Deep learning in next-generation sequencing

Bertil Schmidt and Andreas Hildebrandt

Institut für Informatik, Johannes Gutenberg University Mainz, Germany

Next-generation sequencing (NGS) methods lie at the heart of large parts of biological and medical research. Their fundamental importance has created a continuously increasing demand for processing and analysis methods of the data sets produced, addressing questions such as variant calling, metagenomic classification and quantification, genomic feature detection, or downstream analysis in larger biological or medical contexts. In addition to classical algorithmic approaches, machine-learning (ML) techniques are often used for such tasks. In particular, deep learning (DL) methods that use multilayered artificial neural networks (ANNs) for supervised, semisupervised, and unsupervised learning have gained significant traction for such applications. Here, we highlight important network architectures, application areas, and DL frameworks in a NGS context.

Introduction

Recent years have seen a tremendous increase in the volume of data generated in the life sciences, propelled especially by the rapid progress of high-throughput sequencing (HTS) technologies, such as Illumina systems. Low sequencing cost per genome (www.genome.gov/sequencingcosts) renders large projects feasible, which increases the statistical power provided by larger data sets.

The abundance of generated genomic and transcriptomic sequencing data allows us to address a plethora of questions that could not be answered previously, with numerous applications in precision and personalized medicine as well as drug discovery. Examples include more reliable transcriptomic studies, which can detect disease-related expression patterns, or expression differences as the adverse effect of drug treatment [1,2]. Better variant calling and genome-wide association studies (GWAS) can have tremendous importance for personalized medicine [3]. Improved metagenomics can yield crucial information about ecosystems, such as the human gut microbiome, bacterial populations in hospitals, or viral evolution in patients. NGS methods are also a cornerstone of research into epigenetics associated with a variety of diseases [4]. They also have a role in understanding the origins and epidemiology of severe acute respiratory syndrome coronavirus 2 (SARS-CoV2) [5].

The enormous growth in data continuously shifts the life sciences from a model-driven toward a data-driven science. As in other fields of science, we now see an increasing adoption of methods from statistical learning that are not 'seeded' with a biological model in mind, but rather attempt to learn directly from data. In particular, DL methods based on multilayered ANNs have increased in popularity.

ANNs have a long history in ML, reaching back to Rosenblatt's biologically inspired Perceptron model [6] from 1958. Early ANNs were severely restricted in their size (number of layers and number of neurons per layer) and, consequently, were unable to compete with other popular ML techniques on complex tasks. However, groundbreaking improvements in parallel and vectorized computer hardware, such as graphics processing units (GPUs), in neural network design, and in algorithmic approaches to training of ANNs led to the development of so-called 'deep learning' methods, where ANNs typically contain large numbers of hidden layers. Such deep networks have allowed revolutionary results in fields such as computer vision, voice or text recognition.

We are already starting to see the adoption of several cutting-edge DL techniques for the analysis of NGS data. They have shown to be successful for a variety of important problems and, thus,

Corresponding authors: Schmidt, B. (bertil.schmidt@uni-mainz.de),

1359-6446/© 2020 Elsevier Ltd. All rights reserved.
https://doi.org/10.1016/j.drudis.2020.10.002

could become pivotal as the application of NGS continues to transcend from bench to bedside.

Deep learning

ML models attempt to solve specific tasks without being told explicitly what to do in detail. Instead, ML methods make use of available data related to the task at hand to build statistical models. In supervised ML methods, model inference makes use of labeled training data, where data points are annotated with the 'true' outcome. This allows for the inference of models for the relationship between input and output on this training data set, which then need to be validated on independent test and/or validation data sets to prevent overfitting to the distribution of the training data set. By contrast, unsupervised ML methods are trained from unlabeled data samples. Important examples are clustering and anomaly detection techniques.

The field of ML has generated numerous approaches for the different kinds of learning task. Interestingly, Wolpert's no free lunch theorem for ML [7] states that every possible ML algorithm has the same error rate on previously unobserved data points when averaged over all data-generating distributions. Hence, if no additional assumptions on the data-generating process can be made, no ML method can be assumed universally better or worse than any other on previously unobserved data. In practice, however, this is unrealistic, because prior knowledge about the problem domain will typically render some data-generating distributions more likely than others. Such assumptions can be formulated as priors on the data-generating process. One prior that is common is the smoothness or local constancy prior [8], stating that similar input data points should feature similar output values. Although this is obviously realistic in many application scenarios, relying purely on smoothness requires the availability of large amounts of training data to exploit it. Roughly speaking, if smoothness is all we can rely on to infer values, we need training examples close to each data point for which we want to perform inference. Thus, to successfully build ML models requires knowledge about the expected characteristics of the data-generating distribution on the one hand, and the ability to encode such knowledge in the model on the other.

DL models in general encode one particular additional prior on the data-generating process: the idea that the relation of input to output is expressed in terms of the hierarchical composition of multiple simpler functions or building blocks. This admittedly mild assumption already provides considerably more structure than a simple local constancy prior. In addition, further assumptions about the prior can be encoded through the topology of the deep neural networks.

