# Mathematics of Neural Networks

Gabriel Peyré
CNRS & DMA
PSL, École Normale Supérieure
gabriel.peyre@ens.fr

# Chapter 1

# Discriminative Neural Networks

Since 2012, deep neural networks have revolutionized machine learning. Although relatively old, in recent years this technique has allowed very spectacular advances in the recognition of texts, sounds, images and videos. Understanding the stakes of these methods raises questions at the interfaces between mathematics and algorithmics. In this article, I will explain the structure of these networks as well as the key concepts of their supervised learning.

## 1.1 Algorithmics and mathematics of learning

Neural networks are algorithms, which compute, from an input $x$ (for example an image), an output $y$. As shown in figure 1.1, this output is most often a set of probabilities: for example the first output is the probability that the image contains a cat (the closer this number is to 100%, the more it means that the algorithm is sure of itself), the second is the probability that the image contains a dog, etc. To simplify, we will consider in our examples only two classes: cats and dogs, but in practice one can consider an output $y$ with several thousands of classes. We also restrict ourselves to the example of images, but neural networks are also very efficient in recognizing texts or videos.

Mathematically, such an algorithm defines a function $f_w$ (i.e. $y = f_w(x)$). The computer program that calculates this function is very simple: it is made up of a sequence of several stages, and each stage performs elementary calculations (additions, multiplications, and a maximum). In comparison, the computer programs found in a computer's operating system are much more complex. But what makes the huge difference between a "classical" algorithm and a neural network is that the latter depends on parameters, which are the weights of the neurons. Before using a neural network, these weights must be modified so that the algorithm can best solve the requested task. This is done using mathematical and algorithmic methods which will be explained in the following sections. This process is called "training" a neural network, and this requires a lot of time, machine calculations and energy.

Using these algorithms wisely therefore requires computer science and mathematics skills. It is thus necessary to manipulate key concepts in computer science (iterative methods, computation time, memory space, efficient implementation, . . . ) and mathematics (linear algebra, optimization, statistics, . . . ).

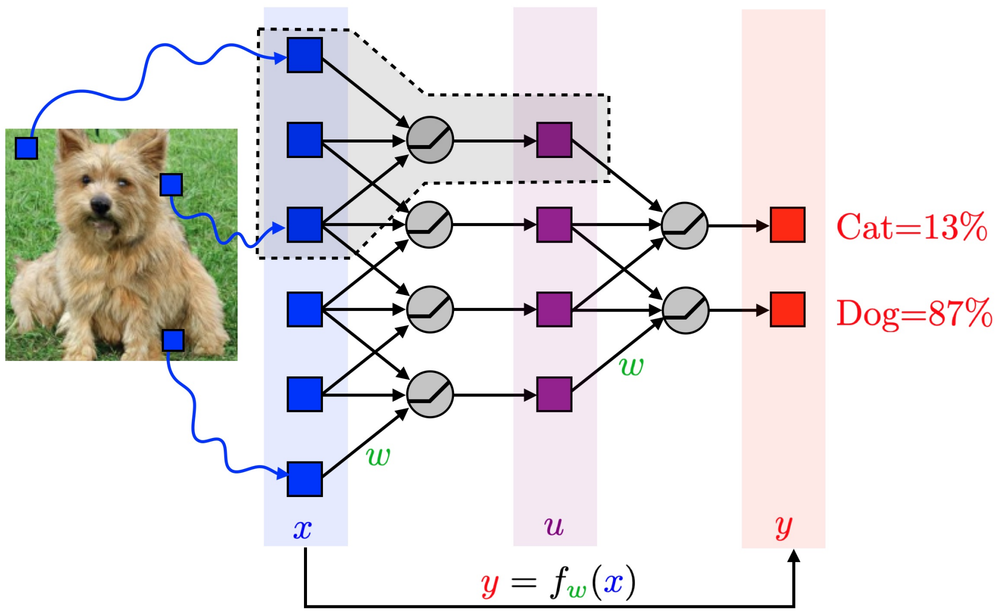

Figure 1.1: Example of a discriminative neural network with two layers.

