Deep Multimodal Representation Learning to Integrate Natural Language Processing with Genomic Interval Data for Tailored Biomedical Discovery

A Technical Report submitted to the Department of Biomedical Engineering

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Peneeta Ann Wojcik

Spring, 2024

Technical Project Team Members

Caitlyn Fay

Lilian Jones

Zachary Mills

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Timothy Allen, Ph.D., Department of Biomedical Engineering,

Nathan Sheffield, Ph.D., Center for Public Health Genomics

Deep Multimodal Representation Learning to Integrate Natural Language Processing with Genomic Interval Data for Tailored Biomedical Discovery

Caitlyn Fay, Lilian Jones, Zachary Mills, Peneeta Wojcik, Nathan LeRoy[1,2], Nathan Sheffield[1,2]

[1] Department of Biomedical Engineering

[2] Center for Public Health Genomics

Abstract

The amount of epigenomic data generated through experiments such as ATAC-seq and ChIP-seq has exponentially increased due to advanced sequencing technologies. These data are summarized in the form of Browser Extensible Data (BED) files, which are text files where each line represents a specific genomic region. These data are challenging to analyze due to their high dimensionality and structure. Promising approaches to extract relationships from these data and determine which genomic regions are similar across many different studies are deep learning models and natural language processing. We sought to generate a set of genomic regions in the form of a BED file based on a user-entered query. Four deep learning models were developed: a Text2BED, direct encoder, diffusion model, and transformer. Each was trained on BED files and associated text files from publicly available databases. All models successfully generated an output consisting of or relating to a BED file; however, more training data is needed. Once trained on more data and further validated, these models will inform researchers on which genomic regions are closely related to a disease or cell type they are interested in, expediting the research process.

Keywords: Deep learning, natural language processing, genomics, ATAC-seq, ChIP-seq

Introduction

The amount of data from ATAC-seq and ChIP-seq experiments has exploded over the past 10 years, increasing exponentially as sequencing technologies continue to improve. This has created a clear demand for complex models to understand the genomic relationships encoded in this large volume of data [1]. One factor driving the increase in sequencing data is that cells with the same DNA can have vastly different phenotypes. Sequencing can be used not just for sequencing a human genome, but to measure these phenotypes. The study of external modifications to DNA that create these various phenotypes is known as epigenomics. Epigenomic signals vary based on cell type and there are important considerations to be made in analyzing epigenomic data. Failing to adjust studies for cell-type heterogeneity can limit the accuracy and sensitivity to locate these modifications [2].

Multimodality representation learning is a deep learning approach that embeds information from two or more input types into a low dimensional vector representation[3]. This practice of embedding information in a vector space is used in natural language processing applications. One example is in search engines where embeddings are used to map search queries to images or webpages[4]. The Word2Vec model developed by Google engineers learns vector representations of words. The vectors represent the meaning of each word based on surrounding words[5].

Deep learning and natural language processing techniques have been successfully applied to genomics data. One model, called Region2Vec, creates vector representations of genomic regions[6]. This is an adapted Word2Vec approach where genomic data is considered as a text document and each region represents a word. Another model called StarSpace embeds genomic regions with associated metadata for use in information retrieval tasks[1].

The focus of this project is to build generative models using prior models to embed both natural language and genomic regions. Generative models are increasingly being integrated into search engines and have many applications in image and text generation[7]. These models can be used to generate new genomic data that reflect relationships based on original data. Four different models were developed: a Text2BED neural network, direct encoder, diffusion model, and transformer. Each model creates

vector representations of BED files and associated text descriptions and outputs a set of genomic regions. The use of both BED and text embeddings allow models to predict genomic regions from text. It is hypothesized the implementation of four deep multimodal representation learning algorithms, including Text2BED, direct encoder, diffusion model, and a transformer, will greatly enhance the efficiency of generating relevant genomic region sets from biomedical data. These models can enable a more holistic approach to epigenomic analysis and research. Through further comparison between models, one can be focused on to deepen its accuracy and performance.

Certain design constraints must be taken into account during this project. First, running these models locally on computers requires significant computational resources, like power and memory. To mitigate this issue, models will be trained using Rivanna, a High-Performance Computing (HPC) system from the University of Virginia. This gives access to more memory and computing power, which will decrease the time required to train the model. There still exists some storage issues within Rivanna though, which is important to consider. While there is a plethora of data provided from the Sheffield Lab, there is limited time to train models on all of this data. It is assumed that this data is of good quality and chose significant file pairs for the most accurate model training possible in the given time span. Lastly, each approach taken during this project is complex, which can limit the ability to interpret results. The models produce outputs in different forms, which is important to consider.

