Radiological Physics and Technology (2019) 12:235–248
https://doi.org/10.1007/s12194-019-00520-y

Overview of image-to-image translation by use of deep neural networks: denoising, super-resolution, modality conversion, and reconstruction in medical imaging

Shizuo Kaji[1,2] · **Satoshi Kida**[3,4]

Received: 30 May 2019 / Revised: 11 June 2019 / Accepted: 12 June 2019 / Published online: 20 June 2019

Abstract
Since the advent of deep convolutional neural networks (DNNs), computer vision has seen an extremely rapid progress that has led to huge advances in medical imaging. Every year, many new methods are reported at conferences such as the International Conference on Medical Image Computing and Computer-Assisted Intervention and Machine Learning for Medical Image Reconstruction, or published online at the preprint server arXiv. There is a plethora of surveys on applications of neural networks in medical imaging (see [1] for a relatively recent comprehensive survey). This article does not aim to cover all aspects of the field, but focuses on a particular topic, *image-to-image translation*. Although the topic may not sound familiar, it turns out that many seemingly irrelevant applications can be understood as instances of image-to-image translation. Such applications include (1) noise reduction, (2) super-resolution, (3) image synthesis, and (4) reconstruction. The same underlying principles and algorithms work for various tasks. Our aim is to introduce some of the key ideas on this topic from a uniform viewpoint. We introduce core ideas and jargon that are specific to image processing by use of DNNs. Having an intuitive grasp of the core ideas of applications of neural networks in medical imaging and a knowledge of technical terms would be of great help to the reader for understanding the existing and future applications. Most of the recent applications which build on image-to-image translation are based on one of two fundamental architectures, called pix2pix and CycleGAN, depending on whether the available training data are *paired* or *unpaired* (see Sect. 1.3). We provide codes ([2, 3]) which implement these two architectures with various enhancements. Our codes are available online with use of the very permissive MIT license. We provide a hands-on tutorial for training a model for denoising based on our codes (see Sect. 6). We hope that this article, together with the codes, will provide both an overview and the details of the key algorithms and that it will serve as a basis for the development of new applications.

Keywords Deep convolutional neural networks · Image-to-image translation · Denoising · Super-resolution · Image synthesis · Reconstruction

1 Image processing with deep neural networks

We begin by considering basic notions about neural networks in general. The reader may want to skip to Sect. 1.3, where we discuss our main topic of image-to-image translation.

Conventional medical image-processing algorithms rely on domain-specific knowledge and assume some underlying model, and hence, each algorithm is tailored for a specific task. On the other hand, in recent years, purely data-driven approaches without specific models have become popular in medical imaging (e.g., [4–6]). In particular, deep neural networks (DNNs) have proved to be very powerful in various

S. Kaji and S. Kida contributed equally to this work.

✉ Satoshi Kida
satoshikida1492@gmail.com

Shizuo Kaji
skaji@imi.kyushu-u.ac.jp

[1] Institute of Mathematics for Industry, Kyushu University, Fukuoka, Japan

[2] JST PRESTO, Saitama, Japan

[3] LPixel Inc., Tokyo, Japan

[4] The University of Tokyo hospital, Tokyo, Japan

image-processing tasks. The most important fact for image processing is that an image is expressed by a real-valued vector. A greyscale image of dimension $w \times h$ is an element of $\mathbb{R}^{w\times h}$, where $\mathbb{R}^{w\times h}$ is the set of real-valued vectors of dimension $w \times h$. Similarly, a full color picture of dimension $w \times h$ is an element of $\mathbb{R}^{w\times h\times 3}$ consisting of three channels corresponding to Red, Green, and Blue. This fact allows us to apply various mathematical operations to the processing of images. We can apply a mapping $\mathbb{R}^{w_1\times h_1} \to \mathbb{R}^{w_2\times h_2}$ to translate one greyscale image to another. Image filters such as Sobel filters, FFT, and bilateral filters are handcrafted mappings[1], whereas DNNs are meant for finding of useful filters automatically from a large amount of data. DNN-based methods have achieved the state-of-the-art performance in many image translation tasks, including denoising, super-resolution, image synthesis, and reconstruction.

1.1 Regression with neural networks

In this subsection, we explain what are neural networks and deep learning. A good introduction to deep learning in general is given in [8]. For those who are interested in the theory of deep learning in depth, we refer to [9].

Neural networks are just a particular class of multivariate functions $f_w : \mathbb{R}^n \to \mathbb{R}^m$ with parameters (called *weights*) $w \in \mathbb{R}^l$. The parameters w are *learnable*, so that they are determined by the given data through *training* for the solution of a given problem. The simplest example is the linear function $f_{(w_1,w_0)} : \mathbb{R}^1 \to \mathbb{R}^1$, defined by $f_{(w_1,w_0)}(x) = w_1x + w_0$, whose weights are $w = (w_1, w_0) \in \mathbb{R}^2$.

A neural network can be used to approximate (*fit* or *regress*) an unknown function g from a finite number of its input–output pairs:

$$\{(x_1, g(x_1)), (x_1, g(x_1)), \dots, (x_k, g(x_k))\}.$$

Here, k is the number of input–output pairs which are observed and known to us. For a new x different from $x_1, \dots, x_k$, we want to use $f_w(x)$ as a prediction of the unknown value $g(x)$.

To measure how well f_w approximates g, we need a *loss* function. A popular choice is the squared L^2*-distance* $L(f_w, g) = \sum_{i=1}^k |f_w(x_i) - g(x_i)|^2$. A loss function is designed so that, the smaller $L(f_w, g)$ is, the better f_w approximates g with respect to the loss function L. Hence, we try to find parameters w which minimize $L(f_w, g)$, or at least, which make $L(f_w, g)$ reasonably small. In the case of the simplest neural network $f_{w_1,w_0}(x) = w_1x + w_0$ with the squared L^2 -distance as the loss function, this is achieved by ordinary least-squares fitting. For expressing functions more complex than simple linear functions, the crucial idea of neural networks is to stack (or compose) linear functions. Of course, the composition of two linear functions is again a linear function. Therefore, we insert the *activation function* (or the *non-linearity*) between linear functions. The simplest two-layer neural network $\mathbb{R}^1 \to \mathbb{R}^1$ is given by

$$f_w(x) = w_{2,1} \max(w_{1,1}x + w_{1,0}, 0) + w_{2,0}.$$

This looks complicated, but it is just the composition of two linear functions sandwiching the *ReLU* (Rectified Linear Unit) activation, which outputs the input itself if it is positive and outputs zero otherwise:

$$\begin{array}{ccccc} \mathbb{R}^1 & \to & \mathbb{R}^1 & \xrightarrow{\text{ReLU}} & \mathbb{R}^1 \\ x & \mapsto & y_1 = w_{1,1}x + w_{1,0} & \mapsto & y_2 = \max(y_1, 0) \\ & & & \to & \mathbb{R}^1 \\ & & & \mapsto & y_3 = w_{2,1}y_2 + w_{2,0}. \end{array}$$

The point is that composition increases the expressive power drastically (see, for example, [10]) while keeping the building blocks rather simple (consisting of only linear functions and activation functions). Neural networks with three or more layers are referred to as deep neural networks (DNNs). The universal approximation theorem (e.g., [11]) roughly states that almost any function $g : \mathbb{R}^n \to \mathbb{R}^m$ which appears in practical applications can be approximated arbitrarily well by a deep neural network if we allow the number of weights to be arbitrarily large. Nowadays, the number of layers of a DNN is of the order of dozens and more, and the parameter w exists in a high-dimensional space $\mathbb{R}^k$ with $k > 10,000,000$.

Remark 1 If three-layer networks suffice for universal approximation, why do we need dozens of layers? In practice, deeper neural networks are better at various jobs than are shallower and wider neural networks with the same number of weights. However, in theory, deep narrow neural networks in which the width is bounded are not universal [12]. This fact poses an interesting question in how to design the structure of neural networks for a particular task.

1.2 Convolutional neural networks

A linear map (without a constant term) $\mathbb{R}^n \to \mathbb{R}^m$ is specified by an $m \times n$ matrix[2]. This means that the number of weights is mn, which is huge; e.g., when $n = m = 256 \times 256$, the figure is over 4 million. In principle, the larger the number of weights, the more data we need to find good values for them by training.

[1] Various image filters are implemented in the free software Fiji [7], and we can easily try them out to see their characteristics.

[2] Usually, a linear layer involves the constant term as well, so that it has the form $x \mapsto Ax + b$ for $b \in \mathbb{R}^m$. The term b is often referred to as the *bias*.

