
Designing Cell-Type-Specific Promoter Sequences Using Conservative Model-Based Optimization

Aniketh Janardhan Reddy^{*,†}, Xinyang Geng^{*,†}, Michael H. Herschl^{*,†},
Sathvik Kolli, Aviral Kumar, Patrick D. Hsu[†], Sergey Levine[†], Nilah M. Ioannidis[†]
University of California, Berkeley

Abstract

Gene therapies have the potential to treat disease by delivering therapeutic genetic cargo to disease-associated cells. One limitation to their widespread use is the lack of short regulatory sequences, or promoters, that differentially induce the expression of delivered genetic cargo in target cells, minimizing side effects in other cell types. Such cell-type-specific promoters are difficult to discover using existing methods, requiring either manual curation or access to large datasets of promoter-driven expression from both targeted and untargeted cells. Model-based optimization (MBO) has emerged as an effective method to design biological sequences in an automated manner, and has recently been used in promoter design methods. However, these methods have only been tested using large training datasets that are expensive to collect, and focus on designing promoters for markedly different cell types, overlooking the complexities associated with designing promoters for closely related cell types that share similar regulatory features. Therefore, we introduce a comprehensive framework for utilizing MBO to design promoters in a data-efficient manner, with an emphasis on discovering promoters for similar cell types. We use conservative objective models (COMs) for MBO and highlight practical considerations such as best practices for improving sequence diversity, getting estimates of model uncertainty, and choosing the optimal set of sequences for experimental validation. Using three leukemia cell lines (Jurkat, K562, and THP1), we show that our approach discovers many novel cell-type-specific promoters after experimentally validating the designed sequences. For K562 cells, in particular, we discover a promoter that has 75.85% higher cell-type-specificity than the best promoter from the initial dataset used to train our models. Our code and data will be available at https://github.com/young-geng/promoter_design.

1 Introduction

Gene therapies treat diseases through the delivery and expression of therapeutic genetic cargo in disease-associated cells and tissues. Expression is controlled by a promoter sequence, a short regulatory DNA sequence (typically up to a few hundred base pairs (bp) long) placed upstream of the coding region. An ideal promoter for gene therapy would differentially induce expression in targeted cells while repressing expression in all other cells to increase effectiveness and reduce side-effects; i.e. it should be cell-type-specific and induce differential expression (DE). However, although there are over 400 types of cells in the human body [1], very few cell-type-specific promoters are known. Traditional methods to design cell-type-specific promoters rely heavily on manual curation or involve tiling known cis-regulatory elements (CREs) or transcription factor (TF) binding motifs ([2–6] among others). These methods are difficult to automate and are not guaranteed to work, especially in less

^{*}Equal contributions

[†]Corresponding authors: aniketh@berkeley.edu, young.geng@berkeley.edu, michael_herschl@berkeley.edu, pdhsu@berkeley.edu, svlevine@eecs.berkeley.edu, nilah@berkeley.edu

studied cell types. Directed evolution can be used to automate promoter design but is expensive to perform, typically requiring many experimental validation steps to continually improve designs, and does not make the best use of available data. Offline model-based optimization (MBO) has recently been used to design promoters and other CREs in a data-driven manner (Linder et al. [7], Wang et al. [8], Jores et al. [9], LaFleur et al. [10], Gosai et al. [11] among others). The offline MBO approach involves building machine learning (ML) models of promoter-driven expression (PE) using experimental measurements and then optimizing the promoter sequence using model predictions as surrogates for experimental measurements. Offline MBO has the potential to accelerate promoter discovery by being automated and relatively data-efficient; however, the field is lacking a generalizable MBO-based framework for designing cell-type-specific promoters in a data-efficient manner, while systematically accounting for practical considerations including:

1. Data-efficiency: Measuring PE is expensive and time-consuming, necessitating the effective use of available data during design. Most previous work on offline MBO for promoter design has used large PE datasets for model training (e.g. from massively parallel reporter assays, MPRA) that are only available for a few well-studied cell types, and their modelling strategies have not been evaluated when designing cell-type-specific promoters for less-studied cell types using small PE datasets.

2. Minimizing adversarial designs: Using models for optimization, rather than just prediction, presents key additional challenges, since naïvely optimizing designs based on predicted DE can lead to *adversarial* designs that fool a model into outputting desirable values while having undesirable values in reality. Previous promoter design workflows using optimization techniques such as *in silico* directed evolution, gradient ascent, or generative models do not account for such designs.

3. Selecting design candidates while accounting for sequence diversity and uncertainty in estimates of a sequence's goodness: Offline MBO can design many sequences that are predicted to be cell-type-specific by the PE model. However, due to limited budgets and time for wetlab experiments, typically only a subset of the candidates can be validated. Thus, we need a systematic approach to choose promising candidates from a larger set of designs in a principled manner. During selection, it is important to consider both the diversity of the chosen candidates and the uncertainty in model predictions. Choosing diverse candidate designs is important (1) to improve the chances of successfully discovering a cell-type-specific promoter upon experimental validation, since we lack complete information about how a sequence controls PE, and (2) to explore sequence space more effectively during several rounds of design, leading to progressively better models and designs. It is also important to choose candidates that have low uncertainty in model predictions while also balancing their potential goodness. This sequence selection process is crucial for a principled MBO workflow and has been overlooked by previous work.

Here, we present a systematic framework and case study for applying offline MBO to the real world problem of promoter design, backed by empirical evidence from wetlab experiments evaluating DE in three closely related blood-based cell types. To improve data-efficiency through transfer learning, we follow previous work [12] and pretrain sequence-based deep learning (DL) models of PE using large existing MPRA datasets that measure PE in different contexts, and fine-tune them using a small PE dataset collected from the target cell types. To address adversarial designs, we extend the conservative objective models (COMs) framework [13] to our setting: while fine-tuning models, we use a loss term that reduces the predicted DE of adversarial designs. These fine-tuned models are then used to design cell-type-specific promoters by performing gradient ascent on the inputs. Upon generating multiple candidate sequences, we propose a selection scheme to choose a diverse yet effective subset of final candidates that uses uncertainty estimates from an independently trained ensemble. We use our workflow to design cell-type-specific promoters for three leukemia cell lines (Jurkat, K562, and THP1) and present experimental evidence of its effectiveness. Existing studies have designed promoters for markedly different cell types derived from different tissues and germ layers [11], a simpler problem since such cell types typically have very different regulatory environments with different transcription factors (TFs) expressed. We show that our MBO approach works in a more difficult setting where we aim to design promoters that are only expressed in one of three hematopoietic leukemia cell lines. Our designs improve upon the cell-type-specificity of sequences from the fine-tuning dataset, **improving 72.12% of sequences in Jurkat cells and 80.48% in K562 cells. Additionally, in K562 cells, we identify a promoter with 75.85% higher DE than the best promoter** from the fine-tuning dataset. These results suggest that applying our approach over multiple rounds of modelling, design, and experimental validation will enable the discovery of promoters with even higher cell-type-specificity, by shifting the training distribution towards higher DE sequences at each iteration.

2 Related work

Promoter design methods: Promoters have traditionally been designed using heuristics and manual curation. For example, Selvakumaran et al. [3] and Yun et al. [4] found cell-type-specific promoters for ovarian and breast cancer cells, respectively, by identifying genes that are differentially expressed in these cells and showing that their promoters exhibit cell-type-specificity. Nissim et al. [5] designed cell-type-specific promoters for ovarian and breast cancer cells by identifying differentially expressed TFs and tiling their binding motifs. However, these studies do not provide a generalizable method to design cell-type-specific promoters for any given cell type. In an attempt to address this issue, recent work has sought to develop more automated and data-driven design methods. Kotopka and Smolke [14] and Jores et al. [9] used sequence-based convolutional neural network (CNN) models alongside simple optimization techniques such as *in silico* directed evolution and gradient ascent to design promoters for high expression in yeast and various plant cells, respectively. Wang et al. [8] generated E. coli promoters by training a generative adversarial network (GAN) on natural E. coli promoters and then using a PE predictor to further filter generated sequences. Gosai et al. [11] trained a sequence-based CNN model of PE using a large MPRA dataset from three very different cell lines. They used this model with three design algorithms to produce cell-type-specific promoters that they show to be effective and diverse. Here, we build on previous work in three crucial ways. First, we exploit the benefits of pretraining on large existing PE datasets to build accurate models for target cell types in a more data efficient manner compared to previous methods. Second, we leverage COMs [13], a powerful offline MBO method, to design promoters using these models more effectively, minimizing adversarial designs. Finally, we propose a sequence selection strategy that explicitly accounts for sequence diversity and model uncertainty to choose a subset of sequences for experimental validation from a larger set of design candidates.

Offline MBO for designing biological sequences: Apart from COMs, other offline MBO methods have been proposed for designing biological sequences. While COMs only requires discriminative models, most of the other methods rely on generative models. For example, Brookes et al. [15] built variational autoencoders (VAEs) that generate desirable designs by iteratively improving the VAE and its designs, guided by a design model. Jain et al. [16] used GFlowNets for sequence design, another class of generative models that are trained to output designs that span all modes of some design model's prediction distribution. Linder et al. [7]'s deep exploration networks (DENs) are generative models that are trained to output diverse yet desirable sequences and again use design model predictions to guide the training process. They experimentally validated their method and showed that alternative polyadenylation (APA) sites designed using DENs were better than those designed using gradient ascent. Here we use DENs as a baseline in our analyses, since it was validated using wetlab experiments.

