Accelerating Systematic Prediction of Variant Effects and Sequence Interpretation with Multiplexer Models

Dennis Tang [1] **Chenlai Shi** [1] **Jian Zhou** [1]

Abstract

Deep learning models have found wide applications in capturing sequence dependencies of biological functions. These models are often used to predict the effect of a large number of mutations in a sequence, a method also known as in silico mutagenesis (ISM). ISM is a common approach for interpreting the sequence dependencies captured by the model. However, this process can be computationally expensive as the naive implementation requires the model to individually process each mutation. Here, we introduce a new framework called "Multiplexer" that can simultaneously predict the effects of a high number of input variations given any "Base" sequence model. Multiplexer models can help identify important sequence patterns underlying its function, such as transcription factor binding, by discovering impactful mutations. We demonstrate that Multiplexer models can be 100x faster than naive methods and 10x faster than existing acceleration methods, while achieving very similar accuracy when applied to experimental datasets. We expect the ability to make fast, large-scale, predictions of mutation effects to accelerate the process of understanding the sequence dependencies of genome regulation. Furthermore, to further facilitate the use of the Multiplexer model, we have developed a Python package that allows users to easily visualize Multiplexer predictions and train sequence-based Multiplexer models.

1. Introduction

Deep learning models are widely used in the biomedical sciences to predict various biological features such as chromatin profiles, gene expression, or 3D genome structure from a DNA sequence (Alipanahi et al., 2015; Zhou & Troyanskaya, 2015; Kelley et al., 2015; Zhou et al., 2018; Jaganathan et al., 2019; Fudenberg et al., 2020; Žiga Avsec et al., 2021; Zhou, 2022) by capturing sequence-dependencies of biological functions. Because sequence models are able to generalize to new unseen sequences, one key application of sequence models is the prediction and analysis of mutation effects. In particular, sequence models allow for the systematic characterization of all possible single-nucleotide mutations, i.e. performing in silico mutagenesis (ISM), which is commonly used to identify key base-pairs in regulatory sequences or to predict all mutation consequences (Zhou & Troyanskaya, 2015; Kelley et al., 2015; Žiga Avsec et al., 2021). However, because the computational cost of sequence models scales linearly with the number of variants, predicting the effects of a high number of sequence mutations can take a considerable amount of time. Accelerated ISM methods including fastISM, Yuzu, and multiplexed ISM (Nair et al., 2022; Schreiber et al., 2021; Zhou, 2022) have recently been developed to leverage model architecture-specific properties to improve model efficiency, but these methods only offer a modest improvement in ISM speed and require specific model designs such as convolution-based architectures.

More broadly, predicting the effects of a large number of input variations is a common task in deep learning model interpretation. By tracking how input perturbations change a model's output, it is possible to determine the critical features of an input space. Therefore, analyzing how input perturbations effect model predictions is a fundamental building block of many deep learning model interpretation methods (Zeiler & Fergus, 2013; Ribeiro et al., 2016; Lundberg & Lee, 2017) and a generally applicable method for accelerating input variation prediction can also accelerate other methods of model interpretation.

Here, we introduce a "Multiplexer" model that provides fast and systematic prediction of all SNV (single nucleotide variant) effects. Given a "Base" model that makes predictions for sequence-based inputs, Multiplexer models are trained to simultaneously generate Base model predictions for a large set of input variations with drastically

[1]Lyda Hill Department of Bioinformatics, University of Texas Southwestern Medical Center, USA. Correspondence to: Jian Zhou <jian.zhou@utsouthwestern.edu>.

The 2023 ICML Workshop on Computational Biology. Baltimore, Maryland, USA, 2023.