The idea of learning relevant features as the (potentially multi-level and hierarchical) composition of simpler features turns out to be one of the most important reasons for the popularity of DL methods. Although most learning approaches require sophisticated feature engineering [i.e., the identification and preparation of informative input features that need to be optimized (mostly manually) for each problem domain], deep networks often identify highly useful features themselves. Their hierarchical nature further allows for the composition of more advanced or refined features from simpler ones, such as building corners and contours from edges, which are themselves identified from raw input pixels.

Once a deep network has identified such hierarchical features for a certain problem domain, it is often possible to transfer the feature representation (i.e., several layers of the deep network) to a new ML problem in the same domain. This has tremendous impact on the practical use of deep networks: training such networks from scratch not only places great demands on the compute infrastructure, but also typically requires very large training data sets. After all, modern deep networks easily have tens of millions (or even billions) of parameters that need to be determined [9]. However, a large percentage of those parameters participate in the build-up of features of interest, transferring the raw data into a representation more amenable to regression or classification tasks.

Here, we briefly review the most important ANN architectures (Fig. 1) in the context of NGS applications.

Artificial neurons

The basic building block of ANNs are artificial neurons (Fig. 1a), which are simply scalar functions of $n+1$ input variables x_i, $0 \le i \le n$. These inputs are processed and fed into a nonlinear function known as the activation or transfer function.

Most activation functions are applied to a weighted sum $s := \sum_{i=0}^{n} w_i x_i = w^t x$ of the input vector x. Usually, x_0 is kept fixed as $x_0 := 1$, such that w_0 takes the role of a bias term in s. The sum s is then fed into the activation function $\varphi(x;\theta)$ to yield the output of the neuron: $y := \varphi(w^t x;\theta)$. In this model, the weights w are treated as additional parameters. Together, the weight vector w and the parameters θ of the activation function form the set of unknowns of the neuron that are learned as part of the training process.

Three kinds of activation function are most frequent in practice. The first are biologically inspired approximations of the action potential. These functions are typically sigmoidal in nature, and are usually modeled by a hyperbolic tangent as $\varphi(x) := \tanh(x)$. The second kind of activation function is either linear or almost linear, such as the popular rectified linear units or ReLU [10], which are defined as $\varphi(x) := \max\{x, 0\}$. The third kind of popular activation function, the so-called softmax-function [11,12], is neither scalar nor does it use a weighted sum of the input variables. Instead, it models a probability distribution for k different outcomes or classes according to the Boltzmann distribution $\varphi(x)_i := \frac{e^{x_i}}{\sum_{i=1}^{n} e^{x_i}}$.

Historically, sigmoidal activation functions used to be the most popular choice for all neurons in the network. Today, they are usually only used in the final output layer. The main reason for this choice is their behavior during gradient-based training, in that sigmoidal activation functions tend to saturate easily into either an on- or off-state where gradients vanish. Hence, training such functions is challenging. As a result, the default activation function for inner layers of a deep network today is usually a variant of ReLUs, whereas (truly) linear, sigmoidal, or softmax-functions mostly occur in the output layer.

Deep network architectures

Deep networks are created by combining several artificial neurons into a common topology. Conceptually, the neurons are organized into layers, where the output of one layer is fed into (at least) one other layer. Layers below the final output layer are also known as hidden layers.

