Deep learning: new computational modelling techniques for genomics

Gökcen Eraslan[1,2,5], *Žiga Avsec*[3,5], *Julien Gagneur*[3]* and *Fabian J. Theis*[1,2,4]*

Abstract | As a data-driven science, genomics largely utilizes machine learning to capture dependencies in data and derive novel biological hypotheses. However, the ability to extract new insights from the exponentially increasing volume of genomics data requires more expressive machine learning models. By effectively leveraging large data sets, deep learning has transformed fields such as computer vision and natural language processing. Now, it is becoming the method of choice for many genomics modelling tasks, including predicting the impact of genetic variation on gene regulatory mechanisms such as DNA accessibility and splicing.

Feature
An individual, measurable property or characteristic of a phenomenon being observed.

Handcrafted features
Features derived from raw data (or other features) using manually specified rules. Unlike learned features, they are specified upfront and do not change during model training. For example, the GC content is a handcrafted feature of a DNA sequence.

[1]*Institute of Computational Biology, Helmholtz Zentrum München, Neuherberg, Germany.*

[2]*School of Life Sciences Weihenstephan, Technical University of Munich, Freising, Germany.*

[3]*Department of Informatics, Technical University of Munich, Garching, Germany.*

[4]*Department of Mathematics, Technical University of Munich, Garching, Germany.*

[5]*These authors contributed equally: Gökcen Eraslan, Žiga Avsec.*

**e-mail: gagneur@in.tum.de; fabian.theis@helmholtz-muenchen.de*

https://doi.org/10.1038/s41576-019-0122-6

Genomics, in the broad sense, also referred to as functional genomics, aims to characterize the function of every genomic element of an organism by using genome-scale assays such as genome sequencing, transcriptome profiling and proteomics[1]. Genomics arose as a data-driven science — it operates by discovering novel properties from explorations of genome-scale data rather than by testing preconceived models and hypotheses[2]. Applications of genomics include finding associations between genotype and phenotype[3], discovering biomarkers for patient stratification[4], predicting the function of genes[5] and charting biochemically active genomic regions such as transcriptional enhancers[6].

Genomics data are too large and too complex to be mined solely by visual investigation of pairwise correlations. Instead, analytical tools are required to support the discovery of unanticipated relationships, to derive novel hypotheses and models and to make predictions. Unlike some algorithms, in which assumptions and domain expertise are hard coded, machine learning algorithms are designed to automatically detect patterns in data[7,8]. Hence, machine learning algorithms are suited to data-driven sciences and, in particular, to genomics[9,10]. However, the performance of machine learning algorithms can strongly depend on how the data are represented, that is, on how each variable (also called a feature) is computed. For instance, to classify a tumour as malign or benign from a fluorescent microscopy image, a preprocessing algorithm could detect cells, identify the cell type and generate a list of cell counts for each cell type. A machine learning model would then take these estimated cell counts, which are examples of handcrafted features, as input features to classify the tumour. A central issue is that classification performance depends heavily on the quality and the relevance of these features. For example, relevant visual features such as cell morphology, distances between cells or localization within an organ are not captured in cell counts, and this incomplete representation of the data may reduce classification accuracy. Deep learning, a subdiscipline of machine learning, addresses this issue by embedding the computation of features into the machine learning model itself to yield end-to-end models[11]. This outcome has been realized through the development of deep neural networks, machine learning models that consist of successive elementary operations, which compute increasingly more complex features by taking the results of preceding operations as input. Deep neural networks are able to improve prediction accuracy by discovering relevant features of high complexity, such as the cell morphology and spatial organization of cells in the above example.

The construction and training of deep neural networks have been enabled by the explosion of data, algorithmic advances and substantial increases in computational capacity, particularly through the use of graphical processing units (GPUs)[12]. Over the past 7 years, deep neural networks have led to multiple performance breakthroughs in computer vision[13–15], speech recognition[16] and machine translation[17]. Seminal studies in 2015 demonstrated the applicability of deep neural networks to DNA sequence data[18,19] and, since then, the number of publications describing the application of deep neural networks to genomics has exploded. In parallel, the deep learning community has substantially improved method quality and expanded its repertoire of modelling techniques, some of which are already starting to impact genomics.

Here, we describe deep learning modelling techniques and their existing genomic applications. We start by presenting four major classes of neural networks (fully connected, convolutional, recurrent and graph convolutional) for supervised machine learning and explain how they can be used to abstract patterns

End-to-end models
Machine learning models that embed the entire data-processing pipeline to transform raw input data into predictions without requiring a preprocessing step.

Deep neural networks
A wide class of machine learning models with a design that is loosely based on biological neural networks.

Fully connected
Referring to a layer that performs an affine transformation of a vector followed by application of an activation function to each value.

Convolutional
Referring to a neural network layer that processes data stored in *n*-dimensional arrays, such as images. The same fully connected layer is applied to multiple local patches of the input array. When applied to DNA sequences, a convolutional layer can be interpreted as a set of position weight matrices scanned across the sequence.

Recurrent
Referring to a neural network layer that processes sequential data. The same neural network is applied at each step of the sequence and updates a memory variable that is provided for the next step.

Graph convolutional
Referring to neural networks that process graph-structured data; they generalize convolution beyond regular structures, such as DNA sequences and images, to graphs with arbitrary structures. The same neural network is applied to each node and edge in the graph.

Autoencoders
Unsupervised neural networks trained to reconstruct the input. One or more bottleneck layers have lower dimensionality than the input, which leads to compression of data and forces the autoencoder to extract useful features and omit unimportant features in the reconstruction.

Generative adversarial networks
(GANs). Unsupervised learning models that aim to generate data points that are indistinguishable from the observed ones.

common in genomics. Next, we describe multitask learning and multimodal learning, two modelling techniques suited to integrating multiple data sets and data types. We then discuss transfer learning, a technique that enables rapid development of new models from existing ones, and techniques to interpret deep learning models, which are both crucial for genomics. We finish with a discussion of two unsupervised learning techniques, autoencoders and generative adversarial networks (GANs), which first found application in single-cell genomics. To facilitate the adoption of deep learning by the genomics community, we provide pointers to code that ease rapid prototyping. For further background on deep learning, we refer readers to the deep learning textbook[11]. As complementary reading, we recommend a hands-on primer[20] and several reviews that provide a broader perspective on deep learning, target computational biologists and cover applications of deep learning beyond genomics[21–25].

Supervised learning

The goal of supervised learning is to obtain a model that takes features as input and returns a prediction for a so-called target variable. An example of a supervised learning problem is one that predicts whether an intron is spliced out or not (the target) given features on the RNA such as the presence or absence of the canonical splice site sequence, the location of the splicing branchpoint or intron length (FIG. 1). Training a machine learning model refers to learning its parameters, which typically involves minimizing a loss function on training data with the aim of making accurate predictions on unseen data (BOX 1).

Complex dependencies can be modelled with deep neural networks. For many supervised learning problems in computational biology, the input data can be represented as a table with multiple columns, or features, each of which contains numerical or categorical data that are potentially useful for making predictions. Some input data are naturally represented as features in a table (such as temperature or time), whereas other input data need to be first transformed (such as DNA sequence into *k*-mer counts) using a process called feature extraction to fit a tabular representation. For the intron-splicing prediction problem, the presence or absence of the canonical splice site sequence, the location of the splicing branchpoint and the intron length can be preprocessed features collected in a tabular format. Tabular data are standard for a wide range of supervised machine learning models, ranging from simple linear models, such as logistic regression[8], to more flexible nonlinear models, such as neural networks and many others[26–29]. Logistic regression is a binary classifier, that is, a supervised learning model that predicts a binary target variable. Specifically, logistic regression predicts the probability of the positive class by computing a weighted sum of the input features mapped to the [0,1] interval using the sigmoid function, a type of activation function. The parameters of logistic regression, or other linear classifiers that use different activation functions, are the weights in the weighted sum. Linear classifiers fail when the classes, for instance, that of an intron spliced out or not, cannot be well discriminated with a weighted sum of input features (FIG. 1a).

To improve predictive performance, new input features can be manually added by transforming or combining existing features in new ways, for example, by

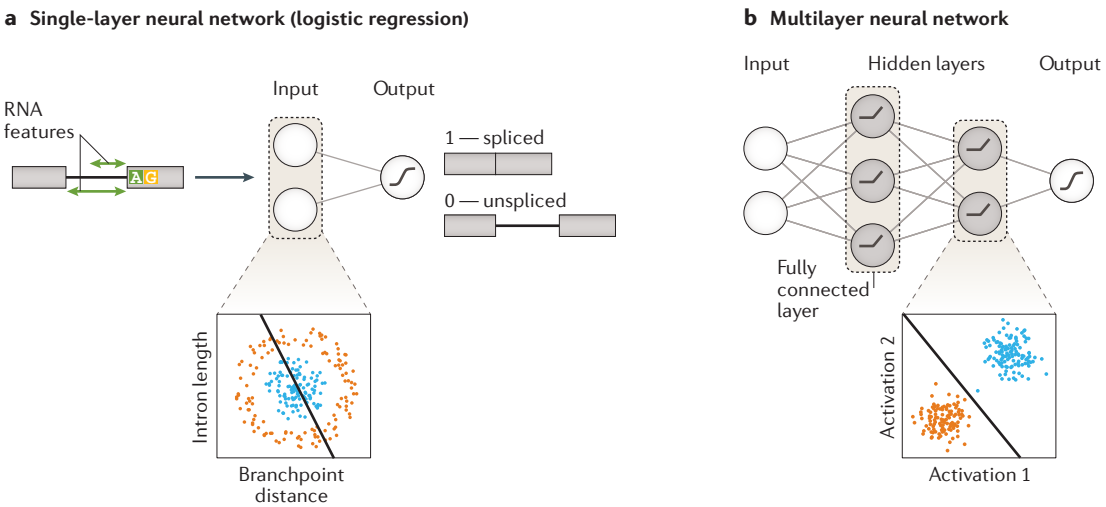


Fig. 1 | **Neural networks with hidden layers used to model nonlinear dependencies. a** | Shown is an example of splice site classification based on two RNA features. Depicted is a single-layer neural network with sigmoid activation function, which corresponds to logistic regression. It predicts the probability of the output being class 1 using a weighted sum (also called linear combination) of the input that is mapped to the [0,1] interval with a sigmoid function. In this example, the aim is to discriminate spliced-out from not-spliced-out introns as a function of the length of the intron and of the distance of the branchpoint to the acceptor site. If either the intron length or the branchpoint distance is too short or too long, splicing will not occur. Hence, linear combinations of these two features, as implemented in logistic regression, cannot separate the spliced (blue) from unspliced (orange) data points. **b** | Neural networks with intermediate layers, also called hidden layers, transform the inputs using intermediate nonlinear transformations into a space where the classes become linearly separable. The depicted layers are said to be fully connected because every neuron receives input from all neurons of the upstream layer. Deep neural networks are neural networks with many hidden layers.

Target
The desired output used to train a supervised model.

Loss function
A function that is optimized during training to fit machine learning model parameters. In the simplest case, it measures the discrepancy between predictions and observations. In the case of quantitative predictions such as regression, mean-squared error loss is frequently used, and for binary classification, the binary cross-entropy, also called logistic loss, is typically used.

***k*-mer**
Character sequence of a certain length. For instance, a dinucleotide is a *k*-mer for which $k=2$.

Logistic regression
A supervised learning algorithm that predicts the log-odds of a binary output to be of the positive class as a weighted sum of the input features. Transformation of the log-odds with the sigmoid activation function leads to predicted probabilities.

Sigmoid function
A function that maps real numbers to [0,1], defined as $1/(1+e^{-x})$.

Activation function
A function applied to an intermediate value *x* within a neural network. Activation functions are usually nonlinear yet very simple, such as the rectified-linear unit or the sigmoid function.

Regularization
A strategy to prevent overfitting that is typically achieved by constraining the model parameters during training by modifying the loss function or the parameter optimization procedure. For example, the so-called L2 regularization adds the sum of the squares of the model parameters to the loss function to penalize large model parameters.

Hidden layers
Layers are a list of artificial neurons that collectively represents a function that take as input an array of real numbers and returns an array of real numbers corresponding to neuron activations. Hidden layers are between the input and output layers.

