Deep Learning

Angshuman Paul

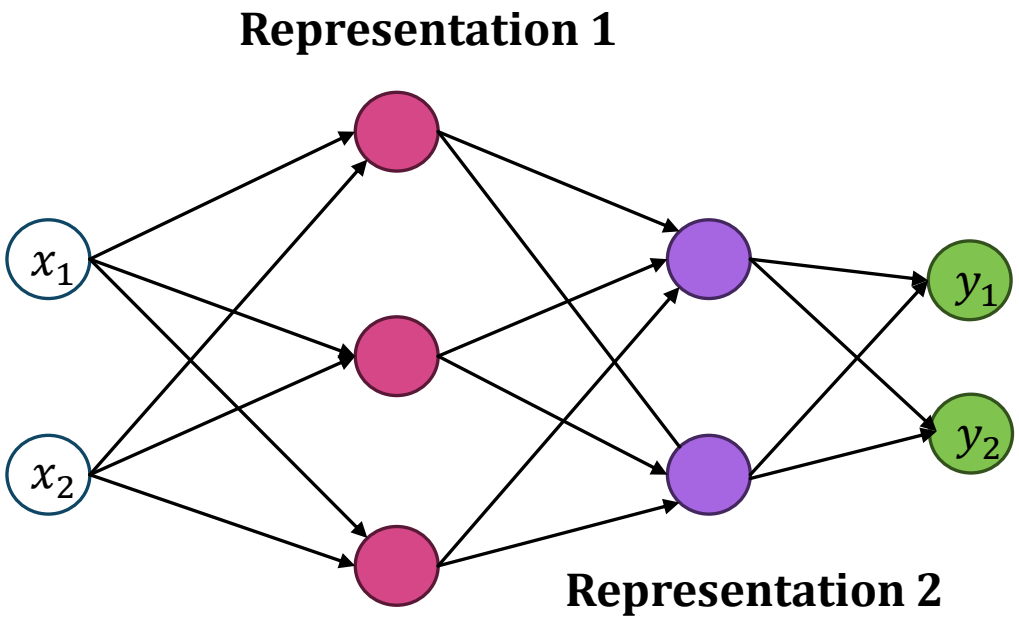
Assistant Professor

Department of Computer Science & Engineering

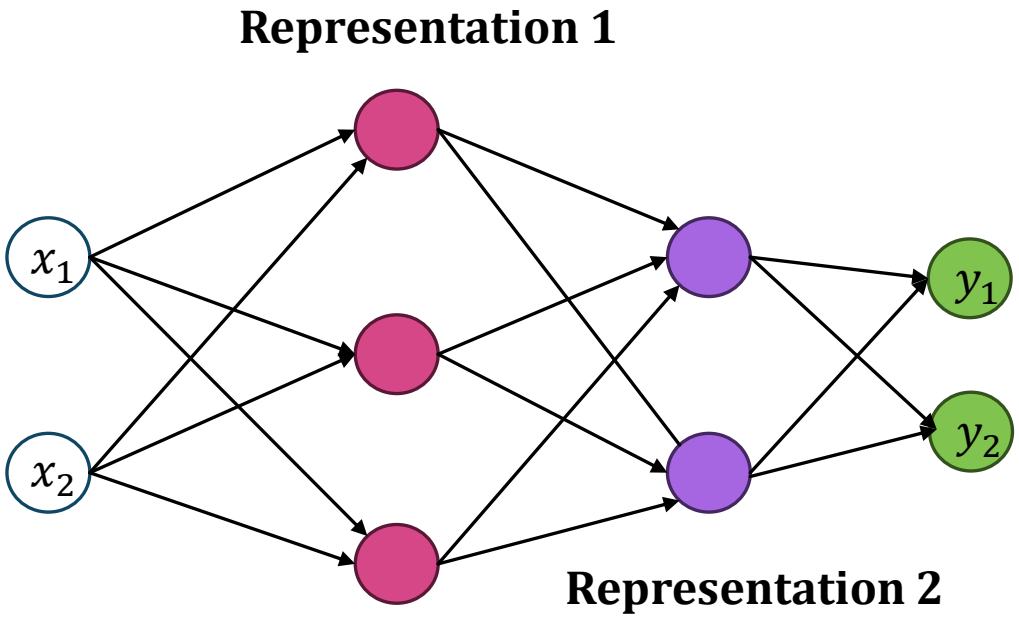
Representation Learning

- Handcrafted features are useful in ML tasks if correct features are extracted
- It's often difficult to know what are the correct features (especially from images, audios, video, etc.)
- Various NNs (such as CNNs, RNNs) not only map the input data (images, videos, audios) to the output label/ value but also learns a representation of the data in the various layers
  - **Representation learning**

Representation Learning

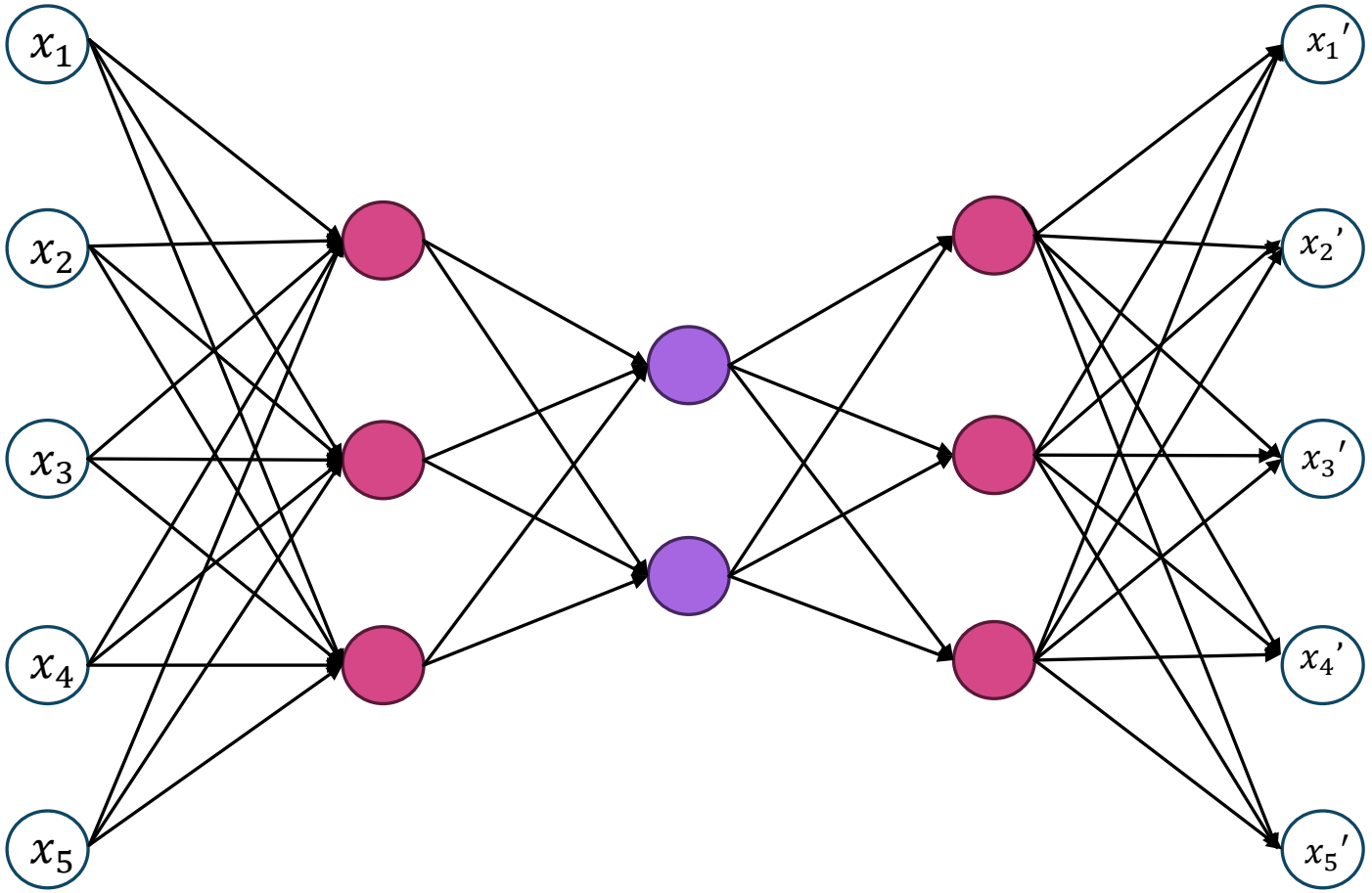


Representation Learning

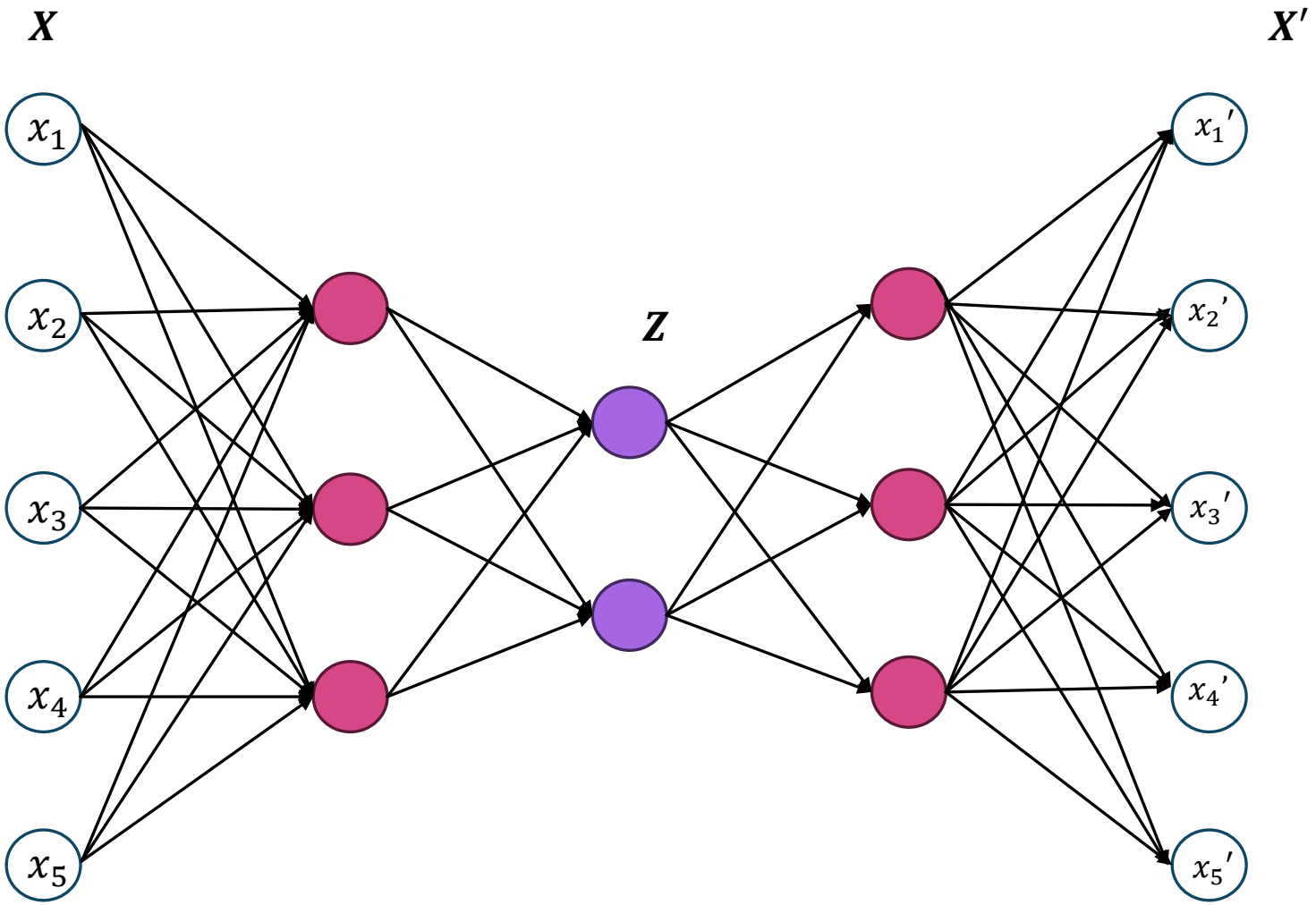


Initial layers learn low level explainable features (such as edge, texture, etc.), deeper layer learn high-level abstract features (by combining suitable low-level features)

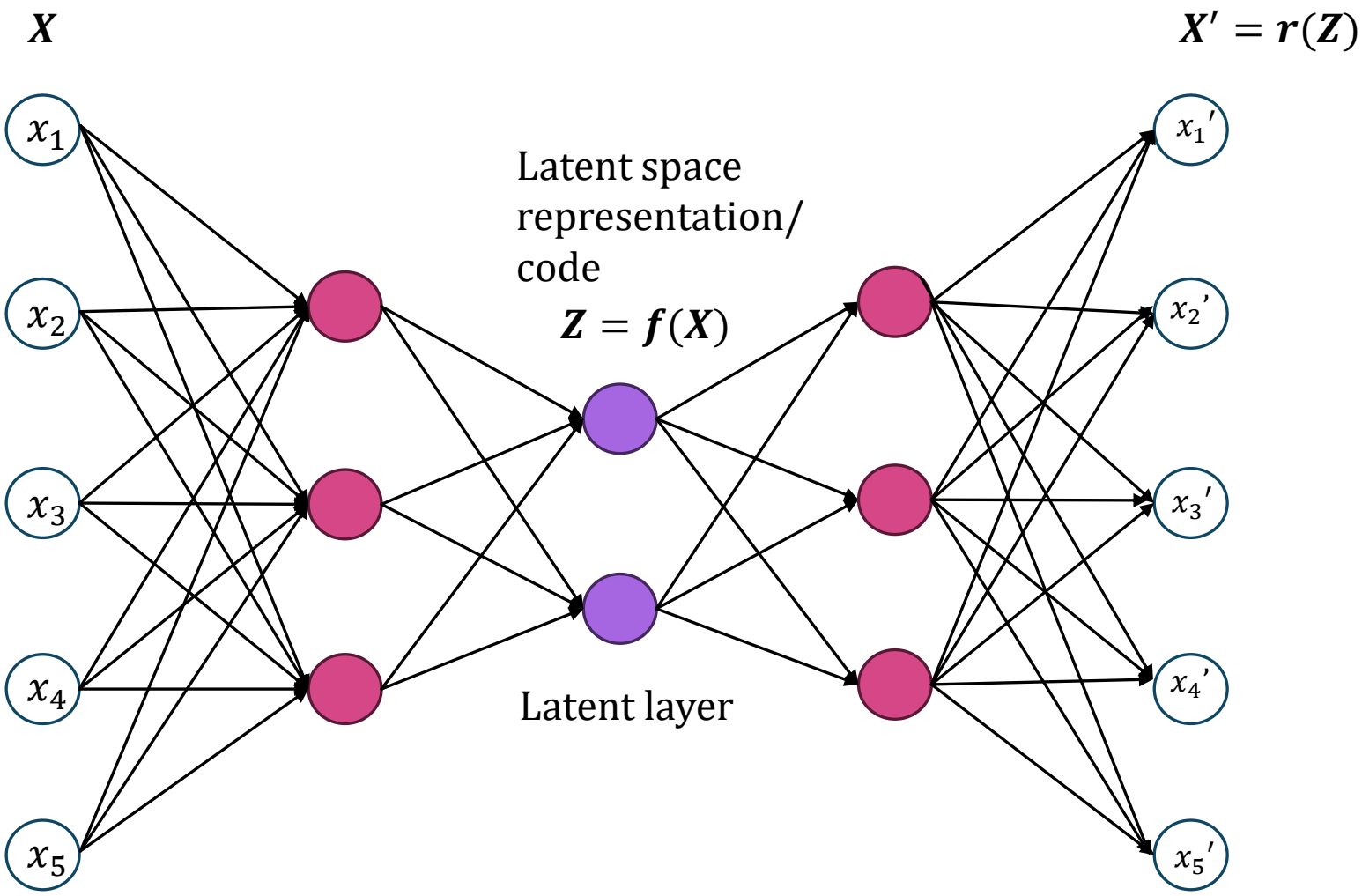
Autoencoders



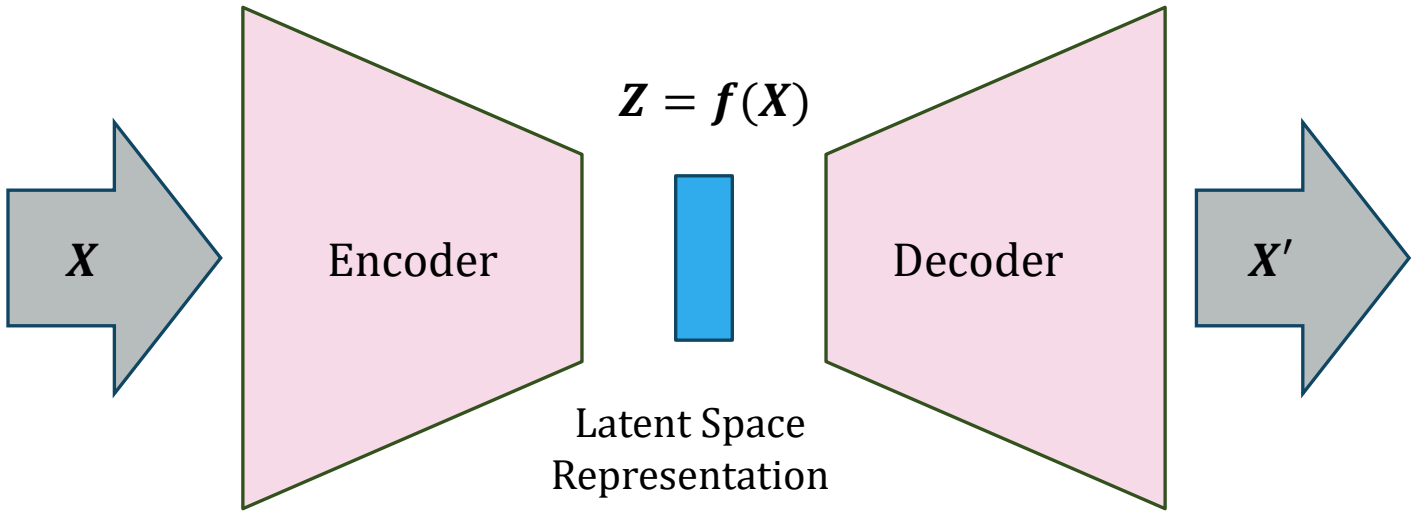
Autoencoders



Autoencoders



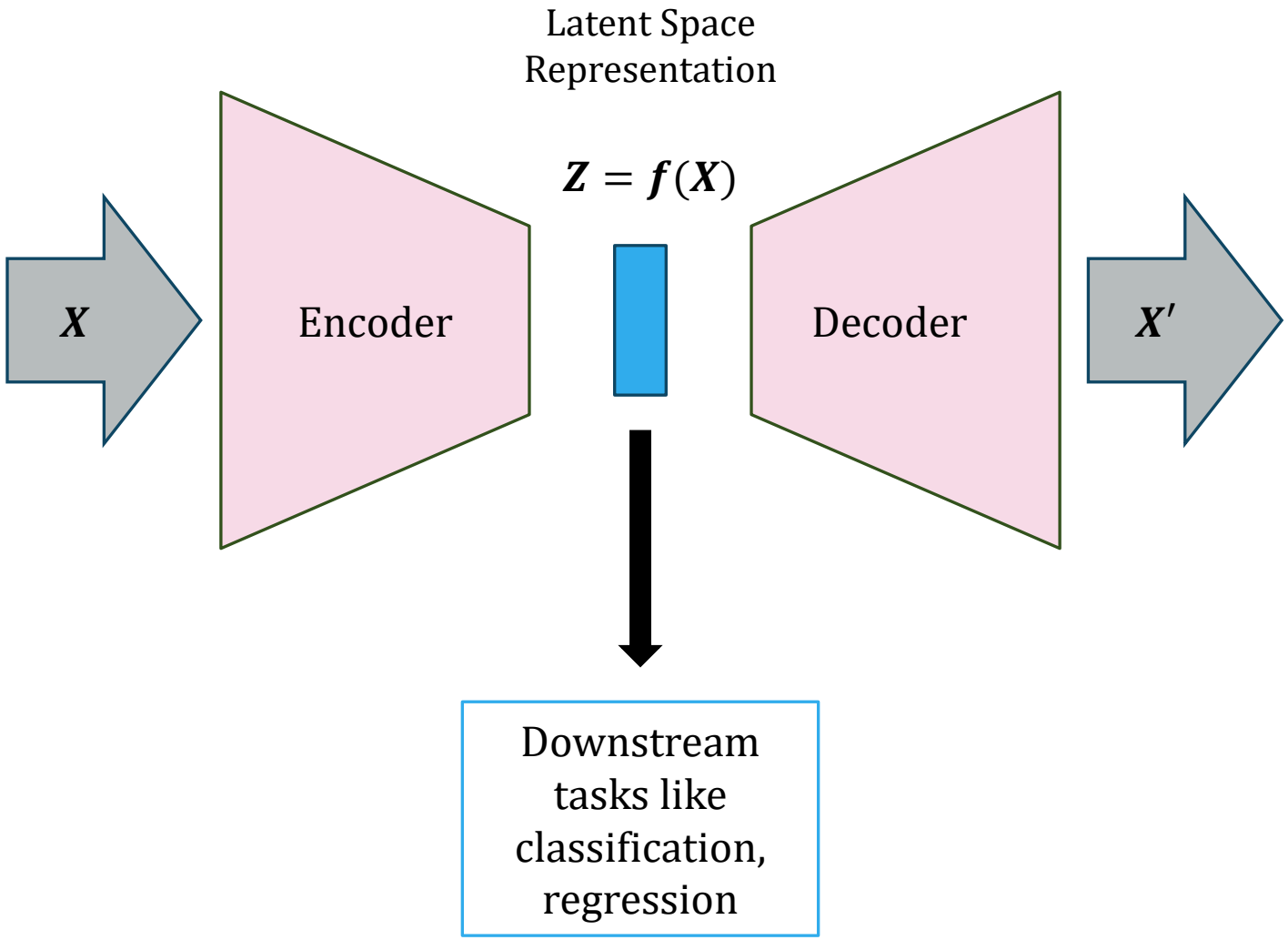
Autoencoders



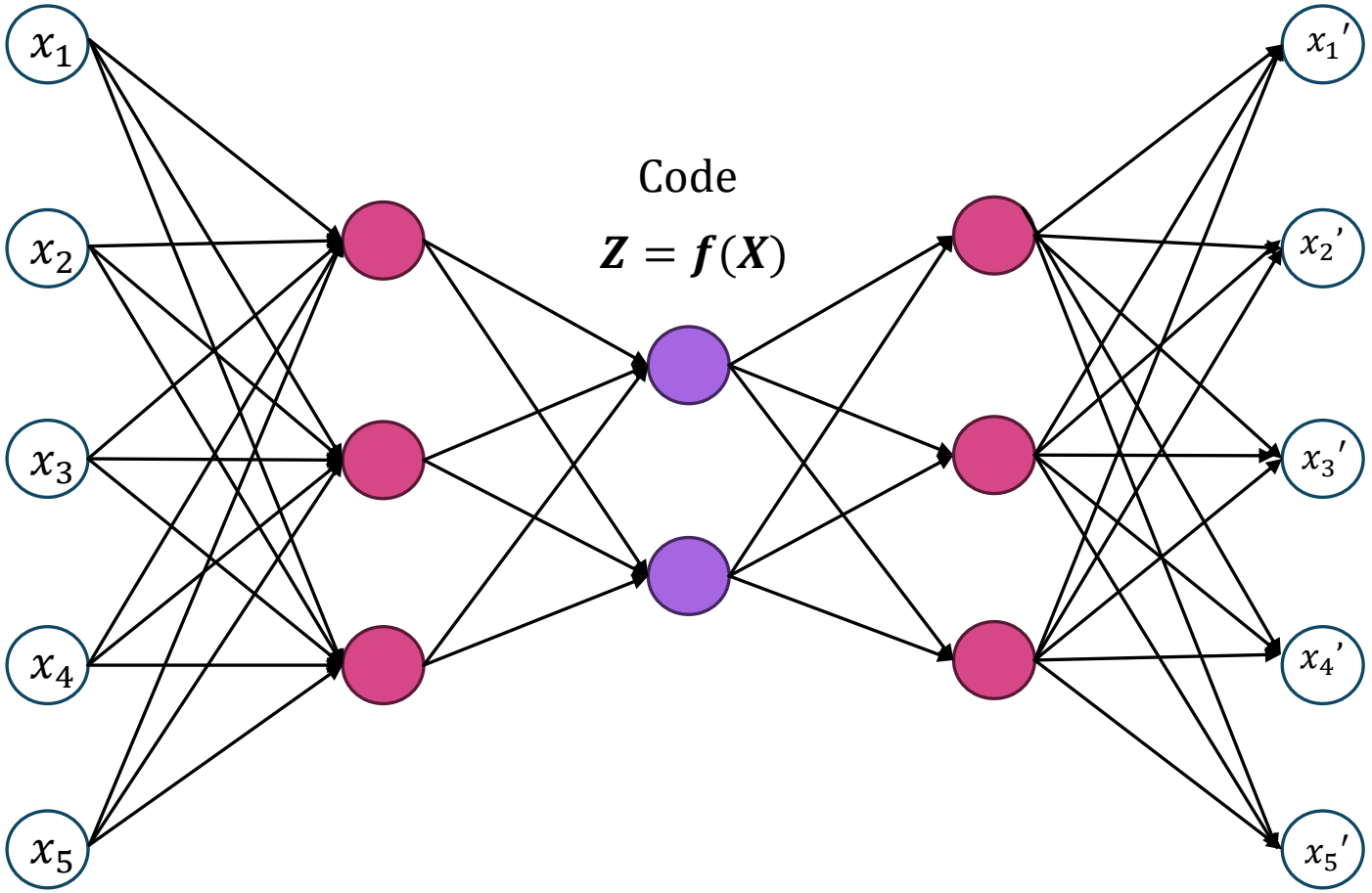
The encoder creates the code Z from X

The decoder reconstructs X from Z. So ideally, $X' = X$

Autoencoders

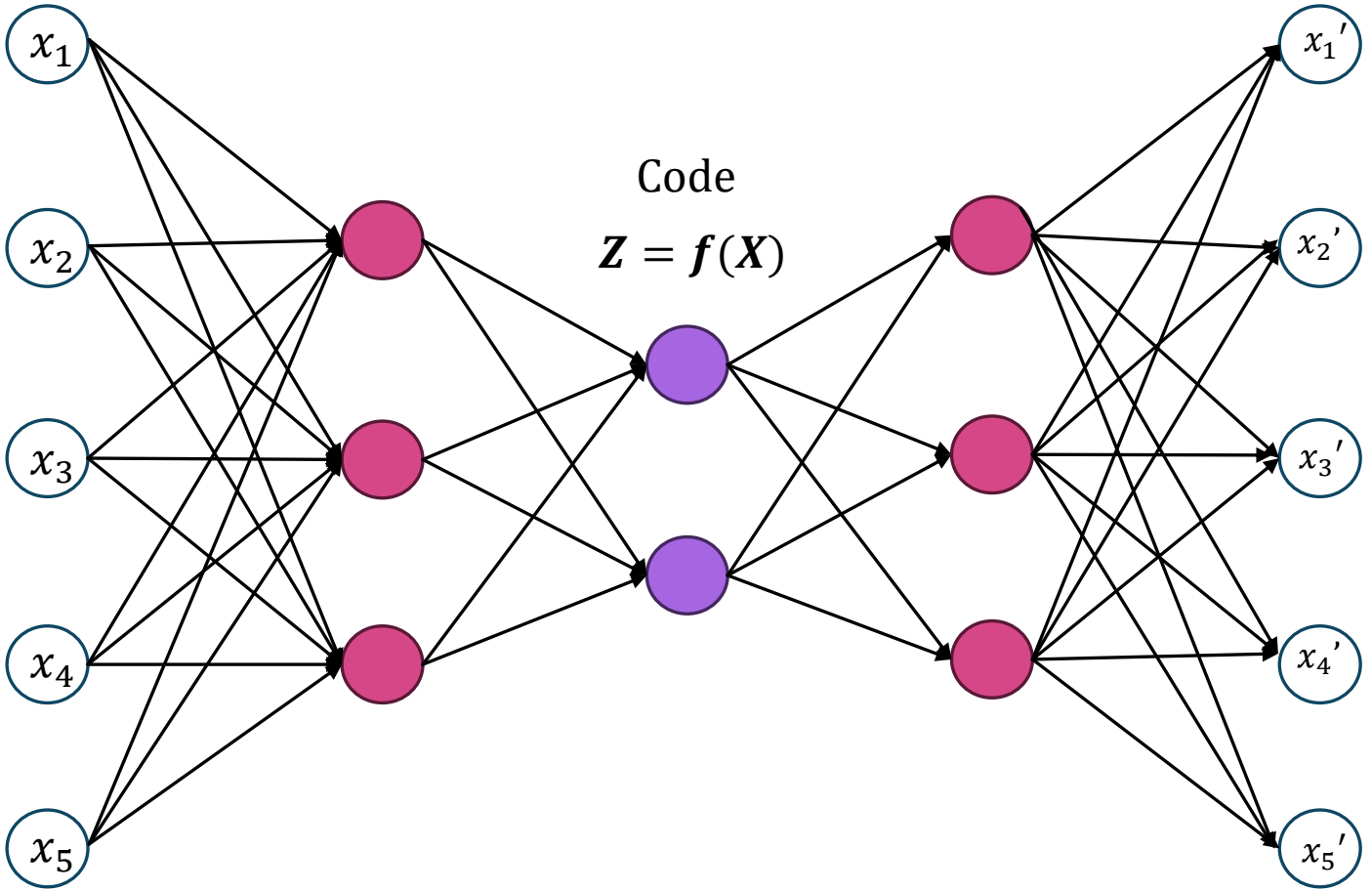


Undercomplete Autoencoders



Dimension of latent space < dimension of input/ output space

Undercomplete Autoencoders

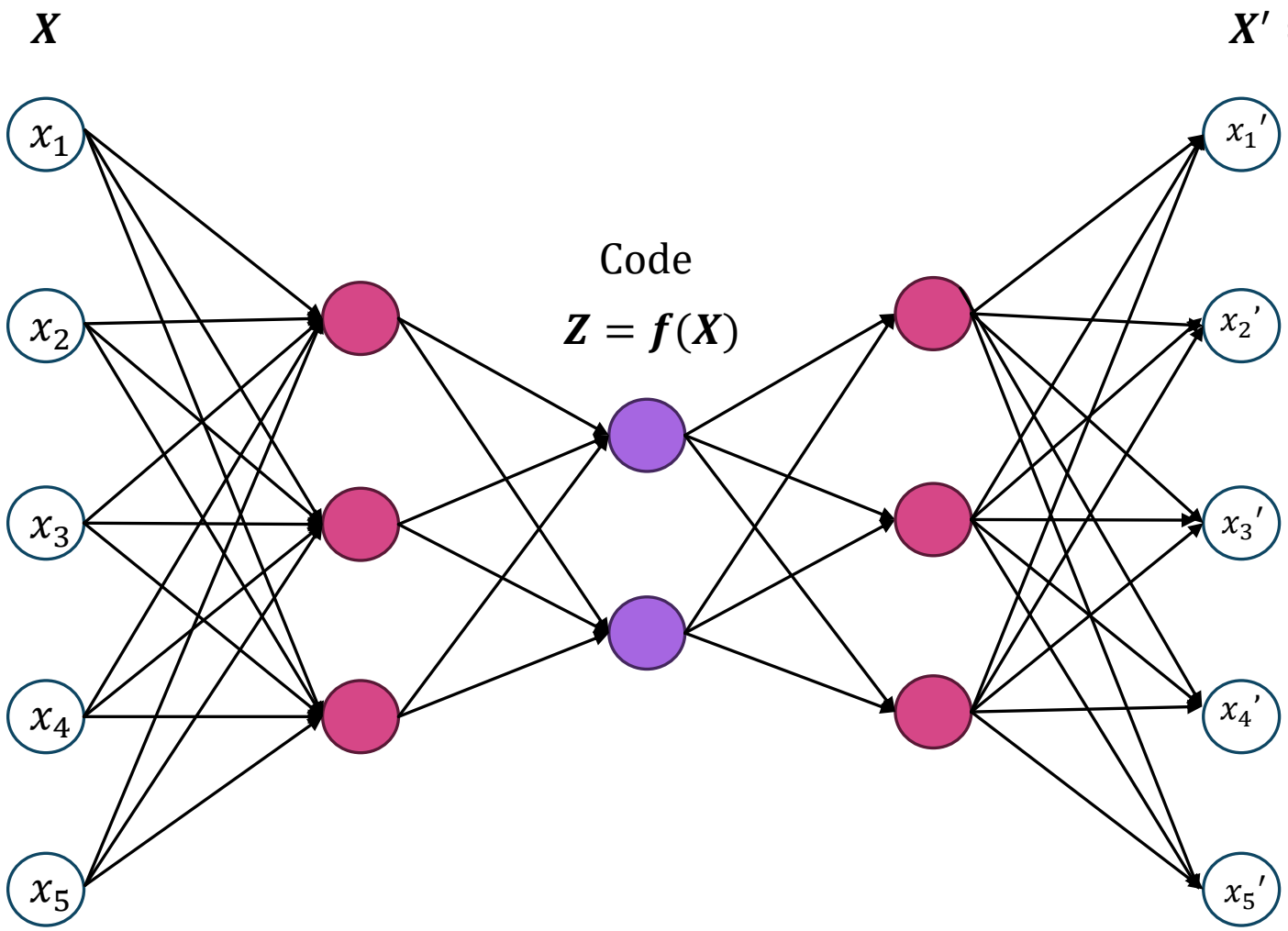


May be used for dimensionality reduction/ compression of data

Extraction of salient features

Dimension of latent space < dimension of input/ output space

Undercomplete Autoencoders

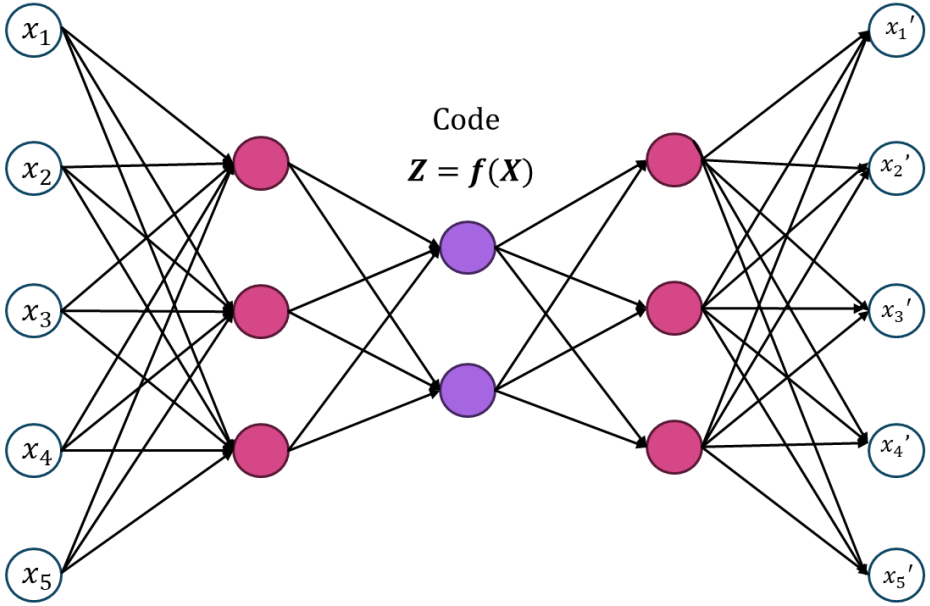


Loss $= MSE(X, X')$

Minimization of this loss requires X' to be as similar as possible to X

In order to do that, Z must contain salient features

Extraction of Salient Features



Dimension of latent space < dimension of input/ output space

x_1: Weight

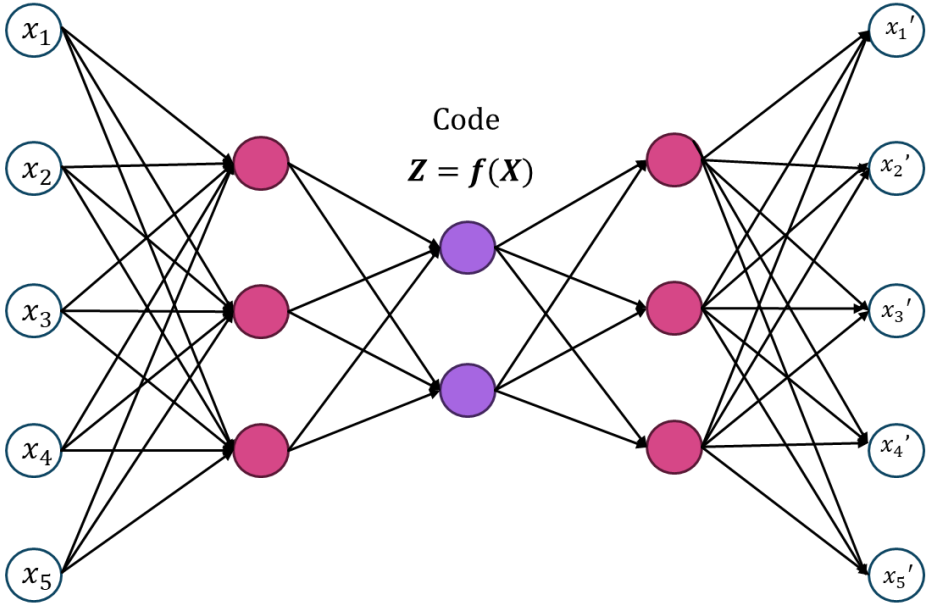
x_2: Height

x_3: Major color 1

x_4: Major color 2

x_5: Repetition of color pattern (yes/no)

Extraction of Salient Features



Dimension of latent space < dimension of input/ output space

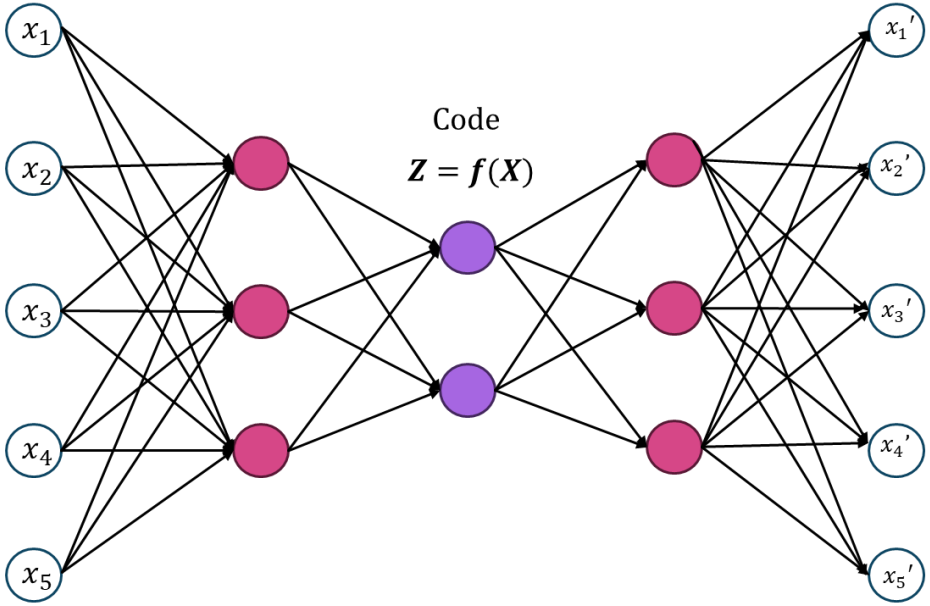
Can x_1 through x_5 indivisually help in distinguishing a dog and a zebra?

x_1: Weight
x_2: Height
x_3: Major color 1
x_4: Major color 2
x_5: Repetition of color pattern (yes/no)

https://www.livescience.com/27443-zebras.html

https://www.pexels.com/photo/a-black-and-white-dog-with-a-harness-on-a-path-in-the-forest-16113021/

Extraction of Salient Features



Dimension of latent space < dimension of input/ output space

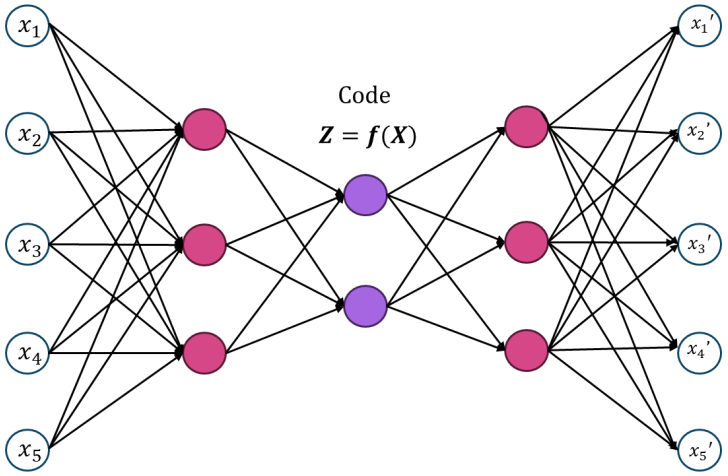
x_1: Weight
x_2: Height
x_3: Major color 1
x_4: Major color 2
x_5: Repetition of color pattern (yes/no)

Can x_1 through x_5 indivisually help in distinguishing a dog and a zebra?

May not be. For example, x_1 and x_2 may provide redundant information.

A proper combination of x_3 (white), x_4 (black) and x_5 may provide a meaningful discriminative information

Extraction of Salient Features



Dimension of latent space < dimension of input/ output space

x_1: Weight
x_2: Height
x_3: Major color 1
x_4: Major color 2
x_5: Repetition of color pattern (yes/no)

Can x_1 through x_5 indivisually help in distinguishing a dog and a zebra?

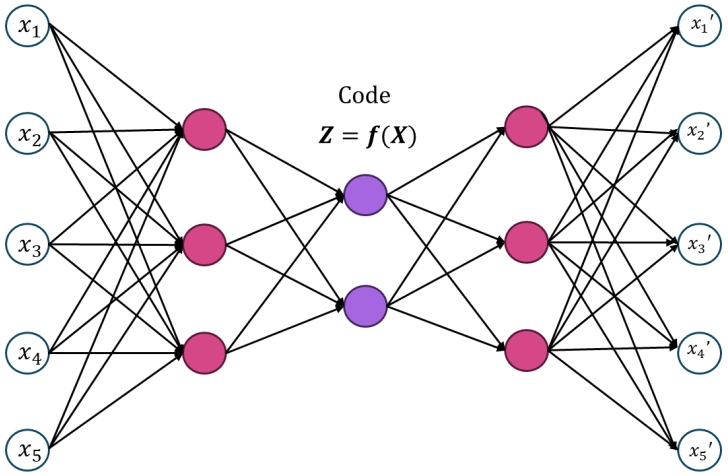
May not be. For example, x_1 and x_2 may provide redundant information.

A proper blending of features x_3 (white), x_4 (black) and x_5 may provide a meaningful discriminative information

This blending of features is created by AE in the latent space

Thus the latent space creates a salient representation of data

Extraction of Salient Features



Dimension of latent space < dimension of input/ output space

x_1: Weight
x_2: Height
x_3: Major color 1
x_4: Major color 2
x_5: Repetition of color pattern (yes/no)

Can x_1 through x_5 indivisually help in distinguishing a dog and a zebra?

May not be. For example, x_1 and x_2 may provide redundant information.

A proper blending of features x_3 (white), x_4 (black) and x_5 may provide a meaningful discriminative information

This blending of features is created by AE in the latent space

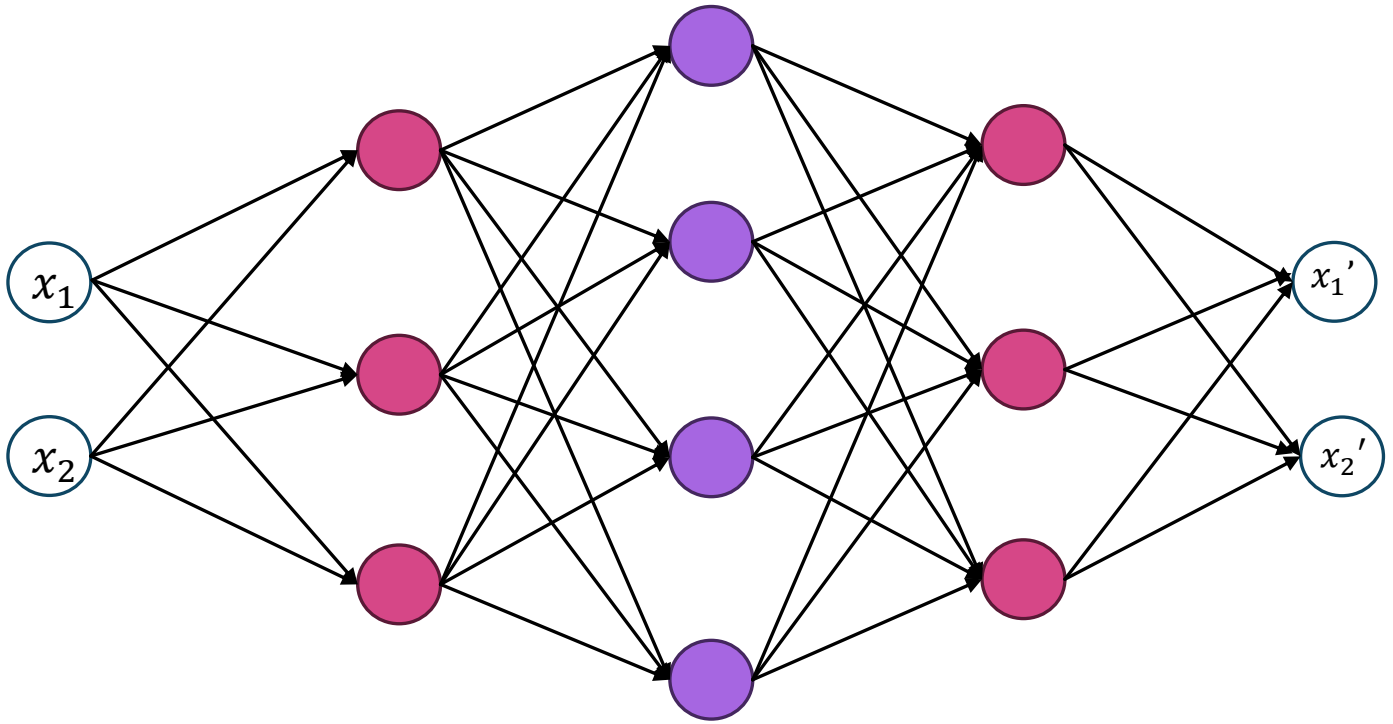
Thus the latent space creates a salient representation of data

If salient features are not created, the correct data (dog, zebra) can't be created

Overcomplete Autoencoders

Code

$$\boldsymbol{Z} = \boldsymbol{f}(\boldsymbol{X})$$

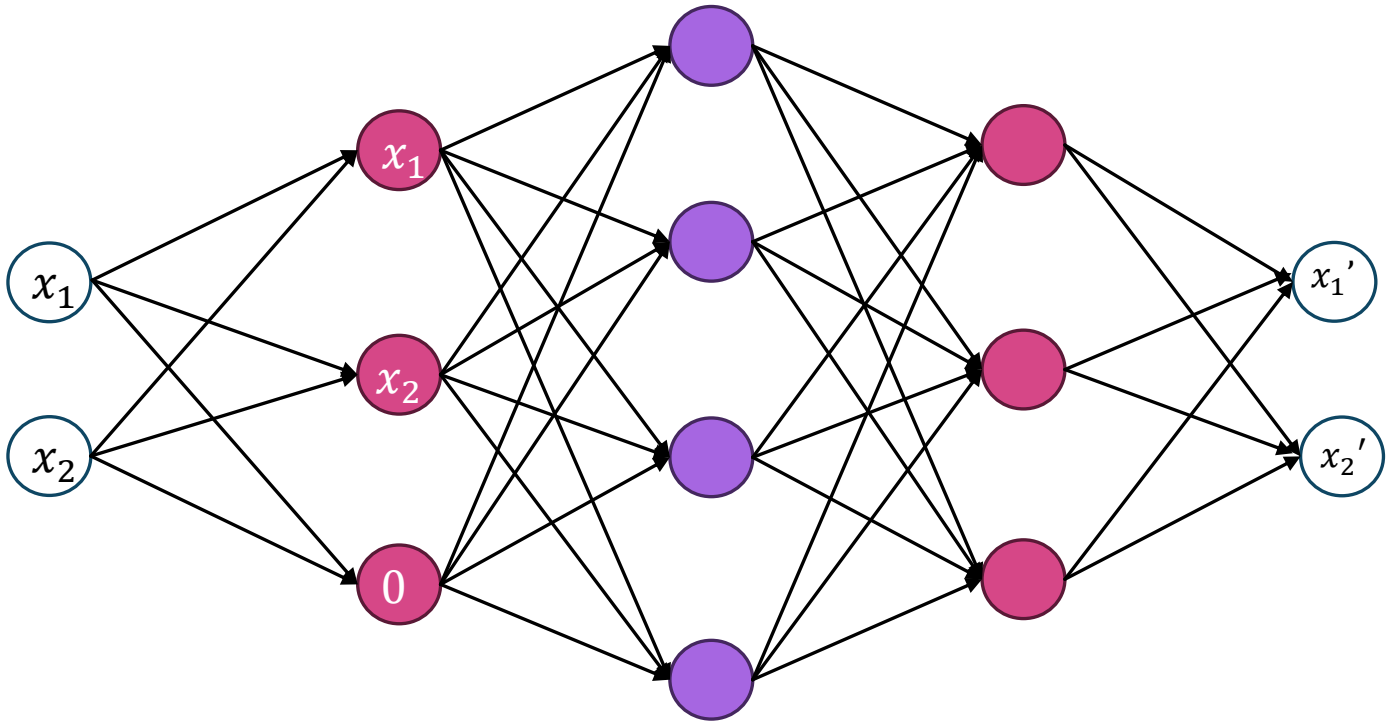


Dimension of latent space > dimension of input/ output space

Overcomplete Autoencoders: Problem

Code

$$\boldsymbol{Z} = \boldsymbol{f}(\boldsymbol{X})$$

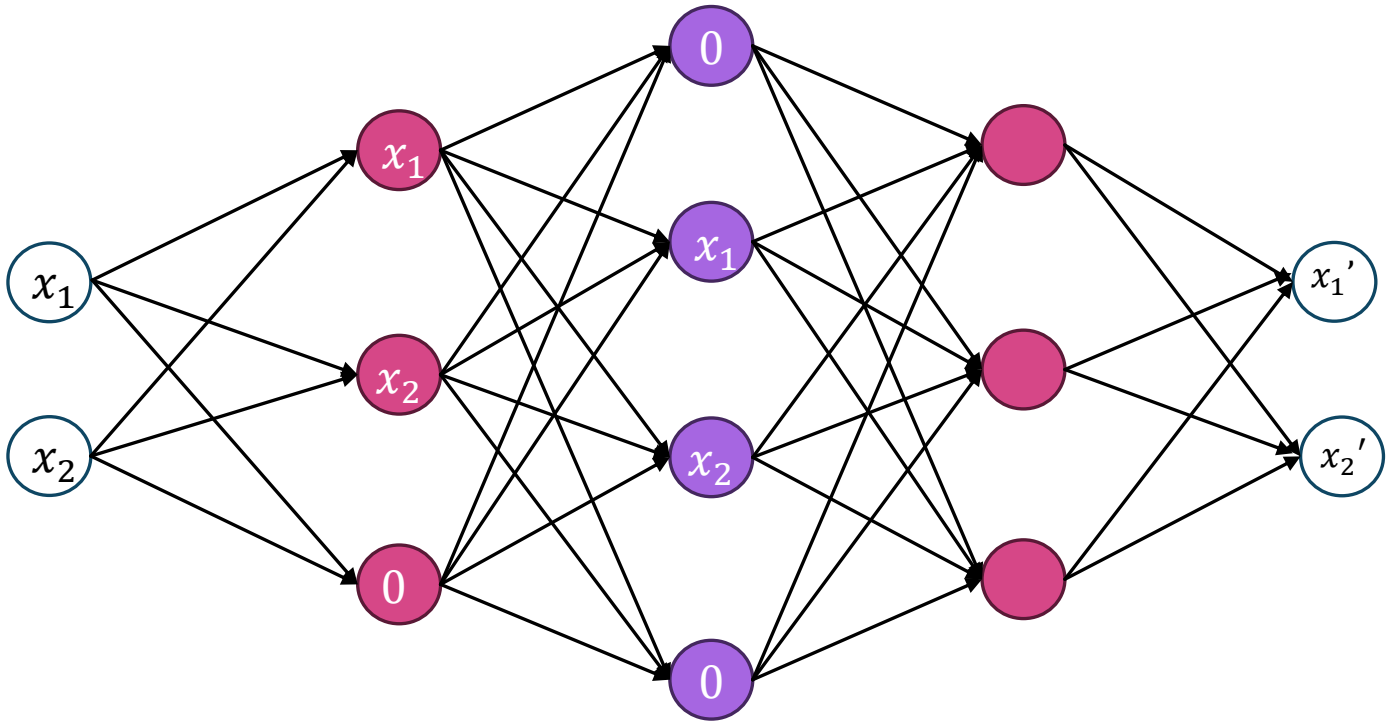


Dimension of latent space > dimension of input/ output space

Overcomplete Autoencoders: Problem

Code

$$\boldsymbol{Z} = \boldsymbol{f}(\boldsymbol{X})$$

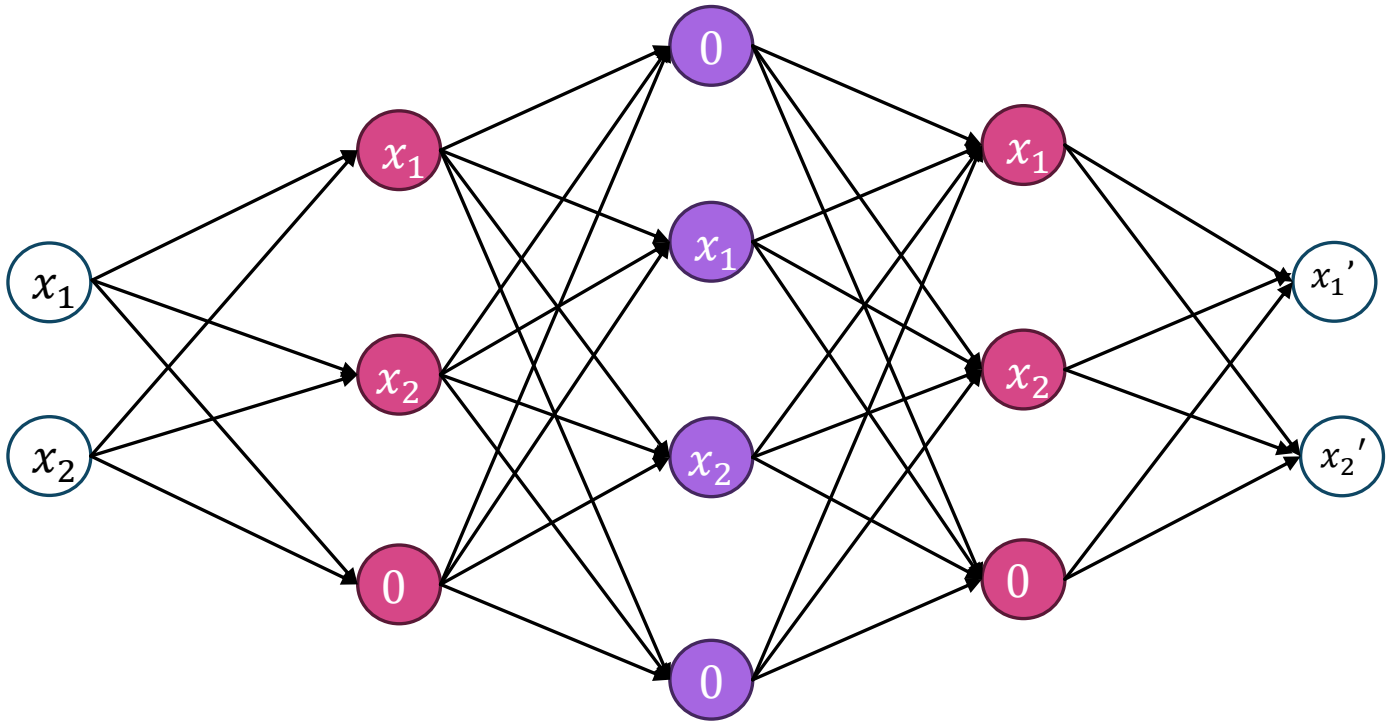


Dimension of latent space > dimension of input/ output space

Overcomplete Autoencoders: Problem

Code

$$\boldsymbol{Z} = \boldsymbol{f}(\boldsymbol{X})$$

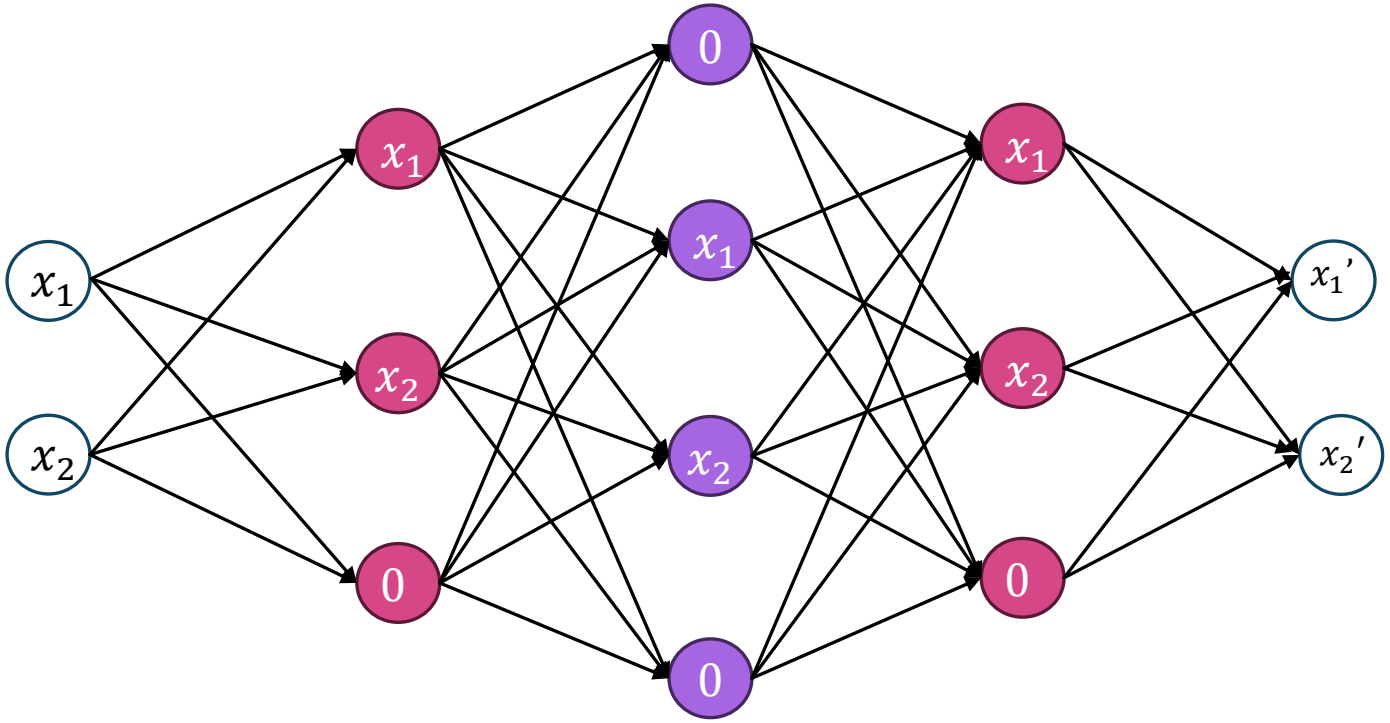


Dimension of latent space > dimension of input/ output space

Overcomplete Autoencoders: Problem

Code

$$\boldsymbol{Z} = \boldsymbol{f}(\boldsymbol{X})$$



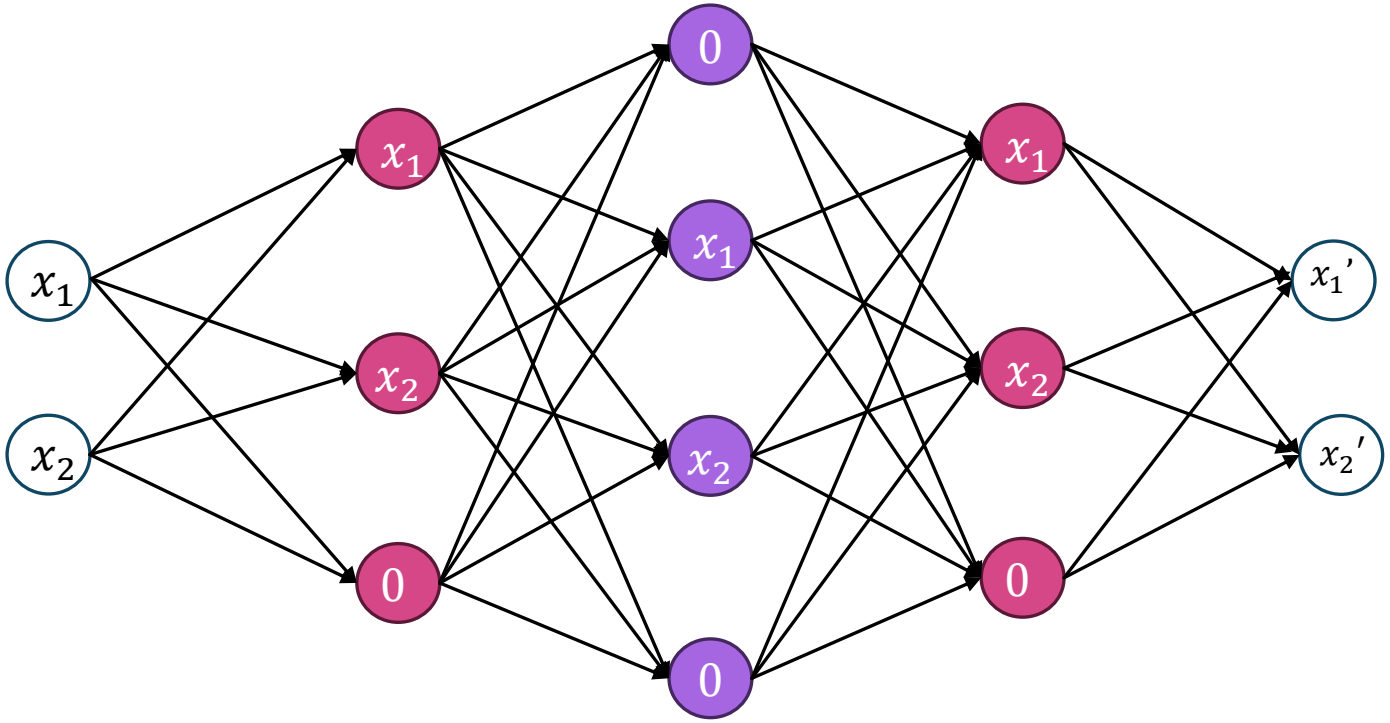
The layers may just copy the input data and append zeroes to reconstruct the data at the output

As a result, no salient representation is created at the latent layer

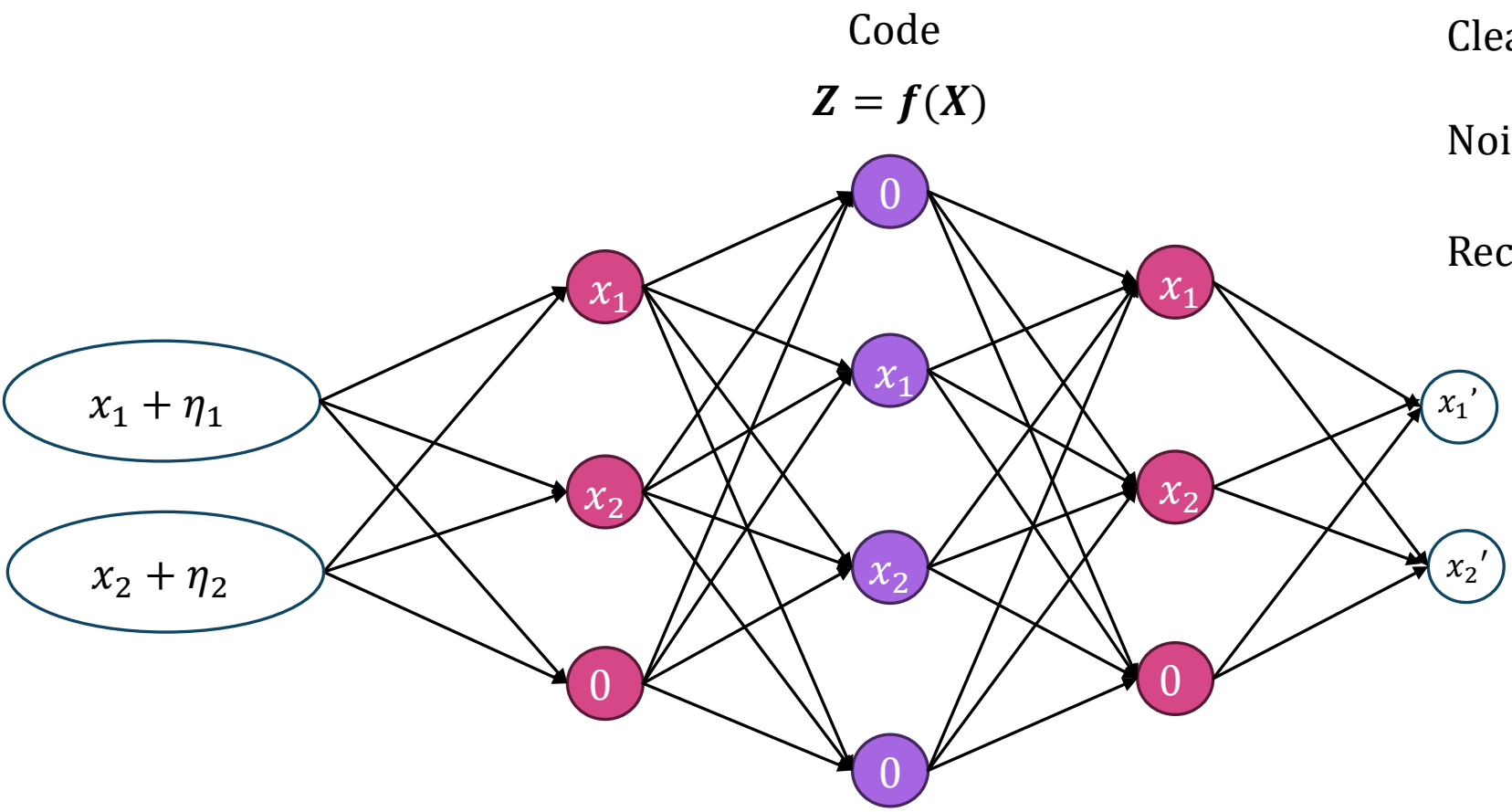
Overcomplete Autoencoders: Can it be Still Useful?