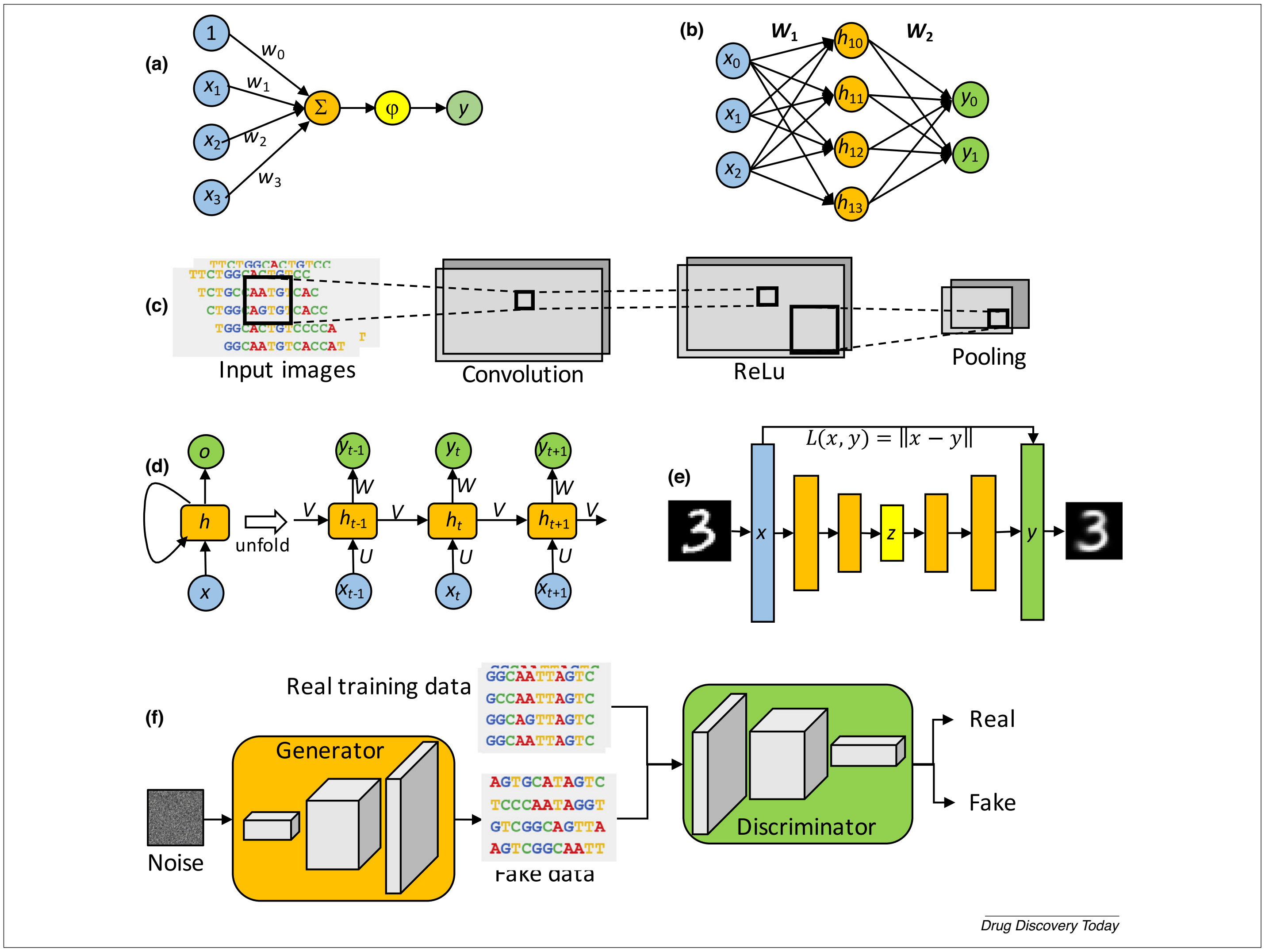

FIGURE 1

Overview of ANN architectures: **(a)** An artificial neuron maps an input vector x_i, $0 \le i \le n$, to a scalar output y by applying a nonlinear activation function φ to a weighted sum $s := \sum_{i=0}^{n} w_i x_i = w^t x$. **(b)** A multilayer perceptron (MLP) comprising an input layer, a fully connected hidden layer, and an output layer. **(c)** A single layer of a convolutional neural network (CNN), where matrix multiplication is replaced by a convolution with a small filter kernel matrix, the entries of which are learned during training followed by a ReLu activation function and (max)pooling. **(d)** Recurrent neural networks (RNNs) feature feedback connections to earlier layers and can be trained to learn time-dependent relations. **(e)** Autoencoders (AEs) are designed to identify useful data encodings in an unsupervised setting. **(f)** Generative adversarial networks (GANs) train two networks simultaneously. The generator produces new data points, whereas the discriminator classifies data points as either genuine or fake.

Feed-forward networks

In the simplest kind of deep network architecture, the so-called feed-forward networks or multilayer perceptrons (MLPs), information only flows forward along the layer stack. Figure 1b shows an MLP with a single hidden layer. In the context of NGS data, such networks are typically not directly used on DNA/RNA sequence data but on derived features or summaries in tabular form, such as *k*-mer histograms. In addition, they are also often an essential part of more complex networks, such as final layers of convolutional neural networks (CNNs) or recurrent neural networks (RNNs).

Convolutional neural networks

In CNNs, the matrix multiplication in at least some of the layers is replaced by a convolution with a (typically rather small) filter kernel matrix the entries of which are learned during training. Given that convolution allows for the replacement of values by weighted averages over their spatial neighborhood, it enables the encoding of information about spatial structures or patterns. Convolution is often followed by an operation known as pooling, which combines the results of several convolutions into a single output. For example, max-pooling replaces a set of convolution results by their maximum value. This allows for a certain degree of invariance; for instance, in computer vision, max-pooling convolution with the same filter kernel over several neighboring pixels can yield translation invariance. Give that it is often useful to compute the same convolutions at each input value (meaning that the same spatial pattern is compared with each input point), CNNs mostly use parameter sharing where several neurons are constrained to use the same filter kernel.

Figure 1c illustrates the design of a single convolutional layer of a CNN. In the context of analyzing NGS data, CNNs are often applied to pile-up images of multiple sequencing read alignments

TABLE 1

Summary of DL methods for the analysis of NGS data in the four selected application areas

Application area	Method	NN architecture	Framework	Refs
Variant calling	Deep Variant	CNN	Nucleus and TF	[14]
	NeuSomatic	CNN	PyTorch	[17]
	Clairvoyante	CNN	TF	[15]
	Clair	RNN	TF	[16]
	DeepSC	CNN	TF	[17]
	CNNScoreVariants	CNN	TF	[18]
Metagenomics	DeepMicrobes	LSTM	TF	[22]
	seq2species	CNN	TF	[23]
	GeNet	CNN	TF	[24]
	Meta^2	MIL	–	[25]
Transcriptomics	AutoImpute	AE	TF	[27]
	DCA	AE	Keras and TF	[28]
	scScope	AE	TF	[29]
	scvis	AE	TF	[30]
	scDeepCluster	AE	Keras and TF	[31]
	DeepImpute	MLP	Keras and TF	[32]
	scIGain	GAN	PyTorch	[33]
Epigenetics	DeepCpG	CNN & RNN	Keras and Theano	[34]
	MRCNN	CNN	Keras and TF	[35]
	FunDMDeep-m^{6}A	CNN	Keras and TF	[36]
	DeepSEA	CNN	Torch	[38]
	DeepBind	CNN	C++ and Python	[39]
	DanQ	CNN & LSTM	Keras and TF	[40]
	DeepHistone	CNN	PyTorch	[41]
	DeepLift	CNN	Keras and TF	[52]

to detect localized patterns for identifying mutations or sequence motifs.

