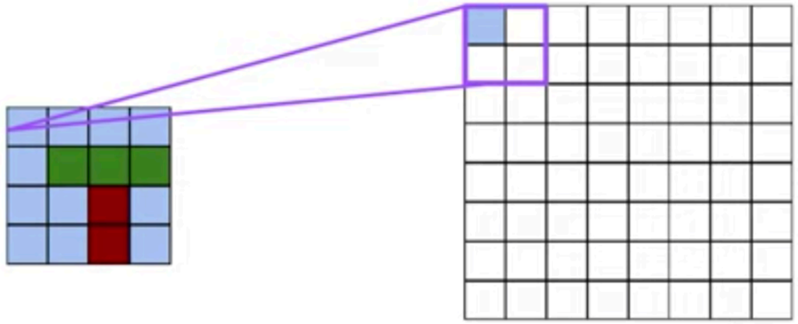
@deepmind (V. Patraucean)

Going beyond classification: increasing complexity

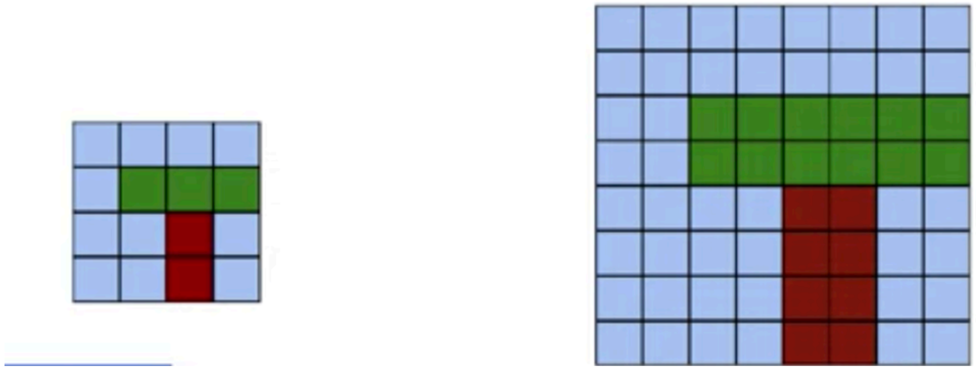


@deepmind (V. Patraucean)

UNPOOLING OPERATION (INVERSE OF POOLING)



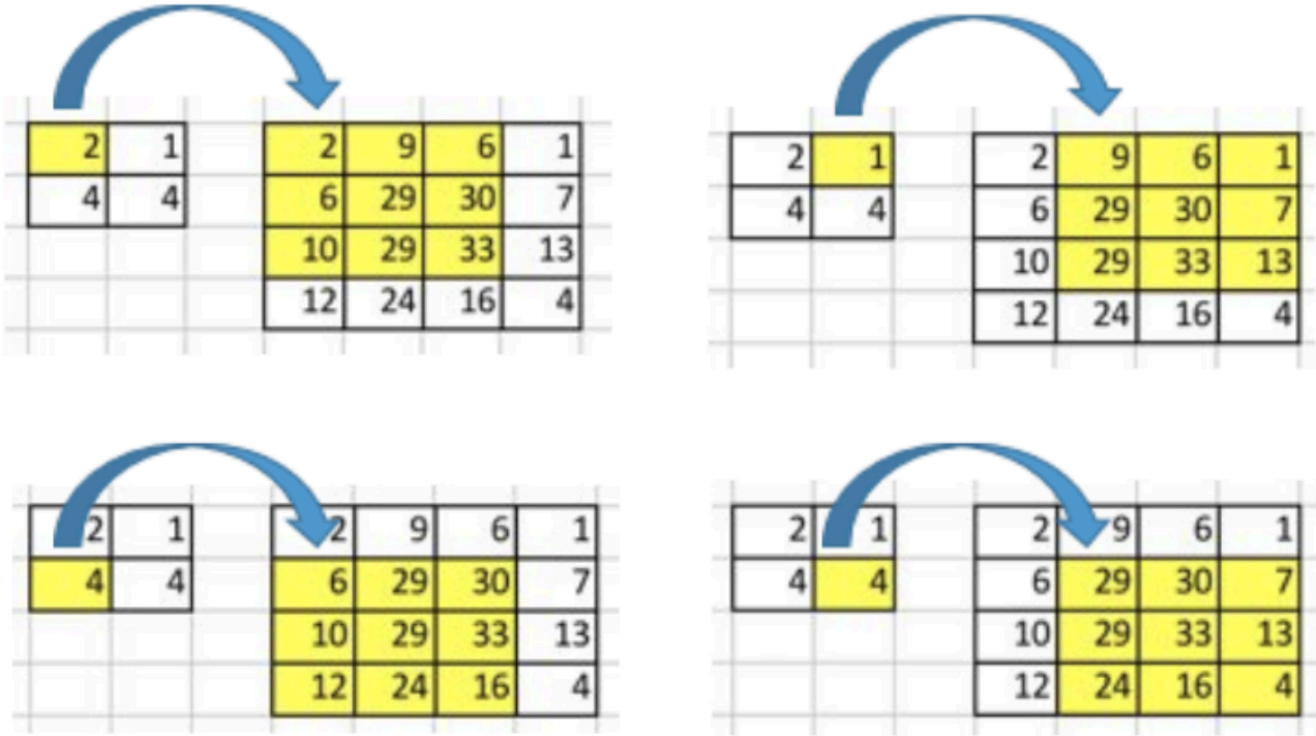
COPY PIXELS IN A GIVEN WINDOW



GENERATES LARGER IMAGES FROM SMALLER ONES

TRANSPOSED CONVOLUTION

ALLOWS TO INCREASE THE SIZE



Going Backward of Convolution

EXAMPLE TAKEN FROM HERE

CONVOLUTION MATRIX

	0	1	2
0	1	4	1
1	1	4	3
2	3	3	1

Kernel (3, 3)

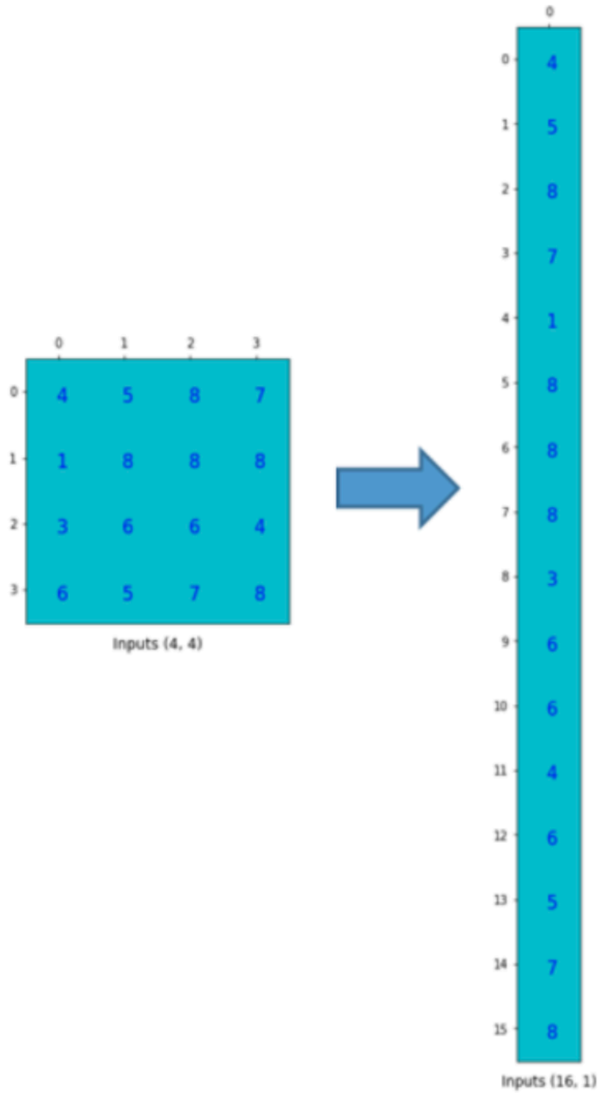
THE KERNEL CAN BE ARRANGED IN FORM OF A MATRIX:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	4	1	0	1	4	3	0	3	3	1	0	0	0	0	0
1	0	1	4	1	0	1	4	3	0	3	3	1	0	0	0	0
2	0	0	0	0	1	4	1	0	1	4	3	0	3	3	1	0
3	0	0	0	0	0	1	4	1	0	1	4	3	0	3	3	1

Convolution Matrix (4, 16)

