npj | systems biology and applications

Article

Published in partnership with the Systems Biology Institute

https://doi.org/10.1038/s41540-026-00683-6

Generative design of synthetic gene circuits for functional and evolutionary properties

Olivia Gallup ✉ & Harrison Steel

In the past decades, a wide suite of design tools for biological systems has been developed, but using these to create biotechnologies that achieve reliable and predictable behaviour remains challenging. Modelling approaches have enabled researchers to traverse the vast search space of genetic circuits more efficiently, while machine learning has proven useful for designing parts and predicting their function or evolutionary properties. Generative algorithms have the potential to leverage these features to design entire genetic circuits from the sequence level, but have only recently begun to be applied to synthetic biological applications. Here, we show that even simple generative models like the conditional variational autoencoder (CVAE) can produce novel genetic circuits that match complex dynamic functions such as signal adaptation. Using in silico RNA simulation, we construct a dataset of RNA sequences and convert them to circuits via RNA interaction predictors, allowing us to estimate functional features alongside evolutionary stability and interpret model-learned features. Our model generates diverse distributions of circuits that match their target adaptation specification well, even when limited to small training data sets. Structures in the embedding space correspond to motifs previously identified as crucial for adaptation and reflect the design rules for adaptable circuits. Framing adaptation as a single design objective outperforms other input representations, reflecting the importance of choosing the correct data encoding for generating genetic circuits. Finally, we show that functional and evolutionary properties can be prompted simultaneously, providing a proof-of-concept for the combined design of phenotype and evotype.

Biological systems must constantly adjust to changes in their environment and return to homoeostasis after perturbations. Adapting to an external signal involves returning a system back to its original state before it was perturbed; this functionality is widely used in natural systems where precise sensing or control is required[1–4], and can be realised by a surprisingly small number of interacting components in an organism[5,6]. In synthetic biology, designing genetic circuits that can adapt to disturbances is crucial for stable functioning, as biodesigns are applied in different applications or environments[7,8].

However, biological circuits can fail for many reasons[6,9,10], such as unforeseen host interactions, parameter-fragile designs, or mutations. As an example, designing a circuit that adapts to perturbations can be achieved with motifs such as incoherent feed-forward loops or negative feedback with a buffer node[5,11–14], yet functionality is often disrupted by host context and in general constrained by practical requirements (such as non-repetitive sequences, low crosstalk, assembly compatibility, and synthesis ease). Design optimisation can encode such requirements in objective (cost) functions[15–17], but some such properties entail significant computational hurdles, such as characterising mutation effects and the surrounding evolutionary landscape. Meanwhile, changing the target specification (e.g. desired adaptation level) often forces a fresh search.

Currently, designing evolutionary properties is most important in directed evolution, where starting variants must be functional yet still evolvable. In synthetic biology, the aim is typically mutation resistance—coupling function to an essential process[18] or imposing growth costs on burden-reducing mutations[19]. Nevertheless, the importance of tuning a circuit's evolutionary 'evotype' alongside its physical 'phenotype' is becoming increasingly recognised[20]. In nature, distinct functions that are separated by small evolutionary distances allow populations to adapt quickly to new environments. A study on the LacI repressor in *E. coli* found that a few mutations could transform its function entirely, yielding inverse dose response or band-pass behaviours, all only a few amino-acid steps

Department of Engineering Science, University of Oxford, Oxford, UK. ✉e-mail: olivia.gallupova@eng.ox.ac.uk

away[21]. In this way, the evolutionary landscape around a genetic circuit can hold a plethora of hidden functional possibilities.

To make building genetic circuits more predictable, researchers developed a variety of mathematical modelling approaches. Algorithms for exploring possible architectures often rely on parameter-level approaches, such as exploring biological part libraries[22,23], optimising directly from a starting set using gradient descent[24], or probabilistic approaches[25,26]. Alongside optimisation, other approaches focus on building predictors, such as physics-based simulators that determine RNA structure and interactions[27,28]. Some tools utilise nucleotide or amino acid sequences directly[28–30] and have become standard in synthetic biology[31–34].

Recently, machine learning (ML) predictors have become widely used for their versatility and accuracy. In biology, applications include tuning expression levels[35–38], building functional switches[39] and linking composition to function[40,41]. These models can be integrated into existing tools for building complex circuits, such as logic gates, from libraries of characterised biological parts[42–47]. Beyond prediction, generative ML models have been used to propose entirely new or modified parts, such as designing DNA and RNA sequences to boost transcription[36,48] and translation[49,50], creating diverse protein libraries[51–53], and elucidating metabolic function[54] and mutation effects[35,55]. Hybrids of these optimisation, physics-based and generative approaches could markedly improve the reliability of biodesign and be useful for downstream applications[56].

So far, generative models have been deployed in synthetic biology to design individual parts[48,57], with the aspiration of designing entire circuits[40,58]. Optimising biological parts (riboswitches, promoters, or repressor proteins) typically requires a model to recognise sequence features that determine the binding interaction. However, designing genetic circuits goes a step further and requires knowledge of features underlying a dynamic response as a result of the binding interactions. Currently, data availability constraints often diminish the impact of ML models in biology, which require large and diverse datasets and become less accurate when applied to situations that significantly differ from those they are trained on[59,60]. Understanding these limitations is critical for the field to develop and become widely adopted—confidence in models is crucial for their deployment in industrial settings, where inaccurate model predictions can have high time- and financial-cost.

To address these challenges, in this paper, we develop a generative model that outputs new genetic circuit topologies when prompted with a target dynamic function. The work has two goals: first, to explore the effectiveness and requirements for generative models to accomplish a synthetic biological design task, and second, to interpret model features and assess their relevance to biological features known to relate to functionality. Since adaptation has been studied in depth for biological networks, we can compare our expectations against the features the model actually utilises. Following this, we further explore whether the generative model can learn evolutionary properties and thus generate circuits that are adaptable and evolvable.

To avoid the throughput limitations of physical experiments and explore a vast number of mutations, we simulated a large dataset of genetic circuits. To ground this in a biologically grounded question of widespread interest, we used an RNA simulator (IntaRNA[27]) to define genetic circuits as RNA circuits and simulated their dynamics. We then trained a simple conditional variational autoencoder (CVAE) on these genetic circuits topologies, using their adaptation levels as labels. A VAE is a probabilistic encoder-decoder model in which the encoder maps each input to a distribution in a lower-dimensional latent space and the decoder samples from that distribution to reconstruct the input. A CVAE extends this framework by conditioning both the encoder and decoder on additional information (such as labels), thus allowing the model to generate outputs that match a specified condition[61]. We found that even this small and simple machine learning model could generate new circuits with a desired level of adaptation. Deeper analysis of the model's embedding space confirmed that the CVAE had successfully captured the specific network motifs that are essential for adaptation[5,12,13]. We found that the training dataset can be as small as 2000 samples to achieve high-quality generative results, which has important practical implications for experimental approaches. Finally, by combining prompts for adaptation with those for evolutionary stability, we were able to generate circuits that were not only adaptable but also either evolvable or stable against mutations, with meaningful representations for evolutionary stability.

Results

We first construct a digital environment for genetic circuits that enables both dynamical simulation and exploration of the evolutionary landscape. In this framework, a genetic circuit is represented as a network whose nodes are biological components (RNA molecules) and whose edges encode pairwise binding strengths. While the CVAE does not take RNA sequences as inputs but only topological parameters, we chose to define genetic circuits in a more concrete way as RNA circuits, as this determines the range for plausible interactions, along with limitations to the equations governing dynamic behaviour. Here, we use 3 nodes (e.g. 3 RNA molecules) per circuit with the binding parameters $[k_{11}, k_{12}, k_{13}, k_{22}, k_{23}, k_{33}]$. Restricting to 3 nodes facilitates comparability with existing literature on adaptation in biological networks[5,12] and reasonably fast computation times. We focus on RNA molecules because robust physics-based simulators of RNA interactions are widely available. For each RNA pair, interaction strength is quantified as the minimum free binding energy predicted by IntaRNA 2.0[27] (Fig. 1a); intuitively, more negative energies correspond to more stable, and therefore stronger, binding. Individual RNAs are generated by randomly sampling nucleotide sequences from a constrained sequence space (see 'Methods'), and these sequences are then used to compute all pairwise interactions with IntaRNA. To simulate circuit dynamics and assess functional properties such as adaptation, we map these energies to reaction rates using experimental parameters from Na et al.[31]. Finally, we simulate the dynamics of all circuits, designating one RNA (node 1) as the signal 'input' and another (node 3) as the 'output'.

By simulating each genetic circuit's dynamics (modelled as RNA concentration trajectories) after adding a step input as the signal, we compute descriptive metrics for every RNA in the circuit, including steady states, overshoot, settling time, signal sensitivity and signal precision[5]. Sensitivity and precision together quantify adaptation, whereby an adaptable circuit responds strongly to the input (high sensitivity) and subsequently returns to its pre-stimulus level (high precision). We therefore characterise adaptability using the definitions in ref. 5 (illustrated in Fig. 1a; formal definitions in Methods) and adopt adaptation as the target function of the circuit (i.e. design objective), as it is well understood yet sufficiently complex to serve as a proof of concept.

In Ma et al.[5], 'effective' adaptation is defined by sensitivity ≥10 and precision ≥1 (dotted red boundary in Fig. 1b). Such motifs are rare (~0.01% of random topologies) and enact integral control, whereby feedback from an input signal is captured by the circuit components and modulates the output based on the area-under-the-curve error between real and target output[62], accounting for the length and intensity of deviation from the desired level. Because this binary criterion discards gradations below the thresholds, we instead use a continuous objective that combines sensitivity and precision into a smooth radial field centred at high values and increasing toward the threshold region (shaded in Fig. 1b; full definition in 'Methods'). This enables circuit designs to climb smoothly toward the optimum across the design space.

Beyond the adaptation-capable region, the space of RNA sequences that interact at all is also sparse. Random sampling mostly yields inert circuits, where components show negligible binding in IntaRNA with minimum free energies near 0. To address this, we supplement the sequence-sampled dataset with a parameter-sampled dataset that draws directly from possible circuit parameters and assigns a minimum free energy to each interaction (Fig. 1b, right). This allows us to bypass sequence sampling entirely but remain in the range of plausible interaction values. The generative model is trained on both datasets, representing each genetic circuit as a numerical matrix of binding strengths.

