Genomic Interpreter: A Hierarchical Genomic Deep Neural Network with 1D Shifted Window Transformer

Anonymous Authors[1]

Abstract

Given the increasing volume and quality of genomics data, extracting new insights requires interpretable machine-learning models. This work presents Genomic Interpreter: a novel architecture for genomic assay prediction. This model outperforms the state-of-the-art models for genomic assay prediction tasks. Our model can identify hierarchical dependencies in genomic sites. This is achieved through the integration of 1D-Swin, a novel Transformer-based block designed by us for modelling long-range hierarchical data. Evaluated on a dataset containing 38,171 DNA segments of 17K base pairs, Genomic Interpreter demonstrates superior performance in chromatin accessibility and gene expression prediction and unmasks the underlying 'syntax' of gene regulation.

1. Introduction

Functional genomics uses a variety of assays to explore the roles of genes in a genome. These assays allow researchers to quantify gene expression (de Hoon & Hayashizaki, 2008), test chromatin accessibility and understand gene regulation (Eraslan et al., 2019). In this paper:

- We introduce Genomic Interpreter, an attention-based model for genomic assay prediction.
- We design a task-agnostic hierarchical Transformer, 1D-Swin, for capturing long-range interaction in 1-D sequences.
- We show that our model performs better than the state-of-the-art.
- We further demonstrate that the hierarchical attention mechanism in Genomic Interpreter provides us with interpretability. This can help biologists to identify and validate relationships between different genomic sequences.

2. Related Work

Deep Learning for genomic assays prediction Deep Neural Networks (DNNs) have seen success in predicting genome-scale sequencing assays such as CAGE (Takahashi et al., 2012), DNase-seq (He et al., 2014), and ChIP-seq (Zhou et al., 2017). In genomic assay prediction, models are provided with a DNA sequence $x \in \mathbb{R}^{n\times 4}$ and are required to forecast sequencing assay outputs $y \in \mathbb{R}^{m\times T}$ for various tracks, where n defines the DNA sequence length, 4 refers to four nucleotides in DNA, m represents the length of the output sequence, with each output value being an average read over a specific DNA segment, and T denotes the track count, referring to the different types of assay outputs being predicted, such as specific sequencing assays performed in particular organisms. In these models, DNA sequences are encoded and transformed into high-dimensional vector representations. These encoded vectors are then processed through a series of linear layers to predict the corresponding real-value assay readings. Existing genomic models can be divided into two categories: the first category uses Convolution Neural Networks (CNNs) and pooling layers as the encoder (Alipanahi et al., 2015; Kelley et al., 2018; Kelley, 2020); the second type of model consists of CNNs and pooling layers followed by Transformer blocks, which is first proposed by (Avsec et al., 2021) as Enformer, serving as the state-of-the-art model used in regulatory DNA study (Vaishnav et al., 2022).

Current state-of-the-art models predict coarse-grained read values, typically over 100 base pairs per read. This requires DNNs to reduce the spatial dimension and create a condensed representation for the input. This property is also shared by most computer vision tasks such as object detection (Girshick, 2015) and semantic segmentation (He et al., 2017). While using CNNs and pooling layers has proven effective in the past. More recently, transformer-based architectures are becoming more popular and are outperforming CNN-based architectures (Dosovitskiy et al., 2020; Liu et al., 2021). The success of these models motivated us to

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by *the 2023 ICML Workshop on Computational Biology*. Do not distribute.

