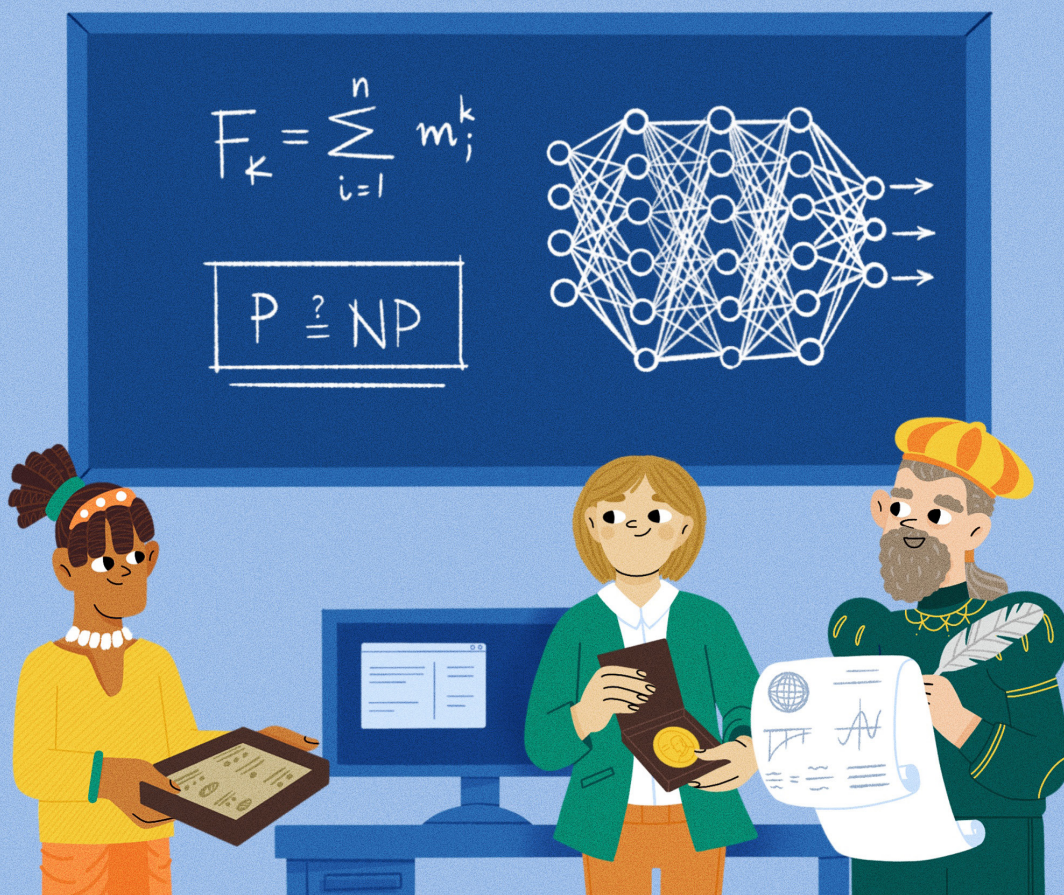


# Math That Changed the World

Edited by

Idan Segev, Jeremy Martin and Robert Knight



**FRONTIERS EBOOK COPYRIGHT STATEMENT**

The copyright in the text of individual articles in this ebook is the property of their respective authors or their respective institutions or funders. The copyright in graphics and images within each article may be subject to copyright of other parties. In both cases this is subject to a license granted to Frontiers.

The compilation of articles constituting this ebook is the property of Frontiers.

Each article within this ebook, and the ebook itself, are published under the most recent version of the Creative Commons CC-BY licence. The version current at the date of publication of this ebook is CC-BY 4.0. If the CC-BY licence is updated, the licence granted by Frontiers is automatically updated to the new version.

When exercising any right under the CC-BY licence, Frontiers must be attributed as the original publisher of the article or ebook, as applicable.

Authors have the responsibility of ensuring that any graphics or other materials which are the property of others may be included in the CC-BY licence, but this should be checked before relying on the CC-BY licence to reproduce those materials. Any copyright notices relating to those materials must be complied with.

Copyright and source acknowledgement notices may not be removed and must be displayed in any copy, derivative work or partial copy which includes the elements in question.

All copyright, and all rights therein, are protected by national and international copyright laws. The above represents a summary only. For further information please read Frontiers' Conditions for Website Use and Copyright Statement, and the applicable CC-BY licence.

ISSN 2296-6846  
ISBN 978-2-8325-4952-0  
DOI 10.3389/978-2-8325-4952-0

**About Frontiers**

Frontiers is more than just an open-access publisher of scholarly articles: it is a pioneering approach to the world of academia, radically improving the way scholarly research is managed. The grand vision of Frontiers is a world where all people have an equal opportunity to seek, share and generate knowledge. Frontiers provides immediate and permanent online open access to all its publications, but this alone is not enough to realize our grand goals.

**About Frontiers for Young Minds**

Frontiers for Young Minds believes that the best way to make cutting-edge science discoveries available to younger audiences is to enable young people and scientists to work together to create articles that are both accurate and exciting. That is why distinguished scientists are invited to write about their cutting-edge discoveries in a language that is accessible for young readers, and it is then up to the kids themselves – with the help of a science mentor – to provide feedback and explain to the authors how to best improve the articles before publication. As a result, Frontiers for Young Minds provides a collection of freely available scientific articles by distinguished scientists that are shaped for younger audiences by the input of their own young peers.

**What are Frontiers for Young Minds Collections?**

A Collection is a series of articles published on a single theme of research and curated by experts in the field. By offering a more comprehensive coverage of perspectives and results around an important subject of research, we hope to provide materials that lead to a higher level of understanding of fundamental science. Alternatively, a collection could feature articles by scientists who received special recognition, such as a Nobel Prize. Frontiers for Young Minds Collections offer our international community of Young Minds access to the latest and most fundamental research; and, most importantly, empowering kids to have their say in how it reaches their peers and the wider public. Every article is peer reviewed according to the Frontiers for Young Minds principles. Find out more on how to host your own Frontiers for Young Minds Collection or contribute to one as an author by contacting the Frontiers Editorial Office: [kids@frontiersin.org](mailto:kids@frontiersin.org)





# Math That Changed the World

## Collection editors

Idan Segev — Hebrew University of Jerusalem, Israel

Jeremy Martin — University of Kansas, United States

Robert Knight — University of California, Berkeley, United States

## Citation

Segev, I., Martin, J., Knight, R., eds. (2024). *Math That Changed the World*.  
Lausanne: Frontiers Media SA. doi: 10.3389/978-2-8325-4952-0

## Cover image

FourPlus Studio

## Participating sections



Mathematics  
and Economics



Engineering  
and Technology

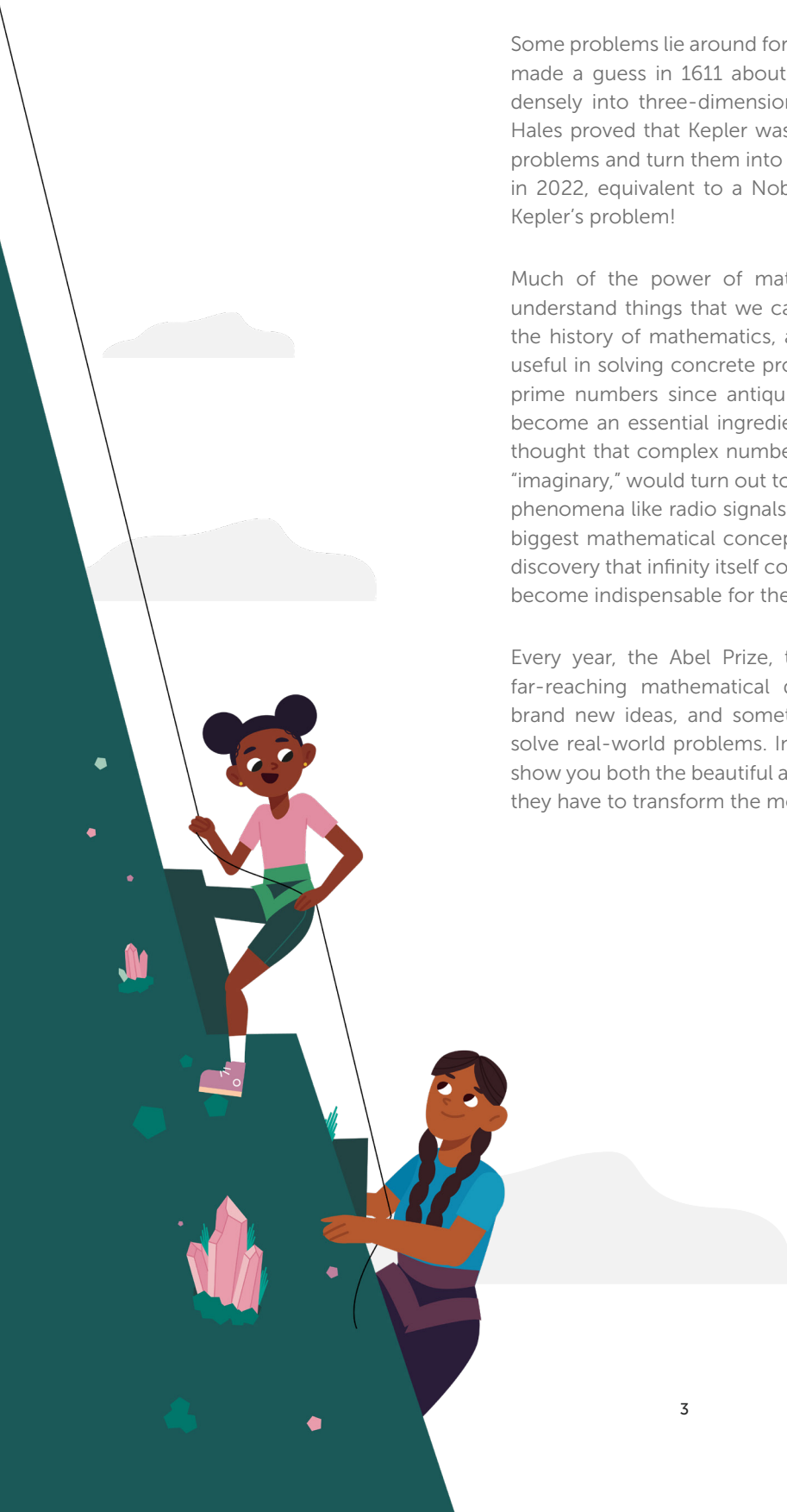
## About this collection

Mathematics is as old as civilization. The Maya developed sophisticated arithmetic to study the stars and the weather, the ancient Greeks used geometry to estimate the size of the earth with remarkable accuracy, and the Chinese mathematician Liu Hui calculated pi to five decimal places two millennia before computers had been invented. And mathematics is alive and well today, with new discoveries made every day, sometimes about problems that are easy to understand but surprisingly difficult to solve.

Some problems lie around for centuries waiting to be solved: Johannes Kepler made a guess in 1611 about how a pile of spheres could be packed most densely into three-dimensional space; it took four centuries until Thomas Hales proved that Kepler was right. And mathematicians love to expand old problems and turn them into new ones: Marina Viazovska won a Fields Medal in 2022, equivalent to a Nobel Prize, for solving an even harder version of Kepler's problem!

Much of the power of mathematics comes from how it enables us to understand things that we can't see or experience directly. And throughout the history of mathematics, abstract ideas have proven to be unexpectedly useful in solving concrete problems. Mathematicians have been playing with prime numbers since antiquity; who would have thought that they would become an essential ingredient of modern cryptography? Who would have thought that complex numbers, first discovered in the 1500s and derided as "imaginary," would turn out to be exactly the right tools to describe real world phenomena like radio signals and electrical circuits? Even infinity, literally the biggest mathematical concept of all, is useful: Georg Cantor's revolutionary discovery that infinity itself comes in different sizes introduced ideas that have become indispensable for the study of computers and computer algorithms.

Every year, the Abel Prize, the Gödel Prize and the Turing Award honor far-reaching mathematical discoveries. Sometimes these discoveries are brand new ideas, and sometimes they are applications of mathematics to solve real-world problems. In this collection, recipients of these awards will show you both the beautiful abstract ideas they study, and the amazing power they have to transform the modern world.





## Table of contents

- 05 **Will Learning Machines Take Over the World?**  
Yann LeCun
- 15 **The Math Behind the Movies**  
Pat Hanrahan
- 23 **Easy Or Hard? Basic Questions in Computational Complexity Theory**  
Noa Segev and Avi Wigderson
- 33 **Combinatorics: The Mathematics of Fair Thieves and Sophisticated Forgetters**  
Noga Alon
- 45 **Wavelets: Mathematical Tools for Image Analysis**  
Ingrid Daubechies





# WILL LEARNING MACHINES TAKE OVER THE WORLD?

**Yann LeCun**<sup>1,2\*</sup>

<sup>1</sup>Courant Institute, New York University (NYU), New York, NY, United States

<sup>2</sup>Meta, AI-FAIR, New York, NY, United States

## YOUNG REVIEWERS:



**OISIN**

AGE: 12



**ZI-AN**

AGE: 8

Learning is an integral part of our lives, and the lives of all animals, but do you realize how wondrous our learning ability is? When we try to build machines that can learn, we are faced with deep questions about the nature and functioning of intelligence. In this article, I will tell you about special artificial networks, called neural networks, that mimic the brain to produce intelligent behavior. Neural networks are an integral part of artificial intelligence, widely used in many daily applications, from face recognition to autonomous driving. I am certain that neural networks will play a key role in our future lives, making them more comfortable and improving our understanding of big riddles such as what intelligence is and how our brains work.

Professor Yann LeCun won the Turing Award in 2018, jointly with Prof. Geoffrey Hinton and Prof. Yoshua Bengio, for conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing.



## ARTIFICIAL INTELLIGENCE (AI)

The ability to reproduce, and even surpass, with machines the capabilities of humans and animals.

## MACHINE LEARNING (ML)

Teaching machines how to learn, or improve their performance, as result of previous experience, with as little human intervention as possible.

## ALGORITHM

A set of instructions that tells machines how to operate.

## NEURONS

Cells in the brain that process information using electrical signals.

## ARE MACHINES SMARTER THAN HUMANS?

Are you intelligent? I assume that all of you would say “yes”. Now, what about your dog or cat? And how about your smartphone? Though intelligence is intuitive for us, it is actually a complex phenomenon [1] and there is no clear definition of intelligence and no absolute measure of how intelligent a system is. What we often do in **artificial intelligence (AI)** is try to reproduce, or surpass, the capabilities of humans and animals, particularly intellectual abilities like understanding language and reasoning.

AI is a moving target. Whenever some problem is solved so that machines outperform humans in a specific task, it is no longer AI. Many things which used to be part of AI are not any more, such as using GPS to navigate the best route or performing complex arithmetical calculations. Through advancing AI we also learned that humans are relatively good, or intelligent, at certain things but pretty bad at other things. For example, until recently, people thought that we were unbeatable at board games like Chess and Go. But today we know that advanced AI systems can beat the best human players—see [this documentary](#) to learn about the first AI system (AlphaGo) to beat a human world champion, in 2015. Today, there are other tasks which AI systems can perform better than humans, such as instantaneous translation and image recognition. But even the most intelligent systems are highly specialized: they are good only at a very narrow range of tasks [2].

Truly intelligent machines must be able to do what all animals do—learn [3]. Learning means adapting, or using knowledge gained in experiences to perform well in a task within new scenarios. Humans and animals learn naturally, but how do machines learn? That is the research of **machine learning (ML)**. In ML, we develop systems and **algorithms** that take in data and use it to continuously update their behavior and improve their performance. ML is the basis of many applications, like automatic translation of webpages to different languages, voice control and face recognition in smartphones, driving autonomous cars and analyzing medical data. But the best learning machines still can not learn like humans. For example, it takes humans only about 20–30 h to learn how to drive properly, but currently no machine can drive autonomously as well as a human, even after thousands of hours. A big ML breakthrough was when we figured out how to imitate the way the brain learns.

## HOW DO WE LEARN?

Our brains are made of about 100 billion (hundred thousand million) **neurons**, each connected to about 10,000 other neurons (this means we have about 1,000 trillion connections in our brain!). They communicate via electrical signals that they transmit and receive.

## SYNAPSE

The junction that connects two neurons. Its strength can vary with time depending on how often it is used to pass signal between the neurons.

## Figure 1

Biological and artificial neurons. **(A)** Biological neurons in the brain whose synapses' strength and location change when we learn. **(B)** Artificial neurons are digital units inside a computer, that receive input and transmit output to other artificial neurons.