Code

$$\boldsymbol{Z} = \boldsymbol{f}(\boldsymbol{X})$$



Denoising Autoencoder



Clean data $x = [x_1 \;\; x_2]$

Noisy data $x_\eta = [(x_1 + \eta_1) \;\; (x_2 + \eta_2)]$

Reconstructed data $x' = [x_1' \;\; x_2']$

Denoising Autoencoder

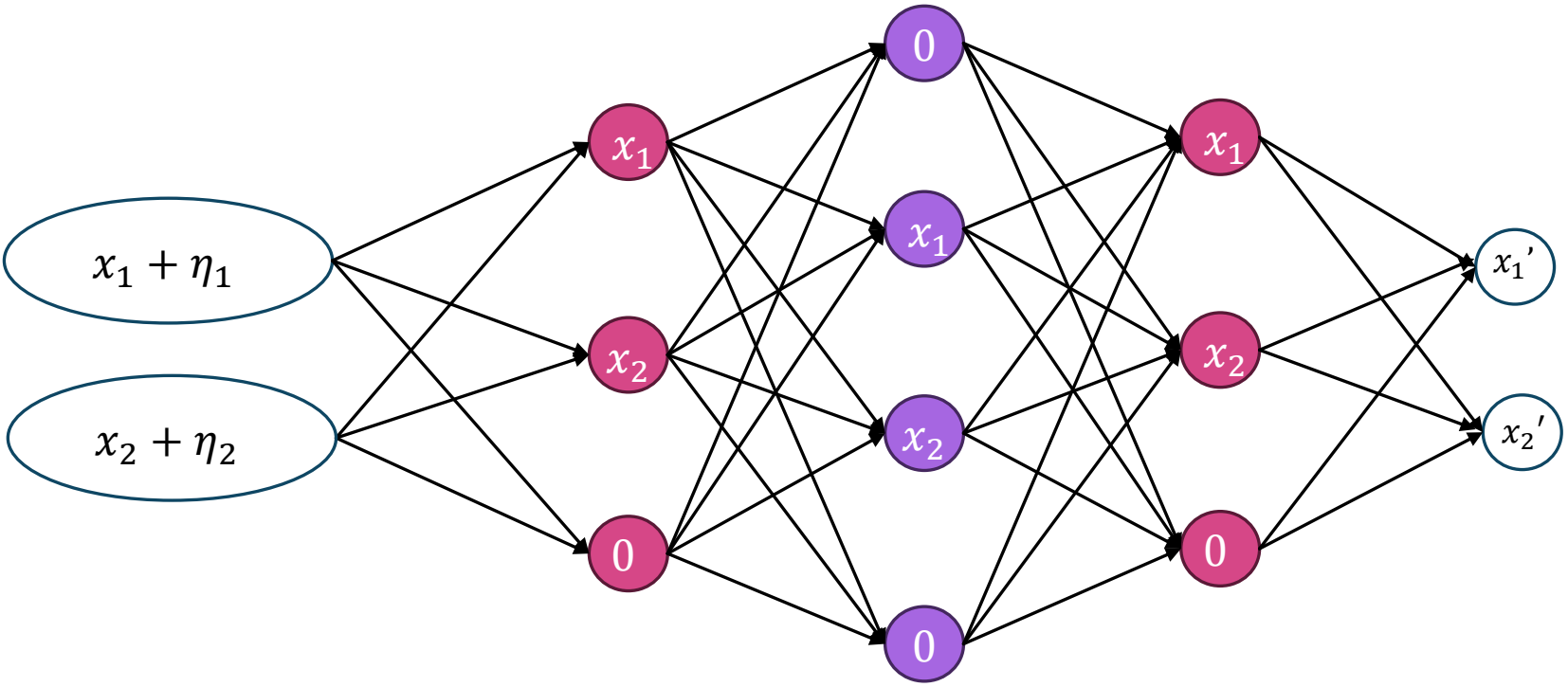
Clean data $x = [x_1 \;\; x_2]$

Noisy data $x_\eta = [(x_1 + \eta_1) \;\; (x_2 + \eta_2)]$

Reconstructed data $x' = [x_1' \;\; x_2']$

Code

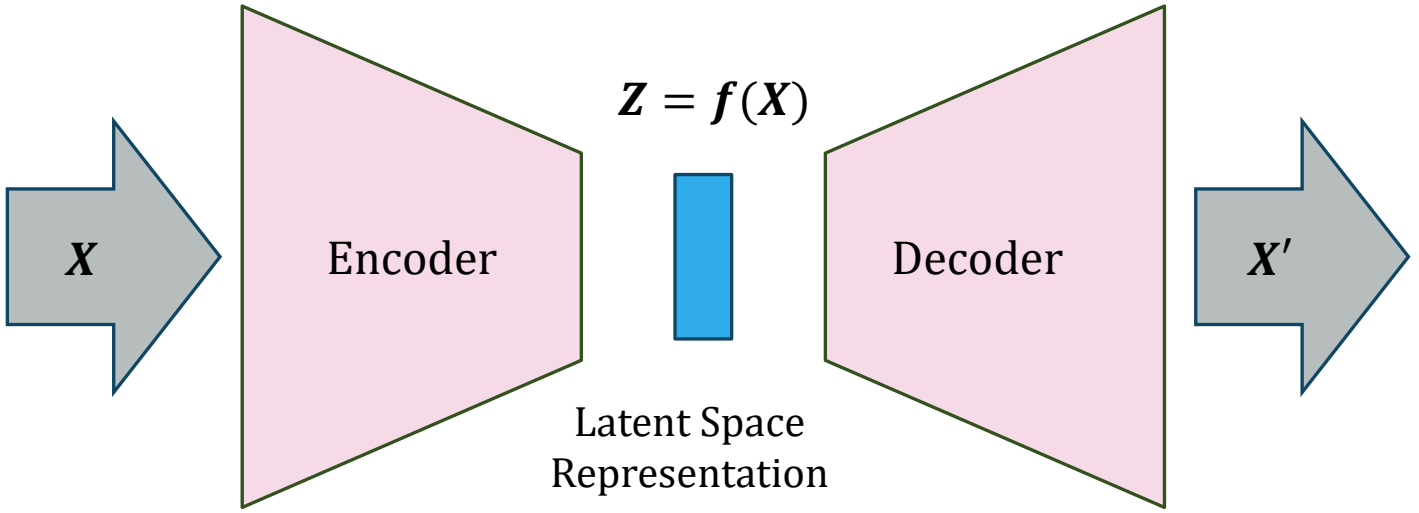
$\boldsymbol{Z} = \boldsymbol{f}(\boldsymbol{X})$



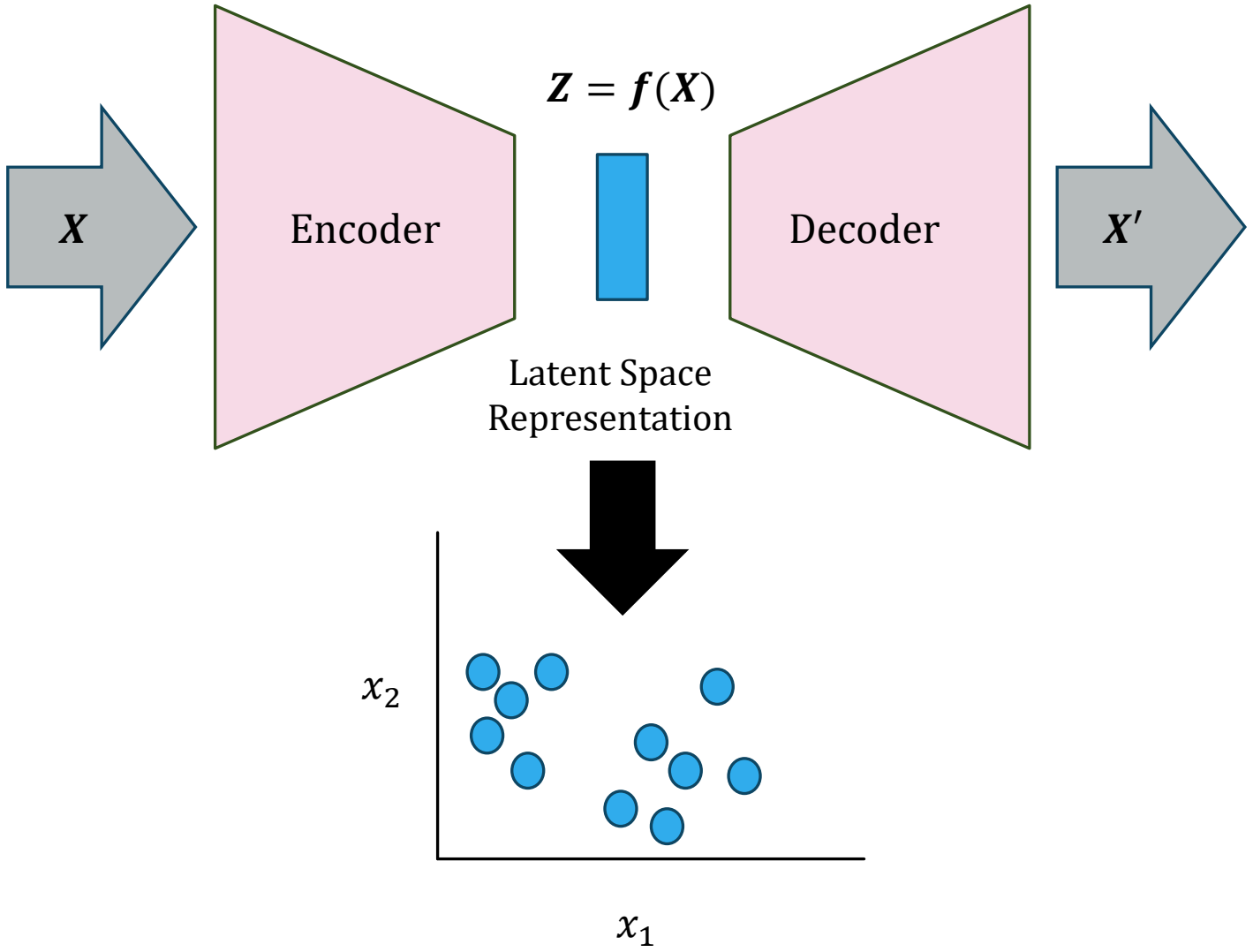
During training, we apply noisy data at input and ask the AE to reconstruct the clean data

For that, we minimize $MSE(x, x')$

Representation by Autoencoders



Representation by Autoencoders

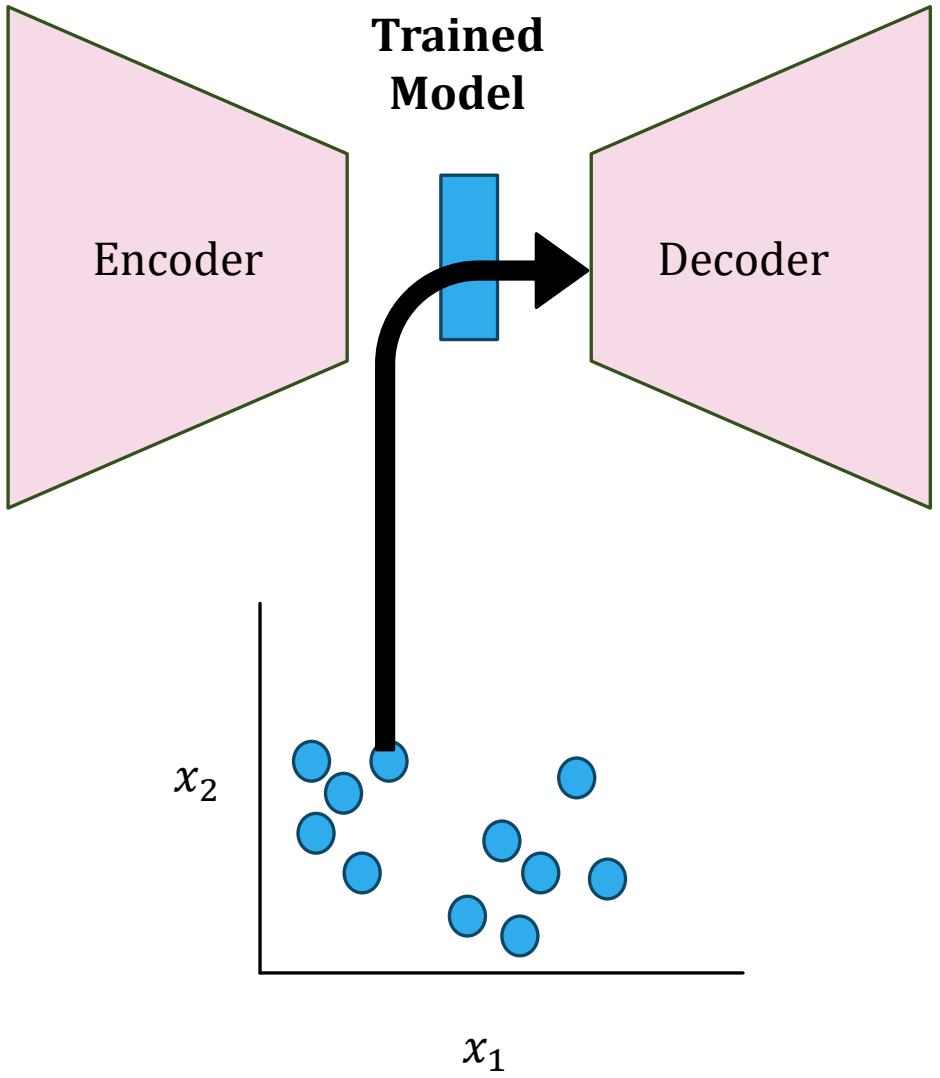


The latent space representation is discrete in nature

Representation by Autoencoders

The latent space representation is discrete in nature

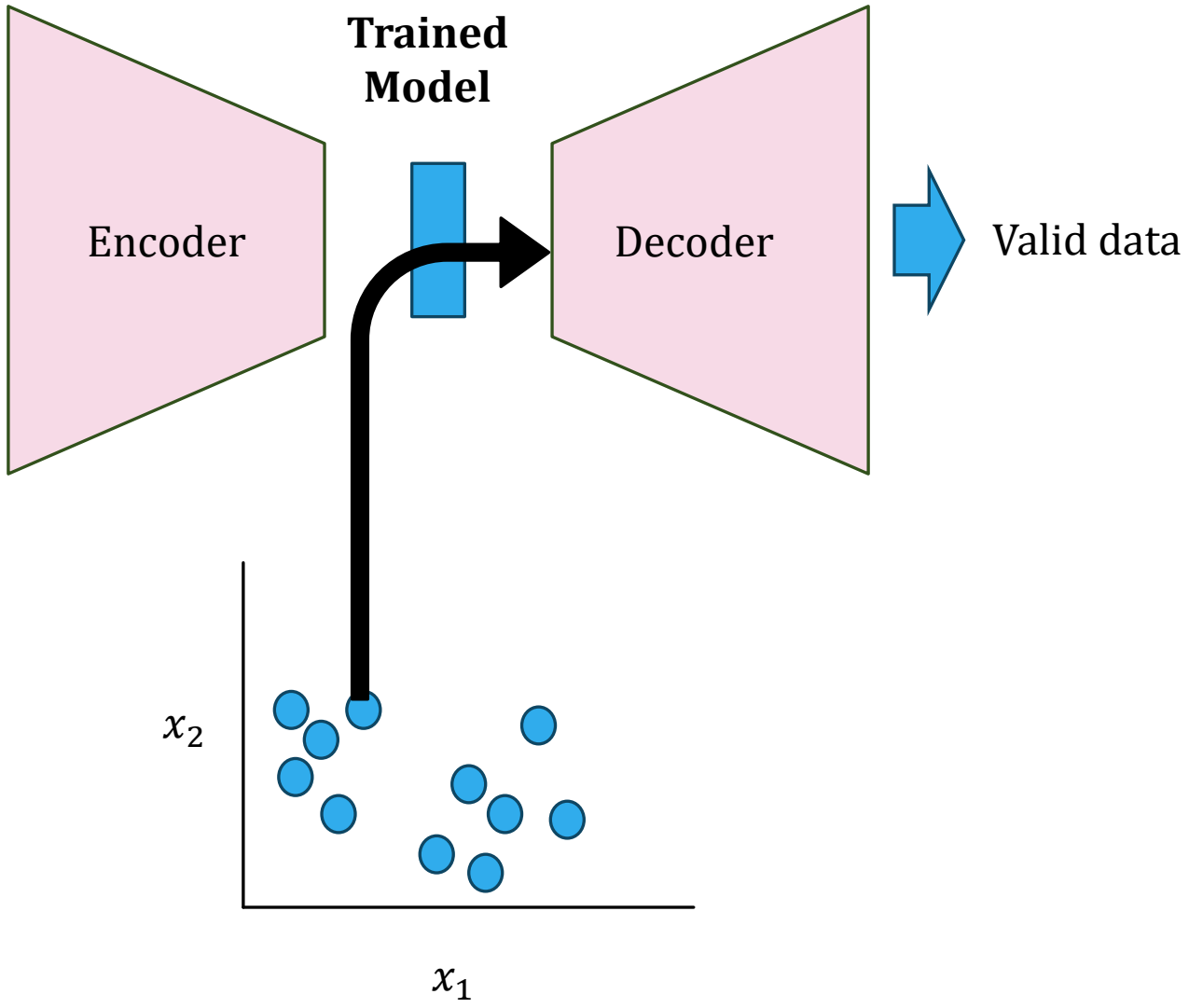
After training, if I take an actual projection point in the latent space and feed it to the decoder, the decoder will produce a valid data



Representation by Autoencoders

The latent space representation is discrete in nature

After training, if I take an actual projection point in the latent space and feed it to the decoder, the decoder will produce a valid data

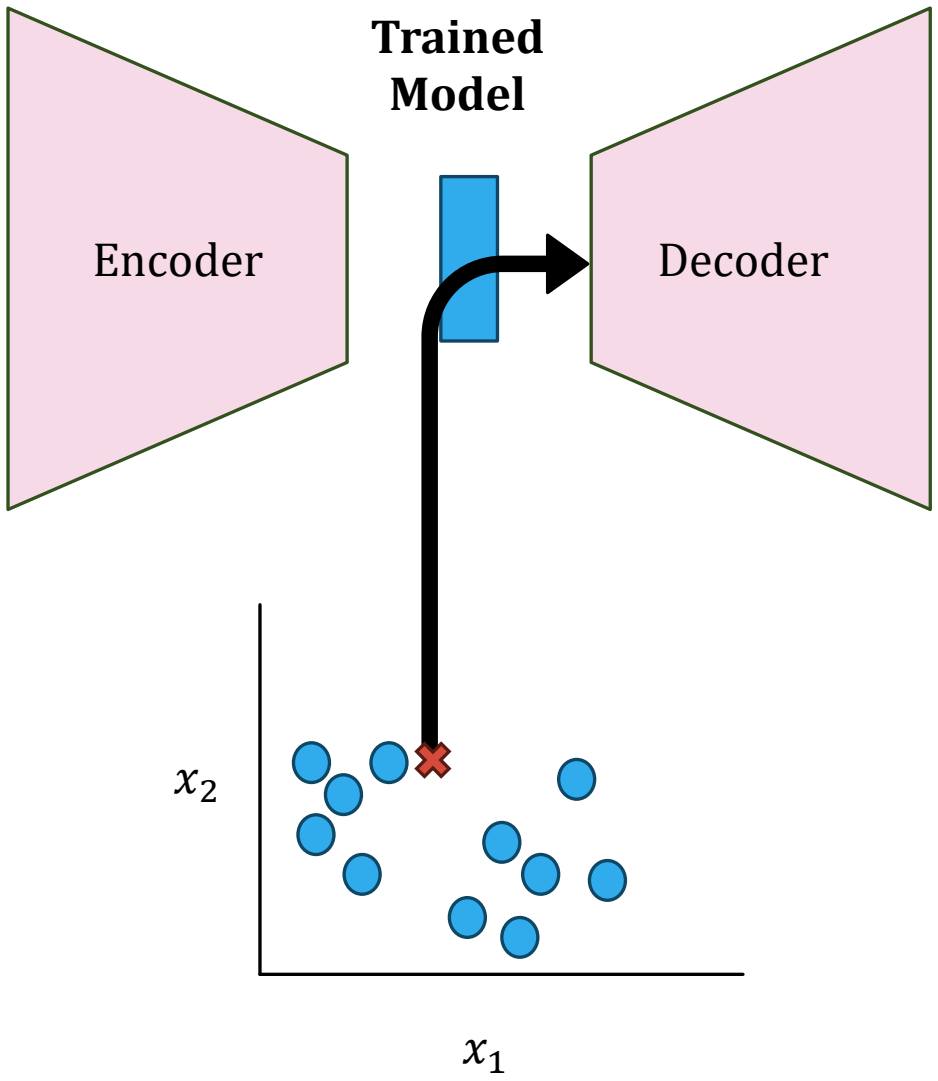


Representation by Autoencoders

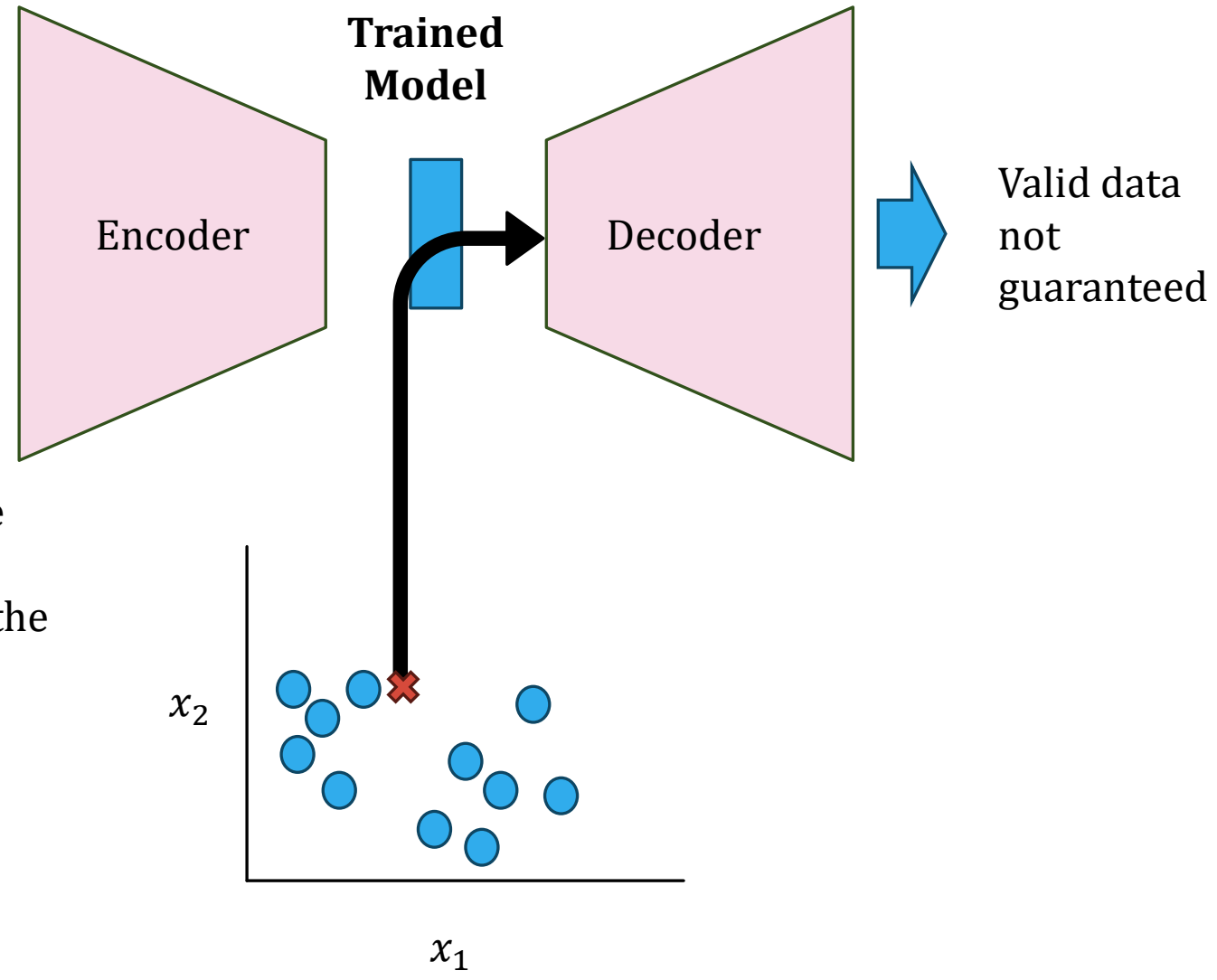
The latent space representation is discrete in nature

After training, if I take an actual projection point in the latent space and feed it to the decoder, the decoder will produce a valid data

However, if a take any other point and feed it to the decoder, it is not guaranteed to provide valid data



Representation by Autoencoders



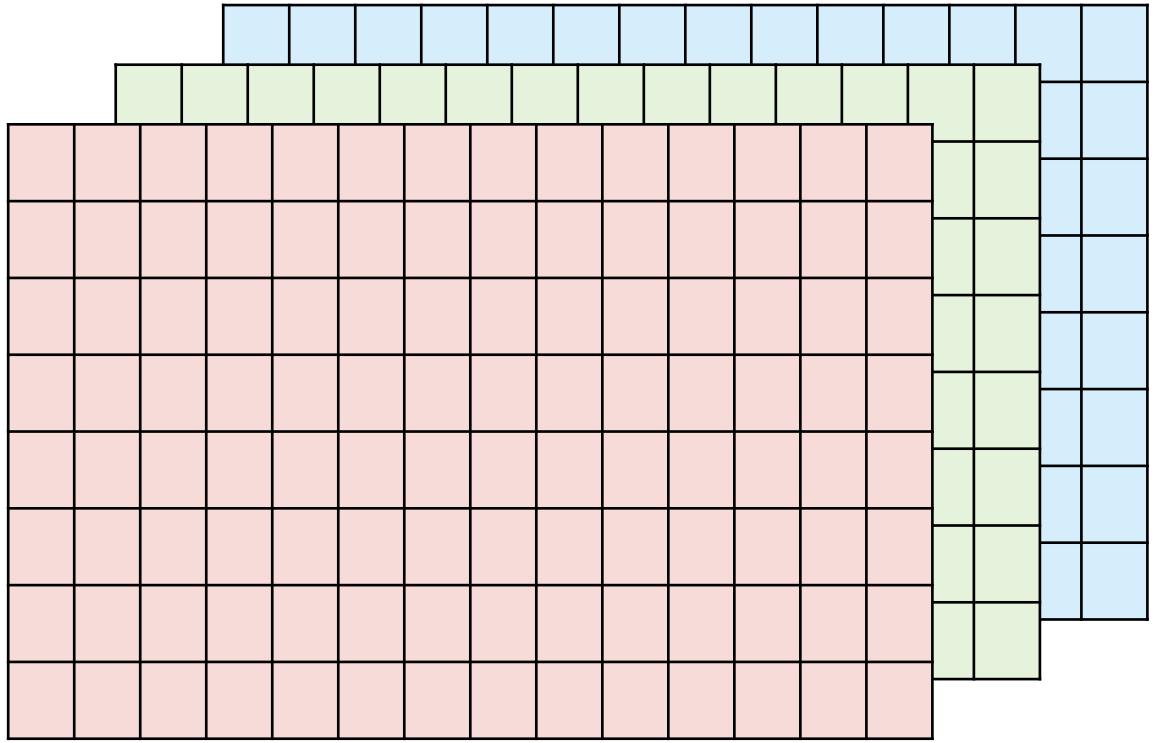
The latent space representation is discrete in nature

After training, if I take an actual projection point in the latent space and feed it to the decoder, the decoder will produce a valid data

However, if a take any other point and feed it to the decoder, it is not guaranteed to provide valid data

Convolutional Neural Networks

Image



An image is 2D (3D for colour) array of pixels

https://www.livescience.com/27443-zebras.html

Image

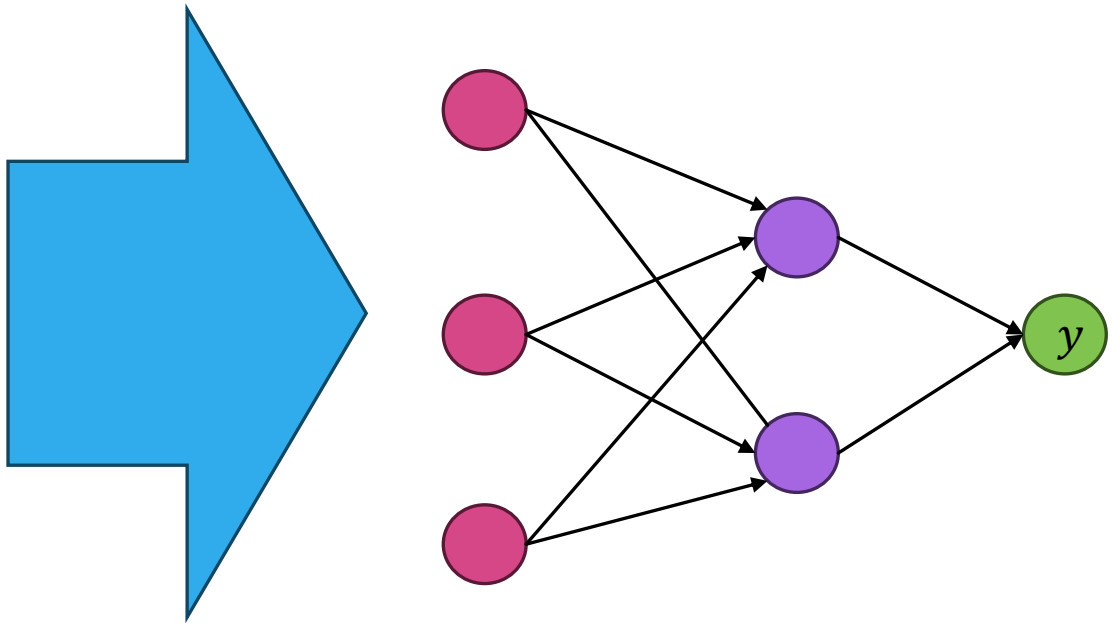
Image A

138	`182	211
79	153	241
101	122	89

NN for Image

Image A

138	`182	211
79	153	241
101	122	89

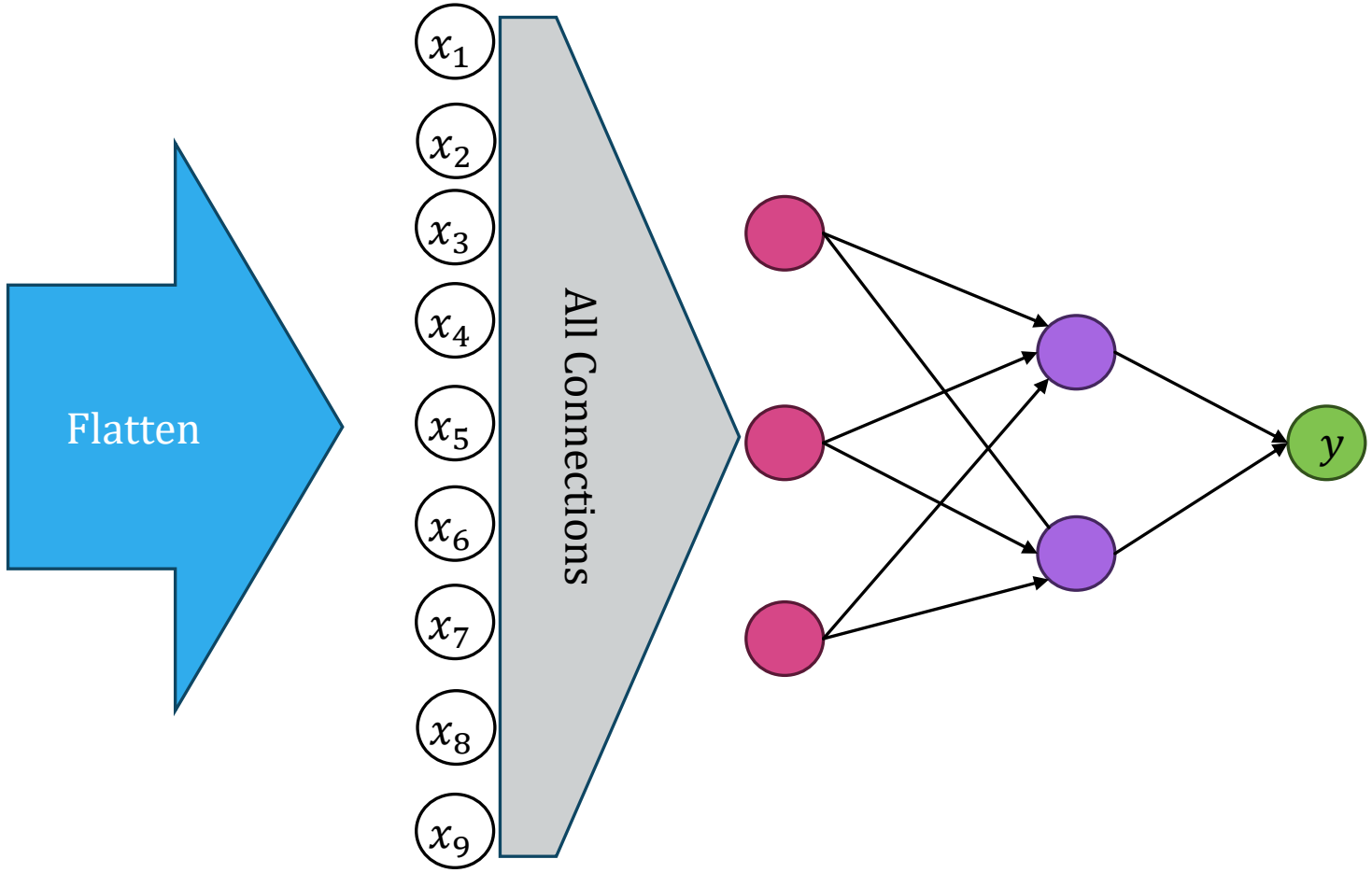


How can I Apply image A to this NN?

NN for Image

Image A

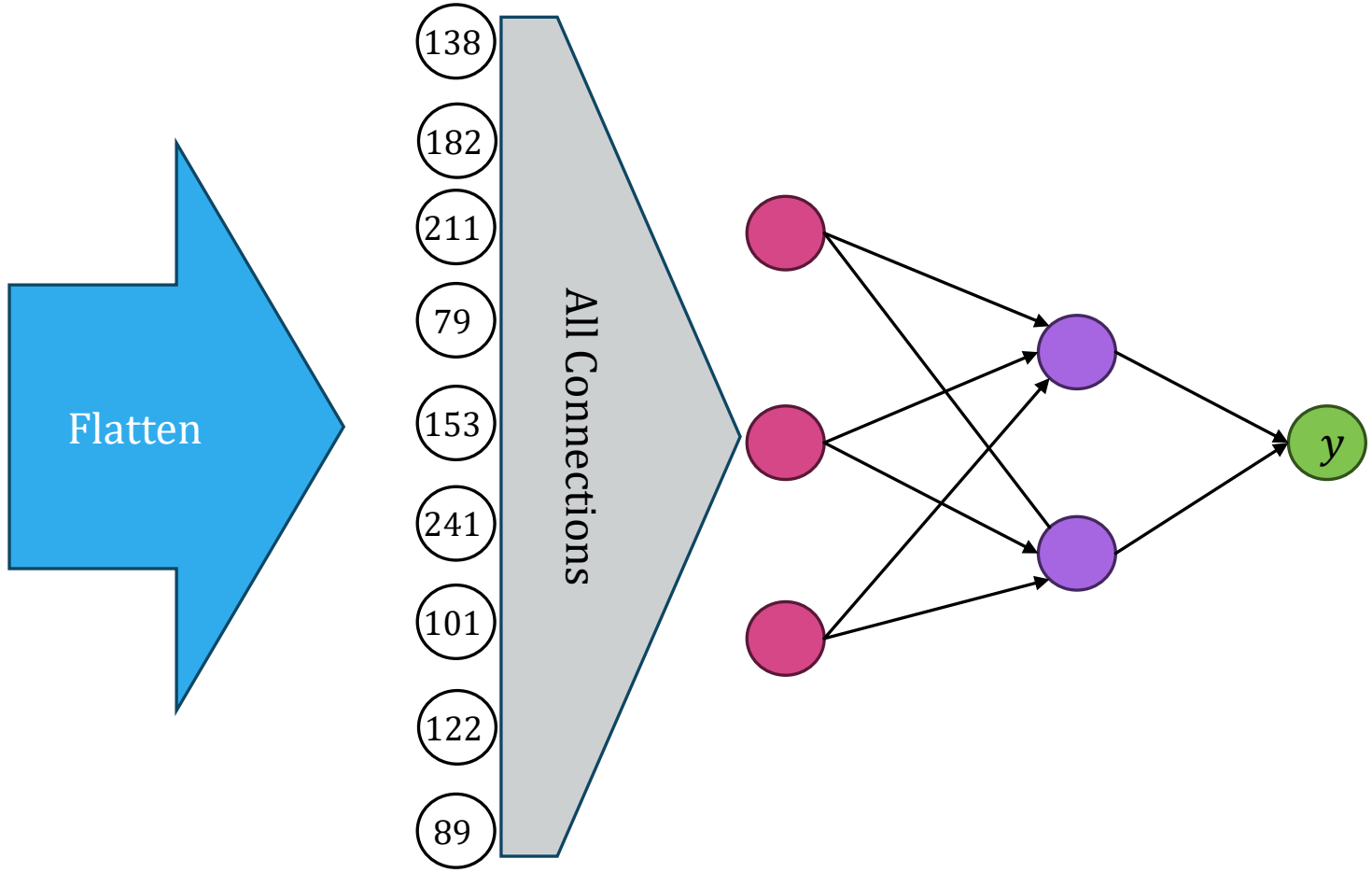
138	`182	211
79	153	241
101	122	89



How can I Apply image A to this NN?

Image A

138	`182	211
79	153	241
101	122	89

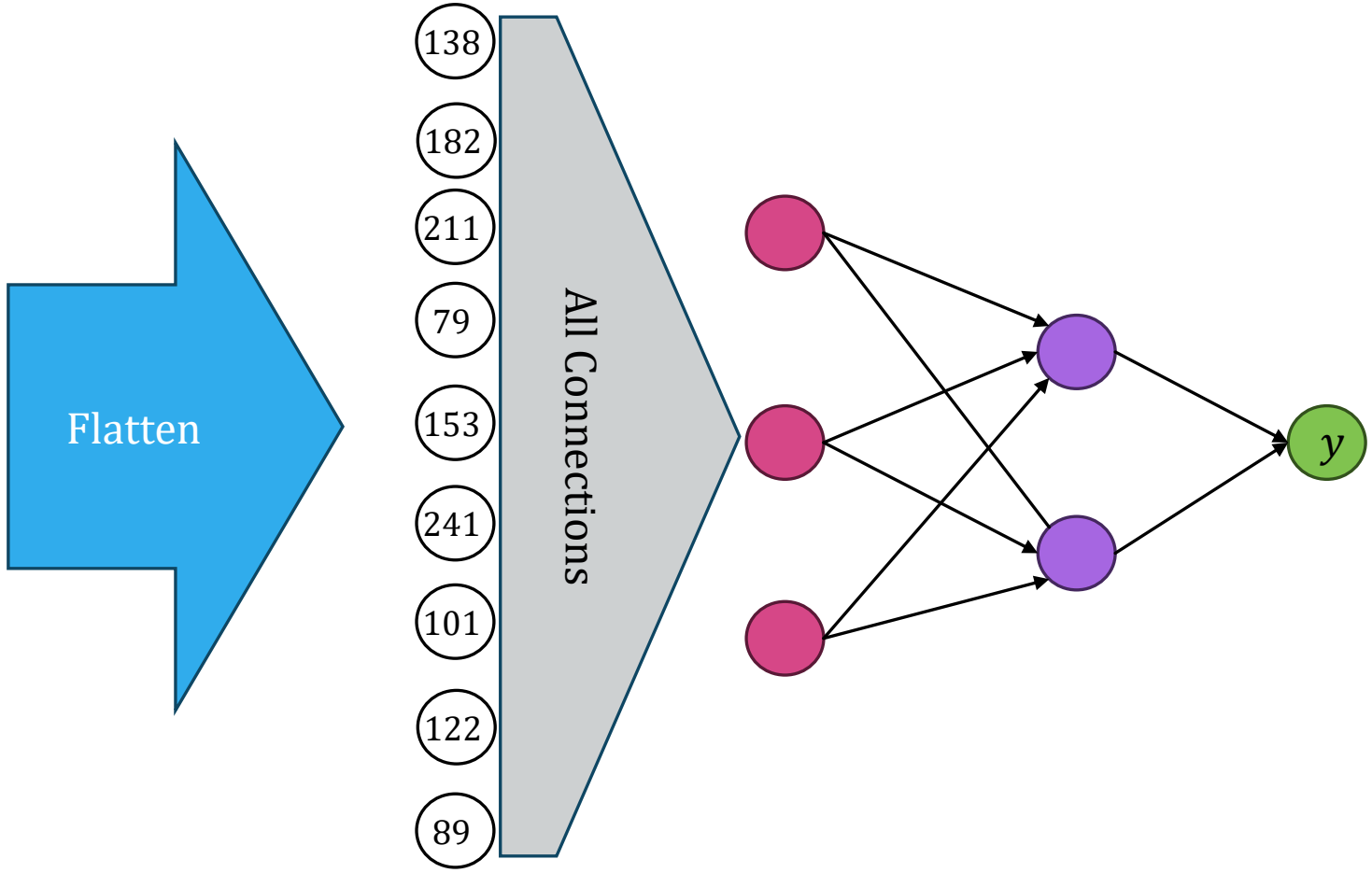


What is the problem with this approach?

Image A

138	`182	211
79	153	241
101	122	89

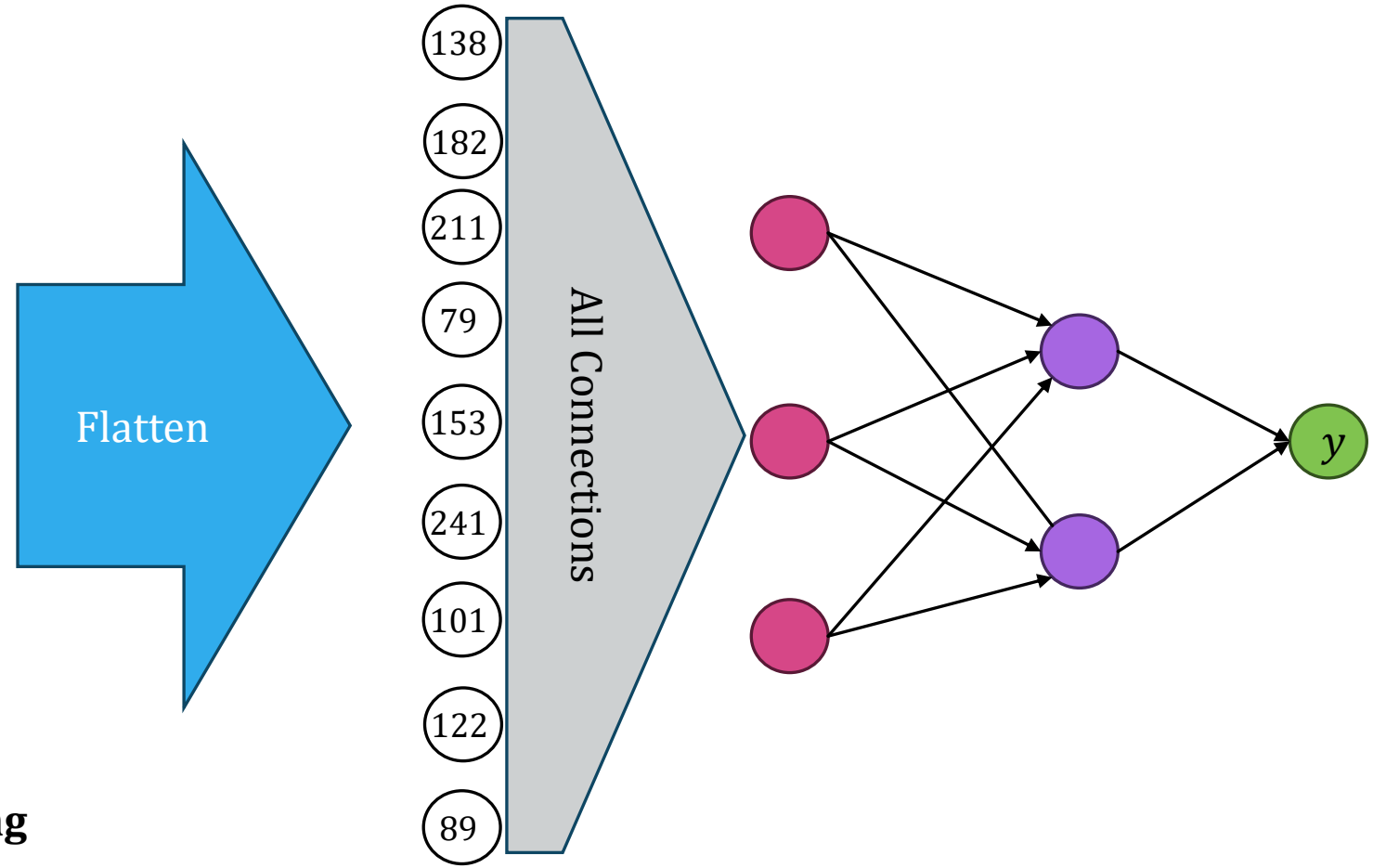
Loss of spatial context



What is the problem with this approach?

Image A

138	`182	211
79	153	241
101	122	89



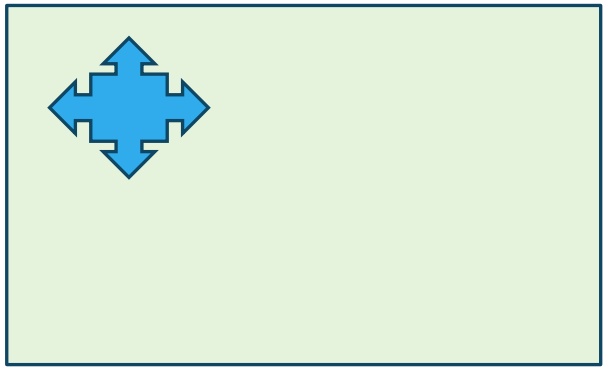
Too may trainable parameters

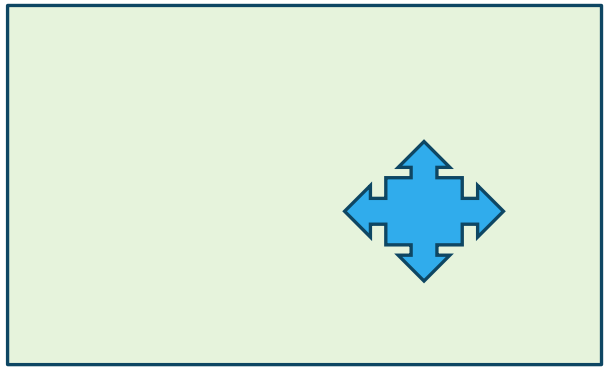
- **Increased possibility of overfitting**

What is the problem with this approach?

NN for Image: What Do We Want?

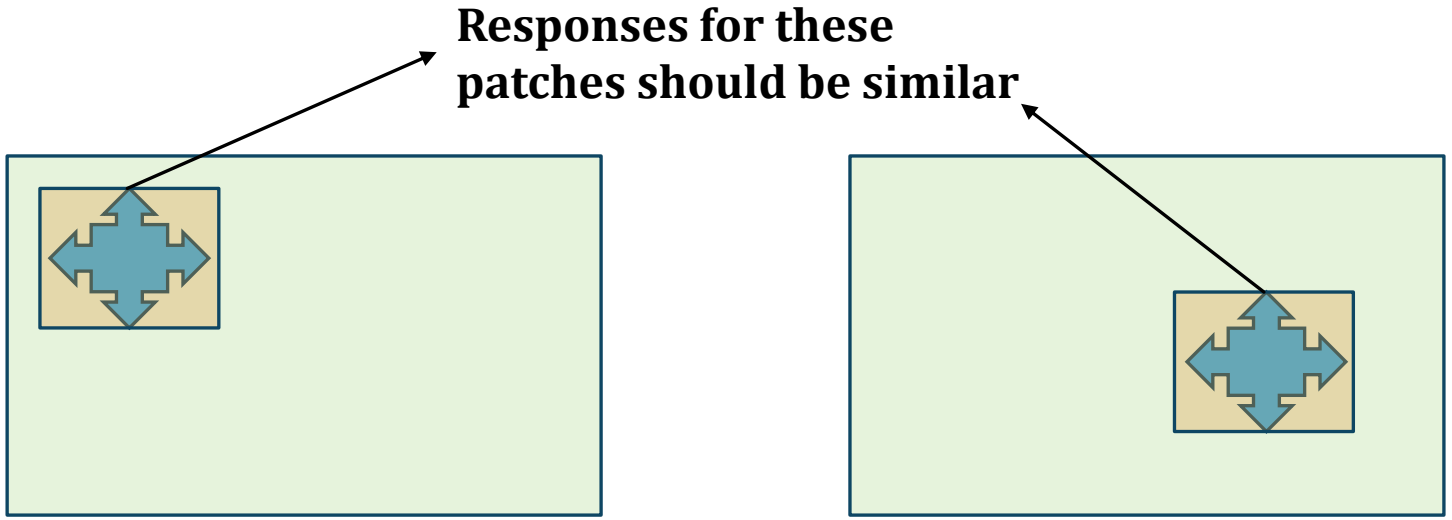
- An object may be located anywhere in an image
  - Our model should respond to same image patch regardless of its position
  - We should not need training data for each position of an object
  - Translation invariance/ equivariance





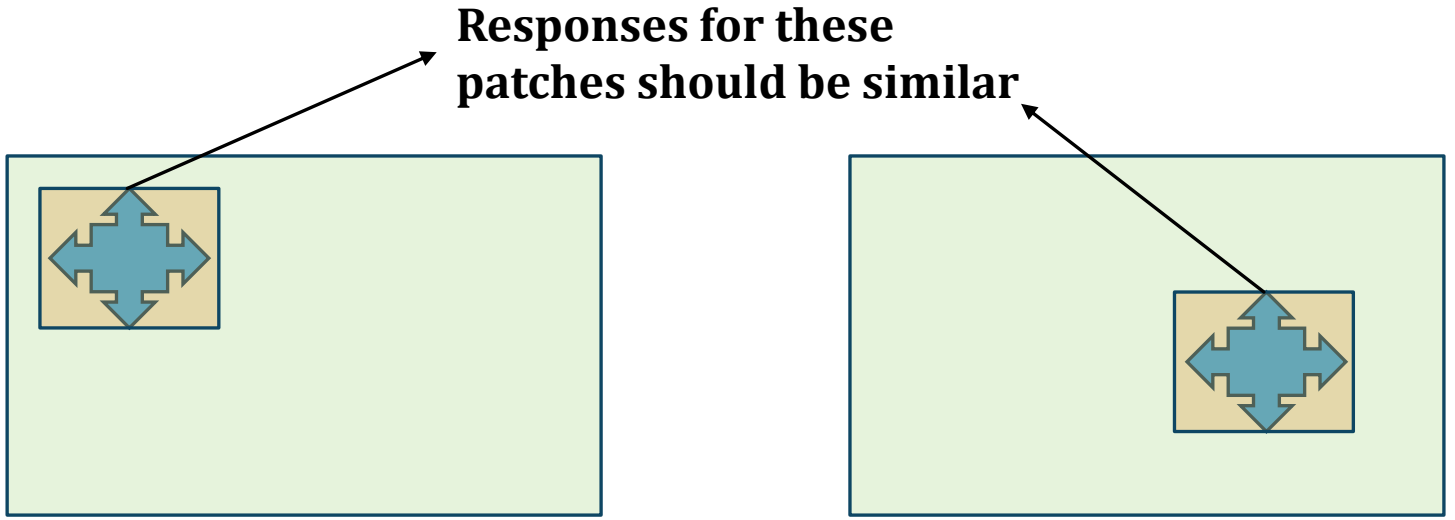
NN for Image: What Do We Want?

- An object may be located anywhere in an image
  - Our model should respond to same image patch regardless of its position
  - Translation invariance/ equivariance



NN for Image: What Do We Want?

- An object may be located anywhere in an image
  - Our model should respond to same image patch regardless of its position
  - Translation invariance/ equivariance



Operate on the patches with same set of numbers

NN for Image: What Do We Want?

- Preserve spatial context
  - Don’t flatten the image
  - Operate on the image itself
    - Operate on the image patches
- Reduce the number of parameters
  - Do parameter sharing across image patches

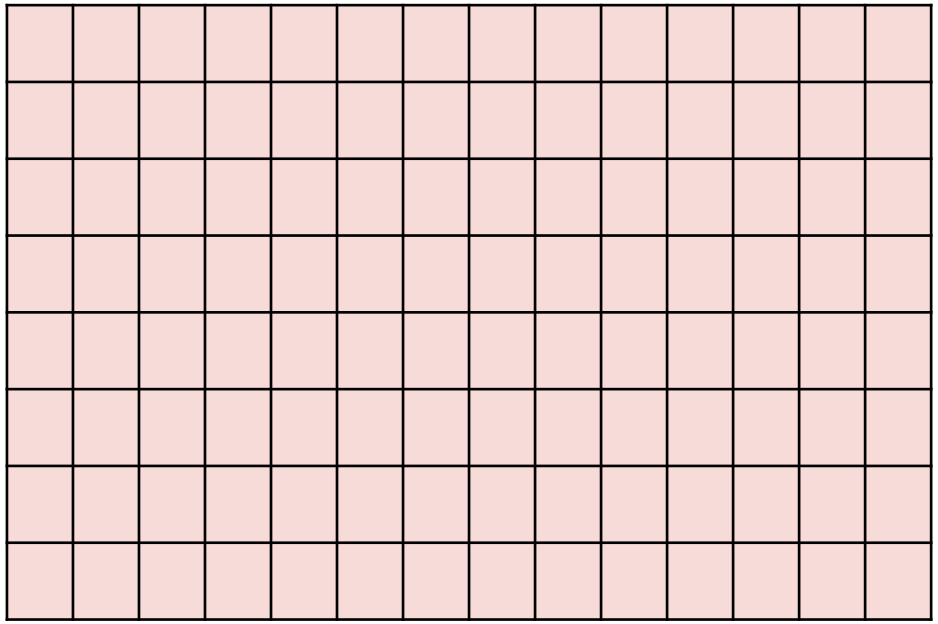
The New Type of NN: Cross-correlation

For NN, we use cross-correlation. So,

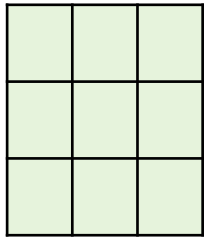
$$S(i,j) = (K * I)(i,j) = \sum_m \sum_m I(i+m, j+n) K(m,n)$$

How to Achieve These?

Image (1-channel)

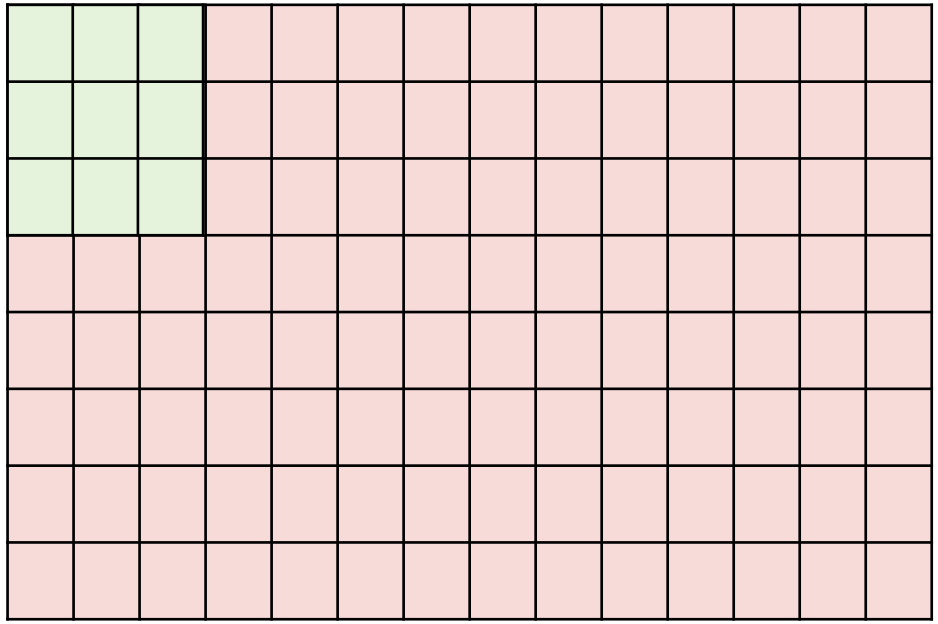


Kernel/ filter



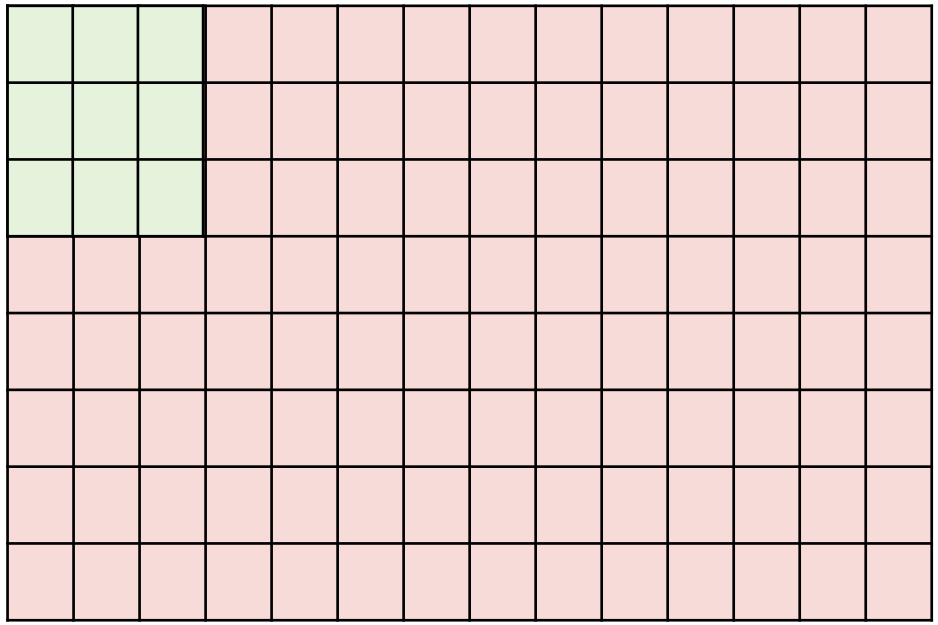
How to Achieve These?

Image (1-channel)



How Convolution Works

Image (1-channel)



Multiply signal elements with kernel elements and add

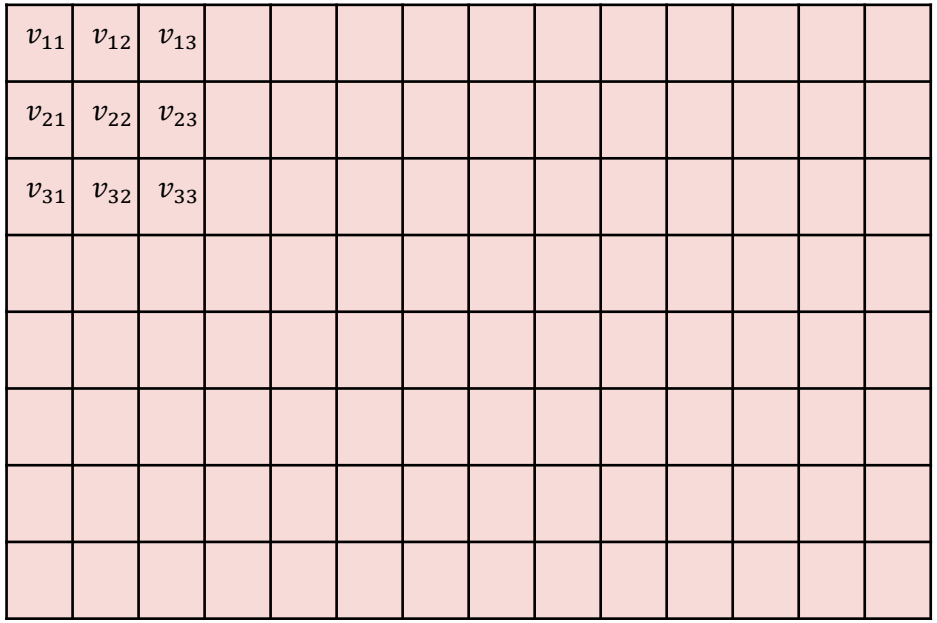
Replace the element at the center with the resultant value

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_m I(i+m, j+n) K(m,n)$$

How to Achieve These?

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Image (1-channel)



Multiply signal elements with kernel elements and add

Replace the element at the center with the resultant value

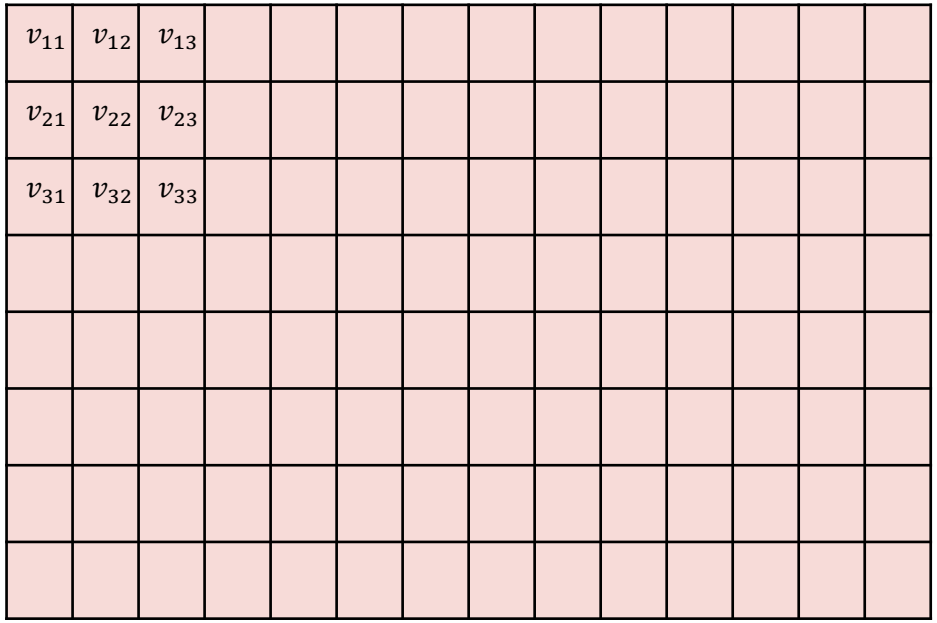
$$S(i,j) = (K * I)(i,j) = \sum_m \sum_m I(i+m, j+n)K(m,n)$$

Convolution

How to Achieve These?

