Introduction to machine and deep learning for medical physicists

Sunan Cui[a)]
Department of Radiation Oncology, University of Michigan, Ann Arbor, MI 48103, USA
Applied Physics Program, University of Michigan, Ann Arbor, MI 48109, USA

Huan-Hsin Tseng
Department of Radiation Oncology, University of Michigan, Ann Arbor, MI 48103, USA

Julia Pakela
Department of Radiation Oncology, University of Michigan, Ann Arbor, MI 48103, USA
Applied Physics Program, University of Michigan, Ann Arbor, MI 48109, USA

Randall K. Ten Haken and Issam El Naqa
Department of Radiation Oncology, University of Michigan, Ann Arbor, MI 48103, USA

(Received 30 September 2019; revised 23 January 2020; accepted for publication 3 March 2020; published 15 May 2020)

Recent years have witnessed tremendous growth in the application of machine learning (ML) and deep learning (DL) techniques in medical physics. Embracing the current big data era, medical physicists equipped with these state-of-the-art tools should be able to solve pressing problems in modern radiation oncology. Here, a review of the basic aspects involved in ML/DL model building, including data processing, model training, and validation for medical physics applications is presented and discussed. Machine learning can be categorized based on the underlying task into supervised learning, unsupervised learning, or reinforcement learning; each of these categories has its own input/output dataset characteristics and aims to solve different classes of problems in medical physics ranging from automation of processes to predictive analytics. It is recognized that data size requirements may vary depending on the specific medical physics application and the nature of the algorithms applied. Data processing, which is a crucial step for model stability and precision, should be performed before training the model. Deep learning as a subset of ML is able to learn multilevel representations from raw input data, eliminating the necessity for hand crafted features in classical ML. It can be thought of as an extension of the classical linear models but with multilayer (deep) structures and nonlinear activation functions. The logic of going “deeper" is related to learning complex data structures and its realization has been aided by recent advancements in parallel computing architectures and the development of more robust optimization methods for efficient training of these algorithms. Model validation is an essential part of ML/DL model building. Without it, the model being developed cannot be easily trusted to generalize to unseen data. Whenever applying ML/DL, one should keep in mind, according to Amara’s law, that humans may tend to overestimate the ability of a technology in the short term and underestimate its capability in the long term. To establish ML/DL role into standard clinical workflow, models considering balance between accuracy and interpretability should be developed. Machine learning/DL algorithms have potential in numerous radiation oncology applications, including automatizing mundane procedures, improving efficiency and safety of auto-contouring, treatment planning, quality assurance, motion management, and outcome predictions. Medical physicists have been at the frontiers of technology translation into medicine and they ought to be prepared to embrace the inevitable role of ML/DL in the practice of radiation oncology and lead its clinical implementation. [https://doi.org/10.1002/mp.14140]
Key words: deep learning, machine learning, medical physics

1. INTRODUCTION

Applications of machine learning (ML) and deep learning (DL), as a branch of intelligence (AI) in medical physics have witnessed rapid growth over the past few years. These techniques have been studied as effective tools for a wide range of applications in medicine and oncology, including computer-aided detection and diagnosis,[1,2] image segmentation[3,4] knowledge-based planning,[5–7] quality assurance,[8,9] radiomics feature extraction,[10,11] and outcomes modeling.[12–16] Numerous studies, as summarized in Fig. 1, demonstrate the potential application of ML and DL models to clinical problems combined with the ongoing efforts toward ushering radiation oncology into the era of Big data analytics[17–21] and serve as evidence that ML and DL are poised to revolutionize the fields of medical physics and radiation oncology. Given the unique role of the medical physicist as a bridge between the clinical team and clinical technology and as a driving force

toward developing new technological innovations in medicine, it stands to reason that medical physicists are the most appropriate member of the clinical team to lead this AI-driven digital revolution for the medical community.

There has been several reviews of AI/ML/DL in the medical literature, introducing its basic concepts and potential roles,[22–27] but with little focus on the personnel that will be tasked to lead its implementation in the field. Therefore, the main goal of this review article is to equip medical physicists (who may have little to no prior experience with ML/DL techniques) with the basic foundational knowledge and examples necessary to develop and analyze models for application to a broad range of problems in medical physics and radiation oncology. While this publication is by no means a comprehensive cookbook of ML algorithms, it should serve as useful resource to help novices answer the question: "where should I start with my AI task?," in regards to what ML/DL models they should select to answer specific research questions in medical physics and what data processing and model validation best practices are recommended to ensure robust results. In addition, this article provides an overview of the underlying practical and ethical issues pertaining to ML/DL applications in radiation oncology. The rest of this tutorial/review is organized as follows: Section 2 provides a general overview of ML and DL algorithms. Section 3 provides a description of data requirements for ML/DL models, such as sample size and necessary pre-processing steps to ensure that the models will perform as intended. Section 4 provides an overview of classical ML techniques, while in Section 5 DL methods are presented. Section 6 discusses how to validate model performance on new and out-of-sample datasets. Section 7 describes the role of humans in the development and use of ML and DL models, while Section 8 discusses the known limitations and pitfalls of these methods. Finally, Section 9 envisions the role of the medical physicist in this new era of *AI-guided radiation oncology* clinics.

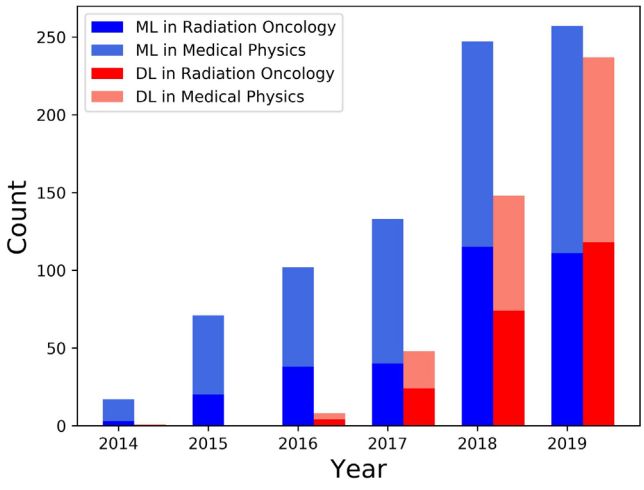

FIG. 1. Frequency counts of published PubMed studies in radiation oncology/medical physics by machine and deep learning.

2. WHAT ARE MACHINE AND DEEP LEARNING?

Machine learning[28] is the scientific study that builds mathematical models and computer algorithms to perform specific tasks by learning patterns and inferences from data using computers, without being explicitly programmed to conduct these tasks. ML algorithms differ in (a) their types of input/output data and (b) the types of problems they are intended to solve. They can be divided accordingly into three categories:[29] supervised learning, unsupervised learning, and reinforcement learning (RL). Table I provides an visual overview of the main categories of ML and their common applications in medical physics and radiation oncology.

Supervised learning[30] requires a labeled dataset, as it is intended to learn the relationship between input variables and outputs (labels). For instance, prediction of radiotherapy outcomes (e.g., tumor control or normal tissue toxicity)[31] is a supervised learning task. First, one collects relevant patient information (e.g., dosimetric information and clinical variables) together with the treatment outcomes (labels). Then, a supervised learning algorithm is applied to learn the mapping from this patient information to the labeled outcomes. Once the model is obtained, it can be adopted to predict response

TABLE I. Common taxonomy for machine learning and deep learning algorithms

Classification of ML and DL algorithms		
Learning style	Definition	Radiation oncology applications
Supervised	Learns relationship between input data and labels	Diagnosis,[1,2] image segmentation,[3,4] radiotherapy outcomes prediction[12–16]
		Requires labeled dataset
Unsupervised	Learns patterns in input data	Anomaly detection for QA,[8,9] radiomics feature extraction[10,11]
		Uses unlabeled dataset
		Used for clustering or data reduction
Reinforcement	Learns to perform actions in response to environment to maximize a reward function	Decision-making in adaptive radiotherapy[19]
Data interaction	**Definition**	**Examples**
Classical ML	An ML algorithm that would require manual feature extraction and selection to perform its task	Supervised: GLM,[75] SVMs,[33] RFs[34]
		Unsupervised: k-means clustering, PCA,[39] t-SNE[40]
		Reinforcement: Q-learning
DL	Can learn feature representation from raw input data and perform learning tasks	Supervised: U-NET[96]
		Unsupervised: VAE,[57] GAN[58]
		Reinforcement: DQN[45]
		Nonspecific: MLP, CNN,[47] RNN[48]

to treatment and be applied to new patients in order to personalize their prescription. Examples of typical supervised learning algorithms are logistic regressions,[32] support vector machines (SVMs),[33] random forests (RFs),[34] and neural networks (NNs).[35]

Unsupervised learning[36] operates on an input dataset without the need for labels. Its goal is to try to draw inferences and identify patterns within the unlabeled data space, for the purposes of clustering or data reduction. For instance, clustering[37] is a typical task of unsupervised learning, which can be applied for quality assurance (QA) in radiotherapy.[38] In this case, one can distinguish outliers or unacceptable treatment plans from acceptable ones by applying the clustering algorithm to the feature set. Other typical unsupervised learning tasks include dimensionality reduction such as principal component analysis (PCA),[39] t-SNE,[40] and autoencoders.[41] These can be used for visualization of complex data in higher dimensions or applied before supervised learning, as a way of learning a more compact data representation for solving complex supervised learning problems.

Reinforcement learning[42] is a ML extension of classical decision-making schemes, Markov Decision Processes (MDPs).[43] It is concerned with how software agents can take actions when interacting with a given environment. Usually the agent needs to achieve a definite goal via maximizing a cumulative reward function,[42] for example, therapeutic index in radiotherapy. The famous Google AlphaGO[44] is an RL application, where an agent learns how to take actions under different situations to win a board game. In radiotherapy, RL can be applied to adaptive treatment planning, for example, how to optimize prescriptions for patients by learning from during treatment information.[45] In this case, an agent will learn how to adapt dose fractionation (action) based on the current condition of the patient undergoing radiotherapy (environment) to achieve the goal (reward) of better treatment response.

Deep learning,[46] which recently demonstrated tremendous success in image recognition problems[47] and natural language processing,[48] is a subcategory of the broad family of ML algorithms. It is generally based on NN architectures,[35] using multiple layers to gradually extract higher level features from the raw inputs; eliminating the necessary and typically problematic feature engineering process[49] in classical ML, and hence showing superior performances. This is a key advancement in multivariable and statistical prediction modeling, where data representation and task learning can be effectively achieved in the same framework. Training a deep NN (DNN) was challenging before and during the 1990s to 2000s due to limitations in computational capacity and lack of robust optimization techniques. The modern framework of DNNs originated from earlier work on restricted Boltzmann machines (RBMs, 1985),[50] deep belief networks (DBNs, 2006),[51] and later the efficient improvement in activation functions using rectified linear unit (ReLU, 2010),[52,53] among others. Certainly, the huge leap in computing power using graphics processing units (GPUs) and advancement in optimization techniques[54] were also critical. Today, a residual network (ResNet[55]) is among the deepest network that can be trained. With its 152 layers, it won the championship of the ILSVRC 2015 classification competition with a top-5 error rate of 3.57%, only rivaling that of human cognitive ability.

Some of the most common architectures of DL include convolutional NNs (CNNs),[47] recurrent NNs (RNNs),[56] variational autoencoders (VAEs),[57] and generative adversarial NNs (GANs)[58] as shown in Figs. 11–13, 15. Convolutional neural network are typically designed for image recognition and computer vision applications. They largely reduce the number of free parameters compared to standard fully connected NNs. They have shown competitive results in medical imaging analysis, including cancer cell classification, lesion detection,[59] organ segmentation[60] and image enhancement. RNNs are usually applied for natural language processing (NLP) and audio recognition problems, as they can exhibit temporal dynamic behavior that can be exploited for sequential data analysis.[56] This property also makes RNNs valuable for aiding fractionated radiotherapy, effectively taking advantage of a variety of previously unused temporal information generated during the treatment course. A VAE is an unsupervised learning algorithm that is able to learn the distribution of compressed data representations from a high-dimension dataset. In another words, it is the equivalent of PCA analysis but for DL applications. It can be widely applied in radiation oncology considering the prevalence of high-dimension data due to the limitation of patient sample sizes.[14] Similar to a VAE, a GAN is also a generative model[61] that can learn the multivariate distribution and describe how the data are generated. GANs learn the distribution by an adversarial competition between its generator and its discriminator. They have been successfully applied in some medical imaging tasks, mapping magnetic resonance imaging (MRI) into computed tomography (CT) images (synthetic CT)[62] or in adaptive radiotherapy[45] for generating synthetic data and enriching the sample size.

3. WHAT DATA ARE NEEDED FOR ML/DL APPLICATIONS?

3.A. What training sample size is required?

When building machine or DL models for solving practical medical problems, one should first consider how much data is needed for successful training, that is, not under- or overfitting the data. Often, the answer can be complicated as there are no off-the-shelf recipes for ML/DL algorithms as compared to traditional power analysis in statistics. Instead, one needs to examine the specific problem/learning algorithm and perform some simulation experimentation using so called *Learning Curves (LC)* on the existing data to determine whether there is a sufficient sample to meet the training requirements of the ML/DL algorithm at hand.

Empirically, the more complex the problem/learning algorithm (e.g., larger number of free parameters), the more data will be naturally required. A simple linear model with two unknown parameters will only require two "perfect" samples to fit, while a complicated nonlinear modern DL architecture may need thousands of data points to train. Applying a small

dataset to train a complex algorithm can be problematic, as it may lead to overfitting pitfalls,[63] where the complex algorithm starts to fit noise or errors in the limited-size training set, in other words, the algorithm memorizes the data rather than learns from it. Under this circumstance, generalization[64] of the model is usually not good, that is, the model performs poorly on new, unseen out-of-sample datasets. Mathematically, we can understand this as a trade-off between variance and bias, colloquially referred to in the ML literature as the *bias-variance dilemma*.[65] Specifically, suppose f describes the underlying real relationship of the data; $\hat{f}$ is the model approximation that being trained on a certain dataset. The amount of $\hat{f}$ that changes as training sets vary is called variance. The difference between f and the model $\hat{f}$ is defined as bias. It is proven that, in a unseen test dataset, both bias and variance add to the total errors of model, moreover, there is a trade-off between the two as in Fig. 2. As the model complexity (e.g., the number of free parameters) increases, the bias (i.e., the training fitting error) decreases, but the variance (i.e., the testing generalizability error on out-of-sample data) increases. The optimal trade-off point between bias and variance or training and testing can be quantified using so called the Vapnik–Chervonenkis (VC) dimension,[66] however, this is still currently a theoretical rather than a practical measure of model complexity.