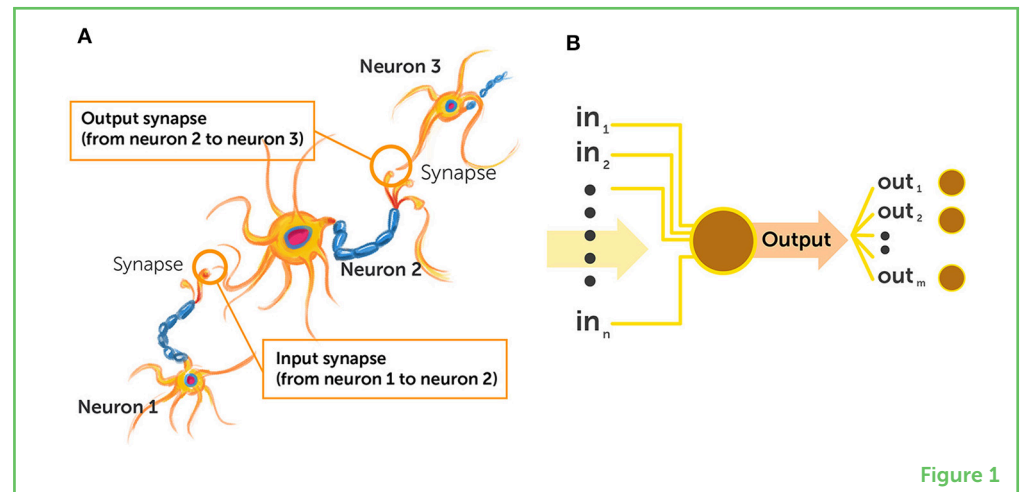


Figure 1

## ARTIFICIAL NEURONS

A representation of a neurons inside a computer. Artificial neurons receive digital inputs, perform a calculation, and transmit a digital output to other artificial neurons.

## NEURAL NETWORK (NN)

A network of artificial computing elements that simulates the operation of neurons in the brain.

## DEEP LEARNING (DL)

A machine learning technique based on neural networks with at least two internal layers.

## NEURAL NETWORKS—BIOLOGICAL AND ARTIFICIAL

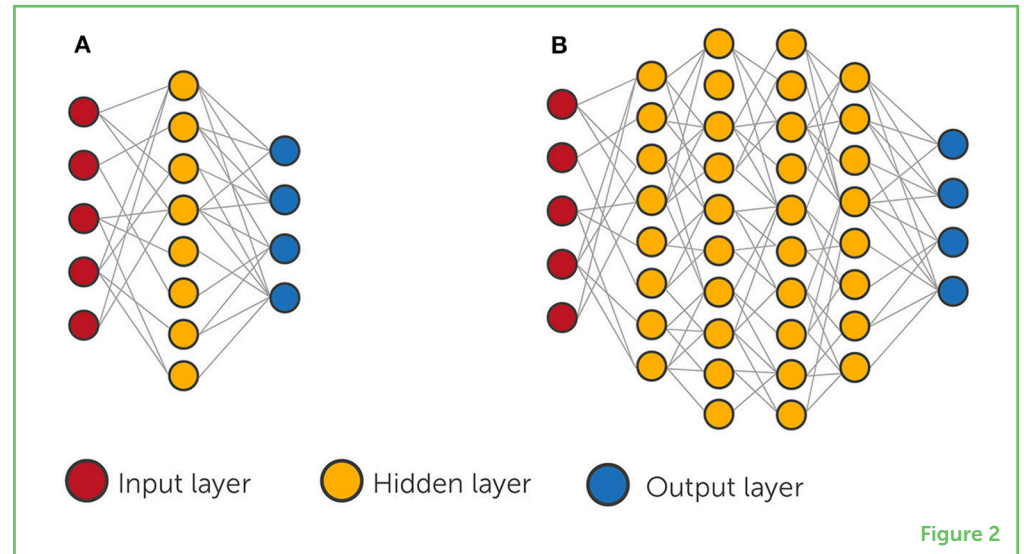
A great leap in ML occurred with the birth of **neural networks (NNs)** [4]. A NN is a network of connected artificial neurons, simulated within a computer and organized in layers (to learn more about NNs, read [this paper](#) on artificial neural networks or [this paper](#) on the similarity between nerve cells and AI). The first layer, which is the input layer ([Figure 2](#)), receives the data that we feed it, for example, an image, video, or audio file. The middle layers, called the hidden layers, process the data, and the output layer gives the result of the computation (for example, identification of objects in an image or the transcription of voice to text). The NN is considered simple, or shallow, if it has only one hidden layer, and a deep neural network (DNN) if it has two or more hidden layers. In this article we will focus on DNN, which perform **deep learning** [5].

Every artificial neuron in the DNN is connected to all other neurons in the next layer of the network, with various strengths of connection, called weights ([Figure 3](#)). Every neuron receives input from all other neurons in the previous layer, performs a calculation based on the weights of connections, and then sends an output to all other neurons in the next layer ([Figure 3](#)).

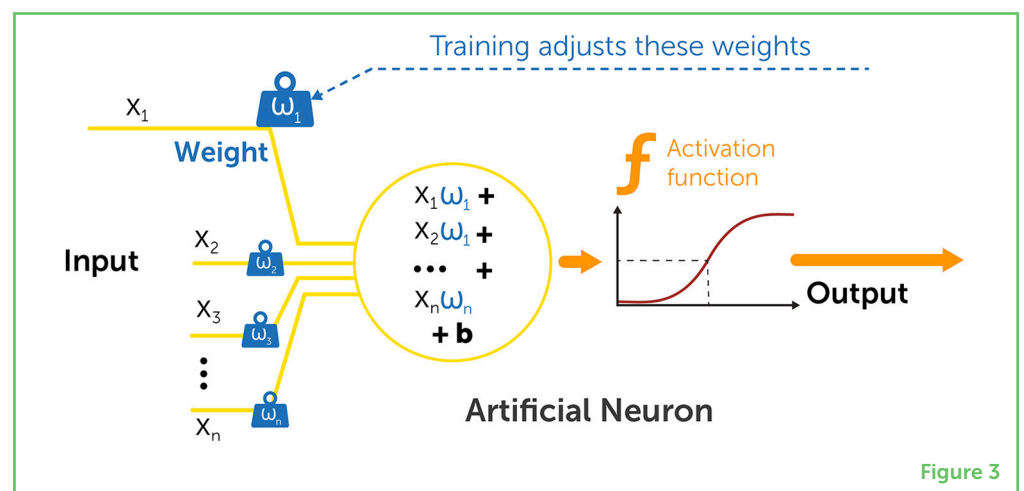


**Figure 2**

Artificial neural networks. Artificial neural networks are collections of artificial neurons organized in layers. **(A)** A simple neural network: one “hidden layer”. **(B)** A deep neural network: two or more hidden layers. Large deep neural networks can contain about a thousand neurons in each layer and a few tens of hidden layers.

**Figure 2****Figure 3**

How neural networks learn. Every artificial neuron in a neural network receives input signals ( $x_1$  to  $x_n$ ) from the artificial neurons in the previous layer. It multiplies these inputs using a set of weights ( $w_1$  to  $w_n$ ), sums them up and adds another constant property called a bias ( $b$ ), that determines how active a neuron will be. Finally, it runs this summation result through a function called an activation function, which determines the final result that the neuron sends, and sends the final output to all the neurons in the next layer. Learning occurs in the network by adjusting the set of weights and bias that the artificial neurons use in their calculations. To learn more about the calculations behind NNs, see [here](#).

**Figure 3**

This way, information flows from the input layer to the output layer in what is called forward propagation. At the output layer, the DNN says what it thinks the right outcome is (for example, whether there is a cat or a dog in the picture).

Now comes an interesting part, where our network must learn. Learning means updating the weights in the DNN (like the changing synapses in the brain), to perform the task better next time. Think of every weight in the DNN as a knob which can be tuned to change the overall result—usually tens of thousands of them. To tune these knobs, the DNN first compares the outcome it got (the activation of its neurons in the output layer, each representing a specific outcome) with the right outcome it should have gotten (expected activation of each neuron in the output layer). Then, it updates the weights in all layers, from the last to the first hidden layer, to minimize the difference between the desired outcome and actual outcome.

## BACKPROPAGATION

A learning algorithm commonly used for learning in deep neural networks.

There are two common ways to train DNNs. The first is supervised training, which is more accurate but less efficient, where systems scan many data samples already labeled by people (e.g., images of cats and dogs correctly categorized). The second way is self-supervised training, where the network tries to reconstruct information it already has (its input) after representing the information within its layers—this does not require human intervention.

The common algorithm used for updating DNN weights is called **backpropagation** [5, 6]. Backpropagation allows the DNN to improve its performance after every trial. The DNNs we currently train keep improving after every trial, but their rate of improvement decreases, so though at some point we stop investing in training, nonetheless the network keeps improving. After a network learns a training set of data, it then receives a test set to check its performance. When its performance is good enough for the specific task, it is ready to be used on new unseen data.

## UNDERSTANDING THE WORLD USING CONVOLUTIONAL NEURAL NETWORKS

In the visual cortex—our brain's visual information processor—neurons respond to different visual features, called motifs, such as the orientation of lines [7], or edges. Collections of neurons, each detecting a different motif, are repeated everywhere inside the visual cortex, detecting the same motifs on different parts of the image we see. Inspired by this architecture of the brain, researchers applied the same basic structure on artificial neural nets [8]. I further developed these neural nets, which were trained using backpropagation [9] and that eventually led to the development of convolutional neural networks (CNNs).

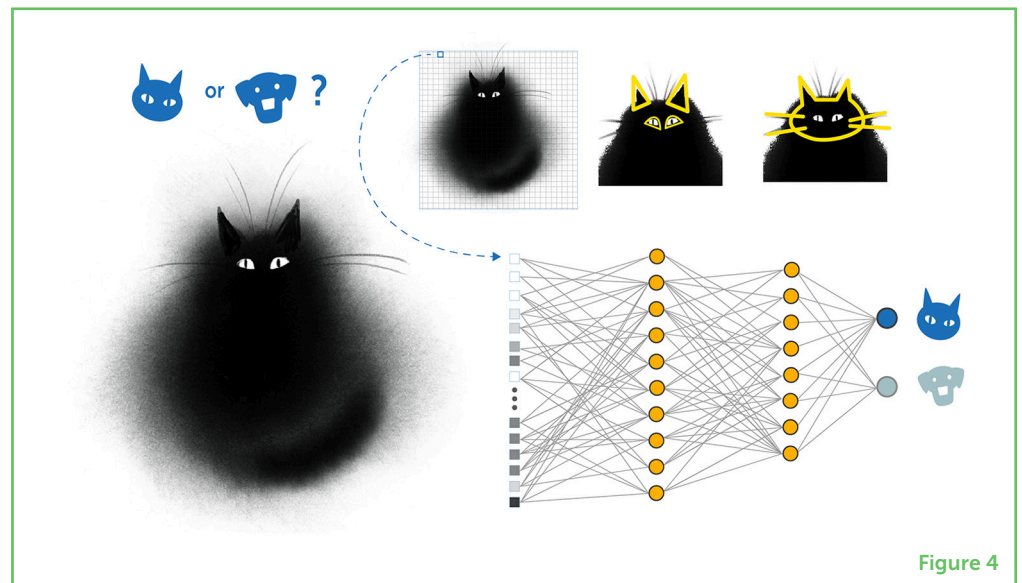
The assumption behind CNNs is that signals we perceive from the world are composited of simple elements that build on top of each other to create complex objects. Simple motifs (e.g., lines and edges) are combined into parts of objects (e.g., square surfaces and legs), which are then combined into more complex objects (e.g., a table) and even categories of objects in a hierarchical manner (Figure 4). In a CNN, we take a set of neurons and connect them to a small patch in our input data, an image for example. Then, we take the same set of neurons and copy them all over the image, so that they look at every different part or patch. The neurons' output determines the presence or absence of a feature within each patch of the image. As the information propagates throughout the CNN layers, these features become more composite, or complex (to see an intuitive demonstration of how CNNs work, see [this video](#)). Amazingly, we do not have to tell the CNNs which features to look for—we just train them end-to-end using backpropagation, and they—quite



magically—figure out themselves which features should be used to identify the object.

**Figure 4**

Image recognition using convolutional neural networks. Convolutional neural networks (CNNs) are used for various applications, including image recognition. The network represents an image in the input layer as a list of values (indicating the color of different pixels in the image). It then propagates the information into more complex objects (such as ears, eyes, whole face), until eventually the network output determines what object is present in the image (cat or dog in this example).



The first CNN I used was trained for recognizing hand-written digits [9]. It was quite successful and was later implemented for reading cheques in banks in the US and Europe. CNNs are useful for various other applications, related to images (such as Optical character recognition, or OCR, face recognition and video surveillance), as well as to speech recognition [10, 11]. CNNs have greatly improved the performance of former DNNs and are now integral to cutting-edge technologies such as medical imagery analysis and autonomous driving. My contribution to CNNs development won me the prestigious Turing Award in 2018, together with my colleagues Geoffrey Hinton and Yoshua Bengio.

## CAN NEURAL NETWORKS HELP US UNDERSTAND THE BRAIN?

As we have seen, the development of artificial neural networks was inspired by the brain's operations. So can NNs be useful in understanding the brain itself? My answer is they are *necessary* to understand the brain. The brain is a very complex organ and there are probably some underlying principles governing its abilities that we have not uncovered yet. Though neuroscientists have collected a huge amount of experimental data, there is no solid theory about how the brain works. To show that a theory of brain functioning is correct, we need to recreate it within a computer and see that it works in ways that are somewhat similar to that of the brain. If we manage to build a computerized brain that acts similarly to the biological brain, this indicates we captured common operating principles, despite the differences between the systems.

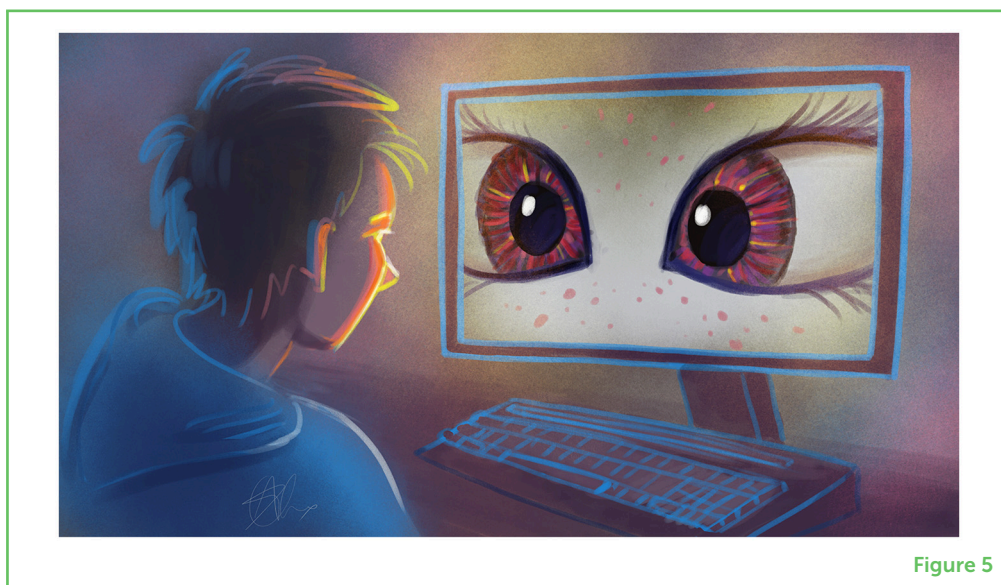
Today, many brain scientists are already using deep learning and NNs as models for explaining brain activity [12], particularly in visual cortex, but it is also relevant for explaining how we process speech and text. NNs allow us to inspect information processes by recording the activity of all the neurons and using that to understand how they represent data. But to understand the collective operation of millions or billions of these elements, at some point we must describe the operation of the whole network at an abstract level. Having NNs that operate in similar ways to the brain would provide a great leap in our overall understanding of the brain, and of intelligence.

## WILL MACHINES TAKE OVER THE WORLD?

I think the issue of controlling AI has become a modern “bogeyman”, with horror forecasts of machines becoming smarter than us and dominating us (Figure 5). However, in humans and thus in machines, intelligence does not equal a will to dominate the other. Another concern is aligning behavior of intelligent machines with humanity’s values. Although it is hard to “educate” machines to behave properly, we can manage it—the same way we educate our kids to behave in society, and with the same regulations to guide social functioning. We can define intrinsic objectives that machines will pursue (think of these like “core values”) that they cannot violate or modify, ensuring that the machines’ behavior stays aligned with our values and goals.

**Figure 5**

Should we be afraid of machines? I believe the worry about machine dominance is unfounded as we can ensure that machines will be friends not foes.



**Figure 5**

Every new technology brings with it some unexpected consequences, so we as a society must correct any unwanted side effects quickly, minimizing their damage. After online services developed, such as YouTube, Facebook, Instagram, we encountered the problem of improper content and developed means for content moderation. I am

confident in our ability to deal with the problems of new technologies successfully as they arise.

What excites me the most about the future of AI is uncovering the underlying principles of intelligence. That would help us explain what human intelligence really is, and enable us to build intelligent systems, which would eventually expand human intelligence. Progressing in our understanding of the world requires more intelligence: at some point we will need other systems beyond our limited brains that we can use. For an interesting example of how AI helped us better understand the world, read [this article](#) about solving the long standing problem of protein folding.

Another, more engineering-based endeavor I want to see in the future is building intelligent systems to help us with our daily lives. For example, domestic robots that will be like intelligent human assistants, managing things we do not want to do and filtering out unimportant information. We call this an AI-complete problem [13], that requires the integration of many abilities and techniques. I work on new basic self-supervised learning algorithms that could hopefully bridge the gap between machine learning today and human learning. I hope we will be able to tackle AI-complete problems more successfully and live even more comfortable lives.