Recurrent neural networks

In contrast to feedforward networks, RNNs feature feedback connections to earlier layers (Fig. 1d). These kinds of network can be trained to learn not only single points, but also sequences, with important applications in areas such as handwriting or voice recognition. Networks aimed at such tasks need the ability to carry information over to later parts of the sequence to identify nonlocal relationships. General RNNs can in principle carry the information infinitely far. However, this renders training challenging for reasons of numerical stability. Long short-term memory (LSTM) networks solve this problem partially by organizing the network into cells with input, output, and forget gates. Information flows into these cells, is processed, and produces an output, but collected information can be forgotten by exiting through the forget gate.

In the context of NGS, RNNs are often used for comparing sequencing reads to entire genomes or genome collections (e.g. for metagenomic read classification), but are also successfully used by base callers for long-read sequencing technologies.

Autoencoders

Autoencoders. (AEs) are particular network architectures designed to identify useful data encodings in an unsupervised setting. The general idea involves simultaneously learning encoding and decoding (Fig. 1e). Training attempts to yield as faithful a reconstruction as possible while simultaneously respecting desirable properties of the encoding, such as resistance to noise or sparsity. AEs are frequently used for unsupervised learning tasks in the area of (single-cell) RNA-seq, with examples including imputation of missing data, cell clustering, and visualization.

Generative adversarial networks

In addition to regression and classification tasks, deep networks can be used to generate virtual or simulated data sets. A popular architecture for this task are generative adversarial networks (GANs), where two networks are trained simultaneously. The first of these networks (the generator) produces new data points whereas the second network (the discriminator) classifies data points as either genuine or generated by the generator. During training, the objective of the generator is to create data points that the discriminator cannot distinguish from the input data sample, whereas the discriminator is trained to recognize generator results. Figure 1f illustrates the architecture of a GAN. Using GANs for NGS data analytics is still new but we are starting to see the first applications in the area of single-cell RNA-seq data imputation.

Applications

We identified four important application areas (variant calling, metagenomics, single-cell transcriptomics, and epigenetics) where the analysis of NGS data based on DL has already shown great potential. The reviewed methods are summarized in Table 1 and highlighted in the following sections.

Variant calling

Variant calling aims to detect genomic variants directly from NGS data. The simplest type of variant are point mutations (single nucleotide variants; SNVs). Typical SNV-calling applications include single nucleotide polymorphism (SNP) genotyping and detecting somatic SNVs within an individual using multiple tissue samples. The identification of more complex genetic variations, such as long indels or copy number variations (CNVs), from NGS data is also important but typically more difficult.

The computational variant-calling process relies on the initial alignment of NGS reads to a given reference genome sequence. Traditional algorithms, such as the GATK [13], apply numerous statistical models and hand-tuned heuristics to predict the likelihood of variation at each position, for instance based on quality scores and allele counts of aligned reads at that position. However, sequencing errors typically depend on alignment positions and the instrument types. This renders the task of designing accurate statistical models to distinguish variants from sequencing errors or alignment artifacts difficult. Using a supervised DL model for variant calling can replace the need for handcrafted heuristics by a system that has the ability to learn characteristic patterns directly from data.

The basic idea is to transform aligned reads centered on the location of interest (candidate variant) into images and apply CNNs known from solving computer vision tasks. This choice is motivated by the fact that inputs can be viewed as images of read pileups and that the complex dependence of variants on neighboring aligned reads could be modeled by convolutional kernels. DeepVariant [14] pioneered this approach for calling SNPs and small indels using the well-known Inception-v3 network architecture, which outputs genotype likelihoods for the candidate location (i.e., homozygous reference, heterozygous or homozygous alternative). The authors showed that it outperformed a variety of state-of-the-art classical methods for calling variants on an unseen Ashkenazi male sample (NA24385 for the PrecisionFDA Truth Challenge) when trained on another sample (Genome-in-a-Bottle, NA12878). Furthermore, generalization of this approach to other instruments and organisms (mouse) could be shown.

This idea paved the way for other architectures that are more specifically optimized towards DNA data. Clairvoyante [15] uses a multitask CNN for single-molecule sequencing data requiring around one order-of-magnitude fewer parameters compared with DeepVariant and achieves particularly good performance for long-read technologies. Its successor, Clair [16], further improves accuracy by using an RNN comprising bidirectional LSTMs followed by feedforward network layers and a softmax activation function that outputs likelihoods of various types of indel and SNP.