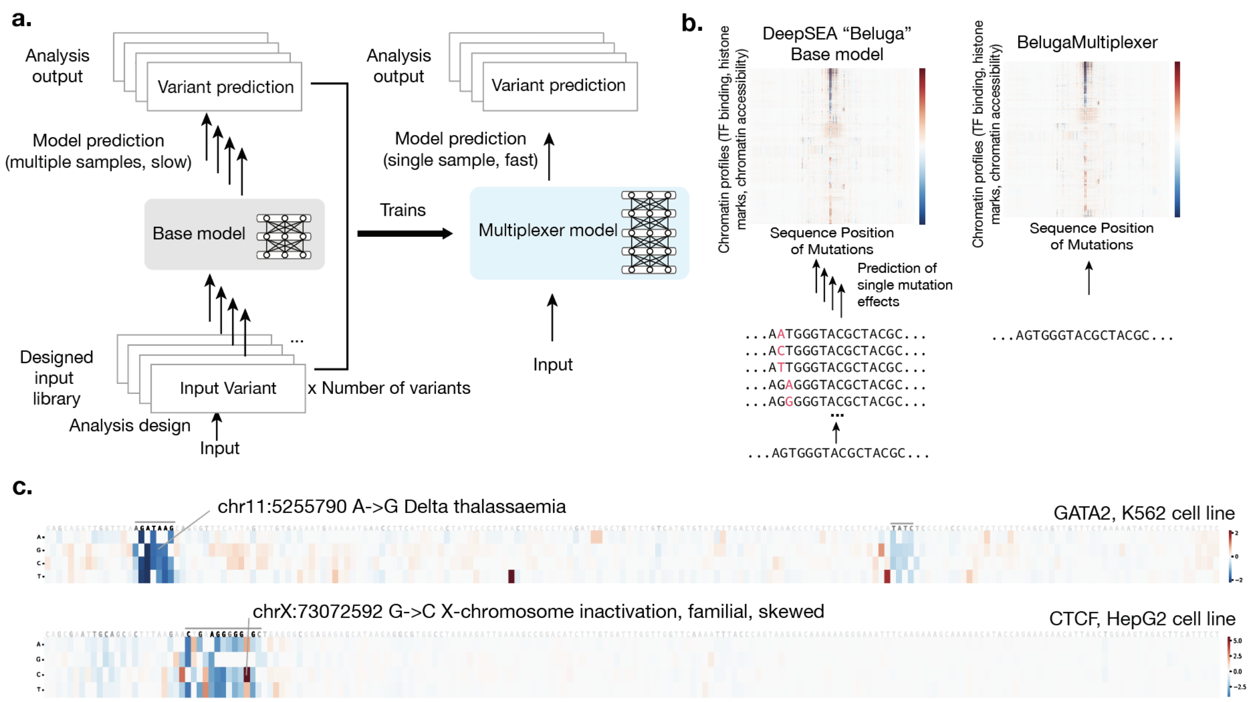

Figure 1. **Multiplexer speeds up systematic prediction of variant effects by deep learning sequence models** (a) Schematic overview of Multiplexer framework applied to sequence models. The Base model takes in one input at a time to generate predictions. Training data for the Multiplexer model is created by using the Base model to generate predictions for all possible input variants and the Multiplexer is trained to predict the same set of variant predictions with only one pass given the base, non-variant input. (b) Schematic overview of the DeepSEA "Beluga" and BelugaMultiplexer models. DNA sequence inputs to Beluga need to be mutated at each position before the forward pass whereas the BelugaMultiplexer only needs to forward propogate a single input. (c) ISM examples for sequences containing known pathogenic mutations. The reference sequence is shown above the plot, each of the four rows indicate a nucleotide mutation, and known motifs are annotated with a horizontal line. Darker letters and bluer heatmap values indicate more negative mutation effects.

reduced computation (Figure 1a). For example, a nucleotide sequence of length n can have $3n$ SNVs and would normally require $3n+1$ instances of forward propagation to compute ISM. However, for Multiplexers, it would only require one. By obviating the need to individually compute predictions for each input variation, Multiplexer models significantly improve the computation speed of ISM and other methods of model interpretation.

2. Methods

2.1. Base Model and Model Design

To demonstrate that a genomic sequence Multiplexer can accelerate both the systematic prediction of genomic mutation effects and the interpretation of DNA-sequences, we train a Multiplexer that uses the DeepSEA 'Beluga' model (Zhou et al., 2018), a sequence model that predicts chromatin profiles (including 2,002 transcription factors, histone marks, and chromatin accessibility profiles), as a Base model. Our trained "BelugaMultiplexer" can simultaneously predict the chromatin profile effects of every SNV in the sequence and thus offers a significant speed-up when performing in silico saturation mutagenesis as it can make the equivalent of 6,001 Base model predictions (1 reference + 2,000 base-pairs x 3 alternative sequences) given a single sequence as input (Figure 1b). Furthermore, BelugaMultiplexer predictions can also directly allow users to interpret which sequence patterns are important for any of the 2002 chromatin profiles. For example, using the sequence containing a Delta thalassaemia mutation (chr11:5255790, Atak et al. (2021)) as input, BelugaMultiplexer identified a sequence element containing the GATA motif and the disease-causing mutation, as well as a second weaker GATA motif nearby (Figure 1c). In another example, for a sequence carrying a familial skewed X-chromosome inactivation mutation (chrX:73072592, Atak et al. (2021)), BelugaMultiplexer identified the sequence element that contained the CTCF motif as well as the skewed X-chromosome inactivation mutation that was known to establish a new CTCF binding site (Figure 1c).