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Image (1-channel)



Multiply signal elements with kernel elements and add

Replace the element at the center with the resultant value

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_m I(i+m, j+n)K(m,n)$$

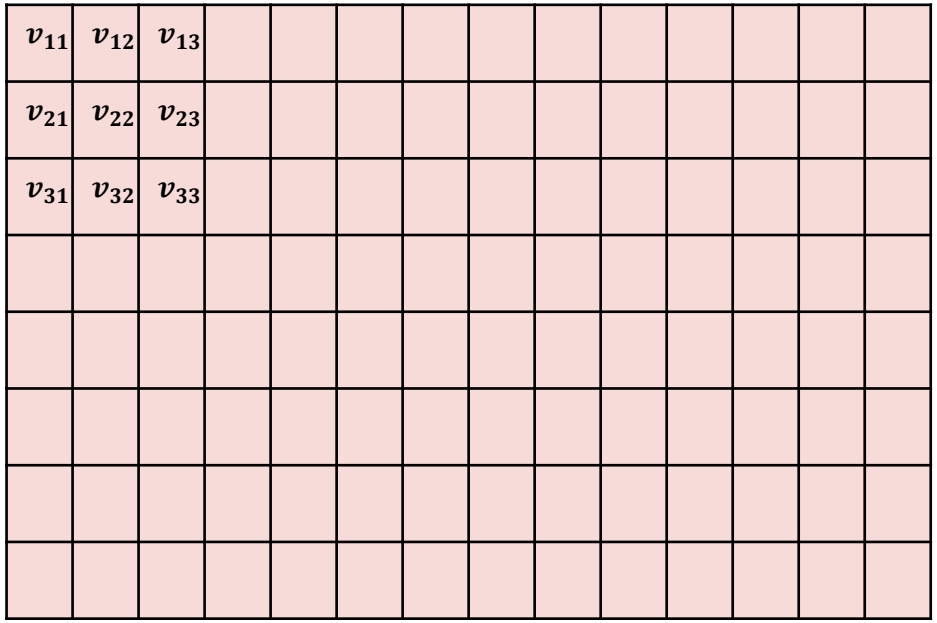
Convolution

(Although mathematically, it's cross-correlation. Every time, we are calculating the correlation between an image patch and the kernel)

How Convolution Works

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Image (1-channel)



Multiply signal elements with kernel elements and add

Replace the element at the center with the resultant value

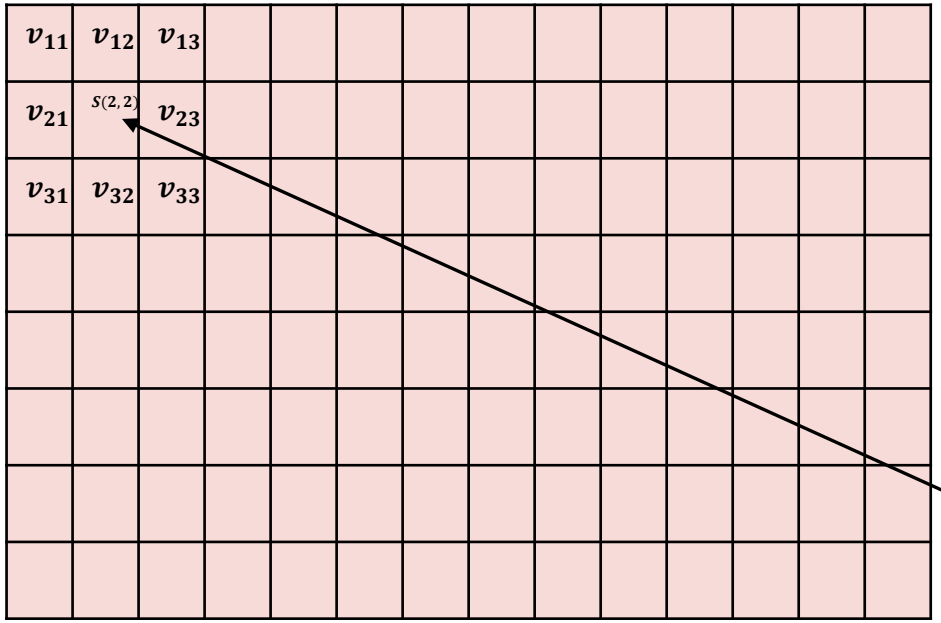
$$S(i,j) = (K * I)(i,j) = \sum_m \sum_m I(i+m, j+n) K(m,n)$$

$$\boldsymbol{S(2,2) = w_1 v_{11} + w_2 v_{12} + \cdots + w_9 v_{33}}$$

How Convolution Works

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Image (1-channel)



Multiply signal elements with kernel elements and add

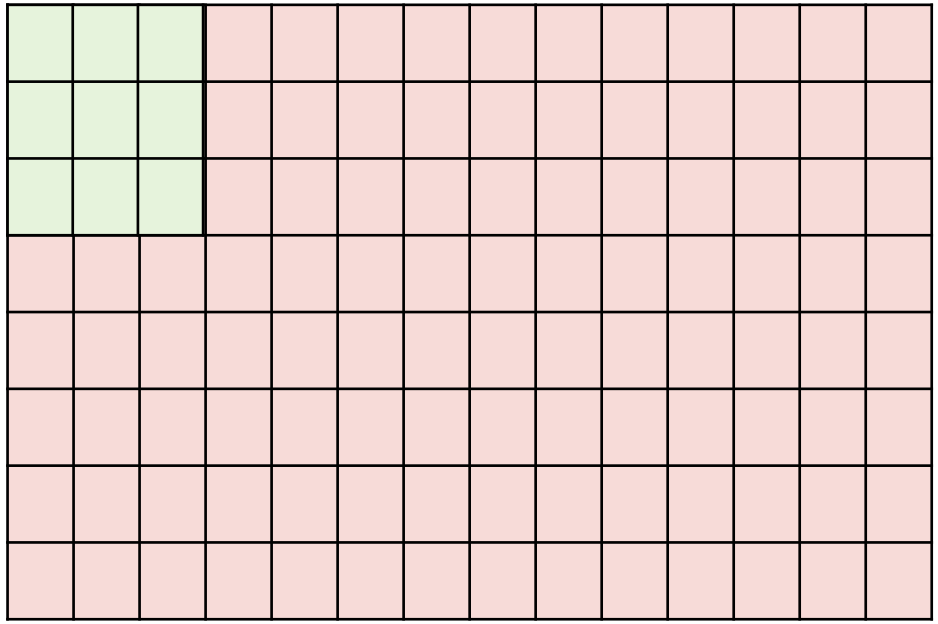
Replace the element at the center with the resultant value

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_m I(i+m, j+n) K(m,n)$$

$$\boldsymbol{S(2,2) = w_1 v_{11} + w_2 v_{12} + \cdots + w_9 v_{33}}$$

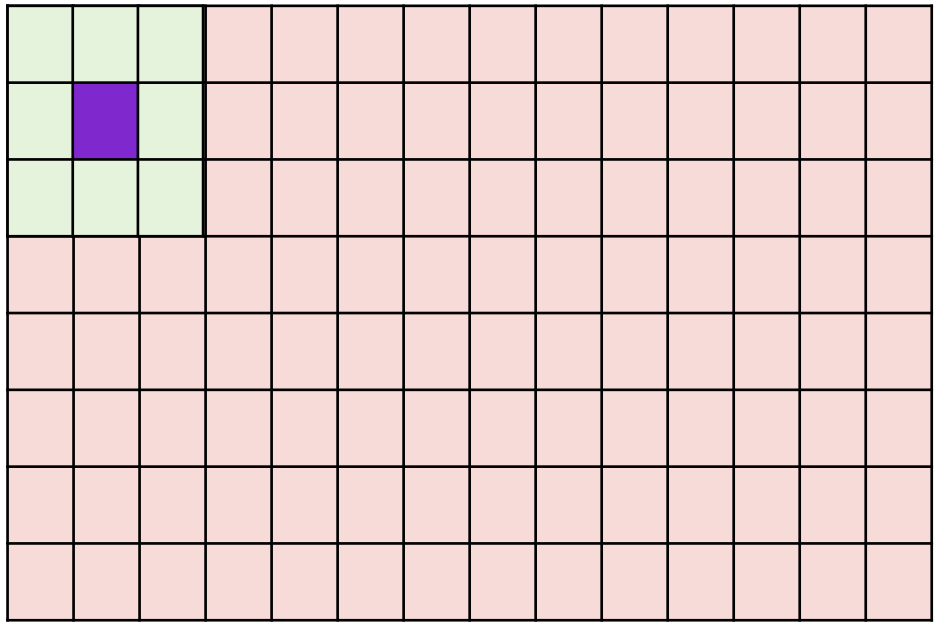
How Convolution Works

Image (1-channel)



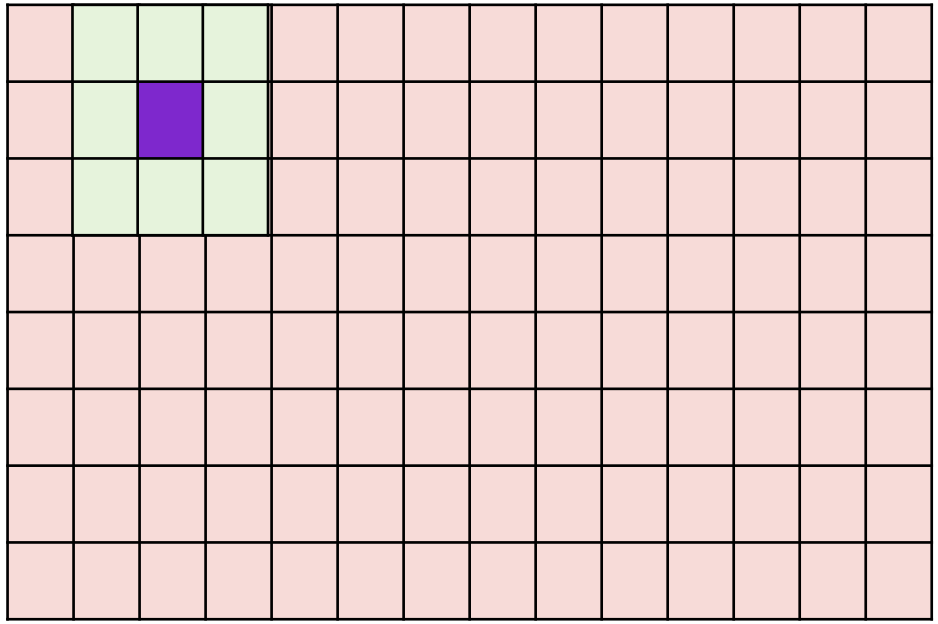
How Convolution Works

Image (1-channel)



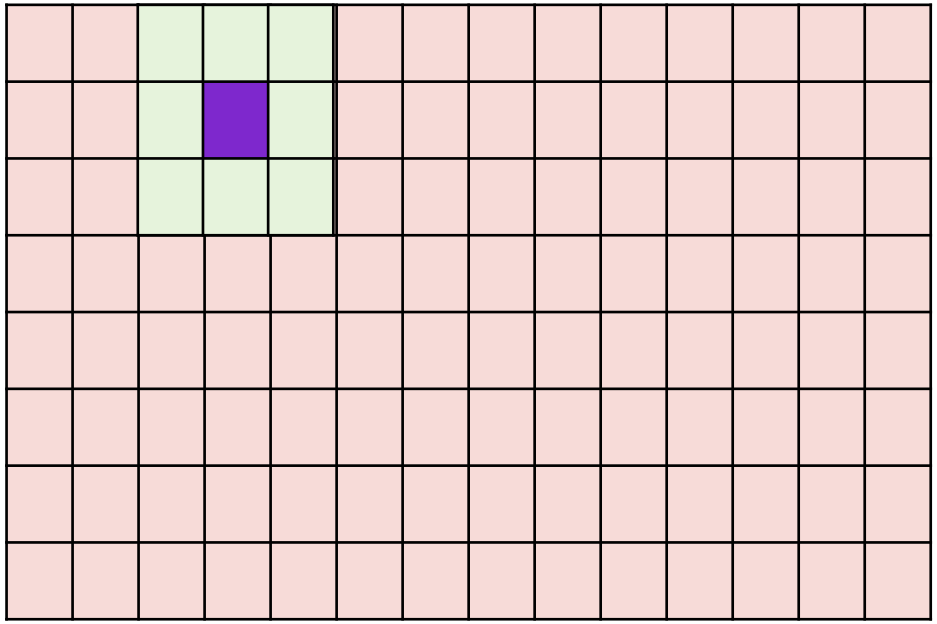
How Convolution Works

Image (1-channel)



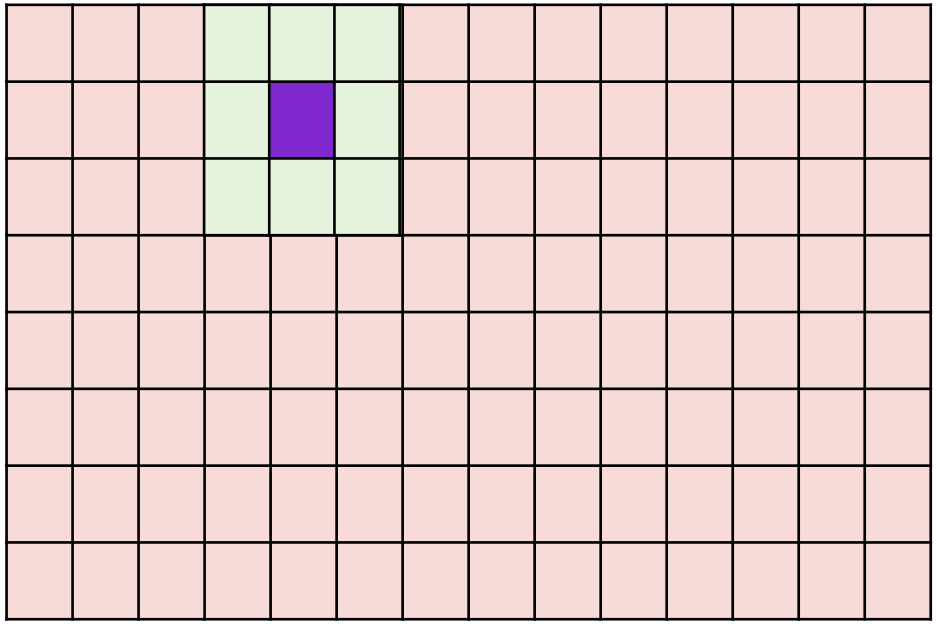
How Convolution Works

Image (1-channel)



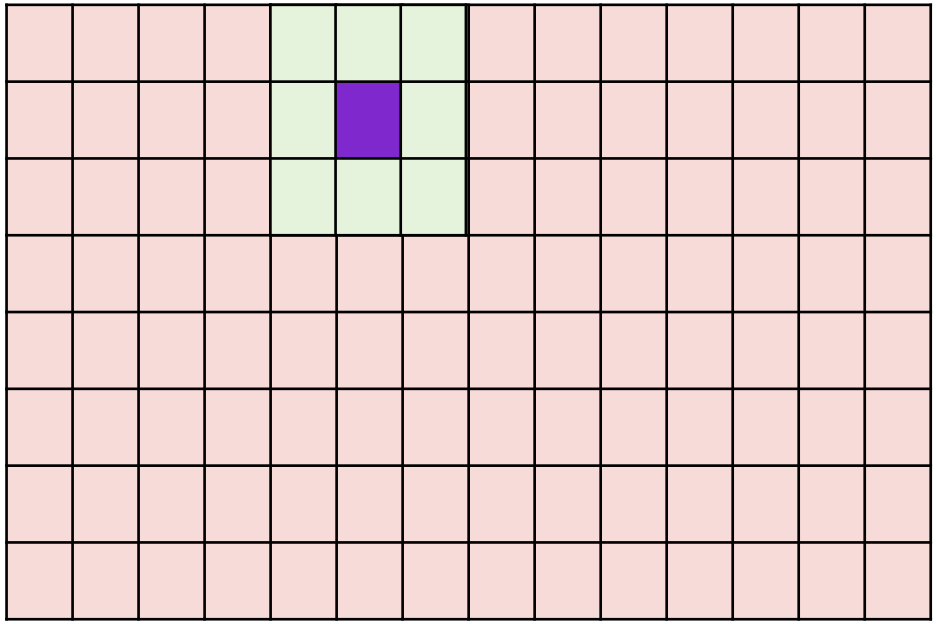
How Convolution Works

Image (1-channel)



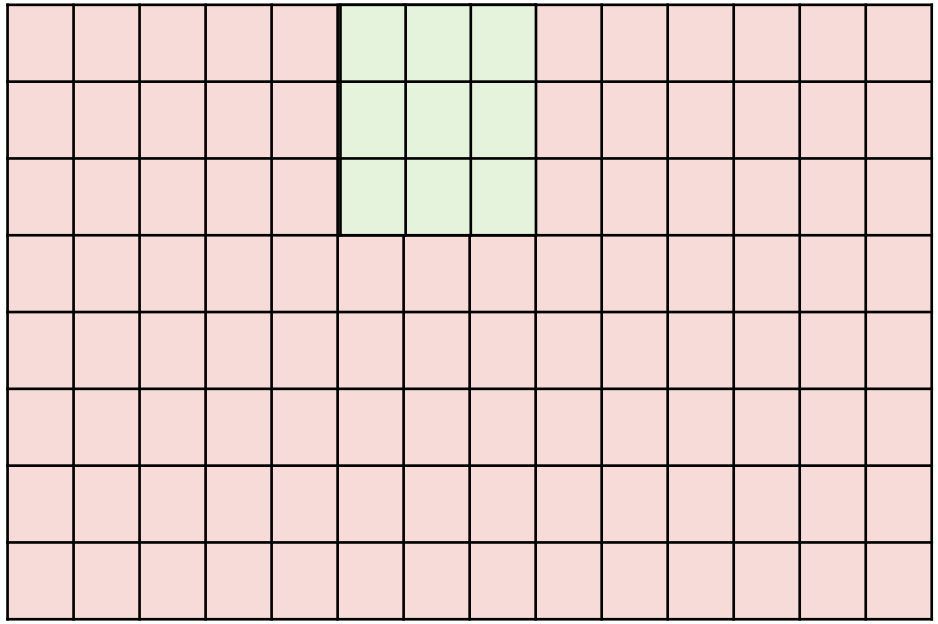
How Convolution Works

Image (1-channel)



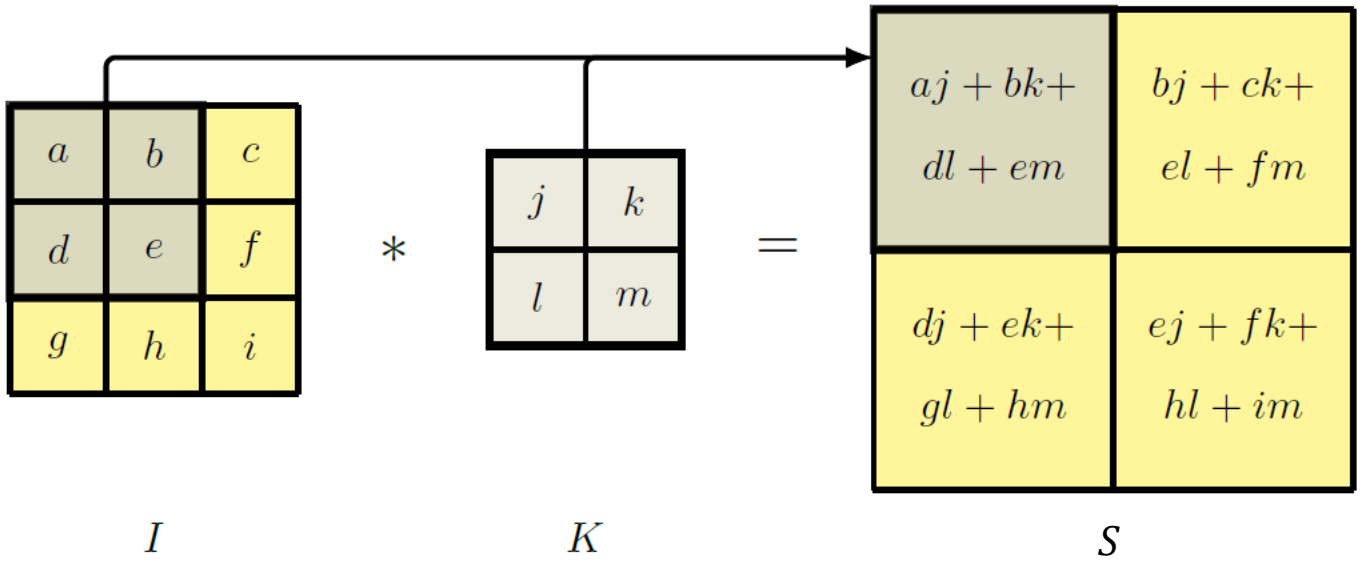
How Convolution Works

Image (1-channel)



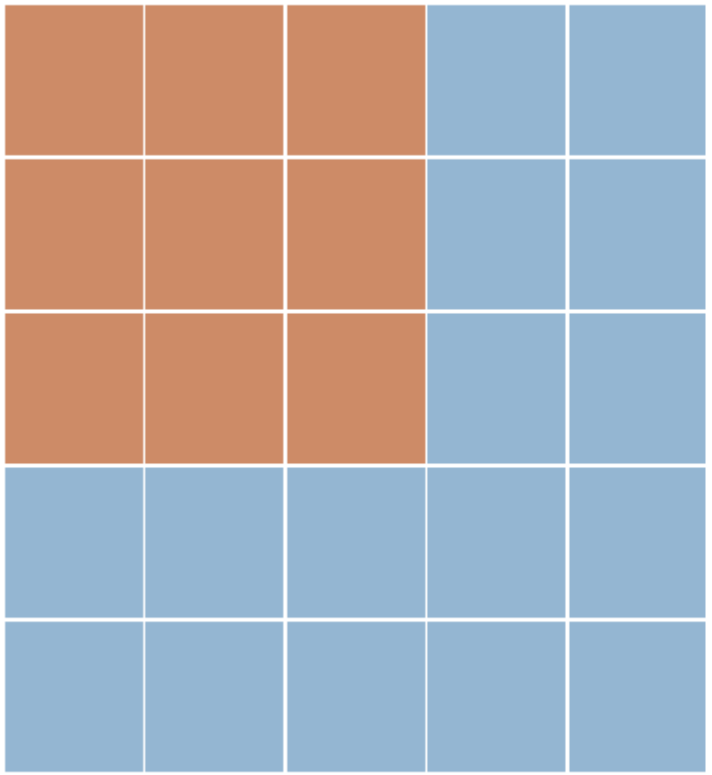
And so on …

How Convolution Works

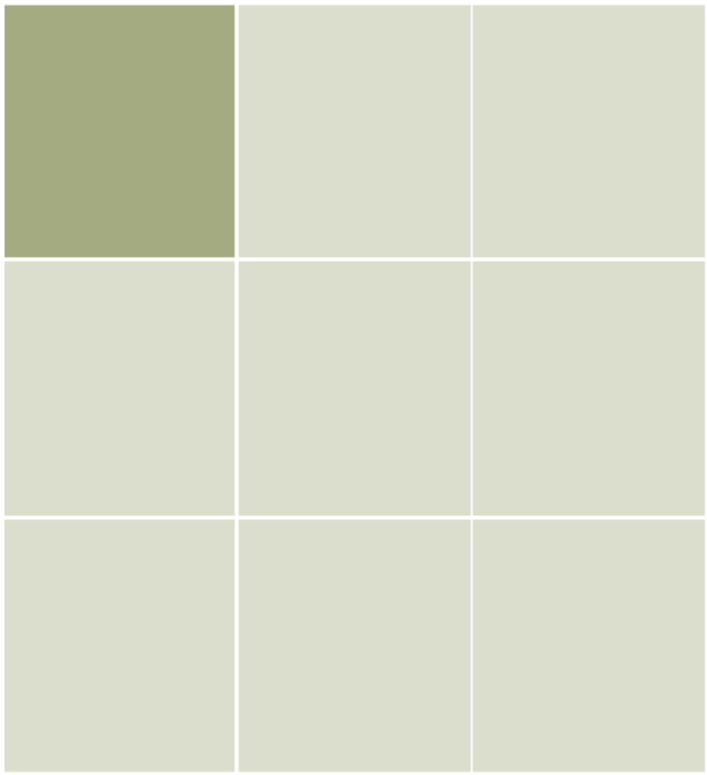


How Convolution Works

Type: conv - Stride: 1 Padding: 0



Input

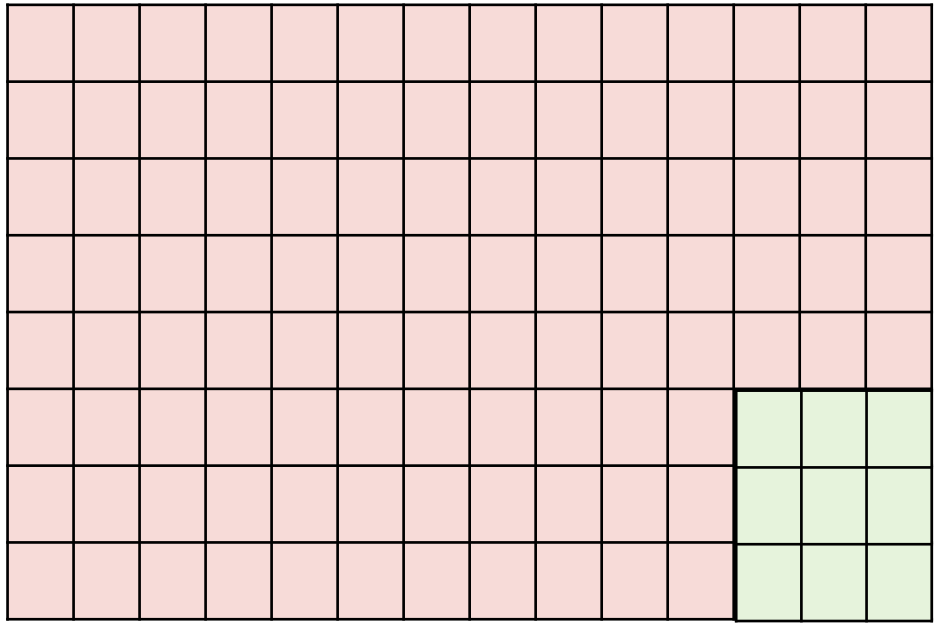


Output

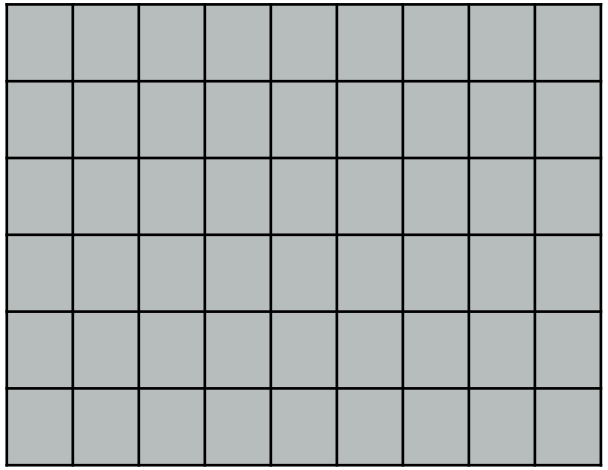
https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11

How Convolution Works

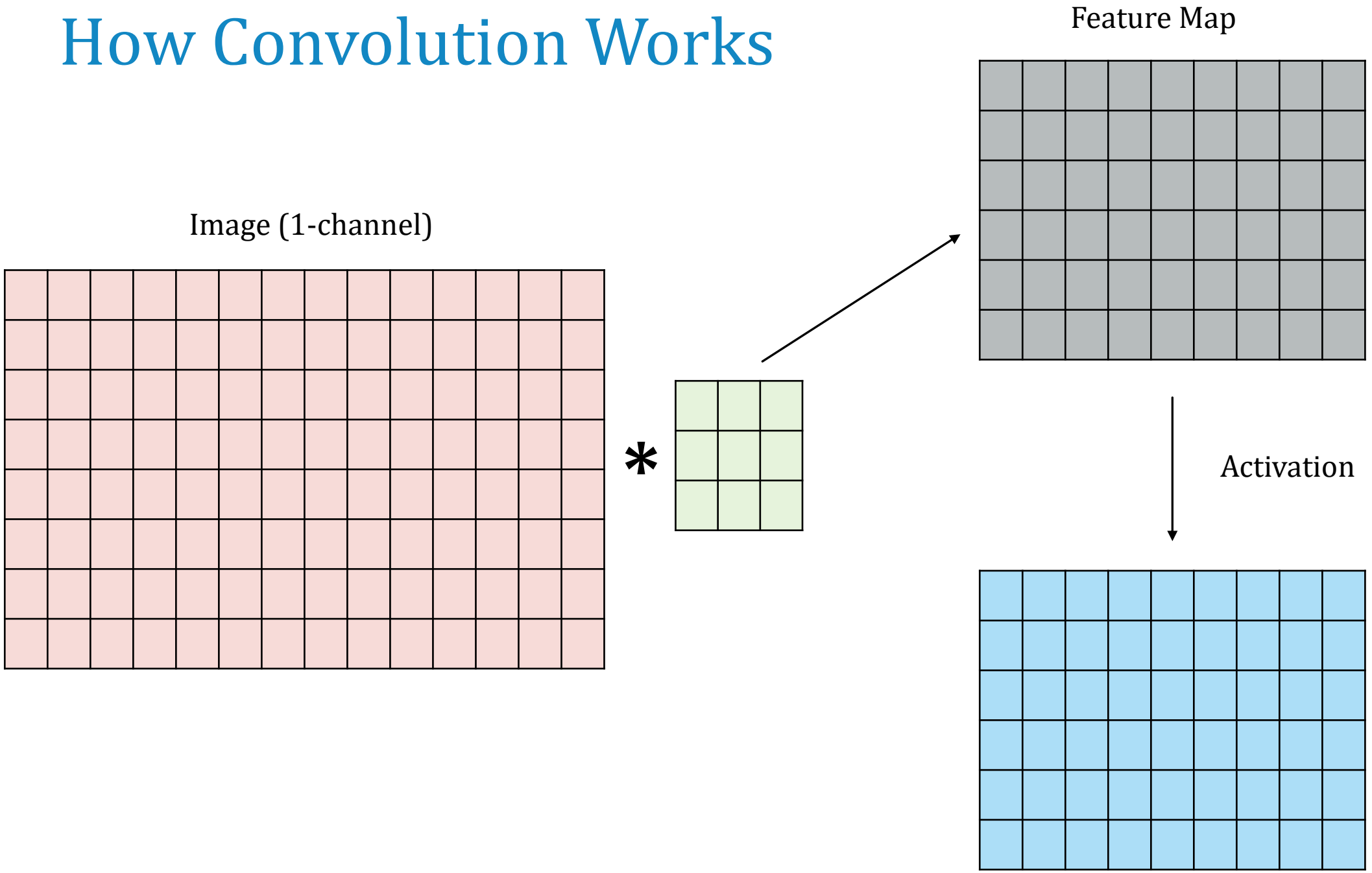
Image (1-channel)



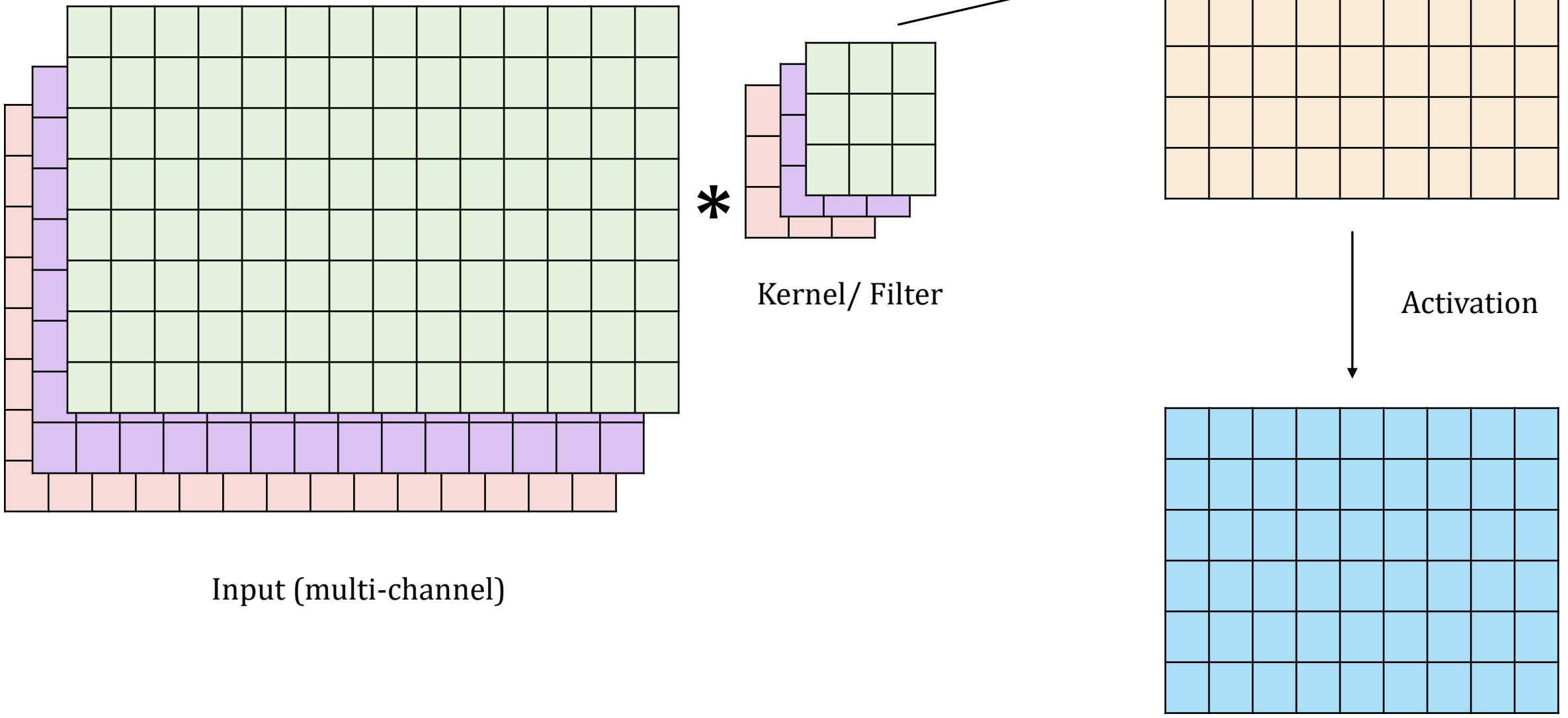
Feature Map



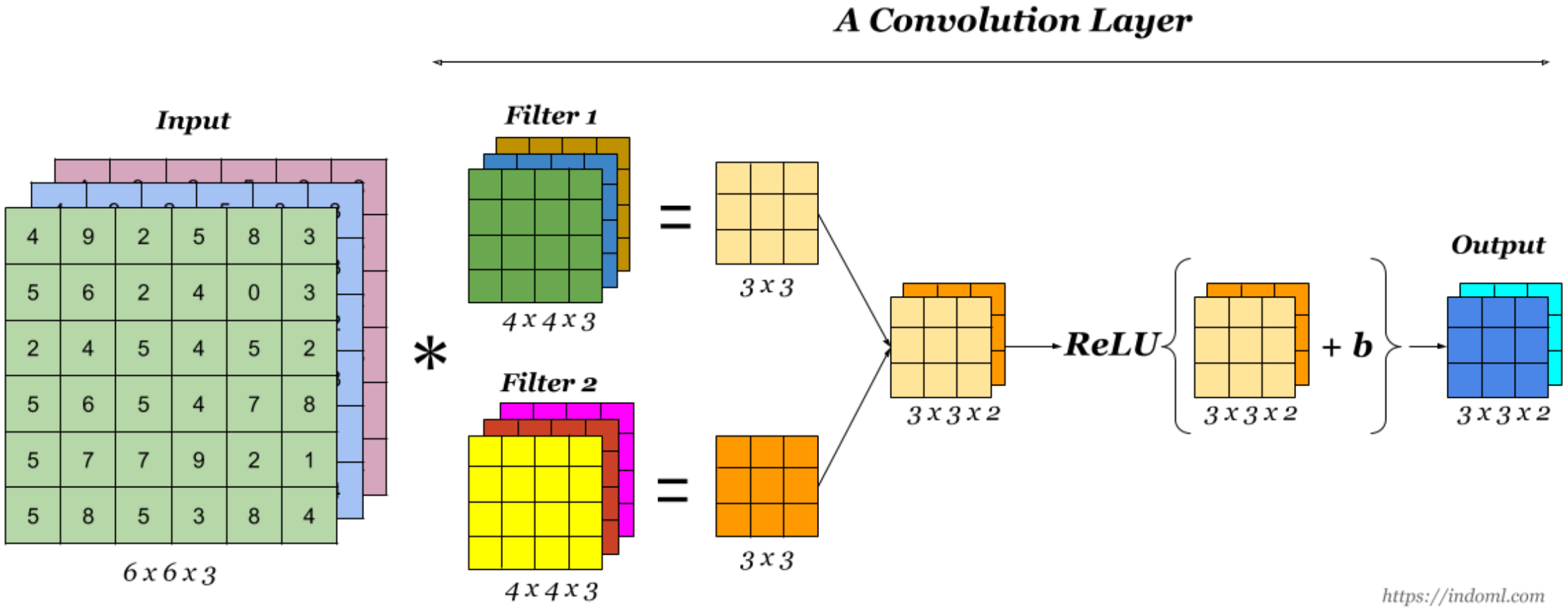
How Convolution Works



How Convolution Works

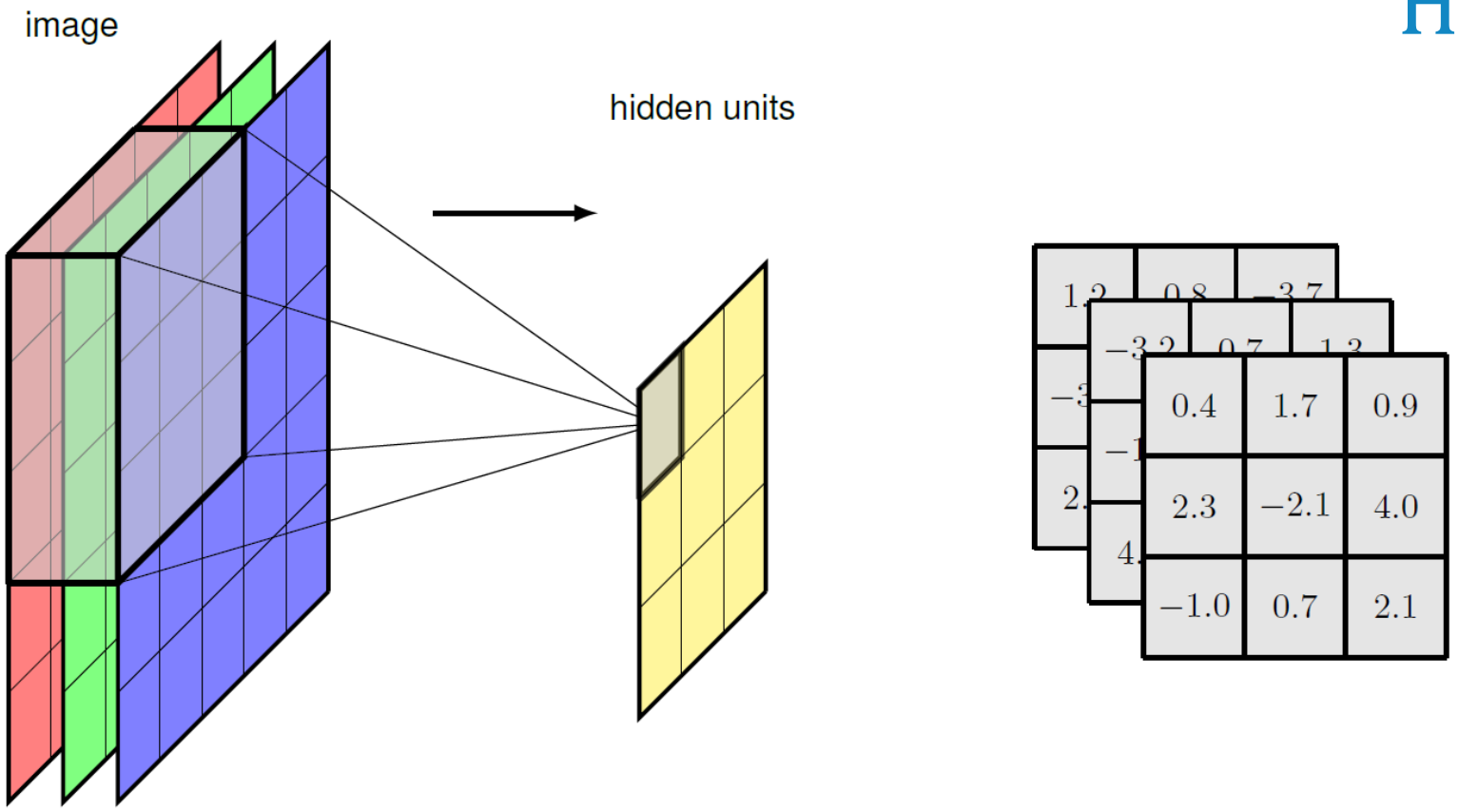


How Convolution Works



https://towardsdatascience.com/beginners-guide-to-understanding-convolutional-neural-networks-ae9ed58bb17d

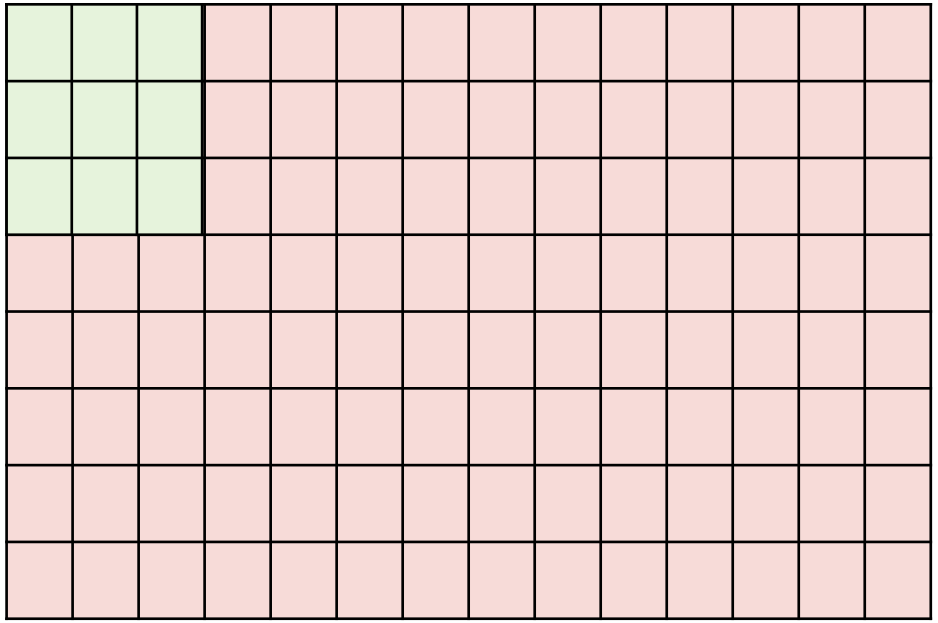
How Convolution Works

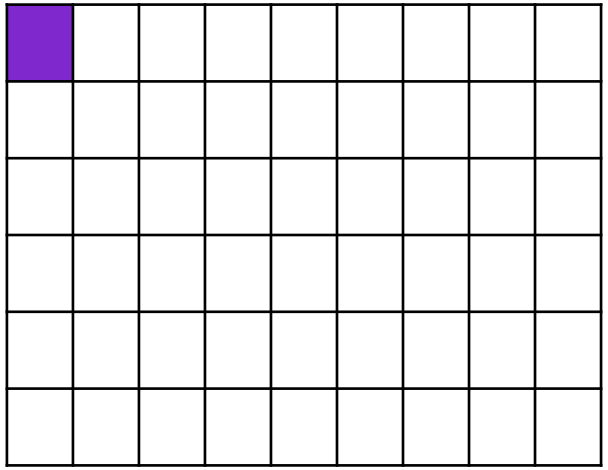


(a) (b)

(a) Illustration of a multi-dimensional filter that takes input from across the R, G, and B channels. (b) The kernel here has 27 weights (plus a bias parameter not shown) and can be visualized as a $3 \times 3 \times 3$ tensor.

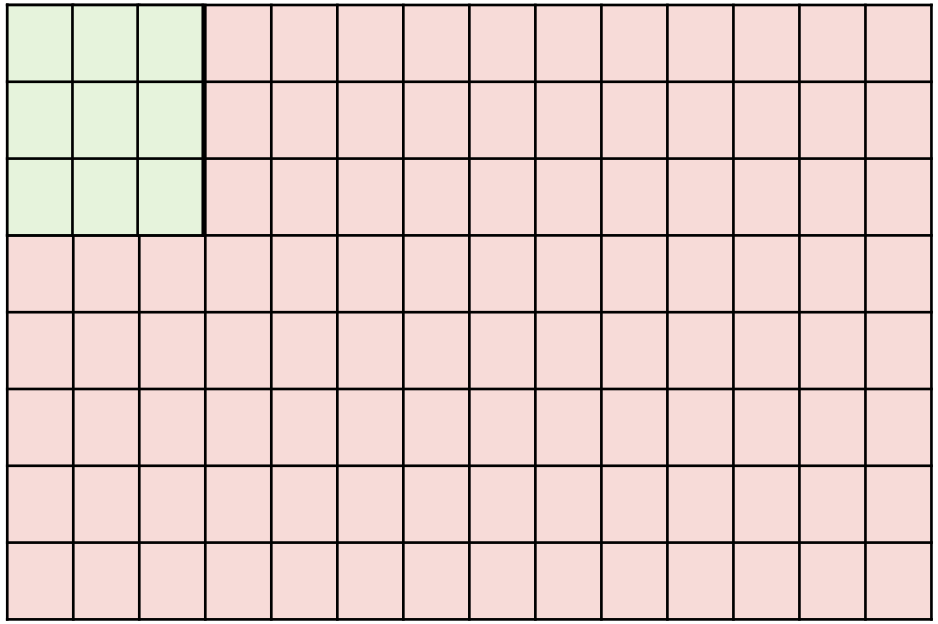
Properties of CNN: Sparse Connectivity

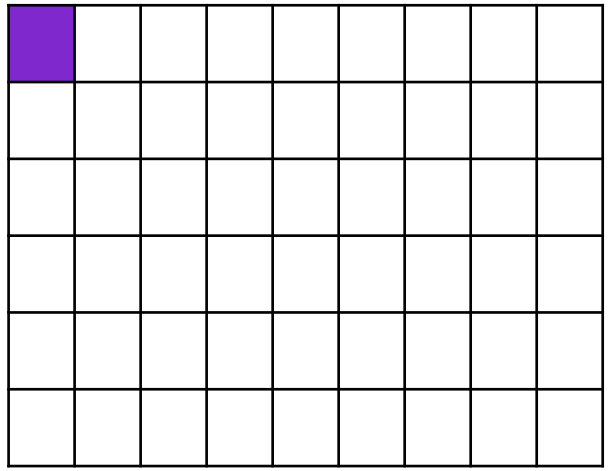




A pixel in the feature map is affected by only a set of pixels from where the feature map is generated (input pixels). It is not affected by all input pixels.

Properties of CNN: Sparse Connectivity



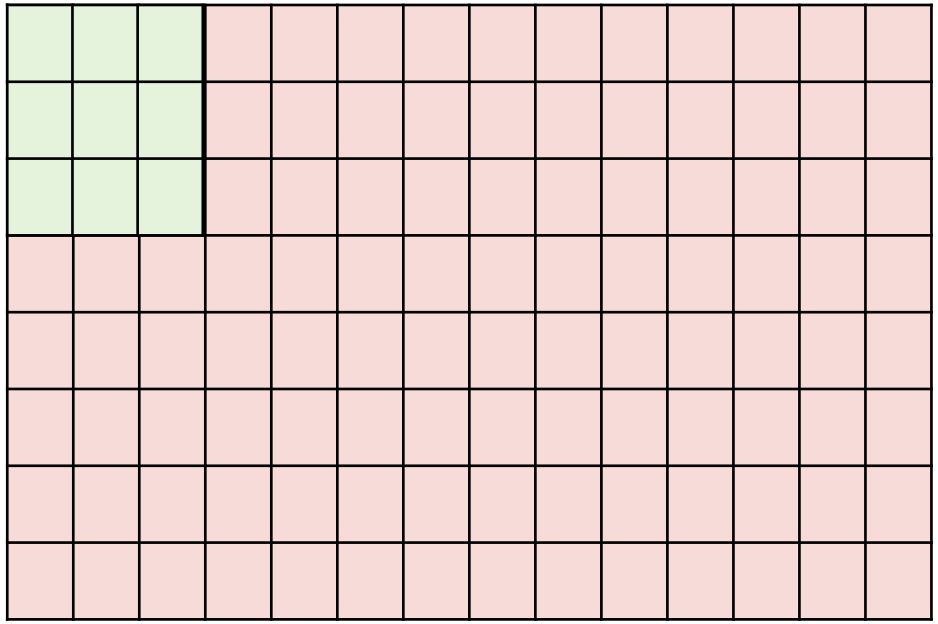


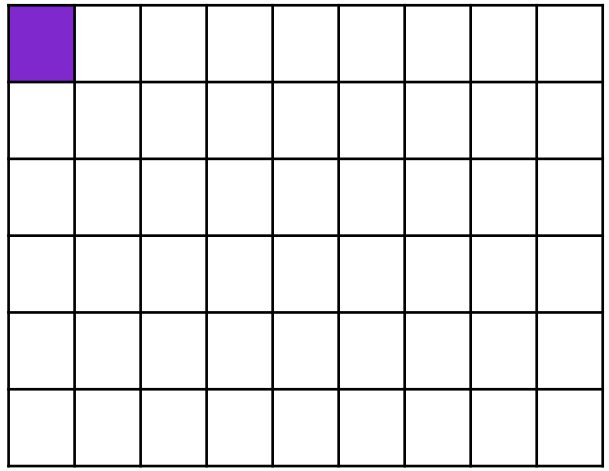
A pixel in the feature map is affected by only a set of pixels from where the feature map is generated (input pixels). It is not affected by all input pixels.

Global context is not used for local characteristics (such as edge, corner, etc.)

Less computing, less storage

Properties of CNN: Sparse Connectivity



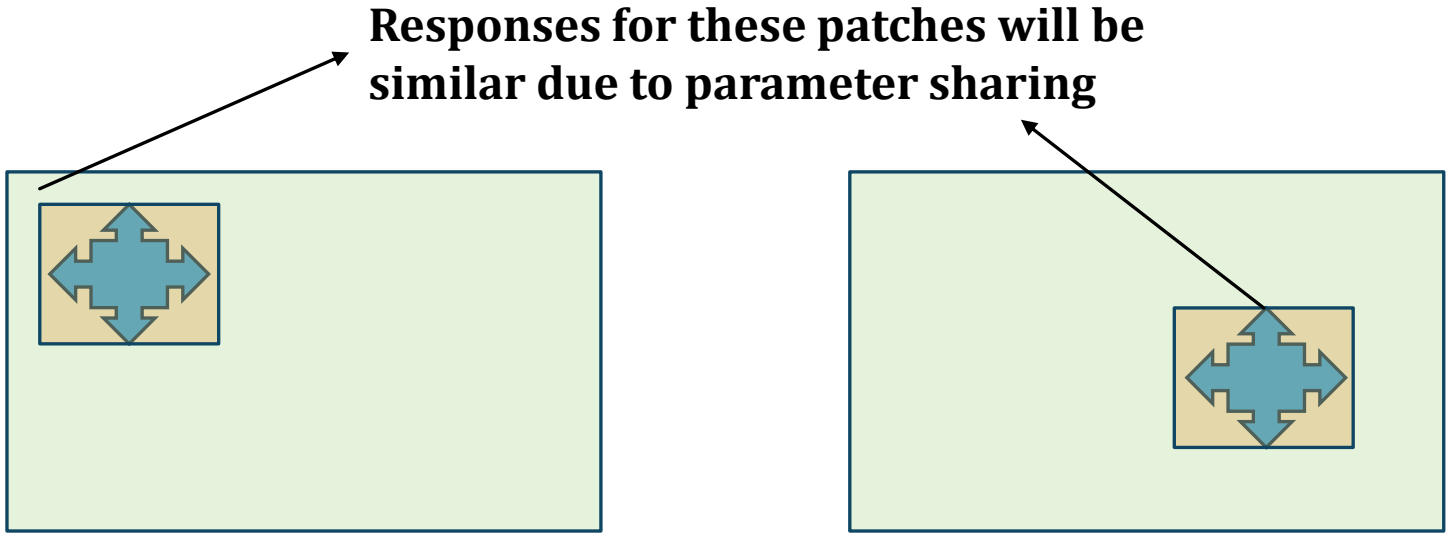


The same kernel (same set of parameters) are applied to all the patches

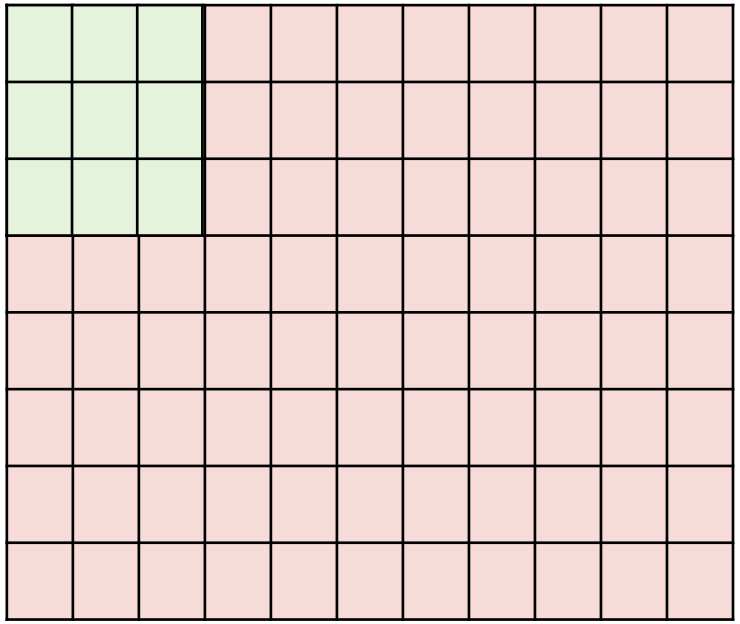
Less parameters, less storage

Properties of CNN: Translation Equivariance

- If an object is translated anywhere in an image
  - Our model will respond to same image patch regardless of its position
  - Translation invariance/ equivariance



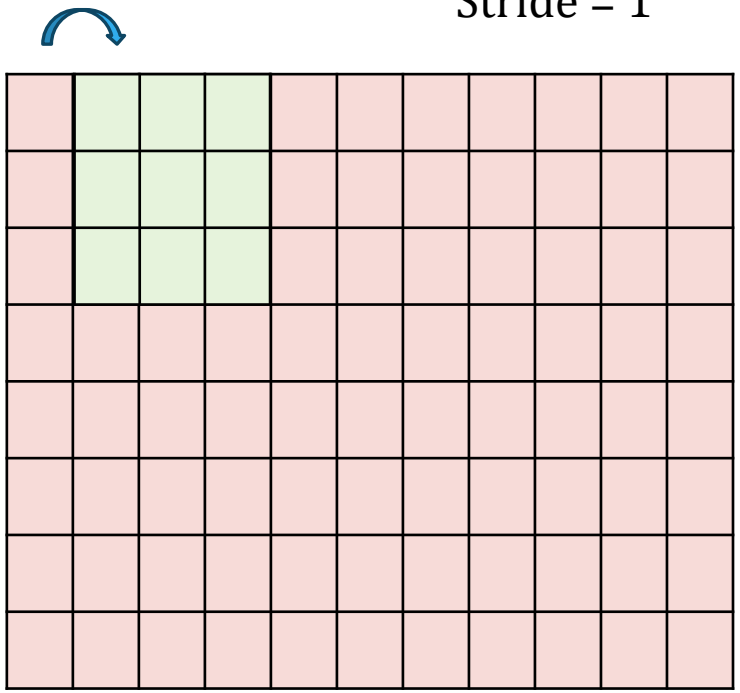
Stride



Stride: Amount of movement (row/ column traversed) of the kernel per slide

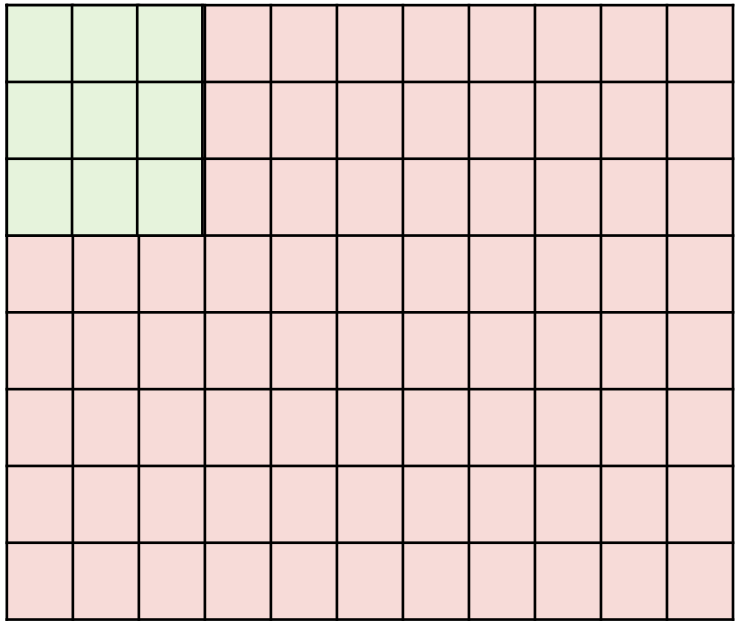
Stride

Stride = 1



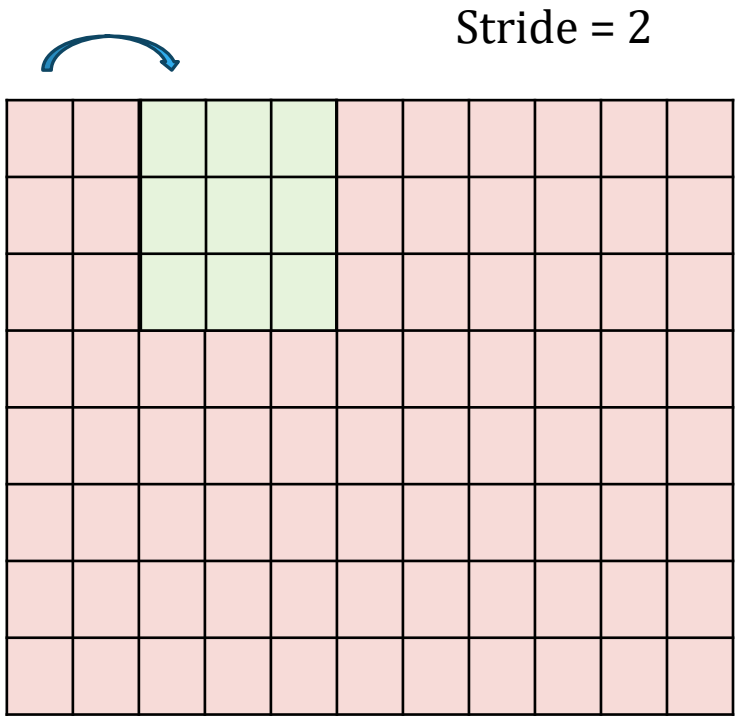
Stride: Amount of movement (row/ column traversed) of the kernel per slide

Stride



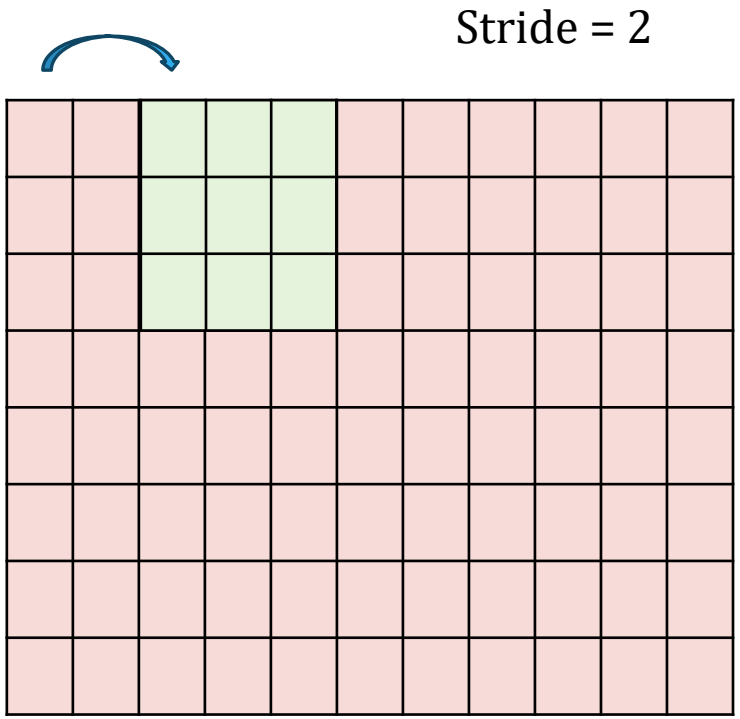
Stride: Amount of movement (row/ column traversed) of the kernel per slide

Stride



Stride: Amount of movement (row/ column traversed) of the kernel per slide

Stride

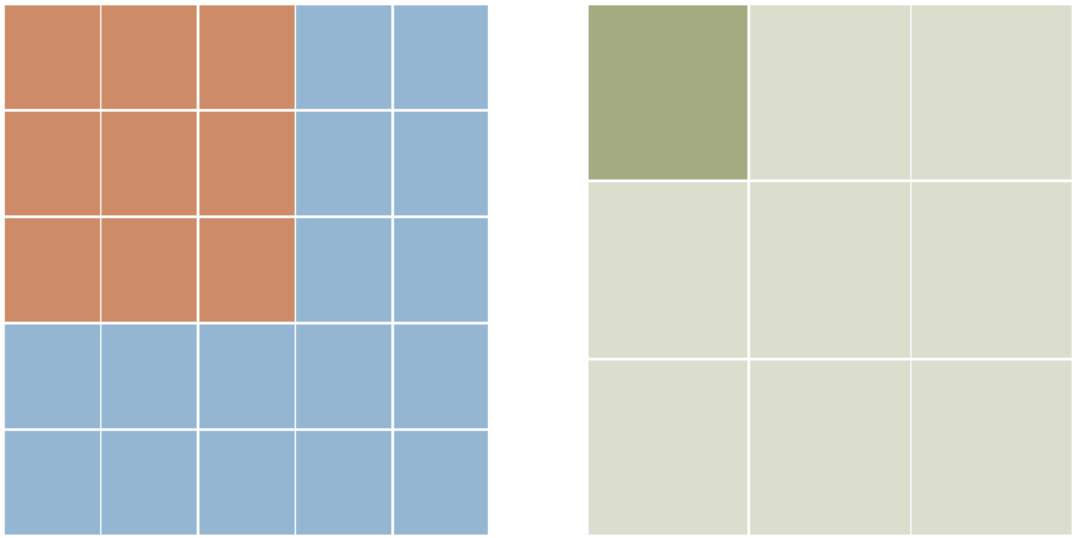


Stride: Amount of movement (row/ column traversed) of the kernel per slide

The more the stride, the smaller the feature map

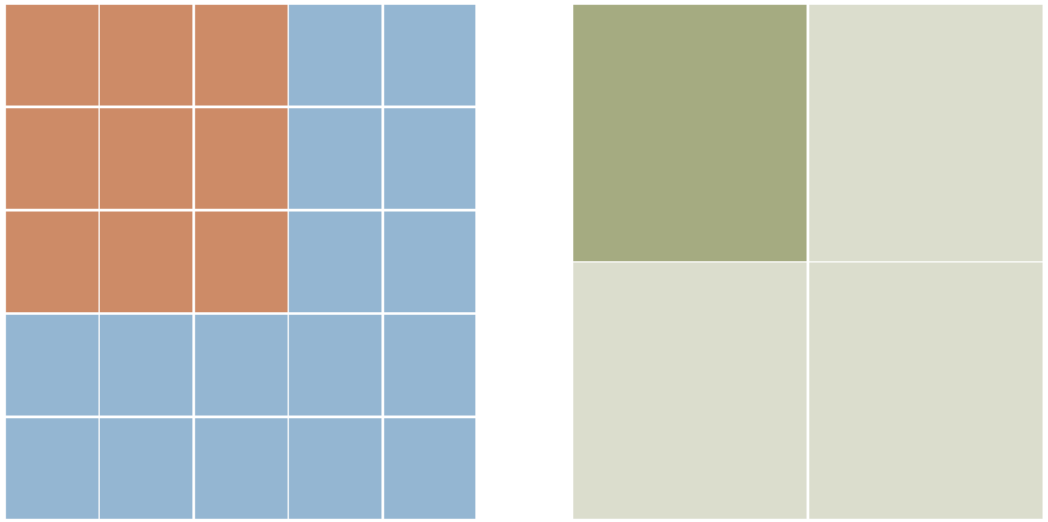
Stride

Type: conv - Stride: 1 Padding: 0



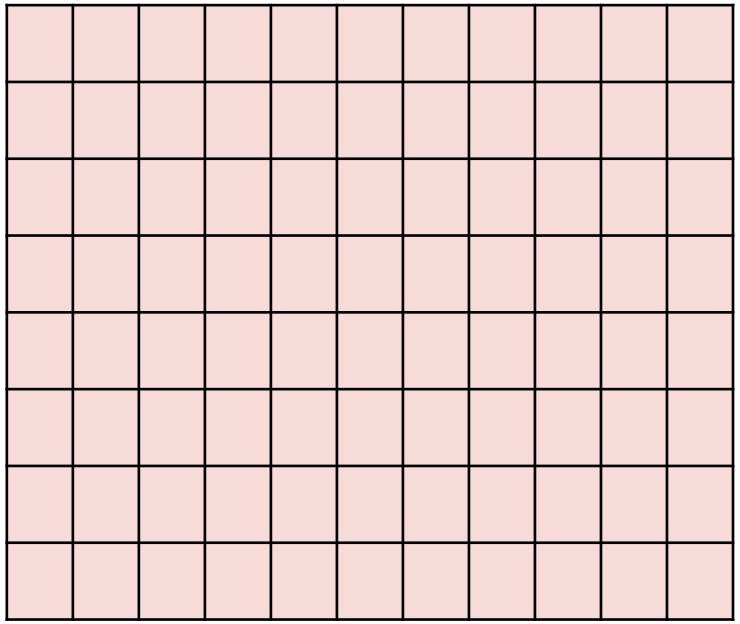
Input Output

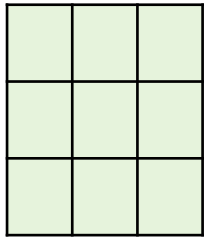
Type: conv - Stride: 2 Padding: 0

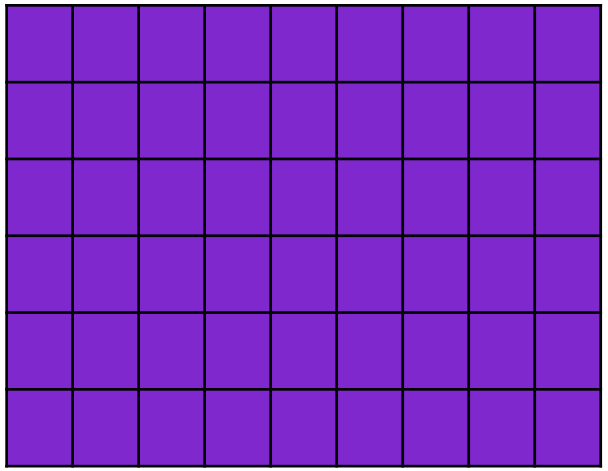


Input Output

Padding



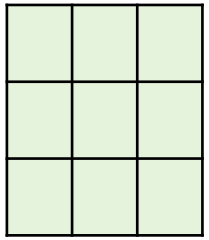




If we use the original input image, the feature map size will be smaller than the input size

Padding

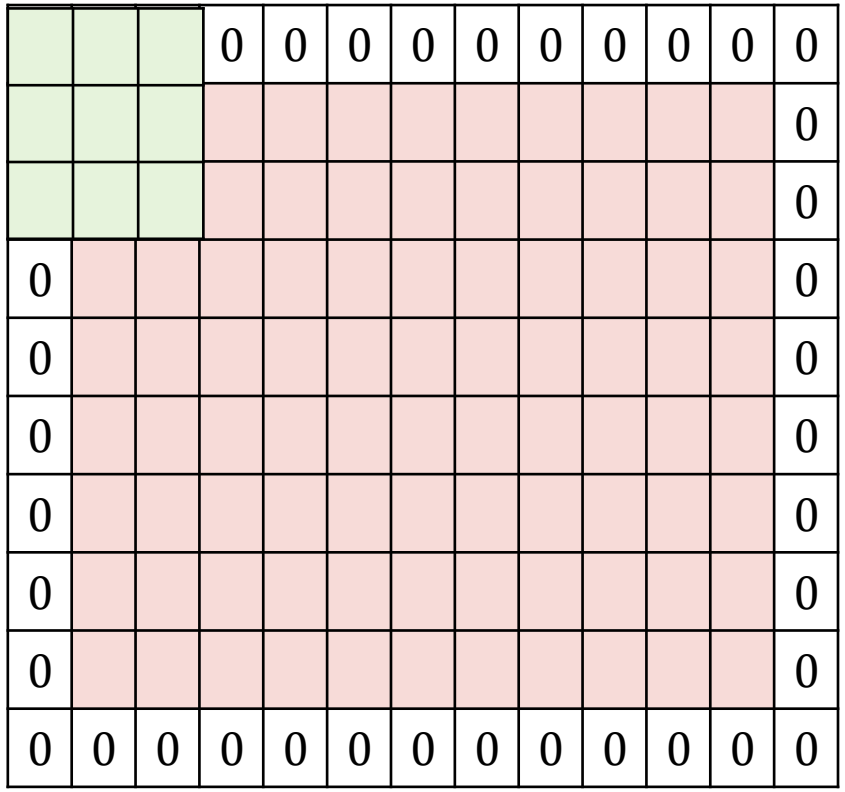
0	0	0	0	0	0	0	0	0	0	0	0	0
0												0
0												0
0												0
0												0
0												0
0												0
0												0
0												0
0	0	0	0	0	0	0	0	0	0	0	0	0



If we use the original input image, the feature map size will be smaller than the input size

To compensate for this, we add appropriate number of rows/ columns in the image

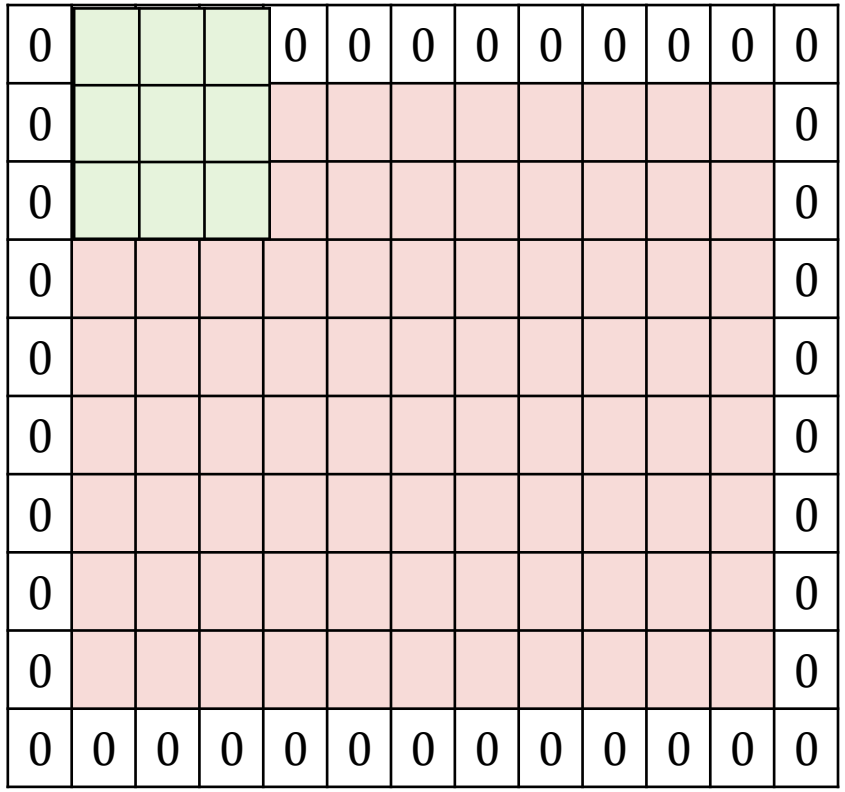
Padding



If we use the original input image, the feature map size will be smaller than the input size

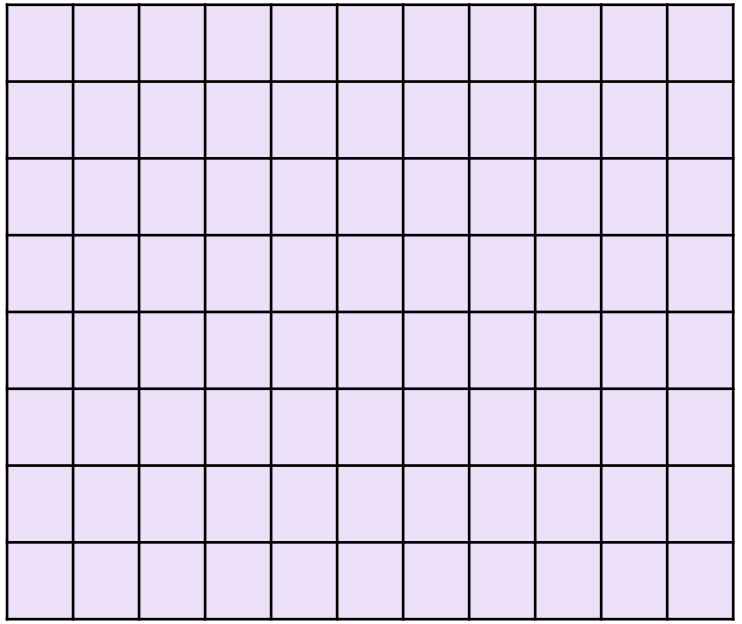
To compensate for this, we add appropriate number of rows/ columns in the image