Thus, we have to keep the number of weights relatively small based on some prior knowledge about images in general. It is reasonable to assume that two pixels far apart are independent, but pixels that are next to each other are highly correlated. In addition, the correlation does not depend on the absolute location of the pixel in a picture, but should be similar everywhere. A mathematical tool called the *convolution* is what we need to exploit this observation. There are many good expository articles on 2D convolutional neural networks (CNNs) (e.g., [13]). Here, we present an illustrative toy example of 1D convolution. Basically, we can think of convolution as a filter application with a sliding window. Assume that we have an input vector $x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$. Given another vector $w = (w_1, w_2, w_3)$ called the *kernel*, we compute

$$\begin{aligned} w * x =& (w_1x_1 + w_2x_2 + w_3x_3, w_1x_2 + w_2x_3 + w_3x_4, \dots, \\ & w_1x_{n-2} + w_2x_{n-1} + w_3x_n) \in \mathbb{R}^{n-2}. \end{aligned} \tag{1.1}$$

This mapping $x \mapsto w * x$ is called the convolution with kernel w, and it defines a linear mapping $\mathbb{R}^n \to \mathbb{R}^{n-2}$ with weight w. Note that the number of weights is fixed to three and is independent of the dimension of the input x. The trick is that the weights are

- *sparse*; each entry of the output $w * x$ depends only on three entries, x_{i-1}, x_i, x_{i+1}, in the input,
- *shared*; the same weights, w_1, w_2, w_3, are used for all entries of the output.

Next, we explain some more technical aspects of convolution. First, if we want to keep the input and the output vector sizes the same, we can use *padding* $(x_1, x_2, \dots, x_n) \mapsto (0, x_1, x_2, \dots, x_n, 0)$, so that the output (1.1) becomes

$$\begin{aligned} &(w_2x_1 + w_3x_2, w_1x_1 + w_2x_2 + w_3x_3, w_1x_2 + w_2x_3 + w_3x_4, \dots, \\ &\quad w_1x_{n-2} + w_2x_{n-1} + w_3x_n, w_1x_{n-1} + w_2x_n) \in \mathbb{R}^n. \end{aligned}$$

Padding also prevents the boundary elements from being treated lightly; without padding, x_1 and x_n contribute only once to the output, whereas $x_3, \dots, x_{n-2}$ appear three times.

A layer with the general form of a linear map $\mathbb{R}^n \to \mathbb{R}^m$ is said to be *fully connected*, in contrast to being convolutional. We can stack convolutional and/or fully connected layers by sandwiching activation functions to form a DNN. With a fully connected layer, we can choose an arbitrary output dimension m. We have some but limited control over the size of the output of a convolutional layer. If we want to down-sample the signal, we use convolution with a *stride*. For example, with stride two and a padding, the output (1.1) becomes

$$\begin{aligned} &(w_2x_1 + w_3x_2, w_1x_2 + w_2x_3 + w_3x_4, w_1x_4 + w_2x_5 + w_3x_6, \dots, \\ &\quad w_1x_{n-3} + w_2x_{n-2} + w_3x_{n-1}, w_1x_{n-1} + w_2x_n) \in \mathbb{R}^{(n+1)/2} \end{aligned}$$

when n is odd and

$$\begin{aligned} &(w_2x_1 + w_3x_2, w_1x_2 + w_2x_3 + w_3x_4, w_1x_4 + w_2x_5 + w_3x_6, \dots, \\ &\quad w_1x_{n-4} + w_2x_{n-3} + w_3x_{n-2}, w_1x_{n-2} + w_2x_{n-1} + w_3x_n) \in \mathbb{R}^{n/2} \end{aligned}$$

when n is even. Down-sampling can also be achieved by a fixed (not learnable) filter such as *max-pooling* and *average-pooling* with a stride.

We can also deal with multi-channel signals. For example, for a two-channel 1D signal $x = \begin{pmatrix} x_{11}, x_{12}, \dots, x_{1n} \\ x_{21}, x_{22}, \dots, x_{2n} \end{pmatrix} \in \mathbb{R}^{2\times n}$, we use a kernel $w = \begin{pmatrix} w_{11}, w_{12}, x_{13} \\ w_{21}, w_{22}, w_{23} \end{pmatrix}$, and the convolution is defined to be

$$\begin{aligned} w * x = &((w_{11}x_{11} + w_{12}x_{12} + w_{13}x_{13}) + (w_{21}x_{21} + w_{22}x_{22} + w_{23}x_{23}), \\ &(w_{11}x_{12} + w_{12}x_{13} + w_{13}x_{14}) + (w_{21}x_{22} + w_{22}x_{23} + w_{23}x_{24}), \\ &\dots, (w_{11}x_{1(n-2)} + w_{12}x_{1(n-1)} + w_{13}x_{1n}) \\ &+ (w_{21}x_{2(n-2)} + w_{22}x_{2(n-1)} + w_{23}x_{2n})) \in \mathbb{R}^{n-2}. \end{aligned}$$

We can also use multiple kernels to make an output multi-channel; the number of output channels is equal to the number of kernels used. For example, multiple gradient-like filters in different orientations may be useful for capturing image features. Each channel in the output is sometimes called the feature map. With CNNs, kernels (filters) are not designed by a human being, but are learned from data. To train a CNN, we have to provide its objective in terms of a loss function (or a cost function, an energy function, a penalty function). Training means optimizing the loss function by finding a set of weights which attains a small value of the given loss function. The training proceeds iteratively by gradual adjustment of the weights to lower the value of the loss functions. This usually requires much time and powerful equipment including GPUs.

Remark 2 An 1D signal with multiple channels should not be confused with a 2D signal with a single channel. Convolution operates differently. Namely, kernels slide only spatially, but not in the direction of channels.

Usually, a down-sampling layer comes with more output channels than input channels. The numbers are chosen, so that the overall size of the output is smaller than that of the input, and the layer serves as a compressor of information. Learning a compact representation in this manner is one of the key ideas of CNN. For 2D signals, a typical down-sampling layer has stride two with doubling of output channels, so that the total size of information is halved.

If we want to up-sample the signal, we can for example, use simple bilinear interpolation or rearrange multiple channels [14]. It should be noted that it has been popular to use *deconvolution* (or *transposed convolution*) for

up-sampling, which is the linear adjoint to convolution (see [13] for details), However, deconvolution often results in a draughtboard-like noise in the output [15].

Often, between a convolution and an activation, various *normalization* methods are applied for better convergence in optimization [16]. In addition, in image translation, normalization is known to have a great impact on the output [17]. The popular combination of convolution followed by batch normalization followed by ReLU activation is abbreviated as CBR.

In image processing, the majority of neural networks are feed-forward networks, so that the data flow in one direction and there is no feedback loop. Designing a DNN architecture mainly involves knowing how to connect layers with different numbers of channels, strides, and kernel sizes. A particular attention should be paid to the receptive field when designing an image-to-image translation network; each pixel in the output image is affected by not all pixels in the input image but by a patch in it. The size of the receptive field can be easily computed by a dedicated calculator which is found online.

In summary, employing DNNs for a specific task requires

1. collecting training data (this is probably the hardest part);
2. designing the network architecture of DNNs, and loss functions (there is a variety of off-the-shelf architecture that one can choose);
3. training (time-consuming, powerful hardware is often required).

Once a trained (learned) model, which consists of DNNs with trained weights, is obtained, using it is not very demanding. We only need the model and do not need the training data. The process of getting outputs from a trained model is called inference, and it usually does not require a great deal of machine power. However, it has to be noted that some information on the training data can be extracted from a trained model by a malicious attacker [18]. Therefore, care has to be taken when one provides a learned model which is trained on data that contain sensitive personal information.

1.3 Image translation with encoder–decoder networks and GANs

Now, we focus on the way to apply DNNs for image-to-image translation. Two powerful inventions, encoder–decoder networks (e.g, [19]) and generative adversarial networks (GANs) [20], have accelerated the use of DNNs in image-to-image translation ([21]). A celebrated paper [22] crystallizes how these are combined to yield image-to-image translation, and we highly recommend that the reader have a look at it. We give an informal account of these two key inventions.

