Deep Learning Models with Applications to Brain Image Analysis

15

Dinggang Shen, Luping Zhou, and Mingxia Liu

Abstract

Deep learning, rooted in artificial neural networks, has received increasing attention in the field of brain image analysis. In this chapter, the pre-processing steps for brain images and the fundamental concepts of deep neural networks are first introduced. After that, four typical types of deep neural networks used for brain image analysis are elaborated, including (i) convolutional neural networks (CNNs) and the variants (i.e., fully convolutional networks and U-net), (ii) recurrent neural networks (RNNs) and the variant (i.e., long short-term memory model), (iii) auto-encoder, and (iv) generative adversarial networks (GANs) and the variants (i.e., Pix2Pix GAN and CycleGAN), as well as their applications in brain image classification, segmentation, registration, and image synthesis/augmentation. In addition, several challenges and future research directions of deep learning in brain image analysis are also pointed out.

Keywords

Deep learning · Brain image analysis · Convolutional neural network · Classification · Segmentation · Registration · Synthesis/augmentation

15.1 Background

The significant advances of neuroimaging techniques have deeply reshaped modern neuroscience in recent decades, offering researchers unprecedented opportunities to noninvasively investigate the anatomy and functions of the brain. The imaging-based measurements are heralded more sensitive and consistent than the traditional cognitive tests, thus critical for the early diagnosis of brain disorders, e.g., Alzheimer's disease (AD) and schizophrenia.

As a relatively new discipline within medicine, neuroscience, and psychology, neuroimaging falls into two broad categories: (1) structural imaging, such as magnetic resonance imaging (MRI), and (2) functional imaging, such as functional magnetic resonance imaging (fMRI) and positron emission tomography (PET). Neuroimaging-based studies not only focus on investigating how the brain is organized around regions (such as local morphometry and functions) but may also consider the

D. Shen (✉) · M. Liu
Department of Radiology and BRIC, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA
e-mail: dgshen@med.unc.edu

L. Zhou
School of Electrical and Information Engineering, The University of Sydney, Sydney, NSW, Australia

B. He (ed.), *Neural Engineering*, https://doi.org/10.1007/978-3-030-43395-6_15

connections between brain regions, in order to identify imaging-based biomarkers for automated diagnosis of brain diseases. Such tasks could be very challenging, giving the highly non-linear mapping between image-based observations and diagnosis outputs. Recently this research field significantly benefits from the emerging deep learning techniques that have demonstrated excellent performance in learning non-linear function mapping in various tasks.

As part of a broader family of machine learning methods, deep learning models are generally based on artificial neural networks. Even though artificial neural networks were proposed in the 1950s, their development still suffered from some limitations, such as lack of computing power, lack of sufficient training data, and difficulty in training deep networks. The rapid development of deep learning in recent years can be attributed to the enhanced computer capabilities of high-tech central processing units (CPUs) and graphics processing units (GPUs), the availability of big data, and the novel algorithms for training deep neural networks. Deep learning algorithms can be supervised (i.e., learning with only labeled data), semi-supervised (i.e., learning with both labeled and unlabeled data), or unsupervised (i.e., learning with only unlabeled data) [1–3]. In particular, deep learning models such as convolutional neural network (CNN), recurrent neural network (RNN), auto-encoder (AE), and generative adversarial network (GAN) have rapidly become a methodology of choice for analyzing brain images in various applications [4,5], such as brain image segmentation [6–8], brain image registration [9], brain disease diagnosis [10–12], and brain image synthesis [13–15]. However, there are several challenges for deep learning models in dealing with neuroimages, as summarized below:

(1) **Small-sample-size problem**. In the domain of brain image analysis, deep learning models generally suffer from the small-sample-size problem, because there are millions of voxels in each 3D volume and a very limited number (e.g., tens or hundreds) of subjects/images for model training. A popular solution is to locate regions of interest (ROIs) in brain images using prior knowledge or data-driven strategies [10]. However, it is usually challenging to define such ROIs in each 3D brain image.
(2) **Missing data problem**. The missing data problem is usually unavoidable in the field of brain image analysis (especially for multi-modality applications) [15], because subjects may lack some modalities (e.g., PET) due to patient dropouts and/or poor data quality. Conventional methods typically discard data-missing subjects, but this will significantly reduce the number of training subjects and affect the robustness of the learned model.
(3) **Spatiotemporal dynamics of the brain**. Previous studies on brain functional connectivity have shown that the human brain is intrinsically organized into spatiotemporal dynamic interaction network [16, 17], demonstrating remarkable spatiotemporal variability over time in its function and structure [18, 19]. Hence, it's essential to model the spatiotemporal dynamics of brain images to improve the performance of deep learning models.

To address these issues, various deep learning models have been proposed for analyzing brain images, resulting in promising results in different applications. In the following, the pre-processing steps for brain images and fundamental concepts of neural networks will be first introduced in Sect. 15.2. Then, four typical deep neural networks for brain image analysis will be presented in Sects. 15.3, 15.4, 15.5, and 15.6. Section 15.7 presents the limitations of current deep learning models and several possible future research directions. Finally, Sect. 15.8 concludes this chapter.

15.2 Image Processing and Concept of Deep Learning

15.2.1 Brain Image Pre-processing

Brain images usually need to be pre-processed so that the acquisition artifacts and undesired tissues could be removed to better serve subsequent tasks. In the following, a typical pipeline

to pre-process structural brain MR images is introduced, since MRI is the most widely used imaging modality to explore human brains. The pipeline includes the steps of skull stripping, bias field correction, intensity normalization, and spatial registration [20]. Based on the characteristic of the analysis task, some steps could be optional.

Skull stripping is used to remove non-brain tissues from MR images. The existence of non-brain tissues, such as skull and eyes, could negatively affect the algorithms for the subsequent segmentation or diagnosis. Therefore, skull stripping is used for pre-processing in many brain image analysis models, including those deep learning-based ones. Either under- or over-segmentation of the brain will lead to inaccurate estimation of brain tissues. A commonly used skull stripping method is based on BET [21, 22].

A common artifact seen in an MR image is the smooth variation of signal intensity across the image, called as bias field. This may be caused by factors such as poor radio-frequency field uniformity and eddy currents driven by the switching of field gradients, etc. This intensity non-uniformity is known as "bias field" which is the low-frequency multiplicative noise in the images. Many methods have been proposed for the bias field correction by estimating both the uncorrupted image intensities and the spatially smooth and multiplicative model of the bias field. Among these methods, a typical representative is the non-parametric non-uniform intensity normalization (N3) algorithm [23].

MRI scans are acquired in arbitrary units, making them not amiable for the comparison of the same tissue across different studies of a same subject or across different subjects. Such intensity difference could make simple operations such as thresholding difficult across images. Therefore, intensity normalization is sometimes involved for pre-processing MR images. Methods for intensity normalization usually manipulate the histograms of MR images so that they are aligned after the normalization, i.e., the discrepancy between histograms is minimized. However, pre-processing images with intensity normalization needs to be done carefully to avoid the elimination of critical or discriminative information carried in image intensities.

Spatial registration, which transforms images into a common coordinate space, is often needed when integrating multiple imaging modalities/MR sequences for analysis. The transformation could be linear (such as translation, rotation, scaling and other affine transforms) or non-linear, which could locally warp images to match them. Image registration by itself is a big category of research problems [24, 25].

Many off-the-shelf toolkits provide basic tools for the pre-processing of brain images, for example, FMRIB Software Library (FSL)[1] and Statistical Parametric Mapping (SPM).[2] Such pre-processing steps are often indispensable in traditional non-deep learning-based models. For deep learning models, despite their strong capacity of learning highly non-linear function mapping, they could still benefit from these pre-processing steps to reduce the complexity of the learning task, especially when there are only a limited number of images for training.

15.2.2 Fundamentals About Neural Network Models

Deep learning is rooted in artificial neural networks. An artificial neural network consists of multiple layers of interconnected processing units, known as *neurons*. If the connections between neurons do not form a circle, the artificial neural network is *feed-forward*. The most common class of feed-forward neural network is known as multilayer perception (MLP). An MLP has at least three layers, (1) an input layer, (2) a hidden layer, and (3) an output layer, as shown in Fig. 15.1. The network takes data from the input layer, non-linearly transforms the data via the hidden layer, and produces the prediction at the output layer. In MLP, neurons in the neighboring layers are fully connected, while neurons within the same layer have no connection. Multiple hidden layers can be stacked to increase the non-linearity of the model.

Specifically, an MLP learns an embedding function $\mathbf{y} = f(\mathbf{x}; \mathbf{w})$, where $\mathbf{x}$ is the input, $\mathbf{y}$

[1] https://fsl.fmrib.ox.ac.uk/fsl/fslwiki

[2] https://www.fil.ion.ucl.ac.uk/spm/

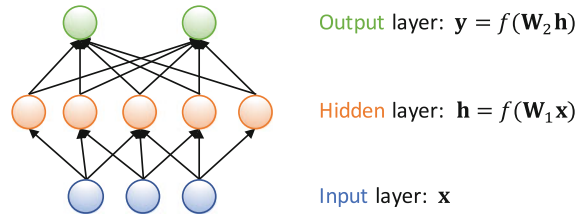


Fig. 15.1 A basic multilayer perception network (MLP)

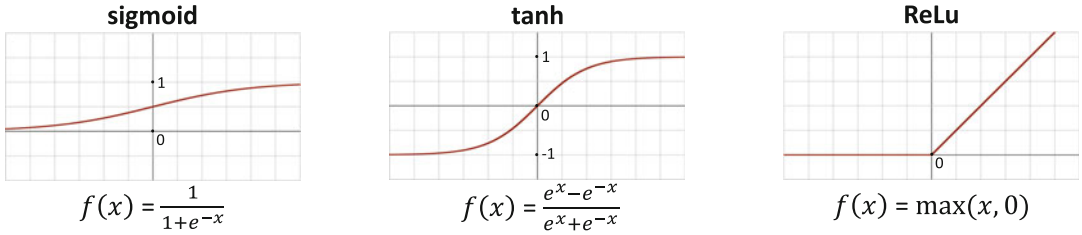


Fig. 15.2 Common activation functions: sigmoid (left), hyperbolic tangent (middle), and rectified linear unit (right)

is the output, $f = (f_n \circ f_{n-1} \cdots \circ f_1)(\mathbf{x})$ is a set of non-linear transforms parameterized by $\mathbf{w}$, and n denotes the number of layers. Usually, a non-linear transform f_i at the i-th layer takes the form of $f_i = \sigma_i(\mathbf{w}_i{}^\top f_{i-1})$, consisting of a linear transform on the output of the previous layer f_{i-1} and a following non-linear activation function σ_i. The linear transform matrix $\mathbf{w}_i$ is called the *parameters* or *weights* of the model, which is automatically learned during training. The commonly used activation function σ_i includes sigmoid function, hyperbolic tangent function (tanh), and rectified linear unit (ReLu), as shown in Fig. 15.2.

Training and testing neural networks require non-overlapped training and test datasets. The training set consists of N paired input and expected output samples $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$. An MLP is trained to minimize the difference between its prediction and the expected output by optimizing the *loss function* $\mathcal{L}$. For example, $\mathcal{L} = \min_{\mathbf{w}} \sum_{i=1}^N \frac{1}{N} \parallel \mathbf{y}_i - f(\mathbf{x}_i; \mathbf{w}) \parallel^2$. The optimal model parameters $\mathbf{w}^*$ can be effectively attained by a family of methods known as "backpropagation." The basic idea is to exploit chain rule to first compute the gradient of the loss function with respect to each model parameter $\mathbf{w}_{ij}$ in $\mathbf{w}$ and then update $\mathbf{w}_{ij}$ using the direction of gradient descent iteratively as

$$\mathbf{w}_{ij}^{t+1} = \mathbf{w}_{ij}^t - \alpha \frac{\partial \mathcal{L}(\mathbf{w}_{ij}^t)}{\partial \mathbf{w}_{ij}}, \tag{15.1}$$

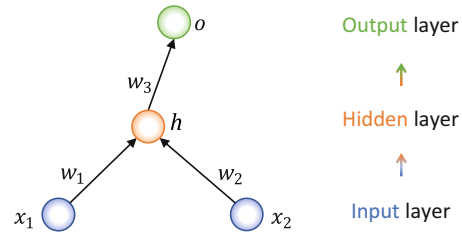


Fig. 15.3 An example for backpropagation

where α is the user-predefined *learning rate* and t is the index of iteration. Repeating in this way, the value of the loss function will be gradually reduced until a certain stopping criterion can be met. In the test stage, the test set (unseen in the training stage) is simply fed forward through the learned neural network model, using $\mathbf{w}^*$ for prediction.

An example for backpropagation Figure 15.3 shows an MLP consisting of an input layer with two scalar input variables x_1 and x_2, a hidden layer with the output h, and an output layer with the output o. The loss function of the MLP is $\mathcal{L} = \sum_i (o_i - y_i)^2$, where y_i indicates ground-truth and

$$h_i = f_1(x_{1,i}, x_{2,i}; w_1, w_2) = w_1 x_{1,i} + w_2 x_{2,i},$$
$$o_i = f_2(h_i) = w_3 h_i.$$

Solution Applying the chain rule, we have the following solution

$$\frac{\partial \mathcal{L}}{\partial w_3} = \sum_i \frac{\partial \mathcal{L}}{\partial o_i}\frac{\partial o_i}{\partial w_3} = \sum_i 2(o_i - y_i)h_i,$$

$$\frac{\partial \mathcal{L}}{\partial w_2} = \sum_i \frac{\partial \mathcal{L}}{\partial o_i}\frac{\partial o_i}{\partial h_i}\frac{\partial h_i}{\partial w_2} = \sum_i 2(o_i - y_i)w_3 x_{2,i},$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \sum_i \frac{\partial \mathcal{L}}{\partial o_i}\frac{\partial o_i}{\partial h_i}\frac{\partial h_i}{\partial w_1} = \sum_i 2(o_i - y_i)w_3 x_{1,i}.$$

Gradient descent computed in backpropagation can be trained in three ways: (1) batch, (2) stochastic, and (3) mini-batch. In gradient descent, *batch* means the total number of samples used to update the gradient in one iteration. A large batch (e.g., the entire training set) may even cause a single iteration to take a long time to compute. On the contrary, *stochastic gradient descent* (SGD) uses a single training sample to calculate the objective loss and update the gradient for each iteration. The increased frequency of model update may lead to faster learning in some problems, but also bring noisy gradient estimation. *Mini-batch stochastic gradient descent* (mini-batch SGD) balances the full-batch training and SGD. It splits the training set into small batches and uses these small batches to calculate objective loss and update the model. Mini-batch SGD improves the efficiency of the full-batch training and reduces the noise in SGD, which is commonly used for training deep learning models.

