Insights into Imaging

About Articles Submission Guidelines Submit manuscript

Download PDF

Review | Open access | Published: 22 June 2018

Convolutional neural networks: an overview and application in radiology

Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do & Kaori Togashi

Insights into Imaging **9**, 611–629 (2018)

504k Accesses | **2755** Citations | **86** Altmetric | Metrics

Abstract

Convolutional neural network (CNN), a class of artificial neural networks that has become dominant in various computer vision tasks, is attracting interest across a variety of domains, including radiology. CNN is designed to automatically and adaptively learn spatial hierarchies of features through backpropagation by using multiple building blocks, such as convolution layers, pooling layers, and fully connected layers. This review article offers a perspective on the basic concepts of CNN and its application to various radiological tasks, and discusses its challenges and future directions in the field of radiology. Two challenges in applying CNN to radiological tasks, small dataset and overfitting, will also be covered in this article, as well as techniques to minimize them. Being familiar with the concepts and advantages, as well as limitations, of CNN is essential to leverage its potential in diagnostic radiology, with the goal of augmenting the performance of radiologists and improving patient care.

Key Points

- *Convolutional neural network is a class of deep learning methods which has become dominant in various computer vision tasks and is attracting interest across a variety of domains, including radiology.*
- *Convolutional neural network is composed of multiple building blocks, such as convolution layers, pooling layers, and fully connected layers, and is designed to automatically and adaptively learn spatial hierarchies of features through a backpropagation algorithm.*
- *Familiarity with the concepts and advantages, as well as limitations, of convolutional neural network is essential to leverage its potential to improve radiologist performance and, eventually, patient care.*

Introduction

A tremendous interest in deep learning has emerged in recent years [1]. The most established algorithm among various deep learning models is convolutional neural network (CNN), a class of artificial neural networks that has been a dominant method in computer vision tasks since the astonishing results were shared on the object recognition competition known as the ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2012 [2, 3]. Medical research is no exception, as CNN has achieved expert-level performances in various fields. Gulshan et al. [4], Esteva et al. [5], and Ehteshami Bejnordi

and several studies have already been published in areas such as lesion detection [7], classification [8], segmentation [9], image reconstruction [10, 11], and natural language processing [12]. Familiarity with this state-of-the-art methodology would help not only researchers who apply CNN to their tasks in radiology and medical imaging, but also clinical radiologists, as deep learning may influence their practice in the near future. This article focuses on the basic concepts of CNN and their application to various radiology tasks, and discusses its challenges and future directions. Other deep learning models, such as recurrent neural networks for sequence models, are beyond the scope of this article.

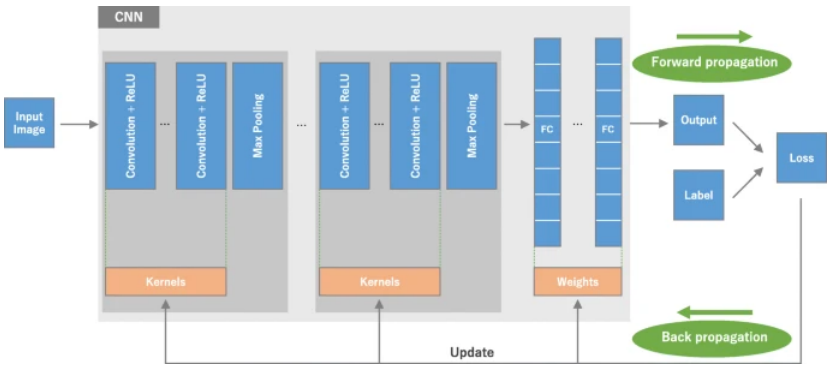
Terminology

The following terms are consistently employed throughout this article so as to avoid confusion. A “parameter” in this article stands for a variable that is automatically learned during the training process. A “hyperparameter” refers to a variable that needs to be set before the training process starts. A “kernel” refers to the sets of learnable parameters applied in convolution operations. A “weight” is generally used interchangeably with “parameter”; however, we tried to employ this term when referring to a parameter outside of convolution layers, i.e., a kernel, for example in fully connected layers.

What is CNN: the big picture (Fig. 1)

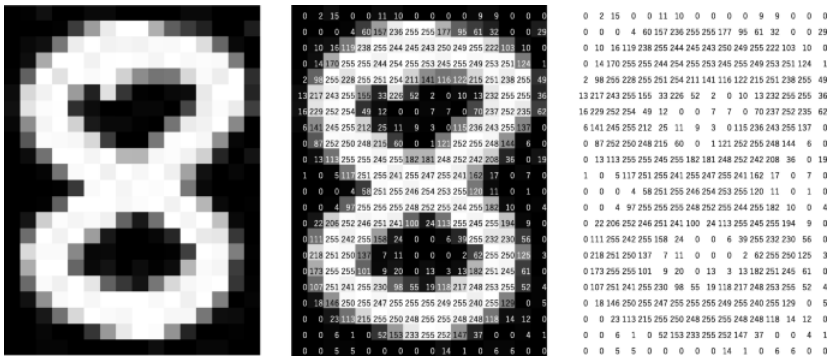
CNN is a type of deep learning model for processing data that has a grid pattern, such as images, which is inspired by the organization of animal visual cortex [13, 14] and designed to automatically and adaptively learn spatial hierarchies of features, from low- to high-level patterns. CNN is a mathematical construct that is typically composed of three types of layers (or building blocks): convolution, pooling, and fully connected layers. The first two, convolution and pooling layers, perform feature extraction, whereas the third, a fully connected layer, maps the extracted features into final output, such as classification. A convolution layer plays a key role in CNN, which is composed of a stack of mathematical operations, such as convolution, a specialized type of linear operation. In digital images, pixel values are stored in a two-dimensional (2D) grid, i.e., an array of numbers (Fig. 2), and a small grid of parameters called kernel, an optimizable feature extractor, is applied at each image position, which makes CNNs highly efficient for image processing, since a feature may occur anywhere in the image. As one layer feeds its output into the next layer, extracted features can hierarchically and progressively become more complex. The process of optimizing parameters such as kernels is called training, which is performed so as to minimize the difference between outputs and ground truth labels through an optimization algorithm called backpropagation and gradient descent, among others.