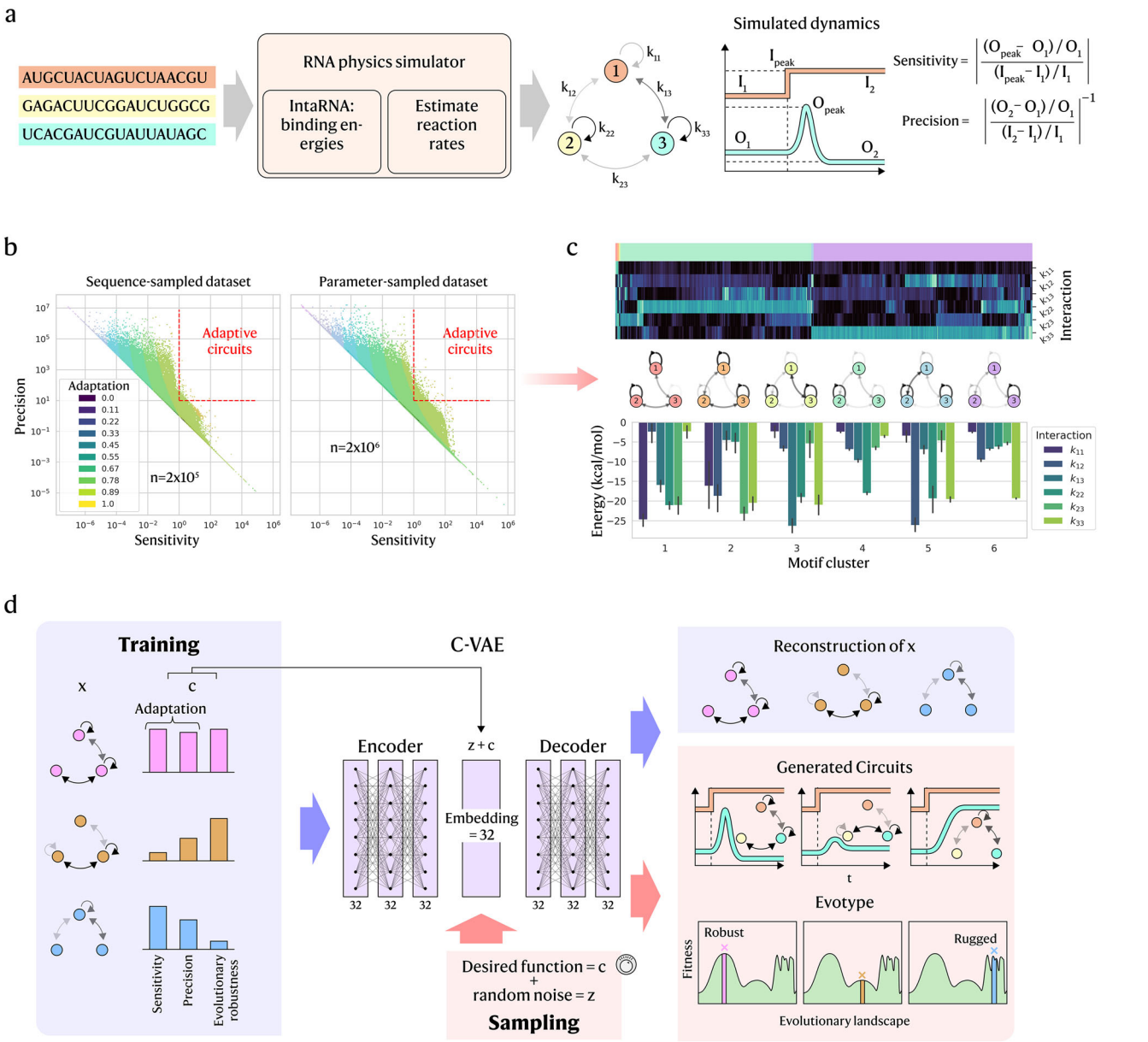

Fig. 1 | Overview of gene circuit simulation and CVAE model. a In silico RNA circuit dataset construction: RNA sequences are randomly generated as circuit constituents, then their interactions are simulated with IntaRNA 2.0 and used to estimate reaction rates. From this, the circuit dynamics are simulated, and a signal response is recorded to calculate sensitivity and precision, among other functional attributes. **b** The custom adaptation function is defined by sensitivity and precision, plotted on a log scale for the training dataset, which is composed of a sequence-sampled (2×10^5 samples) and a parameter-sampled (2×10^6 samples) dataset. The dashed red line denotes the threshold for adaptation as per[5] (sensitivity > 10, precision > 1). **c** Hierarchical clustering (top) on the interactions of the circuits that fall into the adaptable region identifies several motifs for achieving adaptation. Network diagrams (middle) show the 3-node circuits with the distributions of the interactions present in each motif group (bottom). Interaction arrows in the diagram represent the median binding energy of each group. **d** Training and sampling the CVAE: for training (blue boxes/arrows), the circuit interactions (vector x) are concatenated with the functional properties (vector c) and passed through the encoder. The resulting embedding z is then concatenated again with c and passed through the decoder, which outputs reconstructions of the input circuits. For sampling (red boxes/arrows), a random noise vector instead takes the place of z and is concatenated with c, which now serves as the prompt, before being passed through the decoder. The generated circuits are simulated to ascertain their dynamic and evolutionary properties.

Since we expect the generative model to recover adaptive motifs, we first distil candidate network motifs from the training data via hierarchical clustering to serve as baselines for comparison (Fig. 1c). Prior work on enzyme networks shows that adaptability requires at least one of two motif classes, including buffer/opposer or proportioner/balancer[5,11–13]. Although RNA circuits interact only via mutual binding (not explicit activation or repression), the same topological rules apply if we view RNA circuits as enzyme networks restricted to mutual repression. In both motif classes, adaptation emerges through integral action, where one or more nodes temporarily accumulate and then release concentration, driving the output back to the initial concentration. This can arise readily in RNA circuits from self-interactions and imbalanced reaction rates (see Supplementary Note 1), so we expect the generative model to exploit such motifs.

For the generative model, we use a variational autoencoder (VAE) as a simple, strong baseline[54,63], and adopt its conditional form (CVAE) to target specific circuit functions[61,64]. In training (Fig. 1d), the encoder receives the vector x concatenated with the label c, where x encodes the circuit topology via interaction parameters $[k_{11}, k_{12}, k_{13}, k_{22}, k_{23}, k_{33}]$ and c is the adaptation

metric; both are normalised to [0, 1]. The encoder outputs a 32-dimensional latent representation z. Because CVAEs are probabilistic, z parametrises a distribution conditioned on c, with a KL-divergence term encouraging distinct latent representations for different prompts. The decoder then takes $[z, c]$ to reconstruct the original circuit, scored by a reconstruction loss (see 'Methods'). We inject c at the latent stage so that the trained decoder can be used alone later for generative purposes. Overall, training organises the latent space so circuits with similar dynamics cluster together, enabling the decoder to sample new circuits that achieve desired functions.

After training, sampling new circuits is straightforward. We supply the generative model with a target function (e.g. a desired adaptation value) and obtain a circuit as a set of binding parameters predicted to meet that target. At sampling (Fig. 1d), the encoder is bypassed, and the prompt is fed directly to the trained decoder, with random noise used in place of the latent z to produce diverse samples. Because the encoder learned a conditional distribution, the decoder remains sensitive to prompts. We then simulate each generated circuit's dynamics with the same numerical pipeline used in training to evaluate prompt adherence. We next report training results before analysing accuracy and adherence in detail.

For adaptation, the CVAE reconstructs circuits with high accuracy ($R^2 = 0.97$, Fig. 2a) and generates diverse circuits across prompts, despite its simple architecture. We assess model quality based on how well the CVAE reconstructs circuits and how specifically it responds to prompts. While samples need not match a prompt exactly since sampling noise induces variation, the generated distributions should be centred around the prompt.

To evaluate prompt adherence, we sampled 10,000 circuits across 10 adaptation prompts. Since inputs were normalised to [0, 1], we can push the generative model to extrapolate outside of the training range with prompts in the range [−0.2, 1.2] (vertical lines in Fig. 2b). A weak model would yield similar adaptation distributions for all prompts, while a strong one would peak near each target. Our CVAE produces distinct distributions with peaks near their prompts that increase monotonically with prompt order (Fig. 2b) and still generates some circuits beyond the training range, despite sparsity in training data.

Although the prompt-wise distributions peak near their targets, some are broad, especially for lower prompts <0.5, which show long tails (Fig. 2b). Plotting sensitivity and precision separately (Fig. 2c) shows strong overall alignment with targets but a slight high-end bias, as some circuits prompted with adaptation = 1 reach higher sensitivity/precision than those prompted with >1, suggesting limited extrapolation beyond the training range. We also see a peak at $s = 1, p = 1$ for medium-high prompts, likely due to overrepresented weak-response circuits and sparse highly adaptable ones. These patterns reflect biases from both the quantitative objective (our adaptation metric) and training-data imbalance, a common challenge in computational biology[60].

Despite these biases, the CVAE learns meaningful latents that cluster by adaptation level (Fig. 2d right). We project embeddings with uniform manifold approximation and projection (UMAP) to assess structure. A naive UMAP on raw circuit parameters showed only weak clustering with substantial overlap in different levels of adaptation, implying richer representations are needed (Fig. 2d left; see Supplementary Fig. 1 for sensitivity and precision overlay). In contrast, the UMAP of CVAE latents for 10,000 generated circuits separates circuits by adaptation, providing evidence that the model has learned to relate genetic circuit features to functional characteristics. Overlaying interaction energies (Fig. 2e) highlights which parameters drive clustering—k_{11} and k_{33} are most determining, whereas k_{13} and the self-interaction k_{22} vary within clusters. Self- and cross-interactions at the input/output nodes (RNA_1 and RNA_3) thus primarily set adaptation in the latent space, while interactions with the auxiliary node RNA_2 capture finer differences among adaptation-capable motifs consistent with previous literature[5,13].

To examine adaptation-capable motifs, we analysed the dynamics and topology of the most adaptable circuit from each latent cluster (black circles in Fig. 2d, f). The cluster with circuit A contains highly adaptive circuits, while clusters with B and C show medium-high adaptation and clusters with D and E show medium-low adaptation (see trajectories in Fig. 2f). Circuits A–C reach similar adaptation via distinct topologies, whereas D and E respond to the signal but fail to adapt well. UMAPs overlaid with interaction parameters confirm that clusters map to distinct motif classes, indicating that the latent space encodes meaningful diversity.

These cases reveal clear embedding-space trends, showing that the model learns broadly informative representations, though some motif classes remain partially overlapping. Furthermore, the robustness of circuit architectures influences how well the generative model can distinguish it from similar designs. Motifs that are evolutionarily unstable (functional yet on rugged fitness landscapes) sit only a few mutations from non-functionality and are easier to misclassify than robust, constrained motifs. At the same time, the generative models must not miss these unstable motifs. Next, we probe factors that shape generative performance by varying hyperparameters and prompt formats.

Generative model performance acutely depends on many hyperparameters spanning training, optimisation, losses, and architecture, particularly when data is limited. We initially performed broad parameter sweeps to set appropriate ranges (see 'Methods') before focusing on the most consequential factors, including KL-divergence weight, training dataset size, and the objective function. We compare settings by reconstruction accuracy (R^2) and prompt adherence (Fig. 3), evaluating the latter by how distinctly prompt-wise distributions separate across different prompts.

A core feature of VAEs, the KL divergence weight sets how strongly we penalise similar embeddings across different prompts. Parameter sweeps (Fig. 3a) reveal a narrow workable range; model accuracy collapses near 10^{-3}, while prompt adherence is high for weights 10^{-5}–10^{-4} (low overlap among prompt-wise distributions). We then tested data requirements (Fig. 3b) and found that although the sequence-sampled and parameter-grid datasets respectively contain ~100,000 and ~1,000,000 samples, as few as ~1500 training points achieved high R^2 without degrading prompt adherence. In particular, prompt adherence varies with the random initialisation seed, which presents considerable implications for using generative models in experimental settings (illustrated in Supplementary Fig. 8). We found that mitigation strategies such as training with multiple seeds or (to a lesser extent) including contrastive loss were helpful, while regularisations (e.g. L2 regularisation) and architectural adjustments (e.g. changing hidden size) did not make a large difference. A lack of convergence in the output can be identified by assessing the variability in generated circuit topologies before testing them in the lab, although looking at the variability between distributions of individual circuit parameters can be misleading.

