Deep Learning and Its Application to Function Approximation for MR in Medicine: An Overview

Hidenori Takeshima*

This article presents an overview of deep learning (DL) and its applications to function approximation for MR in medicine. The aim of this article is to help readers develop various applications of DL. DL has made a large impact on the literature of many medical sciences, including MR. However, its technical details are not easily understandable for non-experts of machine learning (ML).

The first part of this article presents an overview of DL and its related technologies, such as artificial intelligence (AI) and ML. AI is explained as a function that can receive many inputs and produce many outputs. ML is a process of fitting the function to training data. DL is a kind of ML, which uses a composite of many functions to approximate the function of interest. This composite function is called a deep neural network (DNN), and the functions composited into a DNN are called layers. This first part also covers the underlying technologies required for DL, such as loss functions, optimization, initialization, linear layers, non-linearities, normalization, recurrent neural networks, regularization, data augmentation, residual connections, autoencoders, generative adversarial networks, model and data sizes, and complex-valued neural networks.

The second part of this article presents an overview of the applications of DL in MR and explains how functions represented as DNNs are applied to various applications, such as RF pulse, pulse sequence, reconstruction, motion correction, spectroscopy, parameter mapping, image synthesis, and segmentation.

Keywords: artificial intelligence, machine learning, deep learning

Introduction

In the ImageNet large-scale visual recognition competition 2012, AlexNet showed that a deep neural network (DNN) could significantly improve the performance of an image classification task.1 After this competition, deep learning (DL), which is a machine learning (ML) algorithm using DNNs, has been used in various fields, including MR.

The aim of this article is to help the readers develop various applications of DL for MR in medicine. The first part of this article presents an overview of DL and its related topics, such as artificial intelligence (AI) and ML. The second part of this article presents an overview of the applications of DL. MR images used in this article were approved by the institutional review board, and informed consent was obtained.

Overview of Deep Learning

The aim of this section is to present a technical overview of DL. Overviews of AI and ML, which are necessary for understanding DL, are also provided.

Artificial intelligence

AI is the ability of machines to imitate intelligence. This article will only explain the aspects of AI focused on learning, problem-solving, and planning. Intelligence is not limited to these capabilities. In addition, the term AI is sometimes used interchangeably with ML or DL, regardless of whether or not an implementation actually contains intelligence. The definition of AI is actually not clear.

The purpose of AI is to imitate a function. A process (behavior) of AI can be interpreted as a function. The process accepts some kinds of inputs and generates some kinds of outputs. While hidden states are sometimes used by AI, the states before and after processing can be treated as inputs and outputs, respectively. Example cases of AI are shown in...
In these cases, the functions accept various kinds of images as inputs. Images are typically sampled as discrete signals whose dimensions can range from several tens of thousands to millions. The output of each function is the combination of scalar values. The output of the function (a) contains four scalar values. It represents top-left and bottom-right points of the bounding box around the face. The output of the function (b) contains eight scalar values. It represents two points of the cardiac long axis and two points of the cardiac short axis. The output of the function (c) contains the pixel values of the noise-free image. The number of pixels in the output is the same as that in the input. AI, artificial intelligence.

**Machine learning**
ML is an algorithm, which can be used to implement AI. More specifically, ML can be used for approximating a function, which corresponds to intelligence. ML can process various types of data, such as scalars, vectors, and matrices. In this article, data of various types are simply referred to as data. Depending on the available information, several algorithms, such as supervised learning, unsupervised learning, and reinforcement learning (RL), can be used for function approximation. This article mainly explains supervised learning.

Supervised learning consists of five components: target function, function input, function output, loss function, and training examples. The target function $f$ is an unknown function to be estimated. The function input $x$ is the data, which serves as the input to the target function. The function output $y$ is the data, which is produced by the target function using the function input. The loss function $L$ is the function used to compute the differences between known and estimated function outputs. The training examples (or training data) are used to estimate the target function by providing inputs and the desired outputs during training. Typically, supervised learning algorithms seek a solution for the optimization problem

$$\hat{f}(x) = \arg\min_{f} L(y, f(x))$$

using a set of training examples $T = \{(x_1, y_1), (x_2, y_2), \ldots\}$. 
In supervised learning, the target function is estimated using three datasets: training dataset, validation dataset, and test dataset. The training dataset is a set of training examples \( T \). The validation and test datasets are used for evaluating the performance of the estimated functions.

The validation dataset is used when training processes are performed iteratively with different optimization conditions. Since the target function is optimized for the training dataset, the estimated function \( \hat{f} \) is often unsuitable, for examples, independent from the training dataset. This problem is known as overfitting. Overfitting can be estimated by comparing \( L(y, \hat{f}(x)) \) generated using the training dataset with \( L(y, \hat{f}(x)) \) generated using another set of examples. As a counterpart of overfitting, underfitting means that all candidates of \( \hat{f} \) cannot represent the target function accurately. An example case of overfitting and underfitting is shown in Fig. 2.

The test dataset is used for evaluating the final estimation of \( \hat{f} \). If training and validation processes are performed iteratively and some components of learning are adjusted, then the validation dataset cannot be considered as independent. In this case, it is necessary to prepare one more set of examples.

There are various implementations of supervised learning, such as k-nearest neighbor, logistic regression, decision trees, random forests, support vector machines, linear regression, and artificial neural network (ANN).\(^2,3\)

Unsupervised learning is an algorithm, which does not use examples with outputs during training. Some kinds of clustering and dimensionality reduction are examples of unsupervised learning. RL is an algorithm that does not directly evaluate the function outputs, but rather scores the results generated with the function outputs.

**Artificial neural network**

An ANN is a function used in ML, as shown in Fig. 3. It is a composite function of layers. A layer can be a function or a set of layers. Some layers contain coefficients to be trained. ANNs can be used to represent various target functions by assigning appropriate coefficients to their layers. A loss function and initial coefficients are required to estimate the appropriate coefficients for an ANN. An ANN is often referred to as a neural network.

The loss function defines the score for trained coefficients. The loss function typically represents errors between estimated and desirable outputs of an ANN. The loss function can also contain other functions, such as regularization functions.

Initial coefficients are required for successful optimization based on gradient descent (GD). ANNs are commonly optimized with variants of GD, but other training algorithms can be used as alternatives.