The following terms are related to mini-batch training. *Mini-batch size* refers to the number of training samples in one mini-batch used to update the model. The number of *epochs* refers to the times that the entire training set is passed forward and backward through the neural network model. The number of *iterations* refers to the number of passes using the samples of the mini-batch size, where each pass includes a forward pass and a backward pass. For example, if the training set contains 1,000 samples, and the mini-batch size is 50, it then takes 20 iterations to complete one epoch.

15.3 Convolutional Neural Networks

MLPs have some drawbacks when they are used for image processing. They use one neuron for each input (e.g., a pixel in an image), and every neuron connects to all neurons in the next layer. This makes the parameters of the model increase dramatically when the size of the image is relatively large. Also, flattening an image to MLP causes the loss of spatial information in the image. Instead, convolutional neural networks (CNNs) are the most commonly used deep learning models in medical image analysis. They are biologically inspired variants of MLPs, utilizing local receptive fields, weight sharing, and sparse connectivity to preserve spatial information and reduce the number of network parameters.

Receptive field Being 2D or 3D grids, images form high-dimensional input to neural networks. It is inefficient to fully connect a neuron with every pixel/voxel in this case. Instead, as known, pixels/voxels are mostly useful in the context of their neighbors. Therefore, in CNNs, a neuron is connected only to a local region of the input grid. This input region is known as the receptive field of the neuron. It can be described by its centroid location and size. In CNNs, the receptive field can be increased by stacking more layers.

Weight sharing At every layer in CNNs, filters are applied to detect the presence of specific features or patterns. These filters act on the receptive field of the input image. The numbers within each filter are called weights. Weight sharing happens across the filters in a particular layer. That is, when the filter moves through the image, its weights do not change. The idea behind is that if a detected pattern is important in a particular region of the image, it may be important in other regions of the image too.

Sparse connectivity In CNNs, filters are convolved with the input image to calculate the activation of neurons. When the size of the filter,

or equally the receptive field, is smaller than the input volume, each neuron in the activation map is only connected to a local region of the input, leading to "sparse" (rather than "dense") connectivity across layers.

15.3.1 CNN Fundamentals

A CNN model consists of convolutional layers interspersed with pooling layers, on top of which are often fully connected layers as in standard MLPs. A typical example of CNN, i.e., LeNet, is shown in Fig. 15.4.

Convolutional layers In convolutional layers, the input of the layer is convolved with stacks of filters of predefined size. The weights of filters are automatically learned by optimizing CNN via backpropagation. Filters are also known as convolutional *kernels*. The output of convolution is the sum of element-wise multiplication between pixels in the receptive field and weights in the convolution filter, as shown in Fig. 15.5. Each filter moves across all input locations via a step called *stride* and uses the same weights for convolution. This produces a feature map. In other words, a *feature map* is formed by units that share the same weights and bias in a convolutional layer. The stack of filters produces a tensor of feature maps. The feature maps are further sent through a non-linear activation function, such as ReLu (See Fig. 15.2), to model non-linear mapping and produce activation maps.

Pooling layers After convolutional layer, there is typically a pooling layer to down-sample the feature maps produced in the convolutional layer. By *pooling*, each small region in a feature map is summarized into a single value. There are two common methods for pooling: *max-pooling* and *average-pooling*. In *max-pooling*, a small region is represented by the maximum value inside it. In *average-pooling*, a small region is represented by the average of all values inside it. Figure 15.6 illustrates the ideas of max-pooling and average-pooling. The down-sampled feature maps could

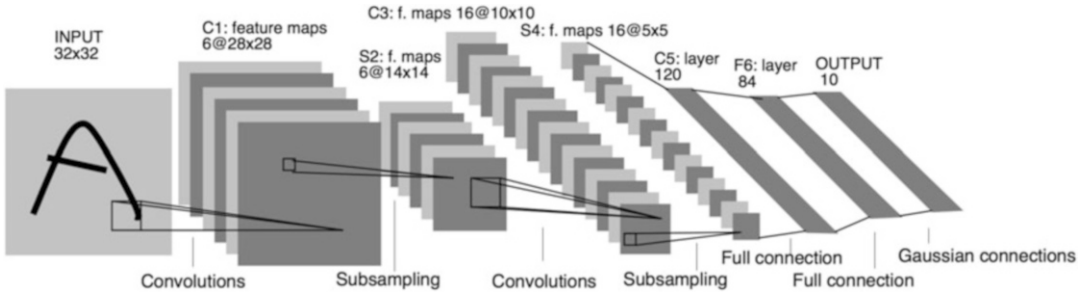


Fig. 15.4 LeNet: an example of CNN

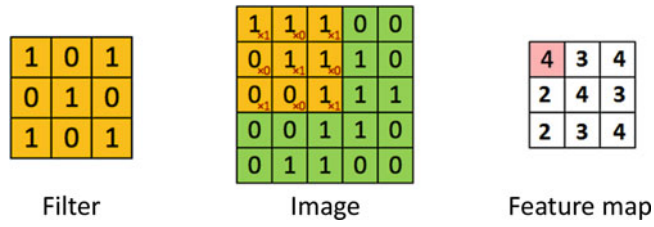


Fig. 15.5 Illustration of convolution operation in 2D. When a filter is convolved with a local region (receptive field) in the image, the element-wise multiplication between the weights in the filter and the pixel values in the region is calculated, and these multiplications are summed up to produce a value in the feature map

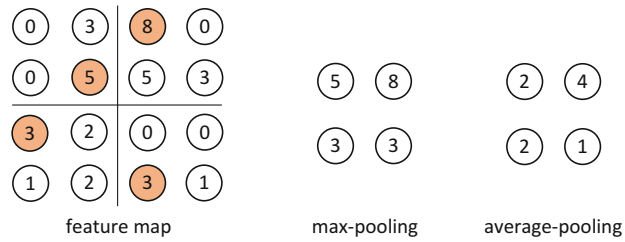


Fig. 15.6 Illustration of max-pooling and average-pooling

be more robust to small changes in the positions of the features. Therefore, pooling could give the network model a certain degree of "translation invariance." Another way to down-sample feature maps is to conduct convolution with the "stride" larger than 1.

Fully connected layers When CNNs are used for classification, following multiple convolution-pooling blocks, there are usually several fully connected (FC) layers that flatten feature maps for prediction. FC layers are in principle the same as the traditional MLP: every neuron in one layer is connected to every neuron in another layer. For example, in LeNet (see Fig. 15.4), there are two FC layers on top of the convolution-pooling blocks. The output of the final FC layer is sent through a softmax function to classify the image with probabilistic values between 0 and 1.

Dropout Deep models may contain a large number of parameters to learn, and they have a lot of freedom to fit complex datasets. This may lead to an *overfitting* problem, i.e., the learned model fits well to the training data but fails to generalize well to unseen test data. It is known that the ensembles of neural networks with different model configurations can reduce overfitting. This can be achieved in a single model by "dropout" (i.e., randomly dropping out neurons during training). When a neuron is dropped out, it is temporally removed from the network with all its incoming and outgoing connections. This leads to slightly different network for each batch of training data, which effectively reduces overfitting. Dropout takes place only in the training stage, but not in the test stage.

15.3.2 CNN Variants

Based on CNNs, many novel network architectures have been proposed to enable the network model to go deeper with less parameters and better performance. For example, the residual module [26] and inception module in [27] have been proved effective in both general and medical image classification. Meanwhile, it is noticed that when CNN models are used for prediction at pixel level (e.g., segmentation and synthesis), fully convolutional networks (FCNs) [28] without using fully connected layers on top of convolutional blocks demonstrate several advantages over the traditional CNNs, such as allowing input images with different image sizes and being more efficient than patch-wise training for pixel-level prediction. In the following, residual CNNs and a typical FCN model called U-net [29] are introduced, respectively, as well as different ways to combine CNNs for analysis.

15.3.3 Residual Learning Based on CNN

When neural networks go deep, they are expected to become more powerful and better approximate complicated functions. However, it turns out that they often encounter the *gradient vanishing* problem. That is, the gradient of the loss function approaches zero, especially at the shallow layers (the layers closer to the input) during backpropagation. In this case, we may observe the performance saturates or even degrades when more layers are added. To mitigate this problem, the residual module is proposed. As illustrated in Fig. 15.7, in the residual module, the inputs $\mathbf{x}$ from the previous layers are directly connected

with the output $f(\mathbf{x})$ of the new convolutional layers in the module, which is known as *residual connection*. They are then added to approximate the target output y. In other words, in common CNNs, we directly learn the output y, while in the residual setting we learn the difference (residual) between the output and the input: $f(\mathbf{x}) = \mathbf{y} - \mathbf{x}$. In this way, the gradient of the input can be better preserved, as $\mathbf{x}$ is transmitted by the identity matrix without loss of information. The residual modules can be stacked to form very deep neural networks, such as ResNet (Fig. 15.8) that is able to train up to hundreds or even thousands of layers with compelling performance.

15.3.4 Fully Convolutional Networks and U-Net

Common CNN models with fully connected layers can be used for pixel-level prediction, such as segmentation and synthesis. This is usually conducted in a patch-wise manner, i.e., the CNN takes image patch(es) extracted around every pixel as input and predicts the label of the patch centroid. Such approaches have some potential disadvantages. First, fully connected layers consist of predetermined number of neurons, which constrains the size of the input image. That is, all input images have to be rescaled to the same size that is predefined. Second, by flattening the feature map output from a convolutional layer into a fully connected layer, some spatial information is lost. Third, predicting pixel label in a patch-wise manner could be very time-consuming as the prediction needs to go through pixels one by one.

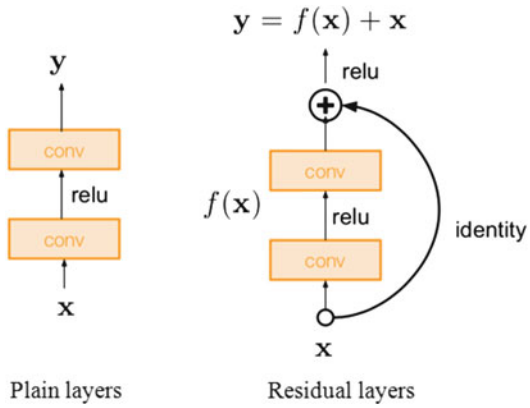


Fig. 15.7 Illustration of the difference between the plain layers (left) and the residual module (right)

Recently, fully convolutional networks (FCNs) [28] are proposed to better deal with per-pixel prediction tasks and demonstrate promising performance, especially in semantic segmentation. Different from CNNs used in patch-wise manner that predicts the label of one pixel each time, FCNs can produce dense outputs from the input images of arbitrary size. For example, FCNs can generate segmentation map that has the same size as the input image at one shot, by concatenating convolutional layers and deconvolutional layers. *Deconvolution* is simply backward strided convolution. Similar to convolutional layers that down-sample the feature maps, deconvolutional layers upsample the feature maps. Figure 15.9 illustrates the deconvolution with 3×3 kernel using stride one on the feature map with size 2×2.

A typical FCN widely used in medical image analysis is U-net [29], as shown in Fig. 15.10. It consists of the contracting and expanding paths. The contracting path consists of convolutional layers, while the expanding path consists of deconvolutional layers. They share the same number of layers. Between these two paths, multiple skip connections are used to link the corre-

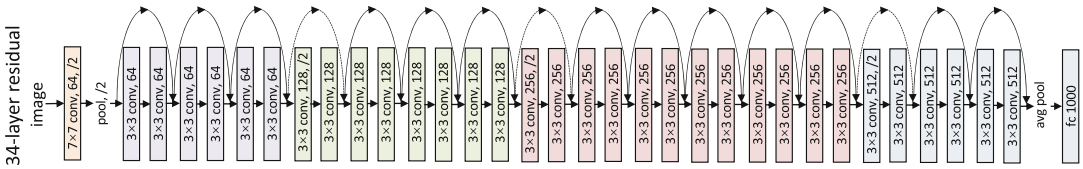


Fig. 15.8 Network architecture of ResNet with 32 layers

sponding convolutional and deconvolutional layers. Similar to the residual module, using the skip connections, U-net can mitigate the common gradient vanishing problem during the training of deep learning models, as the gradient of the deeper layers can be directly backpropagated to the shallower layers via the skip connections. Moreover, this structure allows U-net to acquire multi-depth information of the input image. In this way, it can preserve the contextual information from the input as well as the spatial details in the feature maps of shallow layers, forming a hierarchy of visual clues.

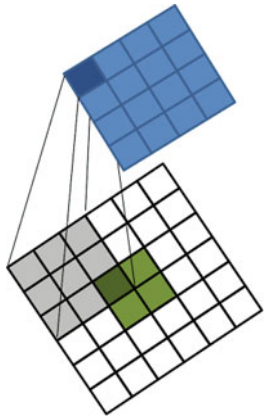


Fig. 15.9 Illustration of deconvolution (kernel size 3×3) using stride one. The original input is only the 2×2 green region, while the white region is filled with zeros

15.3.5 Combination of CNNs

CNN models could also be combined sequentially or in parallel for analysis. When combined sequentially, the output of the preceding network becomes the input of the successive network. Such combination is often used to gradually refine the output results. When combined in parallel, the networks are used to process the same input, and their outputs are integrated for decision. Such combination is often used to extract complementary features from the input image. For example, a cascade of networks is proposed (see Fig. 15.11) in [30]. It sequentially combines CNNs to segment the regions of interest from coarse to fine, including (1) segmentation of whole tumor, (2) segmentation of tumor core segmentation, and (3) segmentation of enhancing tumor core. Each network takes a fully convolutional network using dilated convolutions [31] and residual connections. In contrast, the work in [32] proposed a CNN model (Fig. 15.14) that consists of two parallel CNN pathways, each coping with a different receptive field on the input image, to incorporate both the

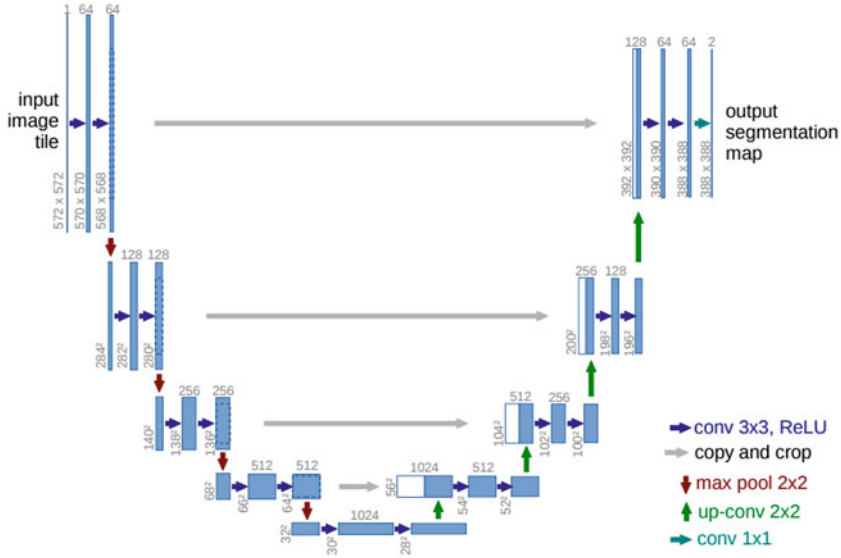


Fig. 15.10 Illustration of U-net model. (Image courtesy to [29])

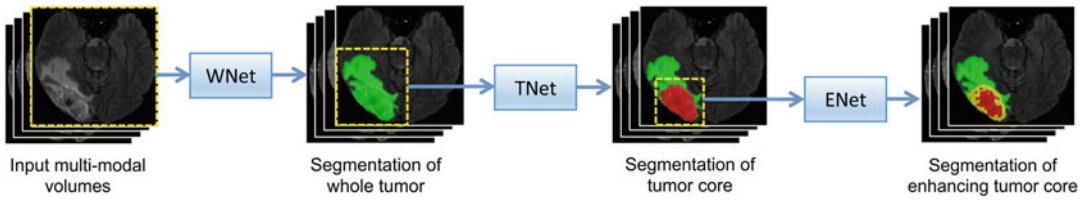


Fig. 15.11 Illustration of Cascade Net. (Image courtesy to [30])

local and the contextual information for brain tumor segmentation. Also, a multi-instance CNN (MICNN) model with multiple parallel sub-networks was designed in [33] for diagnosis and prognosis of brain diseases, and each sub-network was used to extract local patch-level representations of an input image. These patch-level features were further concatenated and fed into several FC layers for brain disease identification. In MICNN, each sub-network was corresponding to a disease-associated location (defined by anatomical landmarks) in the input brain MR image, and these sub-networks share the same architecture but with different network parameters, learn specific features from local patches. This method was further extended to be a multi-task learning model in [11] for joint classification and regression in brain MRI-based disease diagnosis.

