AttentionPert: Accurately Modeling Multiplexed Genetic Perturbations with Multi-scale Effects

Ding Bai, Caleb Ellington, Shentong Mo, Le Song *, Eric Xing*

February 8, 2024

Abstract

Genetic perturbations (i.e. knockouts, variants) have laid the foundation for our understanding of many diseases, implicating pathogenic mechanisms and indicating therapeutic targets. However, experimental assays are fundamentally limited in the number of perturbation conditions they can measure. Computational methods can fill this gap by predicting perturbation effects under unseen conditions, but accurately predicting the transcriptional responses of cells to unseen perturbations remains a significant challenge. We address this by developing a novel attention-based neural network, AttentionPert, which accurately predicts gene expression under multiplexed perturbations and generalizes to unseen conditions. AttentionPert integrates global and local effects in a multi-scale model, representing both the non-uniform system-wide impact of the genetic perturbation and the localized disturbance in a network of gene-gene similarities, enhancing its ability to predict nuanced transcriptional responses to both single and multi-gene perturbations. In comprehensive experiments, AttentionPert demonstrates superior performance across multiple datasets outperforming the state-of-the-art method in predicting differential gene expressions and revealing novel gene regulations. AttentionPert marks a significant improvement over current methods, particularly in handling the diversity of gene perturbations and in predicting out-of-distribution scenarios.

1 Introduction

Single-cell RNA-sequencing (scRNA-seq) has advanced significantly, enabling the generation of transcriptomic datasets encompassing millions of cell samples [28, 1, 8, 15]. This advancement facilitates CRISPR-based perturbational screens and rapid experimental sampling of genetic perturbation outcomes [7, 22, 11, 24]. However, this field faces significant challenges, particularly when expanding the number of genes for perturbation. The potential combinations of two-gene perturbations grow quadratically with an increasing number of genes [22], leading to a combinatorial explosion in scenarios involving even more simultaneous gene perturbations. Consequently, these methods are constrained by scalability and cost-effectiveness, especially when exploring new genetic perturbations [7, 6, 22].

Many computational simulations model the gene regulatory networks (GRN), rather than directly incorporating the effects of perturbations, ranging from traditional Bayesian networks [12] and causal inference algorithms [32] to more contemporary models such as SENIC [2] and CellOracle [17]. Recent studies [23, 27] have indicated that achieving accurate GRN inference from transcriptomic data poses significant challenges, and GRN-predicting models often fall short in performance when tasked with predicting outcomes of single-gene and multi-gene perturbations. Moreover, other mechanistic modeling methods [13, 35] and linear models [7, 17] addressing both chemical and genetic perturbations, face limitations including scalability issues in predicting responses for a vast number of genes, or an inability to capture non-linear dynamics.

Current advancements in the field use deep neural networks trained on datasets encompassing extensive cells and genes with a moderate number of perturbations, which map genes, perturbations, or genetic relationships into a latent space for the prediction of perturbation outcomes [21, 20, 10, 25, 29, 27, 34]. Most of these methods, including scGEN [21], PerturbNet [34] and CPA [20], primarily predict the transcriptional effects of chemical treatments rather than genetic perturbations. They often approach genetic perturbations as analogous to varying doses, which inadequately capture the complex interactions between different genes and the perturbed ones, resulting in limited performances in this particular domain [20].

*Corresponding authors. Le.Song@mbzuai.ac.ae and Eric.Xing@mbzuai.ac.ae

Recently, there has been a growing trend in employing large language models (LLMs) trained on unlabeled single-cell transcriptomic datasets, which often takes predicting perturbational effects as one of their downstream tasks [14, 16, 5, 33]. While these LLMs represent a relative advancement over other methodologies, they also benefit from downstream methods by incorporating expression representations generated by the foundational LLMs as inputs. Furthermore, the training of these foundation models requires extensive resources, utilizing large-scale datasets and numerous GPUs, which incurs significant time and financial costs.

Among all computational methods for genetic perturbation effects prediction, the state-of-the-art (SOTA) method graph-enhanced gene activation and repression simulator (GEARS) [27], in contrast, specifically targets genetic perturbations. GEARS significantly surpasses other methods in this area, illustrating the effectiveness of a genetic-perturbation-centric preprocessing and training approach for future methodologies [27]. Current single-cell LLMs often take GEARS as a downstream probe in predicting perturbational effects [14, 16, 5, 33]. A key innovation of GEARS is its incorporation of a highly expressive and informative knowledge graph based on Gene Ontology similarities. Nonetheless, GEARS as well as other methods still exhibit the three primary limitations that our work aims to address and advance.

We propose **AttentionPert**, a multi-level deep-learning method to accurately predict transcriptional responses to multiplexed genetic perturbations, which integrates the multi-head attention mechanism [3] with graph neural networks on augmented gene interactions, alongside pre-trained co-expressive gene representations. AttentionPert is proposed to improve three major detriments of current deep-learning works on this problem.

- AttentionPert primarily improves existing methods by capturing uneven effects that the perturbation in a particular gene has on various other genes. The attention-based **PertWeight** encoder perturbs all the gene-representing vectors in high-dimensional latent space with non-uniformly weighted offsets. PertWeight is designed to learn both the overarching background and distinct gene-specific global perturbational effects, reflecting the complex gene-gene interactions within the gene regulatory network.

- AttentionPert accounts for localized gene perturbation analysis using the **PertLocal** encoder, complementing the global effects learned by PertWeight. PertLocal is designed to learn the local disturbance in the representation space of perturbed genes on themselves and their neighbors in a knowledge graph of genetic regulations and interactions. The PertLocal encoder also avoids a pure linear combination of effects of perturbed genes and hence learns the non-additive coeffects of multi-gene perturbations.

- AttentionPert utilizes a pre-trained context gene representation to initialize all the gene embeddings rather than random initialization. This approach significantly improves its capacity to predict responses to unseen perturbed genes.

AttentionPert outperforms prior methods including the SOTA GEARS model on *RPE1* and *K562* datasets [26], each of which comprises more than $1,000$ single-gene perturbations, and on the *Norman* dataset containing 131 two-gene perturbations [22]. Particularly, AttentionPert has exceptional capability in the most challenging out-of-distribution (OOD) task: predicting the effects of novel multi-gene perturbations where none of the perturbed genes have been encountered during the training phase. This remarkable performance highlights its ability to generalize effectively to entirely new genetic scenarios. Our experiments also reveal insights into nuanced genetic interactions associated with multi-gene perturbations through detailed gene-specific and perturbation-specific analyses.

2 Method

2.1 Problem setting

AttentionPert predicts post-perturbation gene expressions given specific gene-perturbation conditions. Formally, for a gene-perturbation condition denoted as $c = \{j_1, j_2, ..., j_m\} \subseteq \{1, ..., K\}$, representing a subset of gene indices, the objective is to predict the post-perturbation gene expression distribution $Y_c \in \mathbb{R}^K$. Within the datasets, each perturbation is associated with multiple cells, of which post-perturbation expressions are represented as $\mathbf{y}_c^i \in \mathbb{R}^K, i = 1, ..., n_c$, where n_c is the number of cells corresponding to the condition c. A perturbation condition is termed *control* when $m = 0$ and the unperturbed expressions are considered as known entities denoted by $\{\mathbf{x}^i, i = 1, ..., n_{\text{control}}\}$. The primary goal for a parametric model parameters Y_θ is to minimize the

difference between the expected $Y_\theta(c)$ and the mean of the ground truth samples $\{\mathbf{y}_c^i\}$ for each perturbation c.

Existing deep neural network models process both the perturbation condition and a randomly sampled unperturbed cell to predict a post-perturbation expression vector [20, 27]. These predicted vectors are evaluated by measuring their similarity to actual perturbed samples. In our method, we adhere to the established preprocessing stage by randomly pairing each control vector $\mathbf{x}_c^i$ from the set $\{\mathbf{x}^t, t = 1, ..., n_{\text{control}}\}$ with its corresponding post-perturbation vector $\mathbf{y}_c^i$, where $i = 1, ..., n_c$ for each condition c. While CPA encodes gene expressions into a high-dimensional space and then perturbs them within this space [20], our approach follows GEARS, focusing on directly predicting the expression shift resulting from the perturbation condition and adding it to the control expression, formally expressed as: $\hat{\mathbf{y}}_\theta(c)^i = \mathbf{g}_\theta(c) + \mathbf{x}_c^i$. We also adopt the combined loss function used by GEARS, which integrates auto-focus and direction-aware loss, to compare our predictions $\hat{\mathbf{y}}_\theta(c)^i$ with the ground-truth $\mathbf{y}_c^i$. We detail the preprocessing and loss function formulas in Supplementary Section A.1. The primary innovation of our model is the improved post-perturbation expression change predictor $\mathbf{g}_\theta$.

2.2 Model overview

AttentionPert takes a perturbation condition $c \subseteq \{1, ..., K\}$ as input. The vector $\mathbf{z}_c \in \{0, 1\}^K$ contains the 0/1 perturbation indicators, formally: $\mathbf{z}_{c,j} = 1$ if $j \in c$ and 0 otherwise. AttentionPert feeds the perturbation indicators into two modules called PertWeight and PertLocal encoders respectively. Each encoder outputs a matrix of encoding, noted as $\mathbf{H}^{\text{PW}}$ by PertWeight and $\mathbf{H}^{\text{PL}}$ by PertLocal, which both $\in \mathbb{R}^{K \times D}$, where D is the network hidden size. After that, a gene-specific decoder takes the sum of two encodings and then generates the predicted post-perturbation expression shift $\mathbf{g}_\theta(c)$, as shown in Figure 1. Finally, we add the shift to a random control sample $\mathbf{x}_c^i$ to get a post-perturbation prediction $\hat{\mathbf{y}}_\theta(c)^i = \mathbf{g}_\theta(c) + \mathbf{x}_c^i$, and compute the loss function for the whole batch to learn the model parameter θ. A more intuitive illustration of this method is Supplementary Section A.2.

2.3 PertWeight for non-uniform global effects

Current works treat all affected genes uniformly when computing the effects of a genetic perturbation in the embedding space, which does not differentiate how a perturbed gene might impact other genes in unique ways [21, 20, 27]. To address this issue, the attention-based perturbation encoder **PertWeight** is designed to learn weight matrices representing the interactions from perturbed genes to other genes. These matrices are then applied in a weighted addition to compute post-perturbation states, thus enabling the model to capture the non-uniform gene-to-gene perturbational effects.

PertWeight firstly embeds gene indices $\{1, 2, ...K\}$ into 2 independent sets of embeddings. We define the first embedding as the *weight* embedding noted by $\mathbf{E}_w$, and the second embedding as the *pert* embedding noted by $\mathbf{E}_p$. Here: $\mathbf{E}_w, \mathbf{E}_p \in \mathbb{R}^{K \times D_e}$. Both embeddings can be either randomly initialized or loaded with the pre-trained gene-representing embeddings. Current studies assign each gene a high-dimensional representation that is randomly initialized [20, 27], which is a suboptimal approach considering the limited range of experimental perturbation conditions available in comparison to the potential perturbations. Utilizing pre-trained context representations for initializing gene embeddings could significantly enhance the model's ability to predict responses to unseen perturbed genes. Our proposed method incorporates Gene2Vec [9], which offers 200-dimensional vector representations for human genes, derived from gene co-expression patterns across 984 datasets in the Gene Expression Omnibus (GEO) databases. It is proven to be more expressive through our experiments than random initializing gene representations.

In this module, we apply 4 convolutional graph neural network (GNN) encoders with the Gene Ontology (GO) graph as the input graph $\mathcal{J}$ with an adjacency matrix $\mathbf{J}$. The GNN model *SGConv* [19] has linear projection weights $\mathbf{T} \in \mathbb{R}^{D_{\text{in}} \times D_{\text{out}}}$. Taking the input graph node embedding $\mathbf{E} \in \mathbb{R}^{K \times D_{\text{in}}}$ and the graph $\mathcal{A}$ of K nodes with an adjacency matrix $\mathbf{A} \in \mathbb{R}^{K \times K}$, a t-hop SGConv outputs a graph-convolution encoding $\text{SGC}_{t,\mathbf{T}}(\mathbf{A}, \mathbf{E}) \in \mathbb{R}^{K \times D_{\text{out}}}$.