Fig. 1



An overview of a convolutional neural network (CNN) architecture and the training process. A CNN is composed of a stacking of several building blocks: convolution layers, pooling layers (e.g., max pooling), and fully connected (FC) layers. A model’s performance under particular kernels and weights is calculated with a loss function through forward propagation on a training dataset, and

Fig. 2



A computer sees an image as an array of numbers. The matrix on the right contains numbers between 0 and 255, each of which corresponds to the pixel brightness in the left image. Both are overlaid in the middle image. The source image was downloaded via http://yann.lecun.com/exdb/mnist

How is CNN different from other methods employed in radiomics?

Most recent radiomics studies use hand-crafted feature extraction techniques, such as texture analysis, followed by conventional machine learning classifiers, such as random forests and support vector machines [15, 16]. There are several differences to note between such methods and CNN. First, CNN does not require hand-crafted feature extraction. Second, CNN architectures do not necessarily require segmentation of tumors or organs by human experts. Third, CNN is far more data hungry because of its millions of learnable parameters to estimate, and, thus, is more computationally expensive, resulting in requiring graphical processing units (GPUs) for model training.

Building blocks of CNN architecture

The CNN architecture includes several building blocks, such as convolution layers, pooling layers, and fully connected layers. A typical architecture consists of repetitions of a stack of several convolution layers and a pooling layer, followed by one or more fully connected layers. The step where input data are transformed into output through these layers is called forward propagation (Fig. 1). Although convolution and pooling operations described in this section are for 2D-CNN, similar operations can also be performed for three-dimensional (3D)-CNN.