Encoder–decoder networks refer to a particular structure of DNNs which can be used for converting one input image into another. An encoder–decoder network consists of multiple down-sampling layers with increasing numbers of channels (the encoder part), followed by multiple up-sampling layers with decreasing numbers of channels (the decoder part). The network has the form of an hourglass (Fig. 1). Each down-sampling layer typically reduces the image size down to one quarter, by halving it in every dimension. The number of channels is usually chosen to be twice that of the input to the layer, so that the total information is compressed down to one half. The purpose is to squeeze and abstract the information through the information bottleneck. The output of the middle layer is referred to as the latent feature. Each up-sampling layer then recovers the image size by quadrupling.

Let us consider how this architecture works. Imagine that we have two persons at the ends of a phone line. One person (the encoder) is given a plenty of noisy pictures $x_1, x_2, \ldots$, and the other (the decoder) is given the corresponding clean pictures $y_1, y_2, \ldots$. The encoder randomly chooses $x = x_i$ and tries to describe it over the phone. The time is limited, and the person cannot say "the left most corner has RGB value (30,41,200) and one pixel to the right has ...", but describes "there is a large house in the middle with a blueish roof and ..." The decoder, at the other end of the phone, tries to draw a picture y based on the explanation by the encoder. After finishing the drawing, the decoder is informed of the ID of the picture, which is i, so that the decoder can compare what is drawn, y, with the corresponding clean picture, y_i (the ground truth). The comparison result is fed back to the two persons to improve; the encoder improves the ability of explaining the contents of the noisy pictures, and the decoder improves drawing clean pictures from the encoder's explanation. In the end, they are supposed to master this game and become able to restore new noisy images without ground truth. More specifically, their performance is evaluated by a reconstruction loss function $d(y, y_i)$ which measures the difference between two images, y and y_i. A typical choice for $d(y', y_i)$ is the L^p distance between y' and y (recall that y' and y are just high-dimensional vectors). In particular, the L^2 distance, which is the square root of the mean squared pixelwise error, and the L^1 distance, which is the mean absolute pixelwise error, are popularly used.

Through the above process, visual information on the pictures is abstracted to verbal information. What is important is this abstract representation of the information. A picture with a lot of noise, or with a missing part, can be recovered through this abstraction process. Furthermore, if we train the decoder to draw in the style of Monet, a picture is first translated into words, and from the words the trained painter paints something like a Monet masterpiece. This is the basic

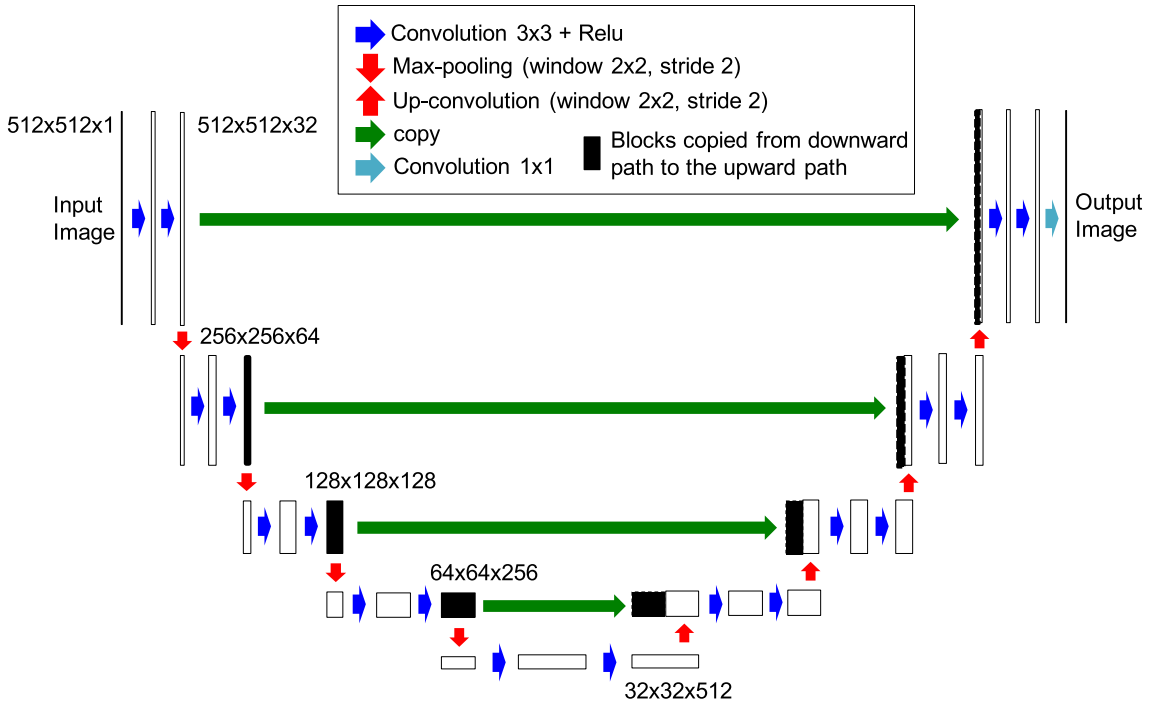

Fig. 1 Encoder–decoder network structure with skip connections (U-net), which were the output of the first down-sampling layer to the input of the last up-sampling layer and similarly for the second and the penultimate, etc. The two inputs to each up-sampling layers are stacked as extra channels. The idea of having skip connections is to transfer the raw, non-abstract information, especially that of the high-frequency signal, directly to the final output. Skip connections are also useful for mitigating the vanishing-gradient problem and accelerating learning

idea of image translation with use of encoder–decoder networks.

However, how can we train the decoder to draw like Monet? This is where the other key component, GANs, comes into the story. If we have many pairs of photos corresponding to Monet's paintings, we can use the same way as above with this training data set, by giving the photos to the encoder and the corresponding paintings to the decoder. However, Monet never drew a smartphone, and the training data set does not contain such a picture. The chance is that the encoder and the decoder cannot do well when they are asked to handle a picture of a smartphone. Therefore, we employ a third player called the discriminator (or the critic), who will be trained to distinguish real Monet paintings from forged ones by the decoder. The encoder and the decoder now have two separate objectives; they collaborate to convert a photo to a painting, whose contents are the same as in the original photo, and at the same time to deceive the third player. The team of the encoder and the decoder is called the generator. The discriminator is trained with real paintings of Monet and generated ones and has to tell how likely it is that a given picture was drawn by Monet. They train themselves through friendly rivalry. Their performance is evaluated in terms of the loss function. Each player is associated with a tailored loss function which measures how well the player achieves his or her goal. The loss function is designed, so that the smaller its value; the better the player is doing.

The discriminator is usually a DNN with down-sampling layers, whose objective and structure are the same as those used for classification tasks. The generator is an encoder–decoder network whose objective is to convert an input image with which the discriminator makes a wrong classification, and at the same time, the contents of the original image are the same as those of the original image. However, how can we be assured that the contents of the original and the converted images are same? We can think of two different situations.

First, a paired-images data set consists of pairs of aligned images in the source domain A and the target domain B. For example, a photo of a cathedral and a painting by Monet of the cathedral with exactly the same angle make a paired image. We want the generator f to learn to convert $x \in A$ to $f(x) \in B$. With a paired-images data set, the discriminator will be trained to discriminate between a pair (x, y) of a photo and the corresponding real painting and a pair $(x, f(x))$ of a photo and the generated painting. The discriminator's task is one of classification, and any classification loss can be used for training the discriminator. The discriminator learns how Monet would paint based on a photo by optimizing the discriminator's loss. This is called conditional GAN,