Model Architectures

Text2BED

The Text2BED model is composed of two functions: create_backend and generate_bed_file. In the first function, create_backend, the inputs are: 1) the path to the region2vec model previously designed and fully trained by the DataBio lab on Hugging Face and 2) an index path from the user's local environment. The region2vec model is loaded and the tokenized region embeddings from the universe are imported. The universe imported consists of 1,063,880 embedded regions. The region embeddings are then accessed through a PyTorch tensor and thus converted from the embedding variable type to a NumPy array for computation. The Hierarchical Navigable Small World (HNSW) backend is then created using the local index path to prepare to store the region embeddings in a backend so that the hierarchical kNN search can be performed. The create_backend function then returns the subsequently generated HNSW backend to be used in searching for the closest regions to a given NL text query.

The HNSW algorithm is a method used in machine learning for efficient nearest neighbor search in high-dimensional spaces. HNSW organizes these points into a hierarchical graph structure where each point is connected to other nearby points and levels representing levels of resolution. When given a query, HNSW efficiently navigates through this graph, starting from coarse levels and gradually refining the search at finer levels until the nearest neighbors are found[8].

In the backend, payloads are created that keep track of the original NL region representations (chr number, start bp, and end bp) from the universe. This backend is then taken as an input into the second function, generate_bed_file, along with the previously mentioned NL text query, a path to a desired pre-trained NN to use to compare similarities between embedding vectors, and an integer value for the desired length of the resulting generated BED file. The function imports a popularly used pre-trained sentence transformer in order to convert the NL text query into a text embedding, representing the text as a vector of numbers. The vector to vector comparison NN and the HNSW backend outputted by the previous function are then used to build a search interface to perform the kNN search. So, taking the NL text query, the embedding of this text is compared to the region embeddings of the Region2Vec universe and then the previously user-specified number of regions to be returned are found using HNSW in order of most to least close in similarity to the text query. From the returned N closest region embeddings, the information contained in their corresponding payload is then written to a new, generated BED file.

Direct Encoder

The Direct Encoder is a neural network for representation learning. The data must be represented numerically to be fed as input to the model. To achieve this, term frequency-inverse document (TF-IDF) frequency is used to transform text vectors into usable input. This is a statistical measure that finds the significance of a word with respect to a large text corpus[9]. For use in this model, the text corpus used was provided by the Sheffield lab to ensure consistent training amongst all models. The text data includes the descriptions of correlated BED files. These text-BED pairs are used to train the model. The text descriptions are vectorized using the tfidf vectorizer from Scikit Learn in Python[10].

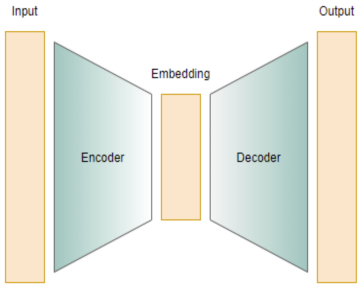


Figure 1. Architecture of Direct Encoder

The actual model definition includes an encoder and decoder portion. The encoder layer compresses the vectorized text input data to highlight its most relevant features. This encoder is made of five linear layers. The decoder then decompresses the data by reconstructing it to fit the size as the genomic vocabulary. This structure is visualized in Figure 1[11].

The vector output from the direct encoder model serves as a numerical representation for genomic regions relevant to the text entered.

To test the efficiency of the model, the loss is calculated by measuring the mean squared error between the predicted and actual values using Pytorch. The actual values were found using the ITTokenizer, which tokenizes genomic regions found in BED files.

Lastly, the model is trained using an optimizer. A dataset class is created that is designed to hold text vectors and bed bit vectors. Next, a data loader is created from the data set to handle batching and shuffling of the data set. Throughout 10 epochs, the data is fed through the data loader to create varying batches of data, which is then passed through the model. The MSE loss is calculated for each batch and the optimizer adjusts model parameters to improve this loss over time. The loss is tracked to ensure model accuracy is improving.

Diffusion Model

Diffusion models are the current state of the art for new image generation[12]. These models work by adding Gaussian noise to the training data until it is unrecognizable from pure noise, and then learning to

predict the noise and remove it to recover the data. After training, randomly sampled noise can be passed to the model and noise will be removed resulting in new data according to an input or the training examples[12]. This general structure can be seen in Supplementary Figure 1 below. These types of models are the basis for the most popular image generation models today such as DALL-E and Stable Diffusion[13].