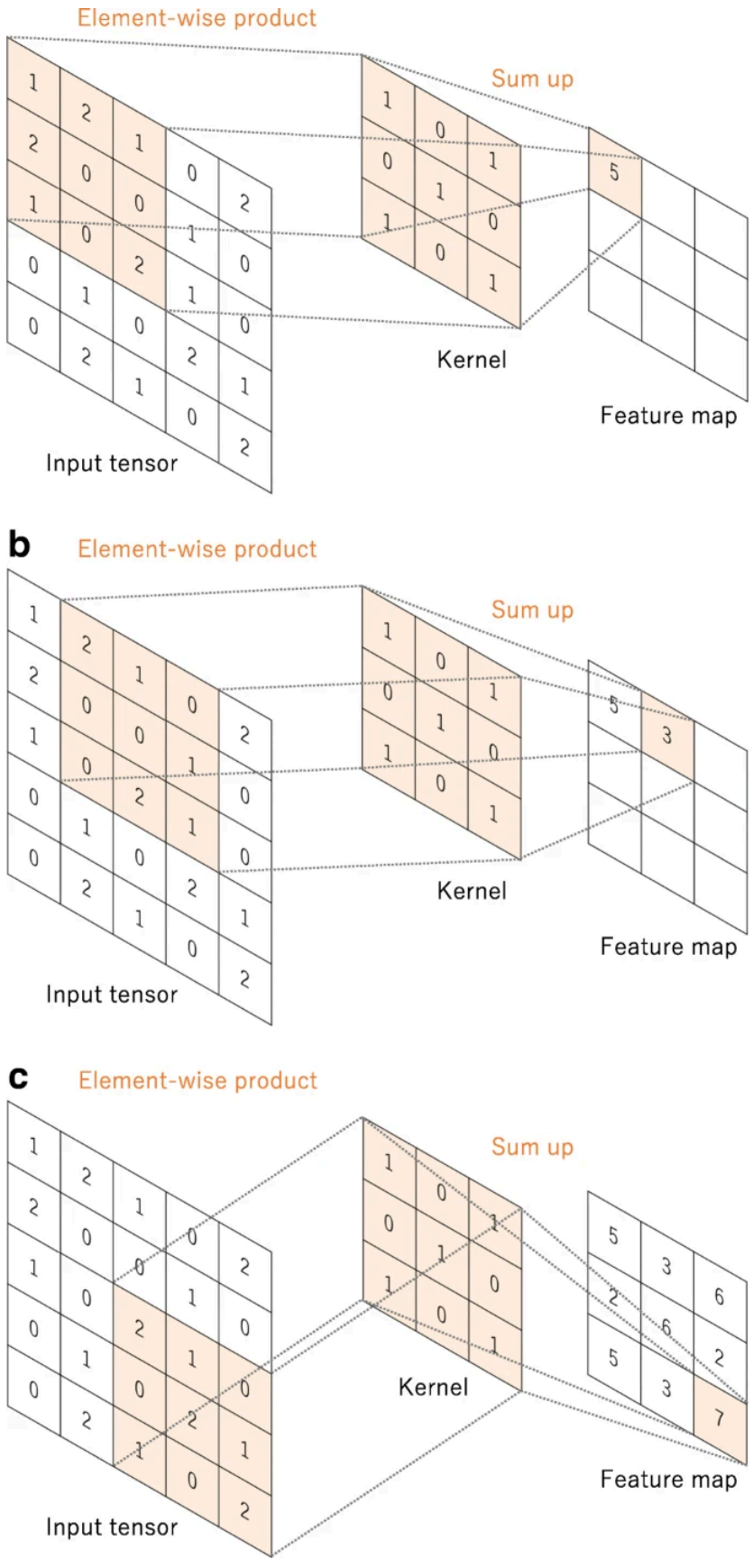
Convolution layer

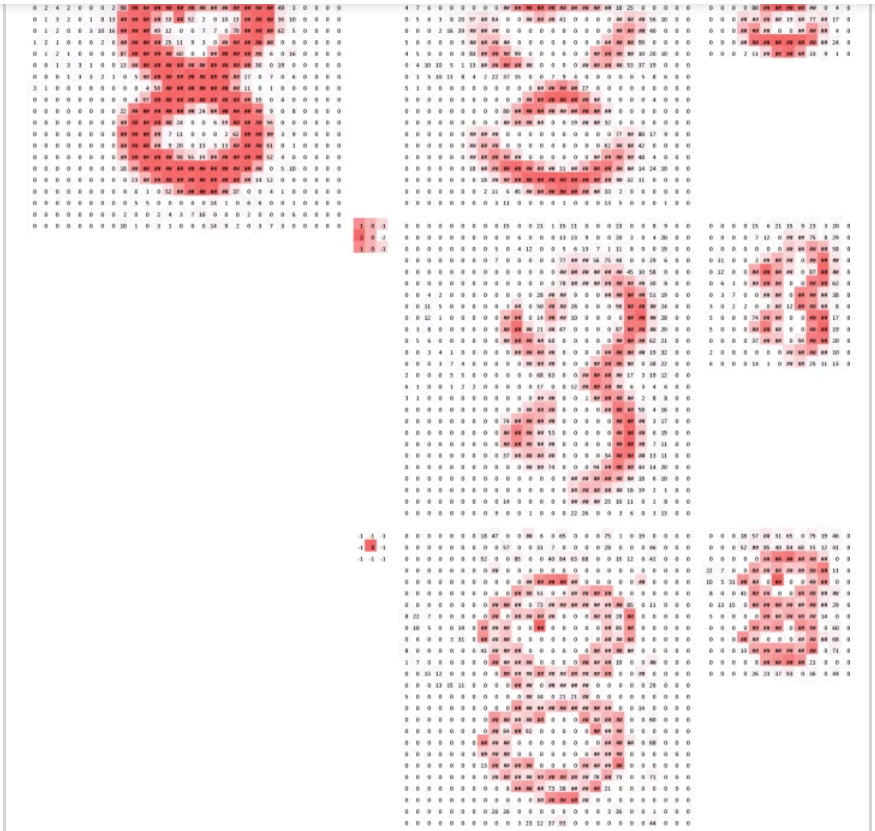
A convolution layer is a fundamental component of the CNN architecture that performs feature extraction, which typically consists of a combination of linear and nonlinear operations, i.e., convolution operation and activation function.

Convolution

Convolution is a specialized type of linear operation used for feature extraction, where a small array of numbers, called a kernel, is applied across the input, which is an array of numbers, called a tensor. An element-wise product between each element of the kernel and the input tensor is calculated at each location of the tensor and summed to obtain the output value in the corresponding position of the output tensor, called a feature map (Fig. 3a–c). This procedure is repeated applying multiple kernels to form an arbitrary number of feature maps, which represent different characteristics of the input tensors; different kernels can, thus, be considered as different feature extractors (Fig. 3d). Two key

Fig. 3

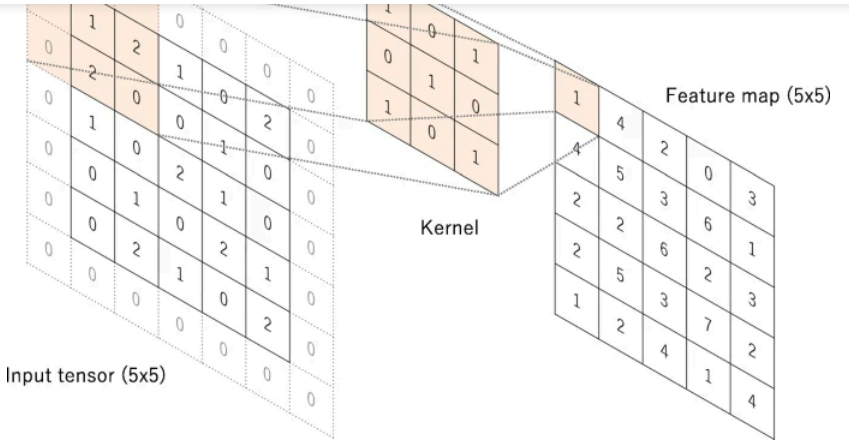




a–**c** An example of convolution operation with a kernel size of 3 × 3, no padding, and a stride of 1. A kernel is applied across the input tensor, and an element-wise product between each element of the kernel and the input tensor is calculated at each location and summed to obtain the output value in the corresponding position of the output tensor, called a feature map. **d** Examples of how kernels in convolution layers extract features from an input tensor are shown. Multiple kernels work as different feature extractors, such as a horizontal edge detector (top), a vertical edge detector (middle), and an outline detector (bottom). Note that the left image is an input, those in the middle are kernels, and those in the right are output feature maps

The convolution operation described above does not allow the center of each kernel to overlap the outermost element of the input tensor, and reduces the height and width of the output feature map compared to the input tensor. Padding, typically zero padding, is a technique to address this issue, where rows and columns of zeros are added on each side of the input tensor, so as to fit the center of a kernel on the outermost element and keep the same in-plane dimension through the convolution operation (Fig. 4). Modern CNN architectures usually employ zero padding to retain in-plane dimensions in order to apply more layers. Without zero padding, each successive feature map would get smaller after the convolution operation.

