Designing Eukaryotic Gene Expression Regulation Using Machine Learning

Trends in Biotechnology

Jongh, Ronald P.H.; Dijk, Aalt D.J.; Julsing, Mattijs K.; Schaap, Peter J.; Ridder, Dick

https://doi.org/10.1016/j.tibtech.2019.07.007

Trends in Biotechnology

CellPress REVIEWS

Review

Designing Eukaryotic Gene Expression Regulation Using Machine Learning

Ronald P.H. de Jongh,[1,4] Aalt D.J. van Dijk,[1,2] Mattijs K. Julsing,[3] Peter J. Schaap,[4] and Dick de Ridder[1,*]

Controlling the expression of genes is one of the key challenges of synthetic biology. Until recently fine-tuned control has been out of reach, particularly in eukaryotes owing to their complexity of gene regulation. With advances in machine learning (ML) and in particular with increasing dataset sizes, models predicting gene expression levels from regulatory sequences can now be successfully constructed. Such models form the cornerstone of algorithms that allow users to design regulatory regions to achieve a specific gene expression level. In this review we discuss strategies for data collection, data encoding, ML practices, design algorithm choices, and finally model interpretation. Ultimately, these developments will provide synthetic biologists with highly specific genetic building blocks to rationally engineer complex pathways and circuits.

Highlights

Machine learning (ML) models can predict gene expression levels from DNA sequences, given sufficiently large datasets. Such datasets are now rapidly becoming available for regions that regulate eukaryotic gene expression, namely promoters and untranslated regions (UTRs).

These predictive models are increasingly used in algorithms for designing novel regulatory regions to achieve a desired fine-tuned expression level.

ML models of gene expression will enable synthetic biologists to rationally engineer complex pathways and circuits.

With increasing attention to interpretability of ML models, they may also help to gain deeper understanding of eukaryotic gene regulation.

Controlling Gene Expression in Eukaryotes

A fundamental challenge in synthetic biology is controlling gene expression of engineered pathways or circuits in an organism of interest. Gene expression levels should be carefully tuned to obtain optimal protein levels and to evoke a desired function or phenotype, regardless of whether the coding sequences of the gene were present natively or introduced from another organism. Protein gene expression is regulated through various regulatory genomic regions. Transcription initiation is mostly regulated by the **promoter** (see Glossary), where transcription factors (TFs) bind to recruit or inhibit RNA polymerases [1], but also by surrounding chromatin [2]. Translation initiation and elongation are subsequently regulated by **untranslated regions (UTRs)** and coding sequences, respectively [3]. Box 1 provides a more detailed discussion.

Although prokaryotic host organisms are used in a wide array of biotechnological applications, eukaryotic hosts play an important role because they can produce larger proteins as well proteins that require post-translational modification [4]. Compared with prokaryotes, the processes of transcription and translation initiation are relatively complex in eukaryotes [5–7]. However, some regions – promoters and UTRs – are now sufficiently well characterized that regulatory sequences can be introduced to achieve desired expression levels. To allow rational, modular engineering, such regulatory sequences should have predictable and robust activities, and ideally be insensitive (i.e., orthogonal) to other processes in the host organism. Until recently these sequences were often selected from a fixed library of characterized sequences. This approach has limitations [8–10] because; (i) the desired gene expression may not lie in the range of the library or cannot be fine-tuned to the level needed, (ii) longer sequences cannot easily be reused for different genes to avoid genetic engineering issues, and (iii) the set of parts available does not scale to engineer larger circuits or pathways. Moreover, libraries characterized in particular hosts and environments may yield different or even unwanted (side) effects when the host or conditions are changed. As a result, there is increasing interest in designing custom sets of regulatory sequences to achieve desired gene expression levels in specific applications.

[1]Bioinformatics Group, Wageningen University and Research, Wageningen, The Netherlands
[2]Biometris, Wageningen University and Research, Wageningen, The Netherlands
[3]Wageningen Food and Biobased Research, Wageningen University and Research, Wageningen, The Netherlands
[4]Laboratory of Systems and Synthetic Biology, Wageningen University and Research, The Netherlands

*Correspondence:
dick.deridder@wur.nl (D. de Ridder).

https://doi.org/10.1016/j.tibtech.2019.07.007

Box 1. Eukaryotic Gene Expression

The immediate genomic regions that are thought to play a role in eukaryotic gene regulation include distal enhancers, promoter (upstream and core), the 5′-UTR, the coding sequence (CDS), introns, the 3′-UTR, and the terminator (Figure I). Beyond this, regulation may be influenced by, among others, surrounding genes, nucleosomal architecture, and the 3D conformation of the chromosomes in the nucleus. Other regulatory influences can be epigenetic, such as histone modifications or DNA methylation, and post-transcriptional, such as miRNA regulation.

The most researched genomic regulatory region is the promoter. The core promoter is where the RNA polymerase is recruited and from where transcription begins [45]. The region is defined as the minimal portion of the promoter that is necessary to initiate transcription. Core promoters can be categorized based on the presence or absence of a consensus TATA box. Promoters lacking a consensus TATA box tend to be more easily accessible owing to depletion of nucleosomes, allowing easier access by TFs and the RNA polymerase complex [12]. It has been shown that sequences as short as 10 nt, in combination with a TATA box-containing core promoter, can yield robust expression [78]. Initial high-throughput experiments have investigated this region by placing it upstream of a minimal TATA box-containing promoter [19,22,23].

The region downstream of the core promoter is called the 5′-UTR. Starting at the transcription start-site (TSS) and ending at the first codon to be translated into an amino acid, the 5′-UTR is important in regulating translation initiation. The 5′-UTR is a common target for optimization of gene expression owing to its relatively short length. One of the most important possible elements is an upstream open reading frame (uORF), which when present tends to inhibit expression. The other area of interest is the location directly upstream of the start codon. The sequence known to produce high expression is the so-called Kozak sequence that is still being researched because it is not always conserved [9].

The region downstream of the coding sequence is the 3′-UTR, which is usually associated with miRNAs and RNA-binding proteins (RBPs) in terms of gene regulation. The region normally contains one or more polyadenylation signals. The last region of each gene is the terminator region. This follows the 3′-UTR, and is mainly involved in cleavage of the 3′ region of the mRNA and the process of polyadenylation [79].