3 Background and motivation

The cell-type-specific promoter design problem: A typical promoter design experiment ([17, 14, 9, 10] among others) starts by identifying target cell types and defining an objective such as increasing the absolute or differential expression levels of some gene. Then, an initial set of candidate promoters is prepared by leveraging existing data sources and heuristics, or even using random sequences. To test the efficacy of these promoters, reporter assays that collect PE measurements for a large batch of promoters simultaneously are used (batch sizes range from a few hundred to many thousands or even millions). Using these initial PE measurements, better promoters can be designed using various methods such as directed evolution, shuffling motifs enriched in desirable sequences, or offline MBO. The designed promoters then need to be experimentally validated using another reporter assay. If a sufficiently desirable promoter is not discovered, the design and validation steps can be repeated.

Here, we focus on designing short cell-type-specific promoters in a data-constrained setting. In this setting, we have a set of target cell types C , from which we have a small PE dataset – a few thousand sequences and corresponding PE measurements from all cell types in C . Current pooled oligonucleotide synthesis methods impose length constraints on DNA sequences. This technical limitation restricts high-throughput synthesis and measurement capabilities to promoters of at most a few hundred base pairs (bp). For any target cell type $tc \in C$, we define the DE of a promoter $\mathbf{x}$ (i.e. its cell-type-specificity) as:

$$\text{DE}^{tc}(\mathbf{x}) = \frac{1}{(|C| - 1)} \sum_{oc \neq tc} [\text{PE}^{tc}(\mathbf{x}) - \text{PE}^{oc}(\mathbf{x})], \quad (1)$$

where $PE^c(\mathbf{x})$, $c \in C$, is the experimentally measured expression value induced by $\mathbf{x}$ in cell type c . Our goal is to design sequences that maximize DE^{tc} - the average difference in PE between the target cell type tc and the other cell types.

Offline MBO: An offline MBO algorithm aims to produce designs that maximize some objective function, using a provided static dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$ of designs $\mathbf{x}_i$ and a corresponding measurement of the objective value y_i . The algorithm analyzes this dataset and produces an optimized candidate design $\mathbf{x}^*$, which is evaluated against the true objective function. This process often involves learning a proxy objective f_θ mapping designs to corresponding objective values, which we hereafter refer to as the **design model**. Optimization is then performed to find an input which maximizes this learned design model: $\mathbf{x}^* = \arg \max_{\mathbf{x}} f_\theta(\mathbf{x})$, using methods such as *in silico* directed evolution, gradient ascent, or by building generative models.

Here, we use offline MBO algorithms to design promoters that maximize the objective function mentioned in Eqn 1. When several rounds of promoter design are performed, we can use all available PE measurements from previous rounds as the “static” dataset for offline MBO. In our workflow, we propose an additional selection step after running offline MBO that diversifies the final designs if needed, making the pipeline more suitable for multi-round optimization.

4 Our MBO workflow for designing promoters in data-constrained settings

This section presents our approach to design cell-type-specific promoters that accounts for key practical considerations. It is summarized in Figure 1, with a detailed flowchart in Figure S.2.

4.1 Building models of promoter-driven expression in a data-efficient manner

As previously mentioned, offline MBO algorithms need a design model that approximates the objective function. When designing cell-type-specific promoters for some target cell type $tc \in C$, our objective function is $DE^{tc}(\mathbf{x})$ that is in turn computed using all $PE^c(\mathbf{x})$, $c \in C$, values. Therefore, instead of directly modelling $DE^{tc}(\mathbf{x})$, we simultaneously model all $PE^c(\mathbf{x})$ values using multi-task learning (MTL). This allows us to use one design model to design cell-type-specific promoters for all $c \in C$. In this section, we provide guidelines for building these models based on the findings of Reddy et al. [12], who studied model architectures and transfer learning strategies for modelling PE.

Model architecture: Recent work has shown that genomic data can be accurately modelled using sequence-based models – models that take in one-hot encoded DNA sequences as input to predict associated experimental assay measurements such as gene expression, histone modifications, and TF-binding (e.g. [18–20]). Reddy et al. [12] find that a model consisting of convolutional layers followed by transformer layers (MTLucifer) outperforms other architectures when used to model PE without any transfer learning.

Transfer learning to boost accuracy: Effective offline MBO hinges on accurate design models, and building accurate models often requires large datasets. However, collecting large datasets with experimental PE measurements in multiple cell types is expensive and time-consuming. In such data-constrained settings, transfer learning can be very beneficial. Reddy et al. [12] showed that pretraining sequence-based models on large related genomic datasets before fine-tuning on a smaller

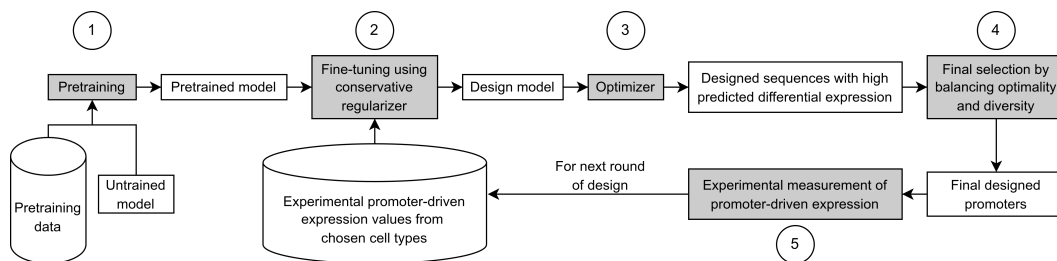


Figure 1: Our workflow for designing cell-type-specific promoters. Five main steps are highlighted in grey: (1) pretrain a base model using existing large genomic datasets; (2) fine-tune the pretrained model using the experimentally measured PE data collected so far, while also using a conservative regularizer; (3) use a gradient ascent-based optimizer to design sequences that have high predicted DE; (4) apply a final sequence selection algorithm that balances optimality and diversity to choose a smaller subset of the designed sequences; (5) experimentally measure the PE of the selected designed sequences. The last four steps can be repeated when running multiple rounds of design iterations.

PE dataset from target cells leads to significantly better modelling of the smaller dataset compared to training exclusively on the smaller dataset. The three best approaches that they identified are:

1. Linear probing¹ of Enformer [20] predictions using Lasso [21]: Enformer is a large sequence-based model (again, consisting of convolutional and transformer layers) trained on thousands of epigenomic datasets from humans and mice, allowing it to accurately learn sequence features that control gene expression. Since it is expensive to train Enformer (3 days on 64 TPU v3 cores), we often need to rely on the pretrained models published by Avsec et al. [20], limiting flexibility in terms of changing architectures or using alternate deep learning (DL) libraries. It can also be difficult to use Lasso with gradient-based optimization techniques when performing offline MBO. Therefore, this approach is only recommended when the published pretrained Enformer model can be easily incorporated into your code base without any architectural changes, and when the offline MBO algorithm you are using is not reliant on gradients (e.g. CbAS [15]).

2. Fine-tuning Enformer by using its embeddings to predict PE in target cells: The performance of this approach is very similar to the previous approach. Moreover, using fine-tuning instead of Lasso allows us to easily employ gradient-based offline MBO methods. Thus, this approach is recommended when the pretrained Enformer model is compatible with your code base and you do not wish to make any changes to its architecture that would necessitate re-training.

3. Pretraining MTLucifer on large existing MPRA datasets before fine-tuning it to predict PE in target cells: Certain MPRA datasets (e.g. [22, 23]) measure PE in selected cell lines from a large number of sequences. These data are useful for transfer learning, since models can learn a broad set of features that are generally useful for predicting PE. This approach is very inexpensive (33 hours for pretraining on a single Nvidia A40 GPU), which makes it easier to experiment with architectures and hyperparameters. Hence, this approach is recommended when you want to use custom architectures for modelling your data, or if you want flexibility in the DL libraries used to build your code base.

Using any of these approaches should yield an accurate model of PE in our target cells. However, since we focus on performing offline MBO using COMs, which require differentiable design models, only the last two approaches are compatible with our workflow.

4.2 Performing offline MBO while mitigating adversarial designs using COMs

Once we have accurate design models, we can couple them with offline MBO algorithms to design promoters. However, during the design process, we need to address adversarial designs that might arise due to the distribution shift problem. We describe the problem below and how it can be mitigated using COMs [13]. Then, we specify how COMs can be used to design cell-type-specific promoters.

Distribution shift problem in offline MBO and conservative regularization: While we can train an accurate design model $f_\theta(\mathbf{x})$, it may still suffer from generalization failures common to supervised regression models, especially when designed sequences are far away from the training data's distribution i.e. when there is a distribution shift. Such adversarial sequences can be easily designed by optimizers by heavily mutating training sequences, especially those that are already good. To alleviate this issue, we suggest learning and using *conservative* models of the objective function, or COMs [13], as design models. These models are trained using a *conservative regularizer* that penalizes high predictions on the set of potentially undesirable promoters $\mu(\mathbf{x})$ that are not in the training set but appear promising under the current design model, thus preventing the optimizer from designing adversarial promoters. Concretely, conservative models use the following loss:

$$\min_{\theta} \underbrace{\mathbb{E}_{\mathbf{x} \sim \mathcal{D}} [(f_\theta(\mathbf{x}) - y)^2]}_{:= \text{supervised loss}} + \alpha \underbrace{(\mathbb{E}_{\mathbf{x} \sim \mu} [f_\theta(\mathbf{x})] - \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} [f_\theta(\mathbf{x})])}_{:= \text{conservative regularizer}} \quad (2)$$

where α is the conservatism coefficient and controls the degree of “conservatism”.