Padding



Padding

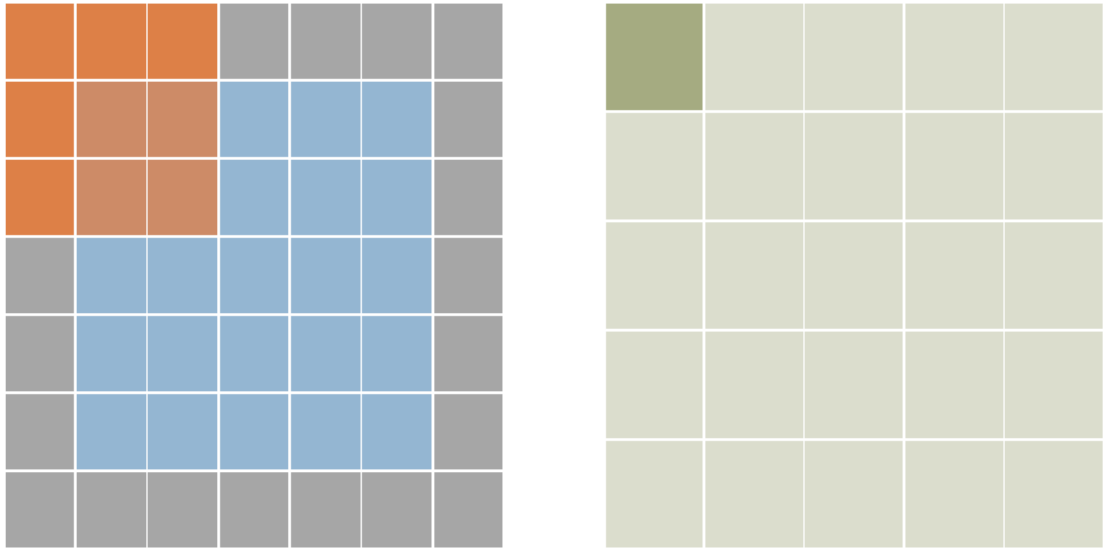
0	0				0	0	0	0	0	0	0	0
0												0
0												0
0												0
0												0
0												0
0												0
0												0
0												0
0	0	0	0	0	0	0	0	0	0	0	0	0



Feature Map

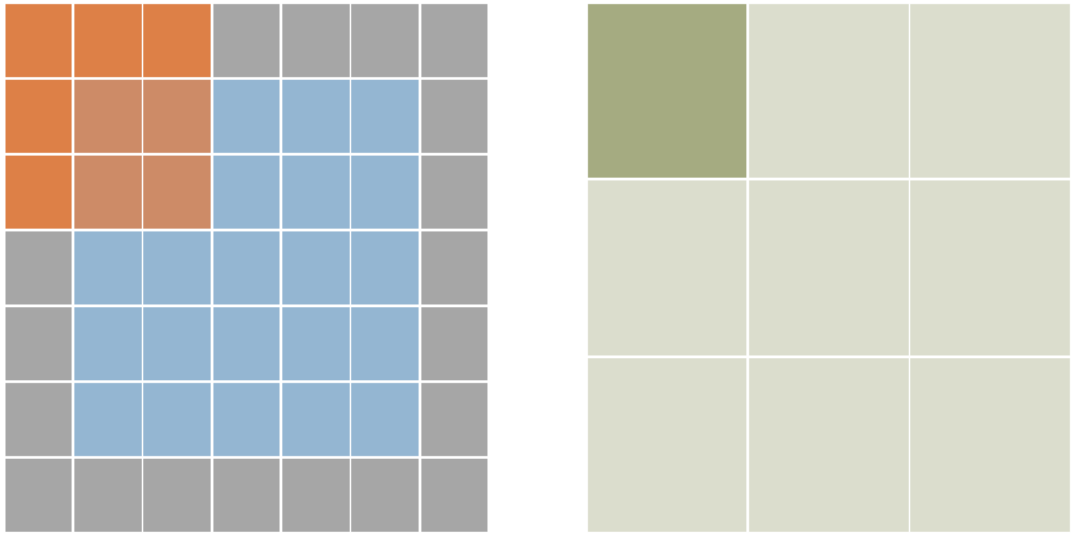
Padding

Type: conv - Stride: 1 Padding: 1



Input Output

Type: conv - Stride: 2 Padding: 1



Input Output

Dimension of the Output

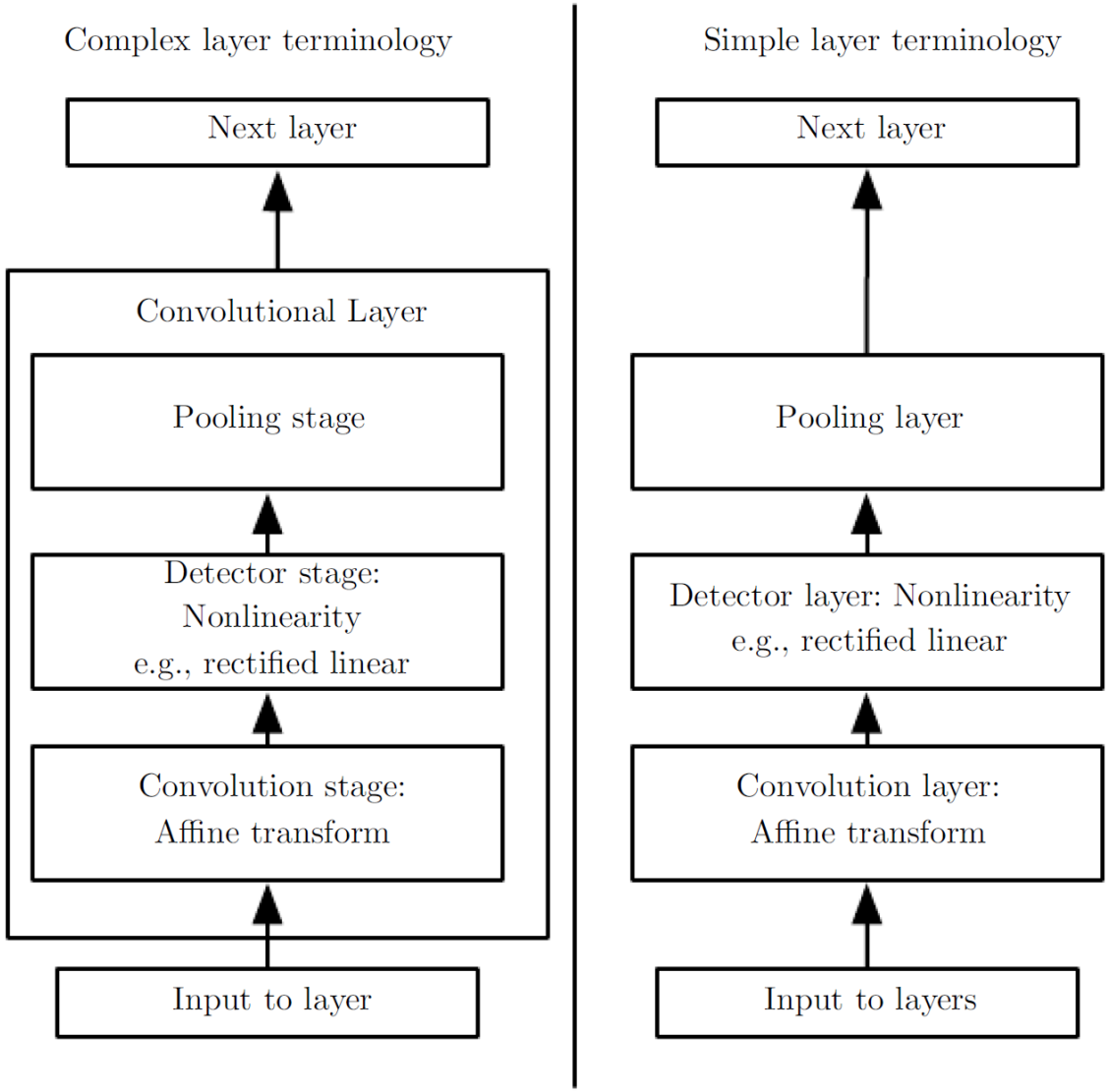
- Let
  - The input size be $m \times n$
  - Kernel size be $k \times k$
  - Padding size: p rows and p columns

- If output feature map size is $m_f \times n_f$, then

Dimension of the Output

- Let
  - The input size be $m \times n$
  - Kernel size be $k \times k$
  - Padding size: p rows and p columns
  - Stride: s
- If output feature map size is $m_f \times n_f$, then

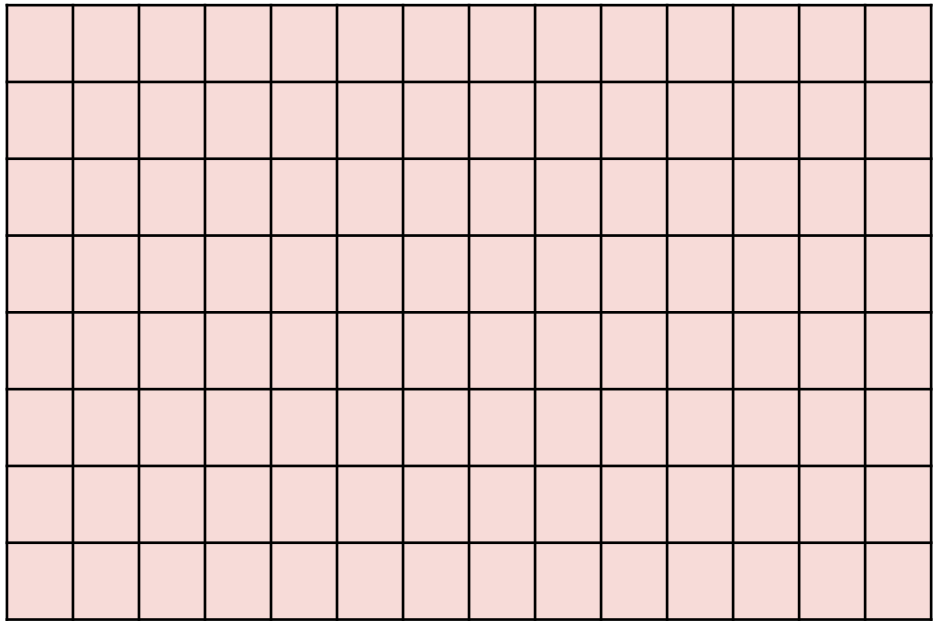
$$m_f = \left\lfloor \frac{m + 2p - k}{s} + 1 \right\rfloor \qquad n_f = \left\lfloor \frac{n + 2p - k}{s} + 1 \right\rfloor$$


Complex layer terminology
Next layer
Convolutional Layer
Pooling stage
Detector stage:
Nonlinearity
e.g., rectified linear
Convolution stage:
Affine transform
Input to layer
Simple layer terminology
Next layer
Pooling layer
Detector layer: Nonlinearity
e.g., rectified linear
Convolution layer:
Affine transform
Input to layers

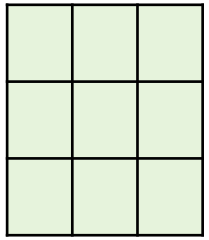
The CNN

Receptive Field

Image (1-channel)

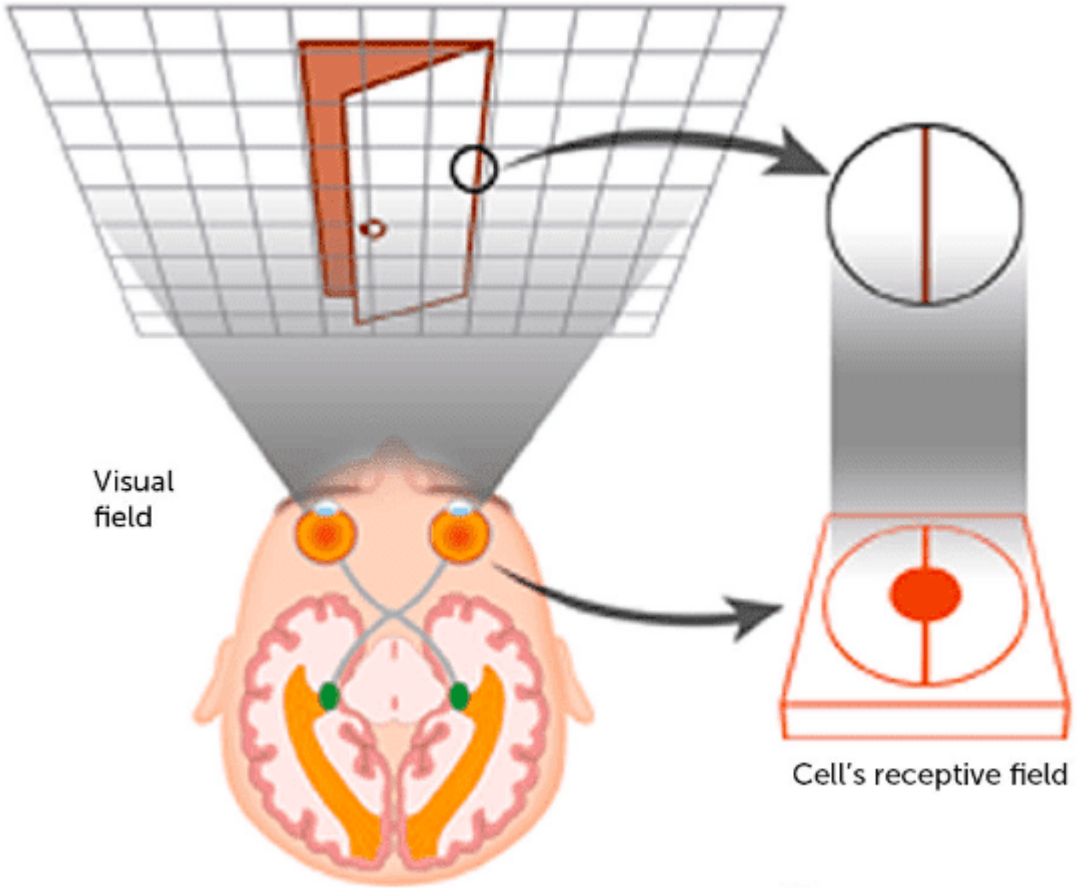


Kernel/ filter



Receptive Field: The portion of the sensory space that can elicit neuronal responses, when stimulated

Receptive Field

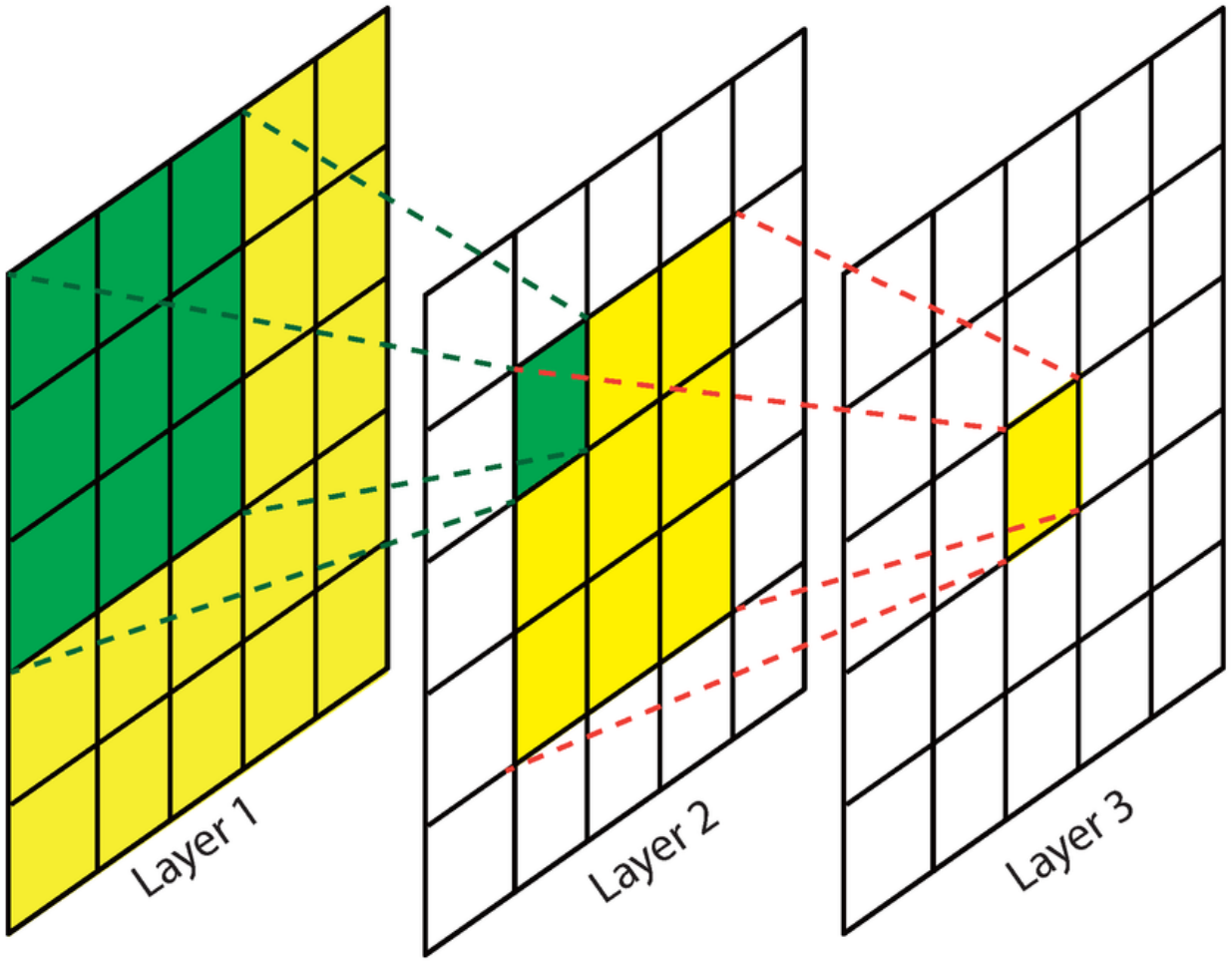


An idea borrowed from neuroscience.

The entire area (the grid in the figure) an eye can see is called the field of view. The human visual system consists of millions of neurons, where each one captures different information. We define the neuron's receptive field as the patch of the total field of view. In other words, what information a single neuron has access to. This is in simple terms the biological cell's receptive field.

https://theaisummer.com/static/ab62e5bb8de79d33ab49cec298230c46/86389/Visual-human-system.png

Receptive Field



The deeper we go in the network, the larger the receptive field (relative to the input) to which each hidden node is sensitive

https://www.researchgate.net/figure/The-receptive-field-of-each-convolution-layer-with-a-3-3-kernel-The-green-area-marks_fig4_316950618

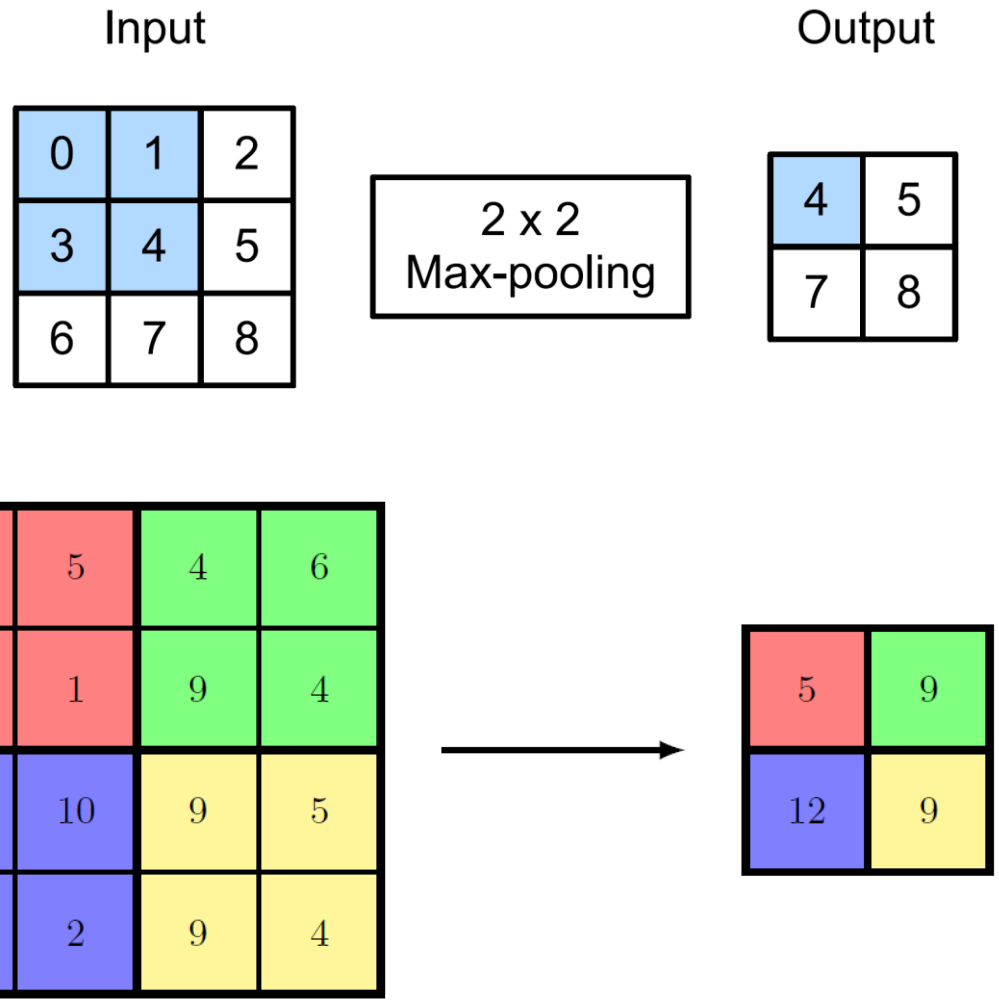
Pooling

- The deeper we go in the network, the larger the receptive field (relative to the input) to which each hidden node is sensitive.
  - Reducing spatial resolution accelerates this process, since the convolution kernels cover a larger effective area.

- Translation in input may change the feature map significantly
  - We often want translation invariant decision making because objects may occur anywhere in images

Pooling

- Pooling provides a summary of statistics
- Pooling enhances translational invariance
  - Makes invariant to small translations
- Types of pooling
  - Average pooling
  - Max pooling (more popular)



Pooling & Translational Invariance

Input

Feature Map & Activation

12	**37**	**11**	**72**
9	**83**	**92**	**11**
5	**17**	**11**	**13**
21	**16**	**15**	**4**

Pooling & Translational Invariance

Input

Feature Map & Activation

12	**37**	**11**	**72**
9	**83**	**92**	**11**
5	**17**	**11**	**13**
21	**16**	**15**	**4**

2 × 2 Max pooling

83	**92**	**92**
83	**92**	**92**
21	**17**	**15**

Pooling & Translational Invariance

Input

Feature Map & Activation

12	37	11	72
9	83	92	11
5	17	11	13
21	16	15	4

2 × 2 Max pooling

83	92	92
83	92	92
21	17	15

Translated Input

7	12	37	11
4	9	83	92
11	5	17	11
2	21	16	15

2 × 2 Max pooling

12	83	92
11	83	92
21	21	17

Pooling & Translational Invariance

Input

Feature Map & Activation

12	37	11	72
9	83	92	11
5	17	11	13
21	16	15	4

2 × 2 Max pooling

83	92	92
83	92	92
21	17	15

Translated Input

7	12	37	11
4	9	83	92
11	5	17	11
2	21	16	15

2 × 2 Max pooling

12	83	92
11	83	92
21	21	17

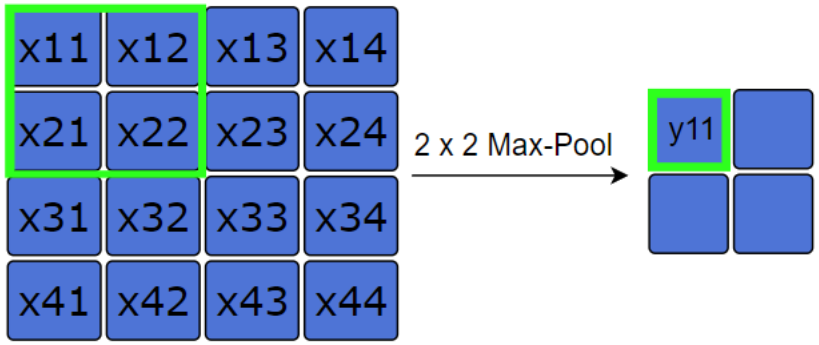
All values changed in same positions

Only nearly half of the values changed in same positions

Backpropagation through Max Pooling

- Can we backpropagate through max pool layer?

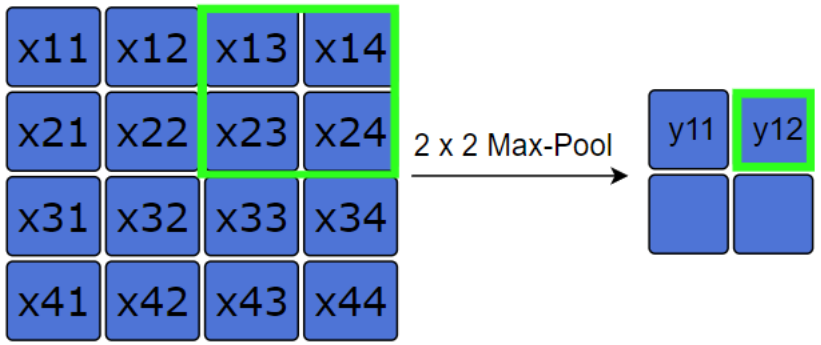
Backpropagation through Max Pooling



$y11 = \max(x11, x12, x21, x22)$

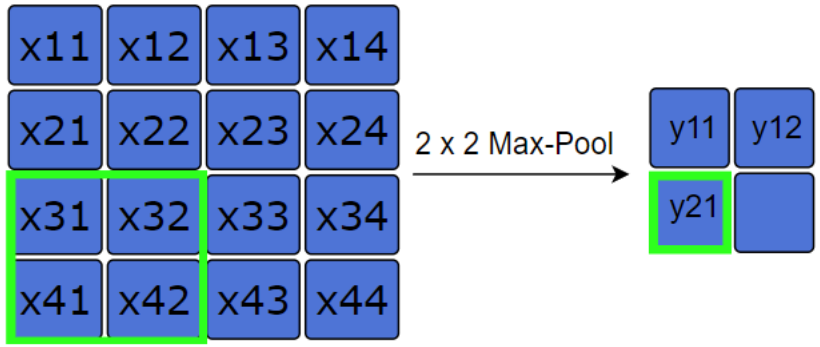
https://www.educative.io/answers/how-to-backpropagate-through-max-pooling-layers

Backpropagation through Max Pooling



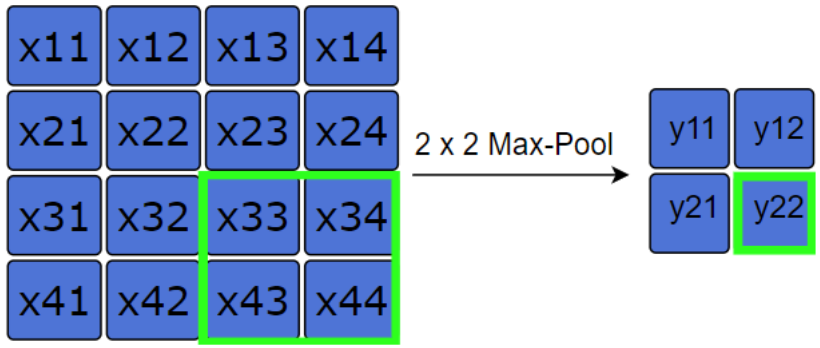
y12 = max (x13, x14, x23, x24)

Backpropagation through Max Pooling



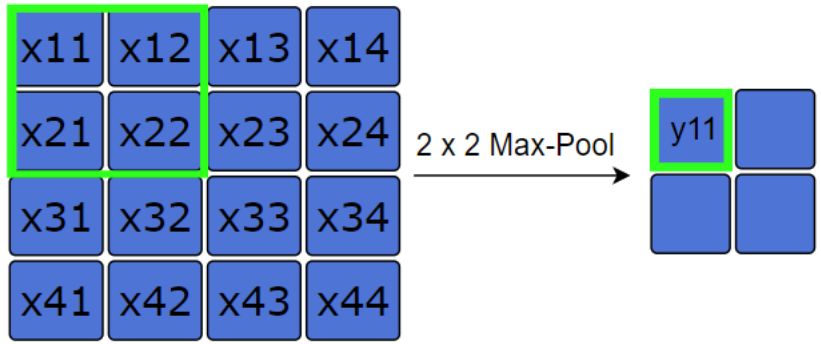
y21 = max (x31, x32, x41, x42)

Backpropagation through Max Pooling



y22 = max (x33, x34, x43, x44)

Backpropagation through Max Pooling



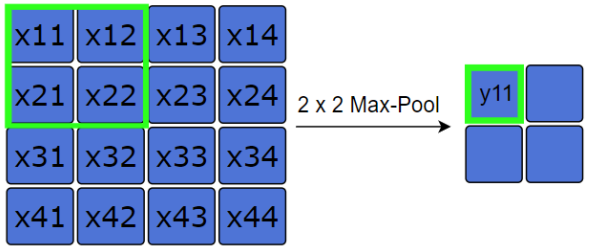
y11 = max (x11, x12, x21, x22)

If E is the loss, we need to find

$$\begin{bmatrix} \frac{\partial E}{\partial x_{11}} & \frac{\partial E}{\partial x_{12}} & \frac{\partial E}{\partial x_{13}} & \frac{\partial E}{\partial x_{14}} \\ \frac{\partial E}{\partial x_{21}} & \frac{\partial E}{\partial x_{22}} & \frac{\partial E}{\partial x_{23}} & \frac{\partial E}{\partial x_{24}} \\ \frac{\partial E}{\partial x_{31}} & \frac{\partial E}{\partial x_{32}} & \frac{\partial E}{\partial x_{33}} & \frac{\partial E}{\partial x_{34}} \\ \frac{\partial E}{\partial x_{41}} & \frac{\partial E}{\partial x_{42}} & \frac{\partial E}{\partial x_{43}} & \frac{\partial E}{\partial x_{44}} \end{bmatrix}$$

https://www.educative.io/answers/how-to-backpropagate-through-max-pooling-layers

Backpropagation through Max Pooling



y11 = max (x11, x12, x21, x22)

If E is the loss, we need to find

$$\begin{bmatrix} \frac{\partial E}{\partial x_{11}} & \frac{\partial E}{\partial x_{12}} & \frac{\partial E}{\partial x_{13}} & \frac{\partial E}{\partial x_{14}} \\ \frac{\partial E}{\partial x_{21}} & \frac{\partial E}{\partial x_{22}} & \frac{\partial E}{\partial x_{23}} & \frac{\partial E}{\partial x_{24}} \\ \frac{\partial E}{\partial x_{31}} & \frac{\partial E}{\partial x_{32}} & \frac{\partial E}{\partial x_{33}} & \frac{\partial E}{\partial x_{34}} \\ \frac{\partial E}{\partial x_{41}} & \frac{\partial E}{\partial x_{42}} & \frac{\partial E}{\partial x_{43}} & \frac{\partial E}{\partial x_{44}} \end{bmatrix}$$

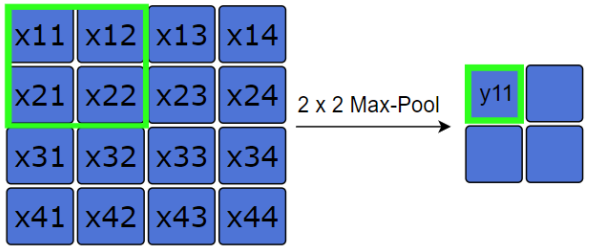
$$\frac{\partial E}{\partial x_{11}} = \frac{\partial E}{\partial y_{11}} \cdot \frac{\partial y_{11}}{\partial x_{11}} + \frac{\partial E}{\partial y_{12}} \cdot \frac{\partial y_{12}}{\partial x_{11}} + \frac{\partial E}{\partial y_{21}} \cdot \frac{\partial y_{21}}{\partial x_{11}} + \frac{\partial E}{\partial y_{22}} \cdot \frac{\partial y_{22}}{\partial x_{11}}$$

Since x_{11} affects only y_{11}, we have

$$\frac{\partial E}{\partial x_{11}} = \frac{\partial E}{\partial y_{11}} \cdot \frac{\partial y_{11}}{\partial x_{11}} = D_{11} \cdot \frac{\partial y_{11}}{\partial x_{11}}$$

https://www.educative.io/answers/how-to-backpropagate-through-max-pooling-layers

Backpropagation through Max Pooling



y11 = max (x11, x12, x21, x22)

If E is the loss, we need to find

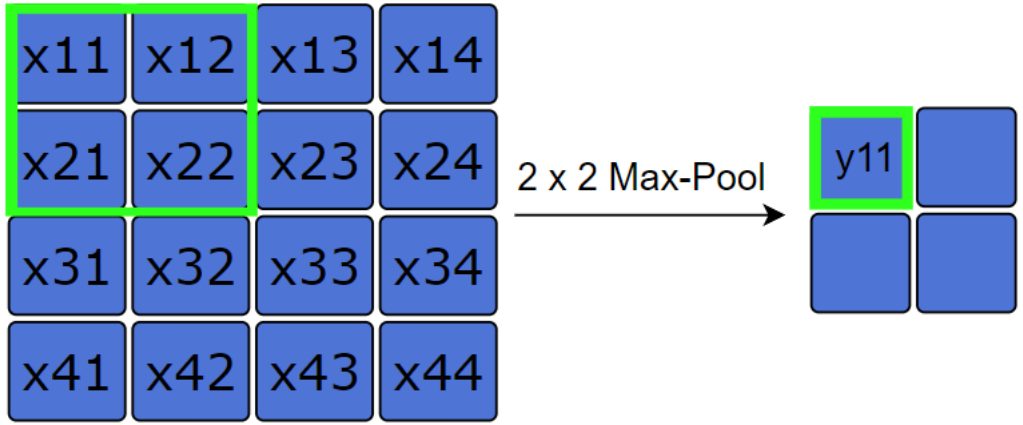
$$\begin{bmatrix} \frac{\partial E}{\partial x_{11}} & \frac{\partial E}{\partial x_{12}} & \frac{\partial E}{\partial x_{13}} & \frac{\partial E}{\partial x_{14}} \\ \frac{\partial E}{\partial x_{21}} & \frac{\partial E}{\partial x_{22}} & \frac{\partial E}{\partial x_{23}} & \frac{\partial E}{\partial x_{24}} \\ \frac{\partial E}{\partial x_{31}} & \frac{\partial E}{\partial x_{32}} & \frac{\partial E}{\partial x_{33}} & \frac{\partial E}{\partial x_{34}} \\ \frac{\partial E}{\partial x_{41}} & \frac{\partial E}{\partial x_{42}} & \frac{\partial E}{\partial x_{43}} & \frac{\partial E}{\partial x_{44}} \end{bmatrix}$$

$$\frac{\partial E}{\partial x_{11}} = \frac{\partial E}{\partial y_{11}} \cdot \frac{\partial y_{11}}{\partial x_{11}} = D_{11} \cdot \frac{\partial y_{11}}{\partial x_{11}}$$

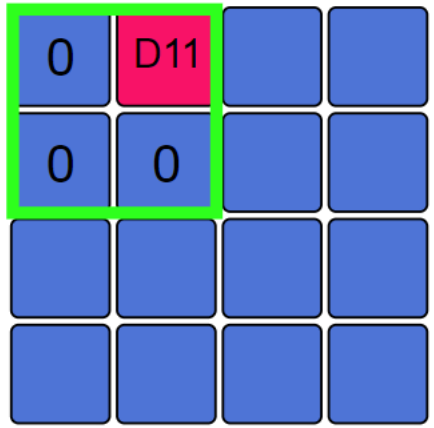
If x_{11} is the max element out of $x_{11}, x_{12}, x_{21}, x_{22}$
We will have $y_{11} = x_{11}$ and consequently $\frac{\partial y_{11}}{\partial x_{11}} = 1$ and $\frac{\partial E}{\partial x_{11}} = D_{11}$

Else, y_{11} will not be a function of x_{11} and consequently $\frac{\partial y_{11}}{\partial x_{11}} = 0$ and $\frac{\partial E}{\partial x_{11}} = 0$

https://www.educative.io/answers/how-to-backpropagate-through-max-pooling-layers

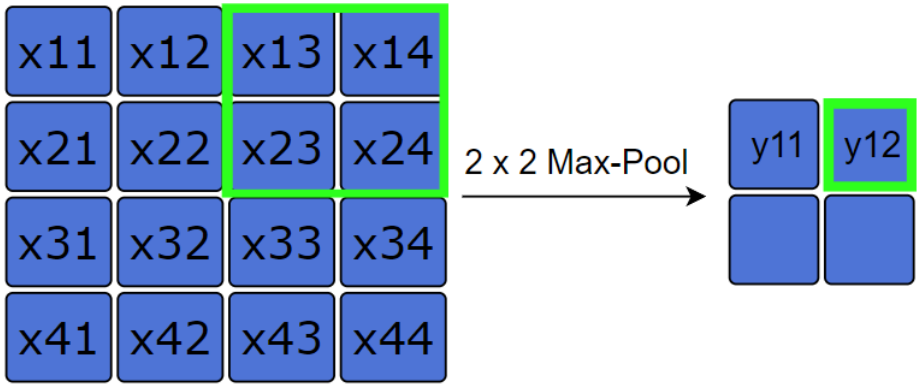


y11 = max (x11, x12, x21, x22) = x12

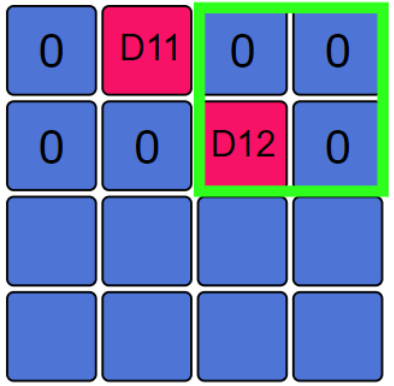


Backward Pass

Backpropagation through Max Pooling



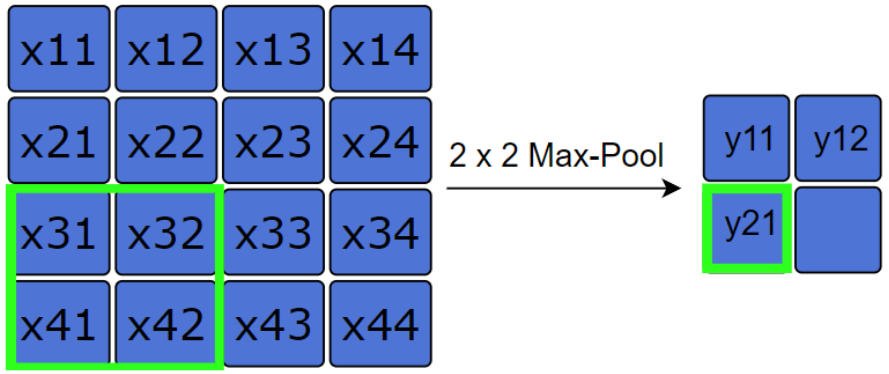
y12 = max (x13, x14, x23, x24) = x23



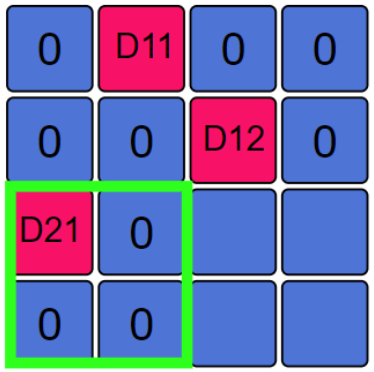
Backward Pass

Backpropagation through Max Pooling

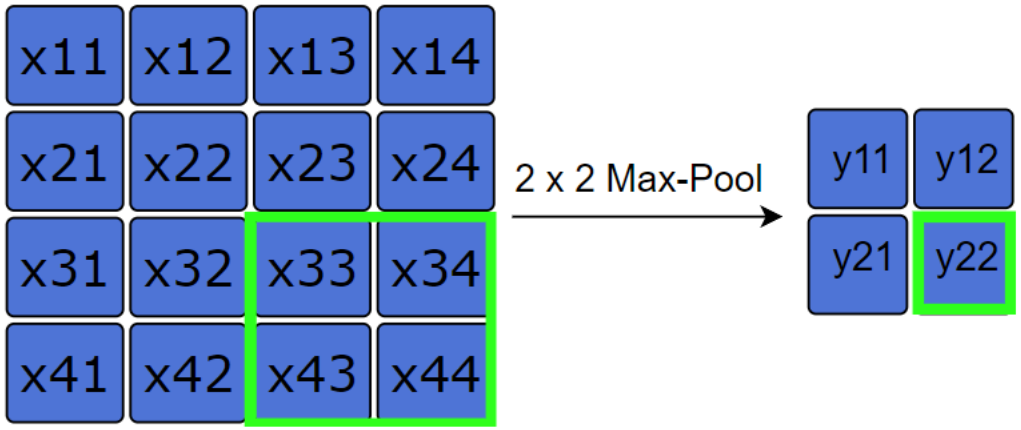
Backpropagation through Max Pooling



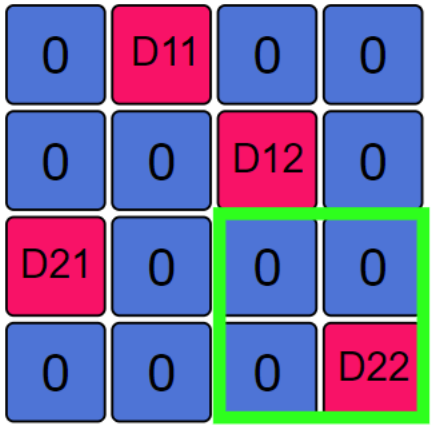
y21 = max (x31, x32, x41, x42) = x31



Backward Pass



y22 = max (x33, x34, x43, x43) = x44



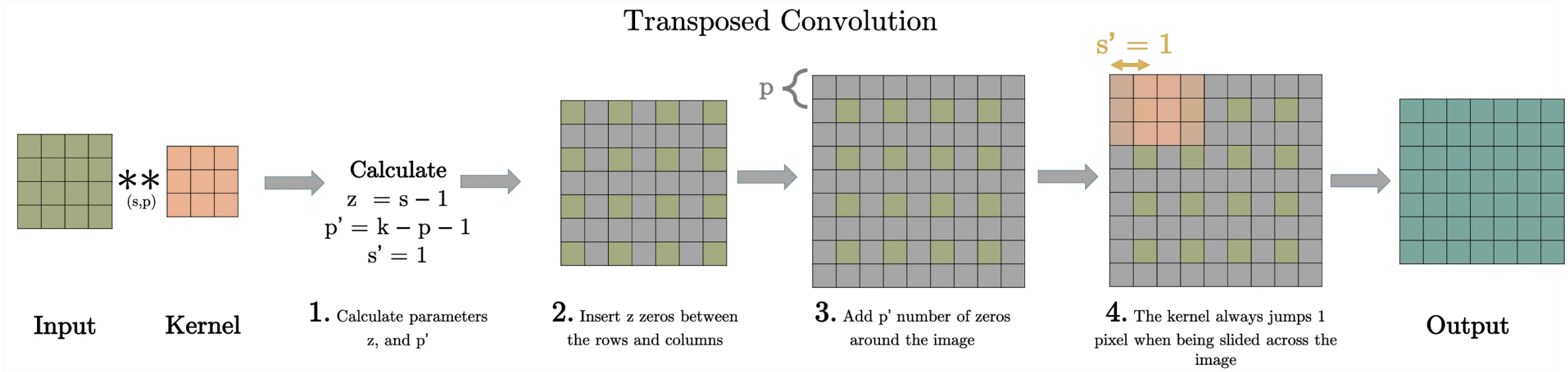
Backward Pass

Backpropagation through Max Pooling

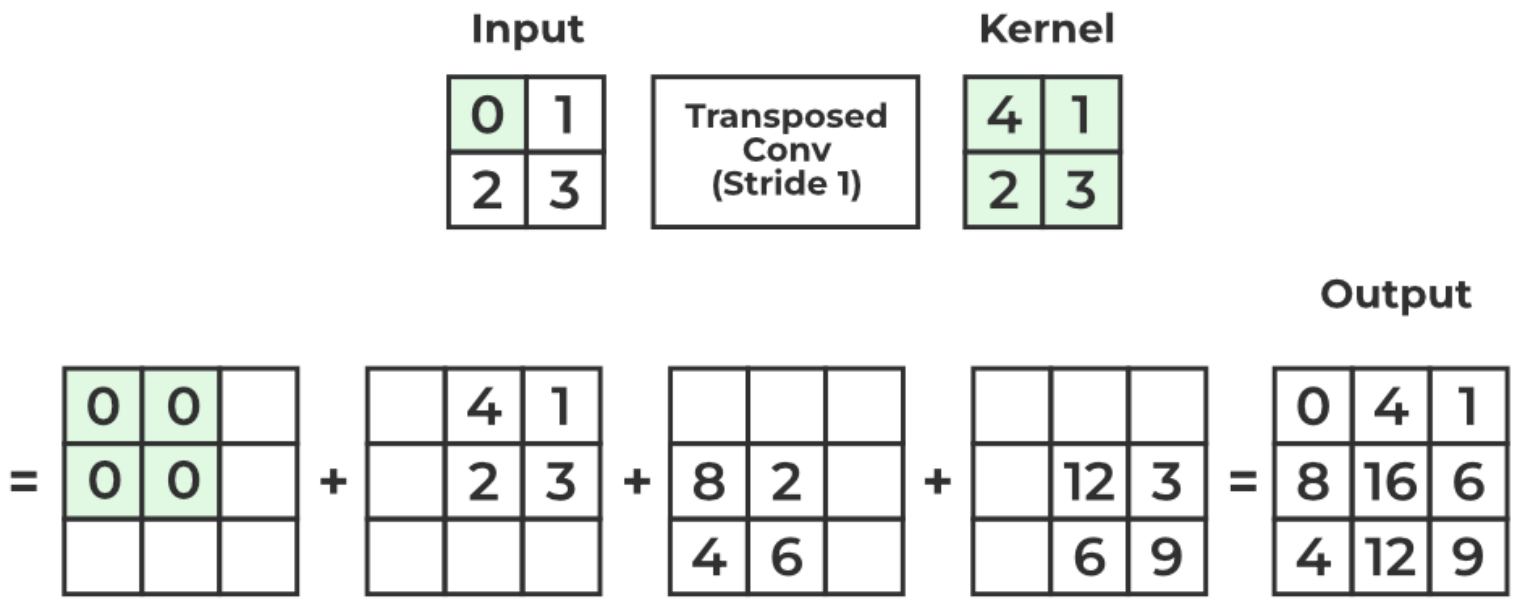
Upsampling

- Standard convolution downsamples the input (if padding is not done)
- Transposed convolution: used for upsampling of input

Transposed Convolution



Transposed Convolution



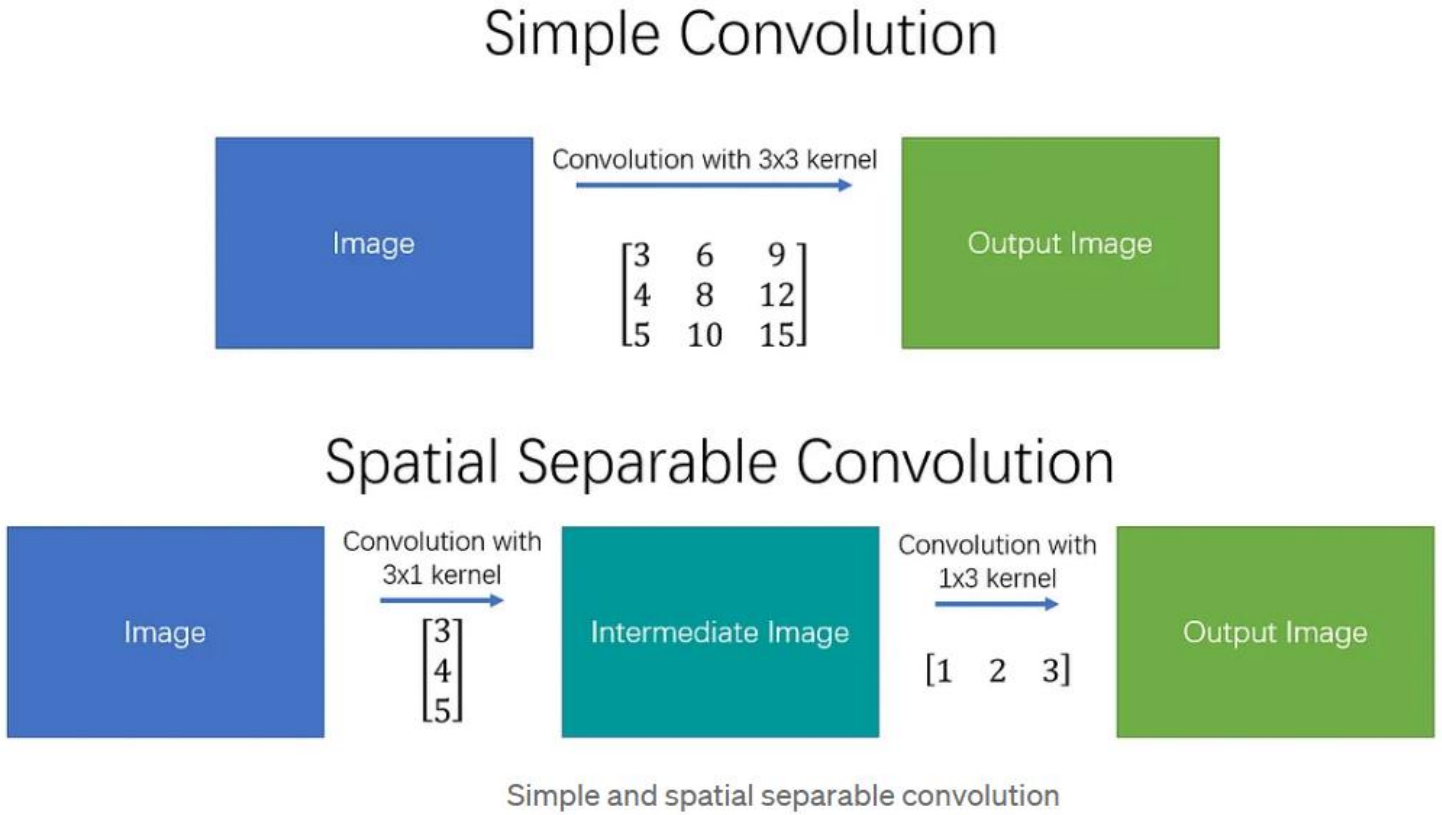
Reducing the Number of Computation: Spatial Separable Convolutions

$$\begin{bmatrix} 3 & 6 & 9 \\ 4 & 8 & 12 \\ 5 & 10 & 15 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

Separating a 3×3 kernel spatially

https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

Spatial Separable Convolutions

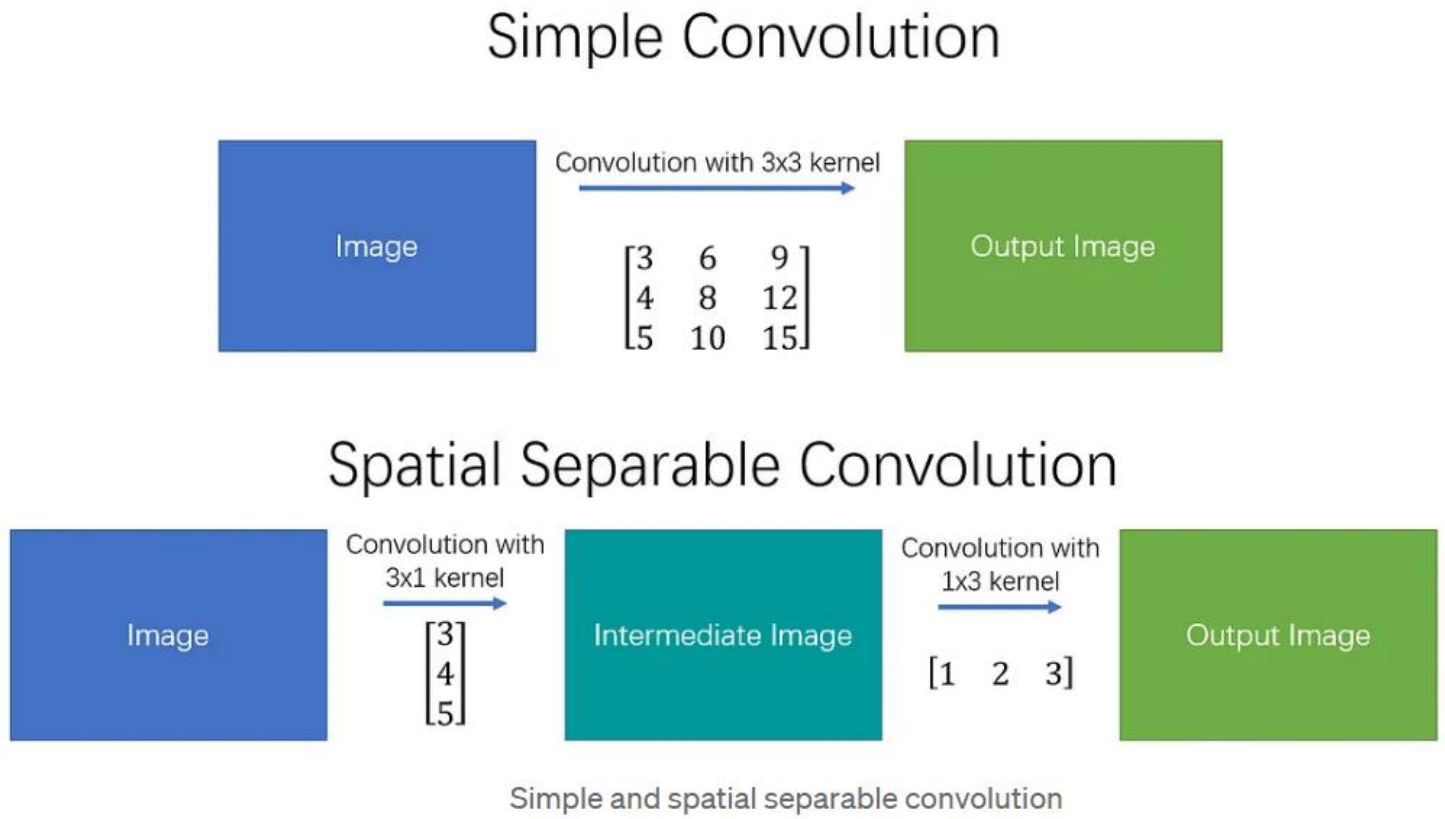


Simple and spatial separable convolution

https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

Spatial Separable Convolutions

How much reduction in computation?

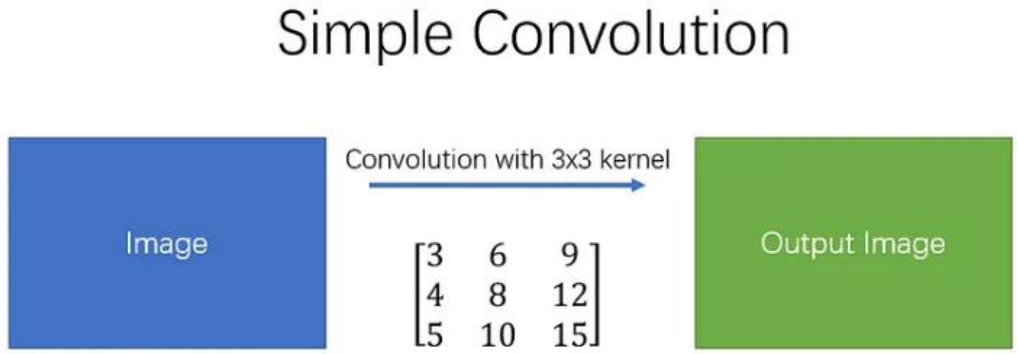


Simple and spatial separable convolution

https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

Spatial Separable Convolutions

Challenge: All kernels are not spatially separable



Simple and spatial separable convolution

https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

Sobel Kernel

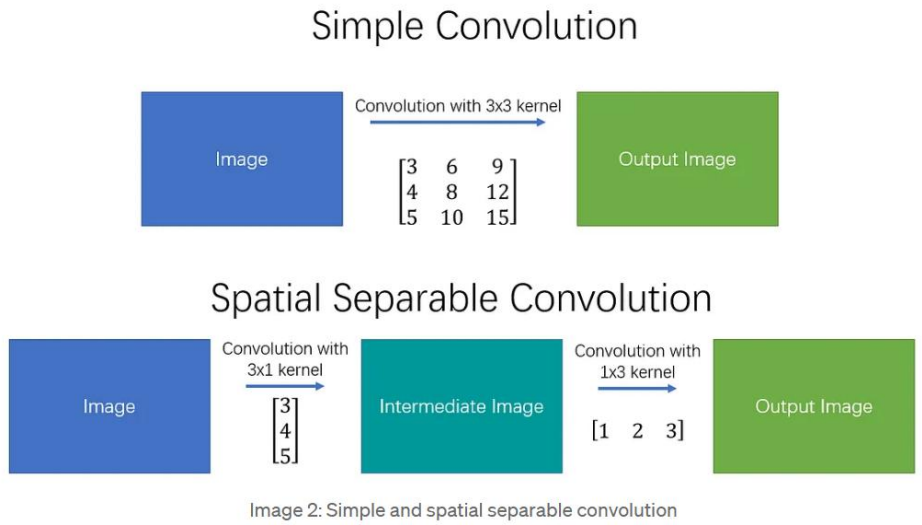


Image 2: Simple and spatial separable convolution

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

Separating the Sobel kernel

https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

Reducing the Number of Computation

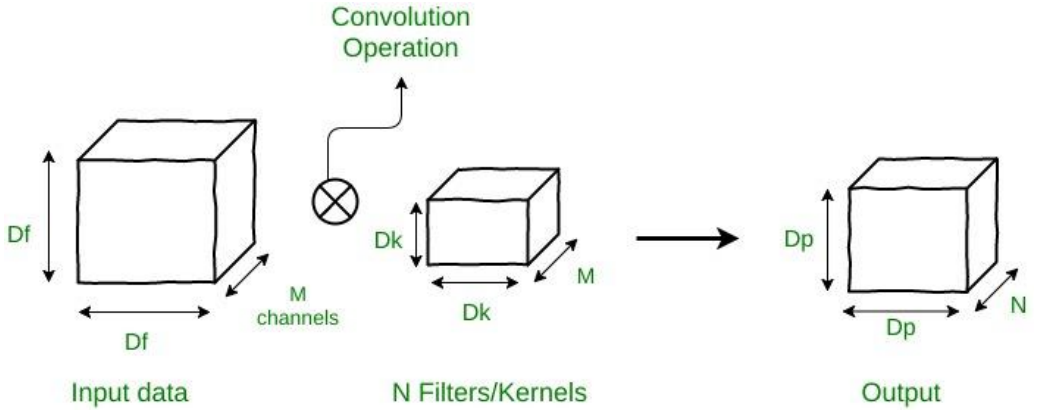
- Normal convolution

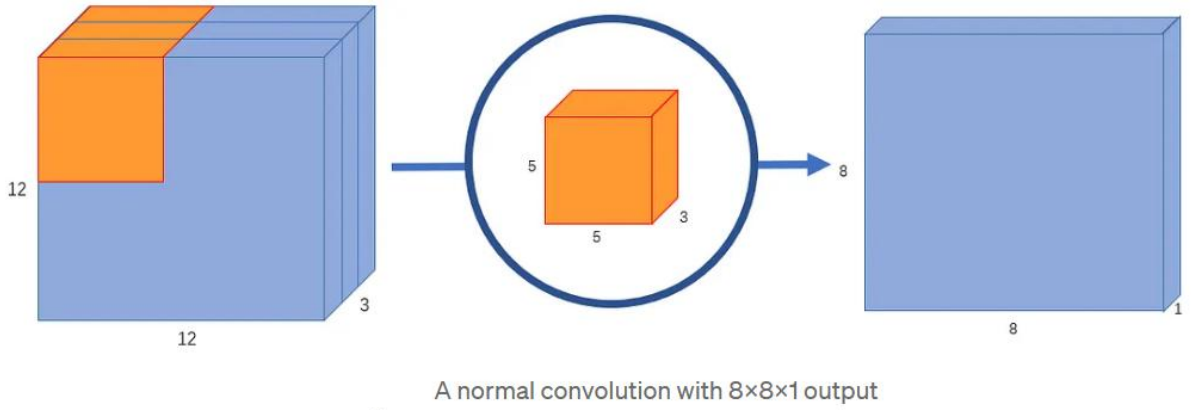
Total multiplications for one output pixel $= D_k \times D_k \times M$

Output feature map size $= D_p \times D_p$ pixels

Total multiplications for one output feature map $= D_k \times D_k \times M \times D_p \times D_p$

Total multiplications for N output feature map $= D_k \times D_k \times M \times D_p \times D_p \times N$





https://www.geeksforgeeks.org/depth-wise-separable-convolutional-neural-networks/

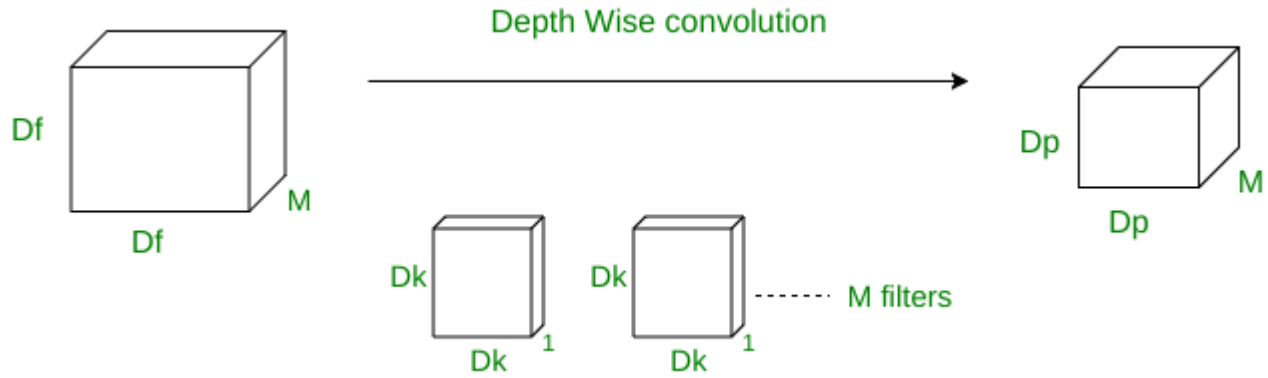
https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

Depthwise-separable Convolution

- Consists of the following steps
  - Depthwise convolution: Convolution is applied to a single channel at a time unlike standard CNN's in which it is done for all the M channels
  - Pointwise convolution: In point-wise operation, a 1×1 convolution operation is applied on the M channels

Reducing the Number of Computation: Depthwise-separable Convolution

- Depthwise convolution: Convolution is applied to a single channel at a time unlike standard CNN's in which it is done for all the M channels

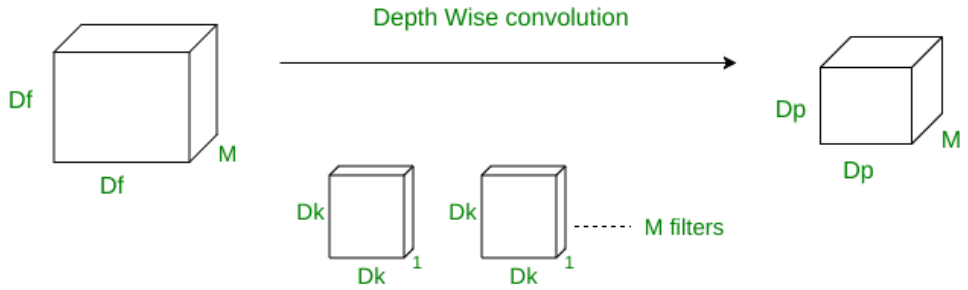


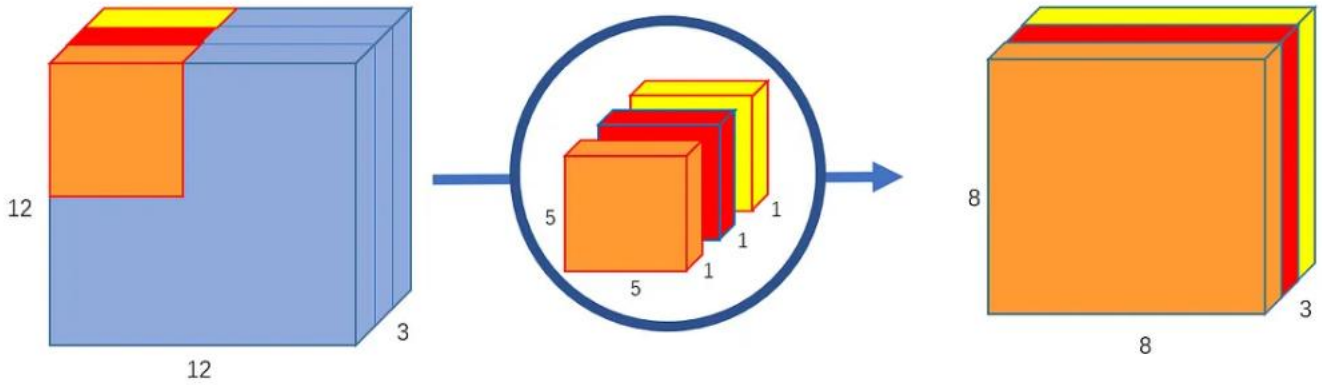
Total multiplications for M output feature maps $= (D_k \times D_k \times 1) \times (M \times D_p \times D_p)$

Depthwise-separable Convolution

Depthwise convolution:
Convolution is applied to a single channel at a time unlike standard CNN's in which it is done for all the M channels

Total multiplications for M output feature maps $= (D_k \times D_k \times 1) \times (M \times D_p \times D_p)$

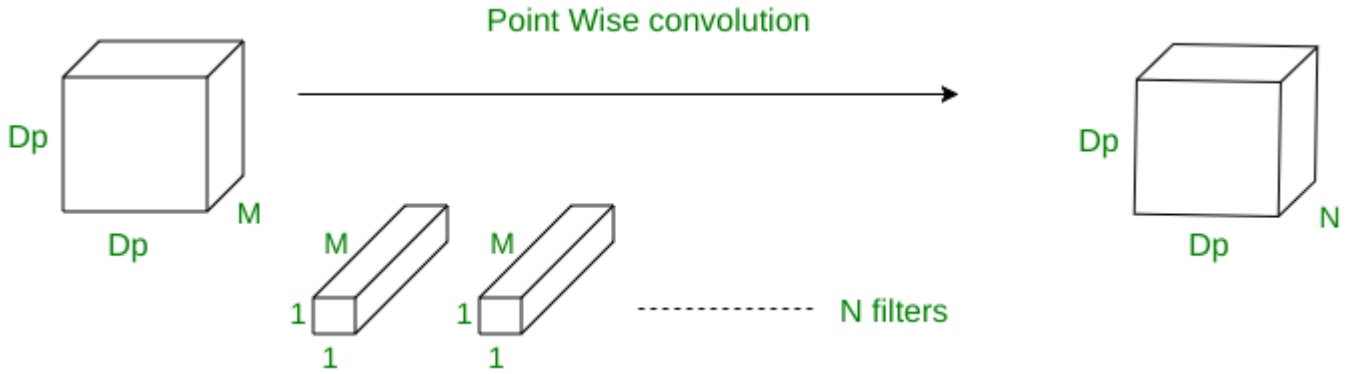




Depthwise convolution, uses 3 kernels to transform a 12×12×3 image to a 8×8×3 image