A learning curve (LC)[67] is a practical graphical tool that can be used to evaluate whether there are enough data empirically. Note that there exist other versions of learning curves for different purposes, for example, determining the training epochs (the number of times that the learning algorithm work through the entire training dataset). However, the idea behind those learning curves is the same; splitting the dataset into training, validation, and testing. Then, one can plot the model performance metric for training/validation separately as a function of the number of the samples to determine a sufficient number of samples for training the ML/DL algorithm before evaluating its generalizability on the testing data. As shown in Fig. 3, when there is a significant change in the performance error for training or validation, it may indicate a larger sample size may be required until they both plateau.

To unlock the usage of more sophisticated models, it is always a good idea to have more training/validation data as long as they are not too noisy. However, if collecting more data becomes infeasible, one can alternatively perform data augmentation or apply transfer learning approaches. *Data augmentation*[68] is an effective way to increase the data size and diversity for the training model by cropping, padding, shifting, flipping, and rotation, which are widely applied in DNNs to support imaging tasks. *Transfer learning*[69] is another important tool in ML/DL to solve the problem of insufficient training data by trying to transfer knowledge from a source domain to the target domain. This approach has been successfully applied to medical image segmentation problems by transferring knowledge from natural image applications (e.g., Google ImageNet database).[70] This idea can be extended to other tasks, where Zhen et al. demonstrated a CNN for predicting rectal toxicity in cervical cancer radiotherapy by fine-tuning a pretrained network (VGG-16) on the natural images from ImageNet.[71]

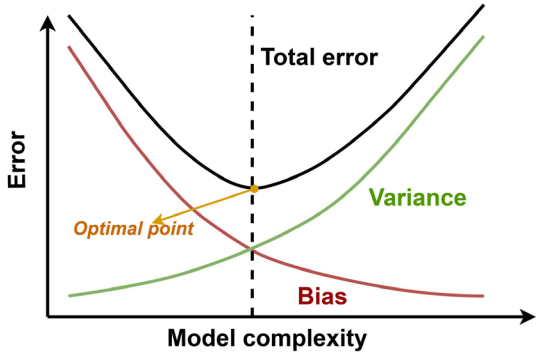

FIG. 2. The trade-off between bias and variance with model complexity.

FIG. 3. The learning curve example for diagnostic purpose of data size.

3.B. How to process the data?

Data are the fuel of ML/DL algorithms, where models are the combustion engines. Only the combination of a good engine and good fuel can bring out the most powerful performance, where the case of "garbage in garbage out" is the least desirable result. Thus, it is important to have high quality data that are properly curated and processed for successful ML applications. At face value, this may seem to contradict the current notion of Big data analytics. However, this is accounted for by the fact that larger sized datasets, though noisy, will benefit from variance reduction by virtue of the law of large numbers.

Depending on the experimental designer's objectives and choices, tabular, text, or imaging data can be represented as vectors, matrices Eq. (1) or even higher-rank tensors.

$$\mathbf{x}_i = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n, \mathbf{x}_i; = \begin{matrix} x_{11} & x_{12} & x_{13} & \ldots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \ldots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{d1} & x_{d2} & x_{d3} & \ldots & x_{dn} \end{matrix} \in \mathbb{R}^{\mathbb{d}\times\mathbb{n}} \quad \begin{matrix} \text{(vector)} \\ \\ \\ \\ \text{(matrix)} \end{matrix} \tag{1}$$

For example, two-dimensional (2D) images are usually presented by a matrix when fed into a CNN, but are usually represented by a long vector when serving as an input to SVMs

or fully connected NNs. This causes a loss of spatial information, which is another reason for the superior performance of CNNs in imaging tasks. Vectors and matrices are only two special cases of tensors (rank 1 and 2, respectively), where a general rank k tensor T is defined as a k-linear functional from a vector space V, $T : \underbrace{V \times \cdots \times V}_{(k\text{times})} \rightarrow \mathbb{R}$. The choice of representation is problem-dependent and determined by the complexity of problem and the size of available data. For instance, a video or a three-dimensional (3D) volume medical image such a CT or an MRI can be presented by a tensor component, if the spatial arrangement is necessary for learning the task at hand (e.g., detection or classification), then T_{ijk} with $(i,j,k)=(1 \sim dimx, 1 \sim dimy, 1 \sim dimz)$, see Fig. 4. In supervised learning, the collection of input data along with ground truths (labels) are denoted by: $\mathcal{X} = \{(\mathbf{x}^{(i)}, \hat{\mathbf{y}}^{(i)}) | \mathbf{x}^{(i)} \in \mathbb{R}^{k_1 \times k_2 \times \cdots}, \hat{y}^{(i)} \in \mathbb{R}^l, i = 1, \ldots, n\}$. In unsupervised learning setting, the data are represented by $\mathcal{X} = \{(\mathbf{x}^{(i)}) | \mathbf{x}^{(i)} \in \mathbb{R}^{k_1 \times k_2 \times \cdots} i = 1, \ldots, n\}$, as no label information is required. When the dataset contains features ($\mathbf{x}$) highly varying in magnitudes and unit, it is important to standardize the feature values,[72] otherwise, model numerical stability and estimation precision may be degraded. Some common standardization methods include *Z-score normalization* and *Min-Max scaling* as in Eq. (2):

$$\begin{aligned} \text{Z-score} \quad & x_{\text{new}} = \frac{x - \mu(x)}{\text{std}(x)} \\ \text{Min-Max} \quad & x_{\text{new}} = \frac{x - \min(x)}{\max(x) - \min(x)} \end{aligned} \tag{2}$$

It should be emphasized that standardization techniques are especially beneficial[73] for learning by DNNs, which is generally a challenging task. During the training, standardization of input data will help the optimizer to find good local minimum more easily and faster by mitigating numerical instability errors. Another big concern when applying ML/DL to medical data is how to deal with missing data. Of course, one can always drop the cases with incomplete information, but it would further reduce the sample size which may already be small. To fully make use of the available data, *imputation* methods,[74] that is, replacing missing values with statistical estimates, are usually applied before analyzing the full dataset. Some simple imputation methods include mean or median imputation. More sophisticated imputation methods also exist based on maximum likelihood estimation (MLE), for instance. However, one should keep in mind, that imputation is also dependent on the quality of observed values, therefore, imputation should be applied with caution, as the imputed values may be inaccurate and noisy. Alternatively, one may consider data augmentation or transfer learning when dealing with insufficient training data, as discussed earlier.

4. WHAT CLASSICAL MODELS EXIST?

Aside from modern DL methods, classical machine learning algorithms such as SVMs, RFs, Naive Bayes, K-nearest neighbor, and so forth are also worth considering when designing applications for radiotherapy. Due to space limitation, the following section will mainly focus on linear models and generalized linear models (GLMs), as they are the foundations of current DL models. Other common classical machine learning methods, including SVMs and RFs will be briefly reviewed in the section that follows.

4.A. Linear models and generalized linear model (GLM)

A linear model $f_w(\cdot)$ considered to be linear in unknown parameter w ($w \in \mathbb{R}^m$), has the form,

$$\mathbf{y} = f_w(\mathbf{x}) = w^T x \tag{3}$$

Note that a linear model is not necessarily a linear function of predictors $x (x \in \mathbb{R}^m)$. A model which has polynomial and interaction terms as in Eq. (4) is also considered as a "generalized" linear model,

$$\mathbf{y} = f_w(\mathbf{x}) = w_0 + w_1 x_1^2 + w_2 x_1 x_2 + w_3 x_2^2 \tag{4}$$

as one can simply re-define x in a way that the model satisfies the general form in Eq. (3). A linear model uses straight lines or linear planes to fit data (see Fig. 5). The fitting process usually involves calculating the optimal estimator $\hat{\beta}$ by minimizing a designated loss function, for example, squared error or likelihood function,

$$\hat{w} = \arg\min \sum_{i=1}^{n} (y^{(i)} - (x^{(i)})^T w)^2 \tag{5}$$

where n is the sample size and i is the index of sample. The best estimator, $\hat{w}$, from Eq. (5) is called least squared estimator which has a closed form,

$$\hat{w} = (X^T X)^{-1} X^T Y \tag{6}$$

where $X = (x^{(1)}, x^{(2)}, \ldots, x^{(n)})^T (X \in \mathbb{R}^{n \times m})$ is called a design matrix and $Y = (y^{(1)}, y^{(2)}, \ldots, y^{(n)})^T$ $(Y \in \mathbb{R}^n)$ is a vector of labels.

A linear model is concise and easy to implement, however, it assumes Y is normally distributed, that is, $Y|X \sim N(0, \sigma^2)$ and there is an identity mapping [Eq. (8)] between Y and Xw, which is usually not the case in practice. To alleviate these issues, a generalized linear model (GLM)[75] was proposed.

A GLM is defined by three components: a random component, which specifies the distribution of Y given X ($Y|X$); a systematic component, which is a linear part relating a parameter η to a predictor X (i.e., $\eta = Xw$); and a link function that connects the expected value of Y to η. In GLMs, the distribution of $Y|X$ typically belongs to the exponential family, for example, normal, exponential, Bernoulli and categorical distribution. Some commonly used link functions include identify, negative inverse, log, and Logit.

Logistic regression[32] which is used for binary classification is a special case of GLM that has Bernoulli distribution and Logit link function, that is,

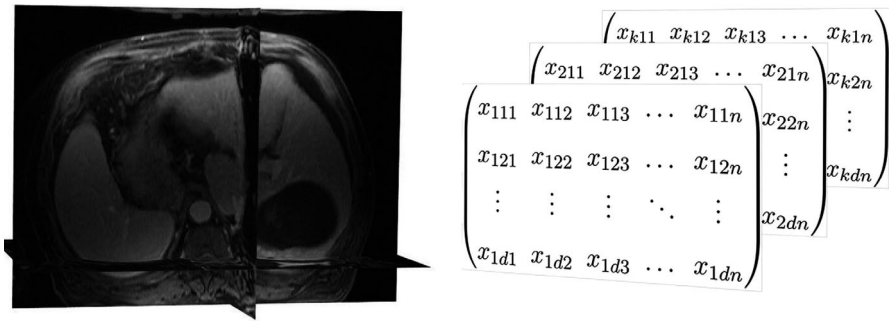

FIG. 4. An magnetic resonance image [left] (along slice planes) is usually represented by a rank three tensor (component) [right].

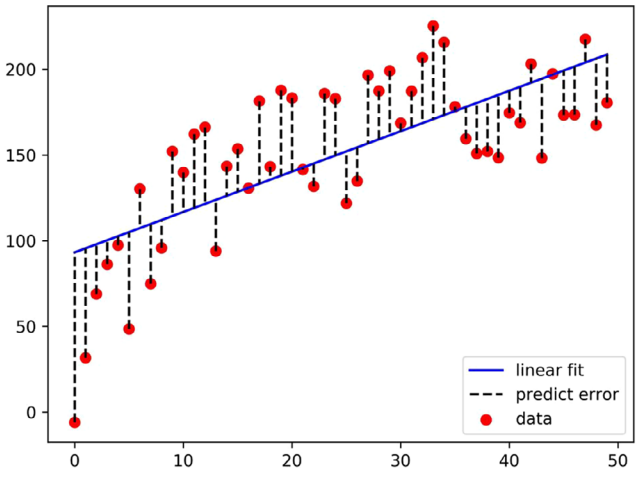

FIG. 5. Geometry intuition of linear regression by Eq. (5), where one wishes to find a straight line fit such that the sum of all residual errors is smallest.

$$\eta = Xw = \log\frac{\pi}{1-\pi}, Y|X \sim \text{Bernoulli}(\pi) \tag{7}$$

One can rewrite the above to arrive at $\pi = \frac{1}{1+\exp(-Xw)}$, where π is the predicted probability of Y=1. Note that π=sigmoid(Xw), see Eq. (8); hence, the logistic model can be seen as a simple NN (in Section 5.A) with only two layers (input/output) and a sigmoid activation function (Fig. 6),

$$\sigma(z_1,z_2,\ldots,z_m)\begin{cases}\text{identity}=(z_1,\ldots,z_m) & (\mathbb{R}^m\text{-regression})\\ \text{sigmoid}=\frac{1}{1+e^{-z_m}} & (m=1\text{ binary classification})\\ \text{softmax}=\left(\frac{e^{z_1}}{\sum_{j=1}^{m}e^{z_j}},\ldots,\frac{e^{z_m}}{\sum_{j=1}^{m}e^{z_j}}\right) & (m>1\text{ classification})\end{cases} \tag{8}$$

where m is the dimension of output. A GLM model is usually optimized via MLE techniques. For logistic regression, this is equivalent to minimizing a binary cross-entropy loss:

$$\mathcal{L}(w) = -\frac{1}{N}\sum_{i=1}^{N} y^{(i)}\log\pi^{(i)} \tag{9}$$

In NNs, cross-entropy is also usually applied as a loss function for the classification problem. This is due to its favorable numerical and theoretical properties, as will be further explained below.

4.B. Model regularizations

Often one wishes to suppress overfitting of a model when feeding data corrupted with noise by adding a penalty term: h ($\mathbf{w}$) into the loss functions $\mathcal{L}(\mathbf{w})$ such as in Eq. (9)

$$\tilde{\mathcal{L}}(\mathbf{w}) = \mathcal{L}(\mathbf{w}) + h(\mathbf{w}) \tag{10}$$

where $h(\mathbf{w})$ only depends on the model $\mathbf{w}$ (not data) for regularization.[76] Typical choices are:

$$\begin{aligned}&(\text{Elastic})h(\mathbf{w}) = \lambda_1\|\mathbf{w}\|_{L_1} + \lambda_2\|\mathbf{w}\|_{L_2}^2\\ &(\text{Ridge})\ h(\mathbf{w}) = \lambda_2\|\mathbf{w}\|_{L_2}^2 \quad (\text{i.e., } \lambda_1 = 0)\\ &(\text{LASSO})h(\mathbf{w}) = \lambda_1\|\mathbf{w}\|_{L_1} \quad (\text{i.e., } \lambda_2 = 0)\end{aligned} \tag{11}$$

In fact, this trick can be widely played in other ML/DL techniques (e.g. SVM, DNNs) leading to better solutions. This is particularly true, if the problem is ill-posed as is commonly the case in many ML/DL applications in medical physics, where small errors in the training may lead to large variations in the estimated ML/DL model.