Box 1 | **Training neural networks for supervised learning**

Data partitioning and prediction goal

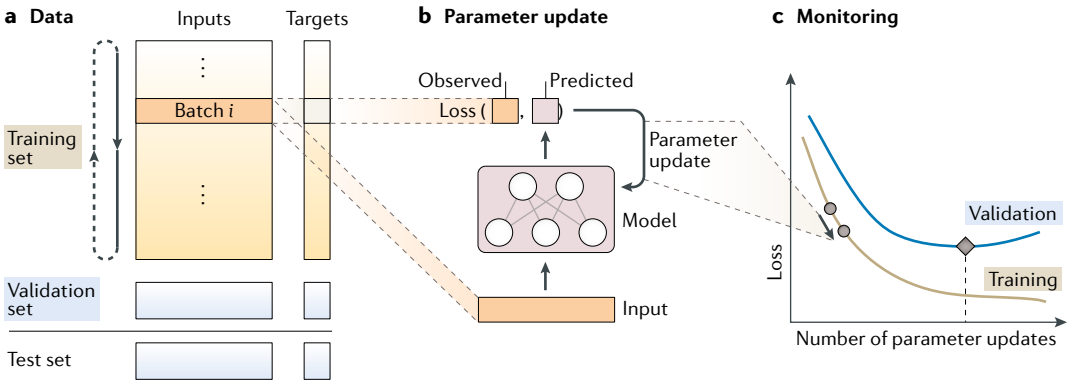
A supervised learning data set consists of input–target pairs split into three distinct sets (see the figure, part **a**): one for optimizing the parameters of the model (training set), one for evaluating the model performance (validation set) and one for the final assessment of the best developed model (test set). During the model development phase, one only has access to the training and validation set. The goal is to develop a model with the most accurate predictions on the test set. The accuracy of predictions is measured by different evaluation metrics such as the Pearson correlation coefficient or Spearman correlation coefficient for regression, area under the receiver operator curve for balanced binary classification or area under the precision-recall curve for imbalanced binary classification[157]. We note that the validation set and test set should be carefully chosen to represent truly unseen samples. For DNA-based models, this is typically implemented by leaving out complete chromosomes or all measurements in new cell types rather than randomly sampling the regions from the genome.

Fitting the parameters using the training set

Parameters of the neural network are first randomly initialized and then iteratively refined using a method called the stochastic gradient descent or its variations[158,159]. Specifically, small random subsets, so-called batches, of input–target pairs of the training data set are iteratively used to make small updates on model parameters in trying to minimize the loss function between the predicted values and the observed targets (see the figure, part **b**). This minimization is performed by using the gradient of the loss function computed using the backpropagation algorithm[160,161]. There are two main benefits to taking only a small random subset of the training set at each optimization step rather than the full training set. First, the algorithm requires a constant amount of memory regardless of the data set size, which allows models to be trained on data sets much larger than the available memory. Second, the random fluctuations between batches were demonstrated to improve the model performance by regularization[162,163]. As operations in neural networks including backpropagation involve matrix operations, graphical processing units (GPUs) can massively parallelize those operations and hence speed up model training by up to two orders of magnitude compared with normal central processing units[12]. In practice, specifying and training neural networks are achieved through the use of deep learning frameworks (BOX 2).

Choosing the hyperparameters using the validation set

The training process is monitored by regularly evaluating the loss or the evaluation metric on the validation data set (see the figure, part **c**). When the metric stops improving or even starts degrading, training is stopped as the model starts to overfit the data. To improve the model performance on the validation data set, the modeller can adjust different hyperparameters, such as the number of layers of the network or batch size, and train a new model. This loop of experimenting with different hyperparameters can be automated using a simple random search[164] or other hyperparameter optimization techniques[165–168]. Finally, after the modeller is satisfied with the performance on the validation set, the generalization performance of the best model or an ensemble of best models is evaluated on a completely separate test set.



We refer interested readers to the deep learning book for more details[11].

taking powers or pairwise products. Neural networks use hidden layers to learn these nonlinear feature transformations automatically. Each hidden layer can be thought of as multiple linear models with their output transformed by a nonlinear activation function, such as the sigmoid function or the more popular rectified-linear unit (ReLU). Together, these layers compose the input features into relevant complex patterns, which facilitates the task of distinguishing two classes (FIG. 1b). Deep neural networks use many hidden layers, and a layer is said to be fully connected when each neuron receives inputs from all neurons of the preceding layer. Neural networks are typically trained using stochastic gradient descent, an algorithm suited to training models on very large data sets (BOX 1). Implementation of neural networks using modern deep learning frameworks enables rapid prototyping with different architectures and data sets (BOX 2).

Fully connected neural networks have been used for a number of genomics applications, which include predicting the percentage of exons spliced in for a given sequence from sequence features such as the presence of binding motifs of splice factors or sequence conservation[30,31]; prioritizing potential disease-causing genetic variants[32]; and predicting *cis*-regulatory elements in a given genomic region using features such as chromatin marks, gene expression and evolutionary conservation[33,34]. Many of

Box 2 | **Example code for training neural networks**

Much of the success of deep learning can be attributed to deep learning frameworks such as Keras, TensorFlow[102] or PyTorch[101]. Deep learning frameworks are software libraries that implement the operations required for building and training neural networks, including matrix multiplication, convolution and automatic differentiation. This enables users to specify the model architecture by composing multiple building blocks — layers — without having to manually derive the gradients required during training (BOX 1). Below is an example that implements the architectures from Figures 1 and 2 using Keras.

```
import keras.layers as kl
from keras.models import Sequential

# Fully connected model architecture (Figure 1)
model = Sequential([
 kl.Dense(3, activation='relu', input_shape=(2,)),
 kl.Dense(2, activation='relu'),
 kl.Dense(1, activation='sigmoid')])

# Convolutional neural network architecture (Figure 2)
model = Sequential([
 kl.Conv1D(2, activation='relu', input_shape=(4, 30), padding='same'),
 kl.MaxPool(6),
 kl.Conv1D(3, activation='relu', padding='same'),
 kl.GlobalMaxPool(),
 kl.Dense(1, activation='sigmoid')])

# Specify optimizer, loss and evaluation metric
model.compile(optimizer='adam',
 loss='binary_crossentropy',
 metrics=['accuracy'])

# Load the dataset
x, y = load_dataset(...)

# Train the model for 10 epochs
model.fit(x, y, epochs=10)
```

Thanks to these frameworks, users can focus on designing the model architecture without having to manually derive the optimization procedure, which makes prototyping new model architectures easy and decouples the model choice from the optimization algorithm. Furthermore, the frameworks enable training of models on GPUs without using extra code. Moreover, as the specification of the architecture is standardized, models and model components can be easily exchanged.

We refer the reader to DragoNN for end-to-end examples of how to implement, train, evaluate and interpret convolutional neural network models based on DNA sequence using Keras.

these methods report improved predictive performance over methods such as linear regression, decision trees or random forests. However, it is important to note that, in many problems with tabular data, other methods such as gradient-boosted decision trees often outperform fully connected neural networks, as can be seen from the results of Kaggle machine learning competitions. Nevertheless, fully connected layers constitute an essential building block in the deep learning toolbox and can be effectively combined with other neural network layers, such as convolutional layers.

Convolutions discover local patterns in sequential data. Local dependencies in spatial and longitudinal data must be taken into account for effective predictions. For example, shuffling a DNA sequence or the pixels of an image severely disrupts informative patterns. These local dependencies set spatial or longitudinal data apart from tabular data, for which the ordering of the features is arbitrary. Consider the problem of classifying genomic regions as bound versus unbound by a particular transcription factor, in which bound regions are defined as high-confidence binding events in chromatin immunoprecipitation following by sequencing (ChIP–seq) data[35–39]. Transcription factors bind to DNA by recognizing sequence motifs. A fully connected layer based on sequence-derived features, such as the number of *k*-mer instances or the position weight matrix (PWM) matches in the sequence[40,41], can be used for this task. As *k*-mer or PWM instance frequencies are robust to shifting motifs within the sequence, such models could generalize well to sequences with the same motifs located at different positions. However, they would fail to recognize patterns in which transcription factor binding depends on a combination of multiple motifs with well-defined spacing. Furthermore, the number of possible *k*-mers increases exponentially with *k*-mer length, which poses both storage and overfitting challenges.

A convolutional layer is a special form of fully connected layer in which the same fully connected layer is applied locally, for example, in a 6 bp window, to all sequence positions. This approach can also be viewed as scanning the sequence using multiple PWMs[42–44], for example, for transcription factors GATA1 and TAL1 (FIG. 2a,b). By using the same model parameters across positions, the total number of parameters is drastically reduced, and the network is able to detect a motif at positions not seen during training. Each convolutional layer scans the sequence with several filters (FIG. 2b) by producing a scalar value at every position, which quantifies the match between the filter and the sequence. As in fully connected neural networks, a nonlinear activation function (typically ReLU) is applied at each layer (FIG. 2c). Next, a pooling operation is applied, which aggregates the activations in contiguous bins across the positional axis, typically taking the maximal or average activation for each channel (FIG. 2d). Pooling reduces the effective sequence length and coarsens the signal. The subsequent convolutional layer composes the output of the previous layer and is able to detect whether a GATA1 motif and TAL1 motif were present at some distance range (FIG. 2e,f). Finally, the output of the convolutional layers can be used as input to a fully connected neural network to perform the final prediction task (FIG. 2g,h). Hence, different types of neural network layers (for example, fully connected and convolutional) can be combined within a single neural network.

Three pivotal methods, DeepBind[18], DeepSEA[19] and Basset[45], were the first convolutional neural networks (CNNs) applied to genomics data. In DeepBind, multiple single-task models (the median number of parameters was 1,586) were trained to predict binarized in vitro and in vivo binding affinities (that is, bound or not bound) of a transcription factor and the in vitro binding affinity of an RNA-binding protein (RBP). The method consistently performed better than existing non-deep learning approaches. The DeepSEA model (52,843,119 parameters) predicted the presence or absence of 919 chromatin features, including transcription factor binding, DNA accessibility and histone modification given

Rectified-linear unit
(ReLU). Widely used activation function defined as max(0, *x*).

Neuron
The elementary unit of a neural network. An artificial neuron aggregates the inputs from other neurons and emits an output called activation. Inputs and activations of artificial neurons are real numbers. The activation of an artificial neuron is computed by applying a nonlinear activation function to a weighted sum of its inputs.

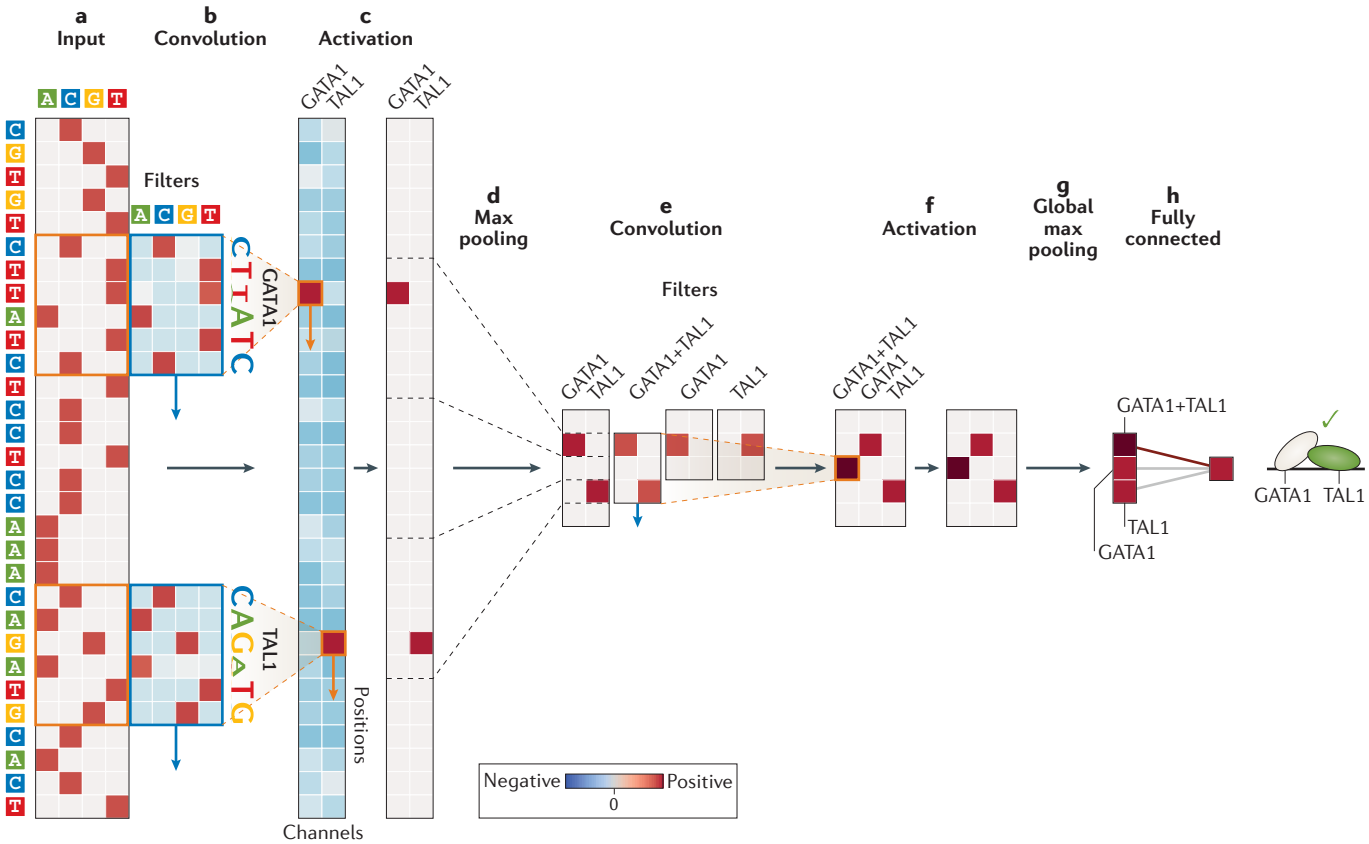


Fig. 2 | **Modelling transcription factor binding sites and spacing with convolutional neural networks.** The depicted convolutional neural network predicts the binding affinity of the TAL1–GATA1 transcription factor complex. **a** | One-hot encoded representation of the DNA sequence. **b** | The first convolutional layer scans the input sequence using filters, which are exemplified by position weight matrices of the GATA1 and TAL1 transcription factors. **c** | Negative values are truncated to 0 using the rectified-linear unit (ReLU) activation function. **d** | In the max pooling operation, contiguous bins of the activation map are summarized by taking the maximum value for each channel in each bin. **e** | The second convolutional layer scans the sequence for pairs of motifs and for instances of individual motifs. **f** | Similarly to that of the first convolution, ReLU activation function is applied. **g** | The maximum value across all positions for each channel is selected. **h** | A fully connected layer is used to make the final prediction.