Neusomatic [17] focuses on somatic variant calling from tumor samples. Instead of using read pile-ups as images, this approach uses features extracted from each alignment column as inputs to the neural network. This leads to a simplified CNN architecture with lower computational complexity for both training and inference, which can still outperform traditional somatic mutation detection methods. CNNScoreVariants [18] proposes some further optimizations of the network architecture and has already been integrated in the new version of GATK. We expect to see more DL methods being developed for the detection of complex structural variations, as has already been proposed by DeepSV [19] for calling long deletions.

Metagenomics

Metagenomics deals with sequencing data obtained from environmental samples. An important processing step is taxonomic read assignment. Classical approaches, such as Kraken [20] or MetaCache [21], simply count exact *k*-mer matches to a reference database to assign a read to a likely species of origin. This problem can also be defined as a supervised learning task by considering each species of interest as the output category and NGS reads as the input. The size of reference genomes (which is significantly longer compared with short reads) and the large number of classes (typically many thousands for bacterial samples) make this classification problem complex and more suitable for RNN architectures. DeepMicrobes [22] investigated the performance of several supervised network models and found that a bidirectional LSTM with self-attention mechanism using *k*-mer embedding surpassed CNNs for this task because of its ability to model taxonomic signatures. CNNs have been applied for the (simpler task of) classification of 16S rRNA reads [23] and for representation learning from metagenomics long reads [24]. However, taxonomic classification of (short) whole-genome shotgun sequencing reads is more challenging because the ANN needs to learn many genome-wide patterns during training, whereas only information from a short genomic fragment is used for inference. Recent work has also extended this approach to the task of abundance estimation from a metagenomics sample by formulating it as a multiple instance-learning problem [25]. However, current DL approaches are still significantly slower than classical *k*-mer counting procedures. Furthermore, the size of their utilized reference genome databases is usually limited. Thus, the design of fast yet highly accurate methods for large-scale reference databases continues to be of relevance in this area.

Single-cell transcriptomics

Single-cell RNA-seq (scRNA-seq) protocols are gaining increasing attention because of their ability to profile the transcriptome of thousands or even millions of cells in a single assay. The input to corresponding analysis tasks is typically a gene expression matrix, which can be computed by mapping reads to a reference transcriptome resulting in an estimation of the expression of each gene within each cell [26]. Clustering is of central importance to identify putative cell types. This is challenging because of large data set sizes, high dimensionalities, and dropout events (because of lowly expressed genes leading to many zero entries in the expression matrix), which motivates the application of unsupervised ML methods for clustering, imputation, and dimensionality reduction.

DL techniques in this area are currently dominated by various types of AE because of their ability to efficiently learn inherent distributions using dimension reduction in an unsupervised manner. Several imputation methods (AutoImpute [27], DCA [28], and scScope [29]) based on this approach have been proposed that can outperform state-of-the-art classical approaches. scvis [30] and scDeepCluster [31] provide generative models based on AEs to compute low-dimensional embeddings while preserving global structure of the high-dimensional measurements that are used for clustering.

It would be interesting to apply other unsupervised methods besides AEs, as has already been investigated in recent work. DeepImpute [32] imputes genes in a divide-and-conquer approach by constructing multiple sub-neural networks, whereas scIGAIN [33] uses a GAN to build a generative model of the data.

Epigenetics

Quantitative detection of epigenetic signals, such as DNA/RNA methylation or histone modification, from NGS data is an important technique for understanding many biological processes. For example, methylation levels can be measured by using bisulfite sequencing (BS-seq) data, whereas histone modifications are often profiled based on chromatin immunoprecipitation followed by sequencing (ChIP-seq) data.

DeepCpG [34] uses a joint DL module based on a CNN (DNA module) and a bidirectional gated RNN (CpG module). CpG methylation is predicted from both local DNA sequence windows and observed neighboring methylation states determined from read counts mapped to a reference genome. Whereas the DNA module is designed to detect motifs, the CpG module compresses patterns of CpG states into a feature vector. MRCNN [35] claims to be more precise for this task by using a CNN. The detection of other methylation signatures from specifically developed transcriptomic sequencing protocols using ML techniques, such as m^{6}A [36] or m^{1}A [37], is also becoming increasingly popular and we expect to see more DL solutions in this area.

DeepSEA [38] and DeepBind [39] were the first to apply CNNs to modeling the sequence specificity of protein binding from large-scale chromatin-profiling data successfully. Instead of handcrafting feature sets, this approach can learn informative sequence features automatically and achieves superior performance to conventional methods. Subsequently, a hybrid model named DanQ [40] was designed for that task and combines a CNN with an LSTM. More recently, DeepHistone [41] proposed a joint neural network module similar to the DeepCpG approach

Deep learning frameworks

Early DL approaches for analyzing NGS data, such as DeepBind, had to be built from scratch, which can be challenging. Fortunately, excellent frameworks for the implementation of deep networks now exist that greatly facilitate the network development. The right column of Table 1 shows the utilized frameworks for the methods reviewed earlier.