2.2. Data Processing And Model Training

To train BelugaMultiplexer, genomic sequences were retrieved by sampling from the hg19 human reference genome, in consistency with the DeepSEA "Beluga" model. To con-

struct a Multiplexer training sample, a sequence of 2,000 base-pairs was sampled (reference sequence) and mutated to create 6,000 SNV sequences (alternative sequences) that represented all possible mutations of the reference sequence. We then used the Base model to make chromatin profiles predictions for both the reference and alternative sequences and took the log-odds difference between them. Specifically, we defined training target y to have entries y_{ijk} where:

$$y_{ijk} = \log\left[\frac{(\text{alt}_{ijk} + \epsilon)(1 - \text{ref}_k + \epsilon)}{(1 - \text{alt}_{ijk} + \epsilon)(\text{ref}_k + \epsilon)}\right]$$

Where alt is the set of chromatin profile predictions made for the alternative sequences, ref is the set of predictions made for the reference sequence, ϵ is $1e{-}6$, and i, j, k index the position, possible mutations, and the chromatin profiles, respectively.

Then, given the reference sequence as input, BelugaMultiplexer was trained to minimize the mean-squared error between its predictions and y. A detailed description of model training can be found in Appendix A.

3. Results

3.1. Model Architecture and Correlation Comparison

To determine the best architecture, we trained four variations of the Multiplexer model and measured the correlation between their predictions and the predictions made by the Base mode on a set of 2,500 validation sequences. For each predicted chromatin profile, we calculated the weighted correlation between the Multiplexer and Base model predictions, where mutations were weighted by the absolute predicted effects from the Base model (as detailed in appendix B). All Multiplexer variations were fundamentally composed of seven convolutional blocks. However, the model architecture with the highest correlation (Figure 2a) additionally employed two design features to facilitate base-pair resolution output: first, it used a U-Net structure (Ronneberger et al., 2015) that allows the model to both integrate sequences across a long range and also reintroduce high-resolution information from early layers to the later layers in the network; second, since early convolution layers have no information about relative position within sequence, it introduces positional information to the input in the forward pass of the Multiplexer model. Because positional information is important for Beluga predictions, positional encodings help BelugaMultiplexer's predictive performance. The model architecture is further detailed in appendix C.

3.2. Speed Comparison

To quantify the speed improvement of the BelugaMultiplexer model, we compared the time the Multiplexer model takes to compute ISM with the time it takes for Beluga and Yuzu, a method that accelerates ISM with compressed sensing (Figure 2b). To facilitate this comparison, we sampled 512 sequences and tracked the time for each method to compute ISM at different batch sizes. Further implementation details are given in appendix D.

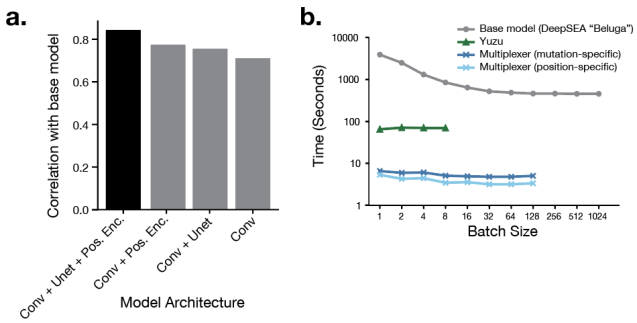

Figure 2. **Multiplexer architecture design achieves high consistency with Base model and 100x speed up** (a) Performance comparison of Multiplexer architectures. The performance of several Multiplexer architectures that combined positionally encoded inputs, U-Net structure, and convolution layers. (b) Speed comparison of several ISM methods. We found that the Multiplexer model is over 10x faster than other published methods. Each method was run up to the largest batch size that fit on GPU memory.

We found that the trained BelugaMultiplexer model was able to compute ISM 94x faster than the Base model and 13x faster than Yuzu, a substantial speed up. Moreover, we found that Yuzu consumes significantly more memory than the Multiplexer model and can only process relatively small batch sizes. BelugaMultiplexer could be further accelerated in tasks where predicting an average mutation effect per position is sufficient, such as assigning sequence importance scores. To demonstrate this, we trained a position-specific Multiplexer model that predicts the average mutation effect at each position and found this model to be 142x faster than the Base model and 20x faster than Yuzu (Figure 2b). Further analysis of the viability of the model is shown in appendix E.