a 1,000 bp sequence. Basset (4,135,064 parameters) predicted 164 binarized DNA accessibility features (for example, as accessible or inaccessible) given a 600 bp sequence. Both methods performed substantially better than the *k*-mer-based approach gkm-SVM[41].

Since their initial applications, CNNs have been applied to predict various molecular phenotypes on the basis of DNA sequence alone and have become the new state-of-the-art models. Applications include classifying transcription factor binding sites[46] and predicting molecular phenotypes such as chromatin features[47], DNA contact maps[48], DNA methylation[49,50], gene expression[51], translation effiency[52], RBP binding[53–55] and microRNA (miRNA) targets[56]. In addition to predicting molecular phenotypes from the sequence, CNNs have been successfully applied to more technical tasks traditionally addressed by handcrafted bioinformatics pipelines. For example, they have been utilized to predict the specificity of guide RNA[57], denoise ChIP–seq[58], enhance Hi-C data resolution[59], predict the laboratory of origin from DNA sequences[60] and call genetic variants[61,62].

CNNs have also been employed to model long-range dependencies in the genome[47]. Although interacting regulatory elements may be distantly located on the unfolded linear DNA sequence, these elements are often proximal in the actual 3D chromatin conformation. Hence, modelling molecular phenotypes from the linear DNA sequence, albeit a crude approximation of the chromatin, can be improved by allowing for long-range dependencies and allowing the model to implicitly learn aspects of the 3D organization, such as promoter–enhancer looping. In Basenji[47] this is achieved by using dilated convolutions, which enabled a receptive field of 32 kb to be achieved. Dilated convolutions have also allowed splice sites to be predicted from sequence using a receptive field of 10 kb, thereby enabling the integration of genetic sequence across distances as long as typical human introns[63].

Recurrent neural networks model long-range dependencies in sequences. Different types of neural network can be characterized by their parameter-sharing schemes. For example, fully connected layers have no parameter sharing (FIG. 3a), whereas convolutional layers impose translational invariance by applying the same filters at every position of their input (FIG. 3b).

Linear regression
A supervised learning algorithm that predicts the output as a weighted sum of the input features.

Decision trees
Supervised learning algorithms in which the prediction is made by making a series of decisions of type 'is feature *i* larger than *x*' (internal nodes of the tree) and then predicting a constant value for all points satisfying the same decisions series (leaf nodes).

Random forests
Supervised learning algorithms that train and average the predictions of many decision trees.

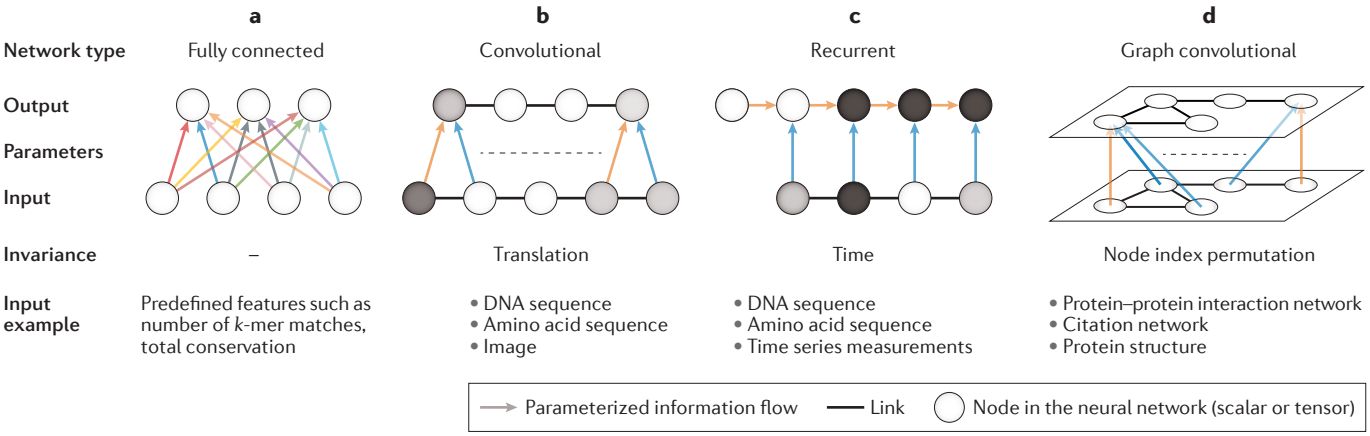


Fig. 3 | **Neural network layers and their parameter-sharing schemes.** Neural network architectures can be categorized into four groups based on their connectivity and parameter-sharing schemes. **a** | Fully connected layers assume that input features do not have any particular ordering and hence apply different parameters across different input features. **b** | Convolutional layers assume that local subsets of input features, such as consecutive bases in DNA, can represent patterns. Therefore, the connectivity and parameter-sharing pattern of convolutional layers reflect locality. **c** | Recurrent layers assume that the input features should be processed sequentially and that the sequence element depends on all the previous sequence elements. At each sequence element, the same operation is applied (blue and orange arrows), and the information from the next input sequence element is incorporated into the memory (orange arrows) and carried over. **d** | Graph convolutional networks assume that the structure of the input features follows the structure of a known graph. The same set of parameters is used to process all the nodes and thereby imposes invariance to node ordering. By exploiting the properties of the raw data, convolutional neural networks, recurrent neural networks and graph convolutional layers can have drastically reduced numbers of parameters compared with fully connected layers while still being able to represent flexible functions. The same colours indicate shared parameters, and arrows indicate the flow of information. Full lines indicate specific ordering or relationships between features represented as nodes (parts **a**–**d**).

Recurrent neural networks (RNNs)[64,65] are an alternative to CNNs for processing sequential data, such as DNA sequences or time series, that implement a different parameter-sharing scheme. RNNs apply the same operation to each sequence element (FIG. 3c). The operation takes as input the memory of the previous sequence element and the new input. It updates the memory and optionally emits an output, which is either passed on to subsequent layers or is directly used as model predictions. By applying the same model at each sequence element, RNNs are invariant to the position index in the processed sequence. For example, an RNN could detect an open reading frame in a DNA sequence regardless of the position in the sequence. This task requires the recognition of a certain series of inputs, such as the start codon followed by an in-frame stop codon. The main advantage of RNNs over CNNs is that they are, in theory, able to carry over information through infinitely long sequences via memory. Furthermore, RNNs can naturally process sequences of widely varying length, such as mRNA sequences. However, recent systematic comparisons show that CNNs combined with various tricks (such as dilated convolutions) are able to reach comparable or even better performances than RNNs on sequence-modelling tasks, such as audio synthesis and machine translation[66]. Moreover, because RNNs apply a sequential operation, they cannot be easily parallelized and are hence much slower to compute than CNNs.

In genomics, RNNs have been used to aggregate the outputs of CNNs for predicting single-cell DNA methylation states[50], RBP binding[67], transcription factor binding and DNA accessibility[68,69]. RNNs have also found applications in miRNA biology: deepTarget[70] performed better than existing models at predicting miRNA binding targets from mRNA–miRNA sequence pairs, and deepMiRGene[71] better predicted the occurrence of precursor miRNAs from the mRNA sequence and its predicted secondary structure than existing methods that use handcrafted features. Base calling from raw DNA-sequencing data is another prediction task for which RNNs have been applied. DeepNano[72] accurately predicted base identity from changes in electric current measured by the Oxford Nanopore MinION sequencer[73]. Despite these numerous applications of RNNs, we note that there is a lack of systematic comparison of recurrent and convolutional architectures for the common sequence-modelling tasks in genomics.

Graph-convolutional neural networks model dependencies in graph-structured data. Graph-structured data, including protein–protein interaction networks and gene regulatory networks, are ubiquitous in genomics[74,75]. Graph convolutional neural (GCN) networks[76–79] (FIG. 3d) use the individual features of nodes in a graph and the node connectivity to solve machine learning tasks on graphs. GCNs sequentially apply multiple graph transformations (layers), whereby each graph transformation aggregates features from the neighbouring nodes or edges in a nonlinear manner and represents nodes or edges with a new set of features. Tasks that GCNs can be trained for include node classification[80,81], unsupervised node embedding (which aims to find informative, low-dimensional representation of nodes)[80], edge classification and graph classification[79].

Gradient-boosted decision trees
Supervised learning algorithms that train multiple decision trees in a sequential manner; at each time step, a new decision tree is trained on the residual or pseudo-residual of the previous decision tree.

Position weight matrix
(PWM). A commonly used representation of sequence motifs in biological sequences. It is based on nucleotide frequencies of aligned sequences at each position and can be used for identifying transcription factor binding sites from DNA sequence.

Overfitting
The scenario in which the model fits the training set very well but does not generalize well to unseen data. Very flexible models with many free parameters are prone to overfitting, whereas models with many fewer parameters than the training data do not overfit.

GCNs have been applied to a number of biological and chemical problems. For instance, one method used an unsupervised approach to derive new features of proteins from protein–protein interaction networks in an unsupervised manner, and these features were then used to predict protein function in different tissues[82]. GCNs have also been used for modelling polypharmacy side effects[83]. In chemistry, graph convolutions have been successfully used to predict various molecular properties including solubility, drug efficacy and photovoltaic efficiency[84,85]. Genomic applications of GCNs include predicting binarized gene expression given the expression of other genes[86] or classification of cancer subtypes[87]. GCNs provide promising tools for exploiting structural patterns of graphs for supervised and unsupervised machine learning problems, and we expect to see more genomics applications in the future.

Sharing information across tasks and integrating data modalities. Genomic data often contain correlated measurements of related biological activities. Correlated measurements can occur within a single data type (such as the expression of co-regulated genes) or across different data types (such as ChIP–seq peaks and DNase I hypersensitive sites sequencing (DNase-seq) peaks) and give rise to related prediction tasks.

Consider an example in which we would like to predict transcription factor binding affinity for multiple transcription factors. Instead of building a single-task model for each prediction task (FIG. 4a), a multitask model can jointly predict binding of multiple transcription factors (FIG. 4b). In such models, the majority of layers are shared and branch out to task-specific layers at the end (FIG. 4b). Owing to co-binding and common protein domains of the modelled transcription factors, using the same layers to extract complex sequence features across multiple transcription factors might improve the predictive performance and require less data per transcription factor. Moreover, by sharing the computation between tasks, multitask models can make predictions faster than single-task models can.

In multitask models, the overall loss function is simply the sum of the losses for each task. When losses are very different across tasks, a weighted sum can be used to balance the losses[88]. Training multitask models can be challenging, as the network needs to simultaneously optimize multiple losses and hence make trade-offs. For example, if class imbalance varies greatly across tasks, the network might successfully learn only the well-balanced classes and completely ignore the difficult imbalanced classes by always predicting the majority class. Various strategies have been proposed to tackle this issue[88–91]. For example, GradNorm[88] adopts task weights during training that ensure the backpropagated gradients corresponding to different tasks will be of equal magnitude. In genomics, multitask models have been successfully used to simultaneously predict multiple molecular phenotypes such as those for transcription factor binding, different histone marks, DNA accessibility and gene expression in different tissues[19,45,47,51].

