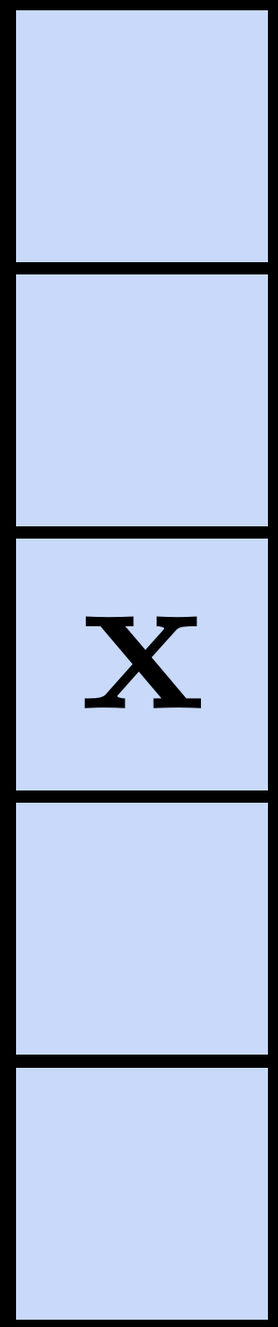
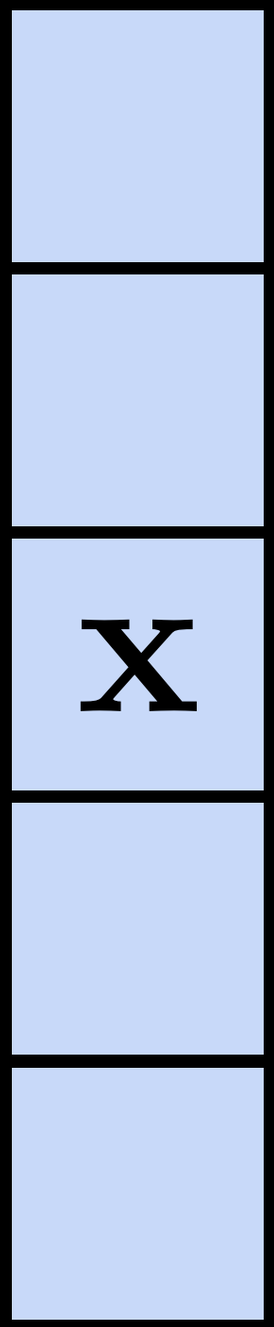
Convolutional Networks

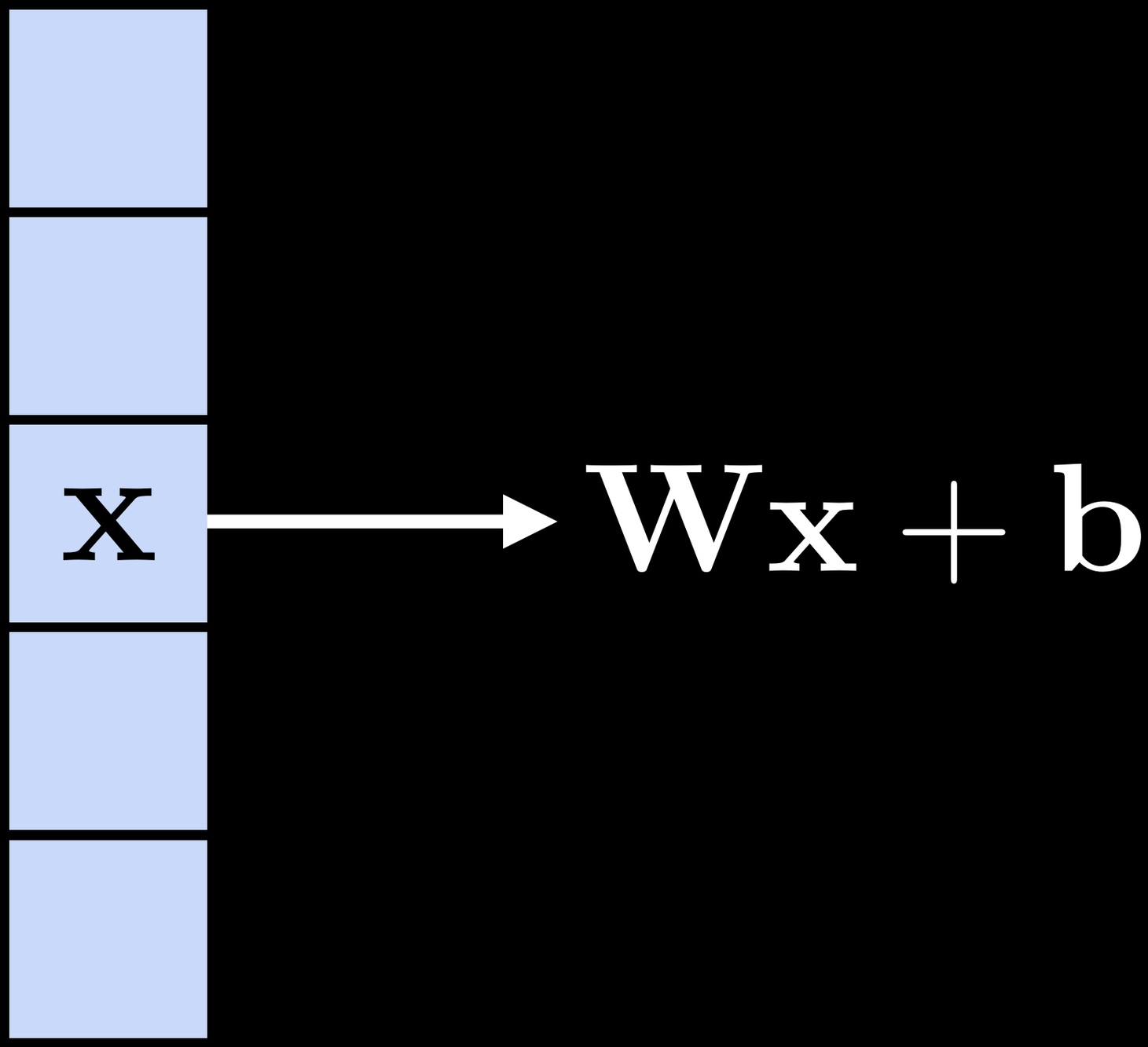
Convolutional Networks

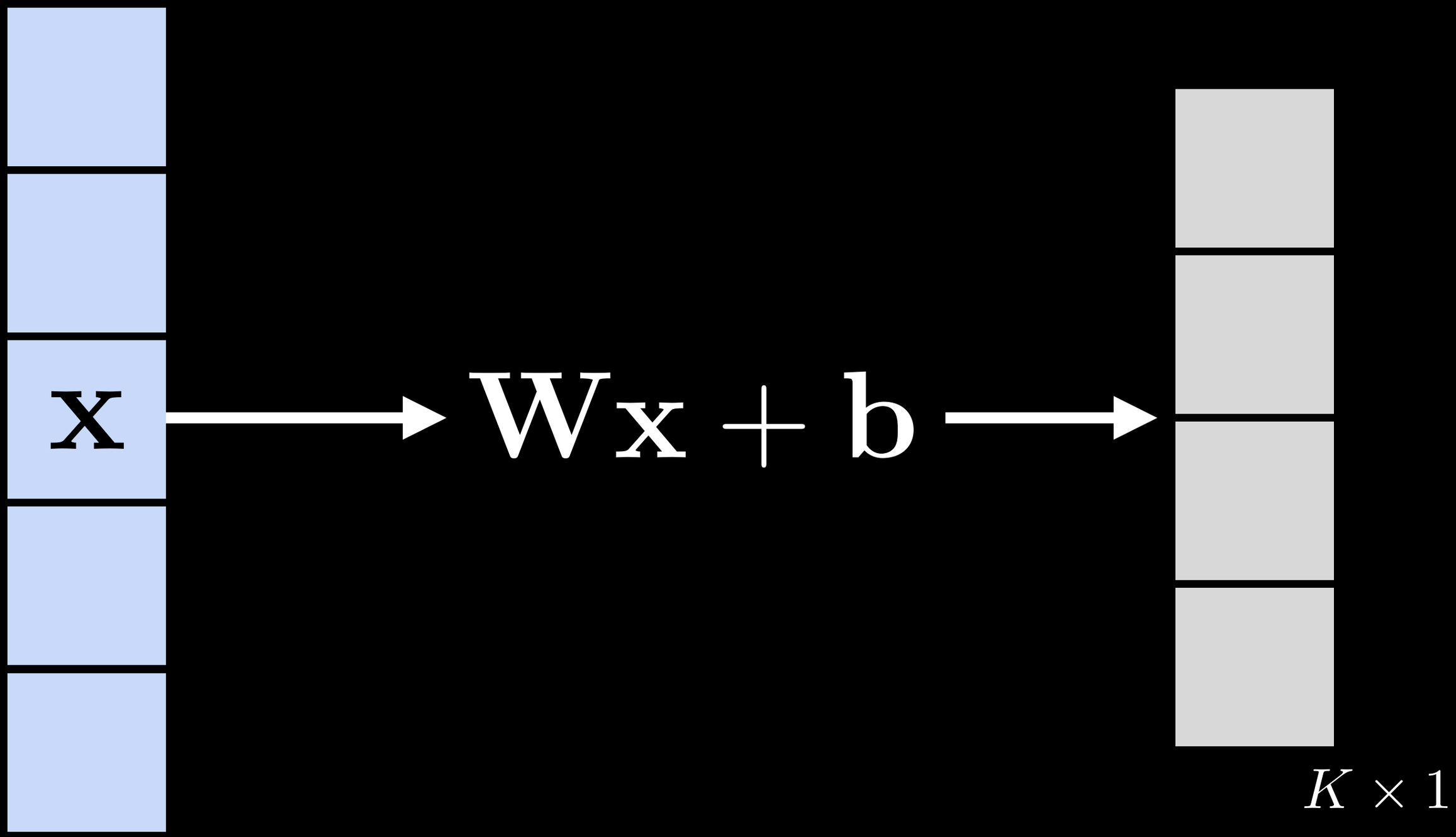

R
U
768 × 1024 × 1

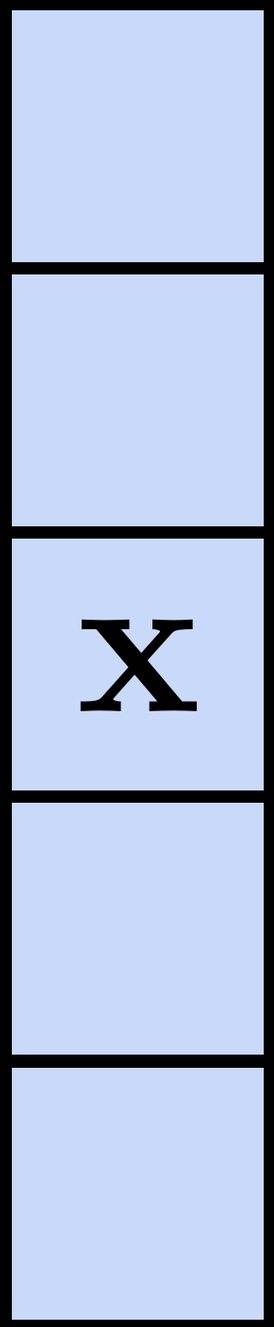

X
786432 × 1



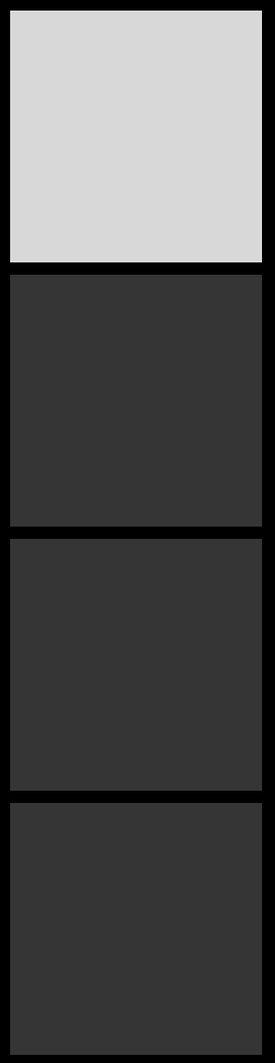
786432×1


x
Wx + b
786432 × 1

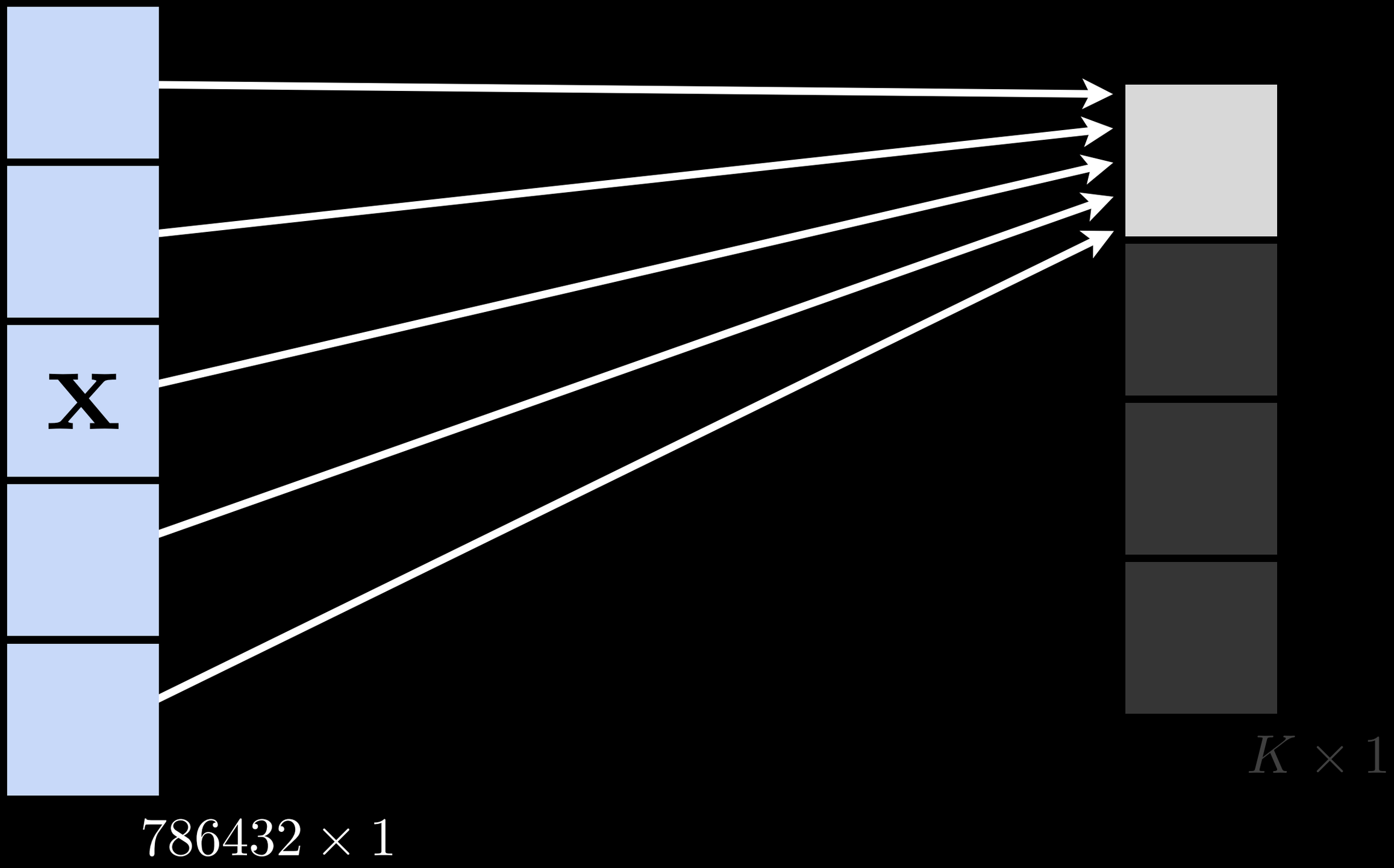

x
Wx + b
786432 × 1
K × 1

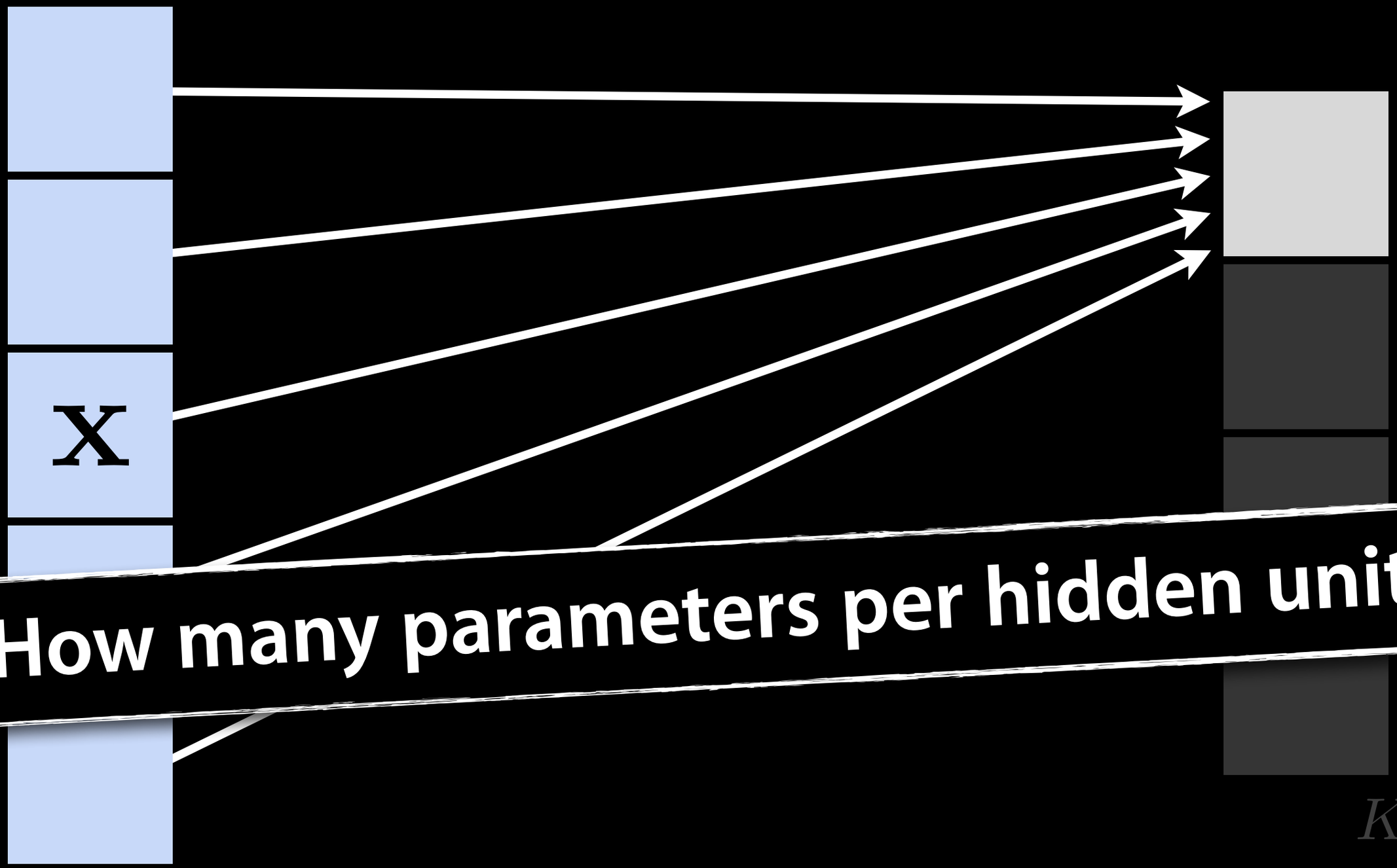


786432×1

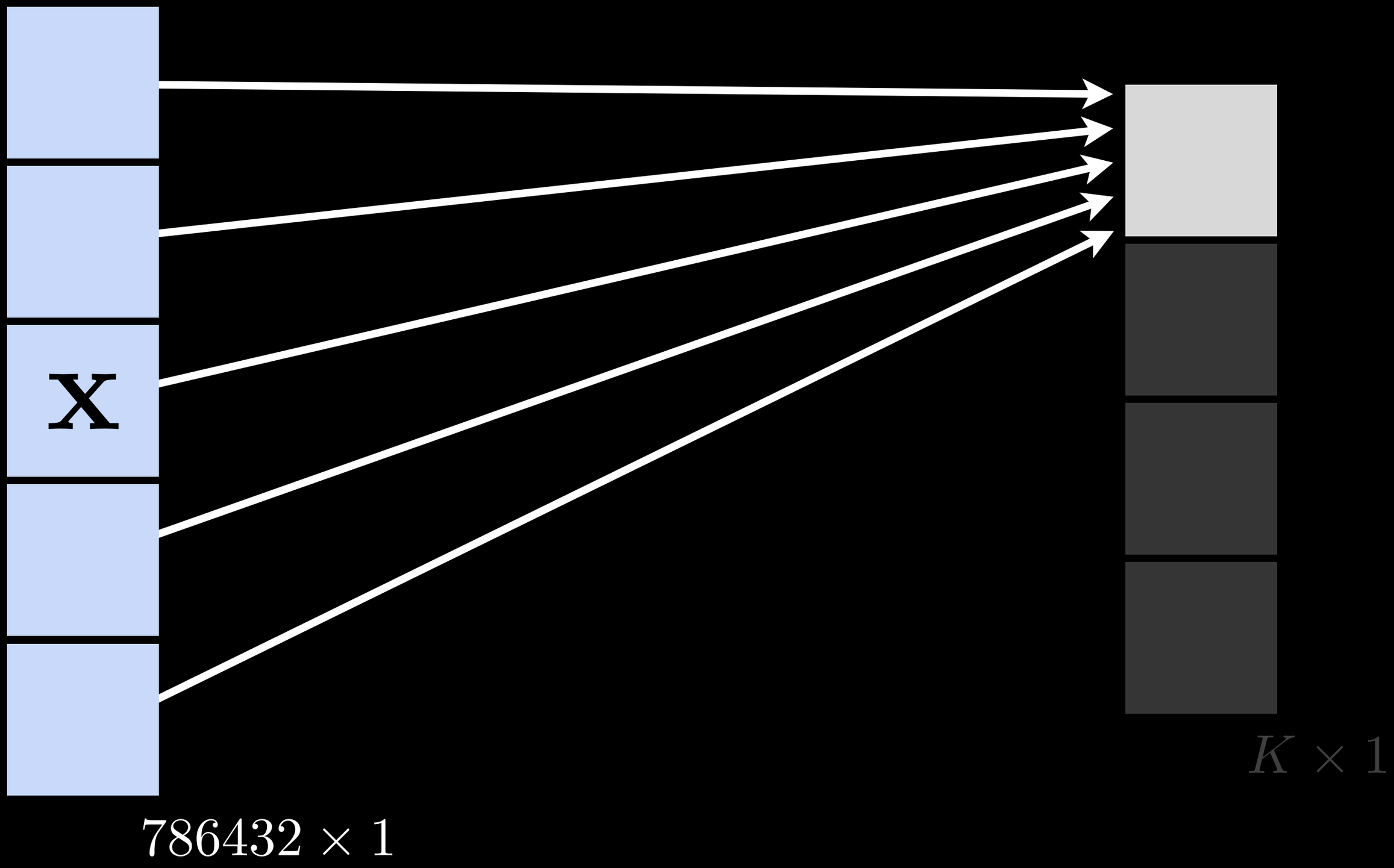


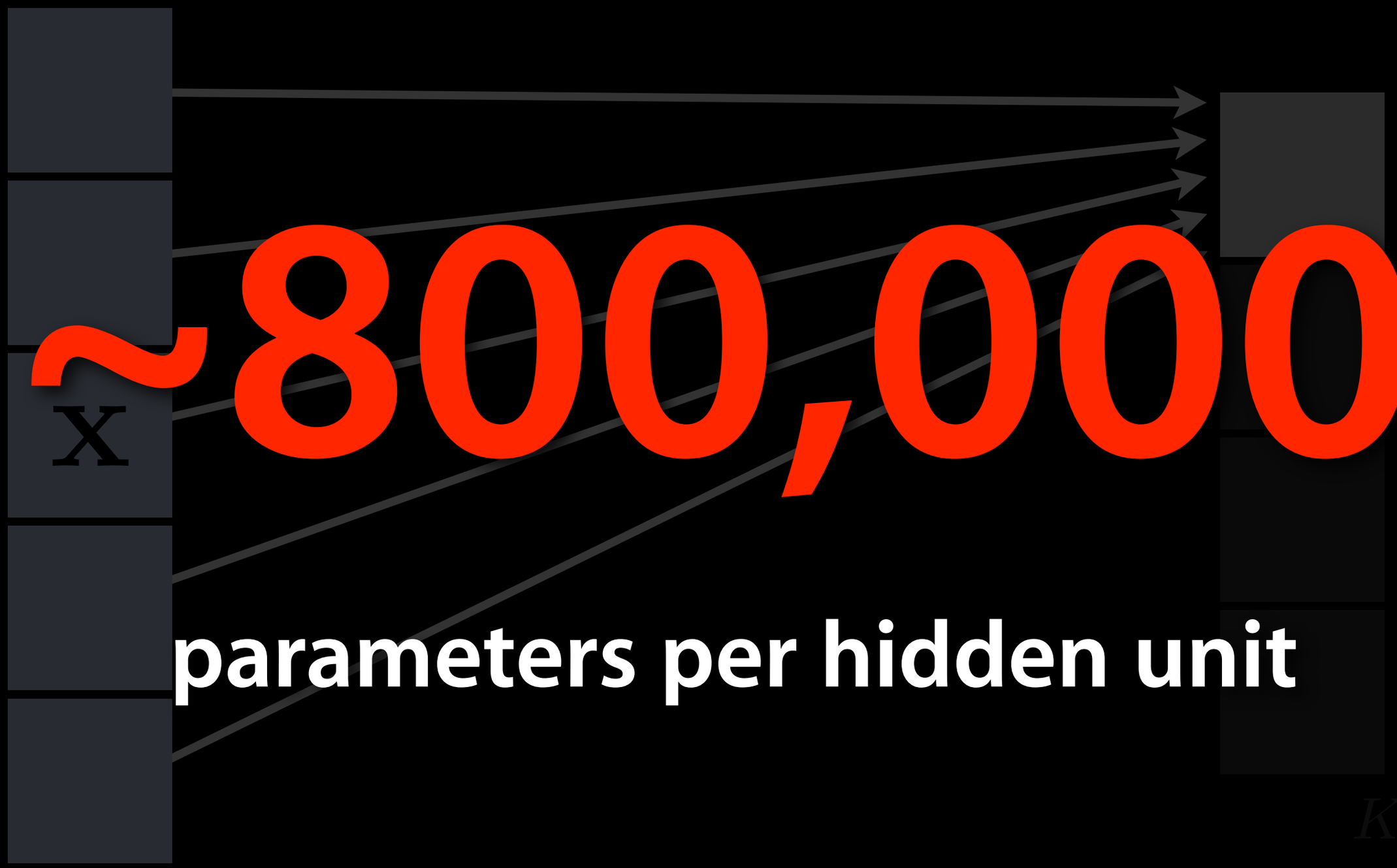
$K \times 1$

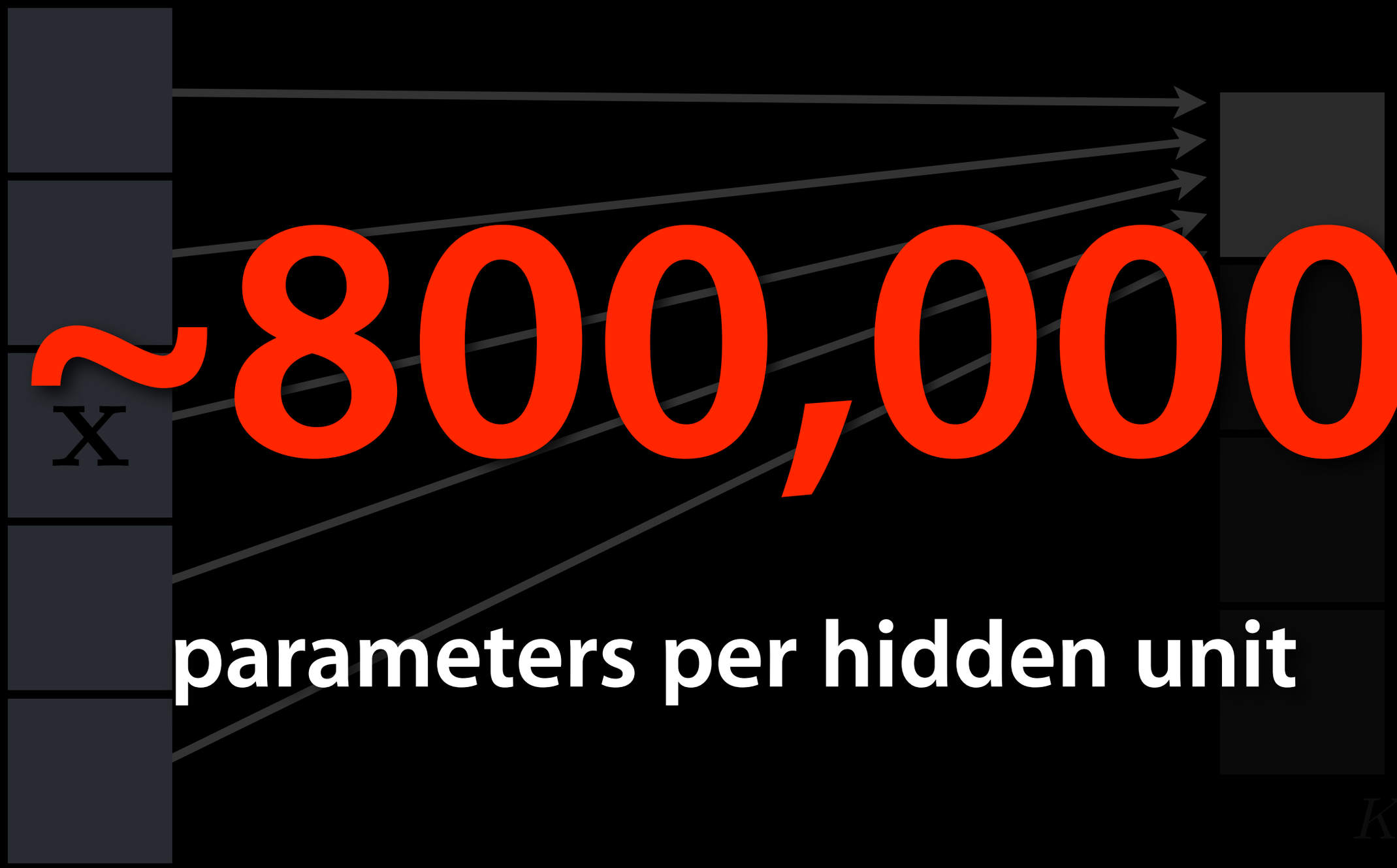

X
786432 × 1
K × 1

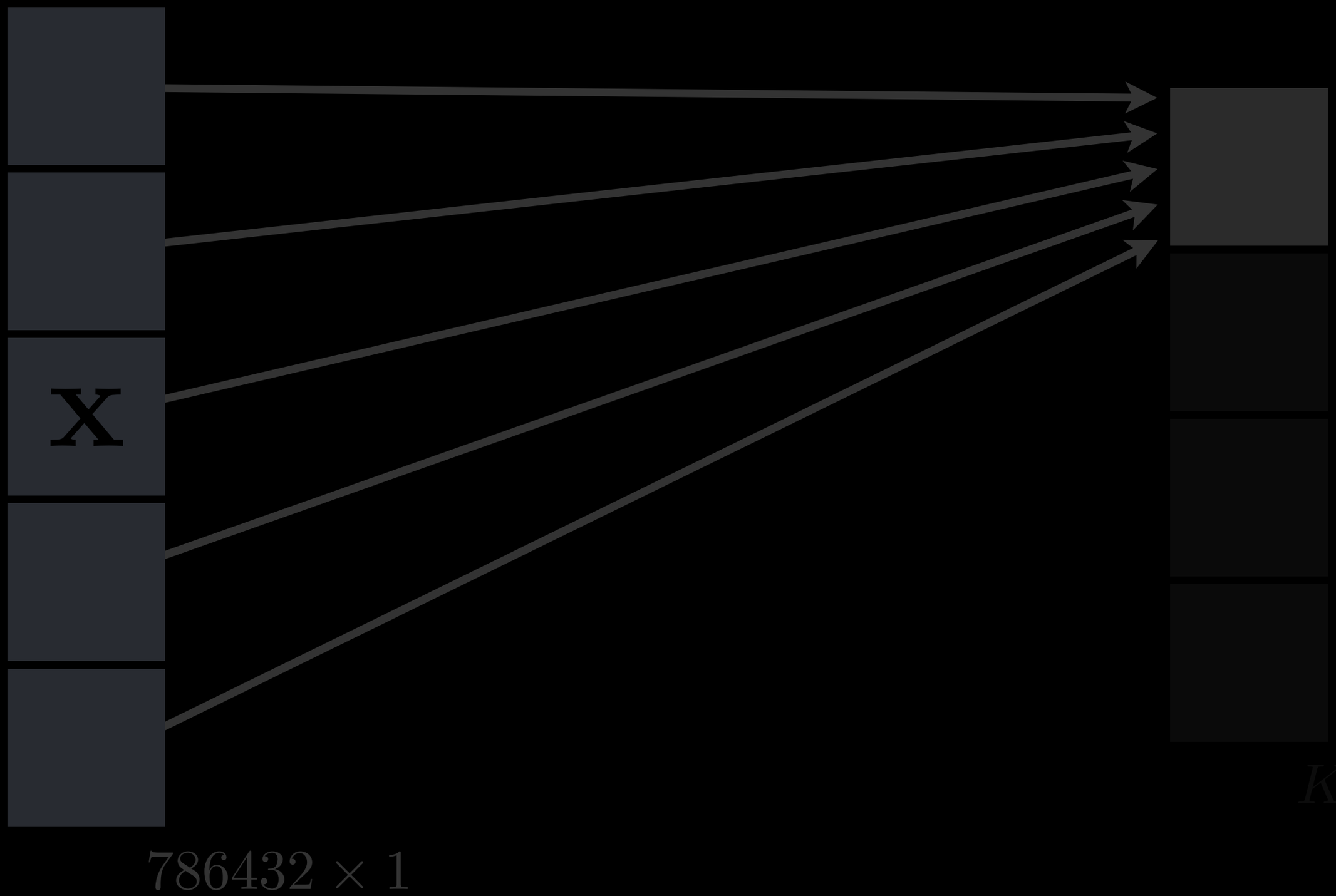


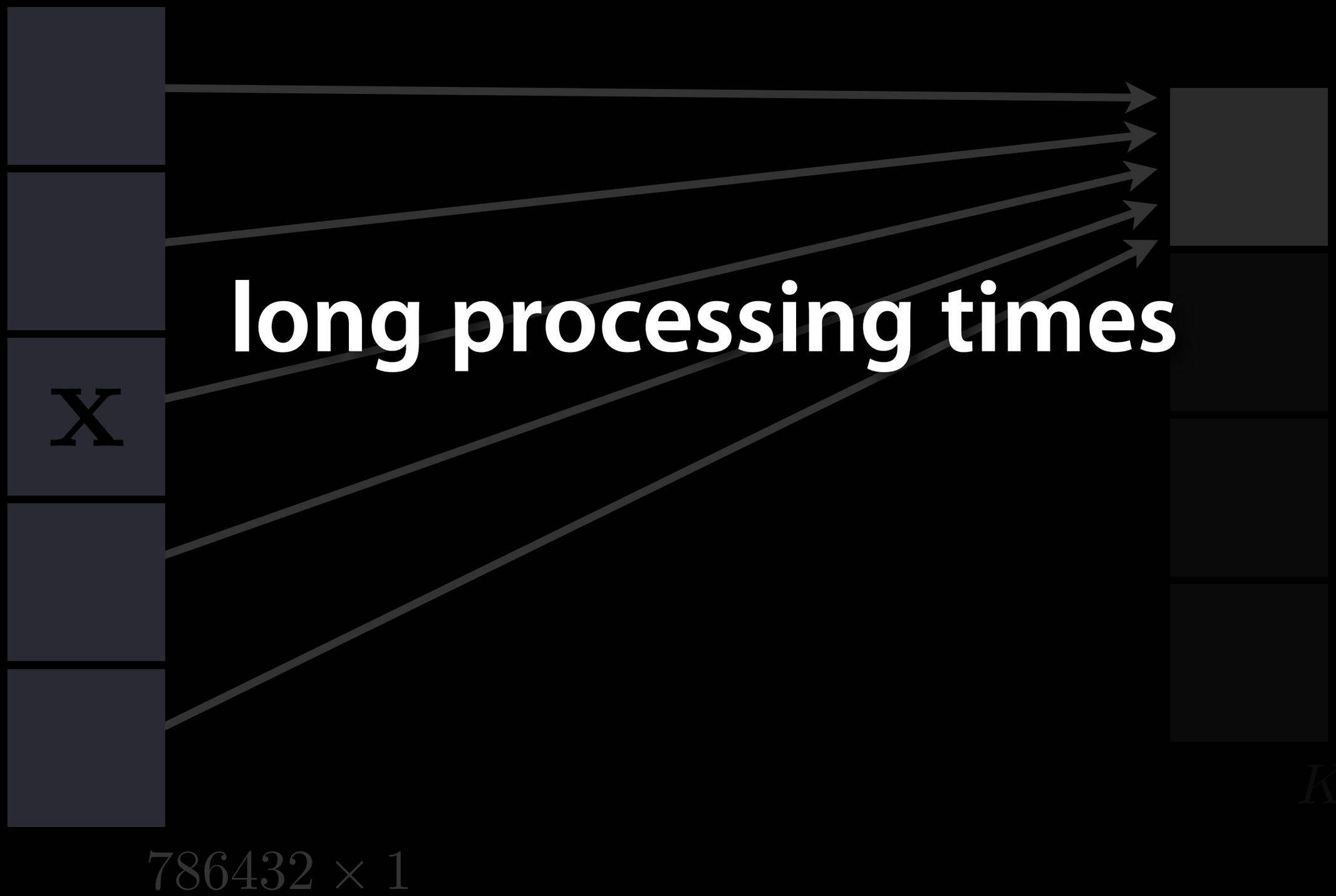
How many parameters per hidden unit?