## ADDITIONAL MATERIALS

1. [Yann's Home Page](#).
2. [Yann's Twitter Posts](#).
3. Check out this [fake speech](#) by the former president of the United States, Barack Obama, made using AI.
4. [Can Computers Understand Humor?](#)
5. [DALL·E 2](#) and [Lexica Aperture](#)—AI-based online softwares for creating images from natural language.

## ACKNOWLEDGMENTS

I wish to thank [Or Raphael](#) for conducting the interview which served as the basis for this paper, and for co-authoring the paper, and [Alex Bernstein](#) for providing the figures.

## REFERENCES

1. Pfeifer, R., and Scheier, C. 2001. *Understanding Intelligence*. Cambridge: MIT Press.
2. Wolpert, D. H., and Macready, W. G. 1997. No free lunch theorems for optimization. *IEEE trans. Evolut. Computat.* 1:67–82. doi: 10.1109/4235.585893
3. Alpaydin, E. 2016. *Machine Learning: The New AI*. Cambridge: MIT press.



4. McCulloch, W. S., and Pitts, W. 1943. A logical calculus of the ideas immanent in nervous activity. *Bullet. Mathemat. Biophys.* 5:115–33.
5. LeCun, Y., Bengio, Y., and Hinton, G. 2015. Deep learning. *Nature*. 521:436–44. doi: 10.1038/nature14539
6. LeCun, Y., Touresky, D., Hinton, G., and Sejnowski, T. 1988. "A theoretical framework for back-propagation", in *Proceedings of the 1988 Connectionist Models Summer School* (Pittsburg, PA: Morgan Kaufmann), 21–28.
7. Hubel, D. H., and Wiesel, T. N. 1959. Receptive fields of single neurones in the cat's striate cortex. *J. Physiol.* 148:574. doi: 10.1113/jphysiol.1959.sp006308
8. Fukushima, K., and Miyake, S. 1982. "Neocognitron: a self-organizing neural network model for a mechanism of visual pattern recognition", in *Competition and Cooperation in Neural Nets* (Berlin: Springer), 267–285.
9. LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., et al. 1989. "Handwritten digit recognition with a back-propagation network", in *Advances in Neural Information Processing Systems (NIPS 1989), Vol. 2* (Denver, CO: Morgan Kaufmann).
10. LeCun, Y., Kavukcuoglu, K., and Farabet, C. 2010. "Convolutional networks and applications in vision", in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems* (Paris: IEEE), 253–256.
11. LeCun, Y., and Bengio, Y. 1995. "Convolutional networks for images, speech, and time series", in *The Handbook of Brain Theory and Neural Networks*, ed. M. A. Arbib (MIT Press).
12. Yamins, D. L., and DiCarlo, J. J. 2016. Using goal-driven deep learning models to understand sensory cortex. *Nature Neurosci.* 19:356–365. Available online at: <https://www.nature.com/articles/nn.4244>
13. Weston, J., Bordes, A., Chopra, S., Rush, A. M., Van Merriënboer, B., Joulin, A., et al. 2015. "Towards AI-complete question answering: a set of prerequisite toy tasks", in *arXiv*.

**SUBMITTED:** 04 April 2023; **ACCEPTED:** 26 March 2024;

**PUBLISHED ONLINE:** 07 May 2024.

**EDITOR:** Idan Segev, Hebrew University of Jerusalem, Israel

**SCIENCE MENTORS:** Yunchao Tang and Tomas Emmanuel Ward

**CITATION:** LeCun Y (2024) Will Learning Machines Take Over the World? *Front. Young Minds* 12:1164958. doi: 10.3389/frym.2024.1164958

**CONFLICT OF INTEREST:** YL was employed by Meta.

**COPYRIGHT** © 2024 LeCun. This is an open-access article distributed under the terms of the [Creative Commons Attribution License \(CC BY\)](https://creativecommons.org/licenses/by/4.0/). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

## YOUNG REVIEWERS

### OISIN, AGE: 12

Oisín likes to play piano, chess, and play video games with his friends. He also likes to draw and play soccer. His favorite video games are Minecraft, City Skylines, and Civilization VI. Oisín loves to read Harry Potter and Discworld books by Terry Pratchett. Oisín lives in Ireland and is in sixth class in national school.

### ZI-AN, AGE: 8

Hi, I am Zi-An, coming from a family of teachers, I have inherited a love for knowledge and learning. But my biggest joy? That is definitely my little brother. I absolutely love goofing around and making him laugh! I am fascinated about science. I want to research the secrets of everlasting life, so that people I love will never grow old or die.

## AUTHORS

### YANN LECUN

Yann LeCun is VP & Chief AI Scientist at Meta and Silver Professor at NYU affiliated with the Courant Institute of Mathematical Sciences & the Center for Data Science. Yann received an Engineering Diploma from ESIEE (Paris) and a PhD in Computer Science from Sorbonne Université. After a postdoc in Toronto, Yann joined AT&T Bell Labs in 1988, and in 1996 he became Head of the Image Processing Lab. at AT&T. He joined NYU as a professor in 2003 and Meta/Facebook in 2013. His interests include AI machine learning, computer perception, robotics and computational neuroscience. Yann won several prestigious awards including the Turing award (2018), AAAI Fellow (2019), and Legion of Honor (2020), and is a member of the National Academy of Sciences, the National Academy of Engineering, the French Académie des Sciences. Yann has several hobbies, including building **model airplanes** with his family and **playing music** (mostly Jazz nowadays). \*[yann@cs.nyu.edu](mailto:yann@cs.nyu.edu)



## THE MATH BEHIND THE MOVIES

**Pat Hanrahan\***

*Computer Graphics Laboratory, School of Engineering, Stanford University, Stanford, CA, United States*

### YOUNG REVIEWERS:



**HRDAYA**

AGE: 9



**MRS.  
MINOGUE'S  
CLASS AT  
KLONDIKE  
ELEMENTARY**

AGES: 9–11

Have you ever wondered why the things you see around you look the way they do? A big part of my work as a computer graphics researcher is to answer this question, so that we can simulate real-world objects using computers. This work is important for creating movies and video games that are visually appealing to the audience. However, there are also many other applications of realistic computer graphics, such as training, product design, architecture, and others. In this article, I will tell you how we create realistic images of the world using computers, and how my work improved the way we simulate skin and hair in virtual characters. It will give you a new perspective on the magic and beauty of the visual scenery that we experience in our everyday lives and on the computer-generated images and movies that imitate these natural scenes.

Professor Pat Hanrahan won the Turing Prize in 2019, jointly with Dr. Edwin Catmull, for fundamental contributions to 3D computer graphics and the impact of computer-generated imagery (CGI) in filmmaking and other applications.

## COMPUTER GRAPHICS

Generation of digital images using computers.

## RENDERING

The process of taking a description of a 3D scene and converting it into a computer image.

## THE WORLD AROUND US

What do you see when you look out of your window? I see some oak trees with beautiful leaf canopies. When it is sunny, the leaves seem to be glowing with glorious light. Because I have worked for so many years in **computer graphics**, I look at our everyday surroundings with an inquisitive, observing mind. When I see a beautiful landscape, I notice subtle variations in lighting of natural objects, such as that on the leaves outside my window, I wonder: what makes it so visually interesting and beautiful? How exactly do the leaves scatter the sunlight that hits them before it reaches my eyes? And what would be the best way to imitate this using a computer so that the leaves on my computer screen will also look beautiful as they do in nature?

If you think about it, everything that we see in the world—from trees to people to clothes—is made up of complex shapes and materials. To describe the composition of a visual scene, we must describe all the geometric shapes that make up the objects within that scene, as well as the materials they are made of. If we look at a human, for example, we see the skin, the face, the hair, the clothes—all made of materials with unique properties. There are also several kinds of light sources—the Sun, a neon light, or a spotlight, for example—and each source creates a unique pattern of lighting. Finally, the observed scene is perceived by the viewer (or recorded by the camera) from a certain point of view. A close look at any scene reveals a huge variety of elements working together to eventually create the image we see. We usually take what we see for granted in our daily lives, but when we must create a realistic image of the world using a computer, we begin to appreciate how interesting and intricate our visual reality is and how challenging it is to represent it convincingly in the computer.

## SIMULATING THE WORLD USING COMPUTERS

In my early career, I worked at Pixar—a studio well-known now for their animated stories. Pixar wanted to make special effects in movies more convincing by combining computer-generated material with photographs taken of live action. To seamlessly combine the computer-generated parts with photographs, the computer-generated images had to be photorealistic, meaning indistinguishable from real-world photos ([Figure 1](#)). This required creating 3D pictures from descriptions of the elements in a scene. This was the first stage in the process called **rendering**—simulating the world using a computer (to learn more about rendering, see [this video](#)) [1]. Eventually, we realized that rendering could be used not only for special effects, but for creating entire movies only in the computer without having to film them in nature. I was fascinated by the challenge of creating a realistic virtual world, and for the last 30–40 years I have been making rendering better and better.



### Figure 1

The road to point Reyes. This image, produced in 1983 by Lucasfilm (which later became Pixar), is an example of a high-resolution photorealistic image created using a computer. It marked an important milestone toward the creation of computer-generated digital films (Image source: [http://www.calgran.net/upf/recursos/ima\\_dig/\\_2\\_/estampes/d3\\_1.html](http://www.calgran.net/upf/recursos/ima_dig/_2_/estampes/d3_1.html)).



Figure 1

### COMPUTER MODEL

Representation, using the computer, of real-life situations and processes.

### RAY TRACING

A method used in rendering to figure out the lighting of a scene by simulating the behavior of light.

### SCATTERING FUNCTION

A mathematical description of how a specific material scatters the light that hits it.

How does rendering work? First, artists, designers, and animators provide us with 3D descriptions (in drawings or in text) of the world we want to simulate. These descriptions are saved within computer files. Once we have a complete description of a specific scene, we must create a **computer model** of it to eventually convert the description files into actual images like the JPEG files you might be familiar with. The simplest way to model complex shapes with a computer is by using polygons, such as triangles and squares (Figure 2). We break all the complex shapes in a scene down into very small triangles, normally billions of polygons per image. Then, one of the biggest challenges for realistic rendering is to correctly simulate the lighting. The simplest way to model light in a computer-simulated world is to assume that light rays move in straight lines. To model light correctly, we use a technique called **ray tracing**, in which we follow the paths of simulated light rays in our virtual scene, to figure out what the lighting of this image should look like.

Every direct light source in a scene, like the Sun or a light bulb, emits light rays. These rays travel in space until they hit the surface of some object and get scattered. Every material scatters light in a unique way, depending on the specific properties of its surface (for example, how smooth and transparent it is). Much of our work as computer graphics researchers is focused on simulating how different materials (say a stone, cloth or the skin) scatter light. For every material we eventually want to find a mathematical function that describes how that material scatters light. This mathematical function is called the **scattering function** of the material.

One way to model the scattering function of a material is to represent the material's surface as if it is composed of many small mirrors

## Figure 2

Simulating light to make images realistic. A complex scene is represented in the computer by a set (of often millions or billions) of small polygons. To determine the brightness and color of each pixel in the rendered image, we trace the paths of rays emanating from the light source and simulate how they are reflected from the surfaces they hit. We account for direct lighting, where the light is reflected directly to the eye or camera. The more challenging problem is indirect lighting, where light from the sun illuminates other objects which scatter light which eventually reaches the eye. An example is skylighting where light from the sun is reflected from the clouds and the atmosphere, or indirect lighting from light fixtures of the walls in a building.

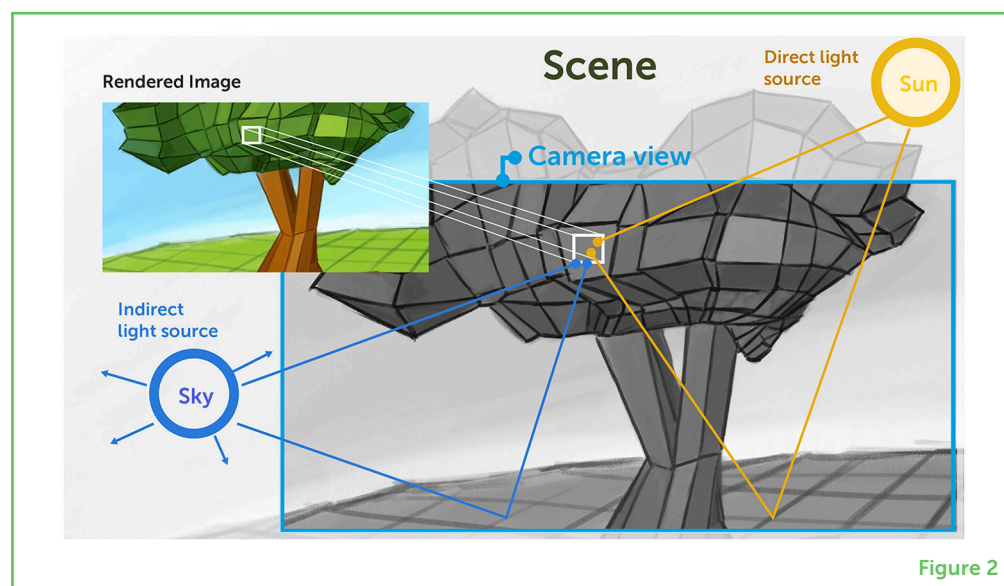


Figure 2

pointing in various directions (Figure 3B). When light rays hit a perfect mirror, they are reflected at an angle that is equal to the angle at which they hit the mirror (Figure 3A). Modeling the scattering function of a material as a random distribution of mirrors on its surface can work well for some materials, but many natural materials require more sophisticated models. We can determine the scattering functions of complex materials by direct measurements of light scattered from these materials. To do this, we shine light on the material, and measure the light scattered off of it in various directions. We can store the reflection in tables or arrays and use this information to model this specific material in the computer. We can also fit physical models to these measurements.

## Figure 3

Modeling materials using mirrors. (A) In a perfect mirror, light rays are reflected such that the angle at which the rays hit the mirror is equal to the angle at which they leave the mirror. (B) A simple way to approximate the scattering function of a material is to model its surface as a collection of perfect mirrors, each scattering the light in a known way. This works well for some materials, while others require a more complex model.

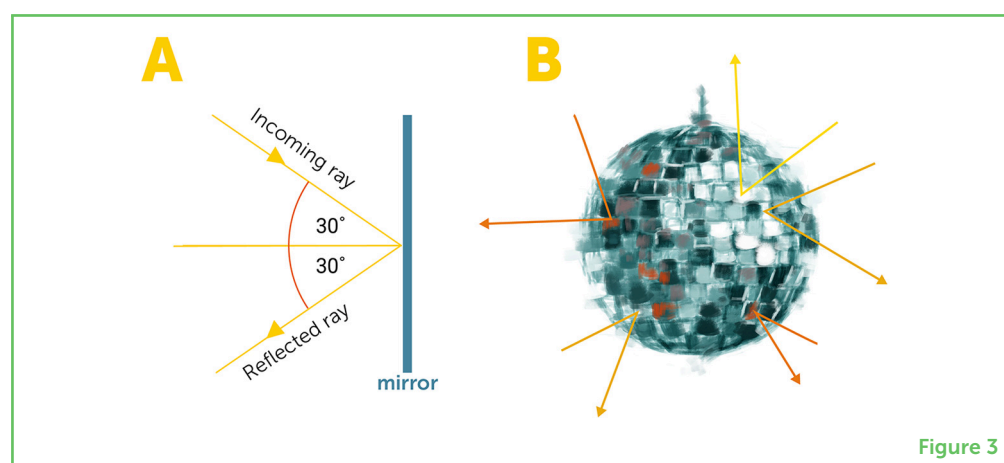


Figure 3

## HOW DOES YOUR SKIN LOOK SO GOOD?

One of the most interesting problems I worked on was how to realistically model skin. In early movies that used rendering, skin looked

quite unhealthy and artificial. No wonder, as it was essentially modeled like plastic! Healthy skin has a special glow to it that makes it look alive. There were no physics books explaining why skin looks this way, but luckily, I found two sources of information to help me crack this riddle. The first was work modeling of the atmospheres of planets (like Venus, Mars, or Jupiter), to explain why planets look the way they do. Planets often have a special glow around them, like skin does—see for example this image of [Jupiter](#). These models of the planet's atmospheres contained a solid core (the planet) surrounded by a thin layer of gases.