4.C. Nonlinear classical machine learning

There is a whole host of nonlinear machine learning algorithms that have been applied to medical physics/radiotherapy applications.[38] Two of the most common ones are briefly reviewed here: SVMs and RFs. *SVMs* represent data samples as points in space, mapped (ϕ) in a way that samples from the different classes can be separated by a margin (gap) that needs to be made as wide as possible, that is, maximized in higher dimensional space, a trick known as the kernel mapping. In practice, it is usually not feasible to completely separate samples, particularly when the data are noisy; hence, some tolerance errors are allowed. A SVM is inherently a binary classifier, which has a hinge loss function. To make the optimization process numerically easier, the non-convex primal problem is usually converted into a convex dual problem defined as follows:

$$\begin{aligned}&\max_{\alpha_i\geq 0}\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n}\alpha_j\alpha_k y_k y_j K(x_j,x_k)\\ &0\leq\alpha_i\leq C,\forall i\\ &\sum_{i=1}^{n}\alpha_i y_i = 0\end{aligned} \tag{12}$$

TABLE II. A snippet from a dataset of 569 breast cancer diagnosis with 30 image extracted features

id	diagnosis	radius_mean	texture_mean	perimeter_mean	...	fractal_dimension_worst
842302	M	17.99	10.38	122.8	...	0.1189
842517	M	20.5	17.77	132.9	...	0.08903
84300903	M	19.69	21.25	130	...	0.08758
84358301	M	11.42	20.38	77.58	...	0.173
84348402	M	20.29	14.34	135.1	...	0.07678
8510426	B	13.54	14.36	87.46	...	0.07259

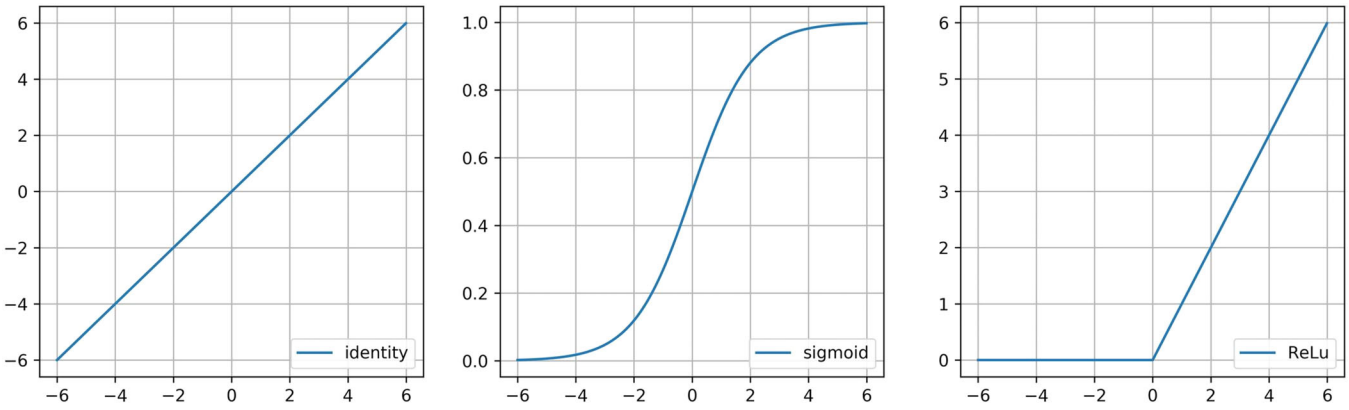

FIG. 6. Activation functions: [Left]$\sigma(\mathbf{z})$=$\mathbf{z}$ for regression; [Middle]$\sigma(\mathbf{z}) = \frac{1}{1+e^{-\mathbf{z}}}$ for classification. [Right]$\sigma(\mathbf{z})$=max(0,$\mathbf{x}$) for deep learning.

where $K(x_j, x_k) = \phi(x_j)^T \phi(x_k)$ is known as a kernel, which is an inner product of feature maps ϕ and acts as a cross-similarity metric in the feature (Hilbert) space. Various types of kernels exists, such as linear, polynomial, and radial basis function (RBF) depending on the desired data support. For instance, polynomials have finite data support while RBFs have infinite data support and is thus commonly used despite being more computationally expensive. Penalty terms such as ridge loss are usually added to the loss function for regularization purposes and for preventing overfitting pitfalls. Support vector machines have been very popular ML techniques due to their global optimal solutions for classification and regression are widely applied to radiation oncology problems. However, as a classical ML approach, they require features to be extracted and selected prior to training.

A *RF* uses an ensemble of decision trees to solve classification or regression problems. A decision tree has a flowchart-like structure where each node represents a test on attributes, splitting examples into different branches. From root nodes to leaf nodes, the decision is gradually made by multilevel classification rules, which make them quite easy to interpret and desirable to use but with limited predictive power. Hence, the corresponding ensemble approach RFs, that is, combining multiple weak classifiers (decision trees) to achieve a stronger classification, is usually applied instead. The splitting at nodes is usually based on a Gini Index, entropy function, or information gain. The ensemble is based on so called bagging, that is, averaging. Recently, a gradient boosting, weighted averaging approach for RFs has been proposed with improved performance.[77] As an ensemble method, RF usually reduces variance and improves generalization compared to a single decision tree but with the caveat of reduced interpretability.

4.D. Example implementations

As a demonstrative implementation example, a logistic regression model application to breast cancer diagnosis is presented here. The dataset (Table II), which contains 569 breast cancer diagnosis cases, was created in 1995 by researchers at the University of Wisconsin. Each case includes 30 features computed from a digitized image of a fine needle aspirate (FNA) of a breast mass and a binary diagnosis label with "M" indicating "malignant" and "B" indicating "benign."

Let the mean-value features (first 10 in all 30) plus a feature $x_{11} = 1$ (serving as intercept term) be the predictor parameters in the trained logistic model. Using $\pi = \frac{1}{1+\exp(-Xw)}$ as in Eq. (7), the logistic regression model will take the design matrix $X(X \in \mathbb{R}^{569\times 11})$ as input and output prediction π which is the probabilities of diagnosis $Y = 1$ (malignant). The associated Python code for the reader reference can be found in https://github.com/sunancui/breast-cancer-diagnosis. As mentioned, logistic regression can be regarded as a simple NN with only a single input/output layer and a sigmoid activation function. We also compared logistic regression and NN in the code from this perspective; one would notice that they yield similar estimation of prediction

parameters (w). A summary of the prediction results of breast cancer diagnosis by logistic regression is presented in Fig. 7 with the ground truth labels shown for comparison.

5. WHAT DEEP LEARNING MODELS EXIST?

5.A. Neural networks

Neural networks in principle[35] can be considered as natural generalizations of linear models (Section 4.A). If we modify Eq. (3) by adding a so-called activation function σ [Eq. (8)],

$$\mathbf{x}^{(j+1)} = \sigma^{(j)}\left(\mathbf{w}^{(j)} \cdot \mathbf{x}^{(j)} + \mathbf{b}^{(j)}\right) \tag{13}$$

j=0,1,. . ., and let it iterate recursively over itself L times, with the upper-right index (0),(1),. . .,(L) (Fig. 8), one then has a composite function

$$\begin{aligned} f_{\mathbf{w}}(\mathbf{x}) &= \mathbf{x}^{(L)} \\ &= \sigma^{(L-1)}\left(\mathbf{w}^{(L-1)} \cdot \sigma^{(L-2)}\left(\mathbf{w}^{(L-2)} \cdots + \mathbf{b}^{(L-2)}\right) + \mathbf{b}^{(L-1)}\right) \end{aligned} \tag{14}$$

This is known as a NN, where now the number of structures is renamed as the number of layers (Fig. 9). In particular, the first layer (0) and the last layer (L) are called the *input layer* and *output layer*, respectively. Any layer (j) with $0 < j < L$ is called a *hidden layer*. Moreover, it is in general referred to as a DNN if $L > 4$ (i.e., more than two hidden layers), which is the fundamental building block for DL. Hence, conceptually, a NN is merely an extension of a linear model. Thus the concept of activation functions in Eq. (8) and loss functions (9) discussed in Section 4.A can be applied to DL without any changes. This is known as a NN, where now the number of structures is renamed as the number of layers. In particular, the first layer (0) and the last layer (L) are called the *input layer* and *output layer*, respectively. Any layer (j) with $0 < j < L$ is called a *hidden layer*. Moreover, it is in general referred to as a DNN if $L > 4$ (i.e., more than two hidden layers), which is the fundamental building block for DL. Hence, conceptually a NN is merely an extension of a linear model. Thus the concept of activation functions in Eq. (8) and loss functions (9) discussed in Section 4.A can be applied to DL without any changes.

However, the nonlinear activation function now plays a prominent role in (deep) NNs as it is the source of nonlinearity, that is, mapping the data into higher dimensions. If the activation function between any two layers, Eq. (13), is an identity map in Eq. (8), then the two layers can be merged since

$$\begin{aligned} \mathbf{x}^{(j+1)} &= \sigma^{(j)}\left(\mathbf{w}^{(j)} \cdot \left(\mathbf{w}^{(j-1)} \cdot \mathbf{x}^{(j-1)} + \mathbf{b}^{(j-1)}\right) + \mathbf{b}^{(j)}\right) \\ &= \sigma^{(j)}\left(\tilde{\mathbf{w}}^{(j)} \cdot \mathbf{x}^{(j-1)} + \tilde{\mathbf{b}}^{(j)}\right) \end{aligned} \tag{15}$$

with $\tilde{\mathbf{w}}^{(j)} = \mathbf{w}^{(j)} \cdot \mathbf{w}^{(j-1)}$ and $\tilde{\mathbf{b}}^{(j)} = \mathbf{w}^{(j)} \cdot \mathbf{b}^{(j-1)} + \mathbf{b}^{(j)}$. Thus, one sees clearly from above discussion that the node $\mathbf{x}^{(j-1)}$ directly connects to node $\mathbf{x}^{(j+1)}$ via $\left(\tilde{\mathbf{w}}^{(j)}, \tilde{\mathbf{b}}^{(j)}\right)$ as if middle layer (j) vanishes. In an extreme case where all activations are identities in Eq. (14), such a NN simply reduces to a linear model Eq. (3) no matter how many layers (deep) it has. Therefore, this argument provides an insight why activation functions are the primary source of nonlinearity of NNs. It is thought that with each nonlinear activation, an additional fold (manifold) in the data can be achieved allowing for better representation or capturing of highly complex relationships. As for NNs' hyperparameters, for example, number of layers L and number of nodes in a specific layer, Bayesian methods[78] can be used to optimize their number. However, most of the current procedures still rely on trial and error schemes.

5.B. Why and how to go deeper?

The Universal Approximation Theorem (UAT) developed by Hornik[79] states that mathematically a NN with one hidden layer of sufficient nodes can approximate any measurable (and hence continuous) function on compact sets under certain mild conditions on the activation functions. This fact explains why NNs are suitable for fitting complex functions and datasets. However, based on this, one probably wonders why we would bother going deeper? In fact, the theorem has several constraints. First, we need to have a "sufficient" (can be infinite) number of nodes. Secondly, it does not guarantee the theoretical performance can be achieved through optimization in practice due to local minima and convergence issues. Thus, it still depends on experimentally designing the right architecture (e.g., activation function, regularization, number and size of layers, etc.) and adopting an appropriate training process (e.g., optimization method) in order to possibly achieve the theoretical performance estimates.[80]

Practically, adding more layers to a NN has been shown to provide a good architecture design vs increasing the number of nodes as suggested by UAT. A NN with more layers will show better performance than a single layer NN that has the same number of parameters. Intuitively, this is possibly because each layer will transform its input, that is, folding, and creating a new representation of the data (appropriate manifold). The multilevel abstraction that is being learned through multiple layers can be hard coded into a single layer with the same number of nodes. Or formally speaking, the multilayer structures enable NNs to recognize the entangled manifolds of the data more easily, so as to solve the designated task.[81]

However, adding too many layers, the performance of a NN can actually be degraded as well. The phenomenon was identified as the *vanishing gradient problem* by Hochreiter's Ph.D thesis[82] and has been a major obstacle of DL studies for a long time. Vanishing gradient is a problem occurring during optimizing NN weights[83] with gradient-based learning methods, where the gradient will become too small through multiplication of many layers, effectively preventing the weight from changing its value. In the modern deep architectures, there exist several mechanisms to prevent such problems, for example, residual blocks, ReLU activation,[52] batch normalization.[84] Hence, they help the architectures grow

deeper and more powerful without encountering such issue. For instance, the residual NN architecture (ResNet)[55] adds one extra term to the usual NN Eq. (13),

$$\mathbf{x}^{(j+1)} = \sigma^{(j)}\left(\mathbf{w}^{(j)} \cdot \mathbf{x}^{(j)} + \mathbf{b}^{(j)} + \underbrace{\mathbf{x}^{(j-1)}}_{\text{skipconnection}}\right) \quad \text{(ResNet)}, \tag{16}$$

where the additional term is called the *skip connection*. Heuristically, it adds an extra path or shortcut to another neuron that was not connected previously. During the back-propagation process, the network will skip some subnetworks, directly forwarding the gradient from higher to lower layers, eliminating the gradient vanishing problem. The new activation functions ReLU in Fig. 6[Right] and its variants[85] eliminate vanishing gradients by avoiding squishing a large input space between 0 and 1 as in Sigmoid activation, thus preventing extremely small gradients from occurring at the edge. Using batch normalization layers between other layers to normalize the intermediate inputs is another solution. This ensures the values of intermediate inputs are inside the range that has the effective gradients, hence, the gradient values will not become too small.