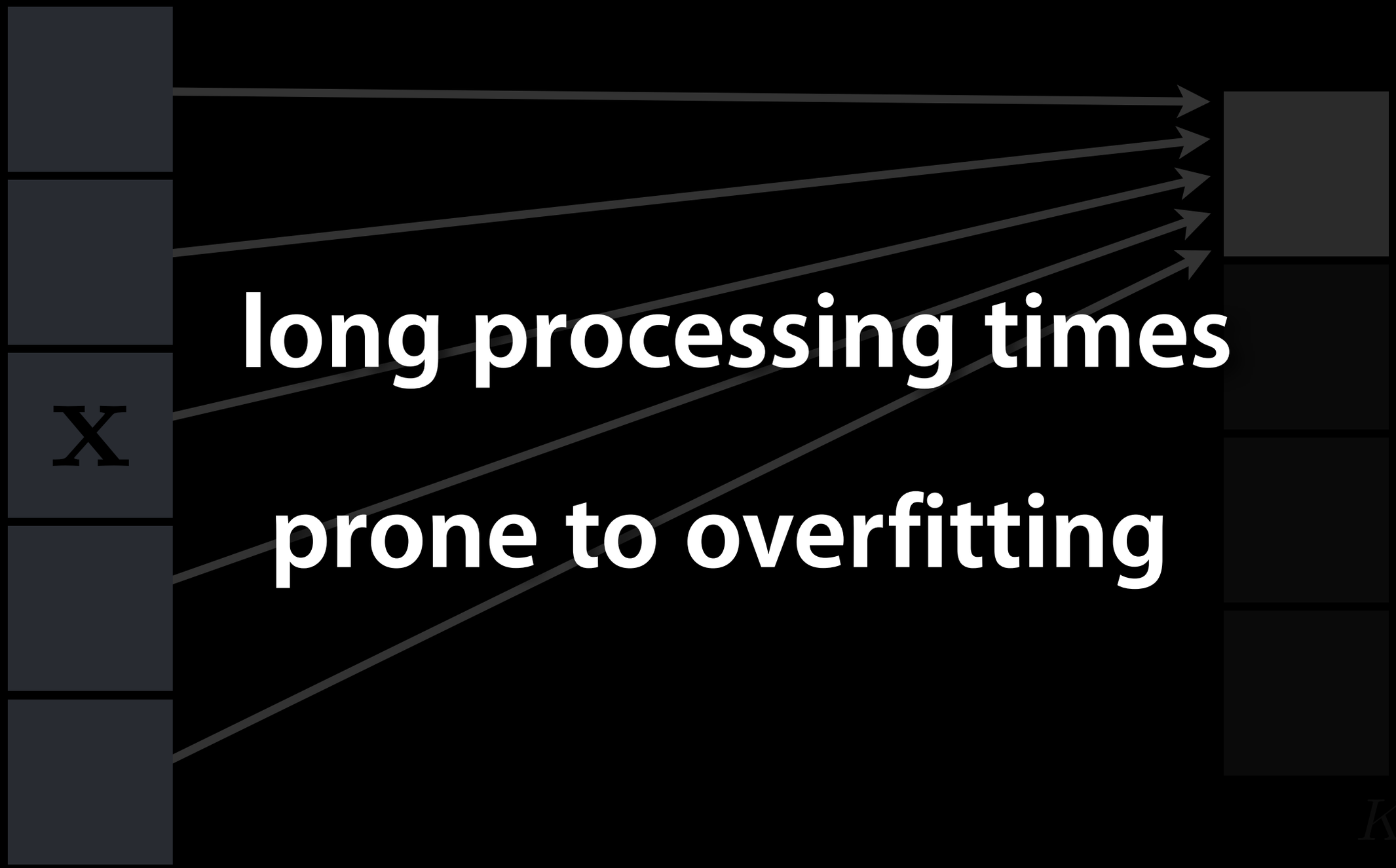

X
786432 × 1
K × 1


~800,000
parameters per hidden unit
786432×1
$K \times 1$


~800,000
parameters per hidden unit
786432×1
$K \times 1$


X
786432×1
$K \times 1$


X
long processing times
786432×1
$K \times 1$



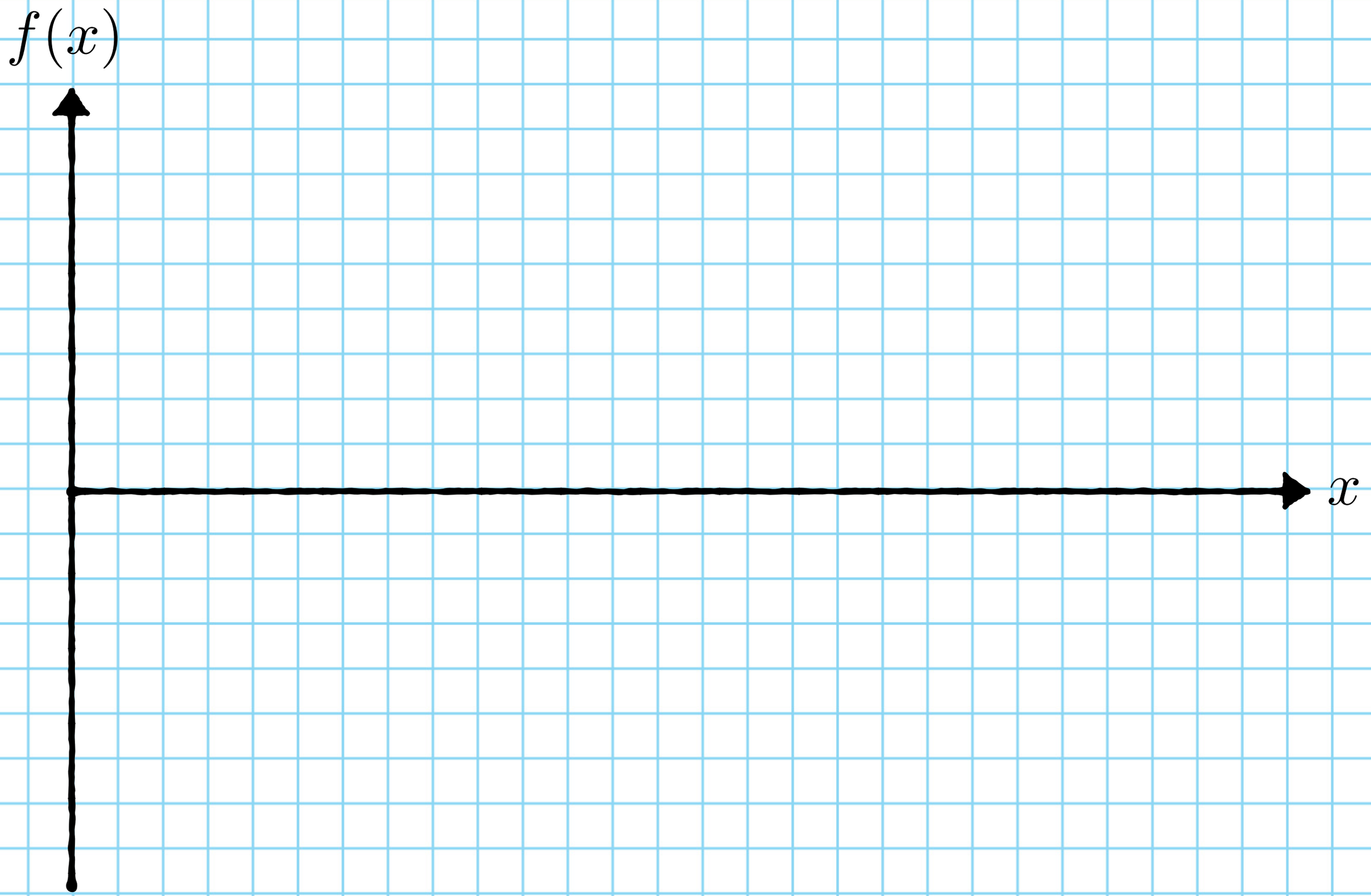
long processing times

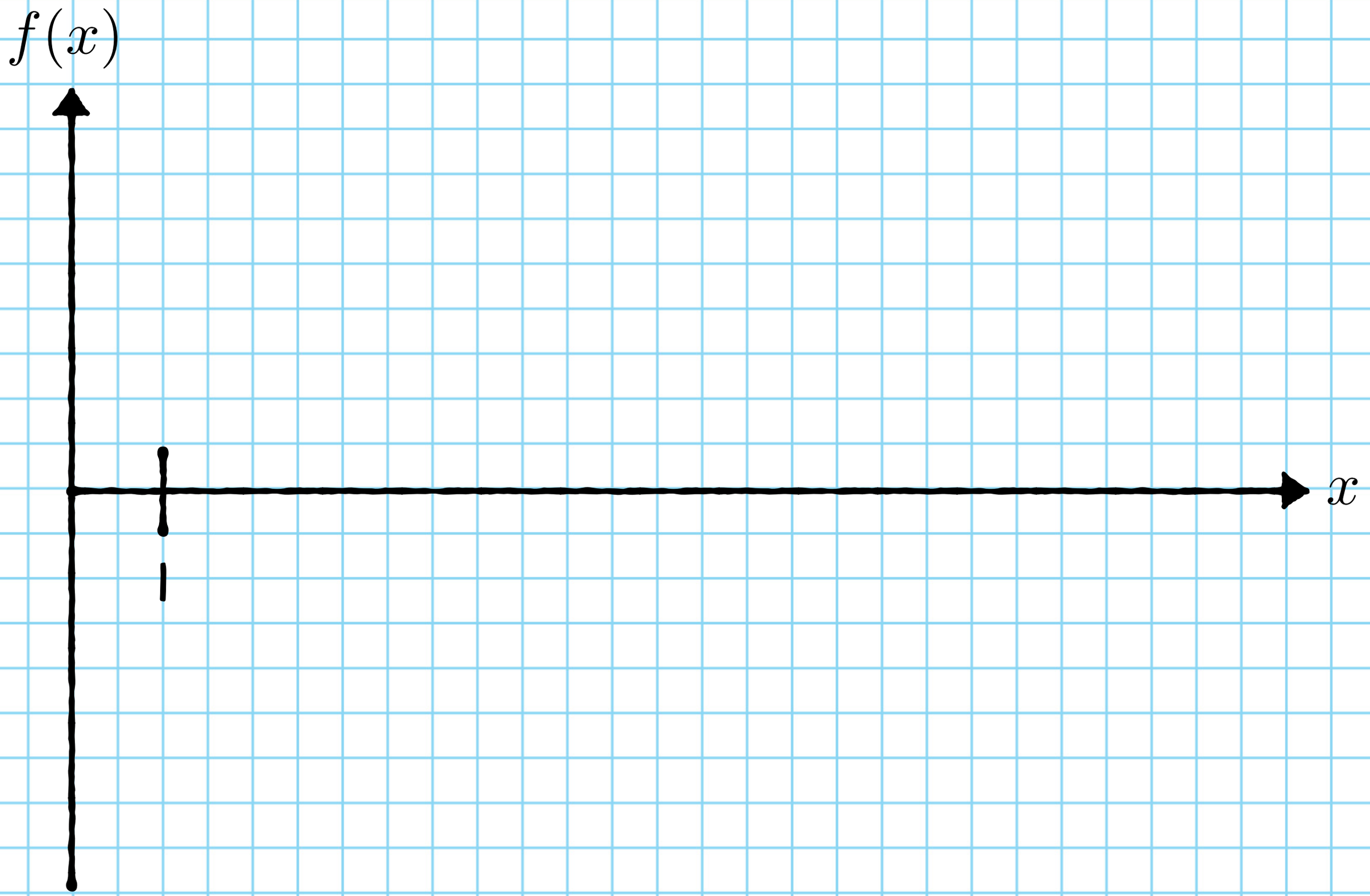
prone to overfitting

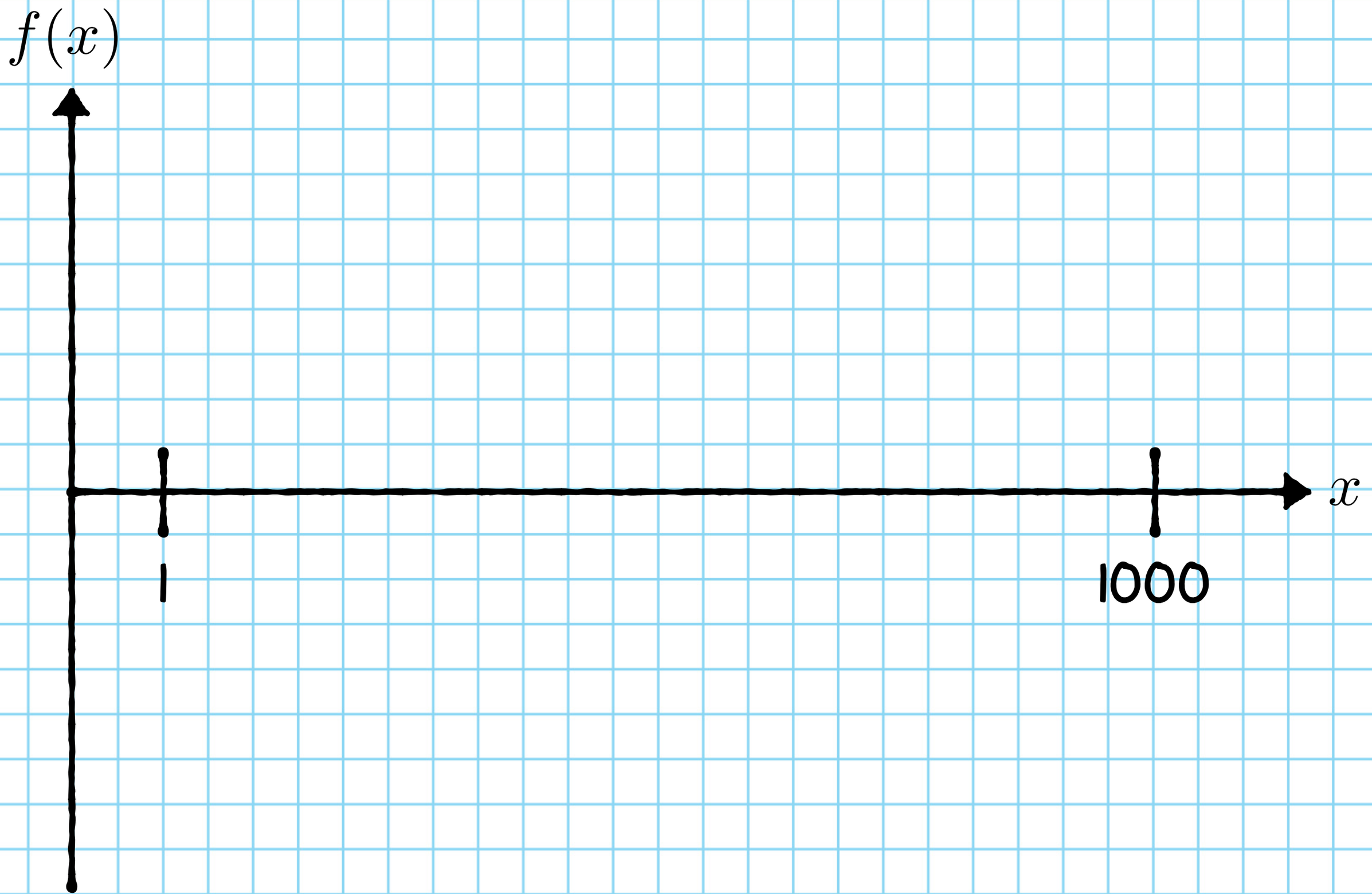
inductive bias

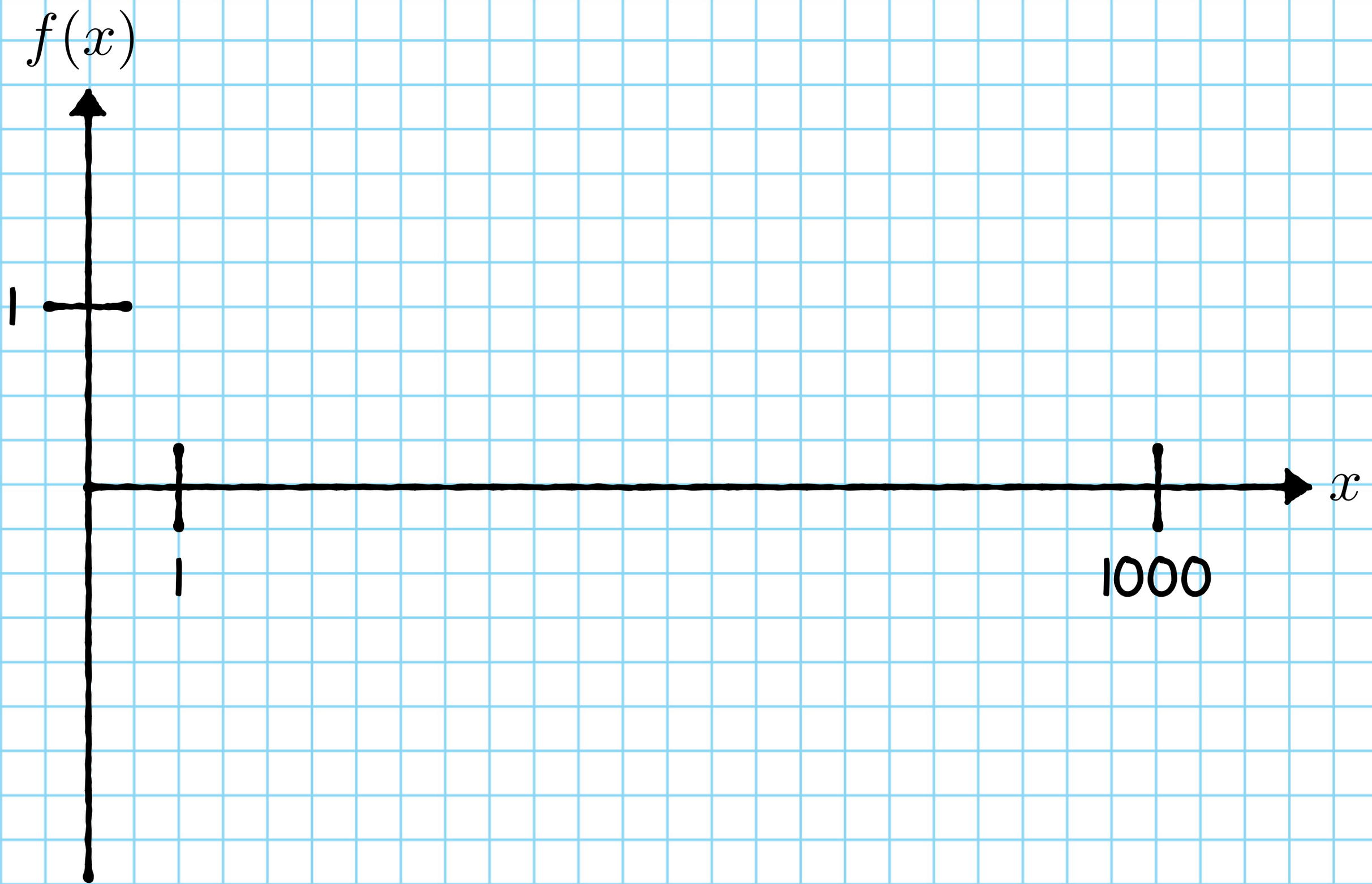
inductive bias

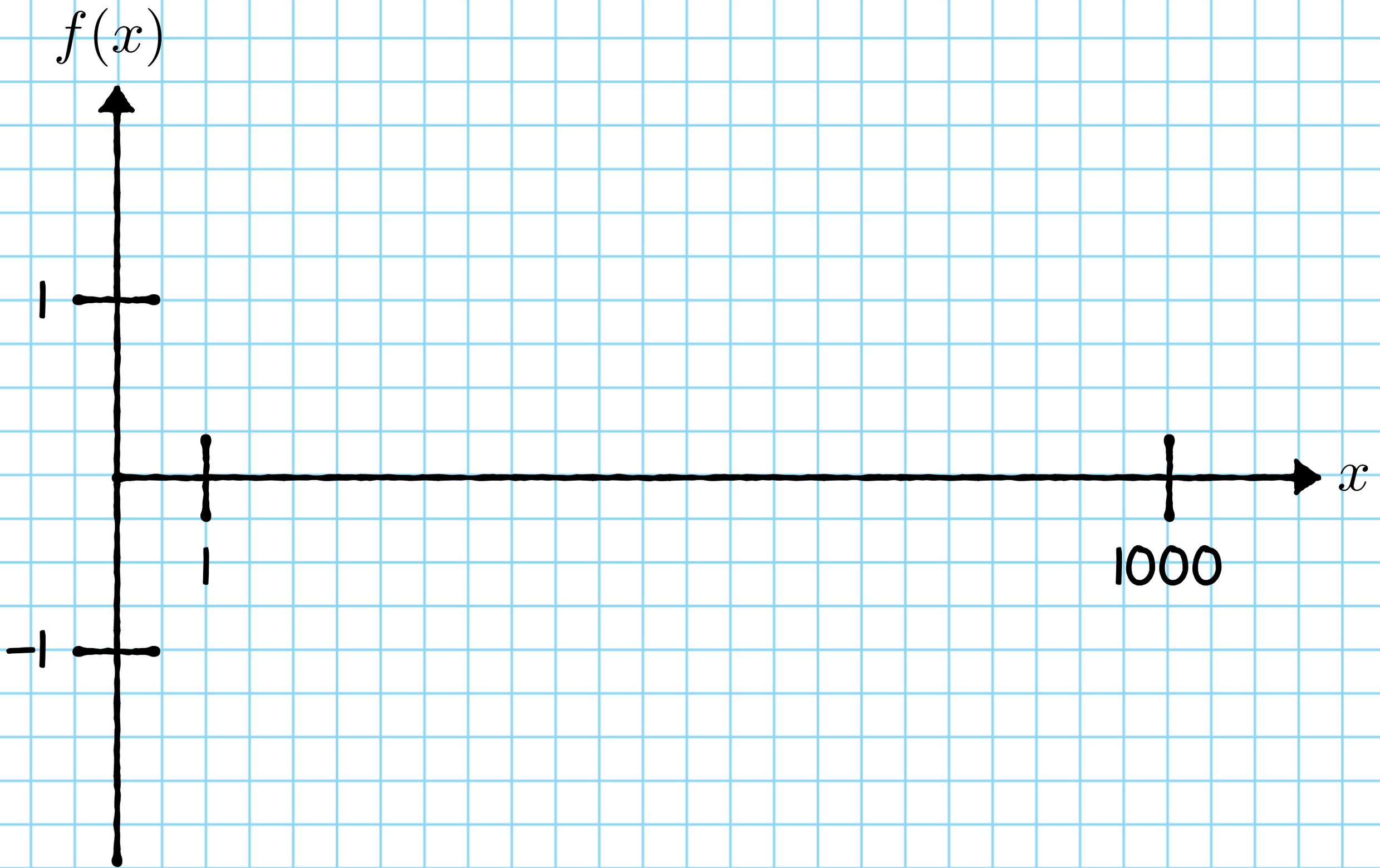
encourages the learning algorithm to prefer solutions with certain properties

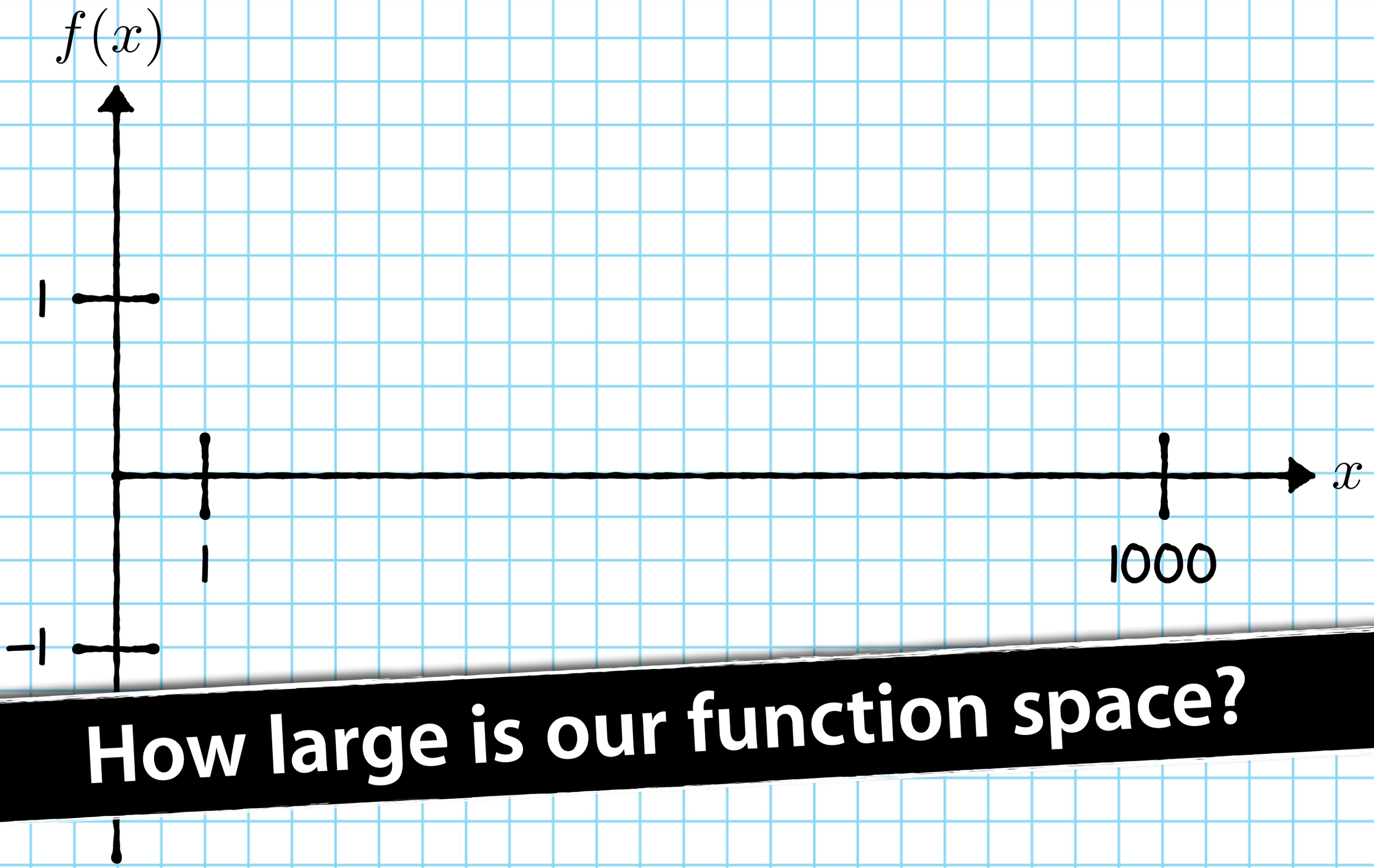

$f(x)$
x


$f(x)$
x


$f(x)$
x
1
1000


$f(x)$
1
1
1000
x


$f(x)$
1
1
x
1000
−1



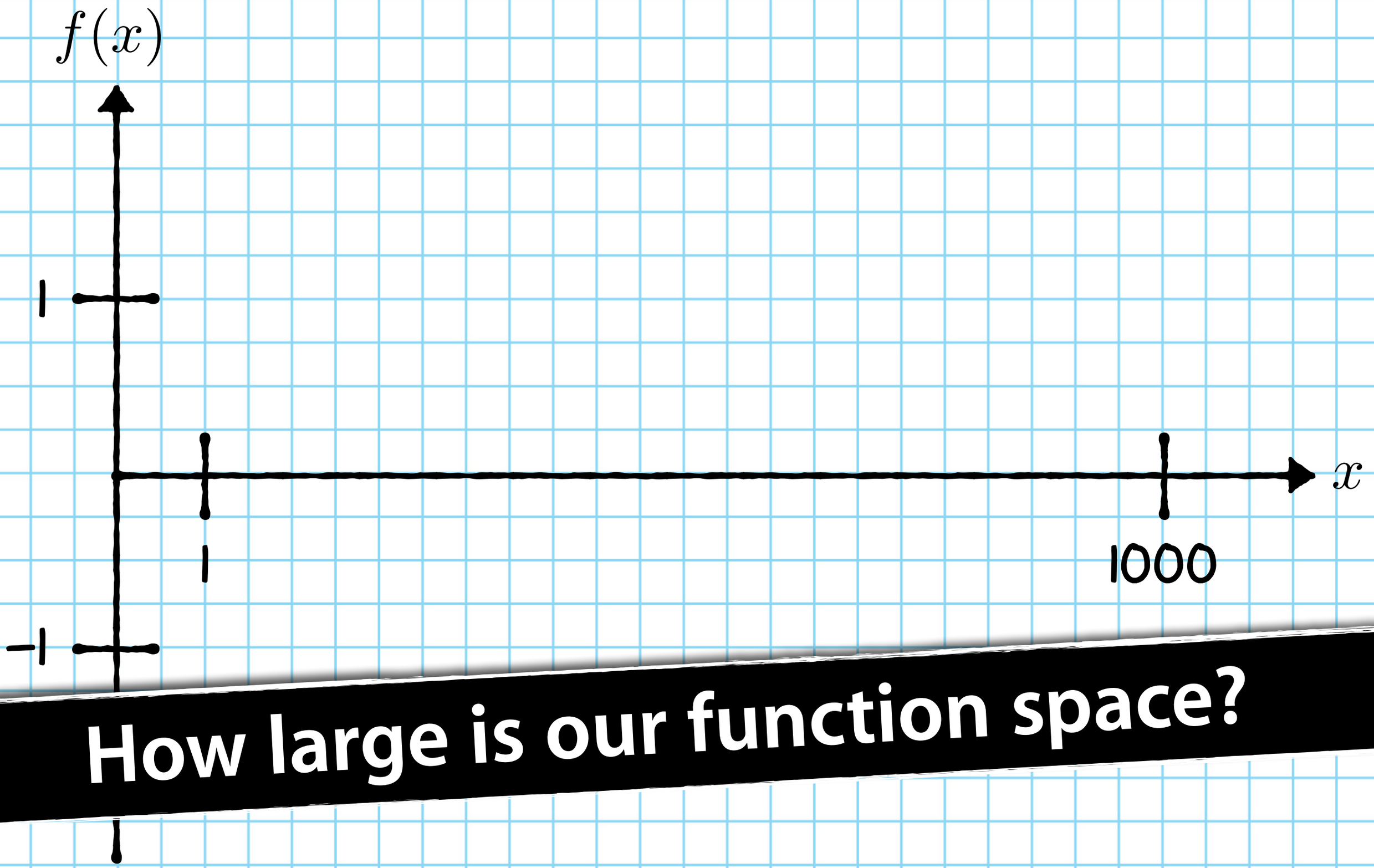
How large is our function space?

$$2^{1000}$$

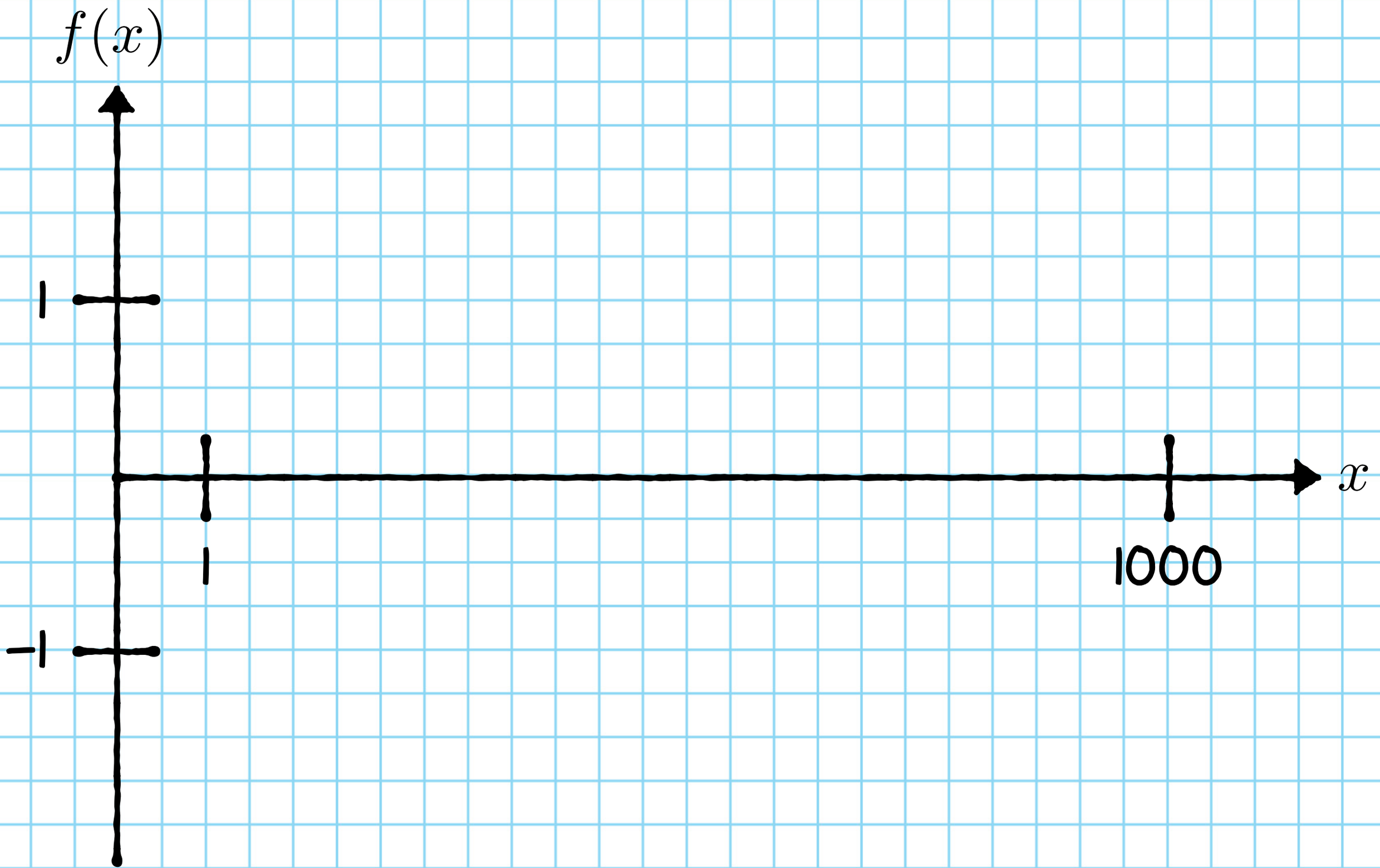
10,715,086,071,862,673,
209,484,250,490,600,018
,105,614,048,117,055,33
6,074,437,503,883,703,5
10,511,249,361,224,931,
983,788,156,958,581,275
,946,729,175,531,468,25
1,871,452,856,923,140,4
35,984,577,574,698,574,

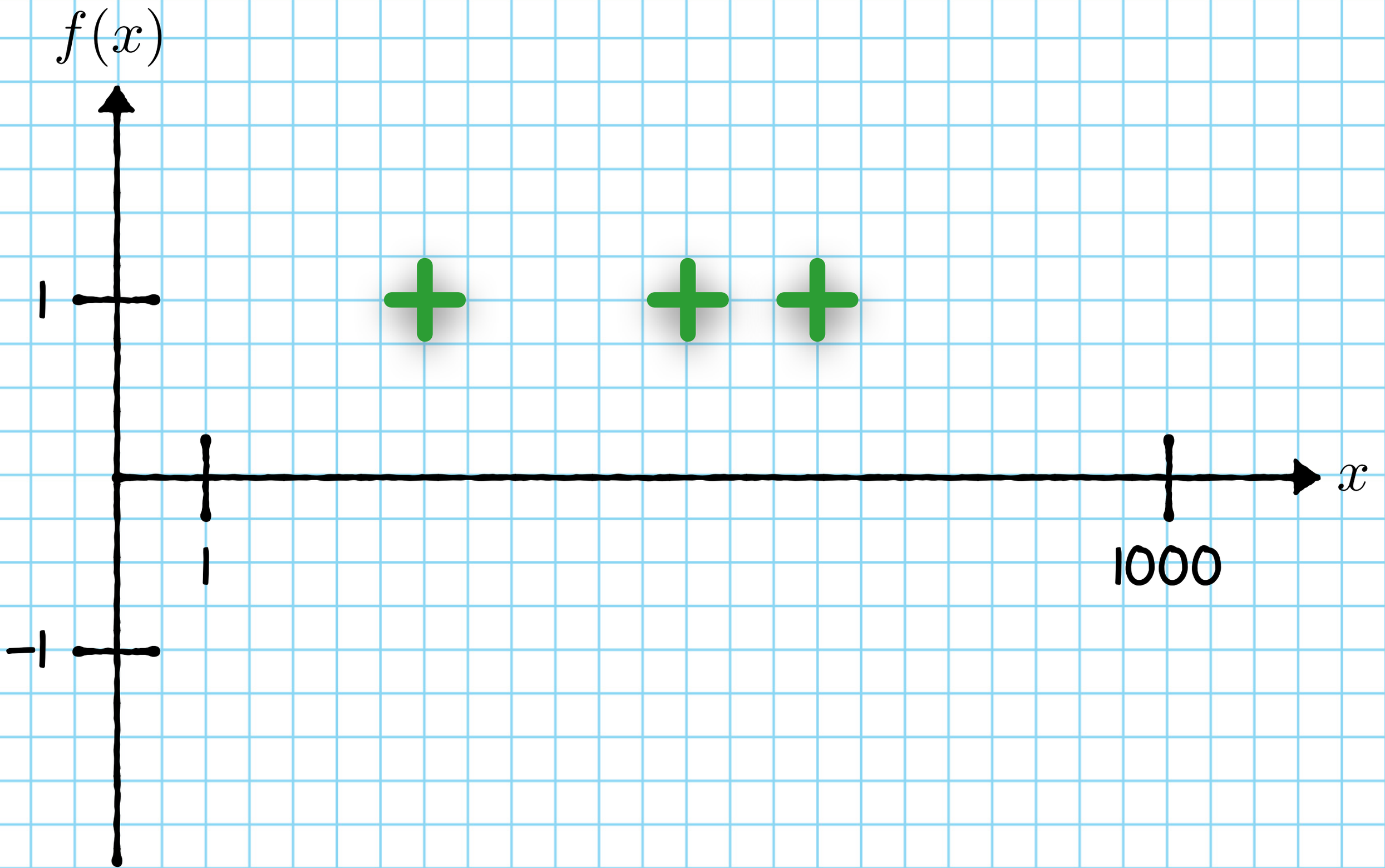
10,715,086,071,862,673,
209,484,250,490,600,018
,105,614,048,117,055,33
6,074,437,503,883,703,5
10,511,249,361,224,931,
983,788,156,958,581,275
,946,729,175,531,468,25
1,871,452,856,923,140,4
35,984,577,574,698,574,

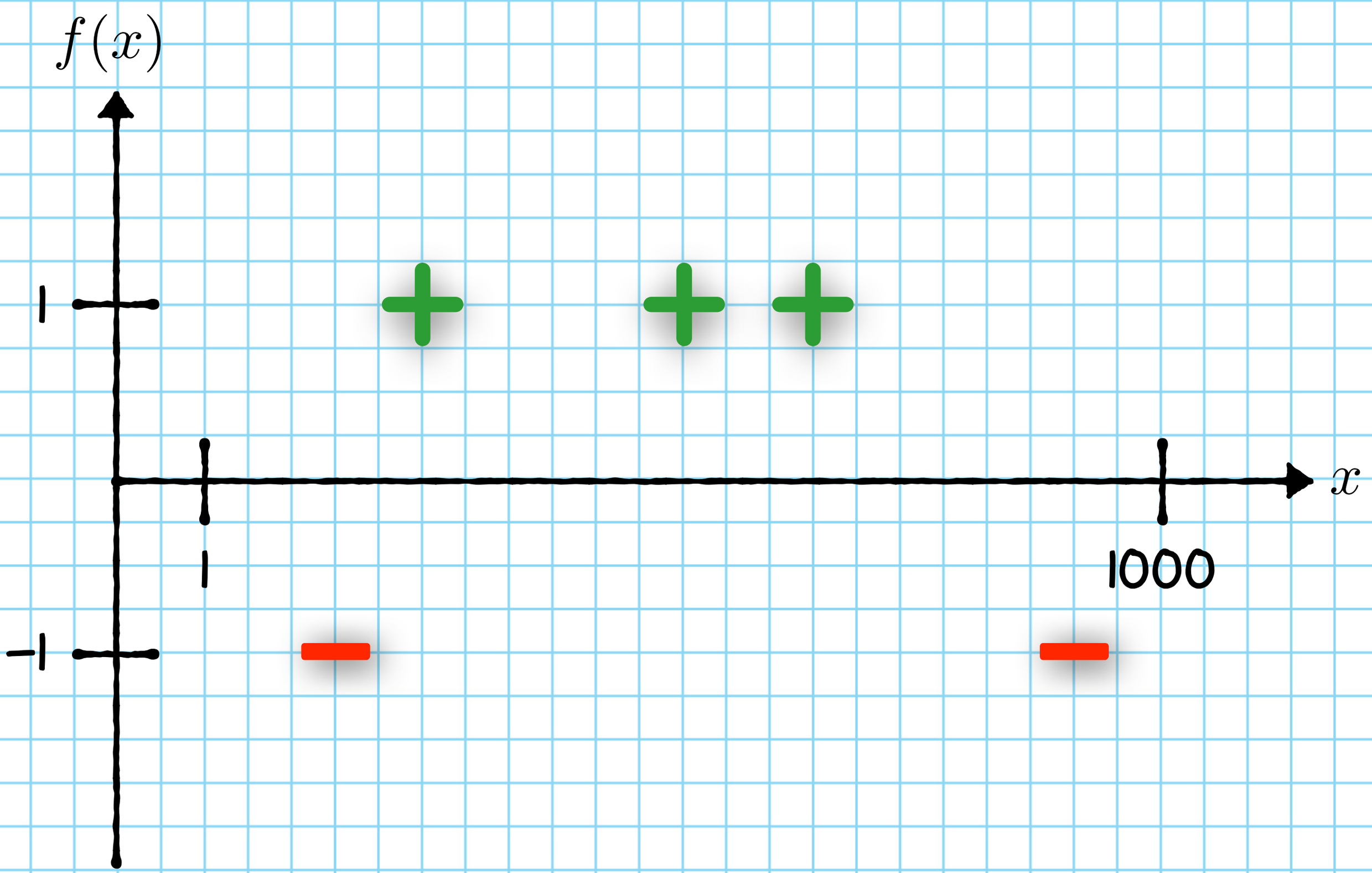
803,934,567,774,824,230
,985,421,074,605,062,37
1,141,877,954,182,153,0
46,474,983,581,941,267,
398,767,559,165,543,946
,077,062,914,571,196,47
7,686,542,167,660,429,8
31,652,624,386,837,205,
668,069,376

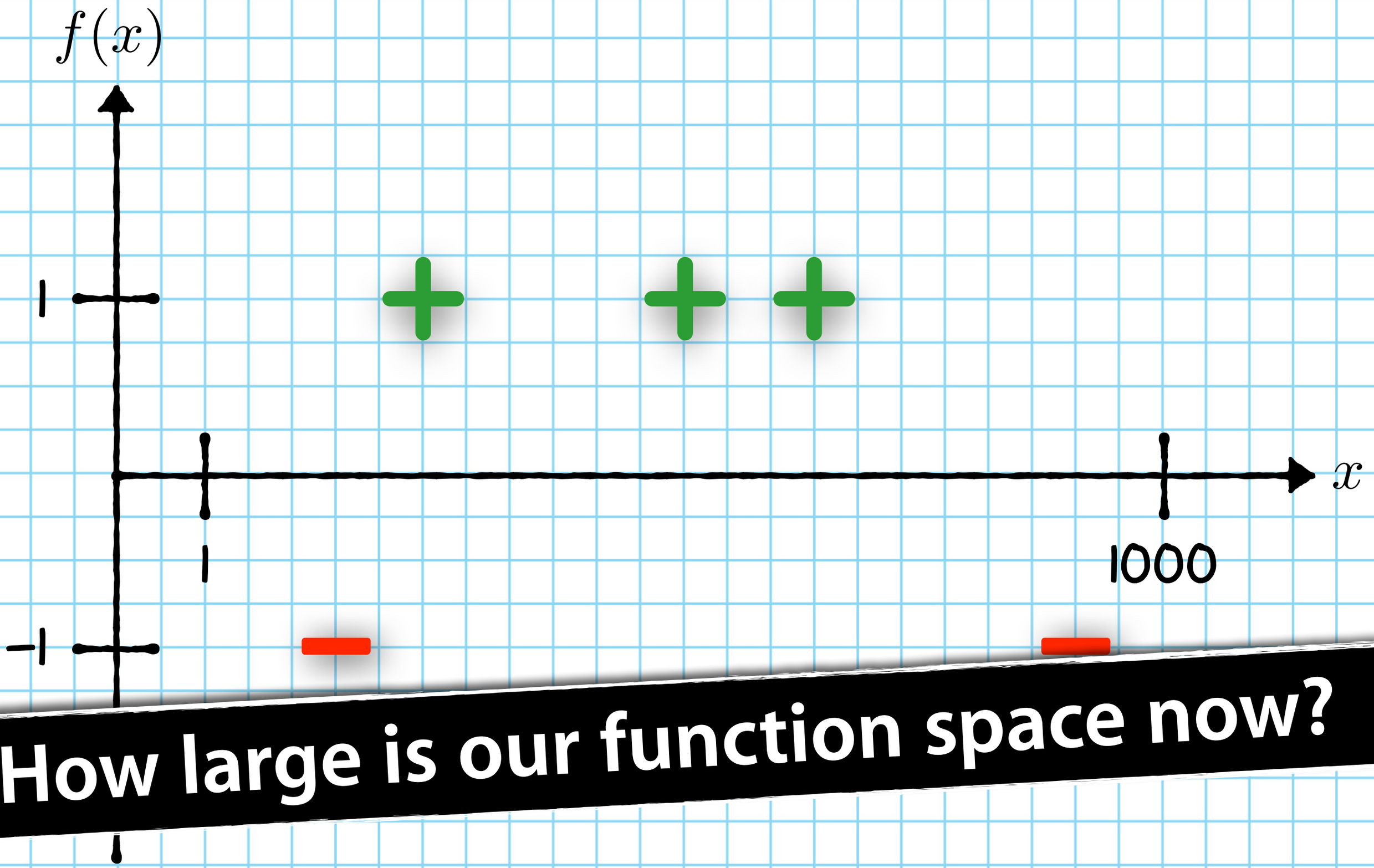


How large is our function space?

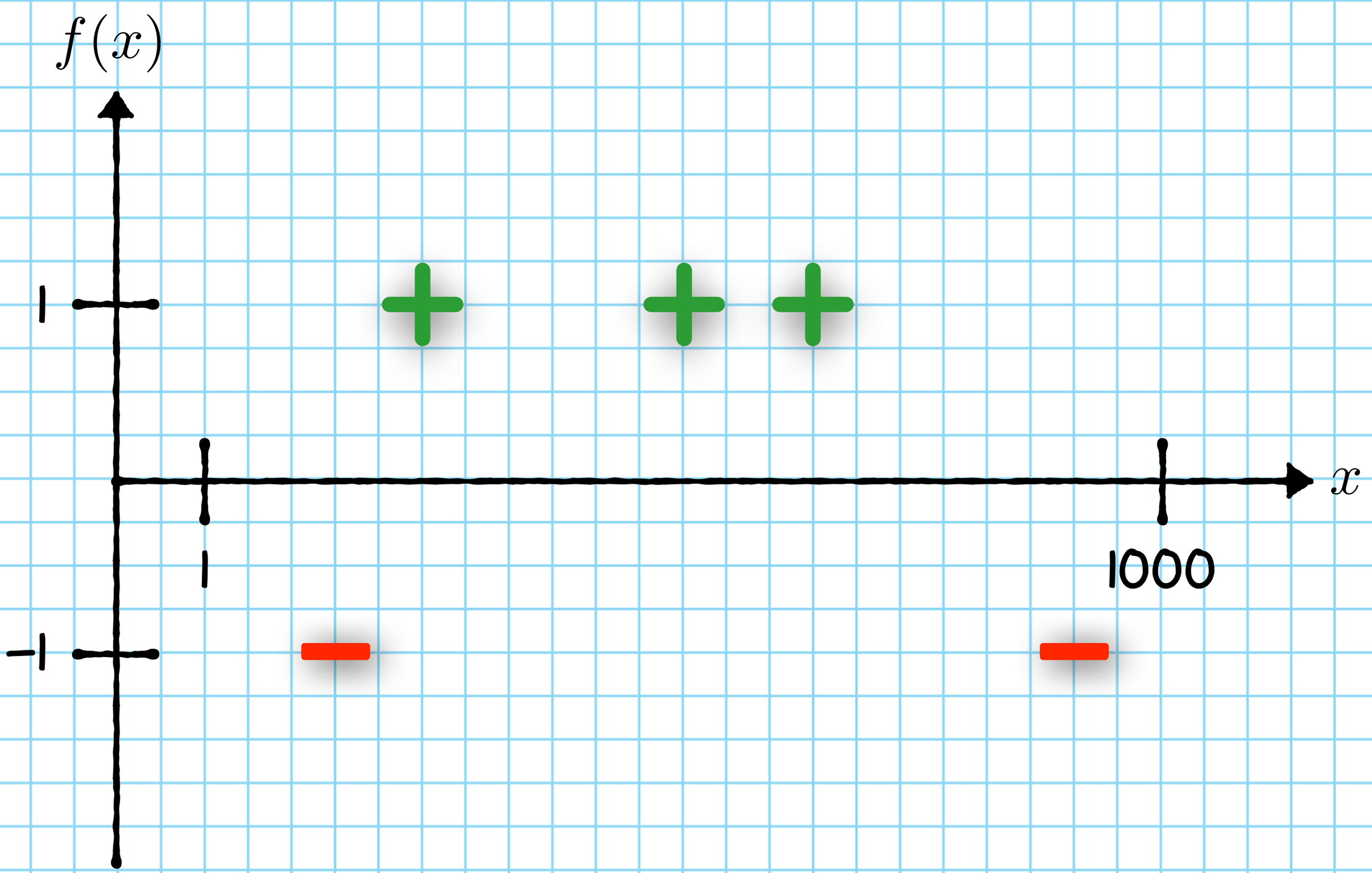

$f(x)$
1
1
x
1000
−1

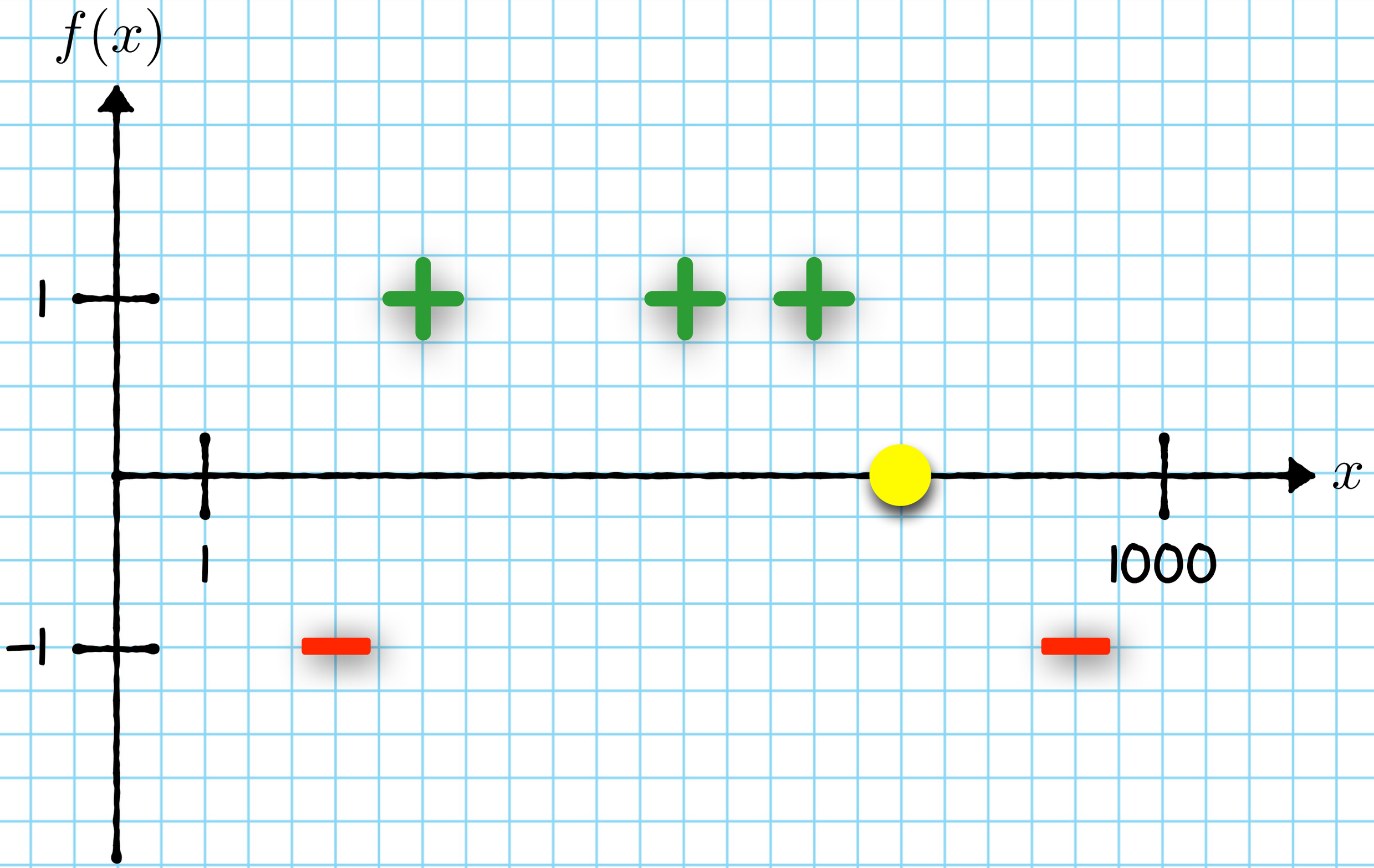

$f(x)$
1
1
−1
1000
x


$f(x)$
1
1
x
1000
−1



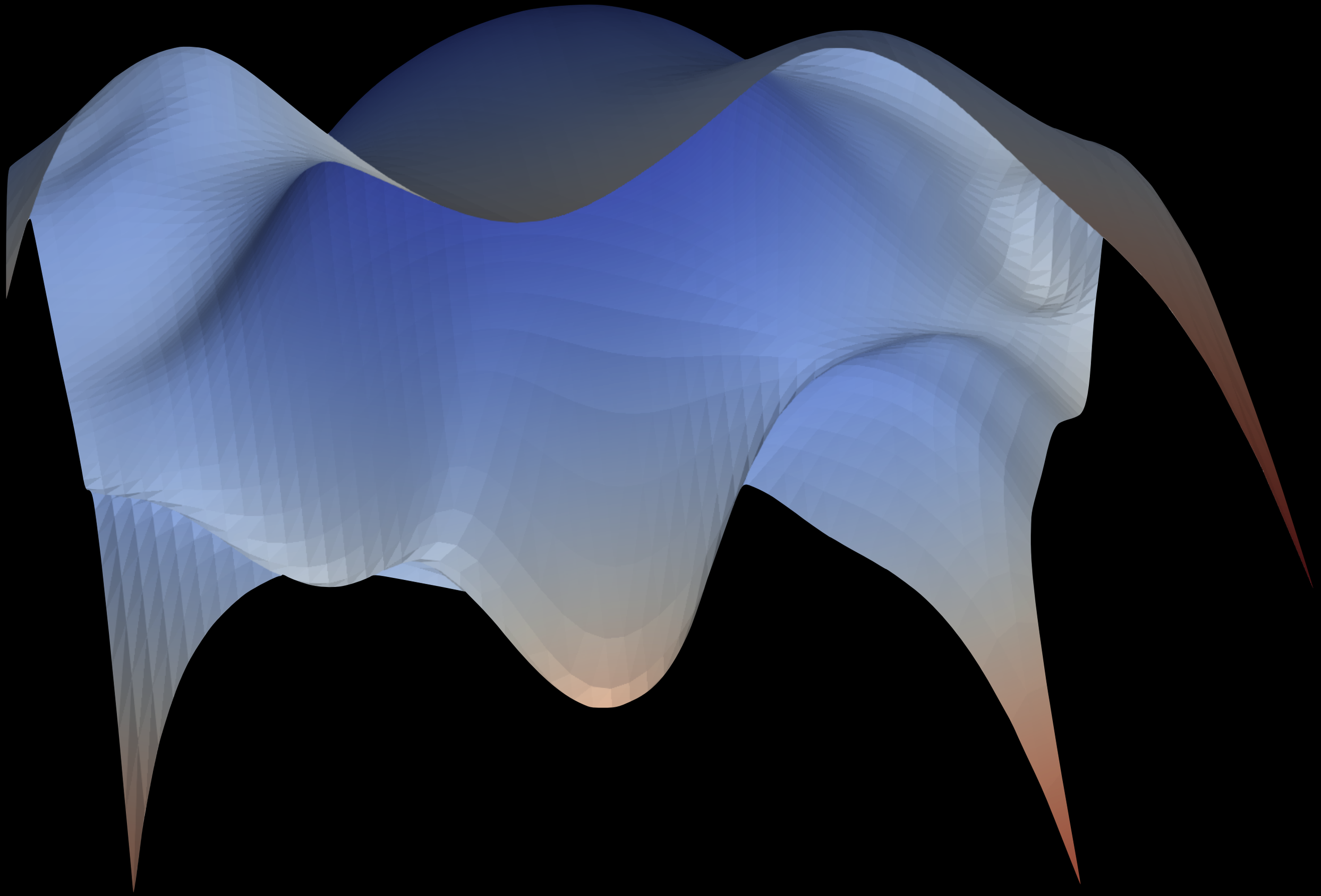
How large is our function space now?