Fig. 4



A convolution operation with zero padding so as to retain in-plane dimensions. Note that an input dimension of 5 × 5 is kept in the output feature map. In this example, a kernel size and a stride are set as 3 × 3 and 1, respectively

The distance between two successive kernel positions is called a stride, which also defines the convolution operation. The common choice of a stride is 1; however, a stride larger than 1 is sometimes used in order to achieve downsampling of the feature maps. An alternative technique to perform downsampling is a pooling operation, as described below. The key feature of a convolution operation is weight sharing: kernels are shared across all the image positions. Weight sharing creates the following characteristics of convolution operations: (1) letting the local feature patterns extracted by kernels translation b invariant as kernels travel across all the image positions and detect learned local patterns, (2) learning spatial hierarchies of feature patterns by downsampling in conjunction with a pooling operation, resulting in capturing an increasingly larger field of view, and (3) increasing model efficiency by reducing the number of parameters to learn in comparison with fully connected neural networks.

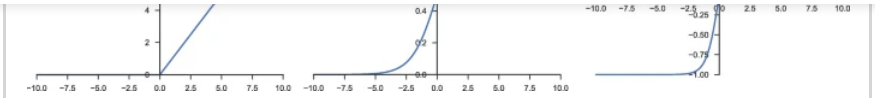
As described later, the process of training a CNN model with regard to the convolution layer is to identify the kernels that work best for a given task based on a given training dataset. Kernels are the only parameters automatically learned during the training process in the convolution layer; on the other hand, the size of the kernels, number of kernels, padding, and stride are hyperparameters that need to be set before the training process starts (Table 1).

Table 1 A list of parameters and hyperparameters in a convolutional neural network (CNN)

Nonlinear activation function

The outputs of a linear operation such as convolution are then passed through a nonlinear activation function. Although smooth nonlinear functions, such as sigmoid or hyperbolic tangent (tanh) function, were used previously because they are mathematical representations of a biological neuron behavior, the most common nonlinear activation function used presently is the rectified linear unit (ReLU), which simply computes the function: $f(x) = \max(0, x)$ (Fig. 5) [1, 3, 17,18,19].

Fig. 5



Activation functions commonly applied to neural networks: **a** rectified linear unit (ReLU), **b** sigmoid, and **c** hyperbolic tangent (tanh)

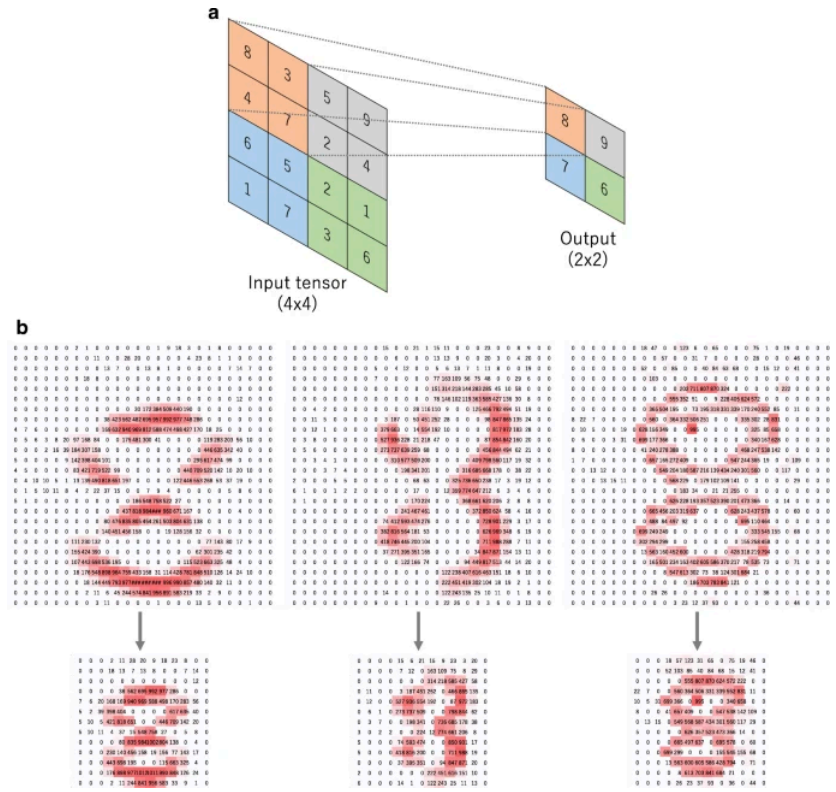
Pooling layer

A pooling layer provides a typical downsampling operation which reduces the in-plane dimensionality of the feature maps in order to introduce a translation invariance to small shifts and distortions, and decrease the number of subsequent learnable parameters. It is of note that there is no learnable parameter in any of the pooling layers, whereas filter size, stride, and padding are hyperparameters in pooling operations, similar to convolution operations.

Max pooling

The most popular form of pooling operation is max pooling, which extracts patches from the input feature maps, outputs the maximum value in each patch, and discards all the other values (Fig. 6). A max pooling with a filter of size 2 × 2 with a stride of 2 is commonly used in practice. This downsamples the in-plane dimension of feature maps by a factor of 2. Unlike height and width, the depth dimension of feature maps remains unchanged.

Fig. 6



a An example of max pooling operation with a filter size of 2 × 2, no padding, and a stride of 2, which extracts 2 × 2 patches from the input tensors, outputs the maximum value in each patch, and discards all the other values, resulting in downsampling the in-plane dimension of an input tensor by a factor of 2. **b** Examples of the max pooling operation on the same images in Fig. 3b. Note that images in the upper row are downsampled by a factor of 2, from 26 × 26 to 13 × 13

Global average pooling

elements in each feature map, whereas the depth of feature maps is retained. This operation is typically applied only once before the fully connected layers. The advantages of applying global average pooling are as follows: (1) reduces the number of learnable parameters and (2) enables the CNN to accept inputs of variable size.