Such frameworks provide common activation functions, handle gradient computation, training, and usually feature optimized implementations for different accelerator architectures, such as GPUs or tensor processing units (TPUs). Tensorflow [42] (TF) has been the most popular framework in the context of NGS and genomics. Although TF is designed to be general and applicable to a variety of computational tasks, it is mostly aimed at ML, in particular at deep network architectures. Currently, TF is mostly seen as a low-level framework. Thus, for many application scenarios that do not require such low-level access, higher-level abstractions based on user-friendly APIs, such as Keras [43], are used, as is the case for many of the newer published DL methods for transcriptomics and epigenetics. The PyTorch framework [44] offers similar functionality but is less popular than TF in the area of NGS. Furthermore, several other frameworks (rarely used in the context of NGS) are currently available (e.g., CNTK [45], Theano [46], or Flux [47]).

Recently, large Cloud providers have also started to offer DL platforms. This includes facilities for distributed computation of TF- or PyTorch-jobs, but also offerings such as Google's Cloud AutoML or Amazon's SageMaker, which promise mostly automated model training. In these packages, end-users can typically, using an intuitive graphical user interface, upload the training data, choose what kind of ML to perform, and highlight the property that is to be inferred. The framework will then automatically build an appropriate ML model and train it on the provided data. Although such automated models are currently rather restricted compared with handcrafted approaches designed by a specialist, they already serve as a means to commodify the generation of learning methods.

With the growing popularity of DL methods for analyzing sequencing data, several software frameworks and packages specifically designed for bioinformatics data have been introduced recently. Libraries, such as Nucleus [48] or Janggu [49], can be used alongside Keras, TF, or PyTorch. They offer dedicated objects for processing biological sequence data, which makes it easy to read, write, analyze, and visualize data in common genomics file formats, such as BAM, FASTA, bigWig, VCF, or BED.

Discussion, outlook, and concluding remarks

The success of DL in computer vision and speech/language processing has motivated the applications of these methods in bioinformatics. Processing of NGS data is of particular importance because corresponding methods lie at the heart of biomedical research, with important applications to many areas, including drug discovery. Consequently, researchers are actively designing and using deep neural networks for corresponding tasks. CNNs, RNNs, and LSTMs are already used in practice based on supervised learning, such as variant calling, metagenomics, or epigenetics. Unsupervised tasks (in particularly for RNA expression data) are currently dominated by AEs, but we are beginning to see the application of more recent architectures, such as GANs.

The intricate connections in deep networks lead to complex and interwoven interactions between input features and eventual outcomes. Often, the nets lead to hierarchical structures of interactions. These are crucial, on the one hand, for the automated learning of representations, and for the connected field of transfer learning. On the other hand, they often render a straightforward interpretation of the influence of input features on model outcome, which is possible for many alternative ML approaches, infeasible. For example, although, for linear approaches, the coefficients of the trained model globally decide the importance of input features on model outcome, deep networks require a more involved analysis. Hence, feature importance scores in DL are usually derived for each input separately and, thus, yield a local answer about feature importance rather than a global one. One way to generate such importance scores for a given input involves perturbing it in a systematic manner (i.e., in the case of genomic sequences, mutating nucleotides, introducing gaps, etc.). The change in outcome then yields information about the importance of the perturbed feature at this point in a high-dimensional input space. Although this approach is very general, the number of perturbations required to yield useful feature importance scores grows rapidly. Alternatively, feature importance scores can be derived directly from a backpropagation pass through the model (e.g., [50]). Although this approach is considerably faster and more scalable than input perturbation, many backpropagation-based feature importance scores suffer from problems arising from neuron saturation [51]. Some methods, such as DeepLIFT [50] try to avoid this and related problems. A recent example in the context of NGS is the design of an interpretation tool that learns predictive motif representations for cooperative transcription factor binding interactions from ChIP-nexus sequencing data using DeepLift [52].

However, there is not yet a consensus on which feature importance scoring methods work best in which scenarios.

As discussed in this review, DL for the analysis of NGS data has great potential. So much potential, in fact, that it is sometimes used as a hammer that makes all problems look like nails. Contrary to this trend, we do not believe that DL will replace the need for classical sequence algorithmic. Instead, ML approaches in general, and DL in particular, will be most useful in scenarios where there is insufficient knowledge to model the system or process of interest accurately. Many classical ML approaches rely on an initial definition of suitable features that are informative of the output. This feature engineering can be complex in practice, and often requires a similar level of understanding of the problem domain as building a non-ML model would. One of the main advantages of DL is that suitable network architectures are often able to perform much of the feature engineering themselves by learning suitable feature representations from the raw data as part of their training. Furthermore, such representations can often be transferred to other ML tasks on the same input data, such that training a sequence-based model for motif detection on a large data set, for example, can help to train sequence-based models for other tasks for which there are fewer data available. ML and classical modeling do not have to exclude each other. Recently, the tight integration of scientific models with ML machinery, sometimes called scientific machine learning, has received considerable attention.

As NGS methods become increasingly important for drug discovery [53,54], DL techniques for NGS data analysis can be expected to have a similar impact as they have had in other fields. In drug development pipelines, the target discovery and validation stages as well as clinical trials and postapproval work are most affected by NGS technologies. Important applications here include targeted cancer sequencing to understand genomic variation [55], RNA sequencing to study differential expression [56], multiplex panels for companion diagnostics [57], or even drug development against gene signatures, as in the case of pembrolizumab [58]. In all these applications, properties of interest are predicted, using statistical means, from input features that include NGS data. Hence, they are ideally suited to profit from DL methodology.