The next question that comes to mind when designing an architecture is how to guarantee good generalization. As the model becomes extremely complex, overfitting can be a main concern. Intuitively, regularization techniques can resolve ill-posed problems by suppressing the noise in the training data. Besides adding a weight penalty as shown in Section 4.B, one can adopt the *dropout*, another neuroscience inspired trick,[86] as a regularization technique. As shown in Fig. 10, during the training, dropout will randomly select some portion (dropout rate, e.g., 20%–50%) of nodes being ignored. They will not affect updated weights, as their contribution to the activation of downstream neurons is temporally removed. Indeed, dropout is currently a very effective ensemble method, performing averaging with NNs while mitigating the risk of memorizing the data. Hence, the resulting NN with many layers (DNN) is capable of better generalization to unseen data and is less likely to overfit (memorize) the training data.

More advanced optimization techniques will also be required when training a "deeper" NN. As compared with the simple case of logistic regression, whose loss function is convex, the deeper NN will tend to have a more sophisticated loss landscape, making finding even the close-to-global optimal solution more difficult.[80] Many efforts have been made to develop effective optimization algorithms[87] for such large-scale nonlinear problems primarily based on gradient descent techniques, including the use of stochastic approaches with momentum, which accelerate learning by increasing the gradient vectors in the direction that past gradients accumulated (velocity). In the adaptive rate scheme, the algorithm will adopt various learning rates per parameter according to their history of momenta and gradients. The common optimization techniques including Adam,[54], RMSProp[87] are popular choices in DL studies as surrogates for the classical stochastic gradient descent techniques.

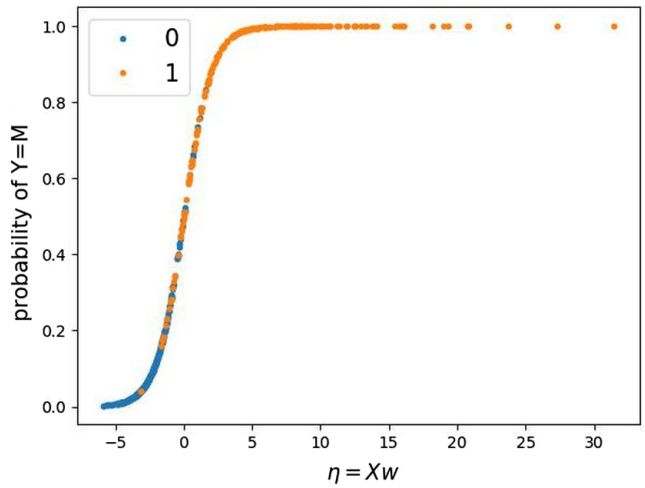

FIG. 7. Plot of fitting of breast cancer diagnosis by logistic regression, cases with malignant (Y = 1), and benign (Y = 0) diagnosis were denoted.

5.C. Prevalent architectures

5.C.1. Convolutional neural networks

CNNs[47] are best known of late for image recognition and image-related predictions, which borrow the concept of convolution from classical linear system filtering, where a 2D image $\mathcal{I} : \mathbb{R}^2 \to \mathcal{C}$ is convolved with a given kernel function $\mathbf{w} : \mathbb{R}^2 \to \mathbb{R}$ such that the output image is of the form,

$$\tilde{\mathcal{I}}(\mathbf{x}) = \int_{\mathbb{R}^2} \mathbf{w}(\mathbf{x}-\mathbf{y})\mathcal{I}(\mathbf{y})d\mathbf{y} \qquad \text{(Fourier convolution)} \tag{17}$$

Therefore, the concept of convolution is naturally blended into NNs to develop the so-called CNN when image-like data are concerned.

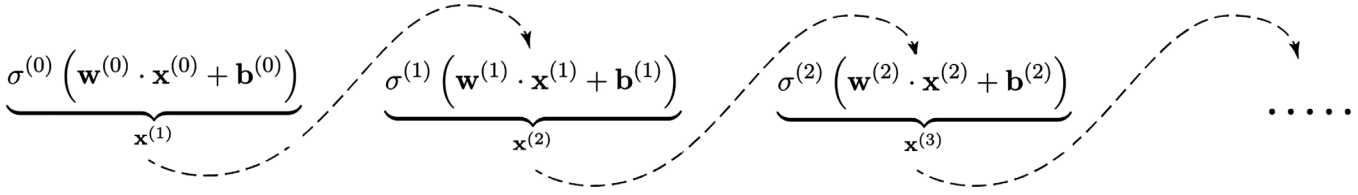

FIG. 8. Self-iterations of linear transformations.

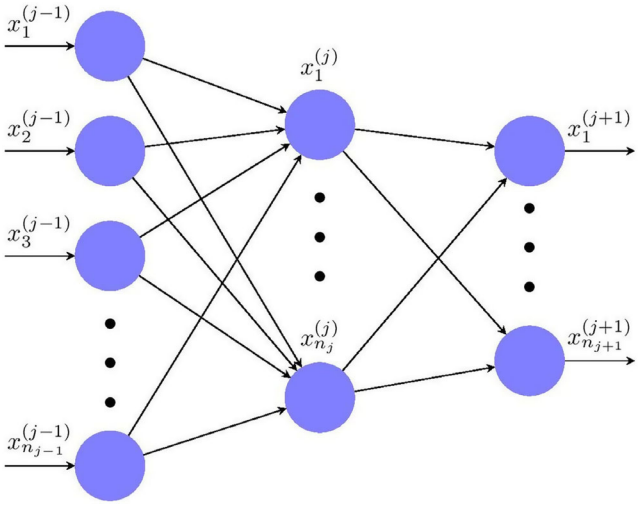

FIG. 9. A graphic plot of Fig. 8 and Eq. (14).

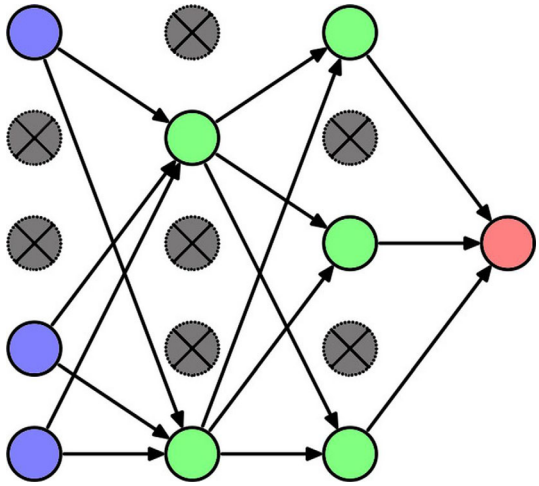

FIG. 10. Illustration of dropout techniques.

A CNN typically consists of several convolutional layers (filtering), pooling layers (down-sampling aimed for reducing dimensionality), and activation functions, where the convolution layer is the core component that operates convolution on an image field to capture its relevant features and contribute to the data representation for the following layers, which is fed subsequently into a fully connected layer to perform the learning task (e.g., detection or classification). In most practical implementations, the "convolution" in CNN is replaced by a cross-correlation operator rather than Eq. (17), for speed-up purposes. In the case of input as a 2D image (size $L_1 \times L_2$) with multicolor channels (C_1) represented by a 3D tensor $\mathcal{I} = \{\mathcal{I}_{i,j,\alpha}\}_{i=1,j=1,\alpha=1}^{L_1,L_2,C_1} \in \mathbb{R}^3$, a convolutional layer with stride s and kernel size $m \times n$ will produce an output image $\tilde{\mathcal{I}}$ (of size $\tilde{L}_1 \times \tilde{L}_2$ with C_2 channels) as below,

$$\tilde{\mathcal{I}}_{k,\ell,\beta} = \sum_{i,j,\alpha}^{m,n,C_1} w_{i,j,\alpha,\beta} \cdot \mathcal{I}_{s(k-1)+i,s(\ell-1)+j,\alpha} \quad (k = 1,\ldots,\tilde{L}_1, \ell = 1,\ldots,\tilde{L}_2, \beta = 1,\ldots C_2) \tag{18}$$

here $\mathbf{w} = \{w_{i,j,\alpha,\beta}\}_{i=1,j=1,\alpha=1,\beta=1}^{m,n,C_1,C_2} \in \mathbb{R}^4$ is a four-tensor convolutional filter (kernel). With stride $s > 1$, the output size would be roughly reduced to $\frac{1}{s}$ of the original input size. Implementing a pooling layer of kernel size s will have the same effect.

One can understand that using kernels in NNs as being equivalent to template matching or "seeing" local information of a neighborhood while blocking information from far apart or less related regions, as depicted in Fig. 11 (left). This can be also visualized by vectorizing the input and the output as in Fig. 11 (right), from which one can realize CNNs only connect to certain nodes within a layer when compared to a fully connected NN; this is called *local connectivity property*. Overall, a CNN is a locally connected NN as it only considers local relations (receptive field) while it decouples information far away in space and/or time allowing for efficient data representation and improved task learning.

The property that makes a CNN distinguishable from other locally connected networks is that CNNs force the weights (kernel) to be repeatedly used, that is, $\{w_{i,j,\alpha,\beta}\}_{i=1,j=1}^{m,n} \in \mathbb{R}^2$ is used everywhere in the input feature map (e.g., $\mathcal{I}$). This parameter sharing scheme effectively takes advantage of the *spatial invariance property* of the imaging data, largely reducing the number of free parameters in the architecture.

5.C.2. Recurrent neural networks

Recurrent neural networks (RNNs)[88] are another variant of NNs especially useful for sequential (time series) data learning, such as voice, text data. Although, a 1D CNN can also model such data by taking into account local sequential relationships, the parameters (kernel) sharing scheme by CNNs is too simple and "shallow" for complex learning objectives. RNNs manage another way of sharing parameters but in a deeper and more sophisticated sense. Suppose we have a sequential data $\{\mathbf{x}^{(t)} \in \mathbf{R}^n | t \in T\}$ as input and $\{\tilde{\mathbf{y}}^{(t)} \in \mathbf{R}^m | t \in T\}$ as the corresponding labels where T denotes an index set labeling separation across time steps. Note that we only consider one sample here for convenience, the superscript no longer stands for sample index. The goal of an RNN is to learn the relation between data $\{\mathbf{x}^{(t)}\}$ and labels $\{\tilde{\mathbf{y}}^{(t)}\}$ via hidden units $\{\mathbf{h}^{(t)} \in \mathbf{R}^k\}$. In RNNs, two functions $f_\theta : \mathbb{R}^k \times \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^k$ and $g_\phi : \mathbb{R}^k \times \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$, relating $\{\mathbf{x}^{(t)}\}$, $\{\tilde{\mathbf{y}}^{(t)}\}$, $\{\mathbf{h}^{(t)}\}$ parameterized by NNs' weights θ,ϕ, are imposed to search for the recurrence relations,

$$\begin{aligned} \mathbf{h}^{(t)} &= f_\theta\left(\mathbf{h}^{(t-1)}, \mathbf{y}^{(t-1)}, \mathbf{x}^{(t)}\right) \in \mathbb{R}^k \\ \mathbf{y}^{(t)} &= g_\phi\left(\mathbf{h}^{(t-1)}, \mathbf{y}^{(t-1)}, \mathbf{x}^{(t)}\right) \in \mathbb{R}^m \end{aligned} \tag{19}$$

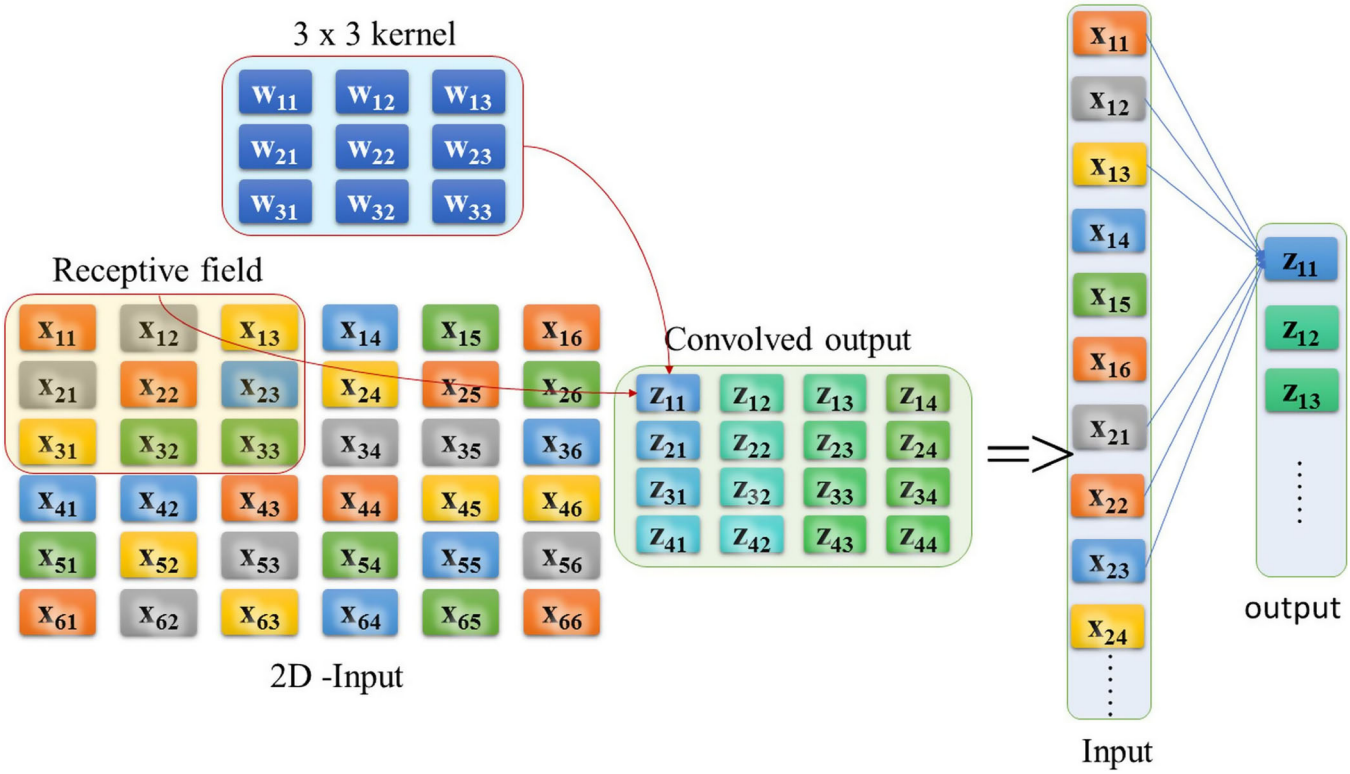

FIG. 11. Illustration of convolutional neural network. (Left) a kernel acts as a mask to consider only neighboring information (pixels) yet block information far part. For example, z_{11} in the convolved output is generated by applying 3 × 3 kernel on the receptive field (denoted) at the left upper corner. In the view from the right, one sees clearly that utilizing kernels (filters) is essentially another implementation of locally connected networks.

where θ and ϕ serving as unknown neural weights to be optimized. As these parameters are shared across different time points, free parameters in an RNN are largely reduced. Two common RNN architectures were invented in the early developments of RNNs, they are known as *Elman*[89] and *Jordan*[90] networks, which are simple RNNs that differed in their connection scheme.