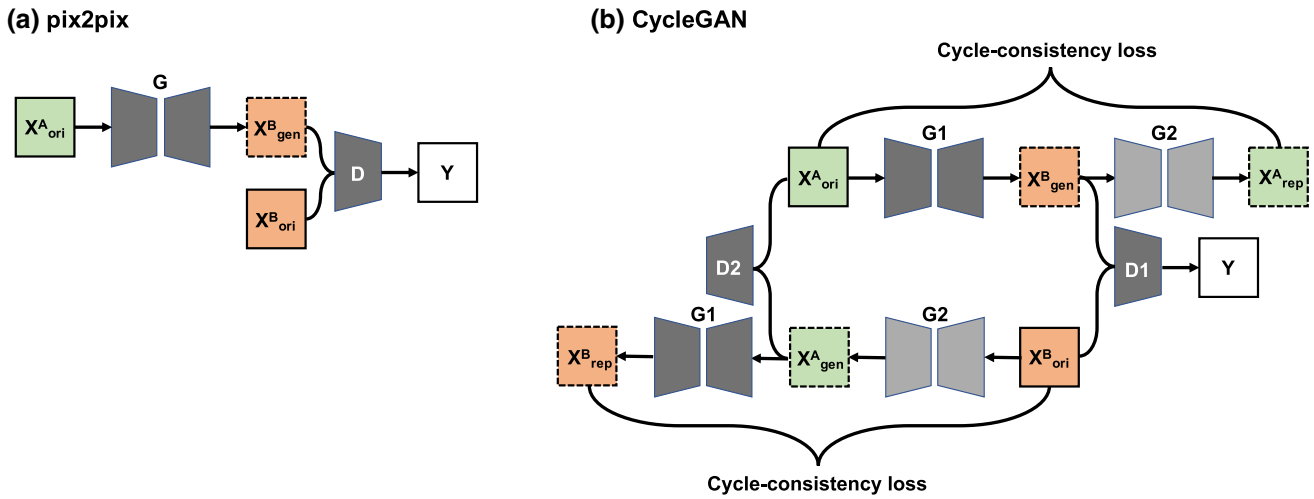

Fig. 2 Architecture of (**a**) pix2pix and (**b**) CycleGAN. **a** pix2pix requires perfectly aligned paired training images. A generator CNN is trained to generate images similar to images in domain *B* from images in domain *A*, and discriminator CNN is trained simultaneously to distinguish the generated images from real images in domain *B*. Reconstruction loss measures how close by the real images in domain B and the generated images. On the other hand, (**b**) CycleGAN can learn a translation mapping in the absence of aligned paired images. The image generated from domain *A* to domain *B* by generator CNN (G1) is converted back to domain *A* by another generator CNN (G2), and vice versa, in the attempt to optimize the cycle-consistency loss in addition to the adversarial loss

because the discriminator is shown the photo in addition to a forged or a real painting to judge its authenticity. The generator is given $x \in A$ and is trained to optimize a weighted sum of the reconstruction loss, measuring the closeness between y and $f(x)$ and the adversarial loss, which is the negative of the discriminator's loss for $(x, f(x))$. This is an instance of supervised learning. The idea of using a conditional GAN for a general-purpose image-to-image translation has been extensively investigated in pix2pix [22]. However, paired images are difficult to acquire in medical imaging, as the images must be aligned almost perfectly for making paired images.

On the other hand, an unpaired image data set consists of a set of images from the source domain *A* and an independent set of images from the target domain *B*. We do not know which image in *A* corresponds to which in *B*. Again, we want the generator f to learn to convert $x \in A$ to $f(x) \in B$. The discriminator is trained to distinguish real images y from generated images $f(x)$. Note that, here, y and x are independent and do not correspond to each other; y could be a painting of a waterlily, whereas x is one of a smartphone. In this case, it is difficult to preserve the contents under translation, as the encoder and the decoder have no way to check if the decoder's drawing corresponds to what the encoder describes. That is, it is difficult to define a reconstruction loss as we do not have the ground truth on what $f(x)$ should be. One trick to encourage the content preservation is to introduce another group of a generator g and a discriminator, who will be trained to convert in the opposite direction from the target domain *B* to the source domain *A*. We demand that a image converted by the first group is converted back to the original, and vice versa. That is, the cycle-consistency loss $|x - g(f(x))|^2 + |y - g(f(y))|^2$ should also be optimized. This encourages both parties to learn one-to-one mappings. The architecture of the whole system is depicted in Fig. 2. This is an instance of unsupervised learning. The cycle-consistency loss is introduced in *CycleGAN* [23][3]. This does not guarantee the preservation of the contents, however, and we have to be careful when using CycleGAN. Anatomic structures can be altered and non-existing tumors can be pretended in the converted images. Further to encourage structure preservation, the perceptual loss (or the content loss) [26] between the original and the converted images can be added as an optimization target of the generators. The perceptual loss between two images x and y is defined to be $|h(x) - h(y)|^2$, where h is another DNN pretrained on natural images (for example, the VGG19 network trained on the ImageNet data set which is truncated at some layer is used for h). The idea is to compare the high-level, abstract features of images extracted by a pretrained DNN; DNNs trained with natural images are like human eyes. The perceptual loss is generally useful when a direct pixel-by-pixel comparison does not make sense (e.g., when images contain strong noise).

[3] Around the same time, very similar architectures such as UNIT, DiscoGAN, and DualGAN were introduced. Walender et al. [24] evaluated UNIT [25] and CycleGAN for transformation between T1- and T2-weighted MRI images and showed that these two frameworks performed almost equally well.

Obviously, it is better to use pix2pix than CycleGAN when a paired image data set is available (see [23] for comparison study). However, in case the alignment of paired images is not perfect, CycleGAN may perform better.

With this purely data-driven approach, any image-to-image translation tasks work almost in the same manner. Only different sets of data consisting of task-specific images are required. In fact, AUTOMAP [27] demonstrated the ubiquity of this approach in a medical setting, showing that the same network can be used for reconstruction of PET, CT, and MRI. AUTOMAP relies on an encoder–decoder network and does not use GANs. What is special with reconstruction is that the spatial correspondence between the original and the reconstructed images is not straightforward. The fundamental assumption behind convolutional neural networks is that spatially close pixels are highly correlated, so that convolution with a small kernel such as 3×3 captures meaningful features. On the other hand, adjacent pixels, say, in the frequency domain as in MRI, do not necessarily affect neighboring pixels in the reconstructed image. The novel idea of AUTOMAP is to use a few fully connected layers before a typical convolutional encoder–decoder network. The fully connected layers are supposed to learn mathematical transformations such as the (inverse) Fourier transformation and the Radon transformation, which require spatially non-local manipulation.

Remark 3 One popular variant of the encoder–decoder networks, called the U-Net [19], uses skip connections which wire the output of the first down-sampling layer to the input of the last up-sampling layer, and similarly for the second and the penultimate, etc. The two inputs to each up-sampling layer are stacked as extra channels. The idea is to transfer the raw, non-abstract information, especially that of a high-frequency signal, directly to the final output. Skip connections are also useful for mitigating the vanishing-gradient problem and accelerating learning.

Remark 4 GANs are notoriously difficult to train, as a balance among different players is required. Several techniques are known for a stable training, such as the Wasserstain GAN with gradient penalty (WGAN-GP) [28], Progressive Growing GAN (PGGAN) [29], and spectral normalization [30].

2 Noise reduction

Noise reduction is the process of removing noise in images acquired through various imaging modalities, typically those with low-dose CT (LDCT). This can be considered as translating of an image with noise to one with reduced noise. Among model-based approaches, iterative reconstruction (IR) with some kind of prior information has been used for the reduction of noise and artifact. However, its slow reconstruction speed and its poor output quality have limited its clinical application. Recently, denoising methods using deep neural networks (DNN) have been intensively studied (see [31]). Most of the DNN-based methods involve post-processing of reconstructed images, which does not rely on raw projection data. For simplicity, we focus on the setting of denoising LDCT to obtain an image resembling normal-dose CT (NDCT).

Chen et al. [32] proposed a shallow encoder–decoder network trained with a paired data set consisting of NDCT images and LDCT images. Here, NDCT images are acquired routinely, and the LDCT images are generated artificially by the addition of Poisson noise to the corresponding NDCT images. The proposed DNN learns the end-to-end mapping from LDCT images to NDCT images. Once trained, the network takes LDCT images as input and converts it to one similar to NDCT images. The output quality is less plausible when it is compared with that of the state-of-the-art algorithms which we review later in this subsection. However, the paper of Chen et al. is simple and well-written and helps us to grasp the basic ideas on denoising with DNNs. Although this simple method greatly reduces the noise of LDCT images, a limitation is that the resulting images look over-smoothed, and sometimes lose structure details, because these methods target minimizing only of reconstruction losses between NDCT images and converted NDCT-like images in the training data set and they are not generalized well for new, unseen images. One solution to this problem is the use of GAN-based methods such as pix2pix. As we discussed in the previous section, pix2pix requires perfectly aligned paired training images, which it may be difficult to obtain in clinical settings. There are two main ways of preparing such paired data sets.

- Noisy images are created by the addition of artificial noise to high-quality NDCT images. The problem with this method is that the noise distribution is different from the real one.
- Multiple acquisitions are performed with different CT radiation doses at different times. The problem with this method is that the images are not perfectly aligned even if they are registered with DIR (deformable image registration).