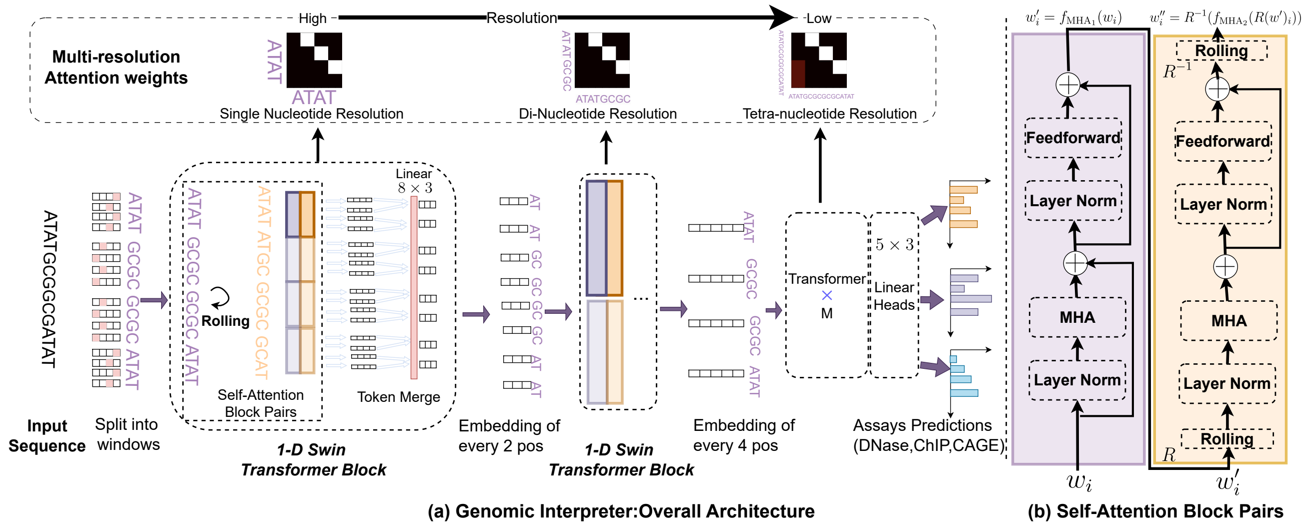
High
Resolution
Low
Multi-resolution Attention weights
Single Nucleotide Resolution
Di-Nucleotide Resolution
Tetra-nucleotide Resolution
Input Sequence
Split into windows
Rolling
Self-Attention Block Pairs
Linear 8×3
Token Merge
1-D Swin Transformer Block
Embedding of every 2 pos
1-D Swin Transformer Block
Embedding of every 4 pos
Transformer × M
5×3 Linear Heads
Assays Predictions (DNase,ChIP,CAGE)
(a) Genomic Interpreter:Overall Architecture
$w'_i = f_{\mathrm{MHA}_1}(w_i)$
$w''_i = R^{-1}(f_{\mathrm{MHA}_2}(R(w')_i))$
Rolling
R^{-1}
Feedforward
Layer Norm
MHA
Layer Norm
Rolling
R
w_i
w'_i
(b) Self-Attention Block Pairs

Figure 1. (a) Genomic Interpreter: given an input sequence $x \in \mathbb{R}^{16\times 4}$, the target is to predict 3×4 read value arrays. x first traverses multiple 1D-Swin blocks. Each forward pass halves the token length; after two passes, the token number is reduced to 4. Then the resulting embedding is fed into Transformer Block and Linear Heads for final prediction. (b)Self-Attention Block Pairs: standard implementation is used for self-attention block

design attention-based models for genomic assay prediction. Furthermore, by calculating the Transformers' attention weights we can reveal genomic site dependencies (Ghotra et al., 2021), thus providing us with better interpretability compared to CNN-based approaches in assay prediction.

Swin Transformers The Shifted Window (Swin) Transformer builds hierarchical feature maps of input images through self-attention operations within a local window and shift operations (Liu et al., 2021). The computed feature maps are merged to higher layers, reducing the spatial size of inputs. Swin Transformer is efficient in processing long input sequences as self-attention is only computed within each window.

While Swin Transformer and its variants have shown improved performance in the computer vision tasks (Liu et al., 2022; Yuan et al., 2021). There is a lack of hierarchical transformer models for 1-dimensional data such as genomic sequence data. The local window approach has been adopted for long-range 1-D Transformer works such as Longformer (Beltagy et al., 2020) and Sparse Transformer (Child et al., 2019) to reduce the computation. However, no work has been done for reducing the spatial dimension in a hierarchical manner. This property is crucial for tasks requiring spatial reduction, such as genomic assay prediction.

3. Method

3.1. Model Architecture Overview

We propose a novel model architecture, termed Genomic Interpreter, that is designed to predict genomic assays. The Genomic Interpreter is composed of several parts: multiple 1-Dimensional Swin (1D-Swin) blocks, a Transformer block (Vaswani et al., 2017), and linear heads at the end. The structure is shown in Figure 1(a).

The process begins with an input sequence which acts as the initial token. This is denoted as $x \in \mathbb{R}^{n\times d}$, where n is the token length and d is the dimension for each token. One round of the 1D-Swin block transforms this matrix where $f_{\mathrm{1dSwin}}: \mathbb{R}^{n\times d} \to \mathbb{R}^{\frac{n}{2}\times\frac{2d}{\alpha_0}}$, where α_0 is the hidden size scaling factor. When α_0 is set to 1, the hidden size of the token is doubled.

The transformed matrix is denoted as $\mathbf{h}$. The number of 1D-Swin blocks (represented by K) is chosen to match the output length to the initial token length. If an exact match is not possible, a Crop operation is used to adjust to output length by removing certain elements from the ends of $\mathbf{h}$.