**Forward and backward propagation**

Forward propagation computes the output data of an ANN using the input data. Since an ANN is a composite function, its output can be computed by applying individual layers in the order of composition. This process is referred to as the forward propagation.

Backpropagation (or backward propagation) computes derivatives of coefficients of an ANN. Since derivatives of a composite function can be computed by the chain rule, derivatives for coefficients can be computed by compositing derivatives of layers in the reverse order of composition. This process is referred to as backpropagation.

To optimize the coefficients with GD, it is required to compute the partial derivatives of the loss function with respect to all the coefficients that require training. By using backpropagation, the derivatives of the loss function can be computed from the derivatives of the error function and the individual layers of the ANN.

As shown in Fig. 3, an update step of GD consists of three steps. In the first step, a value of a loss function is computed by running forward propagation from input data. In the second step, derivatives of all coefficients are computed by

![Fig. 2 Overfitting and underfitting in training. Overfitting means that an estimated function is excessively optimized for training data and not suitable for processing unknown data. It is often caused when the number of coefficients is large and/or the number of examples is small. Underfitting, which is rarely seen for DL, occurs when the estimated function cannot represent a target function accurately. It occurs when the number of coefficients is small. DL, deep learning.](image-url)
running backpropagation starting from the value of the loss function. In the third step, all coefficients are updated using the derivatives propagated in the second step. The scaling factors used for updating coefficients are called the learning rate. Coefficients, which should not be updated, can be fixed by skipping the computation of their derivatives in the second step.

**Fully connected and other linear layers**

There are several kinds of layers, which can represent linear transforms. A fully connected layer applies linear transformation to the input data to produce elements of the output data. It uses different coefficients for all the pairs of input and output elements. A convolutional layer applies cross-correlations, which are often referred to as convolutions, to the input data, as shown in Fig. 4. The coefficients of the convolutional layer are the set of kernel elements used for convolutions. A bias layer adds coefficients to the input data. Bias layers are often considered part of fully connected or convolutional layers.

While a fully connected layer can represent arbitrary linear transforms, the number of coefficients needed for a fully connected layer is large. A convolutional layer reduces the number of coefficients by assuming that input data is locally correlated and its correlation is location-independent. ANNs, which use convolutional layers, are known as convolutional neural networks (CNNs).

It is not necessary to update the coefficients of linear transforms. For example, known linear transforms, such as Fourier transforms, can be used. Pre-trained linear transforms can also be used without updating their coefficients.

**Non-linearities (non-linear activation layers)**

An ANN uses non-linear functions to represent a non-linear target function because a composite function of linear transforms is also a linear transform. Non-linear functions composited into an ANN are called non-linearities or non-linear activation layers.

There are various non-linearities, such as sigmoid, tanh, Gaussian, softplus, rectified linear unit (ReLU), softmax, and kervolution. Examples of some functions are shown in Fig. 5. ReLU is defined as the function max(0, x) for each element of the input data x. Softmax is a normalized exponential function. For the softmax function, the sum of the elements of the output data is normalized to 1. Kervolution is a non-linear layer, which provides non-linearity as an extension of a convolution layer.

ReLU is widely used in ANNs whose number of layers is large. In the cases of image recognition tasks using ANNs with three layers, classification errors with ReLU were lower than those with tanh and softplus, and the errors with ReLU were not improved by unsupervised pre-training. Note that the gradients of ReLU are zeros for negative inputs. In the cases where non-zero gradients are desirable for negative...
inputs, several variants of ReLU can be used, such as exponential linear unit (ELU),\textsuperscript{8} leaky rectified linear unit (LeakyReLU),\textsuperscript{9} and parametric ReLU (PReLU).\textsuperscript{10}

**Loss functions**

Loss functions are chosen to allow for the purposes of ML. Loss functions for training ANNs are generally not specific to ANNs. In the case of classification, loss functions evaluate the differences between the probability distributions: $y$ and $\hat{y}(x)$. Examples of the loss functions include cross-entropy and Kullback-Leibler divergence. Hinge loss functions, which are commonly used for training maximum-margin classifiers such as support vector machines, can also be used. In the case of regression, the loss functions evaluate differences between all elements of $y$ and $\hat{y}(x)$. Examples of such loss functions include mean square errors (MSEs), mean absolute errors (MAEs), and Huber loss function. MSEs and MAEs can be interpreted as errors modeled by Gaussian and Laplacian distributions, respectively.\textsuperscript{11} In the case of image segmentation for unbalanced data, loss functions, such as weighted cross-entropy function and (generalized) dice loss function, can be used.\textsuperscript{12}

Loss functions can include other criteria, such as criteria using perceptual structures. CNNs trained for image classification can progressively remove image details without losing perceptual structures.\textsuperscript{13} Therefore, the intermediate outputs of pre-trained CNNs, such as Visual Geometry Group (VGG),\textsuperscript{14} can be used as loss functions.\textsuperscript{15} Such loss functions are called perceptual losses.

**Deep learning**

DL is a form of ML, which uses a DNN to represent the target function.\textsuperscript{7} A DNN is an ANN whose number of layers is large. A DNN is highly non-linear since it contains a large number of non-linear functions.

The following overview explains DL used for supervised learning. There are learning algorithms other than supervised learning, such as RL.\textsuperscript{16,17} It has been shown that DNNs can be used for approximating various functions used in RL.\textsuperscript{18,19}

**Optimization**

Methods based on stochastic gradient descent (SGD) are commonly used for optimizing DNNs. SGD approximates GD in order to increase the training speed. Instead of using all training data, SGD extracts a random subset from the training data and runs GD using the extracted subset for each update step. The extracted subset is called a minibatch.

There are several improvements to SGD focusing on adaptively adjusting gradients or learning rates. The
momentum method uses a weighed sum of gradients at the previous and current update steps. The adaptive gradient (AdaGrad) method and root mean square propagation (RMSProp) method adjust learning rates using past gradients. The adaptive moment estimation (Adam) method adjusts gradients using the first and second moments of past gradients.

Some DNNs optimized using adaptive methods were shown to produce empirically worse performance than SGD for the test dataset. Switching adaptive algorithms to SGD was shown to help overcome this drawback.