With a paired data set at hand, a generator DNN is trained to generate NDCT-like images from LDCT images in the attempt to optimize a reconstruction loss which measures how close the real NDCT images are the generated NDCT-like images. A discriminator CNN is trained simultaneously to distinguish the generated NDCT-like images from real NDCT images. The generator tries to produce NDCT-like images which fool the discriminator by optimizing the

adversarial loss in addition to the reconstruction loss. For a reconstruction loss, the squared sum of the pixelwise difference in value (the squared L^2 loss) is often used, but it results in blurry outputs. Alternative loss functions including the perceptual feature loss [33], structure loss [34], and sharpness loss [35] have been investigated. Each of these loss functions is dependent on the accuracy of the alignment of paired images, and aligned paired training images are indispensable for using the pix2pix framework. However, in real clinical situations, it is difficult to obtain aligned paired LDCT and NDCT images. Creating LDCT by adding artificial noise to NDCT images involves the problem of a difference from the actual noise of LDCT, and multiple acquisitions at different doses cause problems of an additional radiation dose to patients and of positioning errors between acquisitions.

On the other hand, CycleGAN can learn a translation mapping in the absence of aligned paired images. Kang et al. [36] applied CycleGAN to learn the mapping between the low- and normal-dose cardiac phases, which are not aligned exactly with each other due to the cardiac motion. Their method effectively reduces the noise in low-dose cardiac images while suppressing the deformation and loss of structures.

3 Super-resolution

The purpose of super-resolution (SR) is recovering a high-resolution image from a single low-resolution image. Most existing methods for SR learn mapping functions from external low- and high-resolution example pairs (paired data sets). Conventional SR methods learn the dictionaries [37, 38] or manifolds [39, 40] for modeling of the patch space. On the other hand, DNN-based methods learn an end-to-end mapping between low- and high-resolution images, thus implicitly achieving dictionaries or mapping functions for patch space by hidden layers of DNN. With SRCNN (Super-Resolution Convolutional Neural Network), low-resolution input images are first upscaled to the desired size by use of bicubic interpolation and are then fed to an encoder–decoder network; thus, end-to-end mapping between the bicubic upscaled version of a low-resolution image and a ground-truth high-resolution image is learned. The SRCNN has been applied to mammography images [41], chest CT images [42], and MRI images [43].

SRGAN (Super-Resolution Using a Generative Adversarial Network) [44] can be thought of as a GAN-fortified version of SRCNN. An encoder–decoder network with more up-sampling layers than down-sampling layers is trained to recover detailed textures from heavily downsampled images, and a discriminator is trained to differentiate between the super-resolved images and original high-quality images. SRGAN has been applied to the generation of high-resolution brain MRI images from low-resolution images, and 3D convolution has been adopted for exploiting volumetric information [45].

4 Image synthesis

Medical image synthesis is performed for two purposes. One of these is to expand a small data set with new, plausible examples for training of DNNs for diagnosis and other tasks. The other is to generate images virtually, images which are not acquired due to the clinical workflow, or reduction of the acquisition time, cost, and dose. For the former purpose, transformations such as shifts, flips, zooms, or rotations have been performed traditionally. However, sufficient variations in the size, shape, and contrast of samples cannot be obtained, which results in the deterioration of the accuracy of the detection and classification task. To cope with this problem, unconditional synthesis by use of GAN, which generates images from noise without any other conditional information, has been performed for generating various plausible training images [46–50]. For the latter purpose, cross-modality synthesis such as generating CT-like images from MRI images by use of GAN has been proposed [51–53]. The CT-like images synthesized from MRI images may be applied to dose calculation (treatment planning) for upcoming MRI-only radiotherapy.

4.1 Unconditional synthesis

Unconditional synthesis by use of GAN generates images from noise without any other conditional information such as the boundary of structures. This can be considered as image-to-image translation in which the input images are just randomly generated noise. Deep convolutional GAN (DCGAN) and its improved variant, Progressive Growing of GAN (PGGAN), have been used for generation of medical images. DCGAN has been used for generation of synthetic samples of CT images of lung nodules [46] and liver lesions [47], MRI images of the brain [48], and X-ray images of the chest [49]. PGGAN has been used for generating synthetic samples of mammography images [50]. DCGAN has been applied to image resolution of up to 256×256, but PGGAN can be applied to an image resolution of up to 1024×1024. The key idea of PGGAN is to grow both the generator and the discriminator progressively: starting from a low resolution, and layers that model increasingly fine details as training progresses are added.

These GAN-based augmented images were found to be beneficial for various classification and segmentation tasks when combined with real images. Wu et al. [54] reported that a GAN-based augmentation improved the area under the

curve (AUC) of a mammogram patch-based classification by 0.009 (0.887 → 0.896) over a traditional augmentation approach. Mok et al. [55] reported that a PGGAN-based augmentation improved the dice coefficient of segmentation of a brain tumor by 0.03 (0.81 → 0.84) over a traditional augmentation approach. Frid-Adar et al. [56] reported that a DCGAN-based augmentation improved the classification of a liver lesion by 7.1% (78.6% → 85.7%) in sensitivity and 4.0% (88.4% → 92.4%) in specificity over a traditional augmentation approach.

4.2 Cross-modality synthesis

Cross-modality synthesis by use of encoder–decoder networks has been explored in several studies, which included MRI-to-CT synthesis [57, 58] and cone beam CT-to-planning CT synthesis [59]. In cases, where images from two different modalities can be spatially aligned correctly, such as PET-CT, where PET images and CT images are acquired simultaneously [60], pix2pix can be used. On the other hand, CycleGAN is used in case it is clinically difficult or impossible to acquire aligned paired training images from two different modalities such as CT and MRI images, which are acquired with different modalities [51–53].

Wolterink et al. [51] synthesized CT images of the head from MRI images of the head using CycleGAN, and they showed that CycleGAN using unpaired images created CT images that looked more realistic and contained fewer artifacts and blurring than did GAN combined with voxelwise loss by use of aligned images. The difference could be due to misalignment between MRI and CT images, which is ignored in training with unpaired images by use of CycleGAN. However, CycleGAN sometimes fails to preserve the boundaries of structures in images such as those of the pelvic region, which has large variations in the anatomic structures between images from two different modalities due to the presence of joints, muscles, intestines, and rectum. Hiasa et al. [52] extended CycleGAN by incorporating a loss function named gradient consistency loss, which evaluates consistency of image gradient at each pixel between the original and the synthesized images to preserve the boundary of structures in the pelvic region at MRI-to-CT synthesis. They showed that synthesized images with gradient consistency loss preserved the shape near the femoral head and adductor muscles better than did those without gradient consistency loss. Kida et al. [61] also extended CycleGAN by incorporating several losses for better preservation of the boundary of structures in the pelvic region in cone beam CT-to-planning CT synthesis. They showed that edge sharpness and structures were preserved not only in large, bulky tissues such as the rectum and bladder, but also in small isolated structures such as the small intestine and intestinal gas. They showed, with some initial weightings of the neural network, that the generators completely failed to learn a good mapping and produced totally distorted images. This kind of failure tends not to be reported, but practitioners have to be warned when they plan to deploy DNN-based methods. Zhang et al. [53] employed shape consistency loss that is produced by another image-to-image translation DNN for segmentation, named the segmentor. One segmentor is introduced in each domain, so that it can learn segmentation, in addition to the generator and the discriminator. The segmentor's outputs for the original and the translated images are compared in the shape consistency loss, which is a type of perceptual loss. During training, the segmentor takes advantage of the generator using synthetic images as augmented data, and the generator and the segmentor prompt each other to preserve the anatomic structures under translation through the loss functions. Finally, the segmentors provide implicit shape constraints on the anatomic structures during image synthesis. It should be noted that semantic labels (annotated segmentation) of training images from both modalities are required.

5 Reconstruction

Imaging such as CT, MRI, and PET can also be thought of as a specific instance of image-to-image translation from a volumetric object (density distribution) in three-dimensional space to a (series of) projection image(s). In other words, imaging is a mapping:

$$Im : \mathbb{R}^{d_1 \times h_1 \times w_1} \to \mathbb{R}^{d_2 \times h_2 \times w_2}.$$

Reconstruction is then the translation from the projection image back to the original distribution and is regarded as a mapping:

$$Rec : \mathbb{R}^{d_2 \times h_2 \times w_2} \to \mathbb{R}^{d_1 \times h_1 \times w_1},$$

so that the composition $Rec \circ Im$ is the identity map. In the ideal setting of no noise and infinite resolution, mathematics allows us to find a good reconstruction mapping *Rec* for each imaging mapping *Im*, such as the filtered back-projection for fan-beam CT. However, the reality is that we have to tackle technical difficulties such as noise and low resolution due to scanner and dose limitations, as well as numerical error of the algorithm. In this section, we discuss what machine learning can offer for improved reconstruction.

