

King of Dumbs

Level: ♦ Intermediate

How does a computer turn a photo into numbers?

A photo is millions of numbers, and almost everything AI does with it is multiply and add. Vectors, the dot product and matrices at three levels, with two widgets.

English Published October 17, 2026



Scan to open the interactive lesson

<https://kingofdumbs.com/en/lessons/vectors-and-matrices>

What you'll learn

- A photo is a list of numbers: vectors and dimension
- The dot product: multiply and add, and how much two arrows point the same way
- A matrix moves the plane: its columns, the area and the dimensions that are left
- Multiplying matrices: chaining changes, why order matters and what a neural network does

Symbols you'll meet

E_a subscript: the label underneath (♠) · Σ sum: add them all up (♠) · x_i index i: item number i (♠) · $\approx$ approximately equal (♠) · $\leq \geq$ less or equal / greater or equal (♠) · $|x|$ absolute value: drop the sign (♠) · $\cdot$ dot: multiply (or dot product) (♠♠) · σ sigma: depends on the lesson (♠) · θ theta: an angle (almost always) (♠) · λ lambda: a stretching number (♠) · $\|x\|$ norm: the length of the arrow (♠) · x^T transpose: lay the column down (♠) · $\in$ belongs to (♠) · $\mathbb{R} \mathbb{Z}$ sets of numbers (♠)

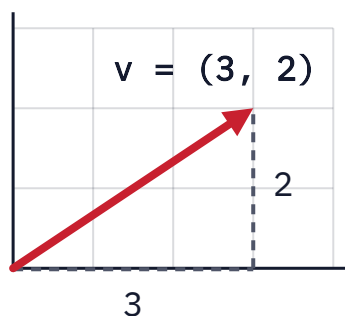
You take a photo of your cat and, if you search for “cat” in your phone’s gallery, it often finds it. But a computer doesn’t see cats: all it knows how to do is add and multiply numbers, very fast.

This lesson’s dumb question is: **how does a computer turn a photo into numbers?** And while we’re at it, what does it do with them? The answer fits into three ideas that sound like an exam, vector, dot product and matrix, and a single sum: multiply and add.

Pick your card: ♣ if you’re starting from zero, ♦ if a simple formula doesn’t scare you, ♠ if you want the notation. You can switch at any time.

♣ ♦ ♠

A photo is a list of numbers



$$\text{length } \sqrt{(3^2 + 2^2)} \approx 3.6$$

2 numbers: the plane

3 numbers: space

36 million: can't be drawn,
but the sums are the same

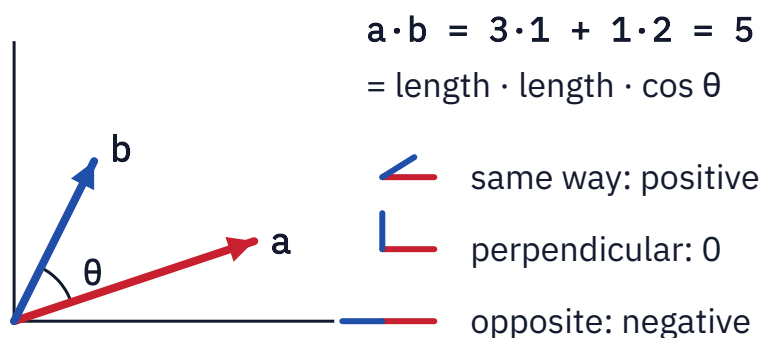
A **vector** is an ordered list of numbers, its **components**. With two components it can be drawn as an arrow from the centre: $v = (3, 2)$ means “3 to the right and 2 up”. Its **length** comes from Pythagoras: $\sqrt{(3^2 + 2^2)} = \sqrt{13} \approx 3.6$ (OpenStax).

A grey photo with two pixels would be an arrow like that, in a plane; with three pixels, an arrow in space. With more it can't be drawn any more, but the sums are the same: the length is still the square root of the sum of the squares. The **dimension** is how many numbers the vector has.

In a colour photo each pixel carries three numbers from 0 to 255, one per **channel** (red, green and blue), because each takes up 8 bits (Pillow, a Python imaging library). So an open photo is a table of **height × width × 3** numbers. The JPEG file takes up less space because it's compressed, but opening it unpacks it into that table. To hand it to a neural network, **it's all put in a row**: $4,000 \times 3,000 \times 3 = 36$ million numbers.