**Initialization of coefficients**

Since DNNs are highly non-linear, proper initialization of coefficients is an important step to produce optimized coefficients using GD-based methods. In the cases of large and small coefficients, the gradients of a composite function can often explode and vanish, respectively. Appropriate initialization helps to avoid such problems. Xavier initialization is designed for initializing variances of each linear layer to be 1. He initialization improves this method by considering ReLU and PReLU. For the number of input and output elements $N_{in}$ and $N_{out}$, coefficients are initialized by random numbers whose variances are $2/(N_{in} + N_{out})$ and $2/(1 + a^2_{Prelu})N_{in}$ for Xavier and He initializations, respectively. In the case of He initialization, $a_{Prelu}$ represents the parameter of PReLU and is 0 for ReLU.

Initialization of coefficients can also be performed by transfer learning. Transfer learning (domain adaptation) uses coefficients, which are already trained for other purposes. In the case of DNNs, some functions can be removed partially and some coefficients can be frozen. There are publicly available functions with pre-trained coefficients (pre-trained models), such as VGG and residual networks (ResNets), which can be used for transfer learning.

**Autoencoders**

An autoencoder, also known as an auto-associative neural network, is developed as a method of dimensionality reduction and can be used for initializing coefficients of ANNs. Its purpose is to keep only the essential non-linear components of the function’s inputs. It is classified as unsupervised learning since the autoencoder uses inputs as the desired outputs when training an ANN.

An autoencoder is a composite function of two functions called an encoder and a decoder. The encoder maps the function inputs to a representation with a reduced number of elements. The decoder attempts to recover the function inputs from the outputs of the encoder. An autoencoder with multiple layers can be efficiently initialized using layer-wise initialization. While the original autoencoder requires the number of output elements in the encoder to be reduced, this restriction can be relaxed by either adding noise to function inputs or enforcing sparseness to output elements in the encoder.

**Normalization functions**

The purpose of normalization is to increase the reliability of the training process. Updating the coefficients of upstream layers changes the distribution of inputs into downstream layers. This, in turn, may cause exploding or vanishing gradients for the training steps of downstream layers. The normalization methods are designed to reduce this problem by using estimated distributions during training. Batch normalization is an elementwise normalization function that uses the statistics of each minibatch, followed by an elementwise linear transform. Layer normalization uses the statistics from all elements in a layer to replace the normalization function of batch normalization.

**Recurrent neural network**

A recurrent neural network (RNN) is an ANN that relies on states from the past time points to help predict states at the present time. RNNs accept a series of inputs sequentially, keep the outputs (a state) of the present inputs, and use the state as future inputs. Backpropagation of an RNN can be performed by unfolding it as a variable-length set of ANNs connected in cascade and applying a backpropagation algorithm to the unfolded set of ANNs. This algorithm is called backpropagation through time (BPTT).

BPTT can lead to gradients vanishing or exploding when there are long-term dependencies. Long short-term memory (LSTM) and gated recurrent unit (GRU) are RNNs that can overcome this problem by using additional layers called gates.

**Model size, dataset size, and regularization**

As shown in Fig. 2, overfitting is often caused when the number of coefficients is large and/or the number of examples is small. Since a DNN has a large number of coefficients that need to be trained, coefficients tend to be excessively optimized for training data and are unsuitable for processing unknown data. Overfitting can be prevented by either increasing number of examples or using regularization.

There are empirical results attempting to visualize the number of coefficients and examples (model and dataset sizes) required for training. These results showed that log-log graphs were approximately linear. This behavior is sometimes referred to as the power-law behavior. The relationship between performance of training and number of coefficients to be trained (model size) was observed to be a power-law behavior. The relationship between performance of training and number of examples used for training (dataset size) was also observed to be a power-law behavior.

There are several regularization methods implemented as layers of DNNs. Dropout is a layer that discards the elements of the inputs using a certain probability $p_{dropout}$, which is a training parameter. The remaining elements of the inputs are scaled by $1/(1 - p_{dropout})$. DropConnect is a layer that
discards elements of coefficients to be trained using a methodology similar to Dropout. The DropBlock layer discards blocks instead of elements to improve effectiveness of regularization for convolutional layers.

Regularization can also be implemented by adding regularization terms to loss functions. Common regularization terms use certain functions of coefficients, such as squared norms of coefficients (L2 regularization) and sums of absolute coefficients (L1 regularization).

Data augmentation

Data augmentation increases the number of samples by generating artificial samples from existing samples. It can be used to reduce overfitting problems in ML methodologies such as DL. In the cases of images, methods of data augmentation include horizontal and vertical flipping, cropping, rotation, shifting, color space conversion, histogram equalization, and noise addition. Data augmentation can also be performed using complicated methods, such as mixing images, erasing randomly, and generating artificial samples using generative adversarial networks (GANs). Some data augmentation methods are shown in Fig. 6.

Minibatch size and parallelism

A significant motivation for using large minibatch size is to enable parallelism such as distributed processing. One easy to use implementation of data parallelism splits a dataset into sub-datasets and stores the sub-datasets distributedly. An alternative implementation of parallelism is model parallelism. Model parallelism splits a DNN into sub-DNNs by hand and trains sub-DNNs stored distributedly.

When the number of training examples is kept constant, increasing minibatch size can degrade metrics for test datasets. When the number of update steps is held constant, increasing minibatch size decreased the required number of update steps proportionally until a certain size limit was reached. Increasing minibatch size above this size limit did not change the number of update steps. It was experimentally shown that this size limit depended on DNNs, algorithms of optimization, and datasets.

Generative adversarial networks

GANs aim to generate high-quality artificial samples whose probability distribution is near that of the training examples. As shown in Fig. 7a and 7b, GANs consist of two functions, which are called generator and discriminator. The generator is a DNN whose input is seed data and output is an artificial sample. The discriminator is a DNN whose input is a sample and output is a value that characterizes the input as a training example or an artificial sample. From the prospective of the generator, the discriminator is a loss function for training. Two DNNs are trained simultaneously. It was shown that GANs could be used for generating artificial images. Stability of training can be improved by using Wasserstein distances. The discriminator can be trained using negative samples generated artificially.

Conditional GANs generate artificial samples for the given conditions. Conditions are added to the inputs of both generator and discriminator as shown in Fig. 7c and 7d. This modification allows GANs to control the kinds of samples generated. An example of conditional GANs is pix2pix.

Composite functions used in various applications

There are composite functions used in various applications. Some of composite functions are shown in Fig. 8.