## 1.2 Discriminative neural networks

An artificial neural network is built around a biological metaphor. The structure of the primary visual cortex is relatively well known, and Hubel and Wiesel won the Nobel Prize in Physiology for the discovery in 1962 of the organization of neurons in the first cortical layers [8]. Thus, in an extremely simplified view of the brain, neurons are organized in layers, each neuron retrieves information from a previous layer, performs a very simple calculation, and communicates its result to neurons in the next layer. It must however be keep in mind that this is only a metaphor and a source of inspiration: biological networks have much more complex connections and the mathematical equations which govern them are also more complex (they have been discovered by Alan Hodgkin and Andrew Huxley in 1952 [7] and they won the Nobel Prize for this). It therefore remains difficult to precisely relate the sometimes surprising performances of artificial neurons to the cognitive capacities of the brain. For example, the techniques to train artificial networks which I will now explain are quite different from the way a child learns.

Figure 1.1 details an example of such an artificial network. This type of neuron was introduced in 1943 by McCulloch and Pitts [14]. To simplify, it is here composed of only two layers of neurons (the first layer between $x$ and $u$, the second between $u$ and $y$), but today's most efficient networks can have several dozen layers, we say that they are deeper. In our example, the inputs $x$ are the pixels of an image. An image typically contains millions of pixels, and the figure voluntarily represents only a small number: a realistic network is indeed more complex. In addition, each pixel that makes up $x$ is actually made up of 3 values (one for each primary color red, green and blue).

Passing from one layer (for example the $x$ layer of inputs) to another (for example the second layer $u$, which is a "hidden" layer in the middle of the network) is done through a set of artificial neurons. A neuron is represented on the figure 1.2. It is the first neuron, the one that calculates the first value $u_1$ which composes the layer $u$. This neuron connects a certain number of elements

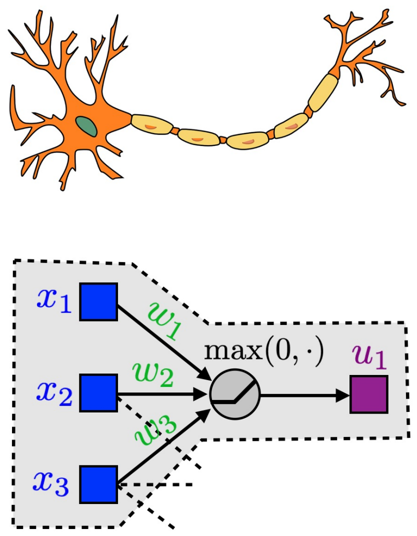

Figure 1.2: Biological and artificial neurons.

of the first layer (here three: $x_1, x_2, x_3$, but there can be more) to a single element of the second, so here $u_1$. The formula calculated by the neuron is

$$u_1 = \max(w_1 \times x_1 + w_2 \times x_2 + w_3 \times x_3 + w_4, 0).$$

The neuron thus performs a weighted sum of the three inputs, with three weights $w_1, w_2, w_3$, and we also add $w_4$, which is a bias. Then the neuron calculates the maximum between this sum and zero. One can also use a function other than the maximum function, but this one is the most popular. It is a thresholding operation. We can compare it to biological neurons which let or not pass information according to whether they are sufficiently excited or not. So if the weighted sum $w_1x_1 + w_2x_2 + w_3x_3 + w_4$ is smaller than 0, then the neuron returns the value $u_1 = 0$, otherwise it returns the value of this sum and places it in $u_1$.

Such neural networks were introduced by Rosenblatt [19] in 1957, who called them "perceptrons". The first perceptrons contained only one layer. Such single-layer architectures are too simple to be able to perform complex tasks. It is by adding several layers that we can calculate more complex functions. Deep neural networks thus use a very large number of layers. In recent years, these architectures have made it possible to obtain very impressive results for the recognition of images and videos as well as for the automatic translation of texts. It was this research on deep networks that enabled French researcher Yann Le Cun as well as Geoffrey Hinton and Yoshua Bengio [11] to obtain the Turing Prize in 2019. This prize is considered to be the equivalent of the Nobel Prize in computer science. To get a better understanding of the behavior of these multi-layered networks, one can use the interactive application `https://playground.tensorflow.org`.