5.1 Hybrid approach

As reconstruction can be thought of as image-to-image translation, the DNNs discussed in Sect. 1.3 (e.g., AUTOMAP [27]) can be directly applicable if we have enough data consisting of pairs of original and translated images. That is, we can approximate the reconstruction mapping by a DNN. Purely data-driven methods do not rely on the fact that

imaging is the result of a physical process that is subject to certain rules. Here, we see how we can exploit knowledge from physics and mathematics and combine it with data-driven approaches to enhance the performance of reconstruction. In this way, we will achieve a better quality of reconstruction with a lesser amount of training image pairs.

Most imaging techniques guarantee at least theoretically that for each projection, there exists a unique distribution, so that the projection and the original distribution contain exactly the same amount of information. However, due to the noise and the finite resolution, some information gets lost in the imaging process, which makes the reconstruction ill-posed and difficult. That is, given a projection image, we cannot expect to find one and only one distribution which projects to it. Therefore, the aim of reconstruction is to find a distribution which translates as closely as possible to the given projection with respect to some goodness and closeness criteria.

In practice, we (assume to) know the mathematical model of the imaging, which is represented by

$$y = Im(x), \tag{5.1}$$

where x is a distribution, and y is the corresponding projection. The imaging mapping Im is often called the forward operator and is determined by the configuration of the imaging equipment such as the scanner geometry and the subsampling scheme. If Im were invertible, we could directly solve $x = Im^{-1}(y)$, but usually Eq. (5.1) is ill-posed due to hardware limitations, and there exists no Im^{-1}. Therefore, reconstruction takes the form of an optimization problem, namely, we would like to find an x^* which minimizes the following:

$$L(x) = d(y, Im(x)) + \lambda E(x), \tag{5.2}$$

where the reconstruction loss term (or the data fidelity term) $d(y, Im(x))$ shows how close y and $Im(x)$ are, the regularisation term $E(x)$ shows how appropriate x is as a reconstructed image, and $\lambda > 0$ determines the balance between the two criteria. Such an x^* is often denoted by $\mathrm{argmin}(L(x))$. There are many choices for d, E, and λ. Once we fix Im, d, E, and λ, all we have to do for the reconstruction is simply optimizing (finding an x which attains the small value of) (5.2). The reader may notice that this is exactly the same as the formulation of iterative reconstruction.

Popular choices for d are the L^1 norm and the L^2 norm. Both are standard distances in Euclidean space. Whereas L^2 greatly penalizes large deviation in values, it permits a small deviation in a large region. These characteristics often lead to a blurry output. On the other hand, L^1 prefers two images to be exactly the same at most of pixels, but allows the rest of pixels to deviate a lot. It is wise to keep these characteristics in mind to design loss functions tailored for specific applications. We can also use a DNN for defining d which measures an abstract distance between images such as a perceptual loss (see Sect. 1.3).

The purpose of the regularization term is threefold:

1. to incorporate prior about images for a higher quality output;
2. to suppress over-fitting for a better generalization;
3. to mitigate ill-posedness for better convergence of the optimization.

For example, it is reasonable to ask for a solution image which has some spatial uniformity. A popular conventional choice is the L^1 norm of the image gradient (the total variation). We can also use DNNs to learn image priors from data [62]. This idea is similar to GANs; a DNN is trained with the images in the reconstructed domain to learn their distribution. In fact, we can use the adversarial loss from a discriminator trained with high-quality reconstructed images and images x which are in the middle of reconstruction. One problem, however, is that such high-quality images may not be available, as we have no way of knowing the ground-truth distribution of our body except for phantoms. Therefore, using a DNN for image priors is useful when high-quality reconstruction images (e.g., planning CT) are available and we would like to do the reconstruction with limited-quality projections (low-dose CT). The regularization term E can consist of many sub-terms. For example, we can take E to be a weighted sum of the total variation and the adversarial loss.

Another interesting way of regularisation, called the deep image prior, is proposed in [63]. This does not take the form of the added term E. Instead, the idea is to replace x with $f_w(0)$ in (5.2), where f is a DNN with weight w. The optimization is performed not directly on x, but indirectly on w. One way to understand this is to think of DNNs as restricted approximators of arbitrary signals, just as trigonometric functions and wavelet functions are conventionally used for approximating signals. We know from sparse modeling that if we restrict ourselves to use only a small number of function bases, we obtain better results. As DNNs are suitable for representing visual information which our eyes can perceive, it may be more reasonable to use DNNs rather than trigonometric functions and wavelets as approximators. In other words, whereas x can take any value in the entire Euclidean space, $f_w(0)$ is constrained within the image of a mapping $w \mapsto f_w(0)$, which is more likely to be visually plausible.

In any case, the motto of regularization is "do not give complete freedom, but restrict in a reasonable manner", as in ordinary education.

Both model-based and data-driven approaches can be integrated in the form of the optimization of (5.2). This hybrid approach has been studied actively in recent years (see, for example, [62, 64, 65]).

Table 1 Brief summary of tasks and networks in the reviewed publications

	Encoder–decoder	pix2pix	CycleGAN	UNIT	SRCNN	SRGAN	DCGAN	PGGAN
Noise reduction	[32]	[33–35]	[36]					
Super-resolution					[41–43]	[44, 45]		
Unconditional synthesis							[46–49, 56]	[50, 55]
Cross-modality synthesis	[57–59]	[60]	[24, 51–53, 61]	[24, 25]				

Image-to-image translation by DNNs as reviewed in the previous sections can be seen as replacing the whole process of the optimization of (5.2) by DNNs which directly find x_* from given inputs y.

6 Sample codes

In this section, we give a hands-on account on what we encounter in the deployment of DNNs for a specific task. We use our codes for image-to-image translation for both a paired and an unpaired data set, which would cover most of the topics in this article except for those in Sect. 5.1. Our codes are written in Python, with use of a deep-learning framework, Chainer [66]. For demonstration purposes, we work on a simple denoising scenario, where we assume that a high-quality training data set is available. For other image-to-image translation tasks, the workflow is more or less the same except for the way in which data sets are acquired. Note that our purpose in the section is to give an overview of how an actual procedure may look, and we do not mean to make a practical denoising model.

For DNNs, we use the code available at [3] for general image-to-image translation DNNs for paired data sets. How to set up the running environment and what to type in to run the code are given in the code's documentation.

For data sets, we use CPTAC-SAR [67], which is made publicly available at The Cancer Imaging Archive. The data set contains normal-dose CT images of the Sarcomas cohort in the DICOM format. We make low-quality noisy CT images by adding artificial Poisson noise. This is done by the Python script included in [3], as instructed in the documentation. Now, we have hundreds of pairs of a noisy and a clean CT image, and we train a DNN based on pix2pix to learn a mapping from a noisy image to a clean one. During the training, the data set is augmented on the fly by random crop and flip, and by addition of random Gaussian noise. There are numerous hyper-parameters for training:

- how long (how many *epochs*) the training lasts; recall that training is an optimization, and the user has to decide when to stop;
- how many layers the encoder–decoder network and the discriminator network have; network architectures have to be fixed;
- how to weight different loss functions,

just to mention a few. The code comes with reasonable default values for the hyper-parameters, but to get the best performance, the user has to tune them depending on the data set and the goal of the task. Basically, we search by trial and error to find the best hyper-parameter setting. One training takes from hours to days, so that tuning can be very time-consuming.

Once we obtain a well-trained model, we can use it to denoise new DICOM images. The model can process dozens of images per second even without a GPU. The trained model may or may not generalize to other CT images. DNNs basically learn how to handle images which are similar to those in the training data set. New images can have totally different noise characteristics, or contain different organs. When the trained model's performance is not satisfactory, we have to find other data sets and/or modify the network architecture, and/or tune hyper-parameters until we are satisfied.

This is a typical procedure of deploying DNNs in an image-processing task. We can also use another code [2] to work with an unpaired data set, e.g., with noisy images from one patient and clean images from another.

7 Conclusion

In recent years, the use of DNNs has been becoming dominant in medical image processing. Many tasks are considered as instances of image-to-image translation. Two key inventions, encoder–decoder networks and GANs, have improved the usability of DNNs in image-to-image translation, and there are two fundamental architectures that utilize them; pix2pix and CycleGAN. They are used for different types of data sets; pix2pix is preferable when a data set consisting of paired images is available, whereas CycleGAN can work with a data set consisting of unpaired images. The following table summarizes various medical applications that

we reviewed in this paper in terms of their base architectures (Table 1).