A Gene Ontology term provides a structured framework to represent information in a specific domain, where every gene is associated with multiple Gene Ontology terms [4, 31, 30]. The GO graph $\mathcal{J}$ is constructed through Gene Ontology following the SOTA method [27], where the edge weight of two genes is the fraction of their shared Gene Ontology terms. Details for the GO graph are shown in Supplementary Section A.3

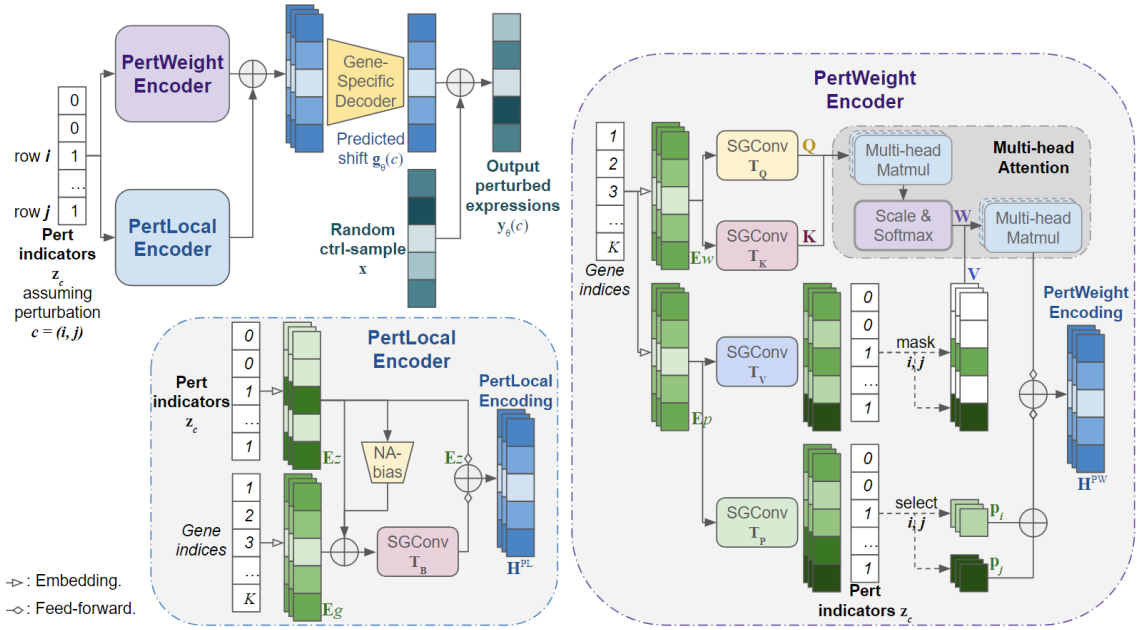


Figure 1: **The overall architecture of AttentionPert.** The illustration assumes that the perturbation condition $c = \{i, j\}$. The indicators vector $\mathbf{z}_c$ is the input of both the PertWeight Encoder and the PertLocal Encoder. PertWeight generates attention weight matrices $\mathbf{W}$ based on the embedding $\mathbf{E}_w$, uses the perturbation indicators to filter the values $\mathbf{V}$ and $\mathbf{P}$, and then combines both the weighted and the general global effects. PertLocal takes the perturbation indicators and outputs the locally disturbed gene states. The gene-specific linear decoder decodes the sum of two encodings into the perturbation effects, and then the predicted expressions are obtained. All SGConvs of PertWeight: $\mathbf{T}_Q, \mathbf{T}_K, \mathbf{T}_V$ and $\mathbf{T}_P$ take the Gene Ontology graph as the input graph, while the SGConv of PertLocal $\mathbf{T}_B$ takes the augmented GO graph. All embeddings for gene indexes are initialized using pre-trained Gene2Vec embeddings and can be learned during training. All the feed-forward layers are multiple linear and batch normalization layers.

The 4 SGConv encoders have different weight matrices: $\mathbf{T}_Q \in \mathbb{R}^{D_e \times Hd_q}, \mathbf{T}_K \in \mathbb{R}^{D_e \times Hd_k}, \mathbf{T}_V \in \mathbb{R}^{D_e \times Hd_v}$ and $\mathbf{T}_P \in \mathbb{R}^{D_e \times D_p}$ respectively, where H is the number of heads in the multi-head attention, and $d_q = d_k$. SGConvs $\mathbf{T}_Q$ and $\mathbf{T}_K$ process the weight embedding $\mathbf{E}_w$ and SGConvs $\mathbf{T}_V$ and $\mathbf{T}_P$ process the pert embedding $\mathbf{E}_p$. Then we got 4 outputs:

$$\begin{cases} \mathbf{Q} &= \mathrm{SGC}_{t,\mathbf{T}_Q}\left(\mathbf{J}, \mathbf{E}_w\right) \in \mathbb{R}^{K \times Hd_q} \\ \mathbf{K} &= \mathrm{SGC}_{t,\mathbf{T}_K}\left(\mathbf{J}, \mathbf{E}_w\right) \in \mathbb{R}^{K \times Hd_k} \\ \mathbf{V} &= \mathrm{SGC}_{t,\mathbf{T}_V}\left(\mathbf{J}, \mathbf{E}_p\right) \cdot \mathbf{z}_c \in \mathbb{R}^{K \times Hd_v} \\ \mathbf{P} &= \mathrm{SGC}_{t,\mathbf{T}_P}\left(\mathbf{J}, \mathbf{E}_p\right) \in \mathbb{R}^{K \times D_p} \end{cases} \tag{1}$$

Note that here the output of the SGConv $\mathbf{T}_V$ is masked using the vector of perturbation indicators $\mathbf{z}_c$, leaving only the values of perturbed genes. $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ are then fed into the multi-head attention as queries, keys and values [3], as shown in Figure 1. Specifically, queries, keys and values are all equally divided into H matrices along the second dimension. Note that for head h, the sub-matrices are $\mathbf{Q}^h$, $\mathbf{K}^h$ and $\mathbf{V}^h$, and the attention matrix of each head is:

$$\mathbf{W}^h = \mathrm{softmax}\left[\frac{Q^h\left(\mathbf{K}^h\right)^T}{\sqrt{d_k}}\right] \tag{2}$$

. And the output of the multi-head attention $\mathbf{H}^{\mathrm{MHA}}$ is the concatenation of all heads $\mathbf{H}^{\mathrm{MHA},h} = \mathbf{W}^h\mathbf{V}^h$. Since $\mathbf{V}$ is masked by the vector of perturbation indicators $\mathbf{z}_c$, the output can also be written as:

$$\mathbf{h}_k^{\mathrm{MHA},h} = \sum_{j \in c} \mathbf{W}_{kj}^h \mathbf{v}_j^h \tag{3}$$

for each gene $k \in \{1, ..., K\}$. This is a weighted summation of perturbational effects from all perturbed genes. All attention matrices $\mathbf{W}^h \in \mathbf{R}^{K \times K}$ are the *Perturbation Weight Matrices*,

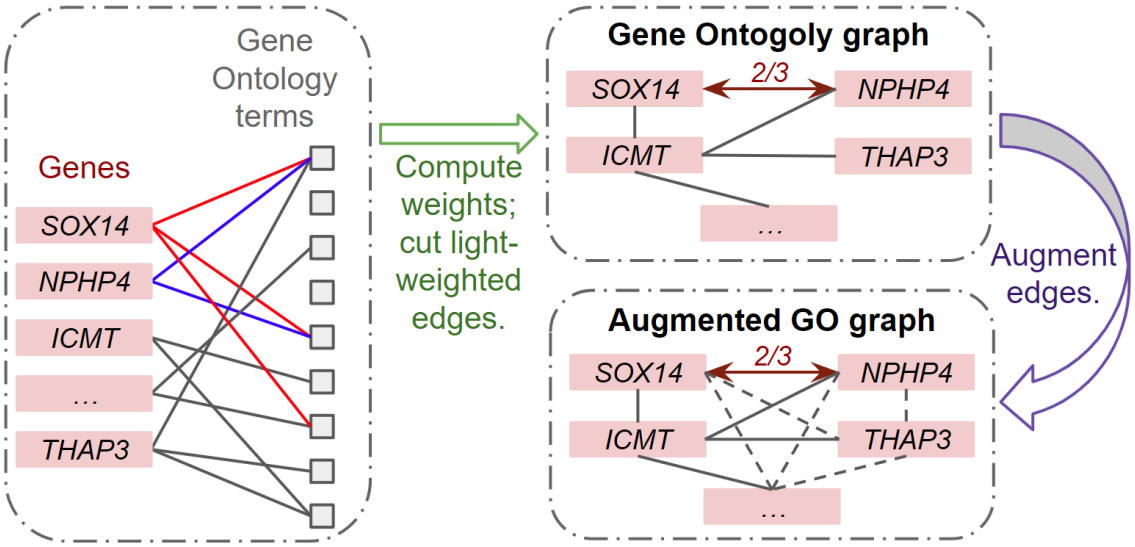


Figure 2: **The process to obtain the Gene Ontology (GO) graph and the augmented GO graph.** Using the GO terms of each pair of genes, for example, those marked in red for gene SOX14 and those marked in blue for NPHP4, the edge weight is defined as the number of shared GO terms divided by the size of the union set of their GO terms, which is 2/3 for genes SOX14 and NPHP4 (only as an illustration in the figure). The GO graph is then obtained by filtering 20 highest edge weights for each gene and cutting all other edges. The augmented GO graph is fully connected, in which each edge either exists in the GO graph or is augmented as α times the minimum edge weight of the GO graph, where $\alpha \in (0, 1]$.

which are the key innovation of PertWeight module featuring the various effects between any perturbed genes and all genes.

The output $\mathbf{P}$ represents the general effects of perturbed genes upon all genes. The row vectors of $\mathbf{P}$ in the perturbed genes $\{\mathbf{p}_j | j \in c\}$ are selected and added to the final output. Hence the final output of the PertWeight Encoder is:

$$\mathbf{h}_k^{\mathrm{PW}} = \mathbf{f}_1(\mathbf{h}_k^{\mathrm{MHA}}) + \mathbf{f}_2(\sum_{j \in c} \mathbf{p}_j) \tag{4}$$

for each gene k, where $\mathbf{f}_1$ and $\mathbf{f}_2$ are linear feed-forward layers that project encodings into the same hidden size D, followed by layer-normalization. The output matrix is $\mathbf{H}^{\mathrm{PW}} = \left[\mathbf{h}_1^{\mathrm{PW}}, ..., \mathbf{h}_K^{\mathrm{PW}}\right]^T \in \mathbb{R}^{K \times D}$.

Obviously, PertWeight is additive for genes in any multi-gene perturbation c, which cannot simulate the non-additive feature, of which details are shown in Supplementary Section A.4. Therefore, we propose PertLocal which learns the nonlinear co-effects of perturbations of multiple genes.

2.4 PertLocal for local disturbance

Many existing models primarily focus on the perturbational effects across all genes within a cell [21, 20], overlooking the different interactions between genes. The SOTA method employs a gene-gene interaction graph, yet it only models global effects [27]. However, genetic regulation studies demonstrate that local and global perturbational effects complement each other. **PertLocal** is designed to overcome this gap by learning the localized disturbance effects of perturbed genes on their closely related counterparts. It complements the global effects captured by the PertWeight encoder, thereby significantly enhancing the overall predictive accuracy of the model.

Similar to PertWeight, PertLocal also starts with 2 embeddings. The first one is the gene base embedding representing unperturbed states of genes that embeds gene indices $\{1, 2, ...K\}$ into dimension D_e, and the second one is the perturbation indicator embedding that embeds binary indicators $\{0, 1\}$ into dimension D_e. The indicator embedding layer embeds the perturbation indicator vector $\mathbf{z}_c \in \{0, 1\}^K$ into a matrix of dimension $K \times D_e$. We denote the gene base embedding as $\mathbf{E}_g \in \mathbb{R}^{K \times D_e}$ and the perturbation indicator embedding as $\mathbf{E}_z \in \mathbb{R}^{K \times D_e}$. Similarly to $\mathbf{E}_w$ and $\mathbf{E}_p$, the gene base embedding $\mathbf{E}_g$ is also initialized with Gene2Vec.

Numerous previous studies have inaccurately employed the additive prediction approach commonly used in multi-gene perturbation modeling [12, 32, 2, 17], which the SOTA work designed an innovative cross-gene layer to avoid [27]. In contrast, PertLocal adopts a non-additive bias layer to effectively learn the complex, non-linear coeffect resulting from multiple perturbed genes. The non-additive (NA) biased embedding for the perturbation indicator embedding is formally:

$$\mathbf{E}'_z = \mathbf{E}_z\big[1 + \beta(m-1)\tanh(\mathbf{K}\mathbf{z}_c)\big] \tag{5}$$

, where $\beta \in (0,1)$ is a hyper-parameter, $m = |c|$ is the number of perturbed genes, and $\mathbf{K} \in \mathbb{R}^{D_e \times K}$ is a trainable weight matrix. PertLocal then applies another t-hop SGConv encoder on the summation of the 2 embeddings $\mathbf{E}_g + \mathbf{E}'_z$, which takes the augmented Gene Ontology graph as the input graph. The augmented GO graph $\mathcal{J}'$ is created by adding virtual edges to all disconnected pairs of nodes in the GO graph with a new minimum edge weight. The process to attain the GO graph and the augmented GO graph is shown in Figure 2. Formally,

$$\mathbf{J}'_{ij} = \begin{cases} \mathbf{J}_{ij}, \mathbf{J}_{ij} > 0 \\ \alpha \min_{k,l}\{\mathbf{J}_{kl} | \mathbf{J}_{kl} > 0\}, \mathbf{J}_{ij} = 0 \end{cases} \tag{6}$$

, where $0 < \alpha <= 1$. Virtually added edges with minimum weights make the graph $\mathcal{J}'$ fully connected, which allows PertLocal to learn lightweight instead of zero global gene-gene reactions. The details of the GO graph and the GNN encoders can be found in Supplementary Section 2.