## 1.3 Supervised learning of a neural network

Training a neural network consists in choosing the "best" possible weights of the set of neurons that make up a network (for example in particular the weights $w_1, w_2$ and $w_3$ of the neuron shown

Figure 1.3: Examples of images from the ImageNet database [5] used for learning.

in figure 1.2). It is thus necessary to choose the values of these weights in order to best solve the task studied on a set of training data. For the recognition of objects in images, this is a supervised learning problem: we have both images $x$ and the associated $y$ (the probability of the presence of a cat and/or a dog in the image). Figure 1.3 shows some examples of images used to train a network, for which we know what they contain (the class of cats and the class of dogs). It is therefore necessary, before the learning phase, that humans do a long and tedious job of labeling thousands or even millions of images.

The training procedure thus consists in modifying the weights $w$ such that for each $x$, the network $f_w$ predicts as precisely as possible the $y$ associated, that is to say at the end of the training that $y \approx f_w(x)$. A simple choice is to minimize the sum $E(w)$ of the squares of the errors, which we write mathematically as

$$\min_{w} E(w) = \sum_{(x,y)} (f_w(x) - y)^2.$$

This corresponds to an optimization problem, because it is necessary to find a set of parameters which optimizes a certain quantity of interest. This is a difficult problem, because there are a lot of parameters, and these parameters, especially those of the hidden layers, have a complicated influence on the result. Fortunately, there are efficient mathematical and algorithmic methods to solve efficiently this type of optimization problem. They are not yet fully understood on a theoretical level and it is a very active area of research. These optimization methods modify the weights $w$ of the network to improve it and reduce the training error $E(w)$. The mathematical rule for deciding on the weight update strategy is called back propagation [20] and is a marvel of ingenuity, it is a special case of a mathematical and algorithmic method which is called backwards automatic differentiation [13].

These supervised learning techniques date mainly from the 1980s. But it was only in 2012 that a paper by Krizhevsky, Sutskever and Hinton [10] created a tsunami by showing that deep networks can solve complex image recognition tasks. This revolution was possible thanks to the combination of three ingredients: new databases much larger than before [5] ; high computing power thanks to graphical processors ("GPUs", which were previously confined to video games) ; the introduction of several optimization techniques that improve and stabilize learning [21].

## 1.4 The efficiency of neural networks

George Cybenko proved in 1989 [3] that a neural network$f_w$ with two layers can approximate as precisely as one wants any continuous function $f^\star$ (so in some sense, it can solve any task, represented by the unknown function $f^\star$, which would be able to recognize objects in any image) as long as the size of the inner layer $u$ (so the number of neurons) is arbitrarily large. This does not mean that such a network $f_w$ with only two layers works well in practice. To apply Cybenko's theorem, it is necessary to have a potentially infinite number of training data, which is far from being the case in practice. The ultimate goal of learning is not to minimize the learning error $E(w)$, but to be able to predict as precisely as possible a new data. With only a limited amount of data, one will not be able to learn precisely enough, and therefore future predictions on unseen data will be bad. The function $f_w$ will actually be very far from the ideal function $f^\star$ that one would like to learn.

In order to make the best possible predictions with a limited number of training data, we are therefore looking for the most suitable network architectures, which can efficiently capture the information present in the data. Deep neural networks (with many layers) but with relatively few connections between layers have proven to be very effective on very "structured" data such as text, sound and images. For example, for an image, the pixels have neighborhood relationships, and one can impose specific connections (an architecture) and not connect a neuron with all the others but only with its neighbors (otherwise there would be too many connections). In addition, one can impose that the weights associated with a neuron are the same as those associated with another neuron. This type of networks is called convolutional networks [12]. For the moment, there is no mathematical analysis which explains this efficiency of deep convolutional networks. There is therefore a need for new mathematical advances to understand the behaviors and limitations of these deep networks.