The dot product: multiply and add



The **dot product** of $a = (a_1, a_2)$ and $b = (b_1, b_2)$ is $a \cdot b = a_1 b_1 + a_2 b_2$: multiply the components one by one and add them up (OpenStax). With more components, the same. With $a = (3, 1)$ and $b = (1, 2)$: $3 \cdot 1 + 1 \cdot 2 = 5$. With two photos of 36 million numbers, that's **36 million multiplications**.

With arrows it has another reading: $a \cdot b = \text{length of } a \times \text{length of } b \times \text{cosine of the angle}$ between them (OpenStax). **The cosine is 1 if they point the same way, 0 if they make a right angle and -1 if they're opposite**. That's why two **perpendicular** arrows give 0.

If you divide the dot product by the two lengths, only the cosine is left, which measures how alike the directions are: it's the **cosine similarity**. With photos it has a nice property. If you multiply every number in a photo by 1.2, it looks lighter: **the arrow gets longer but doesn't turn**, and its similarity with the original is still 1 (our own calculation, as long as no number goes over 255).

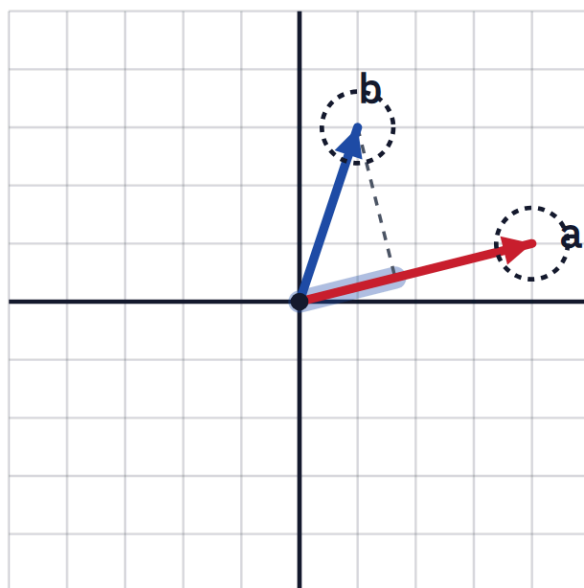
The King of Dumbs says:

Multiply and add. That's all there is. With that I compare prices, arrows and photos, and still have time left for a nap.



Try it: two arrows

Drag the tips of the arrows, or move them with the sliders or the arrow keys. Find a position where the dot product is 0: the arrows will end up at a right angle. The blue band is the shadow of b on a : it gets longer the more they point the same way.



a, across



4

a, up



1

b, across



1

b, up



3

Dot product $a \cdot b$

$$4 \cdot 1 + 1 \cdot 3 = 7$$

Lengths

$$\|a\| = 4.12 \quad \cdot \quad \|b\| = 3.16$$

Angle

$$57.5^\circ$$

Cosine (how alike the directions are) 0.54

$a = (4, 1)$, $b = (1, 3)$, $a \cdot b = 7$. They point the same way: the dot product is positive.



Interactive widget. Try it on the web: <https://kingofdumbs.com/en/lessons/vectors-and-matrices>

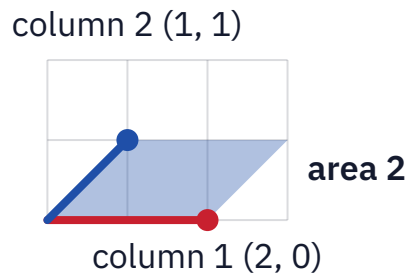


A matrix moves the plane

$$\begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

row 1: $2 \cdot 1 + 1 \cdot 2 = 4$

row 2: $0 \cdot 1 + 1 \cdot 2 = 2$



A **matrix** is a rectangular table of numbers; one with m rows and n columns is $m \times n$ (OpenStax).

Multiplying a matrix by a vector means taking the dot product of **each row** with the vector. With $A = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix}$ and $x = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$, the first row gives $2 \cdot 1 + 1 \cdot 2 = 4$ and the second, $0 \cdot 1 + 1 \cdot 2 = 2$, so $Ax = \begin{pmatrix} 4 \\ 2 \end{pmatrix}$.