A ResNet is a CNN with shortcut connections. A shortcut connection adds inputs of some (e.g. two) stacked layers to outputs of the stacked layers. These shortcut connections are useful for avoiding the vanishing gradient problem caused by backpropagation. A densely connected convolutional network (DenseNet) improves a ResNet by using the inputs of a layer as the inputs of all the succeeding layers.
U-net architecture was developed for the purpose of image segmentation. It uses a CNN, which processes inputs in multiple scales. In the former process of the U-net, CNNs and down-sampling functions are applied alternately. The outputs of CNNs for all scales are used as the inputs for the latter process. The former process is referred to as the encoder. The latter process applies CNNs and up-sampling functions alternately. The inputs of CNNs in the latter process are outputs of both previous functions and CNNs of the encoder. The latter process is referred to as the decoder.

CNNs that process inputs in multiple scales, including the U-net, use down-sampling and up-sampling functions. Examples of down-sampling functions are pooling functions. For each local region, such as a 2x2 block of inputs, max pooling and average pooling compute the max and average values of the region, respectively. Convolution layers whose strides (shifts for each operation) are greater than 1 are also used as down-sampling functions. Up-sampling can be implemented as either repeating inputs or as convolution layers whose strides are less than 1. The latter layers are known as deconvolution layers.
**Complex-valued artificial neural networks**

There is a significant advantage in using ANNs to process complex numbers directly. Complex numbers can be represented as real and imaginary numbers. However, when representing a multiplication of a complex coefficient by a complex number, the required number of real-valued coefficients is 4 for real-valued ANNs and 2 for complex-valued ANNs. Therefore, ANNs using complex numbers directly are expected to optimize easier than ANNs using real and imaginary numbers.

Since backpropagation is an implementation of GD, it can be used for functions that are complex differentiable (called holomorphic). In addition, backpropagation can be performed for non-linear functions that can be separated into the real and imaginary parts of differentiable functions, even if these non-linear functions are not holomorphic.

There are extensions of ReLU, batch normalization, and coefficient initialization for complex-valued ANNs.

**Software tools**

For implementing DNNs, easy-to-use software tools exist that can automatically compute the gradients for functions used in DNNs. There are several software tools that have this capability, known as automatic differentiation. Examples of such tools are TensorFlow, Keras, and PyTorch. These tools provide capabilities of fast computation using general-purpose computing on graphics processing units (GPGPU), which significantly accelerate the computation of optimizing DNNs.

**Limitations**

Similar to other ML algorithms, a major limitation of DL is that it is not robust against dataset shifts. A dataset shift occurs when there are differences between the training and test datasets. It was demonstrated that DNNs trained using examples with eight kinds of distortions failed to classify examples using another unknown kind of distortion. In addition, it was demonstrated that DNNs trained with synthetic data augmentations designed for robustness intervention provide little improvement under dataset shifts on real data.

Another limitation of DNNs is its low interpretability. Since structures of DNNs are often too complicated, they are considered as black-boxes. Since it is desirable to make inference processes human-interpretable in some research fields, there were various attempts to explain inference processes uninterpretable to humans. However, a trade-off between interpretability and accuracy was found. In addition, a paper pointed out that directly training interpretable functions was better than training another function to explain uninterpretable functions such as DNNs.

**Applications of Deep Learning to Function Approximation for MR in Medicine**

This section presents an overview of applications of DL to function approximation for MR in medicine. This section specifically focuses on the inputs and outputs of functions approximated by DNNs. Since there are a wide variety of applications of DL, there are many applications not covered in this article.

**RF pulse and pulse sequence**

For generating RF waveforms, a DNN was used to approximate a function that received desired RF excitation maps as the inputs and produced an RF waveform as the output. The DNN was used for fast processing. While iterative optimization algorithms generate a good multidimensional RF pulse, they require substantial computation time. It was shown that fast generation of RF waveforms could be achieved by utilizing the short computation time of forward propagation.

DNNs could be used for generating entire pulse sequences and the under-sampling pattern of pulse sequences. In the first method, DNNs approximated functions used in RL. In the second method, a DNN approximated a denoising function that received an image as the input and produced a noise-free image as the output. It is a composite function of a forward Fourier transform, an under-sampling function to be estimated, an inverse Fourier transform, and a denoising function.

**Denoising, artifact removal, and super-resolution**

DNNs were used for denoising images in various ways. Typically, DNNs were used to approximate functions that received an image with noise as the input and produced a noise-free image as the output. The DNNs were used for improving image quality with short computation time. For increasing SNR, a DNN was used to approximate functions that received an image with number of excitation (NEX) of 1 as the input and produced an image with NEX of 10 as the output as shown in Fig. 9. DNNs were used for deblurring off-resonance blur.

In the case of dynamic imaging, DNNs approximated functions that received multiple frames as the inputs and produced a single frame as the output and received multiple frames as the inputs and produced multiple frames as the outputs. As shown in Fig. 10, multiple frames reconstructed in multiple temporal resolutions could be used as inputs of DNNs to increase the quality of outputs.

For suppressing artifacts, DNNs were used to approximate functions that received an image with artifacts as the input and produced an artifact-free image as the output. The effectiveness of DNNs was shown for various artifact types, such as streak artifacts of radial k-space data, artifacts caused by simultaneous multi-slice (SMS), Gibbs ringing artifacts, and MR-specific aliasing artifacts. MR-specific aliasing artifacts could be processed using a CNN with additional layers specific to this purpose.

In the case of metal artifact removal, a DNN was trained without artifact-free images. The DNN approximated a function that received distorted images and produced the
distorted images whose geometric shifts were opposite to the input images.

The implementations of super-resolution were similar to those of denoising. The DNNs for super-resolution approximated functions that received a low-resolution (or an interpolated high-resolution) image as the input and produced a high-resolution image as the output. It was shown that a DNN was efficient as a super-resolution method for musculoskeletal images\(^9\) and MR angiography (MRA).\(^9\)

**Alternating optimization for reconstruction**

DNNs were used in alternating optimization algorithms. An image can be reconstructed from k-space data by running minimization of measurement errors and removal of noise (reconstruction artifacts) alternately. DNNs for alternating optimization approximated the functions corresponding to the entire processes of alternating optimization that received k-space data as the input and produced a reconstructed image as the output.\(^9,10,11,12\) They used unrolled functions of alternating optimization algorithms, such as alternating direction method of multipliers (ADMM). It was also shown that a conjugate gradient-based approach was better than a proximal gradients based approach for minimization of measurement errors.\(^11\) A DNN was also used as an alternating optimization algorithm to predict the probability distribution of all pixels.\(^13\)

**Other topics related with reconstruction**

It has been shown that a DNN can reconstruct an image from k-space data without being provided explicit information on the acquisition processes.\(^10\) The DNN approximated functions that received raw acquisition data as the input and produced an image as the output. Reconstruction processes, such as correction of hardware imperfections and Fourier transforms, were also estimated from the training data.