$f(x)$
1
1
x
1000
−1


$f(x)$
1
−1
1
1000
x

Inductive Bias

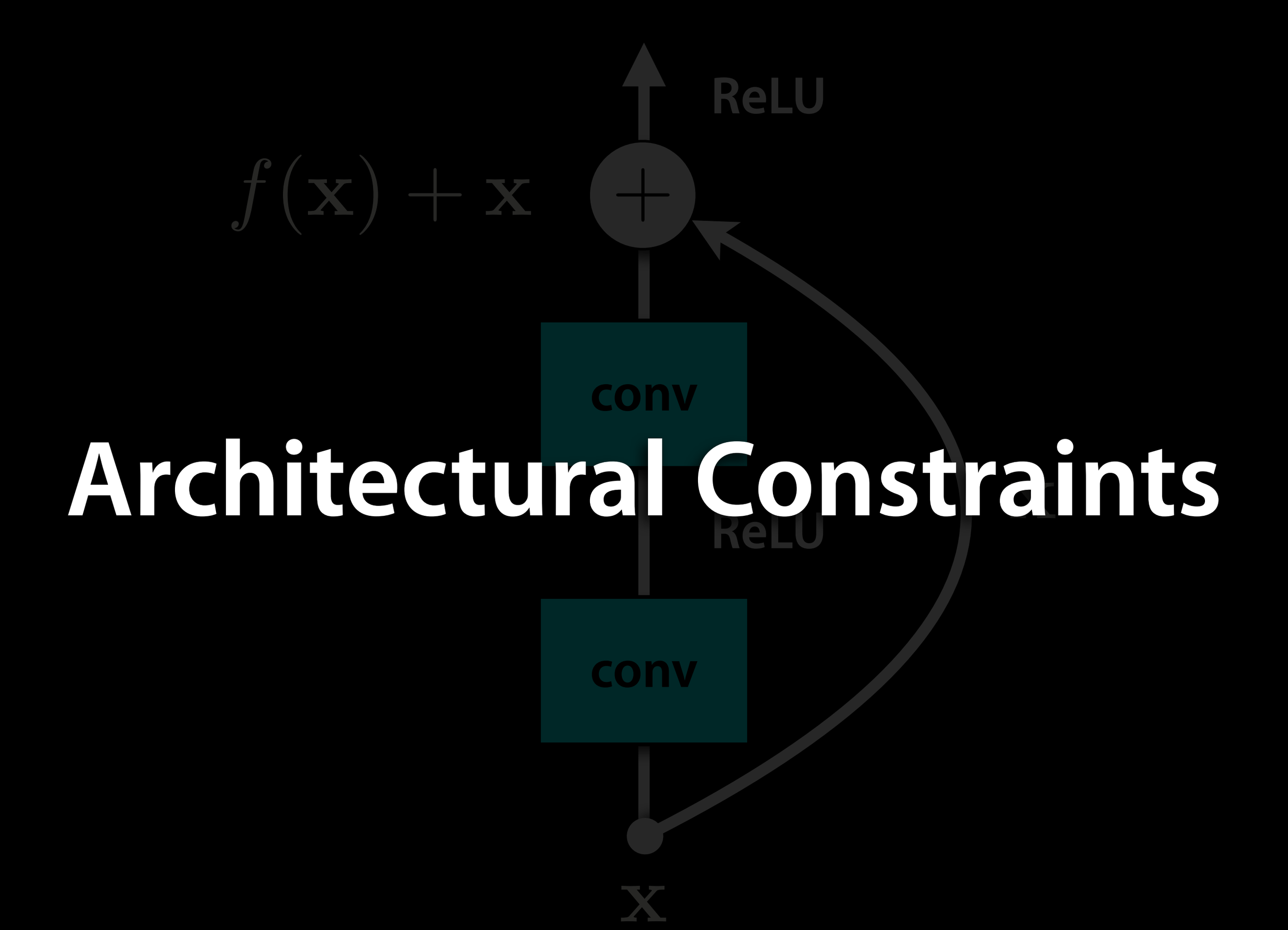
Examples



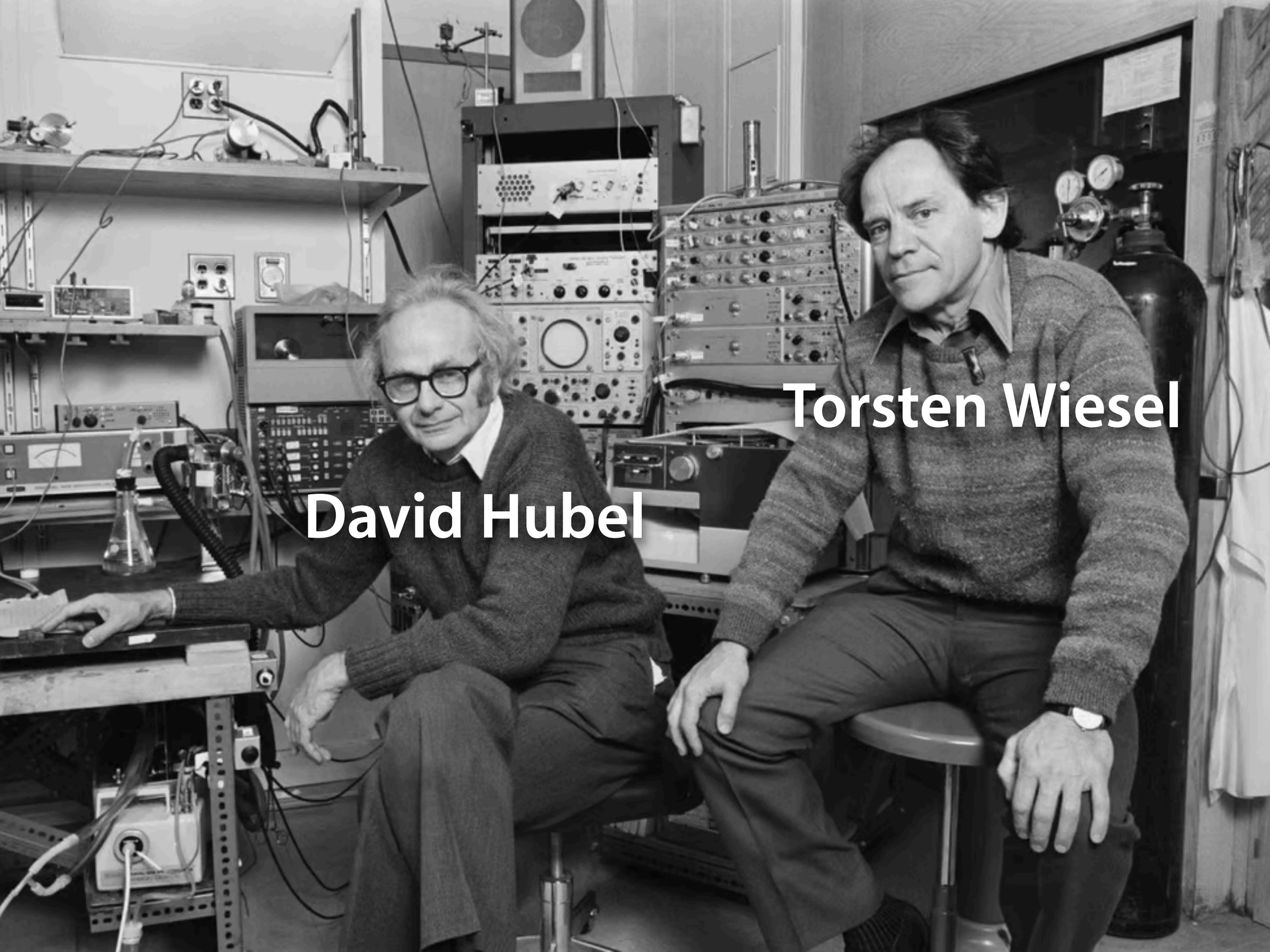
Regularize Objective

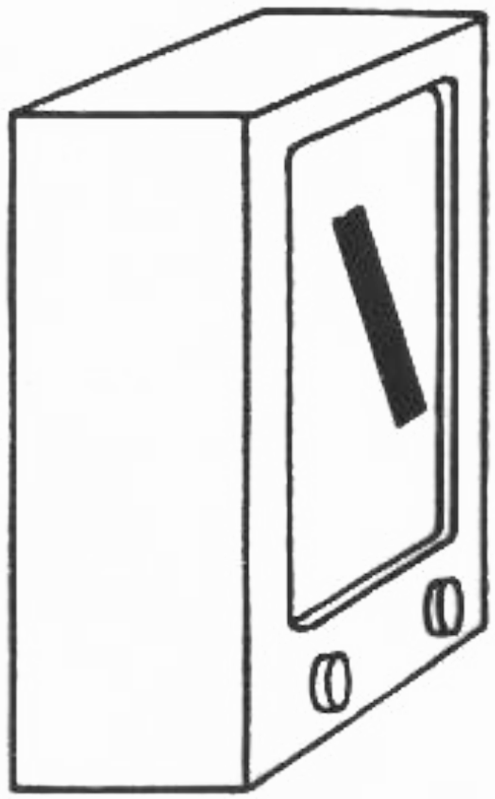
$$\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} \text{Loss}(\hat{y}_i, y_i) + \lambda R(\mathbf{w})$$

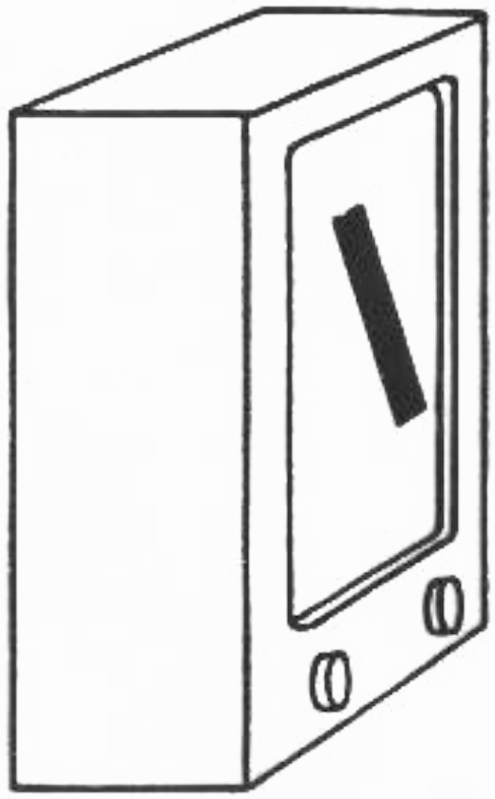
Image Augmentations

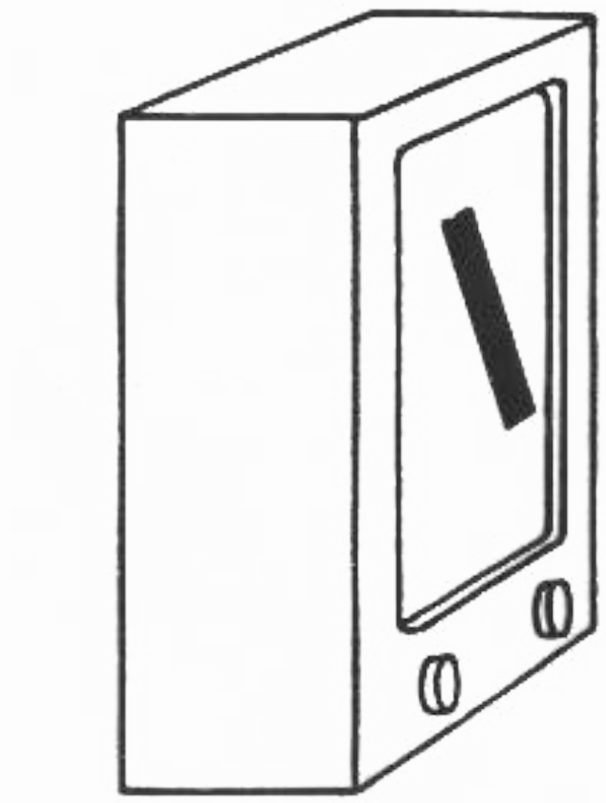


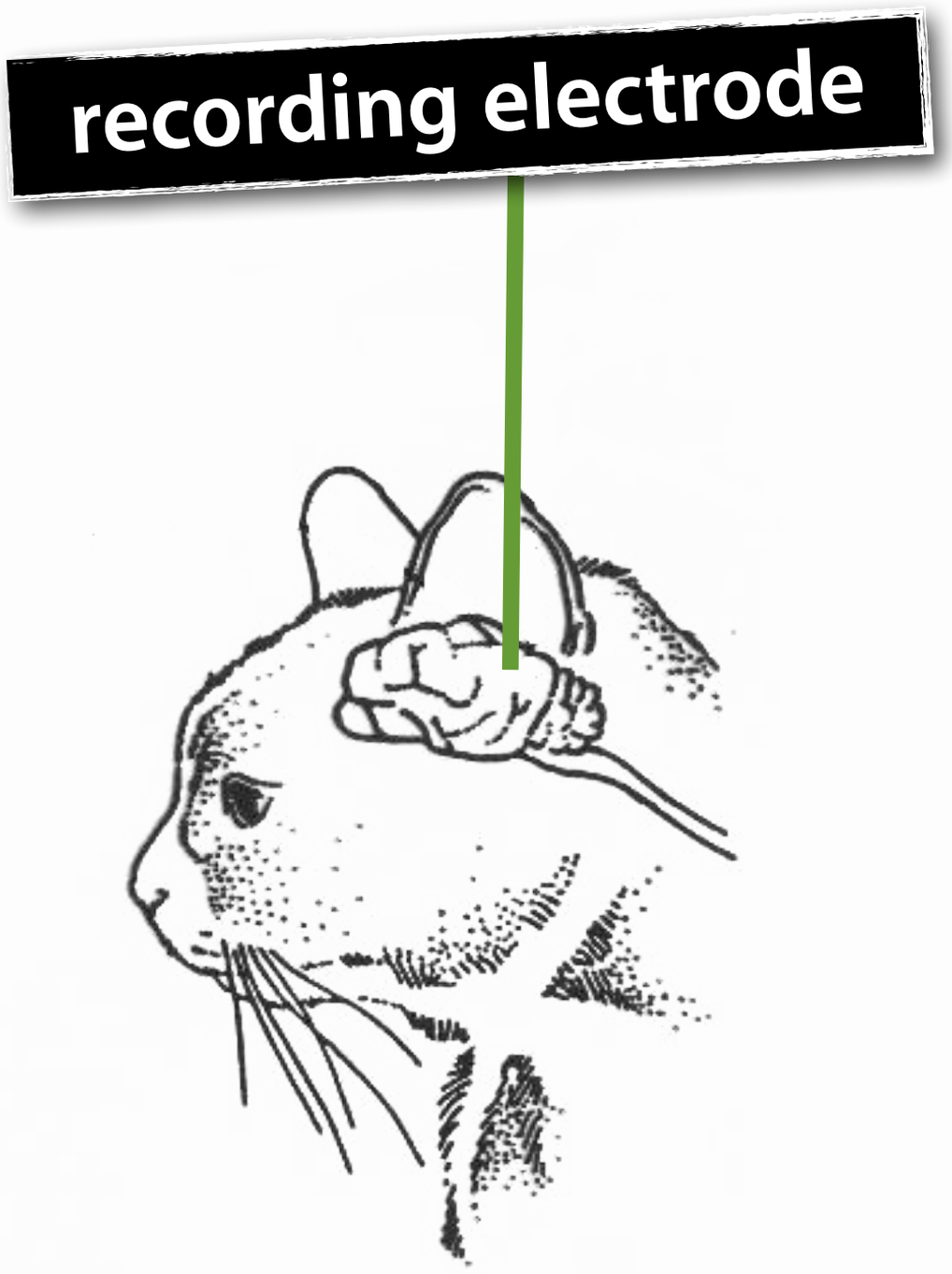
Architectural Constraints

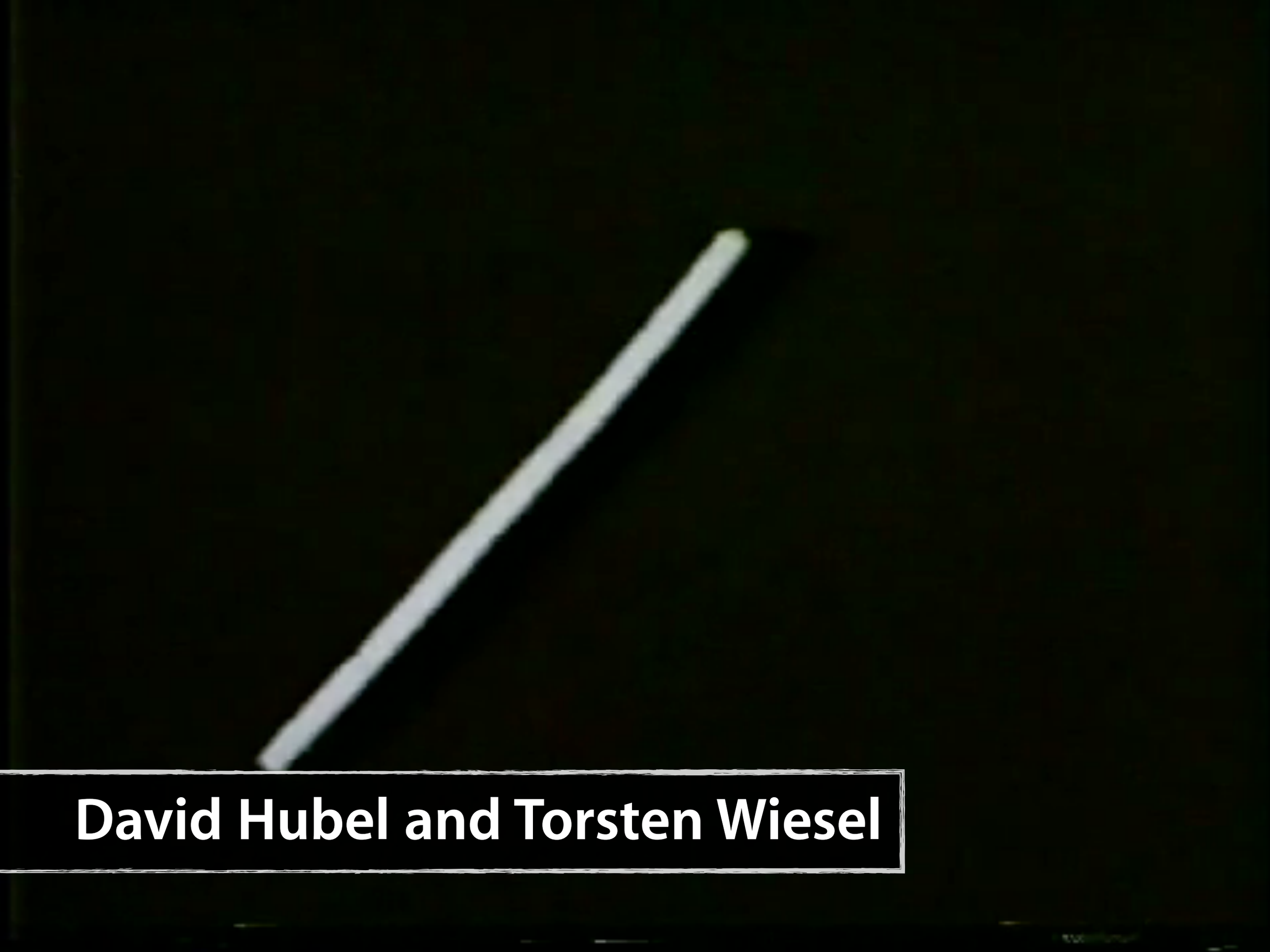

Torsten Wiesel
David Hubel








recording electrode



David Hubel and Torsten Wiesel

neurons are spatially localized

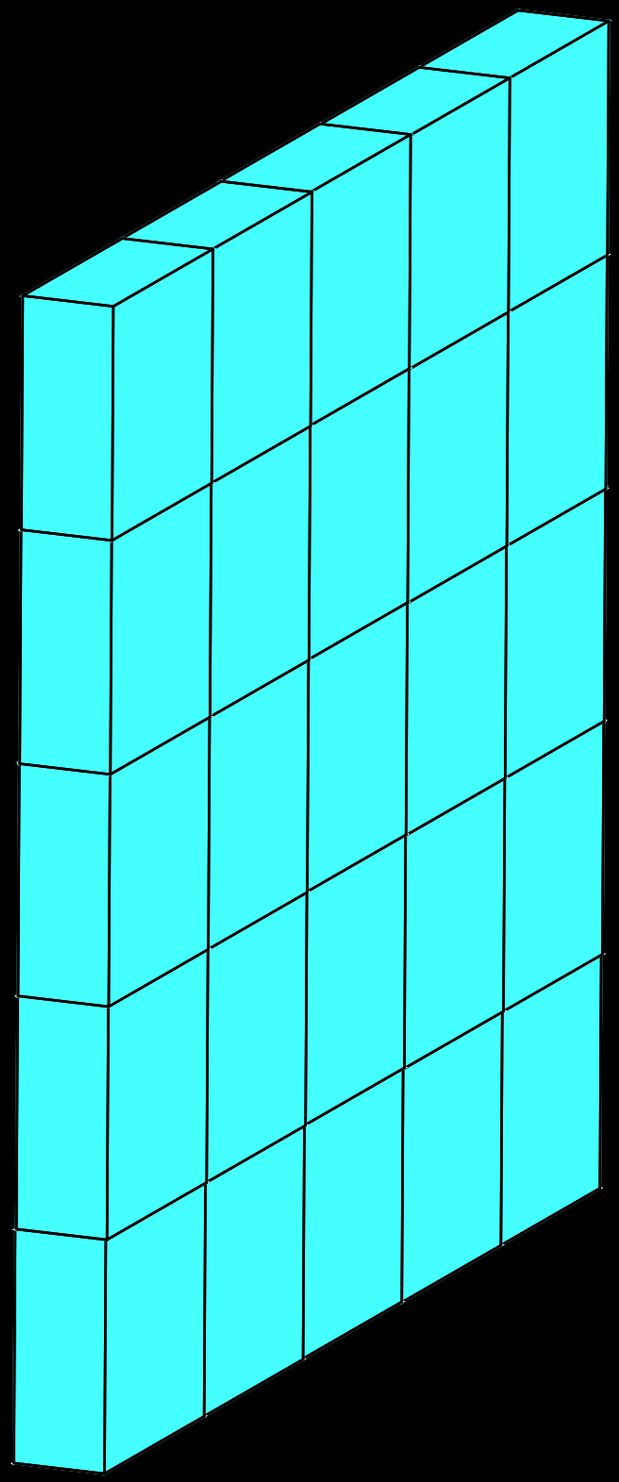
neurons are spatially localized

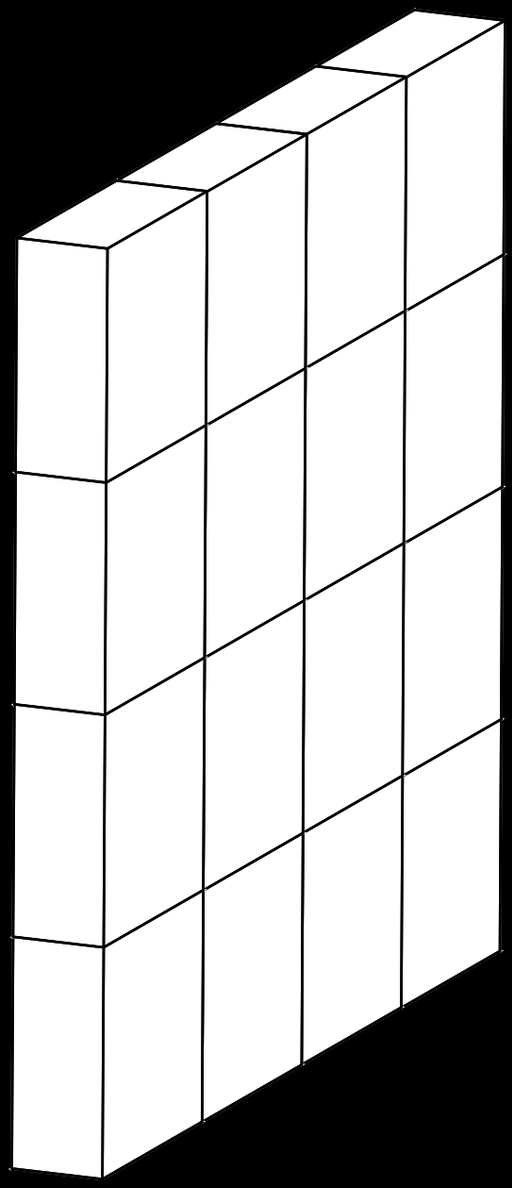
topographic feature maps

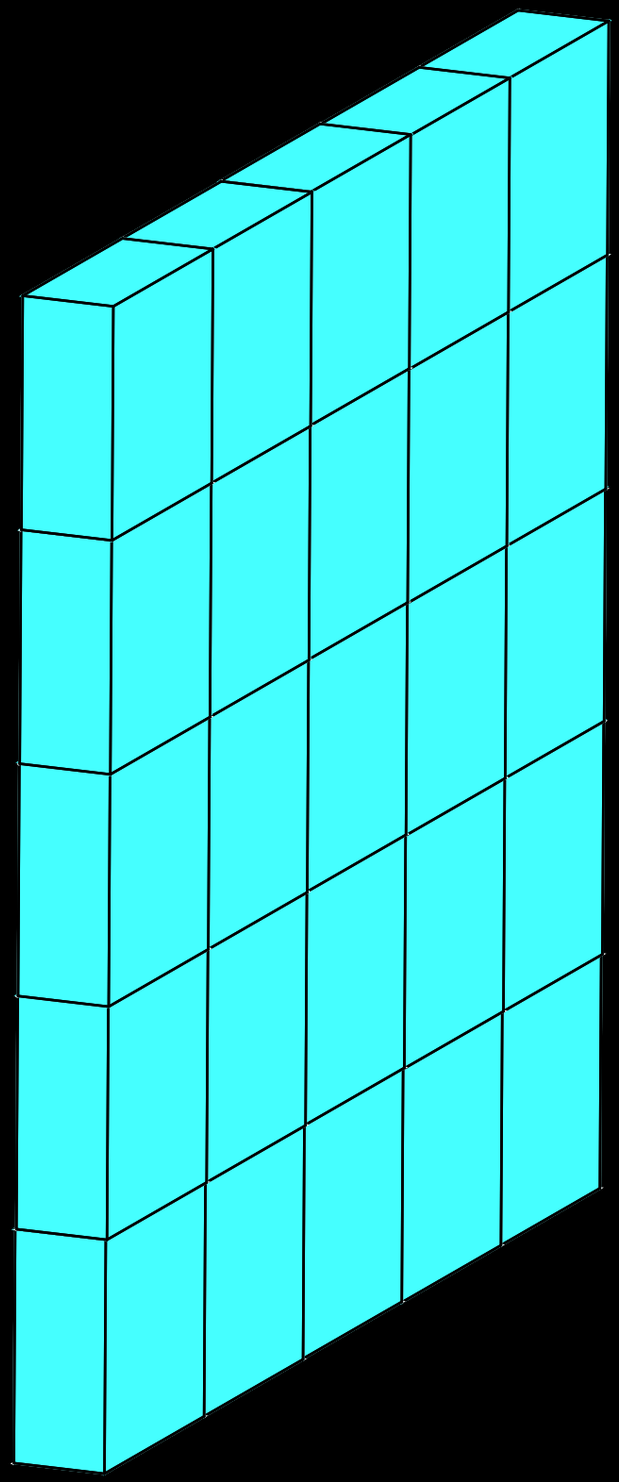
neurons are spatially localized

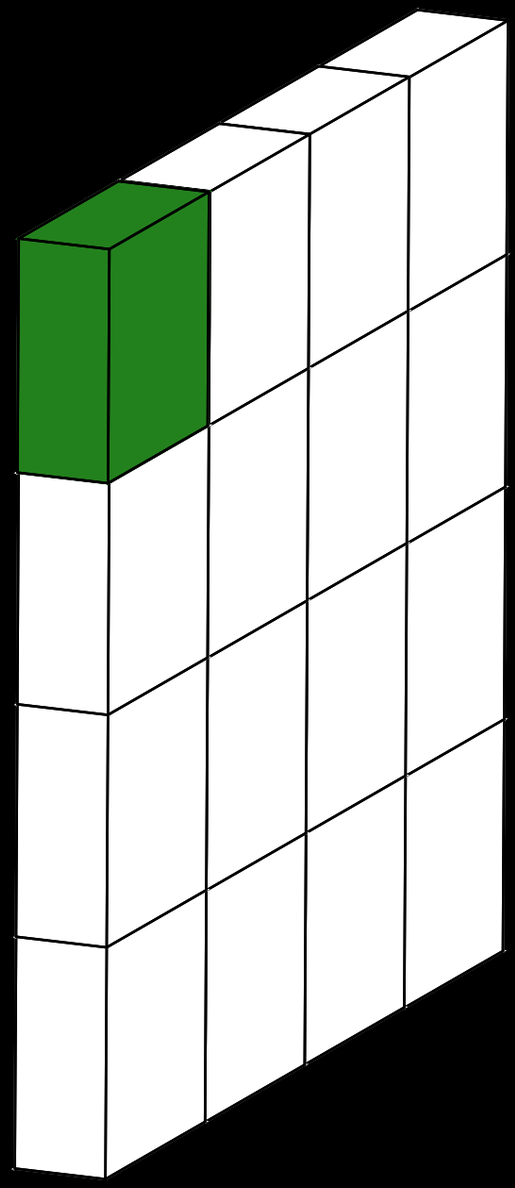
topographic feature maps

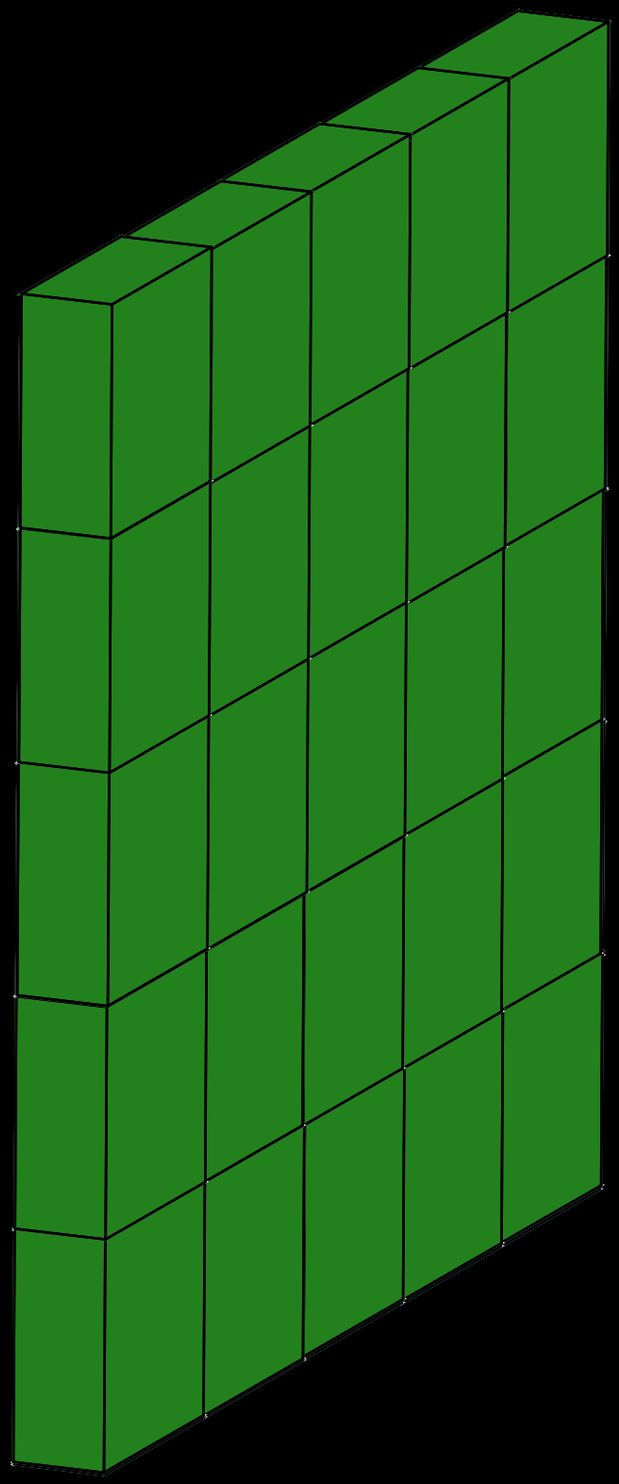
hierarchical feature processing

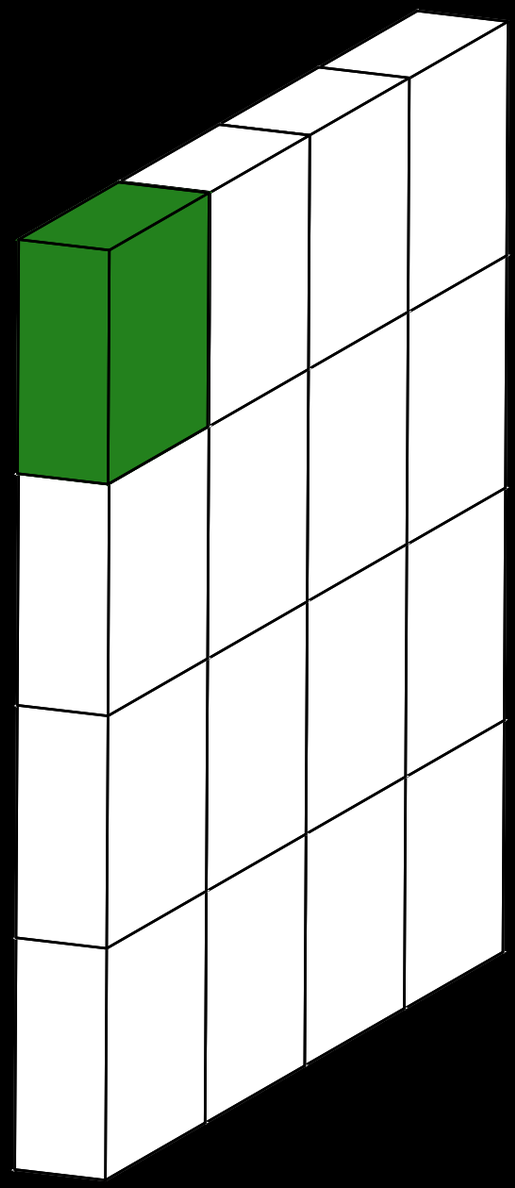


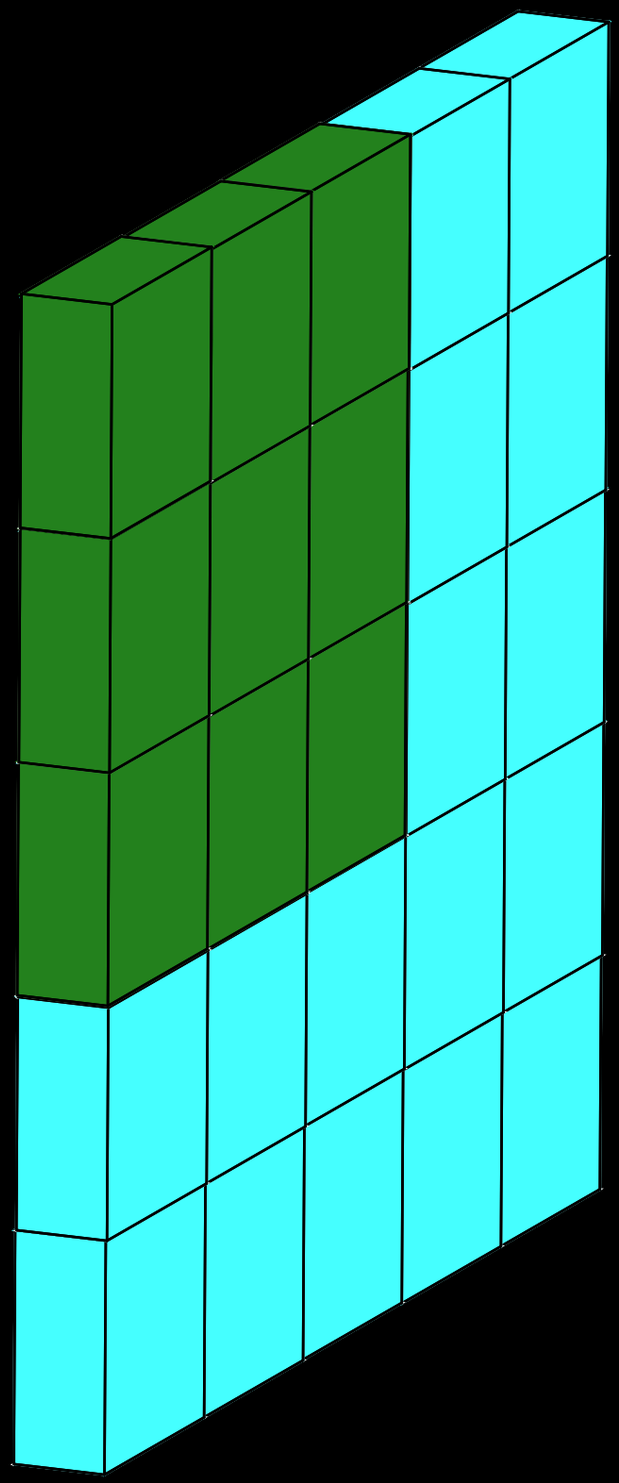


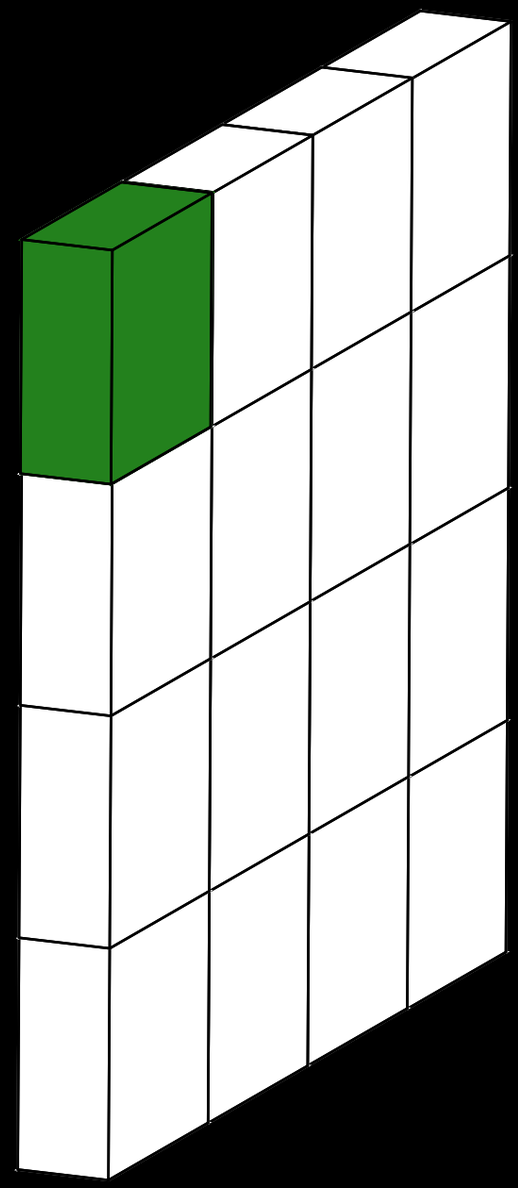


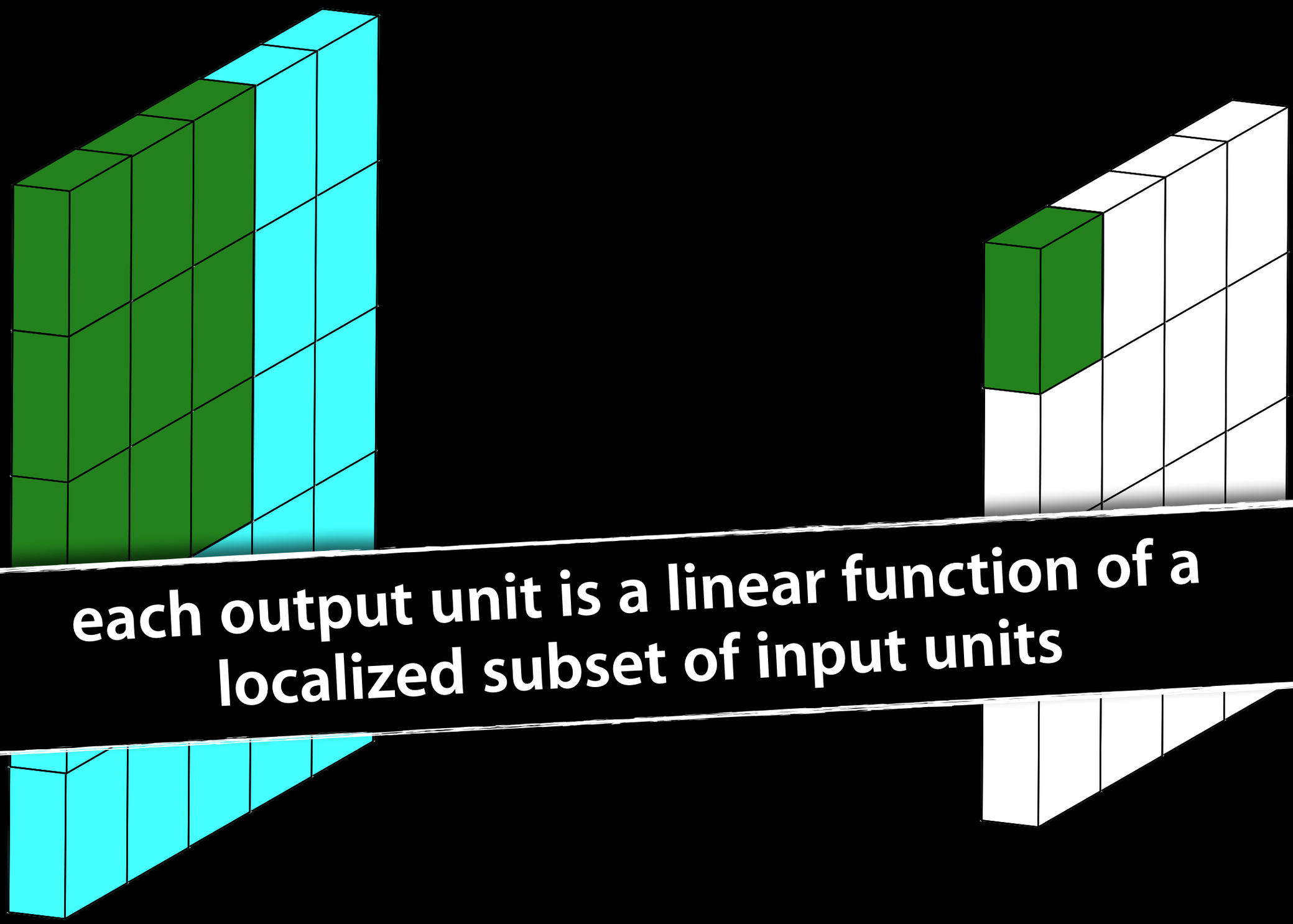




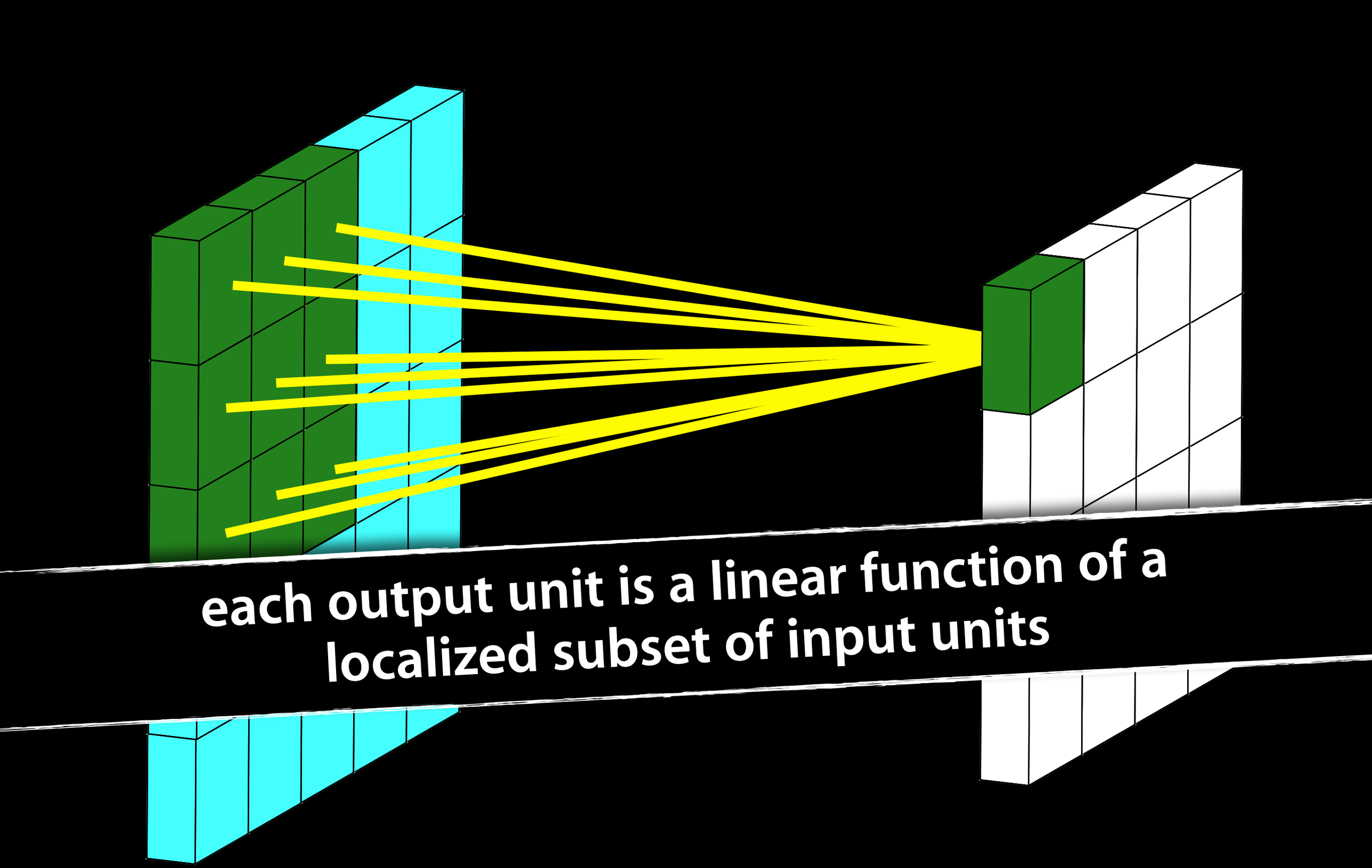




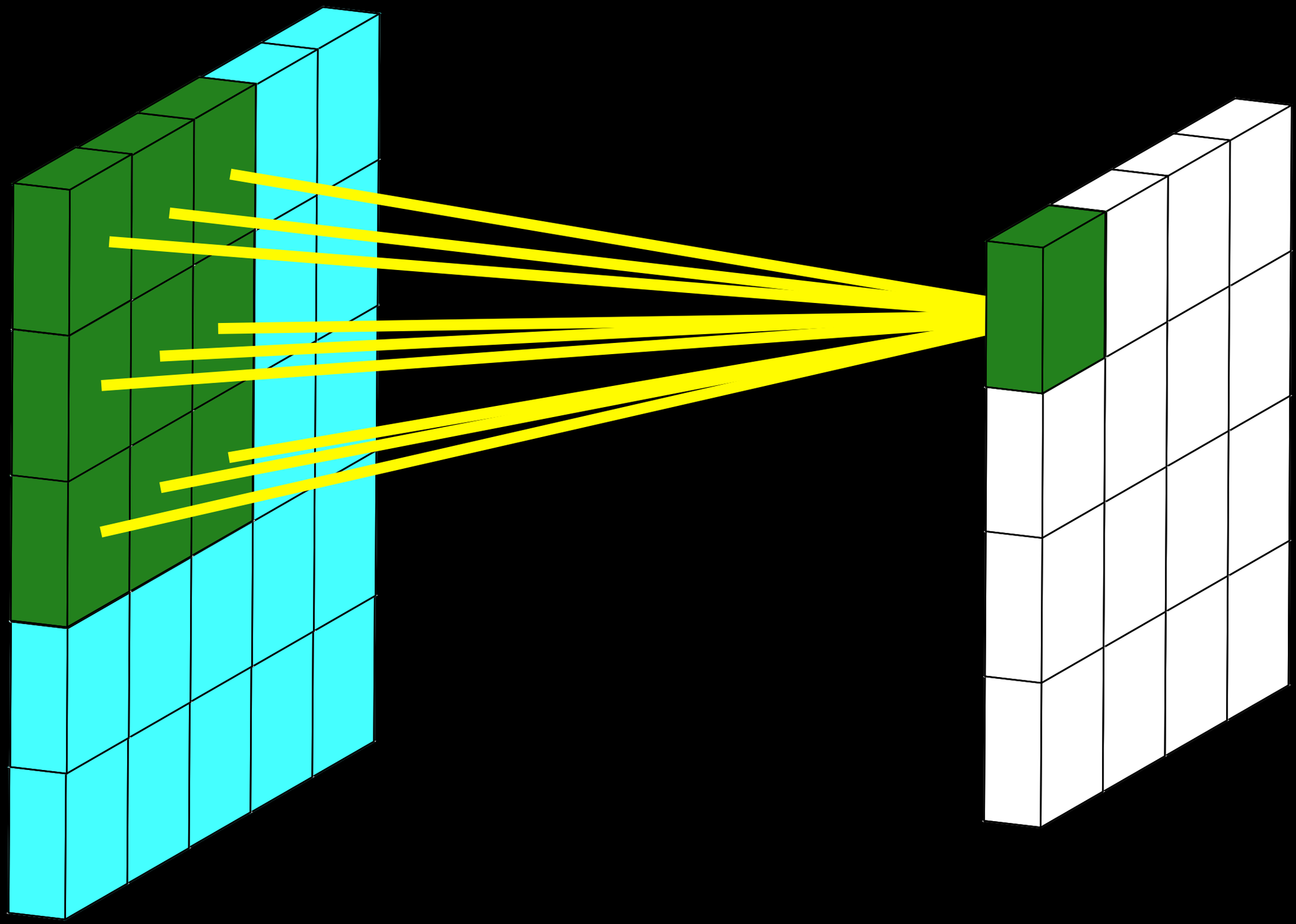


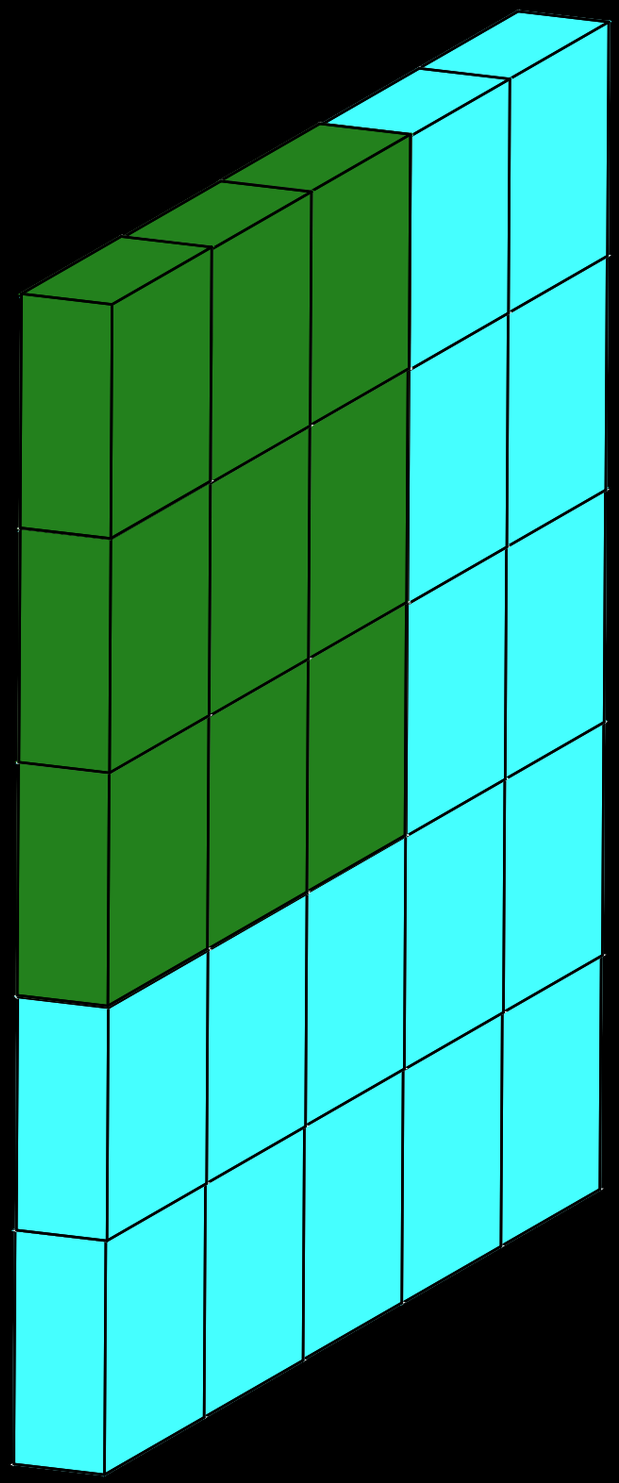


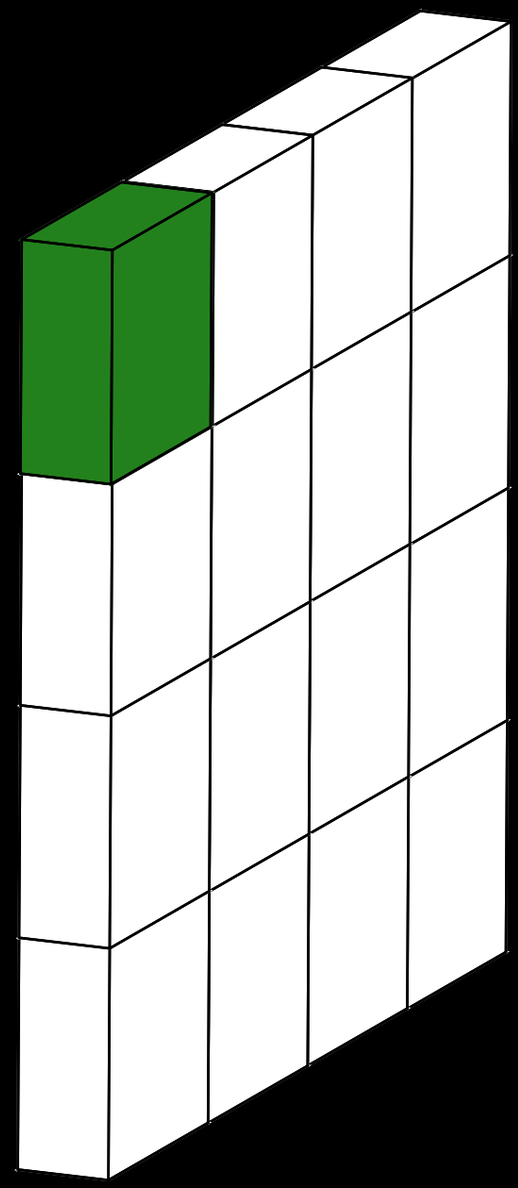
each output unit is a linear function of a localized subset of input units