3.3. Performance on Experimental Data

To evaluate the performance of the Multiplexer model on external experimental data, we compared the accuracy of the BelugaMultiplexer model with the Base model on allele-imbalanced variants from the ATAC-seq (Tehranchi et al., 2018), DNase-seq (Neph et al., 2012), and histone mark QTLs datasets (Chen et al., 2016) where one allele leads to more accessible chromatin or more histone marks compared to the other (Figure 3a). The datasets were first filtered and only the variants with the lowest 8,000 p-values were retained for analysis. Each variant contained a nucleotide mutation and an indication of whether the alternative or reference allele was overrepresented. For each variant, a reference and alternative sequence was generated. The sequences and their reverse-complements were then forward-

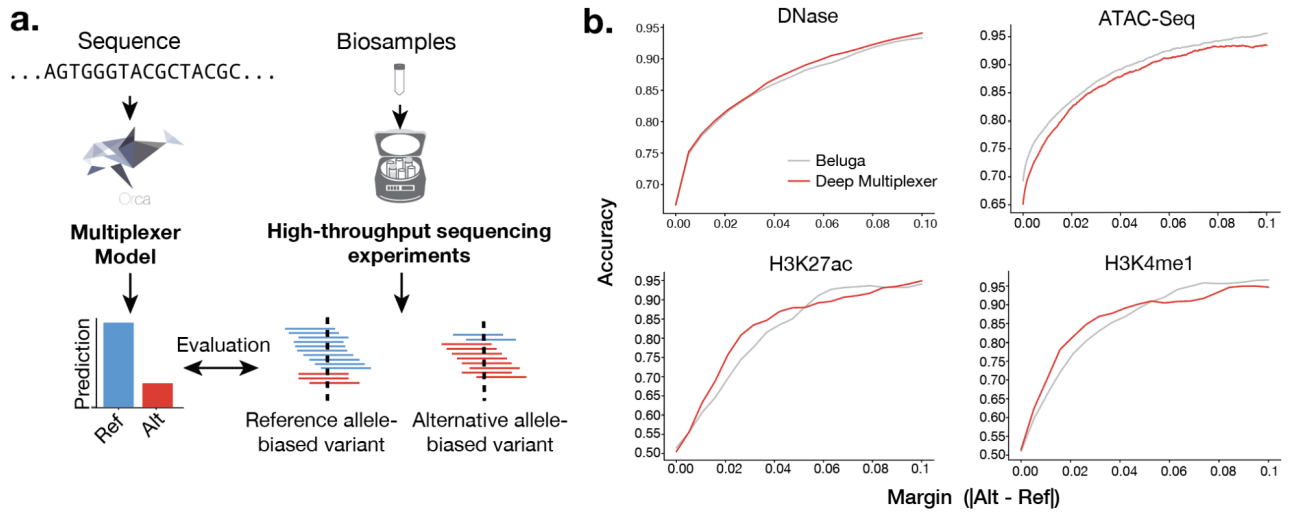

Figure 3. **Multiplexer achieves variant effect prediction accuracy comparable to the base model** (a) Overview of variant effect prediction evaluation with experimental data. Experimental data are used to identify alternative-allele or reference-allele-based variants. These variants were then used to evaluate the predictions provided by computational methods using only the sequence carrying each of the alleles (shown in figure 3b). (b) The variant effect prediction performance of Beluga Multiplexer is comparable with Beluga model on variants from DNase, ATAC-seq, H3K27ac, and H3K4me1 datasets. The y-axis shows the accuracy of predicting the direction of variant effect for variants above a threshold of absolute predicted effect ($|\text{Alt} - \text{Ref}|$) (x-axis).

propagated through the models and averaged to produce chromatin profile predictions. Accuracy was determined by analyzing whether the predictions were in agreement with which allele is over-represented. To provide a confidence threshold for these predictions, we created margins based on the absolute value of the difference between reference and alternative prediction and calculated the accuracy at each margin. Importantly, we observed very similar performance between the two models on all of the allele imbalance datasets (Figure 3b). Both models accurately predicted which allele leads to more accessible chromatin or more enhancer histone marks (H3K27ac or H3K4me3) for variants with strong predicted effects. We also note that DNase predictions from Beluga and BelugaMultiplexer demonstrate comparable performance when evaluated on allele imbalance data from either ATAC-seq or DNase-seq, suggesting that these predictions generalize across techniques. We conclude that Multiplexers show no obvious reduction in generalized performance on external datasets compared to the Base model.