Depthwise-separable Convolution

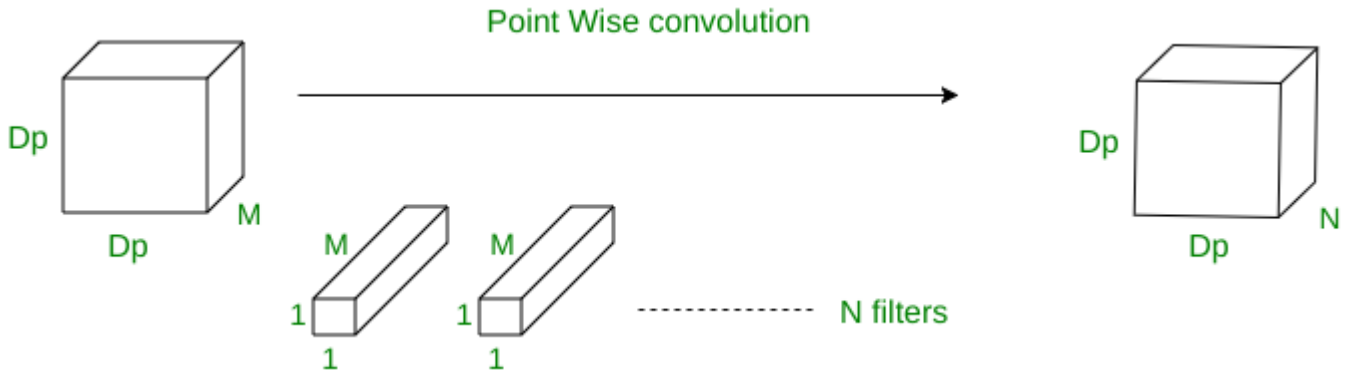
- Pointwise convolution: In point-wise operation, a 1×1 convolution operation is applied on the M channels



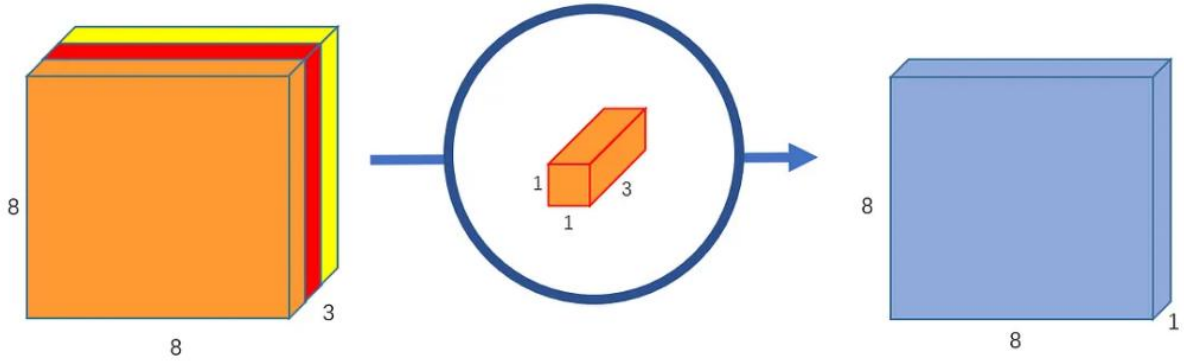
Total multiplications for pointwise convolution$= \left(M \times D_p \times D_p\right) \times N$

Depthwise-separable Convolution

Pointwise convolution: In point-wise operation, a 1×1 convolution operation is applied on the M channels



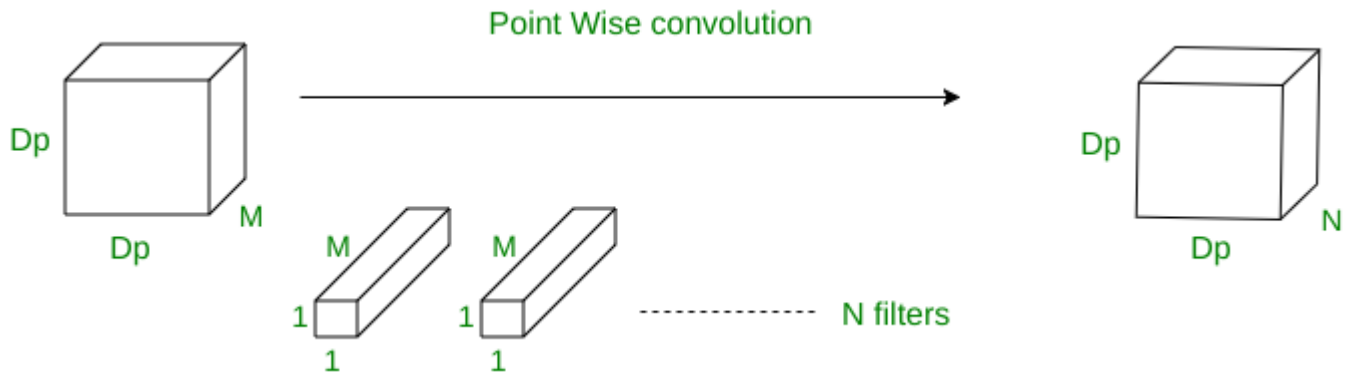
Total multiplications for pointwise convolution$= \left(M \times D_p \times D_p\right) \times N$



Pointwise convolution, transforms an image of 3 channels to an image of 1 channel

Reducing the Number of Computation: Depthwise-separable Convolution

- Pointwise convolution: In point-wise operation, a 1×1 convolution operation is applied on the M channels



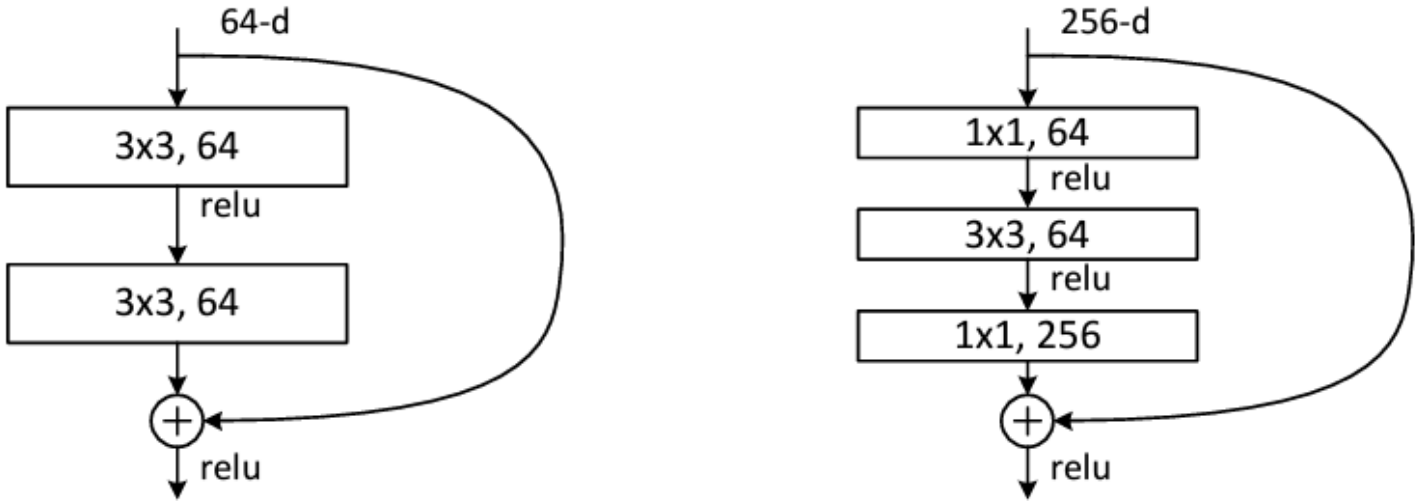
Total multiplications for pointwise convolution= $\left(M \times D_p \times D_p\right) \times N$

Total multiplications for depthwise separable convolution= $MD_p^2 D_k^2 + MD_p^2 N$

Comparing the Computation

- Total multiplications for depthwise separable convolution= $MD_p^2D_k^2 + MD_p^2N$
- Total multiplications for N output feature map in normal convolution = $D_k^2D_p^2MN$
- Ratio of reduction in computation = $\frac{MD_p^2(D_k^2+N)}{D_k^2D_p^2MN} = \frac{D_k^2+N}{D_k^2N} = \frac{1}{N} + \frac{1}{D_k^2}$

Improving the Efficiency of CNN: Residual Connection



Better feature propagation

Dealing with vanishing gradient

Input Standardization

- Suppose one feature of the data has range of 0 to 100000 and another feature has a range of 0 to 10
- If we do not normalize, features with lower value range may loose their meaning when combined response is created in a node
- Large spread in feature value may cause large gradient
  - The parameter can change dramatically
  - Training will become unstable

Input Standardization

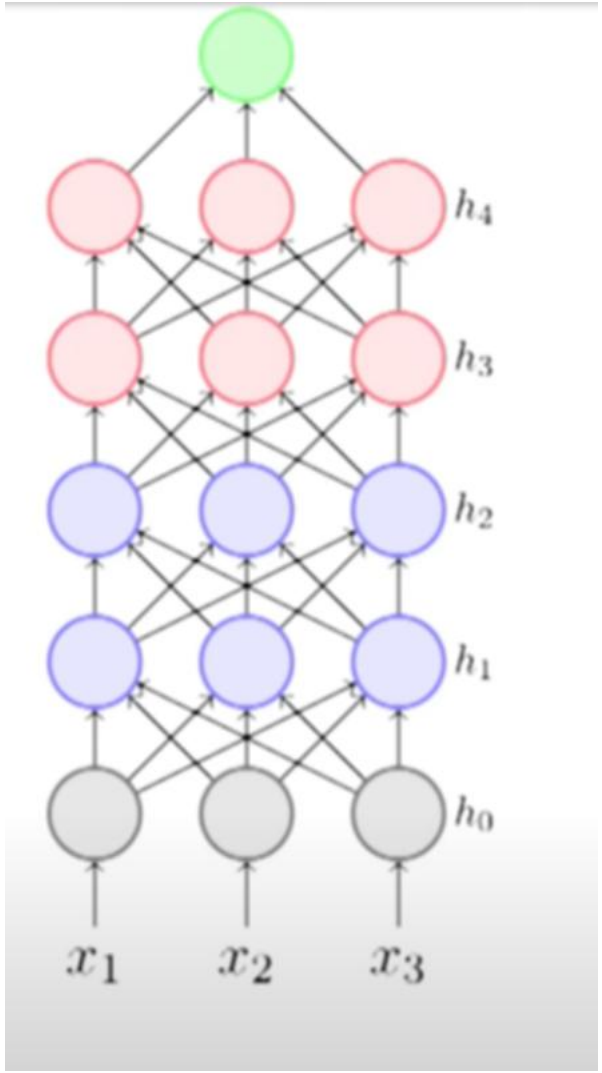
- Standardization at input: create feature values with 0 mean and unit variance
- Suppose, there are n data points
- For a particular feature f, the feature values for n data points are $f_1, f_2, \dots, f_n$
- The normalized feature value is

$$f_i' = \frac{f_i - \mu_f}{\sigma_f}$$

$$\mu_f : Mean\ of\ f_1, f_2, \dots, f_n$$

$$\sigma_f : std\ of\ f_1, f_2, \dots, f_n$$

Batch Normalization

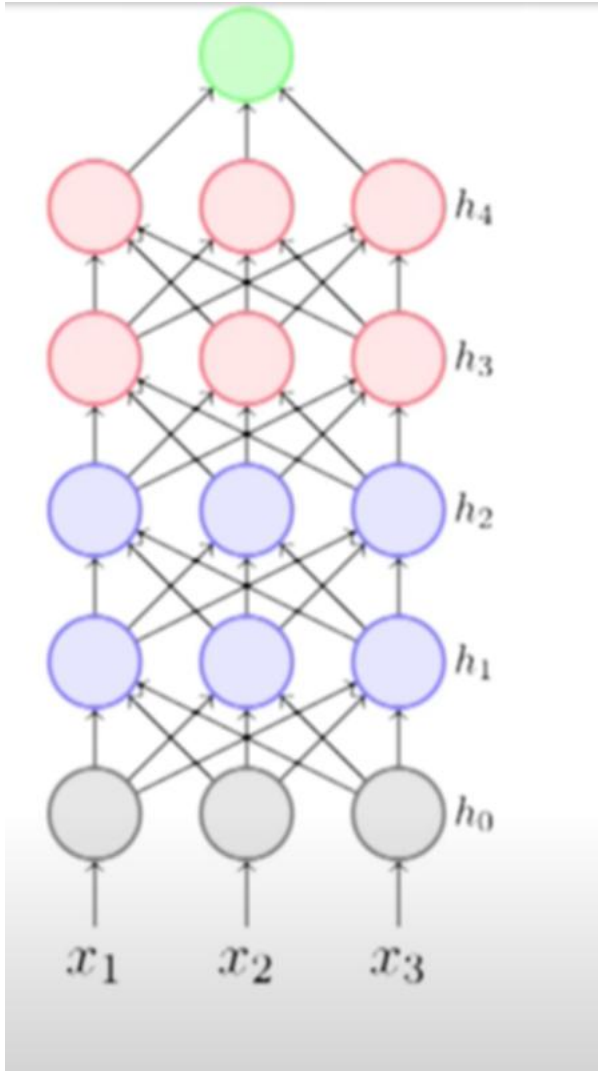


Consider hidden layer $h3$

The distribution of the data at h_3 can change across mini batches

This will make the training difficult

Batch Normalization



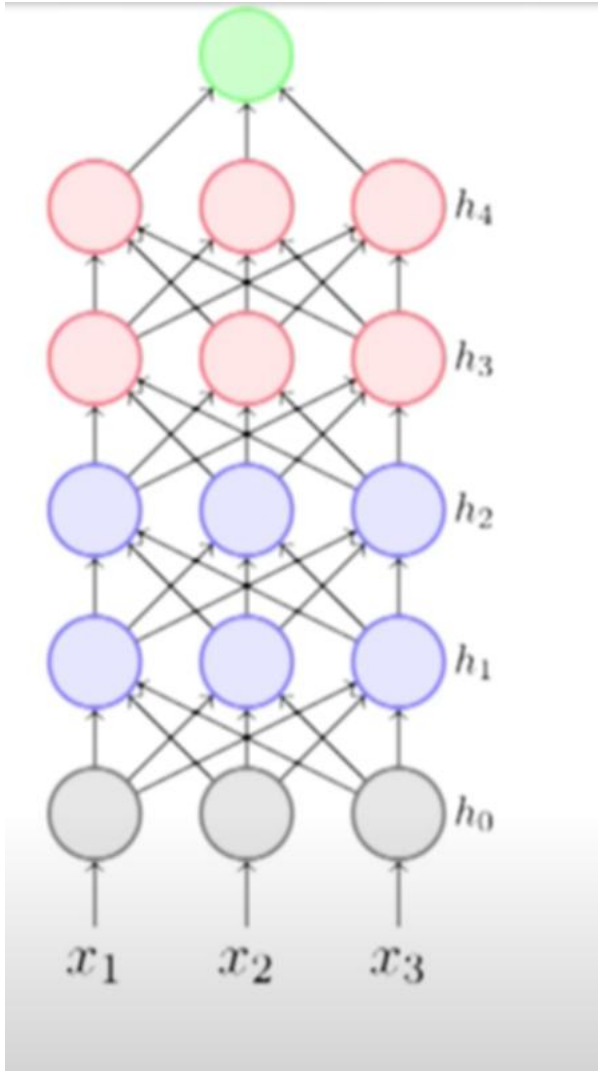
Consider hidden layer $h3$

The distribution of the data at h_3 can change across mini batches

This will make the training difficult

Solution: Make the inputs at each layer a unit Gaussian

Batch Normalization



Solution: Make the inputs at each layer a unit Gaussian

Take the output from the previous layer.

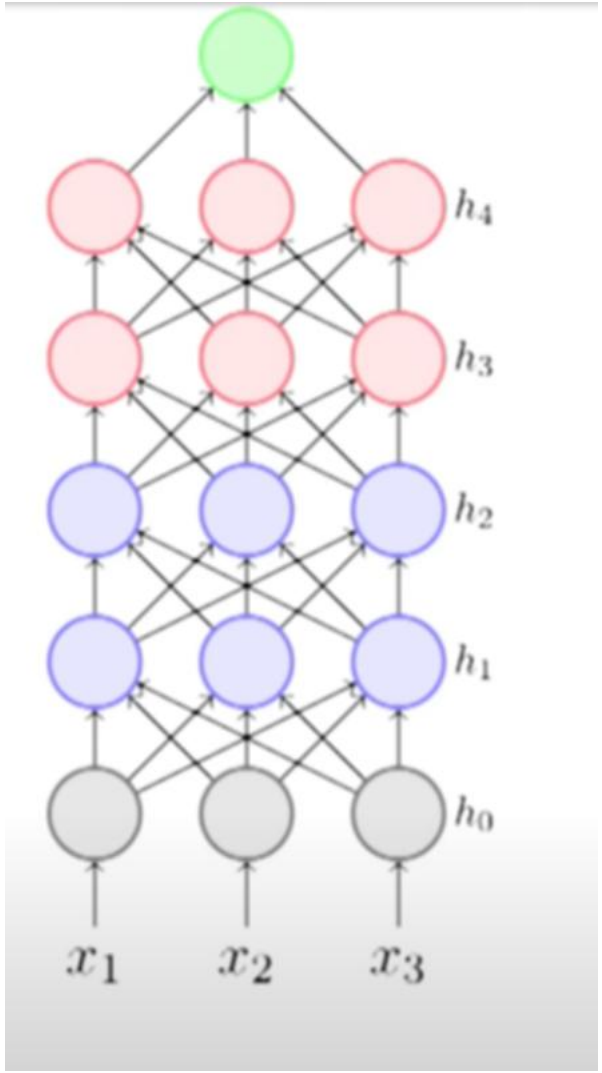
Let z_{ij} be the output of the i^{th} node of the previous layer to j^{th} node of the current layer

$$z_{ij}{}' = \frac{z_{ij} - \mu_{lb}}{\sigma_{lb}}$$

μ_{lb}: $Mean\ of z_{ij}$s for minibatch b

σ_{lb}: $std\ of of z_{ij}$s for minibatch b

Batch Normalization



Solution: Make the inputs at each layer a unit Gaussian

Take the output from the previous layer.

Let z_{ij} be the output of the i^{th} node of the previous layer to j^{th} node of the current layer

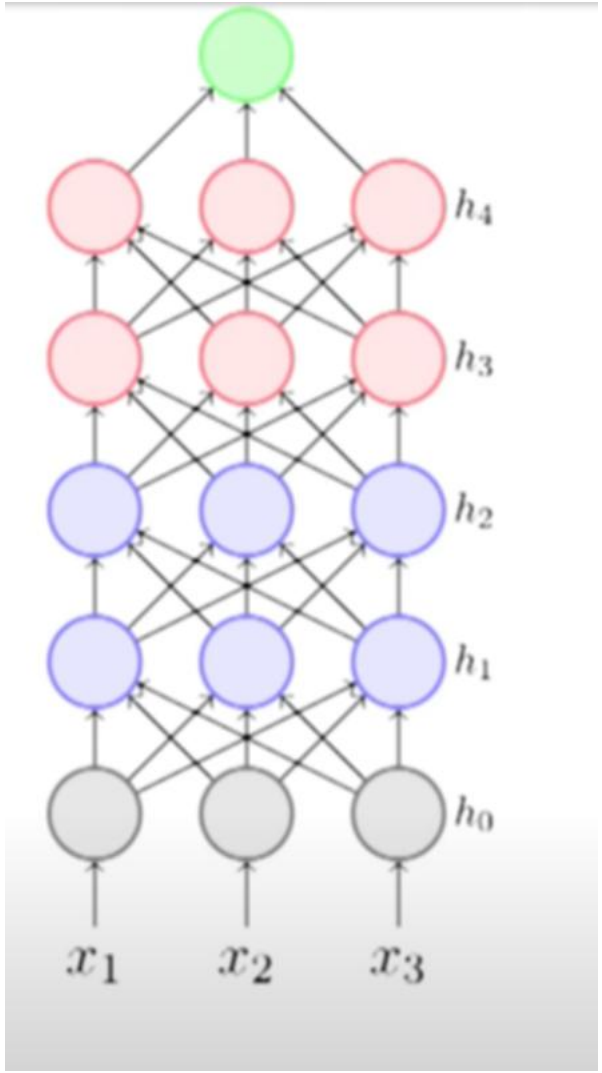
$$z_{ij}{}' = \frac{z_{ij} - \mu_{lb}}{\sigma_{lb}}$$

The final output from the previous layer is

$$zf_{ij} = \beta z_{ij}{}' + \gamma$$

β and γ are learnable parameters

Batch Normalization



After training is done, calculate μ_l and σ_l using the entire training data

Then for the test data from the i^{th} node of the previous layer to j^{th} node of the current layer, we do

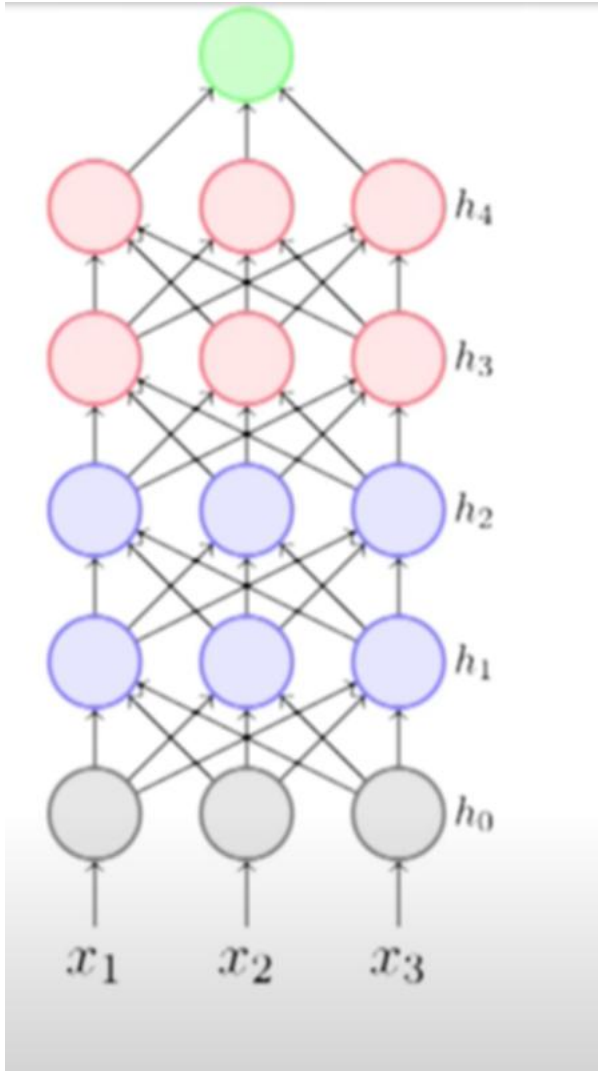
$$zt_{ij}{}' = \frac{zt_{ij} - \mu_l}{\sigma_l}$$

The final output from the previous layer is

$$zft_{ij} = \beta zt_{ij}{}' + \gamma$$

β and γ are learnable parameters

Batch Normalization



After training is done, calculate μ_l and σ_l using the entire training data (Expensive)

Then for the test data from the i^{th} node of the previous layer to j^{th} node of the current layer, we do

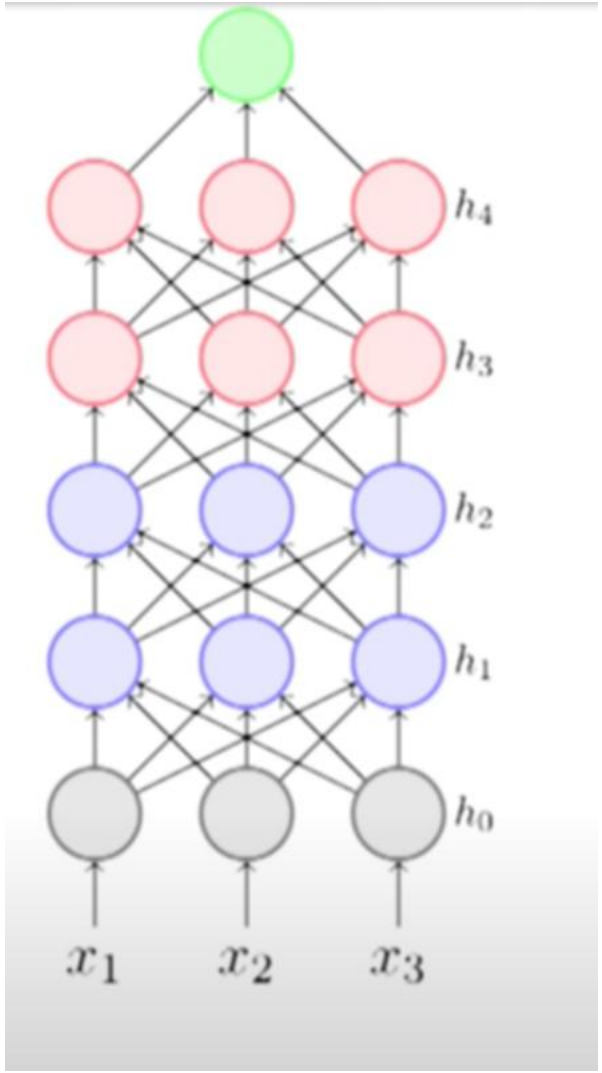
$$zt_{ij}{}' = \frac{zt_{ij} - \mu_l}{\sigma_l}$$

The final output from the previous layer is

$$zft_{ij} = \beta zt_{ij}{}' + \gamma$$

β and γ are learnable parameters

Batch Normalization



After training is done, calculate μ_l and σ_l using the entire training data (Expensive)
Take moving average of mean and std during training μ_m and σ_m

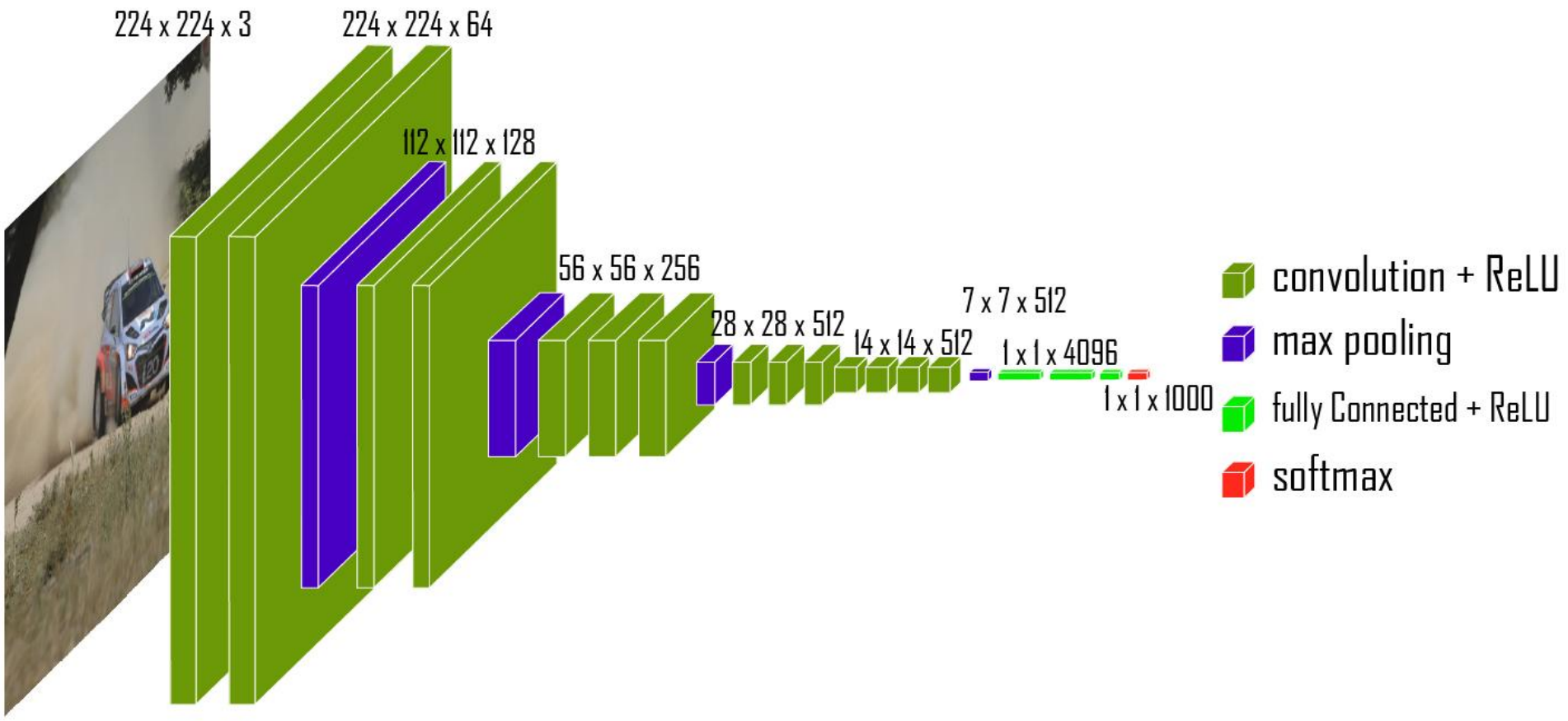
Then for the test data from the i^{th} node of the previous layer to j^{th} node of the current layer, we do

$$zt_{ij}{}' = \frac{zt_{ij} - \mu_m}{\sigma_m}$$

The final output from the previous layer is
$$zft_{ij} = \beta zt_{ij}{}' + \gamma$$

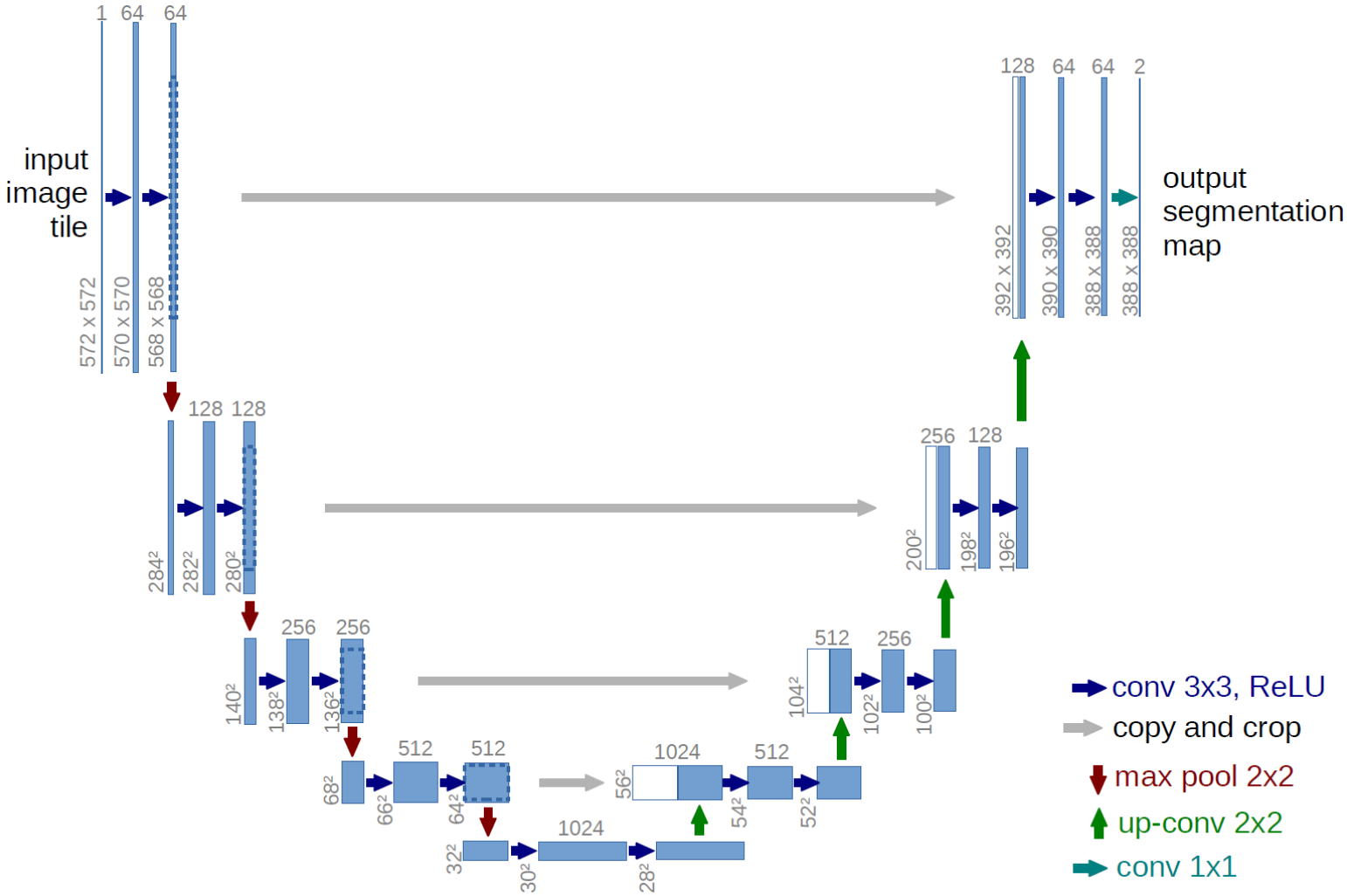
β and γ are learnable parameters

CNN Models: VGG 16

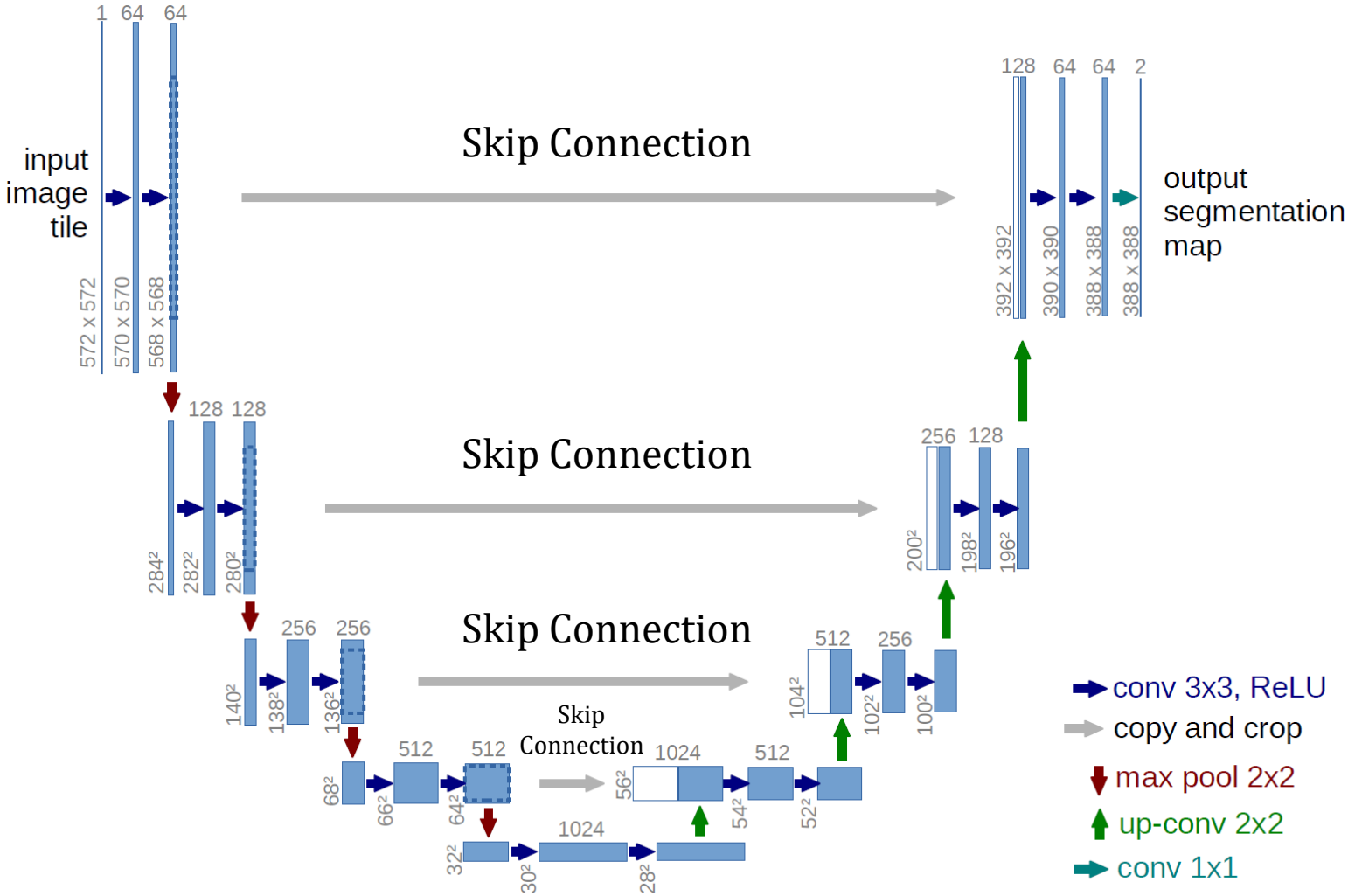


https://www.geeksforgeeks.org/vgg-16-cnn-model/

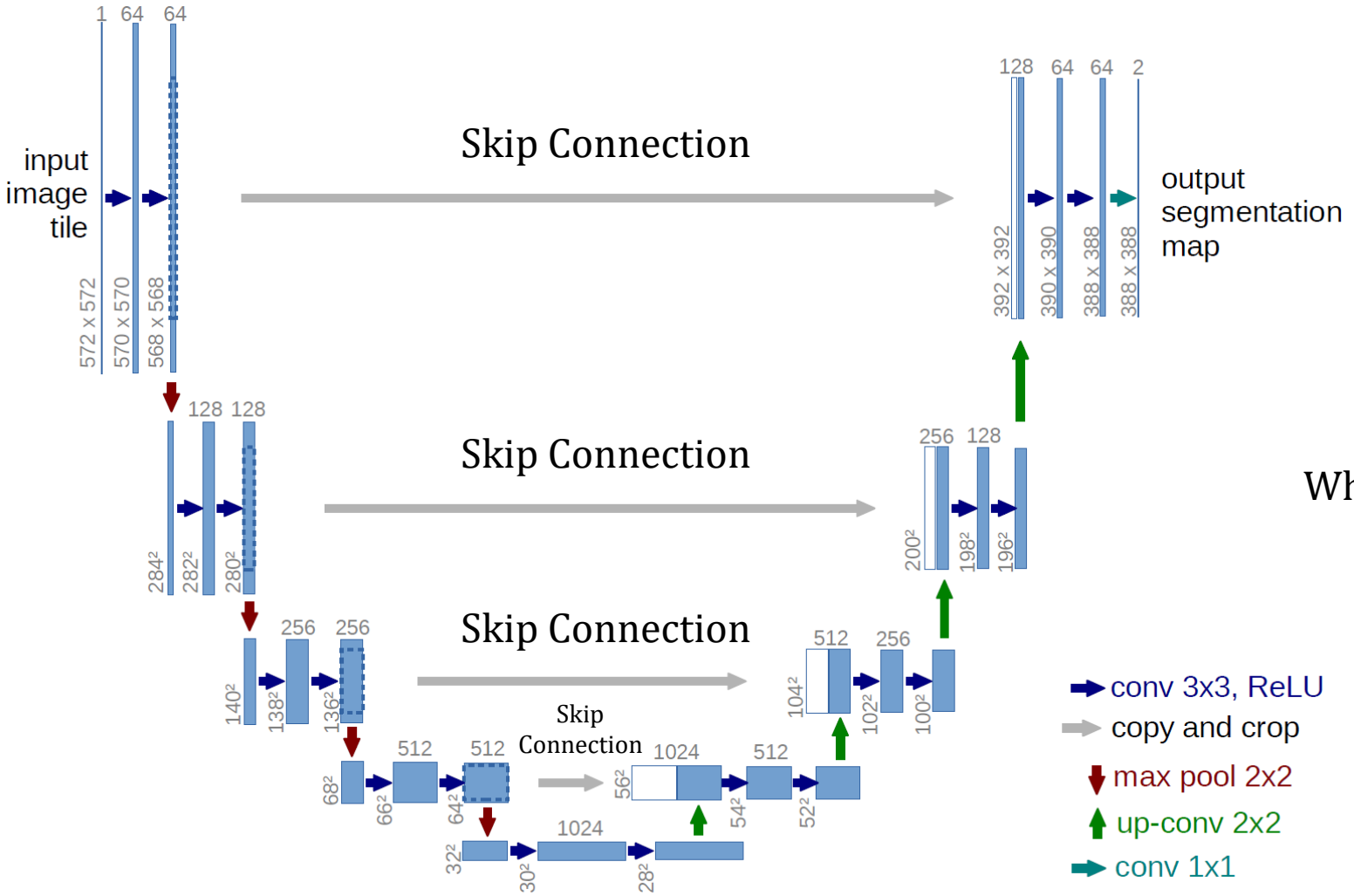
CNN Models: U-Net (Skip Connection)



CNN Models: U-Net (Skip Connection)



CNN Models: U-Net (Skip Connection)



Why skip connection?

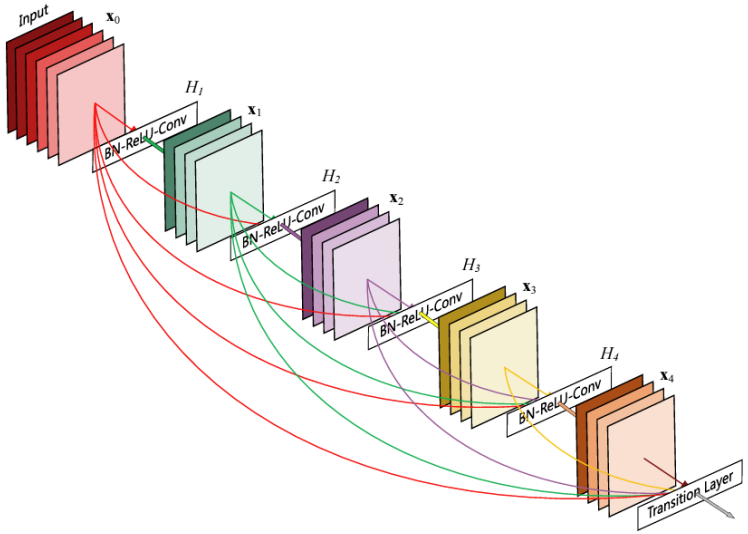
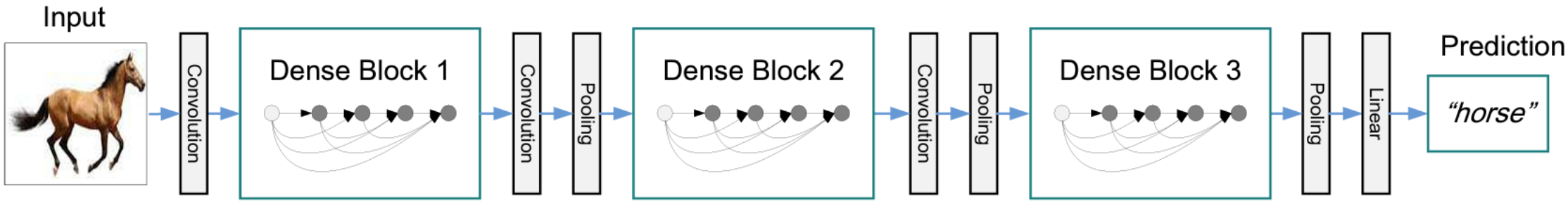


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

CNN Models: DenseNet (Dense Connection)



Sequence Models

Sequence of Information

- So far, we have dealt with data of fixed length
- But, what if we have a sequence of information in one data point
  - e.g, a sequence of images in a video (one data point)
- RNN can process a sequence of information
- Parameter sharing

Sequence of Information

- So far, we have dealt with data of fixed length
- But, what if we have a sequence of information in one data point
  - e.g, a sequence of images in a video (one data point)
- While processing such a sequence of information, the model parameters should be shared across each piece of information

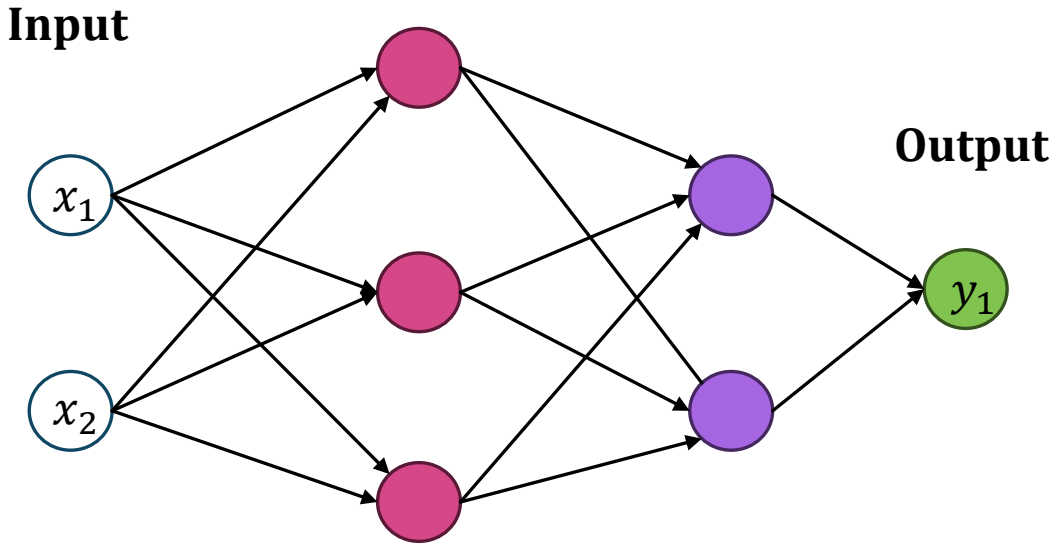
Recurrent Neural Networks

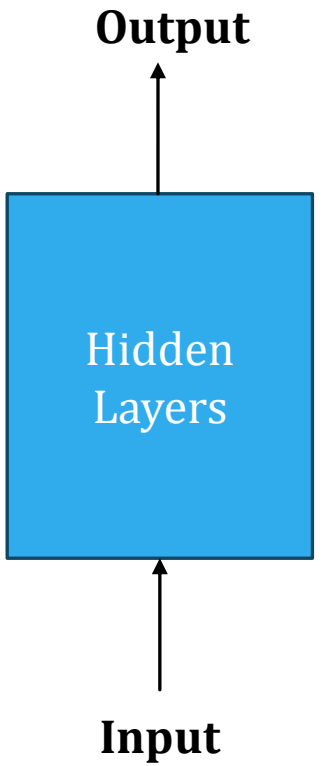
- Recurrent neural networks (RNNs) are deep learning models that capture the dynamics of sequences via recurrent connections

- Can be thought of as cycles in the network of nodes

Representation of Neural Networks

- Two equivalent representations

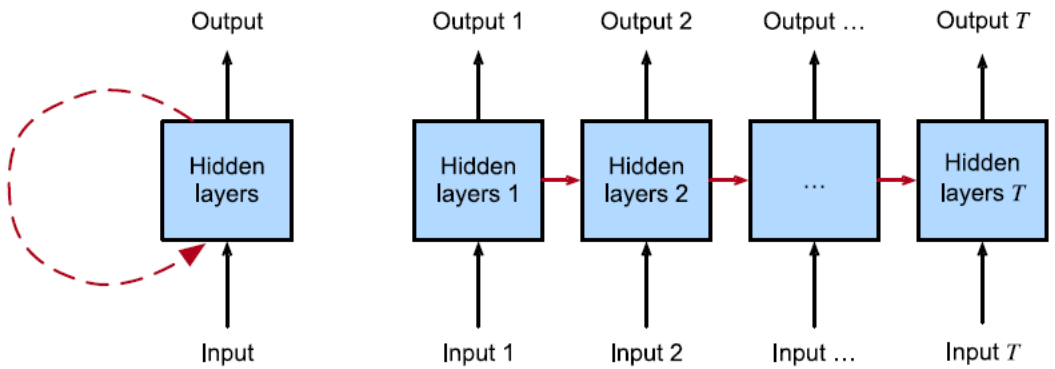




Recurrent Neural Networks

- Recurrent neural networks (RNNs) are deep learning models that capture the dynamics of sequences via recurrent connections

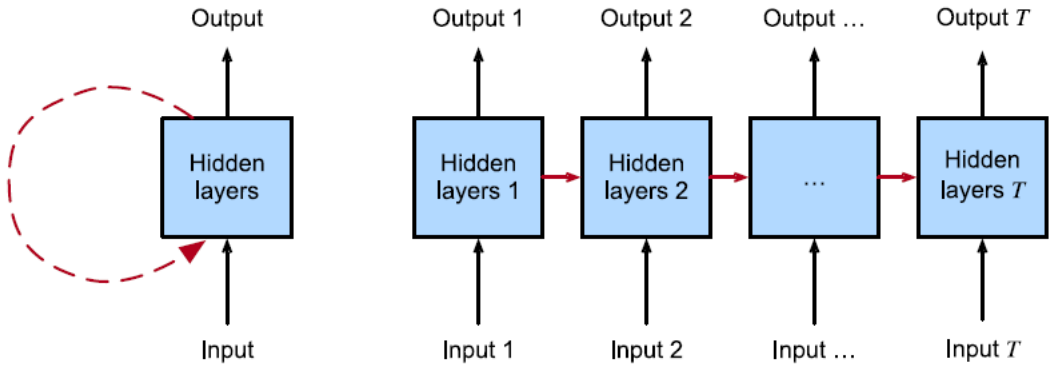
- Can be thought of as cycles in the network of nodes



On the left recurrent connections are depicted via cyclic edges. On the right, we unfold the RNN over time steps. Here, recurrent edges span adjacent time steps, while conventional connections are computed synchronously.

Recurrent Neural Networks

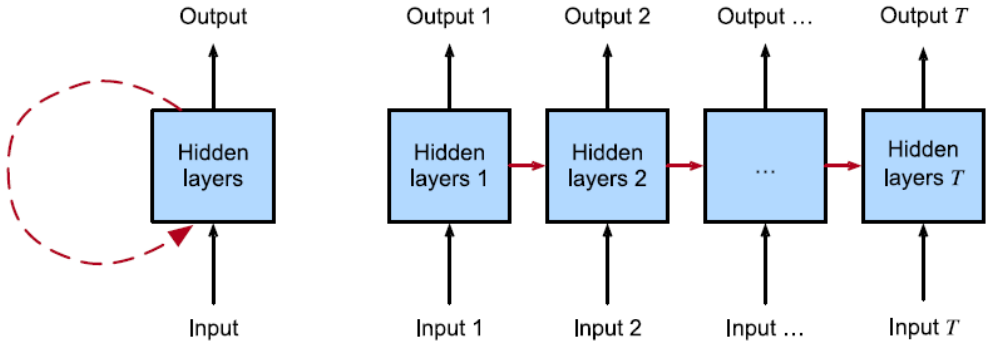
- Input: a sequence of feature vectors $\{x_1, x_2, \ldots, x_T\}$



On the left recurrent connections are depicted via cyclic edges. On the right, we unfold the RNN over time steps. Here, recurrent edges span adjacent time steps, while conventional connections are computed synchronously.

Recurrent Neural Networks

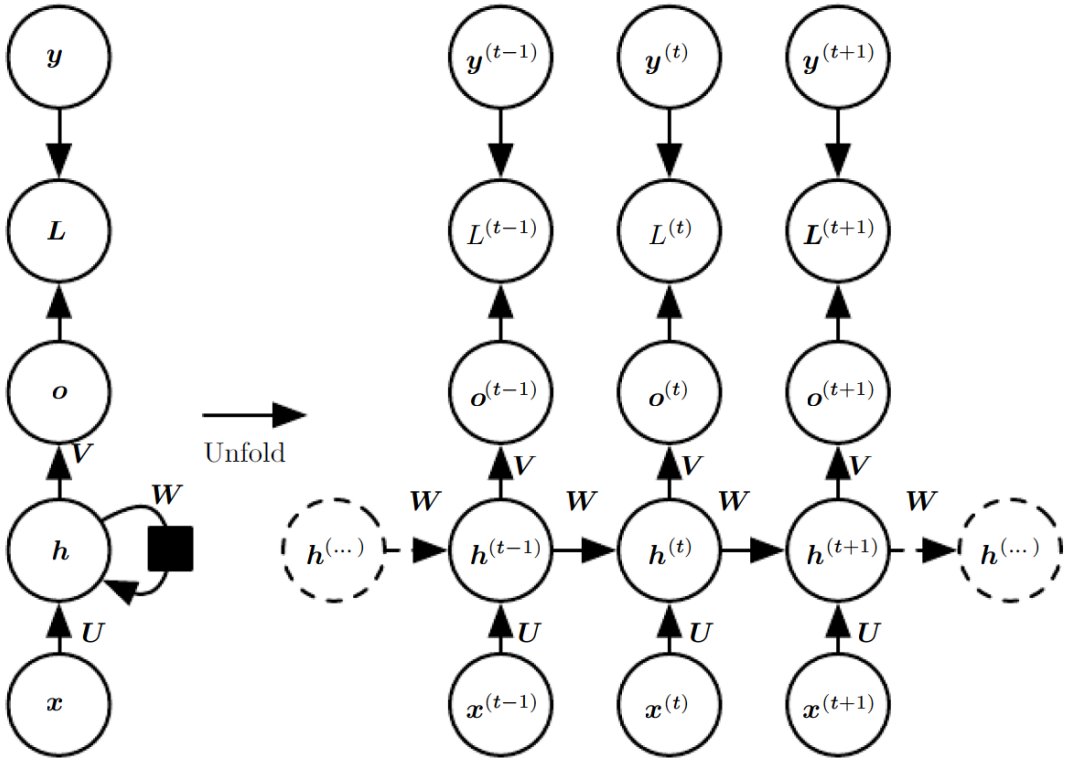
- Input: a sequence of feature vectors $\{x_1, x_2, \dots, x_T\}$
- Information arriving at time instant t is not independent of what happened previously
- For example, consider a movie scene. The frame at time t is not independent of what happened in previous frames

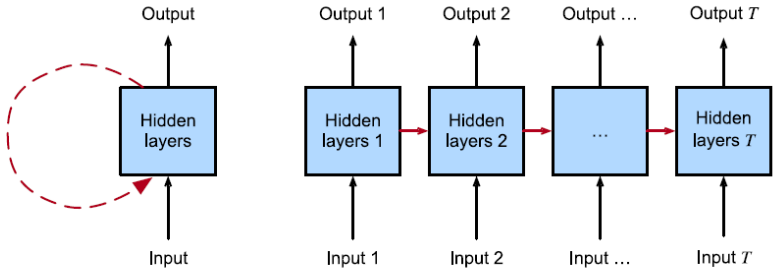


On the left recurrent connections are depicted via cyclic edges. On the right, we unfold the RNN over time steps. Here, recurrent edges span adjacent time steps, while conventional connections are computed synchronously.

Unfolded computational graph

Unfolded Computational Graph





On the left recurrent connections are depicted via cyclic edges. On the right, we unfold the RNN over time steps. Here, recurrent edges span adjacent time steps, while conventional connections are computed synchronously.

Recurrent Neural Networks

- We sometimes wish to predict a fixed target y given sequentially structured input (e.g., sentiment classification based on a movie review)
- At other times, we wish to predict a sequentially structured target $(y_1, \ldots, y_T)$ given a fixed input (e.g., image captioning)
- Still other times, our goal is to predict sequentially structured targets based on sequentially structured inputs (e.g., machine translation or video captioning).

Recurrent Neural Networks

- Such sequence-to-sequence tasks take two forms
  - Aligned: where the input at each time step aligns with a corresponding target (e.g., part of speech tagging)
  - Unaligned: where the input and target do not necessarily exhibit a step-for-step correspondence (e.g., machine translation).

Inputs and Outputs

- For many ML tasks, inputs and outputs may not be represented as fixed length vectors
  - Example: document processing

- However, in may cases, they may be represented as a varying length sequence of fixed length vectors
  - Example: in document processing, each word can be represented as a fixed length vector
  - The entire document is the varying length collection of words

The Basic Task in Sequence Modeling

- Predicting the probability of occurrence of a sequence of tokens
  - $P(x_1, x_2, \dots, x_t)$
  - Joint probability distribution

- From the chain rule of probability
  - $P(x_1, x_2, \dots, x_t) = P(x_t | x_{t-1}, x_{t-2}, \dots, x_1) P(x_{t-1}, x_{t-2}, \dots, x_1)$
  - Similarly, we can expand $P(x_{t-1}, x_{t-2}, \dots, x_1)$ as a conditional probability with terms up to the previous time instant

The Basic Task in Sequence Modeling

- So, the major task is sequence modeling is finding $P(x_t|x_{t-1}, x_{t-2}, \dots, x_1)$
  - Predicting the probability of the t^{th} token given $(t-1)$ tokens
  - This is a recurrent process
  - Hence RNN
- Practically, we don't consider all the terms from the beginning
  - Rather consider past $(n-1)$ tokens
  - We consider that the next token is conditionally independent of tokens from further past

- So, we are interested in $P(x_t|x_{t-1}, x_{t-2}, \dots, x_{t-n+1})$

The Basic Task in Sequence Modeling

- So, we are interested in $P(x_t|x_{t-1}, x_{t-2}, \dots, x_{t-n+1})$
- Technically, we could store all the $(n-1)$ terms and subsequently process with a NN
- But this would increase the number of parameters and storage with increasing n

The Hidden State Model

- So, we are interested in $P(x_t|x_{t-1}, x_{t-2}, \dots, x_{t-n+1})$
- Technically, we could store all the $(n-1)$ terms and subsequently process with a NN
- But this would increase the number of parameters and storage with increasing n
- So, we combine the effect of past $(n-1)$ tokens into one variable h_{t-1}
- Consequently, through the RNN, we are interested to find

$$P(x_t|h_{t-1})$$

The Hidden State

- Through the RNN, we are interested to find

$$P(x_t|h_{t-1})$$

- h_{t-1}: Hidden state that stores the information up to time $(t-1)$

- **Note: Here we consider the independent variable to be time t. However, in other applications, it can be any other independent variable, such as space (e. g. , (x, y) coordinates)**

The Hidden State

- Through the RNN, we are interested to find

$$P(x_t|h_{t-1})$$

- h_{t-1}: Hidden state that stores the information up to time $(t-1)$

- Hidden state at time step t

$$h_t = f(x_t, h_{t-1})$$

Minibatch

- Entire data

Samples	t=1	t=2	t=3	t=4	t=5	t=6
S1	33	21	27	26	30	28
S2	11	15	17	13	9	12
S3	42	41	44	38	35	37
S4	27	27	26	32	29	27
S5	17	16	17	15	11	12
S6	27	28	26	27	30	23

Minibatch

- $\mathbf{X_t}$: Minibatch of tokens at time step t

Samples	t=1	t=2	t=3	t=4	t=5	t=6
S1	33	21	27	26	30	28
S2	11	15	17	13	9	12
S3	42	41	44	38	35	37
S4	27	27	26	32	29	27
S5	17	16	17	15	11	12
S6	27	28	26	27	30	23

Minibatch of size 4 at time $t = 3$

The Hidden State

- $\mathbf{X_t}$: Minibatch of tokens at time step t
- $\mathbf{H_t}$: Minibatch of hidden states at time step t
- Minibatch of hidden states at time step t

$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t}\mathbf{W_{xh}} + \mathbf{H_{t-1}}\mathbf{W_{hh}} + \mathbf{b_h})$$

The Hidden State & Output

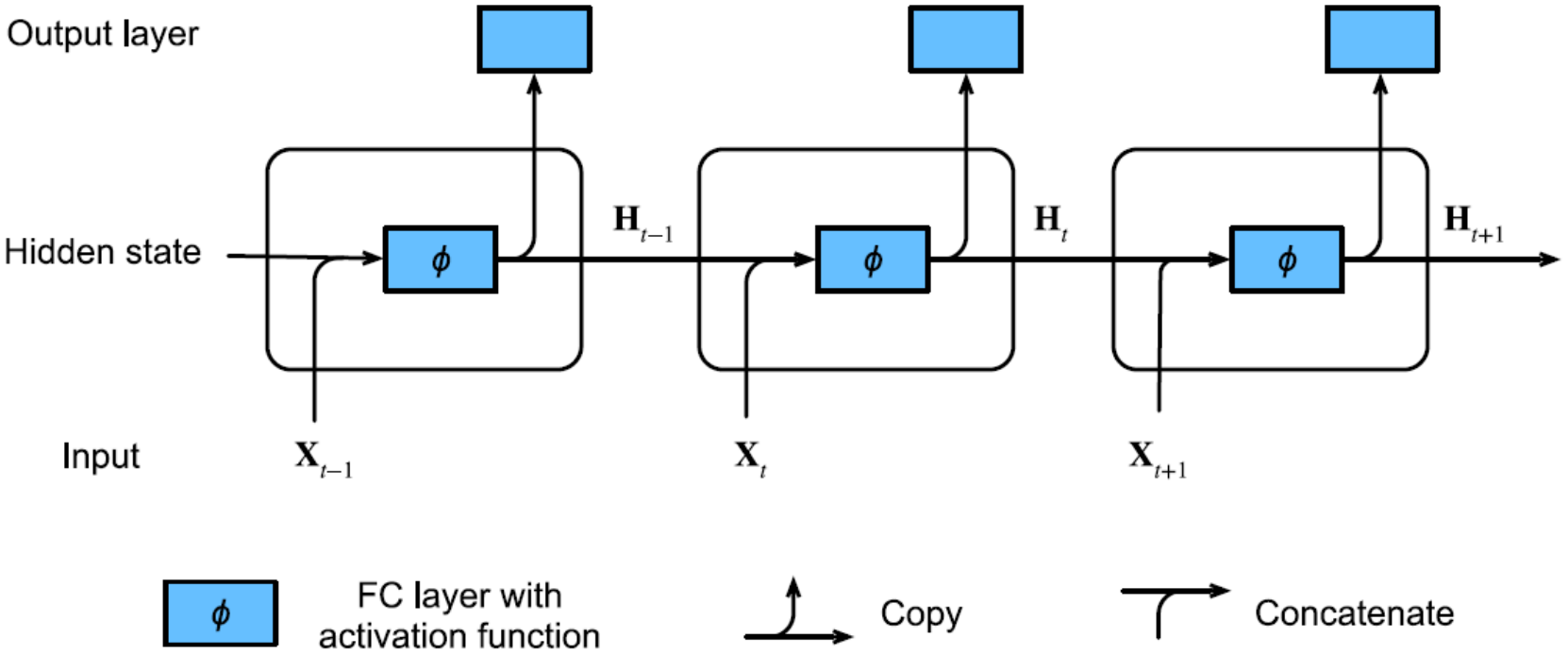
- Minibatch of hidden states at time step t

$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$$

- Output

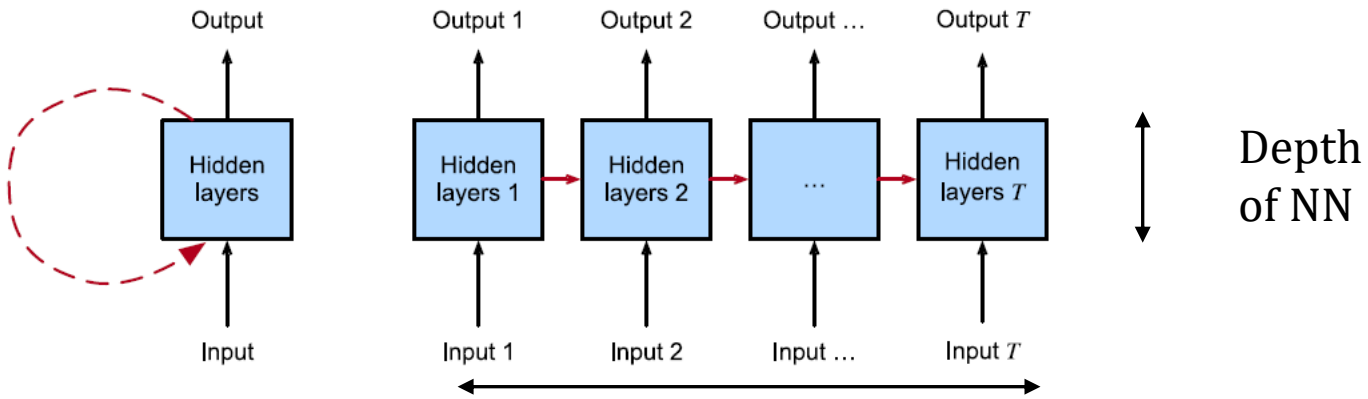
$$\hat{\boldsymbol{y}}_\mathbf{t} = \mathbf{H_t W_{h\boldsymbol{y}}} + \mathbf{b_o}$$

The RNN

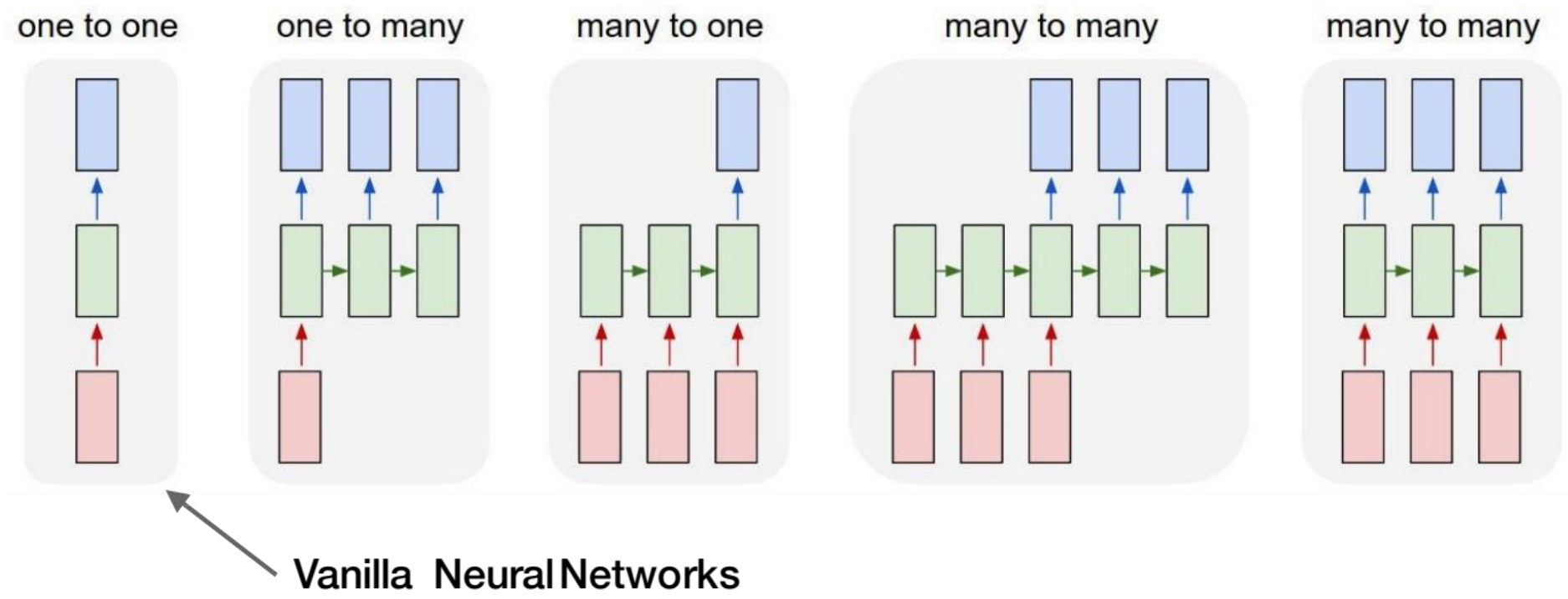


Unfolded computational graph

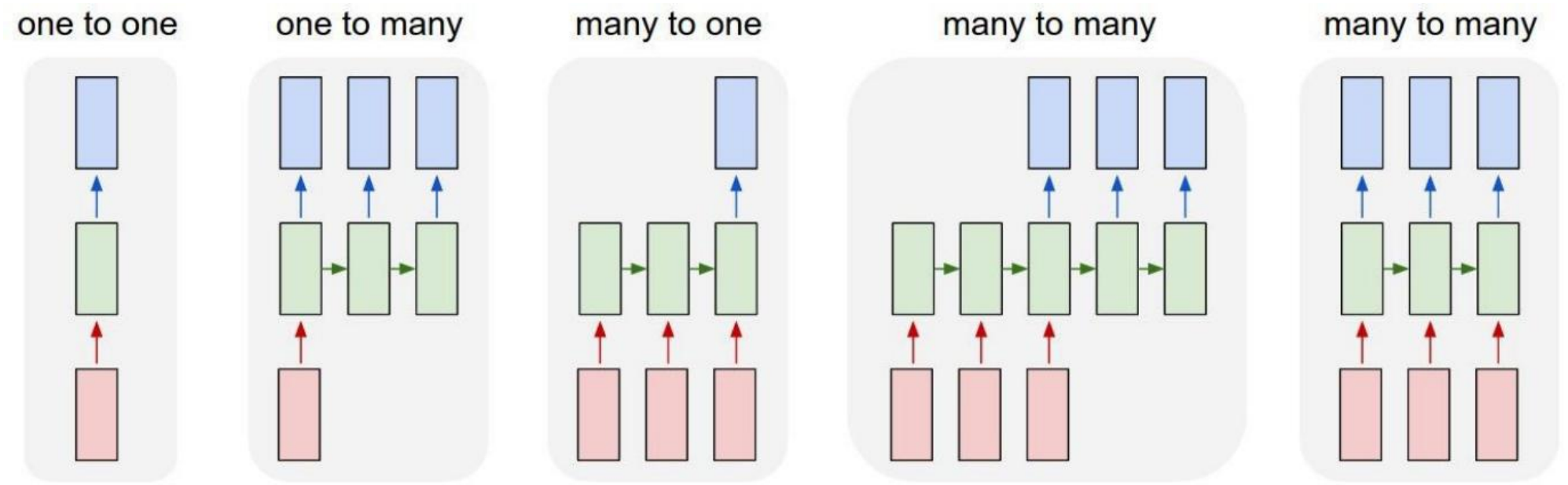
The RNN


Output
Hidden layers
Input
Output 1
Output 2
Output …
Output T
Hidden layers 1
Hidden layers 2
…
Hidden layers T
Input 1
Input 2
Input …
Input T
Depth of NN
Temporal Depth of RNN