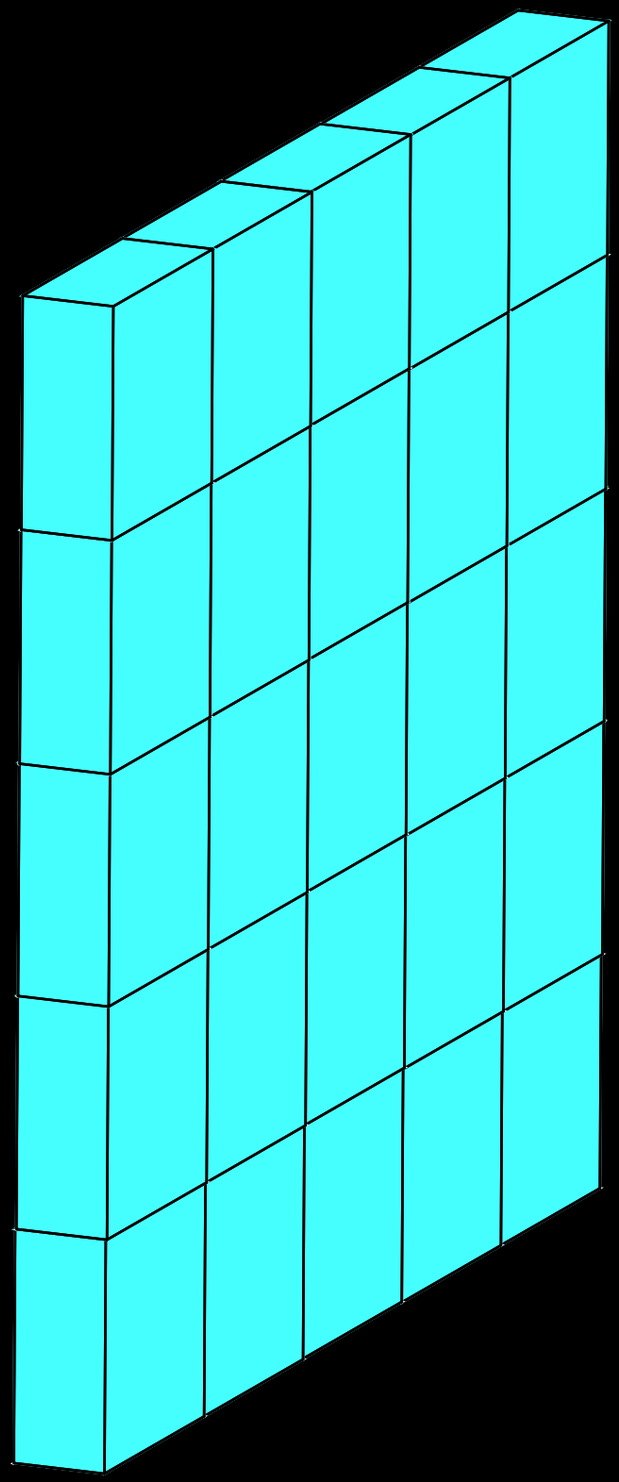


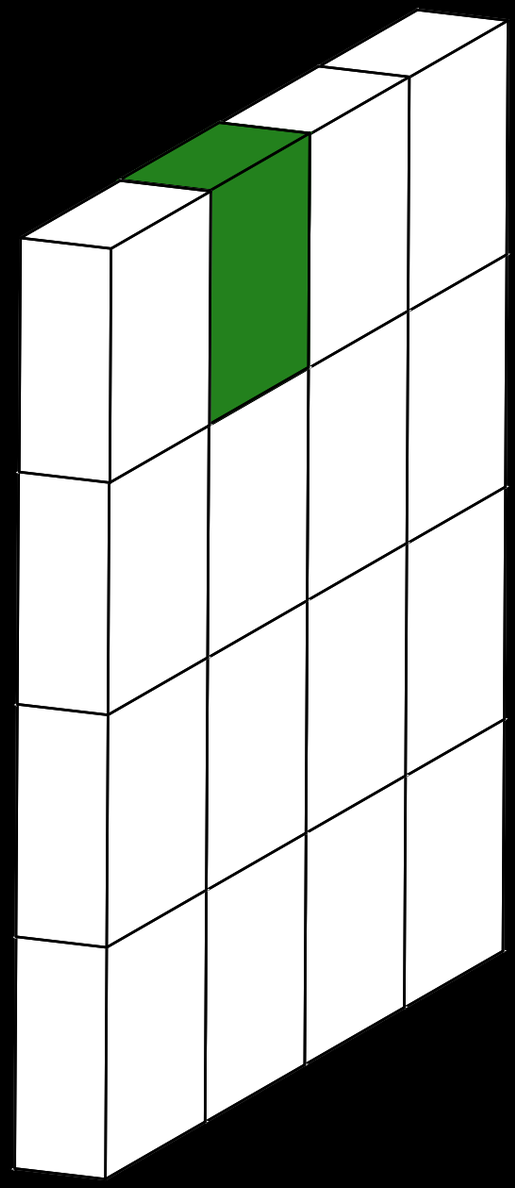
each output unit is a linear function of a localized subset of input units

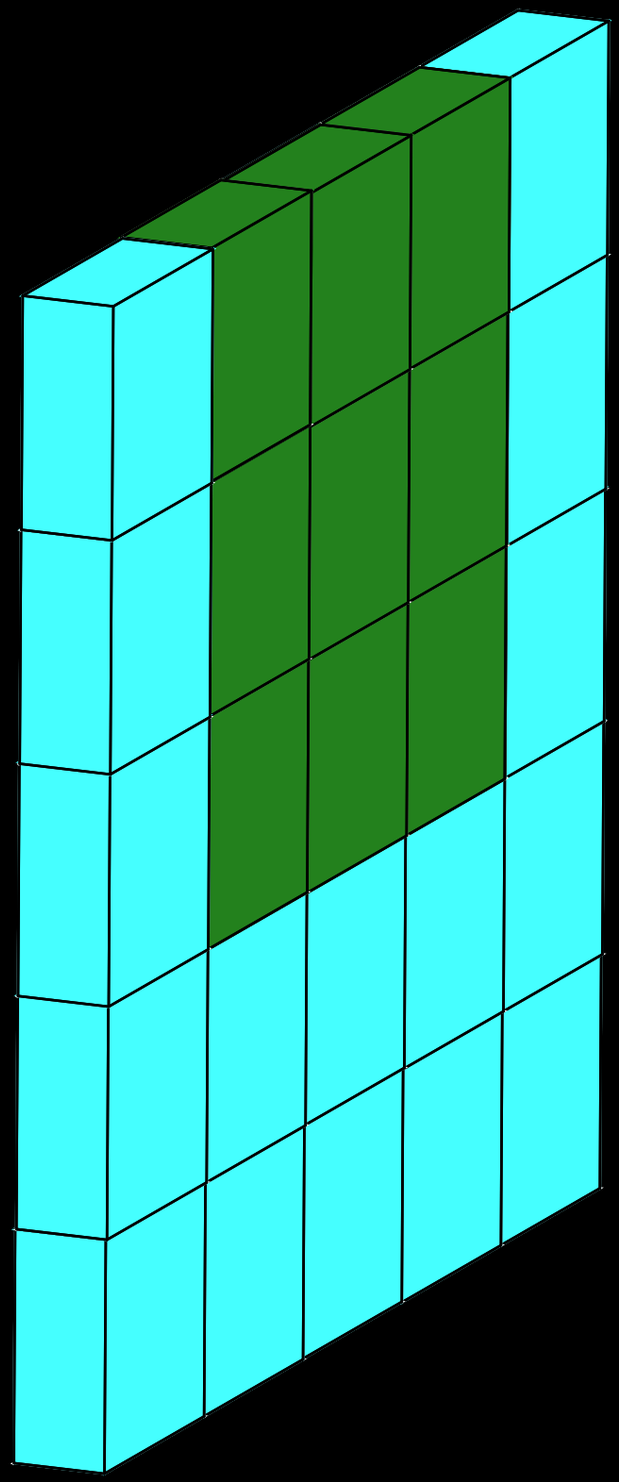


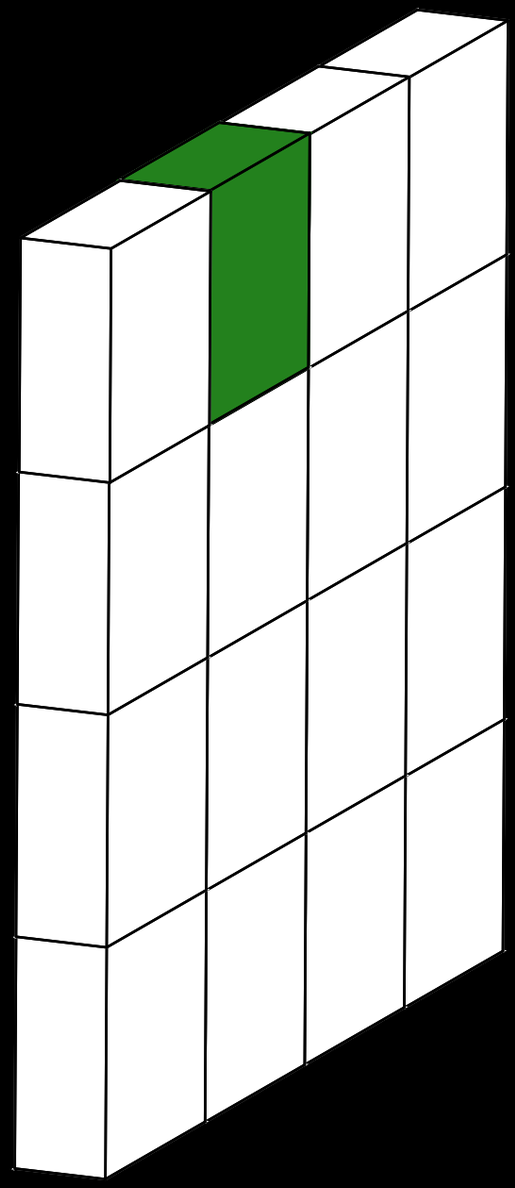


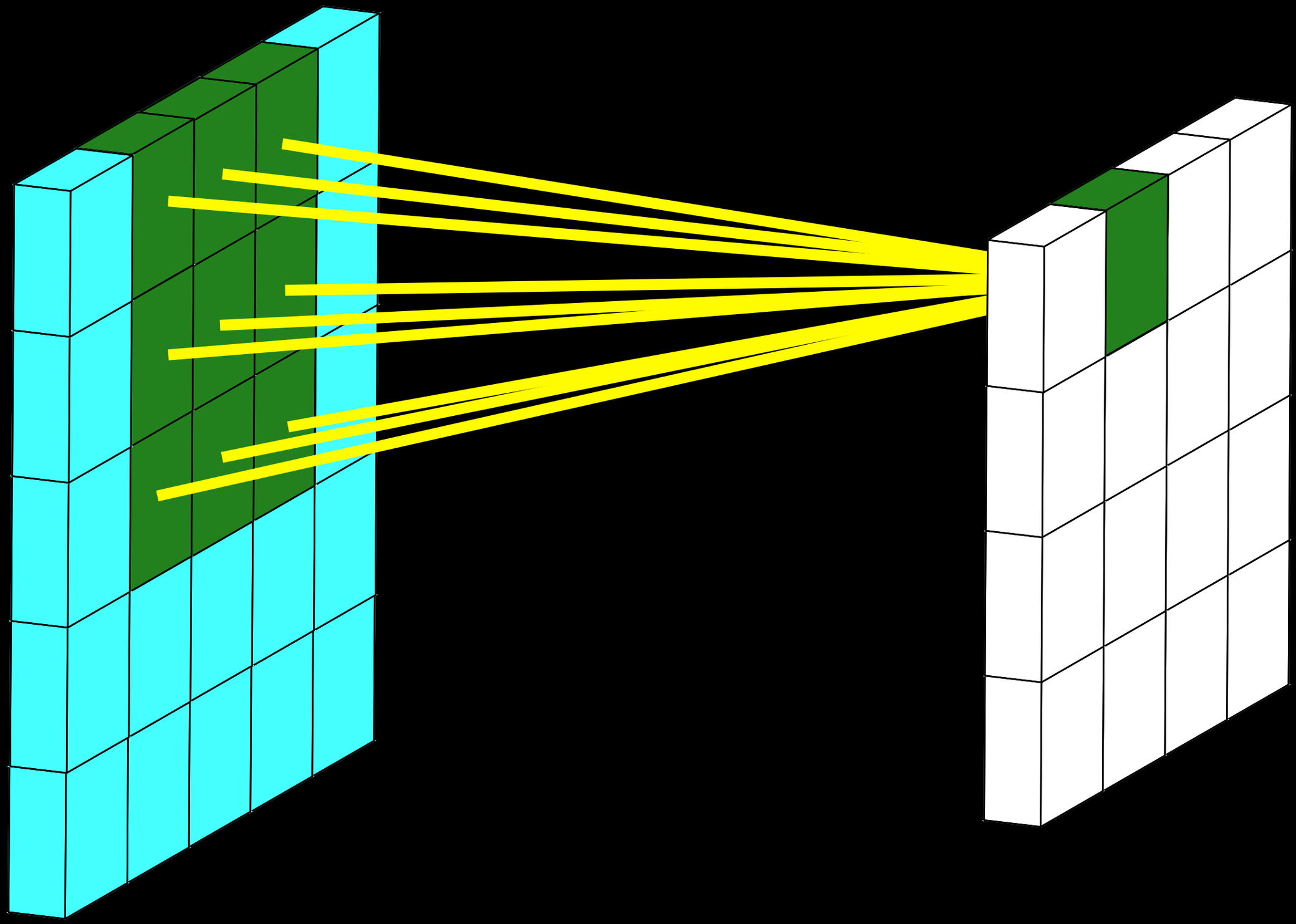


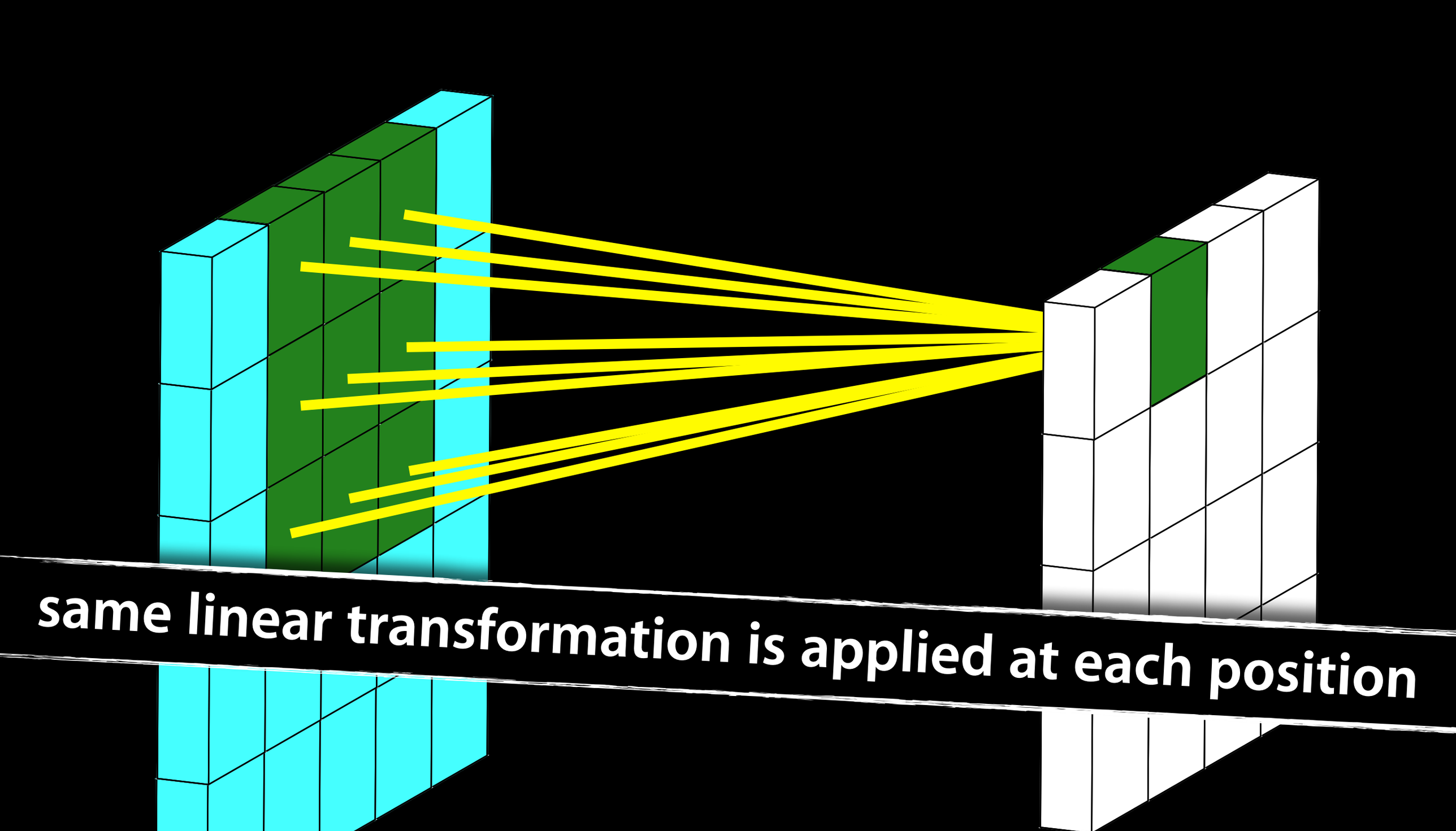


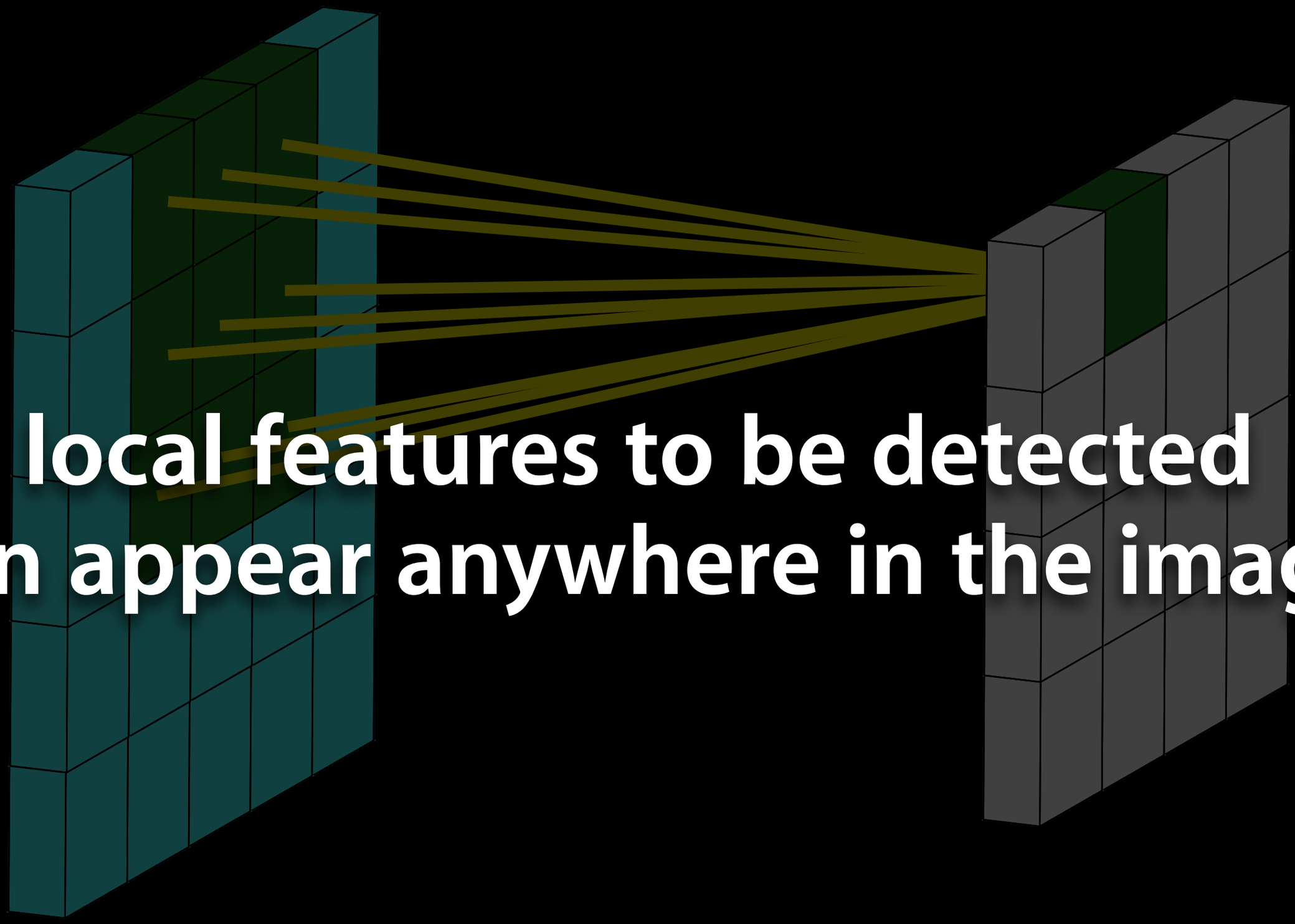




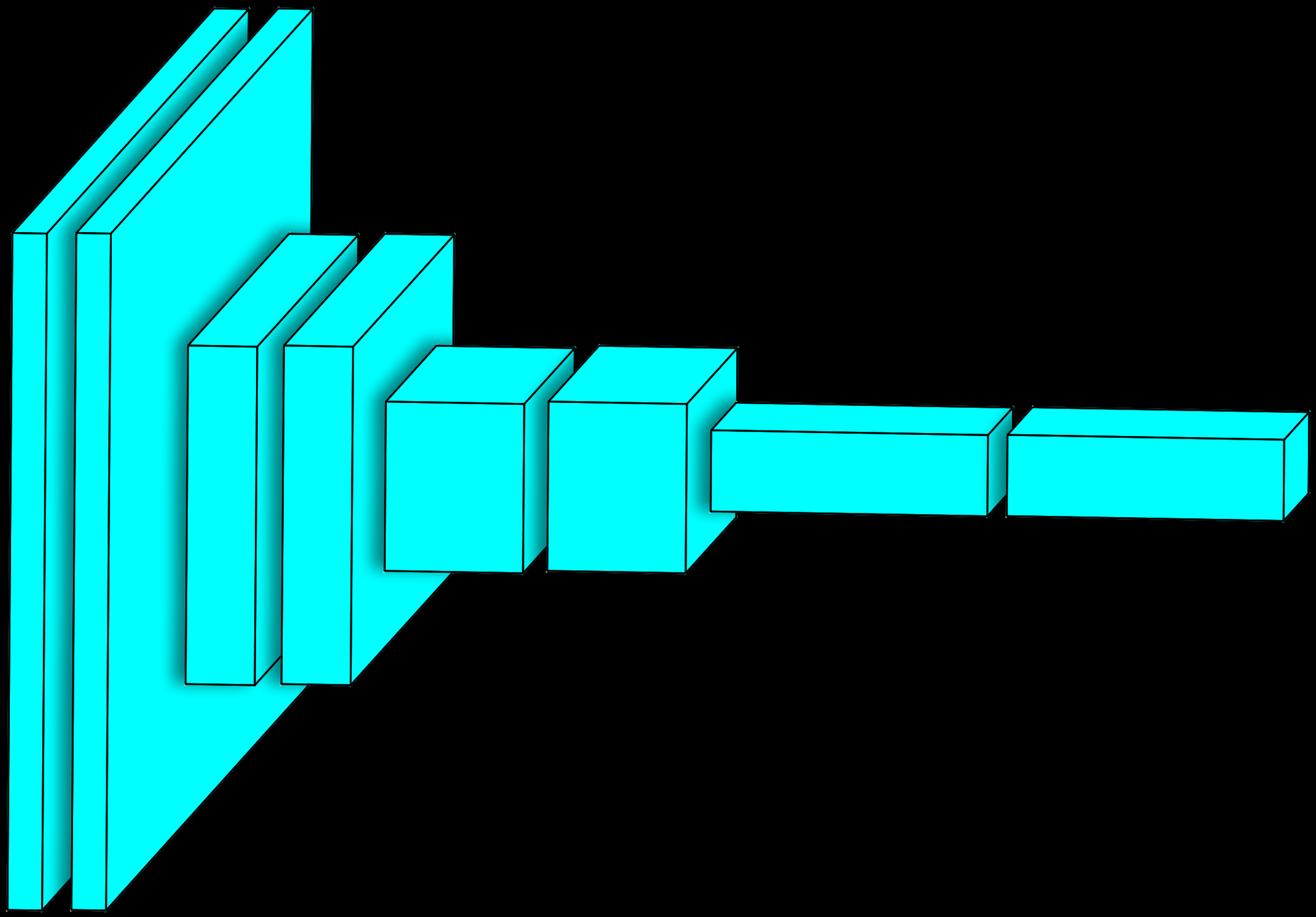





same linear transformation is applied at each position



local features to be detected can appear anywhere in the image



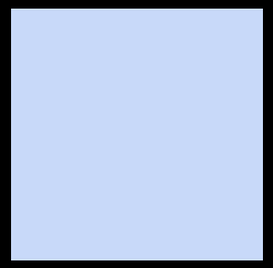
cat

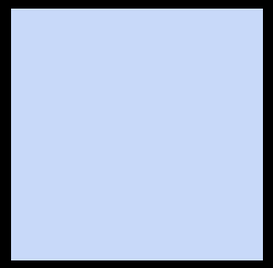
Building Blocks

Building Blocks

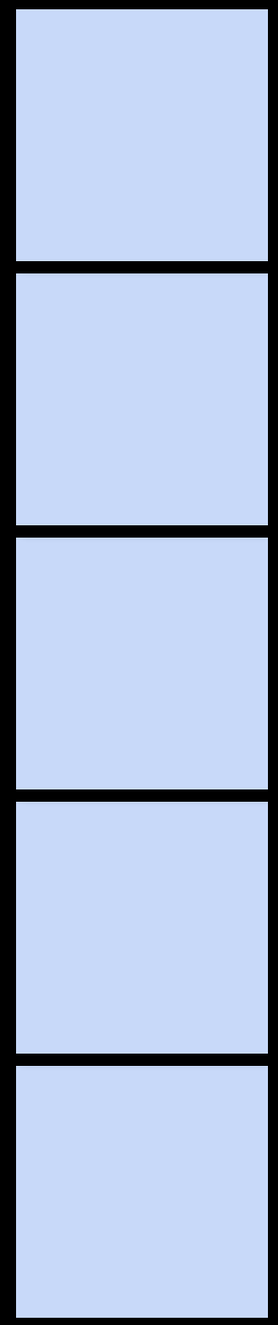
tensor

array of numbers arranged in a regular grid with a variable number of axes

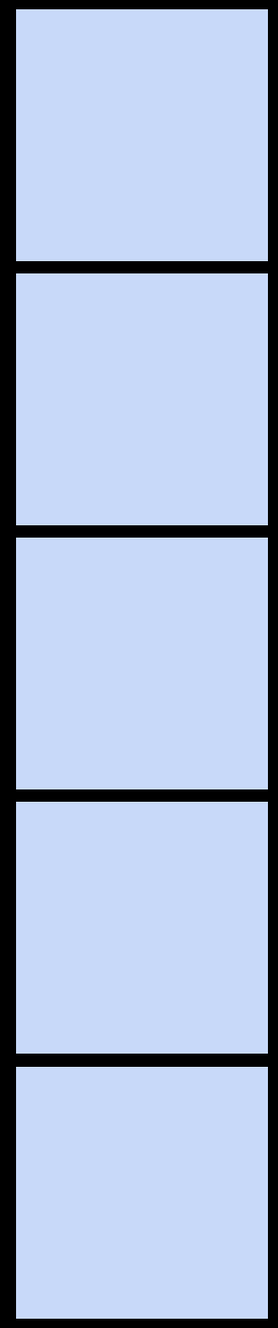




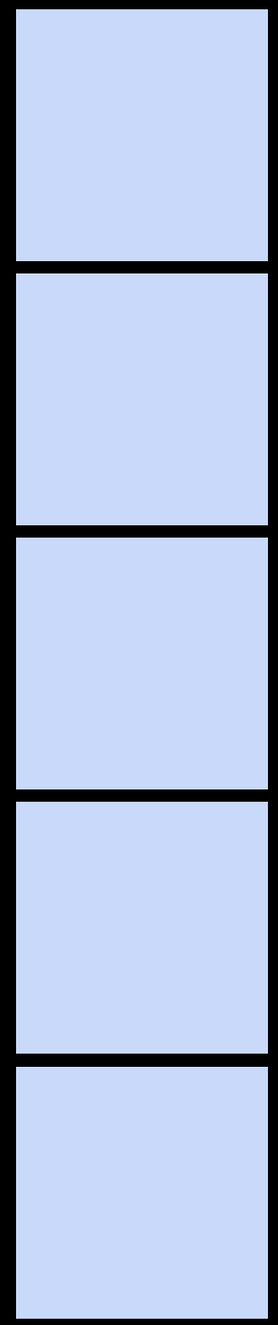
0D tensor



rows × 1

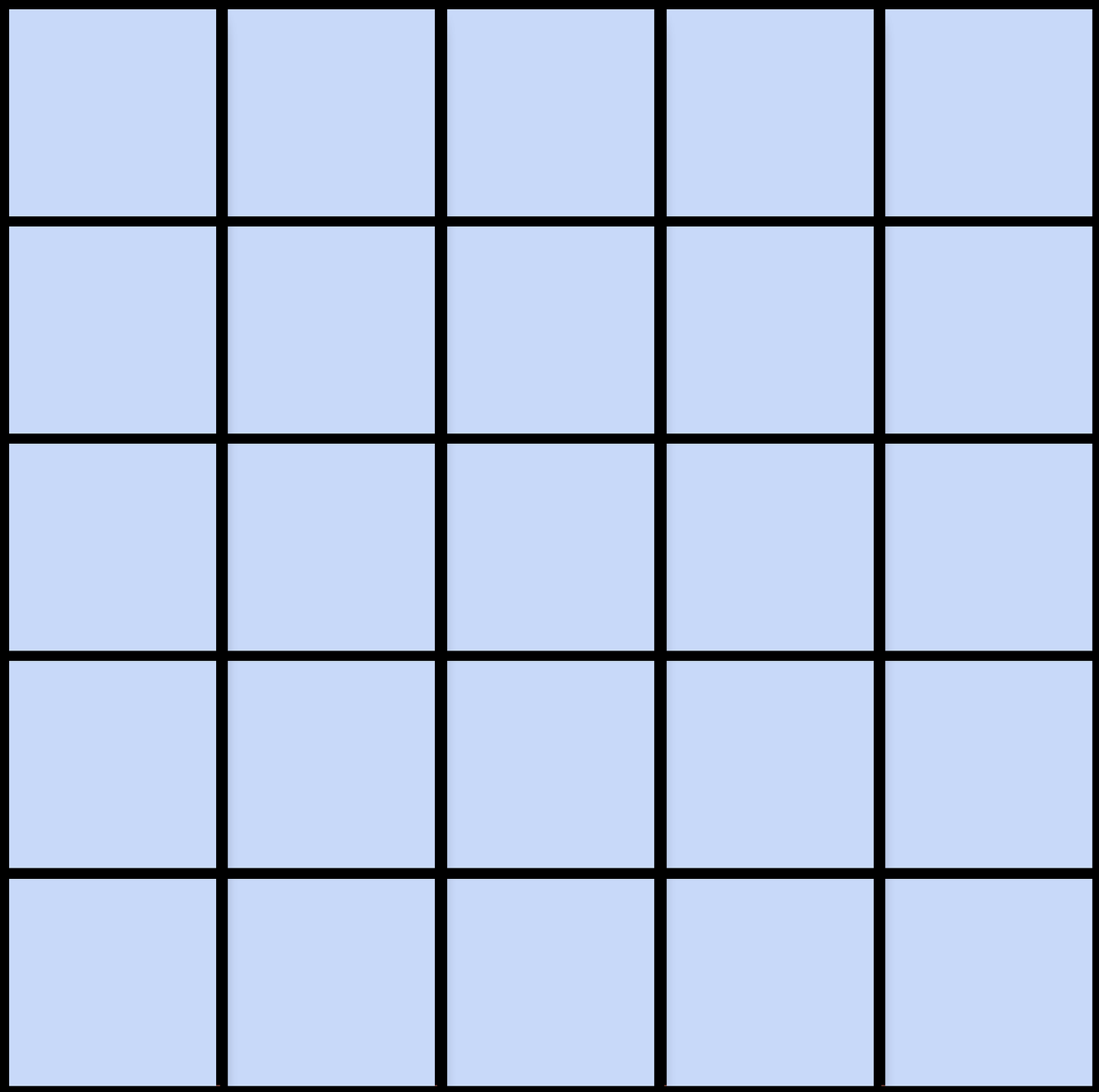


rows × 1

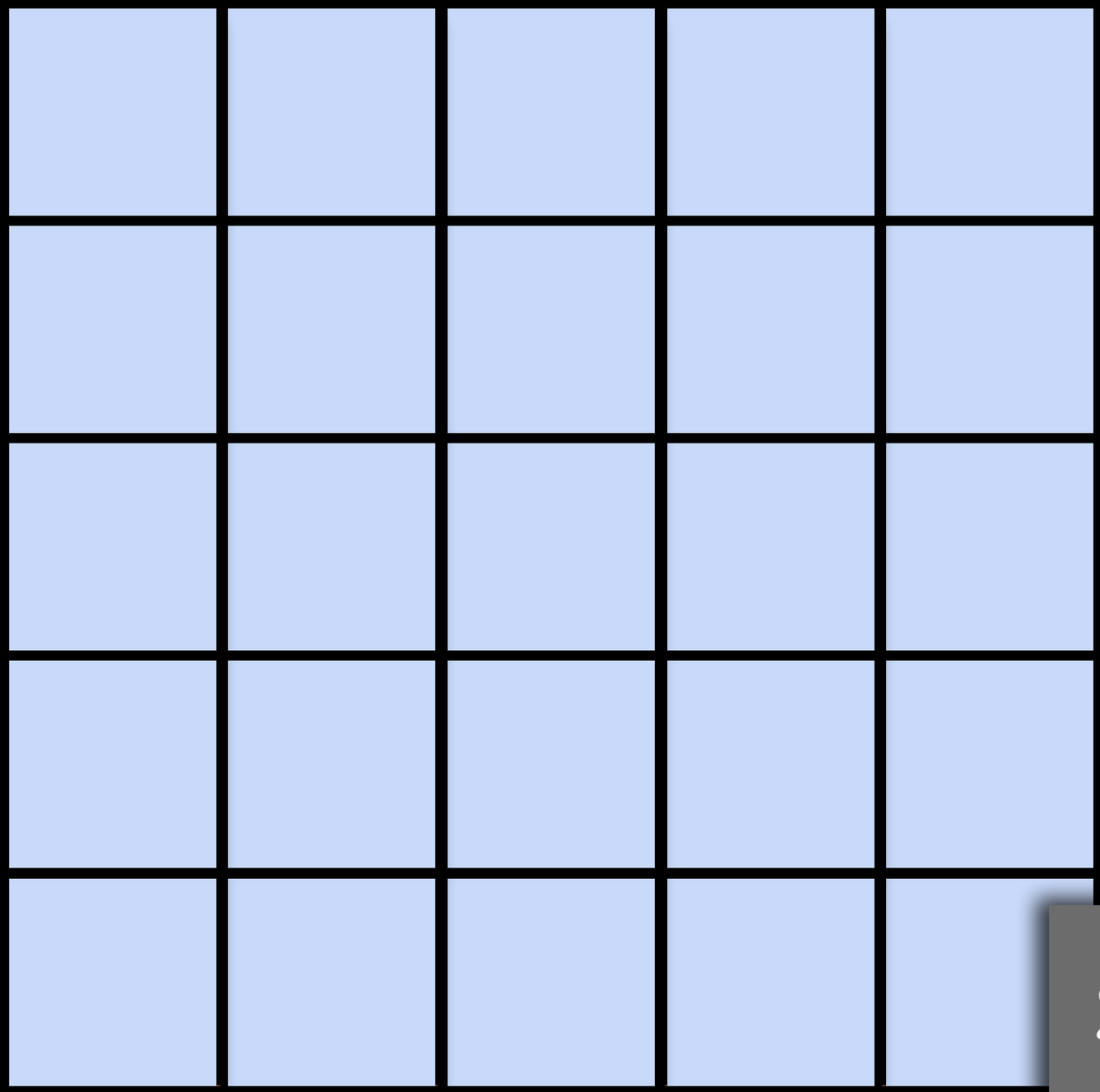


`1D tensor`

rows $\times$ 1

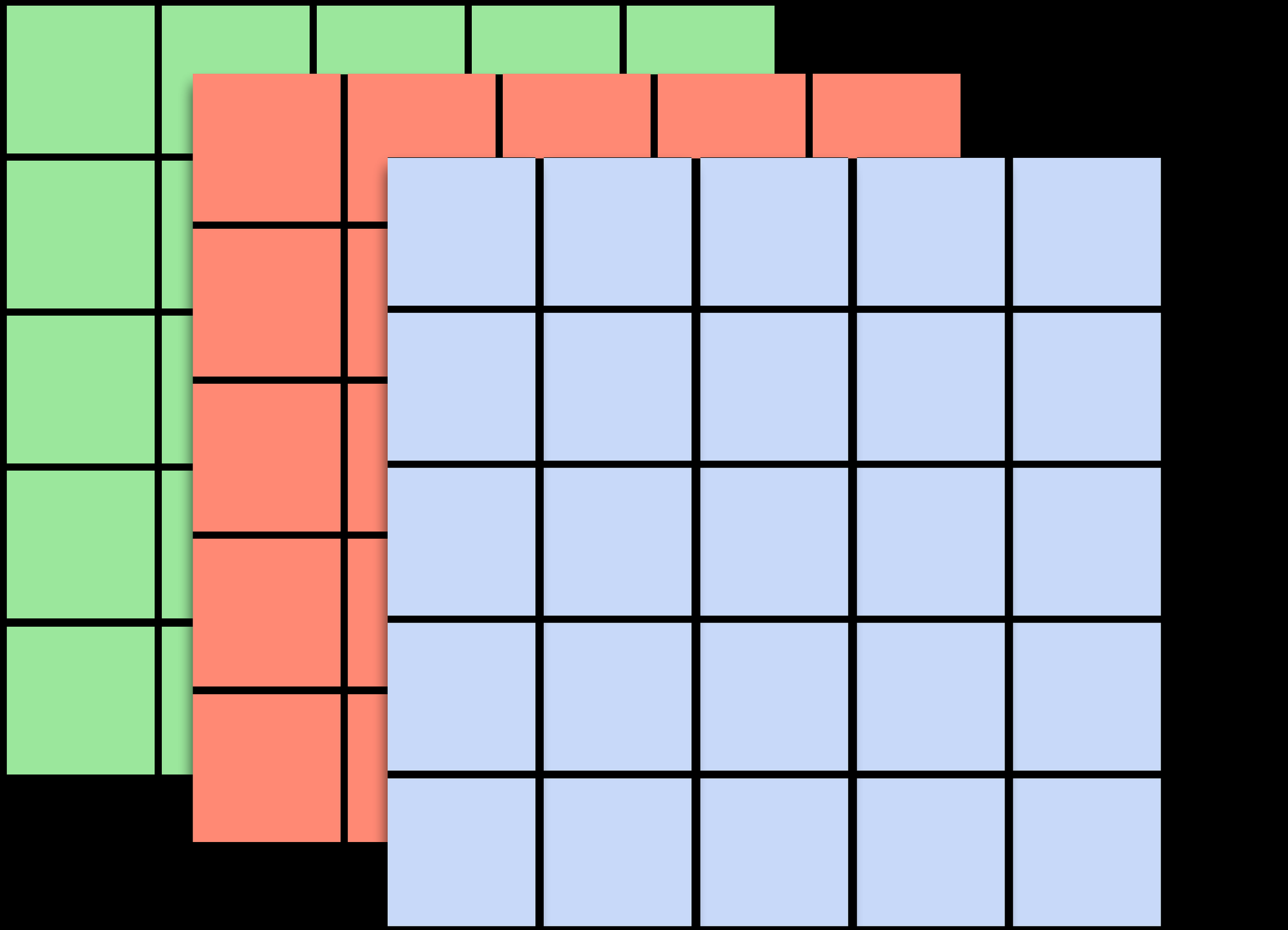


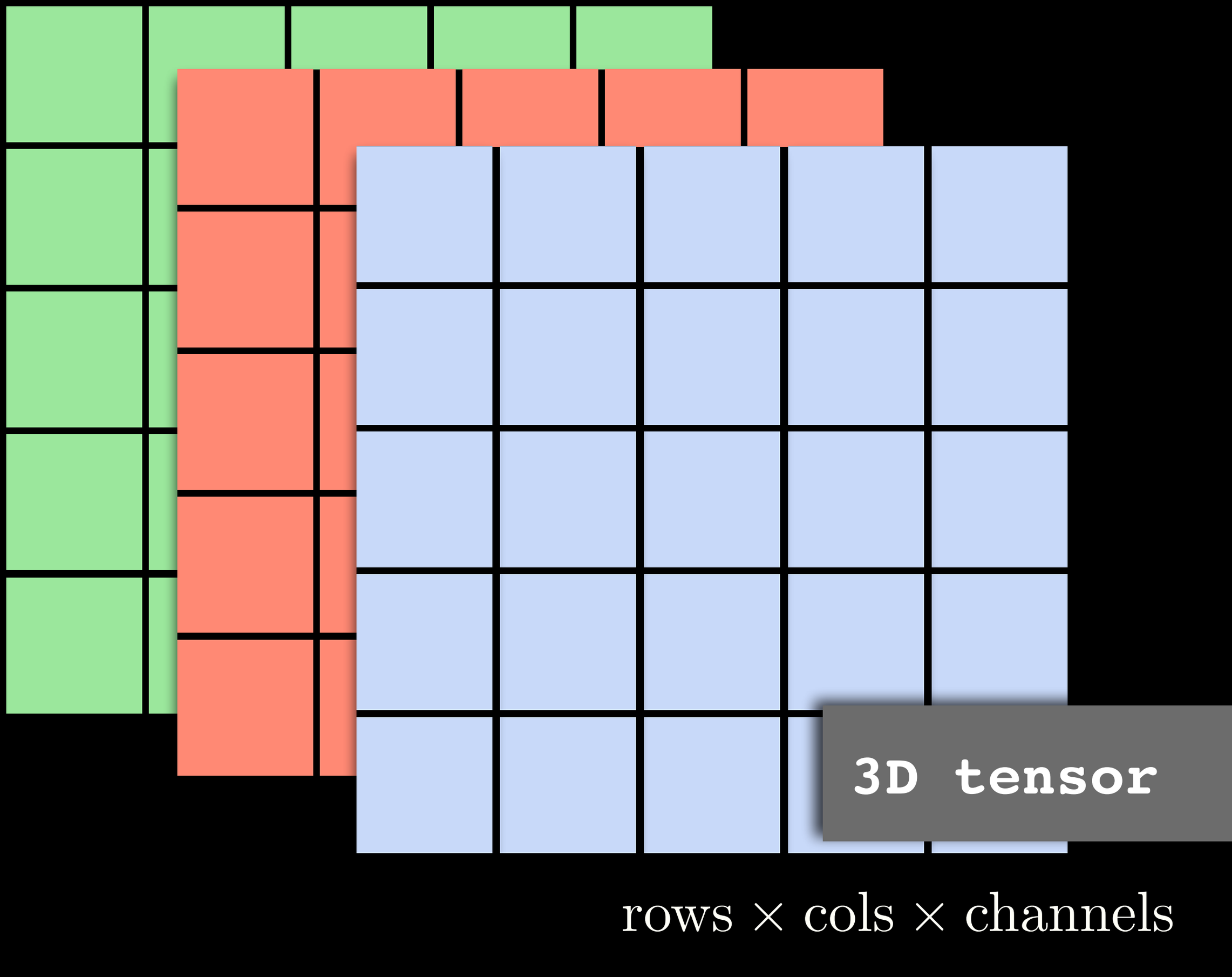
rows × cols



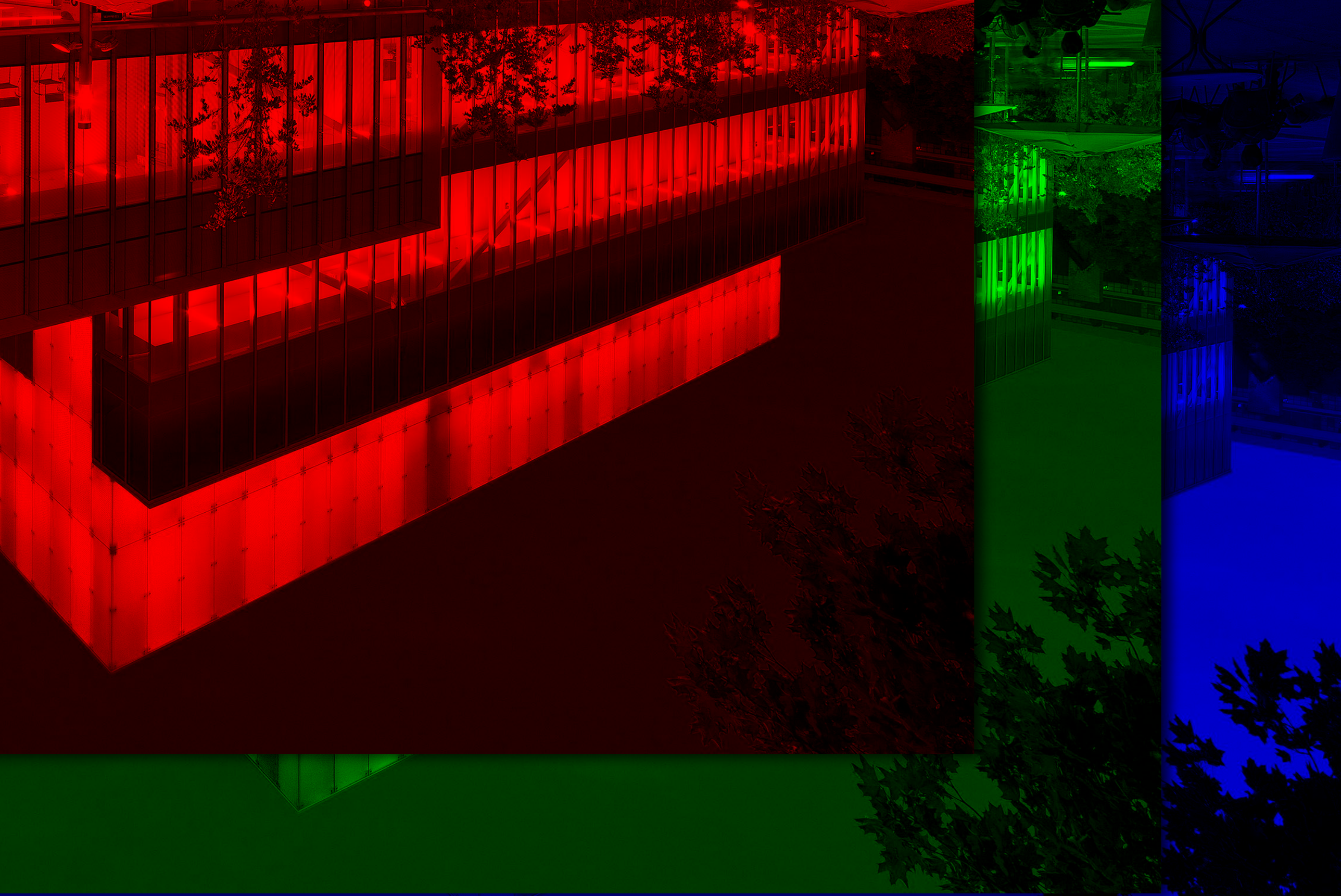
2D tensor

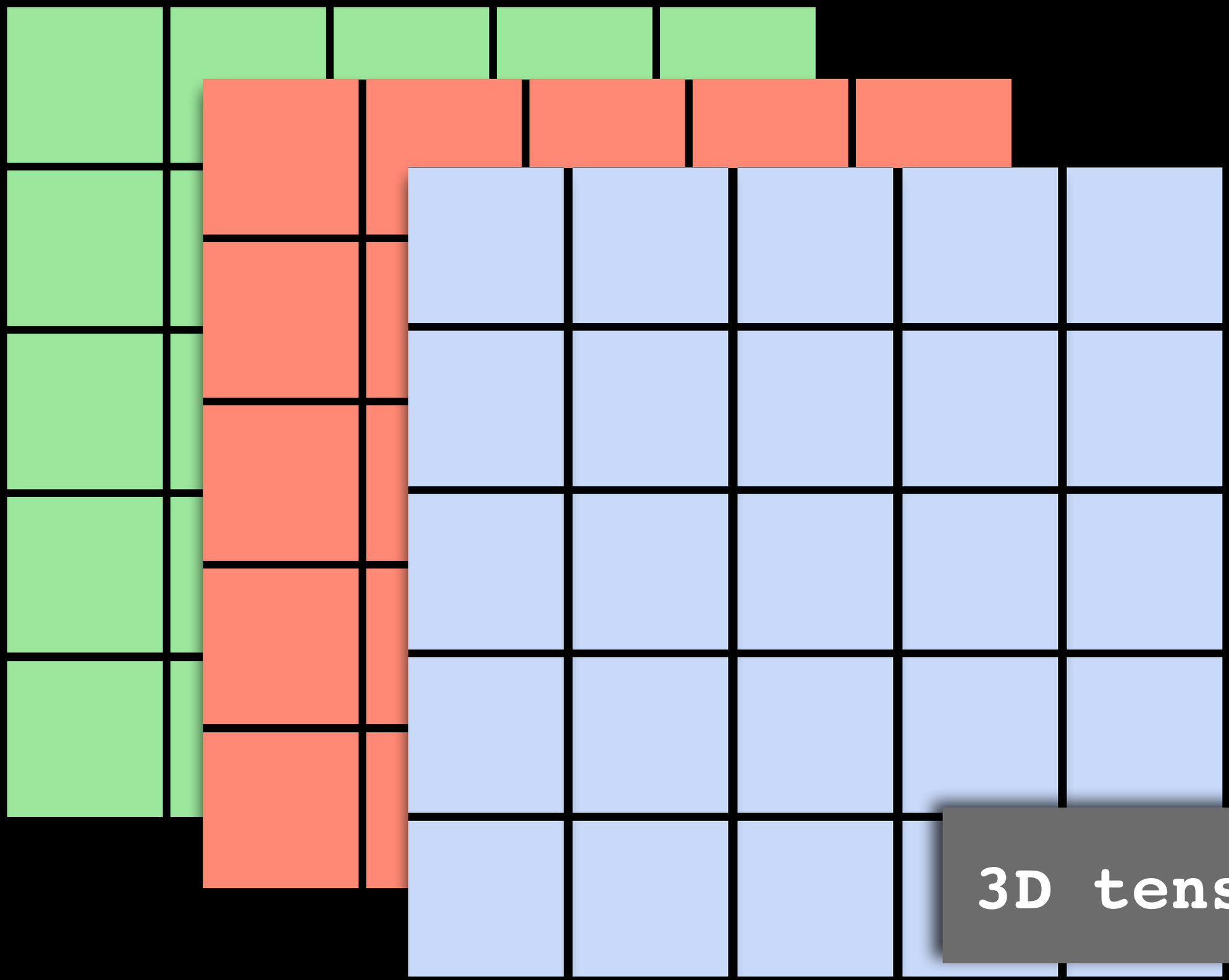
rows × cols

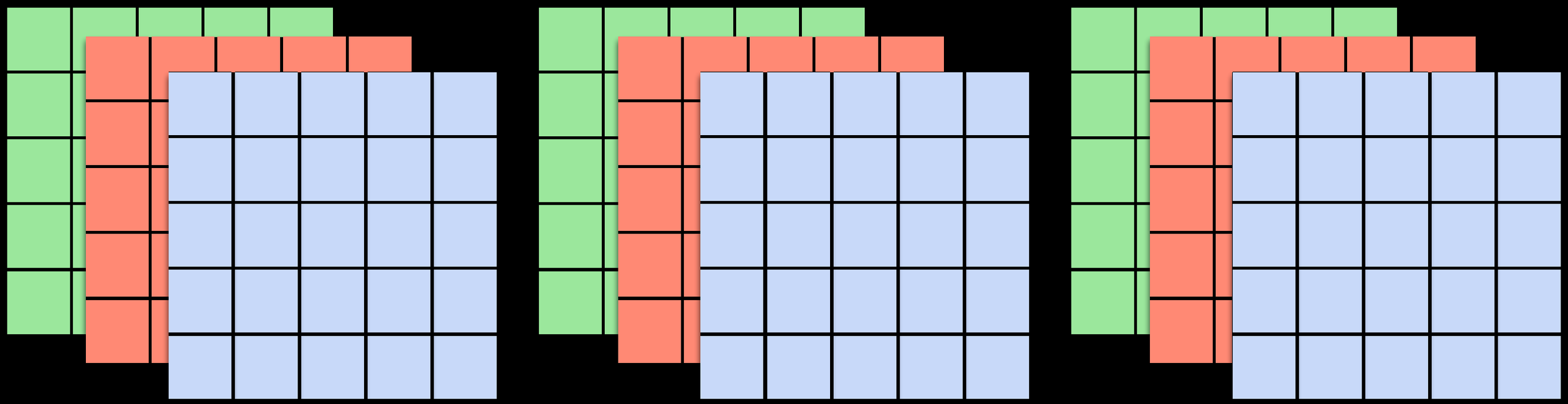

rows × cols × channels


3D tensor
rows × cols × channels

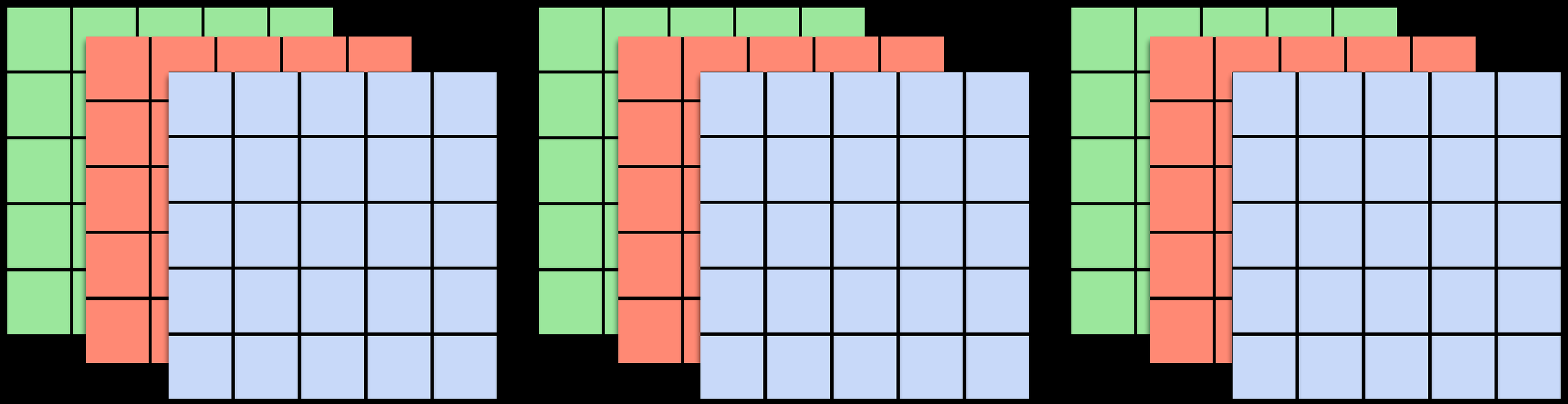

768 × 1024 × 3




3D tensor
rows × cols × channels



$$\text{rows} \times \text{cols} \times \text{channels} \times t$$



$\text{rows} \times \text{cols} \times \text{channels} \times t$

4D tensor

convolution

$$I[x, y] * F[x, y]$$

convolution

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

$*$

0	1	2
2	2	0
0	1	2

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

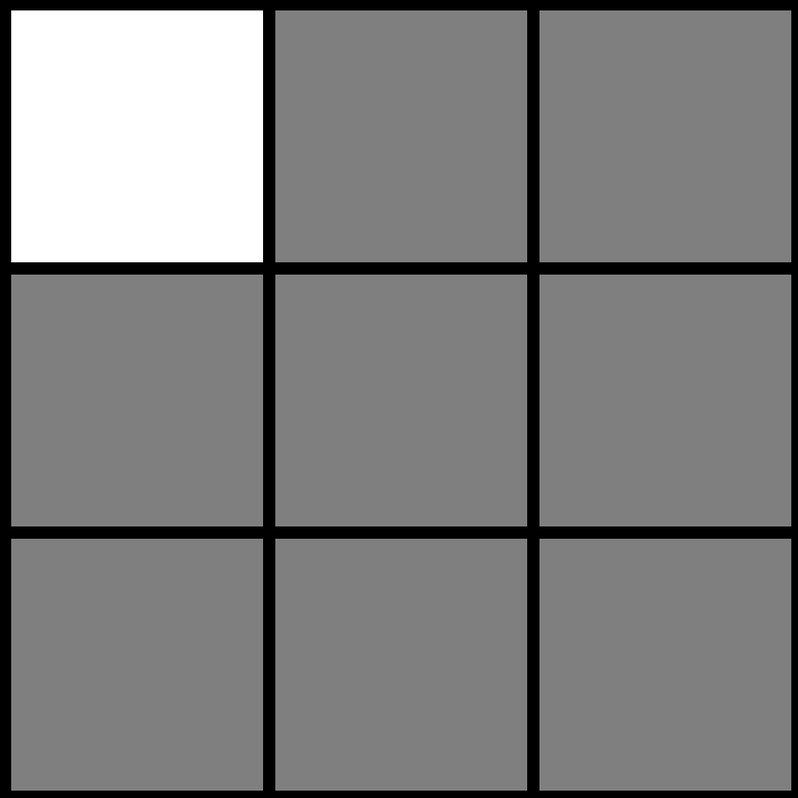
=

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

=

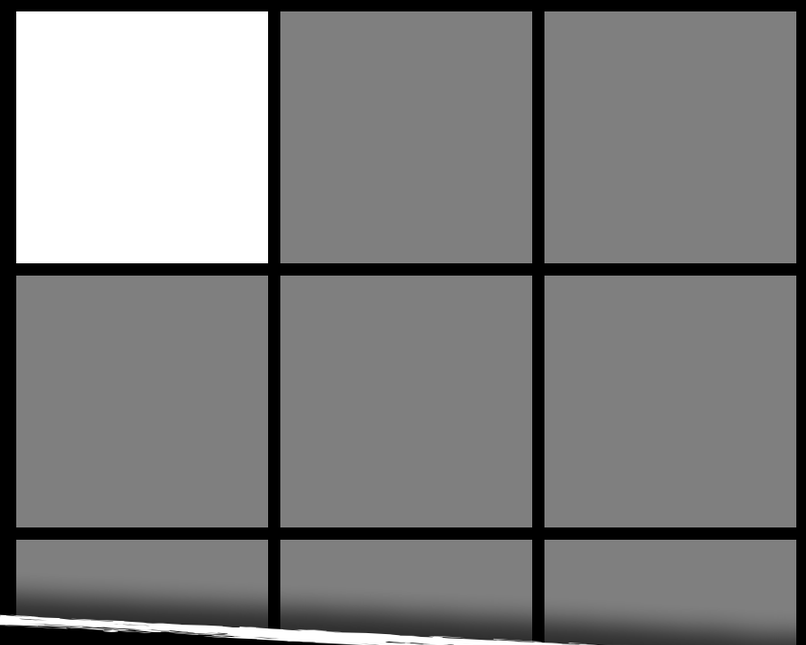
3 0	3 1	2 2	1	0
0 2	0 2	1 0	3	1
3 0	1 1	2 2	2	3
2	0	0	2	2
2	0	0	0	1