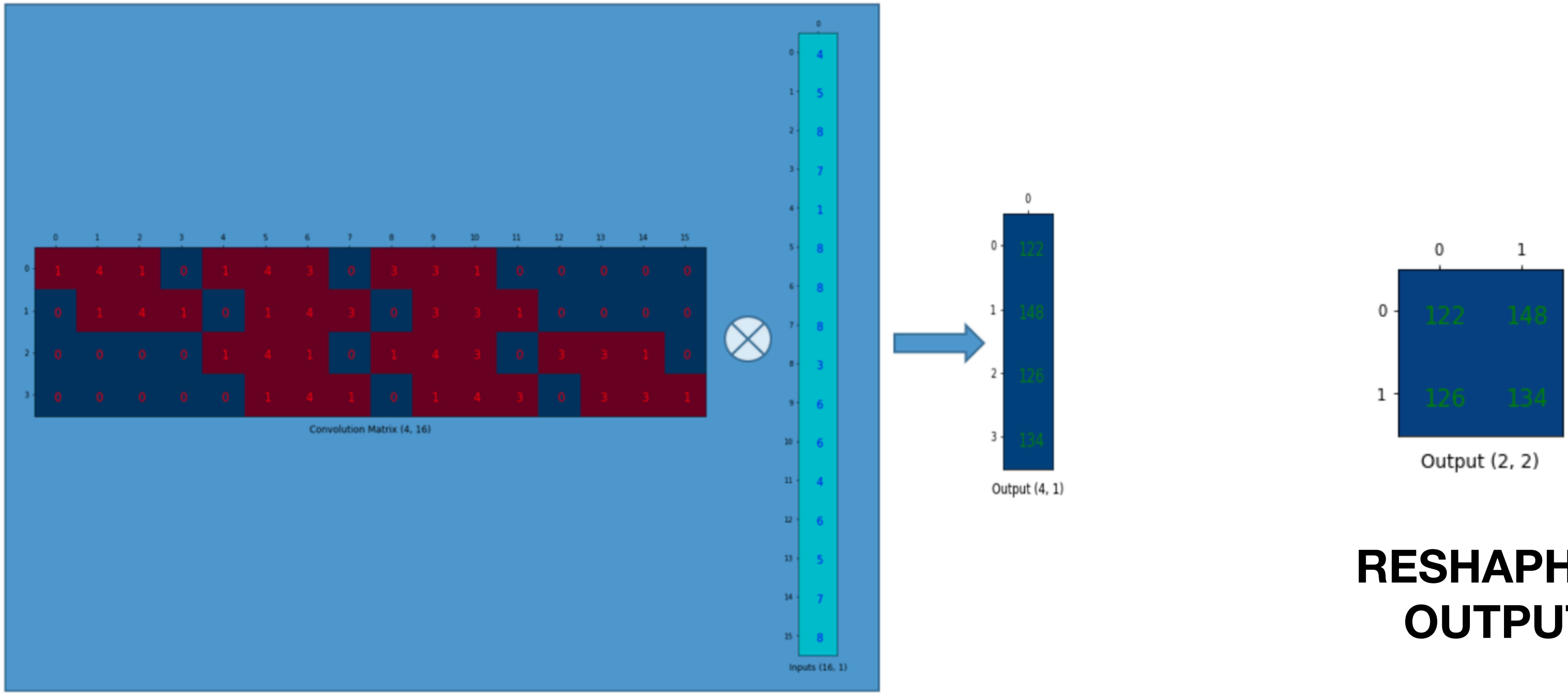
<u>EXAMPLE TAKEN FROM HERE</u>

THE INPUT IS FLATTENED INTO A COLUMN VECTOR



EXAMPLE TAKEN FROM HERE

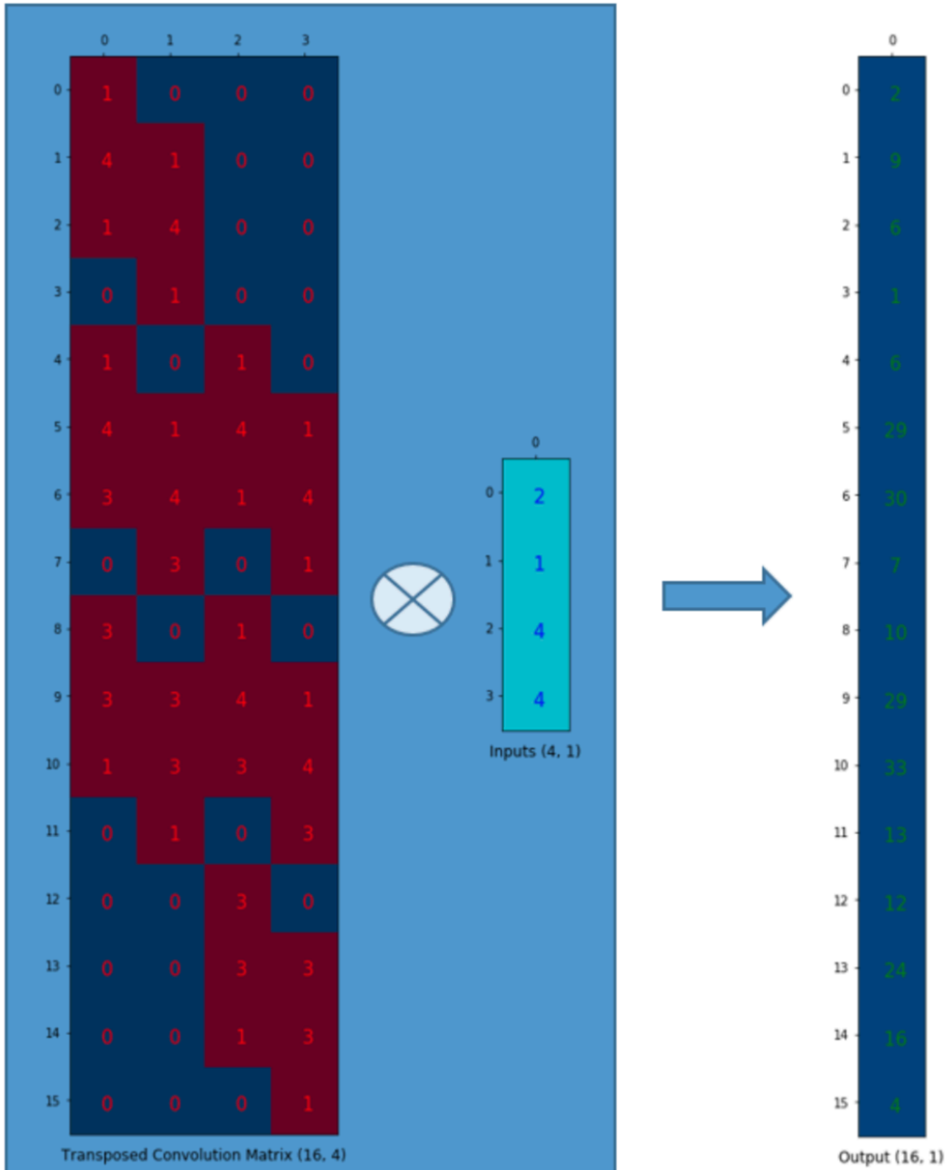
THE CONVOLUTION IS TRANSFORMED INTO A PRODUCT OF MATRICES

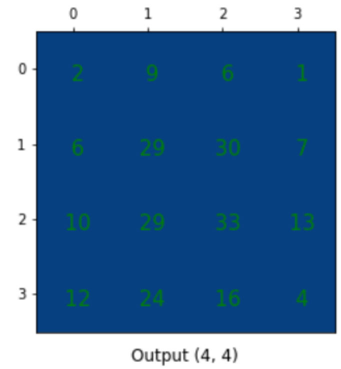


RESHAPED OUTPUT

EXAMPLE TAKEN FROM HERE

THE TRANSPOSED CONVOLUTION IS THE INVERSE OPERATION

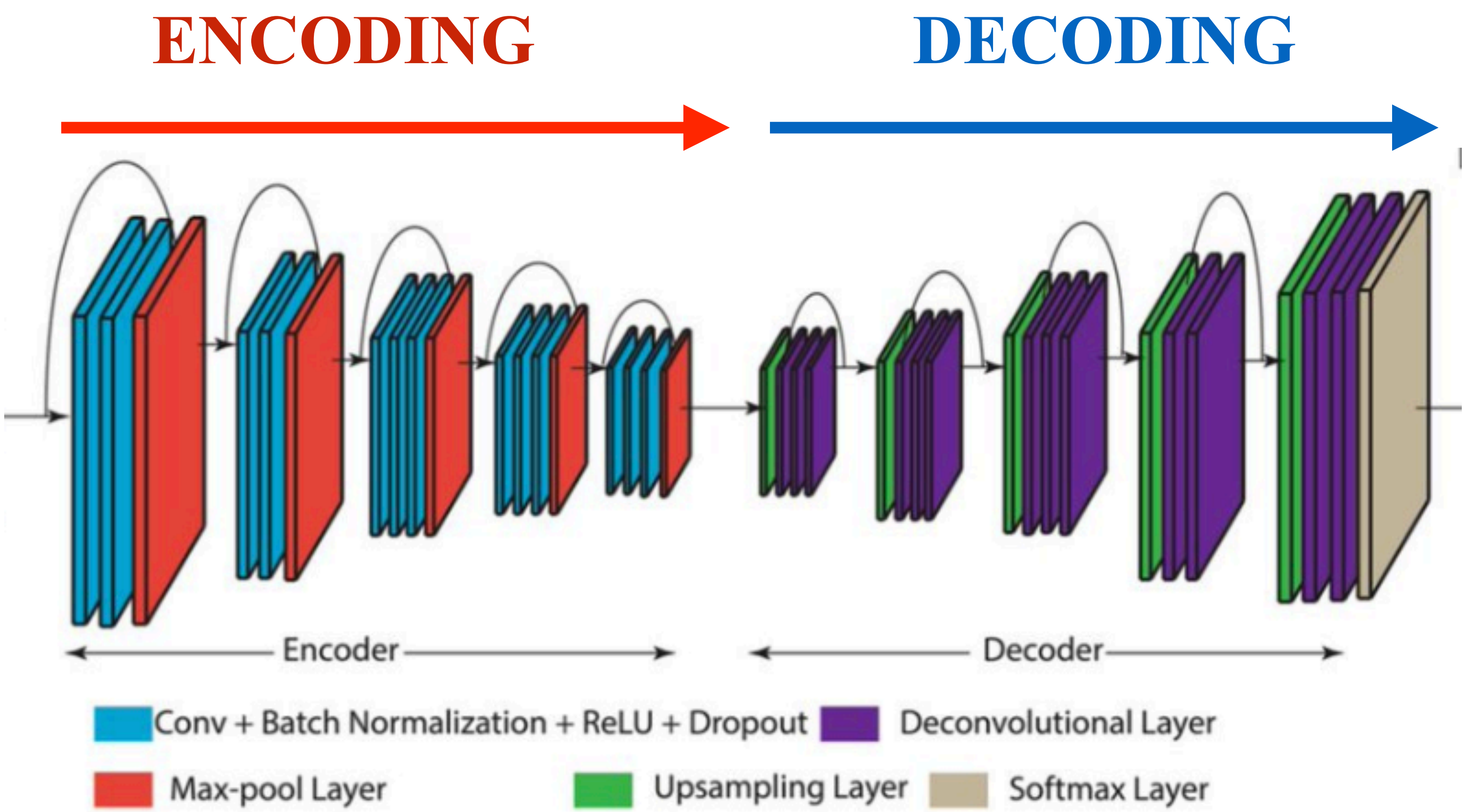




RESHAPED OUTPUT

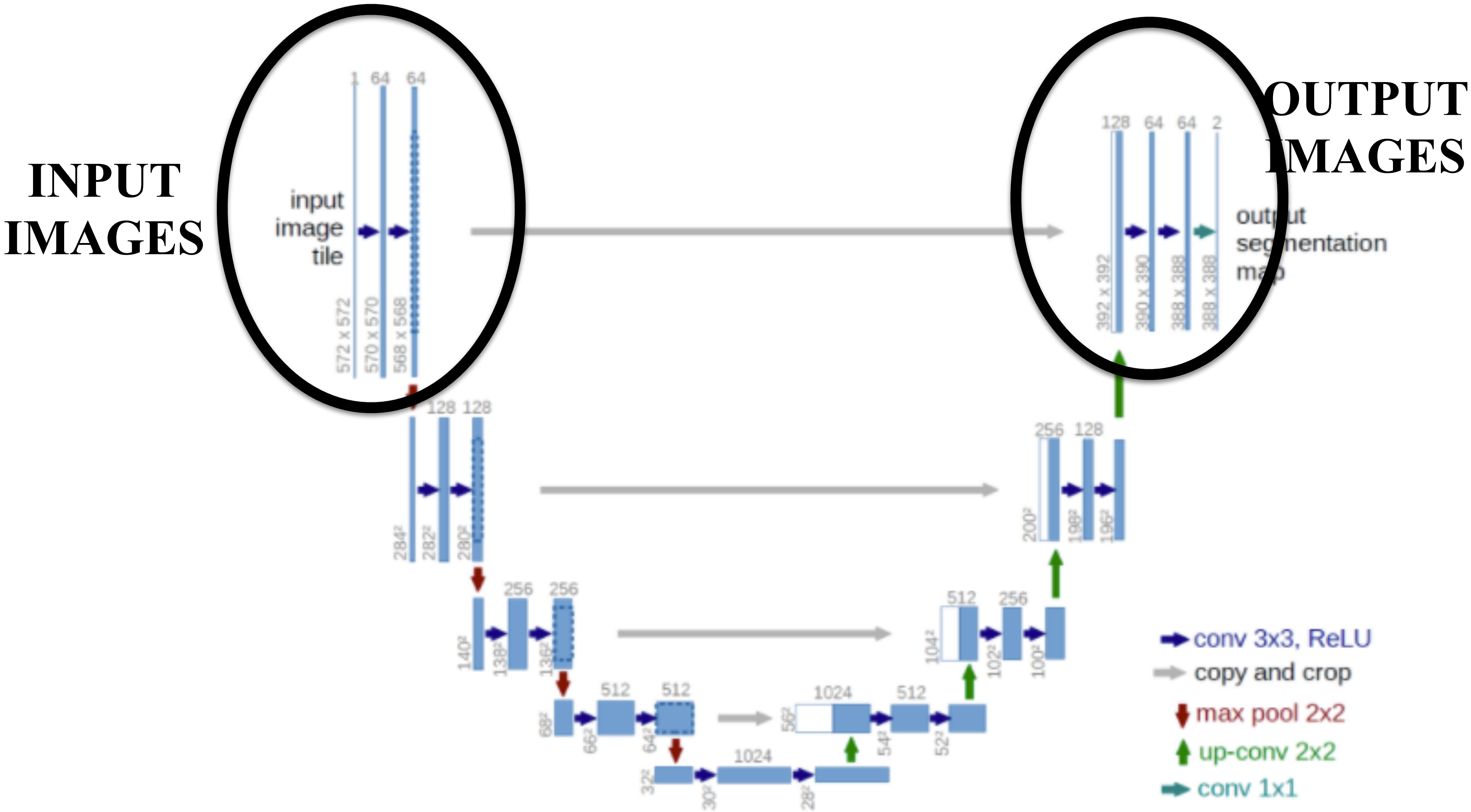
EXAMPLE TAKEN FROM HERE

ENCODER-DECODERS GO FROM IMAGE 2 IMAGE