The overall architecture of the Genomic Interpreter can be summarized by three equations:

$$\mathbf{h}_K = f_{\mathrm{1dSwin}}(\mathbf{h}_{K-1}), \quad \mathbf{h}_0 = \mathbf{x} \tag{1}$$

$$\mathbf{h_c} = f_{\mathrm{crop}}(\mathbf{h_K}) \tag{2}$$

$$\mathbf{y} = \mathrm{LinearHeads}(\mathrm{TransformerBlock}(\mathbf{h_c})) \tag{3}$$

Here, $\mathbf{h^K}$ represents the output after K iterations of the 1D-Swin transformation, with dimensions $\frac{n}{2^K} \times \frac{2d}{(\alpha_0\alpha_1 \ldots \alpha_K)}$. This output is then passed through the Transformer blocks and linear heads in a feedforward fashion, yielding a per-track prediction denoted as $\mathbf{y}$. The dimensions of $\mathbf{y}$ are $m \times T$. As an example, $m = 4$ and $T = 3$ in Figure 1(a).

3.2. 1D-Swin Block

The standard Transformer model's quadratic time-space complexity, denoted as $\Omega(n^2)$ for input tokens $x \in \mathbb{R}^{n\times d}$, hinders its efficiency with long inputs like DNA sequences. The 1D-Swin Transformer reduces this complexity[1] to $\Omega(n)$ through two identical Multi-Head Attention (MHA) blocks.

The first Multi-Head Attention (MHA block), as depicted in Figure 1(b), processes each a subset of tokens with a window to capture local dependencies. And then a rolling operation together with the second MHA block is applied to capture the cross-window dependencies. Finally, pair-wise concatenation and linear transformation are used to form the spatially reduced token set. For the rigorous definition of this process, please refer to Appendix A.

3.3. Multi-resolution genome sites dependency detection

Capturing the interactions between sub-sequences in the genome is crucial for genomics science. Stacked 1D-Swin blocks allow Genomic Interpreter to have hierarchical representations of such genomic sequences. We can look at the learned attention patterns to see how tokens are interacting with each other.

Specifically, as shown at the top of Figure-1(a), the self-attention scores between each token are extracted at each layer, serving as a fine-grained map showing how nucleotide sequences of different lengths interact with each other.

3.4. Data

Understanding the dependencies between genomic sites requires a comprehensive dataset. The dataset provided by Enformer (Avsec et al., 2021) offers a broad spectrum of gene expression data. However, it demands an unrealistic amount of computational resources to replicate their algorithm training[2] for practical usage. To address this issue, we have developed a scaled-down version of genomic datasets named 'BasenjiSmall'.

The 'BasenjiSmall' dataset comprises 38,171 data points (X, Y). This is the same amount of data points as the original Enformer data set. Each data point consists of a DNA segment $\mathbf{x}$ and assay reads $\mathbf{y}$. $\mathbf{x} \in \mathbb{R}^{17,712\times 4}$ represents a DNA segment spanning $17,712$ base pairs (bp). $\mathbf{y} \in \mathbb{R}^{80\times 5313}$ refers to the coarse-grained read values across 80×128 bp and 5313 tracks expanding various cell and assay types, these tracks include: (1) 675 DNase/ATAC tracks, (2) 4001 ChIP Histone and ChIP Transcription Factor tracks and (3) 639 CAGE tracks.

This reduced dataset maintains the richness of gene expression data but is more manageable in terms of computational requirements.

[1]The time complexity of 1D-Swin depends upon a hyperparameter: the window size. It ranges from $O(n)$ for a window size of 1 to $O(n^2)$ for a window size of n

[2]On original dataset, the training time of full-size Enformer model requires 64×3 TPU days

Table 1. Pearson correlation of 5 models for genomic assay prediction on 5313 tracks, tracks are grouped into DNase, CHIP histone/transcription factor, and CAGE.

MODEL NAME	DNASE	CHIP	CAGE	OVERALL
MAXPOOLCNNS	0.4996	0.4320	0.2581	0.4195
ATTPOOLCNNS	0.4970	0.4346	0.2041	0.4146
ATTPOOLDILATE	0.4778	0.4395	0.1803	0.4130
ENFORMER	0.5462	0.4575	**0.3307**	0.4536
1D-SWIN	**0.5583**	**0.4670**	0.3242	**0.4614**

3.5. Training

We utilized Pytorch Lightning framework with (Distributed Data Parallelism) DDP strategy for parallel training.