In some cases, conventional model-based approaches do better than DNNs, especially when the available data are limited. In addition, data-driven approaches often lack explainability, and we may get totally wrong outputs for certain inputs. Thus, quantifying uncertainty and establishing evaluation and validation schemes is crucial for the reliable use in clinical applications. Explainability and quantifying uncertainty are among the high-priority research topics in machine learning (e.g., [68]).

We would like to emphasize that we have to be careful when using DNN-based methods. However, at the same time, we must not be too afraid of using DNNs. After all, they are merely tools. As long as the user knows their proper usage and limitations, they are an extreme powerful companion for practitioners in clinical medicine. They are like spelling helpers for authors. We have written this article with a word processor that has a spelling helper. It corrects spelling automatically, but sometimes, it makes mistakes. Nevertheless, the number of corrected words is much larger than that of mis-corrected words. We just should not trust it completely, but have to check the words again by ourselves.

For each medical image-processing scenario, the best method should be a hybrid combining the advantages of both data-driven and model-based approaches, utilizing domain-specific knowledge to DNNs. Some programming frameworks have been developed that enable hybrid approaches relatively easily. For example, ODL [69] is a Python-based framework for reconstruction algorithms which is integrated with deep-learning frameworks. For innovative advancement, radiologists, physicists, mathematicians, and computer scientists have to team up and conduct interdisciplinary research.

Acknowledgements We thank Mrs. Lanzl at the University of Chicago for her English proofreading.

Compliance with ethical standards

Conflict of Interest The authors declare that they have no conflict of interest.

Funding Kaji was partially supported by JST PRESTO, Grant Number JPMJPR16E3 Japan.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Sahiner B, Pezeshk A, Hadjiiski LM, Wang X, Drukker K, Cha KH, Summers RM, Giger ML. Deep learning in medical imaging and radiation therapy. Med phys. 2018;46:e1–36.
2. Kaji S. Image translation by CNNs trained on unpaired data. 2019. https://github.com/shizuo-kaji/UnpairedImageTranslation. Accessed 18 June 2019.
3. Kaji S. Image translation for paired image datasets (automap + pix2pix). 2019. https://github.com/shizuo-kaji/PairedImageTranslation. Accessed 18 June 2019.
4. Lu L, Zheng Y, Carneiro G, Yang L, editors. Deep learning and convolutional neural networks for medical image computing—precision medicine, high performance and large-scale datasets. Advances in computer vision and pattern recognition. Springer; 2017.
5. Knoll F, Maier AK, Rueckert D. editors. Machine learning for medical image reconstruction—first international workshop, MLMIR 2018, held in conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Proceedings, vol. 11074 of Lecture Notes in Computer Science, Springer; 2018.
6. Litjens GJS, Kooi T, Bejnordi BE, Setio AAA, Ciompi F, Ghafoorian M, van der Laak J, van Ginneken B, Sánchez CI. A survey on deep learning in medical image analysis. Med Image Anal. 2017;42:60–88.
7. Schindelin J, Arganda-Carreras I, Frise E, Kaynig V, Longair M, Pietzsch T, Preibisch S, Rueden C, Saalfeld S, Schmid B, Tinevez J-Y, White DJ, Hartenstein V, Eliceiri K, Tomancak P, Cardona A. Fiji: an open-source platform for biological-image analysis. Nat Meth. 2012;9:676–82.
8. Nielsen MA. Neural networks and deep learning. Determination Press; 2018.
9. Goodfellow IJ, Bengio Y, Courville A. Deep learning. Cambridge, MA, USA: MIT Press; 2016 http://www.deeplearningbook.org.
10. Safran I, Shamir O. Depth-width tradeoffs in approximating natural functions with neural networks. In: Proceedings of the 34th international conference on machine learning, Vol 70, ICML'17, p. 2979–87, JMLR.org; 2017.
11. Scarselli F, Tsoi AC. Universal approximation using feedforward neural networks: a survey of some existing methods, and some new results. Neural Netw. 1998;11:15–37.
12. Lu Z, Pu H, Wang F, Hu Z, Wang L. The expressive power of neural networks: a view from the width. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, editors. Advances in neural information processing systems 30. Red HookRed Hook: Curran Associates, Inc.; 2017. p. 6231–9.
13. Dumoulin V, Visin F. A guide to convolution arithmetic for deep learning, 2016. arXiv:1603.07285.
14. Shi W, Caballero J, Huszar F, Totz J, Aitken AP, Bishop R, Rueckert D, Wang Z. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: 2016 IEEE conference on computer vision and pattern recognition, CVPR 2016, Las Vegas, NV, USA, June 27–30, 2016; 2016. p. 1874–1883.
15. Odena A, Dumoulin V, Olah C. Deconvolution and checkerboard artifacts. Distill; 2016.
16. Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Bach FR, Blei DM (eds) ICML of JMLR workshop and conference proceedings, vol 37. USA: JMLR.org; 2015. p. 448–56.
17. Ulyanov D, Vedaldi A, Lempitsky VS. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In: 2017 IEEE conference on computer vision and pattern recognition, CVPR 2017, Honolulu, HI, USA, July 21–26, 2017; 2017. p. 4105–13.
18. Fredrikson M, Jha S, Ristenpart T. Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22Nd ACM SIGSAC conference on computer and communications security, CCS '15, (New York, NY, USA); 2015. p. 1322–33, ACM.

19. Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. In: Medical image computing and computer-assisted intervention—MICCAI 2015—18th international conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III; 2015. p. 234–41.
20. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. In: Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ, editors. Advances in neural information processing systems 27. Curran Associates, Inc.; 2014. p. 2672–80.
21. Yi X. Awesome GAN for medical imaging. 2019. https://github.com/xinario/awesome-gan-for-medical-imaging.
22. Isola P, Zhu J, Zhou T, Efros AA. Image-to-image translation with conditional adversarial networks. In: 2017 IEEE conference on computer vision and pattern recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017; 2017. p. 5967–76.
23. Zhu J, Park T, Isola P, Efros AA. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: IEEE international conference on computer vision, ICCV 2017, Venice, Italy, October 22-29, 2017; 2017. p. 2242–51.
24. Welander P, Karlsson S, Eklund A. Generative adversarial networks for image-to-image translation on multi-contrast MR images—a comparison of CycleGAN and UNIT, 2018. arXiv:1806.07777.
25. Liu M-Y, Breuel T, Kautz J. Unsupervised image-to-image translation networks. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, editors. Advances in neural information processing systems 30. Red Hook: Curran Associates, Inc.; 2017. p. 700–8.
26. Gatys LA, Ecker AS, Bethge M. Image style transfer using convolutional neural networks. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR); June 2016. p. 2414–23.
27. Zhu BO, Liu JZ, Rosen BR, Rosen MS. Image reconstruction by domain-transform manifold learning. Nature. 2018;555:487–92.
28. Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville AC. Improved training of Wasserstein GANs. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, editors. Advances in neural information processing systems 30. Red Hook: Curran Associates, Inc.; 2017. p. 5767–77.
29. Karras T, Aila T, Laine S, Lehtinen J. Progressive growing of GANs for improved quality, stability, and variation. In: 6th international conference on learning representations, ICLR 2018, Vancouver, BC, Canada, April 30–May 3, 2018, Conference track proceedings; 2018.
30. Miyato T, Kataoka T, Koyama M, Yoshida Y. Spectral normalization for generative adversarial networks. In: 6th international conference on learning representations, ICLR 2018, Vancouver, BC, Canada, April 30–May 3, 2018, Conference track proceedings; 2018.
31. Sinyu J. CT image denoising with deep learning. 2018. https://github.com/SSinyu/CT_DENOISING_REVIEW. Accessed 18 June 2019.
32. Chen H, Zhang Y, Zhang W, Liao P, Li K, Zhou J, Wang G. Low-dose CT via convolutional neural network. Biomed Opt Express. 2017;8:679–94.
33. Yang Q, Yan P, Zhang Y, Yu H, Shi Y, Mou X, Kalra MK, Zhang Y, Sun L, Wang G. Low-dose CT image denoising using a generative adversarial network with Wasserstein distance and perceptual loss. IEEE Trans Med Imaging. 2018;37:1348–57.
34. You C, Yang Q, Shan H, Gjesteby L, Li G, Ju S, Zhang Z, Zhao Z, Zhang Y, Cong W, Wang G. Structurally-sensitive multi-scale deep neural network for low-dose CT denoising. IEEE Access. 2018;6:41839–55.
35. Yi X, Babyn P. Sharpness-aware low-dose CT denoising using conditional generative adversarial network. J Digit Imaging. 2018;31:655–69.
36. Kang E, Koo HJ, Yang DH, Seo JB, Ye JC. Cycle-consistent adversarial denoising network for multiphase coronary CT angiography. Med Phys. 2019;46:550–62.
37. Timofte R, Smet VD, Gool LV. Anchored neighborhood regression for fast example-based super-resolution. In: 2013 IEEE International conference on computer vision, 2013; 1920–1927.
38. Yang J, Wright JN, Huang TS, Ma, Y. Image super-resolution as sparse representation of raw image patches. In: 2008 IEEE conference on computer vision and pattern recognition, 2008; 1–8.
39. Bevilacqua M, Roumy A, Guillemot C, Alberi-Morel M-L. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In: BMVC; 2012.
40. Chang H, Yeung D-Y, Xiong Y. Super-resolution through neighbor embedding. In: Proceedings of the 2004 IEEE computer society conference on computer vision and pattern recognition, 2004. CVPR 2004., 2004; 1:I–I.
41. Kensuke Umehara TI, Ota Junko. Super-resolution imaging of mammograms based on the super-resolution convolutional neural network. Open J Med Imaging. 2017;7:180–95.
42. Umehara K, Ota J, Ishida T. Application of super-resolution convolutional neural network for enhancing image resolution in chest CT. J Digit Imaging. 2018;31(4):441–50.
43. Plenge E, Poot DHJ, Bernsen M, Kotek G, Houston G, Wielopolski P, van der Weerd L, Niessen WJ, Meijering E. Super-resolution methods in MRI: can they improve the trade-off between resolution, signal-to-noise ratio, and acquisition time? Magn Reson Med. 2012;68:1983–93.
44. Ledig C, Theis L, Huszar F, Caballero J, Cunningham A, Acosta A, Aitken AP, Tejani A, Totz J, Wang Z, Shi W. Photo-realistic single image super-resolution using a generative adversarial network. In: CVPR. IEEE computer society; 2017. p. 105–14.
45. Sánchez I, Vilaplana V. Brain MRI super-resolution using generative adversarial networks. In: International conference on medical imaging with deep learning, (Amsterdam, The Netherlands); 2018.
46. Chuquicusma MJM, Hussein S, Burt JR, Bagci U. How to fool radiologists with generative adversarial networks? a visual turing test for lung cancer diagnosis. 2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018), 2018; 240–244.
47. Frid-Adar M, Diamant I, Klang E, Amitai M, Goldberger J, Greenspan H. GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. Neurocomputing. 2018;321:321–31.
48. Bermudez C, Plassard AJ, Davis LT, Newton AT, Resnick SM, Landman BA. Learning implicit brain MRI manifolds with deep learning. In: Proceedings of SPIE-the international society for optical engineering, vol 10574; 2018.
49. Madani A, Moradi M, Karargyris A, Syeda-Mahmood T. Chest x-ray generation and data augmentation for cardiovascular abnormality classification. In: Proceedings of SPIE 10574, Medical Imaging 2018: Image Processing, 105741M; 2018.
50. Korkinof D, Rijken T, O'Neill M, Yearsley J, Harvey H, Glocker B. High-resolution mammogram synthesis using progressive generative adversarial networks; 2018. arXiv:1807.03401.
51. Wolterink JM, Dinkla AM, Savenije MHF, Seevinck PR, van den Berg CAT, Išgum I. Deep MR to CT synthesis using unpaired data. In: Tsaftaris SA, Gooya A, Frangi AF, Prince JL, editors. Simulation and synthesis in medical imaging. Cham: Springer International Publishing; 2017. p. 14–23.
52. Hiasa Y, Otake Y, Takao M, Matsuoka T, Takashima K, Carass A, Prince J, Sugano N, Sato Y. Cross-modality image synthesis from unpaired data using cycleGAN: effects of gradient consistency

loss and training data size. In: Goksel O, Oguz I, Gooya A, Burgos N, editors. Simulation and synthesis in medical imaging—third international workshop, SASHIMI 2018, held in conjunction with MICCAI 2018, proceedings, lecture notes in computer science, vol. 1. Berlin: Springer Verlag; 2018. p. 31–41 (**including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics**).
53. Zhang Z, Yang L, Zheng Y. Translating and segmenting multimodal medical volumes with cycle- and shape-consistency generative adversarial network. In: 2018 IEEE/CVF conference on computer vision and pattern recognition, 2018; 9242–9251.
54. Wu E, Wu K, Cox D, Lotter W. Conditional infilling GANs for data augmentation in mammogram classification. In: Stoyanov D, et al., editors. Image analysis for moving organ, breast, and thoracic images. RAMBO 2018, BIA 2018, TIA 2018. Lecture notes in computer science, vol 11040. Cham: Springer; 2018. p. 98–106.
55. Mok TCW, Chung ACS. Learning data augmentation for brain tumor segmentation with coarse-to-fine generative adversarial networks. In: Crimi A, Bakas S, Kuijf H, Keyvan F, Reyes M, van Walsum T, editors. Brainlesion: glioma, multiple sclerosis, stroke and traumatic brain injuries. Cham: Springer International Publishing; 2019. p. 70–80.
56. Frid-Adar M, Klang E, Amitai M, Goldberger J, Greenspan H. Synthetic data augmentation using GAN for improved liver lesion classification. In: 2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018), 2018; 289–293.
57. Nie D, Cao X, Gao Y, Wang L, Shen D. Estimating CT image from MRI data using 3D fully convolutional networks. In: Carneiro G, Mateus D, Peter L, Bradley A, Tavares JMRS, Belagiannis V, Papa JP, Nascimento JC, Loog M, Lu Z, Cardoso JS, Cornebise J, editors. Deep learning and data labeling for medical applications. Cham: Springer International Publishing; 2016. p. 170–8.
58. Han X. MR-based synthetic CT generation using a deep convolutional neural network method. Med Phys. 2017;44:1408–19.
59. Kida S, Nakamoto T, Nakano M, Nawa K, Haga A, Kotoku J, Yamashita H, Nakagawa K. Cone beam computed tomography image quality improvement using a deep convolutional neural network. Cureus. 2018;10:e2548.
60. Ben-Cohen A, Klang E, Raskin SP, Soffer S, Ben-Haim S, Konen E, Amitai MM, Greenspan H. Cross-modality synthesis from CT to PET using FCN and GAN networks for improved automated lesion detection. Eng Appl Artif Intell. 2019;78:186–94.
61. Kida S, Kaji S, Nawa K, Imae T, Nakamoto T, Ozaki S, Ohta T, Nozawa Y, Nakagawa K. Cone-beam CT to planning CT synthesis using generative adversarial networks; 2019. arXiv:1901.05773.
62. Rick Chang JH, Li C-L, Poczos B, Vijaya Kumar BVK, Sankaranarayanan AC. One network to solve them all–solving linear inverse problems using deep projection models. In: The IEEE international conference on computer vision (ICCV); Oct 2017.
63. Ulyanov D, Vedaldi A, Lempitsky VS. Deep image prior. In: Proceedings of CVPR2018. IEEE Computer Society; 2018. p. 9446–54.
64. Adler J, Öktem O. Solving ill-posed inverse problems using iterative deep neural networks. Inverse Probl. 2017;33:124007.
65. Lucas A, Iliadis M, Molina R, Katsaggelos AK. Using deep neural networks for inverse problems in imaging: Beyond analytical methods. IEEE Signal Process Mag. 2018;35:20–36.
66. Tokui S, Oono K, Hido S, Clayton J. Chainer: a next-generation open source framework for deep learning. In: Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS); 2015.
67. National Cancer Institute Clinical Proteomic Tumor Analysis Consortium (CPTAC), Radiology data from the clinical proteomic tumor analysis consortium sarcomas [CPTAC-SAR] collection [data set]. 2018. https://wiki.cancerimagingarchive.net/display/Public/CPTAC-SAR, Accessed 18 June 2019.
68. Tanno R, Worrall DE, Ghosh A, Kaden E, Sotiropoulos SN, Criminisi A, Alexander DC. Bayesian image quality transfer with CNNs: exploring uncertainty in dMRI super-resolution. In: Proceedings of Medical image computing and computer assisted intervention—MICCAI 2017, Quebec City, QC, Canada, September 11–13; 2017. p. 611–19.
69. Adler J, Kohr H, Öktem O. Operator discretization library (ODL). 2017. https://odlgroup.github.io/odl/. Accessed 18 June 2019.