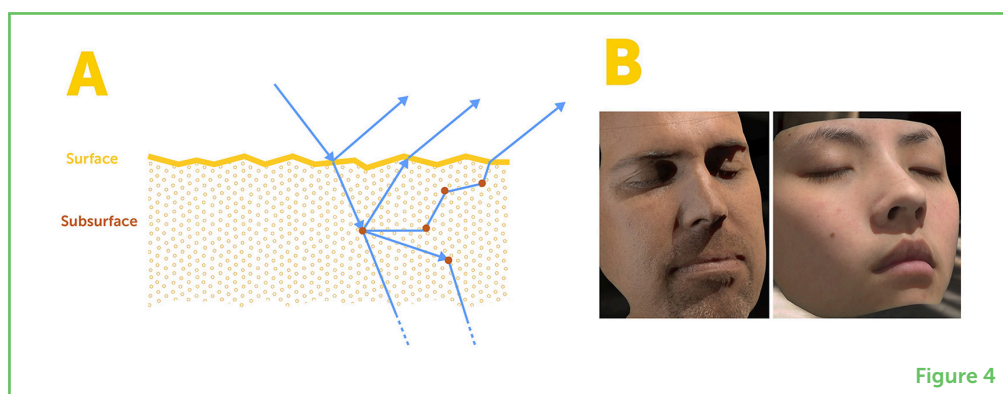
The second clue came from an art history book in the library at Princeton University, describing how the great painter Rembrandt painted skin, which look highly realistic in his paintings. He would layer pigments and oils on top of each other to create the skin's lively glow (to learn more about how Rembrandt painted skin, see [here](#)). Rembrandt was not familiar with the structure of the skin, but he somehow intuitively captured the fact that skin has several layers. Pictures explaining his painting technique show light rays going into the layers of the skin and eventually being reflected back out from the top layer. Using these two clues, I developed the first mathematical model of the scattering function of the skin [2]. This model was based on a process called **subsurface reflection**, in which light enters through the surface of the skin, hits the skin's internal, subsurface layers, gets scattered, and then comes back out (Figure 4). I wanted my first model to be as simple as possible, so I assumed that incoming light that scattered underneath the skin's surface comes out from the same point where it entered. This model was a fairly good simulation of skin, so I knew this was the right approach.

## SUBSURFACE REFLECTION

A type of light scattering in which light enters a material at one point, scatters underneath its surface, and exits at another point.

**Figure 4**

Subsurface reflection. **(A)** The skin is made of multiple layers. When light hits the skin, some of the light is reflected from the skin's surface and some of it enters the skin's subsurface layers. There, the light gets scattered in various directions and some of it comes back out in locations different from where it went in. This property was incorporated by the mathematical model we developed which provided a realistic simulation of skin in computer. **(B)** An example of realistic skin rendering. Adapted from d'Eon et al. [3].



**Figure 4**

However, my simple model made skin look bumpy, so I eventually developed a more complicated skin model, in which the light comes out of the skin at a different point from where it entered. That made the math behind the model more complicated, but it was worth it—the skin finally looked smooth! The reason the skin initially looked bumpy was that, if a small area has one lit side and one dark side, it will look like a bump. But, if the light that enters from the lit side is scattered

inside the skin and comes out on the dark side, the dark side also gets lit. In other words, the subsurface reflection spreads out the light in a way that makes the skin look smooth.

I also had an interesting journey trying to model hair. Early on, hair also looked unrealistic in renderings. It turned out that modeling hair as colored, transparent cylinders works very well [4]. Humans have about 100,000 hairs on their heads and, in current movies, each hair is individually simulated using a cylindrical model. Of course there are different kinds of hair, so human hair is modeled differently than animal fur, for example. It took time, but computer graphics designers eventually realized how to model each type of hair—now, simulated hair looks very good.

## MATH POWER IN COMPUTER GRAPHICS

Simulating light requires math because we must find and use the correct scattering functions to track the way light rays move in virtual scenes. But there is another important use for math in computer graphics simulations. In the real world, a practically infinite number of light rays enters every scene. Each ray hits objects and gets scattered an infinite number of times before the light reaches the observer. To realistically simulate light in a virtual scene, we must sum up all the paths that light could travel to arrive at a certain point, to understand how much light we have at that point. We cannot analytically perform this summation. So, we must take a finite number of samples that accurately represents all the possibilities. How can we figure out how many samples will give us the best estimate of lighting? To do this, we use a mathematical method that was originally used to predict the behavior of randomly bouncing particles, called the **Monte Carlo simulation** [5].

### MONTE CARLO SIMULATION

A mathematical method that uses random sampling to calculate the outcome of an uncertain event. It is used in computer graphics to simulate the lighting in digital scenes.

Today, every pixel in a simulated movie image has about 1,000 light rays coming into it and randomly bouncing around. There are billions of pixels in every image, so we must track an extraordinarily large number of rays for every frame within every scene. This requires a lot of computations and is very expensive. The fastest computers in the world are used to perform the computations for computer-generated movies and computer games. In a typical movie, about 30h of computing by the fastest computers are needed to compute *a single* film image. This can mean that some movies need up to one million hours of computing time! That is a lot of time, but it would be orders of magnitude more if we had not developed the mathematical tools that significantly decreased the number of calculations needed.

The next time you watch a computer-generated scene in a movie, I hope that you will have a new appreciation for the amount of research and computing resources that went into making it! Our everyday world becomes quite magical when we look at it from a computer scientist's



perspective. I invite you to learn to see the beauty of the world through careful observation, with curiosity about why things look the way they do.

## ACKNOWLEDGMENTS

I wish to thank [Or Raphael](#) for conducting the interview which served as the basis for this paper, and for co-authoring the paper, [Alex Bernstein](#) for providing the figures, and Susan Debad for copyediting the manuscript.

## ADDITIONAL MATERIALS

1. [Physically Based Rendering: From Theory To Implementation](#)
2. [Pat Hanrahan's homepage - Stanford](#)

## REFERENCES

1. Pharr, M., Jakob, W., and Humphreys, G. 2016. *Physically Based Rendering: From Theory to Implementation*. Cambridge, MA: Morgan Kaufmann.
2. Hanrahan, P., and Krueger, W. 1993. "Reflection from layered surfaces due to subsurface scattering," in *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY). p. 165–74.
3. d'Eon, E., Luebke, D., and Enderton, E. 2007. "Efficient rendering of human skin," in *Proceedings of the 18th Eurographics Conference on Rendering Techniques* (Goslar). p. 147–57. doi: 10.5555/2383847.2383869
4. Marschner, S. R., Jensen, H. W., Cammarano, M., Worley, S., and Hanrahan, P. 2003. Light scattering from human hair fibers. *ACM Transact. Graph.* 22, 780–91.
5. Harrison, R. L. 2010. "Introduction to monte carlo simulation," in *AIP Conference Proceedings, Vol. 1204* (Bratislava: American Institute of Physics). 17–21.

**SUBMITTED:** 04 April 2023; **ACCEPTED:** 10 November 2023;

**PUBLISHED ONLINE:** 25 January 2024.

**EDITOR:** [Jeremy L. Martin](#), University of Kansas, United States

**SCIENCE MENTORS:** [Srinivasa Ramanujam Kannan](#) and [Samuel Alperin](#)

**CITATION:** Hanrahan P (2024) The Math Behind the Movies. *Front. Young Minds* 11:1166415. doi: 10.3389/frym.2023.1166415

**CONFLICT OF INTEREST:** The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

**COPYRIGHT** © 2024 Hanrahan. This is an open-access article distributed under the terms of the [Creative Commons Attribution License \(CC BY\)](#). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and

the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

## YOUNG REVIEWERS

### HRDAYA, AGE: 9

I love reading books and I am fond of pointing out spelling or grammatical mistakes in magazines since I was 4 years old. English is my favorite subject. My hobbies are reading books, dancing, and playing card games. I am also curious to know more about things around me. I love puppies and birds.



### MRS. MINOGUE'S CLASS AT KLONDIKE ELEMENTARY, AGES: 9–11

Located in West Lafayette, Indiana, we are a classroom of voracious learners full of curiosity, creativity, perseverance, and compassion. We are a family of scholars!

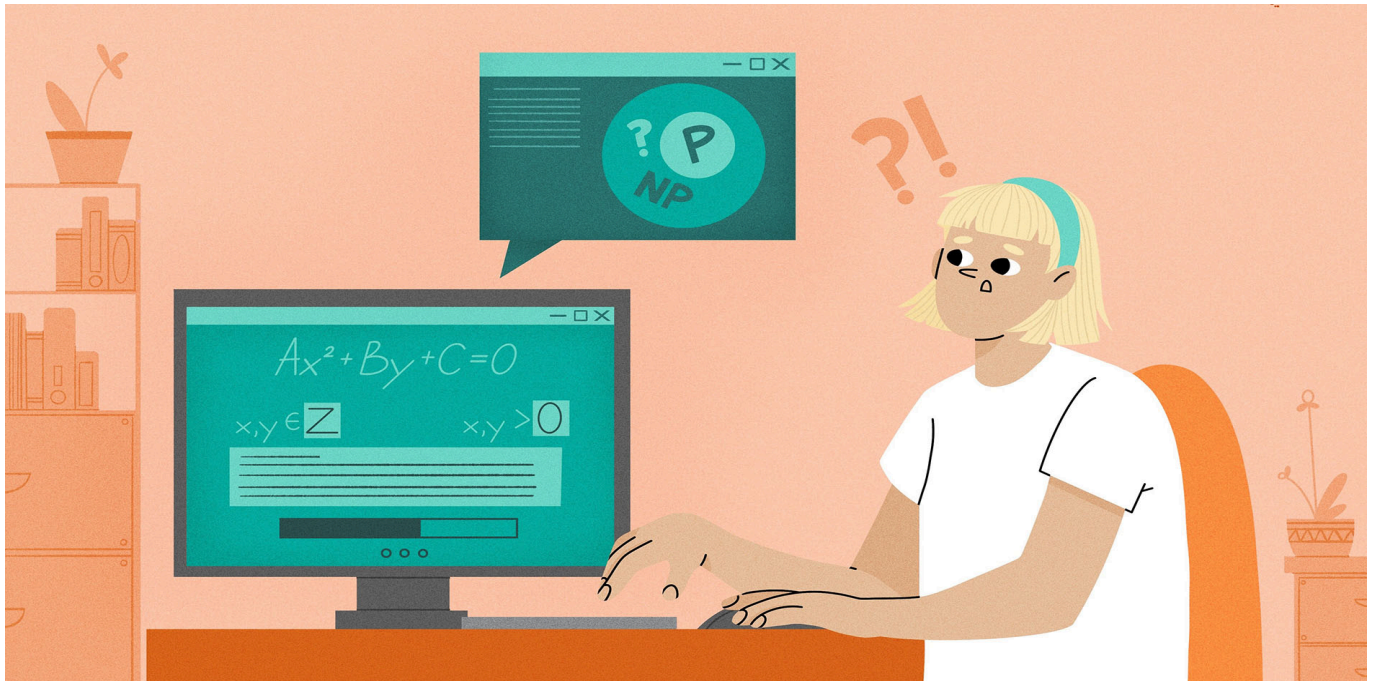
## AUTHORS

### PAT HANRAHAN

Prof. Pat Hanrahan is an American computer graphics researcher and professor in the Computer Graphics Laboratory at Stanford University. Hanrahan completed his B.S. in nuclear engineering at the University of Wisconsin-Madison in 1977, and his Ph.D. in biophysics 1985. During the 1980s, he worked on computer-animated films at Pixar, one of the world's leading animation studios, and was involved in the production of highly successful movies including computer animated movie, Toy Story, in 1995. In 1989, Hanrahan joined Princeton University, and in 1995 he moved to Stanford University, where he currently works. In 2003, he co-founded Tableau Software, a data-visualization software that helps businesses run more efficiently. Hanrahan has won numerous prizes for his contributions to computer graphics, including The Steven A. Coons award for outstanding creative contributions to computer graphics (2003), two technical Oscars (2004 and 2014), the Career Award for Visualization Research (2006), and the Turing Award (2019).

\*[hanrahan@cs.stanford.edu](mailto:hanrahan@cs.stanford.edu)





# EASY OR HARD? BASIC QUESTIONS IN COMPUTATIONAL COMPLEXITY THEORY

Noa Segev<sup>1\*</sup> and Avi Wigderson<sup>2</sup>

<sup>1</sup>Frontiers for Young Minds, Lausanne, Switzerland

<sup>2</sup>School of Mathematics, Institute for Advanced Study, Princeton, NJ, United States

## YOUNG REVIEWERS:



CHRIS  
AGES: 15



LORNA  
AGE: 15

Many things in our lives are designed to solve problems. Whether it is an app on our cellphones, the construction of a new building, or the development of a new drug, solving problems is a big motivator. Did you know that there is a fascinating type of mathematics behind many of the complex problems we face in our daily lives? This mathematics is called computational complexity theory, and it is a field of computer science. This is an active, constantly developing field that is attracting many talented young people—like you! In this article, we will describe computational complexity theory and the kinds of problems it is designed to help with. We hope that by the time you finish reading this article, you will be convinced that computational complexity theory is one of the most exciting fields of science.

Noa Segev is a scientific writer and project coordinator at Frontiers for Young Minds. She has a B.Sc. in physics and an M.E. in renewable energy engineering.

**Prof. Avi Wigderson won the 2021 Abel Prize, jointly with László Lovász, for their foundational contributions to theoretical computer science and discrete mathematics, and their leading role in shaping them into central fields of modern mathematics.**

## COMPUTATIONAL COMPLEXITY THEORY

When you think about the word “*computation*,” you probably think about numbers—maybe about the processes of addition and multiplication that you learned in your school math classes. Although those math operations are indeed computations, the term computation is actually much broader and includes many of the challenges we face in daily life, like the need to compute how to get from one place to another as quickly as possible, or how to keep sensitive information safe using encryption. The field of computer science that deals with many of the complex problems we face in our everyday lives is called *computational complexity theory*. Each computational complexity problem has an input and one or a bunch of valid solutions—not only one. For example, think about a navigation app like google maps. This app must solve the computation problem of how to get from point A to point B the fastest way possible. There could be a few different routes that take you from point A to point B in the same amount of time, and therefore this computation theory has a few equivalent solutions. The inputs in this case are the start and end points (A and B), and the solutions (also called the output) of the computations are all the possible routes you could take to get from point A to point B as quickly as possible.

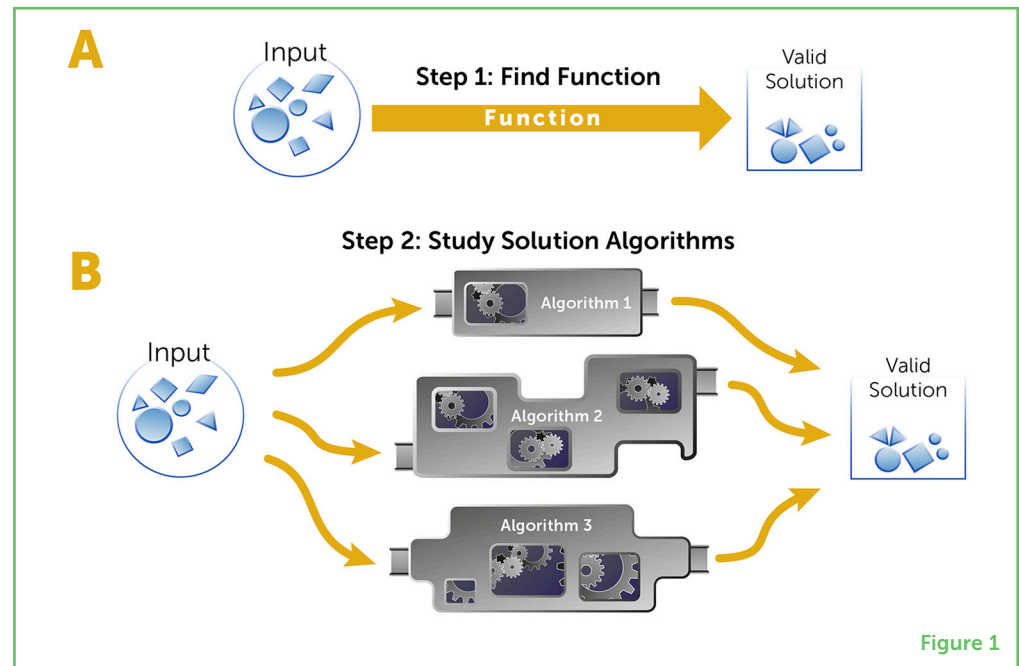
The solution of a computational problem requires two stages ([Figure 1](#)). The first stage is defining the function connecting the input and output. Once a function is given, the second stage involves finding an efficient way to solve the problem, to estimate the solution, or to prove that the problem is hard and cannot be solved in a reasonable time—that is the job of computer scientists who deal with computational complexity theory.

An important problem from biology—called the protein folding problem—can be used to demonstrate the stages of solving a computational problem. As you may know, our bodies contain tiny biological machines called proteins that perform many of our vital functions. Proteins are made of chains of building blocks called amino acids (like a bunch of beads on a string), and after they are made, these chains fold into complex, three-dimensional structures (see Figure 1 in [this article](#)). Proteins only function properly if they fold into the correct three-dimensional shapes. Scientists still do not know the exact physical and chemical laws that make proteins fold in the very specific ways that they do, out of all the millions of possible ways they *could* fold. This protein folding problem is a computational problem—for every specific chain of amino acids (the



**Figure 1**

A computational complexity problem. Computational complexity problems have a collection of valid solutions—not just one—for every input. Solving these problems requires two stages. **(A)** In the first stage, we must define the problem clearly or, in other words, describe the function connecting the input to the collection of valid solutions. This is usually the work of physicists, biologists, chemists, economists, or engineers. **(B)** In the second stage we look for algorithms—the series of operations that can solve the problem most efficiently. This is the stage is generally performed by computer scientists.



input), there is one solution of a specific three-dimensional that allows the protein to function. Scientists have created several models to try to describe how the amino acid sequence of a protein determines its final three-dimensional structure. One model uses the idea that the final structure of the protein is the one that needs the least energy to hold the shape. This can be calculated by adding up the attraction or repulsion forces of each pair of atoms that make up the protein [1]. Defining a model is the first stage in the solution process.