# Chapter 2

# Generative Neural Networks

In the previous article, we saw how to train neural networks in a supervised manner. This makes it possible to effectively solve classification problems, for example image recognition. What is perhaps even more surprising is that these neural networks are also used in an unsupervised manner in order to automatically generate "virtual" texts or images, which are often called "deep fakes". In this second article, I will draw a link between the learning of generative neural networks and the theory of optimal transport. This problem was framed by Gaspard Monge in the 18th century, then it was reformulated by Leonid Kantorovitch in the middle of 20th century. It has now become a tool of choice to tackle important problems in data science.

## 2.1 Generative neural networks

Instead of using neural networks to analyze images, the paper [6] has shown in 2014 that they can be used "backwards" to generate images. These generative neural networks, for example, find applications for special effects, video games and artistic creation. Similar questions and methods can be found in the training of autonomous cars and to solve strategy games. Figure 2.1 shows the structure of such a network $g_w$, which depends on weights $w$. The layers somehow play mirror roles compared to the architecture of the discriminating neural networks exposed in the previous article. From an entry $y$ composed of a small number of values, which are typically drawn randomly, one generates an image $x = g_w(y)$.

The problem of training such networks is unsupervised: there is only a large number of training images, without indication of what they contain. There is no longer any need for human intervention to indicate to the network the content of the images which it must recognize. Data collection is thus simpler than for training discriminatory networks. In addition, this principle of unsupervised learning is close to the way children learn, mainly by observing and manipulating the world around them. The goal is then to select the weights $w$ of the neurons of the network $g_w$ so that the random images generated (the "fake" images) resemble as closely as possible the images of the training set.

## 2.2 Unsupervised learning of generative networks

The goal of generative neural networks is not to solve a task such as recognizing objects in images. In order to train the weights $w$ of the network, the problem must be formalized mathematically. This involves generating a set of "virtual" images (the false ones) that look like real images in a

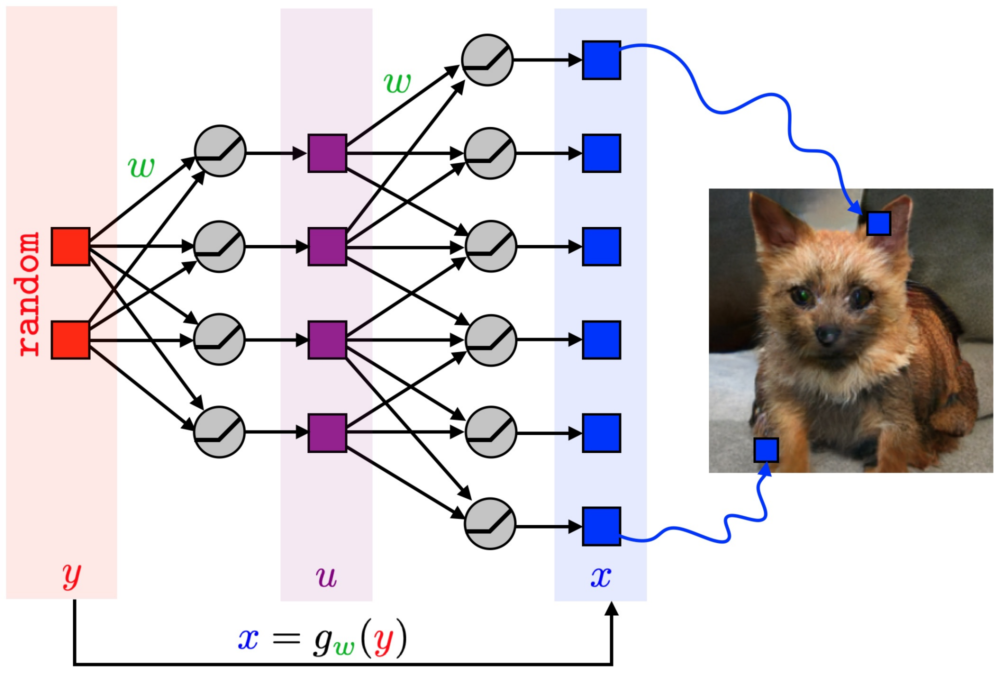

Figure 2.1: Example of a simplified generative neural network (a network for generating such complex images has more layers).