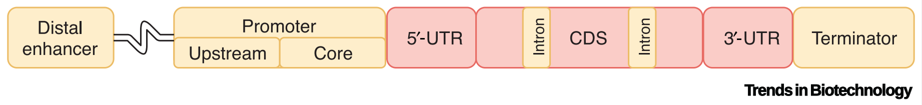


Figure I. Regions Involved in Regulating Eukaryotic Gene Expression. Regions in red are part of the mature mRNA, only the coding sequence (CDS) is translated into a protein product.

Given the plummeting costs of DNA synthesis, *in silico* designed sequences can easily be obtained. This leaves the question as to what sequences achieve a desired expression level. Computational models for designing regulatory sequences have been available for prokaryotes for some time. A well-known example is the ribosome binding calculator [11], which is based on a biophysical model. However, eukaryotic gene regulation is more complex, and sufficiently detailed biophysical models of transcription activation have not yet been developed. Nevertheless, increasingly large genomic (sequence) and transcriptomic (activity) datasets are becoming available [12]. Such datasets offer opportunities for the application of **machine learning (ML)** methods, fitting predictive sequence–activity models without relying on extensive prior knowledge. Given sufficient data, such models can generalize and predict activity for so far unseen sequences. Many relevant problems in biology are already routinely addressed in this way, including the prediction of gene structure [13], protein function, location, and structure [14–16], as well as protein–protein and protein–DNA interactions [17,18].

We discuss here how ML plays an increasingly important role in designing gene regulation in eukaryotes (Figure 1, Key Figure). First, we discuss the issues encountered when collecting data and building models to predict gene expression from sequence data. Next, we focus on design, surveying how such models can be used in DNA design algorithms to achieve a desired gene expression level. Finally, we highlight avenues for future research and discuss how ML can help to provide insights into eukaryotic regulatory biology and be applied in synthetic biology.

Glossary

Active learning: in ML, the use of a model trained on a (small) initial dataset to decide which datapoints are most informative to add; in this context, to decide for which sequences to measure activity next.

Cross-validation: an ML validation technique in which the data are split into *n* parts, after which *n* models are trained on *n* – 1 parts, each time leaving out a different part for testing. The cross-validation performance estimate is then the average performance over these test parts.

Element: a sequence motif that is sufficiently conserved to be identified and has a function related to gene expression. Notable examples are binding sites and sites that promote nucleosome-free regions of the DNA.

Fluorescence-activated cell sorting (FACS): a single cell-based method of sorting cells based on fluorescence signals; can be used to separate populations of cells each showing similar expression of a fluorescent reporter.

Generative adversarial networks (GANs): a pair of neural networks trained in tandem – a generator network to generate realistic 'adversarial' examples, and a discriminator network to distinguish between generated and real samples [80].

Kernel-based models: a class of ML algorithms based on the 'kernel trick' that involves calculating similarities between samples using a kernel. The best-known method is the support vector machine [81].

***K*-mer:** subsequence of length *k*, often used to encode a sequence by storing the frequencies of each possible *k*-mer over the entire sequence into a vector to be used as the input for an ML algorithm.

Machine learning (ML): a subfield of artificial intelligence that develops algorithms that can learn to perform specific tasks from a given dataset.

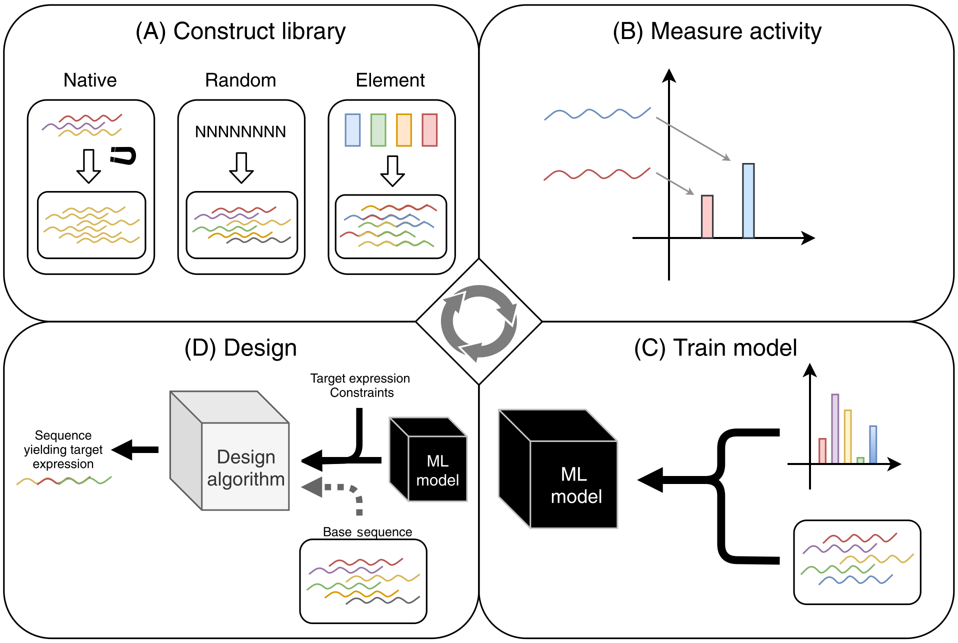
One-hot encoding: a method of transforming sequence data into numerical data by turning each letter into a vector of the length of the alphabet of that sequence.

Overfitting: a situation in which an ML model has become overadapted to the specific training data, to the point that it performs less well on new, unseen samples.

Promoter: the region of a gene where transcription factors (TFs) and the RNA

Key Figure

Design Cycle for Regulatory Sequences



Trends in Biotechnology

Figure 1. (A) Library construction, in one of three ways: (i) native sequences with a known binding site are extracted via capture and clone methods, (ii) random combinations of nucleotides are used to generate the library, or (iii) known elements such as transcription factor binding sites (TFBSs) or nucleosome altering sites are inserted into a background sequence. (B) Measurement of activity, usually involving fluorescence-activated cell sorting (FACS) or RNA-seq. (C) Using the sequences and corresponding activities to train a machine learning (ML) model. (D) Designing new sequences, using the trained model as a design criterion, sometimes requiring a base-sequence to modify. Designed sequences can be used for a desired purpose or can serve as new libraries to close the cycle.