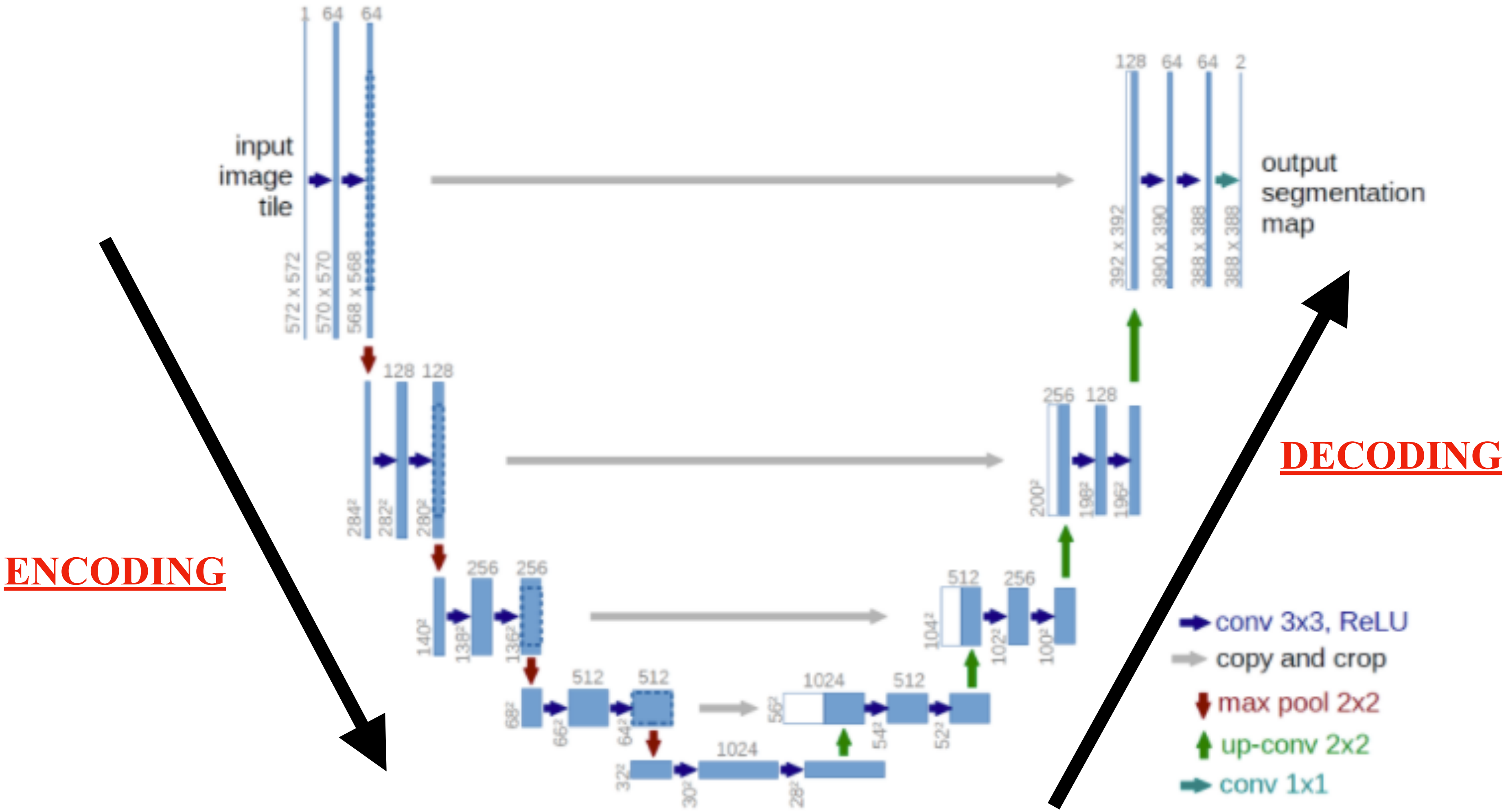
WE CALL THIS FULLY CONVOLUTIONAL NEURAL NETWORKS

ENCODING-DECODING TO EXTRACT IMAGE FEATURES: U-NET



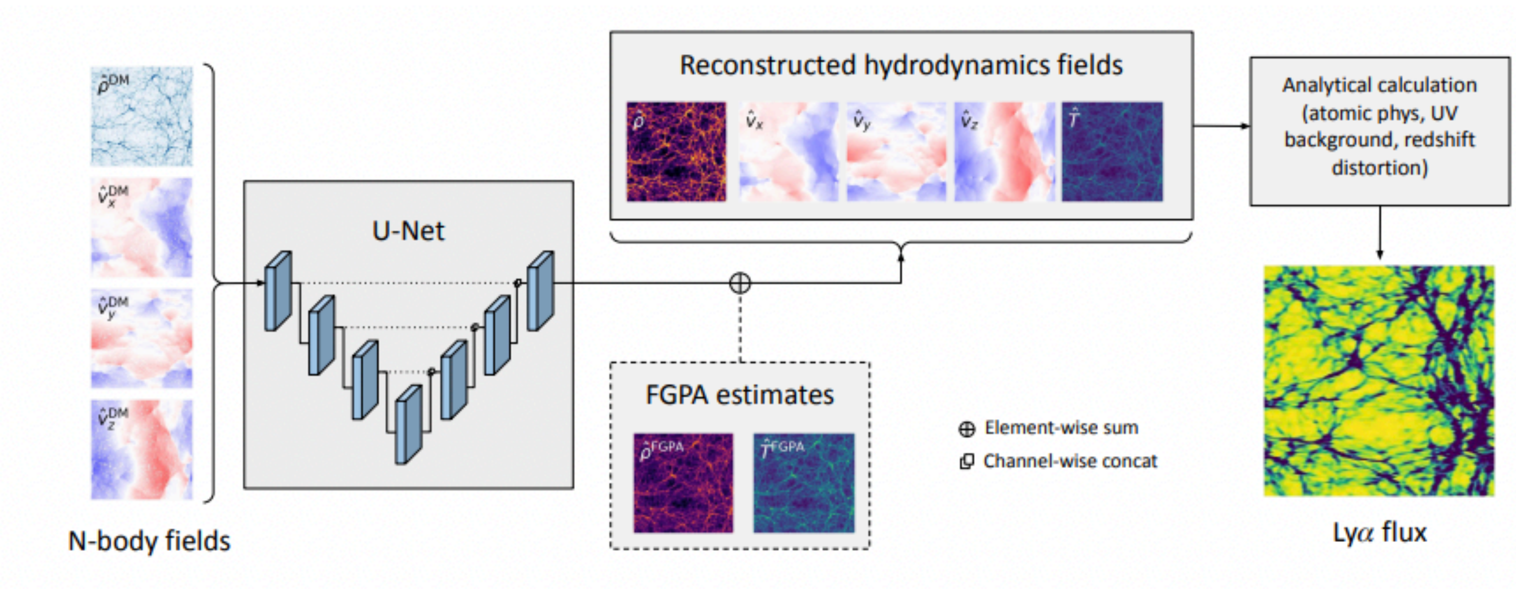
Ronnerberger+15

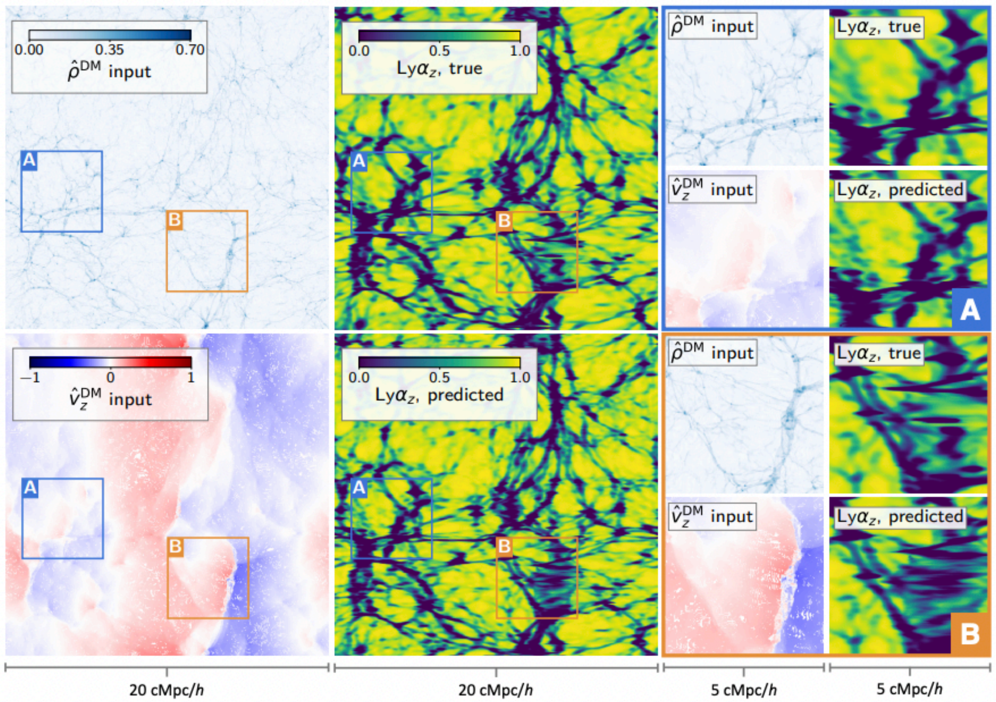
ENCODING-DECODING TO EXTRACT IMAGE FEATURES: <u>THE U-NET</u>



Ronnerberger+15

Painting Baryons





Harrington+21

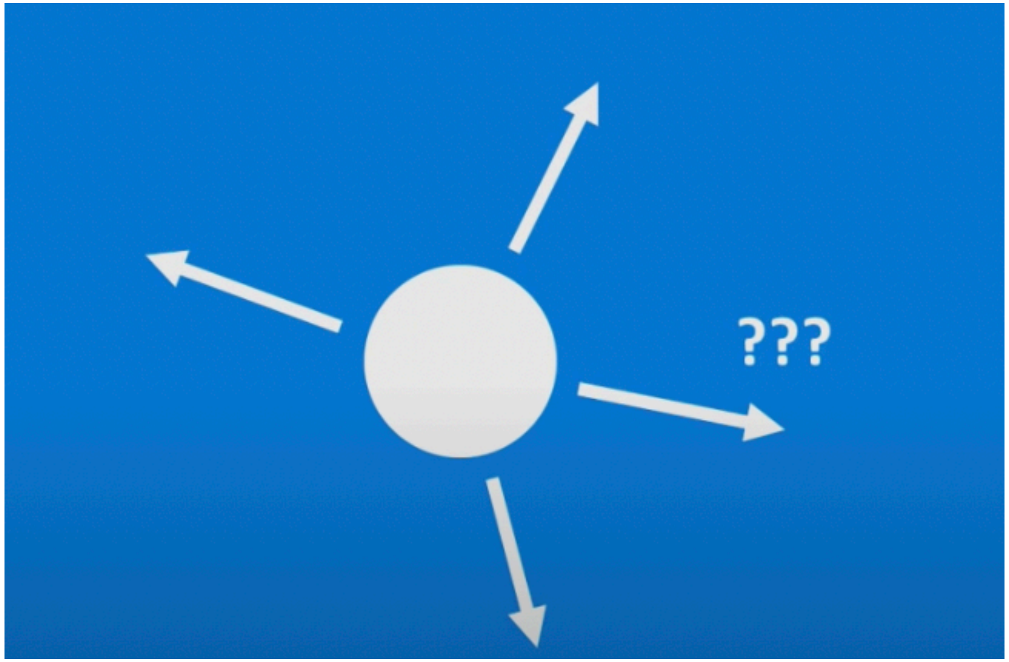
Neural Networks are used to learn the non-linear mapping between cheap dark matter only simulations to expensive baryonic physics