All the models shown in Section 4 are implemented in Pytorch and trained with 3 NVIDIA A100-PCIE-40GB, with a training time of 3*10 GPU hours per model. The batch size is set to 8 for each GPU, with an effective batch size of 24. Adam optimizer (Kingma & Ba, 2014) is used together with the learning rate scheduler, CosineAnnealingLR (Gotmare et al., 2018) and a learning rate of 0.0003 to minimize the Poisson regression loss function for all models.

4. Results

4.1. Gene Expression Prediction

We compared the performance of 1D-Swin with Enformer (Avsec et al., 2021) and other standard genomic assay prediction models using a hold-out test set from BasenjiSmall. This included models implementing CNNs with MaxPooling (Alipanahi et al., 2015), CNNs with AttentionPooling, and Dilated CNNs with Attention Pooling (Kelley et al., 2018). For a detailed view of the implementation process, refer to Appendix B.

Model performance is evaluated using the Pearson correlation between predictions and true values. Table 1 shows the evaluation results for five models across 1937 DNA segments with 5313 tracks, classified into DNase, ChIP, and CAGE groups. These results show that 1D-Swin outperforms other models overall, particularly in the DNase and ChIP groups.

Figure 2 shows a pairwise comparison of 1D-Swin with other methods across track groups. Points above the reference line indicate superior performance by 1D-Swin. Figure 3 visualize the gene expression predictions from two

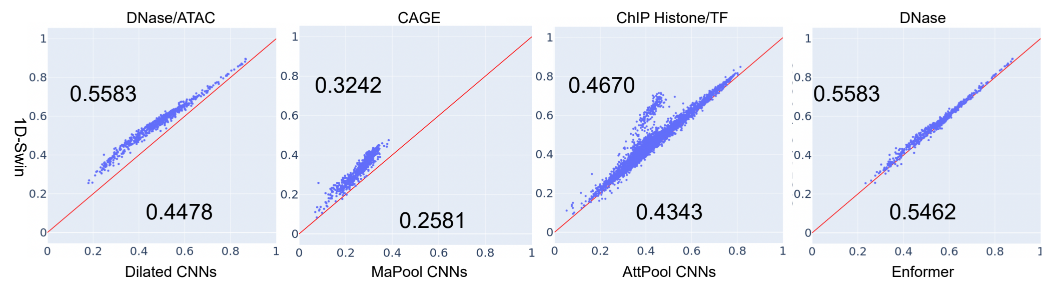

Figure 2. Pairwise Comparison between 1D-Swin and Competing Models: The Pearson correlation for each track is calculated across all DNA segments within the test set. Mean Pearson correlations are denoted on Y-axis for 1D-Swin and X-axis for the reference models.

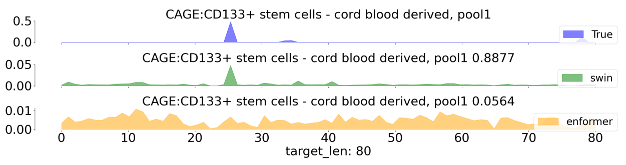

Figure 3. Given a DNA segment, two models will make predictions for the read values of the CAGE-CD133 track.

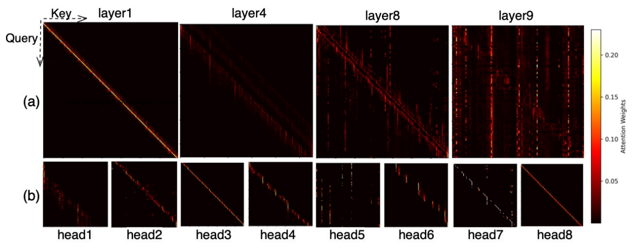

Figure 4. (a) Attention weights from different 1D-swin layers reveal a pattern to have more off-diagonal dependency at Higher Hierarchy (b) Attention weights of different heads on the 8th layer learn to capture various features.

models for CD133 stem cells, the visualisation evident that the prediction provided by 1D-swin matches more with the true values.

4.2. Self-Attention and Gene Regulation

While attention does not equate to explanation (Pruthi et al., 2019), it helps to provide insights into token interaction. Genomic Interpreter leverages this capability to extract hierarchical self-attention values at varying resolutions, potentially unmasking the underlying 'syntax' of gene regulation.

In this hierarchical structure, each matrix position represents a single nucleotide at the first layer and 2^{K-1} nucleotides by the Kth layer. Figure 4(a) illustrates this, showing that lower layers favour local attention while higher layers exhibit long-range token interactions.