Training design models for designing cell-type-specific promoters using conservative regularization: The previous subsection mentions two approaches for building differentiable design models. In either approach, we have a pretrained model (either a pretrained Enformer or MTLucifer model) that is fine-tuned using a small dataset from the target cells. To incorporate conservative regularization into the fine-tuning process, we modify the loss. Let us denote the model prediction for PE in cell

¹As an alternative to fine-tuning a pretrained network, which involves updating all weights in the network, one can extract embeddings or outputs from the pretrained network and fit a simple linear model to make predictions for a downstream task without updating the full pretrained network. This is called linear probing.

type c as $\text{PE}_\theta^c(\mathbf{x})$, and the predicted DE in c as $\text{DE}_\theta^c(\mathbf{x})$, then the overall fine-tuning objective is:

$$\min_{\theta} \sum_{c \in C} \{ \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} [(\text{PE}_\theta^c(\mathbf{x}) - \text{PE}^c(\mathbf{x}))^2] + \alpha (\mathbb{E}_{\mathbf{x} \sim \mu_c} [\text{DE}_\theta^c(\mathbf{x})] - \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} [\text{DE}_\theta^c(\mathbf{x})]) \} \quad (3)$$

In contrast to the canonical formulation of COMs in Eqn 2, we use DE in the regularization term, a value derived from the model's expression predictions, instead of using expression itself, as our goal is to maximize DE and we want to avoid adversarial sequences with erroneously high predicted DE using the conservative regularizer. Furthermore, since we use a single design model for all target cells, the regularization term needs to be computed for all target cells. In contrast, the canonical COMs formulation works for only one objective function. Similar to Trabucco et al. [13], to get distribution $\mu_c(\mathbf{x})$ consisting of sequences with potentially overestimated DE in c in every training step, we can use the Adam optimizer [24] to perform T steps of gradient ascent on $\text{DE}_\theta^c(\mathbf{x})$ starting from sequences in the training batch (in our experiments: learning rate = 0.5, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $T = 100$). Since the DNA sequences are discrete, we perform gradient ascent in the probability simplex in the one-hot encoded space, parameterized by a softmax function. At the end of T steps of gradient ascent, we perform a hard clipping so that the resulting sequence is a valid one-hot encoding. The full fine-tuning objective is optimized until convergence and the model checkpoint with the lowest validation set loss is used as the design model. *Importantly, it is difficult to choose an α a priori. Therefore, we train multiple design models with different α values and use all of them to design candidate promoters (in our experiments, $\alpha \in \{0, 0.0003, 0.001, 0.003, 0.01, 0.03\}$).*

Designing promoters using gradient ascent: After training a design model, starting from each promoter in the fine-tuning dataset (henceforth referred to as the **starting sequence**), we apply the same gradient ascent-based optimization process as that used to build $\mu_c(\mathbf{x})$ to generate a design. This generation is performed using every design model (trained using different α values), yielding a large pool of candidate designed sequences.

4.3 Balancing diversity, optimality, and uncertainty for final sequence selection

From the design process described in the previous subsection, we get a large pool of candidate cell-type-specific promoters. Unlike typical offline MBO problems where an algorithm must produce only a single design for evaluation, as discussed in Section 3, designed promoters are typically evaluated in batches (say, of size K). Moreover, multiple iterations of promoter design can be performed. In this section we describe an algorithm to choose K promoters for experimental validation from the large set of candidates for cell type c , accounting for these facets of the promoter design problem.

Given that we have multiple design models, we cannot simply take the top K designs with highest predicted DE, since the predictions come from different models, making them difficult to compare with each other. Taking the top designs might also yield sequences that are very similar to each other. Finally, we also want to account for model uncertainty during selection and ideally choose high performing sequences that have low uncertainty.

Using an ensemble to get pessimistic estimates of DE that account for model uncertainty: To uniformly evaluate all candidate designed sequences, and potentially get more accurate predictions, we build an ensemble model consisting of models with slightly different architectures, using a train-validation-test split of the fine-tuning data that is different from the one used to build the design models. The constituent models are still pretrained prior to fine-tuning, but the conservative regularizer is not used during fine-tuning since these models are not directly used for design. To reduce the computational overhead for building the ensemble, one can use the same pretrained backbone in all constituent models but use different kinds of fine-tuning layers in each model. In our experiments, we use 36 models in the ensemble (Appendix E). To account for model uncertainty, we use the ensemble to compute $\widehat{\text{DE}}_\gamma^c(\mathbf{x})$, a pessimistic estimate of the DE in cell type c , for all candidates. The pessimistic estimate is defined as the lower confidence bound of the constituent models' predictions and is computed as the mean minus the standard deviation of the predictions. *Using this pessimistic estimate in the final selection process allows us to choose sequences that most models are confident about, further reducing the risk of choosing adversarial designs.*

Final selection algorithm: Our ultimate goal is to maximize the expected efficacy of the designs in the set of final sequences selected for experimental validation. Intuitively, this can be achieved by: **(i)** ensuring that our designs have high predicted DE, and **(ii)** ensuring that the designs are as diverse as possible. Diversity is important as designs predicted to be optimal might not actually be optimal when they are experimentally validated. Having a diverse set of sequences increases the likelihood

of *some* design being optimal, since we cover a broad region of the sequence space. We define a sequence set $\mathcal{S}$'s diversity as:

$$D(\mathcal{S}) = \frac{1}{2} \sum_{\mathbf{x} \in \mathcal{S}, \mathbf{x}' \in \mathcal{S}} \left[\mathcal{H}(\mathbf{x}, \mathbf{x}') + \mathcal{K}(\mathbf{x}, \mathbf{x}') \right]$$

where $\mathcal{H}$ is the normalized Hamming distance between two sequences (ensuring overall diversity) and $\mathcal{K}$ is the normalized Euclidean distance between the 6-mer frequency vectors of two sequences (ensuring diversity in potential TF-binding motifs composition). The distances are normalized by dividing them by their maximum possible values. To instantiate this intuition into a concrete strategy, we aim to select a set of K sequences $\mathcal{S}^* = \{\mathbf{x}_1^*, \mathbf{x}_2^*, \dots, \mathbf{x}_K^*\}$ such that $\mathcal{S}^*$ is the optimal solution to the following optimization problem:

$$\mathcal{S}^* := \arg \max_{\mathcal{S}} \sum_{\mathbf{x} \in \mathcal{S}} \widehat{\text{DE}}_{\gamma}^c(\mathbf{x}) + \beta D(\mathcal{S}) \quad (4)$$

where $\widehat{\text{DE}}_{\gamma}^c(\mathbf{x})$ denotes a pessimistic estimate of the DE computed using an ensemble, and β is the **diversity coefficient** that controls the fitness vs. diversity trade-off. While in theory we can obtain $\mathcal{S}^*$ by optimizing over all sequences, doing so is computationally intractable. So, the optimization in Eqn 4 is performed using a greedy algorithm over a large subset of candidates from the previous step. The subset consists of candidates that have positive ensemble average predicted DE, and also have high predicted expression in the target cell type c ² since this is also crucial for gene therapy applications. The algorithm sequentially chooses K sequences - in each step, given a set of sequences $\mathcal{S}'$ already chosen to be in the final set, it computes $\widehat{\text{DE}}_{\gamma}^c(\mathbf{x}) + \beta D(\mathcal{S}' \cup \{\mathbf{x}\})$ for every candidate sequence $\mathbf{x}$ not in $\mathcal{S}'$, and chooses the sequence that maximizes this value for inclusion in the final set.

4.4 Making adjustments to the design process based on performance and diversity metrics

We discuss common problems that may arise during design and provide heuristics to address them.

Designed sequences are generally not predicted to have higher DE than the starting sequences from the fine-tuning dataset: This issue might arise when the conservative regularizer is too powerful, making it difficult for the optimizer to design good sequences. Here, reducing the conservatism coefficient α should help. However, there might also be problems with the dataset that hinder the optimizer. For example, if the fine-tuning dataset has very few sequences that drive some level of DE in the target cell type, the design models might be poor at modelling such sequences. To determine if this is the case, one can look at the distribution of DE values in the training set and if there are too few sequences that have positive DE in the target cell type (e.g. less than 25% of the dataset), the design problem might be too difficult to solve using the available data.

Designed sequences are not diverse: The diversity of a set of designs can be measured using multiple metrics. We focus on base pair entropy and average pairwise edit distance (can be any edit distance e.g. Hamming distance) as they are easy to interpret. If we are designing sequences of length L , base pair entropy can be computed at every position in the sequence by determining the frequency of the 4 DNA bases across all designed sequences. For a diverse sequence set, this metric should be close to 2 at every position (near uniform usage of the 4 bases, entropy computed using binary logarithms). The average pairwise edit distance should be comparable to L (e.g. $L/2$). If these metrics are low, the diversity coefficient β can be increased to boost diversity. Using a more diverse set of starting sequences during optimization with gradient ascent can also improve the diversity of the designed sequences since we can converge upon different optima of the objective landscape.