15.3.6 CNN Applications to Brain Image Classification and Segmentation

CNNs have been applied to analyze brain images in a variety of tasks, such as mental disease classification [10, 33], neuroanatomy segmentation [34], lesion/tumor detection and segmentation [32, 35], brain image registration [36, 37], etc. In the following, some examples of CNNs used for brain image classification, segmentation, and brain network analysis are introduced, respectively.

15.3.7 Brain Image Classification

With the capability of learning highly non-linear and task-oriented features, CNNs have been applied to diagnosing brain diseases, such as Alzheimer's disease (AD). Many methods train CNN models to extract visual features based on predefined anatomical landmarks, such as hippocampus, for classification. These approaches demonstrated improved diagnostic accuracy over the conventional approaches using handcrafted features. However, in these methods, the localization of atrophy and the diagnosis of diseases are treated separately. Different from them, a hierarchically fully convolutional network (H-FCN) was proposed to jointly learn atrophy location and perform AD diagnosis [10].

The architecture of H-FCN is shown in Fig. 15.12. It consists of four components: location proposal sub-network, patch-level sub-network, region-level sub-network, and subject-level sub-network, aiming to learn features in a hierarchical way. Specifically, co-registered brain images were sent to the location proposal sub-network to generate image patches distributed over the whole brain. These image patches were fed into patch-level sub-networks to output patch-level features and patch-level classification scores. After that, spatially neighboring patches were then grouped into local regions, and their patch-level outputs (features concatenated with classification scores) were processed by the region-level sub-networks to produce regional features and regional classification scores. The outputs of the region-level sub-networks were eventually integrated by the subject-level sub-network to classify each subject. The proposed H-FCN demonstrated promising performance when evaluated on two datasets, i.e., Alzheimer's Disease Neuroimaging Initiative 1 (ADNI-1) and ADNI-2, that contain a large cohort of subjects. Visual examples of discriminative regions identified by H-FCN are provided in Fig. 15.13.

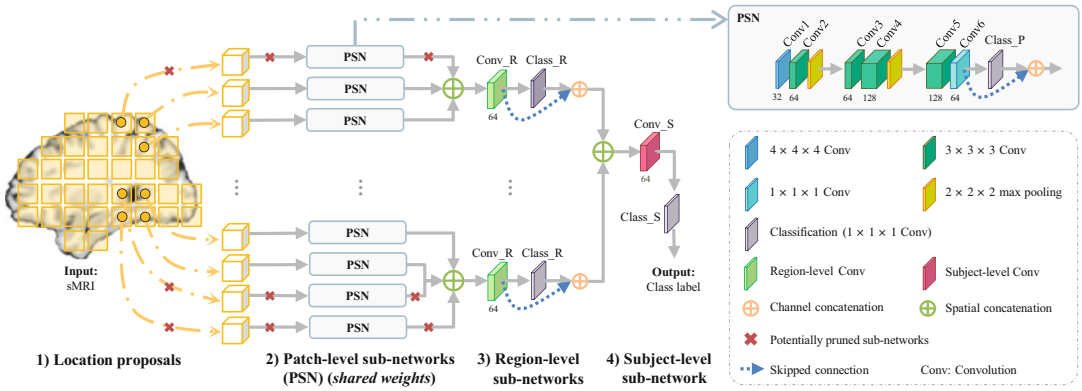


Fig. 15.12 Illustration of the hierarchically fully convolutional network (H-FCN) [10] for joint atrophy localization and disease diagnosis. (Image courtesy to [10])

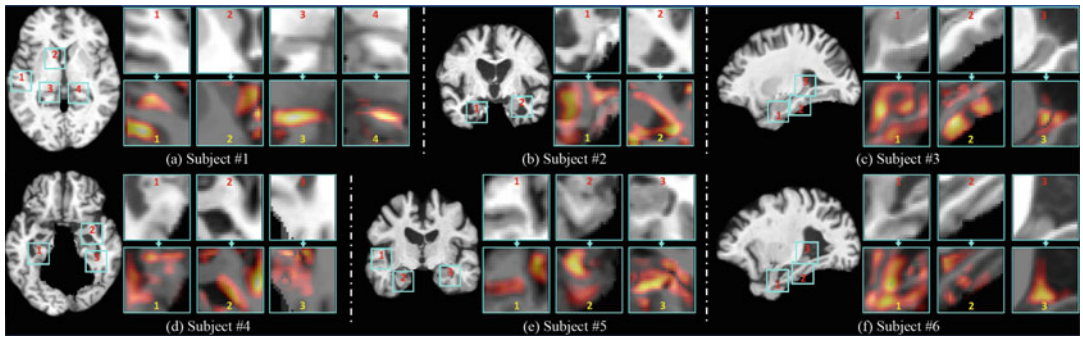


Fig. 15.13 Voxel-level AD heatmaps for discriminative patches automatically identified by H-FCN [10] in six different subjects. Warmer color in each heatmap indicates higher discriminative capacity. (Image courtesy to [10])

15.3.8 Brain Image Segmentation

As mentioned, segmentation of tissues or lesions in brain images could be conducted either by patch-wise classification (predicting the label of the patch centroid) or by FCN-based models that directly generate dense output for segmentation labels. DeepMedic [32] is a representative patch-wise classification method for brain tumor segmentation, with architecture shown in Fig. 15.14. This method proposed a two-pathway 3D CNN architecture to capture multi-scale features that incorporate both the local and the contextual information to improve brain tumor segmentation. As shown in Fig. 15.14, the inputs of the two pathways are 3D patches at the same image location but with different resolutions. The normal-resolution one focused on the local information, while the low-resolution one was extracted from a down-sampled version of the image, providing contextual information. Features extracted from the two parallel pathways were concatenated and processed by two fully connected layers to predict the label of the patch centroid. Additionally, on top of the CNNs' soft segmentation maps, fully connected conditional random field (CRF) model was used for final post-processing. DeepMedic model achieved top rankings in two brain lesion segmentation challenges ISLES2015 [38] and BRAT2015 [39]. Example segmentation results are given in Fig. 15.15.

Rather than classifying the centroid of each patch, another type of CNN-based segmentation approaches directly produce dense outputs corresponding to the segmentation labels. For example, in [6], a 3D U-net architecture was proposed for brain tumor segmentation (see Fig. 15.16). Just like U-net, this model consists of a context aggregation pathway to extract the abstract representation of the input large 3D blocks and a

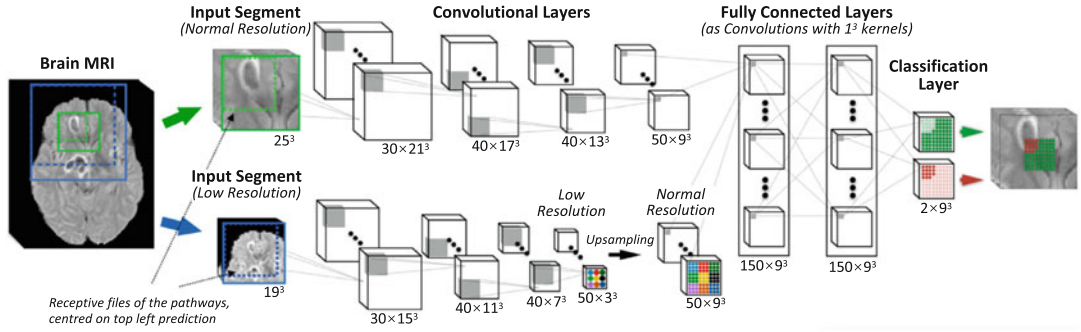


Fig. 15.14 Network architecture of DeepMedic. The inputs of the two parallel pathways are centered at the same image location but with different resolutions. (Image courtesy to [32])

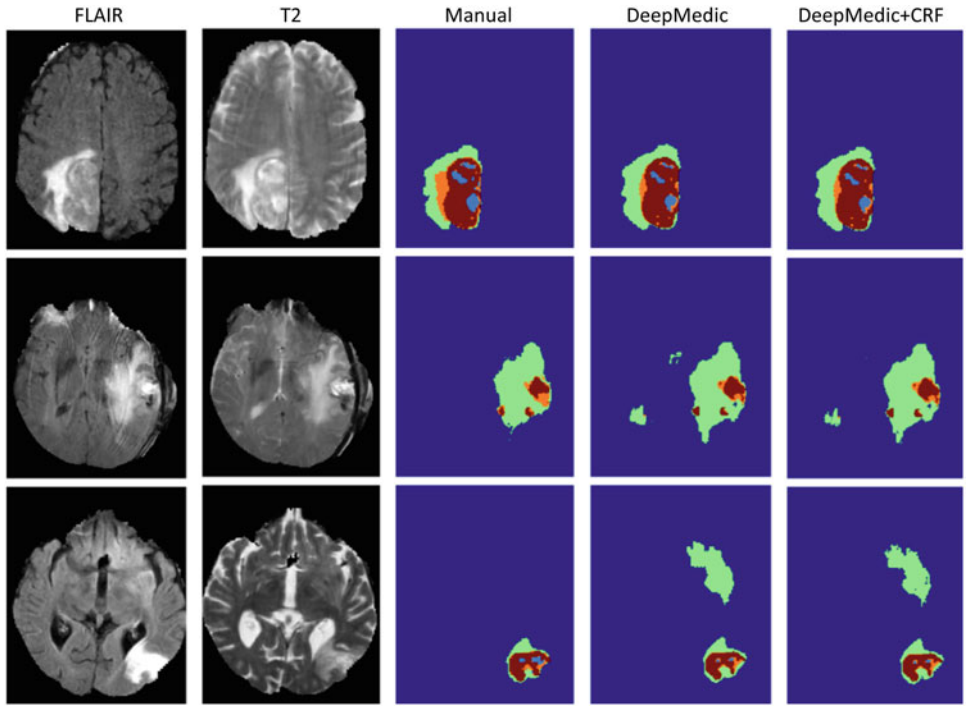


Fig. 15.15 Brain tumor segmentation examples (evaluated on the training set of BRATS2015 dataset) by DeepMedic [32]. Cyan indicates necrotic core; green indicates edema; orange indicates non-enhancing core; and red indicates enhancing core. (Image courtesy to [32])

localization pathway that localizes the structures of interest based on combined features from shallow layers. The context pathway is constructed by residual blocks, while the localization pathway is constructed by deconvolutional blocks for upsampling. Upon all convolution computation, Leaky ReLu is used as the activation function for non-linearity. The final output of the network is the element-wise summation of the segmentation results at different layers in the localization pathway. Meanwhile, the proposed model also benefits from the objective function of Dice loss as well as data augmentation. It achieves promising results on both BRATS2015 and BRATS2017 datasets. Figure 15.17 shows an example segmentation result achieved by [6]. Moreover, based on the segmentation mask, the work in [6] could further extract imaging-based features (such as

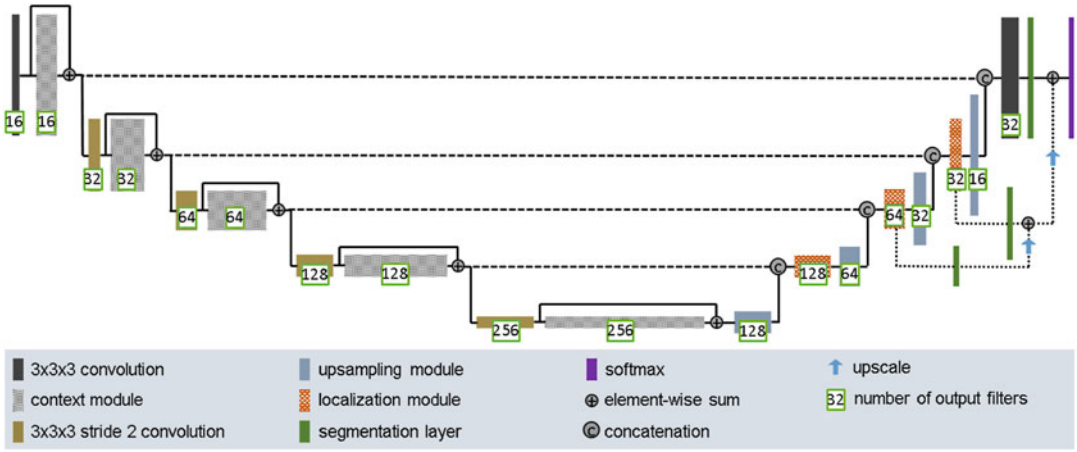


Fig. 15.16 Network architecture of [6]. (Image courtesy to [6])

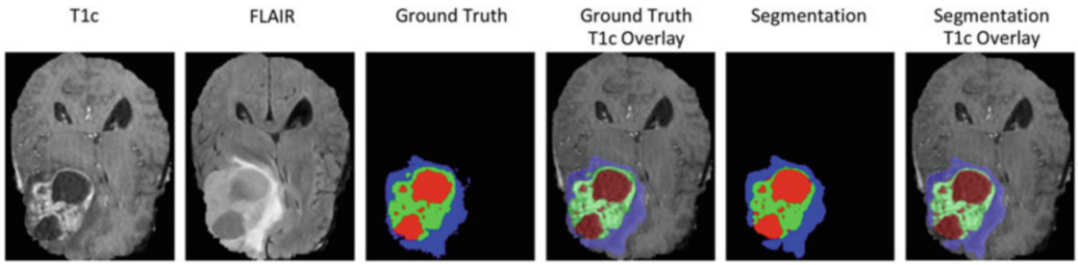


Fig. 15.17 A visual example of brain tumor segmentation. (Image courtesy to [6])

shape and first-order statistics) for survival prediction. Another state-of-the-art multi-scale FCN method was proposed in [8] for segmentation of perivascular spaces (PVSs) in brain images.