Fully connected layer

The output feature maps of the final convolution or pooling layer is typically flattened, i.e., transformed into a one-dimensional (1D) array of numbers (or vector), and connected to one or more fully connected layers, also known as dense layers, in which every input is connected to every output by a learnable weight. Once the features extracted by the convolution layers and downsampled by the pooling layers are created, they are mapped by a subset of fully connected layers to the final outputs of the network, such as the probabilities for each class in classification tasks. The final fully connected layer typically has the same number of output nodes as the number of classes. Each fully connected layer is followed by a nonlinear function, such as ReLU, as described above.

Last layer activation function

The activation function applied to the last fully connected layer is usually different from the others. An appropriate activation function needs to be selected according to each task. An activation function applied to the multiclass classification task is a softmax function which normalizes output real values from the last fully connected layer to target class probabilities, where each value ranges between 0 and 1 and all values sum to 1. Typical choices of the last layer activation function for various types of tasks are summarized in Table 2.

Table 2 A list of commonly applied last layer activation functions for various tasks

Training a network

Training a network is a process of finding kernels in convolution layers and weights in fully connected layers which minimize differences between output predictions and given ground truth labels on a training dataset. Backpropagation algorithm is the method commonly used for training neural networks where loss function and gradient descent optimization algorithm play essential roles. A model performance under particular kernels and weights is calculated by a loss function through forward propagation on a training dataset, and learnable parameters, namely kernels and weights, are updated according to the loss value through an optimization algorithm called backpropagation and gradient descent, among others (Fig. 1).

Loss function

A loss function, also referred to as a cost function, measures the compatibility between output predictions of the network through forward propagation and given ground truth labels. Commonly used loss function for multiclass classification is cross entropy, whereas mean squared error is typically applied to regression to continuous values. A type of loss function is one of the hyperparameters and needs to be determined according to the given tasks.

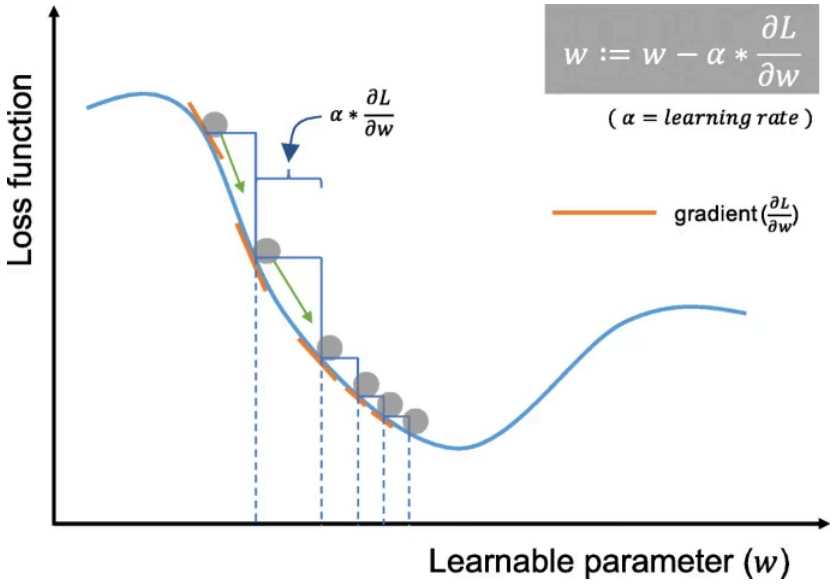
Gradient descent

Gradient descent is commonly used as an optimization algorithm that iteratively updates the learnable parameters, i.e., kernels and weights, of the network so as to minimize the loss. The gradient of the loss function provides us the direction in which the function has the steepest rate of increase, and each learnable parameter is updated in the negative direction of the gradient with an arbitrary step size determined based on a hyperparameter

$$w := w - \alpha * \frac{\partial L}{\partial w}$$

where w stands for each learnable parameter, α stands for a learning rate, and L stands for a loss function. It is of note that, in practice, a learning rate is one of the most important hyperparameters to be set before the training starts. In practice, for reasons such as memory limitations, the gradients of the loss function with regard to the parameters are computed by using a subset of the training dataset called mini-batch, and applied to the parameter updates. This method is called mini-batch gradient descent, also frequently referred to as stochastic gradient descent (SGD), and a mini-batch size is also a hyperparameter. In addition, many improvements on the gradient descent algorithm have been proposed and widely used, such as SGD with momentum, RMSprop, and Adam [21,22,23], though the details of these algorithms are beyond the scope of this article.