5 Implementation and experimental evaluation of our approach

Next, we use the workflow proposed in the previous section to design 250 bp long cell-type-specific promoters for three leukemia cell lines: Jurkat, K562, and THP1. As mentioned in the introduction, designing such promoters is difficult since these cell lines are all mesoderm-derived hematopoietic cells compared to cell lines derived from different tissues or germ layers. Moreover, Jurkat and THP1 cells are relatively understudied and lack large sources of PE measurements. We also perform experimental validation of the designed sequences to show the effectiveness of our approach.

²Designs with ensemble average predicted PE in target cell type c greater than the 90th percentile of predicted PE in c for the fine-tuning dataset's sequences.

5.1 Setup

Model training (design and ensemble models): To train design models, we use an architecture similar to MTLucifer [12] (shown in Figure S.1). We pretrain the models on large existing MPRA datasets before fine-tuning them using a small PE dataset consisting of 17,104 measurements collected by Reddy et al. [12] from Jurkat, K562, and THP1 cells (dataset is described in more detail in Appendix C). We use this approach over fine-tuning Enformer, since our codebase uses Jax [25] for efficiently performing conservative regularization, and pretrained Enformer models are not available for Jax. Following Section 4.2, we train 5 design models, each with a different α value. We also build an ensemble for final sequence selection using a different split of the same dataset. Appendix E describes the training process in more detail.

Sequence design: We design 4,000 sequences for each cell type using our workflow. This number was chosen based on experimental constraints, and to keep the total number of sequences close to the size of the original training dataset. Each design model is used to design ~ 17 K candidate sequences (one design for every starting sequence from the fine-tuning dataset) using the process described in Section 4.2, and the final 4,000 sequences are chosen from these candidates using the algorithm from Section 4.3. As shown later in this section, the designed sequences are naturally diverse, and we set the diversity coefficient β to zero since we did not need to improve diversity.

Baselines: We also design sequences using two baselines, in order to determine the usefulness of various components of our workflow:

1. Motif tiling: This heuristic method to design cell type-specific promoters is similar in spirit to that proposed by Nissim et al. [5], and aims to improve upon known high DE sequences. We use it to determine the overall usefulness of our approach for designing promoters compared to traditional methods. Briefly, the motif tiling approach inserts motifs associated with high DE into sequences from the fine-tuning dataset that already have high DE, with the goal of increasing their DE even further. We run it for the three target cell types to design 520, 630, and 515 sequences for Jurkat, K562, and THP1 cells, respectively. Appendix F.1 presents the details of this method.

2. DENs: In Section 4.2, we use gradient ascent to optimize starting sequences and produce designs that increase the design model-predicted DE. However, as detailed in the Section 2, many previous studies use generative models for this purpose. To determine which optimization algorithm is better, we replace gradient ascent with DENs in our workflow and produce 2000 sequences per cell type (with $\beta = 10$). We specifically use DENs, as their designs were experimentally validated by Linder et al. [7] and they are explicitly trained to produce diverse sequences. Appendix F.2 describes DENs in more detail.

Experimental validation of sequences: We use a similar protocol as Reddy et al. [12] to experimentally determine the DE of designed sequences (Appendix D). Along with the designed sequences, we also measure the DE of the top sequences from the fine-tuning dataset (top 100 sequences for each cell type, so 300 sequences total), and of ~ 200 sequences from the fine-tuning dataset whose expression roughly uniformly spanned the full range of PE values. These sequences with measurements in both the fine-tuning set experiment and the validation experiment are used to determine whether designed sequences improve upon their starting sequences, using a procedure described later in this section. In total, we measure the DE of 20,741 sequences. After filtering out sequences with less than 100 reads in any experiment, we are left with 14,315 high-quality measurements for use in all downstream analyses.

5.2 Main Results

Evaluating the diversity of designed sequences: We analyze the diversity of the designs in terms of their mean base pair entropy and the mean Hamming distance between any two designs (Table 1).

Overall, our approach produces highly diverse sequences for all three cell types. We note that for THP1, DENs produce significantly less diverse designs. This is possibly due to the difficulty of the design problem - the fine-tuning dataset has few sequences with high DE in THP1, leading to the design model having fewer modes. This might make it difficult for a generative model trained from scratch to discover the design model's modes. Since gradient ascent starts from the sequences in the fine-tuning dataset, it can more easily discover the various modes. Finally, to make sure that our designs are distinct from the fine-tuning sequences, we also analyze the mean Hamming distance between a design and the closest fine-tuning sequence (Table S.1). We see that our designs are substantially different from the fine-tuning sequences, with designs being 125-140 bp away from the closest fine-tuning sequence on average. Thus, given a diverse fine-tuning dataset, our workflow produces designs that are not only distinct from one another but are also different from the fine-tuning

Method	Number of sequences			Mean base pair entropy			Mean Hamming distance		
	Jurkat	K562	THP1	Jurkat	K562	THP1	Jurkat	K562	THP1
Our approach	4000	4000	4000	1.87	1.96	1.96	176.18	183.73	184.17
DENs used instead of grad. asc.	2000	2000	2000	1.89	1.87	1.00	178.17	177.39	108.22
Motif tiling	520	630	515	1.79	1.90	1.92	168.71	179.37	180.95

Table 1: **Quantifying the diversity of designs.** Mean base pair entropy (using binary logarithms, maximum value = 2) and mean pairwise Hamming distance (maximum value = 250) are shown for the designed sequences. Mean base pair entropy is calculated by determining the entropy of each position across all designs and then averaging these values.

sequences, even with the application of conservative regularization, which might otherwise inhibit diversity in certain scenarios.

Does our approach design sequences that are better than the corresponding starting sequences?

Next, we analyze the overall effectiveness of our workflow and compare it to motif tiling. Since we aim to design promoters that have higher cell-type-specificity than promoters in the fine-tuning dataset, we need to quantify the improvement in DE from the starting sequences to the designed sequences. Direct comparison of DE values is challenging due to differences in experimental conditions and potential confounding factors like batch effects, and re-measuring the DE of all starting sequences is inefficient. To address this issue, we use a common set of sequences that span the expression range for any given cell type to calibrate DE values between experiments. Since these common set sequences have DE values from both experiments, we can compute percentile scores for both the starting and corresponding designed sequences among this common set. The design process is considered effective if the designed sequence achieves a higher percentile score than the starting sequence. This calibration is valid since the ranks of the sequences in the common set are concordant between experiments, as evidenced by the high Spearman correlation between DE values measured by Reddy et al. [12] and our experiments: 0.953 for Jurkat, 0.921 for K562, and 0.950 for THP1.

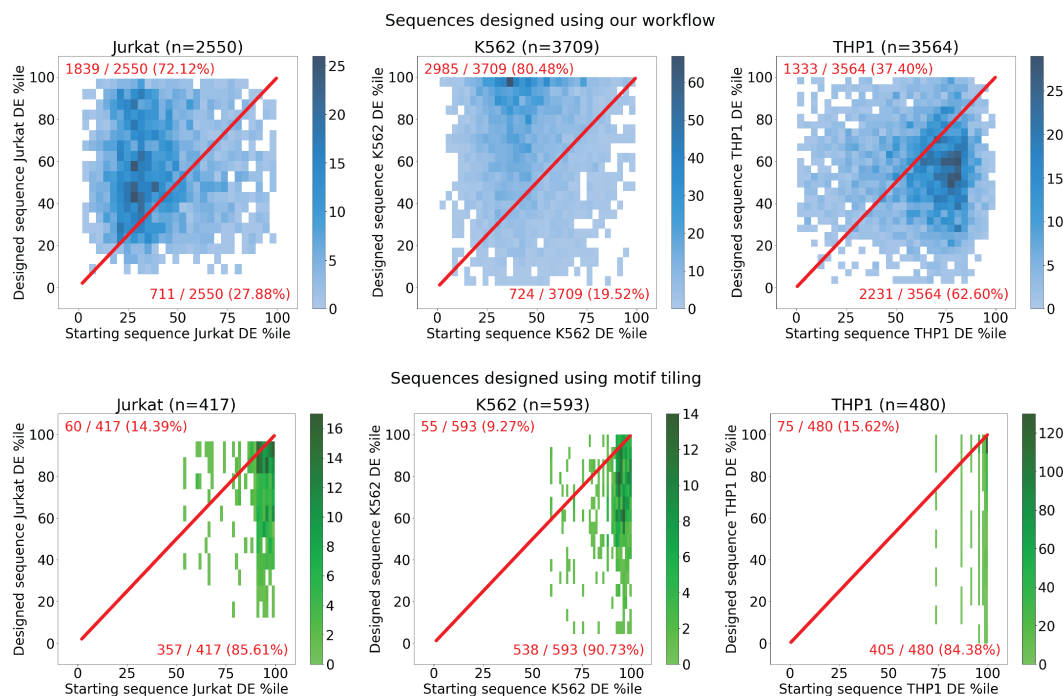


Figure 2: **Comparing the DE of designed sequences with the DE of corresponding starting sequences.** The top row of plots compare the percentile scores of starting and designed sequences from our approach, and the bottom row of plots compare the percentile scores for starting and designed sequences from motif tiling. Each column represents sequences designed for one of the three cell types. The $x = y$ line is shown in red, with the number of sequences above it highlighted in the top left corner and the number below it highlighted in the bottom right corner.