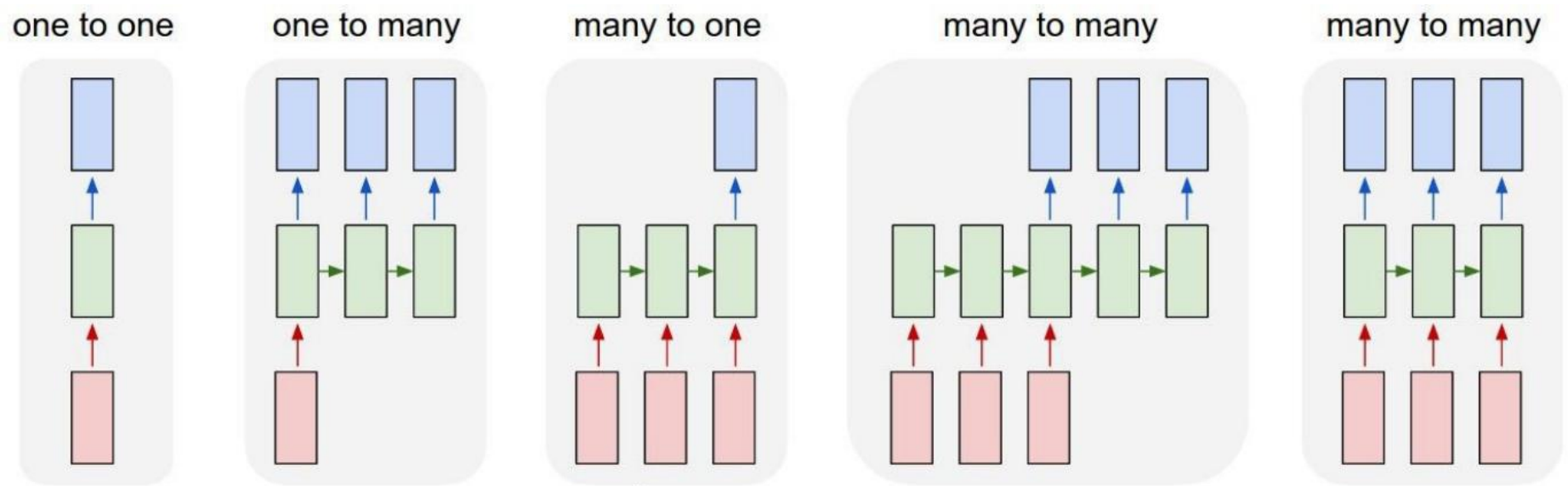
Types of RNN


one to one
one to many
many to one
many to many
many to many
Vanilla Neural Networks

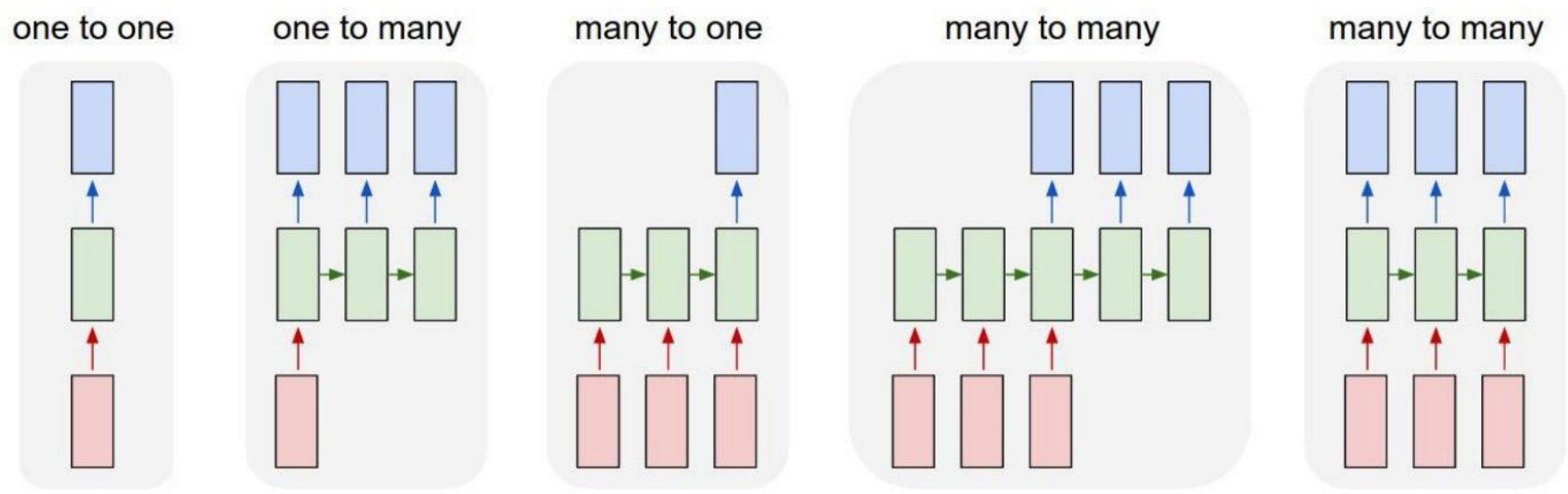
Types of RNN


one to one
one to many
many to one
many to many
many to many
e.g. **Image Captioning**
image -> sequence of words

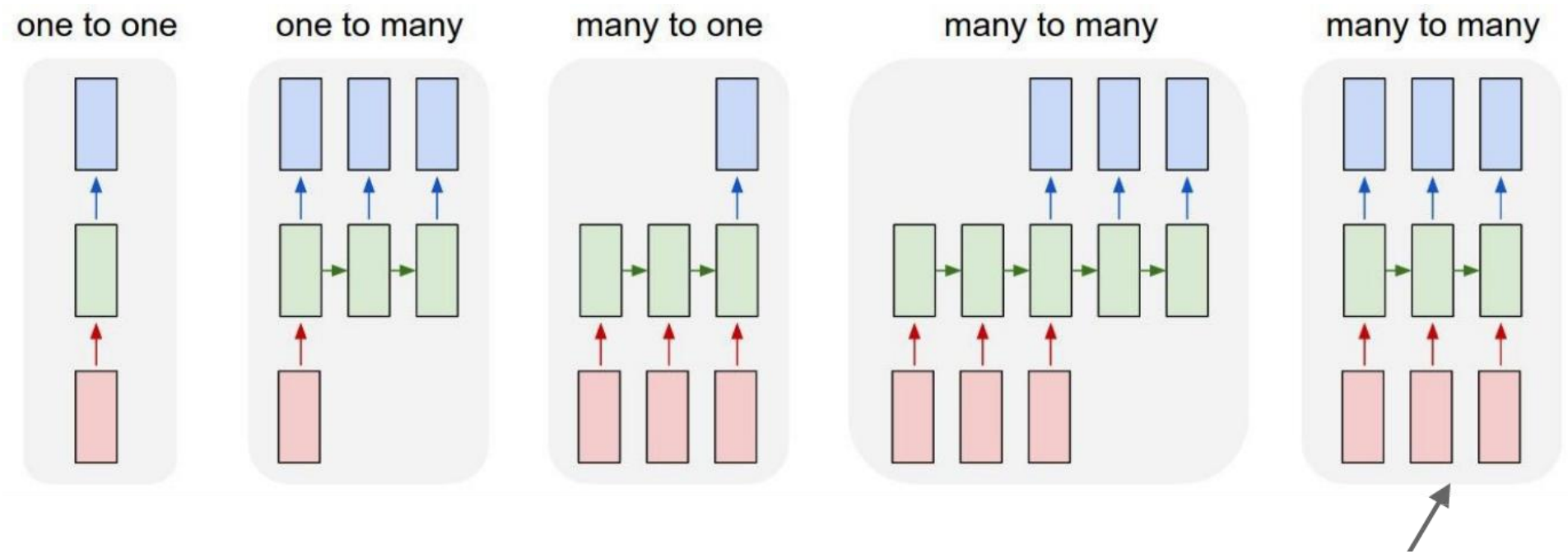
Types of RNN


one to one
one to many
many to one
many to many
many to many
e.g. **Sentiment Classification**
sequence of words -> sentiment

Types of RNN


one to one
one to many
many to one
many to many
many to many
e.g. **Machine Translation**
seq of words -> seq of words
(unaligned)

Types of RNN



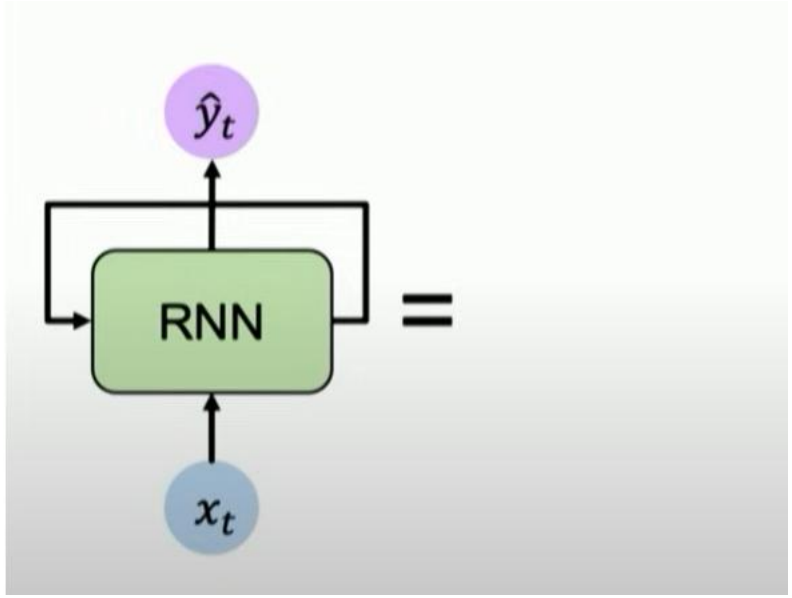
Parts of speech classification (aligned)

Backpropagation through Time

- The unrolled RNN is essentially a feedforward neural network with the special property that the same parameters are repeated throughout the unrolled network, appearing at each time step

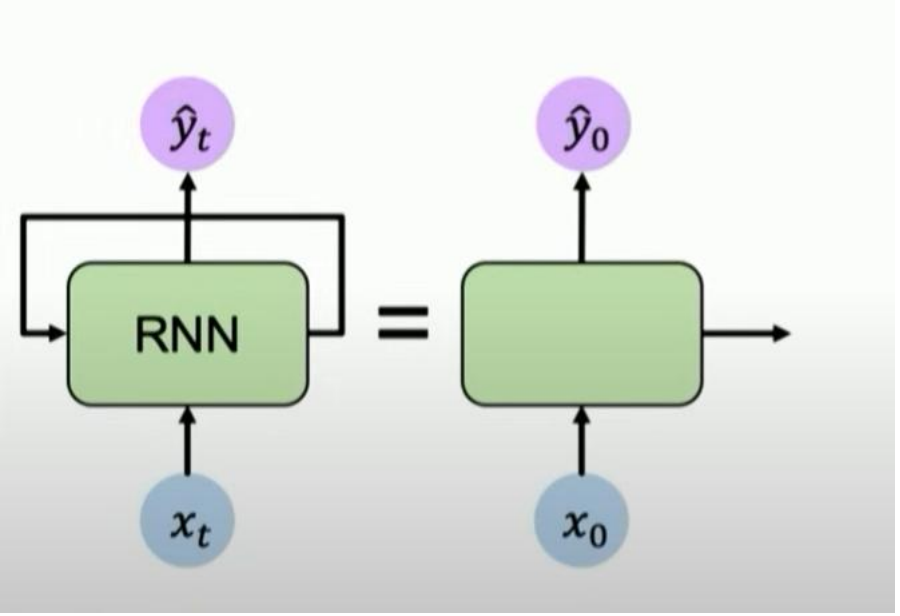
- Like any feedforward neural network, we can apply the chain rule, backpropagating gradients through the unrolled net

Forward Propagation

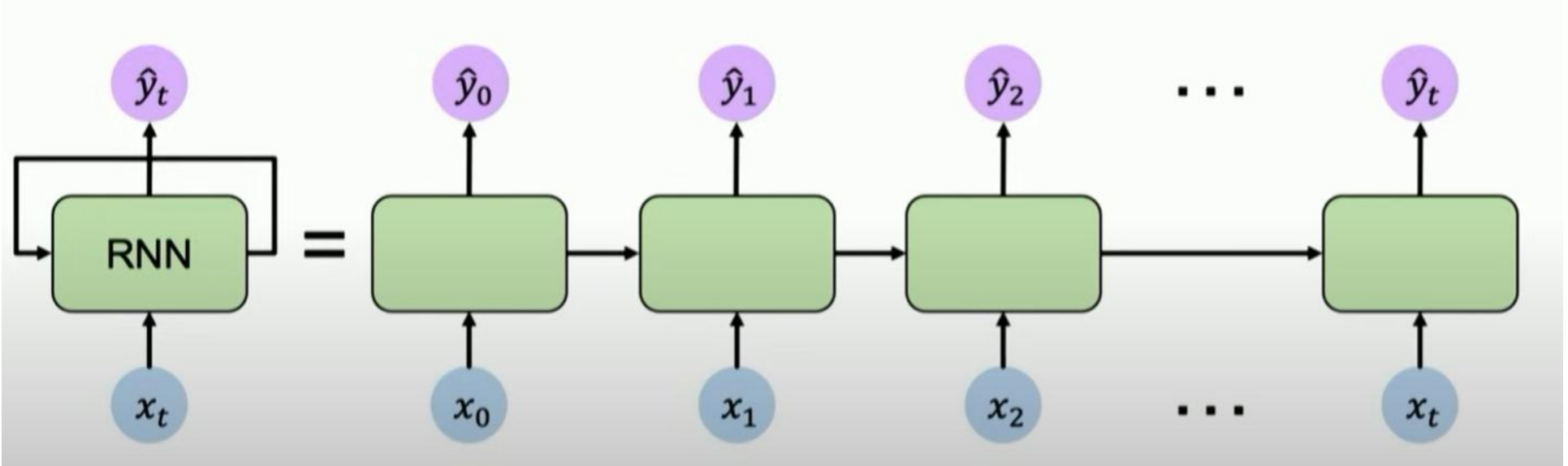


https://www.youtube.com/watch?v=ySEx_Bqxvvo&t=648s

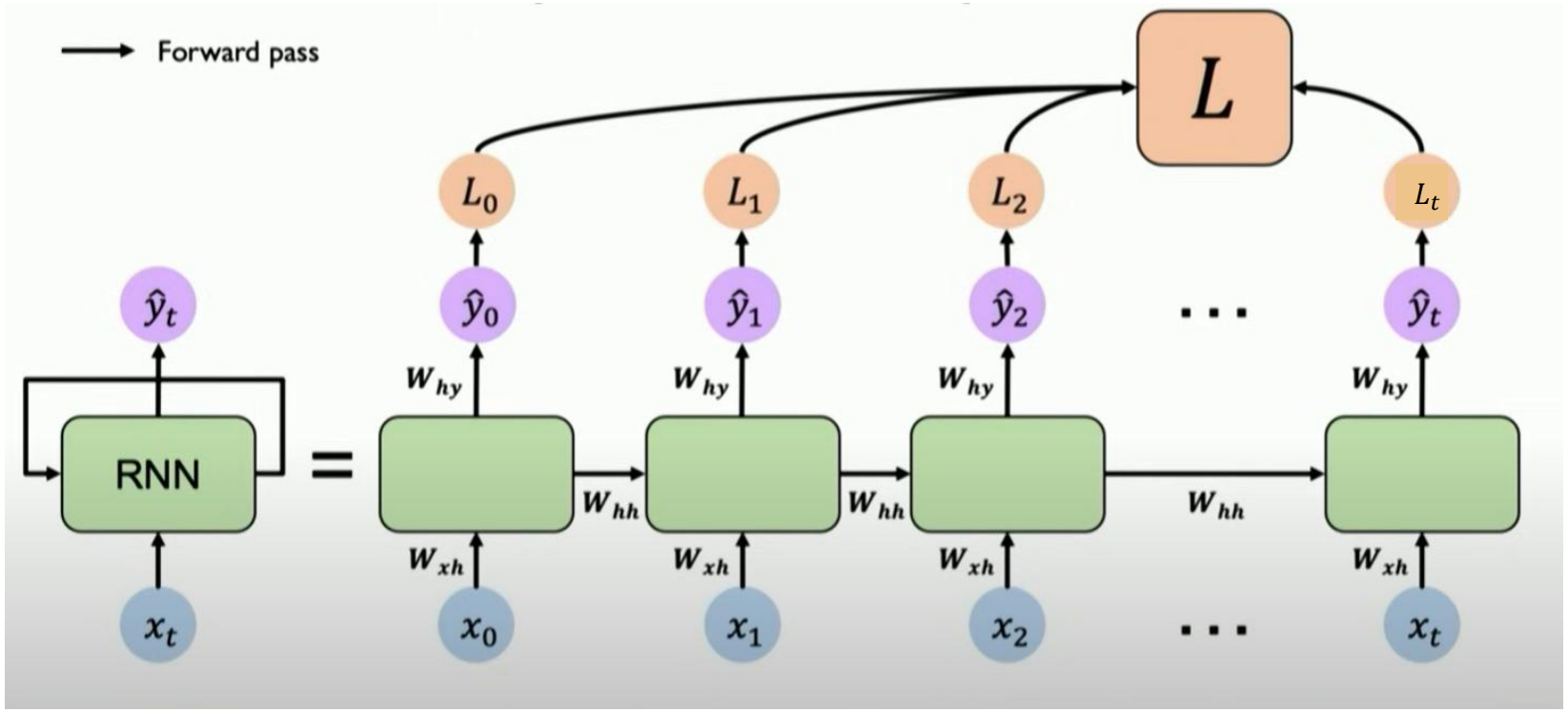
Forward Propagation



Forward Propagation



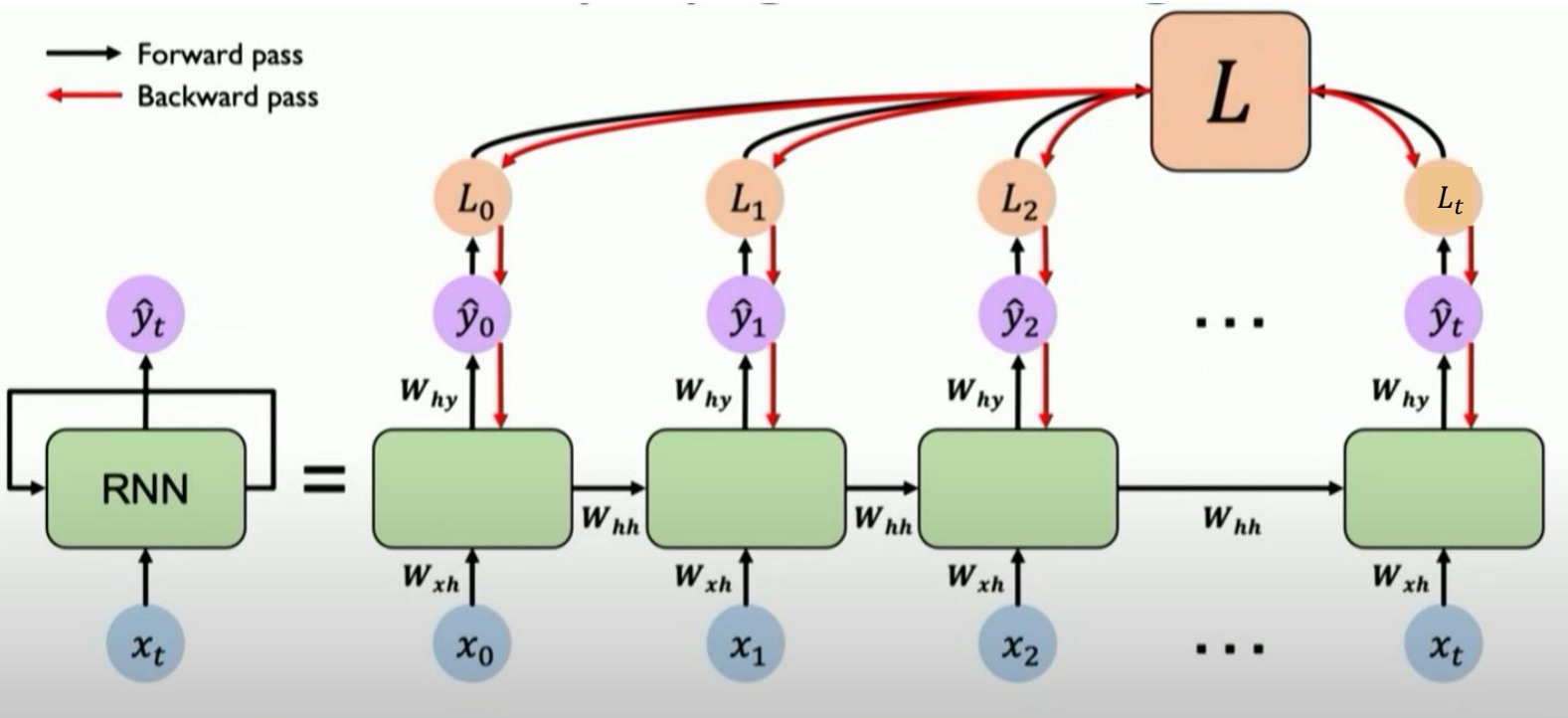
Forward Propagation and Loss



$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$$

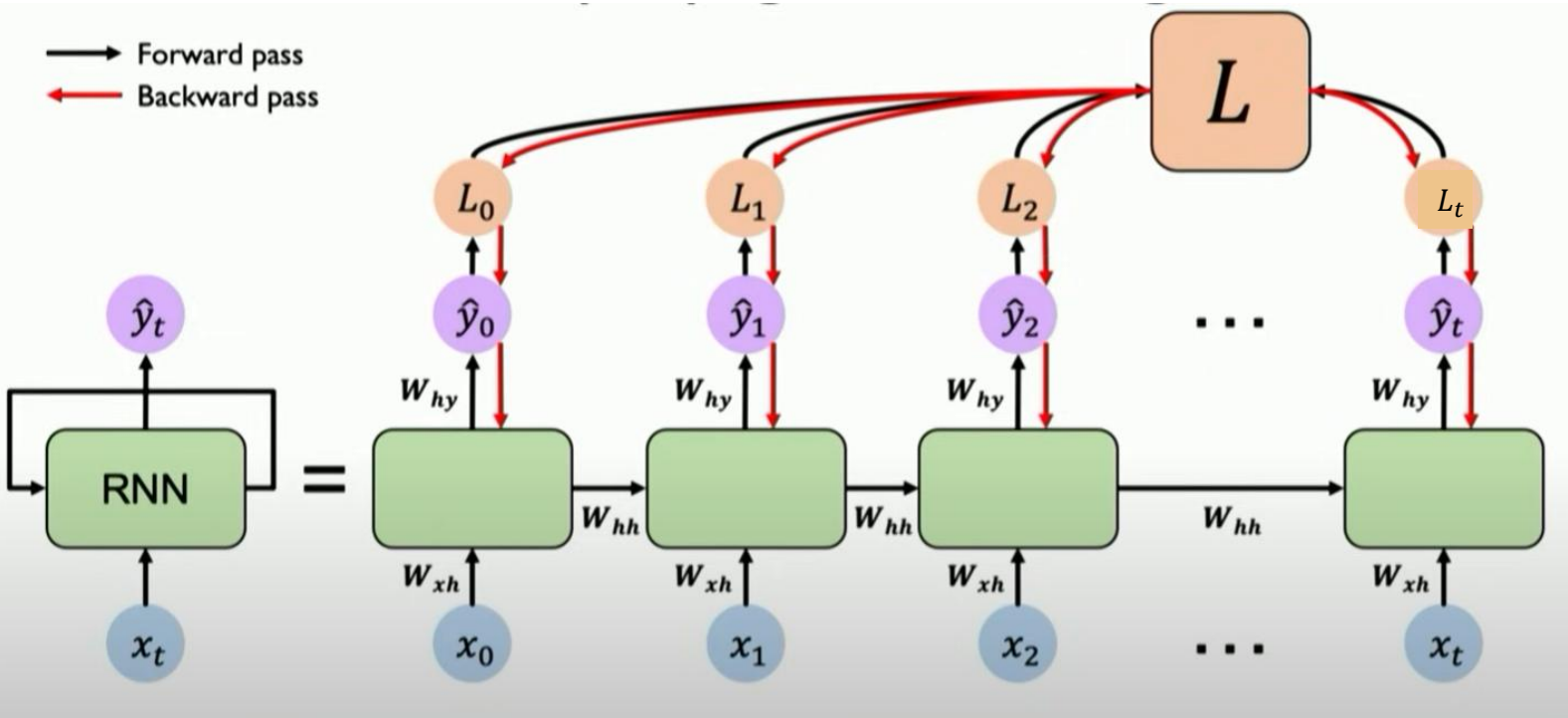
$$\widehat{\boldsymbol{Y}}_\mathbf{t} = \mathbf{H_t W_{hy}} + \mathbf{b_o}$$

Back Propagation



Individual time step-wise loss components are L_i

Back Propagation



Individual time step-wise loss components are L_i

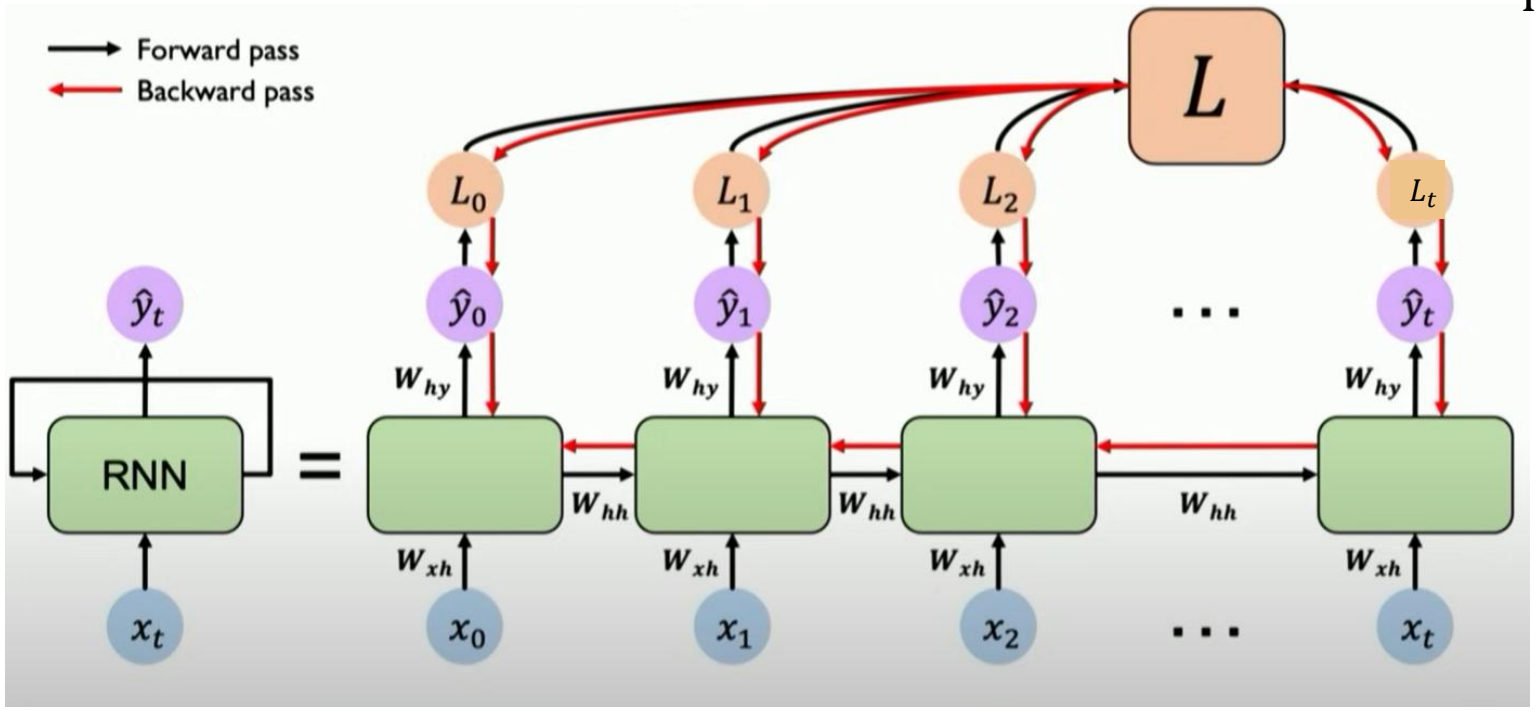
Total Loss

$$L = \frac{1}{t}\sum_{i=1}^{t} L_i$$

Back Propagation through Time

$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$$

$$\hat{\boldsymbol{Y}}_\mathbf{t} = \mathbf{H_t W_{hy}} + \mathbf{b_o}$$



Individual time step-wise loss components are L_i

Total Loss

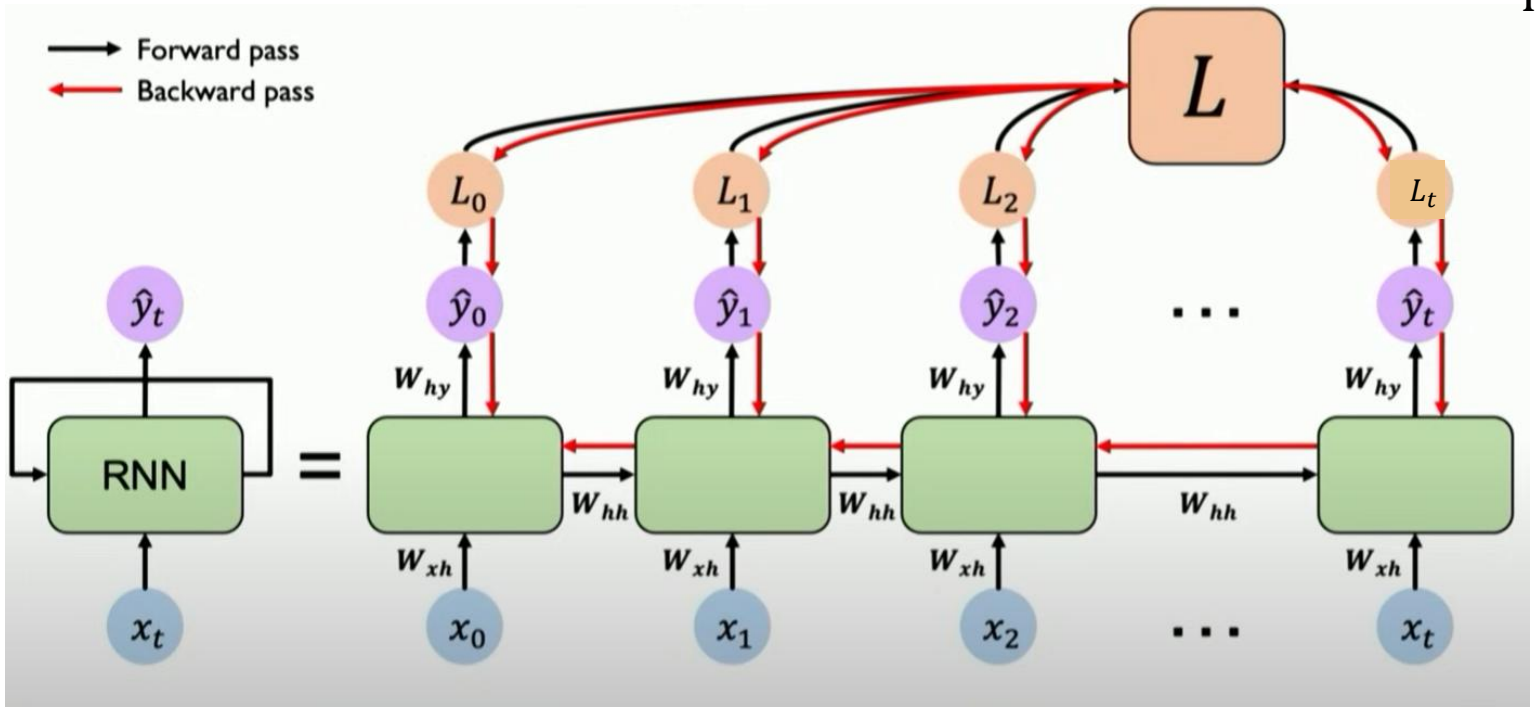
$$L = \frac{1}{t}\sum_{i=1}^{t} L_i$$

$$\frac{\partial L}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial h_i}\frac{\partial h_i}{\partial W_{hh}}$$

Back Propagation through Time

$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$$

$$\hat{\boldsymbol{Y}}_\mathbf{t} = \mathbf{H_t W_{hy}} + \mathbf{b_o}$$



Individual time step-wise loss components are L_i

Total Loss for one input data point

$$L = \frac{1}{t}\sum_{i=1}^{t} L_i$$

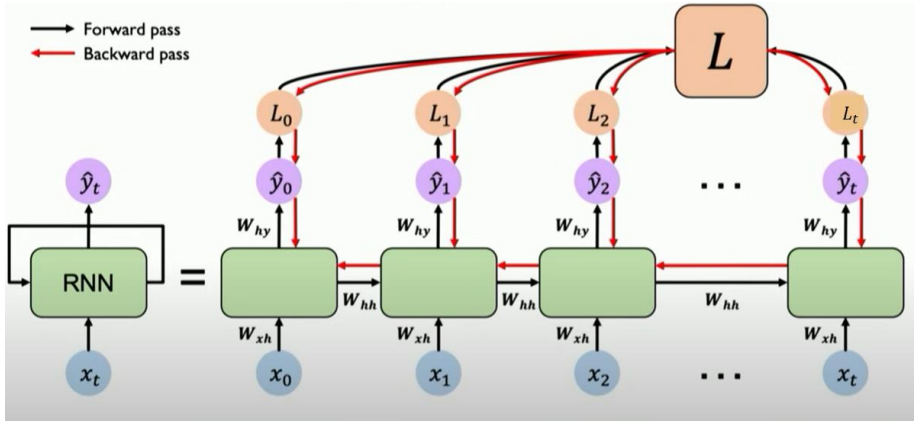
$$\frac{\partial L}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial h_i}\frac{\partial h_i}{\partial W_{hh}}$$

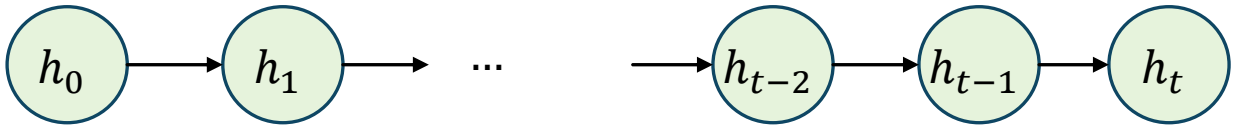
This term is responsible for backpropagation through time

Back Propagation through Time

$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$$

$$\widehat{\boldsymbol{Y}}_\mathbf{t} = \mathbf{H_t W_{hy}} + \mathbf{b_o}$$





Individual time step-wise loss components are L_i

Total Loss

$$L = \frac{1}{t}\sum_{i=1}^{t} L_i$$

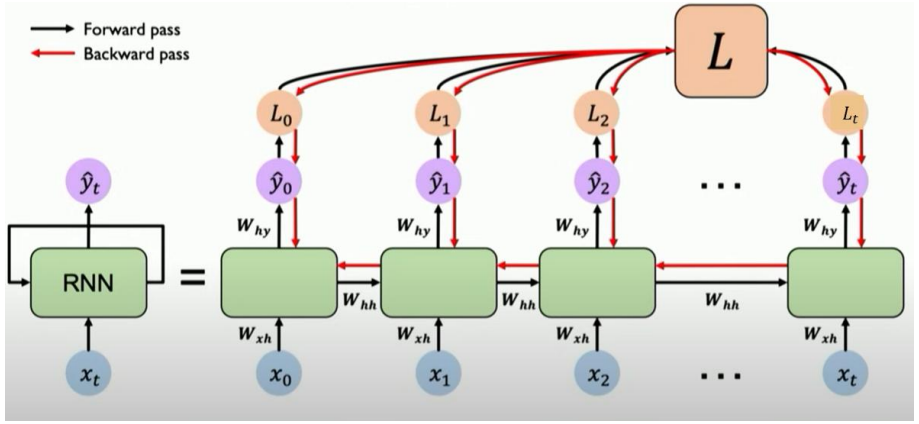
$$\frac{\partial L}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial h_i}\frac{\partial h_i}{\partial W_{hh}}$$

This term is responsible for backpropagation through time

Back Propagation through Time

$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$$

$$\widehat{\boldsymbol{Y}}_\mathbf{t} = \mathbf{H_t W_{hy}} + \mathbf{b_o}$$

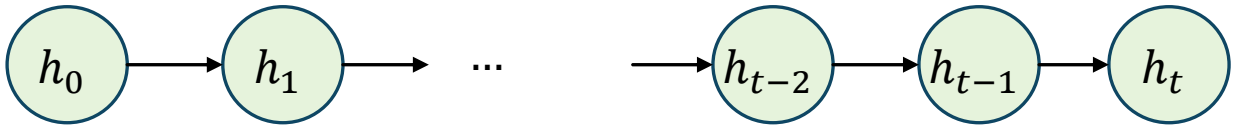


Total Loss

$$L = \frac{1}{t}\sum_{i=1}^{t} L_i$$

$$\frac{\partial L}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial h_i}\frac{\partial h_i}{\partial W_{hh}}$$

For $i = t$, let us try to calculate $\frac{\partial L_t}{\partial W_{hh}}$

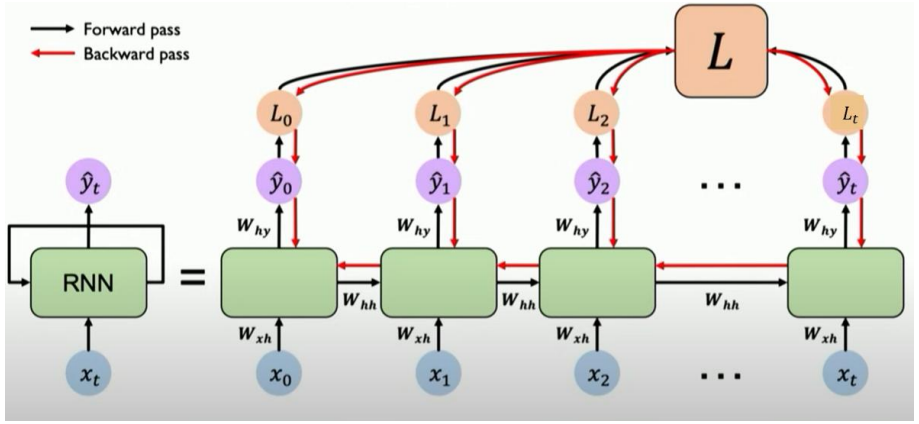


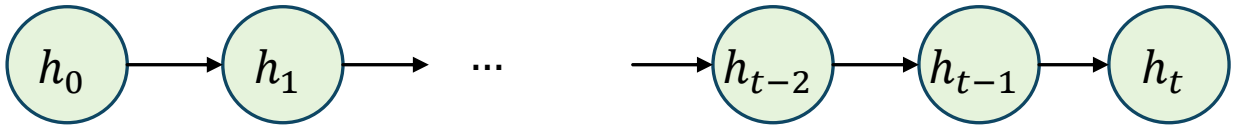
$$\frac{\partial L_t}{\partial W_{hh}} = \frac{\partial L_t}{\partial h_t}\frac{\partial h_t}{\partial W_{hh}}$$

Back Propagation through Time

$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$$

$$\hat{\boldsymbol{Y}}_\mathbf{t} = \mathbf{H_t W_{hy}} + \mathbf{b_o}$$





Total Loss

$$L = \frac{1}{t}\sum_{i=1}^{t} L_i$$

$$\frac{\partial L}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial h_i}\frac{\partial h_i}{\partial W_{hh}}$$

For $i = t$, let us try to calculate $\frac{\partial L_t}{\partial W_{hh}}$

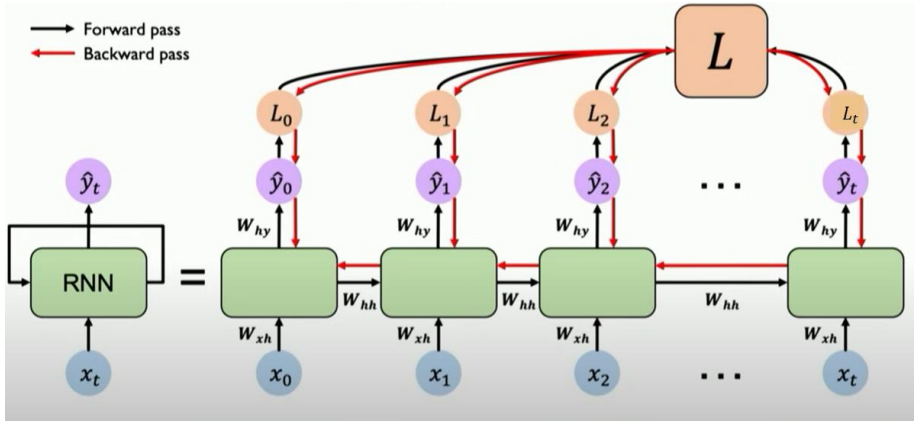
$$\frac{\partial L_t}{\partial W_{hh}} = \frac{\partial L_t}{\partial h_t}\frac{\partial h_t}{\partial W_{hh}}$$

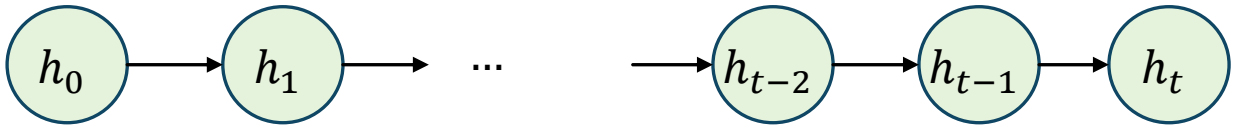
Easy to calculate

Back Propagation through Time

$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$$

$$\widehat{\boldsymbol{Y}}_\mathbf{t} = \mathbf{H_t W_{hy}} + \mathbf{b_o}$$





Total Loss

$$L = \frac{1}{t}\sum_{i=1}^{t} L_i$$

$$\frac{\partial L}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial h_i}\frac{\partial h_i}{\partial W_{hh}}$$

For $i = t$, let us try to calculate $\frac{\partial L_t}{\partial W_{hh}}$

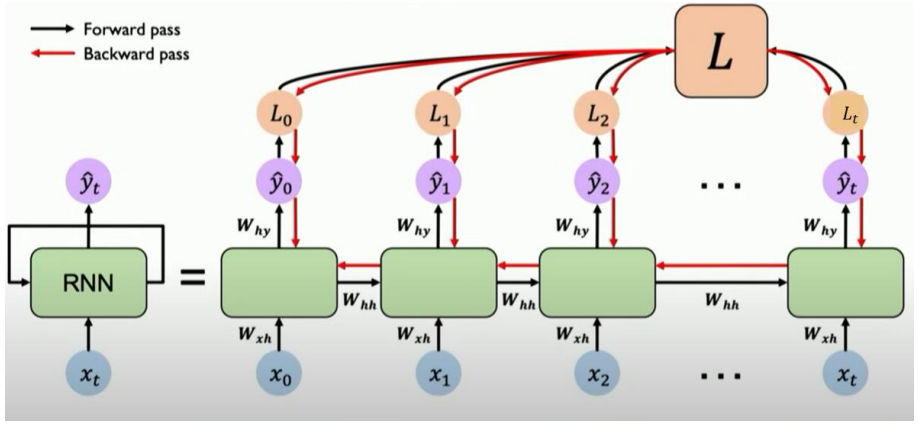
$$\frac{\partial L_t}{\partial W_{hh}} = \frac{\partial L_t}{\partial h_t}\frac{\boldsymbol{\partial h_t}}{\boldsymbol{\partial W_{hh}}}$$

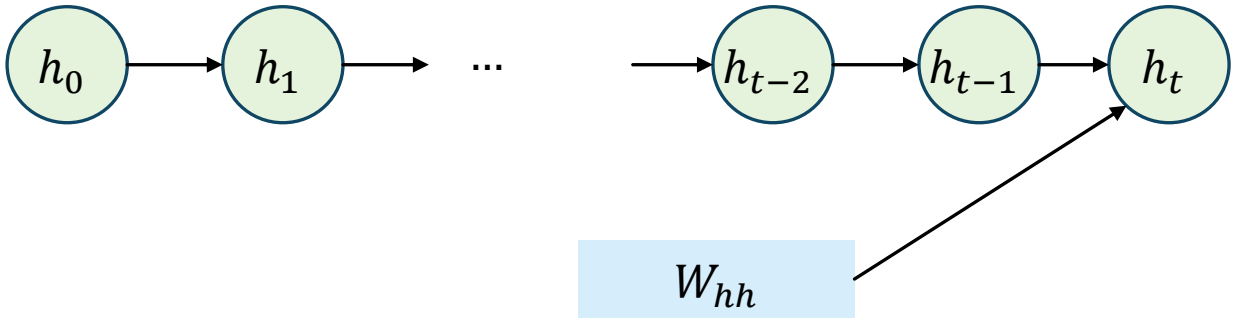
h_t depends on W_{hh}

Back Propagation through Time

$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$$

$$\mathbf{\hat{Y}_t} = \mathbf{H_t W_{hy}} + \mathbf{b_o}$$





Total Loss

$$L = \frac{1}{t}\sum_{i=1}^{t} L_i$$

$$\frac{\partial L}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial H_i}\frac{\partial H_i}{\partial W_{hh}}$$

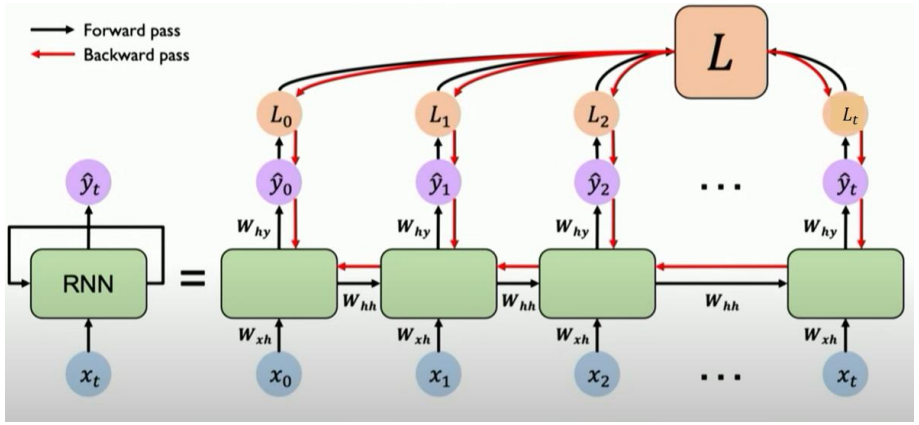
For $i = t$, let us try to calculate $\frac{\partial L_t}{\partial W_{hh}}$

$$\frac{\partial L_t}{\partial W_{hh}} = \frac{\partial L_t}{\partial h_t}\frac{\boldsymbol{\partial h_t}}{\boldsymbol{\partial W_{hh}}}$$

h_t depends on W_{hh}

$$h_t = \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)$$

Back Propagation through Time



$$h_t = \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)$$

$$\hat{y}_t = h_t W_{hy} + b_o$$

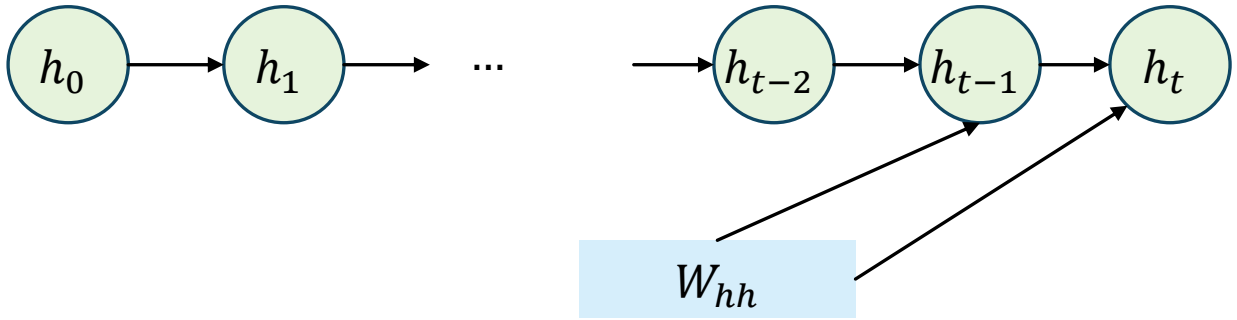
Total Loss

$$L = \frac{1}{t}\sum_{i=1}^{t} L_i$$

$$\frac{\partial L}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial H_i}\frac{\partial H_i}{\partial W_{hh}}$$

For $i = t$, let us try to calculate $\frac{\partial L_t}{\partial W_{hh}}$

$$\frac{\partial L_t}{\partial W_{hh}} = \frac{\partial L_t}{\partial h_t}\frac{\partial h_t}{\partial W_{hh}}$$



h_t depends on W_{hh}

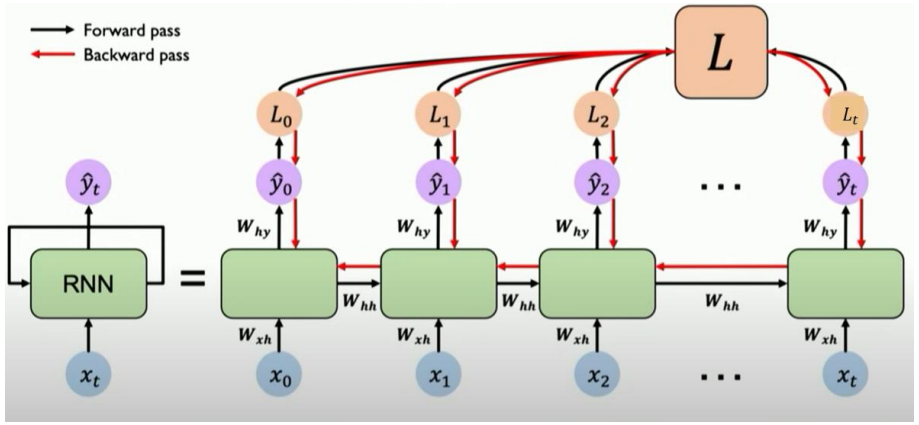
$$h_t = \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)$$

h_{t-1} also depends on W_{hh}. So,

Back Propagation through Time

$$\boldsymbol{h_t = \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)}$$

$$\boldsymbol{\hat{y}_t = h_t W_{hy} + b_o}$$



Total Loss

$$L = \frac{1}{t}\sum_{i=1}^{t} L_i$$

$$\frac{\partial L}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial H_i}\frac{\partial H_i}{\partial W_{hh}}$$

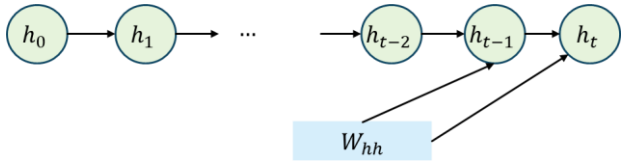
For $i = t$, let us try to calculate $\frac{\partial L_t}{\partial W_{hh}}$

$$\frac{\partial L_t}{\partial W_{hh}} = \frac{\partial L_t}{\partial h_t}\frac{\boldsymbol{\partial h_t}}{\boldsymbol{\partial W_{hh}}}$$

h_t depends on W_{hh}

$$h_t = \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)$$

h_{t-1} also depends on W_{hh}. So,

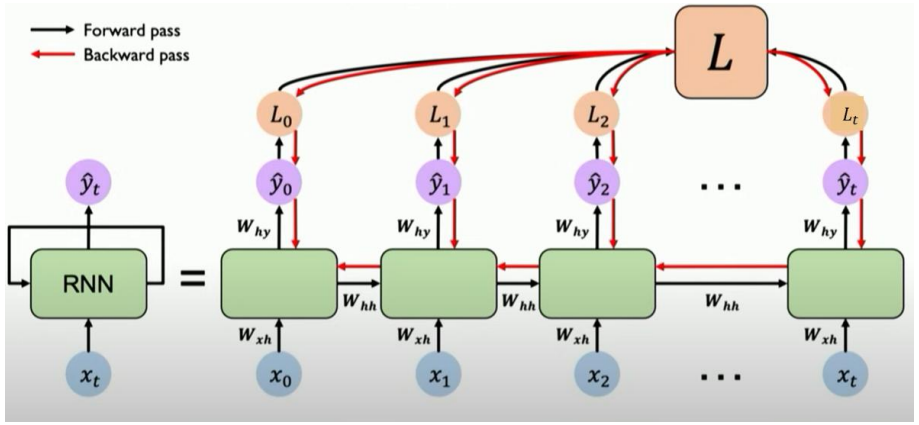


$$\boldsymbol{\frac{\partial h_t}{\partial W_{hh}} = \frac{\partial \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)}{\partial W_{hh}} + \frac{\partial \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial W_{hh}}}$$

Back Propagation through Time

$$\boldsymbol{h_t = \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)}$$

$$\boldsymbol{\hat{y}_t = h_t W_{hy} + b_o}$$



Total Loss

$$L = \frac{1}{t}\sum_{i=1}^{t} L_i$$

$$\frac{\partial L}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial H_i}\frac{\partial H_i}{\partial W_{hh}}$$

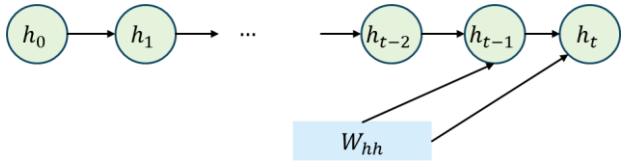
For $i = t$, let us try to calculate $\frac{\partial L_t}{\partial W_{hh}}$

$$\frac{\partial L_t}{\partial W_{hh}} = \frac{\partial L_t}{\partial h_t}\frac{\boldsymbol{\partial h_t}}{\boldsymbol{\partial W_{hh}}}$$

h_t depends on W_{hh}

$$h_t = \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)$$

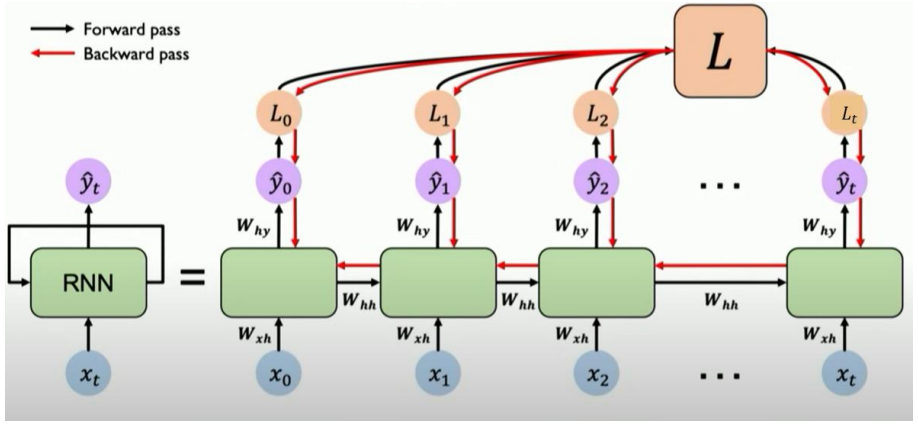
h_{t-1} also depends on W_{hh}. So,

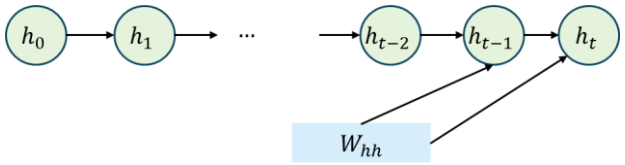


$$\frac{\boldsymbol{\partial h_t}}{\boldsymbol{\partial W_{hh}}} = \frac{\boldsymbol{\partial \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)}}{\boldsymbol{\partial W_{hh}}} + \frac{\boldsymbol{\partial \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)}}{\boldsymbol{\partial h_{t-1}}}\frac{\boldsymbol{\partial h_{t-1}}}{\boldsymbol{\partial W_{hh}}}$$

We may also expand it

Back Propagation through Time





$$\boldsymbol{h_t = \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)}$$

$$\boldsymbol{\hat{y}_t = h_t W_{hy} + b_o}$$

Total Loss

$$L = \frac{1}{t}\sum_{i=1}^{t} L_i$$

$$\frac{\partial L}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial H_i}\frac{\partial H_i}{\partial W_{hh}}$$

For $i = t$, let us try to calculate $\frac{\partial L_t}{\partial W_{hh}}$

$$\frac{\partial L_t}{\partial W_{hh}} = \frac{\partial L_t}{\partial h_t}\frac{\boldsymbol{\partial h_t}}{\boldsymbol{\partial W_{hh}}}$$

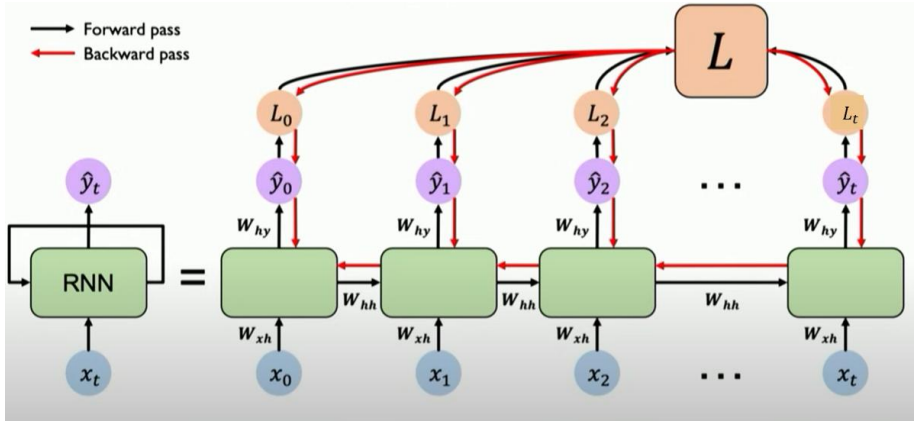
h_t depends on W_{hh}

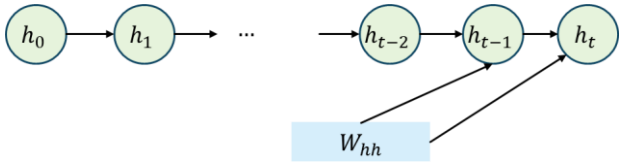
$h_t = \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)$

h_{t-1} also depends on W_{hh}. So,

$$\boldsymbol{\frac{\partial h_t}{\partial W_{hh}} = \frac{\partial \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)}{\partial W_{hh}} + \sum_{i=1}^{t-1}\left(\prod_{j=i+1}^{t} \frac{\partial \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)}{\partial h_{j-1}}\right)\frac{\partial \phi(x_i W_{xh} + h_{i-1} W_{hh} + b_h)}{\partial W_{hh}}}$$

Back Propagation through Time





$$\boldsymbol{h_t = \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)}$$

$$\boldsymbol{\hat{y}_t = h_t W_{hy} + b_o}$$

Total Loss

$$L = \frac{1}{t}\sum_{i=1}^{t} L_i$$

$$\frac{\partial L}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t}\frac{\partial L_i}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t}\frac{\partial L_i}{\partial H_i}\frac{\partial H_i}{\partial W_{hh}}$$

For $i = t$, let us try to calculate $\frac{\partial L_t}{\partial W_{hh}}$

$$\frac{\partial L_t}{\partial W_{hh}} = \frac{\partial L_t}{\partial h_t}\frac{\boldsymbol{\partial h_t}}{\boldsymbol{\partial W_{hh}}}$$

h_t depends on W_{hh}

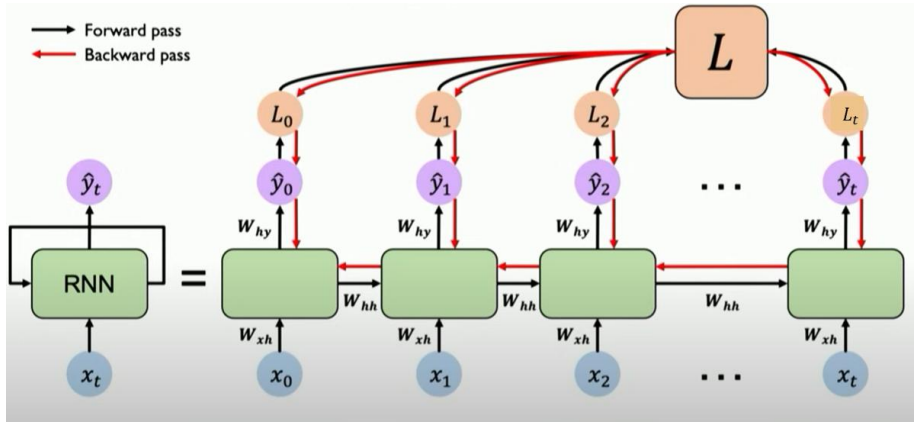
$$h_t = \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)$$

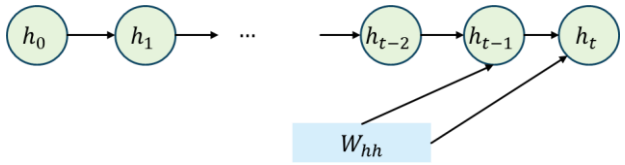
h_{t-1} also depends on W_{hh}. So,

$$\boldsymbol{\frac{\partial h_t}{\partial W_{hh}} = \frac{\partial \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)}{\partial W_{hh}} + \sum_{i=1}^{t-1}\left(\prod_{j=i+1}^{t}\frac{\partial \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)}{\partial h_{j-1}}\right)\frac{\partial \phi(x_i W_{xh} + h_{i-1} W_{hh} + b_h)}{\partial W_{hh}}}$$

Vanishing/ exploding gradient problems

Solution: Truncated BPTT





$$\boldsymbol{h_t = \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)}$$

$$\boldsymbol{\hat{y}_t = h_t W_{hy} + b_o}$$

Total Loss

$$L = \frac{1}{t}\sum_{i=1}^{t} L_i$$

$$\frac{\partial L}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial W_{hh}} = \frac{1}{t}\sum_{i=1}^{t} \frac{\partial L_i}{\partial H_i}\frac{\partial H_i}{\partial W_{hh}}$$

For $i = t$, let us try to calculate $\frac{\partial L_t}{\partial W_{hh}}$

$$\frac{\partial L_t}{\partial W_{hh}} = \frac{\partial L_t}{\partial h_t}\frac{\boldsymbol{\partial h_t}}{\boldsymbol{\partial W_{hh}}}$$

h_t depends on W_{hh}

$$h_t = \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)$$

h_{t-1} also depends on W_{hh}. So,

$$\boldsymbol{\frac{\partial h_t}{\partial W_{hh}} = \frac{\partial \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)}{\partial W_{hh}} + \sum_{i=\tau}^{t-1}\left(\prod_{j=i+1}^{t} \frac{\partial \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)}{\partial h_{j-1}}\right)\frac{\partial \phi(x_i W_{xh} + h_{i-1} W_{hh} + b_h)}{\partial W_{hh}}}$$

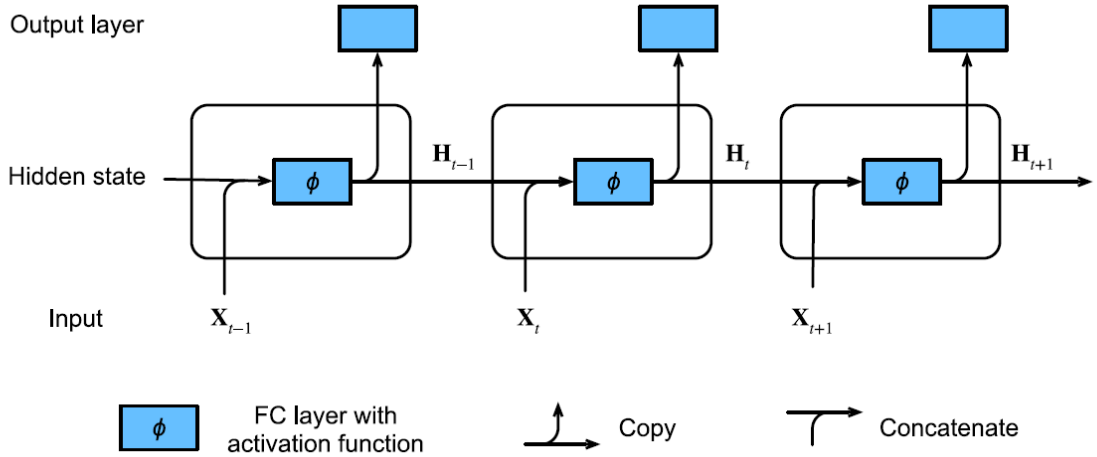
Vanishing/ exploding gradient problems

Challenges in RNN

- Vanishing/exploding gradient
  - Difficult to train
- LSTM, GRU
  - Selective Read, Selective Write, Selective Forget

Selective Read, Selective Write, Selective Forget

- Because of the dependence on previous time steps, the information inside the neural network module in RNN gets morphed over time
- We may not exploit the dependence on all past time steps
- Suppose, $a = 2, b = 10, c = 3, d = 5$
- We have to compute $a(bc + d) + ad$
- If allowed to do 3 operations at a time
- How will I proceed?



Selective Read, Selective Write, Selective Forget

- Suppose, $a = 2, b = 10, c = 3, d = 5$
- We have to compute $a(bc + d) + ac$
- If allowed to do 3 operations at a time
- How will I proceed?