Analogously to multitask models, deep neural networks can be easily extended to take multiple data modalities as inputs in order to leverage complementary information between them. A simple way to integrate multiple data modalities is to concatenate features from each data set (often referred to as early integration). Such concatenation might not be possible with raw data when the data modalities are very different (such as a DNA sequence combined with an image or gene expression). Neural networks enable multiple data modalities to be integrated by first processing each data modality using dedicated layers, concatenating the outputs of dedicated layers and then using further layers to integrate the features extracted from each data modality (FIG. 4c). This approach, also known as intermediate integration, enables the most suitable dedicated layers to be used for each data modality and can hence extract more predictive features. Both early integration and intermediate integration approaches (individually or in combination) have been used by different neural network models in genomics. For example, DNA sequence, gene expression and chromatin accessibility have been integrated to predict transcription factor binding across cell types[69]. In addition, an RNA sequence has been integrated with RNA secondary structure[55] or distances to key genomic landmarks such as splice sites[54] to predict in vivo affinity of RBPs. Another example is the prediction of the pathogenicity of missense variants by integrating amino acid sequences with multiple conservation scores[92]. We refer the reader to Zitnik et al.[93] for more information on data integration with machine learning models.

Training models on small data sets with transfer learning. In the scenario in which data are scarce, training a model from scratch might not be feasible. Instead, the model can be initialized with the majority of parameters from another model trained on a similar task. This approach is called transfer learning[94] and can be viewed as incorporating prior knowledge into the model (FIG. 4d). In the simplest case, in which the parameters of the source model are not modified during training, this approach can be seen as building a separate model on top of features extracted by the source model. Transferred models can learn new tasks more rapidly, require less data to train and generalize better to unseen data than models trained from scratch using randomly initialized parameters[95]. In biological image analysis, pretrained models from the ImageNet competition[96] were successfully adopted to classify skin lesions[97], perform morphological profiling[98] and analyse in situ hybridization images[99,100]. In genomics, the utility of transfer learning has been demonstrated for sequence-based predictive models of chromatin accessibility[45]. In this study, researchers trained the multitask Basset model for predicting binary chromatin accessibility profiles of 149 cell types. They then trained single-task models of chromatin accessibility in 15 other cell types using parameters from the multitask model for initialization. The predictive performance was greater for models initialized with transferred parameters than for models initialized with random parameters[45]. We note that extensive evaluations of how many parameters to share and which models to use for different tasks are still lacking and will require further investigation.

Filters
Parameters of a convolutional layer. In the first layer of a sequence-based convolutional network, they can be interpreted as position weight matrices.

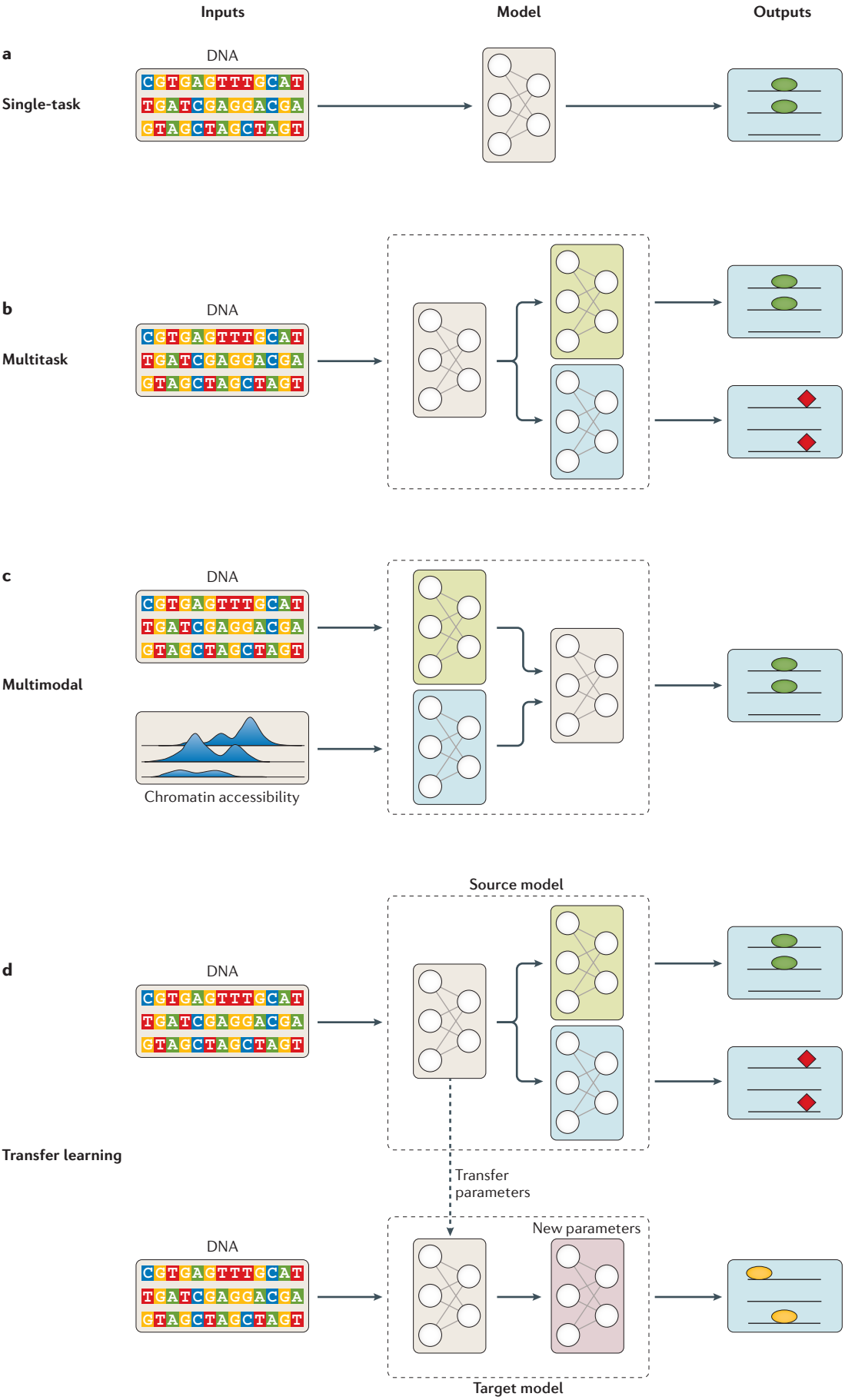
Pooling operation
A function that replaces the output at a certain location with a summary statistic of the nearby outputs. For example, the max pooling operation reports the maximum output within a rectangular neighbourhood.

Channel
An axis other than one of the positional axes. For images, the channel axis encodes different colours (such as red, green and blue), for one-hot-encoded sequences (A: [1, 0, 0, 0], C: [0, 1, 0, 0] and so on), it denotes the bases (A, C, G and T), and for the output of the convolutions, it corresponds to the outputs of different filters.

Dilated convolutions
Filters that skip some values in the input layers. Typically, each subsequent convolutional layer increases the dilation by a factor of two, thus achieving an exponentially increasing receptive field with each additional layer.

Receptive field
The region of the input that affects the output of a convolutional neuron.

Memory
An array that stores the information of the patterns observed in the sequence elements previously processed by a recurrent neural network.


Inputs
Model
Outputs
a
Single-task
DNA
b
Multitask
DNA
c
Multimodal
DNA
Chromatin accessibility
d
Transfer learning
DNA
Source model
Transfer parameters
DNA
New parameters
Target model

◀ Fig. 4 | **Multitask models, multimodal models and transfer learning. a** | Shown is a single-task model predicting the binding of a single transcription factor (green oval). **b** | A multitask model is shown that simultaneously predicts binding for two transcription factors (green oval and red diamond). There are three submodels depicted: a common submodel and two task-specific submodels. **c** | A multimodal model is shown that takes as input DNA sequence and chromatin accessibility. Each data modality is first processed using a dedicated submodel, and the outputs are concatenated and processed using the shared submodel. Parameters of all submodels are trained jointly as shown in both parts **b** and **c**. **d** | Transfer learning is presented. Parameters of the original model trained on a large data set (top) are used for initialization for the second model trained on a related task (target task) but with much less data available (bottom). In this example, the first task of the source model is similar to the target task (both are ovals); hence, the transferred submodel may contain features useful for the target task prediction.

To realize the full potential of transfer and multitask learning, trained models must be easily shared. In the fields of computer vision and natural language processing, trained models are shared through repositories called model zoos and are available for popular machine learning frameworks, for example PyTorch model zoos[101], Keras model zoos and Tensorflow model zoos[102]. We and others recently developed Kipoi, a model zoo for genomics[103], to address the lack of a platform for exchanging models. Kipoi contains over 2,000 predictive models for genomics and allows the user to access, apply and interpret these predictive models through a unified interface as well as score the effect of single-nucleotide variants for a subset of sequence-based models. As the size and the number of data sets grow and predictive models become more accurate and essential, we expect to see a greater emphasis on model distribution, similar to the improvement in data and software sharing over the past decade.

Explaining predictions

Although deep neural networks are not designed to highlight interpretable relationships in data or to guide the formulation of mechanistic hypotheses, they can nevertheless be interrogated for these purposes a posteriori[104]. We refer to these interrogations of the models as model interpretation. In simple models such as linear models, the parameters of the model often measure the contribution of an input feature to prediction. Therefore, they can be directly interpreted in cases where the input features are relatively independent. By contrast, the parameters of a deep neural network are difficult to interpret because of their redundancy and nonlinear relationship with the output. For example, although the CNN presented in FIG. 2 may be interpreted as multiple PWMs scanning the sequence, the filters representing the PWM in the first layer typically only represent parts of the motifs. The reason for this phenomenon is that individual filters are never forced to learn complete motifs. Rather, the network as a whole can detect motifs by assembling multiple filters in the downstream layers.

Feature importance scores interrogate input–output relationships. In complex models, it is imperative to inspect parameters indirectly by probing the input–output relationships for each predicted example. Feature importance scores, also called attribution scores, relevance scores or contribution scores, can be used for this purpose. They highlight the parts of a given input that are most influential for the model prediction and thereby help to explain why such a prediction was made (FIG. 5a). In DNA sequence-based models, the importance scores highlight sequence motifs and are hence widely used in genomics[18,45,47]. They can also be used to probe more complex epistatic interactions[105]. We refer to feature importance scores as scores generated per example, and they should not be confused with the feature importance for supervised models based on tabular data like those of random forests, which are aggregated across the entire data set.

Feature importance scores can be divided into two main categories on the basis of whether they are computed using input perturbations or using backpropagation. Perturbation-based approaches systematically perturb[18,19,45,106] the input features and observe the change in the output (FIG. 5b). For DNA sequence-based models, the induced perturbation can be, for example, a single-nucleotide substitution[18,19] or an insertion of a regulatory motif[15]. The main drawback of perturbation-based importance scores is the high computational cost, which becomes notable when the importance scores for the whole data set need to be computed. For example, a sequence of 1,000 nt requires an additional 3,000 model predictions to assess the effect of every possible single-nucleotide variant. Backpropagation-based approaches[107,108], to the contrary, are much more computationally efficient. In these approaches, importance scores for all the input features are computed using a single backpropagation pass through the network (FIG. 5c), and hence they require only twice the amount of computation as a single prediction. The simplest backpropagation-based importance scores are saliency maps[107] and input-masked gradients[108]. As deep learning frameworks support automatic differentiation (BOX 2), these scores can be efficiently implemented in a few lines of code.

One issue with saliency maps, input-masked gradients or perturbation-based methods is the so-called neuron saturation problem. Consider a neural network that classifies a sequence as positive if it observes a TAL1 transcription factor motif. If there are actually two TAL1 motifs in the sequence, one of them could be deleted, and the model prediction would not change. In the case of perturbation-based gradients or input-masked gradients, the importance scores would be low for both TAL1 motifs, as they are individually not necessary for the prediction. To address this failure mode, so-called reference-based methods like DeepLIFT[108] and integrated gradients[109] were developed. These methods compare the input features with their 'reference' values and thereby avoid the saturation issue. In the case of DNA sequences, a reasonable reference value is the dinucleotide-shuffled version of the original sequence. We note that a rigorous benchmark of feature importance scores and different reference values in genomics are currently lacking. Therefore, we recommend trying multiple methods and comparing them with some well-understood examples or simulated data.