These simple RNNs were known to suffer from two major disadvantages, (a) their back-propagation gradients tended to vanish or explode quickly and (b) the information farther apart cannot be connected. These two problems are resolved by using the so called *long short-term memory (LSTM)* architecture.[56]

An LSTM is a state-of-the-art RNN model composed of basic building blocks denoted as *gated units*, which learns by itself to store and forget internal memories when needed such that it is capable of creating long-term dependencies and paths through a time series as in Fig. 12. An LSTM has the following construction. Specify $f_\theta = f_{\text{LSTM}}$, $g_\phi = g_{\text{LSTM}}$ in Eq. (19) by

$$\begin{aligned}\mathbf{h}^{(t)} &= f_{\text{LSTM}}\left(\mathbf{h}^{(t-1)}, \mathbf{y}^{(t-1)}, \mathbf{x}^{(t)}\right) = \mathbf{G}_1^{(t)} \odot \mathbf{h}^{(t-1)} \\ \mathbf{y}^{(t)} &= g_{\text{LSTM}}\left(\mathbf{h}^{(t-1)}, \mathbf{y}^{(t-1)}, \mathbf{x}^{(t)}\right) = \mathbf{G}_3^{(t)} \odot \left(\sigma_3 \circ \mathbf{h}^{(t)}\right)\end{aligned}, \tag{20}$$

where $\odot$ denotes the component-wise multiplication, (i.e., for two $m \times n$ matrices A and B, $A \odot B$ will produce another $m \times n$ matrix C, where $C_{i,j} = A_{i,j} \times B_{i,j}$). And $\circ$ denotes the component-wise functional composition, that is, the i^{th} element in the resulting vector of function is the functional composition between the i^{th} elements of the two input vector of function. The affine transformation is defined by,

$$\begin{aligned}&\text{Aff}_{W,U,\mathbf{b}}(\xi, \eta) := W \cdot \xi + U \cdot \eta + \mathbf{b} \\ &(W \in \mathbb{R}^{p \times q}, U \in \mathbb{R}^{p \times r}, \mathbf{b} \in \mathbb{R}^r, \xi \in \mathbb{R}^q, \eta \in \mathbb{R}^r)\end{aligned} \tag{21}$$

In Eq. (20), three additional units $\mathbf{G}_\alpha^{(t)}$, where α = 1,2,3 denotes the forget gate, the input gate and the output gate, respectively,

$$\mathbf{G}_\alpha^{(t)} = \sigma_\alpha \circ \text{Aff}_{W_\alpha, U_\alpha, \mathbf{b}_\alpha}\left(\mathbf{x}^{(t)}, \mathbf{y}^{(t-1)}\right) \in \mathbb{R}^k \tag{22}$$

are used to control and determine when and how much should the previous information be kept or forgotten. Total unknown parameters of an LSTM are $(W_h, U_h, \mathbf{b}_h)$ and $\{(W_\alpha, U_\alpha, \mathbf{b}_\alpha) | \alpha = 1, 2, 3\}$. A simplified version of LSTM is called a gated recurrent units (GRU),where the number of gates is reduced to two, namely, reset and update gates, and hence more computationally efficient.[91]

5.C.3. Attention awareness

Attention awareness[92] is a special mechanism that equips NNs with the ability to focus on a subset of a feature map,

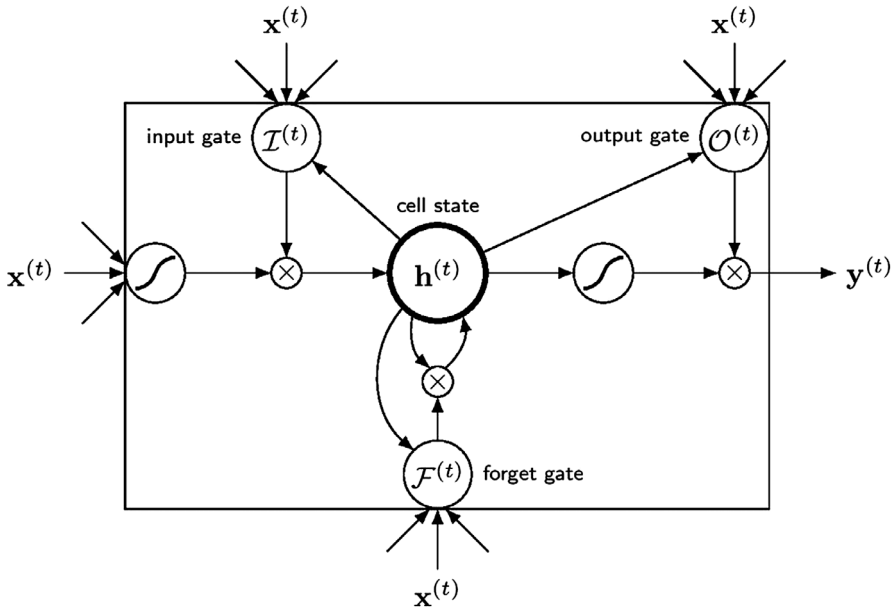

FIG. 12. The diagram of a gated units of long short-term memory, which is consisted of input, output and forget gates.

which is particularly useful when tailoring the architecture for specific tasks. Suppose, the input of an attention network $f_w(\cdot)$ is x and its output is $a, a_{i,j} \in [0,1]$. After applying it on the feature map z, it will produce an attention glimpse g as below:

$$a = f_w(x), g = a \odot z, \tag{23}$$

where $\odot$ is a component-wise multiplication. The attention mechanism assigns weights a to the input feature map. When a is restricted to be either 0 or 1, it is hard attention[93]; When a is between 0 and 1, it is called soft attention.[94]

Attention was first implemented in RNNs[94] for language translation tasks. In the classical *Seq2Seq* model, a context vector will be built out of the last hidden state from an encoder RNN, which takes source sequence as input, then a decoder RNN would take the context vector as input and generate target sequences. However, the model tends to "forget" the early part of source sequences when generating target sequences. Hence, attention networks are proposed to be applied on all the hidden states (z) of encoder RNN to learn which part of source sequences should be focused on when generating a certain piece of a target sequence (g).

Attention mechanisms can be applied to fuse and align information from heterogeneous data. For instance, by adopting an attention mechanism, the relation between an image and its caption can be learned.[95] Specifically, the imaging information extracted from a CNN can be consumed by the RNN, for each position of the caption, an attention mechanism fuses the information and generate the desired caption on a word by word basis.

5.C.4. Other interesting architectures

We have introduced two fundamental architectures for DL algorithms, that is, CNNs and RNNs, together with fully connected NNs. These three basic architectures comprise almost all the deep architectures used today. Other architectures fall into these three categories or are a combination of them.

For instance, a variational autoencoder[57] is a special type of CNN or fully connected NN which has a bottleneck structure and is employed for data compression applications. As Fig. 13 suggests, its input and output dimension are exactly the same and the dimension of its latent variables is usually significantly smaller than its input size. In a VAE, latent variables are sampled from a designate distribution, for example, Gaussian distribution, which is parameterized by the encoder. The decoder is responsible for re-constructing the original input from the latent variables. The loss function of a VAE is defined as a lower bound of the data log-likelihood function. It turns out to be a combination of two terms; one penalizes reconstruction error as in a plain autoencoder, the other forces the similarity [Kullback–Leibler divergence (KL)] between the learned distribution and true prior distribution. The additional KL-term of VAE can be regarded as a way of regularization. Contrasted with the deterministic autoencoder, it helps suppressing the noise in data.

U-Net,[96] which has been successfully applied in many medical imaging segmentations, is also CNN based, with its name derived from its "U" shape as shown in Fig. 14. U-Net is composed of three sections: contraction, bottleneck, and expansion sections. The contraction (encoding) section is made of several blocks of convolutional layers and max pooling layers. It is responsible for learning complex features. The bottleneck section is the bottommost layer mediating between the contraction and expansion layers. The heart of U-net lies in the expansion (decoding) section, which consists of several blocks of convolutional layers and an upsampling layer. Particularly, it appends the outputs from the contraction section to their corresponding components in the expansion

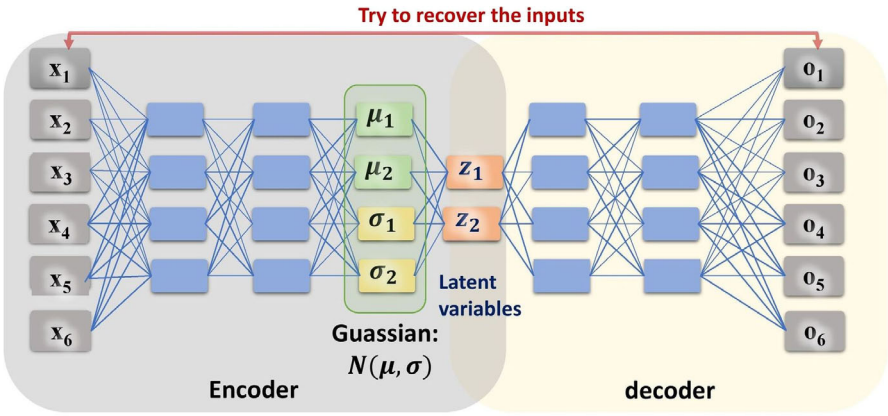

FIG. 13. The diagram of a variational autoencoders, which is composed of encoder and decoder networks.

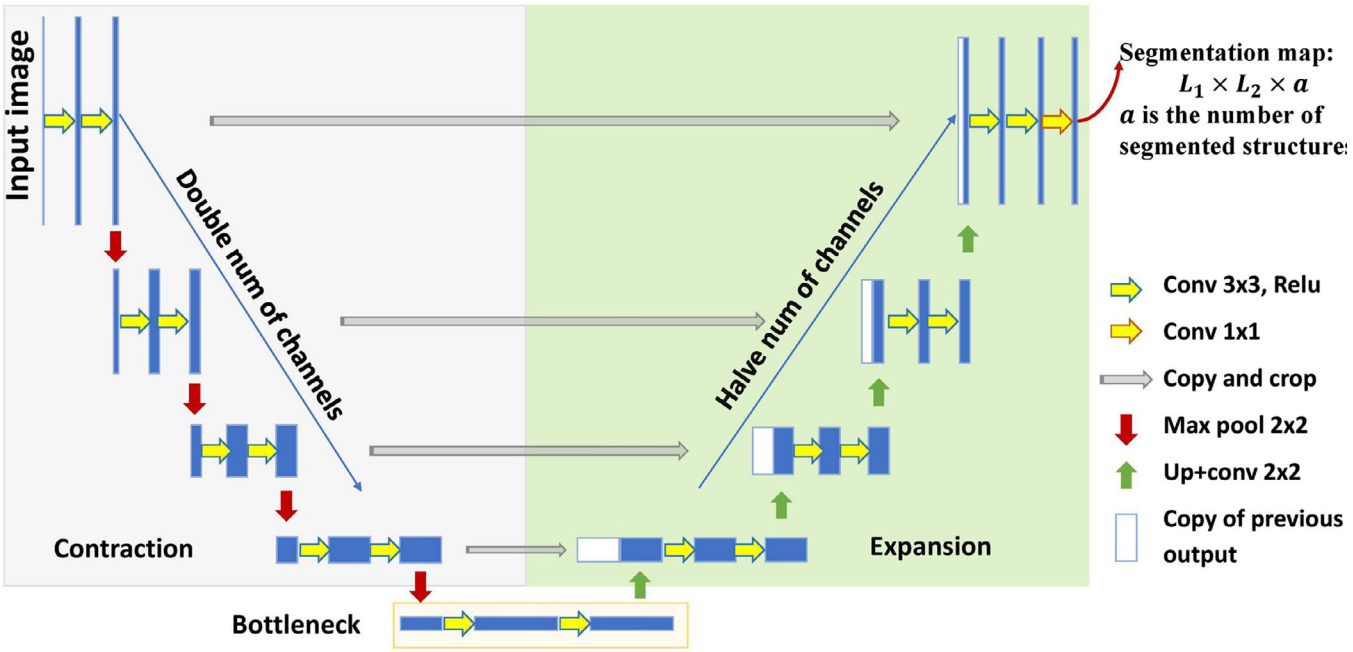

FIG. 14. The diagram of a U-net consisting of contraction, bottleneck, and expansion section. The size of output is $L_1 \times L_2 \times a$, where $L_1 \times L_2$ is the size of original image, and a is the number of structure that needs to be contoured.

section. Such a design helps preserve the structural integrity of the image and reduces distortion enormously. The loss function of U-net is defined as energy function computed by a pixel-wise soft-max over the final feature map combined with the cross-entropy loss function.

A generative adversarial networks (GAN)[58] is another popular NN architecture and is considered among the most exciting breakthroughs in DL of the past decade. Numerous GAN variants have been invented for different purposes following their invention by Ian Goodfellow in 2014. Like a VAE, a GAN is also a generative model,[61] which aims to learn the true data distribution from the training set so as to generate new data points. However, unlike VAEs, which try to minimize the lower bound of likelihood function, GANs learn the data distribution through pitting a competition between two adversarial components, so-called *generator* and *discriminator* networks in some form of a game. The basic idea is that the generator tries to generate new data from random noise to mimic the real data, while the discriminator tries to distinguishable any fake data from the real data in an analogy to a Turing machine. The two parts are alternately trained to compete with each other and overtime the performance of both will be boosted. The original GAN as shown in Fig. 15 works on the dataset without labels (unsupervised learning) and is based on fully connected architectures. However, its idea can be directly applied to CNNs or other architectures to create a deep convolution GAN (DCGAN),[97] which has had wide applications in the imaging domain such as sequence generation. A cGAN[98] is a GAN variant that can be used to learn multimodal data models and can be used for generation of synthetic data for training of limited sample outcome prediction models, for instance. Its generator and discriminator are both trained conditioning on the label/other information.