Generating Sequence Data

ML (Box 2) optimizes predictive models by fitting their parameters based on large datasets of inputs and corresponding desired outputs, so-called training data. The resulting models are then used to make predictions for new, unseen test data. Training a model to predict activity from sequence data requires quantitative measurement of gene expression in a regulatory sequence library that is sufficiently large and varied to capture the relevant parts of sequence space. The two main approaches are inserting subsequences into a background [19] and capturing native sequences [20]. The resulting variants are combined with a reporter gene, usually together with fixed, characterized additional regions (Box 1) to isolate the effects of variation in a regulatory region of interest. These constructs are introduced to cells on custom single-copy plasmids or via integration into the genome, although this is generally more time-consuming [21]. Expression of the reporter gene downstream of the variant sequences is then quantified.

To generate sufficient training data, some type of high-throughput sequence synthesis, measurement, and sequencing must be combined. A commonly used paradigm is the massively parallel reporter assay (MPRA) [22,23] that generally produces thousands to millions of sequence–activity measurements. MPRAs have been reviewed in the context of understanding genetic regulation

polymerase are recruited, and from where transcription begins.
Terminator: following the 3′-UTR, terminators are mainly involved in cleavage of the 3′ region of the mRNA and in polyadenylation [67].
Transfer learning: using knowledge gained training an ML model on one problem and using it to train a model to solve a different but similar task [61].
Untranslated region (UTR): an mRNA region located upstream (5′) or downstream (3′) of the coding sequence. The 3′-UTR starts at the transcription start-site (TSS) and ends at the first codon translated into an amino acid, and regulates translation initiation. The 3′-UTR is usually associated with miRNAs and RNA-binding proteins (RBPs) in terms of gene regulation, and normally contains one or more polyadenylation signals [67].

Box 2. Machine Learning

ML is a subfield of artificial intelligence that develops algorithms that can learn to perform specific tasks from a given dataset. ML is widely used in research and industry, with notable successes including automated translation, medical diagnosis, self-driving cars, etc. [82]. A major distinction is between supervised learning, aiming to predict a desired output (often a label) for a given object, and unsupervised learning that seeks patterns in sets of unlabeled objects. In both cases the input data often need to be preprocessed depending on their type and the choice of algorithm, and care should be taken regarding what measurements are taken or calculated to best represent the objects. Most algorithms work from numerical or categorical data, although some more advanced methods can also directly process structured data such as sequences, words, graphs, etc.

Supervised learning optimizes a predictive model by fitting its parameters to perform well on large datasets of inputs and corresponding desired outputs, the so-called training data. The resulting models can then make predictions for new, unseen test data. Care should be taken to avoid overfitting, situations in which the model performs well on training data but does not generalize well to new data. A common rule of thumb is that training requires at least 10 samples per input feature [83].

The most decisive factor in successful application of ML is the availability of sufficient, well-measured, and correctly labeled training data. However, the choice of model is also important: simpler models are less likely to overfit, but cannot always model more complex relations between input features. Simpler models also tend to be quicker to train, whereas for large deep learning models specialized hardware such as graphics processing units (GPUs) may be required. An important consideration may be to select methods that give interpretable decisions or attach a measure of certainty to their outputs, for deciding on subsequent actions such as sequence design.

[24], and methods for statistical analysis have also been reviewed previously [25]. MPRAs usually create variants by manipulating sequences at either the single-nucleotide level or at the **element** level [26].

Nucleotide-Level Variation

The most basic approach for creating a library is random mutagenesis at the nucleotide level. This is often the method of choice owing to its ease of use. However, the space of potential variants quickly becomes extremely large. For example, a library of 1 million sequences almost covers all 4^{10} possible sequences of length 10, but only 0.0001% of all possible sequences of length 20. Although ML methods can generalize, and thus do not need to be trained on an exhaustive dataset, randomly sampling all possible variants would be expected to yield mostly inactive sequences. Even so, in the gigantic parallel reporter assay [27] a set of variants of 80 nt sequences was constructed without restrictions, and 100 million (10^8) of 4^{80} (10^{48}) possible variants were synthesized and tested in the laboratory. TF binding sites (TFBSs) were found to have such low information content that they arise out of random sequences sufficiently frequently to produce functional *cis*-regulatory regions by chance. This indicates that random sampling may provide sufficiently rich data for training an ML model.

Scanning mutagenesis approaches, starting from one or more known sequences, offer a more systematic way to explore sequence space compared with random mutagenesis. A small region of the DNA, or even a single nucleotide, is mutated at a time [19,28,29]. Large changes in measured expression levels pinpoint the locations of important nucleotides, indicating the presence of binding sites that influence nucleosome binding, or key DNA-shape features that affect the binding affinity of surrounding elements. The exploration–exploitation trade-off, in other words how far to stray from known sequences, is an important design choice here.

Element-Based Variation

An alternative approach is to combine known regulatory elements, either randomly or systematically. Often, a characterized native regulatory sequence is selected, and known elements are removed by mutation to create a background sequence of low activity. Elements are then inserted, replacing the native bases, such that the overall variable region length stays the same [19]. This has the advantage of using available prior knowledge for exploring sequence space. Moreover, the introduction or removal of an element tends to have a larger effect on expression than randomly chosen mutations.

Even though elements can be seen as building blocks of regulation, they still need to be tested in the target environment. They may, for example, rely on TFs present at different levels than in the environment they were characterized in [19]. The same holds for genomic context: if element activity depends on surrounding chromatin structure [30], it may not work on a plasmid. In a clear example of the effect of local DNA sequence, Maricque and colleagues found that it accounted for almost 50% of variance in regulatory activity when they integrated lentivirus-based constructs into a genome [21]. In a follow-up study they showed that the relative effects of *cis*-regulatory sequences are maintained regardless of genomic location [31].