Rodriguez+19, Modi+18, Berger+18, He+18, Zhang+19, Troster+19, Zamudio-Fernandez+19, Perraudin+19, Charnock+19, List+19, Giusarma+19, Bernardini+19, Chardin+19, Mustafa+19, Ramanah+20, Tamosiunas+20, Feder+20, Moster+20, Thiele+20, Wadekar+20, Dai+20, Li+20, Lucie-Smith+20, Kasmanoff+20, Ni+21, Rouhiainen+21, Harrington+21, Horowitz+21, Horowitz+21, Bernardini+21, Schaurecker+21, Etezad-Razavi+21, Curtis+21

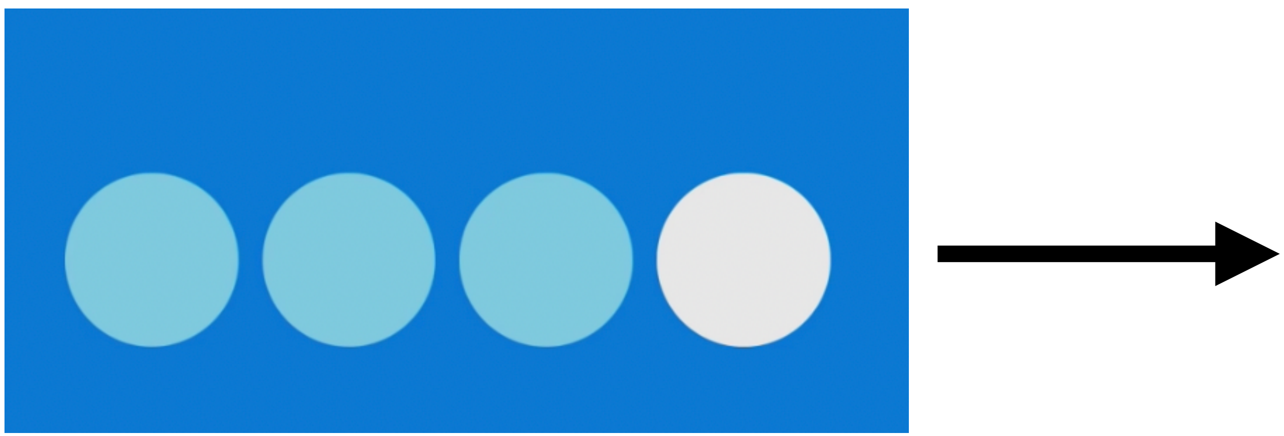
SEQUENCE MODELLING

*based on MIT Lecture by Ava Soleimany

GIVEN AN IMAGE OF A BALL, CAN YOU PREDICT WHERE IT WILL GO NEXT?



GIVEN AN IMAGE OF A BALL, CAN YOU PREDICT WHERE IT WILL GO NEXT?



Previous positions help guessing the future

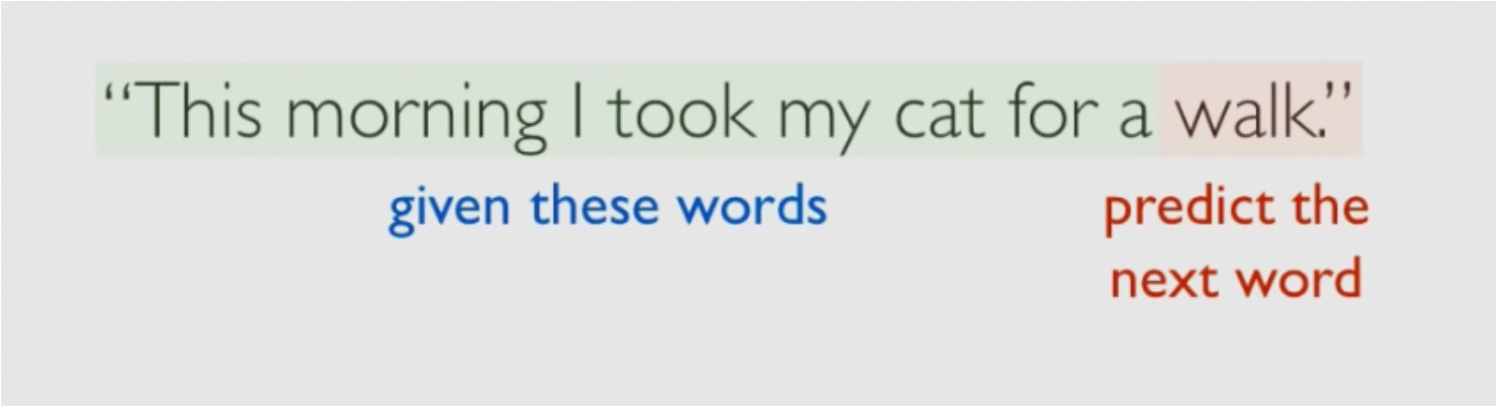
LET'S TAKE A SIMPLE EXAMPLE OF LANGUAGE MODELLING

"This morning I took my cat for a walk."

given these words

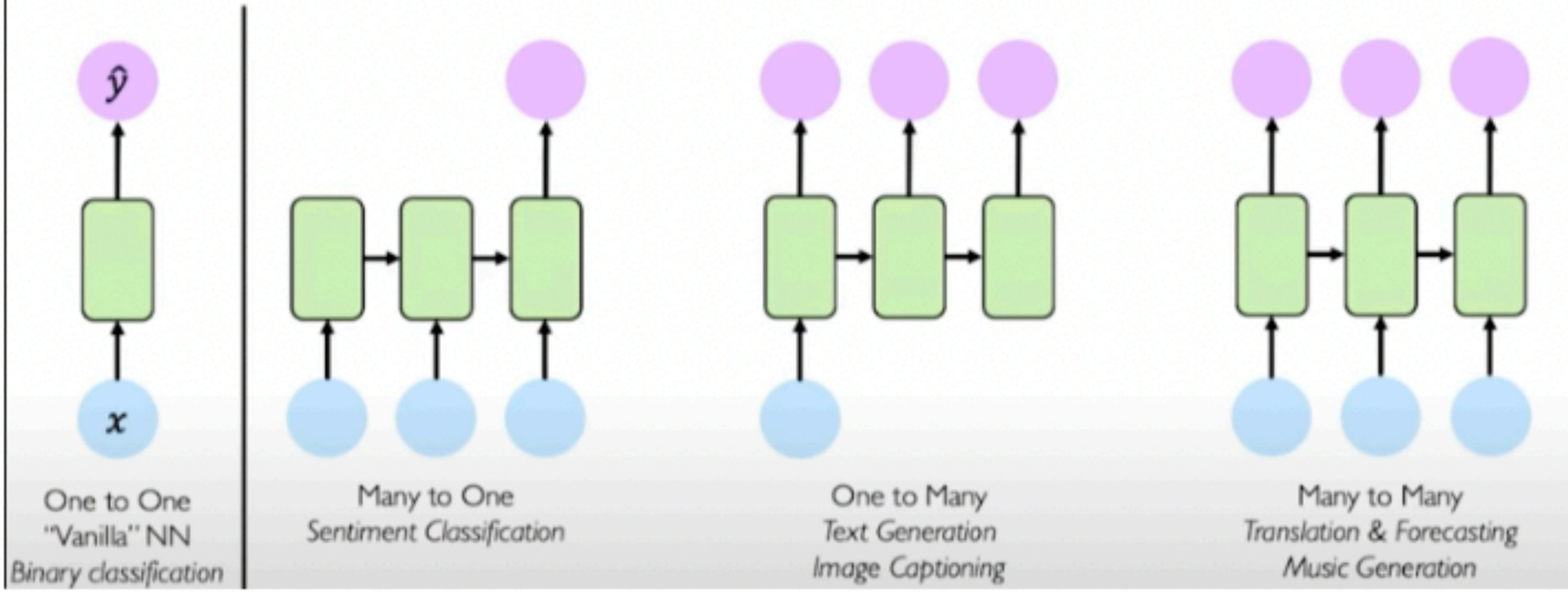
predict the next word

LET'S TAKE A SIMPLE EXAMPLE OF LANGUAGE MODELLING

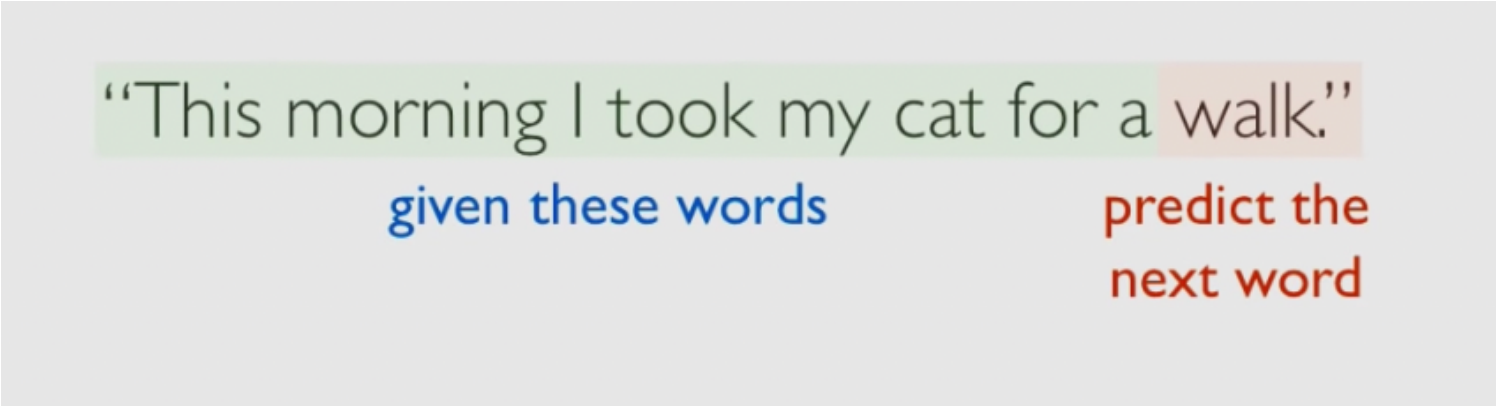


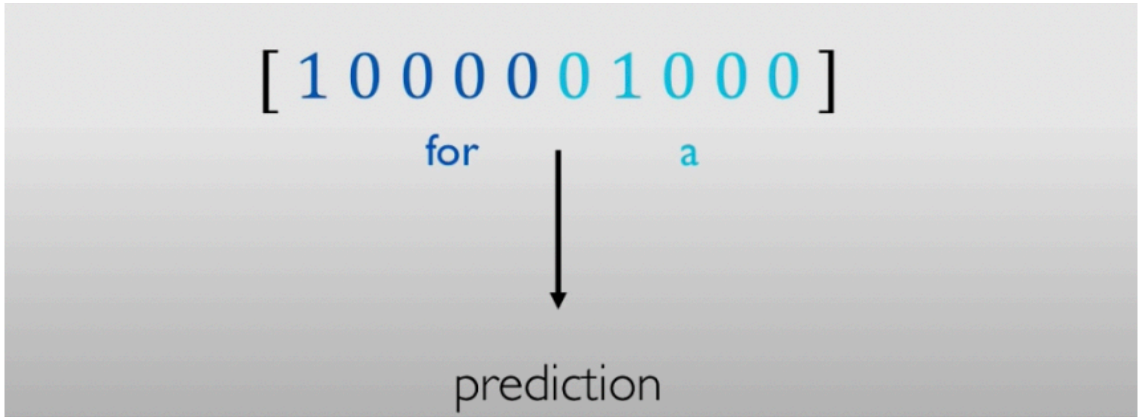
Normal ANNs and CNNs cannot handle variable length inputs …