Parallel imaging in k-space can be implemented as an ANN with three layers.\(^15,16\) This method serves a similar purpose as nonlinear generalized autocalibrating partially parallel acquisition (GRAPPA),\(^17\) which was an extension of GRAPPA\(^18\) using a kernel method for nonlinear mapping. The ANN approximated a function that received undersampled multichannel k-space data as the input and produced full-sampled k-space data as the output. The approximated function was trained on autocalibration signal (ACS) data.

A limitation of DNN-based reconstruction methods is the demonstrated fragility against small amounts of adversarial attacks and structural changes.\(^19\)

**Motion correction**

DNNs were used to remove motion artifacts.\(^10,11\) The DNNs approximated functions which produced a motion-free patch from a motion-affected patch. A discriminator network, which itself was a DNN, was shown to behave as an efficient loss function for this purpose.\(^10\) It was shown that a DNN could be used as a part of alternating optimization.\(^11\)

**Spectroscopy**

DNNs have been used for denoising in the field of magnetic resonance spectroscopy (MRS)\(^11,12,13,14\) and chemical-shift imaging (CSI).\(^15,16\) In the case of MRS, DNNs were used to approximate a function that received a short-time spectrogram as the input and produced the presence of artifacts as the output,\(^11\) a function that received a spectrogram with ghosting artifacts as the input and produced an artifact-free spectrogram as the output,\(^11\) and functions that received a spectrum with several kinds of noise as the input and produced a metabolite-only spectrum as the output.\(^13,14\) In the case of CSI, DNNs were used to approximate a function that received a spectrum as the input and produced a scalar quality value as the output,\(^11\) and a function that received a spectrum with noise as the input and produced a noise-free spectrum as the output.\(^11\)

**Parameter mapping**

ANNs were used for estimating per-pixel tissue parameters, such as T1, T2, and offset of resonance frequency. In the case of MR fingerprinting,\(^17\) ANNs approximated a function that received a series of images as the inputs and produced tissue parameters as the outputs.
parameters as the outputs;\textsuperscript{118} and a function that received a series of images and cardiac R-R intervals as the inputs, and produced tissue parameters as the outputs.\textsuperscript{119} For estimating T1 and T2 values using a MOLLI sequence,\textsuperscript{120} a DNN was used to approximate a function that received a time series of 1-dimensional signals and time stamps as the inputs, and produced T1 and T2 values as the outputs.\textsuperscript{121}

DNNs were used for estimating electrical properties tomography (EPT) by approximating functions, which produce EPT from B1+ amplitudes, transceiver phase, and existence of tissue.\textsuperscript{122,123} A DNN was also used to approximate a function that produces a map of quantitative susceptibility mapping (QSM) from a local field map.\textsuperscript{124}

In the field of diffusion analysis, DNNs were used as a function that received under-sampled q-space data as the input, and produced neurite orientation dispersion and density imaging (NODDI) and generalized fractional anisotropy (GFA) parameters as the outputs;\textsuperscript{125} and a function that received diffusion-weighted images as the inputs, and produced fractional anisotropy (FA) and mean diffusivities (MD) parameters as the outputs.\textsuperscript{126}

**Imaging using magnetization transfer**

DNNs were used for the purpose of estimating quantitative magnetization transfer (qMT) parameters from a small number of MT images.\textsuperscript{127} This method\textsuperscript{127} used DNNs as a function...
that received MT images as the inputs and produced qMT parameters directly as the outputs, and a function that received MT images as the inputs and produced interpolated MT images as the outputs. In the latter case, the interpolated MT images were used for estimating qMT parameters.

In the field of chemical exchange saturation transfer (CEST), DNNs approximated a function that received images with several frequency offsets as the inputs, and produced frequency offset images at 3 ppm and -3 ppm for generating a CEST image as the outputs; and a function that received several images with different frequency offsets and under-sampling patterns as the inputs, and produced a CEST image as the output. In the latter method, input images were acquired using periodically rotated overlapping parallel lines with enhanced reconstruction (PROPELLER).

**Image synthesis**

DNNs were used to synthesize other types of images. In the case of synthetic CT, a DNN approximated a function that received reconstructed images of a Dixon sequence as the inputs and produced a CT image as the output. DNNs were also used to approximate a function that received multi-echo gradient-recalled echo images as the inputs and produced water and fat images as the outputs; and a function that received multi-echo gradient-recalled echo images as the inputs and produced fat fraction maps as the outputs.

For synthesizing a short inversion time inversion recovery (STIR) image, a DNN approximated a function which produced a STIR image from T1-weighted, T2-weighted, and gradient-recalled echo images. As a part of a T2 mapping method, ANNs approximated functions that produced a T2-weighted image for several types of tissues from a T1-weighted image.

**Segmentation**

DNNs were used for segmentation of various tissues. DNNs were used as a function that produced a segmentation map of the brain tumors from images; a function that produced a segmentation map of the knee tissues from an image; a function that produced probability maps and boundary information of the vessel wall from an image patch extracted as the vessel wall; a function that produced a segmentation map of the intervertebral disc from an image; a function that produced visceral adipose tissue (VAT) and subcutaneous adipose tissue (SAT) from water, fat, and fat fraction images at the abdomen; and a function that produced a segmentation map of the thoracic aorta from phase-contrast MR angiogram (PCMRA).

**Other applications**

For estimating Specific Absorption Rate (SAR), a DNN was used to approximate a function that received a transmit field (B1+) as the input and produced a distribution of SAR as the output. The DNN was used as a part of the method for estimating local SAR accurately.