Beyond training parameters, we find that the choice of objective function strongly shapes learning. Our custom adaptation metric collapses sensitivity and precision into a single value, masking different (S, P) combinations. We therefore compared this to multi-parameter objectives that treat metrics separately (Fig. 3c): a continuous target $[\log_{10}S, \log_{10}P]$; a binary categorisation $[\log_{10}S > 0, \log_{10}P > 1]$[5]; and a three-metric target $[\log_{10}S, \log_{10}P,$ overshoot/initial], motivated by the link between overshoot and adaptability. For each objective, we trained 60 models with different seeds and generated 3000 circuits using prompts in [−0.2, 1.2]. For the binary case, we also tested intermediate prompts to assess whether the generative model implicitly learned beyond just categorical labels. To limit combinatorial explosion among prompts, we used 10 prompts for the unified adaptation objective and 5 prompts per dimension for multi-objectives (e.g. $5^2 = 25$ for $[\log_{10}S, \log_{10}P]$ and $5^3 = 125$ for the three-metric target).

Because prompting differs across objectives, we cannot compare models with the overlap of prompted distributions. Instead, we map each multi-objective prompt to an equivalent adaptation value (combining sensitivity and precision) and compare that to the true adaptation of generated circuits (Fig. 3c). For $[\log_{10}S, \log_{10}P]$, distributions are separated but in the wrong order relative to their targets, likely because the precision part of the prompt influences the output disproportionately (Supplementary Fig. 2). Adding overshoot ($[\log_{10}S, \log_{10}P,$ overshoot/initial]) does not substantially improve prompt adherence (3c), although the sensitivity part of

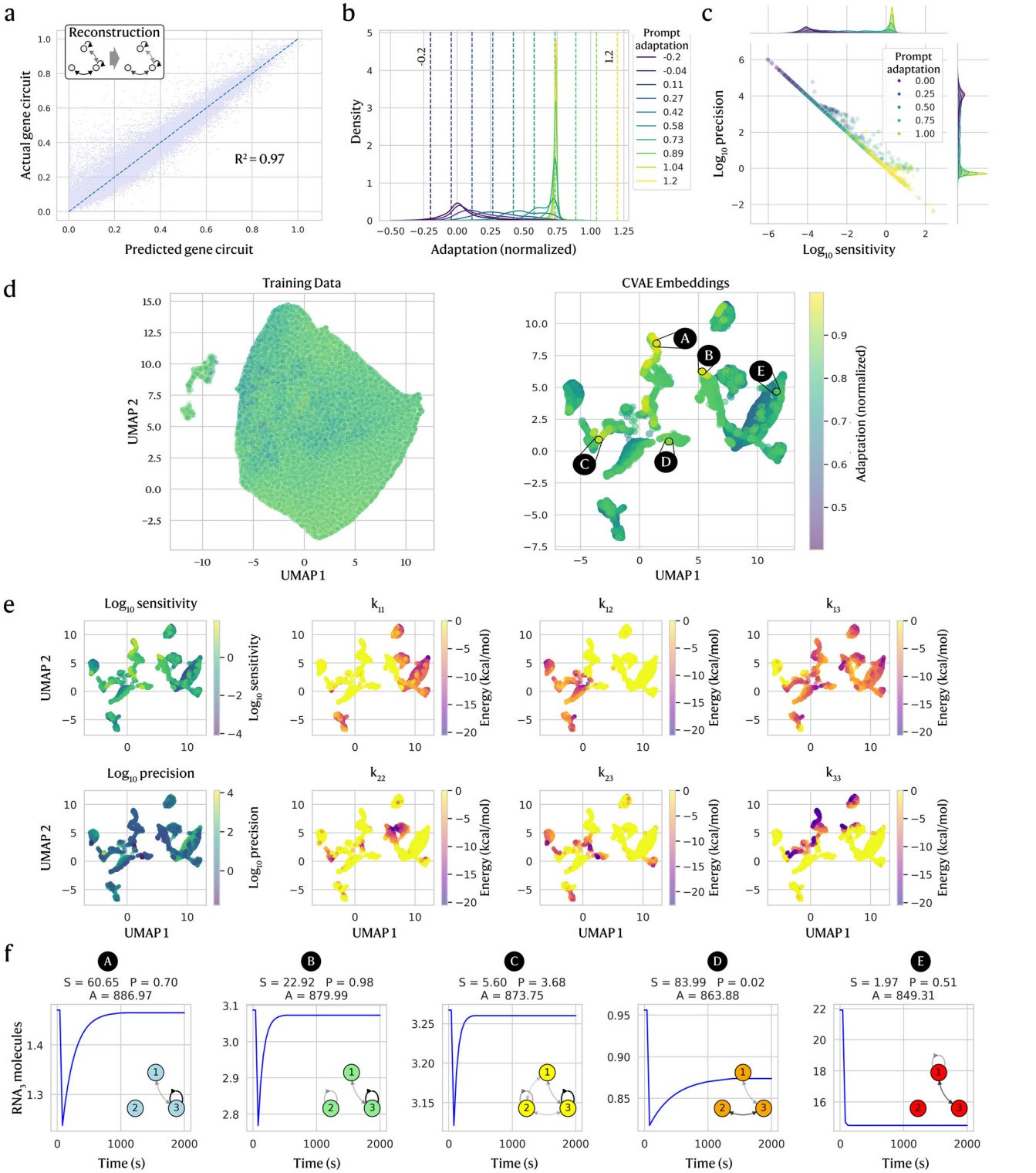

Fig. 2 | CVAE trained on adaptation generates novel circuits true to prompts. **a** Predicted vs. true circuits (interactions in terms of minimum free binding energy) show high accuracy for both training and test data, with an R^2 of 0.97 for test data. **b** Kernel density estimation (KDE) distributions of the adaptation of generated circuits (n = 10,000) (normalised to training data range [0, 1]) largely adhere to the intended adaptation prompt (shown as coloured dashed vertical lines). **c** Generated circuits adhere to prompts in both sensitivity and precision. **d** [Left] UMAP of training data (genetic circuit parameters x) compared to [right] UMAP of the embeddings of CVAE-generated genetic circuits, which reveals a latent space structured by adaptation. **e** CVAE UMAP coloured by other features shows how the latent space is structured by sensitivity and precision, as well as how each unique circuit interaction [k_{11}, k_{12}, k_{13}, k_{22}, k_{23}, k_{33}] varies between clusters. **f** To illustrate actual dynamic behaviours, 5 circuits with the highest adaptation in each of their clusters (A-E) are picked from the CVAE UMAP, with different motifs represented.

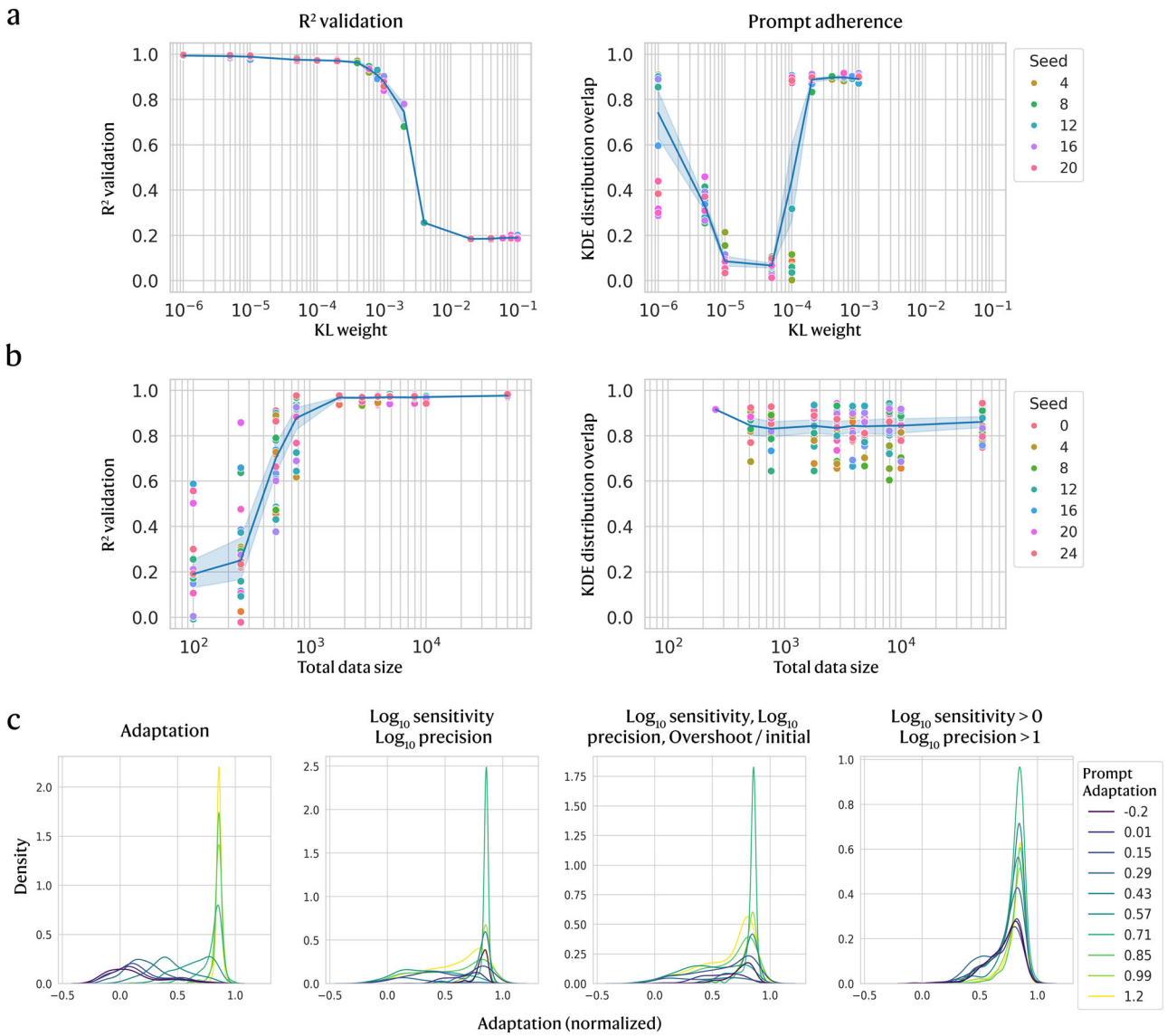

Fig. 3 | Generative model trained with varying KL weight, dataset size, and objective function. a The weight controlling KL divergence loss is varied and compared by model accuracy (R^2) and prompt adherence, which is assessed for the simulated dynamic function adaptation by measuring the average overlap between the KDE distribution of a prompt and all other prompt distributions. For each value, 20 different seeds are used for model initialisation. **b** Similarly to (**a**), the accuracy and prompt adherence are assessed across training dataset sizes, with 24 seeds used for model initialisation. **c** To examine how using alternative representations of adaptation affects model training, we compare the KDE prompt distributions for the best-performing model (out of 60 initialised with different seeds) for each of the 3 new objective functions, $[\log_{10}S, \log_{10}P]$, $[\log_{10}S > 0, \log_{10}P > 1]$, and $[\log_{10}S, \log_{10}P,$ overshoot/initial].

the prompt shows marginally better alignment to actual sensitivity (Supplementary Fig. 2). For the binary objective $[\log_{10}S > 0,\ \log_{10}P > 1]$, the gradation of the prompt-wise distributions indicates that the generative model learned to interpolate beyond just categorical values. Although distributions peak further from targets and the overall adaptation range is narrower than the continuous objectives, prompt-wise distributions are ordered and distinct, showing that even a simple binary objective can outperform more complex ones.