RNNs for Sequence Modeling



WE COULD SIMPLY USE A FIXED WINDOW…

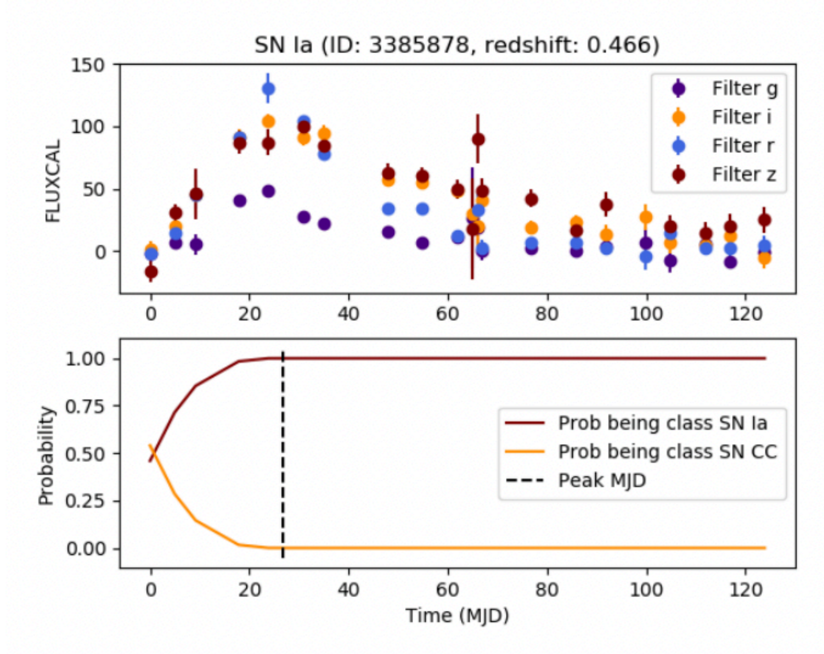




HOWEVER THIS CANNOT HANDLE LONG TERM MEMORY

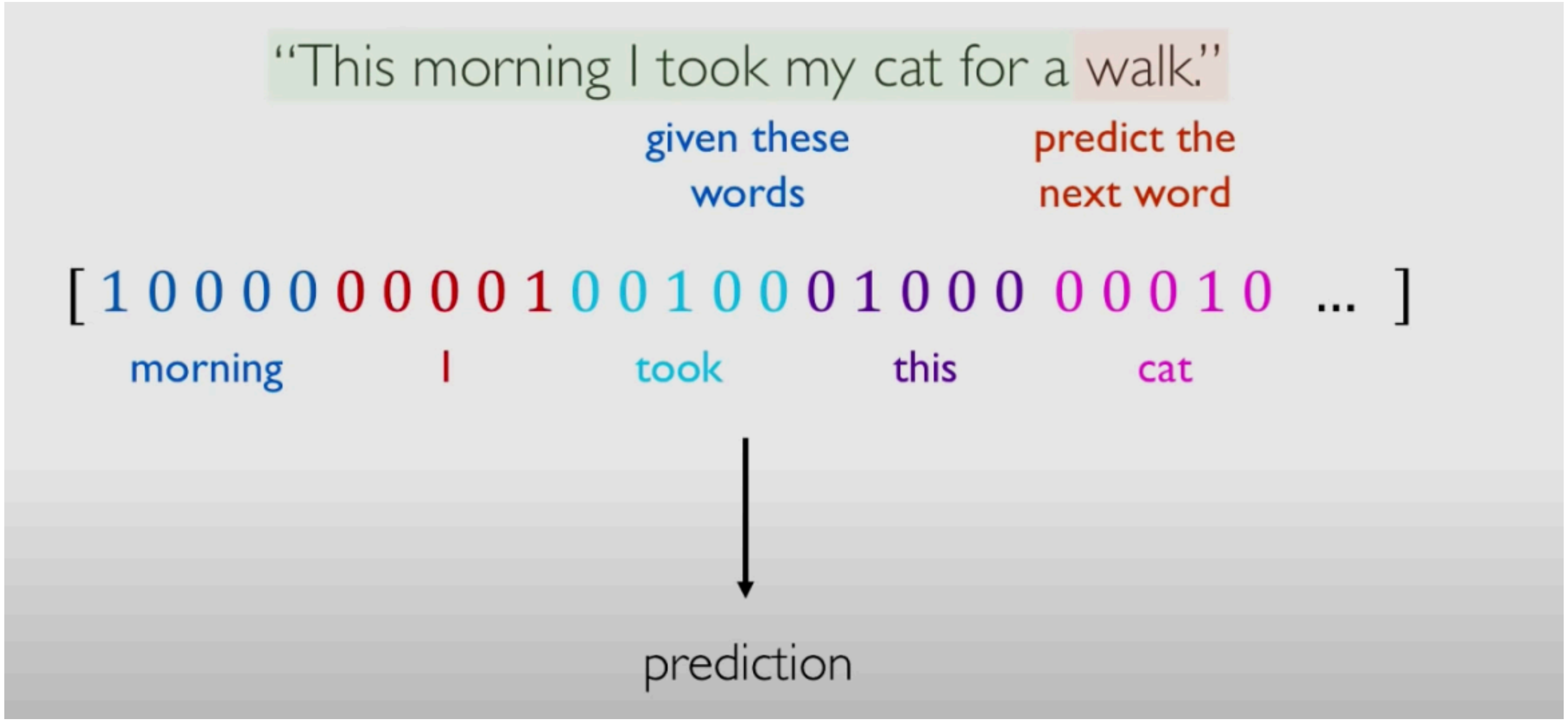
"**France** is where I grew up, but I now live in Boston. I speak fluent ____."

IN SOME CASES, INFORMATION FROM THE DISTANT PAST IS NEEDED FOR A CORRECT PREDICTION



Moller+19

ANOTHER ALTERNATIVE COULD BE TO FEED A REALLY LARGE WINDOW



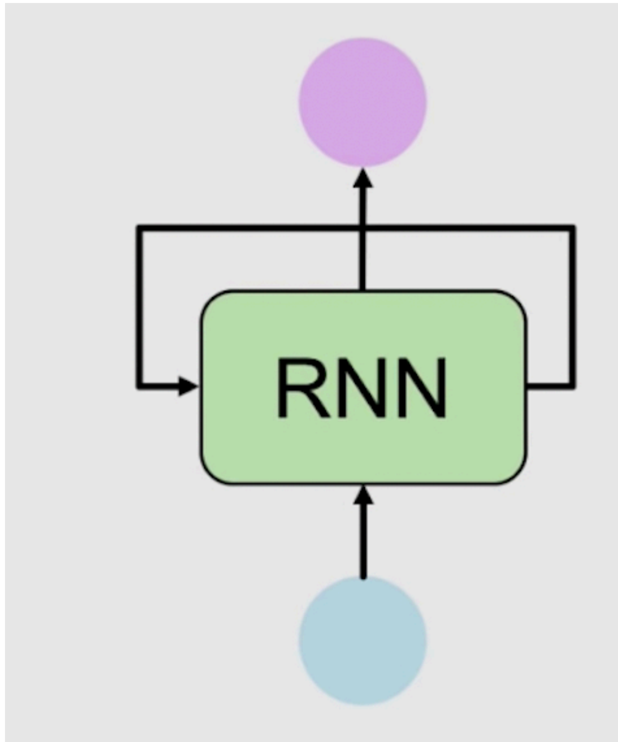
BUT WE STILL HAVE THE PROBLEM THAT THERE IS NO PARAMETER SHARING (SAME AS PIXELS)

RECURRENT NEURAL NETWORKS (RNNs)

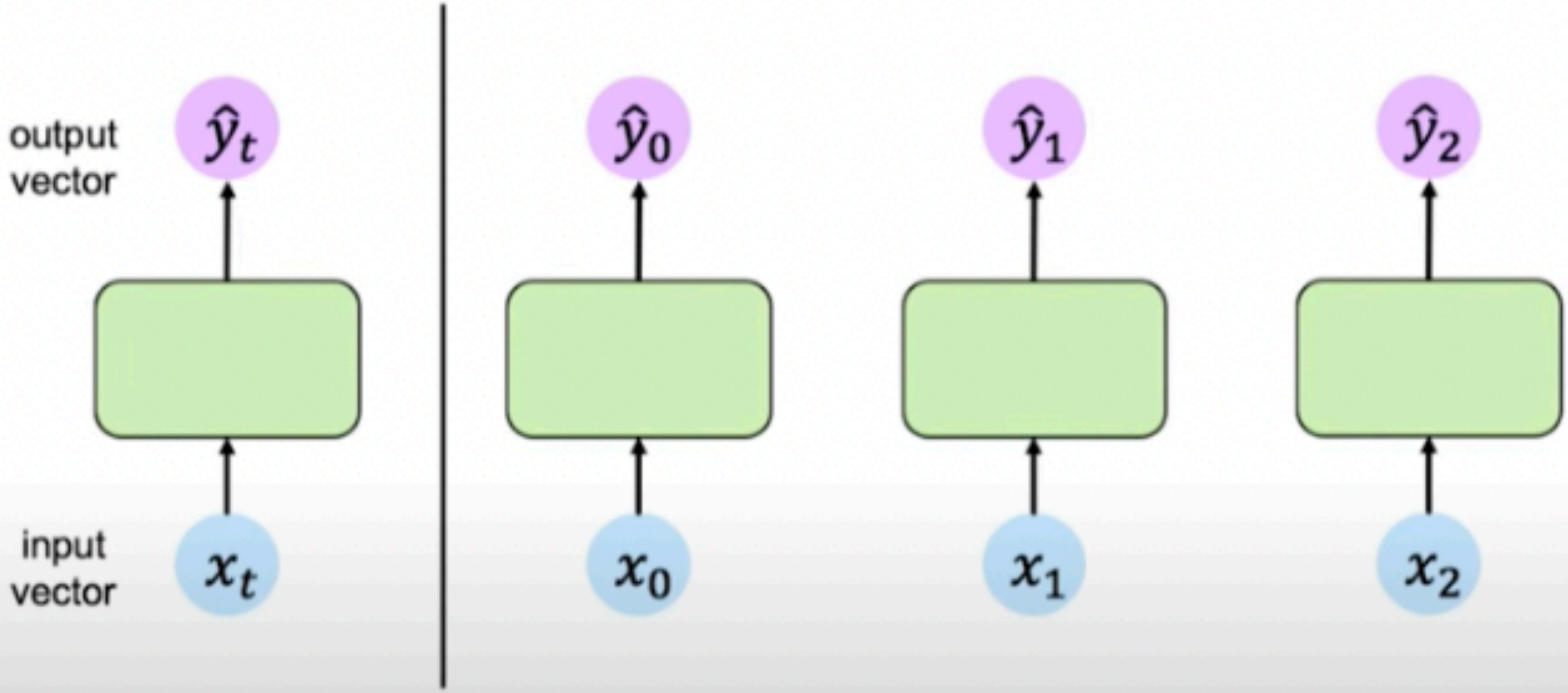
OUR WISH LIST:

1. Handle variable-length sequences
2. Track long term dependencies
3. Maintain information about order
4. Share parameters across the sequence