=



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

=



pointwise multiplication and sum

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

=

12		

pointwise multiplication and sum

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

=

12		

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

=

12	12	

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

=

12	12	

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

=

12	12	17

3	3	2	1	0
0_0	0_1	1_2	3	1
3_2	1_2	2_0	2	3
2_0	0_1	0_2	2	2
2	0	0	0	1

=

12	12	17	

3	3	2	1	0
0_0	0_1	1_2	3	1
3_2	1_2	2_0	2	3
2_0	0_1	0_2	2	2
2	0	0	0	1

=

12	12	17
10		

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

=

12	12	17
10		

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

=

12	12	17
10	17	

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

=

12	12	17
10	17	

3	3	2	1	0
0	0	1_{0}	3_{1}	1_{2}
3	1	2_{2}	2_{2}	3_{0}
2	0	0_{0}	2_{1}	2_{2}
2	0	0	0	1

=

12	12	17
10	17	19

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

=

12	12	17
10	17	19

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

=

12	12	17
10	17	19
9		

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

=

12	12	17
10	17	19
9		

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

=

12	12	17
10	17	19
9	6	

3	3	2	1	0
0	0	1	3	1
3	1	2_0	2_1	3_2
2	0	0_2	2_2	2_0
2	0	0_0	0_1	1_2

=

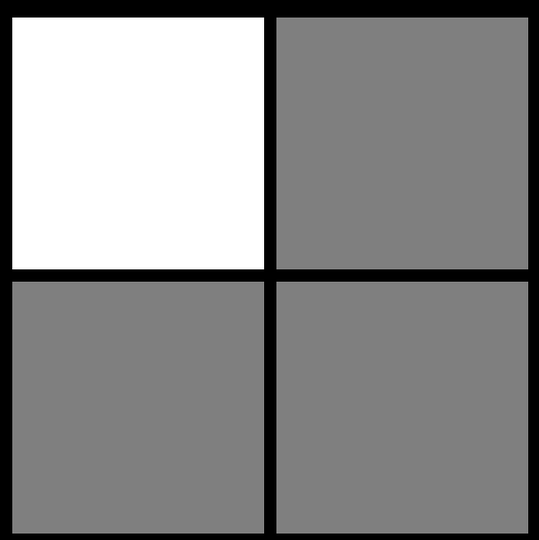
12	12	17
10	17	19
9	6	

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

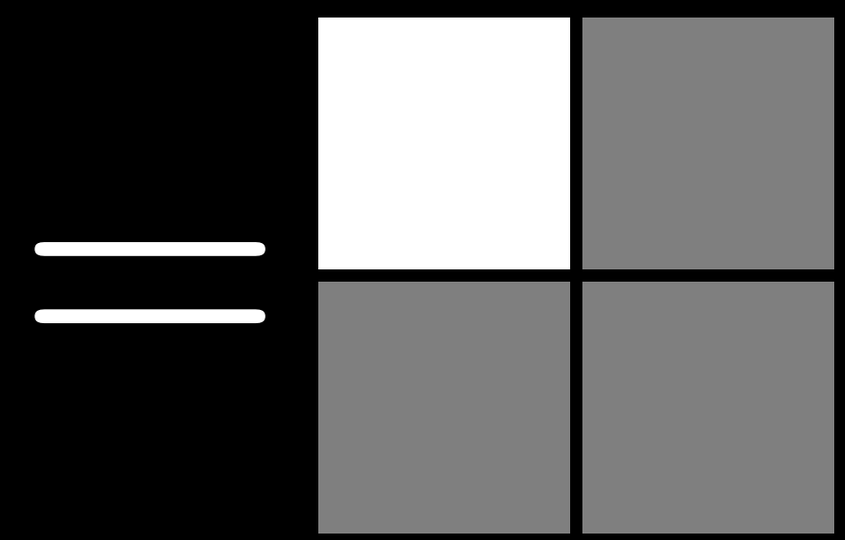
=

12	12	17
10	17	19
9	6	14

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

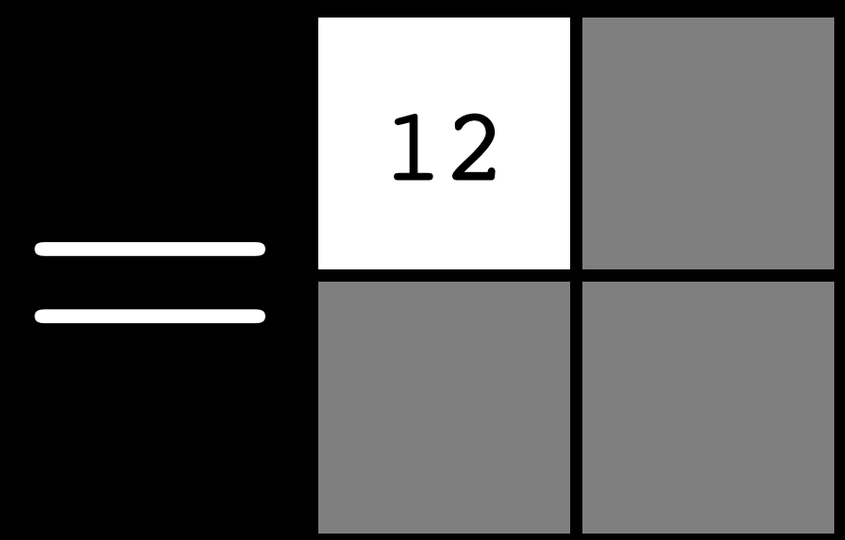


3 0	3 1	2 2	1	0
0 2	0 2	1 0	3	1
3 0	1 1	2 2	2	3
2	0	0	2	2
2	0	0	0	1



stride two convolution

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



stride two convolution

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

=

12	

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

=

12	17

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

=

12	17

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

=

12	17
9	

3	3	2	1	0
0	0	1	3	1
3	1	2_{0}	2_{1}	3_{2}
2	0	0_{2}	2_{2}	2_{0}
2	0	0_{0}	0_{1}	1_{2}

=

12	17
9	

3	3	2	1	0
0	0	1	3	1
3	1	2_{0}	2_{1}	3_{2}
2	0	0_{2}	2_{2}	2_{0}
2	0	0_{0}	0_{1}	1_{2}