Fig. 7

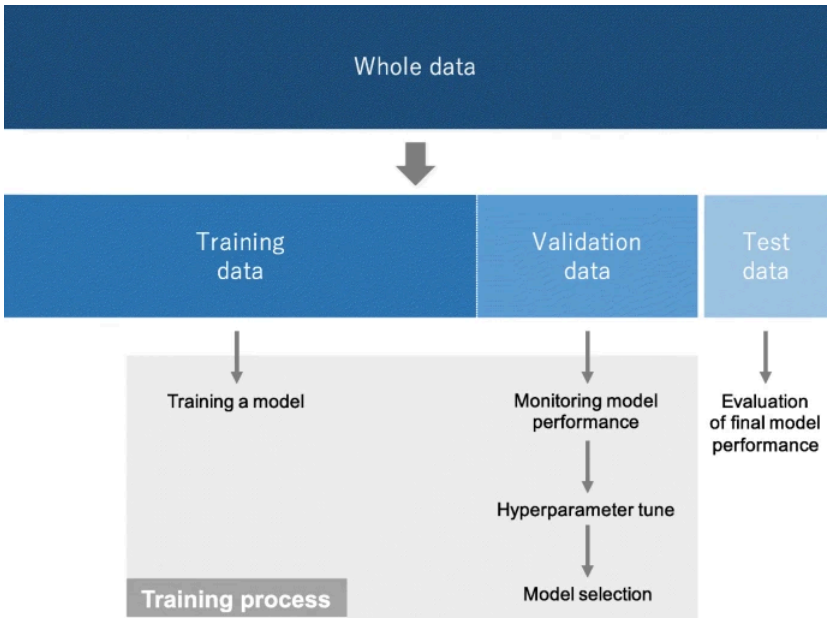


Gradient descent is an optimization algorithm that iteratively updates the learnable parameters so as to minimize the loss, which measures the distance between an output prediction and a ground truth label. The gradient of the loss function provides the direction in which the function has the steepest rate of increase, and all parameters are updated in the negative direction of the gradient with a step size determined based on a learning rate

Data and ground truth labels

Data and ground truth labels are the most important components in research applying deep learning or other machine learning methods. As a famous proverb originating in computer science notes: “Garbage in, garbage out.” Careful collection of data and ground truth labels with which to train and test a model is mandatory for a successful deep learning project, but obtaining high-quality labeled data can be costly and time-consuming. While there may be multiple medical image datasets open to the public [24, 25], special attention should be paid in these cases to the quality of the ground truth labels.
Available data are typically split into three sets: a training, a validation, and a test set (Fig. 8), though there are some variants, such as cross validation. A training set is used to train a network, where loss values are calculated via forward propagation and learnable parameters are updated via backpropagation. A validation set is used to evaluate the model during the training process, fine-tune hyperparameters, and perform model selection. A test set is ideally used only once at the very end of the project in order to evaluate the

Fig. 8



Available data are typically split into three sets: a training, a validation, and a test set. A training set is used to train a network, where loss values are calculated via forward propagation and learnable parameters are updated via backpropagation. A validation set is used to monitor the model performance during the training process, fine-tune hyperparameters, and perform model selection. A test set is ideally used only once at the very end of the project in order to evaluate the performance of the final model that is fine-tuned and selected on the training process with training and validation sets

Separate validation and test sets are needed because training a model always involves fine-tuning its hyperparameters and performing model selection. As this process is performed based on the performance on the validation set, some information about this validation set leaks into the model itself, i.e., overfitting to the validation set, even though the model is never directly trained on it for the learnable parameters. For that reason, it is guaranteed that the model with fine-tuned hyperparameters on the validation set will perform well on this same validation set. Therefore, a completely unseen dataset, i.e., a separate test set, is necessary for the appropriate evaluation of the model performance, as what we care about is the model performance on never-before-seen data, i.e., generalizability.

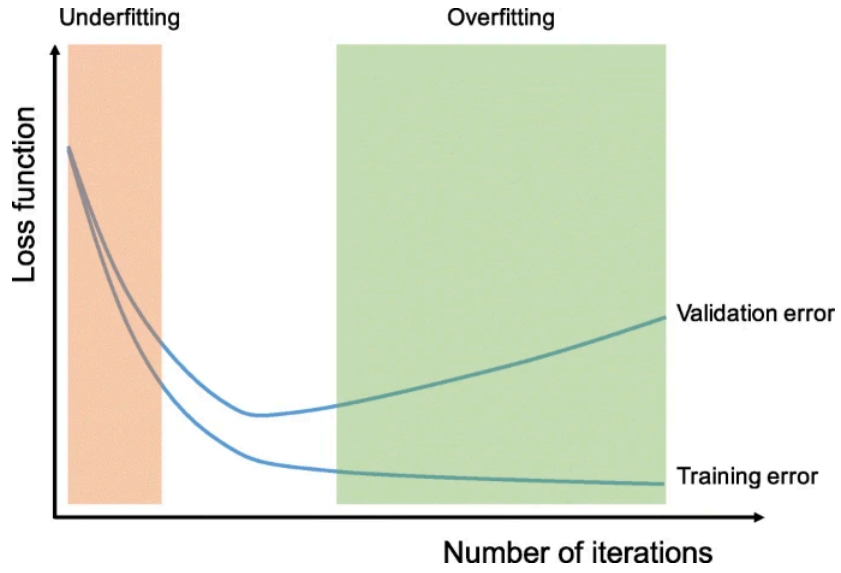
It is worthy of mention that the term “validation” is used differently in the medical field and the machine learning field [26]. As described above, in machine learning, the term “validation” usually refers to a step to fine-tune and select models during the training process. On the other hand, in medicine, “validation” usually stands for the process of verifying the performance of a prediction model, which is analogous to the term “test” in machine learning. In order to avoid this confusion, the word “development set” is sometimes used as a substitute for “validation set”.

Overfitting

Overfitting refers to a situation where a model learns statistical regularities specific to the training set, i.e., ends up memorizing the irrelevant noise instead of learning the signal, and, therefore, performs less well on a subsequent new dataset. This is one of the main challenges in machine learning, as an overfitted model is not generalizable to never-seen-before data. In that sense, a test set plays a pivotal role in the proper performance evaluation of machine learning models, as discussed in the previous section. A routine check for recognizing overfitting to the training data is to monitor the loss and accuracy on the training and validation sets (Fig. 9). If the model performs well on the training set

larger dataset typically generalizes better, though that is not always attainable in medical imaging. The other solutions include regularization with dropout or weight decay, batch normalization, and data augmentation, as well as reducing architectural complexity. Dropout is a recently introduced regularization technique where randomly selected activations are set to 0 during the training, so that the model becomes less sensitive to specific weights in the network [27]. Weight decay, also referred to as L2 regularization, reduces overfitting by penalizing the model's weights so that the weights take only small values. Batch normalization is a type of supplemental layer which adaptively normalizes the input values of the following layer, mitigating the risk of overfitting, as well as improving gradient flow through the network, allowing higher learning rates, and reducing the dependence on initialization [28]. Data augmentation is also effective for the reduction of overfitting, which is a process of modifying the training data through random transformations, such as flipping, translation, cropping, rotating, and random erasing, so that the model will not see exactly the same inputs during the training iterations [29]. In spite of these efforts, there is still a concern of overfitting to the validation set rather than to the training set because of information leakage during the hyperparameter fine-tuning and model selection process. Therefore, reporting the performance of the final model on a separate (unseen) test set, and ideally on external validation datasets if applicable, is crucial for verifying the model generalizability.