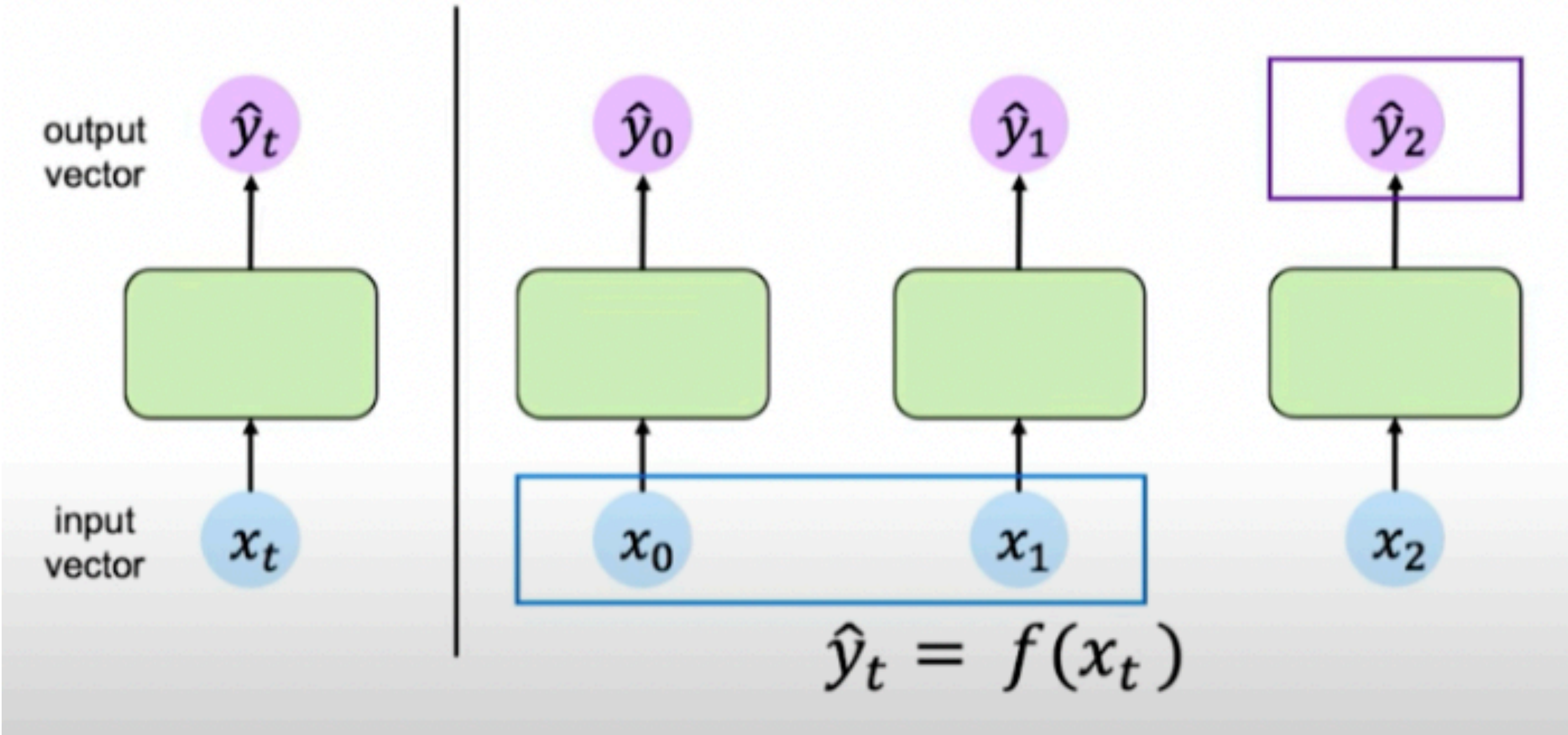
Recurrent Neural Networks
offer a first solution to this problem