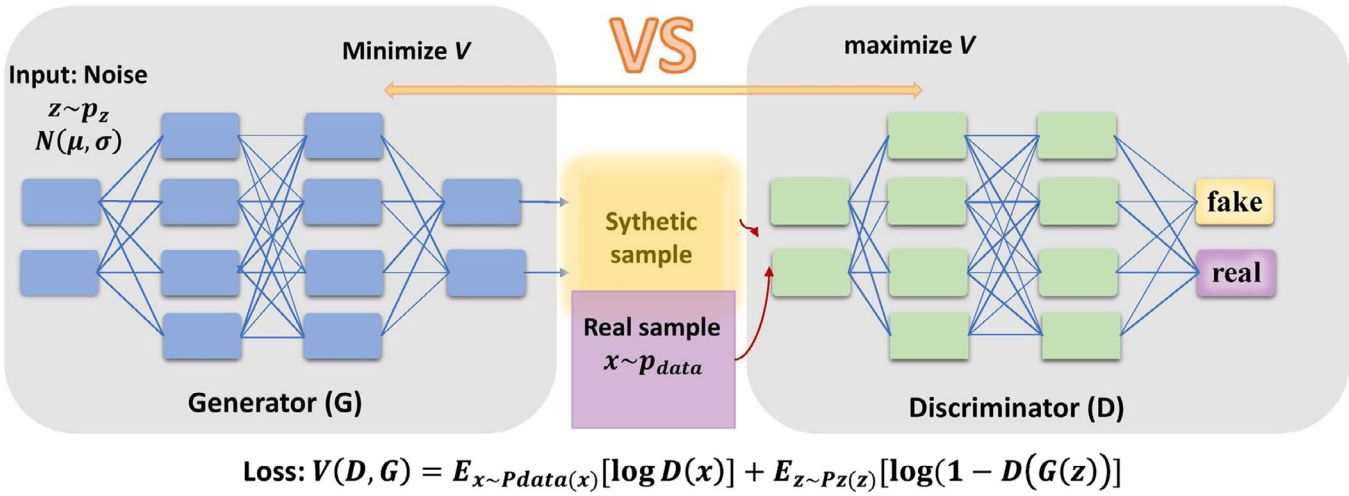

FIG. 15. A generative adversarial neural network which learns data distribution through the competition between two adversarial components.

TABLE III. A snippet of a dataset containing the six predictor for local control prediction

id	Gender	pathologic_N	pathologic_stage	pathologic_T	other_dx	tobacco_smoking_history
1	Male	N0	Stage IIIB	T3	No	4
2	Male	N1	Stage IIA	T1b	No	3
3	Female	N2	Stage IIIA	T1	No	2
4	Female	NX	Stage IB	T2	No	4

InfoGAN[99] is an version of GAN that learns meaningful codes of unlabeled data. CycleGANs[100] were invented for domain image transfer; they can accomplish one-to-one translation with unpaired datasets such as synthetic CT generation from MR images.[101]

5.D. Example implementations

We implemented a fully connected NN in pytorch for a binary classification problem. The dataset in Table III contains 45 patients in TCGA-LUAD and TCGA-LUSC who received external beam radiotherapy for a primary tumor as adjuvant therapy. Only the patients who have complete dose and treatment outcome information were kept. In addition to dose, six clinical variables were selected for local control prediction (progressive = 1/local control = 0).

Before training the model, all the variables were converted into numerical values and were Z-score normalized to avoid numerical instability. The dataset is randomly split into training and testing sets. The code is available for the reader reference at https://github.com/sunancui/lung_TCGA_prediction, where the history of training/test losses is plotted and model performance is evaluated with area under receiver operating curve (AUC). In Fig. 16, AUC results of both training/testing data are shown.

Another implementation of deep U-Net for liver tumor segmentation is also provided for the reader reference at https://github.com/sunancui/liver_segmentation. The dataset is from the Liver Tumor Segmentation Challenge (LiTS), containing 131 training/70 test 3D CT images of size (512, 512, 74 ~ 784). The last dimension of images varied, as patients have different numbers of slices. During the training, each slice from patients was taken as individual inputs. Hence the input of U-Net will be a 2D image. The U-Net is applied to classify each pixel into three different classes: liver, lesion, something else. Dice similarity was used for evaluation and is calculated for the three classes to evaluate the resulting classification models. The code was implemented to split the 131 training patients into training/validation sets. The model was trained up to 17 epochs and was evaluated on the validation set. Some sample results are shown in Fig. 17. Interested readers are also encouraged to play with the architecture definitions in the code and test their final models on the 70 hold-out patients.

5.E. Reinforcement learning

Reinforcement learning is the third category of machine learning apart from supervised and unsupervised. It is designed to achieve a definite goal by optimizing a "reward" function. The learning process of an RL is through interaction with an "environment" so that RL user (also called an agent) tries to earn the most reward to obtain the designated goal.

Reinforcement learning is based on the MDP, five-tuple $(S, \mathcal{A}, P, \gamma, R)$. $S = \{(x_1, \ldots, x_n) \in \mathbb{R}^n\}$ is used to describe an environment, it is the space of its all possible states. $\mathcal{A}$ is a collection of all actions the agent can take. $R : \Omega \to \mathbb{R}$ is the

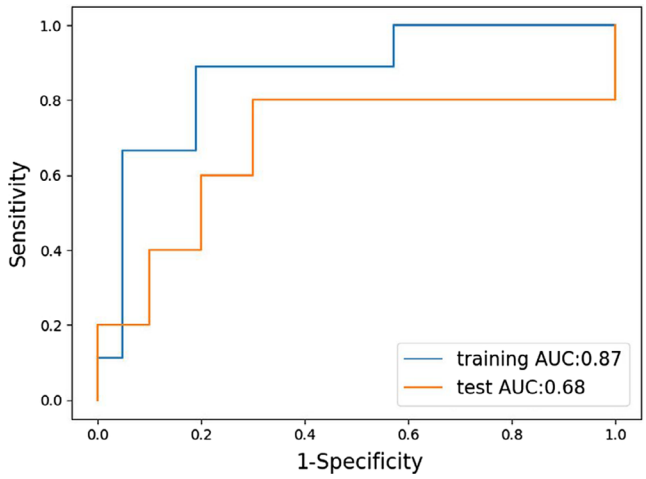

FIG. 16. Area under receiver operating curves of local control prediction in lung cancer by neural network.

reward function given on the product space $\Omega = S \times \mathcal{A} \times S$, that is, the reward is determined by actions and states. $\gamma \in [0,1]$ is the discount factor representing the possible rewards that propagate from future back to the present. And $P : \mathcal{F} \to [0,1]$ is a probability describing the transition between states. The probability mass function (pmf) $(s,a,t) \mapsto P(s,a,t)$ denotes the transition probability from state $s \in S$ to another $t \in S$ under an action $a \in \mathcal{A}$. Consequently, this induces the condition probability:

$$P_{sa}(t) \equiv \text{Prob}(t|s,a) \equiv P(s,a,t)/P(s,a) \tag{24}$$

on space of next states t conditioned on previous state s and current action a.

An MDP is also called an "environment" in modern machine learning development. Each element $s_i \in S$ is called a "state" representing a configuration in MDP. An action $a_i \in \mathcal{A}$ corresponds to a control or a move to be exerted. The purpose of an agent in the RL algorithm is to find a sequence of actions $\{a_0, a_1, \ldots\}$ such that the following path in S collects maximum rewards:

$$s_0 \xrightarrow[\pi]{a_0} \quad s_1 \xrightarrow[\pi]{a_1} \quad s_2 \xrightarrow[\pi]{a_2} \cdot s_3 \cdots \tag{25}$$

In other words, an agent is a policy function $\pi : S \to \mathcal{A}$ who provides an action $a = \pi(s)$ under a state s. There are mainly two ways to construct a policy function by *policy-based* and *value-based* methods in RL where the former tries to find a policy function directly[102] via $\pi^{(\theta)}$ while the latter finds a policy implicitly via a Q-function defined by,

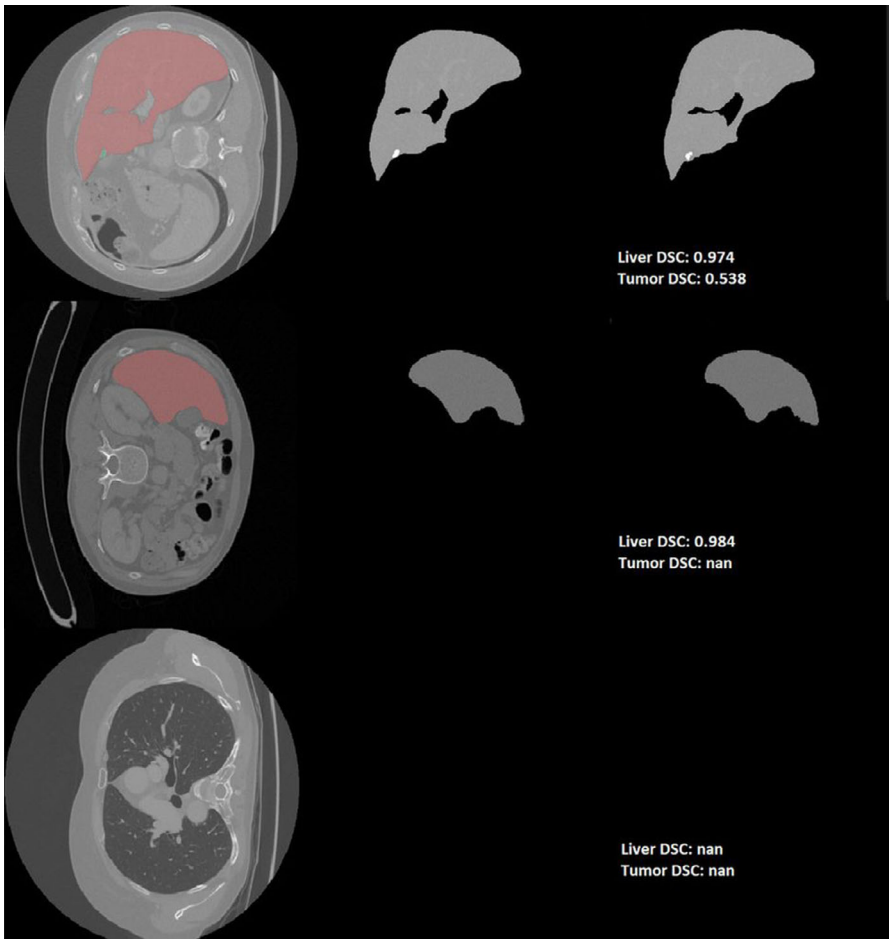

FIG. 17. Example results from the validation set with dice similarity coefficient shown, where "nan" indicates no such structure existing in the ground truth label. (first column: original image slice, second column: ground truth label, third column: prediction).

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{k=0}^{\infty}\gamma^{k}R(s_k,\pi(s_k))\middle|\pi, s_0 = s, a_0 = a\right] \quad (26)$$

An optimal policy $\pi^* : S \rightarrow \mathcal{A}$ is then derived by maximizing the Q-function such that $Q^{\pi^*} = \max_{\pi} Q^{\pi}$. Such RL algorithms that use Q-functions are called Q-learning. A practical way to compute the Q-function Eq. (26) is via the Bellman's iteration defined by:

$$\tilde{Q}_{i+1}(s,a) = \mathbb{E}_{t\sim P_{sa}}\left[R(s,a) + \gamma \max_{b\in A}\tilde{Q}_i(t,b)\right]. \quad (27)$$

such a sequence has a unique converging point where $\{\tilde{Q}_i\}_{i=1}^{\infty} \rightarrow Q^{\pi^*}$ and so

$$\tilde{Q}^{*}(s,a) = \mathbb{E}_{t\sim P_{sa}}\left[R(s,a) + \gamma \max_{b\in A}\tilde{Q}^{ast}(t,b)\right]. \quad (28)$$

Consequently, the optimal value Q^{π^*} can be easily computed by Bellman's equation $\{\tilde{Q}_i\}_{i=1}^{\infty}$ instead.

RL serves as an independent ML field separate from supervised and unsupervised learning in the sense that it does not rely on observed data nor comparing its outputs with any data labels. It is designed to make an optimal decision or a control action and therefore it is particularly suitable for adaptive treatment planning and optimizing sequential decision-making. An implementation in radiotherapy to optimize dose adaptation can be found in.[45]

6. HOW TO VALIDATE?

When building a model one would like to know how good it is. This is particularly true in the context of ML/Dl, where the models have tremendous ability to overfit (memorize) the training data and one would like to confirm that the model can actually have a good generalization, that is, perform well on out-of-sample or unseen data.

Common validation methods are based on statistical resampling techniques and include cross-validation (CV) and bootstrapping. *CV*[65] (e.g., K-Fold CV, leave-one-out CV) is a widely used resampling technique in classification/regression model internal validation analysis. In CV, one would systematically split the data into training/testing sets, train the model on training sets, then test a selected performance metrics (accuracy, area under receiver operating curve, specificity, sensitivity) on the test datasets in a round-robin fashion and average the results. Stratified K-fold CV is usually applied for a imbalanced classification problem, where each fold in CV contains roughly the same number of positive/negative cases.

Bootstrapping[103] is an inherently computationally more intensive procedure than CV but generates more realistic results. Typically, a bootstrap pseudo-dataset is generated by randomly making copies of original data samples, similar to Monte Carlo techniques, with an estimated inclusion probability of 63%. The bootstrap often works acceptably well even when datasets are small or unevenly distributed. To achieve valid results this process must be repeated many times, typically several hundred or thousand times depending on the original dataset size.