The next stage is the computational stage, which tries to determine how hard it is to calculate the solution to the problem, and to figure out the most efficient way to compute the solution—which in this case is to find the final three-dimensional structure of the protein. For example, if the protein only has a small number of atoms, it is easy to compute all the forces acting between every pair of atoms and add them up to figure out the three-dimensional structure that requires the least energy. But, if the protein is composed of many atoms (as most proteins are), it is much harder and sometimes even impossible to compute the exact folding that results in the minimal energy, even if we use very powerful computers. Maybe, in these cases, there is a better calculation that would allow a solution to be found? That is the kind of challenge that computational complexity theory deals with.

Computational complexity theory can deal with many practical questions that are related to our daily lives. In addition to helping us build models of systems we want to understand, like the protein folding problem, computational complexity theory can also be used to figure out what kinds of problems can and cannot be solved, to determine how efficient computations are at solving problems,

and to develop practical tools for solving problems. This may sound complicated, but you will see that it is a beautiful and fascinating field of research.

## ALGORITHMS AND EFFICIENCY

One of the most important problems in computational complexity theory deals with the efficiency of computations [2]. Every computation is defined by a collection of operations that are performed on the input to get to the result. This collection of operations is called an **algorithm**. We all use algorithms in our daily lives. For example, we all learn to add numbers in school. You may have learned to add according to the algorithm shown in Figure 2A. Any two numbers can be added, which means there are an unlimited number of inputs into the addition algorithm. Inputs could be one digit, a million digits, or a billion digits, for example. The most efficient computations are like adding numbers: the number of operations required in the algorithm is *proportional to* the length of the input numbers. This means that, using the algorithm described in Figure 2A, if we had to add numbers that were twice as long as the numbers shown, we would have to perform twice as many operations to get the final result.

### ALGORITHM

A collection of actions that are performed on the input to get to the computation result in a finite number of steps.

**Figure 2**

Computational complexity. **(A)** An easy algorithm for adding numbers. The number of required operations is proportional to the length of numbers being added. If we increase the numbers from 2 to 4 digits, then twice as many operations are needed. **(B)** An algorithm to compute all the possible combinations of pizza toppings is exponential. The number of operations is proportional to 2 to the power of the input's length. If there are three possible toppings (inputs), the number of required operations is  $2^3$ , meaning that there are eight possible topping combinations. For exponential algorithms, the number of required operations grows very quickly as the input length increases, so the computation times needed for these algorithms are only reasonable for small inputs.



**Figure 2**

But let us think about a slightly more difficult problem: multiplying numbers. In school, you were probably taught an algorithm for this, too. If you use the algorithm shown in [this video](#), if you want to multiply numbers that are twice as long, the number of operations you must perform will be 4-fold greater. For example, if we multiply two two-digit numbers, we perform four multiplication operations and a few more addition operations. However, if the numbers have ten digits, we must perform 100 multiplication operations! If we use the algorithm shown for addition, only 10 operations would be required for 10-digit numbers. If the numbers have 100 digits, we need to perform

about 10,000 operations for multiplication as opposed to about 100 operations for addition. So, in the case of multiplication, the number of operations is *proportional to the square* of the numbers' lengths.

The efficiency of computations is very important because it defines which problems we can solve within a reasonable time and which we cannot, and the cost (in effort and time) required for each computation. There are many problems we must solve quickly so that the answer can be available to us almost immediately (for example, like Waze, which should instantly tell us which route to choose), and there are some problems we want to solve within a reasonable time (not necessarily immediately) to find answers to important questions in science or engineering, for instance. As you saw in the addition and multiplication examples, the difference between the efficiency of an algorithm that is proportional to the length of the input and one that is proportional to the *squared* length of the input is large. Let us now think about a case in which the algorithm is proportional not to the squared length of the input, but to an even higher exponent—maybe to the fifth or sixth power. Or we could think about a case in which the algorithm's efficiency is exponential, meaning proportional to two in the power of the input's length, or even a larger number in the power of the input's length. These cases require many computations to get to the solution.

Such complex problems *do* exist in our daily lives. To demonstrate the meaning of an exponential function, here is a tasty example—think about your favorite pizza. Let us say that the pizza restaurant had one possible topping—green olives. Then, you could choose between two options: a plain pizza or a pizza with green olives. Now, let us assume that the pizza place also has a corn topping. Then, you could choose between four options: plain, with green olives, with corn, or with green olives *and* corn. If there were three toppings (for example, mushrooms, olives, and tomatoes) you would have eight possibilities (Figure 2B). So, in this case, there are two to the power of possible topping options ( $2^3 = 8$ ). Therefore, the number of options for pizza topping combinations is exponential with respect to the number of toppings, which is basically the input length in the calculation. For larger inputs, like tens or hundreds of pizza toppings, the number of topping combinations quickly becomes enormous.

## EXPONENTIAL ALGORITHMS

The same exponential type of growth applies also for exponential algorithms. Now, instead of the length of input growing exponentially (like we saw with the number of possible pizza toppings), it is the *number of operations* in the algorithm can grow exponentially and is proportional to some number (say two, like in the pizza example) in the power of the input's length. Since the number of operations a computer can perform is finite—a billion operations per second,

say—we can translate the number of operation into the time it takes for computation. For an exponential algorithm and a 20-character-long input, the computation time for such a computer will be only one-millionth of a second. For a 50-character input, the computation will take more than 18 min, and for an 80-character input, it will take more than 3,800 years!

Since there are many computations we would like to be able to perform in a reasonable time, one major question in complexity theory deals with trying to find the most efficient algorithms to perform certain computations. For example, you may have learned a quicker way than the one we described for multiplying two numbers (you can watch an example [here](#)). In the same way, it might be possible to find more efficient ways to perform other computations for which we have only inefficient solutions. Researchers studying computational complexity theory search for these efficient algorithms and, to do so, they must also ask themselves how they know when they have found the best, most efficient algorithm. Sometimes they can prove that an algorithm is the best possible one for a particular problem, and other times they try to prove that there *is no* efficient way to solve a particular problem for every input, meaning that the problem is inherently hard.

## EASY AND HARD PROBLEMS TO SOLVE

Now let us think about dividing all problems into two groups depending on how difficult it is (i.e., how much time it takes) for a computer to solve them [2]. We will call one group P, for “polynomial”. **Problems in P** be solved quickly, because the number of computations needed to solve them is equal to the input’s length to some power. For example, when adding numbers according to our algorithm, the number of operations required is a polynomial of the first power with respect to the input’s length. We can think of problems in P as all the problems that humanity can solve or has already solved.

The second group of problems is called NP, for “nondeterministic polynomial”. This is the collection of problems that have not been solved yet, but that humanity would like to solve. Real-life **problems in NP** include, for example, all the problems that scientists think about, all mathematical proofs that mathematicians are trying to validate, and engineering problems such as planning a bridge that spans a river (Figure 3A).

Sometimes a problem might be difficult to solve, but once a solution is found, checking to see if that solution is correct might be relatively easy. Imagine a strange navigation app that needs to find the *longest possible route* between two points, instead of the shortest. This is a hard problem—finding the longest possible route between two points is much more complex than finding the shortest route. However, if we

### PROBLEMS IN P

Computational problems whose solution is easy (polynomial with respect to the inputs’ length).

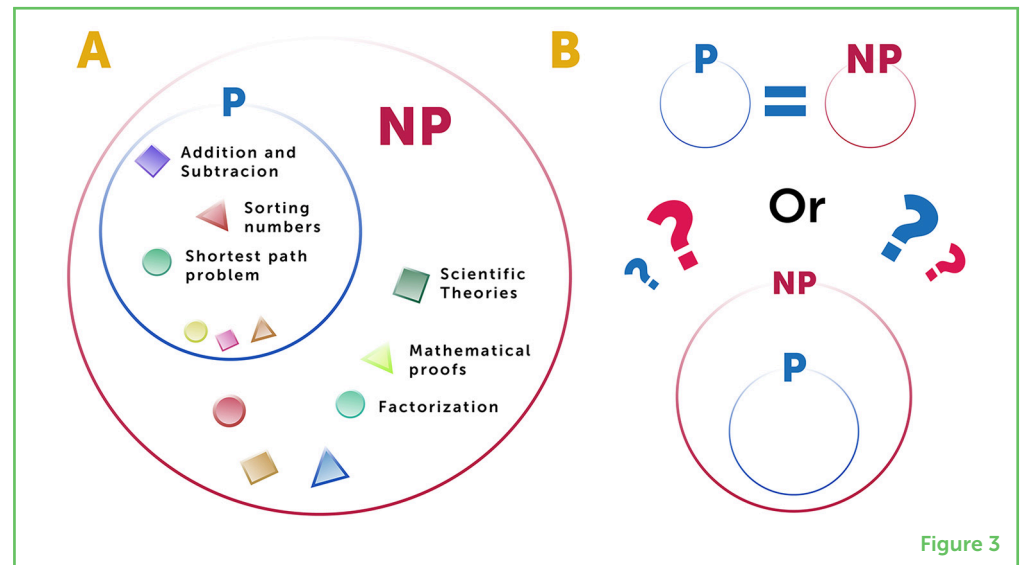
### PROBLEMS IN NP

Computational problems for which checking their solution is a problem in P. Such algorithm does not provide the solution but only makes sure that a suggested solution is correct.



**Figure 3**

Problems in P and NP—are they equal? **(A)** Currently, problems can be divided into two sets. P contains problems that can be solved efficiently and NP contains problems whose solutions can be checked efficiently. **(B)** A very important question within complexity theory is whether for all problems in NP there is an efficient solution algorithm. This would mean that P and NP are equal. Alternatively, NP could include all problems that are in P but also include additional problems that are not in P, that cannot be solved efficiently.

**Figure 3**

were *given* all the possible routes between two points, it would be easy to check which route is the longest, just by comparing them. In other words, even if the computation of the original problem is complex, checking a suggested solution is easy. As another example, if we try to find all the factors of a very large number (meaning, numbers that can be multiplied together to get that large number) this could be a hard problem. In contrast, if we just need to check whether two numbers are factors of a large number, this is an easy problem—we only need to multiply them together to check if the solution equals that large number.

One of the biggest open questions in computational complexity theory is whether the classes of P and NP problems are equal or whether P is a subgroup of NP (Figure 3B). In other words, if we can easily check whether a given solution to a problem is correct, does this definitely mean there is an algorithm that solves that problem efficiently, even if we have not found it yet? We *do* have examples of problems in NP for which we have not found efficient solutions—but this does not mean that such algorithms do not exist—maybe we just have not found them. If they do exist and we will be able to find them, this might be humanity's biggest dream coming true. On the other hand, there are some problems that we hope are too hard to solve (i.e., not in P). For example, the safety of data-encryption systems and other electronic safety system like those used for online shopping are based on extremely complex calculations that we hope are too hard to ever solve efficiently. If someone finds an efficient algorithm to solve these calculations, all of our information-protection systems will break down and the information that is currently safely encrypted will no longer be safe. So, you can see that there are reaching implications to the questions whether P and NP are equal classes of problems.

## RECOMMENDATIONS FOR YOUNG MINDS

Our main recommendation is to find out what you love—what your passion is. People do their best and most significant things when they believe in the importance of what they are doing, and often what is important for them is also fun for them. You might have a strong motivation to do things you do not necessarily enjoy all the time (for example, maybe you want to cure cancer but do not enjoy the lab work), but we think this option is not quite as good as finding something you really love.

If you enjoy problem solving, we recommend that you learn as much as possible and try to understand the kinds of problems that you like to face. Once you choose an area, try to be the best you can (Figure 4). An academic career is not for everyone, and some people might have more fun working in industry in companies that try to solve various problems. But if you *are* interested in an academic career, develop two important characteristics: an unstoppable thirst for knowledge and a love for sharing that knowledge through teaching and guiding students. Patience is also important, along with the ability to handle a competitive environment in a positive way.

**Figure 4**

Recommendations for Young Minds. Learn a lot and discover the kinds of intellectual problems you like to face. If these problems are in mathematics, remember that many mysteries you try to crack will include moments of “failure,” in which you will not solve the problem or even know whether you are progressing toward the solution. Remember that these “failures” give you extremely valuable insights and experience that will help you solve other problems in the future.



**Figure 4**

If you are specifically interested in mathematical research, remember that there are connections between sub-fields that supposedly do not “talk” with each other. These connections are critical, so it is important to have knowledge of as many fields of math as possible. In addition, mathematicians often try to solve problems that no one has solved before, so if you follow that path there is a chance that you will not solve them, either. That means that you might invest a lot of time on things that are ultimately unsuccessful. But if you enjoy the problem-solving process, then even if you have no promise that your solution will work, or even if your progress seems extremely slow, you will still be happy. In other words, good math researchers must

love the process of solving problems, independent of how successful the process is. Finally, it is important to remember that researchers are always learning from the ideas that they try, even if they are not successful in solving the problem. The insights they gain in the process stay with them and can help them to solve future problems.

## AUTHOR'S NOTE

This article is based on an interview between the two authors. It is written in the artistic expression of NS and is backed up scientifically by AW.

## ACKNOWLEDGMENTS

We wish to thank [Alex Bernstein](#) for providing the figures.

## REFERENCES

1. Levinthal, C. 1968. Are there pathways for protein folding? *J. Chim. Phys.* 65:44–5.
2. Wigderson, A. 2019. *Mathematics and Computation*. Princeton, NJ: Princeton University Press.

**SUBMITTED:** 28 August 2023; **ACCEPTED:** 27 November 2023;

**PUBLISHED ONLINE:** 25 January 2024.

**EDITOR:** [Jeremy L. Martin](#), University of Kansas, United States

**SCIENCE MENTORS:** [Kostas Karpouzis](#) and [Carol Jagger](#)

**CITATION:** Segev N and Wigderson A (2024) Easy Or Hard? Basic Questions in Computational Complexity Theory. *Front. Young Minds* 11:1284284. doi: 10.3389/frym.2023.1284284

**CONFLICT OF INTEREST:** NS declared that they were an employee of Frontiers, at the time of submission. This had no impact on the peer review process and the final decision.

The remaining author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

**COPYRIGHT** © 2024 Segev and Wigderson. This is an open-access article distributed under the terms of the [Creative Commons Attribution License \(CC BY\)](#). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication

in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

## YOUNG REVIEWERS

### CHRIS, AGE: 15

Chris is a math and Brooklyn 99 aficionado and in his spare time hunts down Minecraft zombies and creepers. He wants to study Computer Science and eventually become a game developer, mostly to justify still playing computer games as an adult.

### LORNA, AGE: 15

My name is Lorna, and I am doing this young reviewer alone, apart from my mentor, Carol Jagger. I am currently in year 9, and enjoy playing sports (swimming, netball and volleyball), and hanging out with my friends.

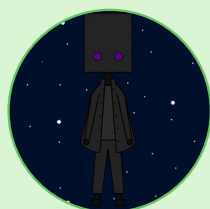
## AUTHORS

### NOA SEGEV

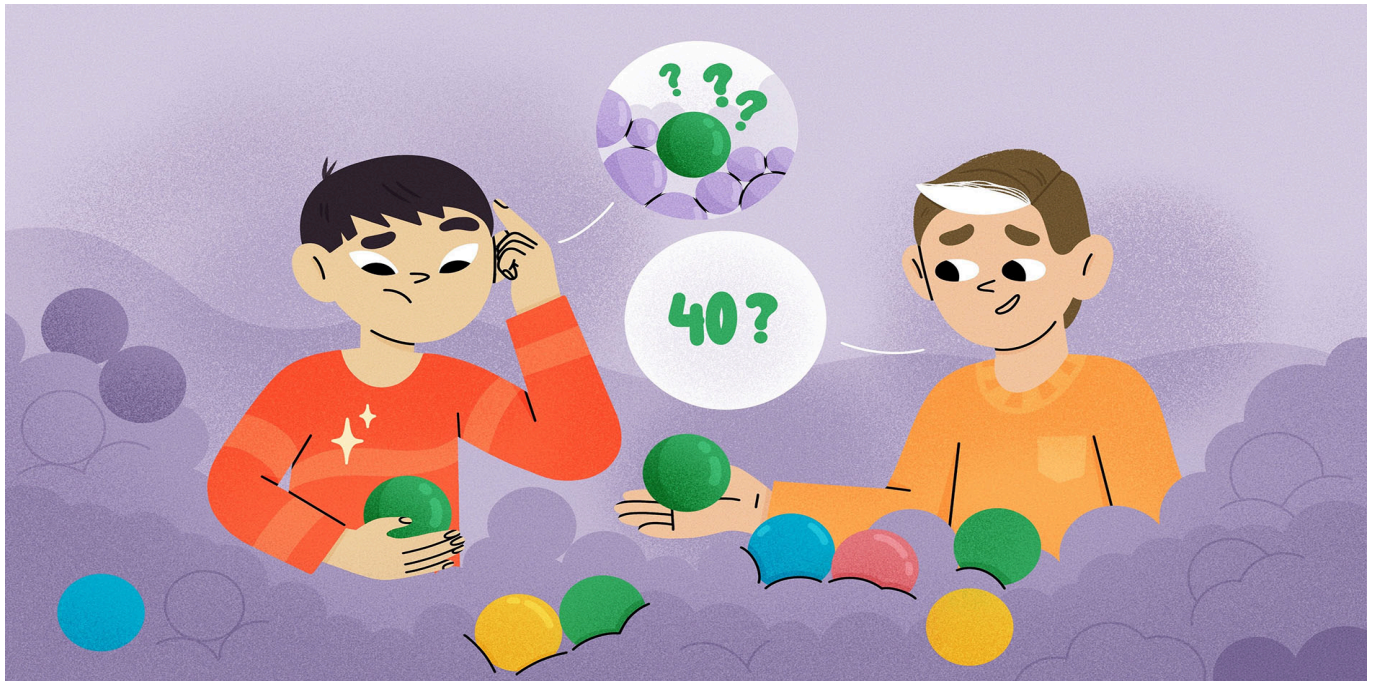
Noa Segev is a scientific writer and project coordinator at Frontiers for Young Minds. She earned her B.Sc. in physics at The Hebrew University of Jerusalem and her M.E. in renewable energy engineering at the Technion—Israel Institute of Technology. Since 2019, she has been interviewing winners of prestigious scientific prizes, including the Nobel Prize, and co-authoring articles with them for Frontiers for Young Minds. Noa aims to make the science behind important scientific discoveries accessible to all, and to share valuable insights from the vast professional and personal experience of world-class scientists. \*[noasegev@gmail.com](mailto:noasegev@gmail.com)

### AVI WIGDERSON

Prof. Avi Wigderson is a researcher in the field of computational complexity theory in computer science and a faculty member in the Institute for Advanced Studies at Princeton University. Wigderson completed his B.Sc. in computer science at the Technion—Israel Institute of Technology in 1980 and continued to do his advanced studies at the Princeton University in computation complexity and combinatorics. In 1986, Wigderson joined the Hebrew University of Jerusalem, Israel, where he was a faculty member for 13 years. In 1999, Wigderson joined Princeton University as a faculty member in the Institute for Advanced Studies, where he currently works. During his career, Wigderson studied varied questions and significantly contributed to many areas in computational complexity theory, including graph theory, P vs. NP problems, randomness in computations, and zero-knowledge proofs. Wigderson won numerous important awards including the Nevanlinna Prize (1994), Gödel Prize (2009), Knuth Prize (2019), and Abel Prize (2021).