Sequence motif discovery. Motif discovery is an essential component of the bioinformatics workflow when regulatory DNA sequences are analysed. Although feature importance scores are able to highlight the instances

Feature importance scores
The quantification values of the contributions of features to a current model prediction. The simplest way to obtain this score is to perturb the feature value and measure the change in the model prediction: the larger the change found, the more important the feature is.

Backpropagation
An algorithm for computing gradients of neural networks. Gradients with respect to the loss function are used to update the neural network parameters during training.

Saliency maps
Feature importance scores defined as the gradient absolute values of the model output with respect to the model input.

Input-masked gradients
Feature importance scores defined as the gradient of the model output with respect to the model input multiplied by the input values.

Automatic differentiation
A set of techniques, which consist of a sequence of elementary arithmetic operations, used to automatically differentiate a computer program.

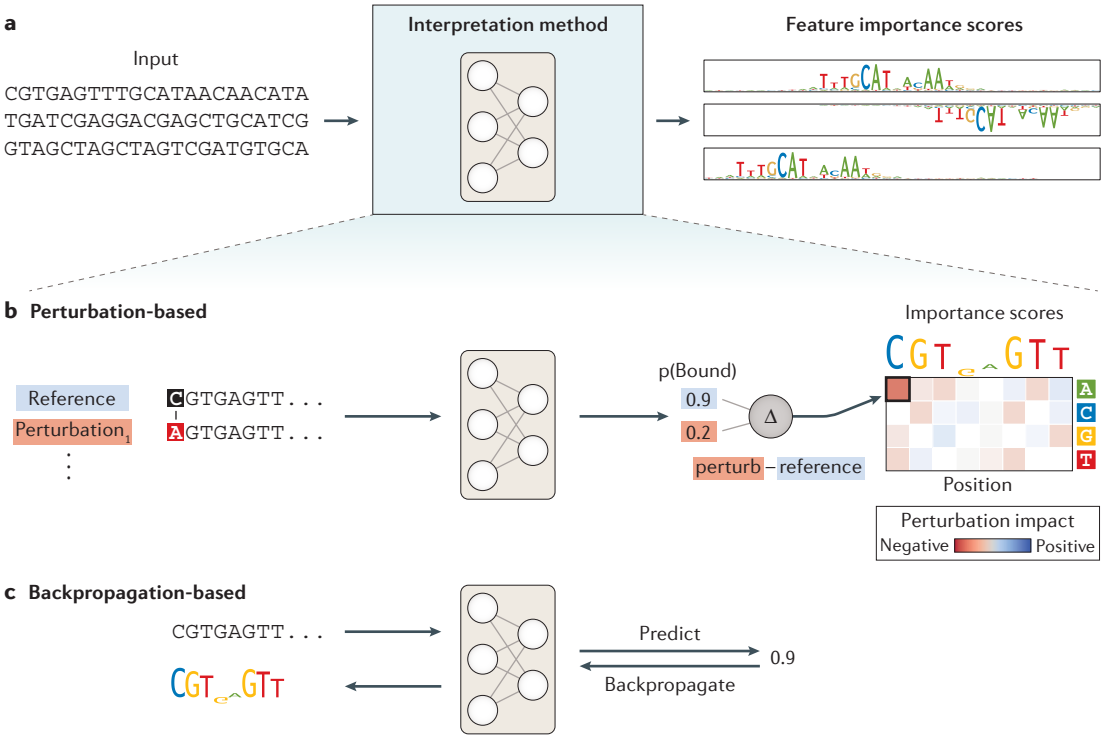


Fig. 5 | **Model interpretation via feature importance scores. a** | Feature importance scores highlight parts of the input most predictive for the output. For DNA sequence-based models, these can be visualized as a sequence logo of the input sequence, with letter heights proportional to the feature importance score, which may also be negative (as visualized by letters facing upside down). There are two classes of feature importance scores: perturbation-based approaches (part **b**) and backpropagation-based approaches (part **c**). **b** | Perturbation-based approaches perturb each input feature (left) and record the change in model prediction (centre) in the feature importance matrix (right). For DNA sequences, the perturbations correspond to single base substitutions[18]. Alternatively, the perturbation matrix can be visualized as a sequence logo with the letter heights corresponding to the average per-base perturbation impact. **c** | Backpropagation-based approaches compute the feature importance scores using gradients[107] or augmented gradients such as DeepLIFT[108] for the input features with respect to model prediction.

of different motifs[18,45,47,110], they have so far been used only to manually inspect individual sequences and not to perform automated motif discovery. Simply averaging the importance scores across multiple examples will not yield the desired results because the motif is not always located at the same position in the input sequence. Owing to this issue, many studies[45,49,50,54] have derived motifs from sequences by aggregating sequences in the training set that strongly activated filters of the first convolutional layer or interpreted filters directly as motifs[52]. Recently, a promising approach to aggregate the importance scores called TF-MoDISco was proposed[111]. TF-MoDISco extracts, aligns and clusters the regions of high importance into sequence motifs. Unlike classical motif discovery, which relies only on plain sequences, TF-MoDISco relies on the predictive model to highlight the important regions within the sequence via feature importance scores, which guides motif discovery.

Neural networks with interpretable parameters and activations. An approach termed 'visible neural networks' has recently been proposed with the DCell model[112] to improve the interpretability of internal neural network activations. The model architecture of DCell corresponds to the hierarchical organization of known molecular subsystems within the cell. Nodes in the neural network correspond to molecular subsystems, such as signalling pathways or large protein complexes, and connections between two nodes (systems) are only permitted if the upstream system (for example, a small protein complex) is part of the downstream system (such as a large protein complex). The neurons in the neural network correspond to known concepts; hence, their activations and parameters can be interpreted. We note that this approach is only feasible for tasks in which the underlying entities and their hierarchical structure are sufficiently well known; it may not be directly applicable to tasks for which the entities or their hierarchical structure are generally unknown, as in the case of transcription factor binding. It will be interesting to see to what extent this approach can be applied in the future to other models and also how it can be combined with modular modelling approaches (such as ExPecto[51]) to tackle predicting and understanding more complex phenotypes such as disease.

Unsupervised learning

The goal of unsupervised learning is to characterize unlabelled data by learning the useful properties of the data set. Classic unsupervised machine learning methods include clustering algorithms such as *k*-means and dimensionality reduction methods such

Model architecture
The structure of a neural network independent of its parameter values. Important aspects of model architecture are the types of layers, their dimensions and how they are connected to each other.

k-means
An unsupervised method for partitioning the observations into clusters by alternating between refining cluster centroids and updating cluster assignments of observations.

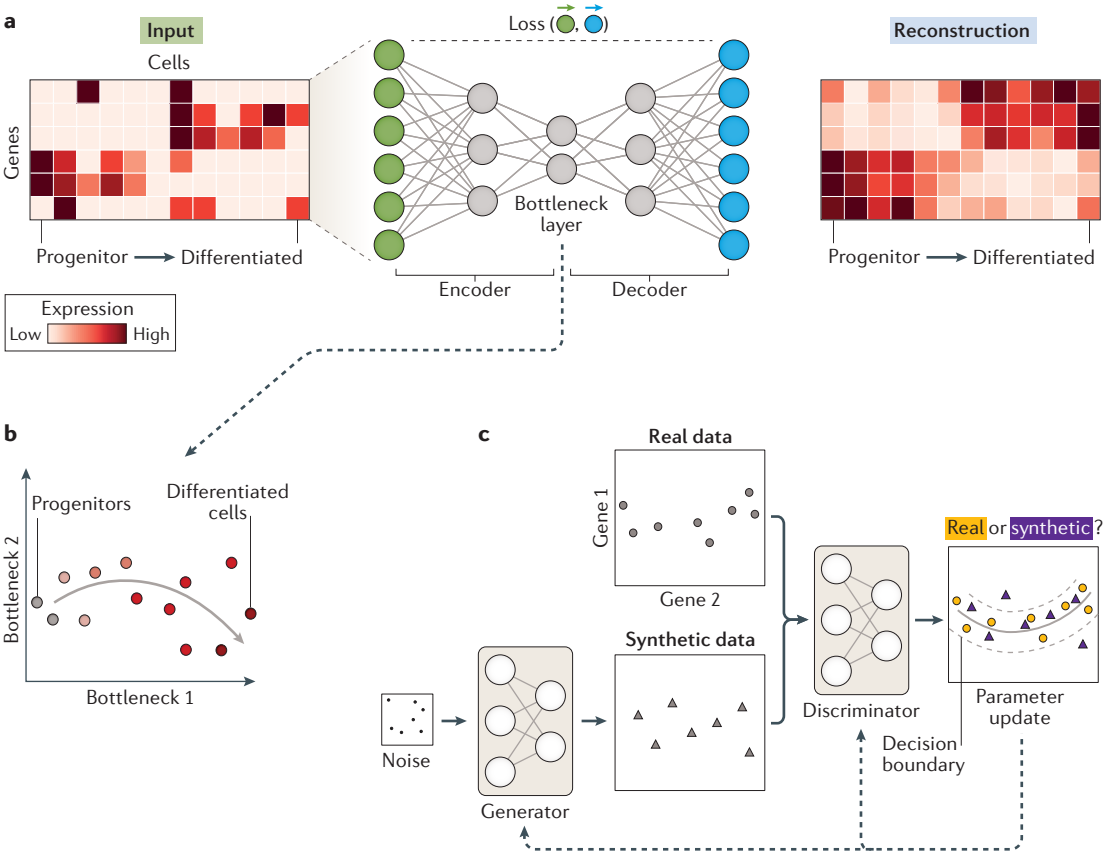


Fig. 6 | **Unsupervised learning. a** | An autoencoder consists of two parts: an encoder and a decoder. The encoder compresses the input data (depicted as gene expression of differentiating single cells) into a fewer (two shown here) dimensions in the so-called bottleneck layer. The decoder tries to reconstruct the original input from the compressed data in the bottleneck layer. Reconstruction accuracy is quantified by the loss function between the original data and the reconstructed data. Although the pseudotime estimation is not a property of autoencoders, the denoising effect of reconstruction can make the underlying structure of the data (for example cellular differentiation process) clearer[130]. **b** | The bottleneck layer is a low-dimensional representation of the original input revealing the cell differentiation process. **c** | Generative adversarial networks consist of generator and discriminator neural networks that are trained jointly. The discriminator classifies whether a given data point was drawn from the real data (circles) or whether it was synthetically generated (triangles). The generator aims to generate realistic samples and thereby tries to deceive the discriminator into mistakenly classifying synthetic samples as real.

as principal component analysis, t-distributed stochastic neighbour embedding (t-SNE) or latent variable models. Neural networks are able to generalize some of these approaches. For example, autoencoders[113,114,115,116] embed the data into a low-dimensional space with a hidden layer, called the bottleneck layer, and reconstruct the original input data (FIG. 6a). This approach forces the network to extract useful features of data, as the bottleneck layer makes it infeasible to learn the perfect reconstruction. Reconstructing the data is often interpreted as denoising because the unimportant variations are automatically left out (FIG. 6b). Principal component analysis is equivalent to a linear autoencoder[117,118,119], in which the principal components correspond to the representations in the bottleneck layer. Multiple nonlinear layers generalize linear autoencoders to a nonlinear dimensionality reduction method.

Autoencoders have been used to impute missing data[120], extract gene expression signatures[121–123] and detect expression outliers[124] in microarray data and bulk RNA sequencing gene expression data. In the field of single-cell genomics, autoencoders have been used for imputation, dimensionality reduction and representation learning[125–130]. Furthermore, prior biological knowledge has been incorporated into the autoencoder architecture in order to infer a new representation that improves clustering and visualization of cells from single-cell RNA sequencing (scRNA-seq) data[131]. Specific noise characteristics of scRNA-seq data, such as sparse count data, are also addressed with tailored loss functions within the autoencoder framework[130].

Neural networks have also greatly contributed to the toolbox of generative models. Unlike the approaches described earlier, generative models aim to learn the data-generating process. Variational autoencoders[132] (VAEs) and GANs[133] are two powerful generative approaches that have emerged in the deep learning field. VAEs are autoencoders with additional distribution assumptions that enable them to generate new random samples[11], and they have been applied to single-cell and bulk RNA sequencing data to find meaningful probabilistic latent representations[134–137]. These methods

Principal component analysis
An unsupervised learning algorithm that linearly projects data from a high-dimensional space to a lower-dimensional space while retaining as much variance as possible.

t-Distributed stochastic neighbour embedding
(t-SNE). An unsupervised learning algorithm that projects data from a high-dimensional space to a lower-dimensional space (typically 2D or 3D) in a nonlinear fashion while trying to preserve the distances between points.

demonstrate that either denoised reconstruction or low-dimensional representation of the single-cell data improves commonly performed unsupervised learning tasks such as visualization and clustering. Another approach, which uses vector arithmetic with VAE latent representations, was reported to predict cell type-specific and species-specific perturbation responses of single cells[138]. We note that the performances of VAEs and other models based on neural networks strongly depend on the choice of hyperparameters[139].