Compared with a single adaptation objective, multi-objective training struggled to learn multiple high-dimensional conditional distributions and to follow prompts. Redefining the objective function with a custom scalar function (adaptation) or simplified formulation (e.g. binary) often produced better generated outputs than prompting with raw metrics. Therefore, simple objectives are a good starting point before introducing more complex custom functions. With balanced data, even small training datasets can yield strong models, which is feasible for circuits with few interacting components, as investigated here. Next, we examine a different case of multi-objective training with wholly distinct objective functions of adaptation and evolutionary ruggedness.

Having shown that the CVAE can generate adaptation-targeted circuits even with limited data or binary labels, we next incorporate evolutionary stability into design. Evolutionary stability is determined by the ruggedness of the evolutionary landscape that a circuit sits in, where higher ruggedness corresponds to a higher sensitivity of the circuit function to small changes in the interaction strengths. We estimate ruggedness by perturbing a circuit's interactions (i.e. changing k's) and measuring the resulting change in function (i.e. the adaptation value it achieves). This can be done by (1) mutating RNA sequences and recomputing binding energies or (2) directly perturbing the interaction energies between RNAs. We opt for (2), since sequence-level details are not provided to the CVAE. Our ruggedness metric slightly perturbs each unique interaction, re-simulates the dynamics, and aggregates the resulting changes in adaptation across all perturbations.

Since multiple genetic circuit topologies can yield the same adaptation, a fixed adaptation level can lie on rugged or smooth regions of the

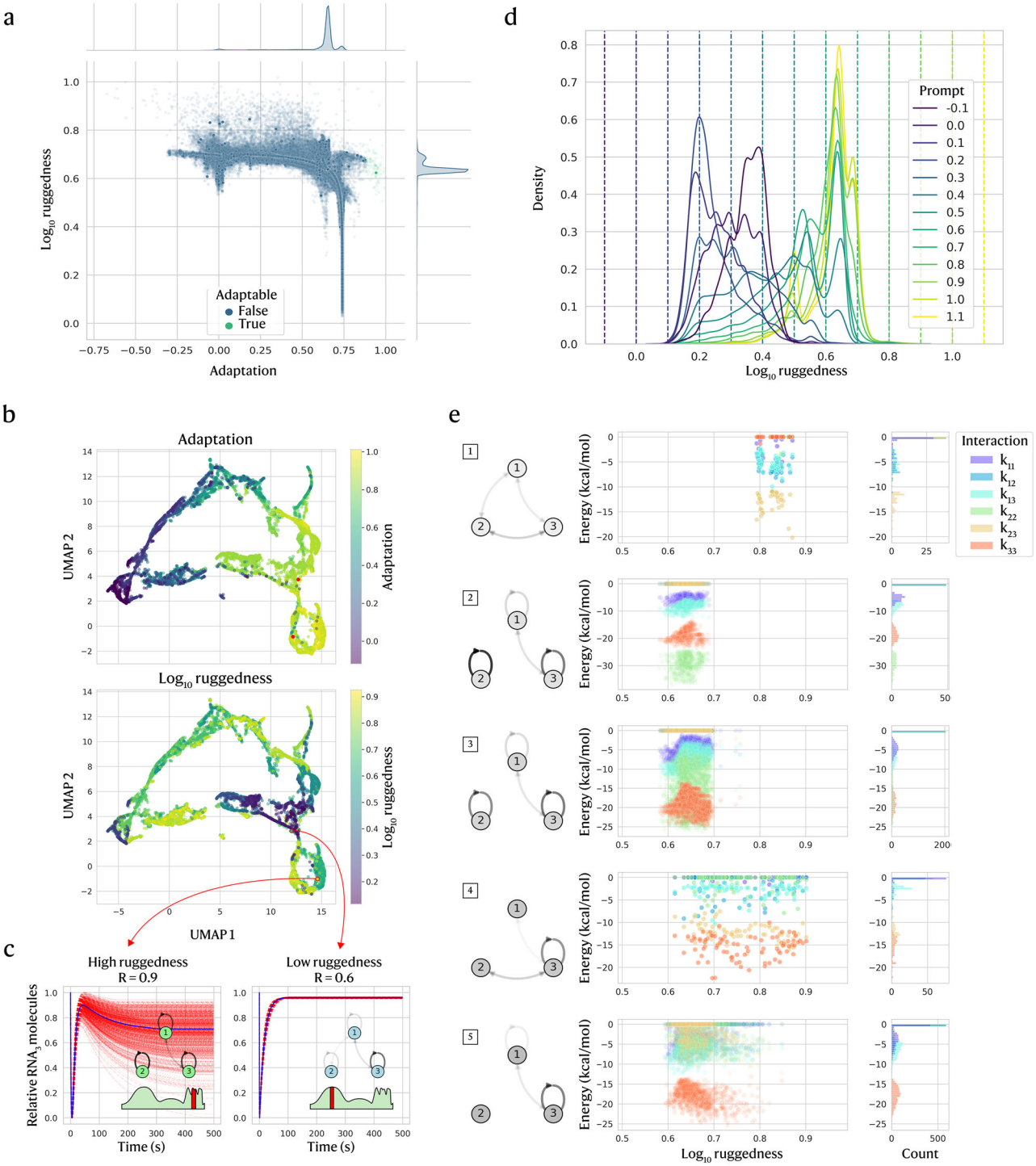

Fig. 4 | Adaptation and evolutionary ruggedness are prompted simultaneously to generate circuits effectively. a Adaptation correlates slightly with $\log_{10}$ ruggedness, shown for training data before pre-processing but normalised to [0, 1] (samples outside of this range are later excluded). **b** UMAP of generated circuits demonstrates that the latent space is structured by both normalised adaptation (top) and ruggedness (bottom). **c** Two circuits chosen from a high-adaptation region with high (left) and low (right) ruggedness are plotted by output RNA (blue) against 1000 mutated versions (dashed red), normalised to a [0, 1] range to show relative changes. Network diagrams show the original circuit topology and an illustration of the evolutionary landscape. **d** KDE prompt-wise distributions of ruggedness prompts (prompted adaptation = 1) are distinct and adhere well to most ruggedness prompts. **e** Hierarchical clustering of interactions in adaptable circuits ($S > 1, P > 10$) identifies 5 motifs that also differ in ruggedness, with network diagrams (left) representing median circuit interactions.

evolutionary landscape. Nevertheless, some correlations arise from the definition of ruggedness, which perturbs each unique interaction and combines the change in adaptation between original and perturbed circuits (Eqs (6) and (7)). Because adaptation weights sensitivity slightly more than precision, ruggedness is biased with perturbations that mainly affect sensitivity. Consequently, ruggedness decreases mildly as adaptation increases (Fig. 4a), while sensitivity and precision individually are largely flat versus ruggedness (Supplementary Fig. 3). A further bias comes from the logarithmic transform used to temper very high adaptation values, which also expands the low-ruggedness region so that roughly half of the [0, 1]

normalised range maps to negligible true ruggedness <0.01. As a result, most circuits are compressed into a narrower effective range starting at a normalised ruggedness of 0.5, especially affecting adaptable circuits that only make up a small proportion of the training data (Fig. 4a).

Beyond simulating the ruggedness metric, inferring stability from topology alone is difficult. Nevertheless, the CVAE's latent space (Fig. 4b) clusters by both adaptation and evolutionary ruggedness, with relatively high-adaptation regions splitting into high- vs. low/medium-ruggedness subregions. We illustrate this with two CVAE-generated circuits that are both largely adaptive but differ in ruggedness (Fig. 4c). We plot the output RNA (blue) alongside perturbed variants (dashed red), normalised to [0, 1] to emphasise relative changes. The high-ruggedness circuit (left) shows large trajectory shifts after perturbation, whereas variants of the low-ruggedness circuit (right) closely track the original, preserving sensitivity and precision (see Supplementary Note 2 for a mechanistic account).

The structured latent space suggests that joint prompting for adaptation and ruggedness is coherent, so we test adherence to combined prompts. Across many pairs of prompts, generated circuits concentrate near their targets. At adaptation prompt = 1, ruggedness distributions are distinct for ruggedness prompts (Fig. 4d), although extremes prove more challenging—very low ruggedness prompts (<0.2) rebound to higher true ruggedness, while low-medium ruggedness prompts peak near the intended level. High prompts (>0.7) yield narrow distributions that saturate around ~0.7, with a small tail to ~0.9. Because ruggedness combines adaptation changes over all perturbed interactions, high ruggedness arises more readily when many topological links are vulnerable. These effects are less pronounced at other adaptation prompts with more training data (Supplementary Fig. 4). Despite data limits, the model still produces low-ruggedness circuits at high adaptation, indicating it learns useful combinations of objectives from the data.

As exemplified by the adaptable pair in Fig. 4c, adaptable motifs can be either mutation-prone or mutation-resistant. The CVAE's latent space organises circuits by ruggedness, suggesting it captures topology links to evolutionary properties. Using hierarchical clustering, we extracted motifs from CVAE-generated adaptive circuits and related them to ruggedness (Fig. 4e and Supplementary Fig. 5). Cluster 1 is high-ruggedness (evolvable), clusters 2–3 are low-ruggedness (stable), and clusters 4–5 span intermediate levels (flexible). As stability increases, allowable interaction-energy ranges narrow (e.g. motifs 2, 3 and 5), consistent with core motif 'constellations' that become more evolvable when pushed toward their boundaries in high-dimensional design space (see Supplementary Note 3). The CVAE reflects this, as ruggedness prompts are higher in genetic circuits that deviate from their most stable settings (radar charts in Supplementary Fig. 6). Thus, the CVAE can generate both evolvable and stable circuits while maintaining adaptation.

Discussion

The challenges facing engineering biology are complex, multi-faceted and well-suited for machine learning approaches. Genotype-phenotype datasets that combine underlying sequences with behavioural features such as dynamic functions or metabolic properties are increasingly being pursued experimentally to fill the data gap for training high-quality models[65–71]. However, more data is not always the answer—understanding the limitations of ML models, especially generative approaches, is an important step in order to make them trustworthy, deployable, and competitive with other discovery methods[60,72,73].

In this paper, we present a novel generative approach for realising genetic circuits that optimise both functional and evolutionary properties, and show that even simple generative models can identify key features that recapitulate known adaptation motifs (as identified in refs. 5,12,13). Even without providing full time-series data (as in ref. 54), summary metrics like sensitivity and precision (even as binary labels) are enough to train our model and output diverse designs for a desired function. Training on other kinds of functions, such as evolutionary features or even response time (see Supplementary Fig. 9), proved feasible too without re-optimising hyperparameters, highlighting the model's versatility. Many generative approaches designed sequences of individual biological parts[48,50,74–79] while other ML approaches predicted optimal circuit composition and culture conditions[40,68,80,81], so it is plausible that multiple sequences could be generated for desired dynamic behaviour. Although our simulated RNA circuits dataset makes many assumptions and approximations, this work serves as an important proof of concept for utilising generative models in genetic circuit design.