Although ML approaches and their combination with classical modeling have many advantages, it is also important to be aware of the significant risk of overfitting. DL approaches are often so versatile and powerful that they can easily adapt to all input distributions they are presented with, but this does not mean that they generalize well to unseen data points. In the context of NGS, this can be illustrated by the example of models trained with NGS data derived from certain species, sequencing technologies, coverage, and read length that do not maintain their accuracy across data from different sequencing technologies, prep methods, and species. Thus, model training needs to be tightly monitored and controlled to avoid bias.

In summary, the application of ML shifts biomedical research from model-driven towards data-driven science. Given that sequencing data grow at an enormous rate, we expect to see an increasing adoption of DL methods, which do not require to be seeded with a biological model in mind, but rather attempt to learn directly from the data. Furthermore, progress in AI techniques continues to be dynamic. For example, recent work demonstrated that training very large transformer models, such as Bidirectional Encoder Representations from Transformers (BERT) or Megatron can significantly advance the state-of-the-art in language processing. We are already seeing initial applications of BERT for biomedical text mining (BioBERT [59]). Thus, the availability of tools built on top of frameworks such as TF that allow for the rapid design and integration of (new) AI techniques into bioinformatics workflows will be vital for the life sciences as a whole, and for drug discovery in particular.

Declaration of Competing Interest

The authors declare that they have no known competing financial interest or personal relationship that could have appeared to influence the work reported in this paper.

Acknowledgments

The Federal Ministry of Education and Research (BMBF) supported this work through the project MetaDL.

References

1 Mavrou, A. *et al.* (2015) Serine–arginine protein kinase 1 (SRPK1) inhibition as a potential novel targeted therapeutic strategy in prostate cancer. *Oncogene* 34, 4311–4319

2 Stephens, Z.D. *et al.* (2015) Big data: astronomical or genomical? *PLoS Biol.* 13, e1002195

3 Harper, A.R. and Topol, E.J. (2012) Pharmacogenomics in clinical practice and drug development. *Nat. Biotechnol.* 30, 1117–1124

4 Heerboth, S. *et al.* (2014) Use of epigenetic drugs in disease: an overview. *Genet. Epigenet.* 6, 9–19

5 Tang, X. *et al.* (2020) On the origin and continuing evolution of SARS-CoV-2. *National Sci. Rev.* 7, 1012–1023

6 Rosenblatt, F. (1958) The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol. Rev.* 65, 386

7 Wolpert, D.H. (1996) The lack of a priori distinctions between learning algorithms. *Neural Comput.* 8, 1341–1390

8 Goodfellow, I. *et al.* (2016) *Deep Learning*. MIT Press

9 Alom, M.Z. *et al.* (2019) A state-of-the-art survey on deep learning theory and architectures. *Electronics* 8, 292

10 Hahnloser, R.H. *et al.* (2000) Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature* 405 (6789), 947–951

11 Bridle, J.S. (1990) Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing* (Soulié, F.F. and Hérault, J., eds), pp. 227–236, Springer

12 Bridle, J.S. (1990) Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. *Adv. Neural Inf. Process. Syst.* 2, 211–217

13 DePristo, M.A. *et al.* (2011) A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nat. Genetics* 43, 491

14 Poplin, R. *et al.* (2018) A universal SNP and small-indel variant caller using deep neural networks. *Nat. Biotechnol.* 36, 983–987

15 Luo, R. *et al.* (2019) A multi-task convolutional deep neural network for variant calling in single molecule sequencing. *Nat. Commun.* 10, 998

16 Luo, R. *et al.* (2020) Exploring the limit of using a deep neural network on pileup data for germline variant calling. *Nat. Mach. Intell.* 2, 220–227

17 Sahraeian, S.M.E. *et al.* (2019) Deep convolutional neural networks for accurate somatic mutation detection. *Nat. Commun.* 10, 1041

18 Friedman, S. *et al.* (2020) Lean and deep models for more accurate filtering of SNP and INDEL variant calls. *Bioinformatics* 36, 2060–2067

19 Cai, L. *et al.* (2019) DeepSV: accurate calling of genomic deletions from high-throughput sequencing data using deep convolutional neural network. *BMC Bioinf.* 20, 665

20 Wood, D.E. and Salzberg, S.L. (2014) Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol.* 15, 1–12

21 Müller, A. *et al.* (2017) MetaCache: context-aware classification of metagenomic reads using minhashing. *Bioinformatics* 33, 3740–3748

22 Liang, Q. *et al.* (2020) DeepMicrobes: taxonomic classification for metagenomics with deep learning. *NAR Genomics Bioinf.* 2, lqaa009

23 Busia, A. *et al.* (2019) A deep learning approach to pattern recognition for short DNA sequences. *bioRxiv* 2019, 353474

24 Rojas-Carulla, M. *et al.* (2019) Genet: deep representations for metagenomics. *arXiv* arXiv:1901, 11015

25 Georgiou, A. *et al.* (2019) Deep multiple instance learning for taxonomic classification of metagenomic read sets. *arXiv* arXiv:1909, 13146