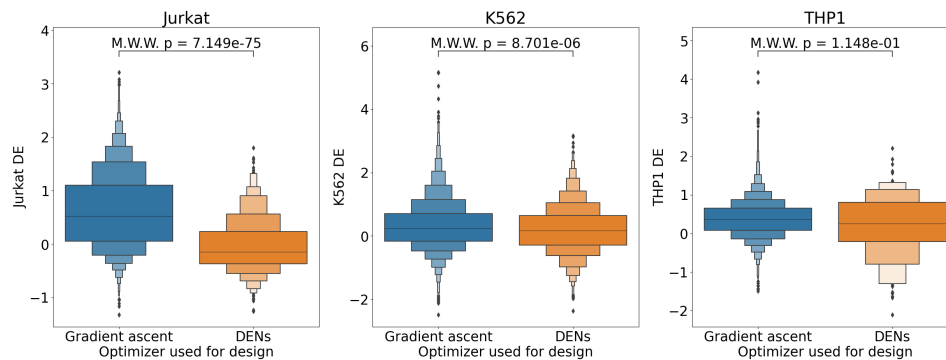


Figure 3: **Box plots comparing the DE of sequences designed using gradient ascent as the optimizer in our workflow vs. those from DENS.** p -values are computed using a Mann-Whitney-Wilcoxon (MWW) test that tests whether sequences from gradient ascent have higher DE than sequences from DENS.

Figure 2 shows the percentile scores of the starting and designed sequences from our approach and from motif tiling. Sequences designed for Jurkat and K562 using our approach improve upon the majority of corresponding starting sequences, highlighting the overall effectiveness of our approach in this difficult setting. **In fact, we design a promoter for K562 that has 75.85% higher DE ($=5.17$) than the best high DE sequence from the fine-tuning dataset ($=2.94$).** On the other hand, we observe that using motif tiling to improve on high DE sequences is ineffective, almost always leading to worse sequences. We note that our workflow tends to choose designs that come from relatively mediocre starting sequences. This is possibly because most sequences in the fine-tuning dataset have DE similar to these sequences, likely making the design and ensemble models accurate and confident near such sequences. Since our workflow significantly improves upon such sequences, we should be able to discover progressively better sequences over multiple rounds of model training, design, and experimental evaluation.

However, we fail to improve upon most starting sequences for THP1. This is likely because of the difficulty of the design problem. As mentioned previously, there are few high DE sequences in the fine-tuning dataset for THP1. Thus, it might be difficult for design models to be accurate in the space of high DE sequences, leading to poor designs. In such cases, collecting more experimental data for training might be the only effective way to improve designs.

Gradient ascent is better than DENS for designing sequences: Finally, we compare gradient ascent to DENS (Figure 3) to determine the better optimizer for use in our workflow. We observe that gradient ascent is generally more effective than DENS. Given that gradient ascent is also easier to implement and tune, we recommend it over generative models such as DENS.

6 Discussion and limitations

Designing cell-type-specific promoters is a crucial step in the effective application of gene delivery technologies. We introduce a novel workflow for promoter design based on COMs [13], tailored for real-world applications due to its data efficiency and ability to produce diverse, non-adversarial designs. We apply this approach to the challenging task of designing cell-type-specific promoters for three leukemia cell lines. We then experimentally validate our designs, demonstrating effectiveness for two out of three cell lines.

We also present limitations that practitioners should consider. The success of our workflow depends on the accuracy of the design models, and effectiveness across the three cell lines was related to design model accuracy for each cell line (Table S.2). This resulted in failure for THP1 cells, where we had lower quality fine-tuning data with few high DE sequences. Pretraining data may also affect performance, and many existing MPRA datasets used for pretraining come from K562 cells, likely explaining our superior performance in K562 compared to Jurkat. Finally, conservative regularization may inhibit diversity in certain cases as using conservatism leads to designs that are close to the training data. Although we do not observe this in our experiments (Tables S.1, and 1), using a fine-tuning dataset with limited sequence diversity might lead to very similar designs and addressing this issue is a good avenue for future work. Our MBO workflow highlights these and other practical considerations. Our approach is valuable for practitioners targeting a wide range of cell types, and will inspire further research in this area.

Acknowledgments

This work was partially supported by an H2H8 Graduate Research Grant to AJR, a Schmidt Futures grant to SL, and grants from the U.S. National Institutes of Health (R00HG009677) and the UC Noyce Initiative for Computational Transformation to NMI. NMI is a Chan Zuckerberg Biohub San Francisco Investigator.

This research used the Savio computational cluster resource provided by the Berkeley Research Computing program at the University of California, Berkeley (supported by the UC Berkeley Chancellor, Vice Chancellor for Research, and Chief Information Officer).

References

- [1] Tabula Sapiens Consortium. The tabula sapiens: A multiple-organ, single-cell transcriptomic atlas of humans. *Science*, 376(6594):eabl4896, 2022.
- [2] Carol H Miao, Kazuo Ohashi, Gijsbert A Patijn, Leonard Meuse, Xin Ye, Arthur R Thompson, and Mark A Kay. Inclusion of the hepatic locus control region, an intron, and untranslated region increases and stabilizes hepatic factor ix gene expression in vivo but not in vitro. *Molecular Therapy*, 1(6):522–532, 2000.
- [3] Muthu Selvakumaran, Rudi Bao, Anne PG Crijns, Denise C Connolly, Jillian K Weinstein, and Thomas C Hamilton. Ovarian epithelial cell lineage-specific gene expression using the promoter of a retrovirus-like element. *Cancer research*, 61(4):1291–1295, 2001.
- [4] Hye Jin Yun, Young-Hwa Cho, Youngsun Moon, Young Woo Park, Hye-Kyoung Yoon, Yeun-Ju Kim, Sung-Ha Cho, Young-Il Lee, Bong-Su Kang, Wun-Jae Kim, et al. Transcriptional targeting of gene expression in breast cancer by the promoters of protein regulator of cytokinesis 1 and ribonuclease reductase 2. *Experimental & Molecular Medicine*, 40(3):345–353, 2008.
- [5] Lior Nissim, Ming-Ru Wu, Erez Pery, Adina Binder-Nissim, Hiroshi I Suzuki, Doron Stupp, Claudia Wehrspaun, Yuval Tabach, Phillip A Sharp, and Timothy K Lu. Synthetic rna-based immunomodulatory gene circuits for cancer immunotherapy. *Cell*, 171(5):1138–1150, 2017.
- [6] Ming-Ru Wu, Lior Nissim, Doron Stupp, Erez Pery, Adina Binder-Nissim, Karen Weisinger, Casper Enghuus, Sebastian R Palacios, Melissa Humphrey, Zhizhuo Zhang, et al. A high-throughput screening and computation platform for identifying synthetic promoters with enhanced cell-state specificity (specs). *Nature communications*, 10(1):1–10, 2019.
- [7] Johannes Linder, Nicholas Bogard, Alexander B Rosenberg, and Georg Seelig. A generative neural network for maximizing fitness and diversity of synthetic dna and protein sequences. *Cell systems*, 11(1):49–62, 2020.
- [8] Ye Wang, Haochen Wang, Lei Wei, Shuailin Li, Liyang Liu, and Xiaowo Wang. Synthetic promoter design in escherichia coli based on a deep generative network. *Nucleic Acids Research*, 48(12):6403–6412, 2020.
- [9] Tobias Jores, Jackson Tonnies, Travis Wrightsman, Edward S Buckler, Josh T Cuperus, Stanley Fields, and Christine Queitsch. Synthetic promoter designs enabled by a comprehensive analysis of plant core promoters. *Nature Plants*, 7(6):842–855, 2021.
- [10] Travis L LaFleur, Ayaan Hossain, and Howard M Salis. Automated model-predictive design of synthetic promoters to control transcriptional profiles in bacteria. *Nature communications*, 13(1):5159, 2022.
- [11] Sager J Gosai, Rodrigo I Castro, Natalia Fuentes, John C Butts, Kousuke Mouri, Michael Alasoadura, Susan Kales, Thanh Thanh L Nguyen, Ramil R Noche, Arya S Rao, et al. Machine-guided design of cell-type-targeting cis-regulatory elements. *Nature*, pages 1–10, 2024.
- [12] Aniketh Janardhan Reddy, Michael H Herschl, Xinyang Geng, Sathvik Kolli, Amy X Lu, Aviral Kumar, Patrick D Hsu, Sergey Levine, and Nilah M Ioannidis. Strategies for effectively modelling promoter-driven gene expression using transfer learning. *bioRxiv*, 2024. doi: 10.1101/2023.02.24.529941. URL <https://www.biorxiv.org/content/early/2024/05/19/2023.02.24.529941>.