# COMBINATORICS: THE MATHEMATICS OF FAIR THIEVES AND SOPHISTICATED FORGETTERS

Noga Alon<sup>1,2\*</sup>

<sup>1</sup>Department of Mathematics, Princeton University, Princeton, NJ, United States

<sup>2</sup>Department of Mathematics, Tel Aviv University, Tel Aviv, Israel

## YOUNG REVIEWERS:



MAOR

AGE: 13

Mathematics is the most logical and unbiased scientific field there is. What is true in mathematics today will forever stay true, and mathematical arguments will always clearly indicate which side “wins”. These are some of the characteristics of mathematics that I have liked very much since I was young. In this article, I will try to demonstrate the beauty of mathematics and show you that it appears in many situations in our daily lives—including places we might not expect. Then, I will give you a taste of two studies I conducted in a mathematical field called combinatorics, and I will explain how they are related to everyday problems. One of these problems is how to perform accurate mathematical analyses of large streams of data that we cannot save in a computer’s memory. Finally, I will try to give you an idea of my life as a mathematician—the challenges and difficulties, alongside the immense enjoyment and satisfaction I derive from them. Join me to an extended dive into my work on the wonders of combinatorics.

Professor Noga Alon was awarded the 2005 Gödel Prize, jointly with Prof. Yossi Matias and Prof. Mario Szegedy, for the work that laid the foundations of the analysis of data streams using limited memory. In 2022, Prof. Alon was awarded the Shaw Prize in Mathematical Sciences, jointly with Prof. Ehud Hrushovski, for their remarkable contributions to discrete mathematics and model theory with interaction notably with algebraic geometry, topology, and computer sciences.

## INTRODUCTION TO THE BEAUTY OF MATHEMATICS

Mathematics is an academic discipline that can provoke strong feelings—some people love it, and some people hate it. Some of the people who love math recognize that love from a very young age. For example, I knew from the age of 8 or 9 that I wanted to be a mathematician. I do not think that everyone should like mathematics or be mathematicians (in fact, I do not think that humanity would do very well if most people were mathematicians). Still, mathematics contains much beauty, and I think that everyone can learn to see and appreciate it—from close up or from further away.

### OBJECTIVE

Not influenced by personal opinion and unable to be argued about.

For me, mathematics is primarily a logical and **objective** way of thinking, with which I try to understand various types of processes. One of the things I especially like about mathematics is its objectivity. In mathematical arguments, one side can convince the other side that its argument is right, and the other side will actually be convinced by the mathematics itself. This does not normally occur in other situations in which there is a difference of opinions. For example, think about political arguments—it is very rare for one side to convince the other side to change its mind. Mathematics also has a dimension of stability to it—what is true now will always stay true; and what is untrue will forever stay untrue. In other scientific disciplines, including physics, this is not necessarily true—theories describing reality may, and often do, change throughout time.

I also like the challenge that mathematics holds. As a mathematician, I try to solve puzzles. Sometimes, I know in advance that the puzzle has a solution and I try to find it. For other puzzles, I am not even sure that a solution exists. When encountering a puzzle, I quite often feel that I have no chance of solving it. But then I think about it a lot, try various strategies, and I suddenly experience an insight. I do not always know what sparked the insight, or why it happened when it did—sometimes half an hour after I start thinking about the problem, and other times months later—but suddenly I know something new that I did not know beforehand. I find this process of deciphering mathematical challenges very satisfactory.

Sometimes I even think of mathematics as an art because it has a special beauty. There are practical parts of mathematics that are

used in other scientific disciplines such as physics, chemistry, and biology, to explain our reality. Other parts of mathematics are more abstract and contain mathematical structures that do not necessarily have any direct connection to the reality we are familiar with; or sometimes such connections are only realized much later. However, in my view, the value of mathematics is independent from whether it describes something in our reality. Like abstract art, which does not deal with concrete physical objects but is nonetheless pleasant to look at, abstract math has its own beauty.

## MATH APPEARS IN SURPRISING PLACES

### SOCIOLOGIST

A person who studies society, including both the groups and individuals that make it up and society as a whole.

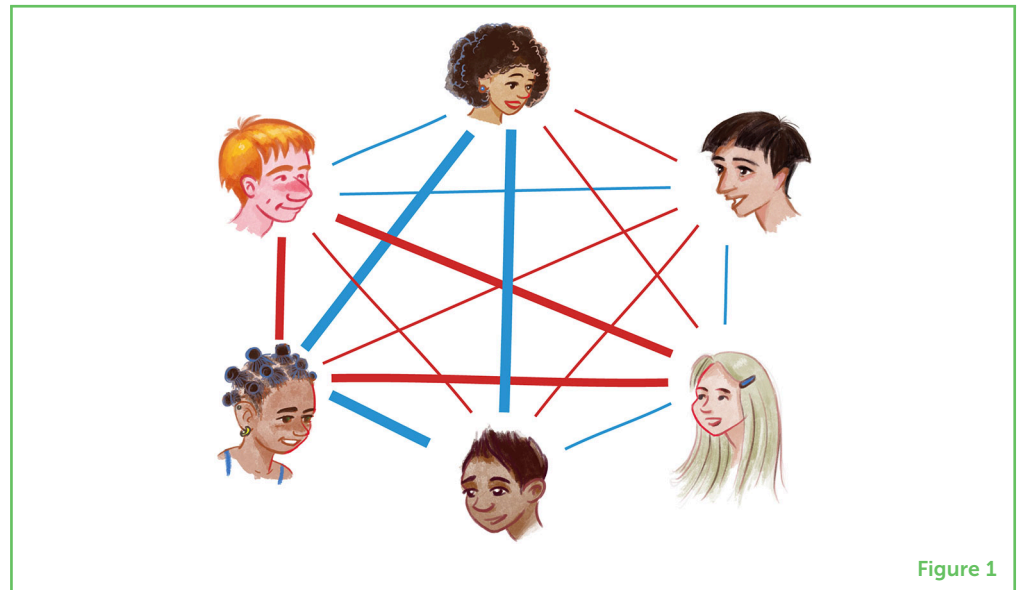
Mathematics is not limited to problems we solve on paper in algebra and geometry classes. Often, it appears in the midst of everyday phenomena. As an example, let me tell you the story of a Hungarian **sociologist** named Sandor Szalai, who lived in the twentieth century. Szalai conducted experiments on friendships between children. He asked every pair of children whether they were friends, where the relation was symmetric (meaning that both members of the pair said they were friends or both said they were not friends; there was no situation in which one said “yes” and the other said “no”). Szalai discovered that, each time there were at least 18 children in a group, there were always four children for which each pair (six pairs overall) were all friends, or not friends. Szalai thought this was a new and exciting sociological discovery. But, after talking to his mathematician colleagues, it became clear that this was not a sociological discovery but rather a mathematical fact! The situation Szalai was studying is a special case of a mathematical theory called Ramsey’s theorem. Ramsey’s theorem claims that, in general situations, it is not possible to have complete disorder—every system that is large enough contains a sub-system that is very organized.

Using our example, we can represent every kid as a dot on paper, and connect every two dots with a blue line to represent a friendship, or a red line to represent no friendship. In a group of 18 children, we will get 18 points among which every two points are connected by a blue or a red line. According to Ramsey’s theorem, no matter how we color these lines, there will always be four dots for which all the lines are colored in red or four dots for which all lines are colored in blue. So, this result is a mathematical one, not a sociological one.

Try to see what happens in a similar situation with six children. Connect every pair of six dots on paper using blue and red lines. You will find that, no matter which color lines you use, you will get a blue triangle or a red triangle, or both blue and red triangles (**Figure 1**). The curious amongst you can try to prove that, if you take five children, this is not the case—you do not always end up with a blue or a red triangle.

**Figure 1**

Social mathematics. If we look at six children, for which every pair can be friends (blue line) or not friends (red line), we will necessarily get three children that are all friends (blue triangle) and/or three children that are not friends (red triangle). This is a mathematical result that influences social structures.

**Figure 1**

## COMBINATORICS—TWO EXAMPLES ON THIEVES AND FORGETTERS

### COMBINATORICS

An area of mathematics that studies the properties of finite groups of objects.

One of the main mathematical fields I study is called **combinatorics**. Combinatorics is the mathematics of finite structures (opposed to continuous mathematics, which deals with infinite structures) [1]. For simplicity, think about these structures as collections of objects. In combinatorics, we deal with finite groups of objects and we try to understand their properties. In the previous example of the children, a group is the collection of all points (children), some of which are connected with blue lines and others with red lines. Classical combinatorics deals with questions like: how many ways are there to organize 10 children if we line them up one next to the other in one line? Or: how many options are there to choose 10 children out of a group of 50? We can also ask more complex questions, like: can we choose seven triples from a group of seven children, such that every possible pair of triples will have exactly one shared kid between them, and every two kids in the group will be contained in exactly one triple? Another example of a combinatorial question on a structure that you might know is related to Sudoku. We could, for example, ask how many possible unique Sudoku games could be made for  $30 \times 30$  Sudoku board?

In modern combinatorics, we often deal with problems in which we must determine the largest or smallest size of a combinatorial structure that satisfies certain properties. These questions have many applications in computer science, and in other areas of mathematics, such as geometry and number theory.



## COMBINATORICS EXAMPLE 1: FAIR THIEVES

One of the problems I delved into in my career deals with the fair splitting of a necklace. Think of a situation in which two (mathematically skilled) thieves steal a necklace with two types of gems—diamonds and rubies (Figure 2). These thieves are fair, and they decide that they will each receive an equal number of diamonds and rubies. To distribute their treasure, they want to cut the necklace using the fewest possible cuts. What is the smallest number of cuts they must perform to split the necklace fairly between them?

**Figure 2**

Fair splitting of a necklace made of diamonds and rubies. Let us assume that two thieves are stealing a necklace made of diamonds and rubies, and they want to split the treasure evenly. What is the least number of cuts they could perform to split the treasure equally? Can you see that, in this example, one cut is sufficient?

**Figure 3**

Fair splitting of a necklace with two types of gems. **(A)** If a necklace has six rubies on the left and eight diamonds on the right, it is easy to see that two cuts (red vertical lines) are sufficient to split the necklace fairly. **(B)** If the same gems were randomly organized, one cut in the middle creates a situation in which there are too many rubies and too few diamonds on the left, and too few rubies and too many diamonds on the right. **(C)** If we shift the cut one gem to the right and add another cut after the first gem from the left, we add one diamond and remove one ruby from the middle section. In this situation, both sections contain three rubies and four diamonds. Thus, even for a randomly organized necklace, two cuts are sufficient to split the gems equally between two people.



Figure 2

If all the diamonds are located on one side of the necklace and all the rubies are on the other side, it is easy to see that two cuts are enough to split the treasure equally (Figure 3A). Now we will prove that, even if the gems are arranged randomly in the necklace, two cuts are still sufficient to split the treasure fairly. Let us assume that the necklace contains eight diamonds and six rubies. For a fair split, each thief must receive four diamonds and three rubies.

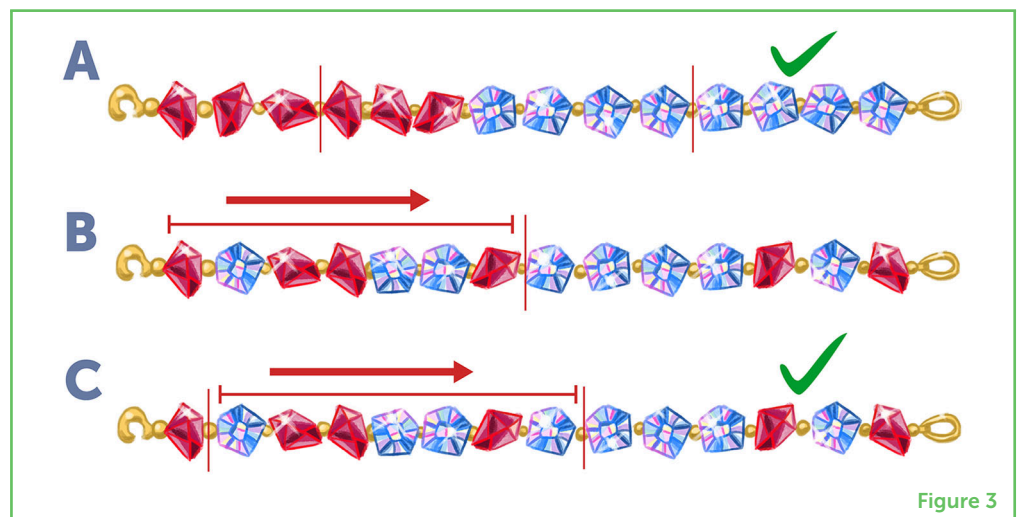


Figure 3

We will start with a situation in which the diamonds and rubies are randomly organized along the necklace (Figure 3B). First, we will cut the necklace in the middle and notice that we have too many rubies on the left side (four gems), and too few rubies on the right side (two gems). But if we shift our cut section one place to the right, we will gain one diamond from the right and lose one ruby from the left (Figure 3C). If we count the number of stones between the two red lines (the first thief's treasure) and outside the two red lines (the second thief's treasure), we will see that in both cases there is an equal number of gems: four diamonds and three rubies. Thus, we succeeded at splitting the necklace fairly using two cuts!

Now we will look at an example of a necklace with an equal number of gems of each type, organized randomly. We first choose a cut section between the end and the middle of the necklace (as in Figure 3B). Each time we shift our cut section to the right, we lose one gem on the left and gain one gem on the right, because the number of gems in both pieces of the necklace should stay equal. One of four options will happen when we shift the cut section:

1. We will lose a ruby on the left and gain a ruby on the right;
2. We will lose a diamond on the left and gain a diamond on the right (in both 1 and 2 the number of rubies in the cutting section will not change);
3. We will lose a ruby on the left and gain a diamond on the right (we will have one less ruby than we had before the shift); or
4. We will lose a diamond on the left and gain a ruby on the right (we will have one more ruby than we had before the shift).

You can see that, in each case, the number of rubies we have can change only by one after the shift. If we start from a situation in which we have too many rubies on the left side of the necklace, and we shift the cut section all the way to the right side of the necklace, we will inevitably end up with too few rubies—because we are back to the initial situation, in which we had too many rubies on the left and too few rubies on the right. If we start from a situation in which we have too many rubies on one side and end in a situation in which we have too few rubies on that side, when we change the situation by no more than one ruby each time, then surely at some point along the way the number of rubies on both sides will be equal (which will also make the number of diamonds equal on both sides). This is the result of a mathematical theorem called the discrete intermediate value theorem, which states that if something starts with a value that is too high, ends with a value that is too low, and shifts only a little bit each time, then along the way we will inevitably get the desired value.