3.4. Python Library and Web Server for Sequence Interpretation with Multiplexer

Because Multiplexer outputs illustrate how specific nucleotide mutations in the reference sequence change the chromatin profile, it becomes easy to identify key regulatory elements in the reference sequence (as demonstrated in Figure 1c). Therefore, the Multiplexer model provides a fast and straightforward method for sequence interpretation. We expect Mutliplexer to be an useful tool for studying regulatory sequence, and have developed a command line tool[1] and web server that enables users to generate Multiplexer predictions to facilitate such analysis. This tool can visualize predictions as heatmaps, which represents the changes in chromatin profiles resulting from every mutation at each position in the reference sequence (shown in appendix F).

4. Discussion

We demonstrated that Multiplexer models can significantly speed up large-scale mutation effect prediction without loss of accuracy and observe that Multiplexers can generalize well to external datasets, reaching similar or better performance compared to the Base model even though the correlation with base model is not perfect. This may be because Multiplexer training has a regularization effect, similar to the case of knowledge distillation (Hinton et al., 2015). Additionally, we have shown that, when applied to sequence interpretation, Multiplexer models can be a powerful tool to identify which sequence patterns contribute the most to biological functions. Finally, even though this paper is focused on Multiplexers' application to sequence models, the Multiplexer approach is generally applicable to diverse problems that involve simultaneous predictions of programmed input variations. This can include predicting other types of input variations such as small indels or random shuffling of small sequence segments. The Multiplexer framework is in principle also applicable to other areas of deep learning such as predicting occlusion effects in image models, or semantic analysis in sentiment classification tasks.

[1] Available at https://github.com/jzhoulab/Multiplexer

References

Alipanahi, B., Delong, A., Weirauch, M. T., and Frey, B. J. Predicting the sequence specificities of dna- and rna-binding proteins by deep learning. *Nature Biotechnology*, 33:831–838, 2015.

Atak, Z. K., Taskiran, I. I., Demeulemeester, J., Flerin, C., Mauduit, D., Minnoye, L., Hulselmans, G., Christiaens, V., Ghanem, G. E., Wouters, J., and Aerts, S. Interpretation of allele-specific chromatin accessibility using cell state–aware deep learning. *Genome Research*, 31:1082 – 1096, 2021.

Chen, L., Ge, B., Casale, F. P., Vasquez, L., Kwan, T., Garrido-Martín, D., Watt, S., Yan, Y., Kundu, K., Ecker, S., Datta, A., Richardson, D., Burden, F., Mead, D., Mann, A. L., Fernández, J. M., Rowlston, S. P., Wilder, S. P., Farrow, S., Shao, X., Lambourne, J. J., Redensek, A., Albers, C. A., Amstislavskiy, V., Ashford, S., Berentsen, K., Bomba, L., Bourque, G., Bujold, D., Busche, S., Caron, M., Chen, S.-H., Cheung, W. A., Delaneau, O., Dermitzakis, E. T., Elding, H., Colgiu, I., Bagger, F. O., Flicek, P., Habibi, E., Iotchkova, V., Janssen-Megens, E. M., Kim, B., Lehrach, H., Lowy, E., Mandoli, A., Matarese, F., Maurano, M. T., Morris, J. A., Pancaldi, V., Pourfarzad, F., Rehnstrom, K., Rendon, A., Risch, T., Sharifi, N., Simon, M.-M., Sultan, M., Valencia, A., Walter, K., Wang, S.-Y., Frontini, M., Antonarakis, S. E., Clarke, L., Yaspo, M.-L., Beck, S., Guigó, R., Rico, D., Martens, J. H. A., Ouwehand, W. H., Kuijpers, T. W., Paul, D. S., Stunnenberg, H. G., Stegle, O., Downes, K., Pastinen, T., and Soranzo, N. Genetic drivers of epigenetic and transcriptional variation in human immune cells. *Cell*, 167:1398 – 1414.e24, 2016.

Fudenberg, G., Kelley, D. R., and Pollard, K. S. Predicting 3d genome folding from dna sequence with akita. *Nature methods*, 17:1111 – 1117, 2020.

Hinton, G. E., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.

Jaganathan, K., Panagiotopoulou, S., McRae, J. F., Darbandi, S. F., Knowles, D. G., Li, Y. I., Kosmicki, J. A., Arbelaez, J., Cui, W., Schwartz, G. B., Chow, E. D., Kanterakis, E., Gao, H., Kia, A., Batzoglou, S., Sanders, S. J., and Farh, K. K.-H. Predicting splicing from primary sequence with deep learning. *Cell*, 176:535–548.e24, 2019.