- [13] Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine. Conservative objective models for effective offline model-based optimization. In *International Conference on Machine Learning*, pages 10358–10368. PMLR, 2021.
- [14] Benjamin J Kotopka and Christina D Smolke. Model-driven generation of artificial yeast promoters. *Nature communications*, 11(1):2113, 2020.
- [15] David Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 773–782. PMLR, 09–15 Jun 2019. URL <https://proceedings.mlr.press/v97/brookes19a.html>.
- [16] Moksh Jain, Emmanuel Bengio, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Bonaventure F. P. Dossou, Chanakya Ajit Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, Lena Simine, Payel Das, and Yoshua Bengio. Biological sequence design with GFlowNets. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 9786–9801. PMLR, 17–23 Jul 2022. URL <https://proceedings.mlr.press/v162/jain22a.html>.
- [17] Michael R Schlabach, Jimmy K Hu, Mamie Li, and Stephen J Elledge. Synthetic design of strong promoters. *Proceedings of the national academy of sciences*, 107(6):2538–2543, 2010.
- [18] Jian Zhou and Olga G Troyanskaya. Predicting effects of noncoding variants with deep learning-based sequence model. *Nature methods*, 12(10):931–934, 2015.
- [19] Vikram Agarwal and Jay Shendure. Predicting mrna abundance directly from genomic sequence using deep convolutional neural networks. *Cell reports*, 31(7), 2020.
- [20] Žiga Avsec, Vikram Agarwal, Daniel Visentin, Joseph R Ledsam, Agnieszka Grabska-Barwinska, Kyle R Taylor, Yannis Assael, John Jumper, Pushmeet Kohli, and David R Kelley. Effective gene expression prediction from sequence by integrating long-range interactions. *Nature methods*, 18(10):1196–1203, 2021.
- [21] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.
- [22] Jason Ernst, Alexandre Melnikov, Xiaolan Zhang, Li Wang, Peter Rogov, Tarjei S Mikkelsen, and Manolis Kellis. Genome-scale high-resolution mapping of activating and repressive nucleotides in regulatory regions. *Nature biotechnology*, 34(11):1180–1190, 2016.
- [23] Joris van Arensbergen, Ludo Pagie, Vincent D FitzPatrick, Marcel de Haas, Marijke P Baltissen, Federico Comoglio, Robin H van der Weide, Hans Teunissen, Urmo Vösa, Lude Franke, et al. High-throughput identification of human snps affecting regulatory element activity. *Nature genetics*, 51(7):1160–1169, 2019.
- [24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL <http://arxiv.org/abs/1412.6980>.
- [25] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Neca, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL <http://github.com/google/jax>.
- [26] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.
- [27] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

- [28] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.
- [29] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.
- [30] Joris van Arensbergen, Vincent D FitzPatrick, Marcel de Haas, Ludo Pagie, Jasper Sluimer, Harmen J Bussemaker, and Bas van Steensel. Genome-wide mapping of autonomous promoter activity in human cells. *Nature biotechnology*, 35(2):145–153, 2017.
- [31] Rajiv Movva, Peyton Greenside, Georgi K Marinov, Surag Nair, Avanti Shrikumar, and Anshul Kundaje. Deciphering regulatory dna sequences and noncoding genetic variants using neural network models of massively parallel reporter assays. *PLoS One*, 14(6):e0218073, 2019.
- [32] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=Bkg6RiCqY7>.
- [33] Charles E Grant, Timothy L Bailey, and William Stafford Noble. Fimo: scanning for occurrences of a given motif. *Bioinformatics*, 27(7):1017–1018, 2011.
- [34] Jeff Vierstra, John Lazar, Richard Sandstrom, Jessica Halow, Kristen Lee, Daniel Bates, Morgan Diegel, Douglas Dunn, Fidencio Neri, Eric Haugen, et al. Global reference mapping of human transcription factor footprints. *Nature*, 583(7818):729–736, 2020.
- [35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

A Supplementary figures and tables

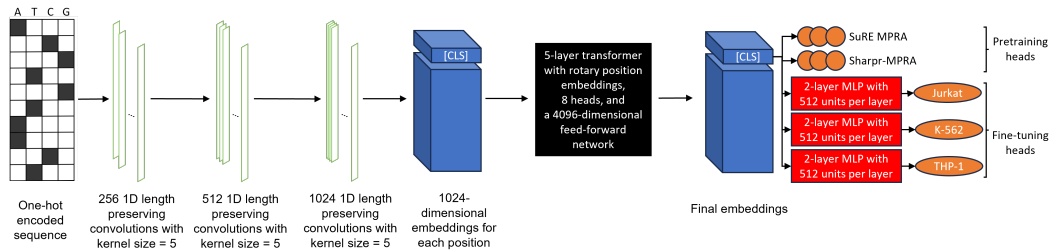


Figure S.1: Design model architecture: the convolutional layers use GELU activation [26], dropout with 0.1 probability [27], and are followed by a group norm layer [28] with each group size being 16. The MLP layers in the fine-tuning heads also use GELU activation. We apply the RoPE position embeddings [29] at each attention layer of the transformer.

Method	Number of sequences			Average Hamming distance to closest fine-tuning sequence		
	Jurkat	K562	THP1	Jurkat	K562	THP1
Our workflow	4000	4000	4000	126.46	132.17	140.27
Using DENs instead of grad. asc.	2000	2000	2000	149.63	155.44	152.89
Motif tiling	520	630	515	96.43	90.17	98.99

Table S.1: Average Hamming distance to the closest fine-tuning sequence for designs from every method (maximum value = 250, minimum value = 0). Even though our workflow optimizes sequences from the fine-tuning dataset to produce designs, the designs are generally quite distinct from the fine-tuning sequences. Since DENs are generative models that produce designs from random noise vectors, if they are trained properly, we expect their designs to be different from the fine-tuning sequences. Finally, motif tiling inserts motifs into sequences from the fine-tuning dataset. Thus, we expect its designs to be relatively more similar to the fine-tuning sequences than designs from the other methods.

B Detailed flowchart describing the steps in our workflow

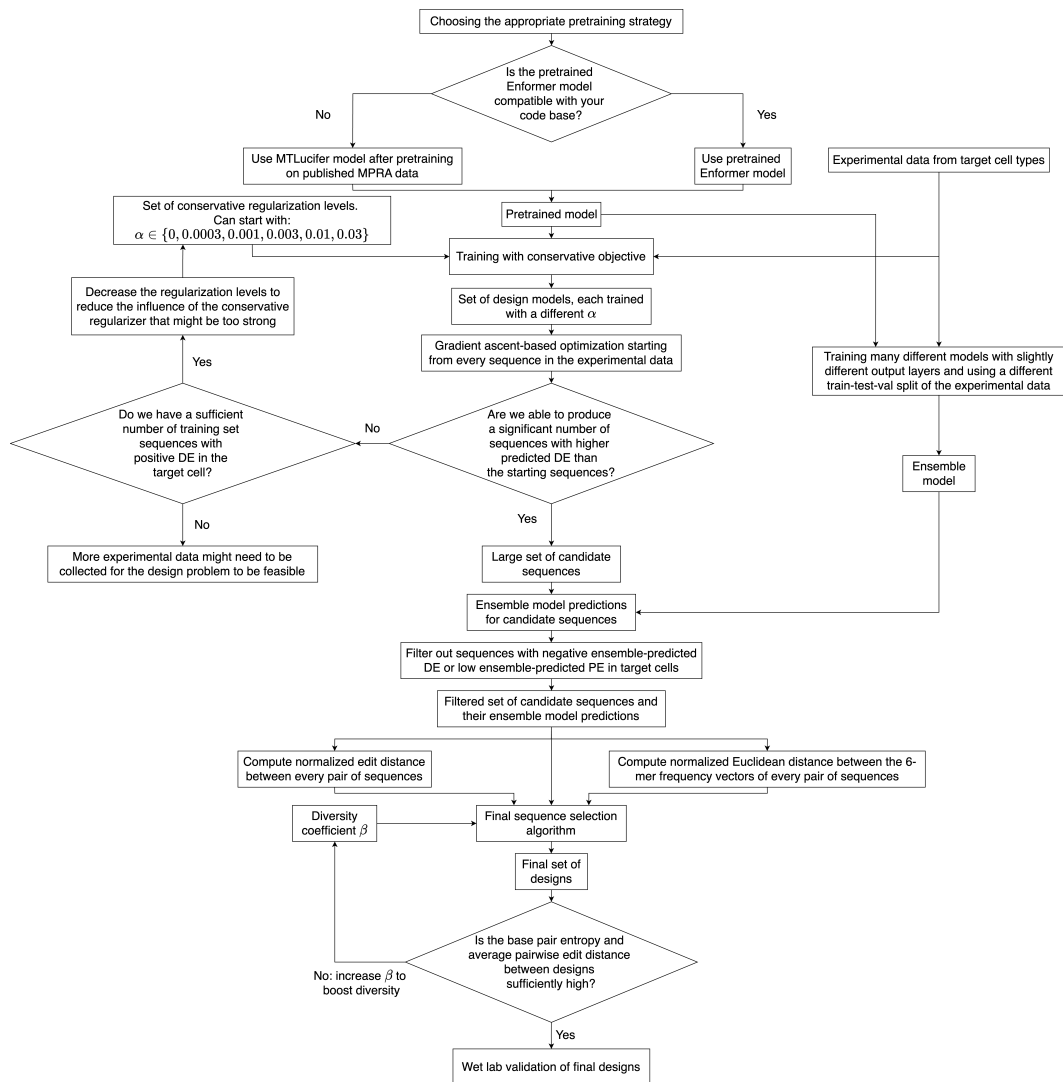


Figure S.2: Flowchart that shows every step of our workflow for a practitioner.