However, in practice, the notion of a comprehensive validation process is more complicated than just implementing CV considering various scenarios of sample size and requirements. The Transparent reporting of a multivariable prediction model for individual prognosis or diagnosis (TRIPOD)[104] guidelines provide a detailed list of different trustworthy-level model validation pipelines. Specifically, it categorizes model validation into four types, from type 1 to type 4, where the validation rules become more stringent. Type 1 uses the whole dataset to develop the model and test its performance, whether using a resampling method or not. Type 2 uses only parts of the data to develop a model and reserve the rest for testing, the split can be either random (subtype a) or by location/time (subtype b). Types 3 and 4 are both refer to external (independent) validation, both using a separate dataset to evaluate the model. Particularly, type 4 further requires the model has been already published and is being evaluated in a meta-analysis like scheme . In all, developers have the responsibility to figure out the best validation pipeline for their specific studies, while higher-level validation is usually desirable when possible to achieve acceptance from the community and demonstrate feasibility for clinical implementation.

7. ARE YOU THE HUMAN STILL RELEVANT?

The human community of ML and DL developers and users will play a decisive role (knowingly or unknowingly) in the continued advancement of machine learning technologies and on the ultimate impact these technologies will have on society including medical physics and radiation oncology. In considering our future role in shaping these technologies, some perspective can be gained by considering Amara's Law[105]: "We tend to overestimate the effect of a technology in the short run and underestimate the effect in the long run." Fig. 18 provides a visual representation of this disconnect between human expectations and how technological productivity develops. It can be conjectured that ML/DL technology is currently in the tail-end of the overestimation phase. The development of ML paradigms and applications has seen explosive growth over the past twenty years,[106] and reports on AI can sometimes overinflate its potential. However, despite the current hype, it is suggested that, based on past trends in this field, the research community's (and eventually by extension the general public's) interest in AI will begin to diminish in the coming decade,[106] even as progress is made to address its most significant limitations. Future development of these technologies will depend on continuing efforts and investments in (a) curation of high quality, accessible datasets, (b) developing algorithms to make high accuracy predictions using available datasets, and (c) buy-in from prospective users.

With respect to medical and radiation oncology community, this user buy-in ultimately needs to come from clinicians in order to establish ML and DL techniques as a standard of care within the clinical workflow. To gain acceptance from

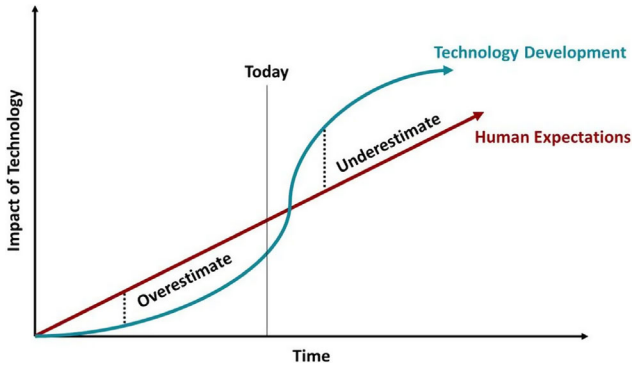

FIG. 18. Graphical representation of Amara's law and today's current state of machine learning and deep learning technologies.

the clinical team, a new technology must prove itself to be useful in a clinical setting, as well as to not subject patients to the risk of unnecessary harm. As a matter of ethical accountability, ML/DL techniques for clinical workflows need to be transparent enough to allow human experts to be part of the decision loop.

7.A. Why interpretability matters?

Although ML/DL have shown promising accuracy in solving practical problems in radiation oncology, interpretability is a necessary requirement for ML and DL to be viable clinical tools. Accuracy in this clinical context refers to the ability of the ML and DL algorithms to perform the tasks they are assigned either as well, or better than what a human could accomplish. Interpretability refers to the ability of the clinician to confidently understand and interpret the results of an algorithm, without necessarily having to understand the minute details of its mechanics.[107] This concept of interpretability is particularly important in a clinical setting because it can help act as a *fail-safe* against instances in which algorithms may produce results that are flawed due to inherent bias in the training data or other unforeseen bugs.

Algorithms which are both accurate and interpretable are able to gain a clinician's trust as clinical tools because they can be expected to perform their function correctly most of the time and do not require the clinician to blindly accept their results. Existing ML and DL techniques are recognized to suffer from a tradeoff between accuracy and interpretability, and therefore more work is necessary to develop ML/DL methods which can achieve a better balance between these two qualities, such as the use of gradient maps or proxy models with DNN or human in the loop with ML approaches as discussed below.[107]

7.B. How to handle human–computer interactions in decision-making?

To meet the needs of practical clinical scenarios in radiation oncology, it is important to involve humans in the loop of model development and decision-making. Human-in-the-loop (HITL) is a practical guide to optimize the entire learning process by incorporating human-computer interaction into the system. Machines are known for their great capabilities of learning from vast datasets, while humans are much better at making decision with scarce information. Human–computer interaction is expected to combine best human intelligence and machine intelligence to make robust and accurate decisions. To develop a robust system to support decision-making in clinical practice, it is important to investigate both (a) how to make machines more accurate with physicians/physicists' input and (b) how to improve a physician/physicist's task with the aid of a machine. The previous one helps leverage human intelligence into AI algorithms for more accurate and robust decisions. The latter one is the real decision-making scenario in practice. During this process, the uncertainty of predictions made by human/machine needs to be estimated. The performance of machine versus machine-aided human should be thoroughly investigated. Specifically, in the scenarios where physicians/physicists and machine make different decisions, a scheme should be designed to resolve the disagreement and to make best use of both physicians/physicists and machine intelligence, which is going to be better than either alone.

8. WHAT PITFALLS MAY EXIST?

There are caveats to be mindful of when medical physicists develop or deploy ML/DL models to solve practical problems in medicine or radiation oncology: bias, data quality, data sharing, and data privacy, to name a few. *Biases* in the data including age, gender and race,[108] or other sources of underrepresented components in the AI algorithm training may potentially aggravate health care disparities. That is, models developed with bias may lead to misleading diagnosis or poor decision support for patients in minority groups whose members have not been sufficiently included in the dataset that is used to develop the models. It is found that some genetic variants, which are more common among black Americans than white Americans, were mis-classified as pathogenic in a study with insufficient inclusion of black Americans in control cohorts.[109] Patients with melanoma and lung cancer treated with the same immunotherapy regimen may respond differently based on sex, with higher remission probability for male patients.[110] A segmentation algorithm designed for one anatomic site may still produce (erroneous, fake) contours if inappropriately applied to the wrong anatomic site. Thus, AI Algorithms can take whatever information we feed into them and output a result that needs to be carefully scrutinized/tested before clinical implementation to mitigate such bias. It is physicists' responsibility to understand ML/DL algorithms' scope of application as well as their limitations and to conduct a thorough battery of quality assurance (QA) on software tools that are used in the clinic. Physicists are also responsible for eliminating any potential bias by ensuring that implemented ML/DL algorithms continue to perform as expected from an accuracy and clinical context perspective as well by monitoring their performance and conducting routine QA tests monthly or yearly, for instance.

Data quality is another important concern when developing and applying ML/DL tools. There are many aspects of data quality, including accuracy, completeness and reliability etc. Accuracy means the data that have been collected are of original source data. Completeness refers to the fact that all required data elements are present. Reliability requires that data remain consistent and the information generated is understandable. Medical physicists should examine the accuracy of data before training new models, for example, filtering out any abnormal outliers, double checking whether patients are correctly linked to their information, etc. By gathering complete high quality data that cover patients with different age, genders and race, medical physicists may also help avoid potential bias in the algorithms. Medical physicists should also inspect the readability of data, that is, data yield the same results on repeated collections, processing, storing and display of information in different medical records. Quantitatively, the Shapley value metric[111] was proposed as a tool for evaluating quality of healthcare data for use in Ml/DL algorithms. It interprets whether the presence of an individual data can help or hurt the overall performance of the ML/DL predictive model. Such a quantitative tool can be useful and may provide guidance for medical physicists to assure data quality in the future.

Data of large volumes are always desirable in the era of precision medicine and predictive analytics. However, proper data sharing protocols are a precondition for generating large volumes given the limited sample size at an individual radiation oncology department. Medical physicists should be aware with some thorny issues including data privacy, data security and data interoperability[112] when involving in multi-institutional studies and assist with other stake holders toward providing secure and safe *data lake* exchange platforms for radiation oncology applications.

9. WHAT DOES THIS ML ERA IMPLY FOR MEDICAL PHYSICISTS?

Machine and deep learning technologies are promising tools to introduce substantial positive changes to the medical physics workflow and is likely to become an indispensable component of its future. Ml/DL are likely to automate many of the mundane procedures as well as some of the essential processes involving treatment planning and quality assurance to name of few. This is likely to lead to changes in the future role of the medical physicist in the clinic. Medical physicists as vanguards of technology in medicine are at the forefront of embracing ML role and its potential. However, this also requires that medical physicists become more acquainted with ML/DL, its algorithms and their implementations, and lead the process of acceptance and commissioning of such algorithms in their clinic. These issues are likely to require changes in the medical physicist training and their job profile. Nevertheless, the benefits are expected to significantly outweigh some of the possible downsides, including achieving better efficiency, improved safety, and potentially better quality of care for patients, which is the ultimate goal of medical practice.

10. CONCLUSIONS

Machine learning/DL algorithms have witnessed increasing applications in treatment planning and quality assurance to improve efficiency and safety in radiation oncology. These tools can also improve outcome prediction and personalizing of radiation oncology treatment. To utilize these tools effectively, medical physicists need to define the problem at hand, know its corresponding category in the ML/DL lexicon, determine suitable algorithms/models to utilize, collect data of appropriate size and conduct validation in the proper way to ensure generalization to unseen data while being mindful of possible pitfalls and how to avoid them. This article aims to serve as a tutorial and stepping stone in that direction. Moreover, establishing ML/DL algorithms into standard clinical workflow will require additional attention to balancing accuracy and interpretability in development of these models. Medical physicists are encouraged to be informed of the latest ML/DL's developments as part of their continued education, get more acquainted with ML/DL literature and become leader of the processes of accepting and commissioning ML/DL for routine clinical practice.

ACKNOWLEDGMENTS

This work was supported in part by the National Institutes of Health P01-CA059827, R37-CA222215 and R01-CA233487.

CONFLICT OF INTEREST

The authors have no relevant conflict of interest to disclose.

[a)]Author to whom correspondence should be addressed. Electronic mail: sunan@umich.edu.

REFERENCES

1. Bi WL, Hosny A, Matthew B, et al. Artificial intelligence in cancer imaging: clinical challenges and applications. *CA*. 2019;69:127–157.
2. Levine AB, Schlosser C, Grewal J, Coope R, Jones SJM, Yip S. Rise of the machines: advances in deep learning for cancer diagnosis. *Trends Cancer*. 2019;5:157–169.
3. Sahiner B, Pezeshk A, Hadjiiski LM, et al. Deep learning in medical imaging and radiation therapy. *Med Phys*. 2019;46:e1–e36.
4. Nikolov S, Blackwell S, Mendes R, et al. Deep learning to achieve clinically applicable segmentation of head and neck anatomy for radiotherapy; 2018.
5. Tseng H-H, Luo Y, Ten Haken RK, El Naqa I. The role of machine learning in knowledge-based response-adapted radiotherapy. *Front Oncol*. 2018;8:266.
6. Shiraishi S, Moore KL. Knowledge-based prediction of three-dimensional dose distributions for external beam radiotherapy. *Med Phys*. 2016;43:378–387.