For the general case of the same problem, we can think about some number of thieves, which we will call "k," and some number of gem types, which we will call "t." I proved that, in every case in which we

want to split a necklace with  $t$  types of gems equally between  $k$  thieves, it is sufficient to perform  $(k - 1) \bullet t$  cuts to split the treasure fairly [2]. In the simple case we previously saw, there were  $k = 2$  thieves and  $t = 2$  types of gems, and we could split the treasure using  $(2 - 1) \bullet 2 = 2$  cuts. In the same way, if there were four thieves and two types of gems, we could split the treasure fairly using  $(4 - 1) \bullet 2 = 6$  cuts (A more detailed example of the case of four thieves and two gem types, and a further explanation of the proof for the general case can be seen in [this video](#)). It is important to understand that the proof for the general case is complex. It is not derived simply and directly from the proof used for the cases of two or four thieves. This is often the case in mathematics—it is not enough to think logically and make intelligent claims; we must also use methods and tools that require a lot of knowledge and understanding. In our case, the proof for fairly splitting a necklace requires the use of advanced mathematical methods, including tools from a mathematical field called **topology**.

## TOPOLOGY

A branch of mathematics dealing with conserved properties of space.

The problem I described using the fair splitting of a necklace also applies to everyday situations. For example, think about cutting a long cake that has various parts—chocolate, sweets, and other surprises—and we want to split it among several people such that each will get the same components of the cake. Or, we can think about a situation in which a piece of land must be split between several people, and the land has a mixture of uses (agricultural land, natural land, and land for construction). In this case, we want each person to get the biggest possible plot of land while also equally dividing up the land in terms of usage types.

Splitting of resources is another example. In this case, we might have a collection of resources to be split between people, but each person has a unique preference and opinion about the value of each resource. Our aim is to split the resources between the people such that everyone feels like they received an equal share. For example, one person will get the share of resources they prefer, which might be undesirable to another person. In this way, no one will envy the others, and everyone will be pleased about what they got. These questions, among others, are studied using a mathematical area called **algorithmic game theory**.

## ALGORITHMIC GAME THEORY

A combination of game theory, which studies interactions between factors with different interests, and computational theory, which deals with models of computation and algorithms for performing computations.

## COMBINATORICS EXAMPLE 2: SOPHISTICATED FORGETTERS

Another problem I studied is an interesting issue in computer science, related to algorithms for dealing with data streams. In such cases, we receive a large collection of data, so big that we cannot save it in a computers' memory, but can only read it as it passes through the computer. One of the questions I studied was: which statistical properties of these data can we know, despite not being able to save it [3]? Without saving the data, there are some things we can know about

it and other things we cannot know—and this is not always evident at first glance. Here is an example: I will tell you all the numbers between 1 and 100 in a random order, except one number, which I will omit on purpose. You cannot write the numbers down as I say them, and you will not be able to remember all of them. Is there a way that you could tell me which number I omitted, without remembering the numbers themselves? Take some time to think about this.

The answer is yes—you can find out which number was omitted without remembering all the numbers. Here is how: for every number I tell you, add it to the previous number I mentioned, and you will get a sum. If I tell you 7 and then 8, you sum  $7 + 8$  and remember 15. Then I tell you 22, you add it to 15, and remember 37, and so on. At the end of this process, you will get the sum of all the numbers I told you. You can subtract this sum from the sum of all the numbers between 1 and 100 [which you can calculate relatively easily as the sum of 50 pairs, each having a value of 101 ( $100 + 1, 99 + 2, 98 + 3, \dots 50 + 51$ )]. This means the sum of all numbers from 1 to 100 is  $50 \bullet 101 = 5,050$ . The difference you get when you subtract the sum of the numbers I told you from 5,050 is exactly the number I omitted from the series! In this case, you were able to answer an important question related to the data you received, without having to remember those data.

The questions I study relate to very large collections of numbers—even trillions of them (one trillion is a million millions, or 1,000,000,000,000). Such large collections of numbers cannot be stored in a computers' memory, so we must find out what we can know about them without saving them. For example, if we want to know whether there is a number that appears twice in the collection, we cannot know this without remembering all the numbers. But if we want to know approximately how many distinct numbers are in the collection, there is a way we can do it without remembering the numbers (Figure 4) [3]. I will not elaborate on how to do so, since it is a bit complicated, but I will mention that we can estimate it in a way that gives us a result that is very likely to be very close to the actual result. For example, it is possible that, with a probability of 99% (meaning 99 cases of out 100) we will get a result that is  $<1\%$  incorrect compared to the real result; and with 1% probability (meaning 1 case out of 100) we will get a result that is not close to the real result. As you see, the mathematics I study allows us to get to surprising and unpredictable results, and that is part of its beauty—it allows us to expand our ideas beyond what we originally thought to be true.

These studies laid the foundation for an organized mathematical analysis of large data streams (called big data) using limited computer memory. This way, users can extract their desired data with very high accuracy, without having to save all the numbers. These studies also have important implications for several other areas of mathematics, including **information theory** and **graph theory**.

### INFORMATION THEORY

A branch of mathematics that studies the transfer of information between a source and a target.

### GRAPH THEORY

An area of mathematics dealing with graphs—combinatorial structures appearing in many contexts.



**Figure 4**

Algorithms for dealing with data streams. There are situations in which we must deal with immense amounts of data, and we are not capable of saving it all in a computer—like constantly streaming data on the internet (illustrated here as the flow of many balls). To conclude something important about these data without saving them, we must use different mathematical “tricks” (bottom right) that allow us, with high probability, to get a result that is close to the actual result that we are unable to compute accurately without saving the data.

**Figure 4**

## RECOMMENDATIONS FOR YOUNG MINDS

### A Mathematician's Life

For people who are not mathematicians, it is hard to imagine what the life of a mathematician looks like. Even as an undergraduate in mathematics, the meaning of conducting mathematical research was not clear to me. As a mathematician at a university, I must fulfill all sorts of demands and requirements: teach courses, mentor students, and publish papers. Sometimes I must write letters of recommendation for my colleagues, or review articles written by other mathematicians dealing with areas that fall within my expertise. When I arrive at the office in the morning, I open my computer and read emails. Often, people write to me with mathematical questions and ask if I can help or direct them to relevant resources. Sometimes the questions I receive are related to something I already know, or to something very similar to what I know—then I know how to answer them. Other times, I get questions about something I do not know but am interested in thinking about, and then I ponder the question for a while to see if any solutions come up.

At other times during the day, I think about mathematical problems related to my personal research. Sometimes I read articles that other people published on mathematical questions that I am interested in, and sometimes I sit and think about how to solve a certain open problem. In contrast to what non-mathematicians might think, direct work with equations and computations is *not* a big part of solving mathematical problems. The common image of a mathematician as someone who does long computations and eventually gets a result is not accurate. Most of the time, I am occupied with finding potential ways of solving a problem, and I ask myself questions like, “what sort of computation should be performed here?” or “what can advance my

understanding of this problem?" Sometimes, at the end of the process, there will be an equation that I will need to solve or a calculation that I must perform, but that is only a small part of the process—it is often the product of an idea that occurred to me first.

It is important to note that researchers—both mathematical and otherwise—often experience frustration from failing to solve the problems we are trying to solve. If we succeed at solving our problems too frequently, that means our problems are too simple and not interesting enough! This ongoing lack of success, which is a substantial part of good research, is often accompanied by nagging thoughts such as "this was probably the last time I could solve something... from now on I will not succeed." My experience proves that this thinking is not true, and part of the confidence you build along the way comes from learning how to avoid believing these self-doubting thoughts. Part of my willingness to deal with the struggles of mathematical research comes from the hope that I will successfully understand something I did not understand before, or solve a problem that has never been solved. I learned to enjoy the challenges in this process—the moments I do not enjoy are important because they make the enjoyment more meaningful and satisfying when it happens. Also, when I feel "stuck" in my work, I make sure to do something to change my internal state, so that I am more likely to have new and different ideas. Sometimes I will choose to think about another mathematical problem, and sometimes I will do something else entirely—read a book, watch a movie, or play a sport. Occasionally, a new idea will be generated from something I hear or read. Most times, I do not know what sparked my new ideas. Although I do not know how to track the sources of my ideas, my experience has taught me that changing what I am doing frequently supports the appearance of new ideas.

### **On Young Mathematicians**

It might interest you to know that young researchers have a certain advantage over experienced researchers, because they are less hesitant about investigating hard problems that many people have failed to solve. As an experienced researcher, I know of a variety of unsolved mathematical problems, and I know how to estimate their level of difficulty. On one hand, a beginning student might not know the history of a certain mathematical problem and not know how to estimate how difficult it is. Sometimes this lack of knowledge might serve them, and they eventually succeed at solving an important, previously unsolved problem. On the other hand, it is important to emphasize—especially to those of you who find school mathematics extremely easy—that you cannot do and know everything yourself. Even if you are very skilled and intelligent, you will still need to learn a lot of the knowledge accumulated by whole generations of smart mathematicians who spent much of their time developing mathematical methods. New students can acquire this knowledge by studying scientific literature, listening to lectures,

and having discussions with other mathematicians interested in the same areas.

### Thoughts About Prizes and Achievements

I have received quite a few prizes over the course of my career. Each time, I was happy to have my work recognized, and each prize proved to me that what I do interests other people as well. The prizes I received positively impacted my career and allowed me flexibility in my academic choices. Although I enjoy receiving prizes, I must emphasize that they are not the aim of my research. Prizes are not guaranteed—they are dependent on factors that I cannot control. In many cases, I can only recognize in retrospect the importance of the things I have worked on. My research on the properties of data streams is a good example—it turned out to be important later on, and it opened a new area of mathematics called streaming algorithms. My first aim in my research is to do what most interests me. This relates to my general recommendation: do what you love, and love what you do. These are two sides of the same coin, and both are important. Instead of doing something that does not interest you very much, thinking that it will eventually lead you to success, choose to do what you love to do. Sometimes it will lead to success and sometimes it will not, but at least you will spend your life engaged in work that is meaningful for you.

### ACKNOWLEDGMENTS

I wish to thank [Or Raphael](#) for conducting the interview which served as the basis for this paper and for co-authoring the paper and [Alex Bernstein](#) for providing the figures.

### REFERENCES

1. Van Lint, J. H., and Wilson, R. M. 2001. *A Course in Combinatorics*. Cambridge: Cambridge University Press.
2. Alon, N. 1987. Splitting necklaces. *Adv. Math.* 63, 247–53.
3. Alon, N., Matias, Y., and Szegedy, M. 1999. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.* 58:137–47.

**SUBMITTED:** 03 February 2023; **ACCEPTED:** 27 June 2023;

**PUBLISHED ONLINE:** 22 November 2023.

**EDITOR:** [Idan Segev](#), Hebrew University of Jerusalem, Israel

**SCIENCE MENTORS:** [Deborah Karaim](#)

**CITATION:** Alon N (2023) Combinatorics: The Mathematics of Fair Thieves and Sophisticated Forgetters. *Front. Young Minds* 11:1158338. doi: 10.3389/frym.2023.1158338

**CONFLICT OF INTEREST:** The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

**COPYRIGHT** © 2023 Alon. This is an open-access article distributed under the terms of the [Creative Commons Attribution License \(CC BY\)](#). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

## YOUNG REVIEWERS

### MAOR, AGE: 13

I really love learning about different topics, especially within social and sciences and humanities. I like reading (also non-fiction literature) and I write a lot (I won the national writing championship in third grade for a poem I wrote, and I write poems to this day). And I am really happy to take part in this initiative!



## AUTHORS

### NOGA ALON

Prof. Noga Alon is a professor of mathematics at Princeton University (New Jersey, United States) and an Emeritus Professor of mathematics and computer science at Tel Aviv University (Israel). His work focuses on combinatorics, graph theory, and their applications in computer science. Prof. Alon earned his B.Sc. in mathematics at the Technion, his M.Sc. in mathematics at Tel Aviv University, and his Ph.D. in mathematics at the Hebrew University of Jerusalem, Israel. He has held various senior positions in many research institutions in the United States, including Massachusetts Institute of Technology (MIT), Harvard University, The Institute for Advanced Study at Princeton, IBM Almaden Research Center, Bell Labs, and Microsoft Research. Prof. Alon joined Tel Aviv University as a faculty in 1985, and moved to Princeton University in 2018. He has served as the head of the school of mathematics at Tel Aviv University, and as the editor of many international scientific journals. He published one scientific book and more than 600 scientific papers. Prof. Alon has won numerous prizes, including the Erdos Prize in Mathematics (1989); The George Pólya Prize (2000); The Gödel Prize (2005); The Israel Prize in Mathematics (2008); The EMET Prize (2011); and the Shaw Prize (2022). He is married to Nurit and is the father of three daughters.  
\*[nalon@math.princeton.edu](mailto:nalon@math.princeton.edu)







# WAVELETS: MATHEMATICAL TOOLS FOR IMAGE ANALYSIS

**Ingrid Daubechies\***

*Departments of Mathematics and Electrical and Computer Engineering, Duke University, Durham, NC, United States*

## YOUNG REVIEWERS:



**ALEX**  
AGE: 14



**ANTHONY**  
AGE: 9



**MATTIA**  
AGE: 13

Mathematics is very human—we all enjoy using our brains, thinking, making connections, and checking to see if our conclusions are true. For me, mathematics is also a way to experience creativity, which is what makes me happy. In this article, I want to share this delight with you, through the story of wavelets. Wavelets are mathematical tools that my colleagues and I developed, which are very useful in analyzing images and other signals like medical scans or audio files. I hope that, by the end of this article, you will see that mathematics is much more interesting and diverse than memorizing formulas and performing calculations. It is a creative endeavor that advances society and can make our lives richer and more beautiful.

**Professor Ingrid Daubechies was awarded the 2023 Wolf Prize for work in wavelet theory and applied harmonic analysis.**

## THE MATHEMATICAL IMPULSE

Mathematics helps us figure things out using just our brains. To me, this is truly magical. I think that all of us have a mathematical impulse because we like using our brains and thinking, we like making connections, coming to conclusions, and checking whether those conclusions are true. This is exactly what I do in my everyday work as a mathematician. I try to understand how mathematical functions work, by taking them apart and putting them back together again. This disassembling and reassembling process happens in many other activities that require learning, too. Even great comedians like Chris Rock do it to improve their acts by breaking apart successful jokes and using their parts to create new jokes! In this article, I will tell you about an achievement made using this kind of learning process—a very useful mathematical tool called **wavelets**.

### WAVELETS

Mathematical functions used for processing signals and images. The word “wavelet” means a small wave.

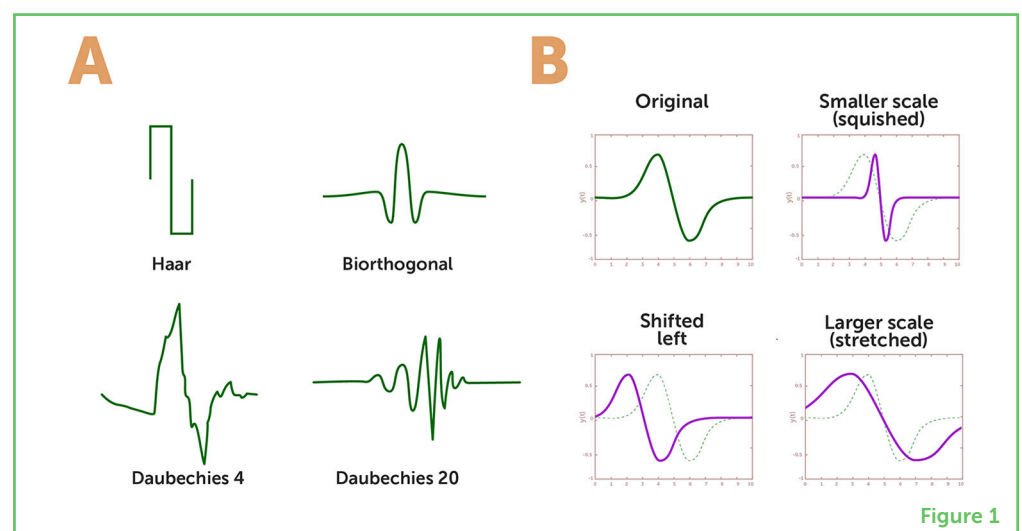
## WHAT ARE WAVELETS?

Wavelets are mathematical functions that help us process and analyze images and other kinds of signals [1], including medical recordings of the body’s activities and even **ripples in space itself**, created by the strongest astronomical events in the universe! (To learn more about wavelets and signal processing, see [this video](#)). In the 1980’s, mathematicians, engineers, and physicists all studied wavelets simultaneously. This was an exciting time, as it was clear that each field was looking at one part of a whole concept that would be extremely useful across multiple fields.

As their name suggests, wavelets are little waves that can come in various forms ([Figure 1A](#)). Wavelets are finite, meaning that their value is non-zero only in a specific area in space. Every wavelet can be stretched or squished to make it narrower or broader and can also be shifted to the left or right along the x-axis ([Figure 1B](#)), in order

### Figure 1

Families of wavelets. Wavelets are small waves that can come in various forms. **(A)** Four different types, or “families,” of wavelets which have different properties and are used for different applications. I developed the bottom two that are called after my name (Daubechies 4 and 20). **(B)** Every wavelet can be scaled to be smaller or larger (“squished” or “stretched;” see original wave in dotted green and new wave in purple). Wavelets can also be shifted along the x-axis (for example, shifted left as you can see in the bottom left).



to be matched to the analyzed signal. You can think of wavelets as building blocks with which we can reconstruct and analyze signals and images.