Denote the weight matrix for the PertLocal SGConv as $\mathbf{T}_B \in \mathbb{R}^{D_e \times D_b}$. After 2 feed-forward and then batch normalization layers $\mathbf{f}_3$ and $\mathbf{f}_4$ of the same output size D, the SGConv output and the perturbation indicator embedding are then added together as the output of the PertLocal Encoder:

$$\mathbf{H}^{\mathrm{PL}} = \mathbf{f}_3\Big(\mathrm{SGC}_{t,\mathbf{T}_B}\big(\mathbf{J}', \mathbf{E}_g + \mathbf{E}'_z\big)\Big) + \mathbf{f}_4\big(\mathbf{E}_z\big) \tag{7}$$

, as shown in Figure 1.

The PertLocal encoder is non-additive since the NA-biased embedding is identical to the indicator embedding when $m = 1$ but not when $m \geq 2$, which then induces the non-additive property (Supplementary Section 1.4 A.4). Through the weighted gene-gene graph structure and the indicator embedding, PertLocal learns the representations of local perturbational effects that complement the PertWeight encoding.

2.5 Decoder for transcriptional effects

After getting the encodings from both encoders, we input the sum of $\mathbf{H}^{\mathrm{PW}}$ and $\mathbf{H}^{\mathrm{PL}}$ into a linear decoder, which simplifies GEARS decoder by removing its cross-gene part. In this decoder, after a feed-forward function for each gene k, a gene-specific linear layer maps the post-perturbation encoding into a scalar that is the predicted perturbation effect. Formally:

$$\mathbf{g}_\theta(c)_k = \mathbf{w}_k^T\Big(\mathbf{f}_5\big(\mathbf{h}_k^{\mathrm{PW}} + \mathbf{h}_k^{\mathrm{PL}}\big)\Big) + b_k \tag{8}$$

, where $\mathbf{w}_k \in \mathbb{R}^D$ and $b_k \in \mathbb{R}$ are learnable weights and bias of the linear projection for each gene. The cross-gene layer is used in GEARS for predicting the non-additive effects of multi-gene perturbations, which is covered in our PertLocal model. Since PertWeight and PertLocal have learned both the uneven global and the local perturbational effects, a relatively less expressive cross-gene layer is abandoned to avoid redundancy. Experiment results demonstrate that AttentionPert achieves superior results, outperforming the need for a cross-gene layer.

The final output is the predicted post-perturbation gene expressions $\hat{\mathbf{y}}_\theta(c)^i$ obtained by adding the predicted post-perturbation expression shift vector $\mathbf{g}_\theta(c)$ to a randomly sampled unperturbed control gene expressions $\mathbf{x}_c^i$, as shown in Figure 1. The combined loss function (inherited from GEARS) is computed by taking the outputs of a mini-batch and then optimized to learn the parameters of $\mathbf{g}_\theta$.

3 Results

3.1 Experimental setup

Evaluation metrics. The evaluation metrics for each perturbation condition c is the mean squared error (MSE) between the mean prediction and the average ground-truth post-perturbation

expression on the top 20 differentially expressed (DE) genes. We also evaluate the Pearson correlation coefficient between the predicted post-perturbation expression shift and the ground-truth shift ρ_Δ. This metric is evaluated throughout all genes to test the model's ability to predict the overall expression change. Detailed descriptions of metrics are shown in Supplementary Section B.

Baselines. We conduct experiments for 3 baselines and compare their performances with AttentionPert. The first one is the model *Ctrl* that directly outputs the mean value of all unperturbed control expressions, which is hence the worst case for any dataset. The performance of *Ctrl* also indicates the difficulty of the post-perturbation prediction since it measures how much effect those perturbations have on unperturbed cells. Due to the differences between datasets and perturbations inside the same dataset, the difficulty of predicting the effect of novel gene perturbations varies much. Hence we also calculate the relative mean squared error (*rel-MSE*) for each model that is the MSE of the given model divided by the MSE of *Ctrl*. The other 2 baselines are CPA [20] and GEARS [27].

Implementation and training. AttentionPert is implemented with the embedding size $D_e = 200$ of all embeddings fit the Gene2Vec. All hyper-parameters of AttentionPert are set as (Supplementary Section C): the hop number of all SGConvs $t = 1$; the output dimension of SGConvs $\mathbf{T}_P$ and $\mathbf{T}_B$ equals to the network hidden size $D_p = D_B = D = 64$; the number of heads $H = 2$ and the head dimension $d_q = d_k = d_v = 64$ in the multi-head attention of the PertWeight encoder; the minimum edge weight of the GO graph is multiplied by $\alpha = 0.75$ when augmenting it to the Augmented GO graph, and the NA-bias parameter $\beta = 0.05$. We train the model in each data split independently for 20 epochs, with a batch size 128 and an Adam optimizer with a learning rate 10^{-3}. We run the training process for all models on 5 splits of 3 datasets that are Norman, RPE1 and K562 [22, 26], of which details are shown in Supplementary Section D. Except for *Ctrl* which is non-parameterized, we conduct each experiment 5 times with randomly different model initialization and take the average and standard deviation of the results.

3.2 Comparison to prior work

The high cost of CRISPR gene-perturbation screens requires an in-silico method that predicts transcriptomic effects of novel single/multi-gene perturbations not in the existing datasets. Therefore in our experiments, all the cells of a given perturbation condition c are either in the train and validation set or in the test set, which simulates the task of predicting unseen perturbations. For single-gene perturbations, we define only one generalization class called *seen 0/1* where the only perturbed gene is never seen in the train and validation set. When considering the combination of multiple perturbed genes, however, there are multiple testing scenarios, depending on the experimental exposure of the genes to perturbations during training.

AttentionPert is adept at predicting transcriptional outcomes for perturbations involving multiple genes, of which performance is evaluated using the Norman dataset [22] that comprised 131 two-gene perturbations. As previously mentioned, two-gene perturbations in the test set are divided into three generalization classes for evaluation purposes. The first class includes scenarios where both genes in a two-gene perturbation set are individually encountered in the training data (*2-gene perturbation, seen 2/2*). The more challenging scenarios involve cases where either one gene (*seen 1/2*) or neither gene (*seen 0/2*) of the two-gene set are perturbed in the training dataset. The comparative analysis of these three scenarios is detailed in Table 1. The results demonstrate that AttentionPert outperforms all baseline models across these scenarios, particularly in the most OOD scenario where both test genes are unseen in the train set.

Table 1: **Seen 0/2, seen 1/2, and seen 2/2 comparison of 3 metrics on Norman dataset.** MSE(DE) is the MSE of the top 20 DE genes. The relative MSE(DE) is in percentage (%) and is relative to the Ctrl model. The ρ_Δ score refers to the Pearson score between the predicted shift and the ground-truth shift. We record the mean and the standard deviation over 5 independent experiments. The best performances are marked in bold.

Model	Seen 0/2			Seen 1/2			Seen 2/2		
	MSE(DE)	rel-MSE(DE)	ρ_Δ(all)	MSE(DE)	rel-MSE(DE)	ρ_Δ(all)	MSE(DE)	rel-MSE(DE)	ρ_Δ(all)
Ctrl	0.503	100	0.013	0.689	100	0.016	0.604	100	−0.016
CPA	0.243 $\pm$0.032	48.3 $\pm$6.3	0.517 $\pm$0.026	0.388 $\pm$0.010	56.3 $\pm$1.4	0.577 $\pm$0.005	0.321 $\pm$0.026	53.2 $\pm$4.3	0.573 $\pm$0.005
GEARS	0.219 $\pm$0.011	43.5 $\pm$2.2	0.558 $\pm$0.016	0.221 $\pm$0.007	32.1 $\pm$0.9	0.595 $\pm$0.015	0.130 $\pm$0.007	21.6 $\pm$1.1	0.657 $\pm$0.027
Ours	**0.159** $\pm$0.012	**31.6** $\pm$2.5	**0.574** $\pm$0.016	**0.193** $\pm$0.003	**28.0** $\pm$0.4	**0.617** $\pm$0.005	**0.097** $\pm$0.008	**16.1** $\pm$1.4	**0.684** $\pm$0.013

Furthermore, Figure 3 shows the specific comparison between AttentionPert and the SOTA method GEARS for each two-gene perturbation out of the 79 tested ones. Among them, Atten-

Table 2: **Seen 0/1 comparison of 3 metrics on RPE1 and K562 datasets.**

Data	Model	MSE(DE)	rel-MSE(DE)	ρ_Δ(all)
RPE1	Ctrl	0.290	100	−0.017
	CPA	0.281 ± 0.001	97.0 ± 0.4	0.109 ± 0.025
	GEARS	0.143 ± 0.005	49.3 ± 1.6	0.527 ± 0.025
	Ours	**0.132** ± 0.005	**45.7** ± 1.7	**0.532** ± 0.010
K562	Ctrl	0.211	100	−0.014
	CPA	0.205 ± 0.000	97.3 ± 0.1	0.088 ± 0.015
	GEARS	0.137 ± 0.003	65.1 ± 1.2	0.327 ± 0.015
	Ours	**0.132** ± 0.003	**62.4** ± 1.3	**0.340** ± 0.012

tionPert outperforms GEARS on all 9 perturbations of the hardest seen 0/2 scenario. Over 3 scenarios, AttentionPert has better performances on 63 perturbations, which is about 80% of the test cases. This reveals that AttentionPert surpasses the SOTA method in predicting the effects of two-gene perturbations.

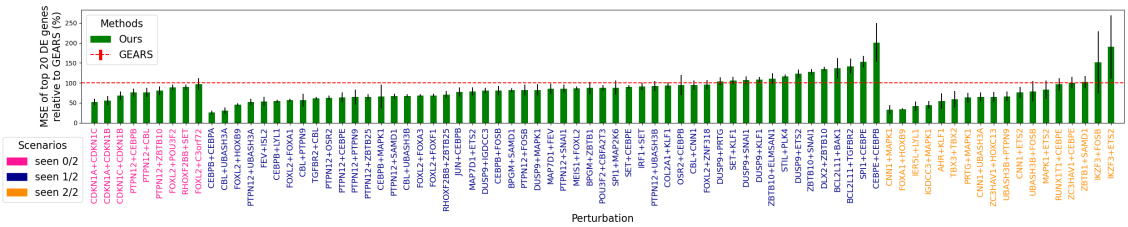


Figure 3: **Perturbation-specific comparison on Norman dataset.** The MSE(DE) values of AttentionPert are normalized as percentages relative to the performance of GEARS.

We apply two metrics evaluating the average error rate when dealing with different genes apart from MSE(DE) and ρ_Δ(all) which represent overall performances across the set of genes. The first metric quantifies the proportion of the top 20 DE genes for a given perturbation predicted to change in an opposite direction compared to their actual change. The second metric is the percentage of DE genes for which the predicted post-perturbation gene expression falls outside one standard deviation of the actual post-perturbation expression distribution mean. Both metrics describe the ratio of wrongly predicted genes, of which lower values are preferable. Figure 4 shows the comparison between baseline models and AttentionPert on these two scores, demonstrating that our model makes fewer errors in predicting both the direction and value of perturbation effects for the top 20 DE genes.

For single-gene perturbations, we employed datasets from two separate genetic perturbation screens: one involving RPE1 cells with 1,543 perturbations and another with K562 cells, comprising 1,092 perturbations. Each dataset contained data from over 160,000 cells [26]. The performance of AttentionPert is compared against baseline models, as detailed in Table 2. The results demonstrate that AttentionPert outperforms all baselines in all metrics.

Results in Supplementary Section E.1 show that AttentionPert beats all the baselines on all other 4 splits of each dataset, which underscores the robustness and adaptability of AttentionPert under diverse test conditions.