In the field of deformable image registration for MR brain images, a stacked autoencoder was used as a feature extraction method of image registration. The encoder of the stacked autoencoder produced a small set of coefficients from a set of inputs. For estimating oxygen extraction fraction, an ANN was used to approximate a function that received blood-oxygenation-level-dependent (BOLD) signals as the inputs and produced four-dimensional parameters, including oxygen extraction fraction as the outputs. For estimating tissue stiffness from displacement images of MR elastography, an ANN approximated a function that received a patch as the input and produced tissue stiffness as the output. A DNN was also used for predicting a brain age by approximating a function that received 3D brain images as the inputs and produced an age value as the output.

**Conclusion**

This article presented an overview of DL and its applications to function approximation for MR in medicine. The author hopes that this article helps readers to develop and use various applications of DL.

**Acknowledgments**

The author would like to thank Kensuke Shinoda and Yutaka Hoshiyama for providing a cardiac image used in Fig. 1. The author would also like to thank Tatsuya Ohkubo and Kensuke Shinoda for providing Fig. 9.

**Conflicts of Interest**

Hidenori Takeshima is an employee of Canon Medical Systems Corporation.

**References**

1. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Proceedings of the 25th International Conference on Neural Information Processing Systems, 2012; 1:1097–1105.
2. Kotsiantis SB, Zaharakis ID, Pintelas PE. Machine learning: a review of classification and combining techniques. Artif Intell Rev 2006; 26:159–190.
3. Dreisell S, Ono-Machado L. Logistic regression and artificial neural network classification models: a methodology review. J Biomed Inform 2002; 35:352–359.
4. LeCun Y, Bottou L, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998; 86:2278–2324.
5. Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks. Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, PMLR, 2011; 15:315–323.
6. Wang C, Yang J, Xie L, et al. Kervolutional neural networks. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019; 31–40.
LeCun Y, Bengio Y, Hinton G. Deep learning. Nature 2015; 521:436–444.

Clevert D, Unterthiner T, Hochreiter S. Fast and accurate deep network learning by exponential linear units (ELUs). arXiv:1511.07289.

Maas AL, Hannun AY, Ng AY. Rectifier Nonlinearities Improve Neural Network Acoustic Models. Proceedings of the ICML 2013 workshop on Deep Learning for Audio, Speech and Language Processing (WDLASL 2013), Atlanta.

Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control using deep reinforcement learning. Nature 2015; 518:475–502.

Francois-Lavet V, Henderson P, Islam R, et al. An introduction to deep reinforcement learning. arXiv:1811.10256.

Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. Nature 2015; 518:459–465.

Mnih V, Badia AP, Mirza M, et al. Asynchronous methods for deep reinforcement learning. Proceedings of The 33rd International Conference on Machine Learning, PMLR 2016; 48:1928–1937.

Qian N. On the momentum term in gradient descent learning algorithms. Neural Netw 1999; 12:145–151.

Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res 2011; 12:2121–2159.

Geoffrey H. rmsprop: divide the gradient by a running average of its recent magnitude. Neural Networks for Machine Learning Lecture 6e; 26–30. http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf. (Accessed: Feb 24, 2021)

Kingma DP, Ba J. Adam: a method for stochastic optimization. arXiv:1412.6980.

Wilson AC, Roelofs R, Stern M, et al. The marginal value of adaptive gradient methods in machine learning. Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, 2017; 4151–4161.

Keskar NS, Socher R. Improving generalization performance by switching from Adam to SGD. arXiv:1712.07628.

Luo L, Xiong Y, Liu Y, et al. Adaptive gradient methods with dynamic bound of learning rate. arXiv:1902.09843.

Glorot X, Bengio Y. Understanding the difficulty of training deep feed forward neural networks. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, PMLR, 2010; 9:249–256.

Shao L, Zhu F, Li X. Transfer learning for visual categorization: a survey. IEEE Trans Neural Netw Learning Syst 2015; 26:1019–1034.

He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2016; 770–778.

Kramer MA. Nonlinear principal component analysis using autoassociative neural networks. AIChE Journal 1991; 37:233–243.

Bengio Y, Lamblin P, Popovici D, et al. Greedy layer-wise training of deep networks. Proceedings of the 19th International Conference on Neural Information Processing Systems, 2006; 153–160.

Vincent P, Larochelle H, Bengio Y, et al. Improving the convergence of generic deep networks by computing decomposition robust features with denoising autoencoders. Proceedings of the 25th International Conference on Machine Learning, Helsinki, 2008; 1096–1103.

Ng A. Sparse autoencoder. CS294A Lecture notes. https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf. (Accessed: May 9, 2021)

Xu J, Xiang L, Liu Q, et al. Stacked Sparse Autoencoder (SSAE) for nuclei detection on breast cancer histopathology images. IEEE Trans Med Imaging 2016; 35:119–130.

Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift. Proceedings of the 32nd International Conference on Machine Learning, PMLR, 2015; 37:448–456.

Ba JL, Kiros JR, Hinton GE. Layer normalization. arXiv:1607.06450.

Werbos PJ. Backpropagation through time: what it does and how to do it. Proc IEEE 1990; 78:1550–1560.

Gers FA, Schmidhuber J, Cummins F. Learning to forget: continual prediction with LSTM. Neural Comput 2000; 12:2451–2471.

Cho K, van Merrienboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv:1406.1078.