There's a trick to reading it: **the first column is where $(1, 0)$ goes and the second, where $(0, 1)$ goes** (Austin). In the example, $(1, 0)$ goes to $(2, 0)$ and $(0, 1)$ goes to $(1, 1)$: the matrix stretches sideways and **slants**. With that you know where everything goes, because **any point is a mix of those two**.

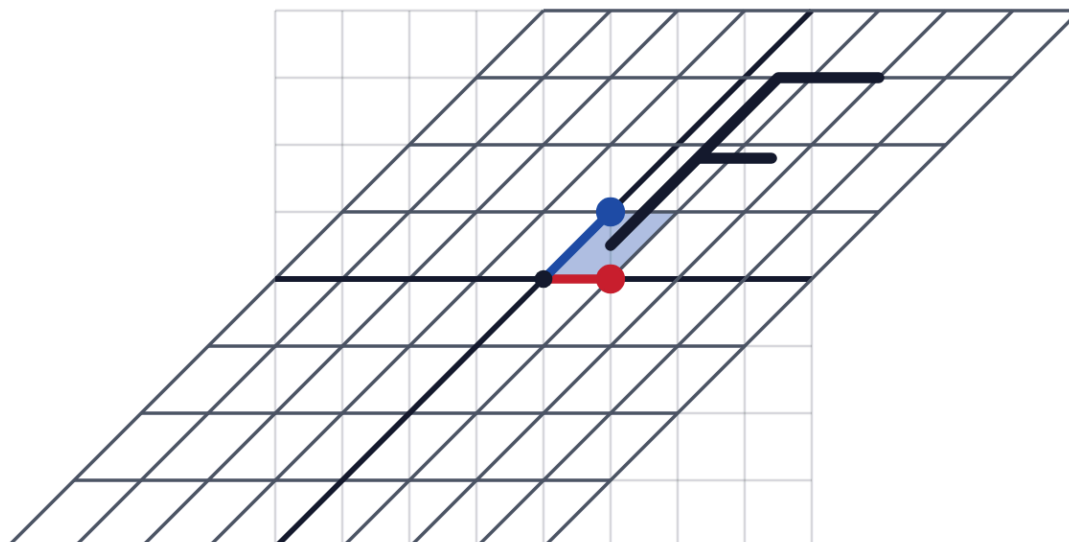
The little square of side 1 becomes a parallelogram. With the matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$, its area is **$ad - bc$** : in the example, $2 \cdot 1 - 1 \cdot 0 = 2$, double (Austin). If it's 0, the matrix **squashes** the whole plane onto a line: a dimension is lost and there's no way back. $\begin{pmatrix} 1 & -1 \\ 2 & -2 \end{pmatrix}$ does that. If it's negative, the drawing comes out **as in a mirror**.

Pillow uses it like this to **rotate** or slant a photo: for each new pixel, it works out with a matrix **which point of the original it comes from** (plus a shift).



Try it: bend the grid

Move the four numbers of the matrix or try the buttons. The red arrow is the first column, where $(1, 0)$ goes; the blue one, the second. Find a matrix that squashes the grid onto a line: the area will become 0. And see what happens to the F with the mirror.



Try one

Do nothing

Stretch

Turn 90°

Slant

Mirror

Squash

a (top left)



1

b (top right)



1

c (bottom left)



0

d (bottom right)



1

(1, 0) goes to

(1, 0)

(0, 1) goes to

(1, 1)

The area is multiplied by $(ad - bc)$

1

Dimensions left

2: the plane is still a plane

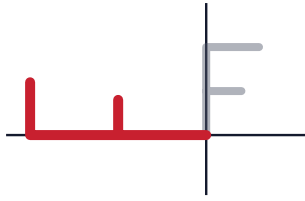
Matrix $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$: the area is multiplied by 1 and 2 dimensions are left.