database. It is not simply that a generated image looks like a real image, it is necessary to match the two sets of images. For example, if in the database there are half images of dogs and half images of cats, the network must also generate half of dogs and half of cats.

We will write $\{z_1, z_2, \ldots, z_n\}$ the set of $n$ images in the database. The number $n$ of images is very large, of the order of several thousand or millions. Given a generative neural network $g_w$, which is parameterized by its weights $w$, we note $\{x_1, x_2, \ldots, x_n\}$ a set of $n$ "false" images randomly generated by the network. To generate the first false image $x_1$, this means that we randomly draw the input values $y_1$ and apply the network to these inputs, to obtain the virtual image $x_1 = g_w(y_1)$. We then do the same thing with $x_2 = g_w(y_2)$ and so on.

The goal of unsupervised learning is therefore to find weights $w$ so that the set of false images $\{x_1, \ldots, x_n\}$ is as close as possible to the set of images $\{z_1, \ldots, z_n\}$ of the database. The optimization problem is written thus

$$\min_w \text{Distance}(\{x_1, \ldots, x_n\}, \{z_1, \ldots, z_n\}).$$

It should here be remembered that the images generated $\{x_1, \ldots, x_n\}$ depend on the network $g_w$ and therefore on the weights $w$. We can reformulate the previous problem as

$$\min_w \text{Distance}(\{g_w(y_1), \ldots, g_w(y_n)\}, \{z_1, \ldots, z_n\}).$$

The mathematical question that arises is therefore to define a notion of distance between two sets of points. There are many ways to do this, and we will explain one that is well suited to this learning problem. It exploits the theory of optimal transport.

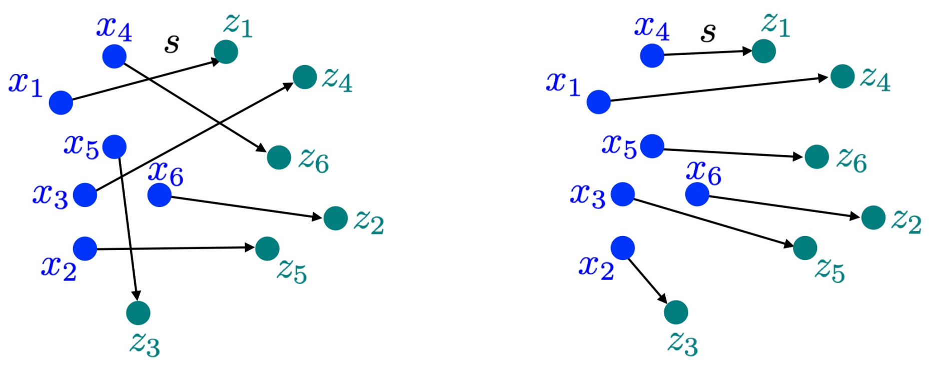

Figure 2.2: Example on the left of a non-optimal permutation $s$ and on the right of the optimal permutation, in the case of 6 points in dimension 2.

## 2.3 Monge's optimal transport

The optimal transport problem is formulated by Gaspard Monge [15] in 1781, for military applications. The question asked is to determine the most economical way to transfer objects from a set of sources $\{x_1, \ldots, x_n\}$ to a set of destinations $\{z_1, \ldots, z_n\}$. For Monge, it is a matter of transferring soil from cuttings to create embankments. But this question finds a multitude of applications. For the problem of training generative networks, the sources are the false images generated by the network and the destinations are the images of the database.