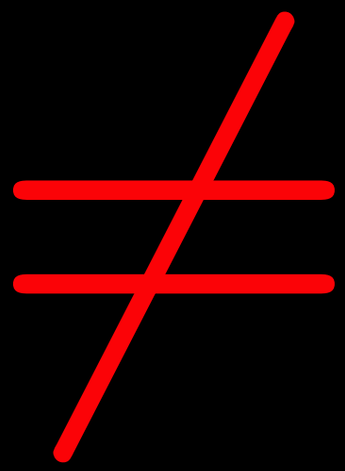
=

12	17
9	14

convolution

$$H[x,y] = \sum_{u=-K}^{K} \sum_{v=-K}^{K} I[x+u, y+v]\, F[u,v]$$

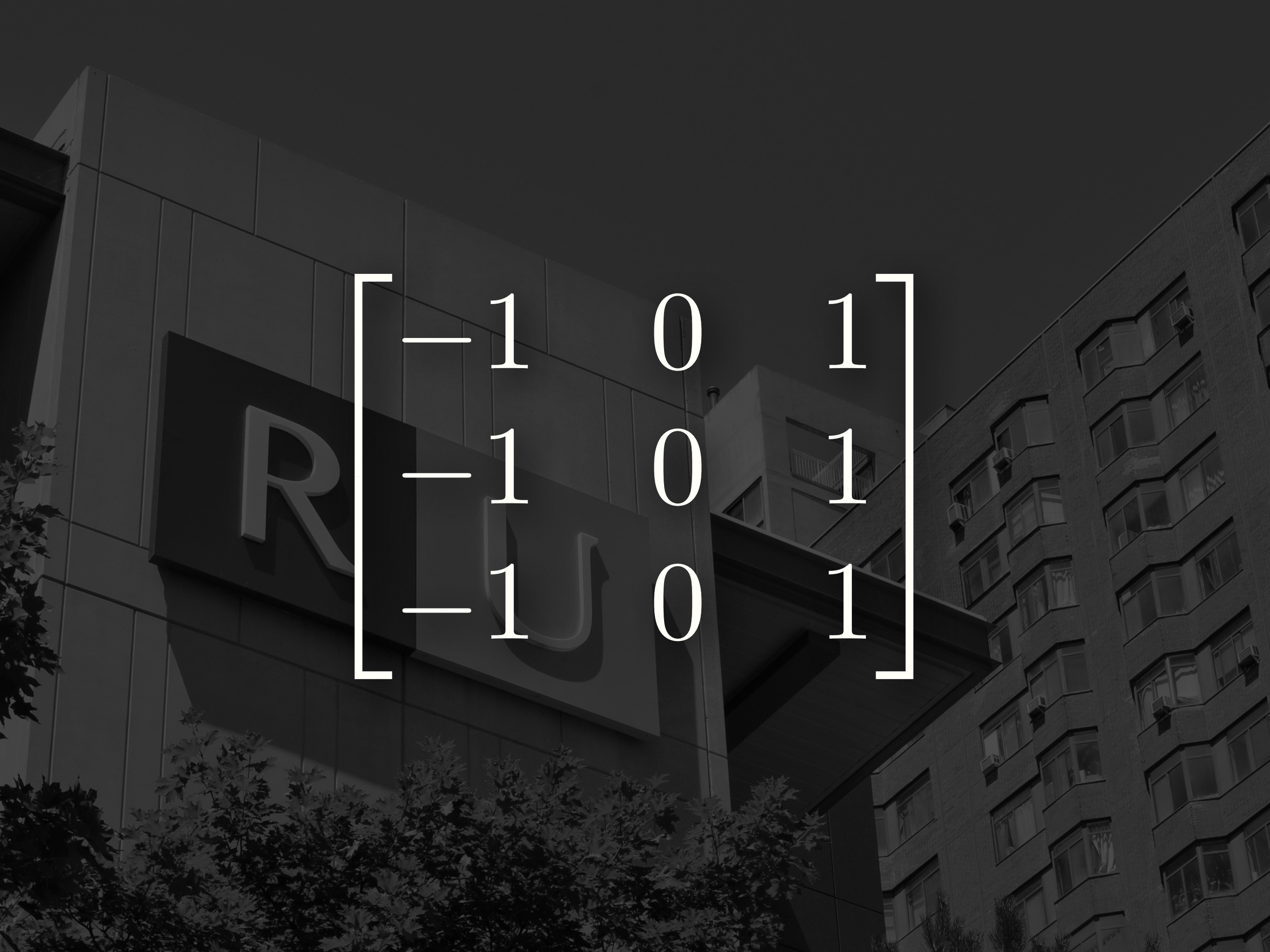
convolution

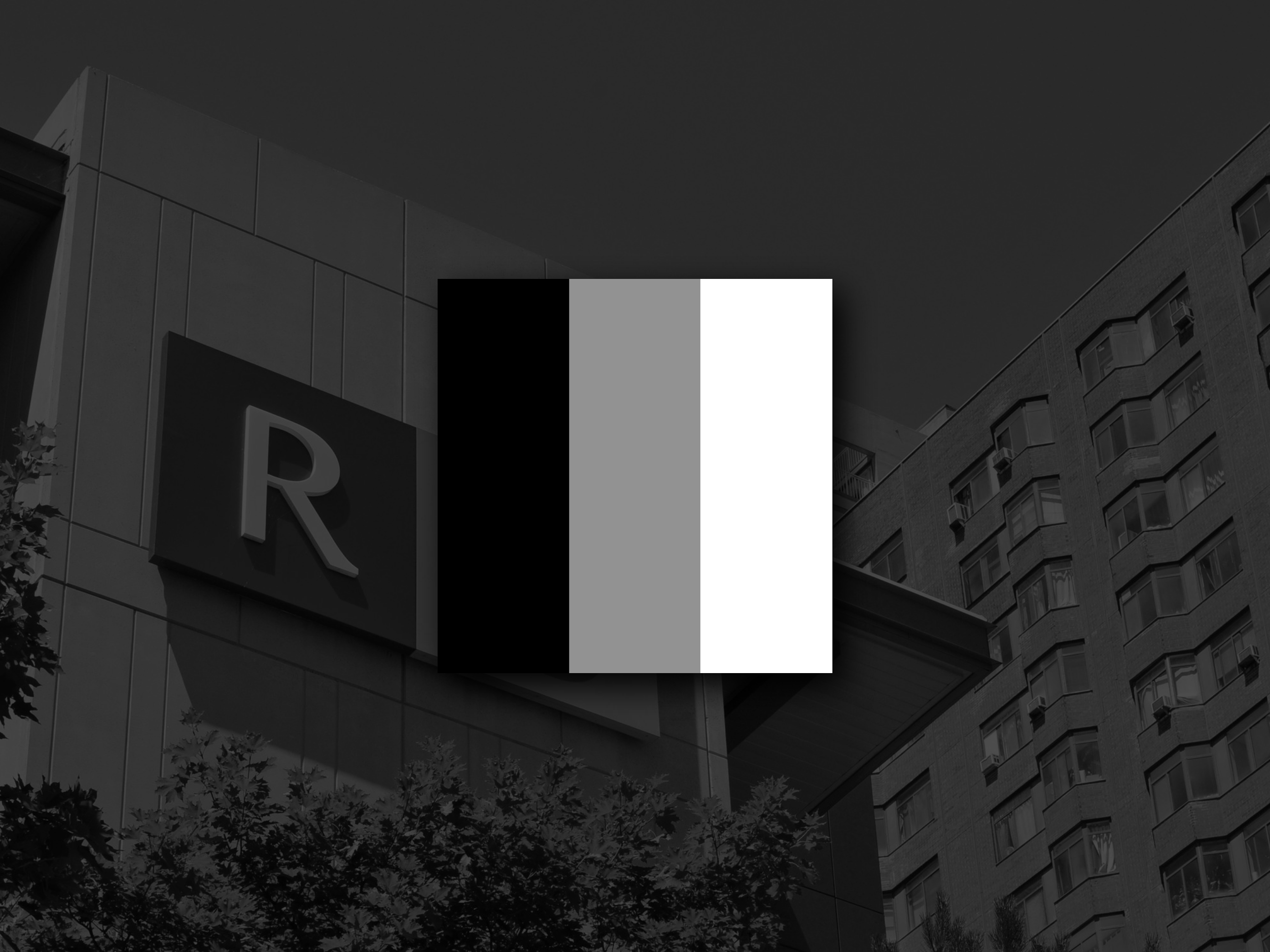


correlation

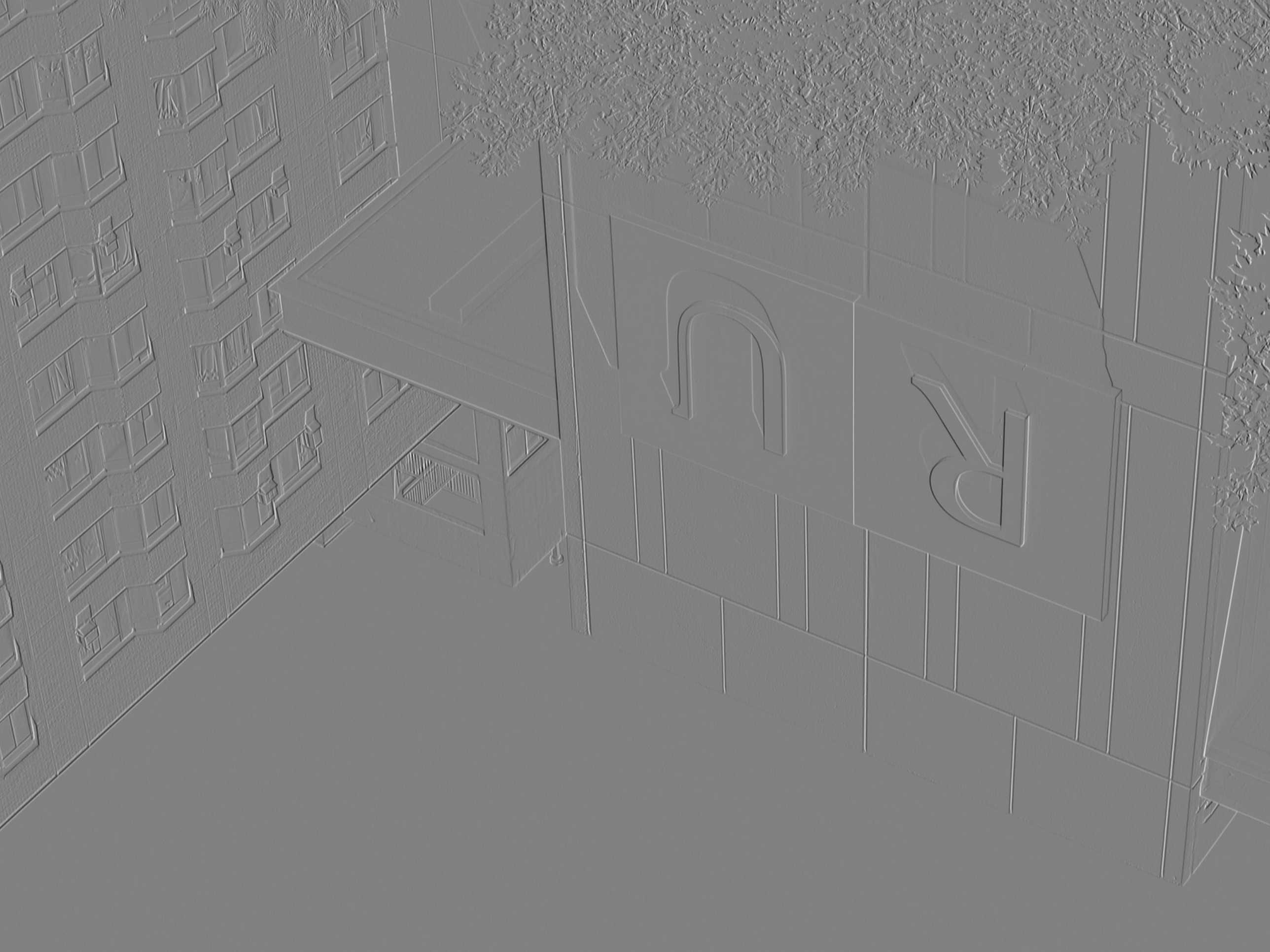

R
U

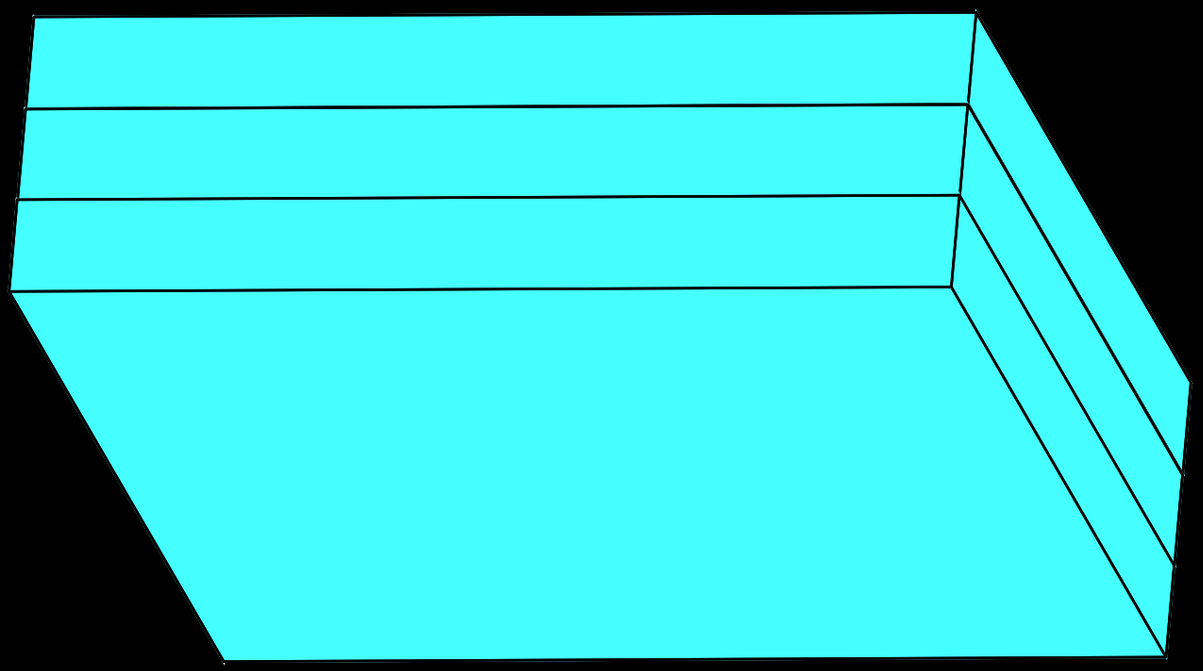

R
U

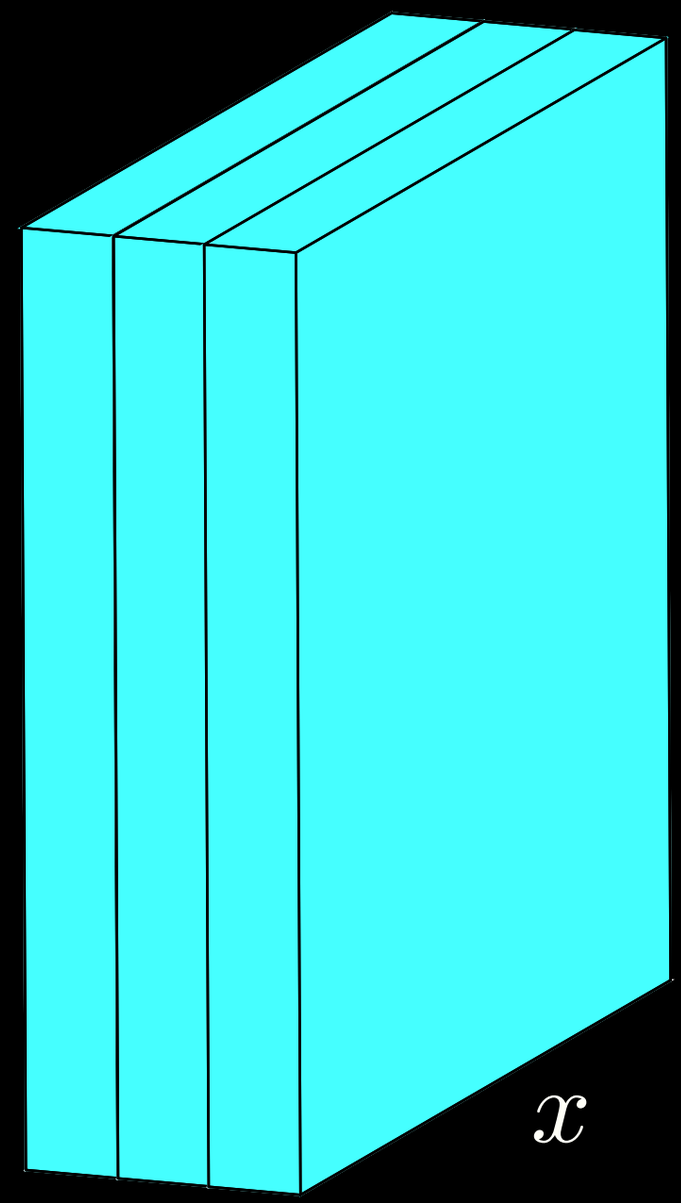

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

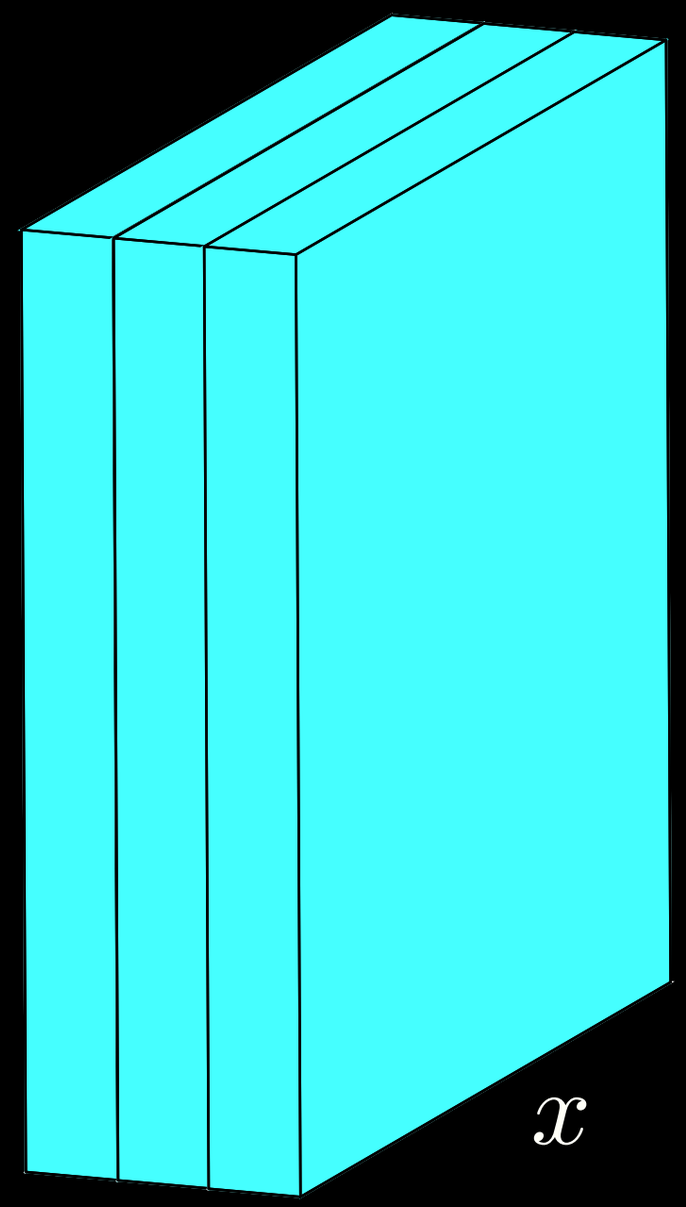


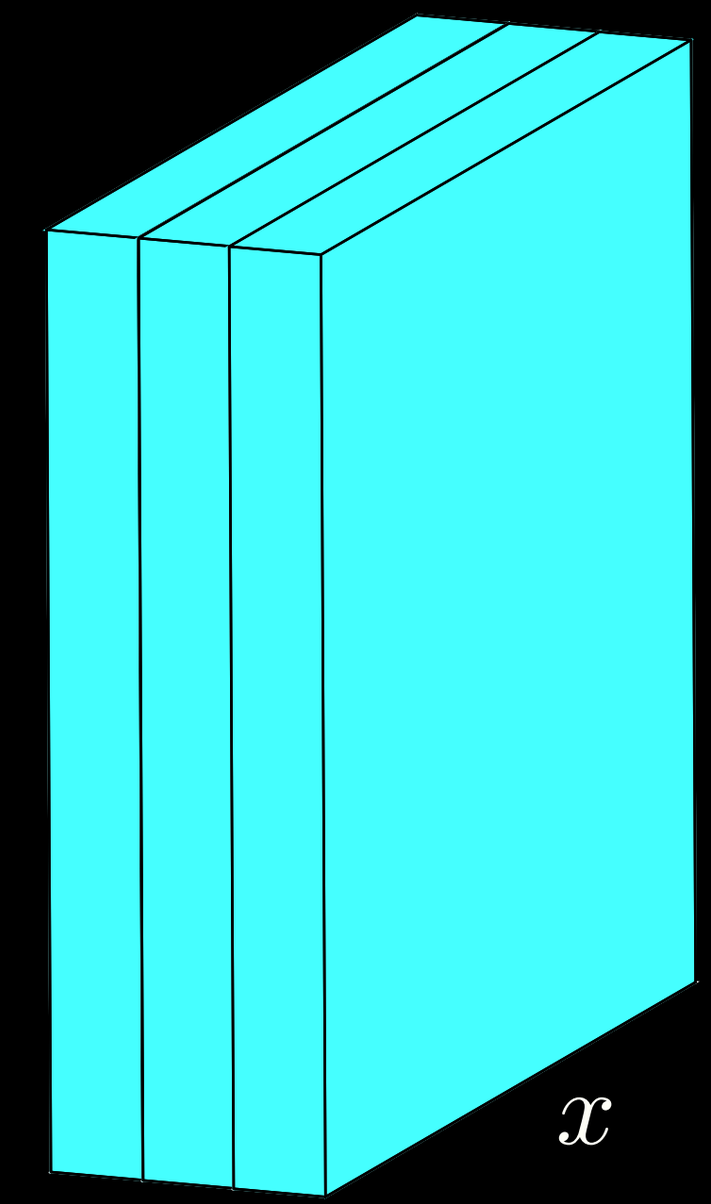

R
U

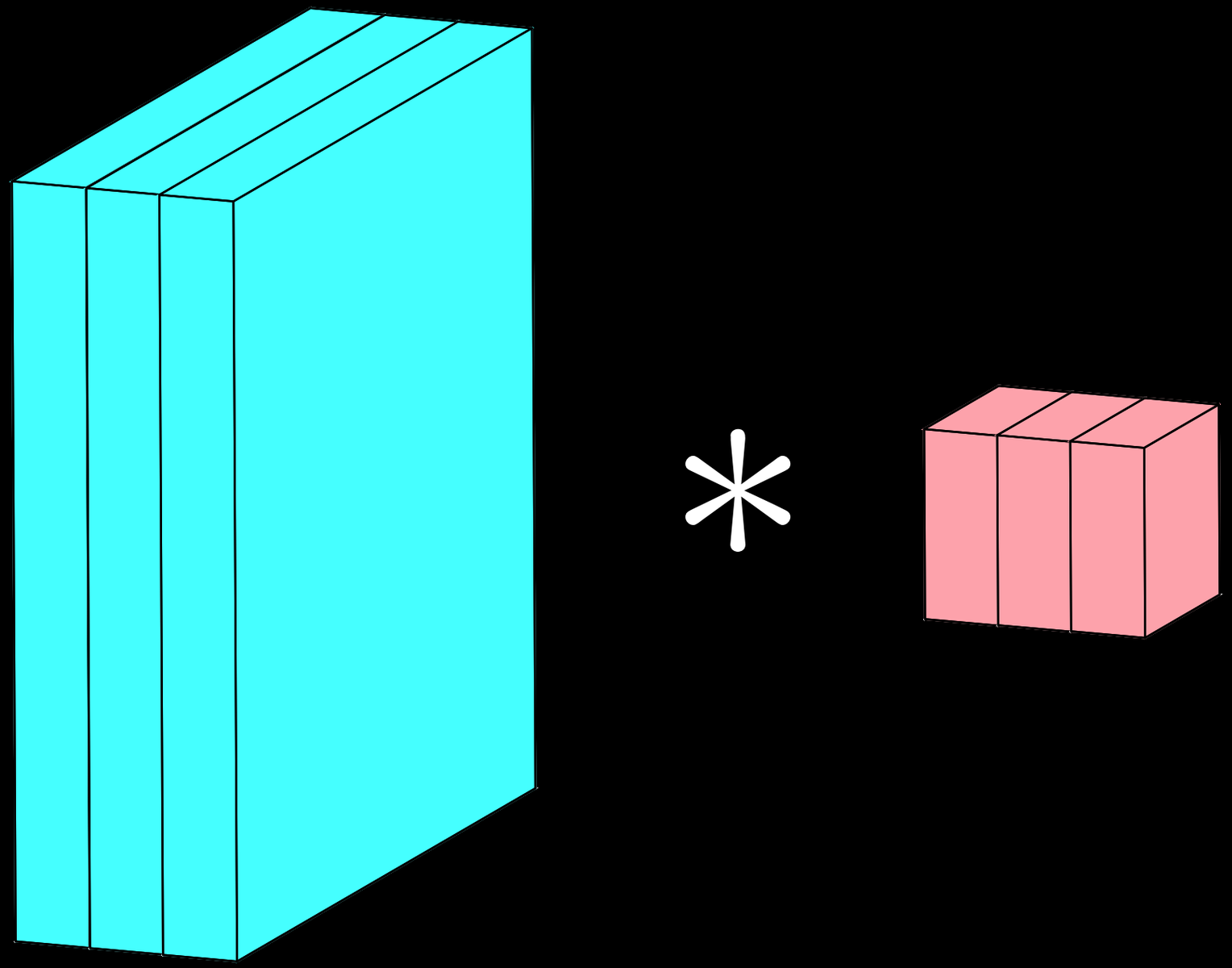


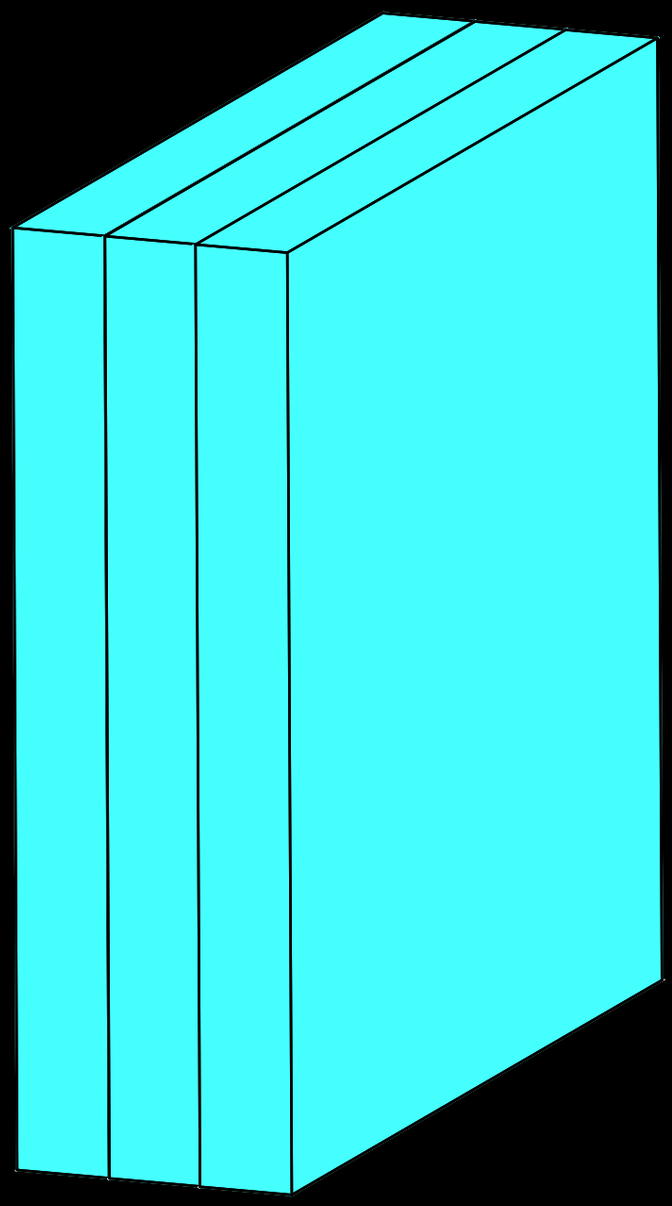



x

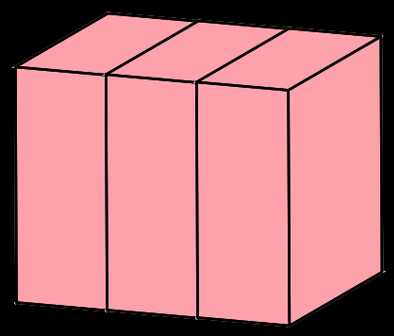

y
x


y
x
channels

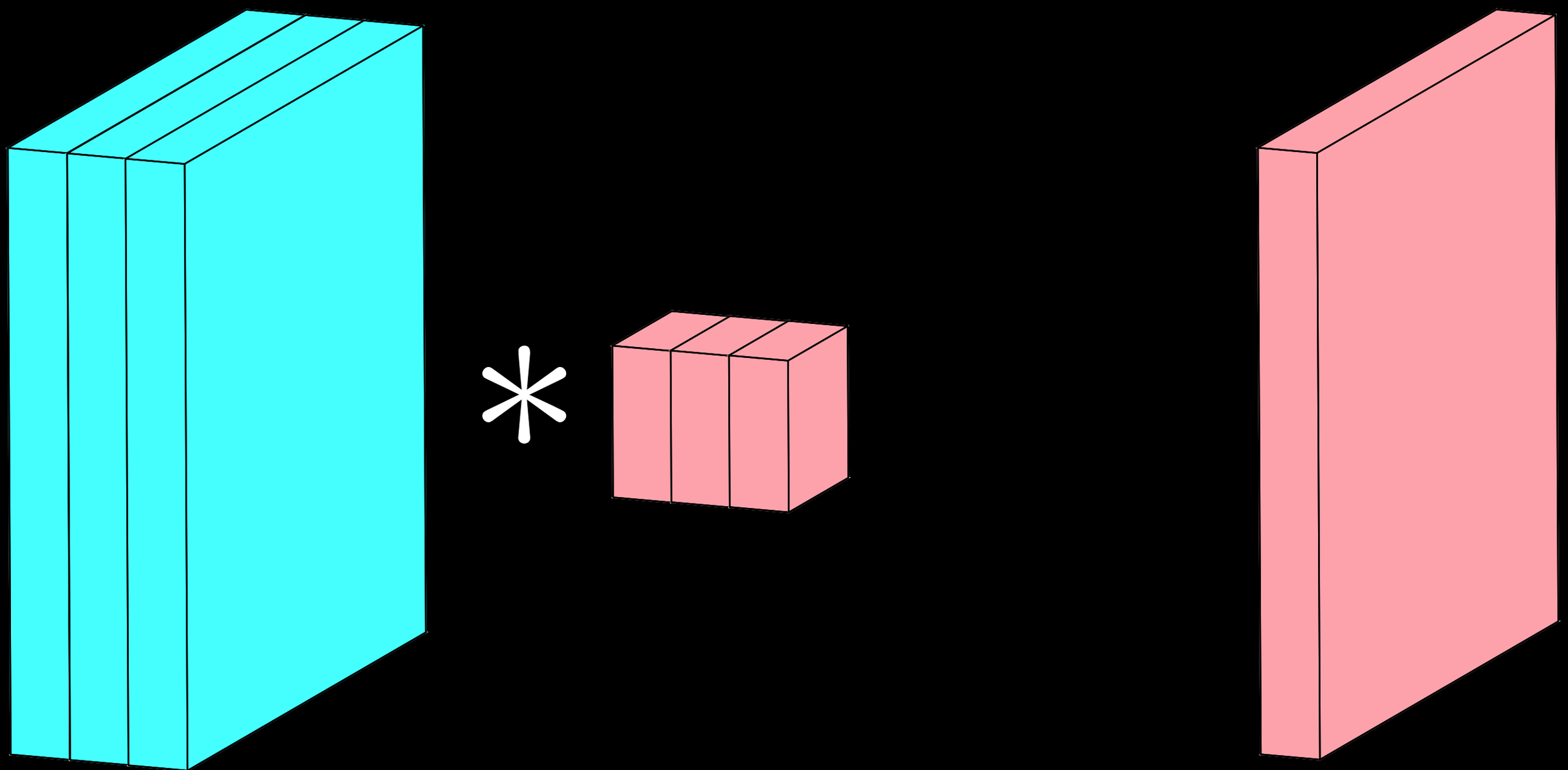


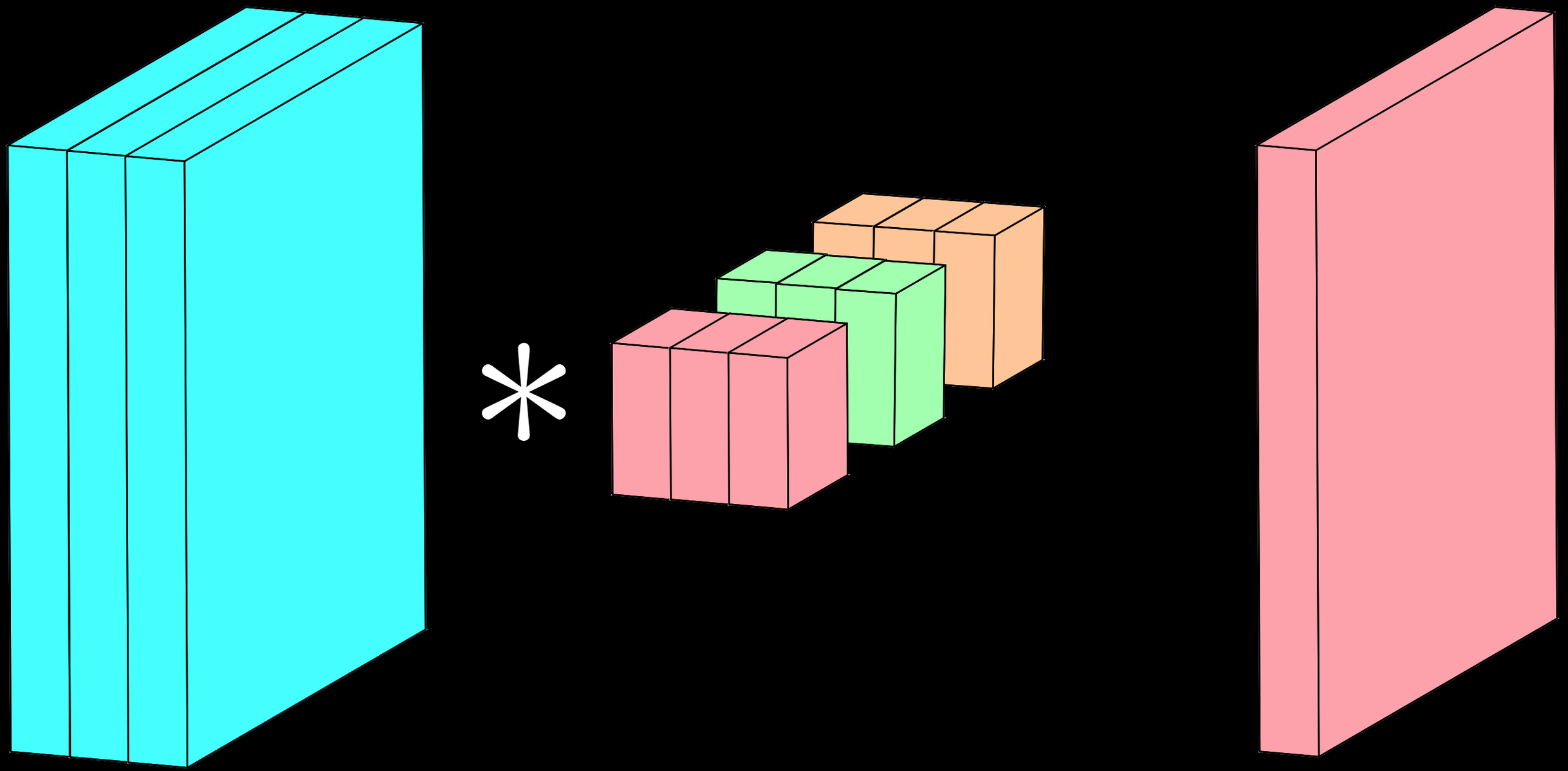


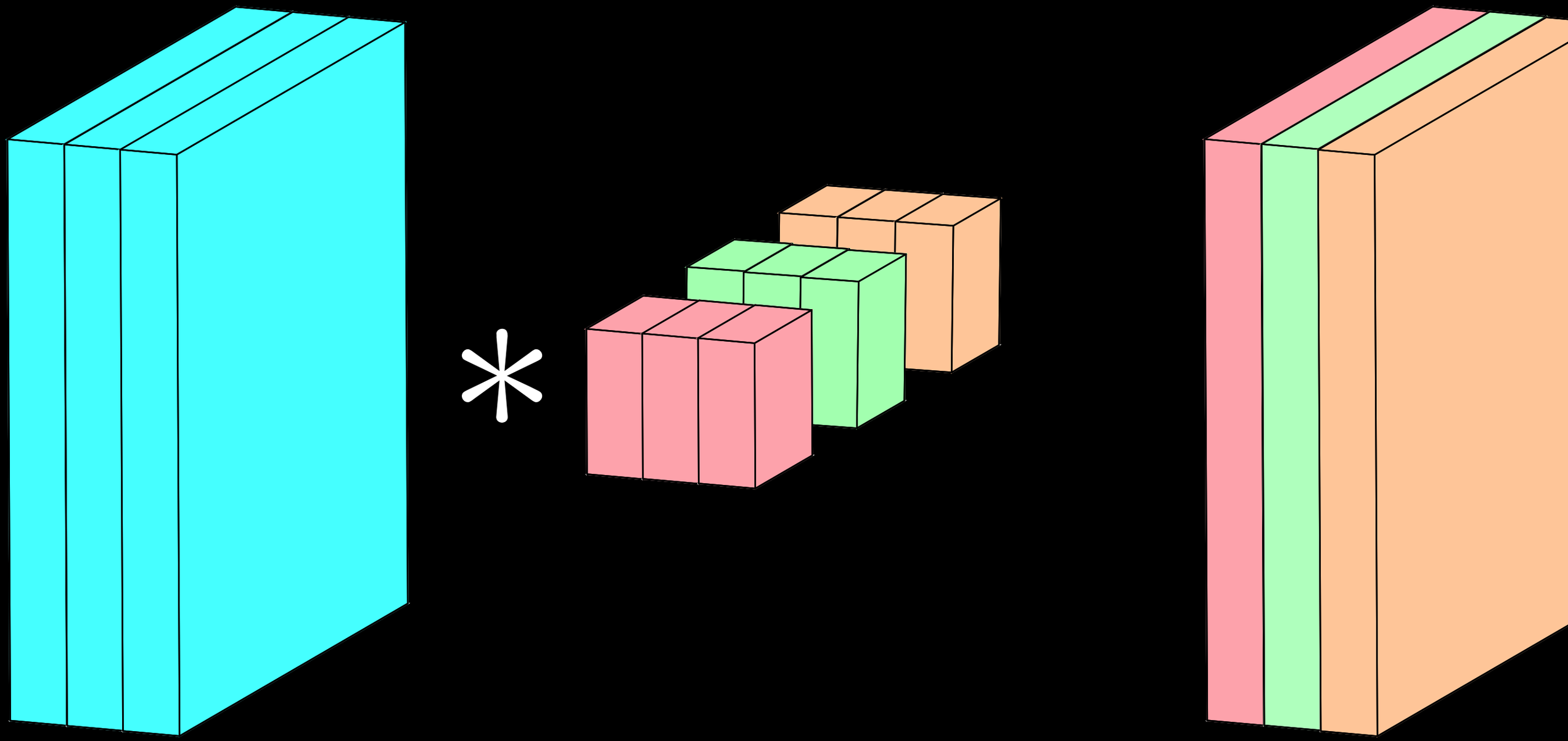
$*$



$+ \text{bias}$



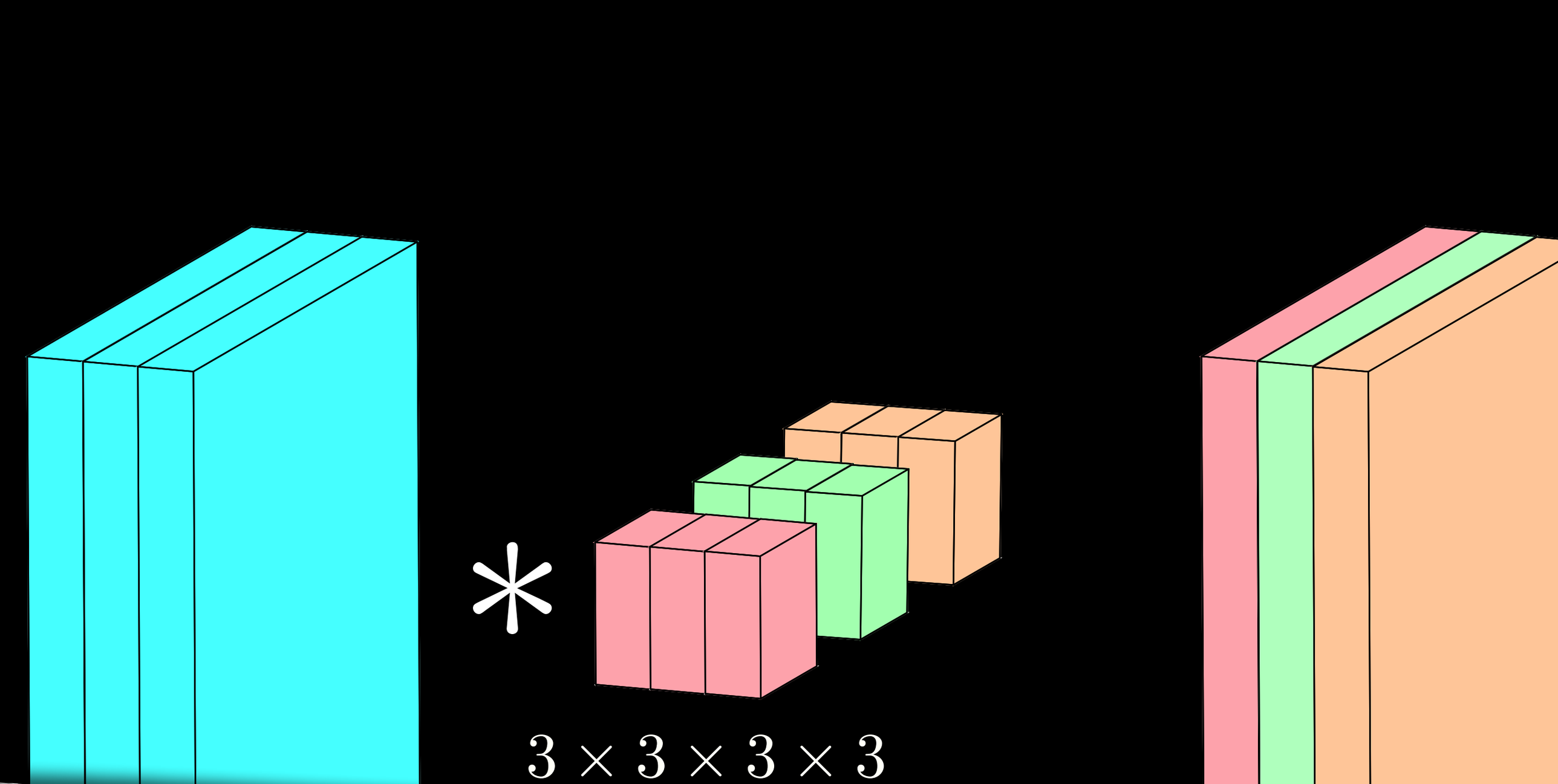




$32 \times 32 \times 3$

$3 \times 3 \times 3 \times 3$

$32 \times 32 \times 3$

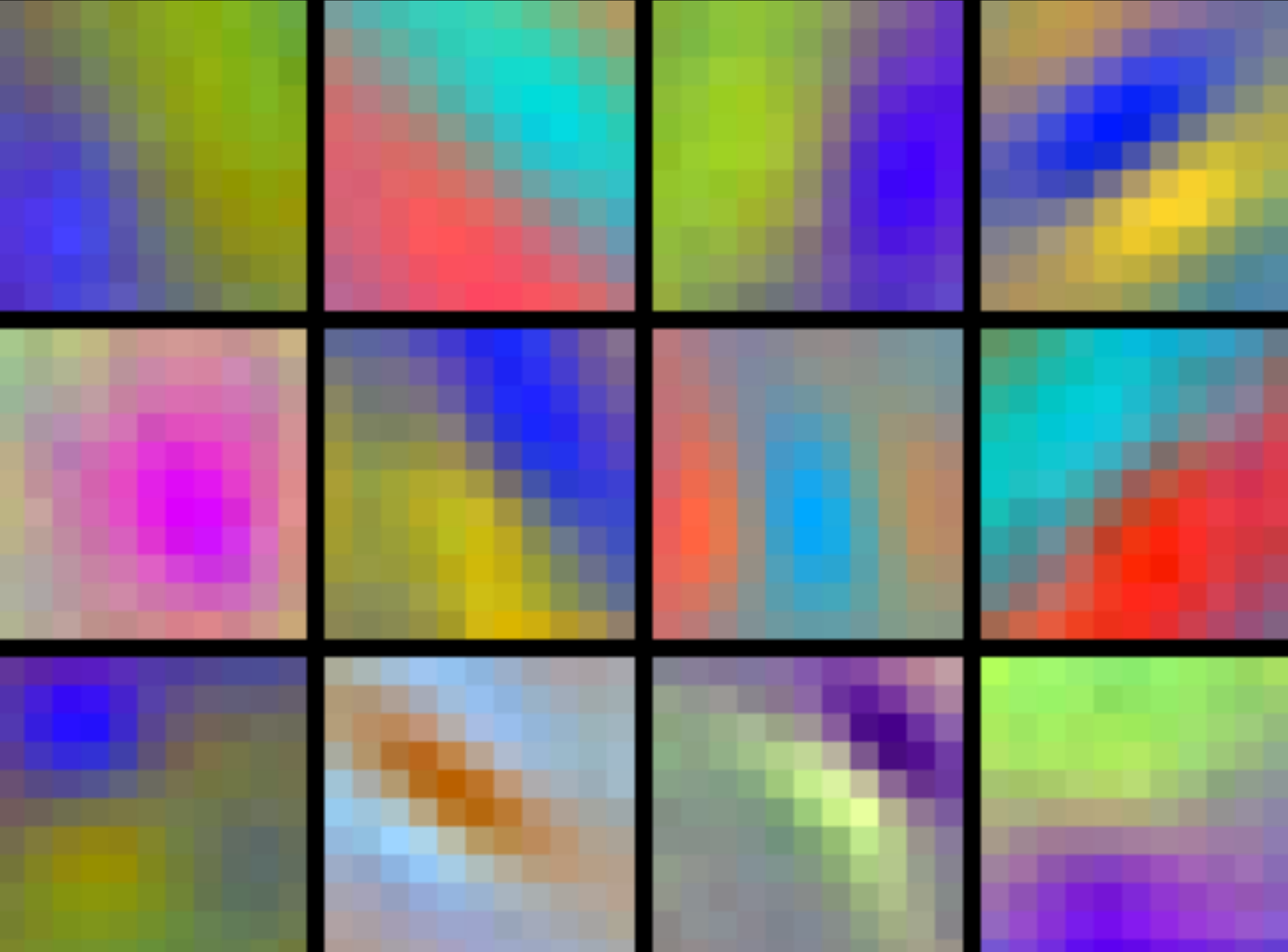


How many parameters in this layer?

$32 \times 32 \times 3$

$32 \times 32 \times 3$

AlexNet
conv layer 1



AlexNet
conv layer 1

dilated convolution

a	b	c
d	e	f
g	h	i

dilated convolution

a b c

d e f

g h i

dilation factor of one

a b c

d e f

g h i

a b c

d e f

g h i

dilation factor of two

a b c

d e f

g h i

a

b

c

d

e

f

g

h

i

dilation factor of three

What about near the image edges?


Eaton Centre
IT'S BETTER TO BE A MEMBER.
BREAK OUT OF THE CITY
What did you get?
AQUOS LCD
THE EVOLUTION OF LCD TV CONTINUES.
SHARP
Any travel on Points.
First Class
VISA
Now that's First Class.
TD Canada Trust
milestones
WOO
SHOPPERS DRUG MART
EXTREME FITNESS
The Karate Kid


H&M
Eaton Centre
BREAK OUT OF THE CITY
IT'S BETTER TO BE A MEMBER.
What did you get?


Eaton Centre
What did you get?
IT'S BETTER TO BE A MEMBER.
BREAK OUT OF THE CITY

clip padding

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

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

clip padding


Eaton Centre
What did you get?

symmetric padding


Eaton Centre
What did you get?

replicate padding


Eaton Centre
What did you get?
IT'S BETTER TO BE A MEMBER.
BREAK OUT OF THE CITY


circular padding
Eaton Centre
What did you get?
IT'S BETTER TO BE A MEMBER.

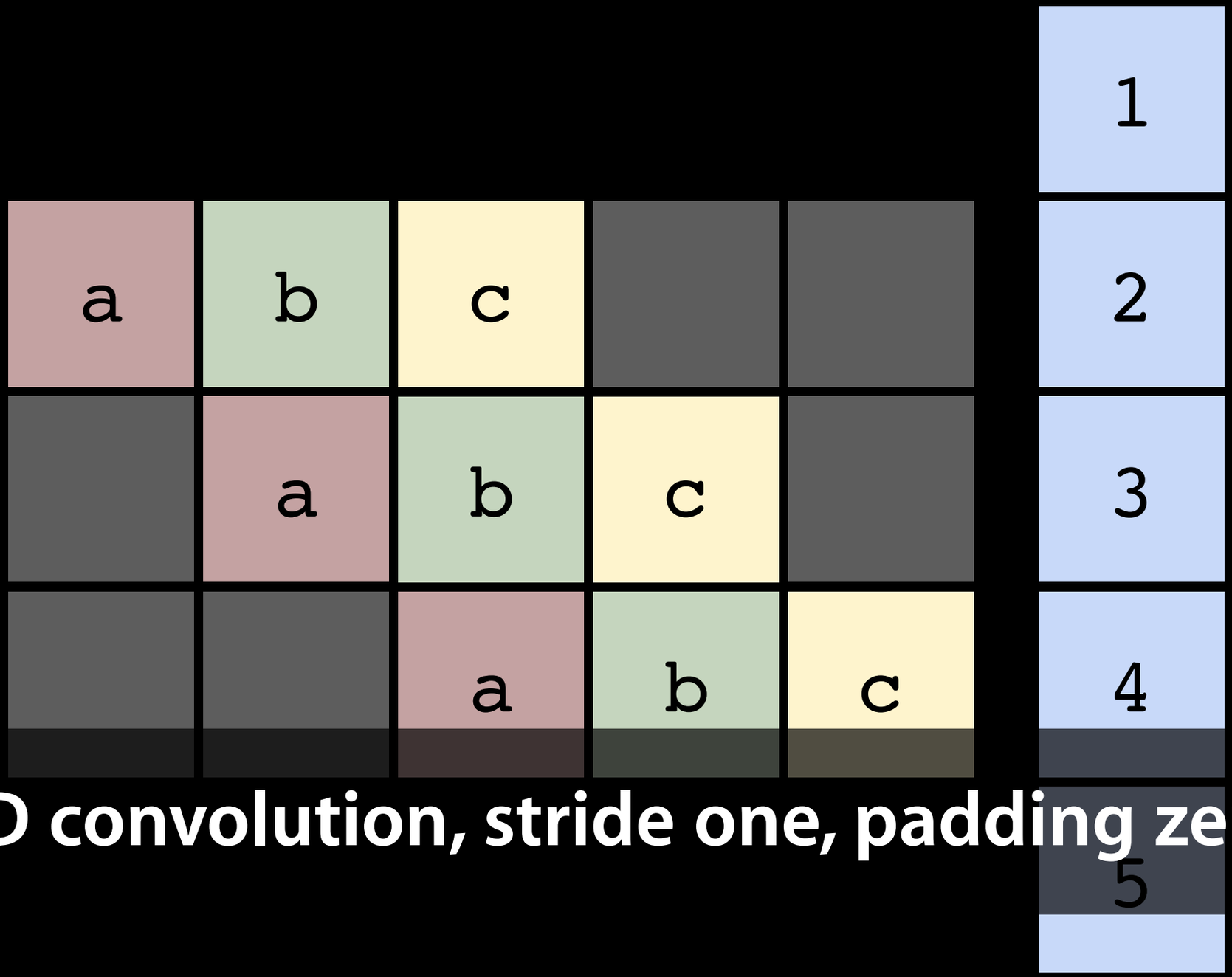
Convolution as Matrix Multiplication

$$I[x,y] * F[x,y]$$

1	2	3	4	5	*	a	b	c

How would you write as a matrix multiplication?

a	b	c			1
	a	b	c		3
		a	b	c	4
					5



1D convolution, stride one, padding zero

optimized convolution

a	b	c

1	2	3	4	5

*

d	e	f

g	h	i

optimized convolution

a	b	c

1	2	3	4	5

*

d	e	f

g	h	i

equivalent to matrix-to-matrix multiplication

1	2	3	4	5

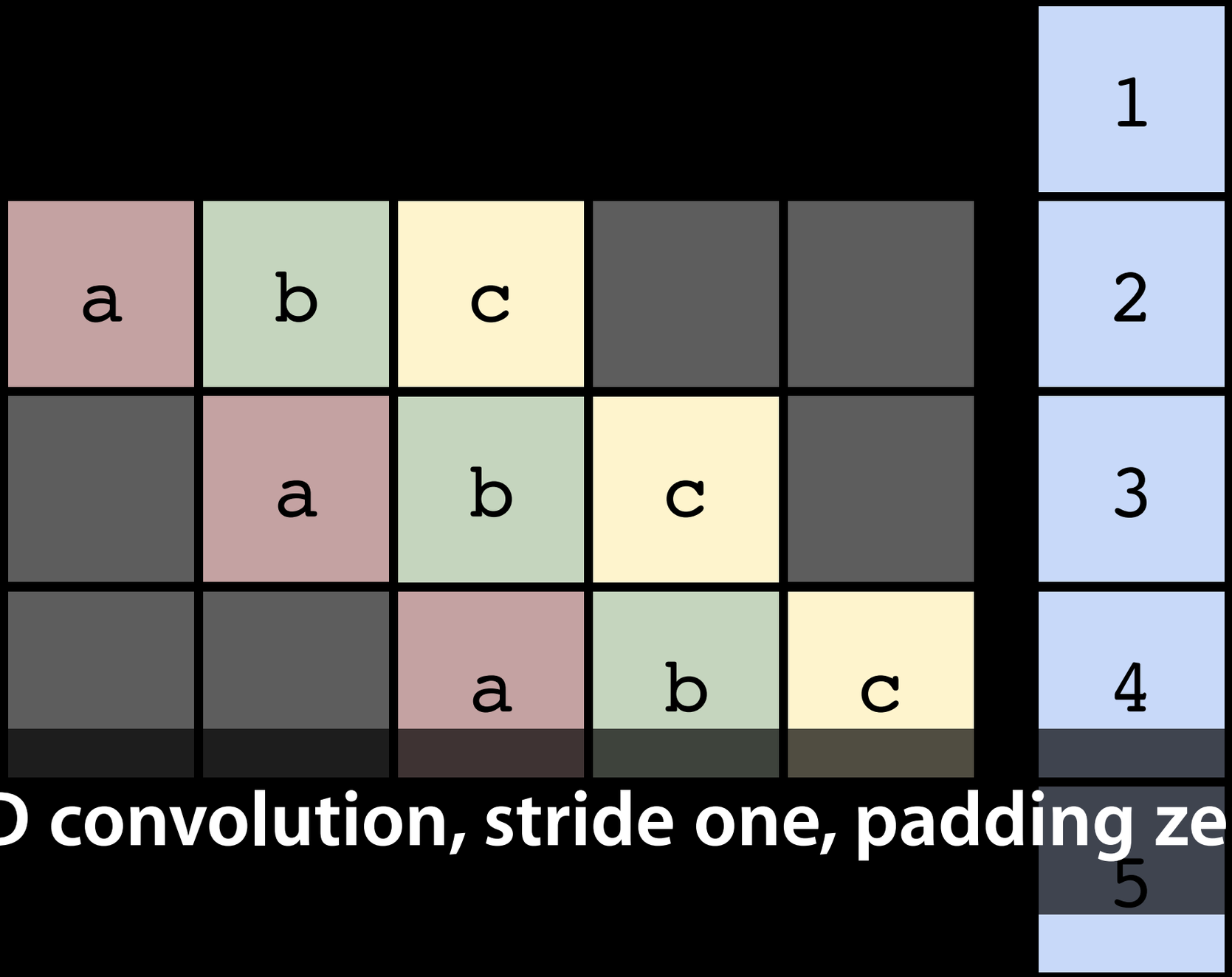
*

a	b	c
d	e	f
g	h	i

equivalent to matrix-to-matrix multiplication

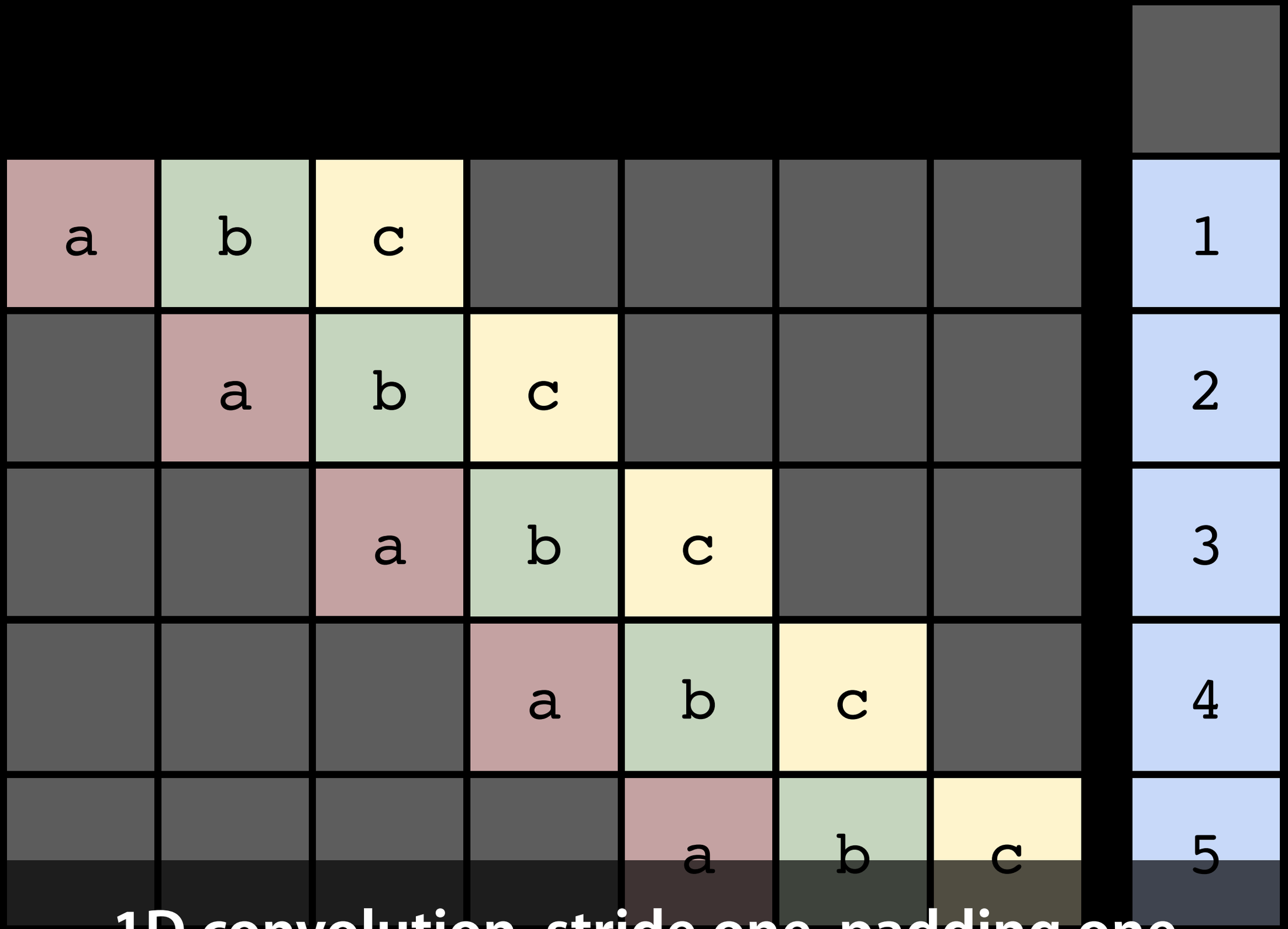
1	2	3
2	3	4
3	4	5

a	d	g
b	e	h
c	f	i



1D convolution, stride one, padding zero

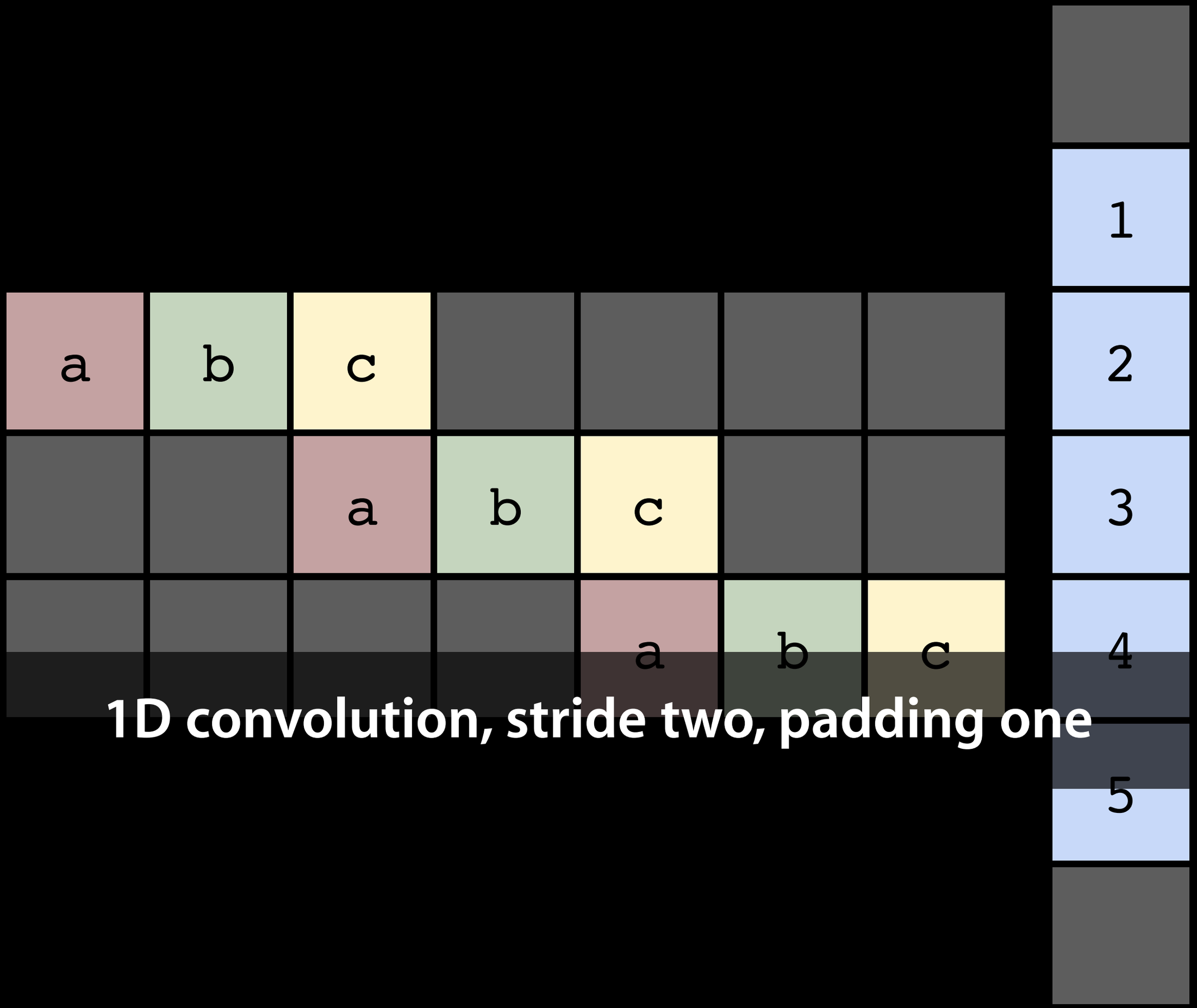
a	b	c					1
	a	b	c				2
		a	b	c			3
			a	b	c		4
				a	b	c	5



1D convolution, stride one, padding one

a	b	c					1
	a	b	c				2
		a	b	c			3
			a	b	c		4
				a	b	c	5

a	b	c					1
		a	b	c			2
				a	b	c	3
							4
							5



1D convolution, stride two, padding one

Feature Pooling Layers

average pooling

1	1	4	2
7	3	9	5
1	1	7	3
4	2	8	6

1	1	4	2
7	3	9	5
1	1	7	3
4	2	8	6

2-by-2 average pool with stride 2

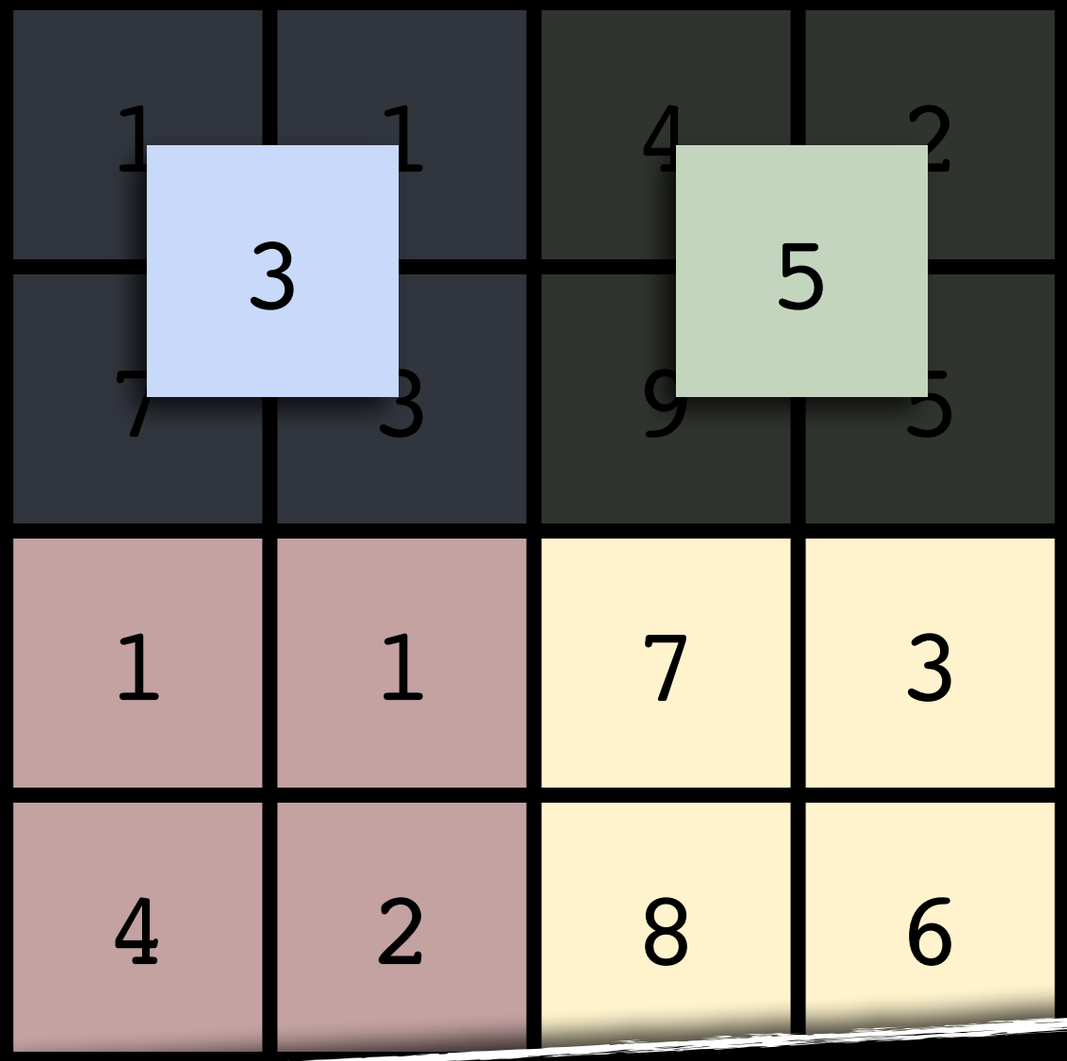
1	1	4	2
7	3	9	5
1	1	7	3
4	2	8	6

2-by-2 average pool with stride 2

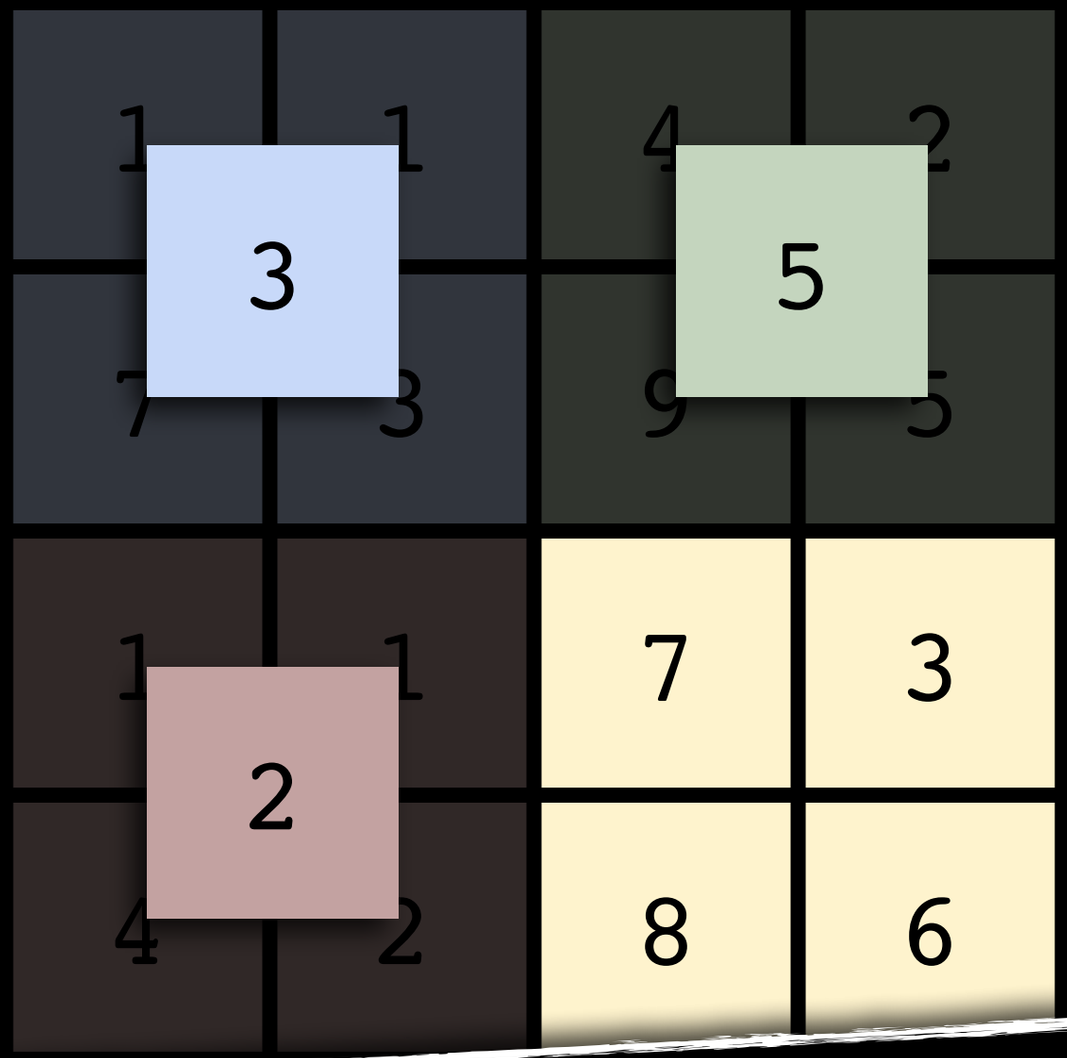
1	1	4	2
7	3	9	5

1	1	7	3
4	2	8	6

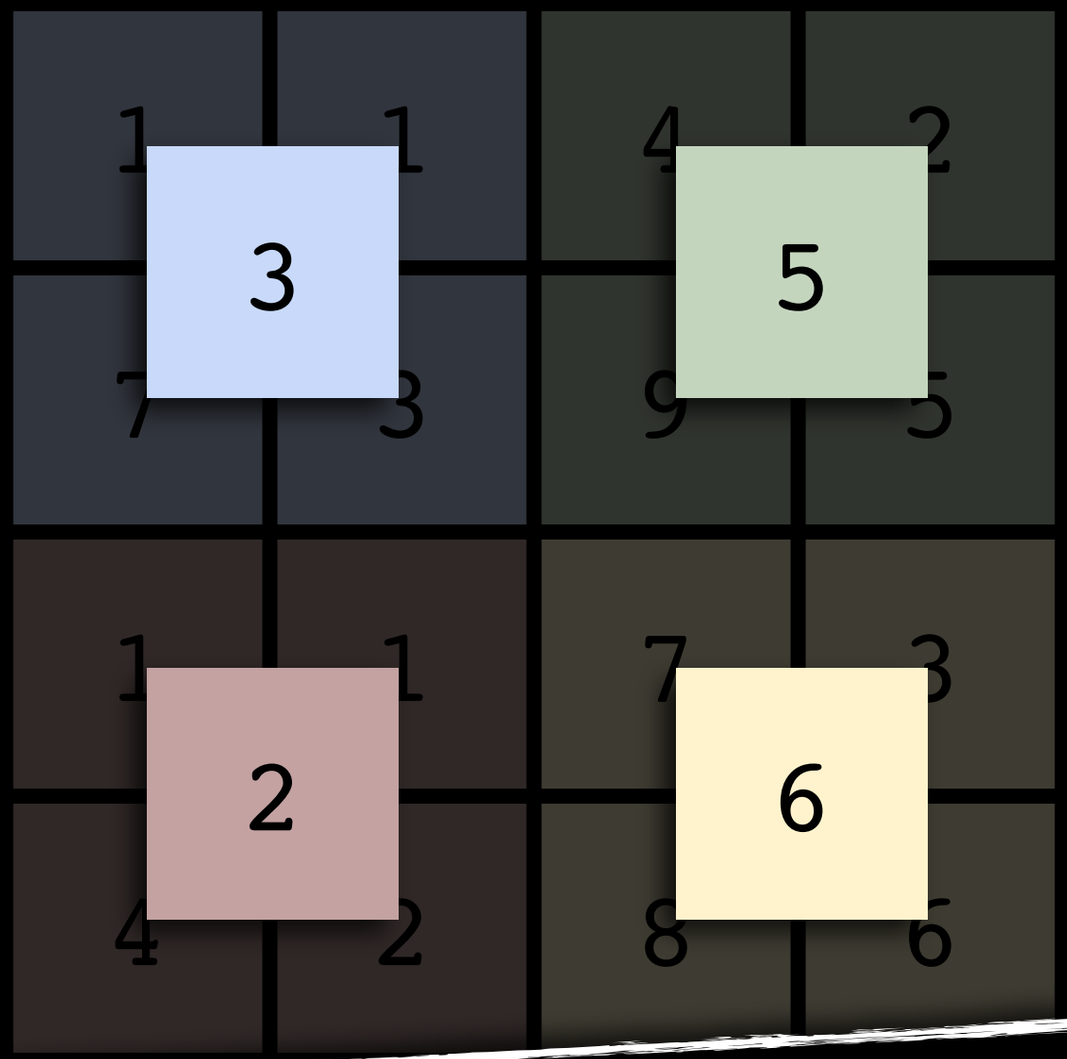
2-by-2 average pool with stride 2



2-by-2 average pool with stride 2



2-by-2 average pool with stride 2



2-by-2 average pool with stride 2

1	1	4	2
7	3	9	5
1	1	7	3
4	2	8	6

3	5
2	6

2-by-2 average pool with stride 2

1	1	4	2
7	3	9	5

3	5

1	1	7	3
4	2	8	6

2	6

$$h_k(x, y, c) = \frac{1}{|\mathcal{N}(x, y)|} \sum_{(i,j) \in \mathcal{N}(x,y)} h_{k-1}(i, j, c)$$

max
pooling

1	1	4	2
7	3	9	5
1	1	7	3
4	2	8	6

1	1	4	2
7	3	9	5
1	1	7	3
4	2	8	6

2-by-2 max pool with stride 2

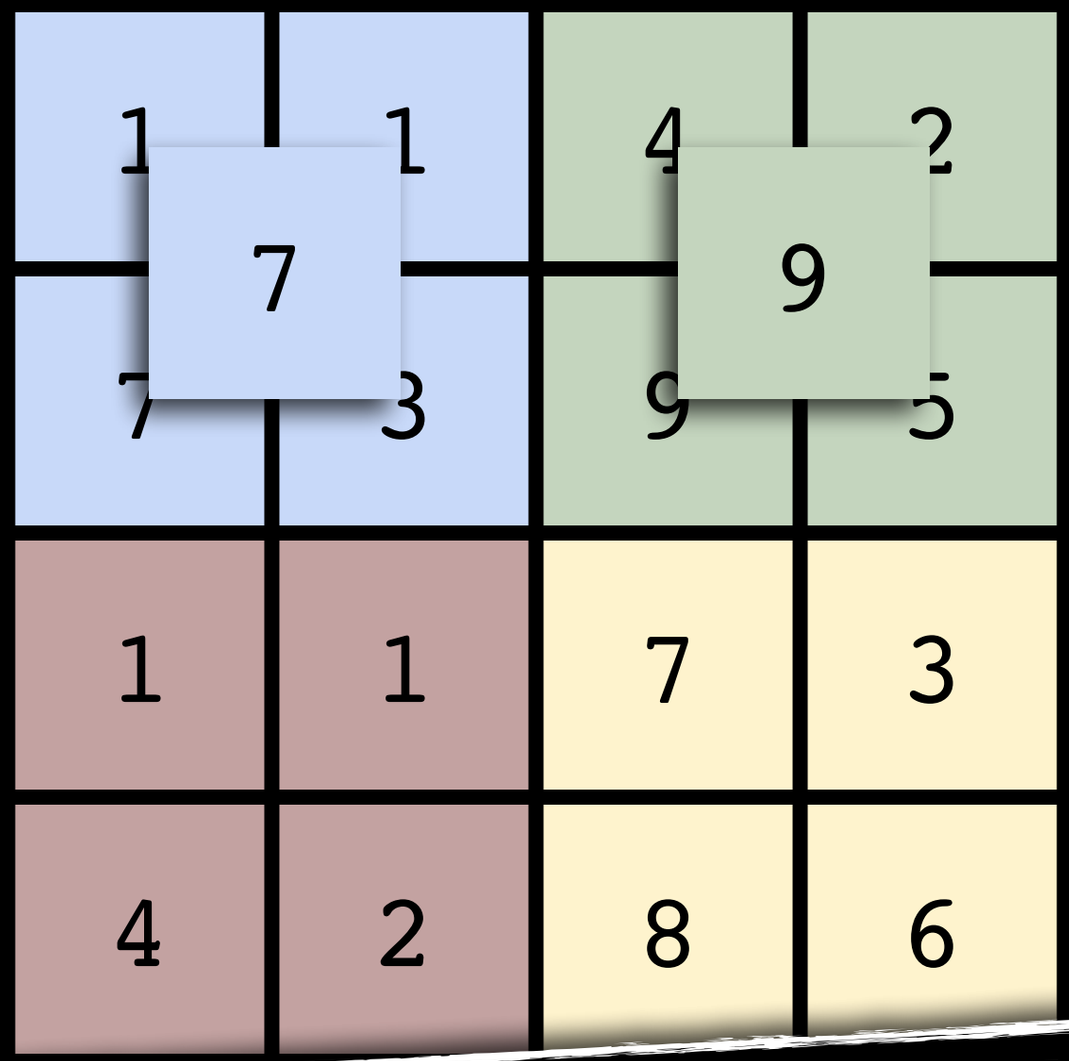
1	1	4	2
7	3	9	5
1	1	7	3
4	2	8	6

2-by-2 max pool with stride 2

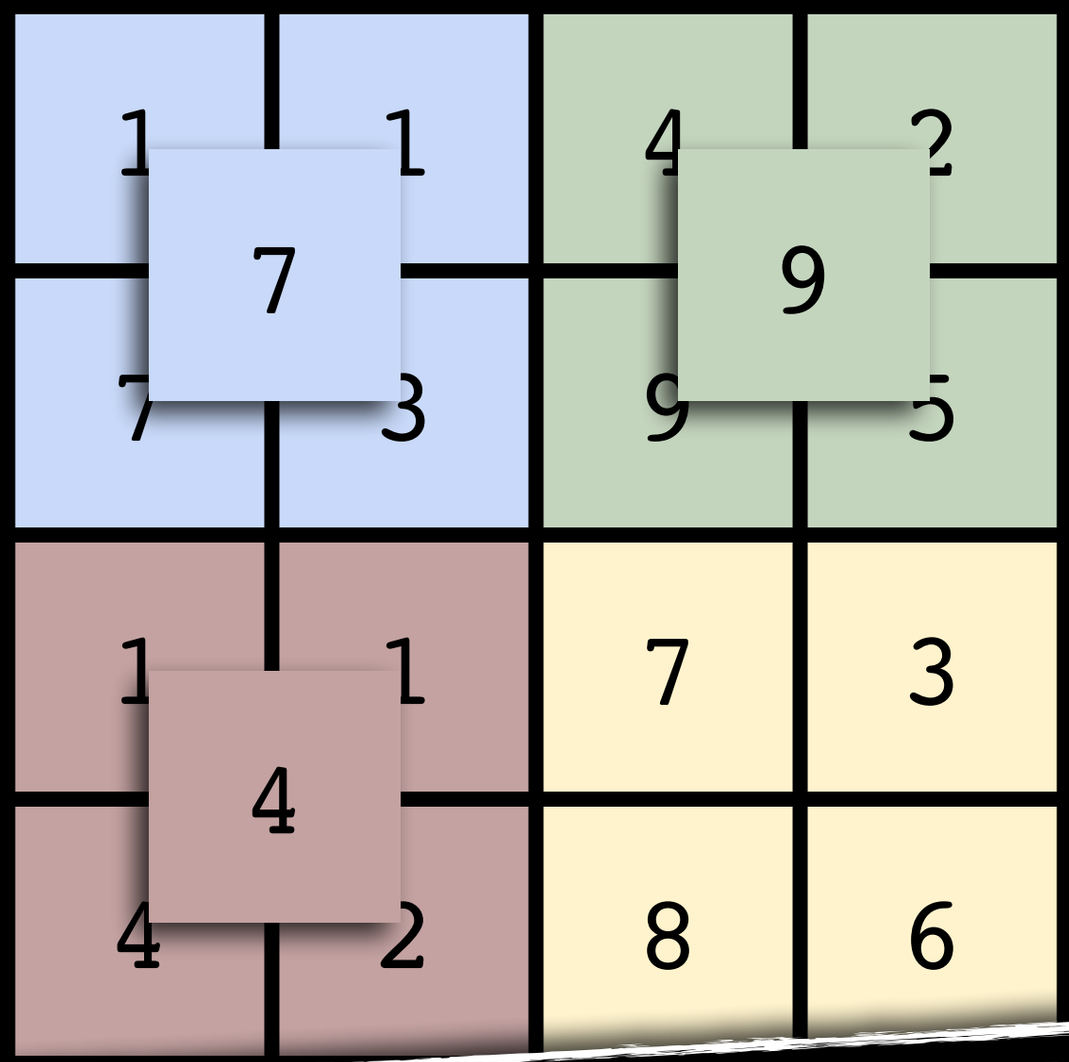
1	1	4	2
7	3	9	5

1	1	7	3
4	2	8	6

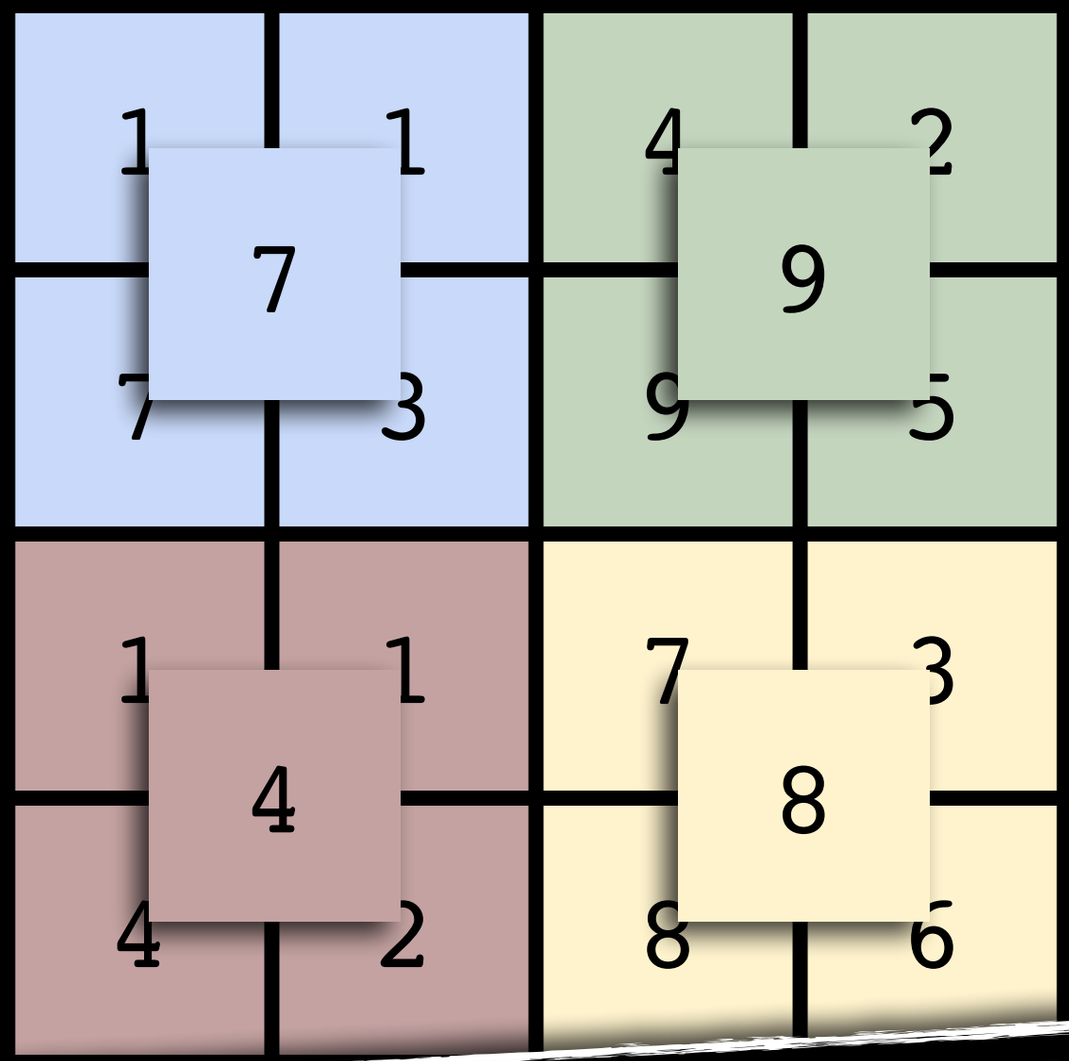
2-by-2 max pool with stride 2



2-by-2 max pool with stride 2



2-by-2 max pool with stride 2



2-by-2 max pool with stride 2

1	1	4	2
7	3	9	5
1	1	7	3
4	2	8	6

7	9
4	8

2-by-2 max pool with stride 2

1	1	4	2
7	3	9	5

7	9

1	1	7	3
4	2	8	6

4	8

$$h_k(x, y, c) = \max_{(i,j)\in\mathcal{N}(x,y)} h_{k-1}(i, j, c)$$

STRIVING FOR SIMPLICITY: THE ALL CONVOLUTIONAL NET

Jost Tobias Springenberg*, **Alexey Dosovitskiy***, **Thomas Brox, Martin Riedmiller**
Department of Computer Science
University of Freiburg
Freiburg, 79110, Germany
{springj, dosovits, brox, riedmiller}@cs.uni-freiburg.de

ABSTRACT

Most modern convolutional neural networks (CNNs) used for object recognition are built using the same principles: Alternating convolution and max-pooling layers followed by a small number of fully connected layers. We re-evaluate the state of the art for object recognition from small images with convolutional networks, questioning the necessity of different components in the pipeline. We find that max-pooling can simply be replaced by a convolutional layer with increased stride without loss in accuracy on several image recognition benchmarks. Following this finding – and building on other recent work for finding simple network structures – we propose a new architecture that consists solely of convolutional layers and yields competitive or state of the art performance on several object recognition datasets (CIFAR-10, CIFAR-100, ImageNet). To analyze the network we introduce a new variant of the "deconvolution approach" for visualizing features learned by CNNs, which can be applied to a broader range of network structures than existing approaches.