GANs were proposed as a radically different approach to generative modelling that involves two neural networks, a discriminator and a generator network. They are trained jointly, whereby the generator aims to generate realistic data points, and the discriminator classifies whether a given sample is real or generated by the generator (FIG. 6c). As a relatively new method, application of GANs is currently rather limited in genomics. They have been used to generate protein-coding DNA sequences[140] and to design DNA probes for protein binding microarrays. It has been reported that GANs are capable of generating sequences that are superior to those in the training data set, as measured by higher protein binding affinity[141]. In the field of single-cell genomics, GANs have been used to simulate scRNA-seq data and dimensionality reduction[142]. Furthermore, the authors interpreted the internal representation of GANs through perturbations. In MAGAN[143], the authors addressed the challenging problem of aligning data sets from different domains, that is, CyTOF data and scRNA-seq data, using an architecture consisting of two GANs.

Impact in genomics

Deep learning methods, both supervised and unsupervised, have found various applications in genomics. Here, we highlight three key areas in which we expect them to have the largest impact now and in the near future.

Predicting the effect of non-coding variants. Models that can predict molecular phenotypes directly from biological sequences can be used as in silico perturbation tools to probe the associations between genetic variation and phenotypic variation and have emerged as new methods for quantitative trait loci identification and variant prioritization. These approaches are of major importance given that the majority of variants identified by genome-wide association studies of complex phenotypes are non-coding[144], which makes it challenging to estimate their effects and contribution to phenotypes. Moreover, linkage disequilibrium results in blocks of variants being co-inherited, which creates difficulties in pinpointing individual causal variants. Thus, sequence-based deep learning models that can be used as interrogation tools for assessing the impact of such variants offer a promising approach to find potential drivers of complex phenotypes. Examples include DeepSEA[19], Basenji[47] and ExPecto[51], which predict the effect of non-coding single-nucleotide variants and short insertions or deletions (indels) indirectly from the difference between two variants in terms of transcription factor binding, chromatin accessibility or gene expression predictions. Furthermore, state-of-the-art models for predicting novel splice site creation[63] from sequence or quantitative effects of genetic variants on splicing[145] are deep learning models. Additionally, end-to-end approaches for variant effect predictions are beginning to appear and have been successfully applied to predict the pathogenicity of missense variants[92] from protein sequence and sequence conservation data.

Deep learning as a fully data-driven refinement of bioinformatics tools. Thanks to their flexibility, deep neural networks can be trained to carry out tasks that have traditionally been addressed by specific bioinformatics algorithms. Training computer programs instead of manually programming them has been shown to yield a significant increase in accuracy in tasks including variant calling[61,62], base calling for novel sequencing technologies[72], denoising ChIP–seq data[58] and enhancing Hi-C data resolution[59]. An additional advantage is that such programs are able to leverage GPUs without the need to write additional code.

Richer representations to reveal the structure of high-dimensional data. In addition to using deep learning as a powerful tool to make accurate predictions, its use in unsupervised settings has given rise to some important applications. Unlike other nonlinear dimensionality reduction methods, such as t-SNE, autoencoders are parametric and can therefore easily be applied to unseen data with similar distributions to the training set[146]. They are highly scalable because the training procedure only requires a small subset of the data at every training step (BOX 1), which is particularly important for fields such as single-cell genomics, in which the number of training examples can now surpass hundreds of thousands[147]. In addition, unsupervised deep learning techniques can help to characterize data sets for which it is not trivial to obtain labels, for example, to enable a data-driven definition of cell identities and states from scRNA-seq data. Finally, unsupervised methods can also be used to integrate scRNA-seq data from different sources[129,134,138,143,148], which is increasingly important not only because of growing data sets for similar tissues but also because of the generation of the first organ atlases, such as the Human Cell Atlas project[149].

Conclusions and future perspectives

The uptake of deep learning in genomics has resulted in early applications with both scientific and economic relevance. Multiple companies and industry research groups are being founded, often under the broader label of artificial intelligence, based on the anticipated economic impact of genomic deep learning on diagnostics and drug development and on its easy integration with imaging data[150]. In particular, pharmacogenomics may profit from more efficient and automated identification of novel regulatory variants in the genome and from more accurate predictions of drug responses and targets using epigenomics data[151].

Regardless of their quantitative advantages (or disadvantages) over alternative methods, some of the qualitative aspects of deep learning will remain relevant

Latent variable models
Unsupervised models describing the observed distribution by imposing latent (unobserved) variables for each data point. The simplest example is the mixture of Gaussian values.

Bottleneck layer
A neural network layer that contains fewer neurons than previous and subsequent layers.

Generative models
Models able to generate points from the desired distribution. Deep generative models are often implemented by a neural network that transforms samples from a standard distribution (normal and uniform) into samples from a complex distribution (gene expression levels or sequences that encode a splice site).

Hyperparameters
Parameters specifying the model or the training procedure that are not optimized by the learning algorithm (for example, by the stochastic gradient descent algorithm). Examples of hyperparameters are the number of layers, regularization strength, batch size and the optimization step size.

for genomics. One of these qualitative advantages is end-to-end learning. Data preprocessing steps can be time-consuming and error prone, especially in the genomics field because of the variety of experimental data sources. Being able to integrate multiple preprocessing steps into a single model and to 'let the data speak' when defining features are important advantages that often increase predictive power. We expect end-to-end learning approaches to become more widely used across genomics, including protein structure prediction[152]. Another qualitative advantage that is particularly important for genomics is the ability of deep learning to deal with multimodal data effectively. Genomics offers extremely heterogeneous data including sequence, counts, mass spectrometry intensity and images. An important application of multimodal modelling will be the development of machine learning models for spatial transcriptomics[153] that integrate scRNA-seq and imaging data, allowing gene expression to be jointly analysed with the morphology of the cell and its position in the tissue. Deep learning is the ideal approach for incorporating spatial patterns into analyses, as has been shown extensively for microscopy data[21,154]. Last but not least, an important advantage is the abstraction of the mathematical details that is offered by deep learning frameworks. Researchers in genomics often do not have the theoretical knowledge, nor do they have the time, to formulate statistical models and devise appropriate parameter fitting algorithms. Deep learning frameworks abstract much of the mathematical and technical details, such as the need for manually deriving gradients and optimization procedures, which lowers the entry barrier to the development of new models.

In the future, we expect deep learning to find new applications across multiple omics data types. We also expect to see an increasing uptake of new techniques from the deep learning research community. A particular challenge in human genomics is data privacy. One appealing direction is the development of federated learning whereby machine learning model instances are deployed on distinct sites and trained on local data while sharing common parameters[155]. By avoiding data transfer, federated learning can reduce total training time and can facilitate the respect of genetic and medical data privacy. Another relevant technique for data privacy is generative models, which could be used to simulate human genomics data that can be analysed by others without privacy violation[156]. Another important area is the prediction of causal effects, which is highly relevant to medical and therapeutic applications. Substantial progress may occur on this front as, on the one hand, the field of machine learning is becoming increasingly interested in causal modelling and, on the other hand, the field of genomics is increasingly generating perturbation data using massively parallel reporter assays or systematic CRISPR screens at the bulk and single-cell level. Although the impact of these novel developments remains to be seen, the magnitude and complexity of genomic data will ensure that deep learning will become an everyday tool for its analysis.

Published online: 10 April 2019

1. Hieter, P. & Boguski, M. Functional genomics: it's all how you read it. *Science* **278**, 601–602 (1997).
2. Brown, P. O. & Botstein, D. Exploring the new world of the genome with DNA microarrays. *Nat. Genet.* **21**, 33–37 (1999).
3. Ozaki, K. et al. Functional SNPs in the lymphotoxin-α gene that are associated with susceptibility to myocardial infarction. *Nat. Genet.* **32**, 650–654 (2002).
4. Golub, T. R. et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* **286**, 531–537 (1999).
5. Oliver, S. Guilt-by-association goes global. *Nature* **403**, 601–603 (2000).
6. The ENCODE Project Consortium. An integrated encyclopedia of DNA elements in the human genome. *Nature* **489**, 57–74 (2012).
7. Murphy, K. P. *Machine Learning: A Probabilistic Perspective* (MIT Press, 2012).
8. Bishop, C. M. *Pattern Recognition and Machine Learning* (Springer, New York, 2016).
9. Libbrecht, M. W. & Noble, W. S. Machine learning applications in genetics and genomics. *Nat. Rev. Genet.* **16**, 321–332 (2015).
10. Durbin, R., Eddy, S. R., Krogh, A. & Mitchison, G. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids* (Cambridge Univ. Press, 1998).
11. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* (MIT Press, 2016).
This textbook covers theoretical and practical aspects of deep learning with introductory sections on linear algebra and machine learning.
12. Shi, S., Wang, Q., Xu, P. & Chu, X. in *2016 7th International Conference on Cloud Computing and Big Data (CCBD)* 99–104 (IEEE, 2016).
13. Krizhevsky, A., Sutskever, I. & Hinton, G. E. in *Advances in Neural Information Processing Systems 25 (NIPS 2012)* (eds Pereira, F., Burges, C. J. C., Bottou, L. & Weinberger, K. Q.) 1097–1105 (Curran Associates, Inc., 2012).
14. Girshick, R., Donahue, J., Darrell, T. & Malik, J. in *2014 IEEE Conference on Computer Vision and Pattern Recognition* 580–587 (IEEE, 2014).
15. Long, J., Shelhamer, E. & Darrell, T. in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 3431–3440 (IEEE, 2015).
16. Hannun, A. et al. Deep speech: scaling up end-to-end speech recognition. Preprint at *arXiv* https://arxiv.org/abs/1412.5567 (2014).
17. Wu, Y. et al. Google's neural machine translation system: bridging the gap between human and machine translation. Preprint at *arXiv* https://arxiv.org/abs/1609.08144 (2016).
18. Alipanahi, B., Delong, A., Weirauch, M. T. & Frey, B. J. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nat. Biotechnol.* **33**, 831–838 (2015).
This paper describes a pioneering convolutional neural network application in genomics.
19. Zhou, J. & Troyanskaya, O. G. Predicting effects of noncoding variants with deep learning-based sequence model. *Nat. Methods* **12**, 931–934 (2015).
This paper applies deep CNNs to predict chromatin features and transcription factor binding from DNA sequence and demonstrates its utility in non-coding variant effect prediction.
20. Zou, J. et al. A primer on deep learning in genomics. *Nat. Genet.* **51**, 12–18 (2019).
21. Angermueller, C., Pärnamaa, T., Parts, L. & Stegle, O. Deep learning for computational biology. *Mol. Syst. Biol.* **12**, 878 (2016).
22. Min, S., Lee, B. & Yoon, S. Deep learning in bioinformatics. *Brief. Bioinform.* **18**, 851–869 (2017).
23. Jones, W., Alasoo, K., Fishman, D. & Parts, L. Computational biology: deep learning. *Emerg. Top. Life Sci.* **1**, 257–274 (2017).
24. Wainberg, M., Merico, D., Delong, A. & Frey, B. J. Deep learning in biomedicine. *Nat. Biotechnol.* **36**, 829–838 (2018).
25. Ching, T. et al. Opportunities and obstacles for deep learning in biology and medicine. *J. R. Soc. Interface* **15**, 20170387 (2018).
26. Morgan, J. N. & Sonquist, J. A. Problems in the analysis of survey data, and a proposal. *J. Am. Stat. Assoc.* **58**, 415–434 (1963).
27. Boser, B. E., Guyon, I. M. & Vapnik, V. N. A. in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* 144–152 (ACM, 1992).
28. Breiman, L. Random forests. *Mach. Learn.* **45**, 5–32 (2001).
29. Friedman, J. H. Greedy function approximation: a gradient boosting machine. *Ann. Stat.* **29**, 1189–1232 (2001).
30. Xiong, H. Y. et al. RNA splicing. The human splicing code reveals new insights into the genetic determinants of disease. *Science* **347**, 1254806 (2015).
31. Jha, A., Gazzara, M. R. & Barash, Y. Integrative deep models for alternative splicing. *Bioinformatics* **33**, i274–i282 (2017).
32. Quang, D., Chen, Y. & Xie, X. DANN: a deep learning approach for annotating the pathogenicity of genetic variants. *Bioinformatics* **31**, 761–763 (2015).
33. Liu, F., Li, H., Ren, C., Bo, X. & Shu, W. PEDLA: predicting enhancers with a deep learning-based algorithmic framework. *Sci. Rep.* **6**, 28517 (2016).
34. Li, Y., Shi, W. & Wasserman, W. W. Genome-wide prediction of *cis*-regulatory regions using supervised deep learning methods. *BMC Bioinformatics* **19**, 202 (2018).
35. Johnson, D. S., Mortazavi, A., Myers, R. M. & Wold, B. Genome-wide mapping of in vivo protein-DNA interactions. *Science* **316**, 1497–1502 (2007).
36. Barski, A. et al. High-resolution profiling of histone methylations in the human genome. *Cell* **129**, 823–837 (2007).
37. Robertson, G. et al. Genome-wide profiles of STAT1 DNA association using chromatin immunoprecipitation and massively parallel sequencing. *Nat. Methods* **4**, 651–657 (2007).
38. Park, P. J. ChIP-seq: advantages and challenges of a maturing technology. *Nat. Rev. Genet.* **10**, 669–680 (2009).
39. Weirauch, M. T. et al. Evaluation of methods for modeling transcription factor sequence specificity. *Nat. Biotechnol.* **31**, 126 (2013).
40. Lee, D., Karchin, R. & Beer, M. A. Discriminative prediction of mammalian enhancers from DNA sequence. *Genome Res.* **21**, 2167–2180 (2011).