26 Niebler, S. *et al.* (2020) RainDrop: rapid activation matrix computation for droplet-based single-cell RNA-seq reads. *BMC Bioinf.* 21, 1–14

27 Talwar, D. *et al.* (2018) AutoImpute: autoencoder based imputation of single-cell RNA-seq data. *Sci. Rep.* 8, 16329

28 Eraslan, G. *et al.* (2019) Single-cell RNA-seq denoising using a deep count autoencoder. *Nat. Commun.* 10, 390

29 Deng, Y. *et al.* (2018) Massive single-cell RNA-seq analysis and imputation via deep learning. *bioRxiv* 2018, 315556

30 Ding, J. *et al.* (2018) Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. *Nat. Commun.* 9, 2002

31 Tian, T. *et al.* (2019) Clustering single-cell RNA-seq data with a model-based deep learning approach. *Nat. Mach. Intell.* 1, 191–198

32 Arisdakessian, C. *et al.* (2019) DeepImpute: an accurate, fast, and scalable deep neural network method to impute single-cell RNA-seq data. *Genome Biol.* 20, 211

33 Xu, Y. *et al.* (2020) scIGANs: single-cell RNA-seq imputation using generative adversarial networks. *Nucleic Acids Res.* 48, e85

34 Angermueller, C. *et al.* (2017) DeepCpG: accurate prediction of single-cell DNA methylation states using deep learning. *Genome Biol.* 18, 67

35 Tian, Q. *et al.* (2019) MRCNN: a deep learning model for regression of genome-wide DNA methylation. *BMC Genomics* 20, 192

36 Zhang, S.Y. *et al.* (2019) FunDMDeep-m6A: identification and prioritization of functional differential m6A methylation genes. *Bioinformatics* 35, i90–i98

37 Schmidt, L. *et al.* (2019) Graphical workflow system for modification calling by machine learning of reverse transcription signatures. *Front. Genetics* 10, 876

38 Zhou, J. and Troyanskaya, O.G. (2015) Predicting effects of noncoding variants with deep learning–based sequence model. *Nat. Methods* 12, 931–934

39 Alipanahi, B. *et al.* (2015) Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nat. Biotechnol.* 33, 831–838

40 Quang, D. and Xie, X. (2016) DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. *Nucleic Acids Res.* 44, e107

41 Yin, Q. *et al.* (2019) DeepHistone: a deep learning approach to predicting histone modifications. *BMC Genomics* 20, 11–23

42 Abadi, M. *et al.* (2016) Tensorflow: large-scale machine learning on heterogeneous distributed systems. *arXiv* arXiv:1603, 04467

43 Chollet, F. (2018) *Keras: The Python Deep Learning Library*. Astrophysics Source Code Library

44 Paszke, A. *et al.* (2019) Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* 32, 8026–8037

45 Seide, F. and Agarwal, A. (2016) CNTK: Microsoft's open-source deep-learning toolkit. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Wallach, H. *et al.* eds), In p. 2135, Curran Associates, Inc.

46 Al-Rfou, R. *et al.* (2016) Theano: a Python framework for fast computation of mathematical expressions. *arXiv* arXiv:1605, 02688v1

47 Innes, M. *et al.* (2018) Fashionable modelling with Flux. *arXiv* arXiv:1811, 01457

48 Baid, G. *et al.* (2019) *Using Nucleus and TensorFlow for DNA Sequencing Error Correction.* Medium

49 Kopp, W. *et al.* (2020) Deep learning for genomics using Janggu. *Nat. Commun.* 11, 3488

50 Shrikumar, A. *et al.* (2017) Learning important features through propagating activation differences. *arXiv* arXiv:1704, 02685

51 Eraslan, G. *et al.* (2019) Deep learning: new computational modelling techniques for genomics. *Nat. Rev. Genet.* 20, 389–403

52 Avsec, *et al.* (2020) Base-resolution models of transcription factor binding reveal soft motif syntax. *bioRxiv* 2020, 737981

53 Schmidt, B. and Hildebrandt, A. (2017) Next-generation sequencing: big data meets high performance computing. *Drug Discovery Today* 22, 712–717

54 Torshizi, A.D. and Wang, K. (2018) Next-generation sequencing in drug development: target identification and genetically stratified clinical trials. *Drug Discovery Today* 23, 1776–1783

55 Bewicke-Copley, F. *et al.* (2019) Applications and analysis of targeted genomic sequencing in cancer studies. *Comput. Struct. Biotechnol. J.* 17, 1348–1359

56 Khatoon, Z. *et al.* (2014) Introduction to RNA-Seq and its applications to drug discovery and development. *Drug Dev. Res.* 75, 324–330

57 Cunha, K.S. *et al.* (2016) Hybridization capture-based next–generation sequencing to evaluate coding sequence and deep intronic mutations in the NF1 gene. *Genes* 7, 133

58 Raedler, L.A. (2015) Keytruda (pembrolizumab): first PD-1 inhibitor approved for previously treated unresectable or metastatic melanoma. *Am. Health Drug Benefits* 8, 96

59 Lee, J. *et al.* (2020) BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 36, 1234–1240