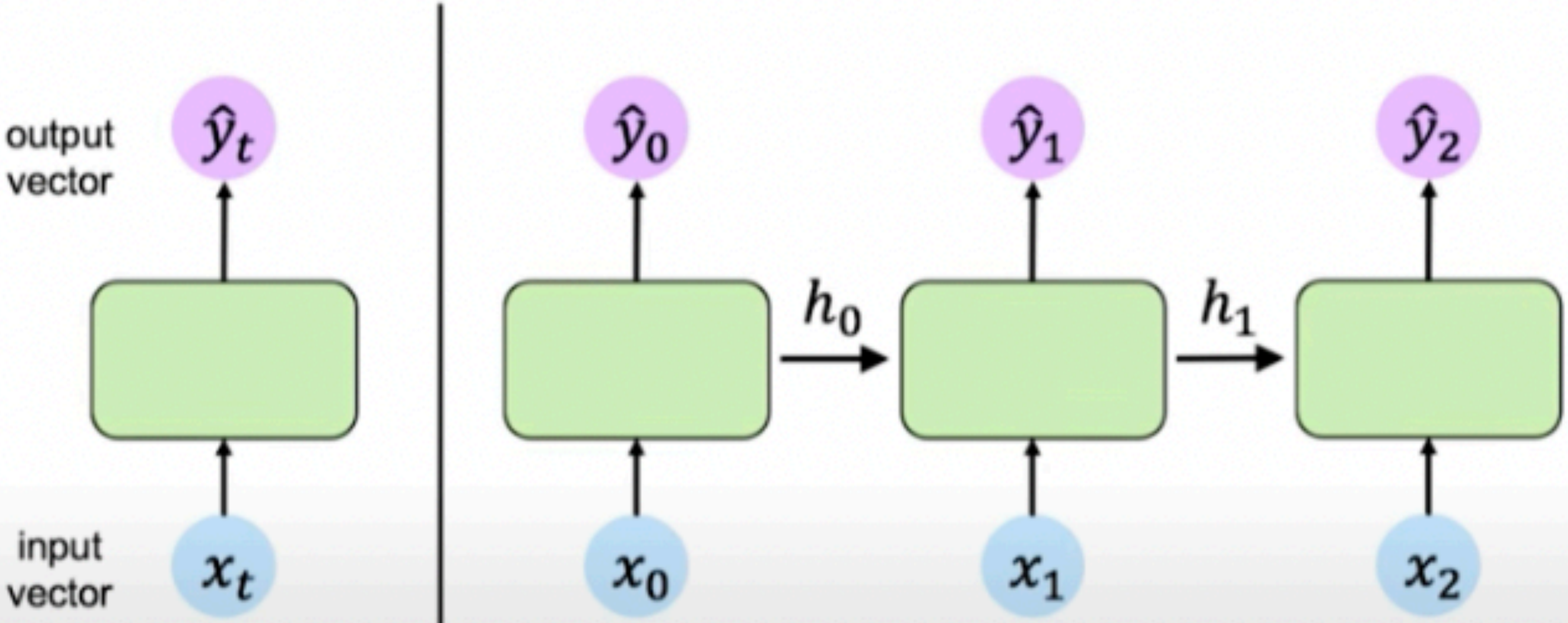
Handling Individual Time Steps



Handling Individual Time Steps

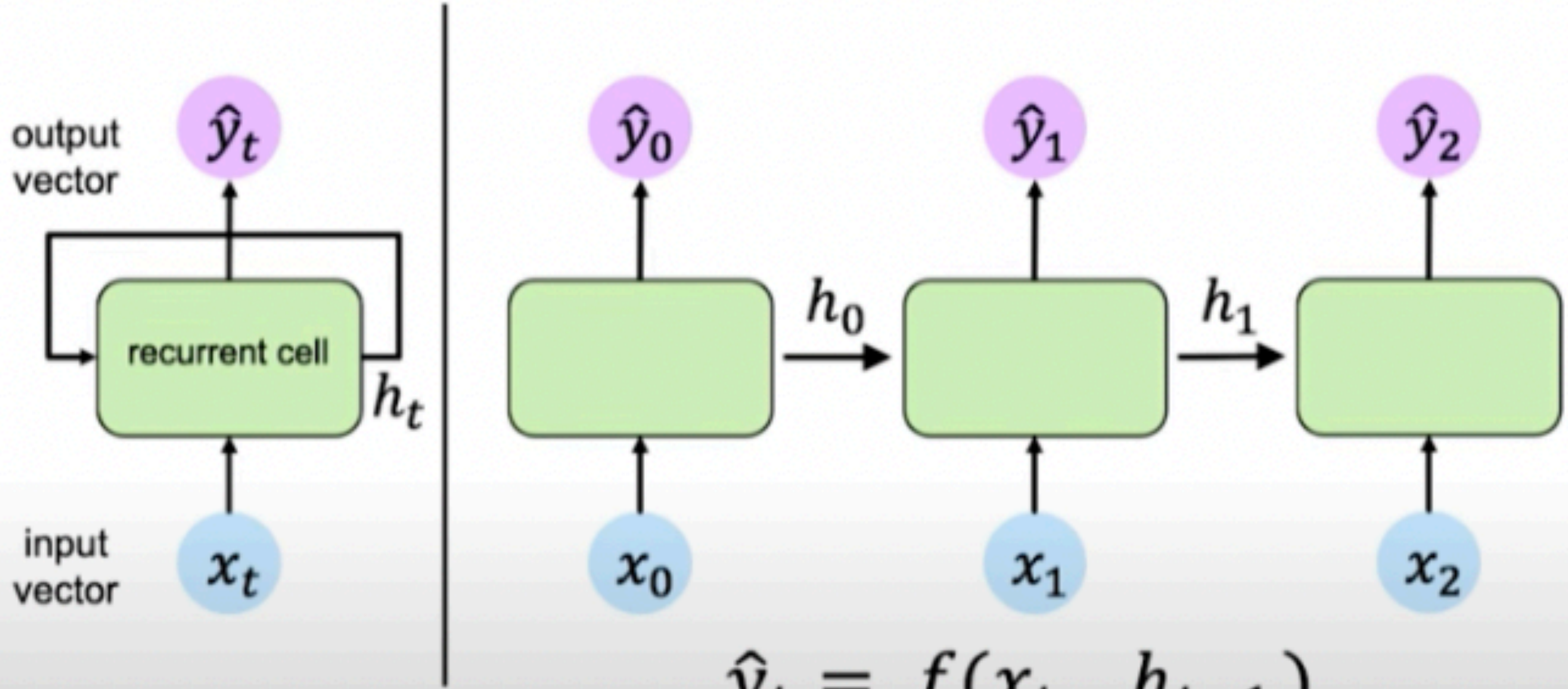


Neurons with Recurrence



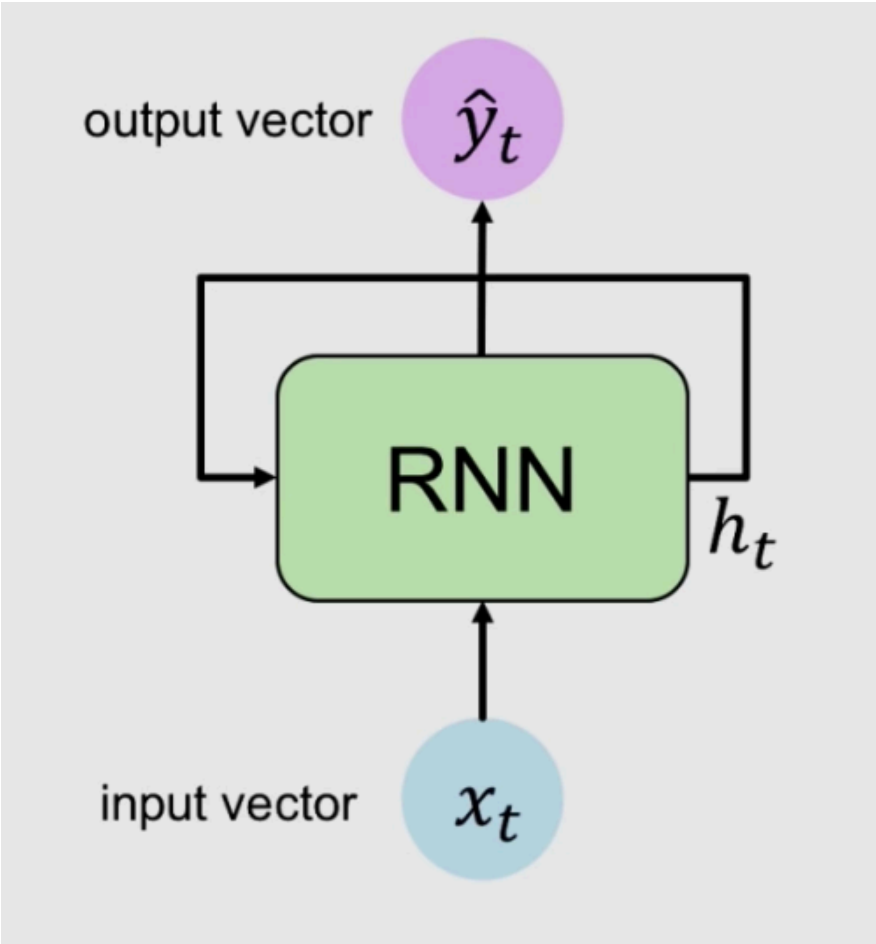
$$\hat{y}_t = f(x_t,\ h_{t-1})$$

Neurons with Recurrence



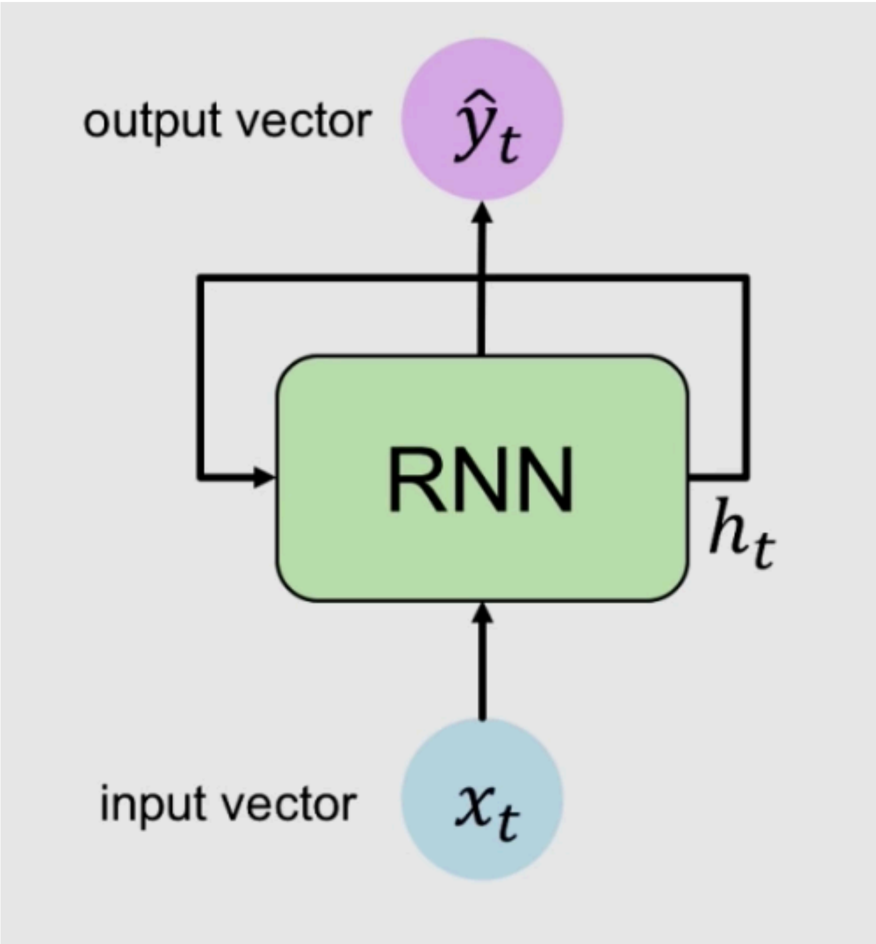
$$\underbrace{\hat{y}_t}_{\text{output}} = f(\underbrace{x_t}_{\text{input}},\ \underbrace{h_{t-1}}_{\text{past memory}})$$

The RNN Block



$$\underbrace{\boxed{h_t}}_{\text{cell state}} = \underbrace{\boxed{f_W}}_{\text{function parameterized by W}}(\underbrace{\boxed{h_{t-1}}}_{\text{old state}}, \underbrace{\boxed{x_t}}_{\text{input vector at time step } t})$$

The RNN Block



Output Vector

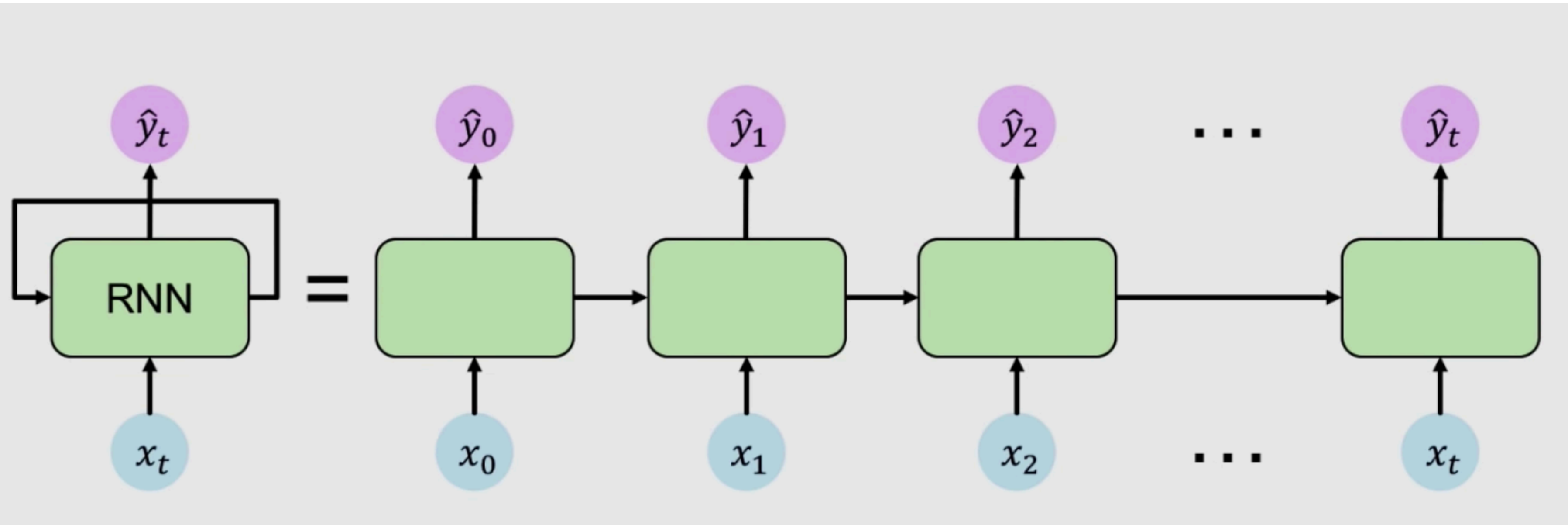
$$\hat{y}_t = \boldsymbol{W}_{\boldsymbol{hy}}^{\boldsymbol{T}} h_t$$

Update Hidden State

$$h_t = \tanh(\boldsymbol{W}_{\boldsymbol{hh}}^{\boldsymbol{T}} h_{t-1} + \boldsymbol{W}_{\boldsymbol{xh}}^{\boldsymbol{T}} x_t)$$

Input Vector

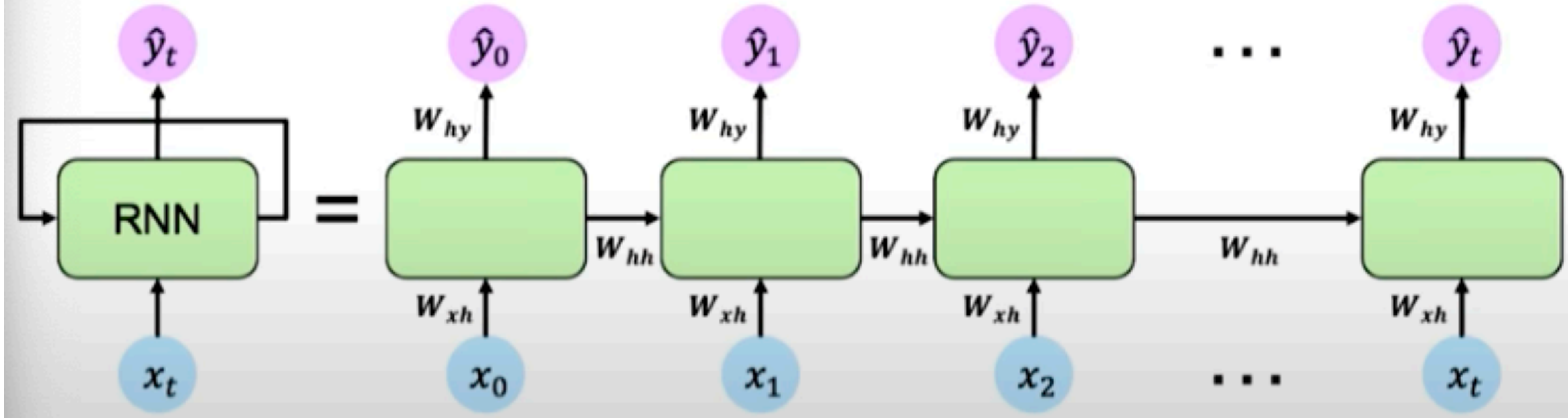
$$x_t$$

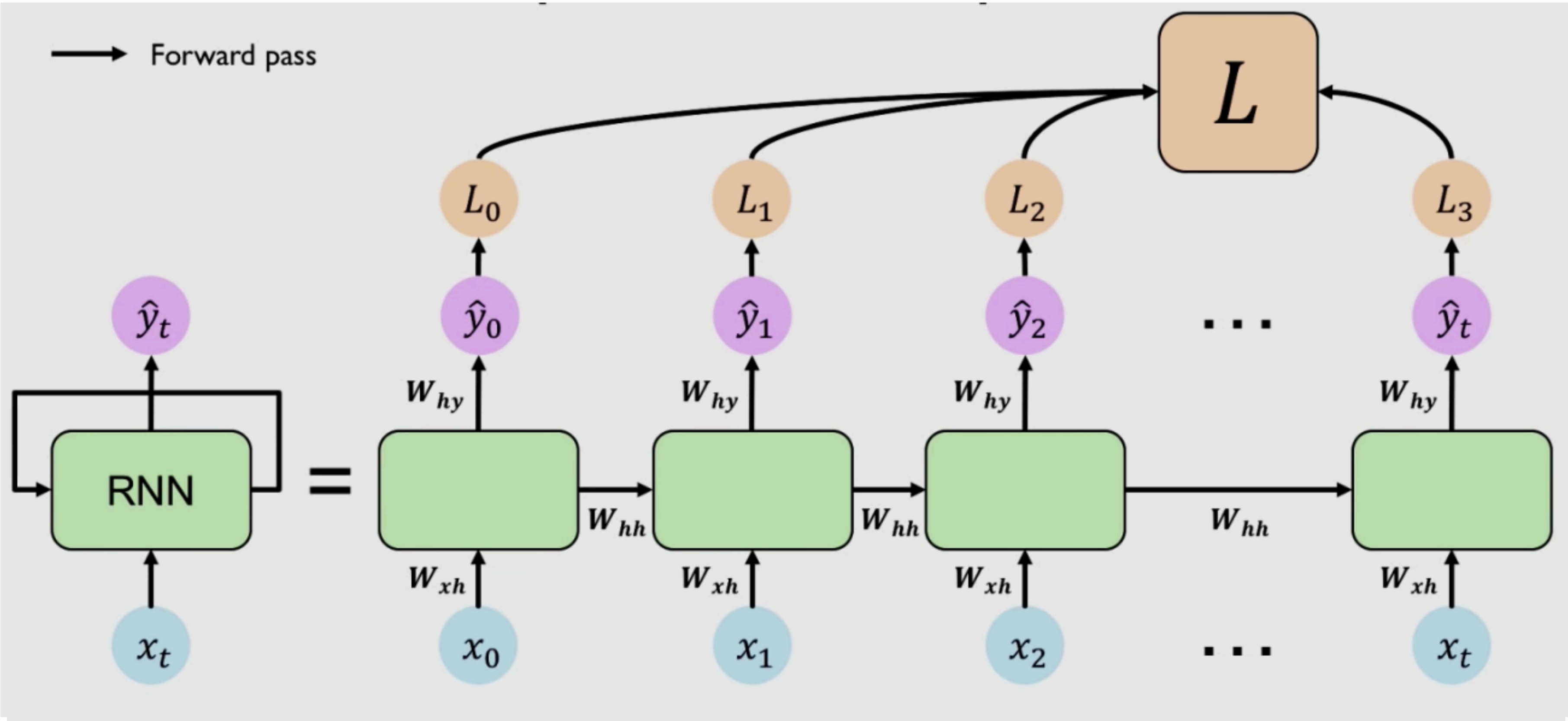


$\hat{y}_t$
RNN
x_t
=
$\hat{y}_0$
$\hat{y}_1$
$\hat{y}_2$
...
$\hat{y}_t$
x_0
x_1
x_2
...
x_t