3.3 Discovering Genetic Interactions

Non-Gaussianity in residual errors reveals outlier complexity in data not captured by the model. Here, we analyze AttentionPert's residual errors to diagnose any failure cases in the model, while also revealing nuanced genetic interactions, outlier complexity in the regulation of some genes, and gene modules with correlated errors (Figure 5). AttentionPert achieves near-zero error on

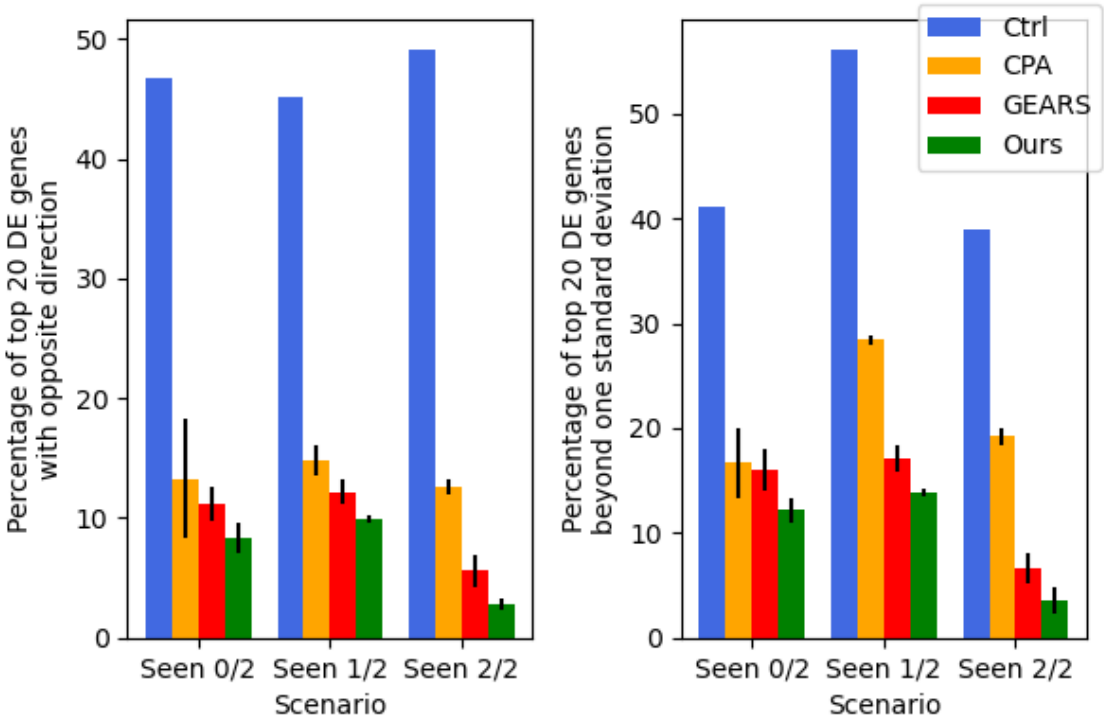


Figure 4: **Comparison of metrics representing error rates on Norman dataset.** These two metrics evaluate the average proportion of genes with incorrect direction and value predictions, respectively.

genes GMFG, ACP5, POU3F2, and NGRN, regardless of perturbation condition, while the genes RUNX1T1, MS4A3, LGI2, and CTSL display high error across all conditions, indicating nuanced regulatory behavior unrelated to gene perturbations. Surprisingly, perturbations to transcription factor families such as the CEB* family and FOX* family have much lower residual error across all genes, suggesting that these are degenerate cases where the complexity of the transcriptome decreases in their absence. In contrast, the two perturbation cases with BCL2L11 – a signaling protein involved in apoptosis regulation – have much higher residuals than average, indicating an induced incoherent system or cellular decay.

Notably, the residuals are uncorrelated with the perturbation domain splits (Figure 5), and uncorrelated with the structure of the Gene Ontology graph (Supplementary Figure 11). Complementing AttentionPert's SOTA performance on all domain generalization tasks (Table 1), this gives us high confidence that AttentionPert generalizes extremely well, and that high error genes and perturbations are indicative of complex and nuanced genetic mechanisms.

Furthermore, non-additive effects arising from the complex interactions between multiple perturbed genes underscore the necessity for a model that can accurately and integrally handle multi-gene perturbations. Figure 6 reveals the ability of AttentionPert to capture the non-additive nature of multiplexed genetic perturbations. When predicting a seen 2/2 perturbation: FOXA1 and HOXB9, AttentionPert is adept at learning the combined response of these genes. This approach differs markedly from a simple additive model, which merely sums the transcriptional effects of each gene perturbed independently.

3.4 Ablation study

The primary distinction between AttentionPert and the SOTA GEARS lies in the expression shift function, $\mathbf{g}_\theta$. Besides the model architecture, AttentionPert differs from GEARS in three aspects: first, we discard the cross-gene layer utilized by GEARS; second, we match the number of potentially perturbed genes to the size of the genes in the dataset, while GEARS includes a different set of potential perturbed genes, adding numerous nodes to the GO graph that are not perturbed in actual datasets; thirdly, we utilize pre-trained Gene2Vec embeddings for initializing all gene embeddings in AttentionPert, while GEARS employs random initialization. The ablation study

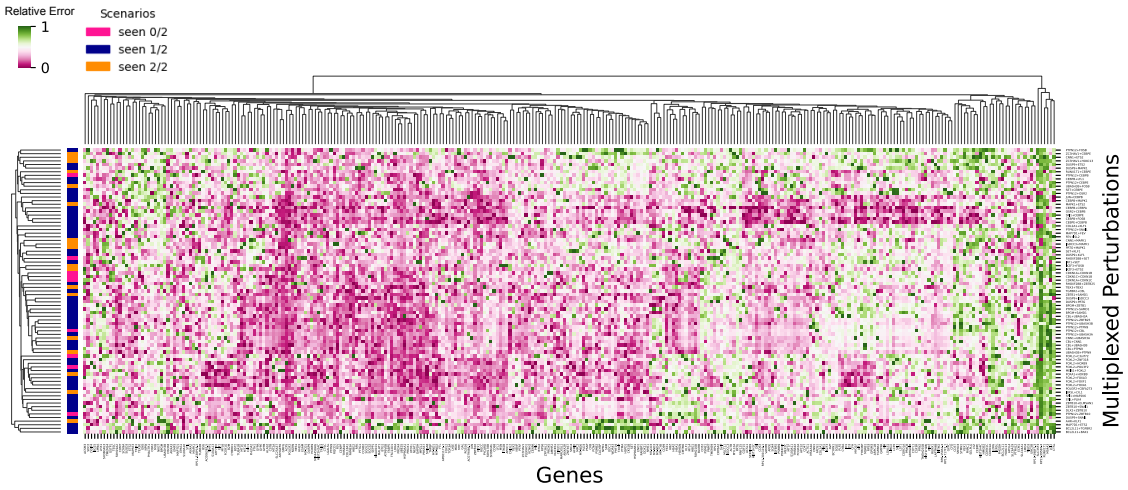


Figure 5: **Gene-specific residual errors of AttentionPert.** The results are shown over tested multiplexed perturbation conditions in the Norman dataset and the union set of the top 20 DE genes of these perturbations. Scenarios of out-of-domain generalization are plotted on the left. Residual errors are log-transformed and column-scaled.

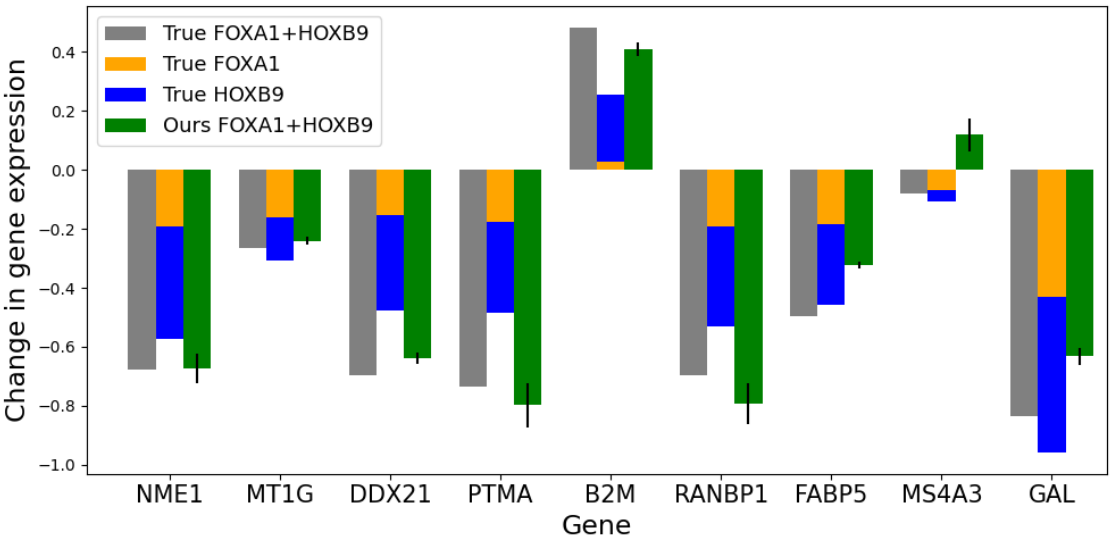


Figure 6: **Change in gene expression after perturbing the combination FOXA1 and HOXB9**. The gray bars indicate the true mean post-perturbation gene expression change. The orange and blue bars show the sum of true shifts for the two individual single-gene perturbations. The green bars show the prediction made by AttentionPert.

presented in Table 3 demonstrates that AttentionPert achieves the lowest MSE(DE) in seen 0/2 scenario, which indicates that the enhanced performance of AttentionPert is not solely attributable to any one of these three changes or their combination. Therefore, the design of AttentionPert is essential for its improved performance.

In our study, we conduct ablation experiments on the two encoders: PertWeight and PertLocal. The results presented in Table 4 reveal that while each encoder contributes to the model's performance, neither encoder individually outperforms the effectiveness of the integrated model that combines both encoders. This underscores the synergistic effect of incorporating both the PertWeight and PertLocal encoders into AttentionPert.

Besides, we conduct experiments where AttentionPert does not use Gene2Vec. Results in Table 4 and Table 3 show that it beats GEARS without using Gene2Vec but has worse results than the final version of AttentionPert where we use Gene2Vec to initialize gene embeddings.

Table 3: **Ablation experimental results comparing other differences from GEARS.** *CG* means utilizing the cross-gene decoding layer. *Rd* means reducing the potentially perturbed genes. *G2V* is initializing the gene embeddings with Gene2Vec. The original GEARS method utilizes cross-gene without reducing genes or using Gene2Vec.

Model	CG	Rd	G2V	Seen 0/2 MSE(DE)	Seen 1/2 MSE(DE)	Seen 2/2 MSE(DE)
GEARS	✓			0.219 $\pm$0.011	0.221 $\pm$0.007	0.130 $\pm$0.007
GEARS	✓		✓	0.293 $\pm$0.010	0.449 $\pm$0.009	0.391 $\pm$0.006
GEARS	✓	✓		0.235 $\pm$0.006	0.234 $\pm$0.008	0.117 $\pm$0.004
GEARS	✓	✓	✓	0.194 $\pm$0.012	0.225 $\pm$0.007	0.110 $\pm$0.003
GEARS		✓		0.246 $\pm$0.022	0.215 $\pm$0.011	**0.087** $\pm$0.003
GEARS		✓	✓	0.208 $\pm$0.015	0.211 $\pm$0.005	0.094 $\pm$0.004
Ours		✓	✓	**0.159** $\pm$0.012	**0.193** $\pm$0.003	0.097 $\pm$0.008

Table 4: **Ablation experimental results on AttentionPert.** We compare our model with/without each of the three components: PertLocal (*PL*), PertWeight (*PW*), and Gene2Vec (*G2V*) initialization.

Model	PL	PW	G2V	Seen 0/2 MSE(DE)	Seen 1/2 MSE(DE)	Seen 2/2 MSE(DE)
only-PL	✓		✓	0.246 $\pm$0.002	0.388 $\pm$0.004	0.324 $\pm$0.002
only-PW		✓	✓	0.172 $\pm$0.009	0.202 $\pm$0.004	0.120 $\pm$0.007
Ours	✓	✓		0.189 $\pm$0.026	0.203 $\pm$0.009	0.105 $\pm$0.018
Ours	✓	✓	✓	**0.159** $\pm$0.012	**0.193** $\pm$0.003	**0.097** $\pm$0.008

Another interesting finding is that the original version of GEARS using Gene2Vec performs worse, while the reduced version of GEARS using Gene2Vec archives a better performance as shown in Table 3. This is probably because the number of possible perturbed genes used by the original GEARS is much larger than the actual number of genes in the Norman dataset. A pre-trained gene context representation undermines its ability when dealing with a smaller set of genes.

Ablation study results for other splits are detailed in Supplementary Section E.2.

4 Conclusion

We have proposed a comprehensive model AttentionPert that effectively integrates attention mechanisms to predict the unique effects of single or multiple perturbed genes. By incorporating the PertWeight and PertLocal encoders that complement each other with encodings representing global and local perturbational effects respectively, we have successfully addressed the challenges of predicting transcriptional responses to genetic perturbations. AttentionPert not only outperforms existing state-of-the-art methods in terms of accuracy and handling diverse gene perturbations but also shows exceptional capability in predicting out-of-distribution scenarios where all simultaneously perturbed genes are unseen. The demonstrated superior performance across multiple datasets highlights its potential in facilitating medical research and treatment development. Besides, detailed gene- and perturbation-specific experimental analyses provide valuable insights into the complex genetic interactions inherent in multi-gene perturbations, which clarify the intricate dynamics of gene interplay under various perturbation conditions. We anticipate further exploration and application of the computational method to predict cellular responses of novel genetic perturbations in personalized medicine and genetic research.

References

[1] A single-cell transcriptomic atlas characterizes ageing tissues in the mouse. *Nature*, 583(7817):590–595, 2020.