Measuring Activity Levels

Once a library has been generated, variants can be inserted into reporter constructs and introduced into cells. Gene expression is commonly measured [26] by either **fluorescence-activated cell sorting (FACS)** or RNA sequencing (RNA-seq) [22,23]. In the first approach, variants activate expression of a fluorescent reporter protein, and cells are sorted according to their fluorescence level into a set of defined bins which quantize the full expression range. Variants are then isolated, barcoded to reflect the expression bin from which they were retrieved, and sequenced by high-throughput DNA sequencing. For each variant this results in a distribution of read-counts over the bins, from which a measure of its corresponding gene expression is derived as the average bin expression level over all reads [19,32]. This means that variation of expression can be estimated, which may be useful in an ML context. Because activity is measured as protein fluorescence levels, the FACS-based approach measures the combination of translation and transcription activity. When libraries are made that only vary in regions known to modulate transcription activity, FACS measurements should largely reflect transcriptional effects.

RNA-seq, by contrast, reports on transcription alone. Sequences should be made such that transcripts contain unique barcode sequences linked to the regulatory variant, often in the 3′-UTR or as a synonymous codon barcode [33]. In either case care should be taken that the barcode in the RNA does not interfere with transcription or RNA processing. Sequencing RNA allows quantization of expression, whereas DNA sequencing makes it possible to link variants to expression levels through the barcode. The activity measurement is then the ratio of RNA reads to the number of DNA reads. This provides a direct measure of transcription, giving a clearer picture compared with FACS-based approaches, but at the same time is not applicable to detecting the effects of 5′-UTR modifications on translation [34].

An alternative to the activity methods discussed above is a competitive growth assay, in which the growth of a host depends on the amount of expression of an essential reporter protein. This growth can then be measured through optical density measurements in a plate reader or by quantitative PCR [35].

Data Encoding and Model Training

Although some ML models can handle sequence data directly [36], most models work with numeric or categorical inputs. Sequence data should thus be encoded as a set of numbers, called features. There are three common ways of doing this, encoding elements, ***k*-mers**, or nucleotides. This is an important choice because it is related to the extent to which prior knowledge is available to the algorithm.

In element encoding, sequences are scanned for specific elements and their counts are recorded as features [8,29,37–43]. This approach is closely linked to element-based data generation because it will already be known which elements can be present in each sequence. However, information about background sequence composition, as well as the relative positioning of elements and other genomic landmarks, is lost when encoding using only counts. A notable exception is the work done by Beer and Tavazoie [44], who mapped such features to patterns of gene expression across many conditions.

Sequences can also be represented by the frequencies of their *k*-mers, in other words overlapping subsequences of *k* nucleotides [21,28,33,45–47] (also known as *q*-grams or *n*-grams [48]). Optionally, *k*-mer frequency profiles from single nucleotides up to the chosen *k* can be aggregated. The parameter *k* determines the coarseness of the encoding: the higher the value the more likely it is that different sequences will have different *k*-mer profiles. However, because the number of features grows exponentially with *k*, and large feature sets lead to sparse training data, it should be chosen with care. The main advantage of *k*-mer counting is that no prior knowledge of elements is required because individual *k*-mers representing biological elements or parts thereof can be learned by the model from the data.

Finally, individual nucleotides can be encoded as numbers. The most common method is called **one-hot encoding** after the fact that each nucleotide is represented as a vector that contains all zeroes except for a single 'hot' position, usually set to one. This is placed at the index of the nucleotide encoded. In this way a DNA sequence is represented by a matrix with four columns and a number of rows equal to the sequence length [27,49–53]. This method is often used with a deep learning model [49] owing to the high interdependence of the features. Importantly, this encoding method keeps positional information, thereby allowing models to pick up important elements or local nucleotide composition.

Features may also be inferred from the original sequence, for example using biophysical models yielding free energy of the 5′-UTR [9,45,54,55], or hybridization energy and target accessibility of miRNA in the 3′-UTR [56]. An important feature is nucleosome occupancy [39,57–59], which has been used to facilitate **terminator** design [60]. ML models can use these predictions and combine them with primary sequence information.

After features have been derived from sequences, a wide array of ML methods are available for training. Simple models such as linear regression are not applicable to all problems, but can be fitted using relatively small training sets and are far less likely to suffer from **overfitting**. At the other end of the spectrum, (deep learning) neural networks can be fitted to almost any problem, but require large amounts of training data and many design and parameter choices must be made. In practice, the art of ML is trying out several approaches to learn which (type of) method is likely to give optimal performance. Care should be taken to ensure the model generalizes well, in other words does not overfit the training data. This can be done through application of methods such as **cross-validation**, although one of the most important aspects is that the dataset is sufficiently large (Box 2 for a more detailed discussion).

ML-Based Design

A trained ML model can be used to predict the activity of unseen sequences. This predictor can then serve as a criterion in a design algorithm searching for sequences to attain a target expression level or to maximize expression, taking specified constraints (e.g., on sequence composition or homology) into account. Table 1 lists examples of ML-based designs and indicates which applications the designs were used for, including metabolic engineering. Two methods are

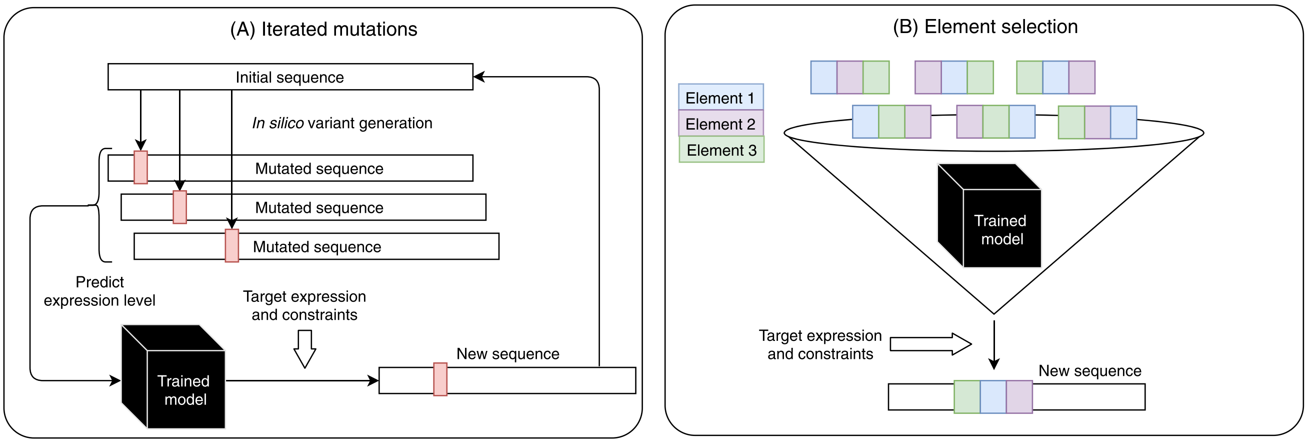
Table 1. Recent Successes of ML Using Design Algorithms for Optimizing Gene Expression in Eukaryotes