Fig. 9



A routine check for recognizing overfitting is to monitor the loss on the training and validation sets during the training iteration. If the model performs well on the training set compared to the validation set, then the model has been overfit to the training data. If the model performs poorly on both training and validation sets, then the model has been underfit to the data. Although the longer a network is trained, the better it performs on the training set, at some point, the network fits too well to the training data and loses its capability to generalize

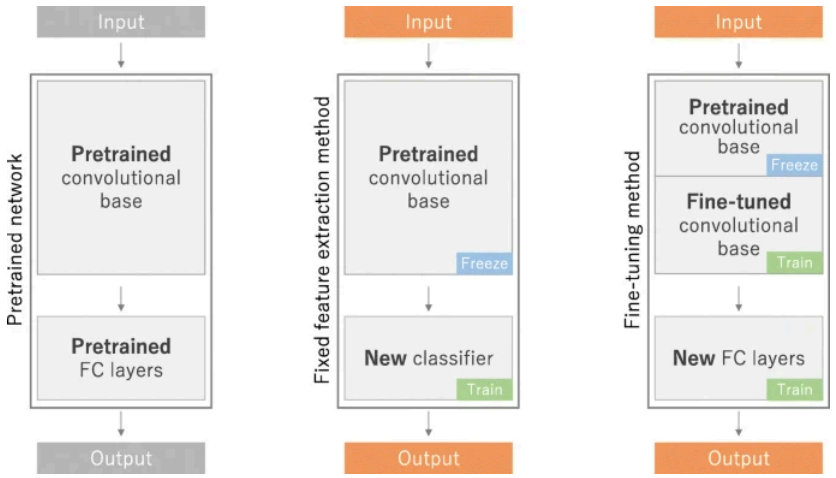
Table 3 A list of common methods to mitigate overfitting

Training on a small dataset

An abundance of well-labeled data in medical imaging is desirable but rarely available due to the cost and necessary workload of radiology experts. There are a couple of techniques available to train a model efficiently on a smaller dataset: data augmentation and transfer

where a network is pretrained on an extremely large dataset, such as ImageNet, which contains 1.4 million images with 1000 classes, then reused and applied to the given task of interest. The underlying assumption of transfer learning is that generic features learned on a large enough dataset can be shared among seemingly disparate datasets. This portability of learned generic features is a unique advantage of deep learning that makes itself useful in various domain tasks with small datasets. At present, many models pretrained on the ImageNet challenge dataset are open to the public and readily accessible, along with their learned kernels and weights, such as AlexNet [3], VGG [30], ResNet [31], Inception [32], and DenseNet [33]. In practice, there are two ways to utilize a pretrained network: fixed feature extraction and fine-tuning (Fig. 10).

Fig. 10



Transfer learning is a common and effective strategy to train a network on a small dataset, where a network is pretrained on an extremely large dataset, such as ImageNet, then reused and applied to the given task of interest. A fixed feature extraction method is a process to remove FC layers from a pretrained network and while maintaining the remaining network, which consists of a series of convolution and pooling layers, referred to as the convolutional base, as a fixed feature extractor. In this scenario, any machine learning classifier, such as random forests and support vector machines, as well as the usual FC layers, can be added on top of the fixed feature extractor, resulting in training limited to the added classifier on a given dataset of interest. A fine-tuning method, which is more often applied to radiology research, is to not only replace FC layers of the pretrained model with a new set of FC layers to retrain them on a given dataset, but to fine-tune all or part of the kernels in the pretrained convolutional base by means of backpropagation. FC, fully connected

A fixed feature extraction method is a process to remove fully connected layers from a network pretrained on ImageNet and while maintaining the remaining network, which consists of a series of convolution and pooling layers, referred to as the convolutional base, as a fixed feature extractor. In this scenario, any machine learning classifier, such as random forests and support vector machines, as well as the usual fully connected layers in CNNs, can be added on top of the fixed feature extractor, resulting in training limited to the added classifier on a given dataset of interest. This approach is not common in deep learning research on medical images because of the dissimilarity between ImageNet and given medical images.

A fine-tuning method, which is more often applied to radiology research, is to not only replace fully connected layers of the pretrained model with a new set of fully connected layers to retrain on a given dataset, but to fine-tune all or part of the kernels in the pretrained convolutional base by means of backpropagation. All the layers in the convolutional base can be fine-tuned or, alternatively, some earlier layers can be fixed

a particular dataset or task [34, 35].

One drawback of transfer learning is its constraints on input dimensions. The input image has to be 2D with three channels relevant to RGB because the ImageNet dataset consists of 2D color images that have three channels (RGB: red, green, and blue), whereas medical grayscale images have only one channel (levels of gray). On the other hand, the height and width of an input image can be arbitrary, but not too small, by adding a global pooling layer between the convolutional base and added fully connected layers.

There has also been increasing interest in taking advantage of unlabeled data, i.e., semi-supervised learning, to overcome a small-data problem. Examples of this attempt include pseudo-label [36] and incorporating generative models, such as generative adversarial networks (GANs) [37]. However, whether these techniques can really help improve the performance of deep learning in radiology is not clear and remains an area of active investigation.