Interactive widget. Try it on the web: <https://kingofdumbs.com/en/lessons/vectors-and-matrices>

Multiplying matrices: order matters

turn 90°, then stretch



$$S \cdot R = \begin{pmatrix} 0 & -2 \\ 1 & 0 \end{pmatrix}$$

stretch, then turn 90°



$$R \cdot S = \begin{pmatrix} 0 & -1 \\ 2 & 0 \end{pmatrix}$$

R: turn 90° · S: stretch ×2 sideways

To **multiply two matrices**, each number in the result is the dot product of **a row of the first with a column of the second** (OpenStax). It only works if the first has as many columns as the second has rows: **an $m \times k$ one times a $k \times n$ one gives an $m \times n$ one.**

Applying first B and then A to a point is the same as applying the matrix **AB** in one go. Careful: it's written the other way round from how it's done (Austin). And **AB usually isn't equal to BA** (OpenStax). With R, turning 90°, and S, stretching to double the width: turning and then stretching is $S \cdot R = \begin{pmatrix} 0 & -2 \\ 1 & 0 \end{pmatrix}$, and stretching and then turning is $R \cdot S = \begin{pmatrix} 0 & -1 \\ 2 & 0 \end{pmatrix}$.

A **layer** of a neural network multiplies its input by a matrix of **weights** (PyTorch). A grey photo of 100×100 is 10,000 numbers; a layer with 1,000 outputs uses a $1,000 \times 10,000$ matrix and does **10 million multiplications** per photo (our own calculation). Between one layer and the next there's **a step that isn't a multiplication**; the lesson "Logistic regression or neural network?" explains what it is and what a network gains from it.

Graphics cards take **lots of photos at once**, one per column, and that turns everything into a single enormous matrix multiplication (NVIDIA).

The King of Dumbs says:

I don't see cats: I see 36 million numbers. The hard part is knowing which matrices to multiply them by.



Final challenge

1. A colour photo of 100×100 pixels. How many numbers does the computer store when it opens it?

- a. 100
- b. 10,000, one per pixel
- c. 30,000: three per pixel, red, green and blue

2. Two arrows make a right angle. What's their dot product?

- a. 1
- b. 0
- c. It depends on how long they are

3. You rotate a photo by 90° and then stretch it sideways. Is that the same as stretching it first and rotating it afterwards?

- a. Yes: multiplying gives the same in any order
- b. No: with matrices, order matters
- c. Only if the photo is square



In three sentences

1. To a computer, a photo is a list of numbers (three per pixel, from 0 to 255): a vector, whose dimension is how many numbers it has.
2. The dot product multiplies two lists term by term and adds: it gives a total, as in the fruit seller's sum, and says how much two arrows point the same way (0 if they're perpendicular).
3. A matrix is a table that moves every point of the plane, and multiplying matrices chains those changes, with the order mattering: it's the sum a neural network does millions of

times with each photo.



Sources

- Vectors, length and the dot product: OpenStax, *Calculus Volume 3*, [2.1](#) and [2.3](#) (the fruit seller, theorems 2.4 and 2.5). Matrices and their product: OpenStax, *College Algebra 2e*, [7.5](#).
- Transformations, determinant and dimension: D. Austin, *Understanding Linear Algebra*, [2.2](#), [2.5](#), [2.6](#), [3.4](#) and [3.5](#).
- Pixels and image transforms: Pillow 13.0, [Concepts](#) and [ImageTransform](#). From photo to tensor: torchvision 0.29, [ToTensor](#).
- Neural networks: PyTorch 2.14, [nn.Linear](#) and [cosine_similarity](#). NVIDIA, [Matrix Multiplication Background](#) and [Linear/Fully-Connected Layers](#).

How this lesson was made: researched and written with help from Claude from the sources above, consulted on 17 October 2026. Figures without their own link are calculations from their definitions. The examples (the 4,000 × 3,000 photo, the two shops, the layer with 1,000 outputs) are made up. The drawings are our own.

Answers

1. **c)** 30,000: three per pixel, red, green and blue. That's it. $100 \times 100 \times 3$: a vector of dimension 30,000.
2. **b)** 0. Right. Perpendicular, dot product 0, however long they are.
3. **b)** No: with matrices, order matters. Exactly. Like the sock and the shoe.

A King of Dumbs lesson, kingofdumbs.com. Offline version; the original, with widgets you can play with, is at:
<https://kingofdumbs.com/en/lessons/vectors-and-matrices>

Updated on October 17, 2026.