Organism	Region	Expression output	Design approach	Application	Success measure	Refs
Yeast	Promoter	Binned fluorescence (protein)	Iterated mutations	Design *de novo* promoters	Designed promoters span a ~20-fold dynamic range	[65]
Yeast	5′-UTR	Competitive growth assay (protein)	Iterated mutations	Optimize 5′-UTRs	Improved expression for ~85–95% of sequences	[49]
Chinese hamster ovarian cells	Promoter	RNA-seq (mRNA)	Element selection	Stable reporter production in batch culture	Designed promoter strengths were maintained after 7 days in culture	[8]
Yeast	5′-UTR	Binned fluorescence (protein)	Iterated mutations	Reliable *p*-coumaric acid production	*p*-Coumaric acid titer correlated well with predicted protein abundance	[9]
Yeast	5′-UTR	Binned fluorescence (protein)	Iterated mutations	Increased production of phloretin	The best result was 50% better than control	[54]

generally used in design: iterative guided mutations and element selection (Figure 2), but alternatives such as adaptive sampling [61], **generative adversarial networks (GANs)**, and paired deep learning models [62,63] are being developed. Note that searching the space of all possible sequences for maximum expression is computationally expensive; as a result, methods are often used that can only be guaranteed to find local (instead of global) maxima.

An algorithm involving iterated mutations, or *in silico* directed evolution [64], starts from a given sequence (or sequence library [54]) and iteratively modifies it. These initial sequences are usually chosen close to a known functional sequence or are based on how the design algorithm is being evaluated; for example, Curran and colleagues [65] aimed to compare their algorithm to an often-used promoter in traditional genetic engineering.

To evaluate the proposed mutations *in silico*, a trained ML model is required to predict expression activity. Modifications are kept small, such that each iteration will only cause a small shift in expression unless an important nucleotide is changed. This process can aim at a target expression



Trends in Biotechnology

Figure 2. Design Algorithm Approaches Using Trained Machine Learning (ML) Models. (A) Iterated mutations, repeatedly modifying a given sequence. A trained ML model is used to select modifications that bring the predicted expression closer to the target expression level. (B) Element selection, combining known elements and selecting combinations based on the expression predicted by a trained ML model.

level [9,65] or, as in the study by Cuperus and colleagues, at optimizing expression until a plateau is reached [49].

In element selection, a fixed set of elements is combined into a new regulatory region. This approach naturally pairs with a model trained on elements. Searching can be exhaustive because, for a reasonable number of elements, all possible combinations can be tested and a global optimum can be found. For example, the workflow applied by Brown and colleagues [8] used a pretrained linear regression model to predict expression for all possible combinations of their regulatory elements. Based on their design criteria, a small set of promoters were selected and synthesized. A downside of element selection is that its search space is limited to the predefined elements, and this can render some expression levels inaccessible or hamper the generation of different sequences attaining the same expression level.

Challenges and Opportunities

The success of models for the design of gene expression regulation depends on choices made in each of the four design cycle stages (Figure 1). These are not necessarily mutually exclusive: some stages allow for combining different approaches, for example, in DNA encoding [9,54] or in response measurement [65]. An important choice in sequence library generation and in the setup of the design algorithm is to what extent prior knowledge is considered. This influences ML model training and can help to constrain the sequence design search space. There is a clear trade-off between providing sufficient prior knowledge to allow accurate designs and providing so much knowledge that all designed sequences will be similar to known sequences and to each other. An important skill in building a successful ML model lies in the realization of ways to use knowledge about the system of interest (e.g., interactions between different features) to guide the learning process.

The sequence library generation step is crucial in that it should provide sufficiently accurate and diverse data for training. Libraries optimal for training differ from those generated to answer specific scientific questions, or to meet specific engineering goals, in that they should (at least partially) explore new parts of sequence space which are not yet well understood. Although the advantage of ML models is their flexibility to adapt to the data at hand, their dependency on data also imposes limitations. Several experimental issues can lead to biases in the training data: some sequences can be over- or under-represented in the training data because they may be easier or harder to synthesize or sequence; extremely high reporter expression may harm the cell, leading to fewer measurements in this range; some sequences may contain restriction sites, hampering transformation; and sequencing may introduce biases for particular sequence characteristics; etc. Other unforeseen *in vivo* interactions may introduce noise into the activity measurements, which is detrimental to training.

It is important to realize that most ML models assume that the training data are an independent sample that represents the distribution of interest. If a model is used in design to predict activity for a sequence that does not look anything like the training set, results are likely unreliable – extrapolation is harder than interpolation. Ideally, a model should be used that provides a measure of confidence in its output (e.g., a posterior probability), or should be combined with a domain description method that determines to what extent a new sequence is sufficiently close to training sequences to allow prediction. Even when deviation from the training sequences is tightly constrained, the design algorithm can accidentally introduce, as a side effect, a repressor site or an insulating sequence that was not observed during training. Extrapolation to organisms, genomic contexts, or even growth conditions other than those for which the training data were obtained is also risky. Regulatory sequences might appear to be (in)active or influence expression in ways not previously observed. An approach to deal with potential side effects is found in the work of Brown and colleagues [8], who used prior knowledge to hand-craft a set of promoters that would suit their needs, avoiding undesired interactions. Although prior knowledge should ideally be used to avoid erroneous designs, for the foreseeable future our knowledge of gene regulation will be too limited to avoid them completely, Therefore, *post hoc* interpretation will still be necessary.