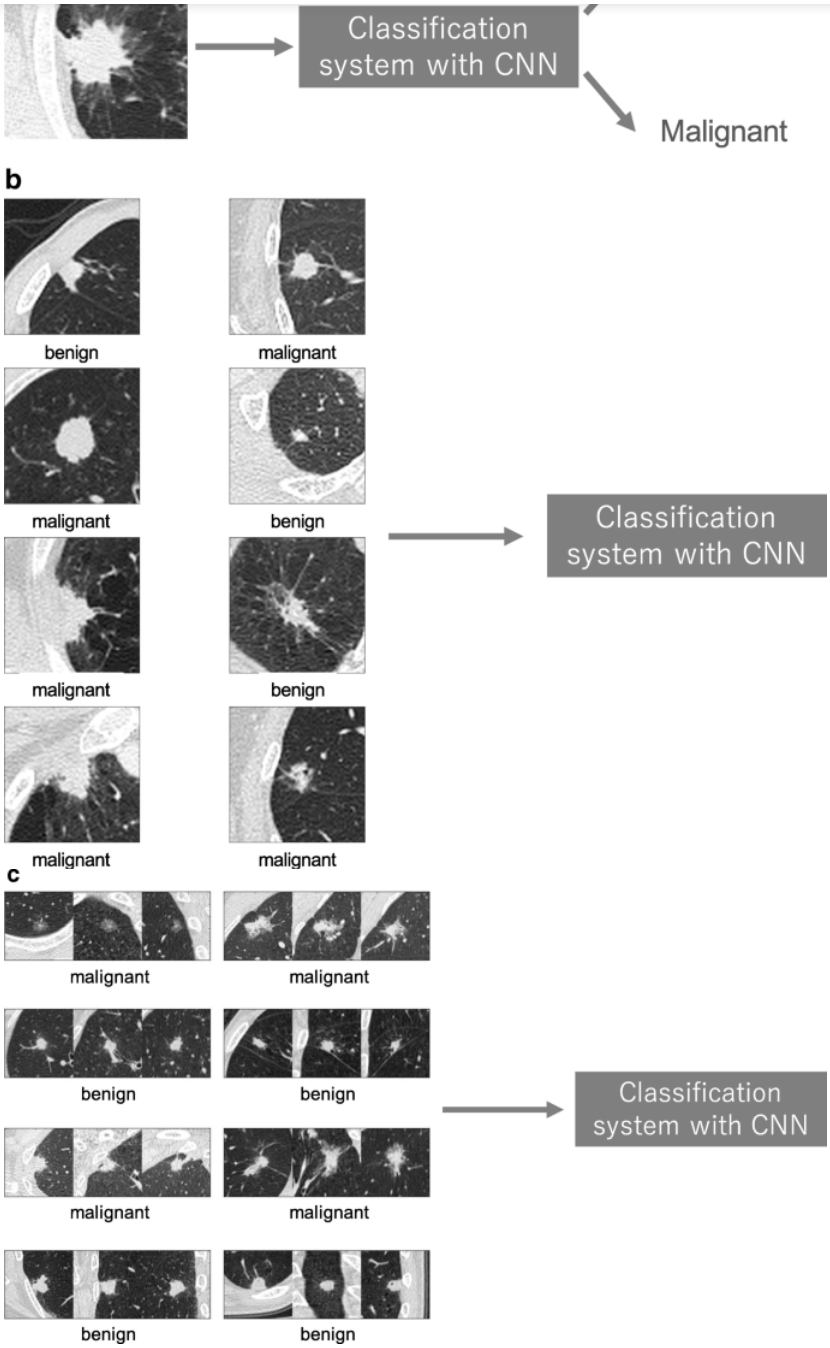
Applications in radiology

This section introduces recent applications within radiology, which are divided into the following categories: classification, segmentation, detection, and others.

Classification

In medical image analysis, classification with deep learning usually utilizes target lesions depicted in medical images, and these lesions are classified into two or more classes. For example, deep learning is frequently used for the classification of lung nodules on computed tomography (CT) images as benign or malignant (Fig. 11a). As shown, it is necessary to prepare a large number of training data with corresponding labels for efficient classification using CNN. For lung nodule classification, CT images of lung nodules and their labels (i.e., benign or cancerous) are used as training data. Figure 11b, c show two examples of training data of lung nodule classification between benign lung nodule and primary lung cancer; Fig. 11b shows the training data where each datum includes an axial image and its label, and Fig. 11c shows the training data where each datum includes three images (axial, coronal, and sagittal images of a lung nodule) and their labels. After training CNN, the target lesions of medical images can be specified in the deployment phase by medical doctors or computer-aided detection (CADe) systems [38].

Fig. 11



A schematic illustration of a classification system with CNN and representative examples of its training data. **a** Classification system with CNN in the deployment phase. **b**, **c** Training data used in training phase

Because 2D images are frequently utilized in computer vision, deep learning networks developed for the 2D images (2D-CNN) are not directly applied to 3D images obtained in radiology [thin-slice CT or 3D-magnetic resonance imaging (MRI) images]. To apply deep learning to 3D radiological images, different approaches such as custom architectures are used. For example, Setio et al. [39] used a multistream CNN to classify nodule candidates of chest CT images between nodules or non-nodules in the databases of the Lung Image Database Consortium and Image Database Resource Initiative (LIDC-IDRI) [40], ANODE09 [41], and the Danish Lung Cancer Screening Trial [42]. They extracted differently oriented 2D image patches based on multiplanar reconstruction from one nodule candidate (one or nine patches per candidate), and these patches were used in