This pattern likely reflects the reality of gene interaction at different levels. At the single-nucleotide level, interactions primarily occur between proximal tokens. As we increase the nucleotide length, longer nucleotide segments begin to form regulatory units such as enhancers and silencers (Maston et al., 2006) that can interact at increased distances.

We can interpret the models in more detail by looking at the attention between tokens at different heads. Figure 4(b) further reveals that distinct attention heads at layer 8 capture varying patterns. For instance, head1 primarily captures a 'look-back' pattern evident in the lower triangle attention, head5 seems to focus on long-range interactions, and head8 attends to immediate proximal interactions. Appendix C provides supplementary figures to visualize Attention Weights.

5. Conclusion and Future Work

This study presents a task-agnostic hierarchical transformer for one-dimensional data, underscoring the importance of efficient, comprehensive hierarchical models. When applied to genomic assay prediction, Genome Interpreter outperforms conventional models while providing interpretable insights.

As a transformer-based architecture, Genomic Interpreter can be reinforced with pretraining (Hendrycks et al., 2020) for improving out-of-distribution prediction and attention flow (Abnar & Zuidema, 2020) for mapping the obtained attention weights to the original input sequence. While genomic science has emphasized the importance of understanding hierarchical structures, this concept is also critical in other fields, including Natural Language Processing (NLP) where language understanding relies heavily on hierarchical concept interpretation. As a result, 1D-Swin may have the potential to be applied to these fields for capturing long-range, hierarchical information.

References

Abnar, S. and Zuidema, W. Quantifying attention flow in transformers. *arXiv preprint arXiv:2005.00928*, 2020.

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(8):831–838, 2015.

Avsec, Ž., Agarwal, V., Visentin, D., Ledsam, J. R., Grabska-Barwinska, A., Taylor, K. R., Assael, Y., Jumper, J., Kohli, P., and Kelley, D. R. Effective gene expression prediction from sequence by integrating long-range interactions. *Nature methods*, 18(10):1196–1203, 2021.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

de Hoon, M. and Hayashizaki, Y. Deep cap analysis gene expression (cage): genome-wide identification of promoters, quantification of their expression, and network inference. *Biotechniques*, 44(5):627–632, 2008.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Eraslan, G., Avsec, Ž., Gagneur, J., and Theis, F. J. Deep learning: new computational modelling techniques for genomics. *Nature Reviews Genetics*, 20(7):389–403, 2019.

Ghotra, R. S., Lee, N. K., and Koo, P. K. Uncovering motif interactions from convolutional-attention networks for genomics. In *NeurIPS 2021 AI for Science Workshop*, 2021.

Girshick, R. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

Gotmare, A., Keskar, N. S., Xiong, C., and Socher, R. A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation. *arXiv preprint arXiv:1810.13243*, 2018.

He, H. H., Meyer, C. A., Hu, S. S., Chen, M.-W., Zang, C., Liu, Y., Rao, P. K., Fei, T., Xu, H., Long, H., et al. Refined dnase-seq protocol and data analysis reveals intrinsic bias in transcription factor footprint identification. *Nature methods*, 11(1):73–78, 2014.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

Hendrycks, D., Liu, X., Wallace, E., Dziedzic, A., Krishnan, R., and Song, D. Pretrained transformers improve out-of-distribution robustness. *arXiv preprint arXiv:2004.06100*, 2020.

Kelley, D. R. Cross-species regulatory sequence activity prediction. *PLoS computational biology*, 16(7):e1008050, 2020.

Kelley, D. R., Reshef, Y. A., Bileschi, M., Belanger, D., McLean, C. Y., and Snoek, J. Sequential regulatory activity prediction across chromosomes with convolutional neural networks. *Genome research*, 28(5):739–750, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12009–12019, 2022.

Maston, G. A., Evans, S. K., and Green, M. R. Transcriptional regulatory elements in the human genome. *Annu. Rev. Genomics Hum. Genet.*, 7:29–59, 2006.

Pruthi, D., Gupta, M., Dhingra, B., Neubig, G., and Lipton, Z. C. Learning to deceive with attention-based explanations. *arXiv preprint arXiv:1909.07913*, 2019.

Takahashi, H., Kato, S., Murata, M., and Carninci, P. Cage (cap analysis of gene expression): a protocol for the detection of promoter and transcriptional networks. *Gene Regulatory Networks: Methods and Protocols*, pp. 181–200, 2012.

Vaishnav, E. D., de Boer, C. G., Molinet, J., Yassour, M., Fan, L., Adiconis, X., Thompson, D. A., Levin, J. Z., Cubillos, F. A., and Regev, A. The evolution, evolvability and engineering of gene regulatory dna. *Nature*, 603 (7901):455–463, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Yuan, L., Chen, D., Chen, Y.-L., Codella, N., Dai, X., Gao, J., Hu, H., Huang, X., Li, B., Li, C., et al. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021.