RNNs: Computational Graph Across Time

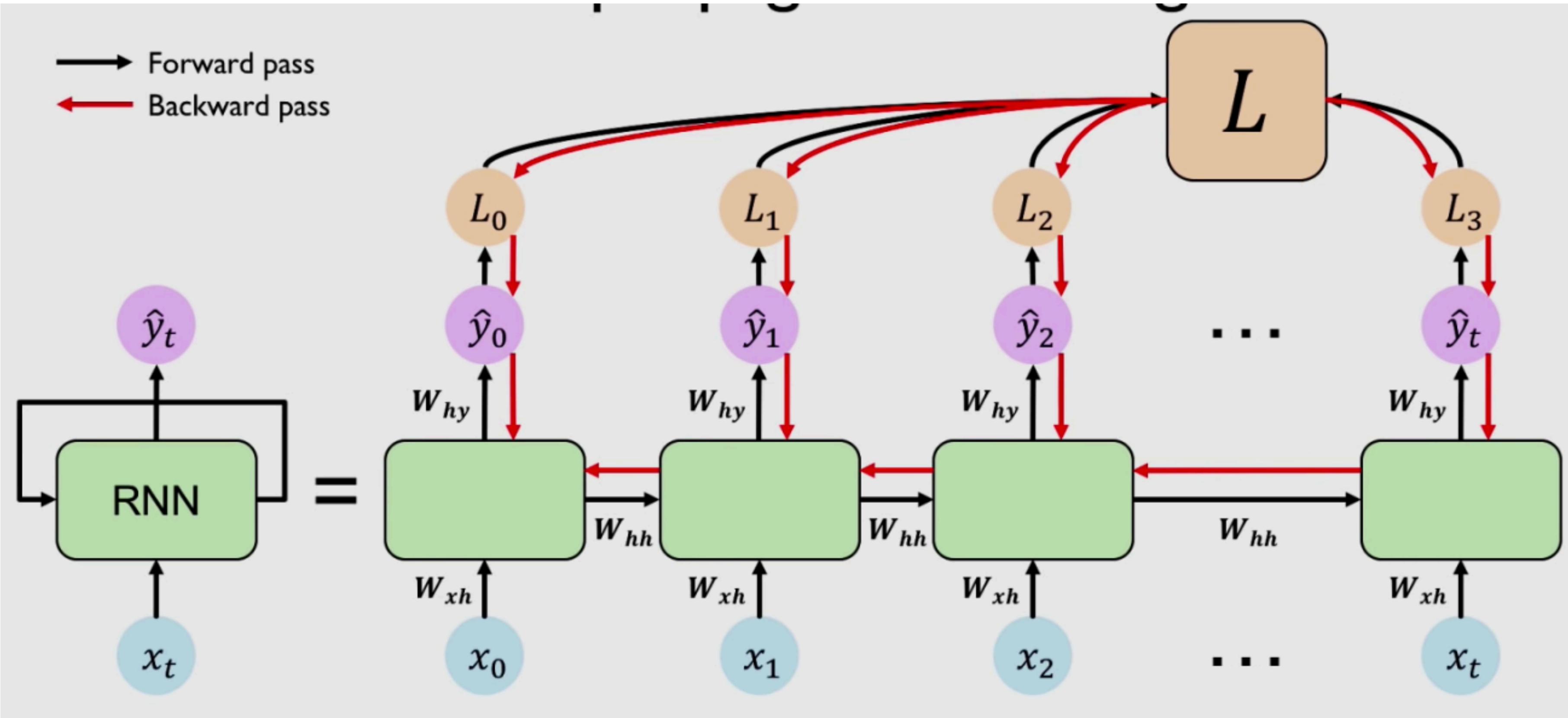
Re-use the **same weight matrices** at every time step

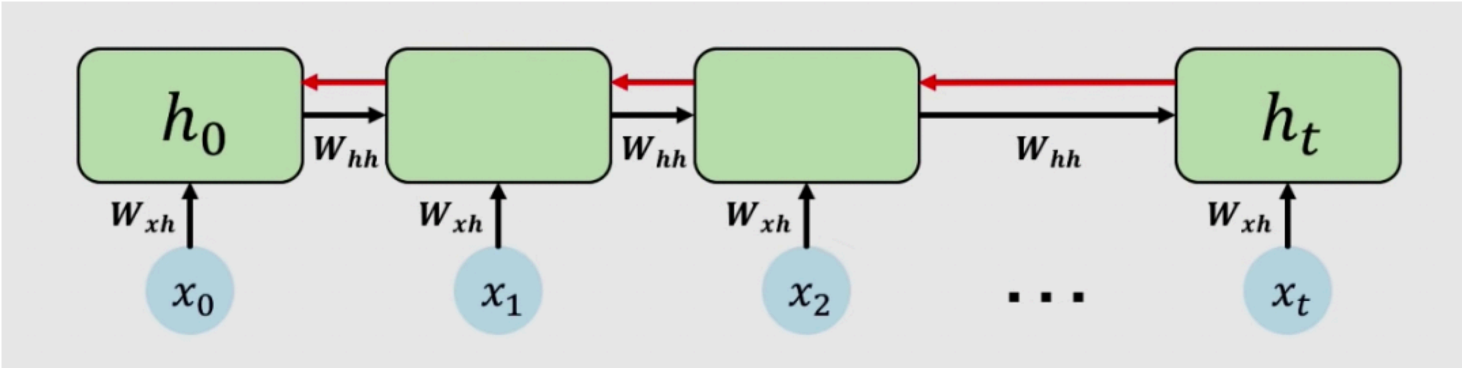




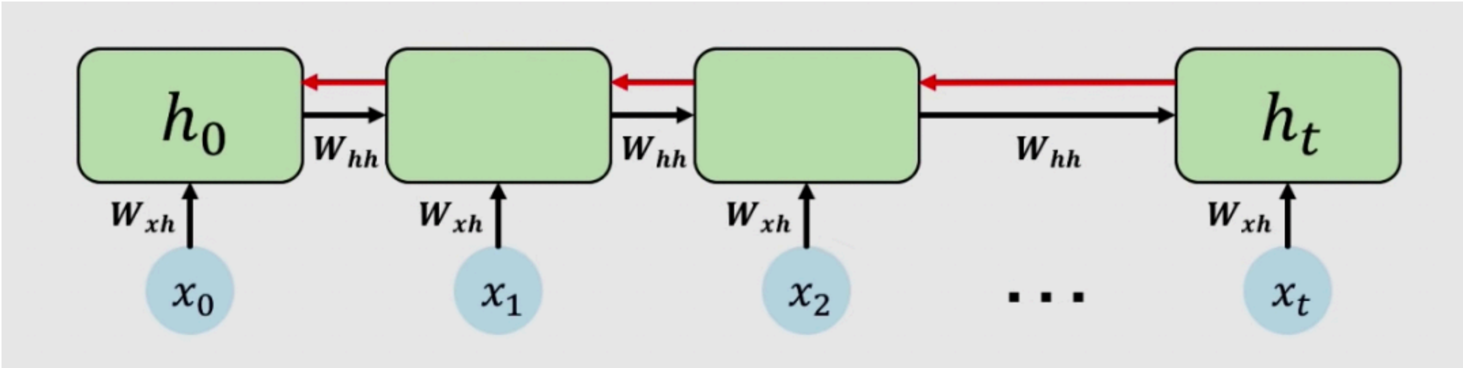
Forward pass
L
L_0
L_1
L_2
L_3
$\hat{y}_t$
$\hat{y}_0$
$\hat{y}_1$
$\hat{y}_2$
...
$\hat{y}_t$
W_{hy}
RNN
=
W_{hh}
W_{xh}
x_t
x_0
x_1
x_2
...
x_t

BACKPROPAGATION THROUGH TIME (BPTT)





BPTT implies a large amount of weight multiplications: if we want to take into account long term memory, networks become quickly very deep and so are subjected to vanishing and exploding gradients problems (see previous lecture)

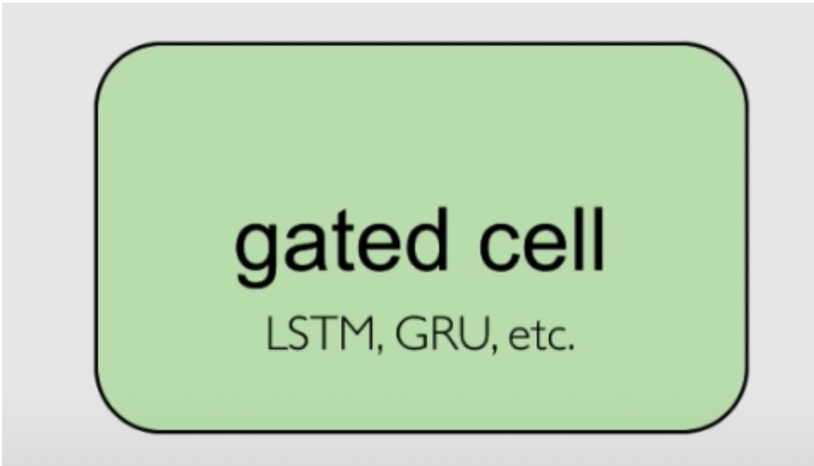


BPTT implies a large amount of weight multiplications: if we want to take into account long term memory, networks become quickly very deep and so are subjected to vanishing and exploding gradients problems (see previous lecture)

ONE CAN USE THE SAME TRICKS THAT WE DISCUSSED FOR DEEP CNNs (WEIGTH INITIALISATION, RELU ACTIVATION FUNCTION ...)

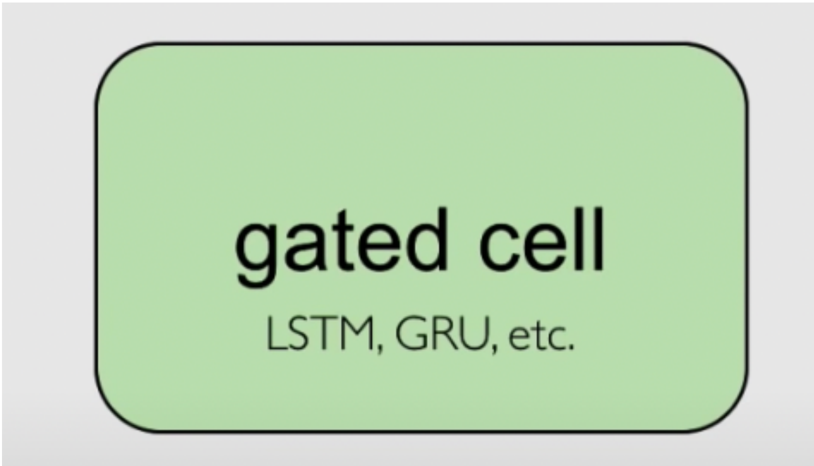
ANOTHER SPECIFIC SOLUTION: GATED CELLS

Use a more complex recurrent unit with gates to control what information is passed through…



ANOTHER SPECIFIC SOLUTION: GATED CELLS

Use a more complex recurrent unit with gates to control what information is passed through…



Long Short Term Memory (LSTMs) networks are an example of this