A major advantage of ML is that it provides an unbiased look at the data and can design sequences that would never be designed by handcrafting a library. Moreover, when additional data become available (perhaps measured under different conditions), models can be retested and even retrained, in what is known as **active learning** [54]. A potential downside is that many ML models are 'black boxes', solving a prediction problem optimally but offering little insight into biology or even into the considerations underlying particular design choices. Interpretability, namely our ability to explain how a model arrived at a decision, is a topic of research in many application areas of ML. In general, the more complex a model becomes, the harder it is to interpret its decisions. In a linear regression model, the influence of a specific coefficient is clear because it, describes the relation between the corresponding input variable and the output. For tree- or **kernel-based models** there are established practices of interpretation [66], whereas for deep learning methods this is still ongoing research, as shown by the work of Shrikumar and colleagues [67,68] and Cuperus and colleagues [49]. Although model interpretability may not be an issue for engineering purposes, it is important to help to gain a better understanding of eukaryotic gene regulation. For an overview on what can be learned from a trained ML model in the context of human gene regulation, we refer the reader to a review by Li and colleagues [69].

Concluding Remarks and Future Perspectives

We have discussed the opportunities and potential pitfalls of using ML for designing regulatory regions for eukaryotes, and have reviewed several recent contributions to the field. There is clearly room for improvement and extension (see Outstanding Questions). First, current ML-based design papers often focus on the 5′-UTR because it is a small region, leading to a more manageable design space, and also has less potential for unforeseen interactions. In the future, we expect this to be extended to larger regulatory regions, and even to combinations of different regions (Box 1). Ultimately, given sufficient training data, it should be feasible to design regulatory regions for tuning the various genes in pathways or circuits, as has been demonstrated in prokaryotes [70,71]. Initial research by Zhou and colleagues used ML for combinatorial optimization of heterologous metabolic pathways in yeast [72]. Second, most research in this area focuses on unicellular yeasts, which have served as a model system for studying transcriptional regulation for many decades in view of their experimental accessibility. A potential issue is that their genome organization and regulatory repertoire may be too different from other branches of the eukaryotic tree that may be of interest in genetic engineering [73]. An intriguing possibility, however, is to use yeast as a design platform for orthogonal regulatory systems for transfer to other organisms. Alternatively, depending on their accessibility to genetic engineering tools, more training data should be generated for each organism and condition. One avenue of interest is the cross-species work done in prokaryotes by Kushwaha and Salis [74]. Given the less complex nature of gene regulation this did not require the use of ML, but in eukaryotes this may be unavoidable. An approach called **transfer learning** [75], in other words repurposing a ML model trained on one task for another, can help to avoid the need to start from scratch.

As in many areas of science and society, the combination of the availability of large datasets and strong ML methods is set to have a major impact in biotechnology, moving from knowledge-based to data-based approaches to problem solving [76]. Most recently, deep learning has

Outstanding Questions

In developing regulatory region design algorithms, how should we strike a good balance between providing prior knowledge to models and allowing them to learn from data in an unbiased manner? What other, possibly biologically inspired, methods can we use to efficiently search sequence space at the nucleotide level?

If we want to exploit the power of deep learning to gain a deeper understanding of eukaryotic gene regulation, what datasets should we ideally measure to allow its application? To what extent will interpretability be important for synthetic biologists?

How can lessons learned from the expression of single genes be used to optimize sets of regulators to control the expression of entire circuits or pathways? Can we design individual regulatory regions to function as isolated modules, or will crosstalk make this impossible and force us to optimize the overall system?

How can we take into account important additional experimental issues such as robustness to changing conditions and transient perturbations?

To what extent does a design success obtained in yeast translate to other eukaryotes? Will the increased complexity of regulation in higher eukaryotes be manageable? How should we integrate dynamic information such as chromatin conformation?

received much attention [77]. Even though trained deep learning models are notoriously hard to interpret, it is clear that they are also finding their way into this application area [49,62]. Whether or not these methods offer advantages at current dataset sizes remains to be seen. An interesting development is to use deep learning networks directly, through gradient-based optimization, to modify an input sequence to one that achieves a desired output, a process called 'deep dreaming'. This approach was used by Killoran and colleagues [62] to navigate regulatory sequence space.

The unbiased data-driven approach of ML offers intriguing opportunities for increasing our understanding of gene regulation, provided that we can open the black boxes that many models are. Even without this ability, genetic engineers striving to gain better control over their expression systems can benefit from using ML models trained on comprehensive sequence–activity datasets to generate specific parts at will.