Our work also highlights many of the challenges and trade-offs faced when quantitatively defining an objective for circuit behaviour. Our custom scalar for adaptation improved accuracy and prompt adherence, but obscures information about mid-range combinations of sensitivity and precision. However, treating these components as separate targets reduced prompt adherence; adding more objectives (e.g. overshoot) skewed the training distribution and hindered conditional learning. One possible improvement could be encoding raw metrics in alternative ways (e.g. one-hot). Another approach could be training the model on each objective sequentially, though this would result in a model that is limited to one prompt and likely only able to reliably generate circuits for which all objectives align in label value. The joint objective (adaptation and ruggedness) likewise suffered from poor training representation of some combinations, despite our large in silico dataset. While specificity was reduced at extreme prompts, the model performed well even for some of these data-poor combinations, suggesting that latents learned in one range can generalise more broadly.

We attempted to improve embedding quality and sought a latent space structured even more by circuit function using several strategies. After tuning the KL weight, architectural tweaks (e.g. embedding size) yielded little improvement, so we added a contrastive loss to pull similar samples together and push dissimilar ones apart[82,83]. This neither improved prompt adherence nor showed better organisation of the latent space (Supplementary Fig. 7). Other contrastive implementations[84–86] or alternative latent encodings[87] may help, as well as VAE variants aimed at disentanglement[88–90]. Some issues are intrinsic to VAEs, including poor approximations of heavy-tailed distributions[91] and posterior collapse[92], while others are broader ML research challenges, such as sensitivity to weight initialisation seeds[93–97] or variations introduced through noise in hardware[98]. Altogether, these issues can determine whether a model responds to prompts at all. We found no reliable heuristic to predict prompt adherence without simulating generated circuits, which is analogous to costly lab tests and illustrates practical challenges in deploying VAE models. Alternative models, such as alternative VAE architectures[92,94] or hybrid graph networks that generalise well and can handle network inputs, may address these limitations[99].

Nevertheless, the CVAE generally performed well, especially considering the sparsity of adaptable circuits and retention of accuracy training on just over 1000 samples, which could be achievable in the lab. For practical applications with expensive data collection and optimisation processes often relying on scientific intuition, flexible ML models can be invaluable in identifying good initial designs for testing, and support existing model-driven algorithms like Bayesian optimisation (BO) and other iterative/active learning approaches that iterate on these designs. These are already heavily used in industry[100–103] and show promise for synthetic biology[25,77,104,105]. Furthermore, many ML models can estimate their own uncertainty about a prediction and utilise diverse inputs such as sequences or images, presenting a compelling upgrade to existing methods that are limited to categorical or continuous inputs. While we could train CVAEs to use sequences as inputs, circuit topologies enable the addition of more flexible constraints and for multiple different sequences to implement the same circuit. There are thus arguments for leveraging separate models for different levels of abstraction within synthetic biology; however, a multi-modal model approach could also see compounding benefits arise from combining data types. Advanced models such as transformers pose an especially suitable option for integrating sequence-level information (such as mutation type and location) with more abstract circuit features (such as topology, switches or logic gates)[106] and predicting evolutionary properties from sequences directly[49,107].

Beyond existing models, we argue that there is an opportunity to further develop machine learning tools and architectures such as hypergraph models, biologically inspired neural networks, self-supervised learning, or hybrids for synthetic biological tasks[99,108–112]. Given the difficulty of predicting binding between biological molecules, using separate models for the tasks of designing circuit topology vs. sequence is perhaps a more feasible option. Novel architectures better geared to operate at multiple levels of abstraction than VAEs could reduce more complex biological networks, such as large adaptable biochemical pathways[13] or >3-node genetic circuits[106], into overarching symbolic motifs. Crucially, models that can predict biological interactions beyond RNA–RNA interactions, like protein-protein or protein-DNA, have transformative potential for synthetic biology by integrating sequence-level approaches with functional[79,113–117] and evolution-aware[107] design. This could broaden our approach from simulated RNA circuits to interacting genes and proteins and allow exploration of new dynamics-predicting models. Ultimately, many exciting innovations on the horizon promise to further tackle the generative design challenge and enable evolution to become a more predictable and engineerable feature of engineering biology.

Methods

For the sequence-sampled dataset, we first generate a set of $n = 3$ RNA sequences of length $m = 20$ by sampling from a distribution of nucleotides. Their probabilities are equivalent to the occurrence of each nucleotide in *Escherichia coli* cells, with the following proportions: [A: 0.2451, C: 0.2458, G: 0.2622, U: 0.2469][118]. We then take this collection of RNAs and simulate the binding energies for all paired combinations of sequences as query and target with the IntaRNA interaction simulator using default settings[27]. For each RNA pair, including self-interactions, we use the most probable predicted minimum free energy (*kcal/mol*) and convert this to a rate of binding. We use the Gibbs free energy equation to convert from binding energy to the ratio of association and dissociation rates:

$$\Delta G^0 = -RTln(K) \tag{1}$$

where ΔG^0 is the minimum free binding energy at equilibrium, R is the gas constant, T is the temperature in Kelvin, and K is the equilibrium constant. This equilibrium constant is a ratio of the forward and reverse binding rates k_f and k_r, respectively (also known as the association rate k_a and dissociation rate k_d). Assuming that the forward rate for RNA binding is very fast[119–121], we fix $k_f = 1 \times 10^6$ and find the reverse rate k_r from the simulated binding energy and the equilibrium constant K. In practice, the K calculated directly from the Gibbs free energy equation (1) does not align well with observed RNA binding, for example, between small RNA (sRNA) and messenger RNA (mRNA) in RNA circuits controlling green fluorescent protein (GFP) fluorescence[31]. We therefore redefine the conversion from binding free energy to the equilibrium constant using an approximate parametrisation of the observations from Na et al.[31] to get the following expression:

$$K \approx \frac{e^{-0.8(\Delta G^0+10)}}{x_0} \tag{2}$$

where x_0 is the initial concentration of RNA reactants. The expression stems from the parametrisation of the relative GFP fluorescence measured by Na et al. across different sRNAs of varying binding energy strength and indicates the amount of binding between the GFP mRNA and an sRNA library acting as inhibitors of mRNA transcription. From this reparametrisation of K and with the forward rate constant k_f, we can approximate the rates of reaction for simulated genetic RNA circuits. From these rates, we construct a system of ordinary differential equations (ODEs) and simulate the dynamics with the solver library diffrax[122] using the numerical solvers Tsit5 (Tsitouras' 5/4 method, 5th order explicit Runge-Kutta) and Dopri5 (Dormand-Prince's 5/4 method) with adaptive step sizing. The simulation is implemented through the `bioreactions` package[123].

Table 1 | Molecular parameters for dynamic simulation

Parameter	Value
Average mRNA per cell (*mRNA*)	100
Cell doubling time (*s*)	1200
Starting copy numbers (*mRNA*)	100
Number of mRNA per circuit	3
Sequence length per mRNA	20
Signal (input) mRNA	RNA_1
Output mRNA	RNA_3
Amount of impulse signal added to steady state	2*x* steady state RNA_1
Degradation rate ($mRNAs^{-1}$)	0.01175
Association binding rate ($mol^{-1}s^{-1}$)	1×10^6
Association binding rate ($mRNA^{-1}s^{-1}$)	$1.50958097 \times 10^{-3}$

This way, the parameter-sampled dataset is constructed by sampling the minimum free binding energy ΔG directly. For each unique interaction, we sample a value from a uniform distribution with the range [0, 1] and scale the interactions to the range [−30, 0] to represent a realistic range of *kcal/mol* that an RNA sequence with 20 base pairs could take. These energies are then transformed into equilibrium constants and reaction rates in the same way as for the sequence-sampled dataset. Because of this fast energy construction step, the computationally intensive RNA sequence interaction prediction simulation can be avoided, and a greater diversity and volume of circuits can be sampled.

We simulated the signal response dynamics in two phases. First, we simulated the steady states of all free and bound RNAs for all circuits and used these as the initial state for calculating sensitivity and precision. We then perturbed them with a spike (step input) in the input RNA (node 1 RNA_1). We simulated the dynamics following the settling of this spike into new steady states and used these as the final states for adaptation calculations. The default parameters (Table 1) relating to the signal response were kept fixed across all simulations.:

The code used to generate the dataset can be found in the GitHub repository https://github.com/olive004/genetic_circuit_generator.

More formally, sensitivity measures how responsive the output is to the input signal, defined as the maximum change in the output relative to the change in the input (Eq (3)). Conversely, precision measures how closely the output returns to its initial value before the input signal was added, defined as the inverse of the relative change in the output compared to the change in the input (Eq (4)):

$$S = \left| \frac{(O_{peak} - O_1)/O_1}{(I_{peak} - I_1)/I_1} \right| \tag{3}$$

$$P = \left| \frac{(O_2 - O_1)/O_1}{(I_2 - I_1)/I_1} \right|^{-1} \tag{4}$$

where S is sensitivity, P is precision, O and I are the output and input species concentrations (here in molecules per cell), the subscript 1 denotes the initial concentration, the subscript 2 denotes the final concentration, and the subscript *peak* denotes the peak concentration (e.g. the maximum or minimum achieved during a simulation period, depending on the direction of change), see Fig. 1a (right) for an illustration of these features.

Sensitivity and precision are metrics used to assess whether a system is able to adapt to a signal[5] and have been defined in Eqs (3) and (4). However, there is no continuous metric for adaptation for biological networks beyond the binary definition that circuits with a sensitivity >1 and a precision >10. For training a model and optimising circuits towards a single objective, we constructed a metric for adaptation that is positive within the region where circuits function, encompassing the ranges $[10^{-7}, 10^2]$ for sensitivity and

$[10^{-2}, 10^{7}]$ for precision. The adaptation metric is defined as a concentric gradient increasing towards the binary threshold as the following:

$$a = -\left(\alpha(\log_{10}(s) - s_{centre})^2 + (\log_{10}(p) - p_{centre})^2\right) + C \quad (5)$$

where s and p are sensitivity and precision, $s_{centre} = 7$ and $p_{centre} = 7.5$ are their respective concentric centres, $\alpha = 3$ is the adjustable extra weight given to sensitivity, and $C = 1000$ is a constant to ensure the metric is positive in the operational region. Making the metric positive facilitates calculating the loss function across all sensitivity and precision values so that the loss always has the same sign. The circuits also always move in the correct direction across both dimensions, thanks to the centres of sensitivity and precision, which allow adaptation to peak at sensitivity = 10^7 and precision = $10^{7.5}$ well within the adaptation threshold region. The precision peak is slightly higher to reflect that the binary boundary also has a higher precision. This alone was not enough to bias optimisation towards more sensitive circuits and high precision values easily dominated, so we added the adjustable weight α to boost sensitivity slightly more. Values that go to infinity due to sensitivity or precision having divisions by zero are set to the Python JAX *NaN* value.

To compare circuits in terms of their evolutionary properties, we perturb their parameters by a small fixed value and resimulate the dynamics of the mutated circuit. Circuits can be defined by the number of unique interactions between nodes, which for a 3-node circuit results in 6 unique interactions $[k_{11}, k_{12}, k_{13}, k_{22}, k_{23}, k_{33}]$. The ruggedness of the evolutionary landscape on which a circuit sits can be seen as the magnitude of its gradient with respect to each interaction. We therefore calculate ruggedness by perturbing each interaction individually and then summing up the squared changes in adaptation relative to the perturbation applied across all mutated versions of a circuit. This can look like the following equation for each circuit:

$$dk = \frac{a_\epsilon - a_0}{\epsilon} \quad (6)$$

$$r = \sum_i^m dk_i^2 \quad (7)$$

where dk is the change in adaptation per perturbation ϵ, which is set to $\approx 1 kcal/mol$ binding energy, a_ϵ and a_0 are the perturbed and initial adaptation values, respectively, and the ruggedness r is a sum of squares across all $m = 6$ unique interactions dk_i. For the perturbed circuits in Fig. 4c, the red traces are mutated versions of the original that have had all of their interactions perturbed by a small, random amount (normally distributed around 0, with a standard deviation of 10% of the strongest binding energy in the training data).