Furthermore, ensemble methods integrating different segmentation networks (e.g., 3D FCN [28], 3D U-net [40] and DeepMedic [32]) and trained with different loss functions and normalization schemes could further improve the segmentation performance over individual models, as demonstrated in [41].

15.4 Recurrent Neural Networks

15.4.1 Recurrent Neural Networks (RNNs): Basic Model

Traditional neural networks assume that all inputs (and outputs) are independent of each other. But for many tasks, this assumption may not hold. For example, the prediction of a word in a sentence usually depends on which words came before it. Recurrent neural networks (RNNs) address this issue. RNNs are called *recurrent* because they perform the same task for every element of a sequence (with the output being depended on the previous computations) and they have a "memory" which captures information about what has been calculated so far [42]. As shown in Fig. 15.18 (with a fold and an unfold structure), the basic RNN model consists of a sequence of non-linear units, and at least one connection of units forms a directed cycle. This chain-like nature reveals that RNNs are intimately related to sequences and lists. RNNs allow us to operate over sequences of vectors: sequences in the input, the output, or in the most general case both.

A typical RNN consists of three types of layers (see Fig. 15.18): (1) input layer, (2) recurrent hidden layers, and (3) output layer. The input

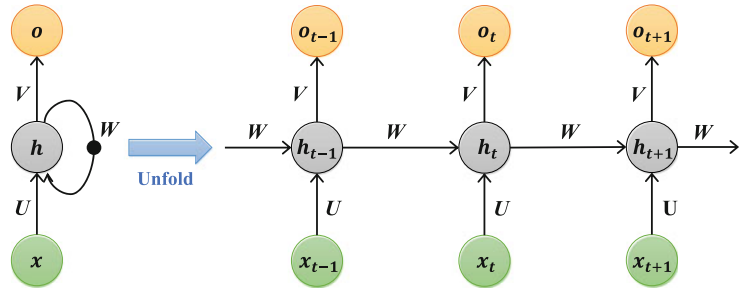


Fig. 15.18 Architecture of the basic recurrent neural network (RNN). Each x is an input example, $\boldsymbol{U}$ is the weight matrix between the input and hidden layers, $\boldsymbol{W}$ is the weight matrix between the previous and current hidden units in the hidden layer, and $\boldsymbol{V}$ is the weight matrix that connects the hidden and output layers. For simplicity, bias terms are not shown in this figure

layers is a sequence of vector through time step t, i.e., $\{\ldots, x_{t-1}, x_t, x_{t+1}, \ldots\}$, where x_t is the input vector. The input units are fully connected with the units in the hidden layers via a weight matrix $\boldsymbol{U}$.

The hidden layers are connected with each other through time via recurrent connections, and $\boldsymbol{W}$ is the weight matrix between the previous and current hidden units of the layer. With the recurrent hidden layers, RNNs can obtain the state space or "memory" as follows:

$$h_t = \sigma_h(\boldsymbol{W}x_t + \boldsymbol{U}h_{t-1} + b_h), \tag{15.2}$$

where h_t and h_{t-1} denote the hidden state at time steps t and $t-1$, respectively, b_h is the bias vector of the hidden units, and σ_h is the activation function used in the hidden layer. The weight matrices $\boldsymbol{W}$ and $\boldsymbol{U}$ are filters that determine how much importance to accord to both the present input and the past hidden states. The error they generate will return via backpropagation and be used to adjust their weights until error cannot go any lower. Because the feedback loop occurs at every time step in the series, each hidden state contains traces not only of the previous hidden state but also of all those that preceded h_{t-1} for as long as memory can persist.

The output units are connected with the hidden units via a weight matrix $\boldsymbol{V}$, and the output can be computed as follows:

$$o_t = \sigma_o(\boldsymbol{V}h_t + b_o), \tag{15.3}$$

where σ_o and b_o denote the activation function and bias term of the output layer. To learn network parameters $(\boldsymbol{W}, \boldsymbol{U}, \boldsymbol{V}, b_h, b_o)$, RNNs use the backpropagation algorithm for network training.

15.4.2 Long Short-Term Memory (LSTM) Model

Even though RNNs with recurrent connections are capable of understanding sequential dependencies, the backpropagation is usually time-consuming and falls victim to exploding and vanishing gradient during network training. Thus, in practice, RNNs don't seem to be able to learn "long-term dependencies" from data. Many methods have been designed to address this problem. Among these methods, long short-term memory (LSTM) network, introduced by Hocheriter et al. [43], is the most popular and efficient method, capable of learning "long-term dependencies."

As shown in Fig. 15.19a, a typical LSTM model consists of a memory cell C_t, an input gate i_t, an output gate o_t, and a forget gate f_t for the time step t. The memory cell transfers relevant information all the way down the sequence chain, and these gates control the activation singles from various sources to decide what information to add to or remove from the memory cell. The input gate i_t, the output gate o_t, and the forget gate f_t of an LSTM at time step t are defined as follows:

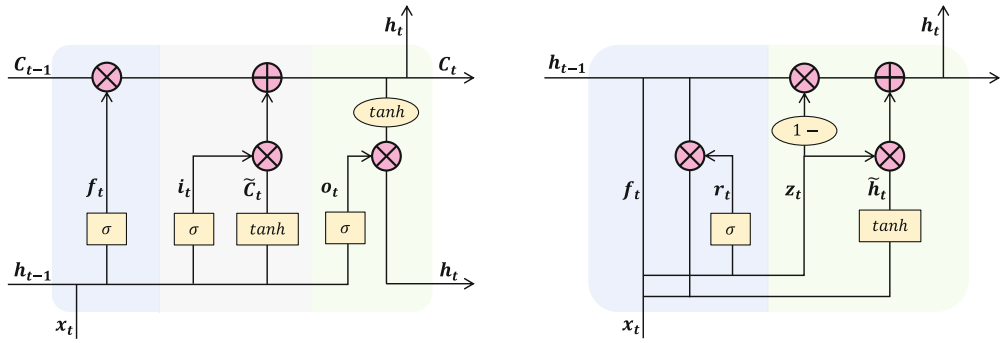


Fig. 15.19 Architectures of (**a**) long short-term memory (LSTM) network and (**b**) gated recurrent unit (GRU). Each $\boldsymbol{x}$ is an input example, $\boldsymbol{U}$ is the weight matrix between the input and hidden layers, $\boldsymbol{W}$ is the weight matrix between the previous and current hidden units in the hidden layer, and $\boldsymbol{V}$ is the weight matrix that connects the hidden and output layers. For simplicity, bias terms are not shown in this figure

$$\begin{aligned} i_t &= \sigma(\boldsymbol{W}_i x_t + \boldsymbol{U}_i h_{t-1} + b_i), \\ o_t &= \sigma(\boldsymbol{W}_o x_t + \boldsymbol{U}_o h_{t-1} + b_o), \\ f_t &= \sigma(\boldsymbol{W}_f x_t + \boldsymbol{U}_f h_{t-1} + b_f), \end{aligned} \tag{15.4}$$

where $\boldsymbol{W}_*$ and $\boldsymbol{U}_*$ are weight matrices from one state to the corresponding gate and b_* is the bias term. The memory cell C_t is updated by forgetting the existing memory and adding the new memory content $\tilde{C}_t$ as follows:

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t, \tag{15.5}$$

where the new memory content $\tilde{C}_t$ is defined as

$$\tilde{C}_t = \tanh(\boldsymbol{W}_c x_t + \boldsymbol{U}_c h_{t-1} + b_c). \tag{15.6}$$

As can be seen, the existing memory and the new memory are modulated by the forget data f_t and the input data i_t, respectively. The hidden state is finally computed as

$$h_t = o_t \tanh(C_t). \tag{15.7}$$

Unlike conventional RNN models, LSTM is able to decide whether to preserve the existing memory by the above-introduced gates. Theoretically, if LSTM learns an important feature from the input sequential data, it can keep this feature over a long time, thus capturing potential long-term dependencies.

A popular LSTM variant, called gated recurrent unit (GRU), is introduced by Cho et al. [44]. It combines the forget and input gates into a single "update gate." It also merges the cell state and hidden state, making each recurrent unit to adaptively capture dependencies of different time scales. The resulting model is simpler than standard LSTM models, with an illustration shown in Fig. 15.19b. The activation h_t in GRU at time step t is linearly modeled as

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t, \tag{15.8}$$

where the update gate z_t and the candidate activation $\tilde{h}_t$ are defined as

$$\begin{aligned} z_t &= \sigma(\boldsymbol{W}_z x_t + \boldsymbol{U}_z h_{t-1} + b_z), \\ \tilde{h}_t &= \tanh(\boldsymbol{W}_h x_t + \boldsymbol{U}_h (h_{t-1} \cdot r_t) + b_n), \end{aligned} \tag{15.9}$$

where the term $r_t = \sigma(\boldsymbol{W}_r x_t + \boldsymbol{U}_r h_{t-1} + b_r)$ denotes the reset gate. The update gate decides how much information to add and throw away, and the reset gate decides how much previous information to forget. Detailed comparisons between LSTM and GRU can be found in [45,46].

15.4.3 RNN Applications to Time Series Data Analysis

RNNs have shown their advantage in exploiting the temporal information in various tasks, such as disease diagnosis and object detection [12,47,48]. In the following, we introduce an example of

RNN models used for brain image classification based on time series data.

Brain functional connectivity (FC) extracted from resting-state fMRI (RS-fMRI) has become a popular approach for disease diagnosis, where discriminating subjects with mild cognitive impairment (MCI) from normal controls (HC) is still one of the most challenging problems. Dynamic functional connectivity (dFC) characterizes "chronnectome" diagnostic information for improving MCI classification, consisting of time-varying spatiotemporal dynamics. In [12], a fully connected bidirectional LSTM model (called Full-BiLSTM) is designed to learn the periodic brain status changes using both past and future information for each brief time segment (of blood oxygen-level-dependent signal of distributed brain regions) for MCI identification. The architecture of Full-BiLSTM is shown in Fig. 15.20. As can be seen from this figure, the outputs of every repeating cell are concatenated into a dense layer (i.e., "concatenation layer"). With this dense layer, one can abstract a common and time-invariant dynamic transition pattern from all the BiLSTM cells which may represent a constant "trait" information of each subject. The dense layer is followed by a softmax layer to get the final classification result. The proposed Full-BiLSTM method demonstrates good performance when evaluated on a rigorously built large-scale multi-site database (i.e., with 164 RS-fMRI scans from HCs and 330 scans from MCIs), achieving an accuracy of 73.6% in the task of MCI vs. HC classification.

15.5 Auto-encoder

The above-mentioned CNNs and RNNs can be treated as supervised deep learning models, since they require labeled data for network training. However, the acquisition of these ground truth labels needs massive human efforts from experts and considerable time cost for manual annotation. Therefore, many unsupervised deep feature learning models, such as stacked auto-encoder [7, 49], deep belief networks [50], and deep Boltzmann machine [51], have been proposed to mitigate this issue by using unlabeled training data. In addition, these unsupervised models learn imaging features without knowing the exact analysis tasks in advance and directly capture the visual clues

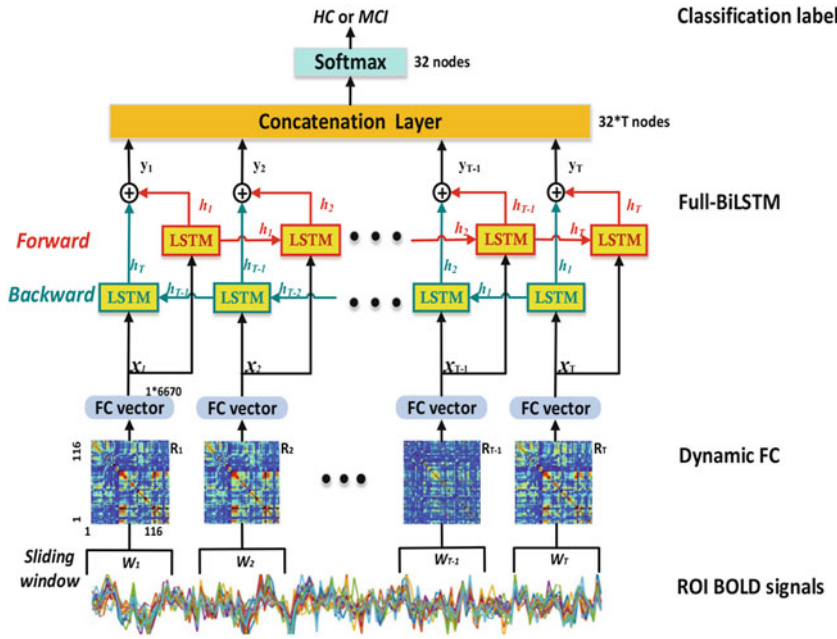


Fig. 15.20 Overview of the fully connected bidirectional LSTM model (Full-BiLSTM) for mild cognitive impairment (MCI) classification. (Image courtesy to [12])

that would be robust for different analysis tasks from brain images. This means that, feature representations extracted by unsupervised learning methods could have good capacity of generalization for the subsequent complex analysis. Among them, the deep variants of auto-encoder (AE) [52] have been widely applied to brain image analysis and achieved promising results.