Zhou, W., Sherwood, B., Ji, Z., Xue, Y., Du, F., Bai, J., Ying, M., and Ji, H. Genome-wide prediction of dnase i hypersensitivity using gene expression. *Nature communications*, 8(1):1038, 2017.

A. 1D-Swin Block

The input sequence $\mathbf{x}=[x_0,x_1,...,x_n]$ is partitioned into non-overlapping local windows $\mathbf{w}=[w_0,w_1,...]$, with each window w_i comprising a subset of the sequence $[x_j,x_{j+1},...,x_{j+k-1}]$, given a pre-defined window size k. The first Multi-Head Attention (MHA block), as depicted in Figure 1(b), processes each window w_i to generate $w_i'=[x_j',x_{j+1}',x_{j+k-1}']$. This transformation encapsulates local dependencies within each window.

Cross window dependencies are captured through a rolling operation, R, which shifts all tokens by a specific distance, t. Elements shifted beyond the final position are reintroduced at the start, resulting in a new sequence, $R_t(\mathbf{x}')=[x_{n-t+1}',...,x_n',x_0',x_1',...,x_{n-t}']$. These shifted elements are reassembled into new windows, denoted as $R(\mathbf{w}')$, where each $R(\mathbf{w}')i$ contains $[x_{j-t}',x_{j+1-t}',x_{j+k-1-t}']$. A second MHA block is then applied to each $R(\mathbf{w}')_i$, after which the inverse rolling operation, R^{-1}, is used to restore the original sequence order.

The total computation for each window, illustrated in Figure-1(b), follows equations 4,5,6:

$$\mathbf{w}_i'=f_{\text{MHA}_1}(\mathbf{w}_i) \tag{4}$$

$$R(\mathbf{w}')_i=R(f_{\text{concat}}(\mathbf{w}_1',\mathbf{w}_2',...))_i \tag{5}$$

$$\mathbf{w}_i''=R^{-1}(f_{\text{MHA}_2}(R(\mathbf{w}')_i)) \tag{6}$$

Given $\mathbf{x}\in\mathbb{R}^{n\times d}$, the spatially reduced token set, $\mathbf{h}\in\mathbb{R}^{\frac{n}{2}\times 2d}$, is computed using a token merging operation, which concatenates pairs of tokens and passes them through a linear layer:

$$\mathbf{h}=f_{\text{Linear}}(f_{\text{PairConcat}}(\mathbf{w}''))=f_{\text{1dSwin}}(\mathbf{x}) \tag{7}$$

B. Architecture Details for Reference Models

Figure 5 shows the overall architecture of reference models. Note that Conv1d represents the 1-d convolutional layers with residual links. Relative Positional Embedding is used in Transformer blocks.