It is thus necessary to link each source, for example $x_1$ to a single destination point, which we will note $z_{s_1}$, where $s_1$ is an integer between 1 and $n$. Similarly, $x_2$ is linked to $z_{s_2}$ and so on. For example, in figure 2.2, we link $x_2$ to $z_5$, which means that $s_2 = 5$. Each of the $n$ destinations must also be supplied by a source. This means for example that $x_1$ and $x_2$ cannot be linked to the same destination, all the sources must be linked to different destinations. This means that $\{s_1, \ldots, s_n\}$ must be a permutation of the first $n$ integer numbers. For example, on the figure 2.2, on a simple example with $n = 6$ elements, we have chosen on the left the permutation

$$(s_1 = 1, s_2 = 5, s_3 = 4, s_4 = 6, s_5 = 3, s_6 = 2).$$

Monge's problem then consists in finding the permutation which minimizes the sum of the transport costs. Monge decided that the cost of transportation between a source $x$ and a destination $z$ is equal to the Euclidean distance $\|x - z\|$ between the two points, but one can choose another cost: for example a travelling time or the price required for gasoline if using trucks, etc. We have to solve the problem

$$\min_{\text{permutation } s} \|x_1 - z_{s_1}\| + \|x_2 - z_{s_2}\| + \ldots + \|x_n - z_{s_n}\|.$$

Once we have calculated an optimal permutation $s^\star = (s_1^\star, \ldots, s_n^\star)$ (i.e. which is solution of the previous problem), we define the distance between the sets of points as the value of the total transport cost

$$\text{Distance}(\{x_1, \ldots, x_n\}, \{z_1, \ldots, z_n\}) \overset{\text{def.}}{=} \|x_1 - z_{s_1^\star}\| + \|x_2 - z_{s_2^\star}\| + \ldots + \|x_n - z_{s_n^\star}\|.$$

The difficulty in calculating this distance is that the total number of permutations to be tested is very large. Indeed, for the choice of $s_1$ there are $n$ possibilities, for that of $s_2$ there is $n-1$ (since the value of $s_1$ is taken), for $s_2$ there is $n-2$, etc. So the total number of permutations is $n!$, The factorial of the number $n$, which is defined as

$$n! = n \times (n-1) \times (n-2) \times \ldots \times 2 \times 1.$$

For $n = 6$ as in the figure 2.2, there is therefore

$$6! = 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 720 \text{ possible permutations.}$$

In this simple case, we can test them all and choose the best one, which is, as shown on the right of the figure 2.2,

$$(s_1 = 4, s_2 = 3, s_3 = 5, s_4 = 1, s_5 = 6, s_6 = 2).$$

The difficulty is that for $n = 70$, there are more than $10^{100}$ possibilities, which is to be compared to the $10^{79}$ atoms in the universe ... And to train neural networks, $n$ is even much bigger! It was therefore necessary to wait for several mathematical and algorithmic revolutions before being able to obtain a method enabling this problem to be resolved.

## 2.4 The optimal transport of Kantorovitch

Monge has noticed that the solutions to his problem have very specific structures. For example, we can observe in the figure 2.2, on the right, that the optimal paths do not cross, and Monge proved it in his article [15]. But this is not enough to solve the problem, because there are still a lot of paths without crossing. It took over 200 years to figure out how to get more information about the solutions in order to calculate them efficiently. It was Leonid Kantorovitch who found, in 1942 [9], a new formulation of the problem of optimal transport. He allowed each source to be divided into several parts, for example two equal parts with a weighting of 1/2 each. This division of production is interesting because it simplifies the optimization problem. It is also natural for the problem studied by Kantorovitch who was trying to model and plan production in economics. He won the Nobel Prize in economics for this idea. Together with these pioneering works by Kantorovitch, George Dantzig introduced in 1947 the simplex algorithm [4], which makes it possible to efficiently solve large scale transport problems. Its numerical complexity to solve an optimal transport problem between $n$ points is of the order of $n^3 = n \times n \times n$, which is much lower than $n! = n \times (n-1) \times \ldots \times 2 \times 1$. It is at the heart of a very large number of industrial systems which must optimize the adequacy between means of production and consumption. And we can also use it to train generative neural networks! One can look at [18] for more details on the optimal transport theory, efficient algorithms and its applications to data science.