[2] Sara Aibar, Carmen Bravo González-Blas, Thomas Moerman, Vân Anh Huynh-Thu, Hana Imrichova, Gert Hulselmans, Florian Rambow, Jean-Christophe Marine, Pierre Geurts, Jan Aerts, et al. Scenic: single-cell regulatory network inference and clustering. *Nature methods*, 14(11):1083–1086, 2017.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Seth Carbon, Amelia Ireland, Christopher J Mungall, ShengQiang Shu, Brad Marshall, Suzanna Lewis, AmiGO Hub, and Web Presence Working Group. Amigo: online access to ontology and annotation data. *Bioinformatics*, 25(2):288–289, 2009.

[5] Haotian Cui, Chloe Wang, Hassaan Maan, Kuan Pang, Fengning Luo, and Bo Wang. scgpt: Towards building a foundation model for single-cell multi-omics using generative ai. *bioRxiv*, pages 2023–04, 2023.

[6] Paul Datlinger, André F Rendeiro, Christian Schmidl, Thomas Krausgruber, Peter Traxler, Johanna Klughammer, Linda C Schuster, Amelie Kuchler, Donat Alpar, and Christoph Bock. Pooled crispr screening with single-cell transcriptome readout. *Nature methods*, 14(3):297–301, 2017.

[7] Atray Dixit, Oren Parnas, Biyu Li, Jenny Chen, Charles P Fulco, Livnat Jerby-Arnon, Nemanja D Marjanovic, Danielle Dionne, Tyler Burks, Raktima Raychowdhury, et al. Perturb-seq: dissecting molecular circuits with scalable single-cell rna profiling of pooled genetic screens. *cell*, 167(7):1853–1866, 2016.

[8] Silvia Domcke, Andrew J Hill, Riza M Daza, Junyue Cao, Diana R O'Day, Hannah A Pliner, Kimberly A Aldinger, Dmitry Pokholok, Fan Zhang, Jennifer H Milbank, et al. A human cell atlas of fetal chromatin accessibility. *Science*, 370(6518):eaba7612, 2020.

[9] Jingcheng Du, Peilin Jia, Yulin Dai, Cui Tao, Zhongming Zhao, and Degui Zhi. Gene2vec: distributed representation of genes based on co-expression. *BMC genomics*, 20:7–15, 2019.

[10] Gökcen Eraslan, Lukas M Simon, Maria Mircea, Nikola S Mueller, and Fabian J Theis. Single-cell rna-seq denoising using a deep count autoencoder. *Nature communications*, 10(1):390, 2019.

[11] Chris J Frangieh, Johannes C Melms, Pratiksha I Thakore, Kathryn R Geiger-Schuller, Patricia Ho, Adrienne M Luoma, Brian Cleary, Livnat Jerby-Arnon, Shruti Malu, Michael S Cuoco, et al. Multimodal pooled perturb-cite-seq screens in patient models define mechanisms of cancer immune evasion. *Nature genetics*, 53(3):332–341, 2021.

[12] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using bayesian networks to analyze expression data. In *Proceedings of the fourth annual international conference on Computational molecular biology*, pages 127–135, 2000.

[13] F Frohlich, T Kessler, D Weindl, A Shadrin, L Schmiester, H Hache, et al. Efficient parameter estimation enables the prediction of drug response using a mechanistic pan-cancer pathway model. cell syst. 2018; 7 (6): 567–79 e6.

[14] Jing Gong, Minsheng Hao, Xingyi Cheng, Xin Zeng, Chiming Liu, Jianzhu Ma, Xuegong Zhang, Taifeng Wang, and Le Song. xtrimogene: An efficient and scalable representation learner for single-cell rna-seq data. *arXiv preprint arXiv:2311.15156*, 2023.

[15] Xiaoping Han, Ziming Zhou, Lijiang Fei, Huiyu Sun, Renying Wang, Yao Chen, Haide Chen, Jingjing Wang, Huanna Tang, Wenhao Ge, et al. Construction of a human cell landscape at single-cell level. *Nature*, 581(7808):303–309, 2020.

[16] Minsheng Hao, Jing Gong, Xin Zeng, Chiming Liu, Yucheng Guo, Xingyi Cheng, Taifeng Wang, Jianzhu Ma, Le Song, and Xuegong Zhang. Large scale foundation model on single-cell transcriptomics. *bioRxiv*, pages 2023–05, 2023.

[17] Kenji Kamimoto, Blerta Stringa, Christy M Hoffmann, Kunal Jindal, Lilianna Solnica-Krezel, and Samantha A Morris. Dissecting cell identity via network inference and in silico gene perturbation. *Nature*, 614(7949):742–751, 2023.

[18] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

[19] Yuhong Li, Tianle Cai, Yi Zhang, Deming Chen, and Debadeepta Dey. What makes convolutional models great on long sequence modeling? *arXiv preprint arXiv:2210.09298*, 2022.

[20] Mohammad Lotfollahi, Anna Klimovskaia Susmelj, Carlo De Donno, Leon Hetzel, Yuge Ji, Ignacio L Ibarra, Sanjay R Srivatsan, Mohsen Naghipourfar, Riza M Daza, Beth Martin, et al. Predicting cellular responses to complex perturbations in high-throughput screens. *Molecular Systems Biology*, page e11517, 2023.

[21] Mohammad Lotfollahi, F Alexander Wolf, and Fabian J Theis. scgen predicts single-cell perturbation responses. *Nature methods*, 16(8):715–721, 2019.

[22] Thomas M Norman, Max A Horlbeck, Joseph M Replogle, Alex Y Ge, Albert Xu, Marco Jost, Luke A Gilbert, and Jonathan S Weissman. Exploring genetic interaction manifolds constructed from rich single-cell phenotypes. *Science*, 365(6455):786–793, 2019.

[23] Aditya Pratapa, Amogh P Jalihal, Jeffrey N Law, Aditya Bharadwaj, and TM Murali. Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data. *Nature methods*, 17(2):147–154, 2020.

[24] Laralynne Przybyla and Luke A Gilbert. A new era in functional genomics screens. *Nature Reviews Genetics*, 23(2):89–103, 2022.

[25] Ladislav Rampášek, Daniel Hidru, Petr Smirnov, Benjamin Haibe-Kains, and Anna Goldenberg. Dr. vae: improving drug response prediction via modeling of drug perturbation effects. *Bioinformatics*, 35(19):3743–3751, 2019.

[26] Joseph M Replogle, Reuben A Saunders, Angela N Pogson, Jeffrey A Hussmann, Alexander Lenail, Alina Guna, Lauren Mascibroda, Eric J Wagner, Karen Adelman, Gila Lithwick-Yanai, et al. Mapping information-rich genotype-phenotype landscapes with genome-scale perturb-seq. *Cell*, 185(14):2559–2575, 2022.

[27] Yusuf Roohani, Kexin Huang, and Jure Leskovec. Predicting transcriptional outcomes of novel multigene perturbations with gears. *Nature Biotechnology*, pages 1–9, 2023.

[28] Orit Rozenblatt-Rosen, Michael JT Stubbington, Aviv Regev, and Sarah A Teichmann. The human cell atlas: from vision to reality. *Nature*, 550(7677):451–453, 2017.

[29] Nikolai Russkikh, Denis Antonets, Dmitry Shtokalo, Alexander Makarov, Yuri Vyatkin, Alexey Zakharov, and Evgeny Terentyev. Style transfer with variational autoencoders is a promising approach to rna-seq data harmonization and analysis. *Bioinformatics*, 36(20):5076–5085, 2020.

[30] Paul D Thomas, Dustin Ebert, Anushya Muruganujan, Tremayne Mushayahama, Laurent-Philippe Albou, and Huaiyu Mi. Panther: Making genome-scale phylogenetics accessible to all. *Protein Science*, 31(1):8–22, 2022.

[31] Paul D Thomas, David P Hill, Huaiyu Mi, David Osumi-Sutherland, Kimberly Van Auken, Seth Carbon, James P Balhoff, Laurent-Philippe Albou, Benjamin Good, Pascale Gaudet, et al. Gene ontology causal activity modeling (go-cam) moves beyond go annotations to structured descriptions of biological functions and systems. *Nature genetics*, 51(10):1429–1433, 2019.

[32] Yuhao Wang, Liam Solus, Karren Yang, and Caroline Uhler. Permutation-based causal inference algorithms with interventions. *Advances in Neural Information Processing Systems*, 30, 2017.

[33] Xiaodong Yang, Guole Liu, Guihai Feng, Dechao Bu, Pengfei Wang, Jie Jiang, Shubai Chen, Qinmeng Yang, Yiyang Zhang, Zhenpeng Man, et al. Genecompass: Deciphering universal gene regulatory mechanisms with knowledge-informed cross-species foundation model. *bioRxiv*, pages 2023–09, 2023.

[34] Hengshi Yu and Joshua D Welch. Perturbnet predicts single-cell responses to unseen chemical and genetic perturbations. *BioRxiv*, pages 2022–07, 2022.

[35] Bo Yuan, Ciyue Shen, Augustin Luna, Anil Korkut, Debora S Marks, John Ingraham, and Chris Sander. Cellbox: interpretable machine learning for perturbation biology with application to the design of cancer combination therapy. *Cell systems*, 12(2):128–140, 2021.

A Method Details

A.1 Data preprocessing and loss functions

We use the three perturbational effects datasets [22, 26] preprocessed by GEARS [27] to evaluate our model. The preprocess of GEARS first normalizes the expression values, discard low-varying genes, then samples an unperturbed gene expression $\mathbf{x}_c^i$ from the set $\{\mathbf{x}^t, t = 1, ..., n_{\text{control}}\}$ to be 1-to-1 correspondent with each post-perturbation expression vector $\mathbf{y}_c^i$ where $i = 1, ..., n_c$. For the Gene Ontology (GO), we use the GO datasets provided by the SOTA method and also the Gene Ontology knowledgebase [4, 31, 30]. After aligning the genes, we have successfully found more GO terms for those genes not shown in the GO graph provided by GEARS.

We also utilize Gene2Vec [9] which are 200-dimension vector representations for human genes. We use a Gaussian distribution to sample the vector for genes shown in each of the datasets that cannot be aligned with the gene symbols provided by Gene2Vec. Formally, assuming the gene vector is $\mathbf{v}_g$ where $g \in \mathcal{G}$ is theset of gene symbols of Gene2Vec, for any unknown gene $x \notin \mathcal{G}$, its representation is sampled as:

$$\begin{cases} V_x \sim \mathcal{N}(\mu, \mathbf{\Sigma}) \\ \mu = \mathrm{E}\big[V_g\big] = \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \mathbf{v}_g \\ \mathbf{\Sigma} = \mathrm{Cov}\big(V_g\big) = \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} (\mathbf{v}_g - \mu)(\mathbf{v}_g - \mu)^T \end{cases} \tag{9}$$

We follow the loss functions used by GEARS [27]. During the training, GEARS uses a loss function that measures the distance between each predicted expression $\hat{\mathbf{y}}_\theta(c)^i = \mathbf{g}_\theta(c) + \mathbf{x}_c^i$ and the ground-truth $\mathbf{y}_c^i$, which is:

$$\mathcal{L}^{\text{autofocus}}(\mathbf{y}_c^i) = \frac{1}{|Z_c|} \sum_{k \in Z_c} \big(\hat{\mathbf{y}}_\theta(c)_k^i - \mathbf{y}_{c,k}^i\big)^{2+\gamma} \tag{10}$$

$$\mathcal{L}^{\text{autofocus}}(\mathcal{B}) = \frac{1}{|\mathcal{B}|} \sum_{\mathbf{y}_c^i \in \mathcal{B}} \mathcal{L}^{\text{autofocus}}(\mathbf{y}_c^i) \tag{11}$$

, where $\mathcal{B}$ is a batch of post-perturbation expressions in the training process, Z_c is the set of genes with non-zero expressions in at least 1 cell under the perturbation condition c, and γ is a hyper-parameter which $= 2$ in GEARS experiment setup.

GEARS also combines a direction-aware loss with a parameter $\lambda = 0.1$ with the autofocus loss:

$$\mathcal{L}^{\text{direction}}(\mathbf{y}_c^i) = \frac{1}{|Z_c|} \sum_{k \in Z_c} \Big[\text{sign}\big(\hat{\mathbf{y}}_\theta(c)_k^i - \mathbf{x}_{c,k}^i\big) - \text{sign}\big(\mathbf{y}_{c,k}^i - \mathbf{x}_{c,k}^i\big)\Big]^2 \tag{12}$$

$$\mathcal{L}^{\text{direction}}(\mathcal{B}) = \frac{1}{|\mathcal{B}|} \sum_{\mathbf{y}_c^i \in \mathcal{B}} \mathcal{L}^{\text{direction}}(\mathbf{y}_c^i) \tag{13}$$

$$\mathcal{L}^{\text{train}}(\mathcal{B}) = \mathcal{L}^{\text{autofocus}}(\mathcal{B}) + \lambda \mathcal{L}^{\text{direction}}(\mathcal{B}) \tag{14}$$

.