7. Valdes G, Simone CB, Chen J, et al. Clinical decision support of radiotherapy treatment planning: a data-driven machine learning strategy for patient-specific dosimetric decision making. *Radiother Oncol*. 2017;125:392–397.
8. El Naqa I, Irrer J, Ritter TA, et al. Machine learning for automated quality assurance in radiotherapy: a proof of principle using EPID data description. *Med Phys*. 2019;46:1914–1921.
9. Good D, Joseph LW, Robert LQ, Jackie W, Yin F-F, Das SK. A knowledge-based approach to improving and homogenizing intensity modulated radiation therapy planning quality among treatment centers: an example application to prostate cancer planning. *Int J Radiat Oncol Biol Phys*. 2013;87:176–181.
10. Avanzo M, Stancanello J, El Naqa I. Beyond imaging: the promise of radiomics. *Phys Med*. 2017;38:122–139.
11. Mattonen SA, Palma DA, Johnson C, et al. Detection of local cancer recurrence after stereotactic ablative radiation therapy for lung cancer: physician performance versus radiomic assessment. *Int J Radiat Oncol Biol Phys*. 2016;94:1121–1128.
12. Deist TM, Dankers FJWM, Valdes G, et al. Machine learning algorithms for outcome prediction in (chemo)radiotherapy: an empirical comparison of classifiers. *Med Phys*. 2018;45:3449–3459.
13. Cui S, Luo Y, Hsin Tseng H, Ten Haken RK, El Naqa I. Artificial neural network with composite architectures for prediction of local control in radiotherapy. *IEEE Trans Radiat Plasma Med Sci*. 2019;3:242–249.
14. Cui S, Luo Y, Tseng HH, Ten Haken RK, El Naqa I. Combining handcrafted features with latent variables in machine learning for prediction of radiation-induced lung damage. *Med Phys*. 2019;46:2497–2511.
15. Men K, Geng H, Zhong H, Fan Y, Lin A, Xiao Y. A deep learning model for predicting xerostomia due to radiation therapy for head and neck squamous cell carcinoma in the RTOG 0522 clinical trial. *Int J Radiat Oncol Biol Phys*. 2019;105:440–447.
16. Ibragimov B, Toesca D, Chang D, Yuan Y, Koong A, Xing L. Development of deep neural network for individualized hepatobiliary toxicity prediction after liver SBRT. *Med Phys*. 2018;45:4763–4774.
17. Schuler T, Kipritidis J, Eade T, et al. Big data readiness in radiation oncology: an efficient approach for relabeling radiation therapy structures with their TG-263 standard name in real-world data sets. *Adv Radiat Oncol*. 2019;4:191–200.
18. Meyer P, Noblet V, Mazzara C, Lallement A. Survey on deep learning for radiotherapy. *Comput Biol Med*. 2018;98:126–146.
19. Tseng HH, Wei L, Cui S, Luo Y, Ten Haken RK, El Naqa I. Machine learning and imaging informatics in oncology. *Oncology*. 2018;1–19. https://doi.org/10.1159/000493575
20. El Naqa I. Perspectives on making big data analytics work for oncology. *Methods*. 2016;111:32–44.
21. McNutt TR, Bowers M, Cheng Z, et al. Practical data collection and extraction for big data applications in radiotherapy. *Med Phys*. 2018;45:e863–e869.
22. Kang J, Schwartz R, Flickinger J, Beriwal S. Machine learning approaches for predicting radiation therapy outcomes: a clinician's perspective. *Int J Radiat Oncol Biol Phys*. 2015;93:1127–1135.
23. Bibault J-E, Giraud P, Burgun A. Big data and machine learning in radiation oncology: state of the art and future prospects. *Cancer Lett*. 2016;382:110–117.
24. Feng M, Valdes G, Dixit N, Solberg TD. Machine learning in radiation oncology: opportunities, requirements, and needs. *Front Oncol*. 2018;8:110.
25. Boldrini L, Bibault J-E, Masciocchi C, Shen Y, Bittner M-I. Deep learning: a review for the radiation oncologist. *Front Oncol*. 2019;9:977.
26. Thompson RF, Valdes G, Fuller CD, et al. Artificial intelligence in radiation oncology: a specialty-wide disruptive transformation? *Radiother Oncol*. 2018;129:421–426.
27. Sahiner B, Pezeshk A, Hadjiiski LM, et al. Deep learning in medical imaging and radiation therapy. *Med Phys*. 2019;46:e1–e36.
28. Domingos P. A few useful things to know about machine learning. *Commun ACM*. 2012;55:78–87.
29. Mohri M, Rostamizadeh A, Talwalkar A. *Foundations of Machine Learning*. Cambridge, MA: The MIT Press; 2012.
30. Kotsiantis SB. Supervised machine learning: a review of classification techniques. In: Real W, ed. *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering*. Amsterdam, The Netherlands: IOS Press; 2007:3–24.
31. El Naqa I, Kerns SL, Coates J, et al. Radiogenomics and radiotherapy response modeling. *Phys Med Biol*. 2017;62:R179.
32. Peng CYJ, Lee KL, Ingersoll GM. An introduction to logistic regression analysis and reporting. *J Educ Res*. 2002;96:3–14.
33. Hearst M. Support vector machines. *IEEE Intell Syst*. 1998;13:18–28.
34. Breiman L. Random forests. *Mach Learn*. 2001;45:5–32.
35. Priddy KL, Keller PE. *Artificial Neural Networks: An Introduction* (Vol. 68). Bellingham: SPIE Press; 2005.
36. Emre Celebi M, Aydin K. *Unsupervised Learning Algorithms*, 1st edn. Berlin: Springer Publishing Company; 2016.
37. Hartigan JA, Wong MA. A k-means clustering algorithm. *J STOR*. 1979;28:100–108.
38. Putora PM, Peters S, Bovet M. Informatics in Radiation Oncology. In: El Naqa I, Li R, Murphy MJ, eds. *Machine Learning in Radiation Oncology: Theory and Applications*. Cham, Switzerland: Springer; 2015:57–70.
39. Abdi H, Williams LJ. Principal component analysis. *WIREs Comput Stat*. 2010;2:433–459.
40. van der Maaten L, Hinton G. Visualizing data using t-SNE; 2008.
41. Baldi P. Autoencoders, unsupervised learning and deep architectures. In: Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27, UTLW'11. JMLR.org; 2011:37–50.
42. Sutton RS, Barto AG. *Reinforcement Learning: An Introduction*, Vol. 1. Cambridge: MIT Press; 1998.
43. Puterman ML. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st edn. New York, NY: John Wiley & Sons Inc.; 1994.
44. Silver D, Huang A, Maddison CJ, et al. Mastering the game of go with deep neural networks and tree search. *Nature*. 2016;529:484–503.
45. Tseng H-H, Luo Y, Sunan C, et al. Deep reinforcement learning for automated radiation adaptation in lung cancer. *Med Phys*. 2017;44:6690–6705.
46. LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature*. 2015;521:436–444.
47. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ, eds. *Advances in Neural Information Processing Systems 25*. Red Hook, NY: Curran Associates Inc; 2012:1097–1105.
48. Mikolov T, Karafiát M, Burget L, Cernocký JH, Khudanpur S. Recurrent neural network based language model. In: INTERSPEECH, edited by Takao Kobayashi, Keikichi Hirose, and Satoshi Nakamura. ISCA; 2010:1045–1048.
49. Guyon I, Elisseeff A. An introduction to variable and feature selection. *J Mach Learn Res*. 2003;3:1157–1182.
50. Ackley DH, Hinton GE, Sejnowski TJ. A learning algorithm for Boltzmann machines. *Cogn Sci*. 1985;9:147–169.
51. Hinton GE, Osindero S, Teh Y-W. A fast learning algorithm for deep belief nets. *Neural Comput*. 2006;18:1527–1554.
52. Nair V, Hinton GE. Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10); 2010:807–814.
53. Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics; 2011:315–323.
54. Kingma D, Ba J. Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980; 2014.
55. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. arXiv preprint arXiv:0706.1234; 2015.
56. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput*. 1997;9:1735–1780.
57. Kingma DP, Welling M. Auto-encoding variational Bayes; 2013. Cite arxiv:1312.6114.
58. Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets. In: Advances in Neural Information Processing Systems; 2014:2672–2680.
59. Esteva A, Kuprel B, Novoa RA, et al. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*. 2017;542:115–118.

60. Dalmiş MU, Litjens G, Holland K, et al. Using deep learning to segment breast and fibroglandular tissue in MRI volumes. *Med Phys*. 2017;44:533–546.
61. Ng AY, Jordan MI. On discriminative vs. generative classifiers: a comparison of logistic regression and naive bayes. In: *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic, NIPS'01*. Cambridge, MA: MIT Press; 2001:841–848.
62. Wolterink JM, Dinkla AM, Savenije MHF, et al. Deep MR to CT synthesis using unpaired data. CoRR abs/1708.01155; 2017. arXiv:1708.01155.
63. Dietterich T. Overfitting and undercomputing in machine learning. *ACM Comput Surv*. 1995;27:326–327.
64. Zhang C, Bengio S, Hardt M, Recht B, Vinyals O. Understanding deep learning requires rethinking generalization; 2017.
65. Bishop CM. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag; 2006.
66. Linial N, Mansour Y, Rivest RL. Results on learnability and the Vapnik-Chervonenkis dimension. *Inf Comput*. 1991;90:33–49.
67. Perlich C, Provost F, Simonoff JS. Tree induction vs. logistic regression: a learning-curve analysis. *J Mach Learn Res*. 2003;4:211–255.
68. Perez L, Wang J. The effectiveness of data augmentation in image classification using deep learning. CoRR abs/1712.04621; 2017, arXiv:1712.04621.
69. Tan C, Sun F, Kong T, Zhang W, Yang C, Liu C. A survey on deep transfer learning. arXiv e-prints, arXiv:1808.01974; 2018, arXiv:1808.01974 [cs.LG].
70. Shin HC, Roth HR, Gao M, et al. Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE Trans Med Imaging*. 2016;35:1285–1298.
71. Zhen X, Chen J, Zhong Z, et al. Deep convolutional neural network with transfer learning for rectum toxicity prediction in cervical cancer radiotherapy: a feasibility study. *Phys Med Biol*. 2017;62:8246–8263.
72. Walach J, Filzmoser P, Hron K. Chapter seven-data normalization and scaling: consequences for the analysis in omics sciences. In: Jaumot J, Bedia C, Tauler R, eds. *Data Analysis for Omic Sciences: Methods and Applications, Comprehensive Analytical Chemistry*, Vol. 82. Amsterdam: Elsevier; y:165–196.
73. LeCun Y, Bottou L, Orr GB, Müller KR. *Efficient backprop*. In: Neural Net-works: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop. London, UK: Springer-Verlag; 1998:9–50.
74. Rubin DB. An overview of multiple imputation. In: Proceedings of the Survey Research Section, American Statistical Association; 1988:79–84.
75. McCullagh P, Nelder JA. *Generalized Linear Models*. London: Chapman Hall CRC; 1989.
76. Ng AY. Feature selection, l1 vs. l2 regularization, and rotational invariance. In: Proceedings of the Twenty-first International Conference on Machine Learning, ICML '04. New York, NY: ACM; 2004:78.
77. Natekin A, Knoll A. Gradient boosting machines, a tutorial. *Front Neurorobot*. 2013;7:21.
78. Springenberg JT, Klein A, Falkner S, Hutter F. Bayesian optimization with robust bayesian neural networks. In: Lee DD, Sugiyama M, Luxburg UV, Guyon I, Garnett R, eds. *Advances in Neural Information Processing Systems 29*. Red Hook, NY: Curran Associates Inc; 2016:4134–4142.
79. Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. *Neural Netw*. 1989;2:359–366.
80. Vidal R, Bruna J, Giryes R, Soatto S. Mathematics of deep learning. CoRR abs/1712.04741; 2017. arXiv:1712.04741.
81. Brahma PP, Wu D, She Y. Why deep learning works: a manifold dis-entanglement perspective. *IEEE Trans Neural Netw Learn Syst*. 2016;27:1997–2008.
82. Hochreiter S. Untersuchungen zu dynamischen neuronalen netzen. Diploma, Technische Universität München 91; 1991.
83. Goller C, Kuchler A. Learning task-dependent distributed representations by backpropagation through structure. In: IEEE International Conference on Neural Networks, Vol. 1. IEEE; 1996: 347–352.
84. Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37, ICML'15. JMLR.org; 2015: 448–456.
85. Clevert AD, Unterthiner T, Hochreiter S. Fast and accurate deep network learning by exponential linear units (ELUs). arXiv preprint arXiv:1511.07289; 2015.
86. Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res*. 2014;15:1929–1958.
87. Ruder S. An overview of gradient descent optimization algorithms. CoRR abs/1609.04747; 2016. arXiv:1609.04747.
88. Jain LC, Medsker LR. *Recurrent Neural Networks: Design and Applications*, 1st edn. Boca Raton, FL: CRC Press Inc; 1999.
89. Elman JL. Finding structure in time. *Cogn Sci*. 1990;14:179–211.
90. Jordan MI. Chapter 25 - serial order: A parallel distributed processing approach. In: Neural-Network Models of Cognition, Advances in Psychology, Vol. 121, edited by John W. Donahoe and Vivian Packard Dorsel. North-Holland; 1997: 471–495.
91. Cho K, van Merrienboer B, Gulcehre C, et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. arXiv e-prints, arXiv:1406.1078; 2014. arXiv:1406.1078 [cs.CL].
92. Vaswani A, Shazeer N, Parmar N. Attention is all you need. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, eds. *Advances in Neural Information Processing Systems 30*. Red Hook: Curran Associates Inc; 2017:5998–6008.
93. Luong M-T, Pham H, Manning CD. Effective approaches to attention-based neural machine translation. CoRR abs/1508.04025;2015, arXiv:1508.04025.
94. Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate; 2014. cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation.
95. Kelvin X, Ba J, Kiros R, et al. Show, attend and tell: neural image caption generation with visual attention. CoRR abs/1502.03044; 2015. arXiv:1502.03044.
96. Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, LNCS, Vol. 9351. Berlin: Springer; 2015:234–241. Available on arXiv:1505.04597 [cs.CV].
97. Radford A, Metz L, Chintala S. *Unsupervised representation learning with deep convolutional generative adversarial networks*; 2015. Cite arxiv:1511.06434Comment: Under review as a conference paper at ICLR 2016.
98. Mirza M, Osindero S. Conditional generative adversarial nets. CoRR abs/1411.1784; 2014. arXiv:1411.1784.
99. Chen X, Duan Y, Houthooft R, Schulman J, Sutskever I, Abbeel P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In: D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds. *Advances in Neural Information Processing Systems 29*. Red Hook: Curran Associates Inc; 2016:2172–2180.
100. Zhu J-Y, Park T, Isola P, Efros AA. Unpaired image-to-image translation using cycle-consistent adversarial networks. CoRR abs/1703.10593; 2017. arXiv:1703.10593.
101. Emami H, Dong M, Nejad-Davarani S, Glide-Hurst C. Generating synthetic CTS from magnetic resonance images using generative adversarial networks. *Med Phys*. 2018;45:3627–3636.
102. Sutton RS, McAllester DA, Singh SP, Mansour Y. Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems; 2000:1057–1063.
103. Efron B, Tibshirani R. *An Introduction to the Bootstrap, Monographs on Statistics and Applied Probability*. New York: Chapman & Hall; 1993: xvi, 436 p.
104. Collins GS, Reitsma JB, Altman DG, Moons KGM. Transparent reporting of a multivariable prediction model for individual prognosis or diagnosis (TRIPOD): the TRIPOD statement. *Ann Intern Med*. 2015;162:55–63.
105. Ratcliffe S. Roy amara; 2016.
106. Hau K. We analyzed 16,625 papers to figure out where AI is headed next. Technical Report; 2019.
107. Luo Y, Tseng H-H, Cui S, Wei L, Ten Haken RK, El Naqa I. Balancing accuracy and interpretability of machine learning

approaches for radiation treatment outcomes modeling. *BJR-Open 1*. 2019;1:20190021.
108. Tannenbaum C, Ellis RP, Eyssel J, Zou F, Schiebinger L. Sex and gender analysis improves science and engineering. *Nature*. 2019;575:137–146.
109. Manrai AK, Funke BH, Rehm HL, et al. Genetic misdiagnoses and the potential for health disparities. *N Engl J Med*. 2016;375:655–665.
110. Conforti F, Pala L, Bagnardi V, et al. Cancer immunotherapy efficacy and patients' sex: a systematic review and meta-analysis. *Lancet Oncol*. 2018;19:737–746.
111. Ghorbani A, Zou J. Data shapley: Equitable valuation of data for machine learning; 2019. arXiv:1904.02868 [stat.ML].
112. Abouelmehdi K, Beni-Hessane A, Khaloufi H. Big healthcare data: preserving security and privacy. *J Big Data*. 2018;5:1.