1 INTRODUCTION AND RELATED WORK

The vast majority of modern convolutional neural networks (CNNs) used for object recognition are

ABSTRACT

Most modern convolutional neural networks (CNNs) used for object recognition are built using the same principles: Alternating convolution and max-pooling layers followed by a small number of fully connected layers. We re-evaluate the state of the art for object recognition from small images with convolutional networks, questioning the necessity of different components in the pipeline. We find that max-pooling can simply be replaced by a convolutional layer with increased stride without loss in accuracy on several image recognition benchmarks. Following this finding – and building on other recent work for finding simple network structures – we propose a new architecture that consists solely of convolutional layers and yields competitive or state of the art performance on several object recognition datasets (CIFAR-10, CIFAR-100, ImageNet). To analyze the network we introduce a new variant of the "deconvolution approach" for visualizing features learned by CNNs, which can be applied to a broader range of network structures than existing approaches.

INTRODUCTION AND RELATED WORK

vast majority of modern convolutional neural networks (CNNs) used for object recognitio

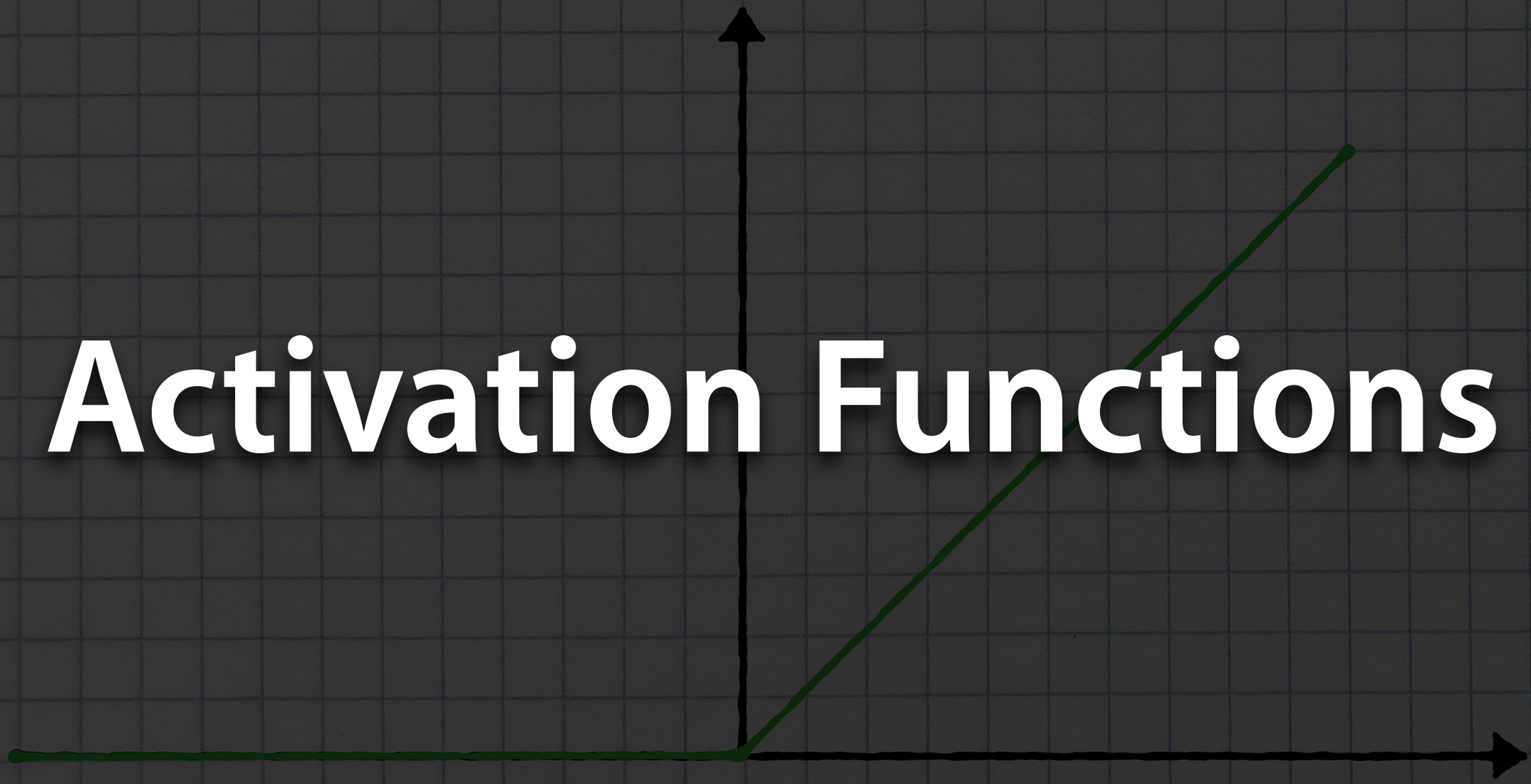
Abstract

Most modern convolutional neural networks (CNNs) used for object recognition are built using the same principles: Alternating convolution and max-pooling layers followed by a small number of fully connected layers. We re-evaluate the state of the art for object recognition from small images with convolutional networks, questioning the necessity of different components in the pipeline. We find that max-pooling can simply be replaced by a convolutional layer with increased stride without loss in accuracy on several image recognition benchmarks. Following this finding – and building on other recent work for finding simple network structures – we propose a new architecture that consists solely of convolutional layers and yields competitive or state of the art performance on several object recognition datasets (CIFAR-10, CIFAR-100, ImageNet). To analyze the network we introduce a new variant of the "deconvolution approach" for visualizing features learned by CNNs, which can be applied to a broader range of network structures than existing approaches.

Introduction and Related Work

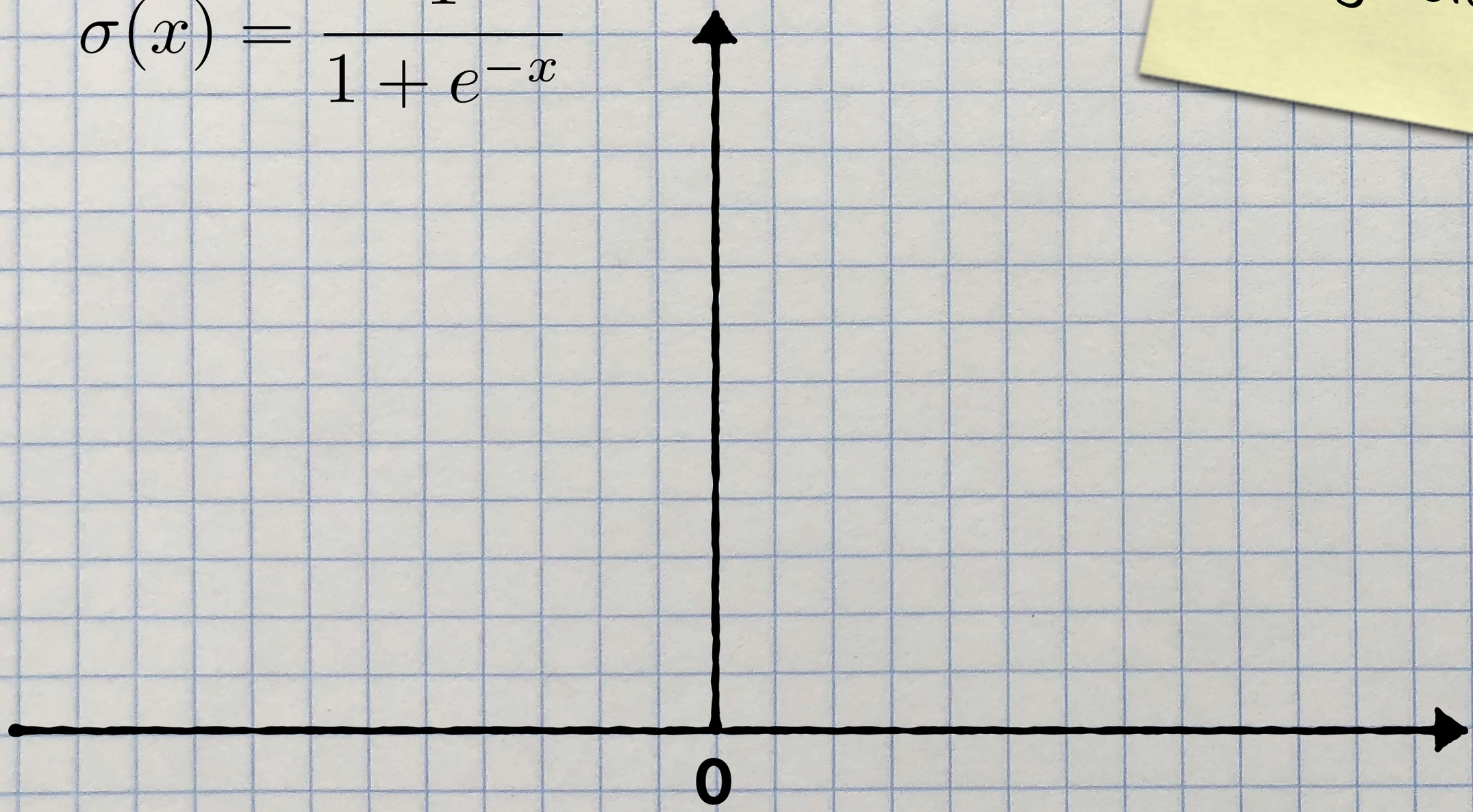
vast majority of modern convolutional neural networks (CNNs) used for object recognitio

Activation Functions



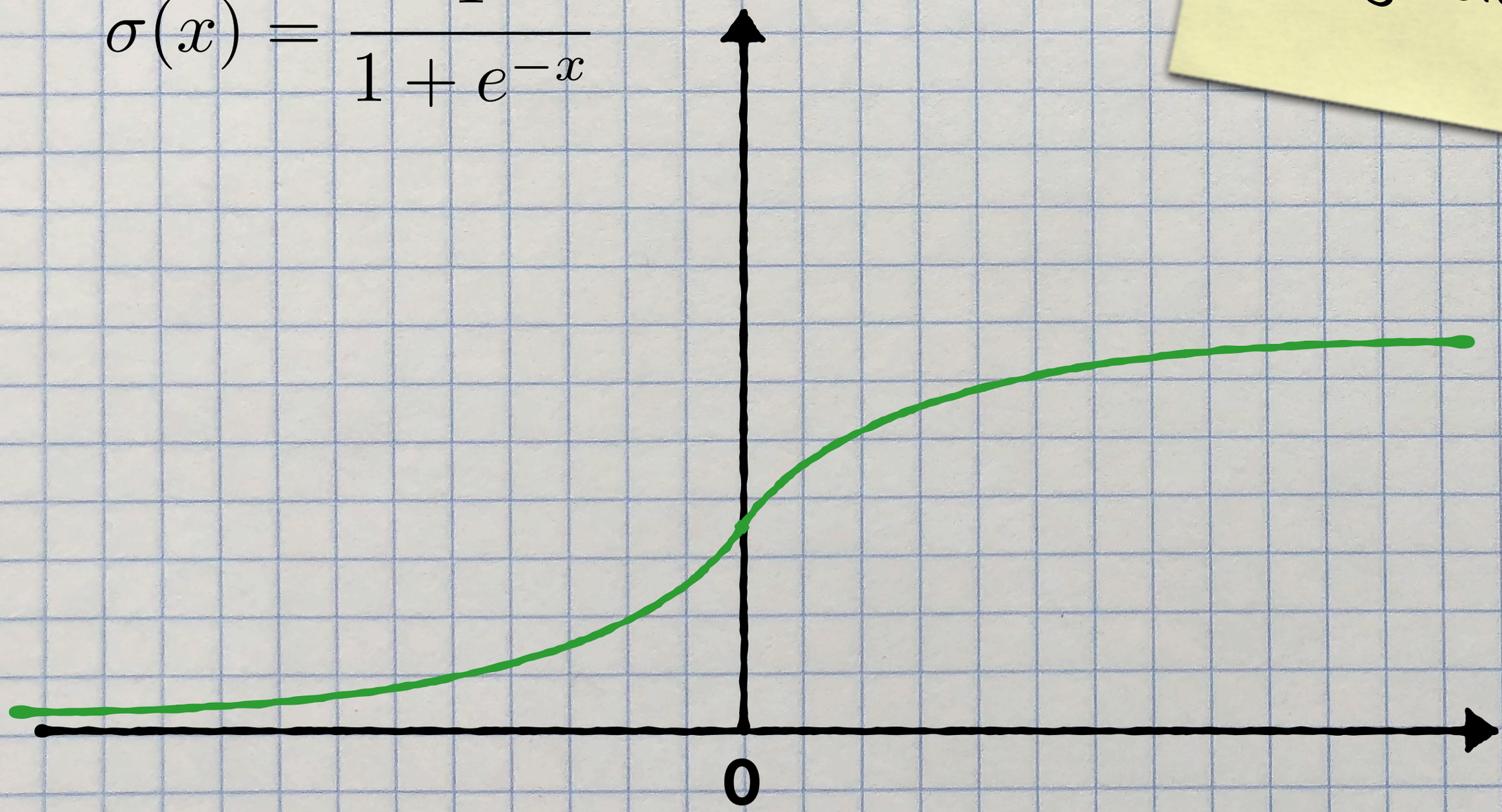
sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



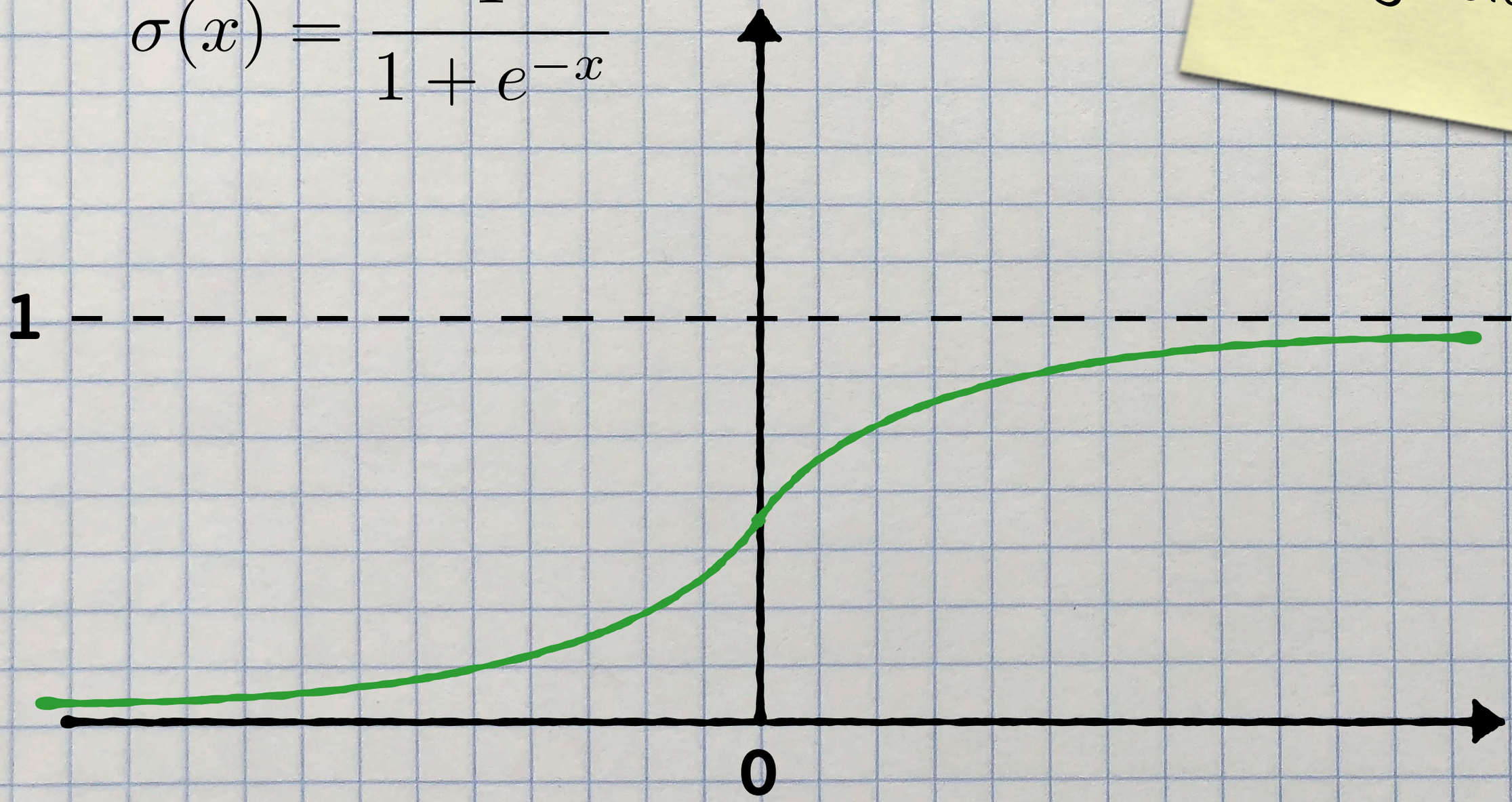
sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



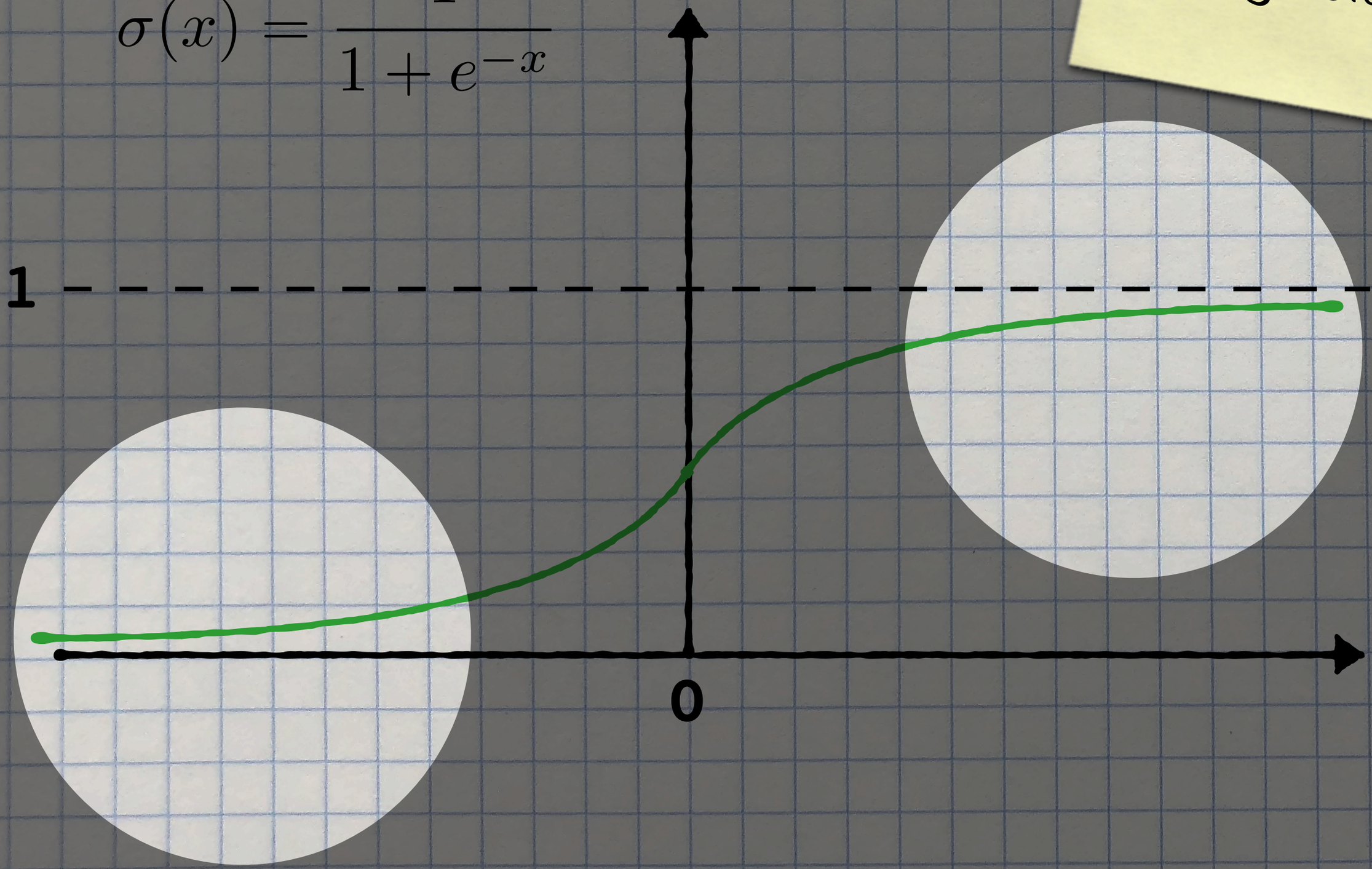
sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

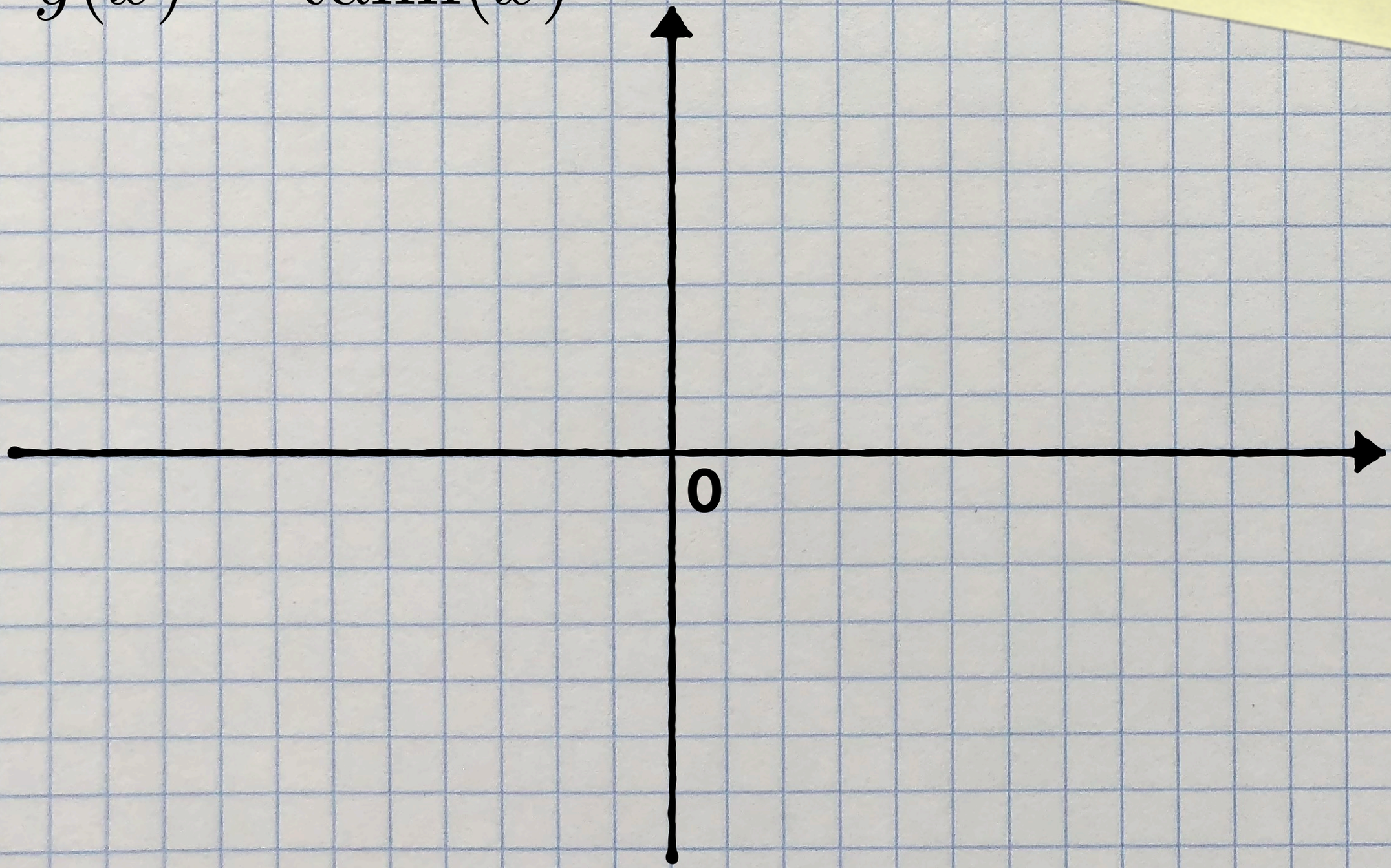


sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



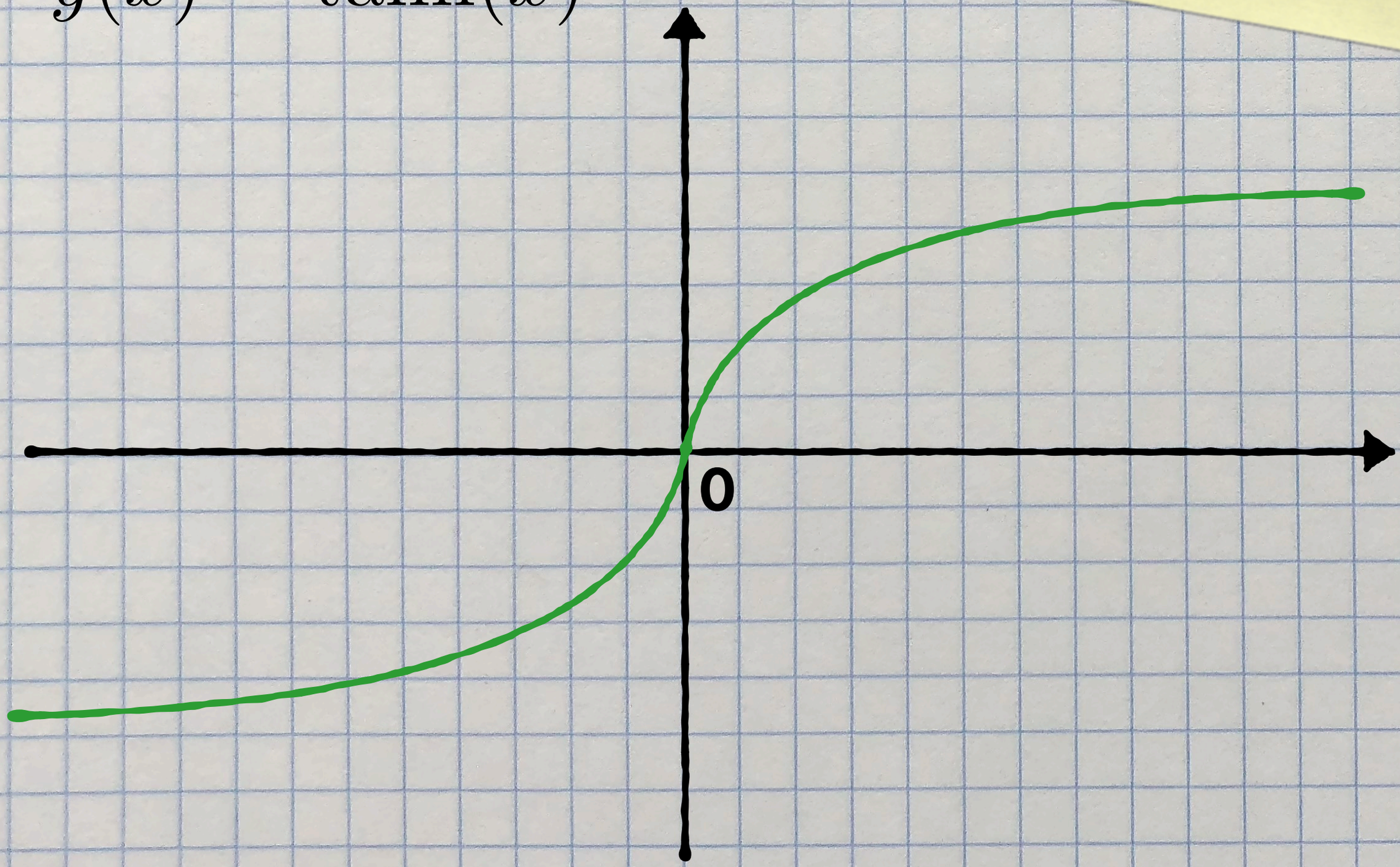
$$y(x) = \tanh(x)$$



hyperbolic tangent

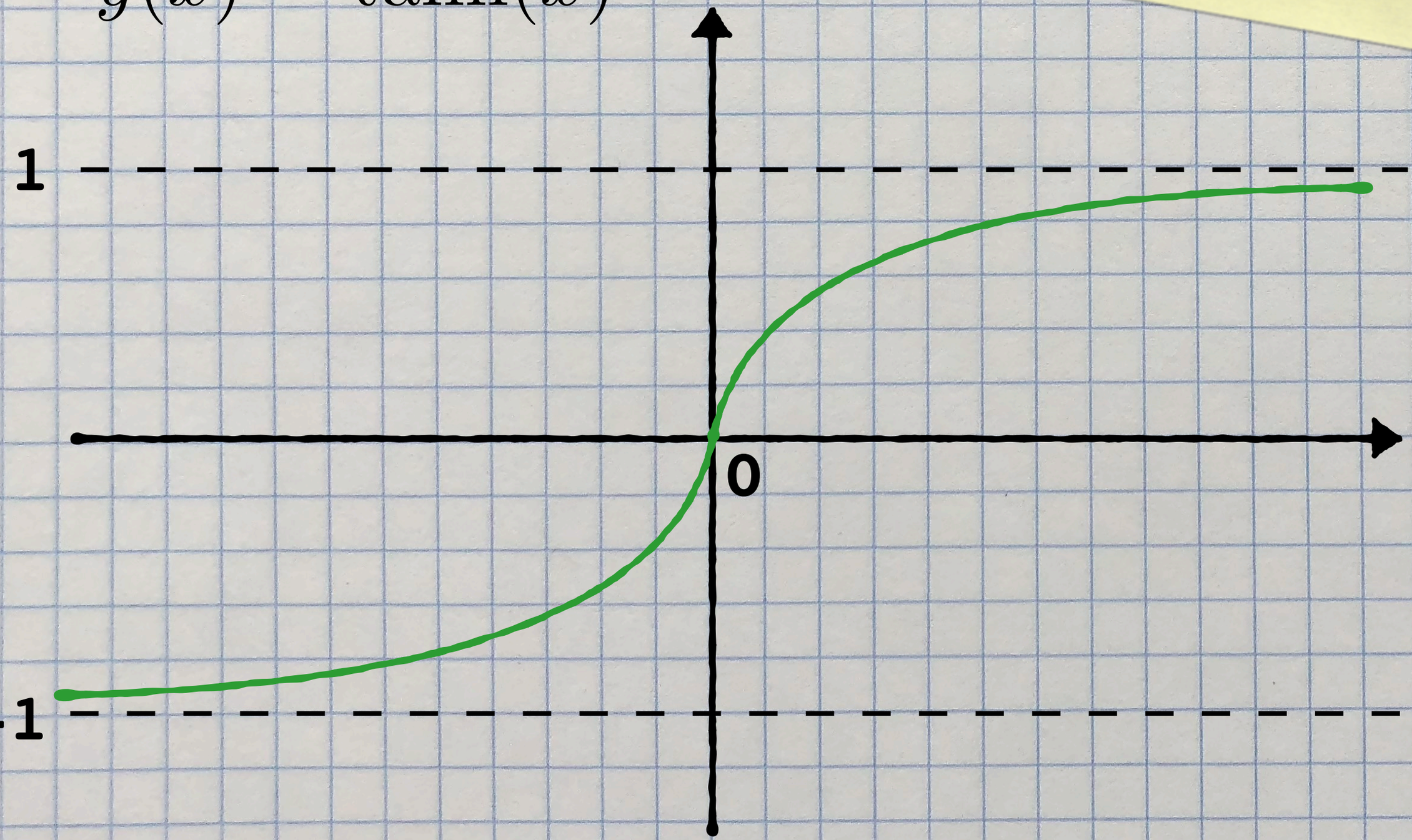
$$y(x) = \tanh(x)$$

hyperbolic tangent



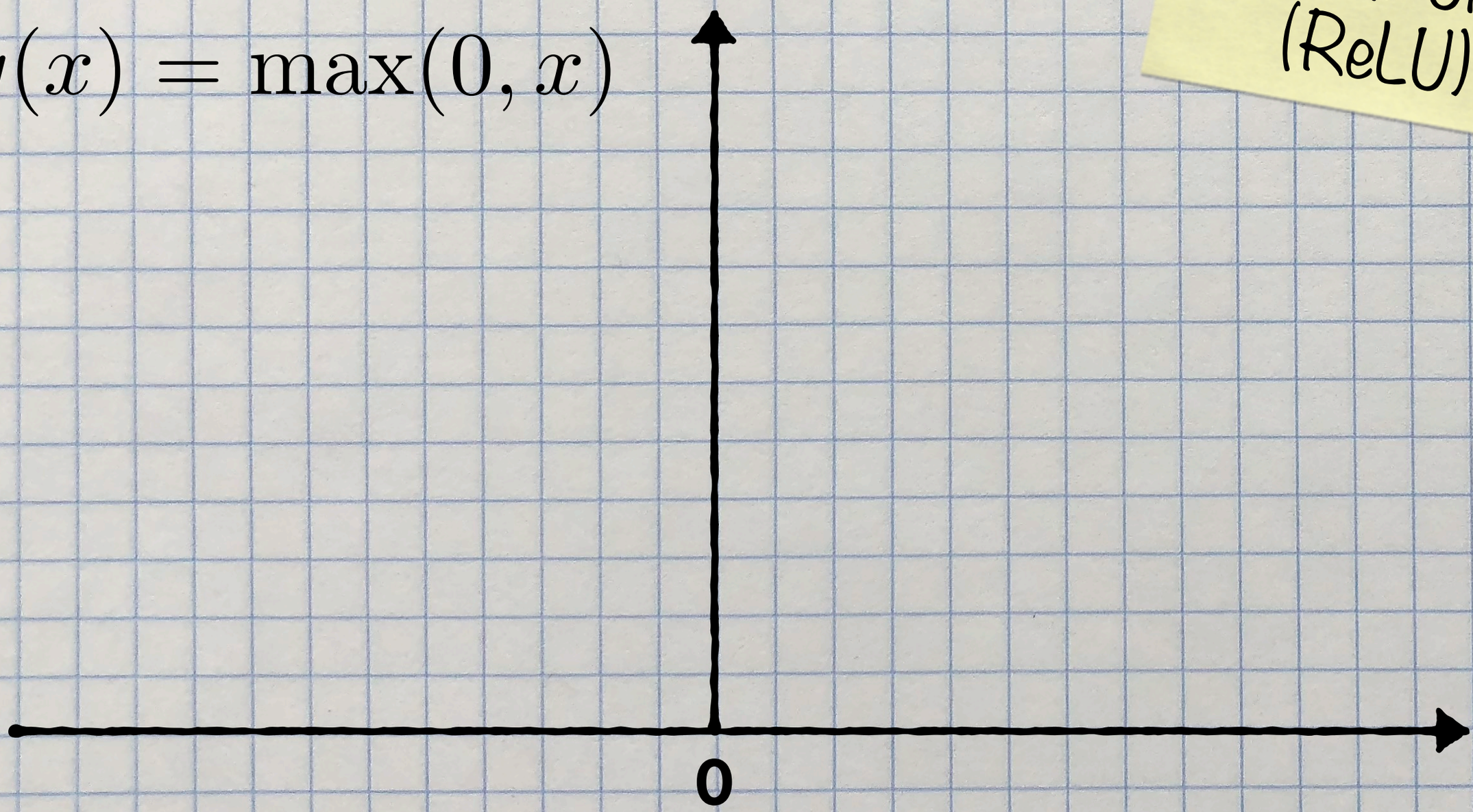
hyperbolic tangent

$$y(x) = \tanh(x)$$



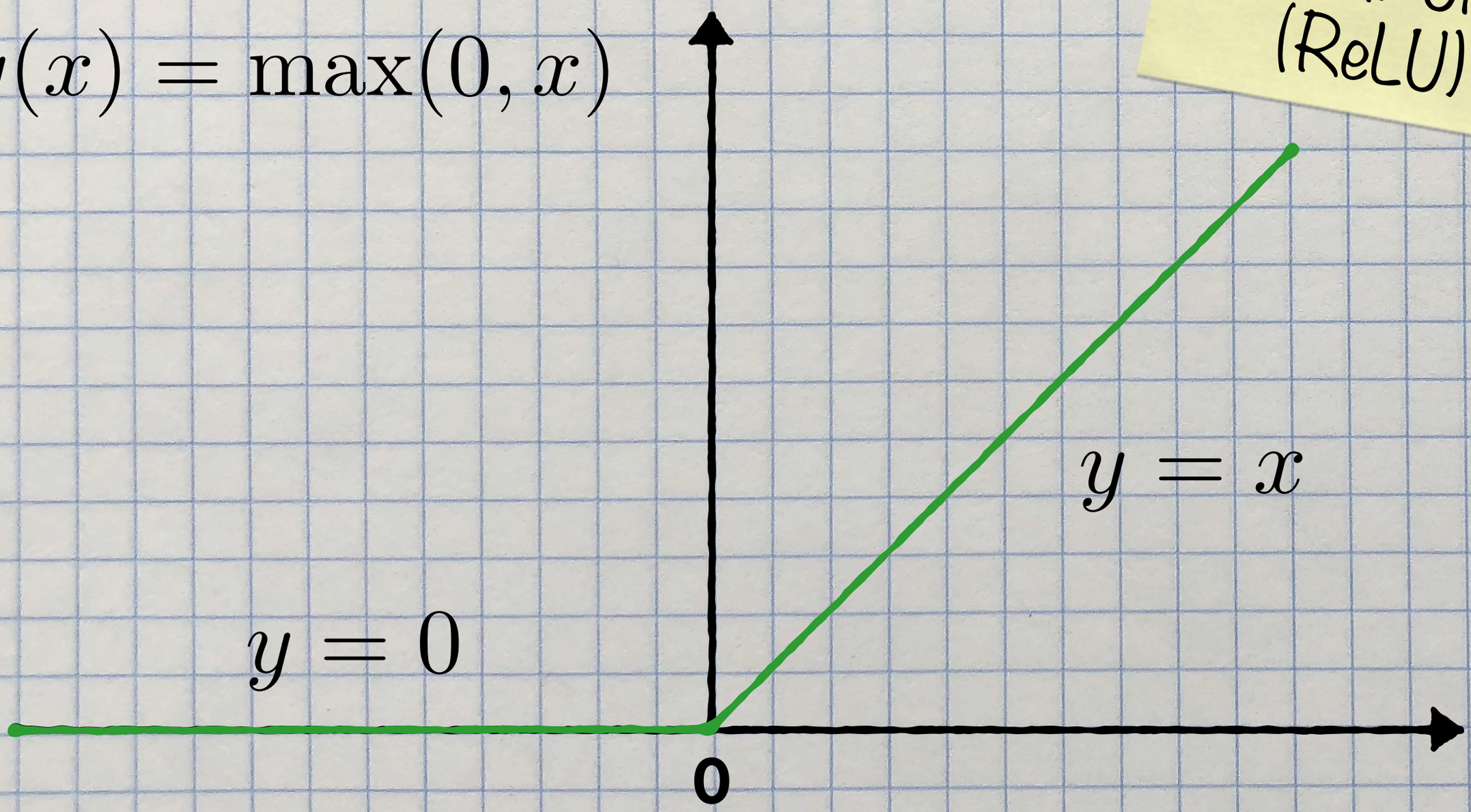
Rectified Linear Unit (ReLU)

$$y(x) = \max(0, x)$$



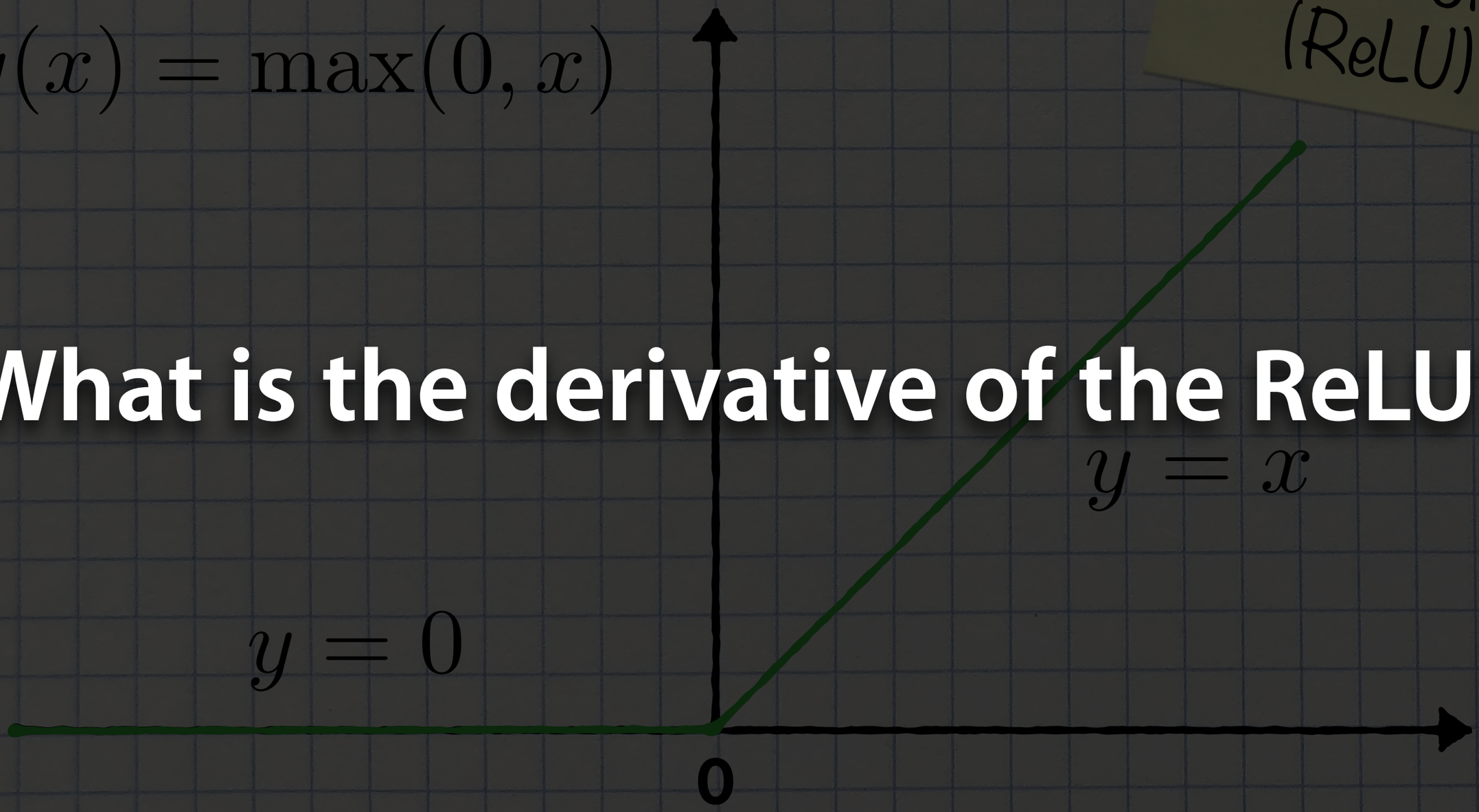
Rectified Linear Unit (ReLU)

$$y(x) = \max(0, x)$$



Rectified Linear Unit (ReLU)

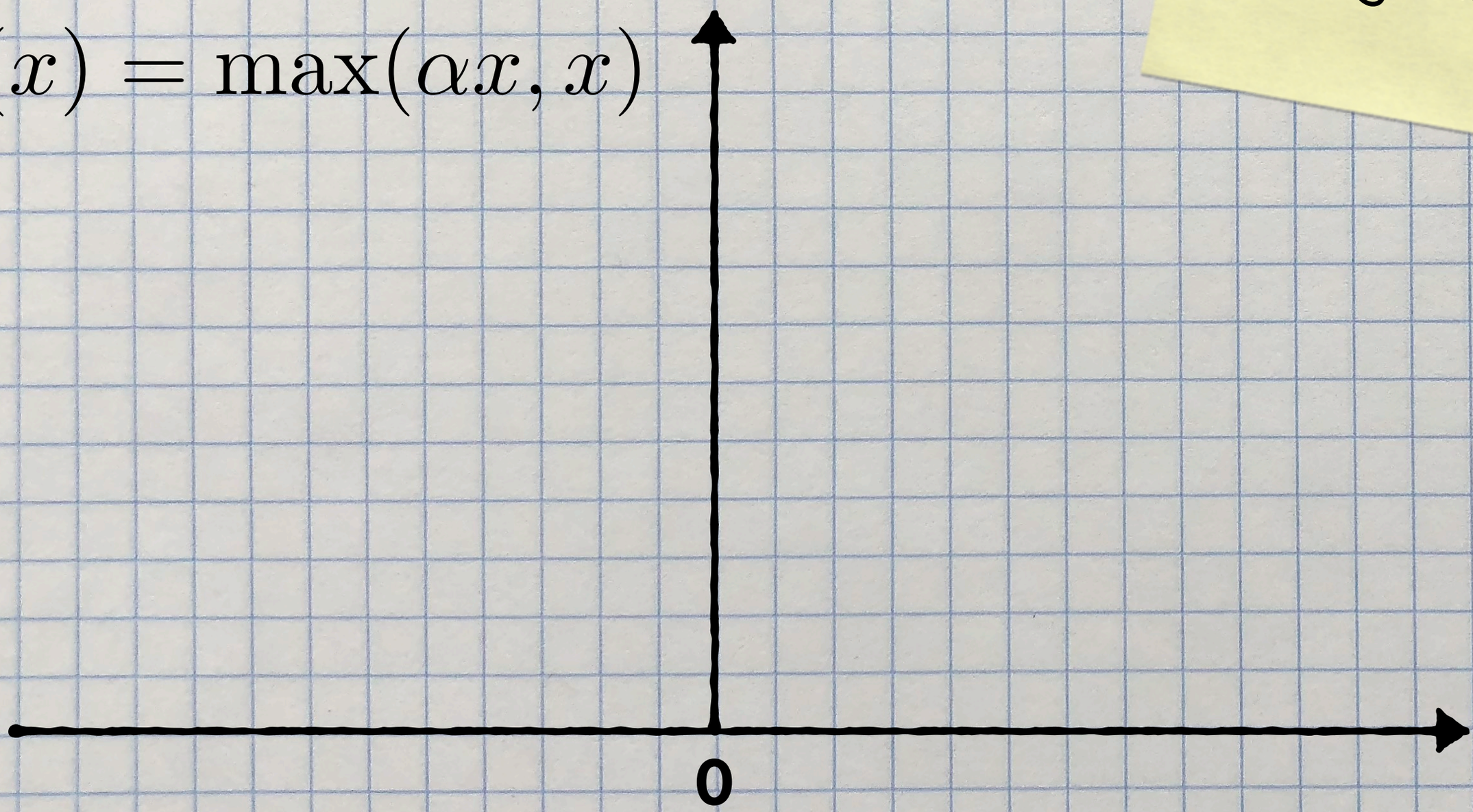
$y(x) = \max(0, x)$



What is the derivative of the ReLU?