In addition, the log variance in GEARS is taught to act as a proxy for model uncertainty by promoting its increase when errors are significant. This is achieved through a gene-specific layer that predicts the log variance for each gene, which is incorporated into the learning process via a modified Bayesian neural network loss, as detailed in [18]. To estimate the post-perturbation gene expression values, a Gaussian likelihood approach is employed. The uncertainty score serves as an indicator of the confidence level of the model when predicting outcomes for new perturbations.

A.2 An intuitive geometric illustration of our method

Our model, AttentionPert, operates on the fundamental assumption that each gene's unperturbed state can be represented as a vector within a D-dimensional hidden space. When certain genes undergo perturbation, we map these perturbation conditions to this high-dimensional space. This mapping alters the basal states of the genes into perturbed states, which are then decoded into scalar values representing the changes in post-perturbation expressions.

AttentionPert comprises two principal encoders: PertLocal and PertWeight. PertLocal is responsible for locally perturbing the basal states of genes. In contrast, PertWeight is designed to calculate both general and gene-specific effects on a more global scale. The terms *locality* and *globality* in this context are defined based on a gene-gene relationship graph. For our model, this graph is the Gene Ontology (GO) graph, which may be used in its standard form or in an augmented version. This entire sequence of events in the hidden space is illustrated in Fig 7.

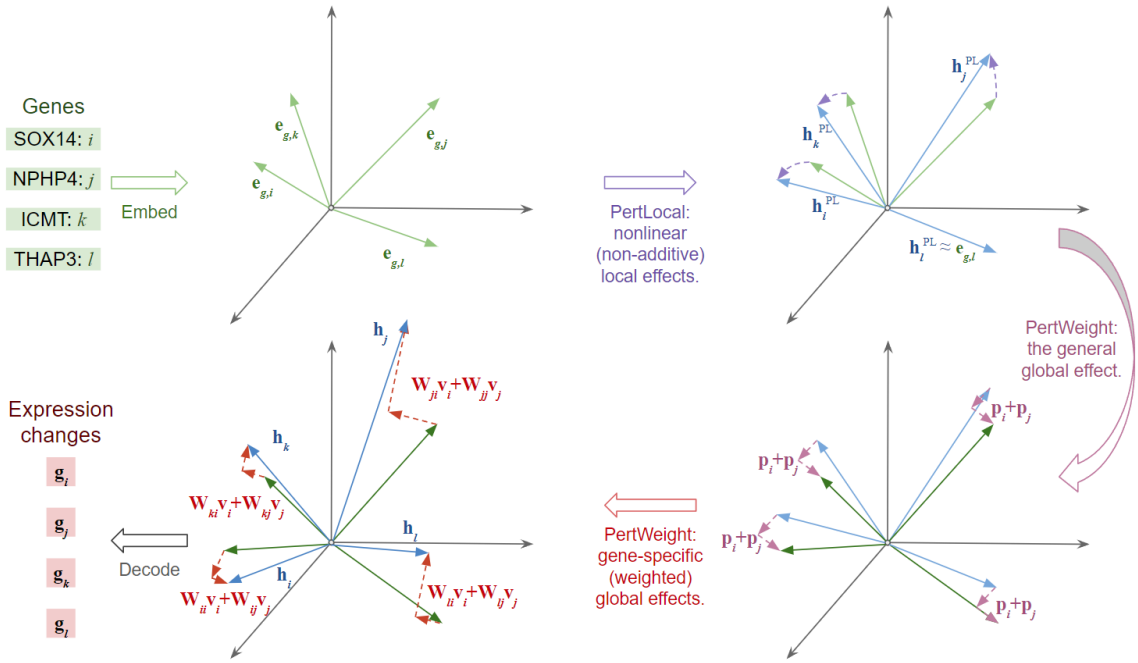


Figure 7: **The geometric illustration of AttentionPert.** First, genes i, j, k, l are embedded into a high-dimensional hidden space as vectors $\mathbf{e}_{g,i}, \mathbf{e}_{g,j}, \mathbf{e}_{g,k}, \mathbf{e}_{g,l}$. Assume that genes i and j are perturbed and are both local to k but distant from l in the GO graph. Then PertLocal locally (with non-additivity) perturbs each gene-representing vector into $\mathbf{H}^{\mathrm{PL}}$. After that, PertWeight adds the general global effect $\mathbf{p}_i + \mathbf{p}_j$ to all vectors and finally adds a gene-specific global effect to each vector, resulting the encoding $\mathbf{H} = \mathbf{H}^{\mathrm{PL}} + \mathbf{H}^{\mathrm{PW}}$. The encoding for each gene is subsequently decoded into a scalar representing the expression change.

A.3 Graphs and GNNs

The GO graph is constructed by incorporating the shared GO terms of each pair gene [27], and those genes in the dataset that have no GO terms or have no intersected ones with other genes are still added to the graph as outlier nodes. Formally, for each pair of genes i and $j \in \{1, 2, ..., K\}$, denote the sets of corresponding Gene Ontology terms to be $\mathcal{N}_i$ and $\mathcal{N}_j$. Then the edge weight between these 2 genes is defined as $\mathbf{J}_{ij} = |\mathcal{N}_i \cap \mathcal{N}_j|/|\mathcal{N}_i \cup \mathcal{N}_j|$, which is the fraction of shared Gene Ontology terms between the 2 genes. After that, for each gene i, the $H_{\mathrm{pert}} = 20$ gene j's with the highest $\mathbf{J}_{i,j}$ are selected to construct the Gene Ontology graph, and this selection gives each edge a direction since the top-weighted neighboring is not bidirectional. Hence the graph $\mathcal{J}$ is an edge-weighted directed graph that uses the Gene Ontology to construct gene-gene interactions, as shown in Figure 2.

Unlike GEARS, we also reduce the GO graph by not including possible perturbed genes that are not actually perturbed in the datasets. Instead of using all possible perturbed genes, we include all the genes in each of the three evaluation datasets to create the GO graph. In conclusion, the GO graph used by GEARS is the same graph of all possible perturbed genes for all three datasets, while our reduced GO graph is unique for each dataset that only contains all the genes from it. Figure 10 shows that the GEARS using the reduced version of the GO graph usually makes better results than its original version.

We use SGConv [19] as the four GNN encoders in our model. A t-hop SGConv with a learnable weight matrix $\mathbf{T} \in \mathbb{R}^{D_{\mathrm{in}} \times D_{\mathrm{out}}}$ outputs:

$$\mathrm{SGC}_{t,\mathbf{T}}(\mathbf{A}, \mathbf{E}) = \left(\hat{\mathbf{D}}_{\mathbf{A}}^{-1/2} \hat{\mathbf{A}} \hat{\mathbf{D}}_{\mathbf{A}}^{-1/2}\right)^t \mathbf{E}\mathbf{T} \tag{15}$$

, where $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ denotes the adjacency matrix with inserted self-loops and $\hat{\mathbf{D}}_{\mathbf{A},ii} = \sum_{j=1}^{K} \hat{\mathbf{A}}_{ij}$ its diagonal degree matrix. The output is in $\mathbb{R}^{K \times D_{\mathrm{out}}}$.

A.4 Non-additive features

As a perturbational effects prediction method, AttentionPert has theoretical non-additive features. Non-additivity requires the predicted $g_\theta(c)_k$ not always equal to $\sum_{i\in c} g_\theta(i)_k$ for any multi-gene perturbation c, and the non-linear coeffect of multiple perturbed genes can be learned.

The PertWeight encoder generates an additive encoding of any multi-gene condition c, formally,

$$\mathbf{H}^{\mathrm{PW}}(c) - \mathbf{H}^{\mathrm{PW}}(\emptyset) \equiv \sum_{i\in c} \Big(\mathbf{H}^{\mathrm{PW}}(\{i\}) - \mathbf{H}^{\mathrm{PW}}(\emptyset)\Big) \tag{16}$$

. Note that here $\mathbf{H}^{\mathrm{PW}}(\emptyset) \equiv 0$. However, PertLocal encoder utilizes the non-additive bias layer to avoid a linear combination of perturbed genes,

$$\mathbf{E}'_z = \mathbf{E}_z\big[1 + \beta(m-1)\tanh(\mathbf{K}\mathbf{z}_c)\big] \tag{17}$$

, where gene indices $i, j \in \{1, ..., K\}$. Since the function tanh is non-linear and m is the number of perturbed genes:

$$\mathbf{E}_z(\{i,j\}) \equiv \mathbf{E}_z(\{i\}) + \mathbf{E}_z(\{j\}) - \mathbf{E}_z(\emptyset) \tag{18}$$

$$\mathbf{E}'_z(\{i,j\}) \not\equiv \mathbf{E}'_z(\{i\}) + \mathbf{E}'_z(\{j\}) - \mathbf{E}'_z(\emptyset) \tag{19}$$

Therefore the PertLocal encoding is non-additive across all perturbed genes in any given condition c:

$$\mathbf{H}^{\mathrm{PL}}(c) - \mathbf{H}^{\mathrm{PL}}(\emptyset) \not\equiv \sum_{i\in c} \Big(\mathbf{H}^{\mathrm{PL}}(\{i\}) - \mathbf{H}^{\mathrm{PL}}(\emptyset)\Big) \tag{20}$$

. This implies that the final output is not additive for multi-gene perturbations either.

B Evaluation Metric Details

The primary metric for evaluating the performance under each perturbation condition c is the Mean Squared Error (MSE). This metric measures the deviation between the expected prediction, approximated as the average predicted expressions $\mathrm{E}\big[Y_\theta(c)\big]$, and the actual ground-truth post-perturbation expression $\mathrm{E}\big[Y_c\big]$ focusing on the top 20 differentially expressed (DE) genes, denoted as $\mathcal{D}_c \subset 1, ..., K$ where $|\mathcal{D}_c| = 20$. The expected expression for both predicted and actual samples is calculated as the mean value across all cells under the given perturbation condition c. Formally:

$$\mathrm{E}\big[Y_c\big] \approx \bar{\mathbf{y}}_c = \frac{\sum_{i=1}^{n_c} \mathbf{y}_c^i}{n_c} \tag{21}$$

$$\mathrm{E}\big[Y_\theta(c)\big] \approx \bar{\mathbf{y}}_\theta(c) = \frac{\sum_{i=1}^{n_c} \mathbf{y}_\theta^i(c)}{n_c} \tag{22}$$

$$\mathrm{MSE(DE)}^\theta(c) = \frac{1}{20}\sum_{k\in\mathcal{D}_c} \Big(\mathrm{E}\big[Y_\theta(c)\big]_k - \mathrm{E}\big[Y_c\big]_k\Big)^2 \tag{23}$$

$$= \frac{1}{20}\sum_{k\in\mathcal{D}_c} \Big(\bar{\mathbf{y}}_\theta(c)_k - \bar{\mathbf{y}}_{c,k}\Big)^2 \tag{24}$$

Due to the substantial cost associated with CRISPR gene-perturbation screens, there is a pressing need for an in-silico method capable of predicting transcriptomic outcomes of novel single or multi-gene perturbations absent from existing datasets. In our experimental design, cells under a specific perturbation condition c are entirely allocated to either the training and validation set or the test set. This allocation strategy effectively simulates the challenge of predicting perturbations that the model has not previously encountered. For any given dataset, we define two distinct sets of perturbation conditions: $\mathcal{C}_{\mathrm{tv}}$, representing the training and validation set, and $\mathcal{C}_{\mathrm{test}}$, representing the test set. As all unperturbed cells are considered known and are therefore included in the training and validation set, the control condition is defined as $c_{\mathrm{control}} = \emptyset$ and is part of $\mathcal{C}_{\mathrm{tv}}$. This approach ensures a clear demarcation between training/validation and testing scenarios, crucial for evaluating the model's efficacy in handling novel perturbations.

When the number of perturbed genes is only 1, there is only one testing scenario that is called *seen 0/1*, because any perturbation condition containing only 1 gene which either shows in the train and validation set or shows in the test set. We denote such a scenario as $\mathcal{S}_{0/1} = \{c \in \mathcal{C}_{\mathrm{test}} | |c| = 1\}$.