Once we have a signal or an image that we want to analyze, we can figure out which wavelets are most similar to the patterns in the signal (Figure 2). Importantly, we can adjust the scale to “zoom in” to see smaller details of the signal (squish the wavelet), or to “zoom out” to get a broader view (stretch the wavelet). Then, using some mathematical tricks, we can use wavelets to either get certain information out of the signal or image, like whether it contains a specific shape or pattern we are looking for, or **compress** that signal or image—meaning to save the image using only a small part of the original memory space, which still gives us enough information for our purpose.

### COMPRESS (DATA)

Save only the necessary part of the data.

### Figure 2

Using wavelets to detect patterns. We often use wavelets to analyze patterns in signals and images. In this artistic representation, the man is using wavelets to study the wave pattern in the painting *The Great Wave off Kanagawa* by Katsushika Hokusai. As you can see on the left, the man has frames with different wavelets and he is searching for the wavelet that would best suit the shape of the wave in the image.

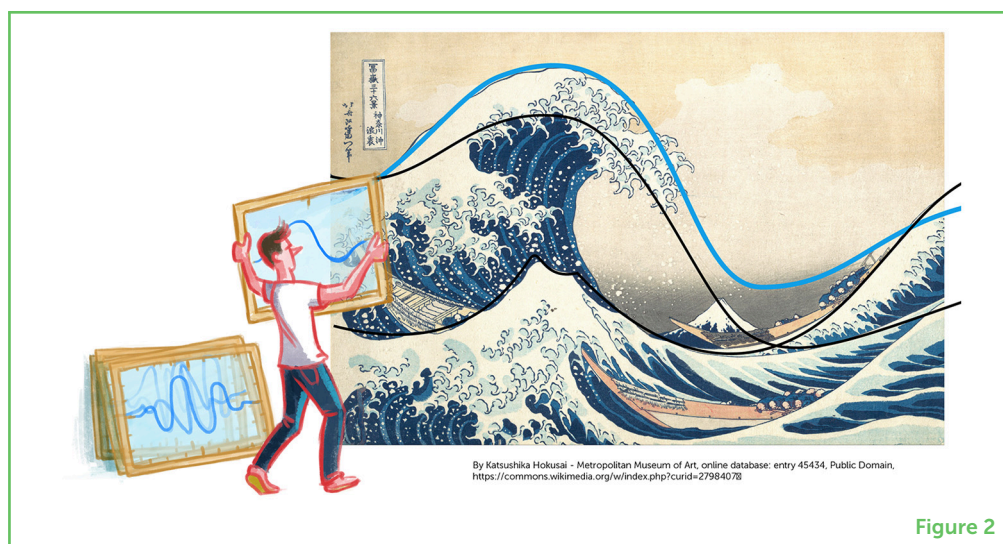


Figure 2

## WAVELETS AS TOOLS FOR IMAGE PROCESSING

As you may know, images are represented in computers as sets of numbers (Figure 3A). In a grayscale image, the color of each **pixel** is represented by one number, often in the range of 0–255 (Figure 3B). The number 0 represents the color black, and 255 represents white. All the numbers in between are various hues of gray (if the image is colorful, then we use three numbers to represent the three components of the color—red, green, and blue). Computers represent all information using only two states: “0” and “1”. Each “0” or “1” unit is called a bit, and each bit can represent the two states. When we combine two bits, since each of them can independently represent two states, both of them together can represent  $2^2 = 4$  states. If we want to represent 256 numbers (from 0 to 255), we need a system of 8 bits ( $2^8 = 256$ ). A typical rectangular digital image is formed of 512 rows, each having 512 pixels, so  $512 \times 512 = 262,144$  pixels overall. Each pixel is represented by 8 bits, so we need  $262,144 \times 8 = 2,097,152$

### PIXEL

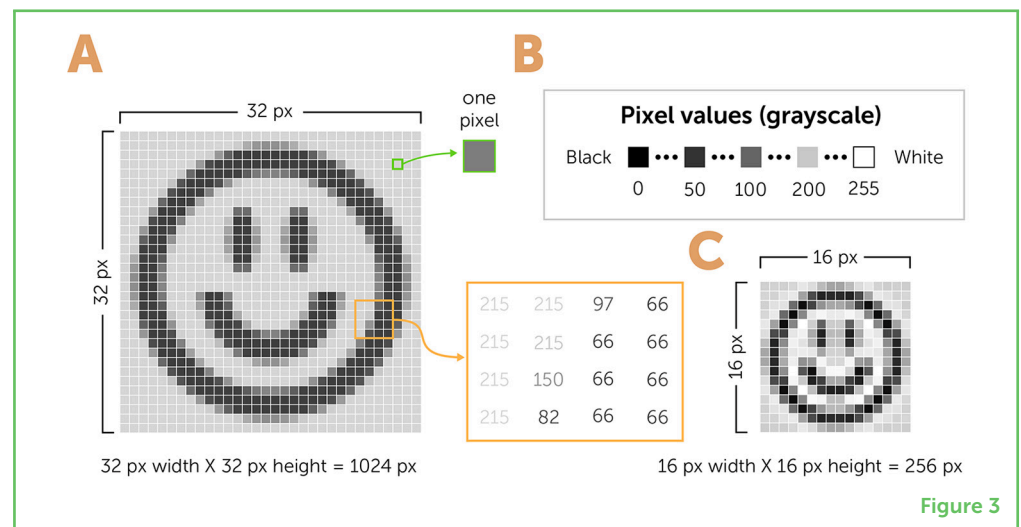
The smallest unit of color (tiny circle) in digital images.



bits, or units of information, to represent only one typical grayscale image (For colored images, we need three times as many)!

### Figure 3

Representing images in computers. **(A)** Images are represented in computers as sets of numbers—one number for every pixel (px). Lower numbers are darker and higher number are brighter (yellow square representing 16 pixels in the image). **(B)** In grayscale images, the numbers usually range from 0 (black) to 255 (white). **(C)** To save computer memory, we can compress images by averaging every two neighboring pixels to get one combined pixel with the average color value.



That is a huge amount of information, especially if we think about movies, in which we have 25 images per second, each represented by millions of bits. We must often find ways to decrease the amount of computer memory needed to store images, without losing important information. To do so, we can try to compress the image and reduce the number of pixels stored in memory. A simple way of doing this is to take every two neighboring pixels in an image, calculate the average number of their colors, and save only one pixel with that average color number (Figure 3C). By doing this, we decrease the number of pixels of an image by a factor of two horizontally (for neighboring pixels in every row) and by a factor of two vertically (for neighboring pixels in every column). We can repeat this process many times and produce a very small image.

When we do this averaging, we lose important information that was contained in the original image—information that might make a reconstructed image blurrier. If we do not want to lose that information, we must ensure that we can reconstruct the original image from the averaged image. To do that, we also save the differences between the original pixels that we averaged, because it is possible to find the original numbers from the average and the difference of the two numbers. It turns out that we can represent these averages and differences using wavelets! If we take the averages and differences only between two near neighbors, we get a staircase-like wavelet (Haar wavelet in Figure 1A). Often, we want to combine larger groups of neighboring pixels, which means each averaged pixel will contain more information. When we do that, we get other wavelets that are smoother (Daubechies 20 wavelet in Figure 1A).

Wavelets can show us the most important parts in our image, which will be areas of the image where “something is happening,” like



## IMAGE PROCESSING

Performing mathematical operations on digital images to get useful information out of them.

### JPEG2000

An image compression method based on Daubechies wavelets, which is widely used on the internet and in digital movies.

boundaries. Wavelets can give us information about general trends in the values of pixels in our image, and about the horizontal, vertical, and diagonal details or sharp changes in color. Wavelets are better at analyzing sharp changes in images (and other signals) compared to other common signal-analysis methods. Wavelets let us identify and save only important features of the image, where there are large changes, and discard information about uniform areas. The information we save about the important features is enough to allow us to reconstruct a fairly accurate version of the original image. Saving as little as 10% or even 3% of the original data can often be enough to reconstruct a pretty good version of the original image (see [this video](#) and check whether you can notice the differences between the various compressions).

I developed wavelets that are particularly useful for image analysis. While most scientists looked at known wavelets and tried to change them to make them useful for image analysis, I first looked at the image-analysis task at hand and asked: which properties should wavelets have to be useful for this task? Based on these properties, my colleagues and I constructed a completely new family of wavelets [2]. We then showed how wavelets can be used for **image processing** and compression [3]. Our suggestion turned out to be very useful, and it was applied in a method that is called the **JPEG2000** image compression standard [4, 5]. JPEG2000 is widely used in the compression of high-resolution images, in many applications on the internet and for digital movies.

## WAVELETS IN INTERESTING PLACES

Wavelets are a powerful tool for analyzing signals and images. In principle, they can be used in any situation for which signal analysis is required, including in the fields of astronomy [for automatic selection of the best-focused images of astronomical objects [6]]; forensics [for detecting manipulations of digital audio files, images, and videos [7]]; and medicine [for detecting heart diseases that cause irregular blood supply to the heart [8]]. With time, as we install more and more sensors in our environments and collect an increasing amount of data that we want to store, I expect that wavelets will have an even bigger role in many applications.

I am currently working with colleagues on using wavelets for interesting and perhaps quite surprising applications. In one of my projects, for example, I am working with art historians. We use wavelets to analyze paintings and understand their history. Specifically, we are trying to get the best possible view of hidden layers, called underpaintings, that are beneath the visible paintings ([Figure 4A](#)). After we have identified an underpainting using x-rays, we use wavelets to “expose” the details of the underpainting and reconstruct it as accurately as possible. We can also use wavelets for other applications

in art, such as studying and removing (or generating) cracks in digital images of paintings (Figure 4B) and rejuvenating aged colors.

### Figure 4

Applying wavelets to art. Wavelets can be used for varied applications, including art conservation and restoration. **(A)** Some paintings, such as *Patch of Grass* by Van Gogh (1), have other paintings beneath them, called underpaintings (2). We can uncover these underpaintings using x-rays, and then use wavelets to reconstruct the original image (3–4). **(B)** We can also use wavelets to remove cracks from digital images of old paintings (1–2) through performing wavelet analysis multiple times (3).

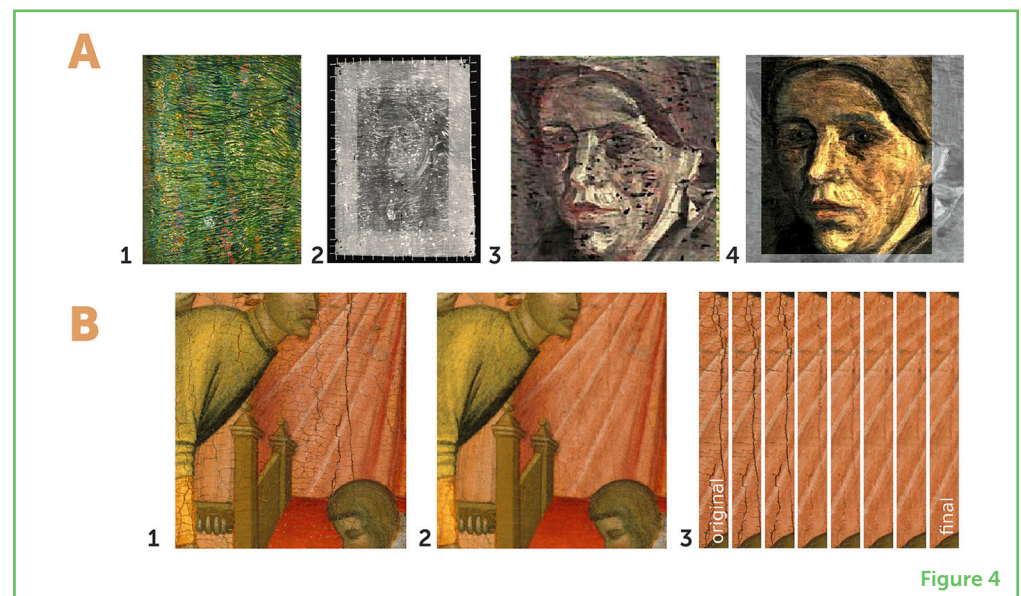


Figure 4

To end, I would like to share with you an important lesson I learned from working on mathematical problems. To solve a mathematical problem, you must try to solve it over and over again, approaching it from multiple “sides.” This is similar to sports, in which you have to work hard and make an effort to get better—but the fact that it is hard does not mean it is impossible. So, keep trying and keep using your creativity to find novel, beautiful ways to understand the things around you, and be sure to share your insights with others.

## ACKNOWLEDGMENTS

I wish to thank [Or Raphael](#) for conducting the interview which served as the basis for this paper and for co-authoring the paper, and [Alex Bernstein](#) for providing the figures.

## ADDITIONAL MATERIALS

1. Compression Applet—Using Wavelets to Compress Files (Duke University).
2. Using Math to Understand Art. Ingrid Daubechies. TEDxDuke.
3. Mathemalchemy—A unique and collaborative art exhibit exploring the beauty of mathematics.
4. How Wavelets Let Researchers Transform and Understand Data.

## REFERENCES

1. Daubechies, I. 1992. “Ten lectures on wavelets,” in *CBMS-NSF Regional Conferences Series in Applied Mathematics* (Philadelphia, PA: SIAM). p. vii.

2. Cohen, A., Daubechies, I., and Feauveau, J. C. 1992. Biorthogonal bases of compactly supported wavelets. *Commun. Pure Appl. Math.* 45:485–560.
3. Antonini, M., Barlaud, M., Mathieu, P., and Daubechies, I. 1992. Image coding using wavelet transform. *IEEE Trans. Image Process.* 1:205–20.
4. Usevitch, B. E. 2001. A tutorial on modern lossy wavelet image compression: foundations of JPEG 2000. *IEEE Sign. Process. Mag.* 18:22–35. doi: 10.1109/79.952803
5. Unser, M., and Blu, T. 2003. Mathematical properties of the JPEG2000 wavelet filters. *IEEE Trans. Image Process.* 12:1080–90. doi: 10.1109/TIP.2003.812329
6. Kautsky, J., Flusser, J., Zitova, B., and Šimberová, S. 2002. A new wavelet-based measure of image focus. *Pat. Recogn. Lett.* 23:1785–94. doi: 10.1016/S0167-8655(02)00152-6
7. Farid, H., and Lyu, S. 2003. "Higher-order wavelet statistics and their application to digital forensics," in *2003 Conference on Computer Vision and Pattern Recognition Workshop*, Vol. 8 (Madison, WI: IEEE). p. 94.
8. Akay, M. 1997. Wavelet applications in medicine. *IEEE Spectr.* 34:50–6.

**SUBMITTED:** 05 April 2023; **ACCEPTED:** 17 August 2023;

**PUBLISHED ONLINE:** 22 November 2023.

**EDITOR:** [Jeremy L. Martin](#), University of Kansas, United States

**SCIENCE MENTORS:** [Deepthi Kolady](#) and [Irene Bottillo](#)

**CITATION:** Daubechies I (2023) Wavelets: Mathematical Tools for Image Analysis. *Front. Young Minds* 11:1200611. doi: 10.3389/frym.2023.1200611

**CONFLICT OF INTEREST:** The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

**COPYRIGHT** © 2023 Daubechies. This is an open-access article distributed under the terms of the [Creative Commons Attribution License \(CC BY\)](#). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

## YOUNG REVIEWERS

### ALEX, AGE: 14

Alex is a 14-year-old 9th grader. He is very much interested in science, particularly in bioinformatics.



**ANTHONY, AGE: 9**

Anthony is a 9-year-old 4th grader. He is interested in robotics engineering and would like to make robots that help disabled people have a better quality of life.

**MATTIA, AGE: 13**

Mattia is a curious guy, and he loves learning and discovering new topics. He also loves playing basketball and watching movies. He loves so much everything he sees and studies, that he has no idea of what job will he do when he grows up.

**AUTHORS****INGRID DAUBECHIES**

Prof. Ingrid Daubechies is a James B. Duke Distinguished Professor of mathematics and electrical and computer engineering at Duke University (North Carolina, United States). Prof. Daubechies earned her B.Sc. and Ph.D. in physics at the Vrije Universiteit Brussel (Brussels, Belgium), where she continued her research until 1987. Then, she joined the Murray Hill AT&T Bell Laboratories (New Jersey, United States) until she joined Princeton University as a professor in 1994. In 2011, Prof. Daubechies moved to Duke University, where she is currently working. During her scientific career, Prof. Daubechies has won numerous prizes, including the Louis Empain Prize for Physics (1984), the NAS Award in Mathematics (2000), the Leroy P. Steele Prize (2011), the William Benter Prize in Applied Mathematics (2018), the L'Oréal-UNESCO For Women in Science Award (2019), and the Wolf Prize in Mathematics (2023).

\*[ingrid.daubechies@duke.edu](mailto:ingrid.daubechies@duke.edu)



# Frontiers for Young Minds

## Our Mission

To publish high-quality, clear science which inspires and directly engages the next generation of scientists and citizens, sharing the revelation that science CAN be accessible for all.

## How we do this?




Top researchers write up their discoveries for kids and our global network of Young Reviewers peer review every article, guided by their mentors, to ensure everything we publish is not only understandable but engaging for their peers aged 8-15.



## Discover our latest Collections

[See more →](#)

### Social Media

 @FrontYoungMinds  
 @FrontiersForYoungMinds  
 @frontiersyoungminds  
#frontiersforyoungminds

### Contact us

+41 21 510 1788  
kids@frontiersin.org