Selective Write

- Suppose, $a = 2, b = 10, c = 3, d = 5$
- We have to compute $a(bc + d) + ac$
- If allowed to do 3 operations at a time
- How will I proceed?

1. $b = 10$
2. $c = 3$
3. $bc = 30$

I won't write $a = 2$ or $d = 5$ because that's not going to help at the present moment

Selective Read

- Suppose, $a = 2, b = 10, c = 3, d = 5$
- We have to compute $a(bc + d) + ac$
- If allowed to do 3 operations at a time
- How will I proceed?

1. $b = 10$
2. $c = 3$
3. $bc = 30$

The next step is computing $bc + d$

For that, do I need to read everything from the above steps?

Selective Read

- Suppose, $a = 2, b = 10, c = 3, d = 5$
- We have to compute $a(bc + d) + ac$
- If allowed to do 3 operations at a time
- How will I proceed?

1. $b = 10$
2. $c = 3$
3. $bc = 30$

The next step is computing $bc + d$

For that, do I need to read everything from the above steps? **No**

I just need to read $bc = 30$

Selective Forget

- Suppose, $a = 2, b = 10, c = 3, d = 5$
- We have to compute $a(bc + d) + ac$
- If allowed to do 3 operations at a time
- How will I proceed?

1. $b = 10$
2. $c = 3$
3. $bc = 30$

The next step is computing $bc + d$

For that, I need to write $d = 5$

To do this writing, I need to forget some information that is not required at present

For example, $b = 10$

Selective Forget

- Suppose, $a = 2, b = 10, c = 3, d = 5$
- We have to compute $a(bc + d) + ac$
- If allowed to do 3 operations at a time
- How will I proceed?

1. $d = 5$
2. $c = 3$
3. $bc = 30$

The next step is computing $bc + d$

For that, I need to write $d = 5$

To do this writing, I need to forget some information that is not required at present

For example, $b = 10$

Next Step

- Suppose, $a = 2, b = 10, c = 3, d = 5$
- We have to compute $a(bc + d) + ac$
- If allowed to do 3 operations at a time
- How will I proceed?

1. $d = 5$
2. $c = 3$ Forget and compute $bc + d$
3. $bc = 30$

The next step is computing $bc + d$

For that, I need to write $d = 5$

To do this writing, I need to forget some information that is not required at present

For example, $b = 10$

Next Step

- Suppose, $a = 2, b = 10, c = 3, d = 5$
- We have to compute $a(bc + d) + ac$
- If allowed to do 3 operations at a time
- How will I proceed?

1. $d = 5$
2. $bc + d = 35$
3. $bc = 30$

Next Step

- Suppose, $a = 2, b = 10, c = 3, d = 5$
- We have to compute $a(bc + d) + ac$
- If allowed to do 3 operations at a time
- How will I proceed?

1. $d = 5$
2. $bc + d = 35$
3. $a = 2$

Next Step

- Suppose, $a = 2, b = 10, c = 3, d = 5$
- We have to compute $a(bc + d) + ac$
- If allowed to do 3 operations at a time
- How will I proceed?

1. $a(bc + d) = 70$
2. $bc + d = 35$
3. $a = 2$

Next Step

- Suppose, $a = 2, b = 10, c = 3, d = 5$
- We have to compute $a(bc + d) + ac$
- If allowed to do 3 operations at a time
- How will I proceed?

1. $a(bc + d) = 70$
2. $c = 3$
3. $a = 2$

Next Step

- Suppose, $a = 2, b = 10, c = 3, d = 5$
- We have to compute $a(bc + d) + ac$
- If allowed to do 3 operations at a time
- How will I proceed?

1. $a(bc + d) = 70$
2. $c = 3$
3. $ac = 6$

Next Step

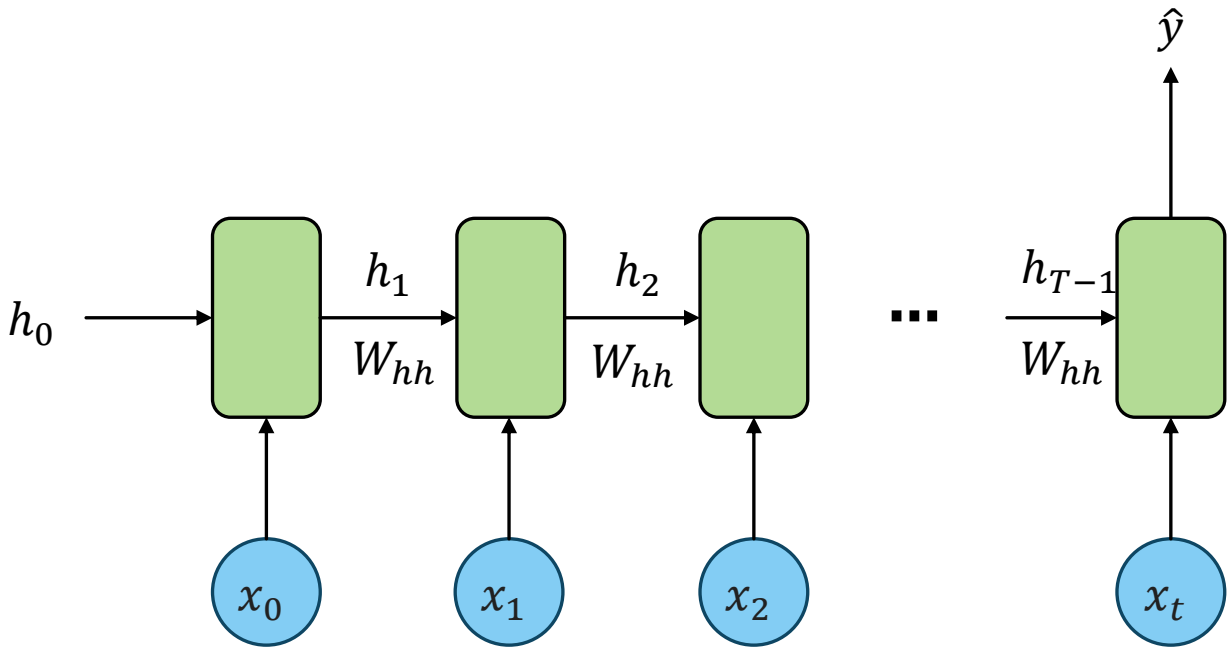
- Suppose, $a = 2, b = 10, c = 3, d = 5$
- We have to compute $a(bc + d) + ac$
- If allowed to do 3 operations at a time
- How will I proceed?

1. $a(bc + d) = 70$
2. $\boldsymbol{a(bc + d) + ac = 76}$
3. $ac = 6$

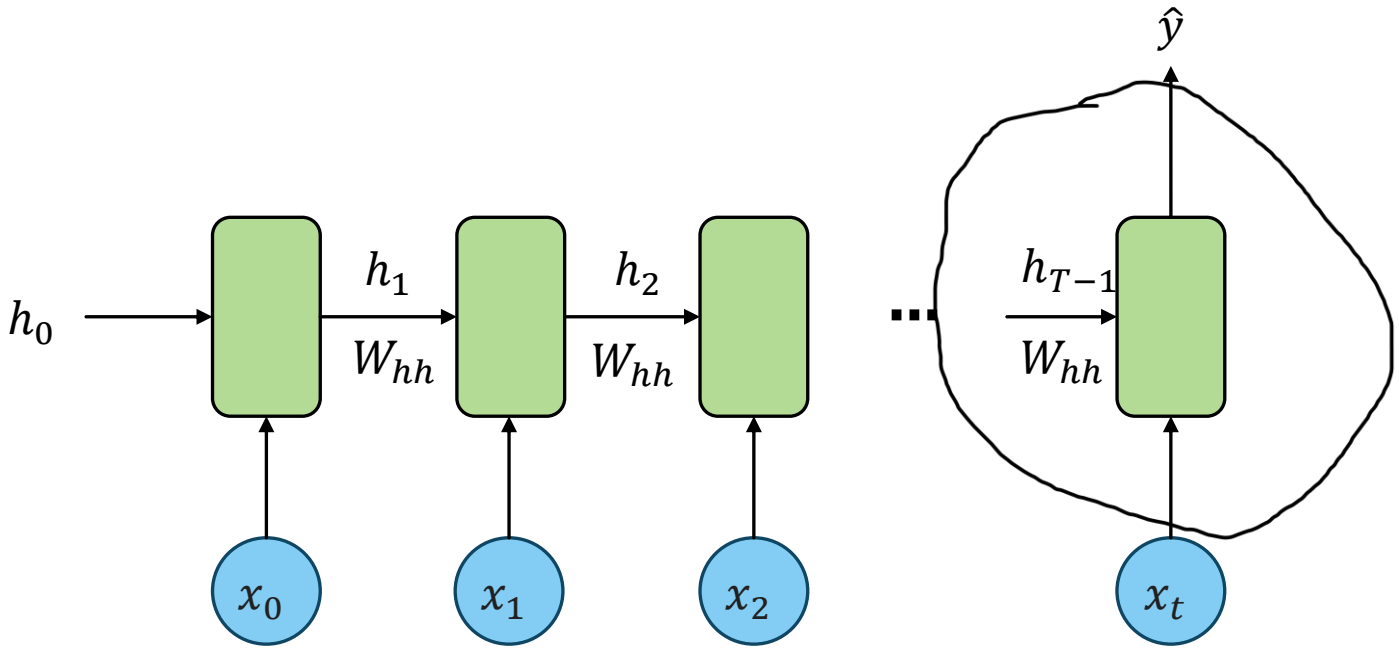
Next Step

- Our brain also selectively reads, writes, and forgets
- RNNs should also selectively read, write and forget

Modifying the RNN for Selective Read, Write and Forget

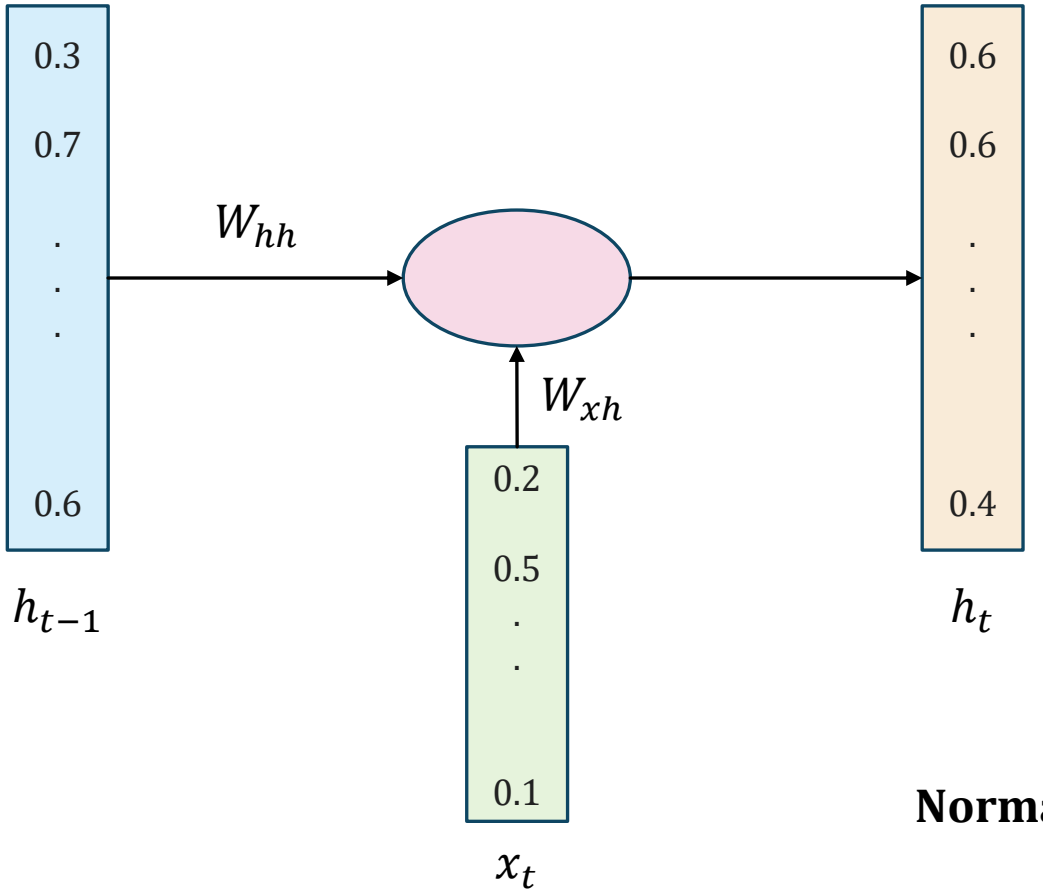


Modifying the RNN for Selective Read, Write and Forget



Let us focus on this segment

Modifying the RNN for Selective Write

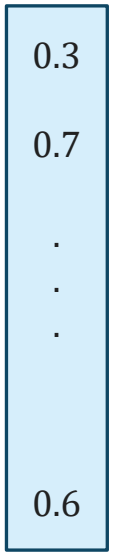


$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$$

$$\widehat{\boldsymbol{Y}}_\mathbf{t} = \mathbf{H_t W_{hy}} + \mathbf{b_o}$$

Normal RNN

Modifying the RNN for Selective Write



h_{t-1}

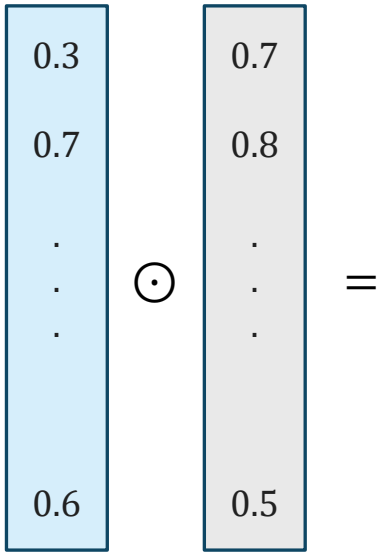
RNN

$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$$

$$\widehat{\boldsymbol{Y}}_\mathbf{t} = \mathbf{H_t W_{hy}} + \mathbf{b_o}$$

Now, I want to take a fraction of h_{t-1} to construct h_t

Modifying the RNN for Selective Write



RNN

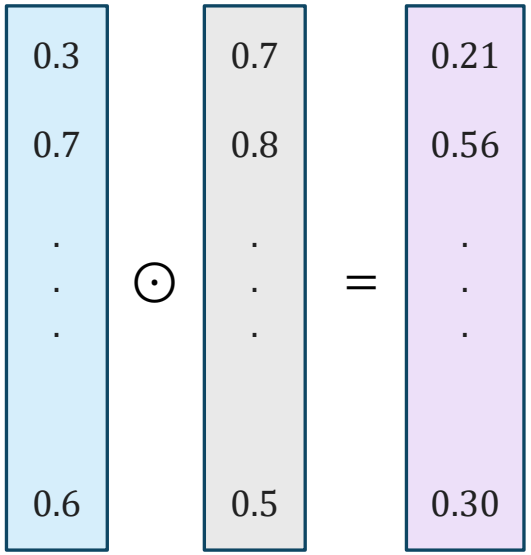
$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$$

$$\widehat{\boldsymbol{Y}}_\mathbf{t} = \mathbf{H_t W_{hy}} + \mathbf{b_o}$$

Now, I want to take a fraction of h_{t-1} to construct h_t

For that, we introduce a term by term multiplier a_{t-1} for h_{t-1}

Modifying the RNN for Selective Write



RNN

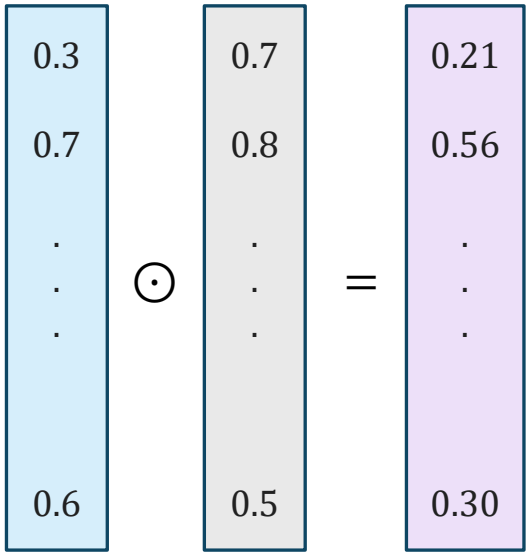
$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$$

$$\widehat{\boldsymbol{Y}}_\mathbf{t} = \mathbf{H_t W_{hy}} + \mathbf{b_o}$$

Now, I want to take a fraction of h_{t-1} to construct h_t

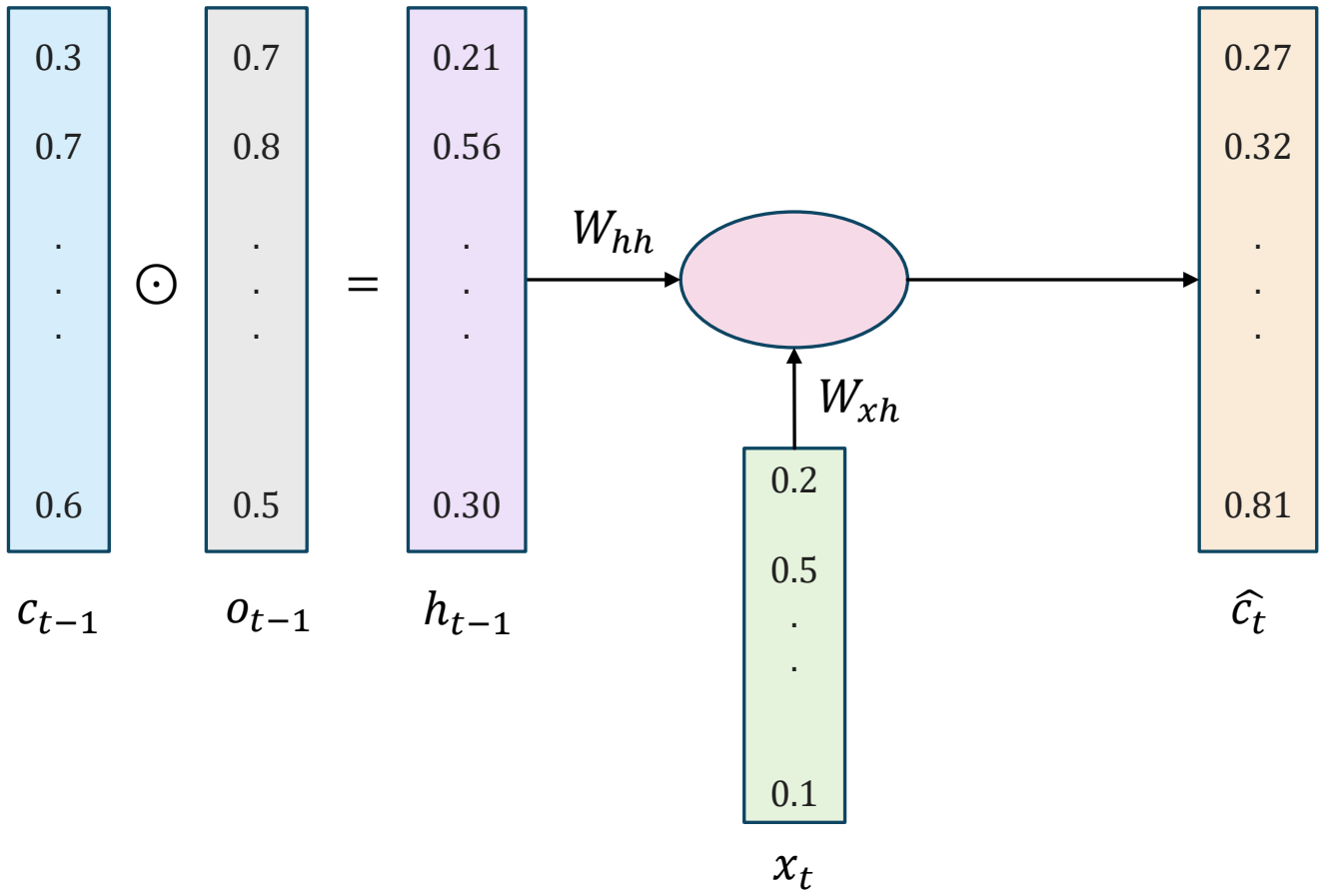
For that, we introduce a term by term multiplier a_{t-1} for h_{t-1}

Modifying the RNN for Selective Write



We now change the names of the variables

Modifying the RNN for Selective Write

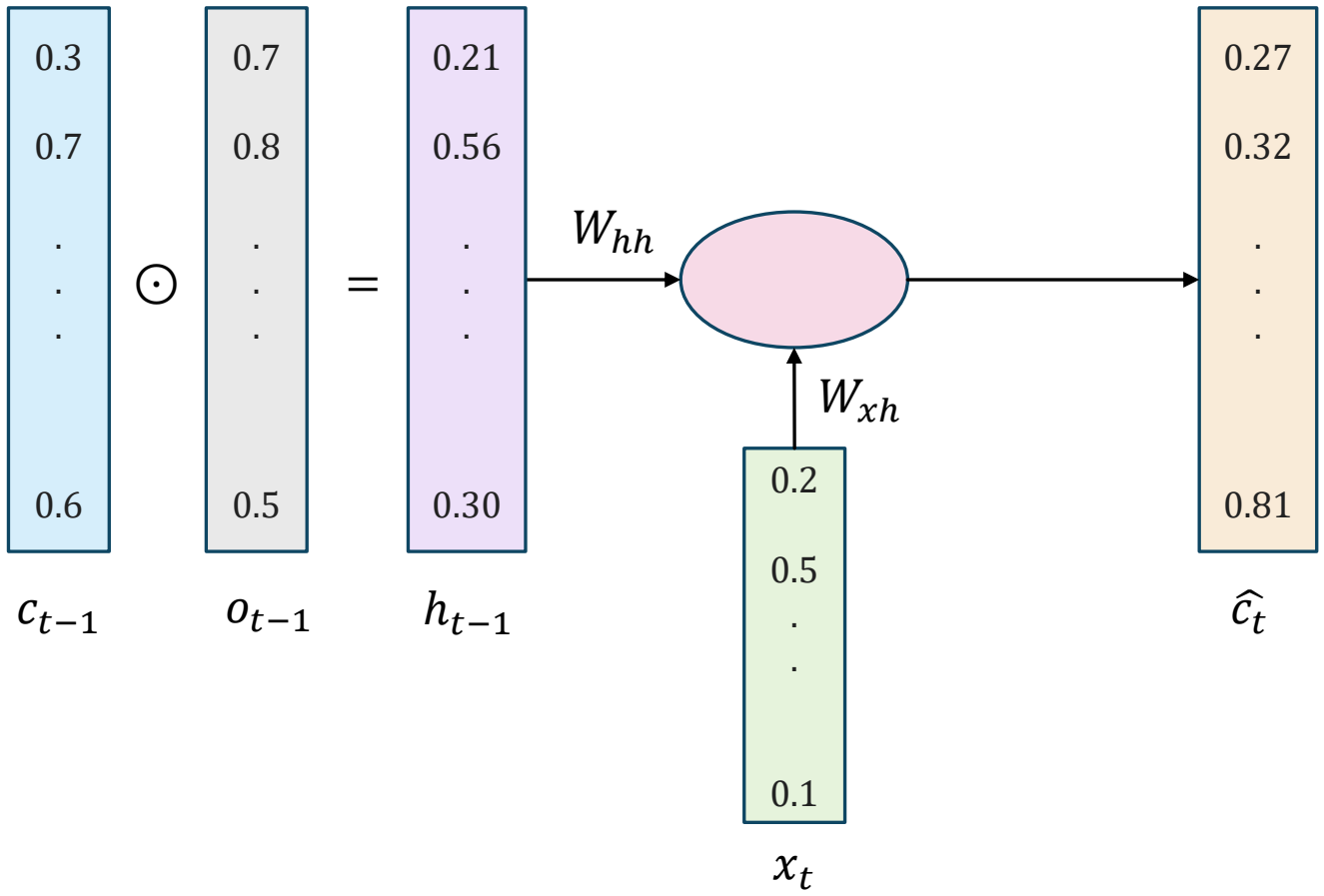


RNN

$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$$

$$\widehat{\boldsymbol{Y}}_\mathbf{t} = \mathbf{H_t W_{hy}} + \mathbf{b_o}$$

Modifying the RNN for Selective Write



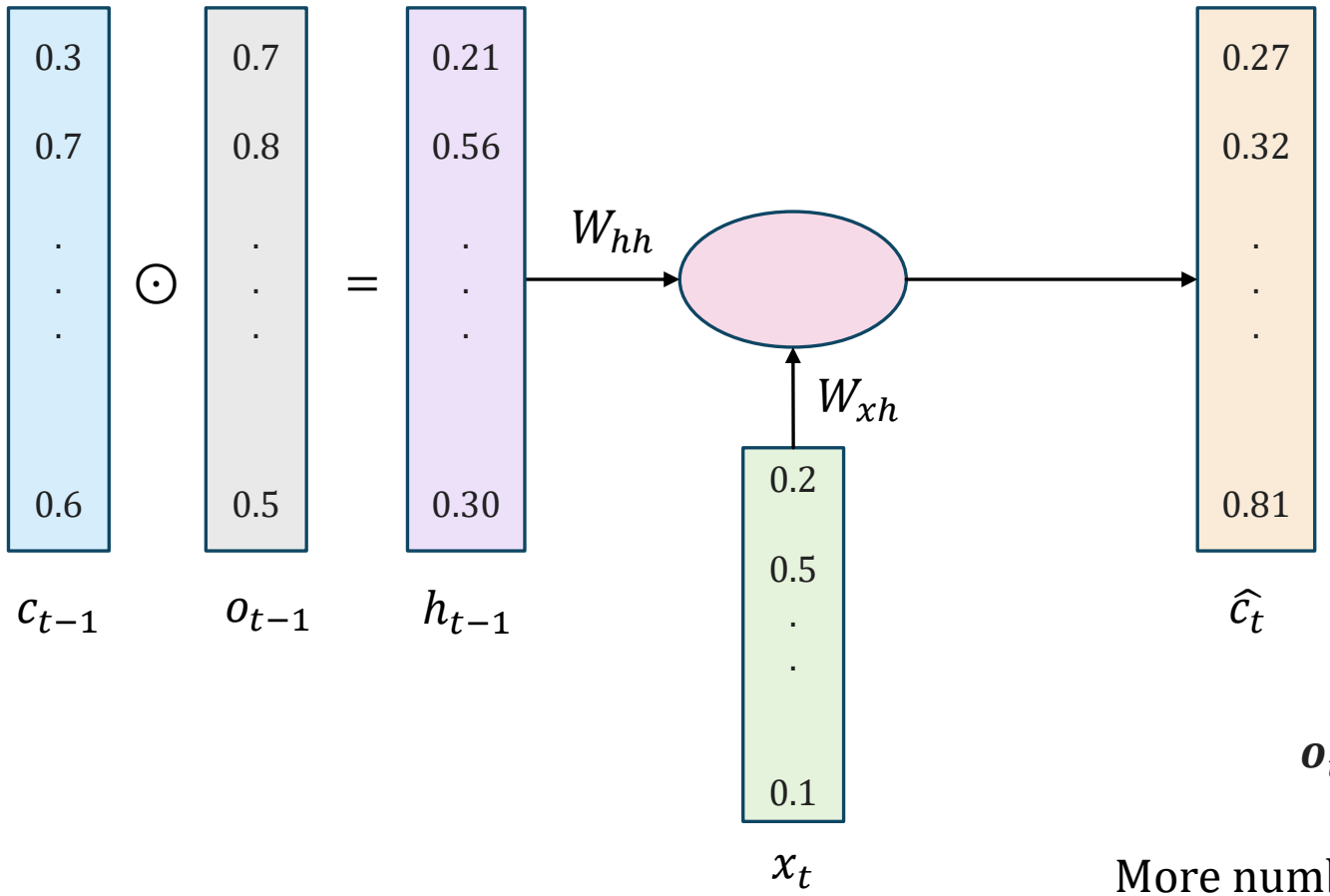
RNN

$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$$

$$\widehat{\boldsymbol{Y}}_\mathbf{t} = \mathbf{H_t W_{hy}} + \mathbf{b_o}$$

How to get the multiplier o_{t-1}?

Modifying the RNN for Selective Write



RNN

$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$$

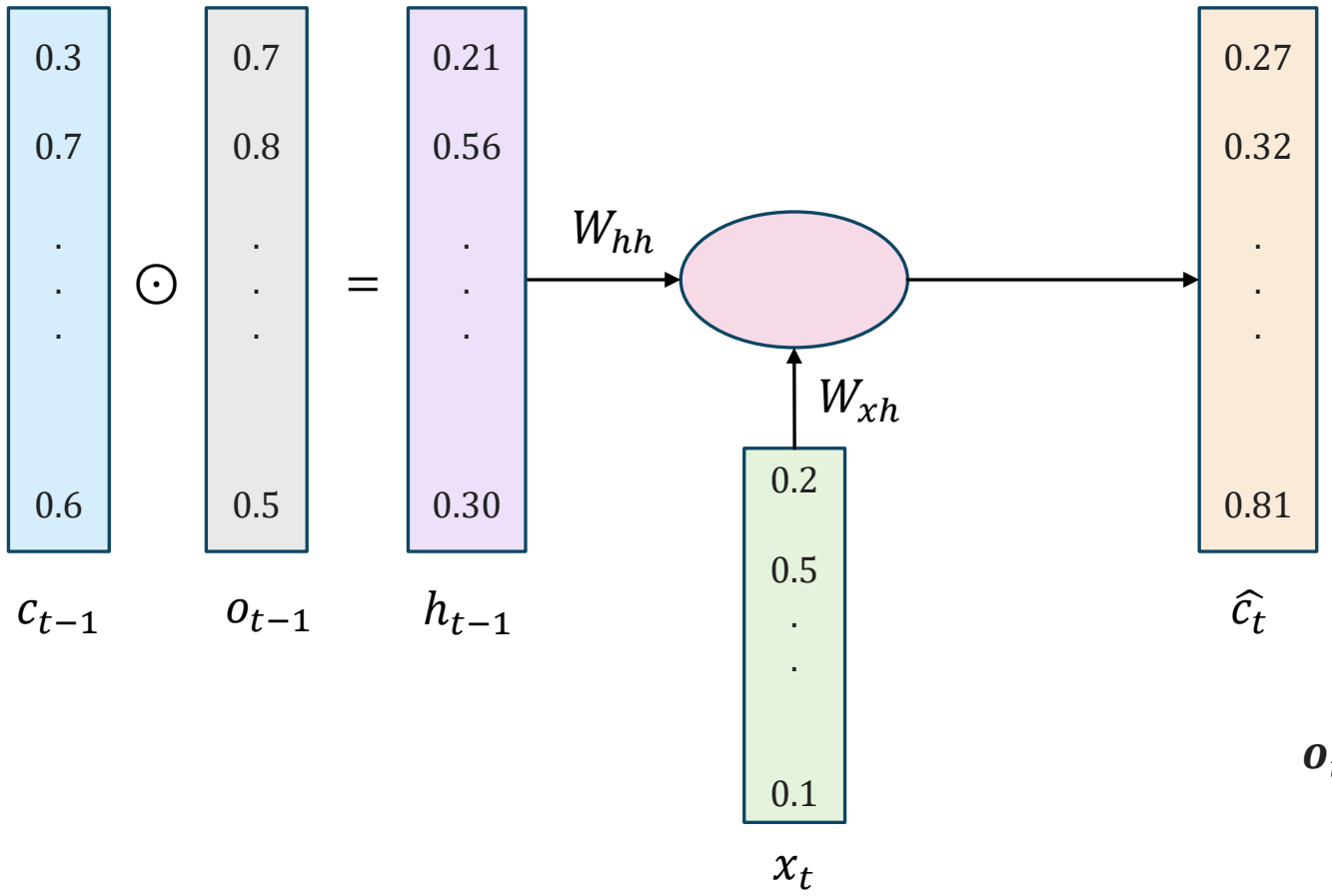
$$\hat{\boldsymbol{Y}}_\mathbf{t} = \mathbf{H_t W_{hy}} + \mathbf{b_o}$$

How to get the multiplier o_{t-1}?

$$\boldsymbol{o_{t-1}} = \boldsymbol{\sigma}(\boldsymbol{h_{t-2} W_a} + \boldsymbol{x_{t-1} W_{ax}} + \boldsymbol{b_a})$$

More number of learnable parameters

Modifying the RNN for Selective Write



RNN

$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$$

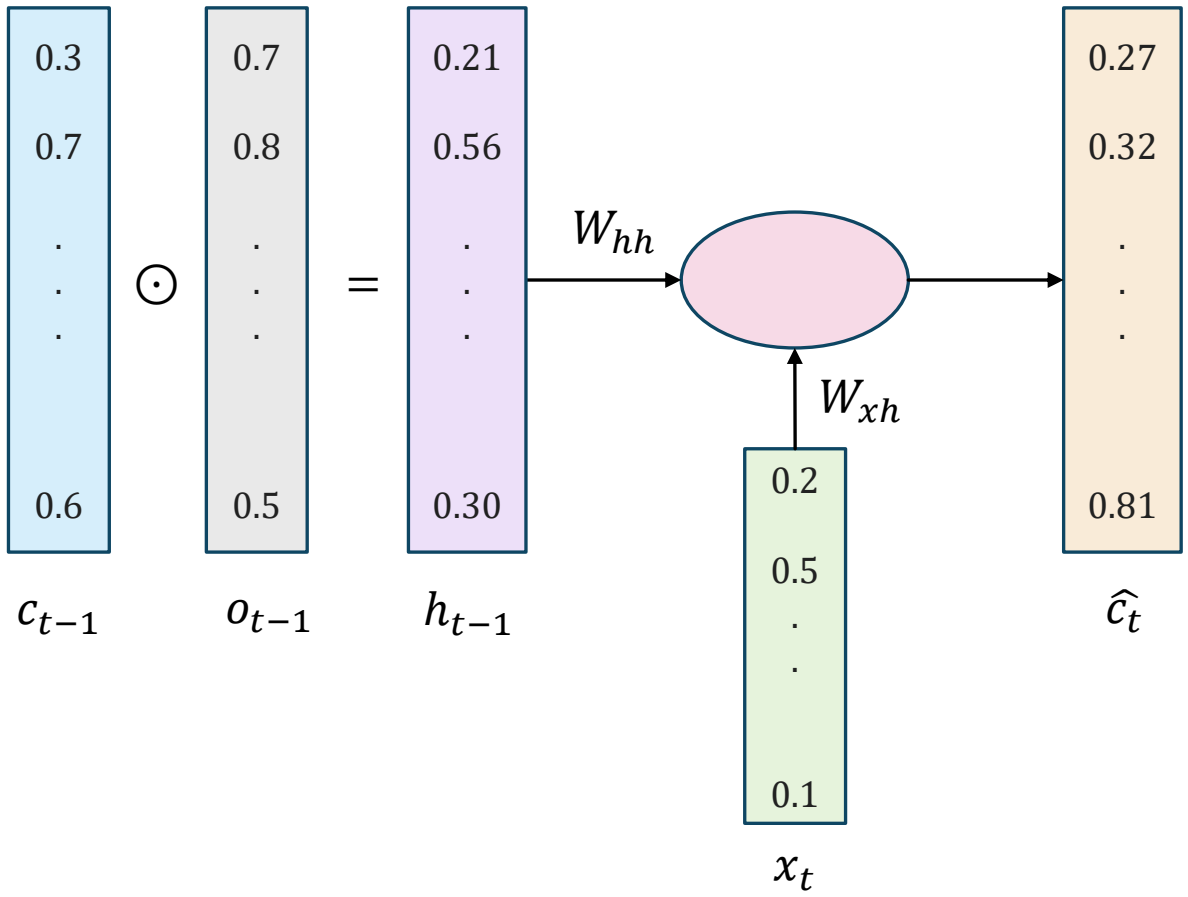
$$\widehat{\boldsymbol{Y}}_\mathbf{t} = \mathbf{H_t W_{hy}} + \mathbf{b_o}$$

How to get the multiplier o_{t-1}?

$$\boldsymbol{o_{t-1}} = \boldsymbol{\sigma}(\boldsymbol{h_{t-2} W_a} + \boldsymbol{x_{t-1} W_{ax}} + \boldsymbol{b_a})$$

$$\boldsymbol{h_{t-1}} = \boldsymbol{\sigma_c}(\boldsymbol{c_{t-1}}) \odot \boldsymbol{o_{t-1}}$$

Modifying the RNN for Selective Read



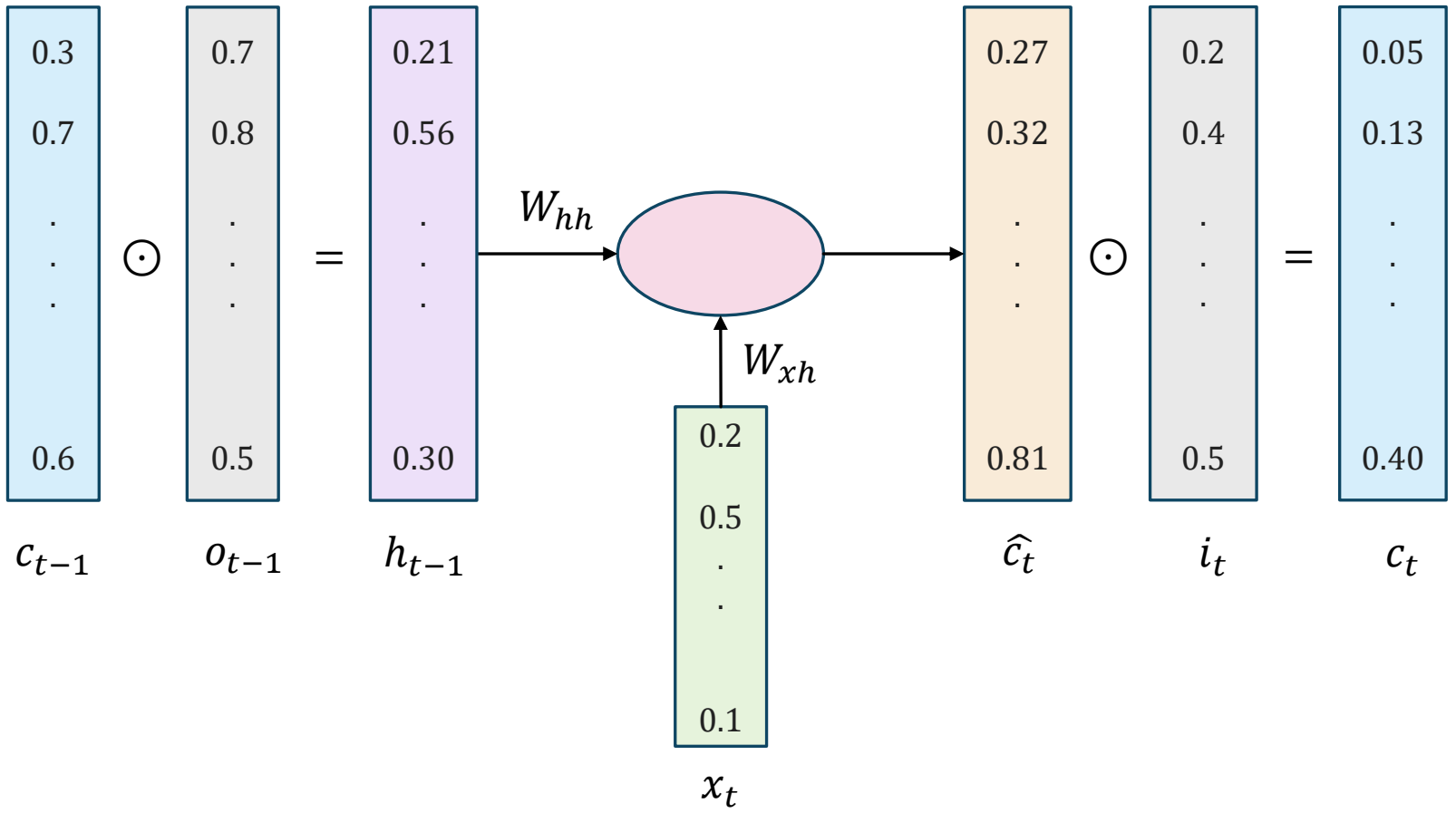
Selectively read from $\hat{c}_t$ to get h_t

RNN

$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$$

$$\widehat{\boldsymbol{Y}}_\mathbf{t} = \mathbf{H_t W_{hy}} + \mathbf{b_o}$$

Modifying the RNN for Selective Read



Selectively read from $\hat{c}_t$ to get c_t

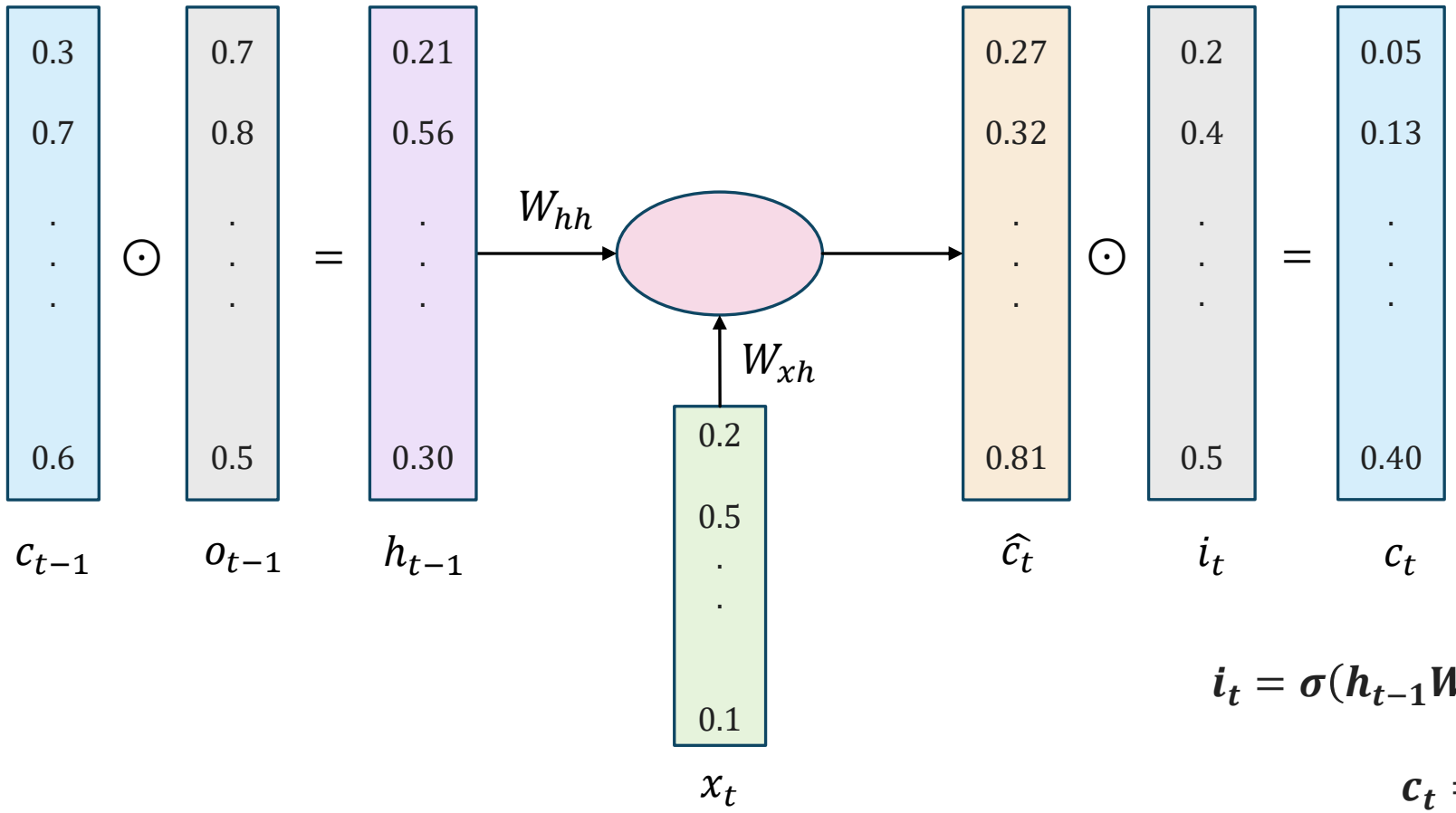
Use a multiplier

RNN

$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$$

$$\widehat{\boldsymbol{Y}}_\mathbf{t} = \mathbf{H_t W_{hy}} + \mathbf{b_o}$$

Modifying the RNN for Selective Read



Selectively read from $\hat{c}_t$ to get c_t

Use a multiplier

How to get the multiplier?

$$\boldsymbol{i_t = \sigma(h_{t-1}W_m + x_tW_{mx} + b_m)}$$

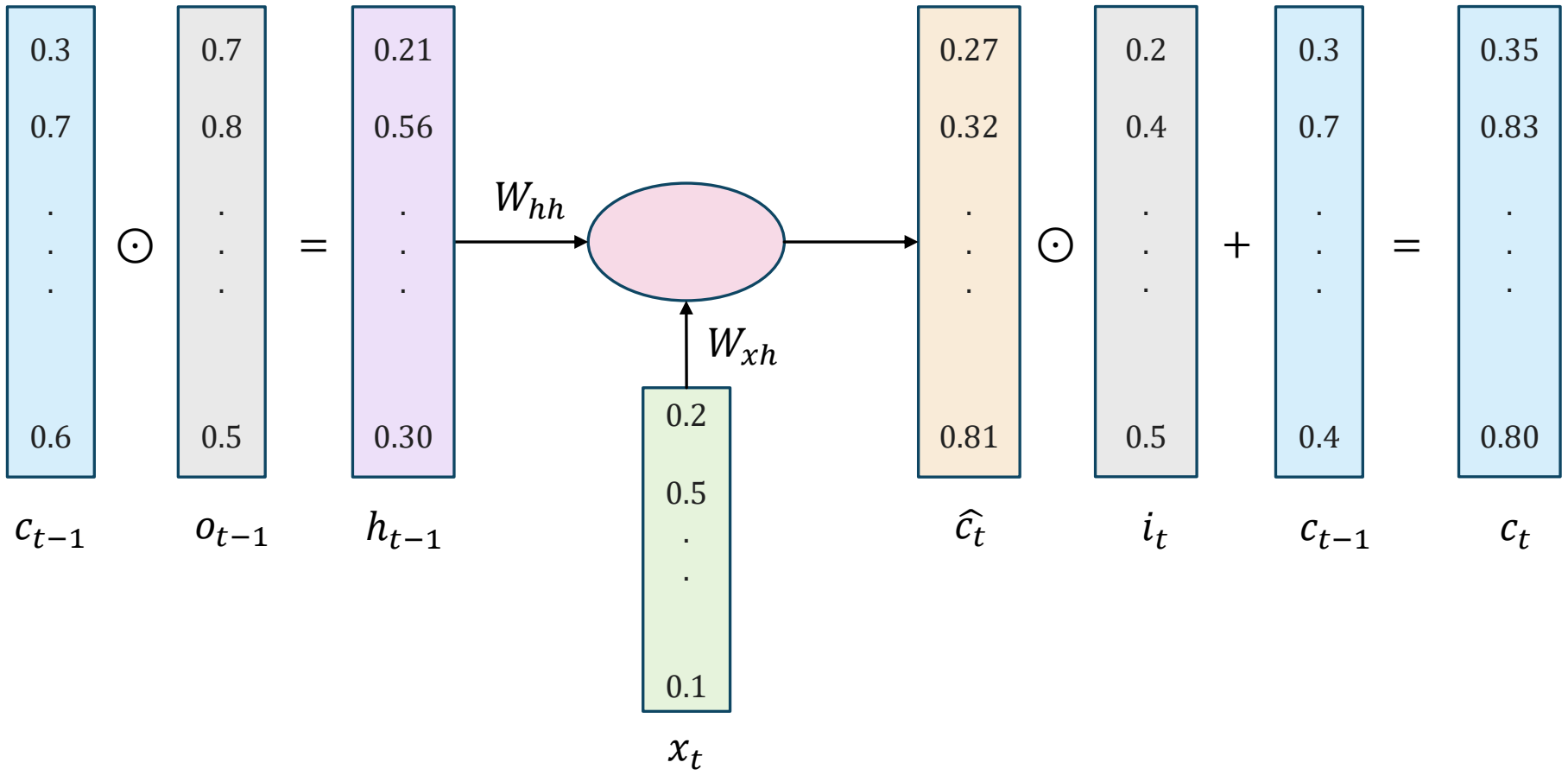
$$\boldsymbol{c_t = \hat{c}_t \odot i_t}$$

RNN

$$\mathbf{H_t = \phi(X_tW_{xh} + H_{t-1}W_{hh} + b_h)}$$

$$\mathbf{\hat{Y}_t = H_tW_{hy} + b_o}$$

Modifying the RNN for Selective Forget



How to combine c_{t-1}, $\hat{c}_t$, and i_t to get c_t

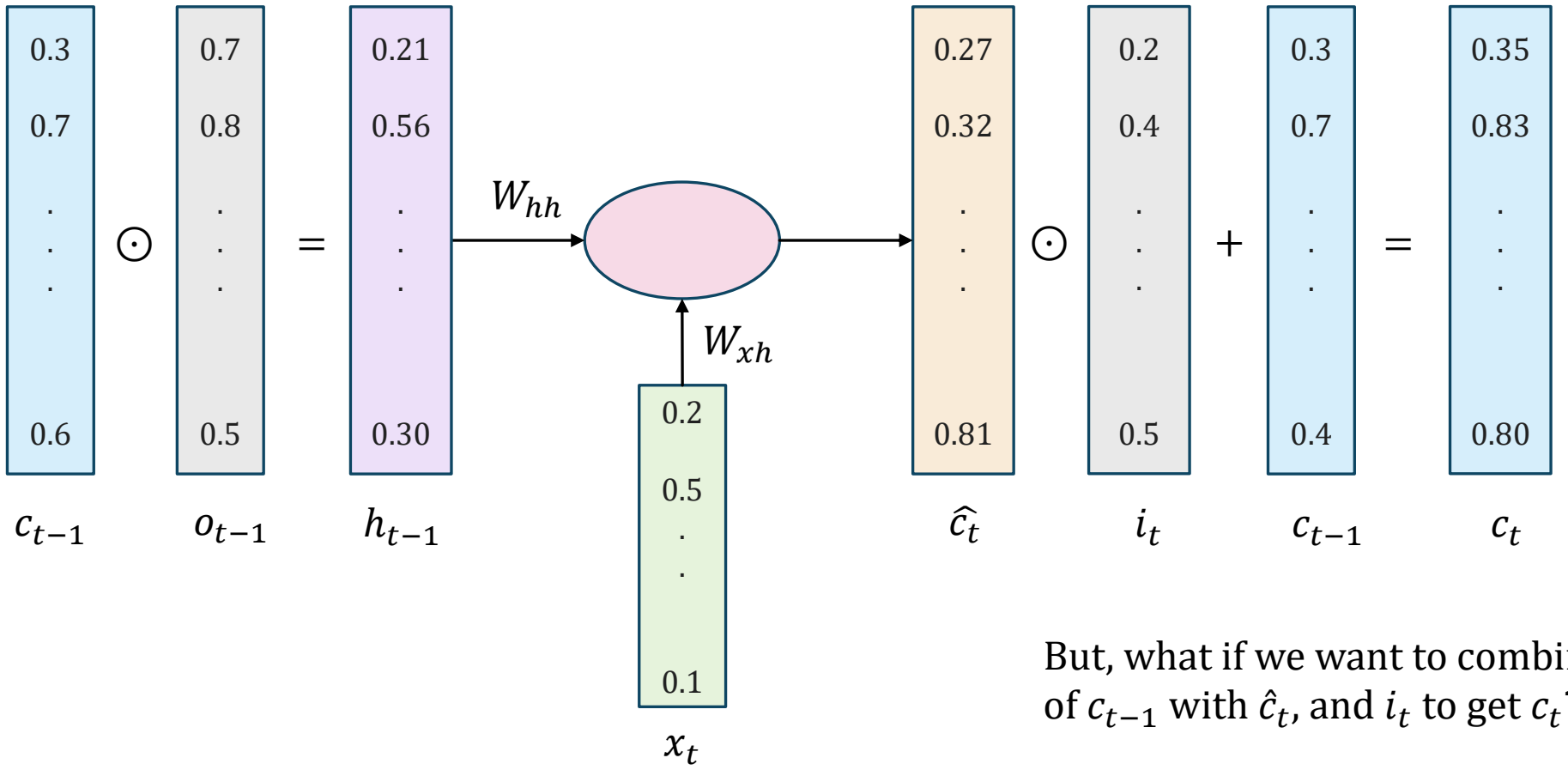
$$\boldsymbol{c_t = \hat{c}_t \odot i_t + c_{t-1}}$$

RNN

$$\mathbf{H_t = \phi(X_t W_{xh} + H_{t-1} W_{hh} + b_h)}$$

$$\mathbf{\hat{Y}_t = H_t W_{hy} + b_o}$$

Modifying the RNN for Selective Forget



How to combine c_{t-1}, $\hat{c}_t$, and i_t to get c_t

$$\boldsymbol{c_t = \hat{c}_t \odot i_t + c_{t-1}}$$

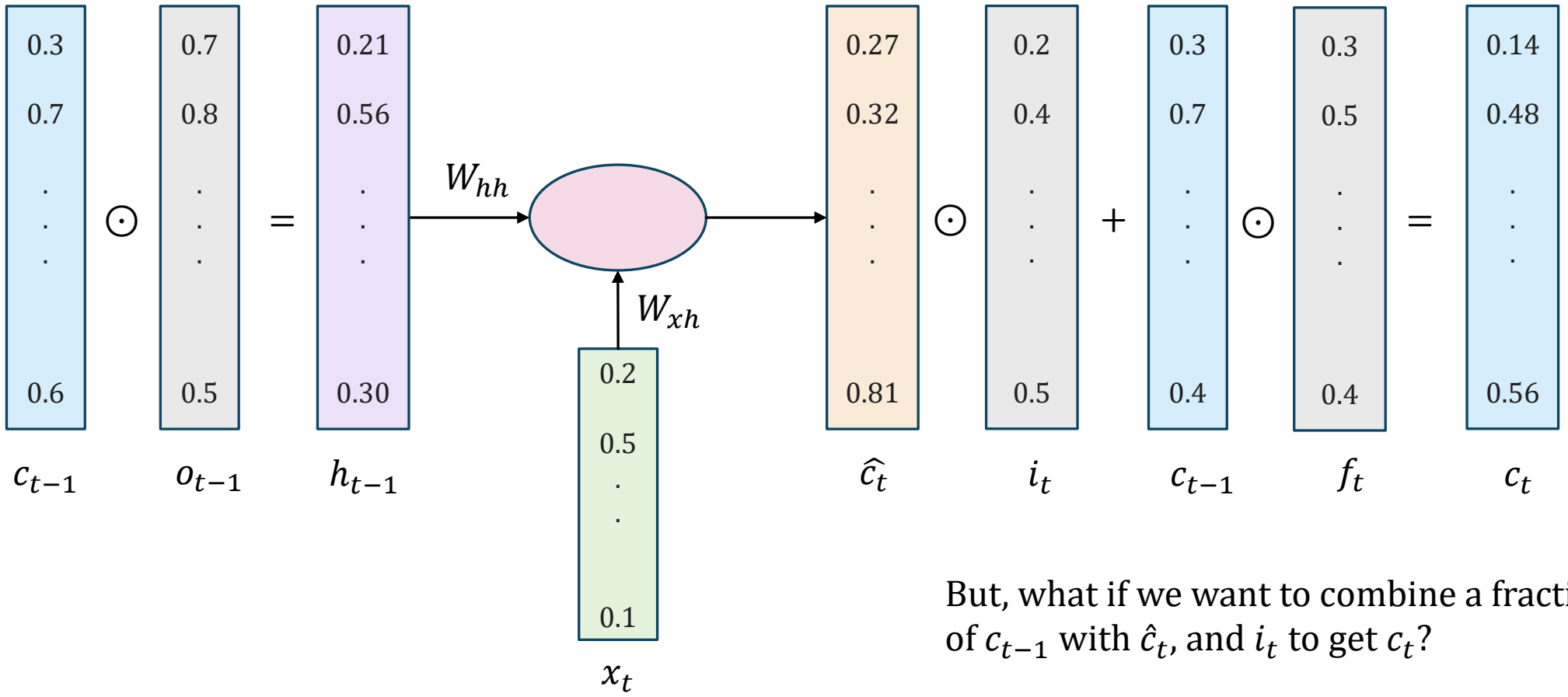
But, what if we want to combine a fraction of c_{t-1} with $\hat{c}_t$, and i_t to get c_t?

RNN

$$\mathbf{H_t = \phi(X_t W_{xh} + H_{t-1} W_{hh} + b_h)}$$

$$\mathbf{\hat{Y}_t = H_t W_{hy} + b_o}$$

Modifying the RNN for Selective Forget



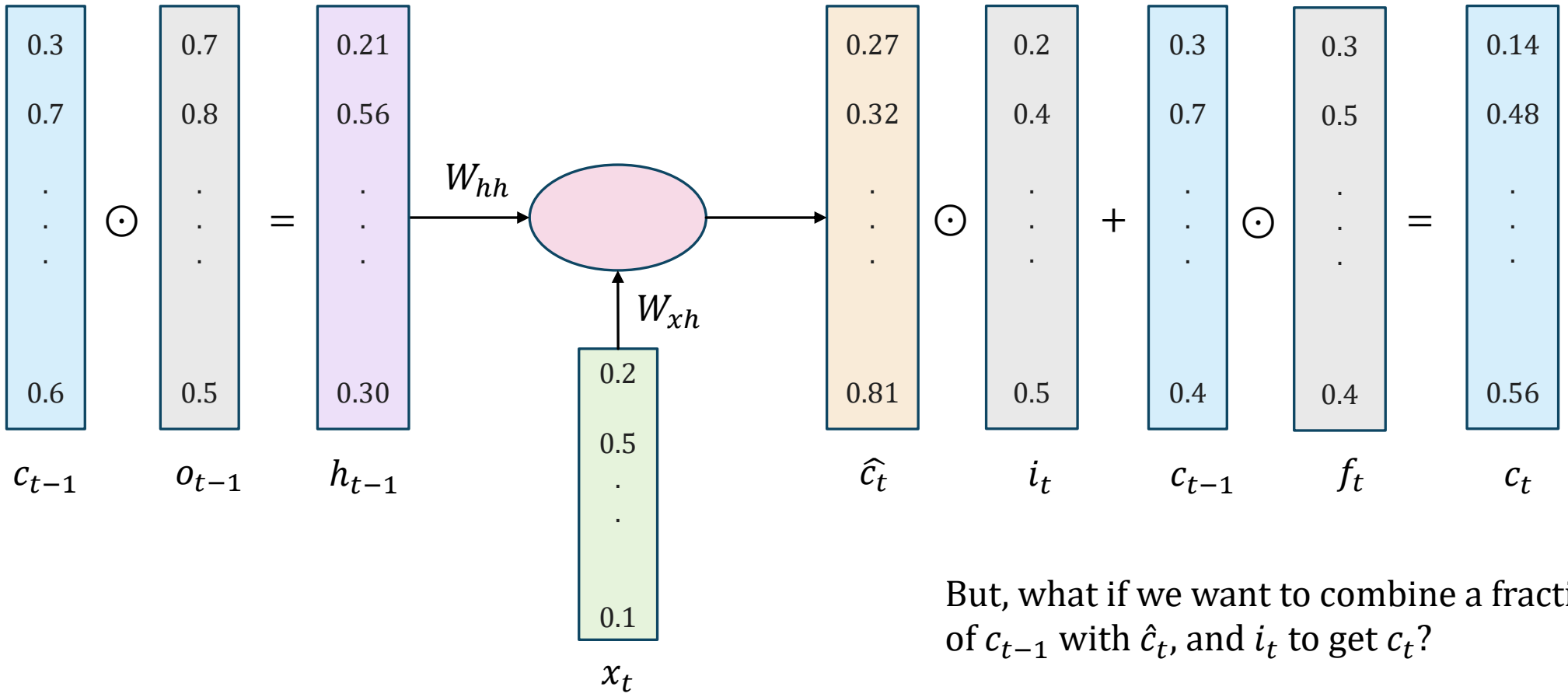
But, what if we want to combine a fraction of c_{t-1} with $\hat{c}_t$, and i_t to get c_t?

RNN

$$\mathbf{H_t} = \boldsymbol{\phi}(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$$

$$\hat{\mathbf{Y}}_\mathbf{t} = \mathbf{H_t W_{hy}} + \mathbf{b_o}$$

Modifying the RNN for Selective Forget



But, what if we want to combine a fraction of c_{t-1} with $\hat{c}_t$, and i_t to get c_t?