41. Ghandi, M., Lee, D., Mohammad-Noori, M. & Beer, M. A. Enhanced regulatory sequence prediction using gapped k-mer features. *PLOS Comput. Biol.* **10**, e1003711 (2014).
42. Stormo, G. D., Schneider, T. D., Gold, L. & Ehrenfeucht, A. Use of the 'Perceptron' algorithm to distinguish translational initiation sites in *E. coli*. *Nucleic Acids Res.* **10**, 2997–3011 (1982).
43. Stormo, G. D. DNA binding sites: representation and discovery. *Bioinformatics* **16**, 16–23 (2000).
44. D'haeseleer, P. What are DNA sequence motifs? *Nat. Biotechnol.* **24**, 423–425 (2006).
45. Kelley, D. R., Snoek, J. & Rinn, J. L. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Res.* **26**, 990–999 (2016).
This paper describes the application of a deep CNN to predict chromatin accessibility in 164 cell types from DNA sequence.
46. Wang, M., Tai, C., E, W. & Wei, L. DeFine: deep convolutional neural networks accurately quantify intensities of transcription factor-DNA binding and facilitate evaluation of functional non-coding variants. *Nucleic Acids Res.* **46**, e69 (2018).
47. Kelley, D. R. et al. Sequential regulatory activity prediction across chromosomes with convolutional neural networks. *Genome Res.* **28**, 739–750 (2018).
In this paper, a deep CNN was trained to predict more than 4,000 genomic measurements including gene expression as measured by cap analysis of gene expression (CAGE) for every 150 bp in the genome using a receptive field of 32 kb.
48. Schreiber, J., Libbrecht, M., Bilmes, J. & Noble, W. Nucleotide sequence and DNaseI sensitivity are predictive of 3D chromatin architecture. Preprint at *bioRxiv* https://doi.org/10.1101/103614 (2018).
49. Zeng, H. & Gifford, D. K. Predicting the impact of non-coding variants on DNA methylation. *Nucleic Acids Res.* **45**, e99 (2017).
50. Angermueller, C., Lee, H. J., Reik, W. & Stegle, O. DeepCpG: accurate prediction of single-cell DNA methylation states using deep learning. *Genome Biol.* **18**, 67 (2017).
51. Zhou, J. et al. Deep learning sequence-based ab initio prediction of variant effects on expression and disease risk. *Nat. Genet.* **50**, 1171–1179 (2018).
In this paper, two models, a deep CNN and a linear model, are stacked to predict tissue-specific gene expression from DNA sequence, which demonstrates the utility of this approach in non-coding variant effect prediction.
52. Cuperus, J. T. et al. Deep learning of the regulatory grammar of yeast 5' untranslated regions from 500,000 random sequences. *Genome Res.* **27**, 2015–2024 (2017).
53. Pan, X. & Shen, H.-B. RNA-protein binding motifs mining with a new hybrid deep learning based cross-domain knowledge integration approach. *BMC Bioinformatics* **18**, 136 (2017).
54. Avsec, Ž., Barekatain, M., Cheng, J. & Gagneur, J. Modeling positional effects of regulatory sequences with spline transformations increases prediction accuracy of deep neural networks. *Bioinformatics* **34**, 1261–1269 (2018).
55. Budach, S. & Marsico, A. pysster: classification of biological sequences by learning sequence and structure motifs with convolutional neural networks. *Bioinformatics* **34**, 3035–3037 (2018).
56. Cheng, S. et al. MiRTDL: a deep learning approach for miRNA target prediction. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **13**, 1161–1169 (2016).
57. Kim, H. K. et al. Deep learning improves prediction of CRISPR-Cpf1 guide RNA activity. *Nat. Biotechnol.* **36**, 239–241 (2018).
58. Koh, P. W., Pierson, E. & Kundaje, A. Denoising genome-wide histone ChIP-seq with convolutional neuralnetworks. *Bioinformatics* **33**, i225–i233 (2017).
59. Zhang, Y. et al. Enhancing Hi-C data resolution with deep convolutional neural network HiCPlus. *Nat. Commun.* **9**, 750 (2018).
60. Nielsen, A. A. K. & Voigt, C. A. Deep learning to predict the lab-of-origin of engineered DNA. *Nat. Commun.* **9**, 3135 (2018).
61. Luo, R., Sedlazeck, F. J., Lam, T.-W. & Schatz, M. Clairvoyante: a multi-task convolutional deep neural network for variant calling in single molecule sequencing. Preprint at *bioRxiv* https://doi.org/10.1101/310458 (2018).
62. Poplin, R. et al. A universal SNP and small-indel variant caller using deep neural networks. *Nat. Biotechnol.* **36**, 983–987 (2018).
In this paper, a deep CNN is trained to call genetic variants from different DNA-sequencing technologies.
63. Jaganathan, K. et al. Predicting splicing from primary sequence with deep learning. *Cell* **176**, 535–548 (2019).
64. Elman, J. L. Finding structure in time. *Cogn. Sci.* **14**, 179–211 (1990).
65. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997).
66. Bai, S., Zico Kolter, J. & Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. Preprint at *arXiv* https://arxiv.org/abs/1803.01271 (2018).
67. Pan, X., Rijnbeek, P., Yan, J. & Shen, H.-B. Prediction of RNA-protein sequence and structure binding preferences using deep convolutional and recurrent neural networks. *BMC Genomics* **19**, 511 (2018).
68. Quang, D. & Xie, X. DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. *Nucleic Acids Res.* **44**, e107 (2016).
69. Quang, D. & Xie, X. FactorNet: a deep learning framework for predicting cell type specific transcription factor binding from nucleotide-resolution sequential data. Preprint at *bioRxiv* https://doi.org/10.1101/151274 (2017).
70. Lee, B., Baek, J., Park, S. & Yoon, S. in *Proceedings of the 7th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics* 434–442 (ACM, 2016).
71. Park, S., Min, S., Choi, H. & Yoon, S. deepMiRGene: deep neural network based precursor microRNA prediction. Preprint at *arXiv* https://arxiv.org/abs/1605.00017 (2016).
72. Boža, V., Brejová, B. & Vinař;, T. DeepNano: deep recurrent neural networks for base calling in MinION nanopore reads. *PLOS ONE* **12**, e0178751 (2017).
73. Mikheyev, A. S. & Tin, M. M. Y. A first look at the Oxford Nanopore MinION sequencer. *Mol. Ecol. Resour.* **14**, 1097–1102 (2014).
74. Barabási, A.-L., Gulbahce, N. & Loscalzo, J. Network medicine: a network-based approach to human disease. *Nat. Rev. Genet.* **12**, 56–68 (2011).
75. Mitra, K., Carvunis, A.-R., Ramesh, S. K. & Ideker, T. Integrative approaches for finding modular structure in biological networks. *Nat. Rev. Genet.* **14**, 719–732 (2013).
76. Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. & Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw.* **20**, 61–80 (2009).
77. Defferrard, M., Bresson, X. & Vandergheynst, P. in *Advances in Neural Information Processing Systems 29 (*NIPS *2016)* (eds Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I. & Garnett, R.) 3844–3852 (Curran Associates Inc., 2016).
78. Kipf, T. N. & Welling, M. Semi-supervised classification with graph convolutional networks. Preprint at *arXiv* https://arxiv.org/abs/1609.02907 (2016).
79. Battaglia, P. W. et al. Relational inductive biases, deep learning, and graph networks. Preprint at *arXiv* https://arxiv.org/abs/1806.01261 (2018).
80. Hamilton, W. L., Ying, R. & Leskovec, J. Inductive representation learning on large graphs. Preprint at *arXiv* https://arxiv.org/abs/1706.02216 (2017).
81. Chen, J., Ma, T. & Xiao, C. FastGCN: fast learning with graph convolutional networks via importance sampling. Preprint at *arXiv* https://arxiv.org/abs/1801.10247 (2018).
82. Zitnik, M. & Leskovec, J. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics* **33**, i190–i198 (2017).
83. Zitnik, M., Agrawal, M. & Leskovec, J. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics* **34**, i457–i466 (2018).
84. Duvenaud, D. K. et al. in *Advances in Neural Information Processing Systems 28 (NIPS 2015)* (eds Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M. & Garnett, R.) 2224–2232 (Curran Associates Inc., 2015).
85. Kearnes, S., McCloskey, K., Berndl, M., Pande, V. & Riley, P. Molecular graph convolutions: moving beyond fingerprints. *J. Comput. Aided Mol. Des.* **30**, 595–608 (2016).
86. Dutil, F., Cohen, J. P., Weiss, M., Derevyanko, G. & Bengio, Y. Towards gene expression convolutions using gene interaction graphs. Preprint at *arXiv* https://arxiv.org/abs/1806.06975 (2018).
87. Rhee, S., Seo, S. & Kim, S. in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence* 3527–3534 (IJCAI, 2018).
88. Chen, Z., Badrinarayanan, V., Lee, C.-Y. & Rabinovich, A. GradNorm: gradient normalization for adaptive loss balancing in deep multitask networks. Preprint at *arXiv* https://arxiv.org/abs/1711.02257 (2017).
89. Sung, K. & Poggio, T. Example-based learning for view-based human face detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **20**, 39–51 (1998).
90. Felzenszwalb, P. F., Girshick, R. B., McAllester, D. & Ramanan, D. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.* **32**, 1627–1645 (2010).
91. Guo, M., Haque, A., Huang, D.-A., Yeung, S. & Fei-Fei, L. in *Computer Vision – ECCV 2018* (eds Ferrari, V., Hebert, M., Sminchisescu, C. & Weiss, Y.) Vol. 11220 282–299 (Springer International Publishing, 2018).
92. Sundaram, L. et al. Predicting the clinical impact of human mutation with deep neural networks. *Nat. Genet.* **50**, 1161–1170 (2018).
93. Zitnik, M. et al. Machine learning for integrating data in biology and medicine: principles, practice, and opportunities. *Inf. Fusion* **50**, 71–91 (2018).
94. Yosinski, J., Clune, J., Bengio, Y. & Lipson, H. in *Advances in Neural Information Processing Systems 27 (NIPS 2014)* (eds Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D. & Weinberger, K. Q.) 3320–3328 (Curran Associates Inc., 2014).
95. Kornblith, S., Shlens, J. & Le, Q. V. Do better ImageNet models transfer better? Preprint at *arXiv* https://arxiv.org/abs/1805.08974 (2018).
96. Russakovsky, O. et al. ImageNet large scale visual recognition challenge. Preprint at *arXiv* https://arxiv.org/abs/1409.0575 (2014).
97. Esteva, A. et al. Dermatologist-level classification of skin cancer with deep neural networks. *Nature* **542**, 115–118 (2017).
98. Pawlowski, N., Caicedo, J. C., Singh, S., Carpenter, A. E. & Storkey, A. Automating morphological profiling with generic deep convolutional networks. Preprint at *bioRxiv* https://doi.org/10.1101/085118 (2016).
99. Zeng, T., Li, R., Mukkamala, R., Ye, J. & Ji, S. Deep convolutional neural networks for annotating gene expression patterns in the mouse brain. *BMC Bioinformatics* **16**, 147 (2015).
100. Zhang, W. et al. in *IEEE Transactions on Big Data* (IEEE, 2018).
101. Adam, P. et al. Automatic differentiation in PyTorch. Presented at 31st Conference on Neural Information Processing Systems (NIPS 2017).
102. Abadi, M. et al. Tensorflow: large-scale machine learning on heterogeneous distributed systems. Preprint at *arXiv* https://arxiv.org/abs/1603.04467 (2016).
103. Avsec, Z. et al. Kipoi: accelerating the community exchange and reuse of predictive models for genomics. Preprint at *bioRxiv* https://doi.org/10.1101/375345 (2018).
This paper describes a platform to exchange trained predictive models in genomics including deep neural networks.
104. Breiman, L. Statistical modeling: the two cultures (with comments and a rejoinder by the author). *Stat. Sci.* **16**, 199–231 (2001).
105. Greenside, P., Shimko, T., Fordyce, P. & Kundaje, A. Discovering epistatic feature interactions from neural network models of regulatory DNA sequences. *Bioinformatics* **34**, i629–i637 (2018).
106. Zeiler, M. D. & Fergus, R. in *Computer Vision – ECCV 2014* (eds Fleet, D., Pajdla, T., Schiele, B. & Tuytelaars, T.) Vol. 8689 818–833 (Springer International Publishing, 2014).
107. Simonyan, K., Vedaldi, A. & Zisserman, A. Deep inside convolutional networks: visualising image classification models and saliency maps. Preprint at *arXiv* https://arxiv.org/abs/1312.6034 (2013).
108. Shrikumar, A., Greenside, P., Shcherbina, A. & Kundaje, A. Not just a black box: learning important features through propagating activation differences. Preprint at *arXiv* https://arxiv.org/abs/1605.01713 (2016).
This paper introduces DeepLIFT, a neural network interpretation method that highlights inputs most influential for the prediction.
109. Sundararajan, M., Taly, A. & Yan, Q. Axiomatic attribution for deep networks. Preprint at *arXiv* https://arxiv.org/abs/1703.01365 (2017).
110. Lanchantin, J., Singh, R., Wang, B. & Qi, Y. Deep motif dashboard: visualizing and understanding genomic sequences using deep neural networks. *Pac. Symp. Biocomput.* **22**, 254–265 (2017).