A typical auto-encoder (AE) model contains an *encoder* to first transform the input into its low-dimensional latent representation space and a *decoder* to reconstruct the initial data from the representation by closing the distance between the input and the output. Once the models are well trained, the latent representations can be leveraged as the extracted features in the following tasks. Originally, the AE model consists of two layers for its encoder and decoder, respectively [52]. The first layer maps the input data $\mathbf{x}$ to its feature representation $\mathbf{h}$ by a specific function $\mathbf{h} = \sigma(\mathbf{W}_x\mathbf{x}+\mathbf{b}_x)$, where $\mathbf{W}_x$ and $\mathbf{b}_x$ are trainable parameters and σ is an activation function. The second layer decodes $\mathbf{h}$ to the output $\mathbf{y}$ by the mapping $\mathbf{y} = \sigma(\mathbf{W}_h\mathbf{h} + \mathbf{b}_h)$ with the parameters $\mathbf{W}_h$ and $\mathbf{b}_h$. The AE is trained to minimize the reconstruction loss to optimize parameters $\mathbf{W}_x$, $\mathbf{b}_x$, $\mathbf{W}_h$, and $\mathbf{b}_h$, as follows:

$$\{\mathbf{W}_x, \mathbf{b}_x, \mathbf{W}_h, \mathbf{b}_h\} = \arg \min_{\{\mathbf{W}_x, \mathbf{b}_x, \mathbf{W}_h, \mathbf{b}_h\}} \sum Distance(\mathbf{x}, \mathbf{y}).$$

Figure 15.21 illustrates the basic structure of an AE model. With this structure, AE models have at least two prominent advantages in feature learning [53]. First, they can be applied as feature extractors without any training labels, which fits the medical cases where only scarce labeled images are available in clinic and research. Second, the generated low-dimensional features largely reduce the complexity of the learning task and benefit the subsequent analysis.

However, due to the simple and shallow structure of the original AE models, they have limited power to capture the complicated non-linear patterns from the input data. To address this problem, deep stacked auto-encoders (SAEs) are constructed to improve the representational power. Specifically, SAEs organize AEs on top of each other by using the hidden features from one AE as the input of the successive AE, as shown in Fig. 15.22. Similar to training the original AE models, SAEs could be trained by directly optimizing the parameters of all layers at the same time. However, this training approach could easily lead these parameters to be stuck in local optimum, reducing the stability of SAEs. Therefore,

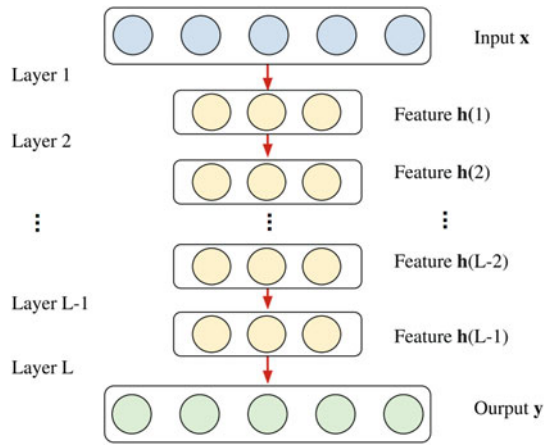
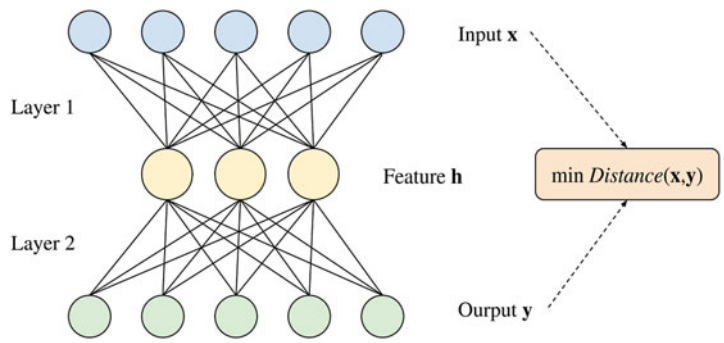


Fig. 15.22 SAE model

Fig. 15.21 Original AE model



a greedy layer-wise strategy is employed to train SAEs [54]. It gradually optimizes the layers in SAEs one by one: when training the l-th layer, the former pre-trained $l-1$ layers only need fine-tuning. In this way, SAEs can benefit from their deep architecture and derive the high-quality hierarchical hidden patterns from the input.

In addition, to better apply SAEs to unsupervised image feature extraction, the conventional MLP layers in SAEs are replaced by 2D or 3D convolutional layers as stacked convolutional auto-encoders (SCAEs) [55]. By building a symmetrical architecture of CNNs, SCAEs could learn the localized features of image structures. Furthermore, the pre-trained SCAEs can also be used to better initialize a CNN model of the same architecture before supervised learning.

15.5.1 AE Applications to Feature Learning in Brain Image Analysis

15.5.2 Brain Image Classification

As mentioned in Sect. 15.3.6, supervised learning- based deep models are usually leveraged to classify MRI and PET images of patients for early diagnosis and prognosis of Alzheimer's disease (AD) and its prodromal stage, i.e., mild cognitive impairment (MCI). However, training a high-quality deep model requires sufficient labeled brain images which are not always accessible. The work in [56] gives an SAE-based approach to mitigate this issue. Figure 15.23 illustrates a diagram of the proposed approach for brain image classification. Specifically, it first extracts the traditional handcrafted features as the low-level representations of input brain images. Then, a deep SAE is trained to reconstruct the low-level features in a greedy layer-wise manner without using disease labels. After this unsupervised learning, SAE can efficiently discover the hidden representation of these target-unrelated samples and be applied to initialize another deep model for supervised classification. This deep model is then fine-tuned by the labeled samples to extract their deep features of brain images. The extracted deep features and the low-level features are concatenated to select the best features for disease diagnosis. Finally, the selected features are passed to a support vector machine (SVM) model for AD/MCI diagnosis. Figure 15.24 compares the diagnosis results by separately using low-level features (LLF), SAE features (SAEF), and the fused LLF and SAEF. Since this work is among the first attempts to use

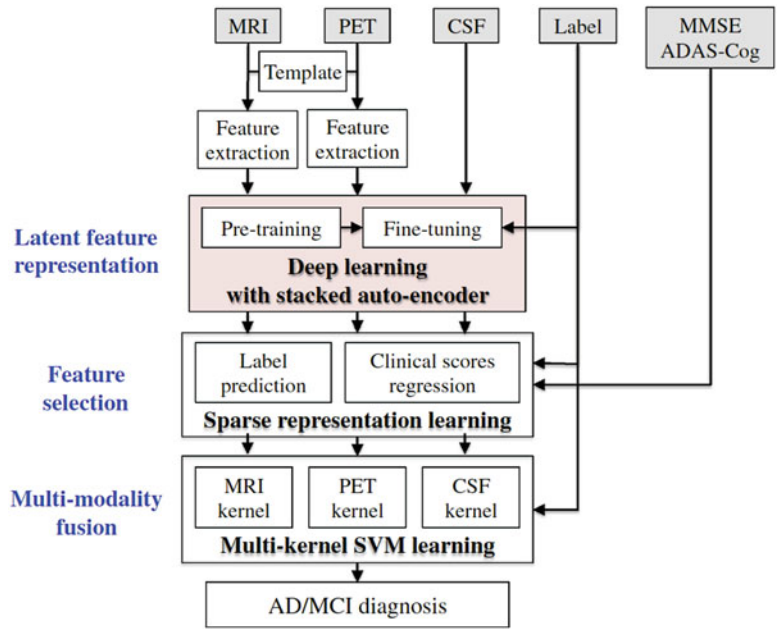


Fig. 15.23 An illustration for AD/MCI diagnosis in [56]. (Image courtesy to [56])

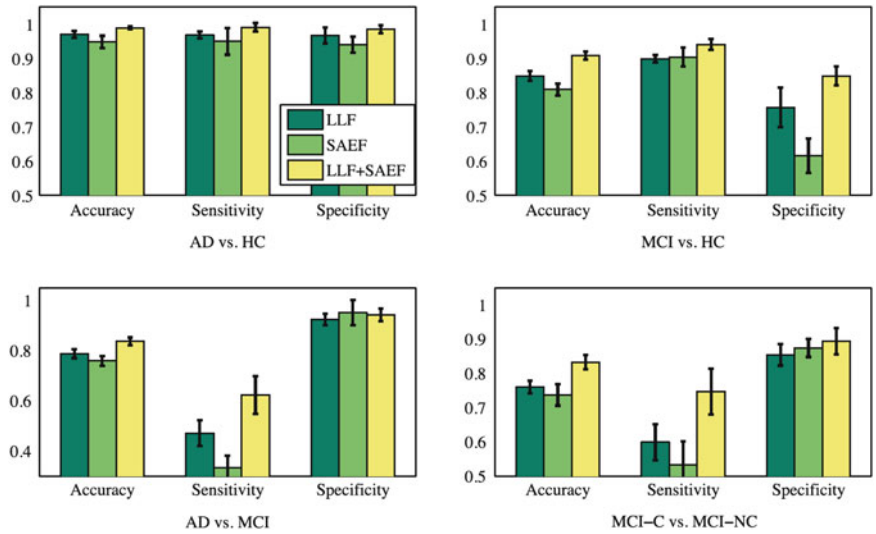


Fig. 15.24 Comparison of different feature extraction methods for AD, MCI, MCI converter (MCI-C), MCI non-converter (MCI-NC), and healthy normal control (HC) classification. (Image courtesy to [56])

deep learning for feature representation in brain disease diagnosis and prognosis in 2013, its SAE is simply constructed with MLP layers rather than convolutional layers. Thus, the extracted SAEF performs worse than LLF in this case. These reported results still validate that the use of SAEF could improve the diagnosis performance by only using traditional LLF.

15.5.3 Brain Image Registration

Deformable image registration, which aims to register medical images to a target template for anatomical alignment, is an important preprocess for various brain image analysis tasks. For more accurate registration, better features should be extracted to reflect the intrinsic local characteristics of both brain images and template. To satisfy this target, [9] proposed a SCAEs model to capture low-dimension anatomical latent representations of brain MR images via unsupervised feature learning. After training the SCAE model in a greedy layer-wise manner, the extracted deep imaging features from its encoder could be directly utilized as the input of existing image registration frameworks. Therefore, the hierarchical features can be learned without using manually annotated labels, and the constructed SCAEs could be directly applied to different types of medical images, such as 1.5-T MR and 7.0-T MR brain images. In [9], the SCAEs (for extracting deep features) are cooperated with the existing HAMMER registration framework, denoted as "H + DP". Another two registration approaches, i.e., image intensity-based Demons [57] and handcrafted feature-based HAMMER [58], are used to evaluate its effectiveness. A visual example of their registration results on 7.0-T MR brain images is shown in Fig. 15.25, where the manually labeled hippocampus on the template image and the deformed subject hippocampi achieved by different registration methods are indicated through red and blue contours, respectively. As observed, H + DP achieves most accurate registration results, while Demons almost fails to register these 7.0-T MR images. These results demonstrate the power of feature representations learned by the SCAEs model for brain image registration.

15.6 Generative Adversarial Networks

The recent occurrence of generative adversarial networks (GANs) has well tackled many challenging problems in brain image analysis.

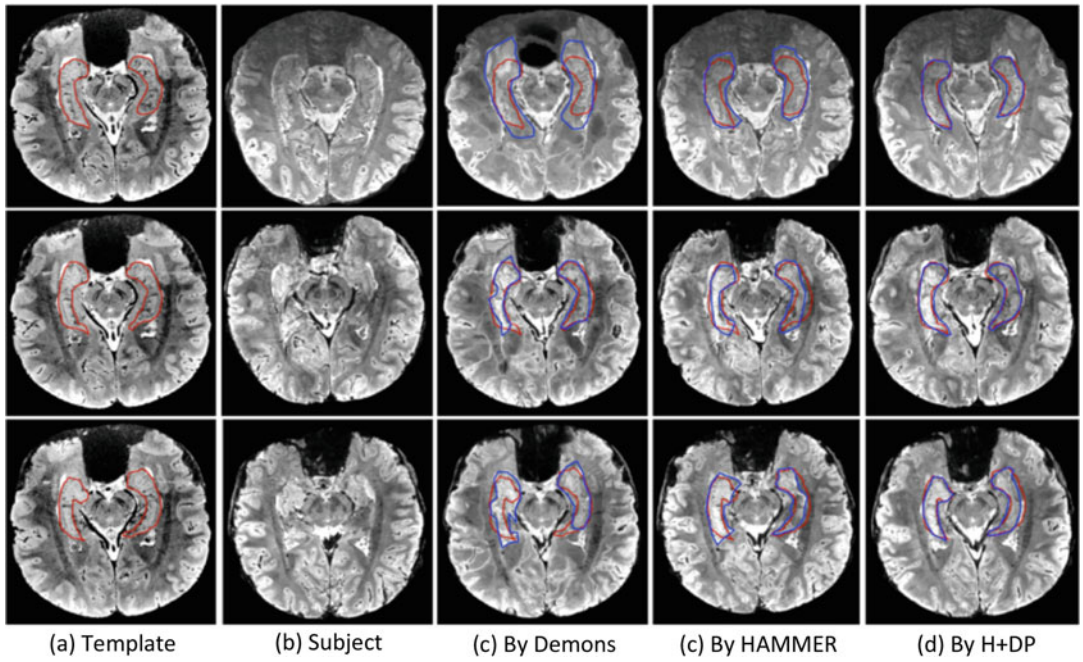


Fig. 15.25 Typical registration results on 7.0-T MR brain images by Demons, HAMMER, and HAMMER with SCAEs features (H + DP). Three rows represent three different slices in the template, subject, and registered subjects. (**a**) Template. (**b**) Subject. (**c**) By Demons. (**d**) By HAMMER. (**e**) By H + DP. (Image courtesy to [9])

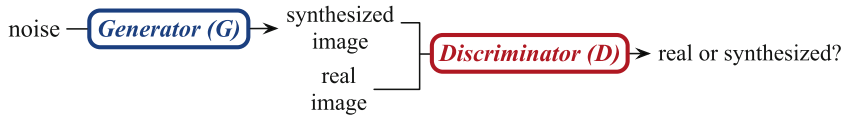


Fig. 15.26 Original GAN model

GANs have been used for a wide variety of applications such as brain lesion detection and segmentation, brain image registration, brain image reconstruction and super-resolution, cross-modality brain image synthesis, etc. GANs were originally proposed as *generative* models for unsupervised learning, which focus on learning the distribution of given data and therefore can generate new samples from the learned distribution. That is, given the input data $\mathbf{x}$, GANs focus on learning the probability $P(\mathbf{x})$. This is different from the *discriminative* models, such as CNNs used for classification, which focus on classifying the input data, i.e., learning $P(\mathbf{y}|\mathbf{x})$, where $\mathbf{y}$ indicates class labels. The key idea of GANs is *adversarial training*. It refers to the simultaneous training of two agents in a GAN model, i.e., a *generator* and a *discriminator*, with the goal of one beating the other. Specifically, the generator tries to produce fake samples that resemble the real ones to fool the discriminator, while the discriminator struggles to tell the fake samples from the real ones. Through the competition, both the generator and the discriminator could improve their models for better performance.