Kelley, D. R., Snoek, J., and Rinn, J. L. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Research*, 26: 990 – 999, 2015.

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. *ArXiv*, abs/1705.07874, 2017.

Nair, S., Shrikumar, A., Schreiber, J., and Kundaje, A. fastism: Performant in-silico saturation mutagenesis for convolutional neural networks. *Bioinformatics*, 2022.

Neph, S. J., Vierstra, J., Stergachis, A. B., Reynolds, A. P., Haugen, E., Vernot, B., Thurman, R. E., Sandstrom, R. S., Johnson, A. K., Maurano, M. T., Humbert, R., Rynes, E., Wang, H., Vong, S., Lee, K., Bates, D. L., Diegel, M., Roach, V., Dunn, D., Neri, J., Schafer, A., Hansen, R. S., Kutyavin, T., Giste, E., Weaver, M., Canfield, T. K., Sabo, P. J., Zhang, M., Balasundaram, G., Byron, R., MacCoss, M. J., Akey, J. M., Bender, M., Groudine, M., Kaul, R., and Stamatoyannopoulos, J. A. An expansive human regulatory lexicon encoded in transcription factor footprints. *Nature*, 489:83 – 90, 2012.

Ribeiro, M. T., Singh, S., and Guestrin, C. "why should i trust you?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. *ArXiv*, abs/1505.04597, 2015.

Schreiber, J., Nair, S., Balsubramani, A., and Kundaje, A. Accelerating in-silico saturation mutagenesis using compressed sensing. *bioRxiv*, 2021.

Tehranchi, A. K., Hie, B. L., Dacre, M., Kaplow, I. M., Pettie, K. P., Combs, P. A., and Fraser, H. B. Fine-mapping cis-regulatory variants in diverse human populations. *eLife*, 8, 2018.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, 2013.

Zhou, J. Sequence-based modeling of three-dimensional genome architecture from kilobase to chromosome scale. *Nature genetics*, 54:725 – 734, 2022.

Zhou, J. and Troyanskaya, O. G. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature Methods*, 12:931–934, 2015.

Zhou, J., Theesfeld, C. L., Yao, K., Chen, K. M., Wong, A. K., and Troyanskaya, O. G. Deep learning sequence-based ab initio prediction of variant effects on expression and disease risk. *Nature genetics*, 50:1171 – 1179, 2018.

Žiga Avsec, Agarwal, V., Visentin, D., Ledsam, J. R., Grabska-Barwinska, A., Taylor, K. R., Assael, Y., Jumper,

J. M., Kohli, P., and Kelley, D. R. Effective gene expression prediction from sequence by integrating long-range interactions. *Nature Methods*, 18:1196 – 1203, 2021.

Appendix

A. Model Training

Training sequences were drawn from all chromosomes except chr8 and chr9 which were reserved for validation. For both training and validation sequences, the chromosomes were sampled with probability proportional to their length while the position within each chromosome was sampled uniformly at random. Specifically, BelugaMultiplexer was trained to minimize the objective defined by:

$$loss = ||y - \hat{y}||^2$$

Where y is the training target defined previously and $\hat{y}$ is the output of BelugaMultiplexer.

Additionally, the BelugaMultiplexer was trained with Adam optimization and a 0.0001 learning rate. During each training epoch, a new batch of 16 training samples was generated and the validation loss was computed every 1,000 epochs. Validation was conducted on a fixed set of 288 validation samples that were randomly drawn before the start training. Training was stopped when there were no further decreases in validation loss at which point the model parameters with the lowest validation loss were returned. In total, the best performing Multiplexer model was trained for 50,000 epochs.

B. Weighted Correlation

To calculate the weighted correlation, let:

$$WM(x, y) = \frac{\sum_i x_i y_i}{\sum_i y_i}$$

$$WV(x, y, z) = \frac{\sum_i z_i (x - WM(x, z))(y - WM(y, z))}{\sum_i z_i}$$

$$WC(x, y, z) = \frac{WV(x, y, z)}{\sqrt{WV(x, x, z) WV(y, y, z)}}$$

Where WM calculates the weighted-mean, WV calculates the weighted-covariance, and WC calculates the weighted-correlation. For our comparison, we calculate:

$$WC(Base, Multiplexer, |Base|)$$

Where $Base$ is the log-odds difference between the predictions made by the Base Model for the alternative and reference sequences, $Multiplexer$ is the set of predictions make by the Multiplexer model, and $||$ indicates absolute value.