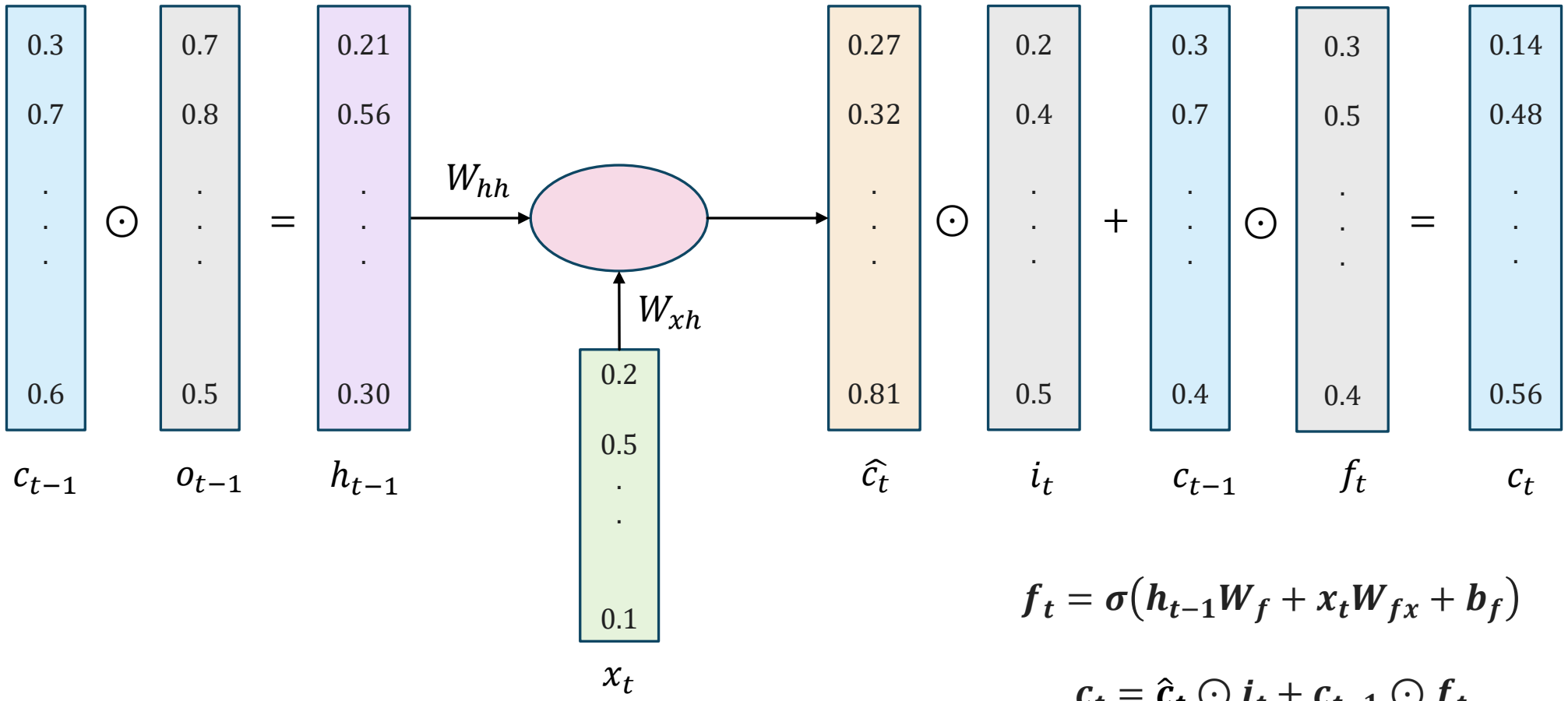
$$\boldsymbol{c_t = \hat{c}_t \odot i_t + c_{t-1} \odot f_t}$$

RNN

$$\mathbf{H_t = \phi(X_t W_{xh} + H_{t-1} W_{hh} + b_h)}$$

$$\mathbf{\hat{Y}_t = H_t W_{hy} + b_o}$$

Modifying the RNN for Selective Forget



$$\boldsymbol{f_t = \sigma\left(h_{t-1}W_f + x_tW_{fx} + b_f\right)}$$

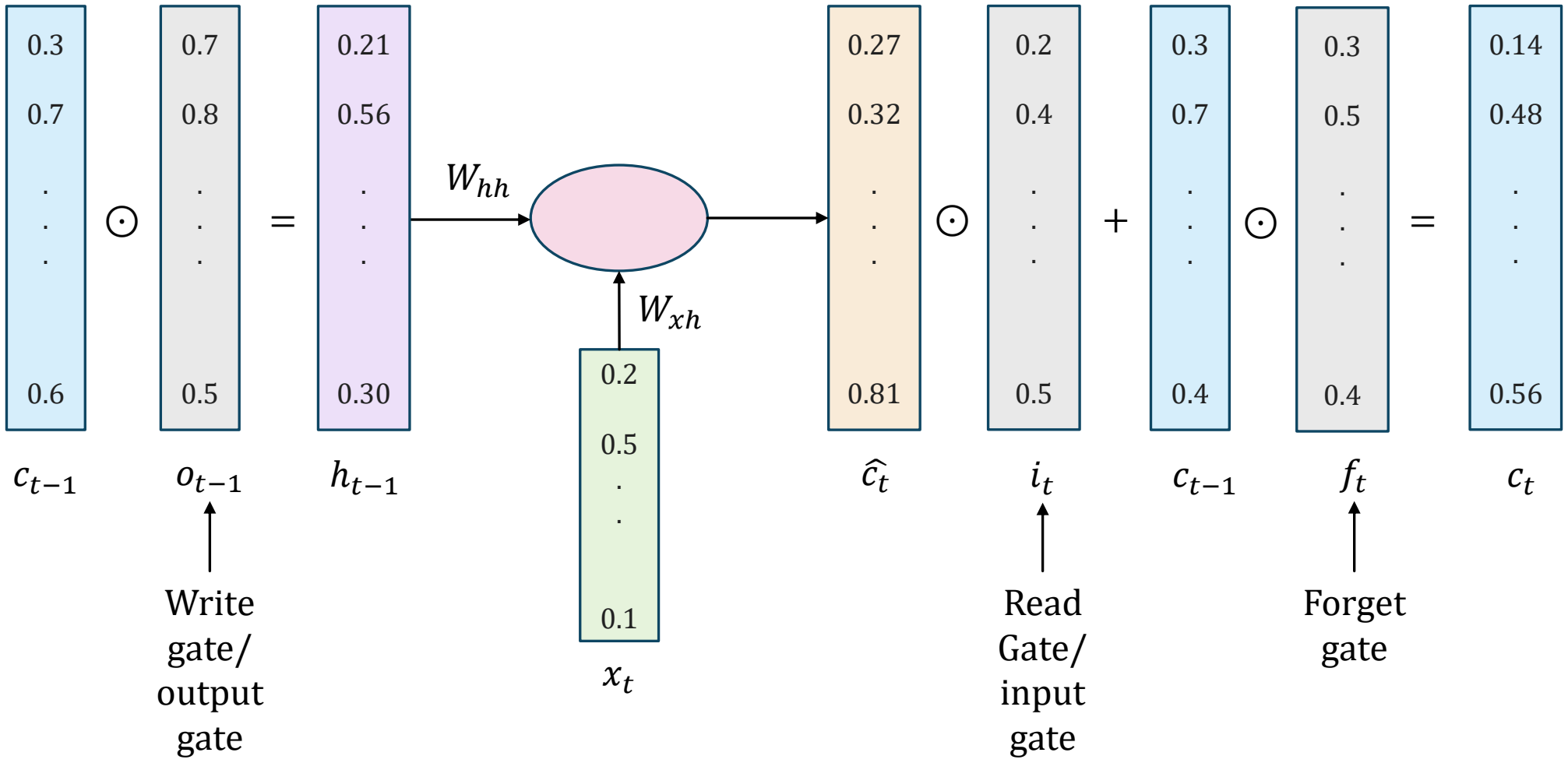
$$\boldsymbol{c_t = \hat{c}_t \odot i_t + c_{t-1} \odot f_t}$$

RNN

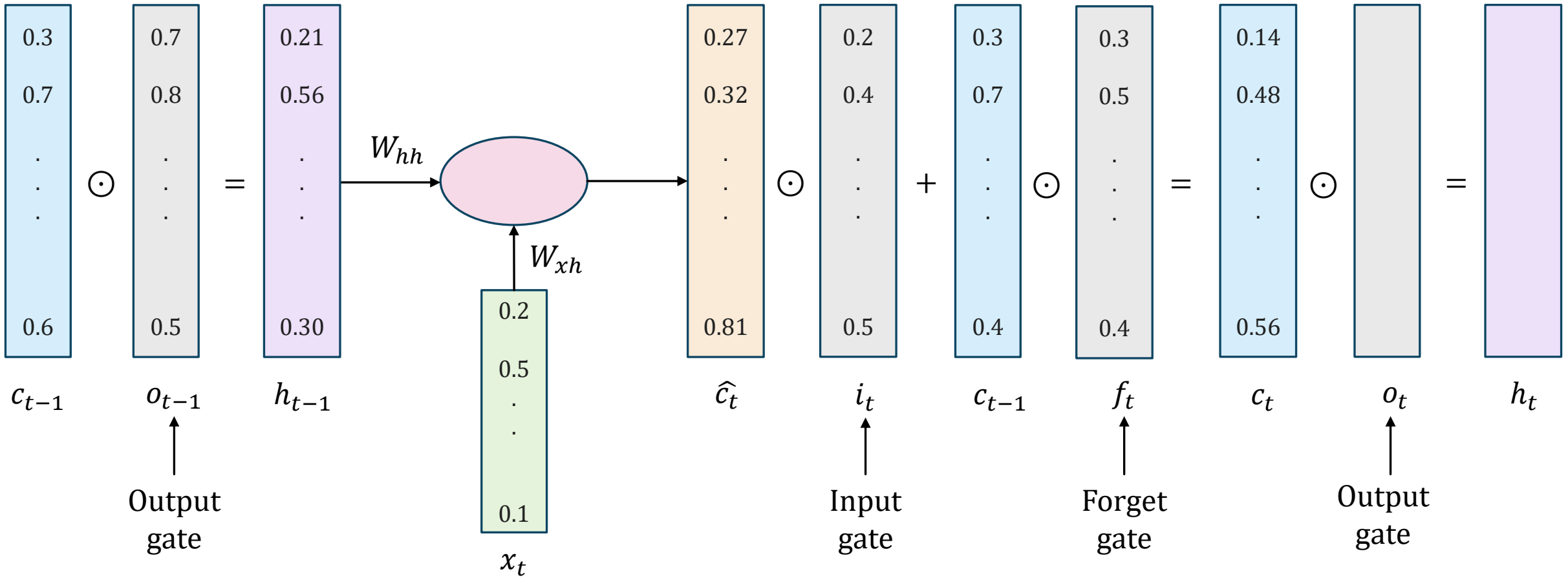
$$\mathbf{H_t = \phi(X_tW_{xh} + H_{t-1}W_{hh} + b_h)}$$

$$\mathbf{\hat{Y}_t = H_tW_{hy} + b_o}$$

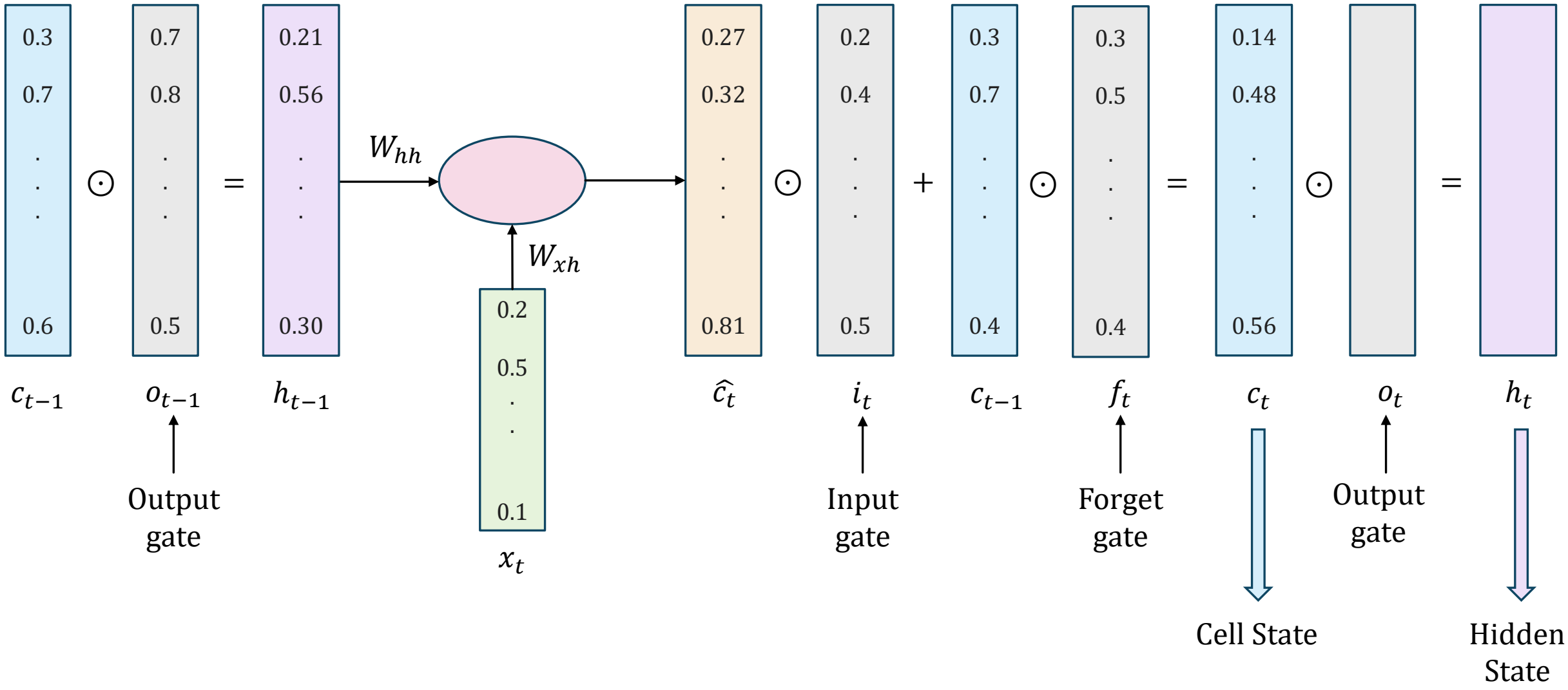
The Long Short Term Memory (LSTM) Unit



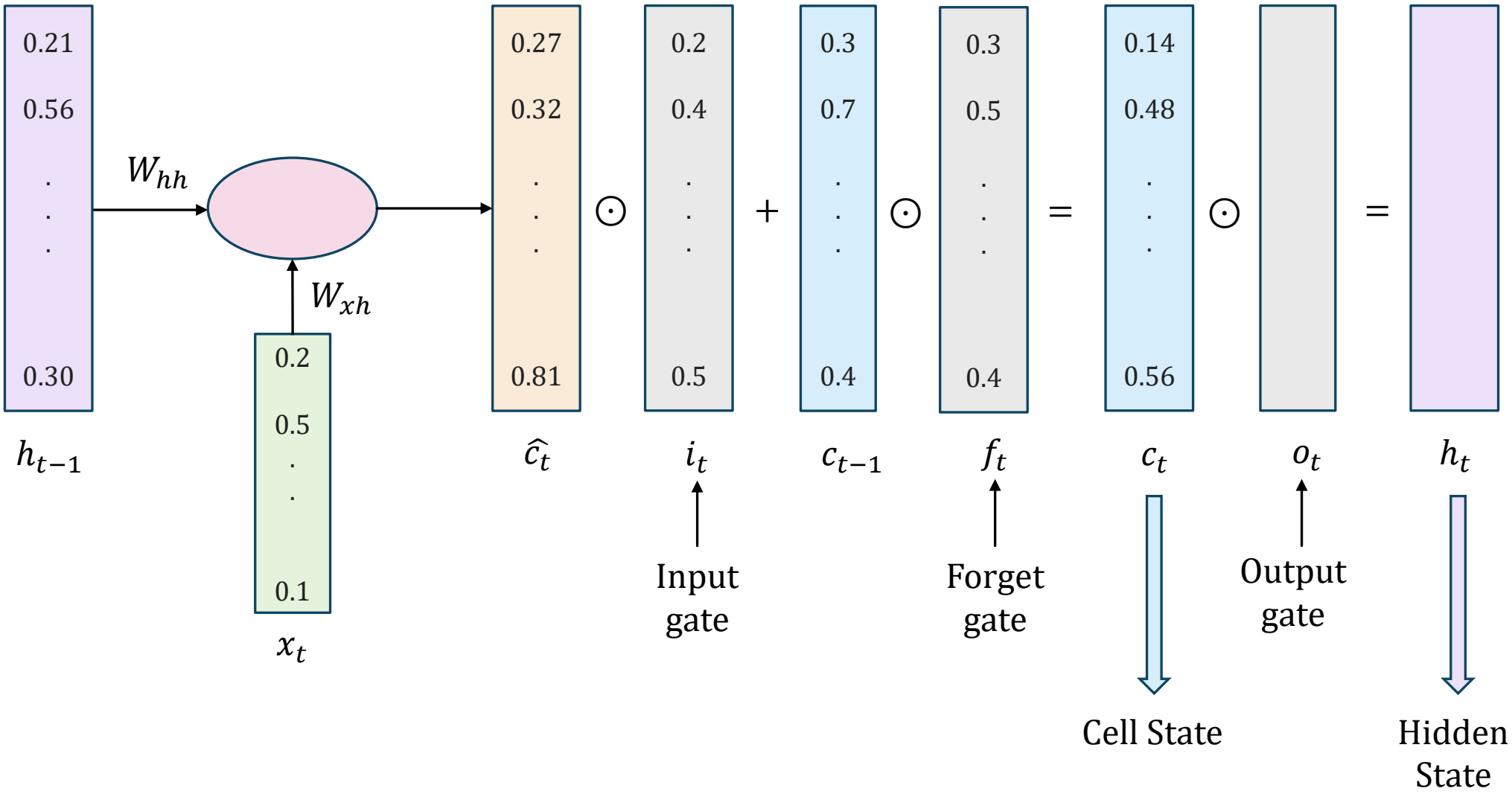
The Long Short Term Memory (LSTM) Unit



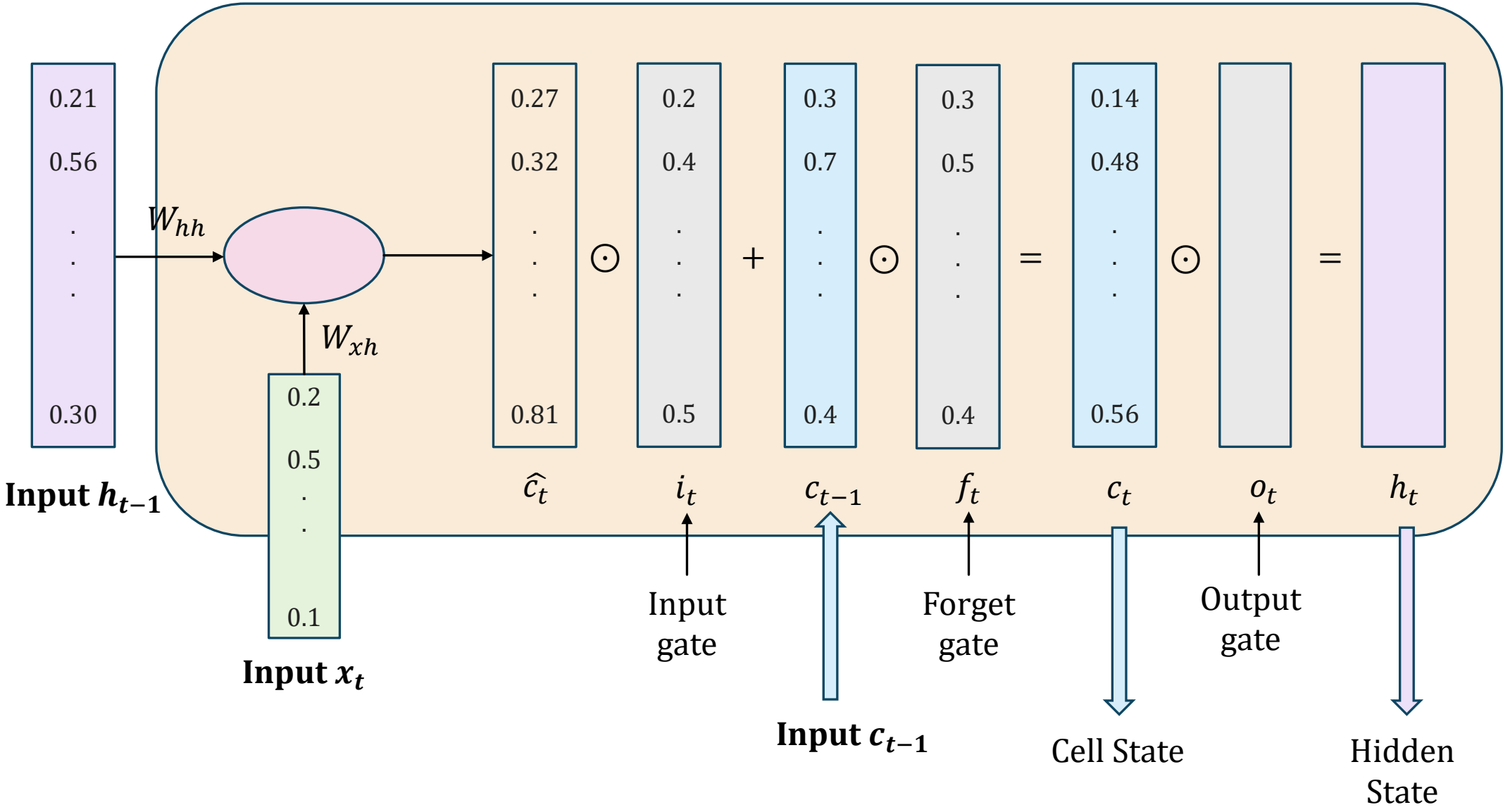
The Long Short Term Memory (LSTM) Unit



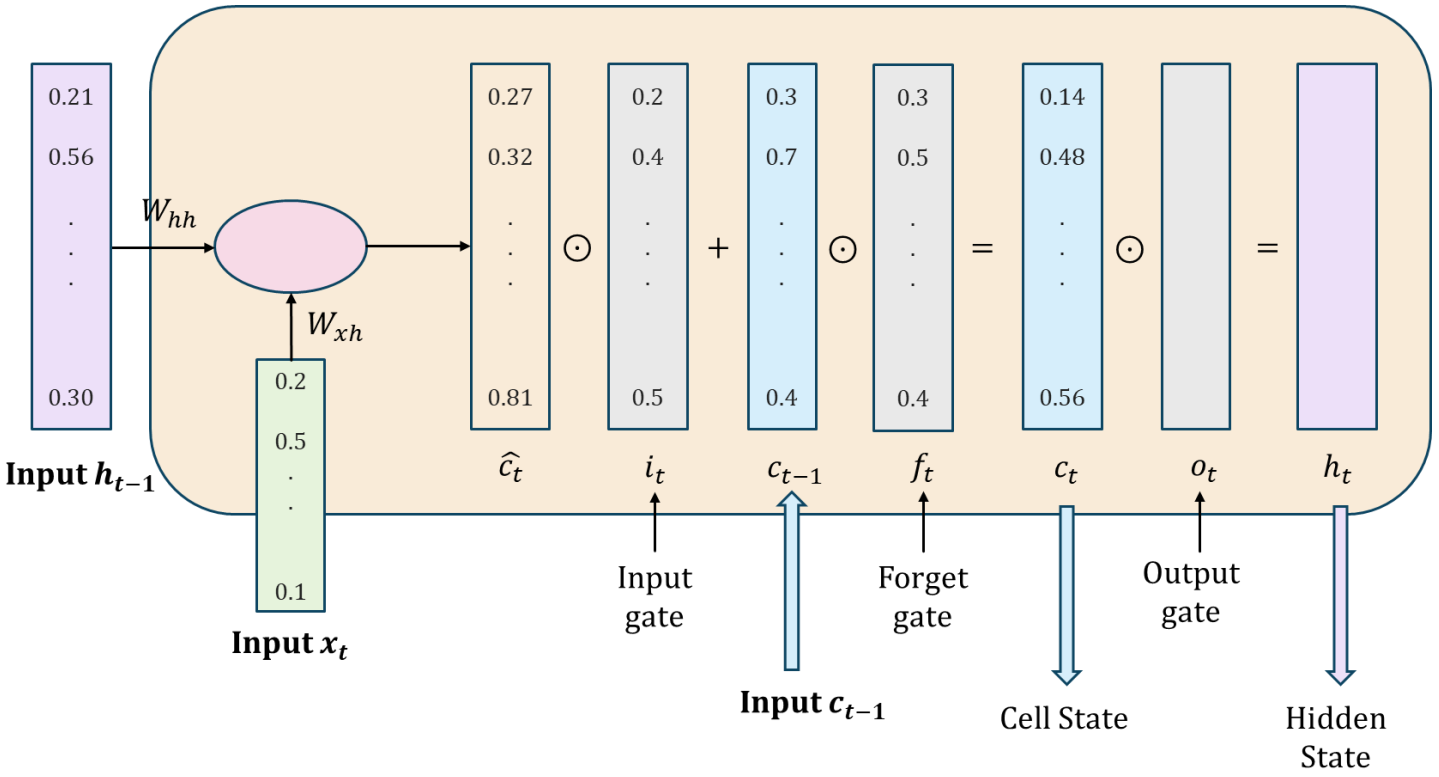
The Long Short Term Memory (LSTM) Unit



The Long Short Term Memory (LSTM) Unit



The LSTM Unit



$$o_{t-1} = \sigma(h_{t-2}W_a + x_{t-1}W_{ax} + b_a)$$

$$h_{t-1} = \sigma_c(c_{t-1}) \odot o_{t-1}$$

$$\hat{c}_t = \phi(x_t W_{xh} + h_{t-1} W_{hh} + b_h)$$

$$i_t = \sigma(h_{t-1}W_m + x_t W_{mx} + b_m)$$

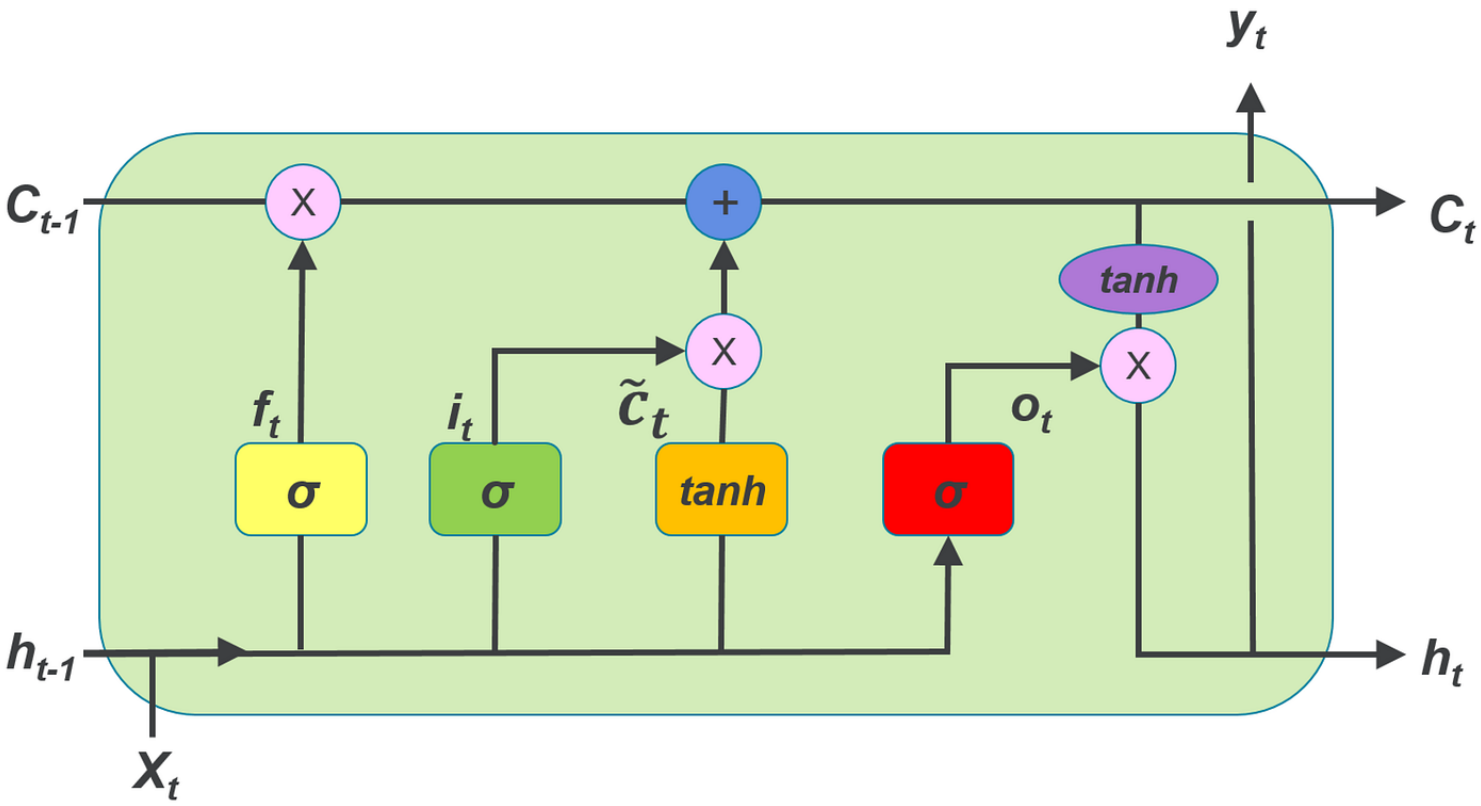
$$f_t = \sigma\left(h_{t-1}W_f + x_t W_{fx} + b_f\right)$$

$$c_t = \hat{c}_t \odot i_t + c_{t-1} \odot f_t$$

$$h_t = \sigma_c(c_t) \odot o_{t-1}$$

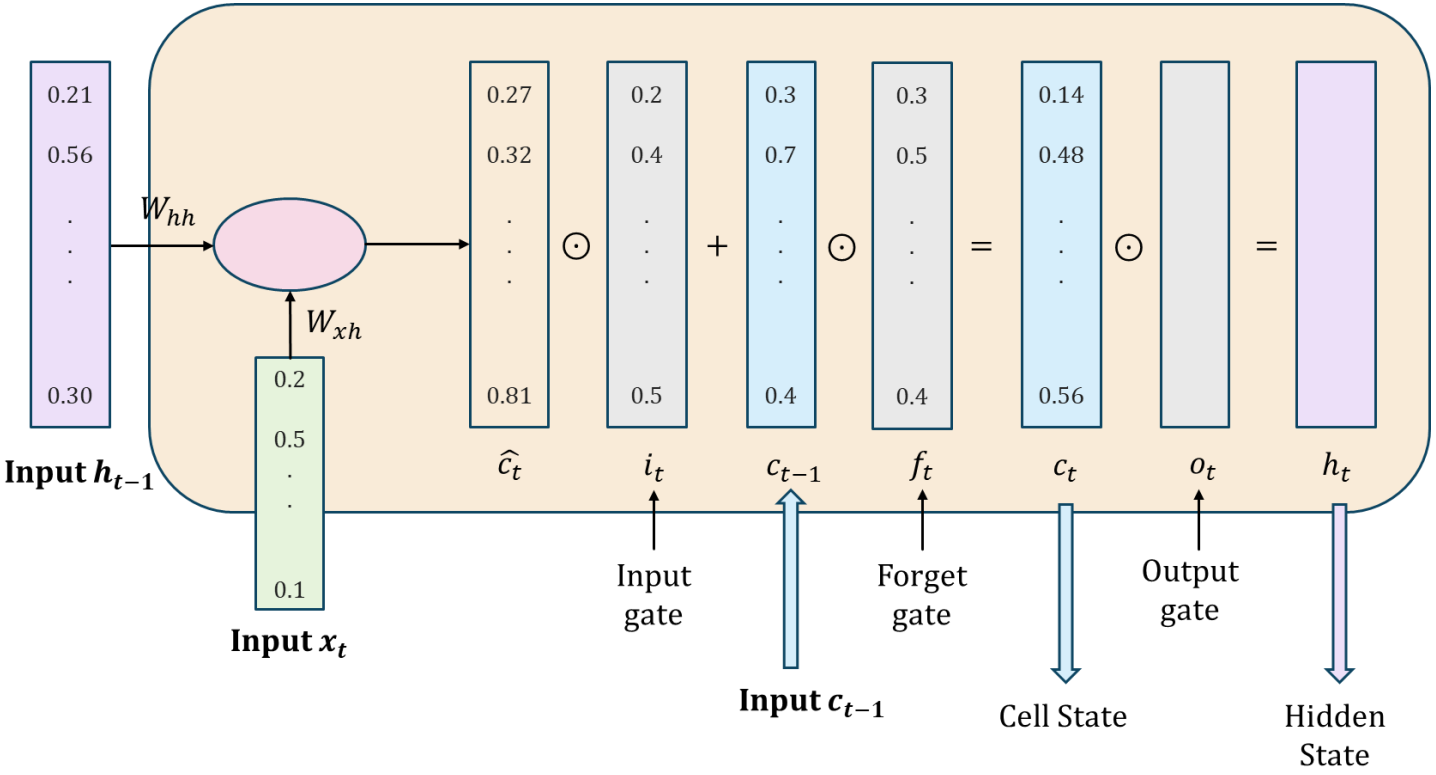
$$\hat{y}_t = h_t W_{hy} + b_o$$

LSTM Unit: A Popular Architecture



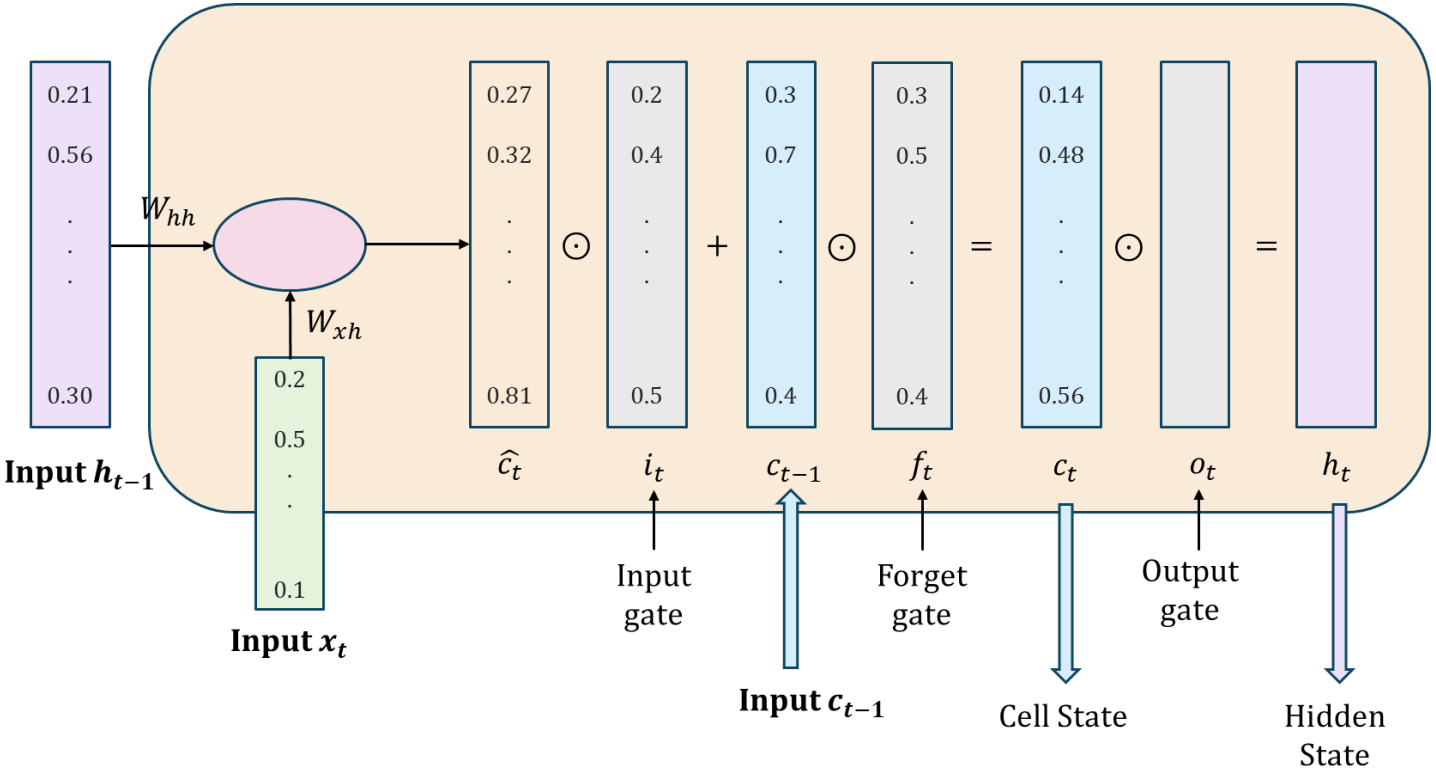
$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \sigma_h(c_t)
\end{aligned}
$$

The LSTM Unit



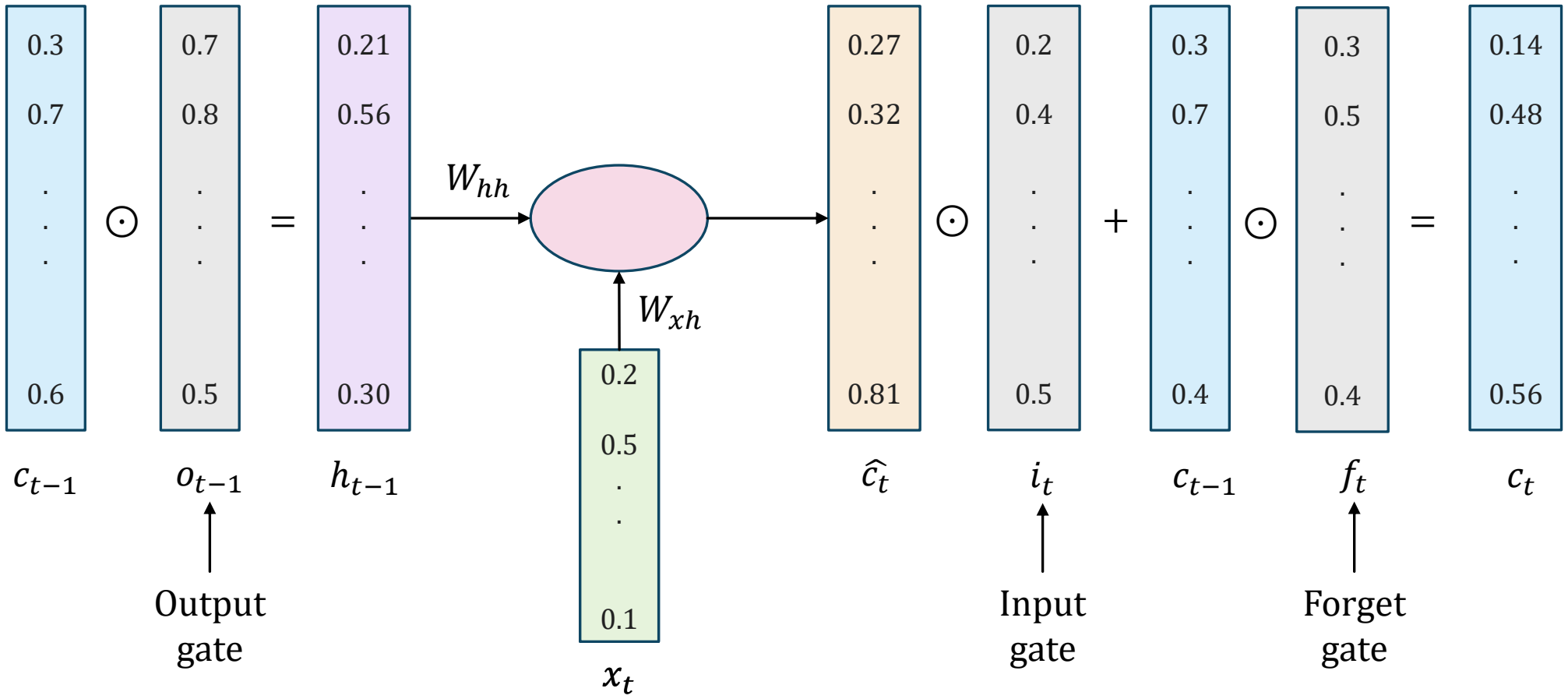
Allows selective propagation of information through the gates

The LSTM Unit



There are many variants of LSTM

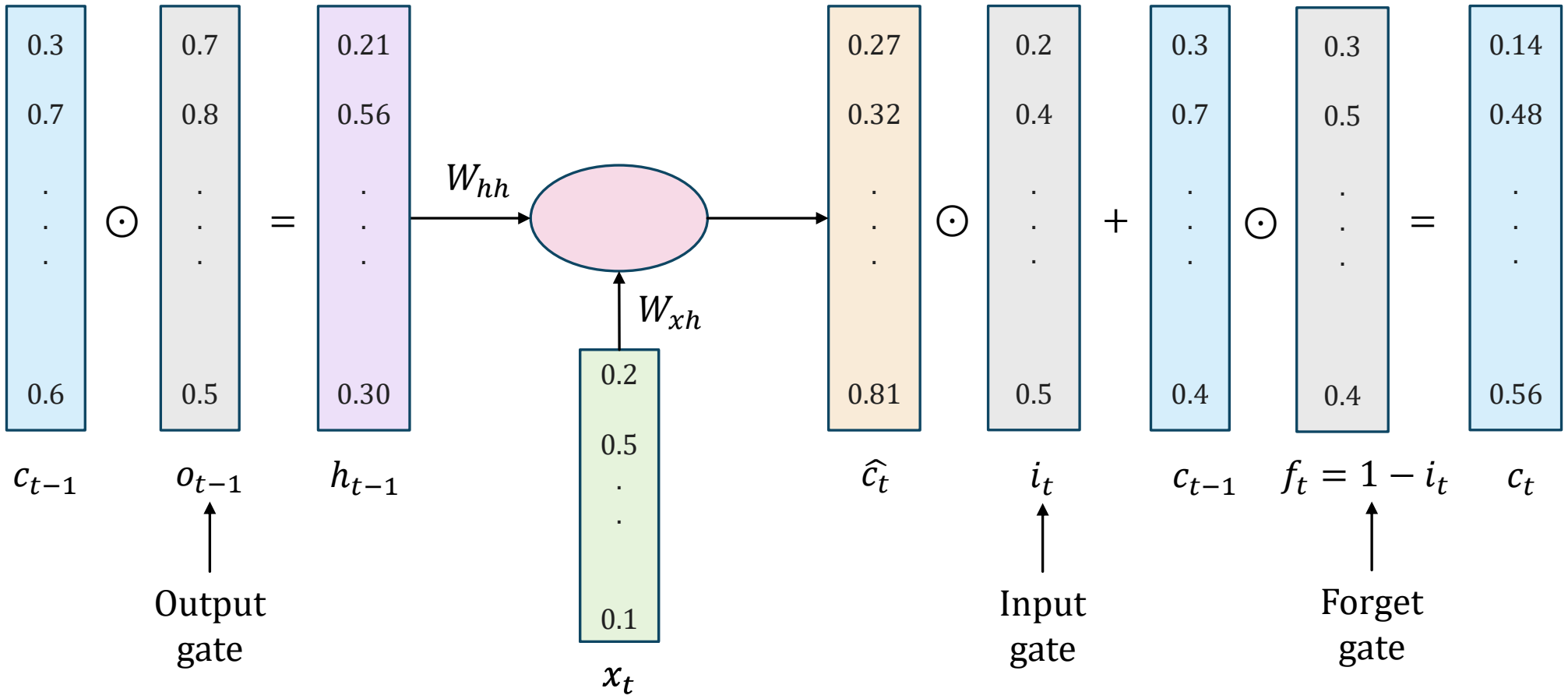
Modifying the LSTM



We do not have completely separate parameters for forget gate

We take $f_t = (1 - i_t)$

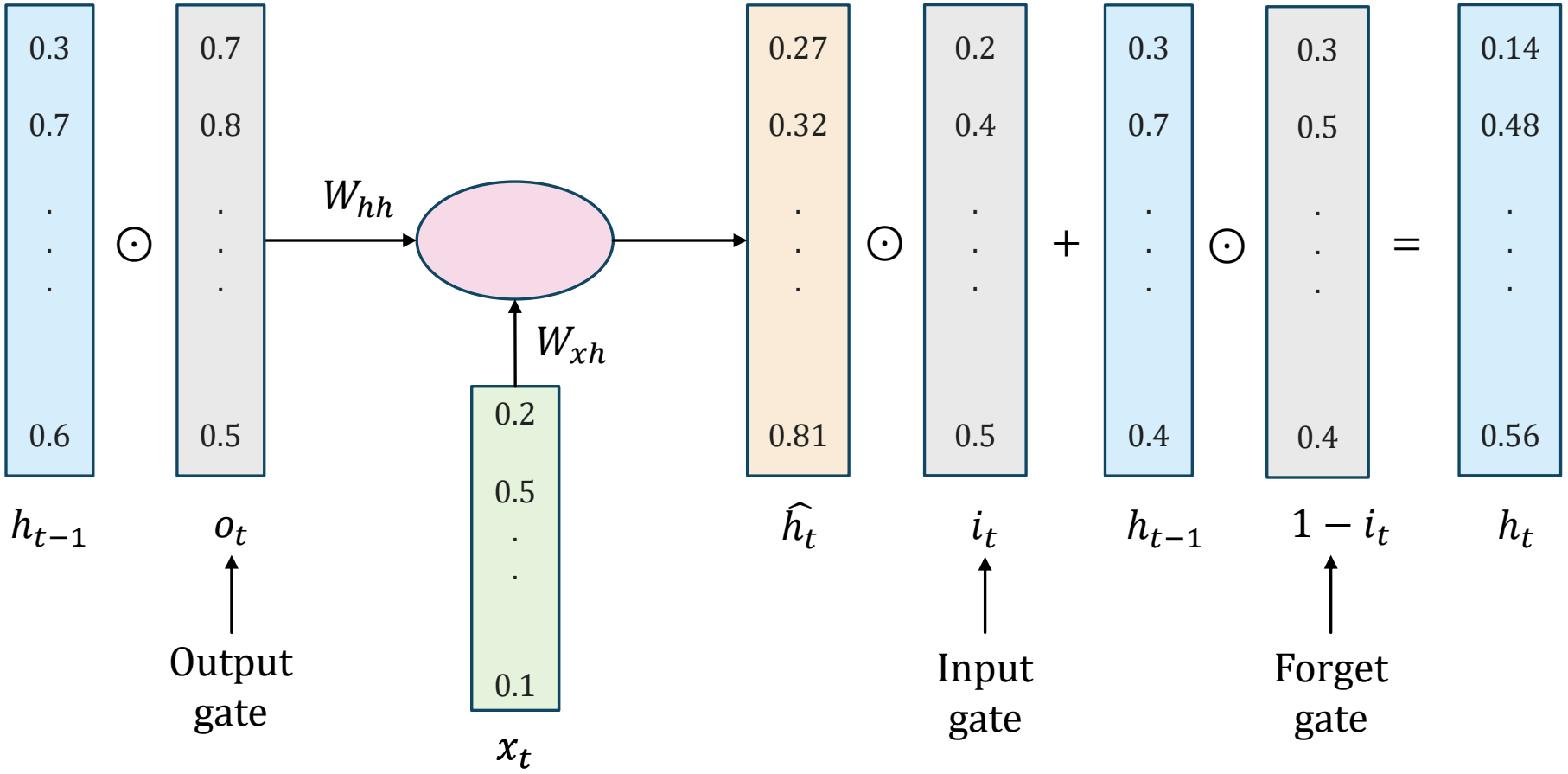
Modifying the LSTM



We do not have completely separate parameters for forget gate

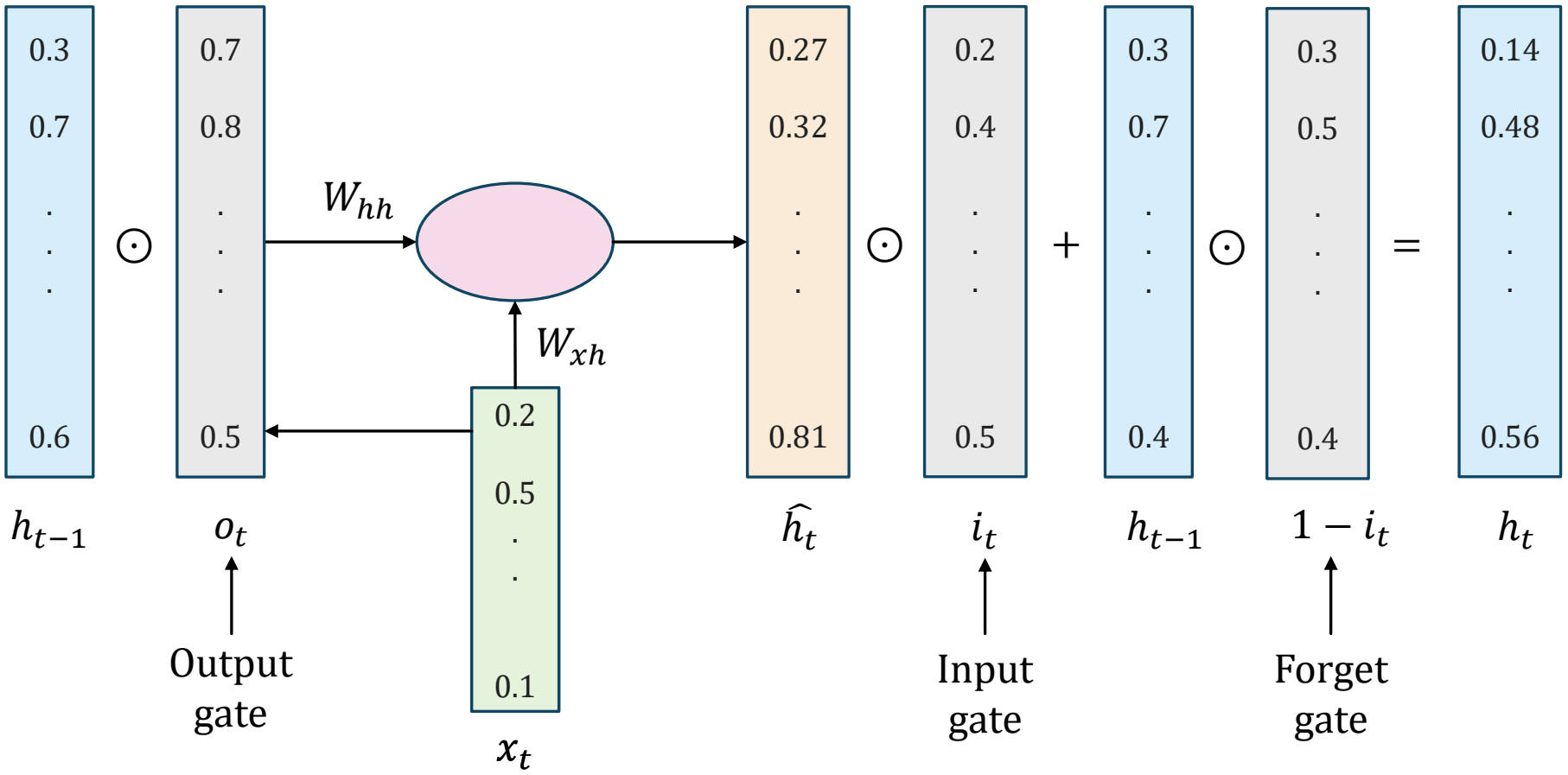
We take $f_t = (1 - i_t)$

Modifying the LSTM



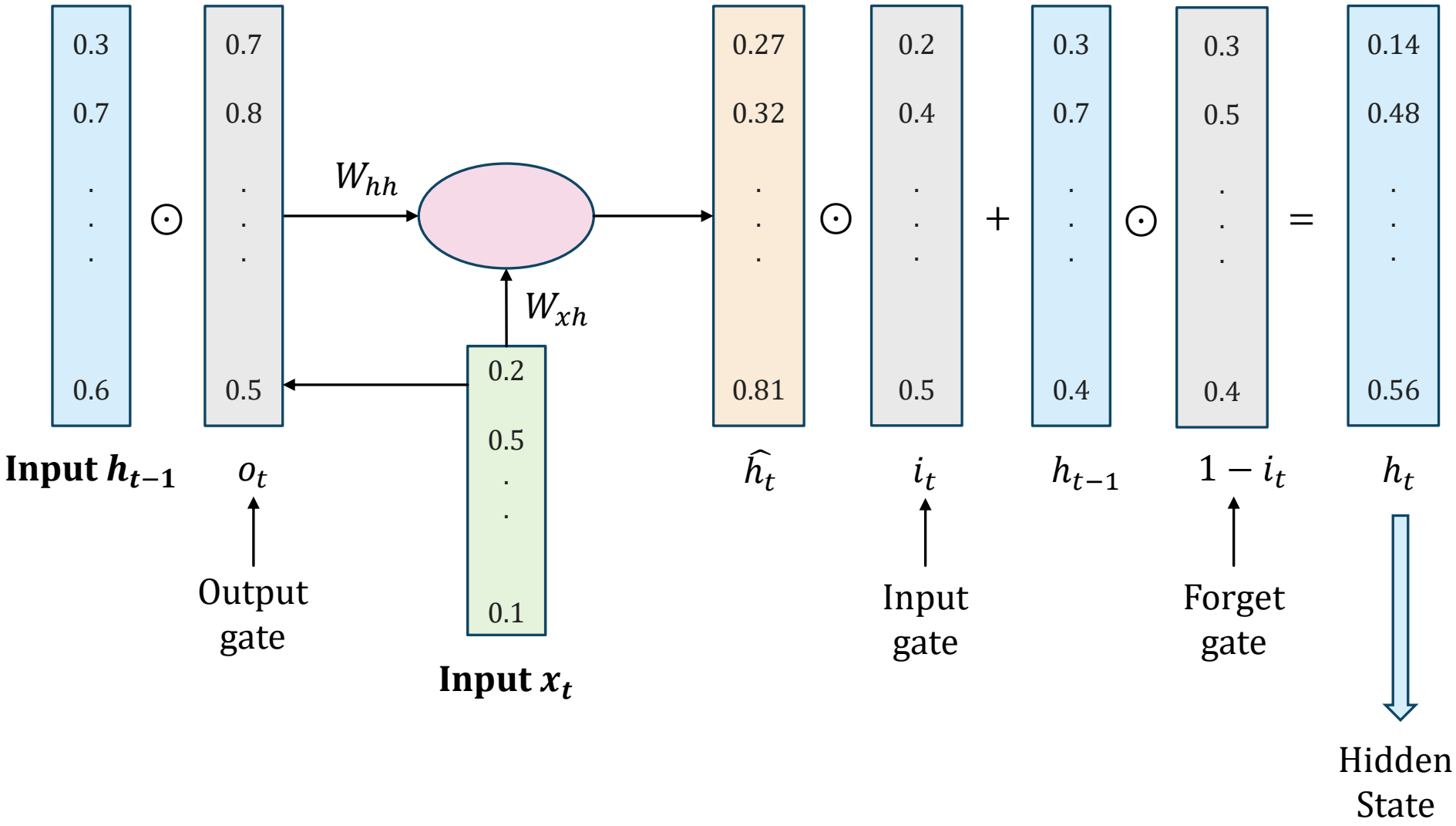
With changes in symbol names

Modifying the LSTM

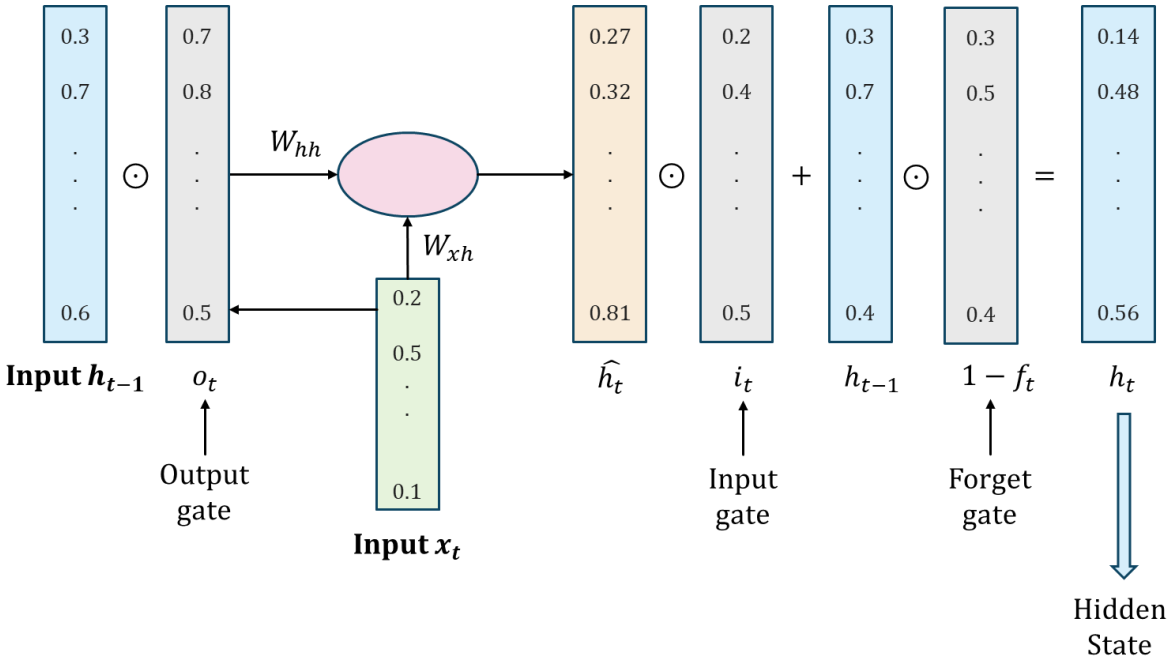


$$\boldsymbol{o_t = \sigma(h_{t-1}W_a + x_tW_{ax} + b_a)}$$

The Gated Recurrent Unit (GRU)



The Gated Recurrent Unit (GRU)



$$o_t = \sigma(h_{t-1}W_a + x_tW_{ax} + b_a)$$

$$i_t = \sigma(h_{t-1}W_m + x_tW_{mx} + b_m)$$

$$\hat{h}_t = \phi(x_t\mathbf{W_{xh}} + (h_{t-1} \odot o_t)\mathbf{W_{hh}} + \mathbf{b_h})$$

$$h_t = \hat{h}_t \odot i_t + h_{t-1} \odot (1 - i_t)$$

$$\hat{y}_t = h_t\mathbf{W_{hy}} + \mathbf{b_o}$$

Not Only the Past, the Future may Also Matter

- Fill in the sentence

  It is _____

Not Only the Past, the Future may Also Matter

- Fill in the sentence

  It is _____

- The blank could contain anything
  - e.g, It is interesting; It is summer, etc.

Not Only the Past, the Future may Also Matter

- But if I tell you to fill in the sentence

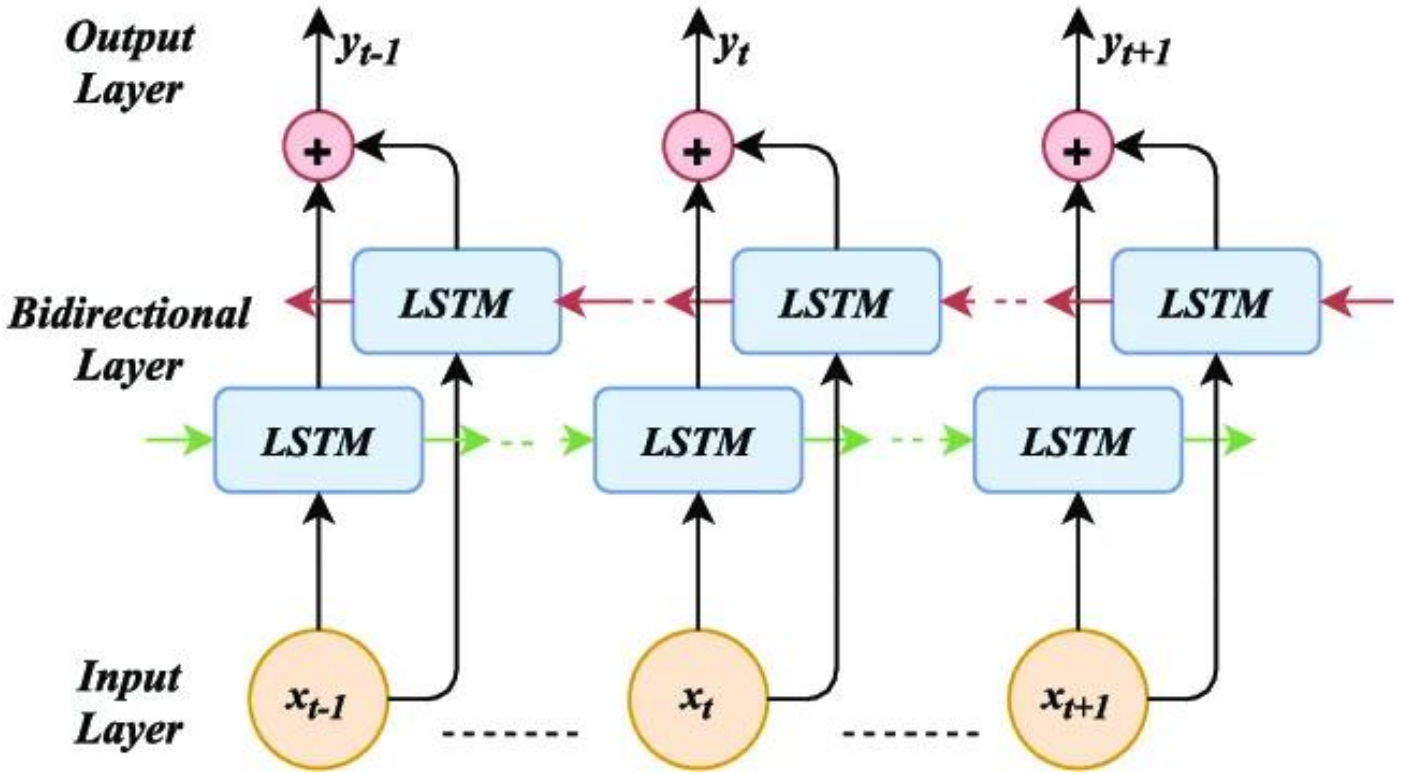
  It is _____. So, I have to take the umbrella before I go outside.

Not Only the Past, the Future may Also Matter

- But if I tell you to fill in the sentence

  It is raining. So, I have to take the umbrella before I go outside. (most probably)

Bi-directional LSTM

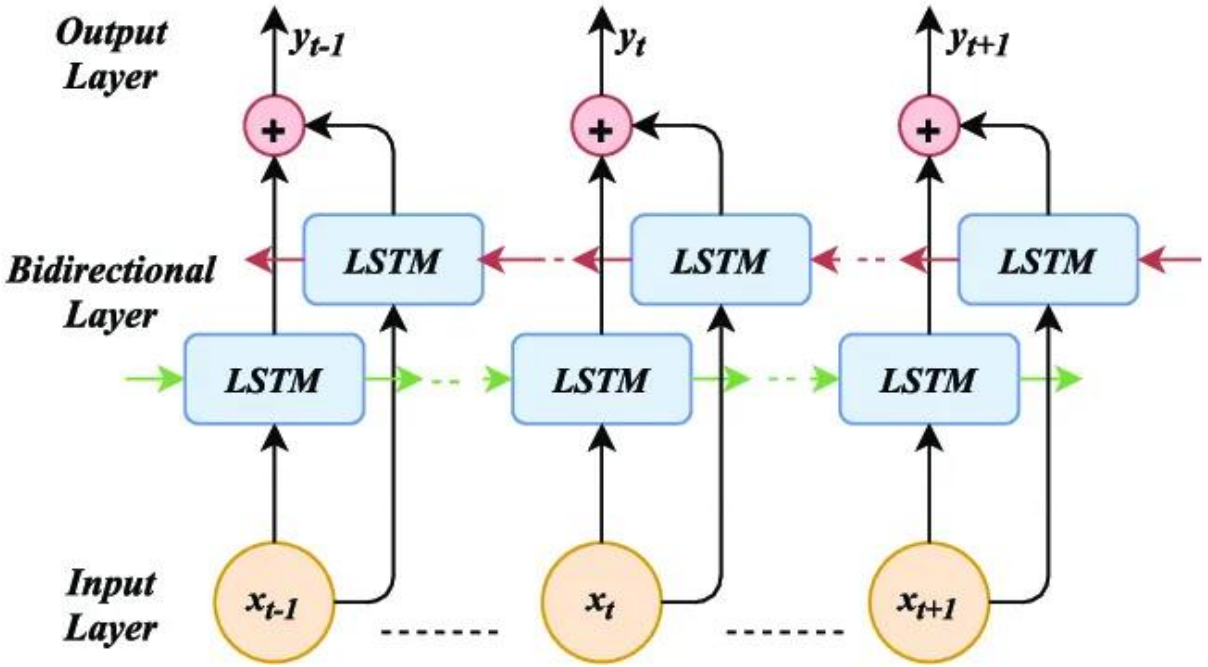


Two LSTM units

Captures the context of the previous and the next time steps

https://www.researchgate.net/figure/Bidirectional-LSTM-model-showing-the-input-and-output-layers-The-red-arrows-represent_fig3_344554659

Bi-directional LSTM

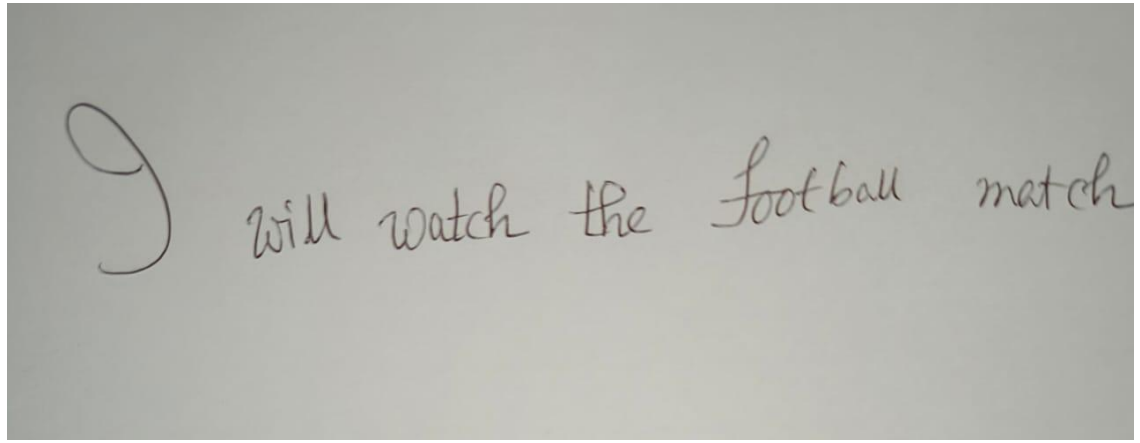


May not be very useful for sequence generation

Because at test time, future data won't be available

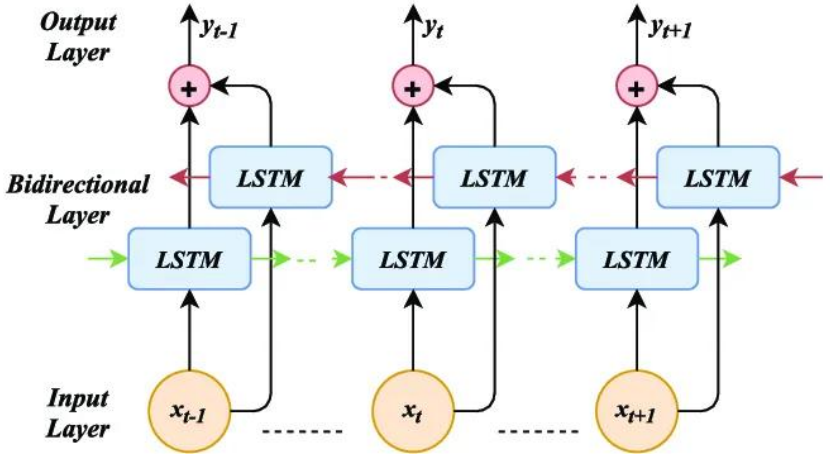
https://www.researchgate.net/figure/Bidirectional-LSTM-model-showing-the-input-and-output-layers-The-red-arrows-represent_fig3_344554659

Bi-directional LSTM



Suppose, I am designing an OCR system
It may output

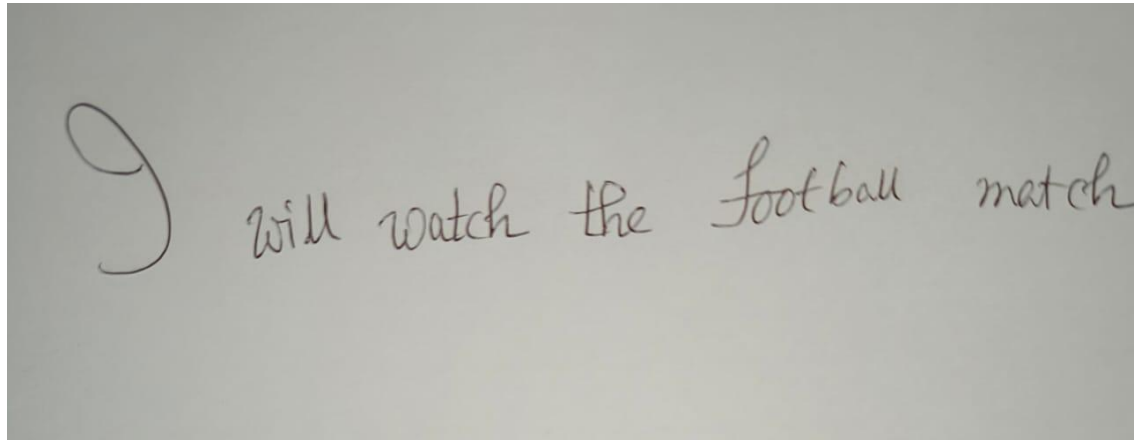
I will watch the fooEball match



The system may make error and output the following if it only considers the past

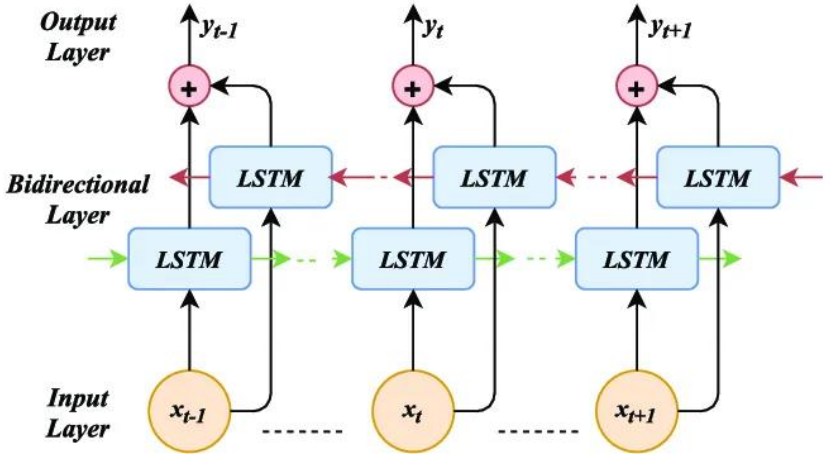
https://www.researchgate.net/figure/Bidirectional-LSTM-model-showing-the-input-and-output-layers-The-red-arrows-represent_fig3_344554659

Bi-directional LSTM



Suppose, I am designing an OCR system
It may output

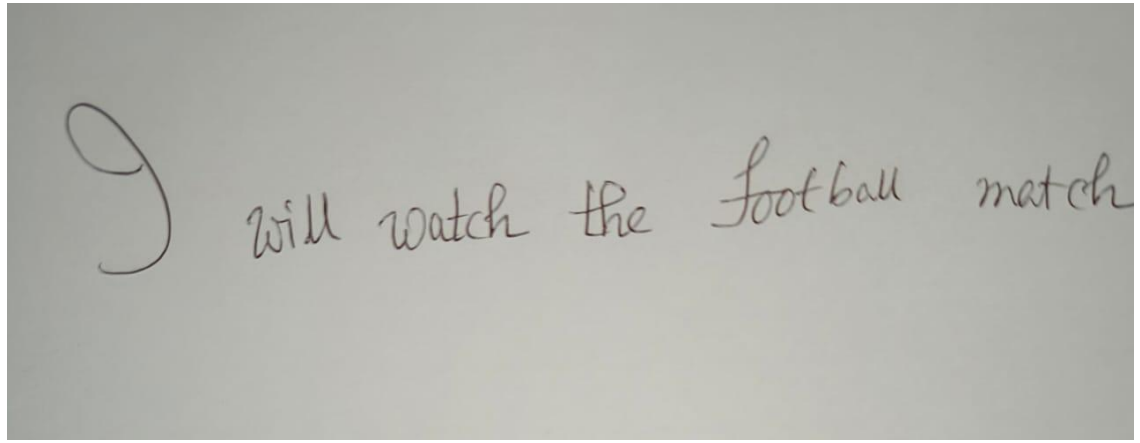
I will watch the fooEball match



But if the system considers the future, it may be able to understand that the letter is not E. It should be t

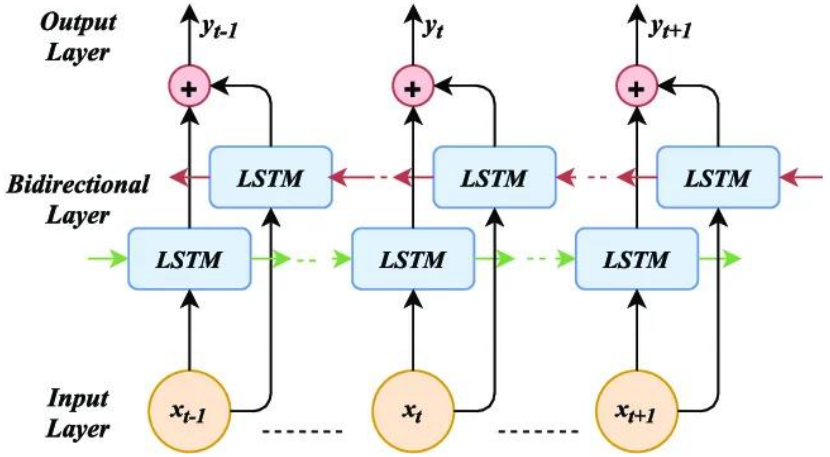
https://www.researchgate.net/figure/Bidirectional-LSTM-model-showing-the-input-and-output-layers-The-red-arrows-represent_fig3_344554659

Bi-directional LSTM



Suppose, I am designing an OCR system
It may output

I will watch the football **match**



But if the system considers the future, it may be able to understand that the letter is not E. It should be t.

Autocorrect may employ such a mechanism

https://www.researchgate.net/figure/Bidirectional-LSTM-model-showing-the-input-and-output-layers-The-red-arrows-represent_fig3_344554659

Tokens

- Home Assignment: Write two methods (one learning-based and other should not involve learning) to create tokens from text.
  - Brief and to-the-point description
  - Penalty for writing unnecessary stuffs
  - All equations and figures must be explained

- Deadline: February 28, 11:59 PM