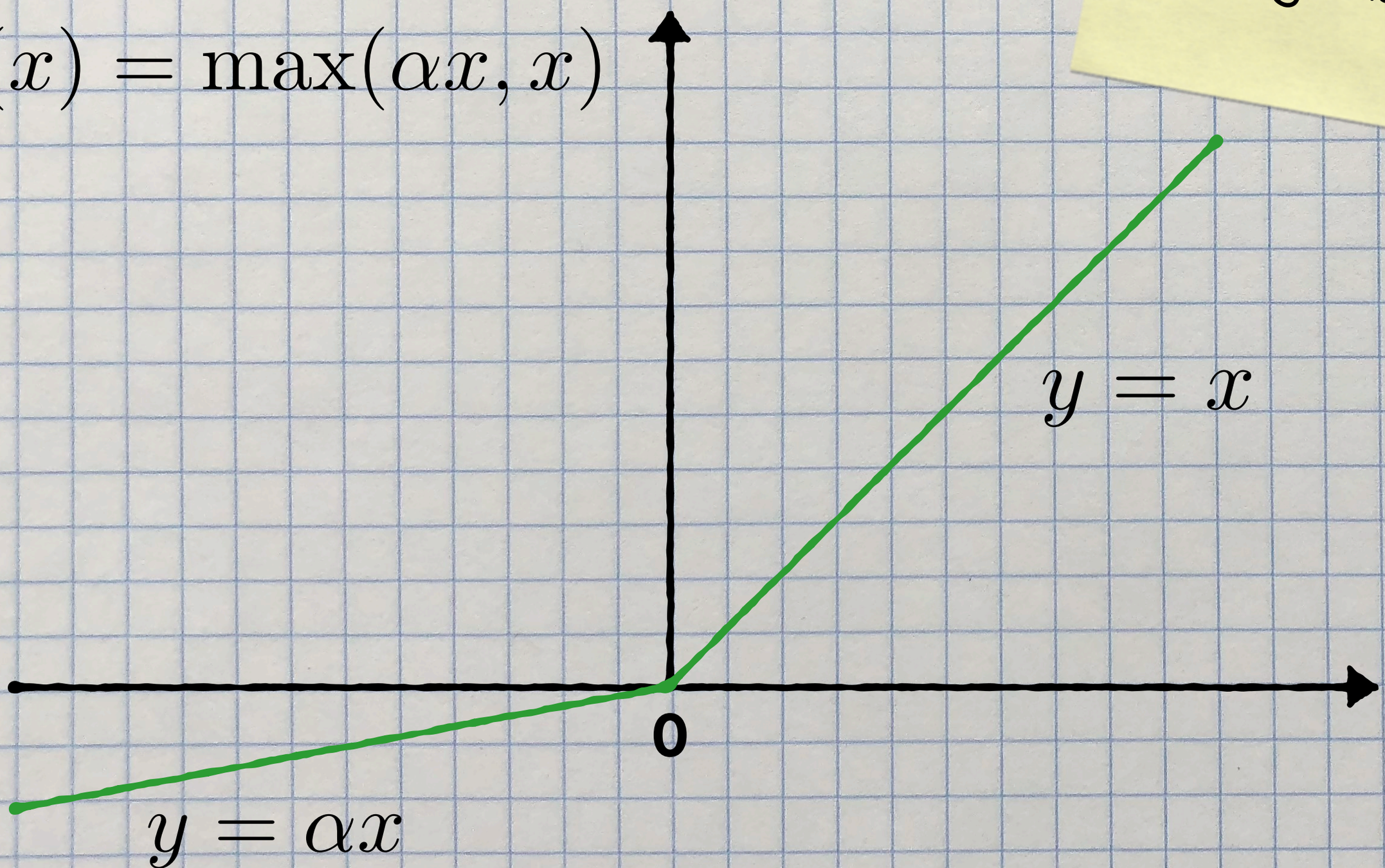
Leaky-ReLU

$$y(x) = \max(\alpha x, x)$$



Leaky-ReLU

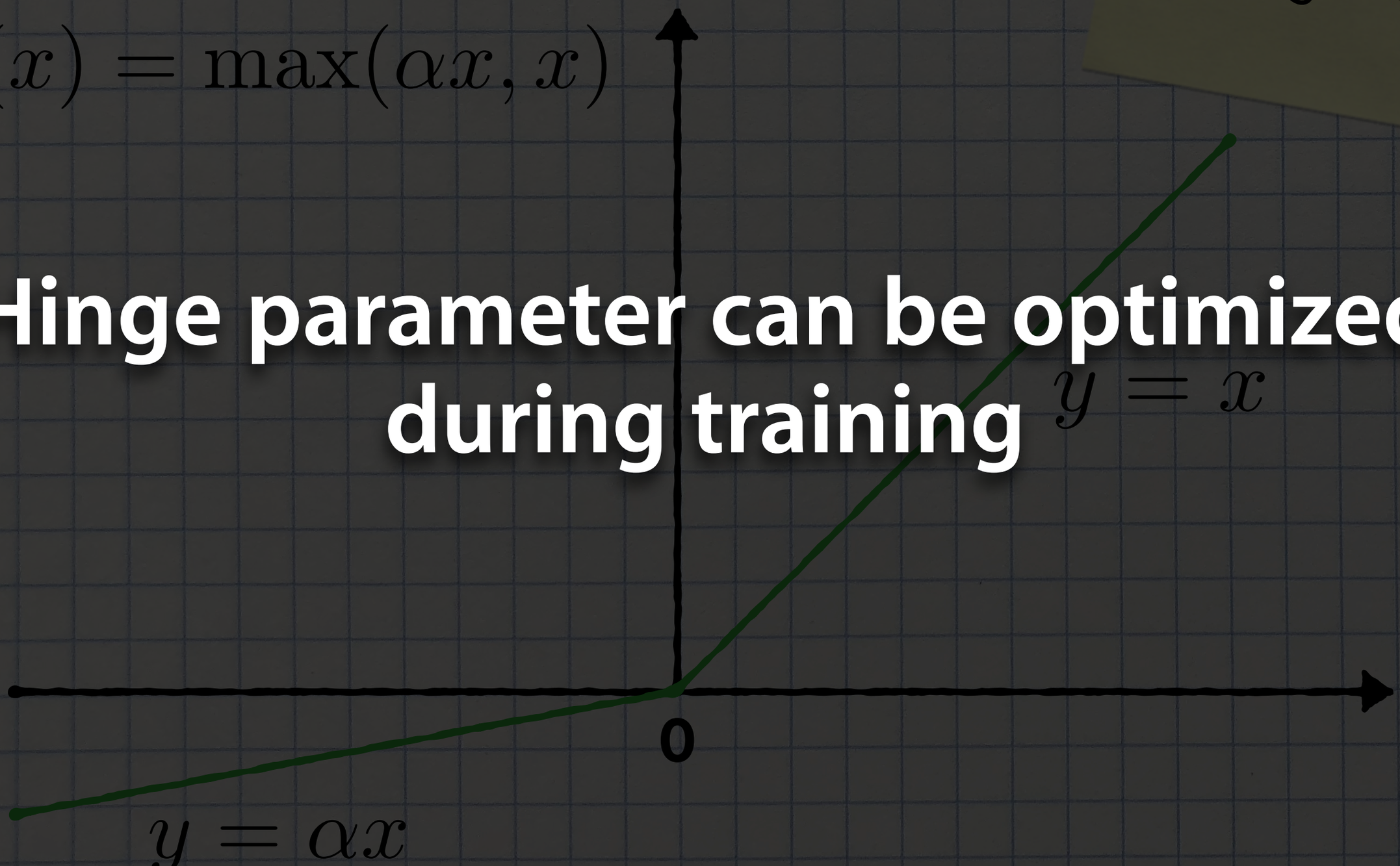
$$y(x) = \max(\alpha x, x)$$



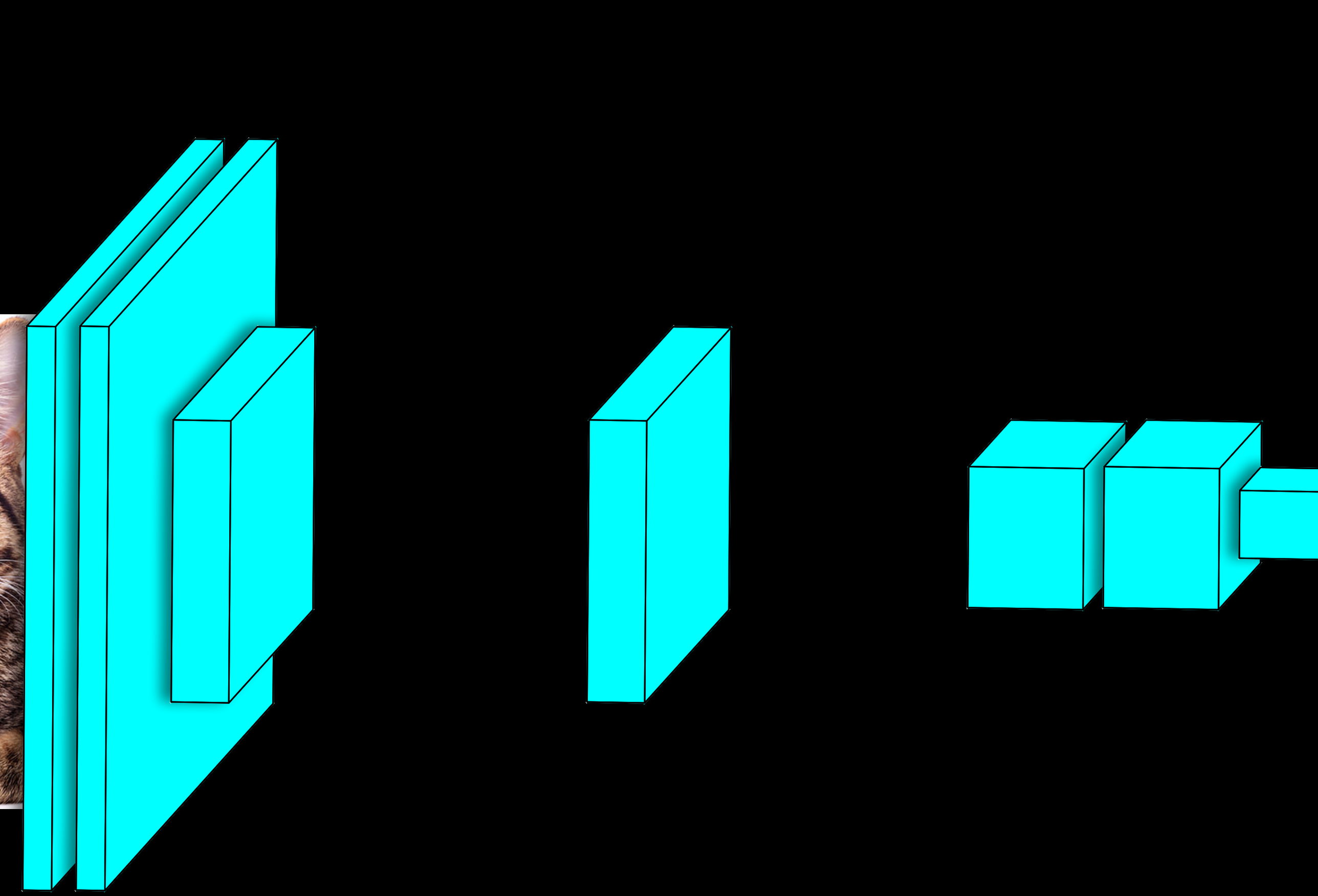
Leaky-ReLU

$$y(x) = \max(\alpha x, x)$$

Hinge parameter can be optimized during training



cat



x

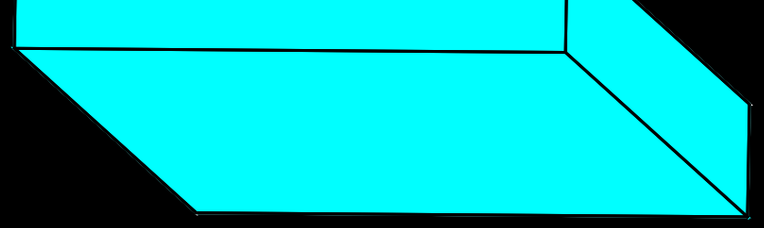
y

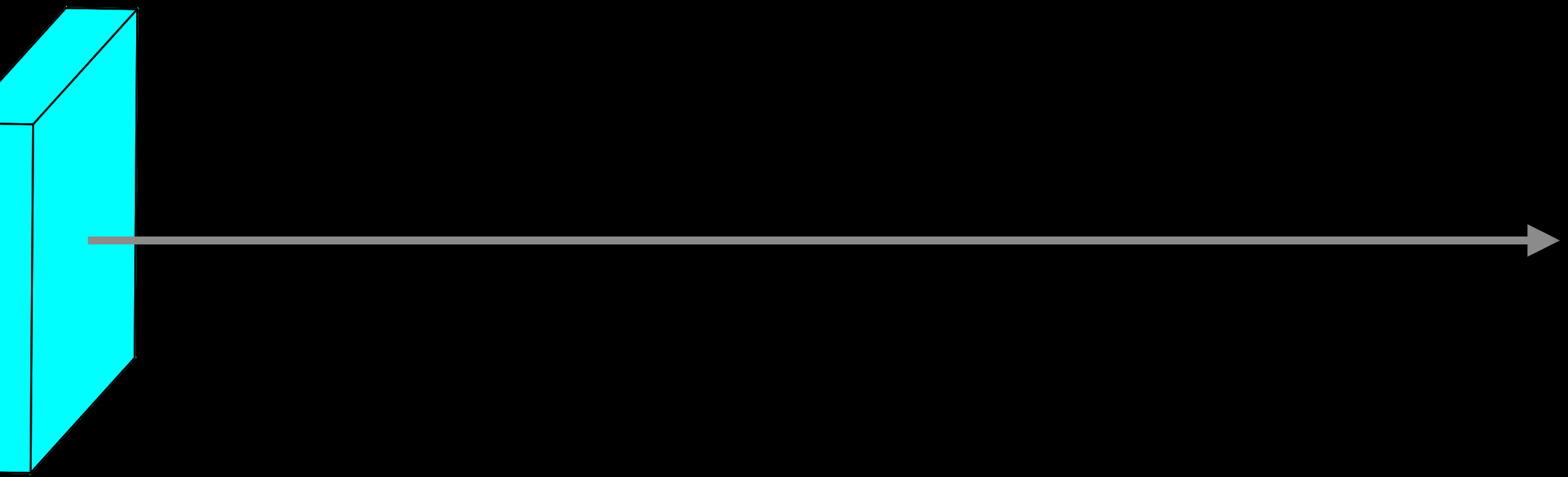
x

y

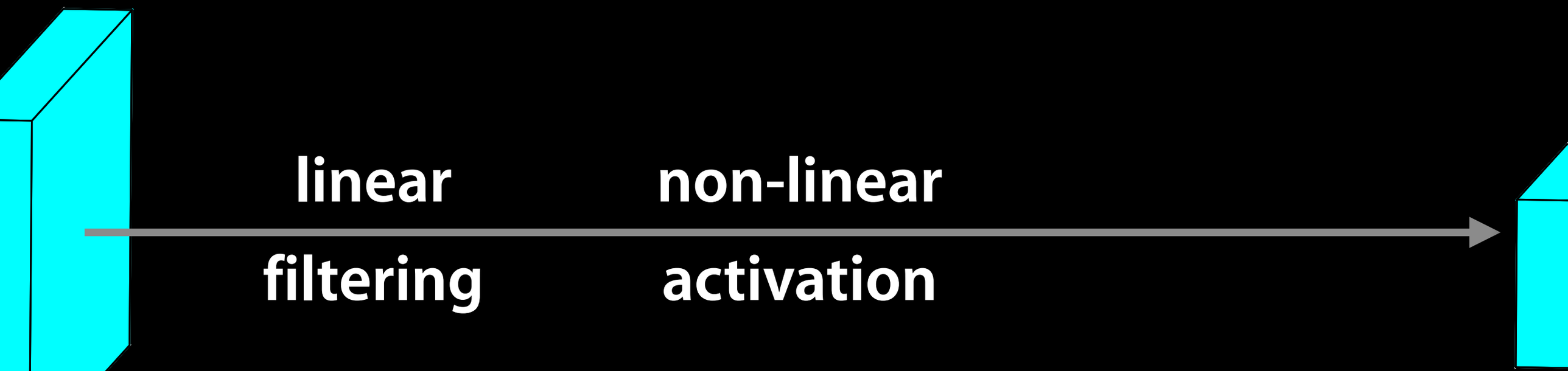
x

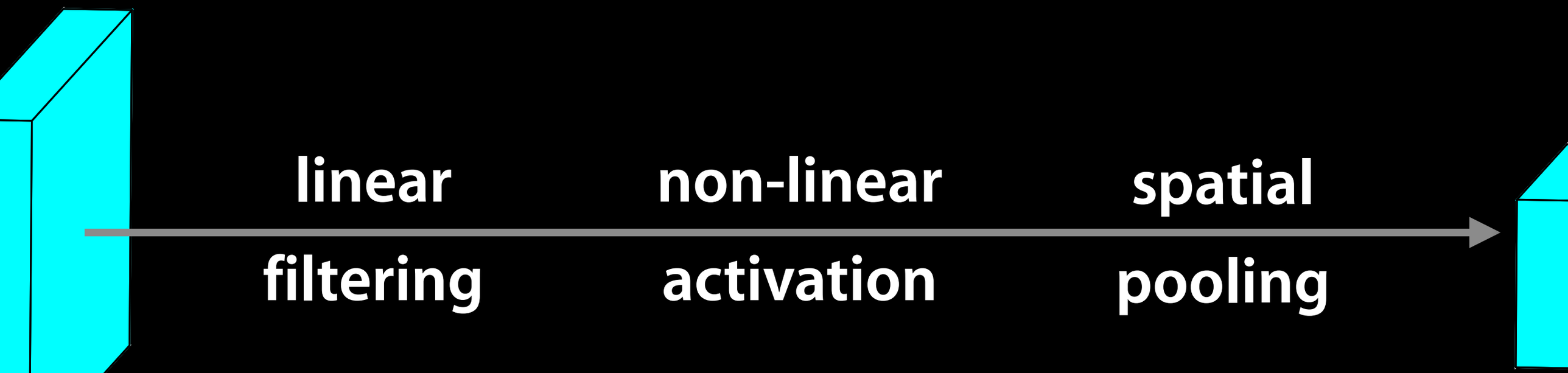
channels






linear
filtering


linear
non-linear
filtering
activation


linear
non-linear
spatial
filtering
activation
pooling

cat

receptive field

spatial region in the original image that can impact the output of a unit in a particular layer


input image

Layer 1

Layer 2

Layer 3

Layer 8

Understanding the Effective Receptive Field in Deep Convolutional Neural Networks

Wenjie Luo* **Yujia Li*** **Raquel Urtasun** **Richard Zemel**
Department of Computer Science
University of Toronto
{wenjie, yujiali, urtasun, zemel}@cs.toronto.edu

Abstract

We study characteristics of receptive fields of units in deep convolutional networks. The receptive field size is a crucial issue in many visual tasks, as the output must respond to large enough areas in the image to capture information about large objects. We introduce the notion of an effective receptive field, and show that it both has a Gaussian distribution and only occupies a fraction of the full theoretical receptive field. We analyze the effective receptive field in several architecture designs, and the effect of nonlinear activations, dropout, sub-sampling and skip connections on it. This leads to suggestions for ways to address its tendency to be too small.

1 Introduction

Deep convolutional neural networks (CNNs) have achieved great success in a wide range of problems in the last few years. In this paper we focus on their application to computer vision: where they are the driving force behind the significant improvement of the state-of-the-art for many tasks recently, including image recognition [10, 8], object detection [17, 2], semantic segmentation [12, 1], image captioning [20], and many more.

One of the basic concepts in deep CNNs is the *receptive field*, or *field of view*, of a unit in a certain layer in the network. Unlike in fully connected networks, where the value of each unit depends on the entire input to the network, a unit in convolutional networks only depends on a region of the input.

ConvNet History

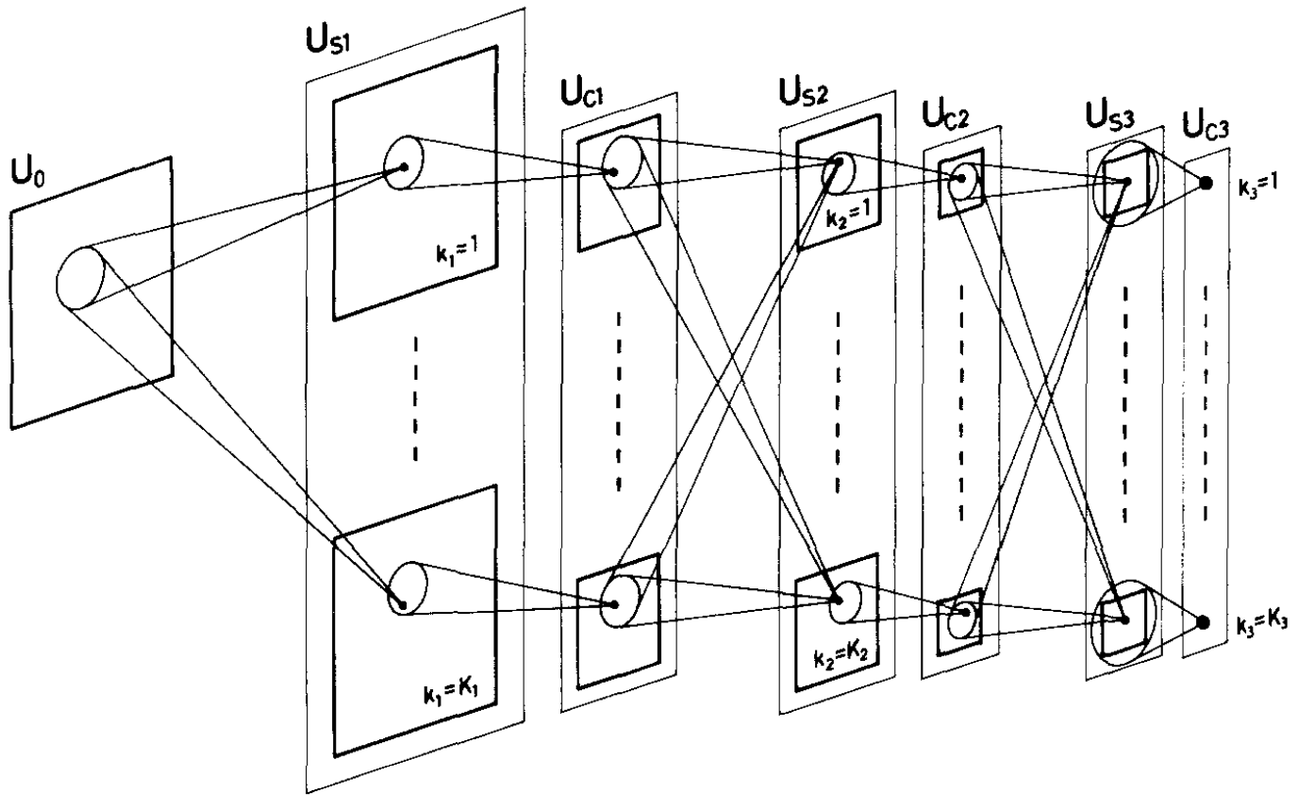


Fig. 2. Schematic diagram illustrating the interconnections between layers in the neocognitron

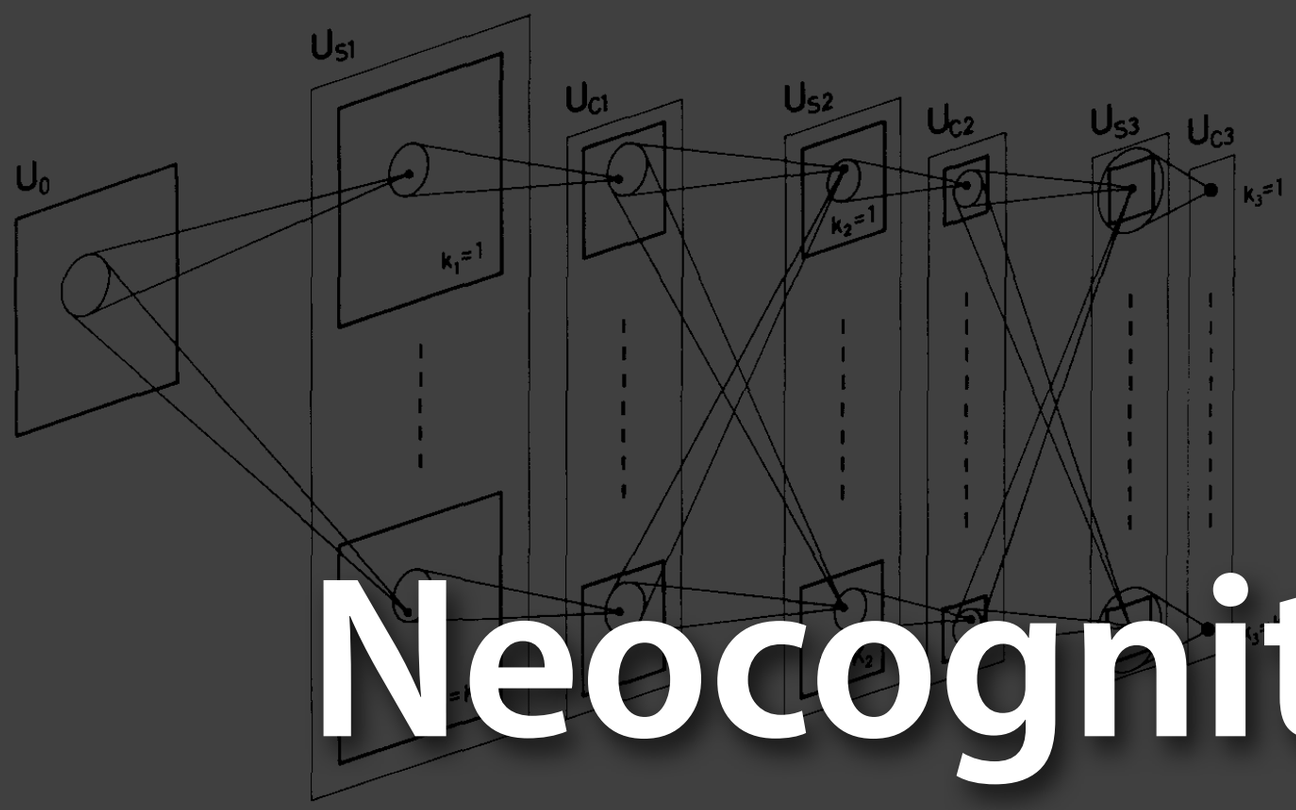
shifted in parallel from cell to cell. Hence, all the cells in a single cell-plane have receptive fields of the same function, but at different positions.

We will use notations $u_{Sl}(k_l, \mathbf{n})$ to represent the output of an S-cell in the k_l-th S-plane in the l-th

Since the cells in the network are interconnected in a cascade as shown in Fig. 2, the deeper the layer is, the larger becomes the receptive field of each cell of that layer. The density of the cells in each cell-plane is so determined as to decrease in accordance with the

Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position (1980)

Kunihiko Fukushima



Neocognitron

Fig. Schematic diagram illustrating the interconnections between layers in the neocognitron

shifted in parallel from cell to cell. Hence, all the cells in a single cell-plane have receptive fields of the same function, but at different positions.

We will use notations $u_{Sl}(k_l, \mathbf{n})$ to represent the output of an S-cell in the k_l-th S-plane in the l-th

Since the cells in the network are interconnected in a cascade as shown in Fig. 2, the deeper the layer is, the larger becomes the receptive field of each cell of that layer. The density of the cells in each cell-plane is so determined as to decrease in accordance with the

Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position (1980)

Kunihiko Fukushima

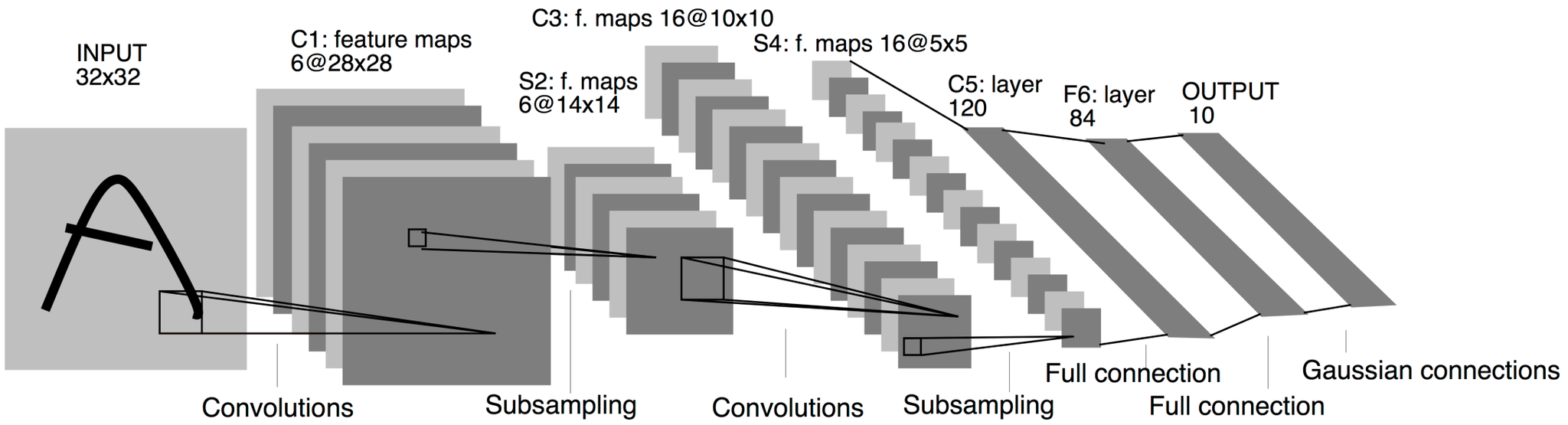


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

as the feature maps in the previous layer. The trainable coefficient and bias control the effect of the sigmoid non-linearity. If the coefficient is small, then the unit operates

B. LeNet-5

This section describes in more detail the architecture of

Gradient-based Learning Applied to Document Recognition (1998)

Yann Lecun, Leon Bottou, Yoshua Bengio and Patrick Haffner

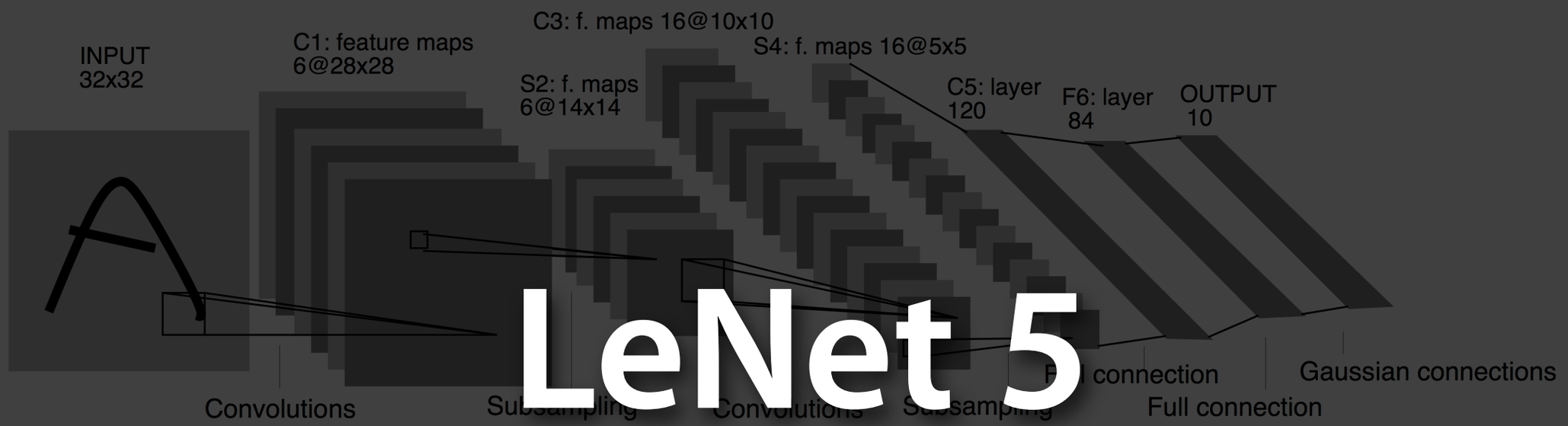


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

as the feature maps in the previous layer. The trainable coefficient and bias control the effect of the sigmoid non-linearity. If the coefficient is small, then the unit operates

B. LeNet-5

This section describes in more detail the architecture of

Gradient-based Learning Applied to Document Recognition (1998)

Yann Lecun, Leon Bottou, Yoshua Bengio and Patrick Haffner

United States Patent [19]

Denker et al.

[11] **Patent Number: 5,058,179**

[45] **Date of Patent: Oct. 15, 1991**

[54] **HIERARCHICAL CONSTRAINED AUTOMATIC LEARNING NETWORK FOR CHARACTER RECOGNITION**

[75] Inventors: **John S. Denker**, Leonardo; **Richard E. Howard**, Highland Park; **Lawrence D. Jackel**, Holmdel; **Yann LeCun**, Middletown, all of N.J.

[73] Assignee: **AT&T Bell Laboratories**, Murray Hill, N.J.

[21] Appl. No.: **472,991**

[22] Filed: **Jan. 31, 1990**

[51] **Int. Cl.**[5] **G06K 9/66**

[52] **U.S. Cl.** **382/14**; 382/49

[58] **Field of Search** 382/14, 10, 49; 364/513, 274.9, 276.6

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,275,986	9/1966	Dunn et al.	340/146.3
3,295,103	12/1966	Driese et al.	340/146.3
3,701,095	10/1972	Yamaguchi et al.	340/146.3 MA
4,750,211	6/1988	Wray	382/41
4,918,742	4/1990	Simonds	382/41
4,933,872	6/1990	Vandenberg et al.	364/513

OTHER PUBLICATIONS

D. E. Rumelhart et al., Parallel Distr. Proc.: Explorations in Microstructure of Cognition, vol. 1, 1986, "Learning Internal Representations by Error Propagation", pp. 318–362.

R. P. Lippmann, IEEE ASSP Magazine, vol. 4, No. 2, Apr. 1987, "An Introduction to Computing with Neural Nets", pp. 4–22.

Y. LeCun, Connectionism in Perspective, R. Pfeifer, Z. Schreter, F. Fogelman, L. Steels (Eds.), 1989, "Generalization and Network Design Strategies", pp. 143–155.

Y. LeCun et al., IEEE Comm. Magazine, Nov. 1989, "Handwritten Digit Recognition: Applications of Neural . . . ", pp. 41–46.

Primary Examiner—Leo H. Boudreau

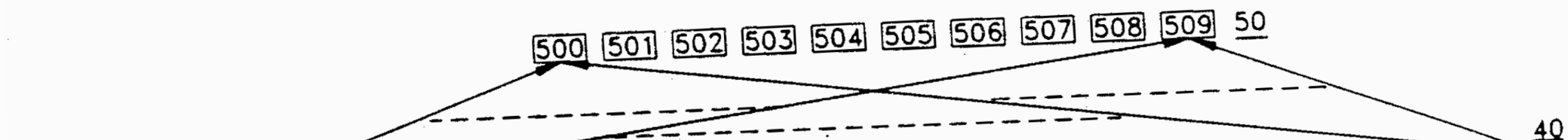
Assistant Examiner—Andrew W. Johns

Attorney, Agent, or Firm—Gregory C. Ranieri

[57] **ABSTRACT**

Highly accurate, reliable optical character recognition is afforded by a hierarchically layered network having several layers of parallel constrained feature detection for localized feature extraction followed by several fully connected layers for dimensionality reduction. Character classification is also performed in the ultimate fully connected layer. Each layer of parallel constrained feature detection comprises a plurality of constrained feature maps and a corresponding plurality of kernels wherein a predetermined kernel is directly related to a single constrained feature map. Undersampling is performed from layer to layer.

14 Claims, 15 Drawing Sheets



United States Patent [19]

Denker et al.

[11] **Patent Number: 5,058,179**

[45] **Date of Patent: Oct. 15, 1991**

[54] **HIERARCHICAL CONSTRAINED AUTOMATIC LEARNING NETWORK FOR CHARACTER RECOGNITION**

[75] Inventors: **John S. Denker,** Leonardo; **Richard E. Howard,** Highland Park; **Lawrence D. Jackel,** Holmdel; **Yann LeCun,** Middletown, all of N.J.

[73] **...Bell Laboratories,** Murray

[21]

[22]

[5

[5

[5

[

"Learning Internal Representations by Error Propagation", pp. 318–362.

R. P. Lippmann, IEEE ASSP Magazine, vol. 4, No. 2, Apr. 1987, "An Introduction to Computing with Neural Nets", pp. 4–22.

Y. LeCun, Connectionism in Perspective, R. Pfeifer, Z. Schreter, F. Fogelman, L. Steels (Eds.), 1989, "Generalization and Network Design Strategies", pp. 143–155.

Y. LeCun et al., IEEE Comm. Magazine, Nov. 1989, "Handwritten Digit Recognition: Applications of Neu-

...41, 46.

United States Patent [19]

Denker et al.

[11] **Patent Number: 5,067,164**

[45] **Date of Patent: Nov. 19, 1991**

[54] **HIERARCHICAL CONSTRAINED AUTOMATIC LEARNING NEURAL NETWORK FOR CHARACTER RECOGNITION**

[75] Inventors: **John S. Denker,** Leonardo; **Richard E. Howard,** Highland Park; **Lawrence D. Jackel,** Holmdel; **Yann LeCun,** Middletown, all of N.J.

[73] Assignee: **AT&T Bell Laboratories,** Murray Hill, N.J.

[21] Appl. No.: **444,455**

[22] Filed: **Nov. 30, 1989**

[51] **Int. Cl.**[5] **G06K 9/62**

[52] **U.S. Cl.** **382/15;** 364/276.6

[58] **Field of Search** 382/14, 15; 364/513, 364/274.9, 276.6, 972.4

Schreter, F. Fogelman, L. Steels, (Eds.), 1989, "Generalization and Network Design Strategies", pp. 143–55.

Y. LeCun et al., IEEE Comm. Magazine, Nov. 1989, "Handwritten Digit Recognition: Applications of Neural . . . ", pp. 41–46.

Primary Examiner—David K. Moore

Assistant Examiner—Michael Cammarata

Attorney, Agent, or Firm—Gregory C. Ranieri

[57] **ABSTRACT**

Highly accurate, reliable optical character recognition is afforded by a layered network having several layers of constrained feature detection wherein each layer of constrained feature detection includes a plurality of constrained feature maps and a corresponding plurality of feature reduction maps. Each feature reduction map is connected to only one constrained feature map in the same layer for undersampling that constrained feature-

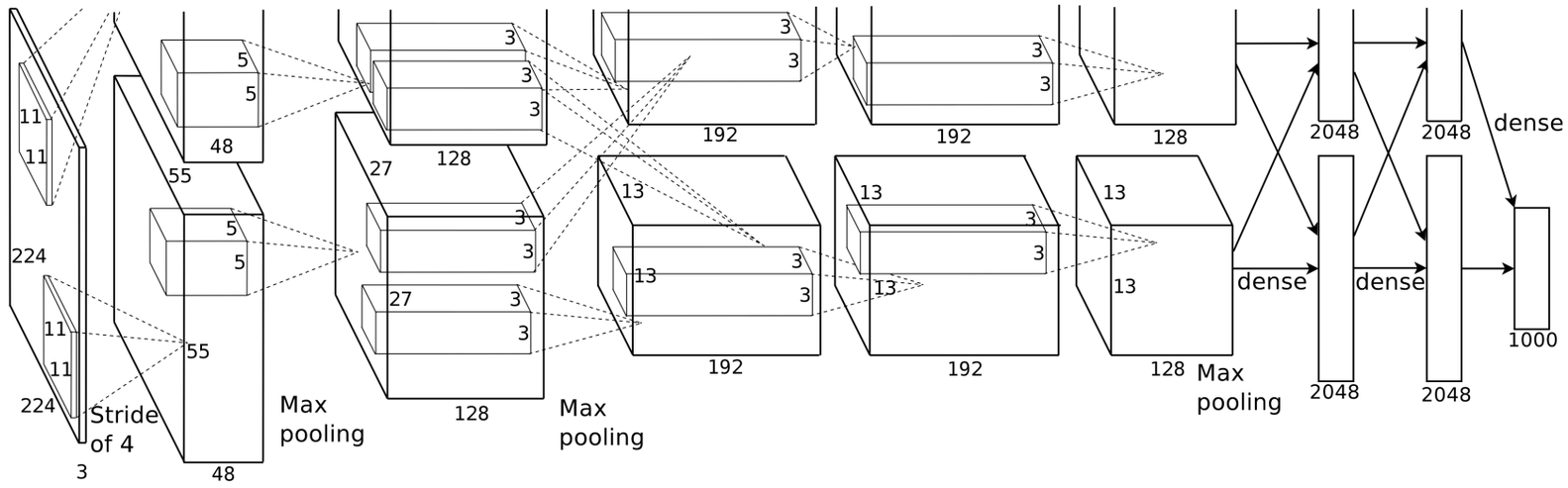
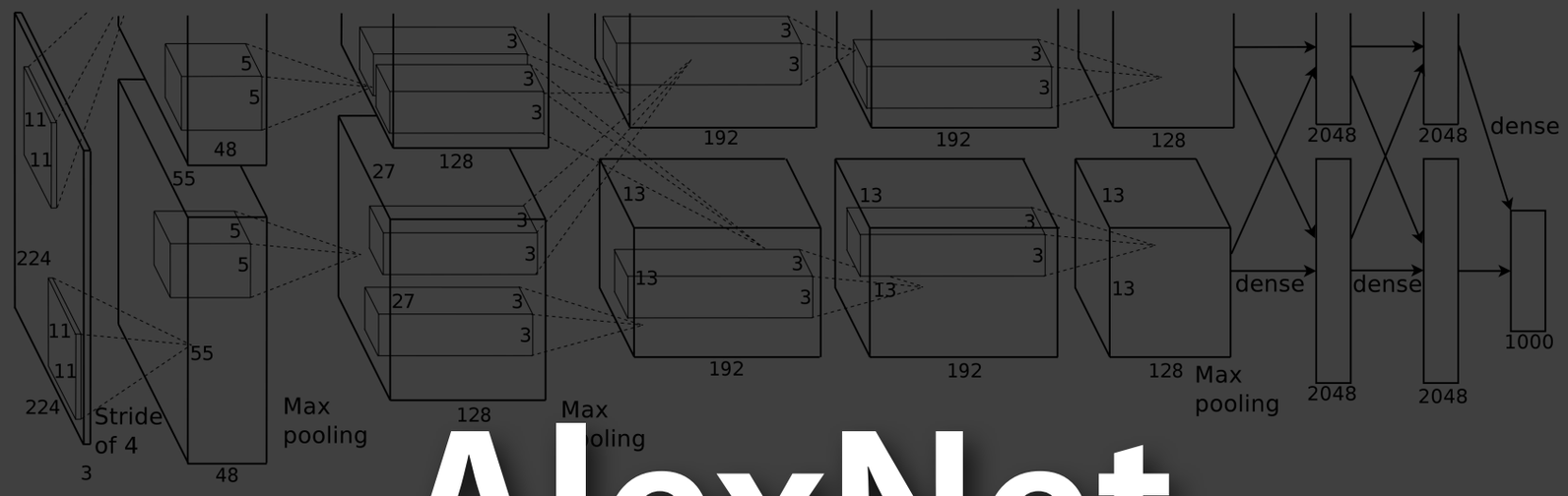


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

neurons in a kernel map). The second convolutional layer takes as input the (response-normalized and pooled) output of the first convolutional layer and filters it with 256 kernels of size $5 \times 5 \times 48$. The third, fourth, and fifth convolutional layers are connected to one another without any intervening pooling or normalization layers. The third convolutional layer has 384 kernels of size $3 \times 3 \times 256$ connected to the (normalized, pooled) outputs of the second convolutional layer. The fourth convolutional layer has 384 kernels of size $3 \times 3 \times 192$, and the fifth convolutional layer has 256 kernels of size $3 \times 3 \times 192$. The fully-connected layers have 4096 neurons each.

ImageNet Classification with Deep Convolutional Neural Networks (2013)

Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton



AlexNet

Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

neurons in a kernel map). The second convolutional layer takes as input the (response-normalized and pooled) output of the first convolutional layer and filters it with 256 kernels of size $5 \times 5 \times 48$. The third, fourth, and fifth convolutional layers are connected to one another without any intervening pooling or normalization layers. The third convolutional layer has 384 kernels of size $3 \times 3 \times 256$ connected to the (normalized, pooled) outputs of the second convolutional layer. The fourth convolutional layer has 384 kernels of size $3 \times 3 \times 192$, and the fifth convolutional layer has 256 kernels of size $3 \times 3 \times 192$. The fully-connected layers have 4096 neurons each.

ImageNet Classification with Deep Convolutional Neural Networks (2013)

Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton

A ConvNet for the 2020s

Zhuang Liu[1,2*] Hanzi Mao[1] Chao-Yuan Wu[1] Christoph Feichtenhofer[1] Trevor Darrell[2] Saining Xie[1†]

[1]Facebook AI Research (FAIR) [2]UC Berkeley

Code: https://github.com/facebookresearch/ConvNeXt

Abstract

The "Roaring 20s" of visual recognition began with the introduction of Vision Transformers (ViTs), which quickly superseded ConvNets as the state-of-the-art image classification model. A vanilla ViT, on the other hand, faces difficulties when applied to general computer vision tasks such as object detection and semantic segmentation. It is the hierarchical Transformers (e.g., Swin Transformers) that reintroduced several ConvNet priors, making Transformers practically viable as a generic vision backbone and demonstrating remarkable performance on a wide variety of vision tasks. However, the effectiveness of such hybrid approaches is still largely credited to the intrinsic superiority of Transformers, rather than the inherent inductive biases of convolutions. In this work, we reexamine the design spaces and test the limits of what a pure ConvNet can achieve. We gradually "modernize" a standard ResNet toward the design of a vision Transformer, and discover several key components that contribute to the performance difference along the way. The outcome of this exploration is a family of pure ConvNet models dubbed ConvNeXt. Constructed entirely from standard ConvNet modules, ConvNeXts compete favorably with Transformers in terms of accuracy and scalability, achieving 87.8% ImageNet top-1 accuracy and outperforming Swin Transformers on COCO

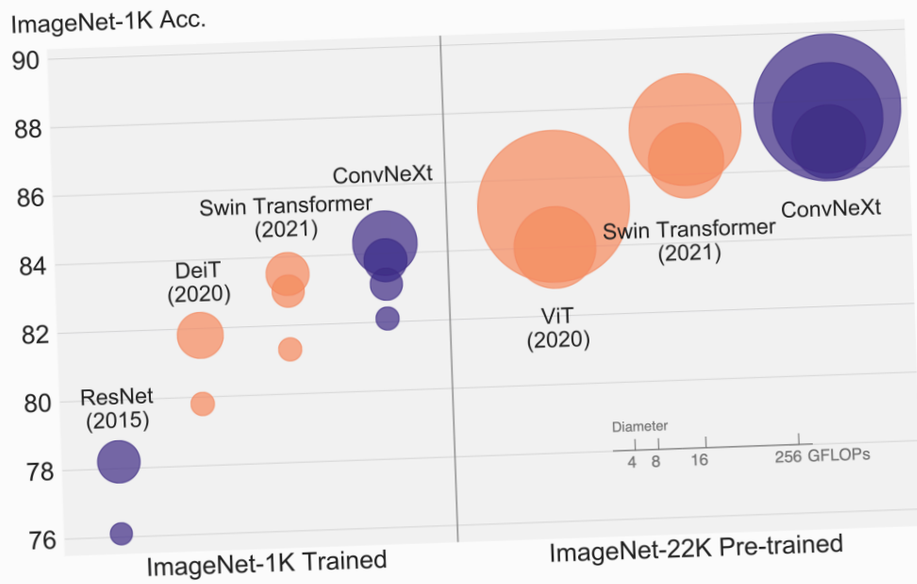


Figure 1. **ImageNet-1K classification** results for • ConvNets and ○ vision Transformers. Each bubble's area is proportional to FLOPs of a variant in a model family. ImageNet-1K/22K models here take 224^2/384^2 images respectively. We demonstrate that a standard ConvNet model can achieve the same level of scalability as hierarchical vision Transformers while being much simpler in design.

visual feature learning. The introduction of AlexNet [37] precipitated the "ImageNet moment" [56], ushering in a new era of computer vision. The field has since evolved at a rapid speed. Representative ConvNets like VGGNet [61], Inceptions [64], ResNe(X)t [26, 82], DenseNet [33], MobileNet [32], EfficientNet [67] and RegNet [51] focused on

github.com/facebookresearch/ConvNeXt

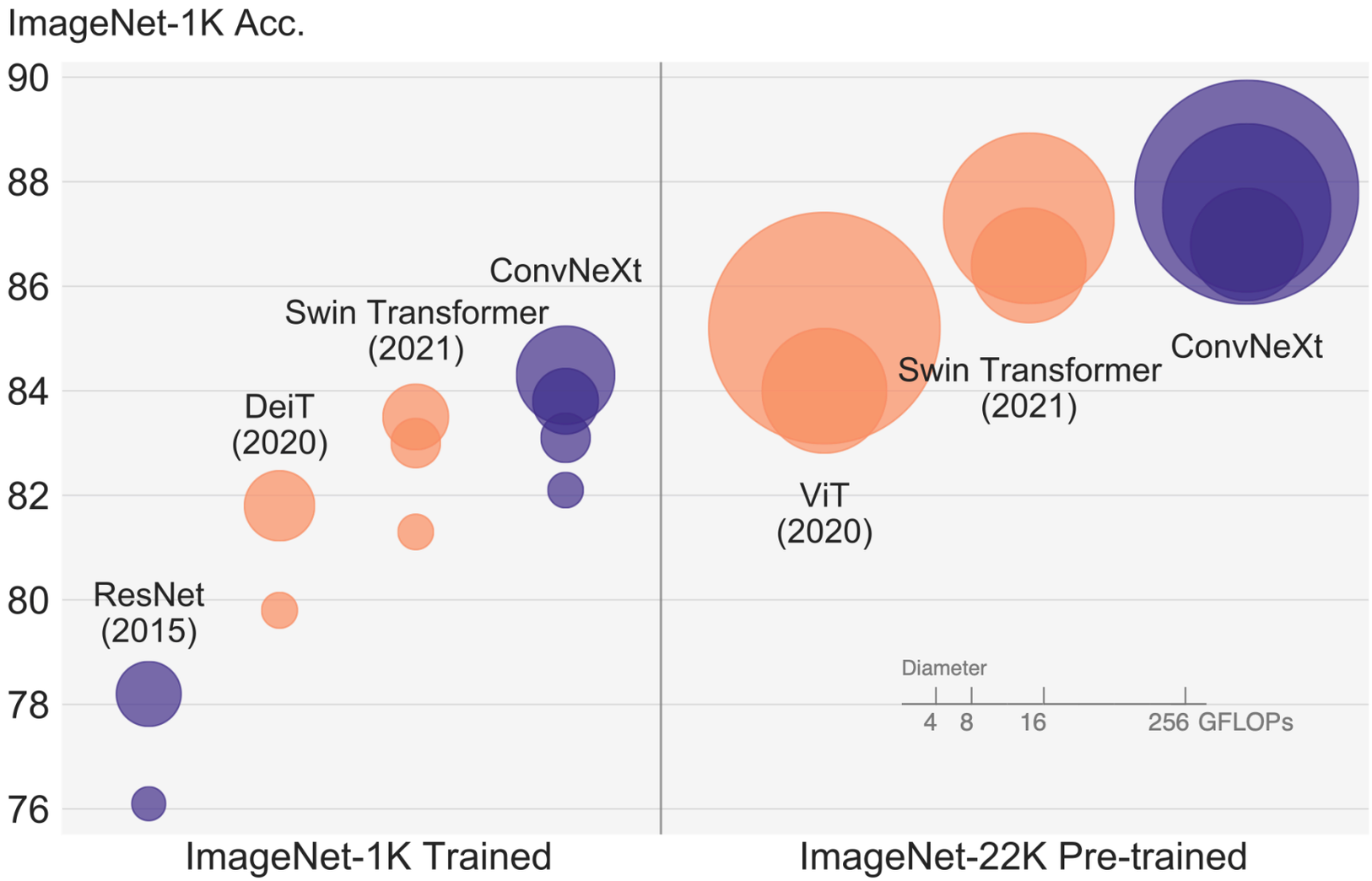


Figure 1. **ImageNet-1K classification** results for • ConvNets and ○ vision Transformers. Each bubble's area is proportional to FLOPs of a variant in a model family. ImageNet-1K/22K models here take 224^2/384^2 images respectively. We demonstrate that a standard ConvNet model can achieve the same level of scalability as hierarchical vision Transformers while being much simpler in design.

gan with the hich quickly ge classifica- s difficulties ch as object hierarchical roduced sev- ically viable remarkable s. However, still largely ners, rather ons. In this the limits of "modernize" Transformer, ribute to the come of this dubbed Con-

visual feature learning. The introduction of AlexNet [37]