15.6.1 Principle of GAN

In 2014, the original GANs were first proposed for the generic image synthesis tasks [59]. Different from the common CNN-based deep learning models, a GAN model consists of two agents, i.e., a generator G and a discriminator D, which are trained by adversarial learning, as illustrated in Fig. 15.26. Given a training set of real samples

$\mathbf{X}$ with the distribution P_{data}, the goal of the generator is to learn an embedding function $G(\cdot)$ that transforms the random inputs $\mathbf{z}$ (drawn from the distribution P_{noise}) to the output synthetic images $G(\mathbf{z})$ whose distribution matches the data distribution P_{data}. Meanwhile, the discriminator D is trained to learn an embedding function $D(\cdot)$ to maximize the probability of the correct label assignment to discriminate the real and the fake samples. The generator and the discriminator play a two-player minmax game with the following objective function

$$\arg\min_G \max_D V(G, D) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log(D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\text{noise}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}))], \tag{15.10}$$

where the symbol $\mathbb{E}$ denotes mathematical expectation.

In addition, prior information could also be incorporated via conditional GANs (cGANs) [60]. With the condition variable $\mathbf{c}$, the objective of cGANs becomes

$$\begin{aligned}\arg\min_G \max_D V(G, D) =& \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log(D(\mathbf{x}|\mathbf{c})] \\ &+ \mathbb{E}_{\mathbf{z}\sim p_{\text{noise}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\mathbf{c}))].\end{aligned} \tag{15.11}$$

When the prior information is an input image $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$, cGANs can be trained for paired image-to-image translation. That is, generating the corresponding image $\mathbf{y} \sim p_{\text{data}}(\mathbf{y})$ with the specific control from $\mathbf{x}$. For example, when $\mathbf{x}$ is an input brain image and $\mathbf{y}$ is the corresponding segmentation map of $\mathbf{x}$, the cGAN model can be trained for segmentation tasks.

15.6.2 GAN Variants

15.6.3 Pix2Pix GANs

Many GAN models used in medical image analysis [14, 61–63] follow the image translation framework Pix2Pix proposed in [64] and achieve promising results. As a cGAN model, given a source image $\mathbf{x}$, the generator G in Pix2Pix produces an image $G(\mathbf{x})$ that resembles the real target image $\mathbf{y}$. At the same time, the discriminator D is trained to differentiate between the fake image pair $(\mathbf{x}, G(\mathbf{x}))$ and the real image pair $(\mathbf{x}, \mathbf{y})$. The training loss of the generator G is as follows:

$$\begin{aligned}\mathcal{L}^G_{cGAN} =& \mathbb{E}_{\mathbf{x}\sim P_{\text{data}}(\mathbf{x})}[\log(1 - D(\mathbf{x}, G(\mathbf{x})))] \\ &+ \lambda_{l1}\mathbb{E}_{\mathbf{x},\mathbf{y}\sim P_{\text{data}}(\mathbf{x},\mathbf{y})}[\|y - G(\mathbf{x})\|_1],\end{aligned} \tag{15.12}$$

where the first term in Eq. 15.12 is the common adversarial loss of a generator as in the original GANs. In addition, Pix2Pix also enforces the pixel-wise similarity between the generated image and the real image, as described in the second term. Here $\|\cdot\|_1$ indicates the l_1 norm, i.e., the average absolute pixel-wise difference between $G(\mathbf{x})$ and $\mathbf{y}$. The hyper-parameter λ_{l1} is user-defined to balance these two terms.

The loss function of the discriminator D is defined as

$$\begin{aligned}\mathcal{L}^D_{cGAN} =& -\mathbb{E}_{\mathbf{x},\mathbf{y}\sim P_{\text{data}}(\mathbf{x},y)}[\log D(\mathbf{x}, \mathbf{y})] \\ &- \mathbb{E}_{\mathbf{x}\sim P_{\text{data}}(\mathbf{x})}[\log(1 - D(\mathbf{x}, G(\mathbf{x})))].\end{aligned} \tag{15.13}$$

By minimizing Eq. 15.13, the discriminator D is trained to assign the correct labels (0 or 1) to the fake or the real image pairs.

As image generation and image discrimination are trained together, the final loss function that integrates the objectives of G and D becomes

$$\mathcal{L}_{cGAN} = \mathcal{L}^{G}_{cGAN} + \mathcal{L}^{D}_{cGAN}. \tag{15.14}$$

In Pix2Pix, CNN architectures are used for both the generator and the discriminator to extract powerful deep features. Especially, the generator uses a U-net-like architecture to utilize the hierarchy of contextual information for image generation. The discriminator follows the common CNNs used for classification.

15.6.4 CycleGAN

The Pix2Pix model transforms images between two domains with one-to-one correspondence. That is, the training data consists of paired images $\{\mathbf{x}, \mathbf{y}\}$, where $\mathbf{x}$ and $\mathbf{y}$ are the corresponding samples in the two domains $\mathbf{X}$ and $\mathbf{Y}$, respectively. This requirement could be relaxed by cycle GAN (CycleGAN) [65] that only needs *unpaired* training data to learn the mapping between images in two domains, such as computerized tomography (CT) and MR images. The needs of paired images are eliminated by learning two mappings $\mathbf{X} \rightarrow \mathbf{Y}$ and $\mathbf{Y} \rightarrow \mathbf{X}$ simultaneously and enforcing a cycle consistency loss during training. CycleGAN consists of two generators – G to learn the mapping $\mathbf{X} \rightarrow \mathbf{Y}$ and F to learn the mapping $\mathbf{Y} \rightarrow \mathbf{X}$ – as well as two discriminators, D_X to differentiate $\hat{\mathbf{x}} = F(G(\mathbf{x}))$ from $\mathbf{x}$ and D_Y to differentiate $\hat{\mathbf{y}} = G(F(\mathbf{y}))$ from $\mathbf{y}$. The basic idea is illustrated in Fig. 15.27. The objective function of CycleGAN is as follows:

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}^{G, D_Y, X, Y}_{GAN} + \mathcal{L}^{G, D_X, Y, X}_{GAN} + \lambda \mathcal{L}^{G, F}_{\text{cycle}}, \tag{15.15}$$

where the first and the second terms are the adversarial loss of GANs and the third term is the cycle consistency loss, which is defined as

$$\mathcal{L}^{G, F}_{\text{cycle}} = \mathbb{E}_{\mathbf{x} \sim P_{\text{data}}(\mathbf{x})}[\|F(G(\mathbf{x})) - \mathbf{x}\|_1] + \mathbb{E}_{\mathbf{y} \sim P_{\text{data}}(\mathbf{y})}[\|G(F(\mathbf{y})) - \mathbf{y}\|_1]. \tag{15.16}$$

As can be seen, the forward cycle $\mathbf{x} \rightarrow G(\mathbf{x}) \rightarrow F(G(\mathbf{x})) \approx \mathbf{x}$ should be able to bring $\mathbf{x}$ back. Similarly, the backward cycle $\mathbf{y} \rightarrow F(\mathbf{y}) \rightarrow G(F(\mathbf{y})) \approx \mathbf{y}$ should be able to bring $\mathbf{y}$ back. In this way, the samples $\mathbf{x}$ and $\mathbf{y}$ in two domains do not need to have one-to-one correspondence in CycleGAN.

15.6.5 GAN Applications to Brain Image Analysis

15.6.6 Brain Image Synthesis

GANs have been widely used for brain image synthesis either within the same imaging modality or across different imaging modalities. For example, a 3D cGAN model was proposed in [13] to synthesize full-dose brain positron emission tomography (PET) images from the low dose ones. PET imaging reveals the metabolism processes of human and is widely exploited in clinics and research. During PET scanning, radioactive tracers are injected into the patient's body for imaging. Usually, a full dose of radioactive tracer is needed to generate PET images of diagnostic quality. The exposure to radiation inevitably brings healthy concerns, especially for those patients who have to undertake multiple scanning during their treatment. On the other hand, lowering the dosage of tracers could significantly reduce the quality of PET images, as shown in Fig. 15.28. Therefore, the work in [13] proposed to fill the gap between

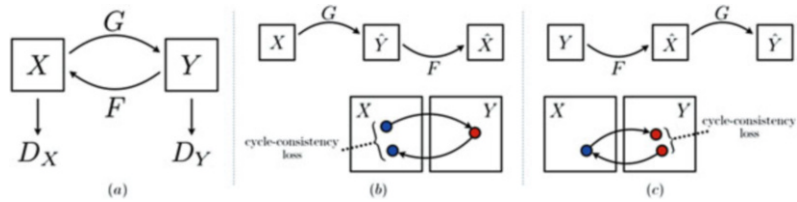


Fig. 15.27 Illustration of CycleGAN for unpaired image-to-image translation. (Image courtesy to [65])

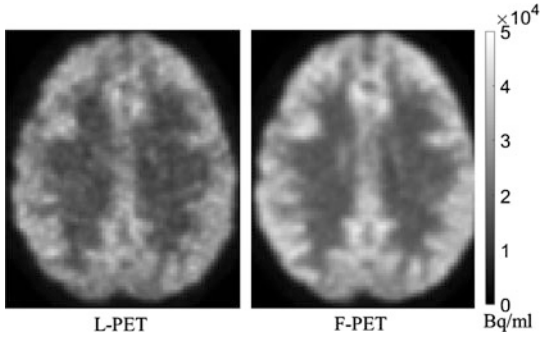


Fig. 15.28 Comparison between a low-dose PET (L-PET) image and its corresponding full-dose PET (F-PET). (Image courtesy to [13])

the low-dose PET images and the full-dose ones by using a 3D conditional GAN (cGAN) model. The overview of the proposed method is given in Fig. 15.29. It follows the Pix2Pix framework, where the generator is a U-net-like structure and the discriminator is a CNN-based classifier to differentiate the real and the fake image pairs. Different from Pix2Pix and many other cGAN-based medical image synthesis models [61, 63, 66, 67] that use 2D slices of a PET image as the input, the cGAN model [13] is completely 3D. It takes 3D patches of PET images as input and processes them with 3D up- and down-convolutions. In this way, it mitigates the problem of discontinuous estimation across slices, which is however often observed in 2D-based synthesis models. A visual comparison of the results using 2D and 3D cGAN models, respectively, is given in Fig. 15.30. As shown, the full-dose PET images synthesized by 3D cGAN show high image quality in all three views. However, three 2D cGAN models only produce good results in their corresponding trained views as indicated in the red circles, but not along with the other two directions since they lose the 3D structural information during the synthesis.

GANs have also been intensively studied for cross-modality brain image synthesis. For example, when setting different scanning parameters, MRI can generate multi-modality images (e.g., T1-weighted, T2-weighted, and FLAIR) to reflect soft tissues with different contrast, providing complementary information for disease diagnosis [68] and treatment planning [69]. Cross-modality MR image synthesis is therefore often needed to deal with the potential modality loss in clinics so that the diagnosis could benefit from the enriched information in the multiple imaging modalities [70, 71]. Such tasks could be more challenging than the synthesis within the same imaging modality. Meanwhile, generating the images of new modality is often not the end: these images are expected to well preserve the pathology that is critical for the subsequent analysis. For example, when the brain images contain tumors, the boundaries of tumors are expected to be well depicted in the generated images. To develop pathology-centered synthesis, different regularization terms have been proposed and embedded into the learning process of GANs models. For example, the work in [14] proposed edge-aware conditional GAN models (Ea-GANs) to enforce the preservation of edges for cross-modality MR image synthesis to assist brain tumor segmentation. In addition to enforcing the pixel-wise intensity similarity as in Pix2Pix, Ea-GANs also require that the edge maps extracted from the synthesized images should resemble those from the real images. These edge maps are computed by the commonly used Sobel filters. As shown in Fig. 15.31, the work in [14] proposed two frameworks to incorporate the edge maps, i.e., a generator-induced Ea-GAN (gEa-GAN) and a discriminator-induced Ea-GAN (dEa-GAN). In gEa-GAN, the edge maps are incorporated into the generator side only, while in dEa-GAN, the edge maps are also introduced to

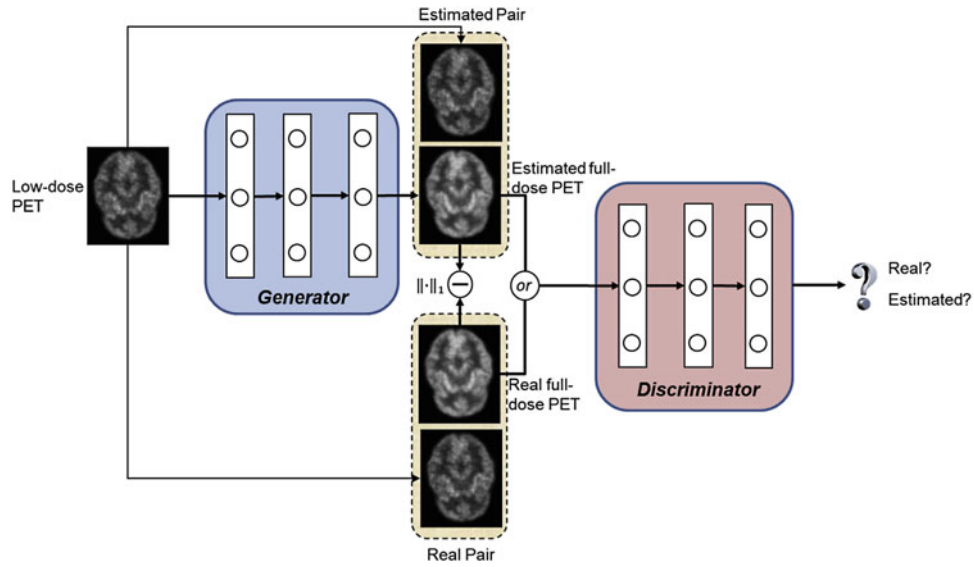


Fig. 15.29 Framework of training a 3D conditional GAN (cGAN) to estimate the full-dose PET image from low-dose counterpart. (Image courtesy to [62])

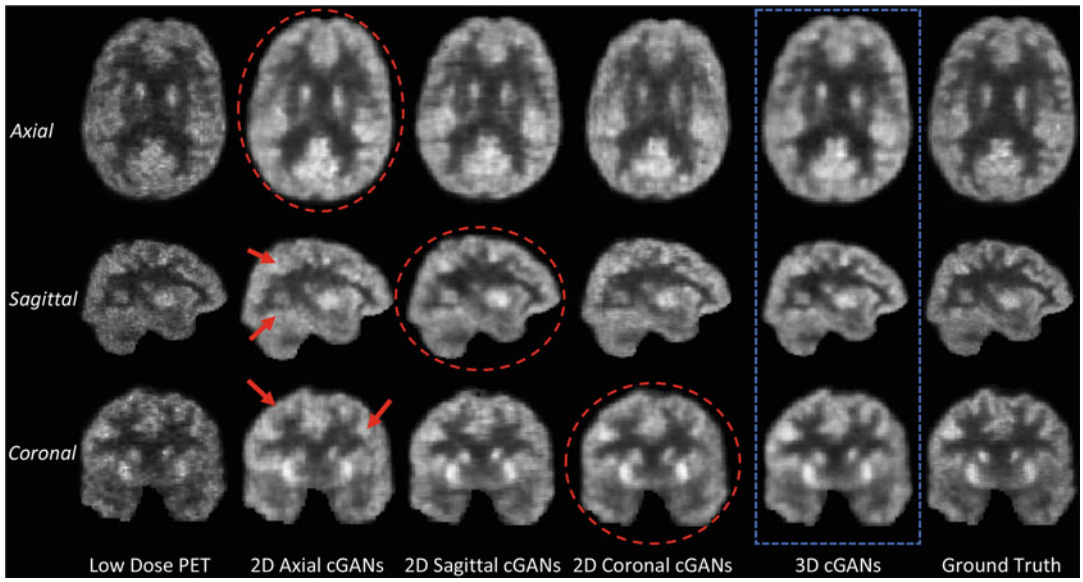


Fig. 15.30 Visual comparison between the results estimated by 2D cGAN and the 3D cGAN in [62]. These 2D cGANs are separately trained with the 2D slices from the corresponding axial, coronal, and sagittal views. (Image courtesy to [62])

the discriminator side, so that they participate the adversarial training to help improve the synthesis quality. As shown in Table 15.1, in a synthesis task from T1-weighted MRI to FLAIR MRI, Ea-GANs using the edge information outperformed the 3D cGAN in both the whole image and the tumor areas. This is consistent with the visual comparison in Fig. 15.32.