C. Model Architecture

The Multiplexer model consists of 7 sequential convolutional blocks which each contain 1-dimension convolutional layers, batch normalization layers, and ReLU activation layers. Our implementation of the U-Net structure organizes these blocks such that the inputs of blocks one through five are the outputs of the previous block, the input to the sixth block is the sum of the outputs of the fifth and third block, and the input to the seventh block is the sum of the outputs of the second and sixth block. The output of the seventh block is then reshaped and returned as the prediction. Furthermore, positional encodings are included as a feature of the model's forward propagation method and the value of the positional encoding concatenated to each value of the input is a function of the position defined by:

$$f(position) = min(0.001 * position, 2 - 0.001 * position)$$

$$\text{where } 0 \leq position \leq 2000$$

A visualization of the BelugaMultiplexer architecture can be found in figure C.1 .

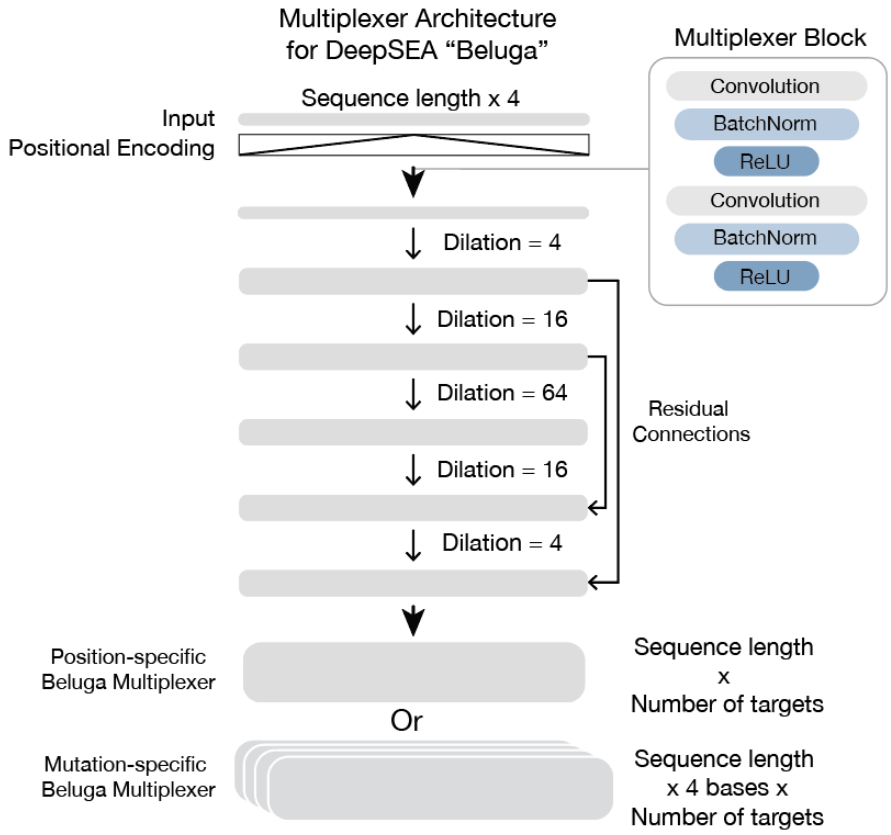

Figure C.1. **BelugaMultiplexer architecture** The BelugaMultiplexer model consists of seven "Multiplexer Blocks". The best performing model also concatenates a positional encoding vector to the input and uses residual connections that sum earlier activations with later activations in the network.

D. Speed Comparison

Yuzu is an ISM acceleration method that improves upon fastISM, which was the previous fastest ISM method (Nair et al., 2022). Yuzu works by modifying a sequence model with compressed sensing to accelerate the speed of its' ISM computations. In our comparison, we apply Yuzu to the Beluga model.

Since ISM requires predictions for every possible input variation, the naive implementation requires preparing the same number of mutated input sequences as the number of variations. In our comparison of model speeds, we assumed that these mutated sequences are pre-computed and did not include the time to generate them in our measurement of the Base model's speed. In practice, generating one sequence input for Multiplexer takes significantly less time than generating all input sequences for naive ISM, so including the time for sequence generation would result in a greater speed-up than our reported figure. Additionally, the Yuzu method is currently limited in the architectures it supports (e.g reshapes and residual connections are not supported, maxpool layers must be rewritten, etc) so the size of the Base model had to be altered before Yuzu was applied which marginally increased the padding of the convolution layers. In order to account for this increase, we computed ISM with both this larger, padded Base model and our original Base model and then proportionally scaled down the computed times for Yuzu (the larger model was 10% slower so the calculated Yuzu time by multiplied by a factor of 0.91) .