When considering the combination of multiple perturbed genes, however, there could be multiple testing scenarios, depending on the experimental exposure of the genes to perturbations during training. For s-gene perturbations test where $s > 1$ (in real-life datasets we only have $s = 2$), there could be $s + 1$ scenarios, which is $\mathcal{S}_{t/s} = \{c \in \mathcal{C}_{\text{test}} | |c| = s, |c \cap \bigcup_{c' \in \mathcal{C}_{\text{tv}}} c'| = t\}$ where $t = 0, 1, ...s$ meaning the number of genes in this perturbation condition that has shown in the train and validation sets. By that definition, over a s-perturbation test, a scenario with smaller t is a more OOD case and harder for the model to predict. The top 20 DE MSE metric of a test scenario $\mathcal{S}_{t/s}$ can be formulated as:

$$\text{MSE(DE)}^{\theta}(\mathcal{S}_{t/s}) = \frac{1}{|\mathcal{S}_{t/s}|} \sum_{c \in \mathcal{S}_{t/s}} \text{MSE}^{\theta}(c) \tag{25}$$

The relative MSE (rel-MSE) in percentage, which is the proportion relative to the Ctrl model, is calculated by:

$$\text{rel-MSE(DE)}^{\theta}(\mathcal{S}_{t/s}) = \frac{\text{MSE(DE)}^{\theta}(\mathcal{S}_{t/s})}{\text{MSE(DE)}^{\text{Ctrl}}(\mathcal{S}_{t/s})} \times 100\% \tag{26}$$

, and by definition of the Ctrl model that only outputs the average control sample:

$$\text{rel-MSE(DE)}^{\theta}(\mathcal{S}_{t/s}) = \frac{\sum_{c \in \mathcal{S}_{t/s}} \sum_{k \in \mathcal{D}_c} \left(\bar{\mathbf{y}}_{\theta}(c)_k - \bar{\mathbf{y}}_{c,k}\right)^2}{\sum_{c \in \mathcal{S}_{t/s}} \sum_{k \in \mathcal{D}_c} \left(\bar{\mathbf{x}}_k - \bar{\mathbf{y}}_{c,k}\right)^2} \times 100\% \tag{27}$$

In addition to MSE, we assess our model's performance using the Pearson correlation coefficient, which measures the relationship between the predicted and actual post-perturbation expression shifts. Formally:

$$\rho_{\Delta}(\text{all})^{\theta}(c) = \rho\Big(\text{E}\big[Y_{\theta}(c)\big] - \text{E}\big[X\big], \text{E}\big[Y_c\big] - \text{E}\big[X\big]\Big) \tag{28}$$

$$= \rho\Big(\bar{\mathbf{y}}_{\theta}(c) - \bar{\mathbf{x}}, \bar{\mathbf{y}}_c - \bar{\mathbf{x}}\Big) \tag{29}$$

, where $\rho(\cdot, \cdot)$ is the Pearson correlation coefficient of 2 vectors. The metric for a set of perturbations is also the average $\rho_{\Delta}(\text{all})^{\theta}$ over all perturbations in the given set. This coefficient is calculated across all genes to gauge the model's proficiency in predicting overall expression changes.

Beyond MSE and the Pearson score, we employ two additional metrics for a more comprehensive evaluation of predictive error rates of top 20 DE genes. The proportion of top 20 DE genes for a given perturbation predicted to change in an opposite direction of the ground truth is formally:

$$P_{\text{opposite direction}}(\text{DE})^{\theta}(c) = \frac{1}{20} \sum_{k \in \mathcal{D}_c} \mathbb{I}\Big(\text{sign}\big(\bar{\mathbf{y}}_{\theta}(c)^i_k - \bar{\mathbf{x}}^i_{c,k}\big) \neq \text{sign}\big(\bar{\mathbf{y}}^i_{c,k} - \bar{\mathbf{x}}^i_{c,k}\big)\Big) \tag{30}$$

. The proportion of DE genes for which the predicted post-perturbation gene expression falls beyond one standard deviation of the actual post-perturbation expression distribution mean is formally:

$$P_{\text{beyond one std}}(\text{DE})^{\theta}(c) = \frac{1}{20} \sum_{k \in \mathcal{D}_c} \mathbb{I}\Big(|\bar{\mathbf{y}}_{\theta}(c)^i_k - \bar{\mathbf{y}}^i_{c,k}| > \sigma_{c,k}\Big) \tag{31}$$

. Here $\mathbb{I}(\cdot)$ is the indicator function which equals 1 if the variable is true and 0 otherwise. The standard deviation of perturbation c on gene k: $\sigma_{c,k}$ is over the set of ground truth expressions : $\{\mathbf{y}^i_{c,k} | i = 1, ..., n_c\}$. Both metrics are averaged as the score of a set of perturbations as well.

C Hyperparameter Search

We tune and fix the best set of hyperparameters on the validation set of the first split of the Norman dataset [22]. We search hyperparameters in ranges: the number of hops of SGConvs $t \in \{1, 2\}$; the network hidden size $D \in \{32, 64, 128\}$; for the multi-head attention, the number of heads $H \in \{1, 2, 4, 8\}$ and the dimension of each head $d_q \in \{8, 16, 32, 64, 128, 256\}$, where $H \times d_q \in [64, 256]$; the minimum edge weight parameter in the augmented GO graph $\alpha \in \{0.25, 0.5, 0.75, 1.0\}$; the NA-bias parameter $\beta \in \{0.001, 0.005, 0.01, 0.05\}$; batch size in $\{32, 64, 128, 256, 512\}$. The best

Table 5: **Details of the perturbational effects datasets.** Here # means the number of something. The Norman dataset is a two-gene perturbation dataset [22]. The RPE1 and K562 datasets are single-gene perturbation datasets [26]. All three datasets are pre-processed by GEARS [27].

Dataset	# 1-gene perturbations	# 2-gene perturbations	# Cells	# 1-gene perturbed cells	# 2-gene perturbed cells	# Genes	Gene Ontology coverage (%)	Gen cover
Norman	105	131	91,205	48,407	35,445	5,045	63.2%	68
RPE1	1,543	0	162,733	151,248	0	5,892	96.0%	97
K562	1,092	0	162,751	152,060	0	5,680	94.6%	96

Table 6: **Different splits of Norman, RPE1 and K562 datasets.** Here # means the number of something. For each dataset, the gene ratio of the train-validation set is 75%. The Norman dataset is a two-gene perturbation dataset [22] that contains 3 scenarios in the test set. The RPE1 and K562 datasets are single-gene perturbation datasets [26] with only seen 0/1 scenario. GEARS pre-processes all three datasets [27].

Dataset	Split	# Perturbed genes	# Perturbed genes in the train-val set	# Seen 0/1 perturbations	# Seen 0/2 perturbations	# Seen 1/2 perturbations	# Seen 2/2 perturbations
Norman	1	105	78	27	9	52	18
	2				7	52	18
	3				2	39	23
	4				6	44	21
	5				10	58	16
RPE1	all	1,543	1,157	386	0	0	0
K562	all	1,092	819	273	0	0	0

hyperparameters are listed in the main body, where $t = 1, D = 64, H = 2, d_q = 64, \alpha = 0.75, \beta = 0.05$. To ensure a fair comparison with the GEARS model, we optimized the batch size, selecting 128 as the most effective value for both models. Additionally, we search the number of top-weighted edges retained in the Gene Ontology (GO) graph. The results indicated similar performance across a range of values $10, 15, 20, 25$, leading us to select 20, aligning with the configuration used in GEARS. Our model is versatile in its representation of prior knowledge regarding gene-gene relationships. While it can utilize various types of networks, such as protein-protein interactions or gene coessentiality networks, we choose to use the GO graph, as detailed in Supplementary Note 2 by GEARS [27]. This decision is driven by the desire to maximize gene coverage and maintain consistency for a fair comparison with GEARS.

D Datasets details

Table 5 shows the details of each dataset, including the numbers of different perturbations, corresponding cell samples, and genes in each dataset covered in the Gene Ontology database or Gene2Vec. During the experiments, each dataset has 5 different splits for training, validation, and testing. We use the same split ratio of genes that the SOTA method conducts its experiments, where for both single-gene and multi-gene cases, the number of perturbed genes in the train and validation set is 75% of the total number of perturbed genes of the dataset. This makes 5 different sets of numbers of 3 scenarios in the 2-gene perturbational dataset [22] but the same number of the only 1 scenario for 1-gene datasets [26], shown in Table 6.

E Additional Experiment Results

E.1 Detailed results for comparison on all 5 splits of 3 datasets

The difficulty across different splits of the same dataset can vary considerably since we do not randomly split each dataset across all cells. Instead, we ensure that all perturbations in the test set are not encountered during the training phase, leading to significant variability in model performance across different experimental splits. Therefore, rather than averaging evaluation metrics

Table 7: **Seen 0/2, seen 1/2, and seen 2/2 comparison of MSE, relative MSE (%) and ρ_Δ on other 4 splits of Norman dataset.** MSE stands for the MSE of the top 20 DE genes. The relative mean squared error (rel-MSE) is in percentage (%). The ρ_Δ score refers to the Pearson score between the predicted shift and the ground-truth shift on all genes. We record the mean and the standard deviation over 5 independent experiments. The best performances of each split are marked in bold.

	Model	Seen 0/2			Seen 1/2			Seen 2/2		
		MSE(DE)	rel-MSE(DE)	ρ_Δ(all)	MSE(DE)	rel-MSE(DE)	ρ_Δ(all)	MSE(DE)	rel-MSE(DE)	ρ_Δ(all)
Split 2	Ctrl	0.798	100	−0.020	0.689	100	0.007	0.575	100	0.007
	CPA	0.546 ±0.004	68.4 ±0.5	0.576 ±0.009	0.380 ±0.008	55.1 ±1.2	0.584 ±0.011	0.191 ±0.020	33.1 ±3.5	0.637 ±0.005
	GEARS	0.403 ±0.032	50.4 ±4.0	0.591 ±0.023	0.241 ±0.011	35.0 ±1.7	0.621 ±0.010	0.101 ±0.004	17.6 ±0.6	0.643 ±0.014
	Ours	**0.329** ±0.021	**41.2** ±2.6	**0.592** ±0.011	**0.220** ±0.010	**31.9** ±1.5	**0.623** ±0.015	**0.088** ±0.007	**15.4** ±1.2	**0.679** ±0.007
Split 3	Ctrl	0.951	100	0.010	0.656	100	0.003	0.762	100	0.012
	CPA	0.542 ±0.067	57.0 ±7.1	0.569 ±0.070	0.300 ±0.031	45.8 ±4.7	**0.590** ±0.011	0.287 ±0.018	37.6 ±2.4	0.639 ±0.017
	GEARS	0.532 ±0.076	56.0 ±8.0	**0.593** ±0.047	0.208 ±0.018	31.7 ±2.7	0.585 ±0.016	0.146 ±0.009	19.2 ±1.1	0.651 ±0.018
	Ours	**0.532** ±0.100	**55.9** ±10.5	0.532 ±0.058	**0.198** ±0.014	**30.1** ±2.2	0.584 ±0.021	**0.137** ±0.012	**18.0** ±1.5	**0.651** ±0.024
Split 4	Ctrl	0.630	100	−5.609	0.607	100	0.008	0.727	100	0.007
	CPA	0.385 ±0.018	61.2 ±2.8	**0.538** ±0.014	0.285 ±0.000	47.0 ±0.1	**0.634** ±0.007	0.203 ±0.012	27.9 ±1.6	0.663 ±0.002
	GEARS	0.375 ±0.040	59.5 ±6.4	0.461 ±0.035	0.220 ±0.018	36.2 ±3.0	0.583 ±0.022	0.139 ±0.024	19.1 ±3.4	0.621 ±0.034
	Ours	**0.275** ±0.037	**43.7** ±5.9	0.516 ±0.016	**0.193** ±0.010	**31.8** ±1.7	0.622 ±0.004	**0.099** ±0.002	**13.6** ±0.3	**0.697** ±0.008
Split 5	Ctrl	0.637	100	0.006	0.670	100	0.013	0.722	100	0.018
	CPA	0.449 ±0.006	70.4 ±1.0	0.374 ±0.003	0.407 ±0.013	60.8 ±1.9	0.470 ±0.028	0.389 ±0.047	53.9 ±6.5	0.507 ±0.047
	GEARS	0.371 ±0.007	58.2 ±1.1	0.449 ±0.025	0.321 ±0.011	47.9 ±1.7	0.530 ±0.042	0.235 ±0.019	32.5 ±2.6	0.581 ±0.050
	Ours	**0.338** ±0.010	**53.0** ±1.6	**0.484** ±0.010	**0.281** ±0.004	**41.8** ±0.5	**0.578** ±0.011	**0.205** ±0.009	**28.4** ±1.2	**0.623** ±0.004

Table 8: **Seen 0/1 MSE, relative MSE (%) and ρ_Δ results on other 4 splits of RPE1 and K562 datasets.** The best performances of each split are marked in bold.