References

1. Lelli, K.M. *et al.* (2012) Disentangling the many layers of eukaryotic transcriptional regulation. *Annu. Rev. Genet.* 46, 43–68
2. Delaneau, O. *et al.* (2019) Chromatin three-dimensional interactions mediate genetic effects on gene expression. *Science* 364, eaat8266
3. Andreev, D.E. *et al.* (2017) Insights into the mechanisms of eukaryotic translation gained with ribosome profiling. *Nucleic Acids Res.* 45, 513–526
4. Demain, A.L. and Vaishnav, P. (2009) Production of recombinant proteins by microbes and higher organisms. *Biotechnol. Adv.* 27, 297–306
5. Si, T. *et al.* (2017) Automated multiplex genome-scale engineering in yeast. *Nat. Commun.* 8, 15187
6. Dai, Z. *et al.* (2015) Yeast synthetic biology for high-value metabolites. *FEMS Yeast Res.* 15, 1–11
7. Walker, R. and Pretorius, I. (2018) Applications of yeast synthetic biology geared towards the production of biopharmaceuticals. *Genes* 9, 340
8. Brown, A.J. *et al.* (2017) In silico design of context-responsive mammalian promoters with user-defined functionality. *Nucleic Acids Res.* 45, 10906–10919
9. Decoene, T. *et al.* (2018) Toward predictable 5′UTRs in *Saccharomyces cerevisiae*: development of a yUTR calculator. *ACS Synth. Biol.* 7, 622–634
10. Hohnholz, R. and Achstetter, T. (2019) Recombination in yeast based on six base pairs of homologous sequences: structural instability in two sets of isomeric model expression plasmids. *Yeast* Published online April 16, 2019. https://doi.org/10.1002/yea.3393
11. Salis, H.M. *et al.* (2009) Automated design of synthetic ribosome binding sites to control protein expression. *Nat. Biotechnol.* 27, 946–950
12. Espinar, L. *et al.* (2018) Promoter architecture determines cotranslational regulation of mRNA. *Genome Res.* 28, 509–518
13. Mudge, J.M. and Harrow, J. (2016) The state of play in higher eukaryote gene annotation. *Nat. Rev. Genet.* 17, 758–772
14. Kulmanov, M. *et al.* (2018) DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics* 34, 660–668
15. Almagro Armenteros, J.J. *et al.* (2019) SignalP 5.0 improves signal peptide predictions using deep neural networks. *Nat. Biotechnol.* 37, 420–423
16. Evans, R. *et al.* (2018) *De novo* structure prediction with deep-learning based scoring. In *Critical Assessment of Techniques for Protein Structure Prediction (CASP13), A7D*, Protein Structure Prediction Center
17. Alipanahi, B. *et al.* (2015) Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nat. Biotechnol.* 33, 831–838
18. Hashemifar, S. *et al.* (2018) Predicting protein–protein interactions through sequence-based deep learning. *Bioinformatics* 34, i802–i810
19. Sharon, E. *et al.* (2012) Inferring gene regulatory logic from high-throughput measurements of thousands of systematically designed promoters. *Nat. Biotechnol.* 30, 521–530
20. Shen, S.Q. *et al.* (2016) Massively parallel cis-regulatory analysis in the mammalian central nervous system. *Genome Res.* 26, 238–255
21. Maricque, B.B. *et al.* (2017) A genome-integrated massively parallel reporter assay reveals DNA sequence determinants of cis-regulatory activity in neural cells. *Nucleic Acids Res.* 45, e16
22. Melnikov, A. *et al.* (2012) Systematic dissection and optimization of inducible enhancers in human cells using a massively parallel reporter assay. *Nat. Biotechnol.* 30, 271–277
23. Patwardhan, R.P. *et al.* (2012) Massively parallel functional dissection of mammalian enhancers *in vivo*. *Nat. Biotechnol.* 30, 265–270
24. White, M.A. (2015) Understanding how cis-regulatory function is encoded in DNA sequence using massively parallel reporter assays and designed sequences. *Genomics* 106, 165–170
25. Myint, L. *et al.* (2019) Linear models enable powerful differential activity analysis in massively parallel reporter assays. *BMC Genomics* 20, 209
26. Levo, M. and Segal, E. (2014) In pursuit of design principles of regulatory sequences. *Nat. Rev. Genet.* 15, 453–468
27. de Boer, C.G. *et al.* (2018) Deciphering eukaryotic cis-regulatory logic with 100 million random promoters. *bioRxiv* Published online March 7, 2018. https://doi.org/10.1101/224907
28. Shalem, O. *et al.* (2015) systematic dissection of the sequence determinants of gene 3' end mediated expression control. *PLoS Genet.* 11, e1005147
29. Weingarten-Gabbay, S. *et al.* (2019) Systematic interrogation of human promoters. *Genome Res.* 29, 171–183
30. Chen, M. *et al.* (2013) Decoupling epigenetic and genetic effects through systematic analysis of gene position. *Cell Rep.* 3, 128–137
31. Maricque, B.B. *et al.* (2019) A massively parallel reporter assay dissects the influence of chromatin structure on cis-regulatory activity. *Nat. Biotechnol.* 37, 9
32. Kinney, J.B. *et al.* (2010) Using deep sequencing to characterize the biophysical mechanism of a transcriptional regulatory sequence. *P. Natl. Acad. Sci. USA* 107, 9158–9163
33. Lubliner, S. *et al.* (2015) Core promoter sequence in yeast is a major determinant of expression level. *Genome Res.* 25, 1008–1017
34. Quax, T.E.F. *et al.* (2015) Codon bias as a means to fine-tune gene expression. *Mol. Cell* 59, 149–161
35. Cuperus, J.T. *et al.* (2015) A tetO toolkit to alter expression of genes in *Saccharomyces cerevisiae*. *ACS Synth. Biol.* 4, 842–852