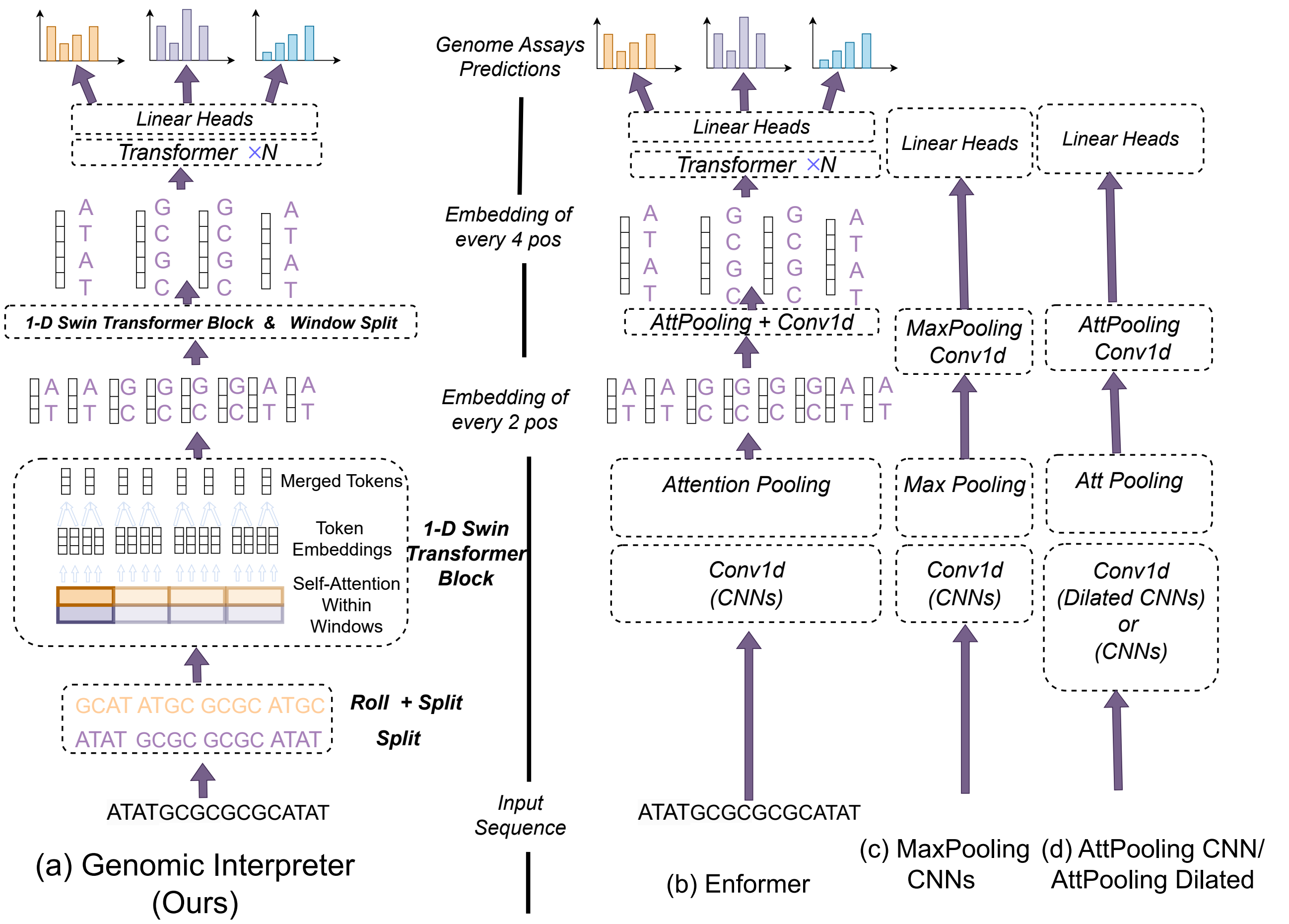

Figure 5. (a) is the architecture of Genomic Interpreter, where no pooling layer is used. (b) shows the architecture of Enformer, which uses attention pooling and CNNs for reducing the spatial dimension and increasing the hidden dimension of each token, finally a transformer layer is appended to aggregate information. (c) Max pooling CNNs use the max pooling layer and remove the transformer layer at the end. (d) Attention pooling CNNs and dilated CNNs use attention pooling layers with normal CNNs or Dilated CNNs.

C. Analysis of Attention Weights of 1D-Swin

C.1. Per-Window View

Each layer of 1D-Swin has multiple windows. While windows at the same layer share the same attention block pairs, these windows output distinct attention weights, highlighting the variety of contextual relations captured within the same layer. The following shows the attention weights obtained at the sixth layer, providing a detailed view across both window and head dimensions.