We implemented the conditional VAE model with the Python packages JAX[124] and Haiku[125]. The general VAE architecture was kept the same as the default model[64,125] composed of an encoder and a decoder, with the main changes occurring in the number and size of layers. Both the encoder and decoder are a stack of 3 linear layers of size 32 with leaky ReLU (rectified linear unit) activations and He normal (Kaiming normal) weight initialisation. While encoders and decoders typically do not have layers of the same size, an initial hyperparameter sweep changing the layer sizes did not yield significantly better results, and the same-sized layers were sufficient for our purposes. We do acknowledge that this would be a source of improvement for optimising model architecture, especially for models deployed beyond simulated genetic circuits. The CVAE model additionally has two layers for transforming the encoder output h into the probabilistic parameters μ and *logvar*, which are combined to determine the embedding space in a reparametrisation step:

$$z = \mu + (e^{logvar/2})\epsilon \quad (8)$$

where z is the embedding, μ is the mean or centre of the distribution, *logvar* is the log variance of the distribution, and ϵ is random, normally-distributed noise that can be added to provide sample diversity or stochasticity during training for better model robustness.

The model was trained with the circuits and their dynamic function of adaptation as input. The circuit input x has dimensions $[n, m]$ and the input conditional variable c has dimensions $[n, c_n]$, where n is the total number of training samples, e.g. $\approx 1 \times 10^6$, m is the number of unique interactions per circuit (6), and c_n is the number of objectives part of the conditional variable. For adaptation, this would be 1, while using both sensitivity and precision as the conditional variables is also possible and makes $c_n = 2$. Within the CVAE, the latent variable z representing the embedding space has dimensions $[n, H + c_n]$, with H representing the size of the hidden variables h and z. The conditional variable c is concatenated with z before being passed to the decoder, hence the dimensions of z also depend on c_n. The output y is just a reconstruction of the circuits, so it has the same dimensions as x. At the model inference stage, where circuits are generated, a noise vector of dimensions $[n, H]$ is concatenated as a stand-in for z with the prompt c.

The inputs x and c are prepared by filtering and normalising the training data. Circuits with sensitivity and precision with NaN values are removed, along with duplicate circuits and those with exceedingly slow response times that make calculating adaptation difficult. We further normalise circuits with a robust scaling transformation and with a min-max scale to make sure they are in a [0, 1] range. The robust scaling transforms features by subtracting the median from the training data and then dividing by the interquartile range, e.g. the range between the 25th and 75th percentile in the data. For the binding energies represented by x, we add a negative multiplier to align high model outputs with strong binding interactions. These filtering and normalisation steps stabilise the training process.

We train the model by optimising its weights with gradient descent over a set number of epochs. At the beginning of every epoch, we shuffle the order of the training data and split it into batches. In each training step, the batches are run through the model iteratively to get a predicted output that is used to calculate the loss and gradients. Since the model outputs reconstructions of the input circuits, a mean square error (MSE) loss is applied initially to penalise differences between the input x and the predicted output y. For validation accuracy, we calculate how many samples the model reconstructed are within an error bound of 0.1 (out of the [0, 1] range). While we experimented with an L2 loss to ensure that weights remained within a reasonable region, this did not have a significant effect on model accuracy or training stability, so we stopped using it. We next apply a Kullback-Leibler (KL) divergence loss, calculated as the mean of the Gaussian KL divergence loss of the following form[63]:

$$L_{KL} = \frac{-logvar - 1 + e^{logvar} + \mu^2}{2} \quad (9)$$

where the μ and *logvar* are output by the VAE as the parametrisation of its embeddings. This KL divergence loss is averaged together across all samples and multiplied by a weighting factor, which we explore in Fig. 3.

Following this, we also tried applying a contrastive loss to improve the quality of latent space clustering, but this did not significantly improve outcomes, as discussed previously (Supplementary Fig. 7). We nevertheless report our methods for applying contrastive loss in this context, as there is still reason to believe this could be useful in the future. Contrastive loss typically punishes embeddings that are similar but for which the labels are actually different. Normally, a pair of 'same' and 'different' samples is compared to an anchor sample based on their membership of a particular category[82,83]. A batch of similar and dissimilar samples may even be supplied. However, since in our data we are mostly dealing with continuous labels in the form of the adaptation metric, we calculate the distance between all pairs of normalised conditional labels in a batch using a dot product. We then subtract a similarity threshold value to recenter the distance as either positive (similar) or negative (dissimilar). This is then multiplied by the negative of the similarity of the encoder embeddings h, which is also calculated with a dot product and scaled by the temperature factor. For a

threshold of 0.9, a normalised distance of 0.95 would decrease the loss, while a distance of 0.85 would contribute positively to the loss, depending on the similarity between the embeddings. This can be summarised as the following:

$$L_{CL} = -\frac{h \cdot h^T}{T}((c \cdot c^T) - d) \tag{10}$$

where T is the temperature and d is the threshold similarity distance. The diagonal of this is also subtracted to eliminate counting self-similarities. Improvements to this approach could include simplifying the comparison between samples, for example, by comparing pairs of samples to an anchor instead of all samples in the batch, or adjusting the threshold adaptation value at which circuits are considered similar.

After summing together all relevant losses, we use an Adam optimiser to calculate the weight updates from the gradients[126]. After 20 warm-up epochs in which the optimiser slowly ramps up the learning rate from 0 to 1×10^{-3} and then sets out on a cosine decay schedule until 2000 epochs are reached. The training loop can stop early if a validation accuracy of over 0.98 is reached or if there is no improvement in the model outputs for 500 epochs. This helps to prevent a model from overfitting on the dataset.

The training process is governed by many hyperparameters, which we determined through automated hyperparameter tuning followed by sweeps of specific parameters. We used the neural network intelligence (NNI) package[127] to find a good starting point and ensure that good parameter combinations are not being missed out on. We quickly found working values for specific variables that had a more noticeable initial effect on model training, such as the learning rate, the validation accuracy threshold, and whether to bin the conditional variable into categories. While categorical labels performed similarly well in terms of accuracy and prompt adherence as continuous adaptation labels, we saw no reason not to favour the continuous labels. More experimentation in this area may be warranted depending on the particular target circuit function.

While a CVAE can attain high accuracy in reconstructing inputs, this does not automatically mean adherence to prompts in the inference phase. Initially, we used the set of metrics commonly employed in generative AI literature, namely precision, recall, and the combined F1 score[128]. Generally, precision measures the accuracy of positive predictions, while recall measures a model's ability to find all actual positive instances. In this context, precision is the proportion of circuits generated for a particular prompt that actually fulfil that prompt (within a threshold bound). Recall measures the proportion of circuits that match one of the prompts that were actually generated according to that prompt. For example, high adaptation circuits that were generated with a different prompt would be considered missed by the model. The F1 score combines these to highlight the trade-off between them in the following way:

$$F1 - score = \frac{2(precision)(recall)}{precision + recall} \tag{11}$$

The main reason using these metrics for prompt adherence was difficult was due to the continuous nature of our prompts, meaning that the threshold bounds within which a circuit is considered to adhere to its prompt played a significant role in assessing precision, recall and F1 score.

While these metrics provided some information about prompt adherence, we constructed a simpler, more heuristic area-under-the-curve metric based on the overlap between distributions of circuits generated with different prompts. We fitted a Gaussian kernel density estimate (KDE) to each prompted collection of generated circuits to create a smooth distribution of their functional properties (e.g. adaptation). We then compare prompt distributions by evaluating the KDEs at 1000 points and summing the number of overlapping points. Overlaps from KDEs are on a [0, 1] scale, with 0 being no overlap and 1 being complete overlap. For each prompt distribution, we use the average of its overlap with all other prompt distributions and finally take the average of these averages. This metric is mostly a heuristic, since we do not account for the number of prompt distributions used to calculate the average area under the curve.

A typical training run with the CVAE on the full dataset (pre-processed to ensure balance and validity of the simulated dynamics) takes approximately 5 min with default training parameters in Supplementary Tables 1–4. Generating the dataset is more computationally intensive; sampling 1,000,000 randomly-generated sequences and then simulating them with IntaRNA (as 3-node RNA circuits) took 1 day, 2 h and 31 min on CPU (single-threading) to determine circuit parameters. Simulating the dynamics based on the interaction parameters of these RNA circuits took around 3 days, with around 1 h for initial steady states (after all RNAs are initialised at $n = 100$ each), 45 min for post-signal steady states, and a few minutes for gathering analytics (adaptation, response times, peak amounts) for a batch size of around 20000.

A single NVIDIA GeForce RTX 4090 GPU (24GB) and a CPU with 24 cores and 128GB of RAM were used for simulations. JAX[124] was used to facilitate batching and scaling, speed up compilation in Python, and easily run on GPUs. The `diffrax`[122] library was used for solving dynamics. A Docker container was set up for running all simulations, with the base images including IntaRNA's image 3.4.1 from Docker biocontainers hosted on Red Hat's quay.io (URL to container: https://quay.io/repository/biocontainers/intarna?tab=tags&tag=3.4.1--pl5321hdcf5f25_0) and the NVIDIA Ubuntu 22.04 image with CUDA 12.6.0 (instructions for environment setup are all within the code repository in a Dockerfile and requirements files).

Data availability

Data is available on Zenodo at the https://doi.org/10.5281/zenodo.17194085 (https://doi.org/10.5281/zenodo.17194084).

Code availability

The underlying code is available in the GitHub repositories https://github.com/olive004/EvoScaper and https://github.com/olive004/genetic_circuit_generator.

Received: 14 October 2025; Accepted: 1 March 2026;
Published online: 16 March 2026

References

1. Barkai, N. & Leibler, S. Robustness in simple biochemical networks. *Nature* **387**, 913–917 (1997).
2. Repoila, F., Majdalani, N. & Gottesman, S. Small non-coding RNAs, co-ordinators of adaptation processes in *Escherichia coli*: the RpoS paradigm. *Mol. Microbiol.* **48**, 855–861 (2003).
3. Kollmann, M., Løvdok, L., Bartholomé, K., Timmer, J. & Sourjik, V. Design principles of a bacterial signalling network. *Nature* **438**, 504–507 (2005).
4. Jørgensen, M. G., Pettersen, J. S. & Kallipolitis, B. H. sRNA-mediated control in bacteria: an increasing diversity of regulatory mechanisms. *Biochim. Biophys. Acta* **1863**, 194504 (2020).
5. Ma, W., Trusina, A., El-Samad, H., Lim, W. A. & Tang, C. Defining network topologies that can achieve biochemical adaptation. *Cell* **138**, 760–773 (2009).
6. Cardelli, L. Morphisms of reaction networks that couple structure to function. *BMC Syst. Biol.* **8**, 84 (2014).
7. Sequeiros, C., Vázquez, C., Banga, J. R. & Otero-Muras, I. Automated design of synthetic gene circuits in the presence of molecular noise. *ACS Synth. Biol.* https://doi.org/10.1021/acssynbio.3c00033 (2023).
8. Aoki, S. K. et al. A universal biomolecular integral feedback controller for robust perfect adaptation. *Nature* **570**, 533–537 (2019).
9. Gyorgy, A., Menezes, A. & Arcak, M. A blueprint for a synthetic genetic feedback optimizer. *Nat. Commun.* **14**, 2554 (2023).
10. Buecherl, L. et al. Stochastic hazard analysis of genetic circuits in iBioSim and STAMINA. *ACS Synth. Biol.* **10**, 2532–2540 (2021).