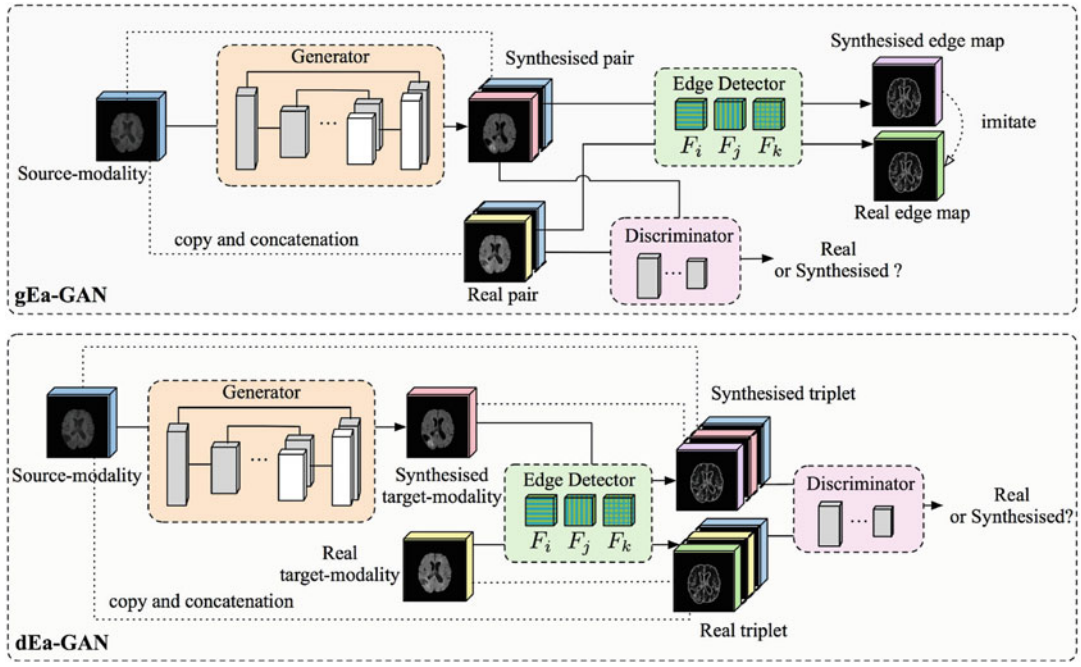


Fig. 15.31 Frameworks of Ea-GANs proposed in [14]. (Image courtesy to [14]). In gEa-GAN, the generator enforces the similarity between the real and the generated images, as well as the similarity of their corresponding edge maps. In dEa-GAN, the edge maps are also used to train the discriminator to classify the triplets comprising of the source-modality image, the real/generated target-modality image, and the edge map of the corresponding real/generated target-modality image

Table 15.1 Method comparison: synthesizing FLAIR-like images from T1-weighted images on the BRATS2015 dataset (mean)

Methods	Whole image			Tumor part		
	PSNR	NMSE	SSIM	PSNR	NMSE	SSIM
3D cGAN [72]	29.26	0.119	0.958	15.95	0.098	0.681
gEa-GAN [14]	29.55	0.115	0.960	16.37	0.090	0.697
dEa-GAN [14]	**30.11**	**0.105**	**0.963**	**16.90**	**0.084**	**0.705**

15.6.7 Brain Image Augmentation

Brain disease diagnosis benefits from multi-modality imaging data that provides complementary information. For example, the structural imaging MRI and the functional imaging PET have been widely used for the diagnosis of Alzheimer's disease (AD). However, the missing-modality problem often occurs in clinic, for example, patients taking MRI scanning may reject to also take PET scanning due to the concerns about the cost. Such a problem also exists in the widely used Alzheimer's Disease Neuroimaging Initiative (ADNI) database, which limits the number of subjects available for the research. A common practice to deal with the missing-modality problem is to impute the images of the missing modality. For example, in [15], a 3D CycleGAN model was proposed to impute the missing PET images by learning the bidirectional mapping between MRI and PET. Based on complete (after imputation) MRI-PET pairs, a multi-modal multi-instance learning method was further proposed for AD diagnosis. The architecture of the 3D CycleGAN is shown in Fig. 15.33. It consists of two generators to learn

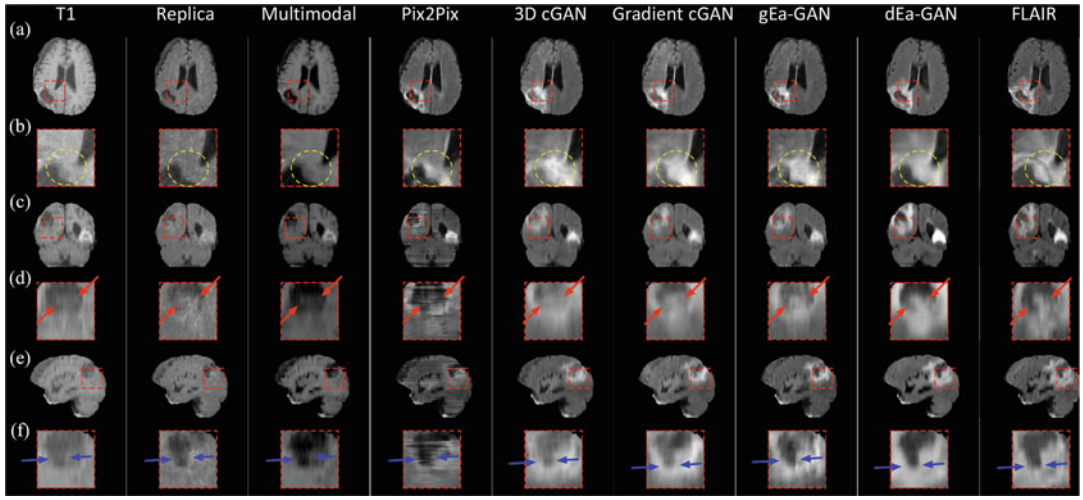


Fig. 15.32 FLAIR image synthesis from T1-weighted MR images: visual comparison between the results estimated by methods proposed in [14] and several competing approaches. (**a**) Axial slices, (**b**) zoomed parts of axial slices, (**c**) coronal slices, (**d**) zoomed parts of coronal slices, (**e**) sagittal slices, and (**f**) zoomed parts of sagittal slices

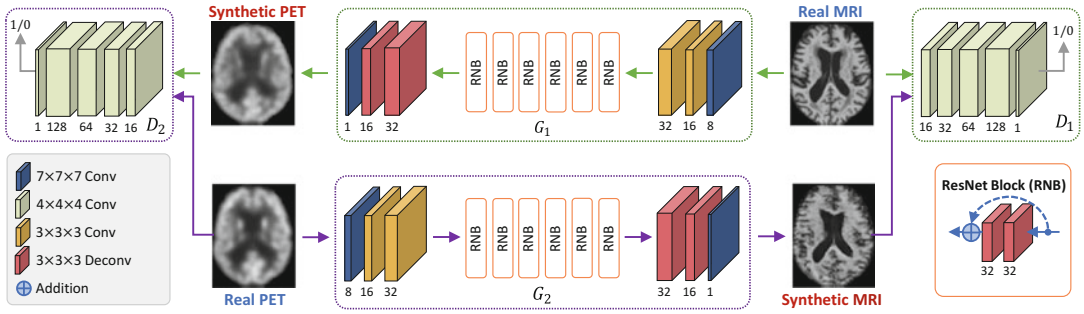


Fig. 15.33 The architecture of the 3D CycleGAN proposed in [15]. (Image courtesy to [15])

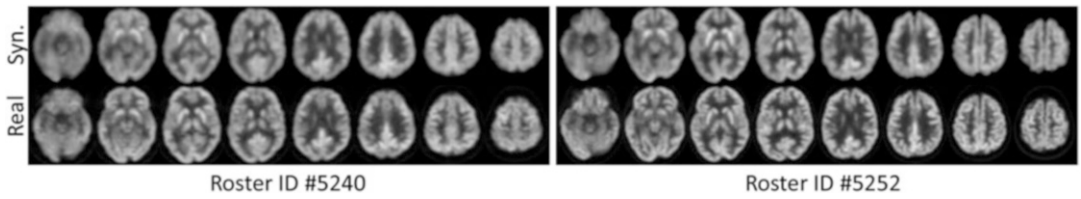


Fig. 15.34 Visual comparison on two subjects: synthetic PET images generated by 3D CycleGAN proposed in [15] (top) *vs.* the corresponding real PET images (bottom). (Image courtesy to [15])

the mappings from MRI to PET and the mapping from PET to MRI, respectively. Each generator consists of three parts: encoding, transferring, and decoding components. The encoding part consists of three convolutional layers to extract features in the source domain (e.g., MRI). The transferring part is constructed by six residual network blocks to transfer the features from the source domain to the target domain (e.g., PET). The decoding part contains two deconvolutional layers and one convolutional layer to generate images in the target domain. Examples of the synthesized PET images are shown in Fig. 15.34. By integrating image synthesis and disease

diagnosis into a unified framework, this work was further extended to a more advanced model to generate disease-specific PET/MRI scans [73], providing an exciting research direction for synthesizing task-oriented neuroimages through GANs.

15.7 Discussion

Even though deep learning has achieved record-breaking performance in brain image analysis, there are still several potential limitations to consider.

First, deep learning models usually have a very high computational cost for network training because of the high dimensionality of input brain images as well as the huge number of to-be-optimized network parameters. Using GPUs with higher computation power and designing models in a parallel way can partly address this issue. It is also interesting to perform dimension reduction for input brain images, by defining regions of interests in the brain (empirically or in a data-driven manner) to reduce the negative influence of uninformative regions [8].

Second, they generally require a large number of training images for generating reliable models. The latest success of GAN models in the synthesis of neuroimages has brought new solutions to the augmentation of training samples. Transfer learning [33, 74], which can enable knowledge sharing between related tasks/domains, is also an interesting solution that reduces the need for a large number of training samples required for network training.

In addition, deep learning models have often been described as "black boxes," without explicitly articulating themselves in a certain way. In many neuroimaging-based applications, it is often not enough to have a good prediction system. To understand what intermediate layers of convolutional networks are responding to, several strategies have been proposed, such as deconvolution networks [75], deep Taylor composition backpropagation [76], and Bayesian deep networks [77]. It is desired to develop new strategies to further understand deep learning methods in brain image analysis, which could accelerate the acceptance of deep learning applications among clinicians and patients.

15.8 Conclusion

In this chapter, deep learning models and their applications to brain image analysis are introduced. Specifically, four typical deep learning models (i.e., CNN, RNN, AE, and GAN) and their applications (i.e., brain image segmentation, brain image registration, neuroimaging-based brain disease diagnosis, and brain image synthesis) are introduced. Limitations of current deep learning models and possible future research directions are also discussed. It is expected that deep learning will have a great impact on brain image analysis.

Homework

1. In practice, we usually have only limited number of brain images to train deep models for analysis. Please explain the problem and list at least two strategies to deal with this situation.
2. Consider the following MLP model with two hidden layers and the loss function shown in Table 15.2. Please calculate $\frac{\partial h_{1,i}}{\partial w_{1,2}}$ and $\frac{\partial J}{\partial w_{1,2}}$.
3. What are the benefits to use CNN to analyze brain images, compared with MLP?
4. What are the key difference and advantages of the fully convolutional networks (FCNs) over convolutional neural networks (CNNs) in brain image segmentation (pixel-level prediction)?
5. Please describe the network structure of U-net. What are the benefits to use skip connections in U-net?

Table 15.2 An MLP model with two hidden layers

Input	$x_{1,i}$, $x_{2,i}$, where $i = 1, \cdots N$ and N is the number of samples
Layer 1	$h_{1,i} = \max(w_{1,1}x_{1,i} + w_{1,2}x_{2,i}, 0)$, $h_{2,i} = \max(w_{2,1}x_{1,i} + w_{2,2}x_{2,i}, 0)$
Layer 2	$p_i = w_5 h_{1,i} + w_6 h_{2,i}$
Loss	$J(p, \mathbf{w}) = \sum_i (p_i - y_i)^2$

6. What's the difference between RNNs and CNNs? What is the advantage of LSTM over RNN?
7. The auto-encoder (AE) model can extract imaging features in an unsupervised manner. What is the principle of AE?
8. What's the purpose of using the generator and discriminator in GANs?
9. What's the advantage of CycleGAN over GAN?
10. Please list possible deep learning models that could be used for the following brain image analysis tasks: (a) brain disease diagnosis, (b) brain lesion segmentation, (c) brain network analysis based on fMRI images, (d) brain image transferring across modality, (e) brain disease diagnosis without labeled training samples, (f) brain image generation without paired training samples.