111. Shrikumar, A. et al. TF-MoDISco v0.4.4.2-alpha: technical note. Preprint at *arXiv* https://arxiv.org/abs/1811.00416v2 (2018).
112. Ma, J. et al. Using deep learning to model the hierarchical structure and function of a cell. *Nat. Methods* **15**, 290–298 (2018).
113. Hinton, G. E. & Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science* **313**, 504–507 (2006).
114. Kramer, M. A. Nonlinear principal component analysis using autoassociative neural networks. *AIChE J.* **37**, 233–243 (1991).
115. Vincent, P., Larochelle, H., Bengio, Y. & Manzagol, P.-A. in *Proceedings of the 25th International Conference on Machine Learning* 1096–1103 (ACM, 2008).
116. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y. & Manzagol, P.-A. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.* **11**, 3371–3408 (2010).
117. Jolliffe, I. in *International Encyclopedia of Statistical Science* (ed. Lovric, M.) 1094–1096 (Springer Berlin Heidelberg, 2011).
118. Plaut, E. From principal subspaces to principal components with linear autoencoders. Preprint at *arXiv* https://arxiv.org/abs/1804.10253 (2018).
119. Kunin, D., Bloom, J. M., Goeva, A. & Seed, C. Loss landscapes of regularized linear autoencoders. Preprint at *arXiv* https://arxiv.org/abs/1901.08168 (2019).
120. Scholz, M., Kaplan, F., Guy, C. L., Kopka, J. & Selbig, J. Non-linear PCA: a missing data approach. *Bioinformatics* **21**, 3887–3895 (2005).
121. Tan, J., Hammond, J. H., Hogan, D. A. & Greene, C. S. ADAGE-based integration of publicly available *Pseudomonas aeruginosa* gene expression data with denoising autoencoders illuminates microbe-host interactions. *mSystems* **1**, e00025–15 (2016).
122. Tan, J. et al. ADAGE signature analysis: differential expression analysis with data-defined gene sets. *BMC Bioinformatics* **18**, 512 (2017).
123. Tan, J. et al. Unsupervised extraction of stable expression signatures from public compendia with an ensemble of neural networks. *Cell Syst.* **5**, 63–71 (2017).
124. Brechtmann, F. et al. OUTRIDER: a statistical method for detecting aberrantly expressed genes in RNA sequencing data. *Am. J. Hum. Genet.* **103**, 907–917 (2018).
125. Ding, J., Condon, A. & Shah, S. P. Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. *Nat. Commun.* **9**, 2002 (2018).
126. Cho, H., Berger, B. & Peng, J. Generalizable and scalable visualization of single-cell data using neural networks. *Cell Syst.* **7**, 185–191 (2018).
127. Deng, Y., Bao, F., Dai, Q., Wu, L. & Altschuler, S. Massive single-cell RNA-seq analysis and imputation via deep learning. Preprint at *bioRxiv* https://doi.org/10.1101/315556 (2018).
128. Talwar, D., Mongia, A., Sengupta, D. & Majumdar, A. AutoImpute: autoencoder based imputation of single-cell RNA-seq data. *Sci. Rep.* **8**, 16329 (2018).
129. Amodio, M. et al. Exploring single-cell data with deep multitasking neural networks. Preprint at *bioRxiv* https://doi.org/10.1101/237065 (2019).
130. Eraslan, G., Simon, L. M., Mircea, M., Mueller, N. S. & Theis, F. J. Single-cell RNA-seq denoising using a deep count autoencoder. *Nat. Commun.* **10**, 390 (2019).
131. Lin, C., Jain, S., Kim, H. & Bar-Joseph, Z. Using neural networks for reducing the dimensions of single-cell RNA-Seq data. *Nucleic Acids Res.* **45**, e156 (2017).
132. Kingma, D. P. & Welling, M. *Auto-encoding variational bayes*. Preprint at *arXiv* https://arxiv.org/abs/1312.6114 (2013).
133. Goodfellow, I. et al. in *Advances in Neural Information Processing Systems 27 (NIPS 2014)* (eds Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D. & Weinberger, K. Q.) 2672–2680 (Curran Associates Inc., 2014).
134. Lopez, R., Regier, J., Cole, M. B., Jordan, M. I. & Yosef, N. Deep generative modeling for single-cell transcriptomics. *Nat. Methods* **15**, 1053–1058 (2018).
135. Way, G. P. & Greene, C. S. in *Biocomputing 2018: Proceedings of the Pacific Symposium* (eds Altman, R. B. et al.) 80–91 (World Scientific, 2018).
136. Grønbech, C. H. et al. scVAE: variational auto-encoders for single-cell gene expression data. Preprint at *bioRxiv* https://doi.org/10.1101/318295 (2018).
137. Wang, D. & Gu, J. VASC: dimension reduction and visualization of single-cell RNA-seq data by deep variational autoencoder. *Genomics Proteomics Bioinformatics* **16**, 320–331 (2018).
138. Lotfollahi, M., Alexander Wolf, F. & Theis, F. J. Generative modeling and latent space arithmetics predict single-cell perturbation response across cell types, studies and species. Preprint at *bioRxiv* https://doi.org/10.1101/478503 (2018).
139. Hu, Q. & Greene, C. S. Parameter tuning is a key part of dimensionality reduction via deep variational autoencoders for single cell RNA transcriptomics. Preprint at *bioRxiv* https://doi.org/10.1101/385534 (2018).
140. Gupta, A. & Zou, J. Feedback GAN (FBGAN) for DNA: a novel feedback-loop architecture for optimizing protein functions. Preprint at *arXiv* https://arxiv.org/abs/1804.01694 (2018).
141. Killoran, N., Lee, L. J., Delong, A., Duvenaud, D. & Frey, B. J. Generating and designing DNA with deep generative models. Preprint at *arXiv* https://arxiv.org/abs/1712.06148 (2017).
142. Ghahramani, A., Watt, F. M. & Luscombe, N. M. Generative adversarial networks simulate gene expression and predict perturbations in single cells. Preprint at *bioRxiv* https://doi.org/10.1101/262501 (2018).
143. Amodio, M. & Krishnaswamy, S. MAGAN: aligning biological manifolds. Preprint at *arXiv* https://arxiv.org/abs/1803.00385 (2018).
144. Maurano, M. T. et al. Systematic localization of common disease-associated variation in regulatory DNA. *Science* **337**, 1190–1195 (2012).
145. Cheng, J. et al. MMSplice: modular modeling improves the predictions of genetic variant effects on splicing. *Genome Biol.* **20**, 48 (2019).
146. van der Maaten, L. in *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics* (eds van Dyk, D. & Welling, M.) Vol. 5 384–391 (PMLR, 2009).
147. Angerer, P. et al. Single cells make big data: new challenges and opportunities in transcriptomics. *Curr. Opin. Syst. Biol.* **4**, 85–91 (2017).
148. Shaham, U. et al. Removal of batch effects using distribution-matching residual networks. *Bioinformatics* **33**, 2539–2546 (2017).
149. Regev, A. et al. The human cell atlas. *eLife* **6**, e27041 (2017).
150. Fleming, N. How artificial intelligence is changing drug discovery. *Nature* **557**, S55–S57 (2018).
151. Kalinin, A. A. et al. Deep learning in pharmacogenomics: from gene regulation to patient stratification. *Pharmacogenomics* **19**, 629–650 (2018).
152. AlQuraishi, M. End-to-end differentiable learning of protein structure. Preprint at *bioRxiv* https://doi.org/10.1101/265231 (2018).
153. Nawy, T. Spatial transcriptomics. *Nat. Methods* **15**, 30 (2018).
154. Eulenberg, P. et al. Reconstructing cell cycle and disease progression using deep learning. *Nat. Commun.* **8**, 463 (2017).
155. Konečný, J., McMahan, H. B., Ramage, D. & Richtárik, P. Federated optimization: distributed machine learning for on-device intelligence. Preprint at *arXiv* https://arxiv.org/abs/1610.02527 (2016).
156. Beaulieu-Jones, B. K. et al. Privacy-preserving generative deep neural networks support clinical data sharing. Preprint at *bioRxiv* https://doi.org/10.1101/159756 (2018).
157. Lever, J., Krzywinski, M. & Altman, N. Classification evaluation. *Nat. Methods* **13**, 603 (2016).
158. Tieleman, T. & Hinton, G. Lecture 6.5 - RMSProp, COURSERA: neural networks for machine learning (2012).
159. Kingma, D. P. & Ba, J. Adam: a method for stochastic optimization. Preprint at *arXiv* https://arxiv.org/abs/1412.6980 (2014).
160. Schmidhuber, J. Deep learning in neural networks: an overview. *Neural Netw.* **61**, 85–117 (2015).
161. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
162. Bottou, L. in *Proceedings of Neuro-Nımes '91* 12 (EC2, 1991).
163. Bengio, Y. Practical recommendations for gradient-based training of deep architectures. Preprint at *arXiv* https://arxiv.org/abs/1206.5533 (2012).
164. Bergstra, J. & Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **13**, 281–305 (2012).
165. Bergstra, J., Yamins, D. & Cox, D. in *Proceedings of the 30th International Conference on Machine Learning* Vol. 28 115–123 (JMLR W&CP, 2013).
166. Shahriari, B., Swersky, K., Wang, Z., Adams, R. P. & de Freitas, N. Taking the human out of the loop: a review of bayesian optimization. *Proc. IEEE* **104**, 148–175 (2016).
167. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A. & Talwalkar, A. Hyperband: a novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.* **18**, 6765–6816 (2017).
168. Elsken, T., Metzen, J. H. & Hutter, F. Neural architecture search: a survey. Preprint at *arXiv* https://arxiv.org/abs/1808.05377 (2018).

Acknowledgements
Ž.A. was supported by the German Bundesministerium für Bildung und Forschung (BMBF) through the project MechML (01IS18053F). The authors acknowledge M. Heinig and A. Raue for valuable feedback.

Author contributions
The authors contributed equally to all aspects of the article.

Competing interests
The authors declare no competing interests.

Publisher's note
Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Reviewer information
Nature Reviews Genetics thanks C. Greene and the other anonymous reviewer(s) for their contribution to the peer review of this work.

RELATED LINKS
DragoNN: https://kundajelab.github.io/dragonn/tutorials.html
Kaggle machine learning competitions: https://www.kaggle.com/sudalairajkumar/winning-solutions-of-kaggle-competitions
Keras: https://keras.io/
Keras model zoos: https://keras.io/applications/
PyTorch: http://pytorch.org
PyTorch model zoos: https://pytorch.org/docs/stable/torchvision/models.html
Tensorflow model zoos: https://github.com/tensorflow/models; https://www.tensorflow.org/hub/