## 2.5 Adversarial networks

One difficulty in applying optimal transport to generate generative networks is that it is necessary to choose the transport cost between two images. We could calculate the Euclidean distance between the pixels of the images, but this does not work well, because it does not take into account the geometry of the objects present in the images. A very successful idea was introduced in 2014 by Ian Goodfellow and his collaborators [6]. It can be interpreted as using a second neural network to determine this transport cost [1]. This second network $f$, called adversary network, plays a discriminative role. While the goal of the generator $g$ is to generate fake images which looks real,

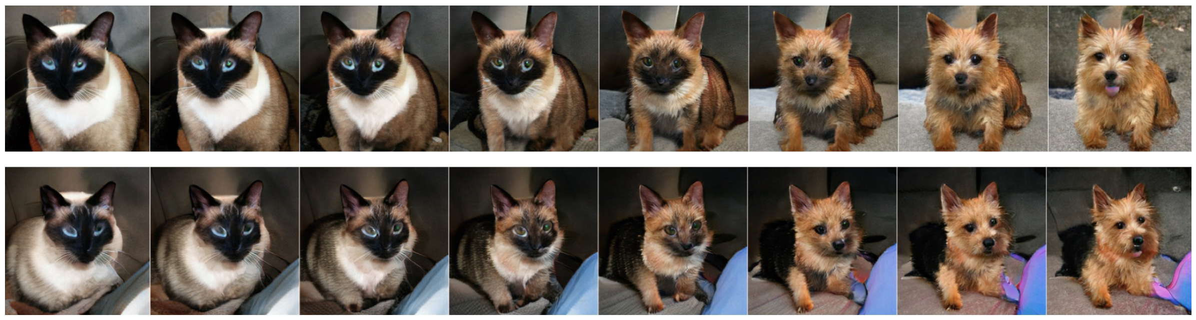

Figure 2.3: Two examples of “deep fakes” which are virtual images interpolating between cats and dogs.

the goal of $f$ is, on the contrary, to do its best to recognize true and false images. These two networks are jointly trained, this is why one speaks of adversarial networks. The training of these two networks corresponds to what is called a zero-sum game, introduced by John Von Neumann in 1944 [16] and then generalized by John Nash in 1950 [17], which just like Kantorovitch obtained the Nobel Prize in economics.

These recent advances [6] have made it possible to obtain excellent results for image generation. The figure 2.3 shows results obtained with the method explained in [2] and its use to calculate “paths” of images between dogs and cats.

## Acknowledgements

I thank Gwenn Guichaoua for her attentive proofreading, as well as Sébastien Racanière and Vincent Barra for their corrections.

# Bibliography

[1] Martin Arjovsky and Leon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34 th International Conference on Machine Learning, Sydney, Australia*, 2017.

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *Proc. ICLR 2019*, 2019.

[3] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[4] George B Dantzig. Origins of the simplex method. In *A history of scientific computing*, pages 141–151. 1990.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[7] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500–544, 1952.

[8] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

[9] Leonid Kantorovich. On the transfer of masses (in russian). *Doklady Akademii Nauk*, 37(2):227–229, 1942.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[11] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[12] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[13] Seppo Linnainmaa. Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics*, 16(2):146–160, 1976.

[14] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[15] Gaspard Monge. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences*, pages 666–704, 1781.

[16] Oskar Morgenstern and John Von Neumann. *Theory of games and economic behavior*. Princeton university press, 1953.

[17] John F Nash. Equilibrium points in $n$-person games. *Proceedings of the national academy of sciences*, 36(1):48–49, 1950.

[18] Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5–6):355–607, 2019.

[19] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[20] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[21] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.