E. Position-Specific Multiplexer

To demonstrate the viability of the position-specific Multipexer model, we also compared the predictive performance of the position-specific Multiplexer and the mutation-specific Mutliplexer. We found that prediction of averaged mutation

effects by the position-specific Multiplexer model is slightly improved over the mutation-specific Multiplexer model, while the mutation-specific Multiplexer predicts individual mutation effects better than using the averaged effect from the position-specific Multiplexer model (Figure E.1). Thus, the position-specific Multiplexer is superior in both speed and performance in applications where only average mutation effects are needed.

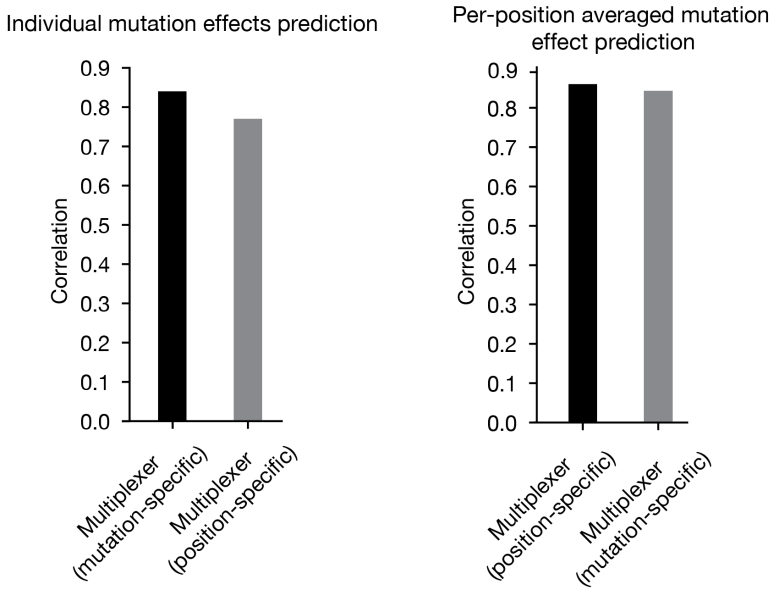

Figure E.1. **Performance comparison between mutation-specific Multiplexer and position-specific Multiplexer on individual mutation effects prediction (left) and per-position averaged mutation effect prediction tasks (right)**. Performance on the former task was measured by calculating the correlation between Beluga predictions and the mutation-specific Multiplexer predictions as well as the correlation between Beluga predictions and repeated position-specific Multiplexer predictions (repeated 3 additional times to match the dimensions of the mutation-specific Multiplexer). The latter task similarly computed correlations but averaged the alternative predictions in the mutation-specific Multiplexer predictions and Beluga model and did not repeat the position-specific Multiplexer predictions.

F. Webserver

Here, we also include a visualization of the Mutliplexer python library's capabilities.

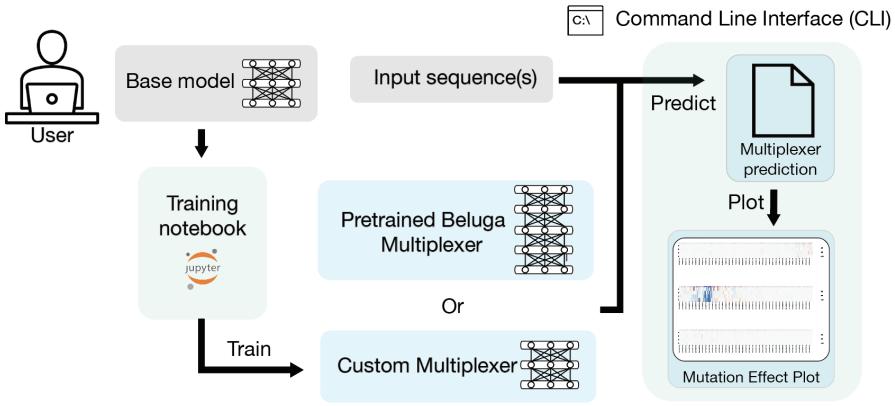

Figure F.1. **Schematic illustration of the Multiplexer python library** This library includes a training notebook that enables users to provide a Base model and train a custom SNV Multiplexer, and also a command line interface where users can make predictions and plots with either the trained BelugaMultiplexer model or a custom model.