Tan M, Le Q. EfficientNet: rethinking model scaling for convolutional neural networks. Proceedings of the 36th International Conference on Machine Learning, PMLR, 2019; 97:6105–6114.
41. Rosenfeld JS, Rosenfeld A, Belinkov Y, et al. A constructive prediction of the generalization error across scales. arXiv:1909.12673.
42. Hestness J, Narang S, Ardalani N, et al. Deep learning scaling is predictable, empirically. arXiv:1712.00409.
43. Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 2014; 15:1929–1958.
44. Wan L, Zeiler M, Zhang S, et al. Regularization of neural networks using DropConnect. Proceedings of the 30th International Conference on Machine Learning, PMLR, 2013; 28:1058–1066.
45. Ghiasi G, Lin T, Le QV. DropBlock: a regularization method for convolutional networks. Proceedings of Advances in Neural Information Processing Systems 31, Montréal, 2018.
46. Ng AY. Feature selection, L1 vs. L2 regularization, and rotational invariance. Proceedings of the 21st International Conference on Machine Learning, Banff, 2004; 615–622.
47. Shorten C, Khoshgoftaar TM. A survey on image data augmentation for deep learning. J Big Data 2019; 6:60.
48. Inoue H. Data augmentation by pairing samples for images classification. arXiv:1801.02929.
49. Summers C, Dinneen MJ. Improved mixed-example data augmentation. arXiv:1805.11272.
50. Zhong Z, Zheng L, Kang G, et al. Random erasing data augmentation. arXiv:1708.04896.
51. Bowles C, Chen L, Guerrero R, et al. GAN augmentation: augmenting training data using generative adversarial networks. arXiv:1810.10863.
52. Sandfort V, Yan K, Pickhardt PJ, et al. Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks. Sci Rep 2019; 9:16884.
53. Krizhevsky A. One weird trick for parallelizing convolutional neural networks. arXiv:1404.5997.
54. You Y, Zhang Z, Hsieh CJ, et al. ImageNet training in minutes. Proceedings of the 47th International Conference on Parallel Processing, Eugene, 2018; 1–10.
55. Coates A, Hwuval B, Wang T, et al. Deep learning with COTS HPC systems. Proceedings of the 30th International Conference on Machine Learning, Atlanta, 2013; 28: 1337–1345.
56. Wilson DR, Martinez TR. The general inefficiency of batch training for gradient descent learning. Neural Netw 2003; 16:1429–1451.
57. Masters D, Luschi C. Revisiting small batch training for deep neural networks. arXiv:1804.07612.
58. Shallue CJ, Lee J, Antognini J, et al. Measuring the effects of data parallelism on neural network training. J Mach Learn Res 2019; 20:1–49.
59. Goodfellow IJ, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets. Proceedings of Advances in Neural Information Processing Systems 27, Montréal, 2014; 2672–2680.
60. Radford A, Metz L, Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv:1511.06434.
61. Arjovsky M, Chintala S, Bottou L. Wasserstein generative adversarial networks. Proceedings of the 34th International Conference on Machine Learning, PMLR 2017; 70:214–223.
62. Gulrajani I, Ahmed F, Arjovsky M, et al. Improved training of Wasserstein GANs. Proceedings of Advances in Neural Information Processing Systems 30, Long Beach, 2017; 5769–5779.
63. Shinha A, Ayush K, Song J, et al. Negative data augmentation. arXiv:2102.05113.
64. Mirza M, Osindero S. Conditional generative adversarial nets. arXiv:1411.1784.
65. Isola P, Zhu J, Zhou T, et al. Image-to-image translation with conditional adversarial networks. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2017; 1125–1134.
66. Huang G, Liu Z, van der Maaten L, et al. Densely connected convolutional networks. Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, 2017; 2261–2269.
67. Ronneberger O, Fischer P, Brox T. U-net: convolutional networks for biomedical image segmentation. Proceedings of the 18th International Conference on Medical Image Computing and Computer-Assisted Intervention, Munich, 2015; 234–241.
68. Hirose A. Complex-valued neural networks (studies in computational intelligence). Berlin Heidelberg:Springer, 2006.
69. Nitta T. An extension of the back-propagation algorithm to complex numbers. Neural Netw 1997; 10:1391–1415.
70. Trabelsi C, Bilaniuk O, Zhang Y, et al. Deep complex networks. arXiv:1705.09792.
71. Baydin AG, Pearlmutter BA, Radul AA, et al. Automatic differentiation in machine learning: a survey. J Mach Learn Res 2018; 18:1–43.
72. Abadi M, Agarwal A, Barham P, et al. TensorFlow: large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/ (Accessed: Feb 24, 2021)
73. Chollet F, et al. Keras. https://keras.io/ (Accessed: Feb 24, 2021)
74. Abadi M, Barham P, Chen J, et al. TensorFlow: A system for large-scale machine learning. Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah, 2016; 263–283.
75. Owens JD, Luebke D, Govindaraju N, et al. A survey of general-purpose computation on graphics hardware. Proceedings of Eurographics 2005 STAR (State of the Art Reports), 2005; 21–51. [https://research.nvidia.com/sites/default/files/pubs/2005-08_A-Survey-of/ASurveyofGeneralPurposeComputationonGraphicsHardware.pdf] (Accessed: Feb 24, 2021)
76. Raina R, Madhavan A, Ng AY. Large-scale deep unsupervised learning using graphics processors. Proceedings of the 26th International Conference on Machine Learning, Montréal, 2009; 873–880.
77. Sugiyama M, Krauledat M, Muller KR. Covariate shift adaptation by importance weighted cross validation. J Mach Learn Res 2007; 8:985–1005.
78. Taori R, Dave A, Shankar V, et al. Measuring robustness to natural distribution shifts in image classification. Proceedings of Advances in Neural Information Processing Systems 33, online, 2020; 18583–18599.
79. Geirhos R, Temme CRM, Rauber J, et al. Generalisation in humans and deep neural networks. Proceedings of Advances in Neural Information Processing Systems, Montréal, 2018; 7549–7561.