11. Araujo, R. P. & Liotta, L. A. The topological requirements for robust perfect adaptation in networks of any size. *Nat. Commun.* **9**, 1757 (2018).
12. Bhattacharya, P., Raman, K. & Tangirala, A. K. Discovering adaptation-capable biological network structures using control-theoretic approaches. *PLoS Comput. Biol.* **18**, e1009769 (2022).
13. Araujo, R. P. & Liotta, L. A. Design principles underlying robust adaptation of complex biochemical networks. in *Computational Modeling of Signaling Networks* 3–32. https://doi.org/10.1007/978-1-0716-3008-2_1 (Humana, 2023).
14. Proverbio, D., Katz, R. & Giordano, G. Bridging robustness and resilience for dynamical systems in nature. *IFAC Pap.* **58**, 43–48 (2024).
15. Brophy, J. A. N. & Voigt, C. A. Principles of genetic circuit design. *Nat. Methods* **11**, 508–520 (2014).
16. Hughes, R. A. & Ellington, A. D. Synthetic DNA synthesis and assembly: putting the synthetic in synthetic biology. *Cold Spring Harb. Perspect. Biol.* **9**, a023812 (2017).
17. Shaytan, A. K., Novikov, R. V., Vinnikov, R. S., Gribkova, A. K. & Glukhov, G. S. From DNA-protein interactions to the genetic circuit design using CRISPR-dCas systems. *Front. Mol. Biosci.* **9**, 1070526 (2022).
18. Chlebek, J. L. et al. Prolonging genetic circuit stability through adaptive evolution of overlapping genes. *Nucleic Acids Res.* **51**, 7094–7108 (2023).
19. Sechkar, K. & Steel, H. Model-guided gene circuit design for engineering genetically stable cell populations in diverse applications. *J. R. Soc. Interface* **22**, 20240602 (2025).
20. Castle, S. D., Grierson, C. S. & Gorochowski, T. E. Towards an engineering theory of evolution. *Nat. Commun.* **12**, 3326 (2021).
21. Tack, D. S. et al. The genotype-phenotype landscape of an allosteric protein. *Mol. Syst. Biol.* **17**, e10179 (2021).
22. Boada, Y., Reynoso-Meza, G., Picó, J. & Vignoni, A. Multi-objective optimization framework to obtain model-based guidelines for tuning biological synthetic devices: an adaptive network case. *BMC Syst. Biol.* **10**, 27 (2016).
23. Etcheverry, M., Moulin-Frier, C., Oudeyer, P.-Y. & Levin, M. AI-driven automated discovery tools reveal diverse behavioral competencies of biological networks. *eLife* **13**, RP92683 (2024).
24. Frank, S. A. Optimization of transcription factor genetic circuits. *Biology* **11**, 1294 (2022).
25. Merzbacher, C., Mac Aodha, O. & Oyarzún, D. A. Bayesian optimization for design of multiscale biological circuits. *ACS Synth. Biol.* **12**, 2073–2082 (2023).
26. Rodrigo, G., Carrera, J. & Jaramillo, A. Genetdes: automatic design of transcriptional networks. *Bioinformatics* **23**, 1857–1858 (2007).
27. Mann, M., Wright, P. R. & Backofen, R. IntaRNA 2.0: enhanced and customizable prediction of RNA-RNA interactions. *Nucleic Acids Res.* **45**, W435–W439 (2017).
28. Zadeh, J. N. et al. NUPACK: analysis and design of nucleic acid systems. *J. Comput. Chem.* **32**, 170–173 (2011).
29. Salis, H. M. The ribosome binding site calculator. *Methods Enzymol.* **498**, 19–42 (2011).
30. Ye, J. et al. Primer-BLAST: a tool to design target-specific primers for polymerase chain reaction. *BMC Bioinform.* **13**, 134 (2012).
31. Na, D. et al. Metabolic engineering of *Escherichia coli* using synthetic small regulatory RNAs. *Nat. Biotechnol.* **31**, 170–174 (2013).
32. Matthies, M. C., Krueger, R., Torda, A. E. & Ward, M. Differentiable partition function calculation for RNA. *Nucleic Acids Res.* **52**, e14 (2024).
33. Singh, A. H. et al. An automated scientist to design and optimize microbial strains for the industrial production of small molecules. https://doi.org/10.1101/2023.01.03.521657v1 (2023).
34. Carbonell, P. et al. An automated design-build-test-learn pipeline for enhanced microbial production of fine chemicals. *Commun. Biol.* **1**, 1–10 (2018).
35. Riesselman, A. J., Ingraham, J. B. & Marks, D. S. Deep generative models of genetic variation capture the effects of mutations. *Nat. Methods* **15**, 816–822 (2018).
36. Zrimec, J. et al. Controlling gene expression with deep generative design of regulatory DNA. *Nat. Commun.* **13**, 5099 (2022).
37. Bogard, N., Linder, J., Rosenberg, A. B. & Seelig, G. A deep neural network for predicting and engineering alternative polyadenylation. *Cell* **178**, 91–106.e23 (2019).
38. Linder, J., Bogard, N., Rosenberg, A. B. & Seelig, G. A generative neural network for maximizing fitness and diversity of synthetic DNA and protein sequences. *Cell Syst.* **11**, 49–62.e16 (2020).
39. Riley, A. T., Robson, J. M. & Green, A. A. Generative and predictive neural networks for the design of functional RNA molecules. https://doi.org/10.1101/2023.07.14.549043v1 (2023).
40. Rai, K., Wang, Y., O'Connell, R. W., Patel, A. B. & Bashor, C. J. Using machine learning to enhance and accelerate synthetic biology. *Curr. Opin. Biomed. Eng.* **31**, 100553 (2024).
41. Avsec, ˇZ. et al. Effective gene expression prediction from sequence by integrating long-range interactions. *Nat. Methods* **18**, 1196–1203 (2021).
42. Nielsen, A. A. K. et al. Genetic circuit design automation. *Science* **352**, aac7341 (2016).
43. Funahashi, A. et al. CellDesigner 3.5: a versatile modeling tool for biochemical networks. *Proc. IEEE* **96**, 1254–1265 (2008).
44. Misirli, G. et al. A computational workflow for the automated generation of models of genetic designs. *ACS Synth. Biol.* **8**, 1548–1559 (2019).
45. Appleton, E., Madsen, C., Roehner, N. & Densmore, D. Design automation in synthetic biology. *Cold Spring Harb. Perspect. Biol.* **9**, a023978 (2017).
46. Roehner, N. et al. GOLDBAR: a framework for combinatorial biological design. *ACS Synth. Biol.* https://doi.org/10.1021/acssynbio.4c00296 (2024).
47. Chandran, D., Bergmann, F. T. & Sauro, H. M. TinkerCell: modular CAD tool for synthetic biology. *J. Biol. Eng.* **3**, 19 (2009).
48. Sarkar, A., Tang, Z., Zhao, C. & Koo, P. K. Designing DNA with tunable regulatory activity using discrete diffusion. https://doi.org/10.1101/2024.05.23.595630v1 (2024).
49. Shulgina, Y. et al. RNA language models predict mutations that improve RNA function. https://doi.org/10.1101/2024.04.05.588317v2 (2024).
50. Zhang, H. et al. Deep generative models generate mRNA sequences with enhanced translation capacity and stability. https://doi.org/10.1101/2024.06.20.599727v1 (2024).
51. Repecka, D. et al. Expanding functional protein sequence spaces using generative adversarial networks. *Nat. Mach. Intell.* **3**, 324–333 (2021).
52. Klarner, L., Rudner, T. G. J., Morris, G. M., Deane, C. M. & Teh, Y. W. Context-guided diffusion for out-of-distribution molecular and protein design. *Proceedings of Machine Learning Research* **235**, 24770–24807 (2024).
53. Ingraham, J. B. et al. Illuminating protein space with a programmable generative model. *Nature* **623**, 1070–1078 (2023).
54. Baig, Y., Ma, H. R., Xu, H. & You, L. Autoencoder neural networks enable low dimensional structure analyses of microbial growth dynamics. *Nat. Commun.* **14**, 7937 (2023).
55. Zhou, B. et al. Accurate and definite mutational effect prediction with lightweight equivariant graph neural networks. Preprint at https://doi.org/10.48550/arXiv.2304.08299 (2023).
56. Schneider, S., Lee, J. H. & Mathis, M. W. Learnable latent embeddings for joint behavioural and neural analysis. *Nature* **617**, 360–368 (2023).
57. Sumi, S., Hamada, M. & Saito, H. Deep generative design of RNA family sequences. *Nat. Methods* **21**, 435–443 (2024).
58. Blazejewski, T. *Generative Models for Synthetic Biology*. PhD thesis (Columbia University, 2020).