	Model	Data: RPE1			Data: K562		
		MSE(DE)	rel-MSE(DE)	ρ_Δ(all)	MSE(DE)	rel-MSE(DE)	ρ_Δ(all)
Split 2	Ctrl	0.282	100	−0.017	0.235	100	−0.015
	CPA	0.272 ±0.001	96.3 ±0.2	0.162 ±0.006	0.228 ±0.002	97.1 ±0.7	0.088 ±0.023
	GEARS	0.150 ±0.003	53.1 ±1.1	**0.512** ±0.011	0.145 ±0.002	61.9 ±0.7	0.358 ±0.013
	Ours	**0.147** ±0.006	**52.0** ±2.2	0.501 ±0.030	**0.130** ±0.004	**55.3** ±1.6	**0.376** ±0.010
Split 3	Ctrl	0.302	100	−0.017	0.249	100	−0.014
	CPA	0.296 ±0.002	97.8 ±0.6	0.118 ±0.025	0.245 ±0.001	98.4 ±0.5	0.081 ±0.011
	GEARS	0.169 ±0.009	56.1 ±3.1	0.482 ±0.031	0.161 ±0.004	64.4 ±1.6	0.345 ±0.012
	Ours	**0.156** ±0.008	**51.7** ±2.7	**0.505** ±0.015	**0.143** ±0.002	**57.5** ±0.8	**0.367** ±0.014
Split 4	Ctrl	0.312	100	−0.016	0.219	100	−0.014
	CPA	0.305 ±0.000	97.6 ±0.1	0.112 ±0.001	0.213 ±0.001	97.5 ±0.6	0.074 ±0.009
	GEARS	0.163 ±0.008	52.2 ±2.7	0.517 ±0.012	0.135 ±0.002	61.6 ±0.9	0.345 ±0.009
	Ours	**0.152** ±0.003	**48.6** ±1.1	**0.527** ±0.016	**0.120** ±0.003	**54.7** ±1.5	**0.378** ±0.012
Split 5	Ctrl	0.289	100	−0.016	0.225	100	−0.014
	CPA	0.279 ±0.002	96.6 ±0.5	0.141 ±0.019	0.221 ±0.001	98.0 ±0.4	0.082 ±0.018
	GEARS	0.153 ±0.006	53.1 ±2.2	0.506 ±0.012	0.143 ±0.002	63.4 ±0.9	0.345 ±0.008
	Ours	**0.148** ±0.005	**51.3** ±1.7	**0.509** ±0.016	**0.134** ±0.006	**59.5** ±2.5	**0.364** ±0.016

across all splits, we present a split-by-split comparison of the three datasets. Table 7 shows the detailed MSE and ρ_Δ results on the other 4 splits of the Norman dataset (results for the first split are shown in the main body). Besides, percentages of DE genes with an opposite direction and beyond one standard deviation results of all 5 splits of the Norman dataset are shown in Fig 8. Results show that AttentionPert outperforms other methods in most of the metrics on every split when dealing with multi-gene perturbations.

The detailed MSE, rel-MSE and ρ_Δ results on the other 4 splits of RPE1 and K562 datasets (results for the first split are shown in the main body) are shown in Table 8, while error-rate representing results of all 5 splits of these two datasets are shown in Fig 9. Based on the experimental performances, AttentionPert also beats other methods in most of the metrics on every split for single-gene perturbation datasets.

E.2 Detailed results of the ablation study

The rel-MSE results for seen 0/2, seen 1/2 and seen 2/2 scenarios of the ablation experiments are shown in Figure 10. We can see that AttentionPert does not beat all other revised GEARS models in seen 1/2 and seen 2/2 metrics. However, the results for seen 0/2 demonstrate that for the most OOD case where either of the perturbed genes is unseen when training, our model outperforms

the revised version of GEARS and also any other methods. Therefore we still conclude that our model generally performs better than any revised version of GEARS.

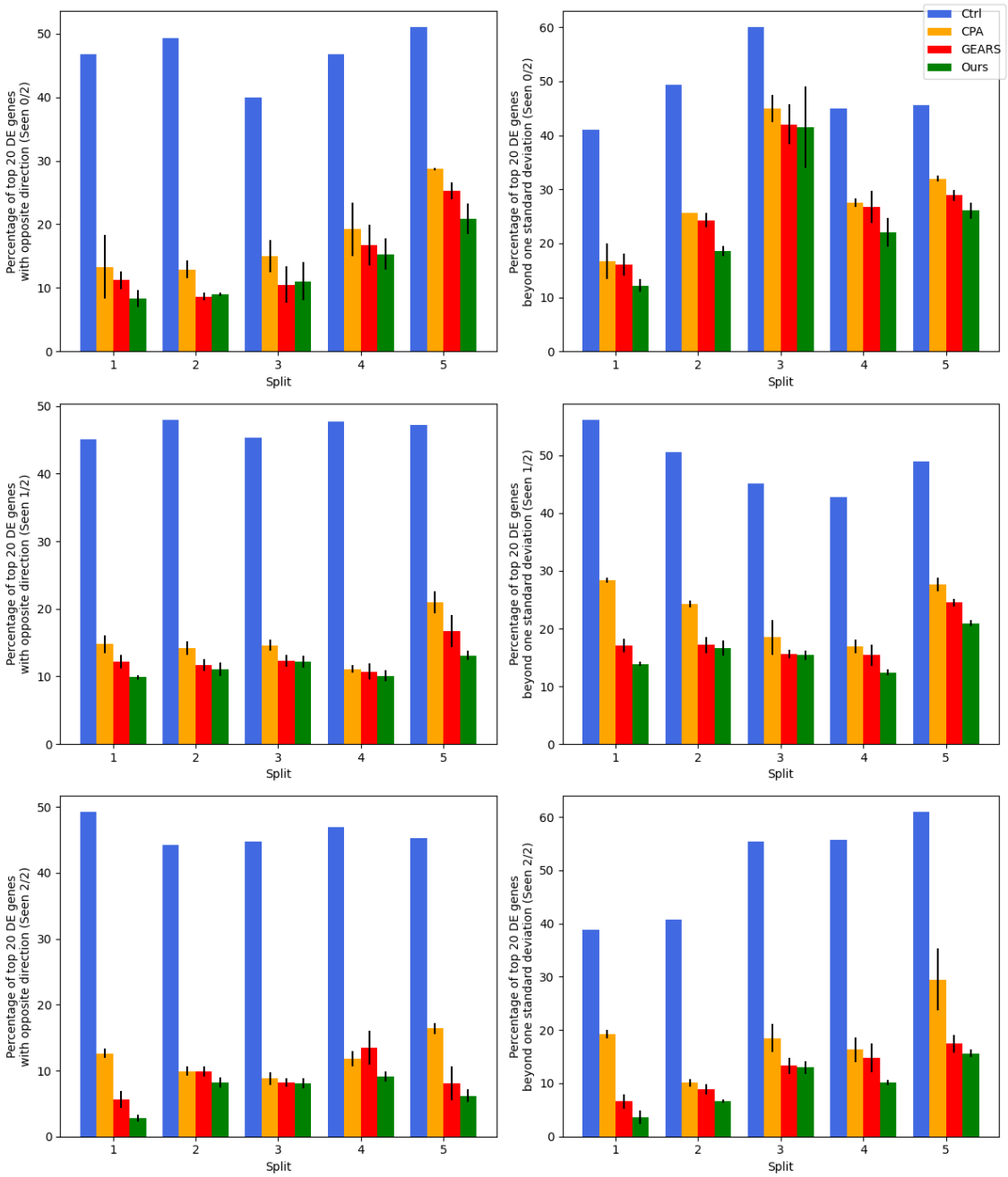
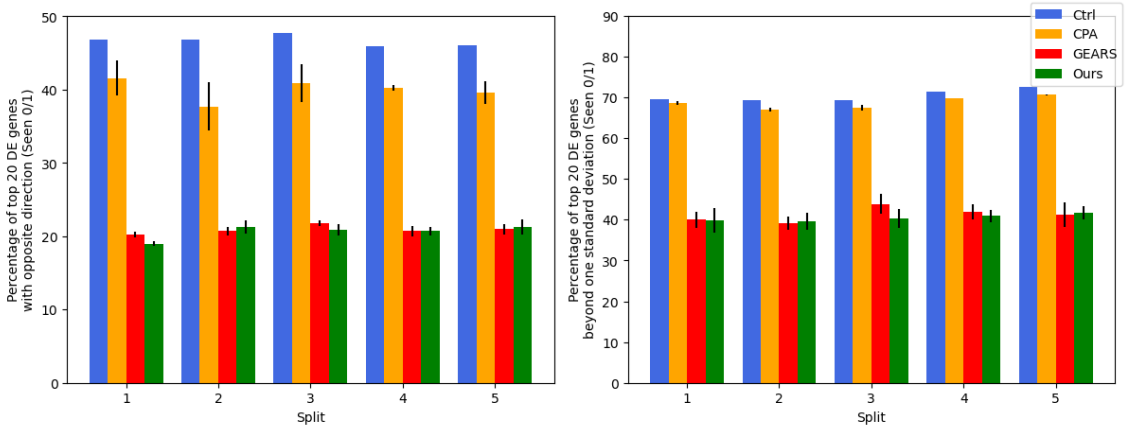
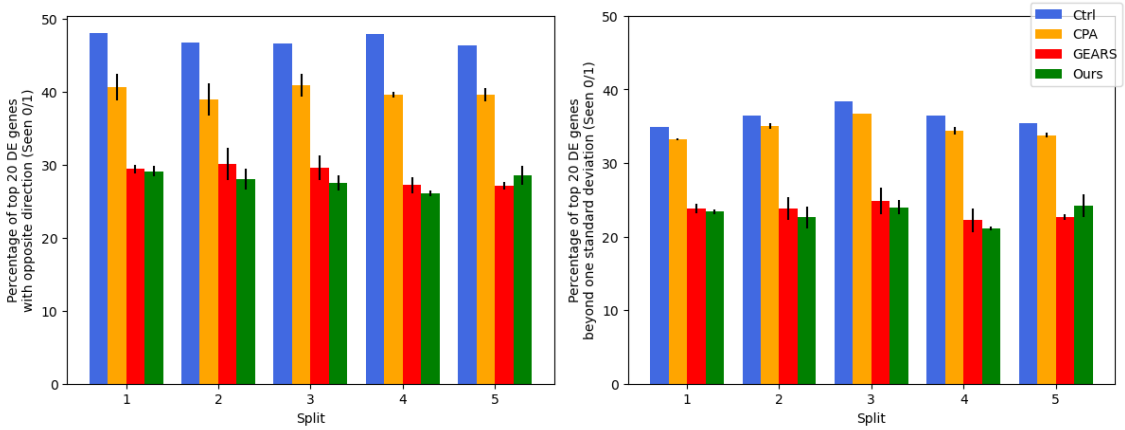


Figure 8: **Comparison of two error-rate representing metrics on Norman dataset.**

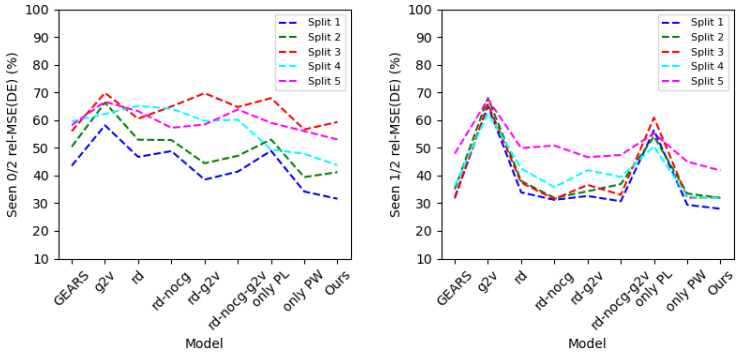


(a) RPE1



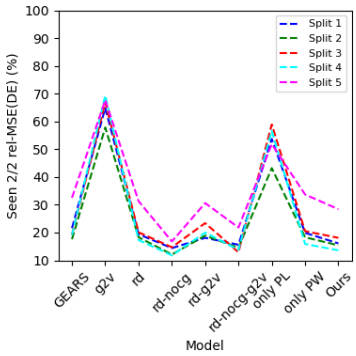
(b) K562

Figure 9: **Comparison of two error-rate representing metrics of RPE1 (a) and K562 (b) datasets.** Both are single-gene perturbation datasets with only seen 0/1 to compare.



(a) Seen 0/2

(b) Seen 1/2



(c) Seen 2/2

Figure 10: **The ablation study comparing rel-MSE (%) of top 20 DE genes over seen 0/2 (a), seen 1/2 (b) and seen 2/2 scenarios (b), on 5 splits of Norman dataset**. Here, *g2v* is GEARS that also uses Gene2Vec to initialize the gene embedding, *rd* model is GEARS changed by merely reducing the possible perturbed genes to the genes in the dataset, and *nocg* is GEARS without the cross-gene layer. For our design, *only PL/PW* means the model with only the PertLocal/PertWeight encoder.

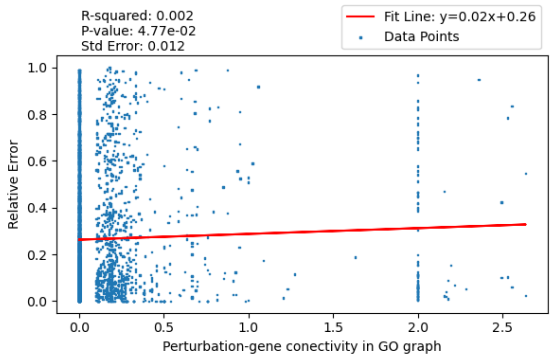


Figure 11: **Relative error of AttentionPert versus connectivity of perturbation-gene pair.** The connectivity is defined as the sum of edge weights between the perturbed genes and the affected gene in the Gene Ontology graph.