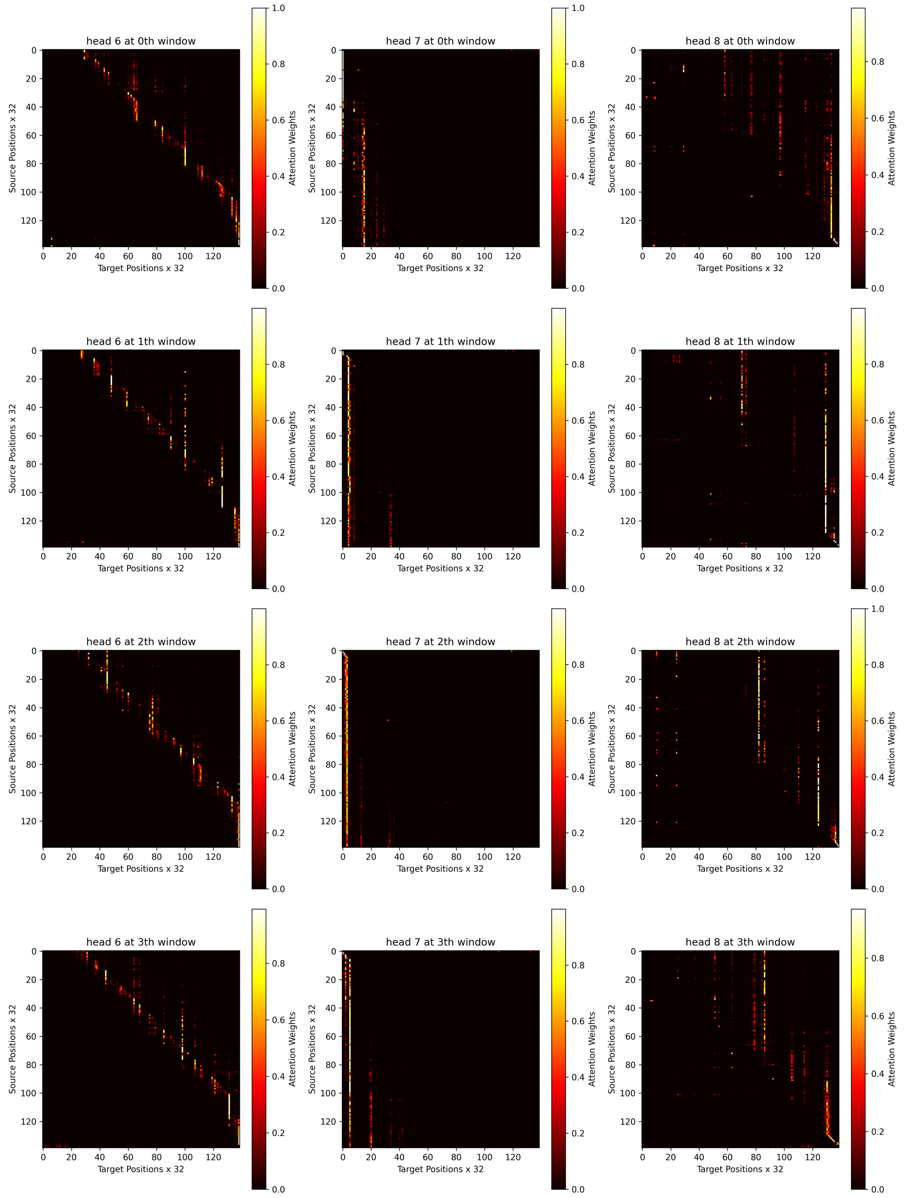

Figure 6. Layer 6 has a window size of 140 tokens, each token corresponds to 32 nucleotides, there are hereby $4 = f_{\text{floor}}(\frac{17712}{32*140})$ windows. MHA block pairs share parameters across different windows and are trained to look for certain patterns at different distances, but provide a distinct map providing different sequences within each window.

C.2. Zoom-in View

The attention weights can also be visualized in a zoom-in view, showing the dependency between tokens in more detail.

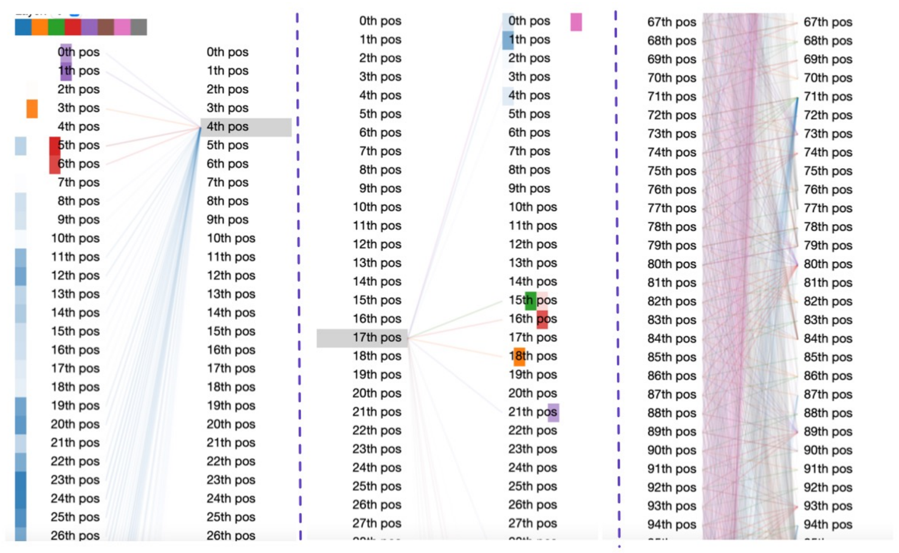

Figure 7. A zoom-in view of how individual tokens attend to each other

D. Training Tricks

Batch Norm is not used in the model, we apply the layer norm within all the models. This was chosen by considering the potential of numerical instability of batch normalization on the small batches necessitated by GPU memory limitations. Further, removing batch norm eliminates the need to synchronise the operation across multiple GPUs in the distributed training setting.