59. Nikolados, E.-M., Aodha, O. M., Cambray, G. & Oyarzún, D. A. From sequence to yield: deep learning for protein production systems. *Tech. Rep*. https://doi.org/10.1101/2021.11.18.468948v1 (2021).
60. Hummer, A. M., Schneider, C., Chinery, L. & Deane, C. M. Investigating the volume and diversity of data needed for generalizable antibody-antigen $\Delta\Delta$G prediction. *Nat. Comput. Sci* **5**, 635–647 (2025).
61. Fan, X. et al. ICVAE: interpretable conditional variational autoencoder for de novo molecular design. *Int. J. Mol. Sci.* **26**, 3980 (2025).
62. Araujo, R. P. & Liotta, L. A. Universal structures for adaptation in biochemical reaction networks. *Nat. Commun.* **14**, 2251 (2023).
63. Kingma, D. P. & Welling, M. Auto-Encoding Variational Bayes. *Proc. 2nd International Conference on Learning Representations (ICLR)* (2014).
64. Sohn, K., Lee, H. & Yan, X. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, Vol. 28 (Curran Associates, Inc., 2015).
65. Zhang, J. et al. Tahoe-100M: a giga-scale single-cell perturbation atlas for context-dependent gene function and cellular modeling. https://doi.org/10.1101/2025.02.20.639398v3 (2025).
66. Phaneuf, P. V., Gosting, D., Palsson, B. O. & Feist, A. M. ALEdb 1.0: a database of mutations from adaptive laboratory evolution experimentation. *Nucleic Acids Res.* **47**, D1164–D1171 (2019).
67. Tonner, P. D., Pressman, A. & Ross, D. Interpretable modeling of genotype-phenotype landscapes with state-of-the-art predictive power. https://doi.org/10.1101/2021.06.11.448129v1 (2021).
68. Tack, D. S. et al. Precision engineering of biological function with large-scale measurements and machine learning. *PLoS ONE* **18**, e0283548 (2023).
69. Eaton, D. S. et al. Essentialome-wide multigenerational imaging reveals mechanistic origins of cell growth laws. https://doi.org/10.1101/2025.06.10.658525v2 (2025).
70. Gutiérrez-Sacristán, A. et al. GenoPheno: cataloging large-scale phenotypic and next-generation sequencing data within human datasets. *Brief. Bioinform.* **22**, 55–65 (2021).
71. Bajić, D., Vila, J. C. C., Blount, Z. D. & Sánchez, A. On the deformability of an empirical fitness landscape by microbial evolution. *Proc. Natl. Acad. Sci. USA* **115**, 11286–11291 (2018).
72. Greenman, K. P., Amini, A. P. & Yang, K. K. Benchmarking uncertainty quantification for protein engineering. *PLoS Comput. Biol.* **21**, e1012639 (2025).
73. Yeo, H. C. & Selvarajoo, K. Machine learning alternative to systems biology should not solely depend on data. *Brief. Bioinform.* **23**, bbac436 (2022).
74. Yan, Z. et al. Integrating deep learning and synthetic biology: a co-design approach for enhancing gene expression via N-terminal coding sequences. *ACS Synth. Biol.* **13**, 2960–2968 (2024).
75. Yin, C. H. et al. Iterative deep learning-design of human enhancers exploits condensed sequence grammar to achieve cell type-specificity. https://doi.org/10.1101/2024.06.14.599076v1 (2024).
76. Lin, J., Wang, X., Liu, T., Teng, Y. & Cui, W. Diffusion-based generative network for de novo synthetic promoter design. *ACS Synth. Biol.* **13**, 1513–1522 (2024).
77. Friedman, R. Z. et al. Active learning of enhancer and silencer regulatory grammar in a developing neural tissue. https://doi.org/10.1101/2023.08.21.554146v2 (2024).
78. Gosai, S. J. et al. Machine-guided design of cell-type-targeting cis-regulatory elements. *Nature* **634**, 1211–1220 (2024).
79. Hayes, T. et al. Simulating 500 million years of evolution with a language model. *Science* **387**, 850–858 (2025).
80. Xu, Z. et al. A machine learning-based approach for improving plasmid DNA production in *Escherichia coli* fed-batch fermentations. *Biotechnol. J.* **19**, 2400140 (2024).
81. Lugagne, J.-B., Blassick, C. M. & Dunlop, M. J. Deep model predictive control of gene expression in thousands of single cells. *Nat. Commun.* **15**, 2148 (2024).
82. Chopra, S., Hadsell, R. & LeCun, Y. Learning a similarity metric discriminatively, with application to face verification. In *Proc. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 1, 539–546 (IEEE, 2005).
83. Khosla, P. et al. Supervised contrastive learning. In *Proc. 34th International Conference on Neural Information Processing Systems* (2020).
84. Schroff, F., Kalenichenko, D. & Philbin, J. FaceNet: a unified embedding for face recognition and clustering. In *Proc. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 815–823 (IEEE, 2015).
85. Oord, A. v. d., Li, Y. & Vinyals, O. Representation learning with contrastive predictive coding. Preprint at https://doi.org/10.48550/arXiv.1807.03748 (2019).
86. Frosst, N., Papernot, N. & Hinton, G. Analyzing and Improving Representations with the Soft Nearest Neighbor Loss. *Proc. 36th International Conference on Machine Learning (ICML)*, In Proceedings of Machine Learning Research **97**:1986−1995 (2019).
87. Ascarate, A., Lebrat, L., Cruz, R. S., Fookes, C. & Salvado, O. Improving the generation of VAEs with high dimensional latent spaces by the use of hyperspherical coordinates. Preprint at https://doi.org/10.48550/arXiv.2507.15900 (2025).
88. Amigo, G., Perea, P. R. & Marks, R. J. Mitigating algorithmic bias on facial expression recognition. Preprint at https://doi.org/10.48550/arXiv.2312.15307 (2023).
89. Kopf, A., Fortuin, V., Somnath, V. R. & Claassen, M. Mixture-of-experts variational autoencoder for clustering and generating from similarity-based representations on single cell data. *PLoS Comput. Biol.* **17**, e1009086 (2021).
90. Higgins, I. et al. beta-VAE: Learning basic visual concepts with a constrained variational framework. https://openreview.net/forum?id=Sy2fzU9gl (2017).
91. Tam, E. & Dunson, D. B. On the statistical capacity of deep generative models. Preprint at https://doi.org/10.48550/arXiv.2501.07763 (2025).
92. Wang, Y., Blei, D. M. & Cunningham, J. P. Posterior collapse and latent variable non-identifiability. In *Proc. 35th International Conference on Neural Information Processing Systems. Curran Associates Inc.*, Red Hook, NY, USA, Article 416, 5443–5455 (2021).
93. Koran, A., Hojjati, H. & Armanfard, N. Unveiling the flaws: a critical analysis of initialization effect on time series anomaly detection. Preprint at https://doi.org/10.48550/arXiv.2408.06620 (2024).
94. Tonolini, F., Aletras, N., Jiao, Y. & Kazai, G. Robust weak supervision with variational auto-encoders. In *Proc. 40th International Conference on Machine Learning*, 34394–34408 (PMLR, 2023).
95. Bui, N. T., Savova, G. & Wang, L. 2025. Assessing the Macro and Micro Effects of Random Seeds on Fine-Tuning Large Language Models. In *Proc. 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 41–46 (2025).
96. Hawkins-Hooker, A. et al. Generating functional protein variants with variational autoencoders. *PLoS Comput. Biol.* **17**, e1008736 (2021).
97. Åkesson, J., Töger, J. & Heiberg, E. Random effects during training: implications for deep learning-based medical image segmentation. *Comput. Biol. Med.* **180**, 108944 (2024).
98. Eryılmaz, B., Koraş, O. A., Schlötterer, J. & Seifert, C. Investigating the Impact of Randomness on Reproducibility in Computer Vision: A Study on Applications in Civil Engineering and Medicine. *IEEE 6th International Conference on Cognitive Machine Intelligence*, pp. 265−274 (2024).
99. Yi, X., Liu, S., Wu, Y., McCloskey, D. & Meng, Z. BPP: a platform for automatic biochemical pathway prediction. *Briefings in Bioinformatics* **25**, https://doi.org/10.1093/bib/bbae355 (2024).

100. Torres, J. G. et al. A multi-objective active learning platform and web app for reaction optimization. https://doi.org/10.26434/chemrxiv-2022-cljcp (2022).
101. Morita, Y. et al. Applying Bayesian optimization with Gaussian process regression to computational fluid dynamics problems. *J. Comput. Phys.* **449**, 110788 (2022).
102. Lei, B. et al. Bayesian optimization with adaptive surrogate models for automated experimental design. *npj Comput. Mater.* **7**, 1–12 (2021).
103. Greenhill, S., Rana, S., Gupta, S., Vellanki, P. & Venkatesh, S. Bayesian optimization for adaptive experimental design: a review. *IEEE Access* **8**, 13937–13948 (2020).
104. Borkowski, O. et al. Large scale active-learning-guided exploration for in vitro protein production optimization. *Nat. Commun.* **11**, 1872 (2020).
105. Pandi, A. et al. A versatile active learning workflow for optimization of genetic and metabolic networks. *Tech. Rep*. https://doi.org/10.1101/2021.12.28.474323v2 (2021).
106. Crowther, M., Wipat, A. & Goñi-Moreno, N. A network approach to genetic circuit designs. *ACS Synth. Biol*. https://doi.org/10.1021/acssynbio.2c00255 (2022).
107. Brixi, G. et al. Genome modeling and design across all domains of life with Evo 2. https://doi.org/10.1101/2025.02.18.638918v1 (2025).
108. Hasibi, R., Michoel, T. & Oyarzún, D. A. Integration of graph neural networks and genome-scale metabolic models for predicting gene essentiality. *npj Syst. Biol. Appl.* **10**, 1–10 (2024).
109. Hartman, E. et al. Interpreting biologically informed neural networks for enhanced proteomic biomarker discovery and pathway analysis. *Nat. Commun.* **14**, 5359 (2023).
110. Burkhart, J. G. et al. Biology-inspired graph neural network encodes reactome and reveals biochemical reactions of disease. *Patterns* **4**, 100758 (2023).
111. Baranwal, M. et al. A deep learning architecture for metabolic pathway prediction. *Bioinformatics* **40**, btae359 (2024).
112. Shah, H. A., Liu, J., Yang, Z. & Feng, J. Review of machine learning methods for the prediction and reconstruction of metabolic pathways. *Front. Mol. Biosci.* **8**, https://doi.org/10.3389/fmolb.2021.634141 (2021).
113. Jendrusch, M. A. et al. AlphaDesign: a de novo protein design framework based on AlphaFold. *Mol. Syst. Biol.* 1–24, https://doi.org/10.1038/s44320-025-00119-z (2025).
114. Kyro, G. W., Brent, R. I. & Batista, V. S. HAC-Net: a hybrid attention-based convolutional neural network for highly accurate protein-ligand binding affinity prediction. *J. Chem. Inf. Model.* **63**, 1947–1960 (2023).
115. Wang, Y., Wu, S., Duan, Y. & Huang, Y. ResAtom system: protein and ligand affinity prediction model based on deep learning. Preprint at https://doi.org/10.48550/arXiv.2105.05125 (2021).
116. Baek, M. et al. Accurate prediction of protein–nucleic acid complexes using RoseTTAFoldNA. *Nat. Methods* **21**, 117–121 (2024).
117. Abramson, J. et al. Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature* **630**, 493–500 (2024).
118. Blattner, F. R. et al. The complete genome sequence of *Escherichia coli* K-12. *Science* **277**, 1453–1462 (1997).
119. Cabello-Garcia, J., Bae, W., Stan, G.-B. V. & Ouldridge, T. E. Handhold-mediated strand displacement: a nucleic acid based mechanism for generating far-from-equilibrium assemblies through templated reactions. *ACS Nano* **15**, 3272–3283 (2021).
120. Fern, J. et al. DNA strand-displacement timer circuits. *ACS Synth. Biol.* **6**, 190–193 (2017).
121. Nordgren, S., Slagter-Jäger, J. G. & Wagner, G. H. Real time kinetic studies of the interaction between folded antisense and target RNAs using surface plasmon resonance. *J. Mol. Biol.* **310**, 1125–1134 (2001).
122. Kidger, P. On Neural Differential Equations. University of Oxford (2021).
123. Gallup, O., Sechkar, K., Towers, S. & Steel, H. Computational synthetic biology enabled through JAX: a showcase. *ACS Synth. Biol.* **13**, 3046–3050 (2024).
124. Bradbury, J. et al. JAX: composable transformations of Python +NumPy programs. http://github.com/google/jax (2018).
125. Hennigan, T., Cai, T., Norman, T. & Martens, L. Haiku: Sonnet for JAX. https://github.com/google-deepmind/dm-haiku (2020).
126. Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations* (2015).
127. Microsoft. Neural Network Intelligence https://github.com/microsoft/nni (2021).
128. Powers, D. M. W. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *J. Mach. Learn. Technol* **2**, 37–63 (2011).

Acknowledgements

H.S. is supported in part by the UKRI-EPSRC under the EEBio Programme Grant, EP/Y014073/1. O.G. recognizes support from the John Brookman Scholarship and Wadham College, University of Oxford.

Author contributions

Software, implementation, and visualisation: O.G. Conceptualisation, investigation, methodology, writing and editing: O.G. and H.S. Supervision: H.S.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41540-026-00683-6.

Correspondence and requests for materials should be addressed to Olivia Gallup.