References

1. Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1798–1828 (2013)
2. Y. LeCun, Y. Bengio, G. Hinton, Deep learning. Nature **521**(7553), 436 (2015)
3. J. Schmidhuber, Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015)
4. D. Shen, G. Wu, H.I. Suk, Deep learning in medical image analysis. Ann. Rev. Biomed. Eng. **19**, 221–248 (2017)
5. J.G. Lee, S. Jun, Y.W. Cho, H. Lee, G.B. Kim, J.B. Seo, N. Kim, Deep learning in medical imaging: general overview. Korean J. Radiol. **18**(4), 570–584 (2017)
6. F. Isensee, P. Kickingereder, W. Wick, M. Bendszus, K.H. Maier-Hein, Brain tumor segmentation and radiomics survival prediction: contribution to the BRATS 2017 challenge. In: Brainlesion: glioma, multiple sclerosis, stroke and traumatic brain injuries. BrainLes 2017 (2018)
7. Y. Zhu, L. Wang, M. Liu, C. Qian, A. Yousuf, A. Oto, D. Shen, MRI-based prostate cancer detection with high-level representation and hierarchical classification. Med. Phys. **44**(3), 1028–1039 (2017)
8. C. Lian, J. Zhang, M. Liu, X. Zong, S.C. Hung, W. Lin, D. Shen, Multi-channel multi-scale fully convolutional network for 3D perivascular spaces segmentation in 7T MR images. Med. Image Anal. **46**, 106–117 (2018)
9. G. Wu, M. Kim, Q. Wang, B.C. Munsell, D. Shen, Scalable high-performance image registration framework by unsupervised deep feature representations learning. IEEE Trans. Biomed. Eng. **63**(7), 1505–1516 (2015)
10. C. Lian, M. Liu, J. Zhang, D. Shen, Hierarchical fully convolutional network for joint atrophy localization and Alzheimer's disease diagnosis using structural MRI. IEEE Trans. Pattern Anal. Mach. Intell. **42**(4), 880–893 (2020)
11. M. Liu, J. Zhang, E. Adeli, D. Shen, Joint classification and regression via deep multi-task multi-channel learning for Alzheimer's disease diagnosis. IEEE Trans. Biomed. Eng. **66**(5), 1195–1206 (2018)
12. W. Yan, H. Zhang, J. Sui, D. Shen, Deep chronnectome learning via full bidirectional long short-term memory networks for MCI diagnosis. In: International conference on medical image computing and computer-assisted intervention (Springer, 2018), pp. 249–257
13. Y. Wang, L. Zhou, B. Yu, L. Wang, C. Zu, D.S. Lalush, W. Lin, X. Wu, J. Zhou, D. Shen, 3D auto-context-based locality adaptive multi-modality GANs for PET synthesis. IEEE Trans. Med. Imaging **38**(6), 1328 (2019)
14. B. Yu, L. Zhou, L. Wang, Y. Shi, J. Fripp, P. Bourgeat, Ea-GANs: edge-aware generative adversarial networks for cross-modality MR image synthesis. IEEE Trans. Med. Imaging **38**(7), 1750–1762 (2019)
15. Y. Pan, M. Liu, C. Lian, T. Zhou, Y. Xia, D. Shen, Synthesizing missing PET from MRI with cycle-consistent generative adversarial networks for Alzheimer's disease diagnosis. In: International conference on medical image computing and computer assisted intervention, pp. 455–463 (2018)
16. S. Sadaghiani, G. Hesselmann, K.J. Friston, A. Kleinschmidt, The relation of ongoing brain activity, evoked neural responses, and cognition. Front. Syst. Neurosci. **4**, 20 (2020)
17. V. Kiviniemi, T. Vire, J. Remes, A.A. Elseoud, T. Starck, O. Tervonen, J. Nikkinen, A sliding time-window ICA reveals spatial variability of the default mode network in time. Brain Connect. **1**(4), 339–347 (2011)
18. M. Kudela, J. Harezlak, M.A. Lindquist, Assessing uncertainty in dynamic functional connectivity. NeuroImage **149**, 165–177 (2017)
19. S. Sadaghiani, A. Kleinschmidt, Functional interactions between intrinsic brain activity and behavior. NeuroImage **80**, 379–386 (2013)
20. J. Bernal, K. Kushibar, D.S. Asfaw, S. Valverde, A. Oliver, R. Marti, X. Llado, Deep convolutional neural networks for brain image analysis on magnetic resonance imaging: a review. Artif. Intell. Med. **95**, 64–81 (2019)
21. S.M. Smith, Fast robust automated brain extraction. Hum. Brain Mapp. **17**(3), 143–155 (2002)
22. M. Jenkinson, M. Pechaud, S. Smith, BET2: MR-based estimation of brain, skull and scalp surfaces.

In: Eleventh annual meeting of the organization for human brain mapping (2005)
23. J.G. Sled, A.P. Zijdenbos, A.C. Evans, A nonparametric method for automatic correction of intensity nonuniformity in MRI data. IEEE Trans. Med. Imaging **17**(1), 87–97 (1998)
24. D. Shen, C. Davatzikos, Hammer: hierarchical attribute matching mechanism for elastic registration. IEEE Trans. Med. Imaging **21**(11), 1421–1439 (2002)
25. A. Sotiras, C. Davatzikos, N. Paragios, Deformable medical image registration: a survey. IEEE Trans. Med. Imaging **32**(7), 1153 (2013)
26. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778 (2016)
27. C. Szegedy, S. Ioffe, V. Vanhoucke, A.A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning. In: Thirty-first AAAI conference on artificial intelligence (2017)
28. J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3431–3440 (2015)
29. O. Ronneberger, P. Fischer, T. Brox, U-net: convolutional networks for biomedical image segmentation. In: International conference on medical image computing and computer-assisted intervention (Springer, 2015), pp. 234–241
30. G. Wang, W. Li, S. Ourselin, T. Vercauteren, Automatic brain tumor segmentation using cascaded anisotropic convolutional neural networks. In: International MICCAI brainlesion workshop, 2017, pp. 178–190
31. F. Yu, V. Koltun, T. Funkhouser, Dilated residual networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 472–480
32. K. Kamnitsas, C. Ledig, V.F. Newcombe, J.P. Simpson, A.D. Kane, D.K. Menon, D. Rueckert, B. Glocker, Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation. Med. Image Anal. **36**, 61–78 (2017)
33. M. Liu, J. Zhang, E. Adeli, D. Shen, Landmark-based deep multi-instance learning for brain disease diagnosis. Med. Image Anal. **43**, 157–168 (2018)
34. A.G. Roy, S. Conjeti, N. Navab, C. Wachinger, QuickNAT: a fully convolutional network for quick and accurate segmentation of neuroanatomy. NeuroImage **186**, 713–727 (2019)
35. M. Havaei, A. Davy, D. Warde-Farley, A. Biard, A. Courville, Y. Bengio, C. Pal, P.M. Jodoin, H. Larochelle, Brain tumor segmentation with deep neural networks. Med. Image Anal. **35**, 18–31 (2017)
36. X. Yang, R. Kwitt, M. Styner, M. Niethammer, Quicksilver: fast predictive image registration–a deep learning approach. NeuroImage **158**, 378–396 (2017)
37. J. Fan, X. Cao, P.T. Yap, D. Shen, BIRNet: brain image registration using dual-supervised fully convolutional networks. Med. Image Anal. **54**, 193–206 (2019)
38. O. Maier, B. Menze, J. von der Gablentz, L. Häni, M. Heinrich, M. Liebrand, S. Winzeck et al., ISLES 2015 – a public evaluation benchmark for ischemic stroke lesion segmentation from multi-spectral MRI. Med. Image Anal. **35**, 250–269 (2015)
39. B. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren et al., The multimodal brain tumor image segmentation benchmark (BRATS). IEEE Trans. Med. Imaging **36**(10), 1993–2024 (2015)
40. Ö. Çiçek, A. Abdulkadir, S.S. Lienkamp, T. Brox, O. Ronneberger, 3D U-Net: learning dense volumetric segmentation from sparse annotation. In: International conference on medical image computing and computer-assisted intervention (Springer, 2016), pp. 424–432
41. K. Kamnitsas, W. Bai, E. Ferrante, S. McDonagh, M. Sinclair, N. Pawlowski, M. Rajchl, M. Lee, B. Kainz, D. Rueckert, B. Glocker, Ensembles of multiple models and architectures for robust brain tumour segmentation. In: International MICCAI brainlesion workshop, 2017, pp. 450–462
42. J.L. Elman, Finding structure in time. Cogn. Sci. **14**(2), 179–211 (1990)
43. S. Hochreiter, J. Schmidhuber, Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
44. K. Cho, B. Van Merriënboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: encoder-decoder approaches (2014). arXiv preprint arXiv:1409.1259
45. J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling (2014). arXiv preprint arXiv:1412.3555
46. K. Greff, R.K. Srivastava, J. Koutník, B.R. Steunebrink, J. Schmidhuber, LSTM: a search space odyssey. IEEE Trans. Neural Netw. Learn. Syst. **28**(10), 2222–2232 (2017)
47. H. Chen, Q. Dou, D. Ni, J.Z. Cheng, J. Qin, S. Li, P.A. Heng, Automatic fetal ultrasound standard plane detection using knowledge transferred recurrent neural networks. In: International conference on medical image computing and computer-assisted intervention (Springer, 2015), pp. 507–514
48. B. Kong, Y. Zhan, M. Shin, T. Denny, S. Zhang, Recognizing end-diastole and end-systole frames via deep temporal regression network. In: International conference on medical image computing and computer-assisted intervention (Springer, 2016), pp. 264–272
49. H.C. Shin, M.R. Orton, D.J. Collins, S.J. Doran, M.O. Leach, Stacked autoencoders for unsupervised feature learning and multiple organ detection in a pilot study using 4D patient data. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1930–1943 (2012)

50. Y. Bengio et al., Learning deep architectures for AI. Found. Trends Mach. Learn. **2**(1), 1–127 (2009)
51. R. Salakhutdinov, G. Hinton, Deep Boltzmann machines. In: Artificial intelligence and statistics, 2009, pp. 448–455
52. H. Bourlard, Y. Kamp, Auto-association by multilayer perceptrons and singular value decomposition. Biol. Cybern. **59**(4–5), 291–294 (1988)
53. J. Ker, L. Wang, J. Rao, T. Lim, Deep learning applications in medical image analysis. IEEE Access **6**, 9375–9389 (2017)
54. Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks. In: Advances in neural information processing systems, 2007, pp. 153–160
55. J. Masci, U. Meier, D. Cireşan, J. Schmidhuber, Stacked convolutional auto-encoders for hierarchical feature extraction. In: International conference on artificial neural networks (Springer, 2011), pp. 52–59
56. H.I. Suk, S.W. Lee, D. Shen, Latent feature representation with stacked auto-encoder for AD/MCI diagnosis. Brain Struct. Funct. **220**(2), 841–859 (2015)
57. T. Vercauteren, X. Pennec, A. Perchant, N. Ayache, Diffeomorphic demons: efficient non-parametric image registration. NeuroImage **45**(1), S61–S72 (2009)
58. G. Wu, P.T. Yap, M. Kim, D. Shen, TPS-HAMMER: improving HAMMER registration algorithm by soft correspondence matching and thin-plate splines based deformation interpolation. NeuroImage **49**(3), 2225–2233 (2010)
59. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets. In: Advances in neural information processing systems, 2014, pp. 2672–2680
60. M. Mirza, S. Osindero, Conditional generative adversarial nets (2014). arXiv preprint arXiv:1411.1784
61. P. Costa, A. Galdran, M.I. Meyer, M. Niemeijer, M. Abràmoff, A.M. Mendonça, A. Campilho, End-to-end adversarial retinal image synthesis. IEEE Trans. Med. Imaging **37**(3), 781–791 (2018)
62. Y. Wang, B. Yu, L. Wang, C. Zu, D.S. Lalush, W. Lin, X. Wu, J. Zhou, D. Shen, L. Zhou, 3D conditional generative adversarial networks for high-quality PET image estimation at low dose. NeuroImage **174**, 550–562 (2018)
63. S.U.H. Dar, M. Yurt, L. Karacan, A. Erdem, E. Erdem, T. Çukur, Image synthesis in multi-contrast MRI with conditional generative adversarial networks (2018). arXiv preprint arXiv:1802.01221
64. P. Isola, J.Y. Zhu, T. Zhou, A.A. Efros, Image-to-image translation with conditional adversarial networks (2016). arXiv preprint arXiv:1611.07004
65. J.Y. Zhu, T. Park, P. Isola, A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks. In: IEEE international conference on computer vision, 2017, pp. 2242–2251
66. X. Yi, P. Babyn, Sharpness-aware low-dose CT denoising using conditional generative adversarial network. J. Digit. Imaging **31**(5), 655–669 (2018)
67. L. Bi, J. Kim, A. Kumar, D. Feng, M. Fulham, Synthesis of positron emission tomography (PET) images via multi-channel generative adversarial networks (GANs). In: Molecular imaging, reconstruction and analysis of moving body organs, and stroke imaging and treatment (Springer, 2017), pp. 43–51
68. M. Dadar, T.A. Pascoal, S. Manitsirikul, K. Misquitta, V.S. Fonov, M.C. Tartaglia, J. Breitner, P. Rosa-Neto, O.T. Carmichael, C. Decarli et al., Validation of a regression technique for segmentation of white matter hyperintensities in Alzheimer's disease. IEEE Trans. Med. Imaging **99**, 1–1 (2017)
69. M. Lê, H. Delingette, J. Kalpathy-Cramer, E.R. Gerstner, T. Batchelor, J. Unkelbach, N. Ayache, Personalized radiotherapy planning based on a computational tumor growth model. IEEE Trans. Med. Imaging **36**(3), 815–825 (2017)
70. D.H. Ye, D. Zikic, B. Glocker, A. Criminisi, E. Konukoglu, Modality propagation: coherent synthesis of subject-specific scans with data-driven regularization. In: International conference on medical image computing and computer-assisted intervention (Springer, 2013), pp. 606–613
71. A. Jog, A. Carass, S. Roy, D.L. Pham, J.L. Prince, Random forest regression for magnetic resonance image synthesis. Med. Image Anal. **35**, 475–488 (2017)
72. B. Yu, L. Zhou, L. Wang, J. Fripp, P. Bourgeat, 3D cGAN based cross-modality MR image synthesis for brain tumor segmentation. In: 2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018) (IEEE, 2018), pp. 626–630
73. Y. Pan, M. Liu, C. Lian, Y. Xia, D. Shen, Disease-image specific generative adversarial network for brain disease diagnosis with incomplete multi-modal neuroimages. In: International conference on medical image computing and computer assisted intervention, 2019, pp. 1–9
74. B. Cheng, M. Liu, D. Zhang, B.C. Munsell, D. Shen, Domain transfer learning for MCI conversion prediction. IEEE Trans. Biomed. Eng. **62**(7), 1805–1817 (2015)
75. M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks. In: European conference on computer vision (Springer, 2014), pp. 818–833
76. G. Montavon, S. Lapuschkin, A. Binder, W. Samek, K.R. Müller, Explaining nonlinear classification decisions with deep Taylor decomposition. Pattern Recogn. **65**, 211–222 (2017)
77. A. Kendall, Y. Gal, What uncertainties do we need in Bayesian deep learning for computer vision? In: Advances in neural information processing systems, 2017, pp. 5574–5584