C Experimental data used for training design models

We use the PE measurements collected by Reddy et al. [12] for designing cell type-specific promoters. They provide 17,104 PE values in Jurkat, K562, and THP1 cells derived from 250 bp long manually designed promoters. They set out to measure the PE of 20,000 promoters chosen using heuristics that try to maximize the number of differentially expressed promoters. Nearly 50% of the tested promoters are from differentially expressed endogenous genes, another $\sim 40\%$ are crafted by assembling known and de-novo motifs which are abundant in the promoters of differentially expressed endogenous genes, and the remaining $\sim 10\%$ of promoters originate from highly expressed endogenous genes' promoters. Each promoter is integrated upstream of a minimal CMV promoter and the enhanced green fluorescent protein (EGFP) reporter gene within a lentiviral vector. The resulting expression in each cell line post-transduction is quantified by the levels of induced fluorescence. They get 17,104 PE values of adequate quality with two replicates and average the values from each replicate to get the final PE values - we use these values in our experiments.

This data was released with their codebase (https://github.com/anikethjr/promoter_models) under an AGPL-3.0 license.

D Experimental protocol for measuring promoter-driven expression of designed sequences

We used the same protocol as Reddy et al. [12] with the following modifications for testing the newly designed sequences. First, the cycle number for PCR amplification of the Twist oligopool was reduced from 12 to 10 cycles to reduce amplification bias, and the lentiviral backbone used a CMV promoter instead of an RSV promoter to drive the expression of the cargo from the lentiviral transfer plasmid to increase titer. To generate lentivirus, we used the LV-MAX kit (ThermoFisher A35684) following the protocol for a 1L flask, followed by the same concentration method. For lentiviral titration and full scale transduction, we used the same protocol, except we did not perform spinfection as this was deemed not necessary for adequate transduction efficiency. 24 hours after transduction, medium containing 8 μ g/mL polybrene and lentivirus was changed to fresh medium, and cells were allowed to recover for another 48 hours before performing puromycin selection. This additional time allowed for better expression of the puromycin resistance gene before selection, increasing functional titer. Full scale transductions were performed at $>500\times$ library coverage and at an MOI <0.3 . Cells were sorted using the same general methodology at a similar coverage; however, gDNA was extracted using the NucleoSpin Blood L (Machery-Nagel 740954) with the addition of RNase A (NEB T3018L) following the manufacturer's instructions. No changes were made to NGS library preparation, but sequencing was performed using an Illumina NovaSeq X Series, pooled with other NGS libraries with non overlapping indices.

E Training of design and ensemble models and their prediction performance

In this section, we detail the process we use to train our design and ensemble models. We train 5 design models with varying values of the conservatism coefficient α ($\alpha \in \{0, 0.0003, 0.001, 0.003, 0.01, 0.03\}$). We also highlight their prediction performances.

E.1 Pretraining details

Here, we provide details about our pretraining process. All design models and all constituent models of the ensembles are pretrained before being fine-tuned. Following Reddy et al. [12], we pretrain our models using data from Sharpr-MPRA [22] and SuRE MPRA [30, 23]. SuRE MPRA measures the expression induced by 150-500bp genomic fragments from 4 individuals from 4 different populations in the K562 and HepG2 cell lines. $\sim 2.4\text{B}$ and $\sim 1.2\text{B}$ fragments were found to be expressed in K562 and HepG2 respectively. Most fragments have very low expression and training on all measurements is time-consuming. Thus, Reddy et al. [12] define a classification task using this data that subsets the data and bins each sequence into one of 5 expression bins. We also use this classification task for pretraining. Sharpr-MPRA is a smaller MPRA that measures the expression from $\sim 487\text{K}$ 145bp sequences centered at DNase I peaks in K562 and HepG2 cells and in two different settings. Reddy et al. [12] use a preprocessed version of this data from Movva et al. [31] that builds a regression task with 12 outputs (2 replicates for expression measured in 2 settings in 2 cell lines, and 4 outputs that

correspond to the average expression across replicates). We use the same formulation for pretraining our models.

During pretraining, our models' output layers produce the probability of a sequence belonging to each of the expression bins for the SuRE MPRA-based pretraining task and directly predicts the expression values for the Sharpr-MPRA-based pretraining task. We minimize the sum of the negative log-likelihood (NLL) loss for the SuRE MPRA task and the mean squared error (MSE) loss for the Sharpr-MPRA task (since both tasks have distinct sequences, a training sequence only contributes to one of the two loss terms, the other loss is set to zero for that sequence).

We use the dataset splits defined by Reddy et al. [12] and train with a batch size of 448 using the AdamW optimizer [32] with 1e-4 learning rate and 3e-3 weight decay. We train the models for 20000 steps in total, with a cosine learning rate decay schedule that decays to 0. We retain the best checkpoint that is selected by monitoring the validation loss throughout the training process.

E.2 Building an ensemble by combining models with slightly different architectures

The ensemble uses constituent models that slightly modify the design model architecture presented in Figure S.1 - each model uses a different type of MLP layer (shown in red in Figure S.1) in the fine-tuning head. Within the set of constituent models, we vary the depth (2, 4, or 8 layers), number of hidden units (512, 1024, or 2048), and activation functions (tanh, GELU, ReLU, or SiLU [26]) of the MLP layers. This gives us 36 constituent models for ensembling.

E.3 Fine-tuning details

After pretraining, all design models are fine-tuned using a certain split of the PE dataset collected by Reddy et al. [12] (Split 1 in Table S.2). The ensemble's constituent models are fine-tuned using a different split (Splits 2 in Table S.2). In each split, following Reddy et al. [12], we use ~ 70% of the assayed promoters for training, ~ 10% for validation, and ~ 20% for testing. Since there are distinct classes of promoters in the dataset with varying levels of GC content, our splits are stratified by both promoter class and GC content.

When fine-tuning the design models using the conservative regularizer, the full fine-tuning objective from Eqn 3 is optimized for up to 200 steps using the AdamW optimizer [32] (batch size = 512, learning rate = 5e-5, weight decay = 3e-3, $\beta_1 = 0.9$, $\beta_2 = 0.999$) with a cosine learning rate decay schedule that decays to 0 after 10 warm-up steps and model checkpoints with the lowest validation set loss are retained. Since the THP1 data is noisier compared to data from Jurkat and K562 cells (illustrated by Reddy et al. [12] reporting significantly lower concordance between replicate PE measurements), our fine-tuned models have generally lower prediction performance for this cell line. Furthermore, there are few promoters in the training data with high DE and PE in THP1. These problems make it difficult to design sequences for THP1 using sequence optimization – we often could not find many designs with high predicted DE and PE in THP1 using the vanilla DE definition presented in Eqn 1 during fine-tuning and sequence optimization. Therefore, we slightly modified the DE definition for THP1 cells to boost the number of such desirable designs (especially those with high predicted PE in THP1 since this is a requirement for gene therapy applications). This definition is used both during fine-tuning (i.e. in Eqn 3) and sequence optimization:

$$\text{DE}^{\text{THP1}}(\mathbf{x}) = \frac{1}{(|C| - 1)} \sum_{oc \neq \text{THP1}} [1.5 * \text{PE}^{\text{THP1}}(\mathbf{x}) - \text{PE}^{oc}(\mathbf{x})], \quad (5)$$

The constituent models of the ensemble are fine-tuned without the conservative regularizer using the AdamW optimizer with a batch size of 512, 5e-5 learning rate, 3e-3 weight decay, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and with a cosine learning rate decay schedule that decays to 0 after 10 warm-up steps. They are fine-tuned for upto 250 steps, and we retain the best checkpoints that are selected by monitoring the validation loss throughout the training process.

E.4 Prediction performance

The prediction performance of our models on their respective test sets is shown in Table S.2. These results indicate that all of our models are accurate predictors of PE in the target cell types, justifying their usage in designing promoters.

Model	Dataset Split	Jurkat		K562		THP1	
		r	ρ	r	ρ	r	ρ
Design model with $\alpha = 0$	Split 1	0.699	0.666	0.705	0.698	0.644	0.542
Design model with $\alpha = 0.0003$	Split 1	0.700	0.666	0.706	0.698	0.644	0.542
Design model with $\alpha = 0.001$	Split 1	0.700	0.666	0.706	0.698	0.643	0.542
Design model with $\alpha = 0.003$	Split 1	0.701	0.667	0.706	0.698	0.642	0.543
Design model with $\alpha = 0.01$	Split 1	0.700	0.668	0.705	0.698	0.639	0.545
Ensemble model used for final selection	Split 2	0.702	0.659	0.691	0.690	0.627	0.529
Test Set Replicate Concordance	Split 1	0.819	0.735	0.737	0.666	0.730	0.483

Table S.2: Prediction performance obtained using our models.

F Details about baselines

In this section, we describe the baselines in more detail.