36. Gärtner, T. (2003) A survey of kernels for structured data. *ACM SIGKDD Explor Newsl.* 5, 49–58
37. de Boer, C.G. *et al.* (2014) A unified model for yeast transcript definition. *Genome Res.* 24, 154–166
38. Grossman, S.R. *et al.* (2017) Systematic dissection of genomic features determining transcription factor binding and enhancer function. *Proc. Natl. Acad. Sci. U. S. A.* 114, E1291–E1300
39. Levo, M. *et al.* (2017) Systematic investigation of transcription factor activity in the context of chromatin using massively parallel binding and expression assays. *Mol. Cell* 65, 604–617
40. Liu, B. (2017) BioSeq-Analysis: a platform for DNA, RNA and protein sequence analysis based on machine learning approaches. *Brief. Bioinform.* 2017, bbx165
41. Portela, R.M.C. *et al.* (2017) Synthetic core promoters as universal parts for fine-tuning expression in different yeast species. *ACS Synth. Biol.* 6, 471–484
42. Smith, R.P. *et al.* (2013) Massively parallel decoding of mammalian regulatory sequences supports a flexible organizational model. *Nat. Genet.* 45, 1021–1028
43. Zeevi, D. *et al.* (2014) Molecular dissection of the genetic mechanisms that underlie expression conservation in orthologous yeast ribosomal promoters. *Genome Res.* 24, 1991–1999
44. Beer, M.A. and Tavazoie, S. (2004) Predicting gene expression from sequence. *Cell* 117, 185–198
45. Dvir, S. *et al.* (2013) Deciphering the rules by which 5′-UTR sequences affect protein expression in yeast. *Proc. Natl. Acad. Sci. U. S. A.* 110, E2792–E2801
46. Lubliner, S. *et al.* (2013) Sequence features of yeast and human core promoters that are predictive of maximal promoter activity. *Nucleic Acids Res.* 41, 5569–5581
47. Siwo, G. *et al.* (2016) Prediction of fine-tuned promoter activity from DNA sequence. *F1000Res* 5, 158
48. Ukkonen, E. (1992) Approximate string matching with q-grams and maximal matches. *Theor. Comput. Sci.* 92, 191–211
49. Cuperus, J.T. *et al.* (2017) Deep learning of the regulatory grammar of yeast 5′ untranslated regions from 500,000 random sequences. *Genome Res.* 27, 2015–2024
50. Liu, B. *et al.* (2017) A multi-modal neural network for learning cis and trans regulation of stress response in yeast. In *Conference on Neural Information Processing Systems: Machine Learning in Computational Biology (MLCB) Workshop*, pp. 5, NIPS
51. Movva, R. *et al.* (2019) Deciphering regulatory DNA sequences and noncoding genetic variants using neural network models of massively parallel reporter assays. *PLoS One* 14, e0218073
52. Xie, R. *et al.* (2017) A deep auto-encoder model for gene expression prediction. *BMC Genomics* 18, 845
53. Zeng, H. *et al.* (2017) Accurate eQTL prioritization with an ensemble-based framework. *Hum. Mutat.* 38, 1259–1265
54. Ding, W. *et al.* (2018) Engineering the 5′ UTR-mediated regulation of protein abundance in yeast using nucleotide sequence activity relationships. *ACS Synth. Biol.* 7, 2709–2714
55. Weenink, T. *et al.* (2018) Design of RNA hairpin modules that predictably tune translation in yeast. *Synth. Biol.* 3, ysy019
56. Slutskin, I.V. *et al.* (2018) Unraveling the determinants of microRNA mediated regulation using a massively parallel reporter assay. *Nat. Commun.* 9, 529
57. Portela, R.M.C. *et al.* (2018) *Pichia pastoris* alcohol oxidase 1 (aox1) core promoter engineering by high resolution systematic mutagenesis. *Biotechnol. J.* 13, 1700340
58. Yan, C. *et al.* (2018) systematic study of nucleosome-displacing factors in budding yeast. *Mol. Cell* 71, 294–305
59. Yang, J. *et al.* (2018) Controlling AOX1 promoter strength in *Pichia pastoris* by manipulating poly (dA:dT) tracts. *Sci. Rep.* 8, 1401
60. Morse, N.J. *et al.* (2017) Yeast terminator function can be modulated and designed on the basis of predictions of nucleosome occupancy. *ACS Synth. Biol.* 6, 2086–2095
61. Brookes, D.H. and Listgarten, J. (2018) Design by adaptive sampling. *arXiv* Published online October 31, 2018. https://arxiv.org/abs/1810.03714
62. Killoran, N. *et al.* (2017) Generating and designing DNA with deep generative models. *arXiv* Published online December 17, 2017. http://arxiv.org/abs/1712.06148
63. Gupta, A. and Zou, J. (2018) Feedback GAN (FBGAN) for DNA: a novel feedback-loop architecture for optimizing protein functions. *arXiv* Published online April 5, 2018. http://arxiv.org/abs/1804.01694
64. Dougherty, M.J. and Arnold, F.H. (2009) Directed evolution: new parts and optimized function. *Curr. Opin. Biotech.* 20, 486–491
65. Curran, K.A. *et al.* (2014) Design of synthetic yeast promoters via tuning of nucleosome architecture. *Nat. Commun.* 5, 4002
66. Molnar, C. (2019) *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*, Lulu Publishing
67. Shrikumar, A. *et al.* (2017) Learning important features through propagating activation differences. *arXiv* Published online April 10, 2017. http://arxiv.org/abs/1704.02685
68. Shrikumar, A. *et al.* (2018) TF-MoDISco v0.4.4.2-alpha: technical note. *arXiv* Published online 31 October 2018. http://arxiv.org/abs/1811.00416
69. Li, Y. *et al.* (2015) The identification of cis-regulatory elements: A review from a machine learning perspective. *Biosystems* 138, 6–17
70. Zelcbuch, L. *et al.* (2013) Spanning high-dimensional expression space using ribosome-binding site combinatorics. *Nucleic Acids Res.* 41, e98
71. Farasat, I. *et al.* (2014) Efficient search, mapping, and optimization of multi-protein genetic systems in diverse bacteria. *Mol. Syst. Biol.* 10, 731
72. Zhou, Y. *et al.* (2018) MiYA, an efficient machine-learning workflow in conjunction with the YeastFab assembly strategy for combinatorial optimization of heterologous metabolic pathways in *Saccharomyces cerevisiae*. *Metab. Eng.* 47, 294–302
73. Marinov, G.K. and Kundaje, A. (2018) ChIP-ping the branches of the tree: functional genomics and the evolution of eukaryotic gene regulation. *Brief. Funct. Genom.* 17, 116–137
74. Kushwaha, M. and Salis, H.M. (2015) A portable expression resource for engineering cross-species genetic circuits and pathways. *Nat. Commun.* 6, 7832
75. Weiss, K. *et al.* (2016) A survey of transfer learning. *J. Big Data* 3, 9
76. de Ridder, D. (2018) Artificial intelligence in the lab: ask not what your computer can do for you. *Microb. Biotechnol.* 12, 38–40
77. Webb, S. (2018) Deep learning for biology. *Nature* 554, 555
78. Redden, H. and Alper, H.S. (2015) The development and characterization of synthetic minimal yeast promoters. *Nat. Commun.* 6, 7810
79. Ito, Y. *et al.* (2013) Characterization of five terminator regions that increase the protein yield of a transgene in *Saccharomyces cerevisiae*. *J. Biotechnol.* 168, 486–492
80. Goodfellow, I.J. *et al.* (2014) Generative adversarial networks. *arXiv* Published online June 10, 2014. https://arxiv.org/abs/1406.2661
81. Cortes, C. and Vapnik, V. (1995) Support-vector networks. *Mach. Learn.* 20, 273–297
82. Jordan, M.I. and Mitchell, T.M. (2015) Machine learning: trends, perspectives, and prospects. *Science* 349, 255–260
83. Jain, A.K. *et al.* (2000) Statistical pattern recognition: a review. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* (22), pp. 4–37, IEEE