H. Takeshima
80. Arrieta AB, Diaz-Rodriguez N, Ser JD, et al. Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. Inform Fusion 2020; 58:82–115.
81. Rudin C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence 2019; 1:206–215.
82. Vinding MS, Skyum B, Sangill R, et al. Ultrafast (milliseconds), multidimensional RF pulse design with deep learning. Magn Reson Med 2019; 82:586–599.
83. Bahadir CD, Wang AQ, Dalca AV, et al. Deep-learning-based optimization of the under-sampling pattern in MRI. IEEE Trans Comput Imaging 2020; 6: 1139–1152.
84. Zhu B, Liu J, Koonjoo N, et al. Automated pulse sequence generation (AUTOSEQ) using Bayesian reinforcement learning in an MRI physics simulation environment. Proceedings of Joint Annual Meeting ISMRM-ESMRMB 2018, Paris, 2018: 438.
85. Kidoh M, Shinoda K, Kitajima M, et al. Deep learning based noise reduction for brain MR imaging: tests on phantoms and healthy volunteers. Magn Reson Med Sci 2020; 19:195–206.
86. Lim Y, Bliesener Y, Narayanan S, et al. Deblurring for spiral real-time MRI using convolutional neural networks. Magn Reson Med 2020; 84:3438–3452.
87. Zeng DY, Shaikh J, Holmes S, et al. Deep residual network for off-resonance artifact correction with application to pediatric body MRA with 3D cones. Magn Reson Med 2019; 82:1398–1411.
88. Takeshima H. Integrating spatial and temporal correlations into a deep neural network for low-delay reconstruction of highly undersampled radial dynamic images. Proceedings of Joint Annual Meeting ISMRM-ESMRMB 2018, Paris, 2018; 2796.
89. Hauptmann A, Arridge S, Lucka F, et al. Real-time cardiovascular MR with spatio-temporal artifact suppression using deep learning-proof of concept in congenital heart disease. Magn Reson Med 2019; 81:1143–1156.
90. Takeshima H, Kutsuna H. Fast acquisition and low-delay reconstruction of stack-of-stars trajectory using temporal multisolution images and a convolutional neural network. Proceedings of ISMRM 27th Annual Meeting & Exhibition, Montreal, 2019; 467.
91. Han Y, Yoo J, Kim HH, et al. Deep learning with domain adaptation for accelerated projection-reconstruction MR. Magn Reson Med 2018; 80:1189–1205.
92. Bilgic B, Chatnuntawech I, Manhard MK, et al. Highly accelerated multishot echo planar imaging through synergistic machine learning and joint reconstruction. Magn Reson Med 2019; 82:1343–1358.
93. Zhang Q, Ruan G, Yang W, et al. MRI Gibbs-ringing artifact reduction by means of machine learning using convolutional neural networks. Magn Reson Med 2019; 82:2133–2145.
94. Muckley MJ, Ades-Aron B, Papaioannou A, et al. Training a neural network for Gibbs and noise removal in diffusion MRI. Magn Reson Med 2021; 85:413–428.
95. Takeshima H. Aliasing layers for processing parallel imaging and EPI ghost artifacts efficiently in convolutional neural networks. Magn Reson Med 2021; 86:820–834.
114. Lee HH, Kim H. Deep learning-based target metabolite isolation and big data-driven measurement uncertainty estimation in proton magnetic resonance spectroscopy of the brain. Magn Reson Med 2020; 84:1689–1706.

115. Gurbani SS, Schreibmann E, Maudsley AA, et al. A convolutional neural network to filter artifacts in spectroscopic MRI. Magn Reson Med 2018; 80:1765–1775.

116. Gurbani SS, Sheriff S, Maudsley AA, et al. Incorporation of a spectral model in a convolutional neural network for accelerated spectral fitting. Magn Reson Med 2019; 81:3346–3357.

117. Ma D, Gulani V, Seiberlich N, et al. Magnetic resonance fingerprinting. Nature 2013; 495:187–192.

118. Cohen O, Zhu B, Rosen MS. MR fingerprinting deep reconstruction network (DRONE). Magn Reson Med 2018; 80:885–894.

119. Hamilton JI, Currey D, Rajagopalan S, et al. Deep learning reconstruction for cardiac magnetic resonance fingerprinting T1 and T2 mapping. Magn Reson Med 2021; 85:2127–2135.

120. Messroghli DR, Radjenovic A, Kozerke S, et al. Modified Look-Locker inversion recovery (MOLLI) for high-resolution T1 mapping of the heart. Magn Reson Med 2004; 52:141–146.

121. Shao J, Ghodrati V, Nguyen KL, et al. Fast and accurate calculation of myocardial T1 and T2 values using deep learning Bloch equation simulations (DeepBLESS). Magn Reson Med 2020; 84:2831–2845.

122. Mandija S, Meliaðò EF, Huttinga NRF, et al. Opening a new window on MR-based electrical properties tomography with deep learning. Sci Rep 2019; 9:88–95.

123. Gavazzi S, van den Berg CAT, Savenije MHF, et al. Deep learning to perform quantitative susceptibility mapping using deep neural network: QSMnet. Magn Reson Med 2020; 83:1109–1122.

124. Yoon J, Gong E, Chatnuntawech I, et al. Artificial neural network to filter artifacts in spectroscopic MRI. Magn Reson Med 2020; 83:1109–1122.

125. Meng Z, Guo R, Li Y, et al. Accelerating T2 mapping of the brain by integrating deep learning priors with low-rank and sparse modeling. Magn Reson Med 2021; 85:1455–1467.

126. Pereira S, Pinto A, Alves V, et al. Brain tumor segmentation using convolutional neural networks in MRI images. IEEE Trans Med Imaging 2016; 35:1240–1251.

127. Byra M, Wu M, Zhang X, et al. Knee menisci segmentation and relaxometry of 3D ultrashort echo time cones MR imaging using attention U-Net with transfer learning. Magn Reson Med 2020; 83:1109–1122.

128. Chen L, Canton G, Liu W, et al. Fully automated and robust analysis technique for popliteal artery vessel wall evaluation (FRAPPE) using neural network models from standardized knee MRI. Magn Reson Med 2020; 84:2147–2160.

129. Iriondo C, Pedoia V, Majumdar S. Lumbar intervertebral disc characterization through quantitative MRI analysis: an automatic voxel-based relaxometry approach. Magn Reson Med 2020; 84:1376–1390.

130. Langner T, Hedström A, Mörlwald K, et al. Fully convolutional networks for automated segmentation of abdominal adipose tissue depots in multicenter water-fat MRI. Magn Reson Med 2019; 81:2736–2745.

131. Florkow MC, Zijlstra F, WillemSEN K, et al. Deep learning-based MR-to-CT synthesis: the influence of varying gradient echo-based MR images as input channels. Magn Reson Med 2020; 83:1429–1441.

132. Cho J, Park H. Robust water-fat separation for multi-echo gradient-recalled echo sequence using convolutional neural network. Magn Reson Med 2019; 82:476–484.

133. Andersson J, Ahlström H, Kullberg J. Separation of water and fat signal in whole-body gradient echo scans using convolutional neural networks. Magn Reson Med 2019; 82:1177–1186.

134. Kim S, Jang H, Jang J, et al. Deep-learned short tau inversion recovery imaging using multi-contrast MR images. Magn Reson Med 2020; 84:2994–3008.

135. Gurbani SS, Sheriff S, Maudsley AA, et al. Incorporation of a spectral model in a convolutional neural network for accelerated spectral fitting. Magn Reson Med 2021; 85:3346–3357.

136. Luu HM, Kim DH, Kim JW, et al. qMTNet: Accelerated saturation transfer imaging based on PROPELLER acquisition and deep neural network reconstruction. Magn Reson Med 2020; 84:3192–3205.

137. Pipe JG. Motion correction with PROPELLER MRI: application to head motion and free-breathing cardiac imaging. Magn Reson Med 1999; 42:963–969.