F.1 Designing cell type-specific promoters using motif tiling

In this section, we detail how we design cell type-specific promoters using motif tiling. First, to identify motifs that might contribute to DE, we use FIMO [33] with default settings to detect instances of clustered TF-binding motifs defined by Vierstra et al. [34]³ in the sequences assayed by Reddy et al. [12], and retain detected motif occurrences with q-value < 0.01. Let's now consider designing a cell type-specific promoter for Jurkat. For every motif, we first run a Welch's t-test to determine if sequences that contain it have higher expression in Jurkat than sequences that do not contain it, and retain motifs with q-values < 0.01. Then, for every motif, we run 2 pairwise Welch's t-tests to determine if its presence leads to higher expression in Jurkat compared to K562 or THP1. Motifs that have positive effect sizes in both t-tests (i.e. leads to higher expression in Jurkat compared to both K562 and THP1) with q-values < 0.01 are retained as those that could be contributing towards DE in Jurkat. Then, these motifs are used to design two sets of sequences - one set of sequences is designed by inserting the same motif into a background sequence as many times as possible while separating the motifs by 10bp (we get 5 sequences per motif using different background sequences), and the second set is designed by randomly sampling motifs (weighted by average effect size from the two pairwise tests) from the list of retained motifs and inserting as many of them as possible into a background sequence while separating the motifs by 10bp. While generating each sequence, the background sequence is a randomly chosen sequence from those assayed by Reddy et al. [12] that exhibits a DE of at least 2 and also has PE in the target cell that is greater than the 90th percentile of PE. Inserted motif sequences are sampled from the motif's position weight matrix (PWM). The same process is repeated for K562 and THP1.

We discover 4, 26, and 3 motifs that may be causing DE in Jurkat, K562 and THP1 respectively. Thus, we get 20, 130, and 15 sequences in Jurkat, K562, and THP1 respectively by tiling the same motif repeatedly. Then, for each cell line, we design 500 sequences by randomly sampling motifs. Therefore, we get a total of 520, 630, and 515 sequences for Jurkat, K562, and THP1 respectively using this design method.

F.2 Deep Exploration Networks (DENs)

DENs are generative models that are trained to output diverse sequences that maximize a design model's predictions. The generator takes random noise as input and transforms it into a sequence PWM. We use a UNet-style [35] generator that first transforms the noise vector into a sequence PWM-sized matrix (i.e. of size (250, 4)). Then, it applies 6 downsizing convolutional layers followed by 5 upsizing convolutional layers. A final convolutional layer then pools information across the final set of filters' outputs to produce the sequence PWM. The PWM is used to sample sequences that are fed to the design model to get its predictions and a fitness-based loss that trains the DEN to output high-fitness sequences is computed. Additionally, to explicitly increase the diversity of the generated

³<https://resources.altius.org/~jvierstra/projects/motif-clustering-v2.0beta/>

sequences, in every training step, random noise vectors are input to the DEN in pairs to get two sequence PWMs per pair. Then, a diversity-based loss is computed that incentivizes the sequences generated using the two different noise vectors to be distinct from each other, both in sequence and design model embedding space. An entropy-based loss is also minimized to reduce the entropy of the PWM output by the DEN at every position.

Thus, when training a DEN to generate cell type-specific promoters for a target cell $i \in \{\text{Jurkat, K562, THP1}\}$, the training objective we use is:

$$\begin{aligned} \min_{\phi} \quad & \underbrace{\left[\sum_{j=1, s_j \sim g_{\phi}(u_1)}^2 \text{DE}_{\theta}^i(s_j) + \sum_{j=1, q_j \sim g_{\phi}(u_2)}^2 \text{DE}_{\theta}^i(q_j) \right]}_{\text{:= fitness loss}} \\ & + \underbrace{\beta_{\text{diversity}} \max \left[-0.3 + \max_{\sigma \in [0, 10]} \frac{1}{N - \sigma} \left[\frac{1}{2} \sum_{j=1, s_j \sim g_{\phi}(u_1)}^2 \sum_{k=\sigma, q_j \sim g_{\phi}(u_2)}^N s_{j,k} \cdot q_{j,k-\sigma} \right], 0 \right]}_{\text{:= sequence-based diversity loss}} \\ & + \underbrace{\beta_{\text{diversity}} \max \left[-0.3 + \frac{1}{2} \sum_{j=1, s_j \sim g_{\phi}(u_1)}^2 \sum_{q_j \sim g_{\phi}(u_2)} \frac{R_{\theta}(s_j) \cdot R_{\theta}(q_j)}{\|R_{\theta}(s_j)\| \cdot \|R_{\theta}(q_j)\|}, 0 \right]}_{\text{:= embedding-based diversity loss}} \\ & + \underbrace{\beta_{\text{entropy}} \max \left[1.8 - \frac{1}{N} \sum_{k=1}^N \left[\log_2 4 - \sum -g_{\phi}(u_1)_k \log_2 (g_{\phi}(u_1)_k + 10^{-8}) \right], 0 \right]}_{\text{:= entropy loss}} \end{aligned}$$

where ϕ is the set of trainable parameters of DEN g_{ϕ} which outputs $N = 250$ base pairs long sequence PWMs by taking u_1 or u_2 - 200-dimensional random noise vectors sampled from the uniform distribution over $[-1, 1]$, as inputs. From the sequence PWMs output by g_{ϕ} , we sample two one-hot encoded sequences per noise vector denoted by s_j and q_j . These sequences are then input to the trained design model that predicts PE induced in each of the three cell types. Then, the predicted DE in the target cell i induced by a sequence $\mathbf{x}$ is given by Eqn 1. The fitness loss maximizes this predicted DE. The other loss terms increase sequence diversity and reduce entropy in the sequence PWM. Here, $s_{j,k}$ is the one-hot encoded base pair at position k in s_j (similarly for $q_{j,k}$), $R_{\theta}(s_j)$ is an embedding for s_j extracted from the design model f_{θ} , $g_{\phi}(u_1)_k$ is the probability distribution over base pairs in the sequence PWM $g_{\phi}(u_1)$ at position k . Finally, the coefficients $\beta_{\text{diversity}}$ and β_{entropy} are used to weight the diversity and entropy losses relative to the fitness loss and to one another. They can be varied to regulate the diversity vs. fitness trade-off. We refer readers to the original work by Linder et al. [7] that proposed DENs for more details on the method. We tune the various hyperparameters reflected in the training objective by observing the overall quality of the generated sequences.

We train a total of 15 DENs per target cell type, each using a different design model to compute the fitness loss, or making a different diversity vs. fitness trade-off by using different $\beta_{\text{diversity}}$ values. We have 5 different design models, each trained using a different conservatism coefficient α (Table S.2). We also try 3 different $\beta_{\text{diversity}}$ values - 1, 5, and 10, yielding 15 DENs in total. Each DEN is used to generate 20000 sequences and we use the final sequence selection algorithm from Section 4.3 to choose the final set of 2000 sequences per target cell.

G Computational resources used for training

Our workflow is mostly run on a system equipped with 128 CPU cores and 8 Nvidia A100 GPUs with 80GB VRAM each. Pretraining takes around 3.5 hours and it takes around 2 hours to fine-tune a design model and optimize sequences using gradient ascent. We performed a hyperparameter sweep consisting of 4 pretraining runs to tune model architectures. Overall, we estimate that it took around 50 hours on this machine to produce designs using our workflow.

DENs are trained a system equipped with 32 CPU cores and 8 Nvidia A5000 GPUs with 24GB VRAM each. It takes about 3 hours to train each model on a single GPU. Overall, we estimate that it took around 20 hours on this machine to produce designs using DENs.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: We support all claims made in the abstract and introduction using our workflow description and experimental evaluation sections.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: Yes, they are discussed in the last section titled "Discussion and limitations".

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory Assumptions and Proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: We do not present any theoretical results in this paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental Result Reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide high-level details about the methods used to produce our experimental results in Section 5 and provide more details in the appendix with the experimental protocol in Section D, training details in Section E, and details about the baselines in Section F. Our code, data, and instructions to reproduce our main results will be available at https://github.com/young-geng/promoter_design.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [\[Yes\]](#)

Justification: Our code, data, and instructions to reproduce our main results will be available at https://github.com/young-geng/promoter_design.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental Setting/Details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: Details about our workflow are provided in Section 4, the high-level setup for our experiments is described in Section 5, and the appendix has more details about the training and testing in Sections E and F.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment Statistical Significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [\[Yes\]](#)

Justification: Wherever appropriate, we add error bars and details about the significance testing (only used to produce Figure 3).

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments Compute Resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [\[Yes\]](#)

Justification: These details are provided in appendix section G.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code Of Ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [\[Yes\]](#)

Justification: We have reviewed the NeurIPS Code of Ethics and our work conforms to it in every respect.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader Impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [\[Yes\]](#)

Justification: We talk about the potential positive impacts of this work in the abstract and introduction. We do not think our work can have any negative societal impacts.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We do not think our paper poses such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The only existing asset we use is the data released by Reddy et al. [12]. We mention that it was released with their codebase under an AGPL-3.0 license in appendix section C.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [\[Yes\]](#)

Justification: Our code, data, and instructions to reproduce our results will be available at https://github.com/young-geng/promoter_design. We provide details on how we collected the data in appendix section D.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [\[NA\]](#)

Justification: We do not use crowdsourcing or human subjects to collect data.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [\[NA\]](#)

Justification: We do not use crowdsourcing or human subjects to collect data.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.