Data Transformation

Diffusion models are used to generate images and as such require images to train on. However, the goal of this model is to generate region sets and as such the data being used to train the model are BED files. In order to adapt diffusion models to work with BED files, there needs to be a method for converting between BED files and images and vice versa. In order to accomplish this, the ITTokenizer from the geniml package is used to tokenize the region sets according to a universe comprised of 1,063,878 candidate cis-Regulatory Elements (cCREs) in the GRCh38 reference genome from the ENCODE project[14,15]. This tokenizer will return which of the regions in the universe are contained in the specific BED file. This is then transformed into a 1,063,878 dimensional binary NumPy array where each value corresponds to a region in the universe. If the index contains a 0 then the BED file does not contain the associated region, and if it contains a 1, then that region is present in the BED file. This array is reshaped into a 761 by 1398 matrix to represent the first channel of a 3-channel image. The other channels are made up of matrices of the same size containing all zeros. These 3 channels are together transformed into a PIL image to be used in the model. This process is performed in reverse for the output images from the model. The images are transformed into a 3-channel matrix where the first channel is the target data. This matrix is transformed into an array and the indexes of each 1 value are stored in a separate list. The corresponding region of each index is then compiled into a new BED file.

In order for the model to be able to generate new data according to user input, it must also account for the associated text descriptions of each region set. To do this, the text description of the files are turned into text embeddings using the all-MiniLM-L6-v2 sentence transformer found on HuggingFace[16]. These text embeddings are then paired with the corresponding transformed images. Once the data has been transformed into image-embedding pairs as seen in supplementary Figure 2, it is ready to be used in the training of the model.

Forward Process/Addition of Noise

Now that the data is ready, the first step in the model is to add noise to the training images. This is done iteratively across multiple steps according to a noise schedule. This is a Markov process where each step depends only on the one immediately preceding it. The noise schedule determines how much sampled Gaussian noise is added to the image at each time step and it ensures that an appropriate amount of noise is added so that the final image ends as a Gaussian distribution with a mean of zero and a fixed variance. This model utilizes a linear noise schedule where the same amount of noise is added at each step. This process of adding noise can be seen below in Figure 2. The specific noise for each time step can actually be computed independently, if the closed form of the mean and variance is precomputed based on the cumulative process.

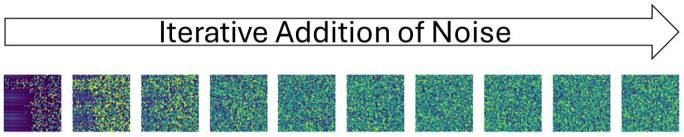


Figure 2: Diagram of the iterative addition of Gaussian noise to the data images to prepare for model training.

Backward Process/Noise Removal

The backward process of the model works to predict the noise and iteratively remove it. It does this through the use of a U-net architecture. This architecture utilizes a series of convolutional, down sampling, and up sampling layers that mimic the shape of the letter "U" where the output is the same dimension as the input. The model follows an encoder-decoder structure. The encoder portion performs repeated convolutional layers on the input image, and then down sampling using a max pooling layer. This process is repeated for a total of 5 downsamplings where at each layer the image becomes smaller, but has more channels. After the encoder layer, the decoder conducts the reverse operation. It performs repeated convolutional layers followed by an upsampling layer where the dimension of the image is increased and the number of channels is decreased. This is also repeated for a total of five up samples to result in a new image that is the same dimension of the input. Each layer of the encoder and the decoder are connected by concatenating some of the encoder features with the decoder features to capture more information. An overview of the U-net model structure can be seen in Figure 3 below. This whole process is done at each step of the iterative addition of noise so that the model is able to perform denoising across various noise levels.

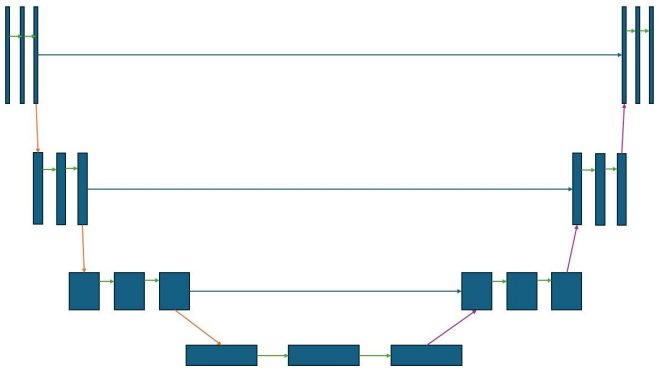


Figure 3: Diagram showing the general structure of the U-net architecture, where the green arrows represent convolutional layers, the orange arrows represent max pooling down sampling, and the purple arrows represent up sampling.

Training and Loss Function

This model was trained using BED files and associated text descriptions from the ENCODE project[15]. This data was collected into a data loader as associated pairs. The model was then trained over 10 epochs using the Adam optimizer and l1 loss. 10 epochs was chosen by determining when the loss naturally plateaued and stopped noticeable decreasing.

Transformer

The transformer model is a neural network model first proposed as an improvement to the recurrent neural network, which was the prior state-of-the-art for sequence modeling problems[17]. It has an encoder-decoder structure with both blocks using attention mechanisms to capture context of sequences. The general structure is shown in Figure 4. This model has previously been used for text data and is the primary model used in ChatGPT[18].

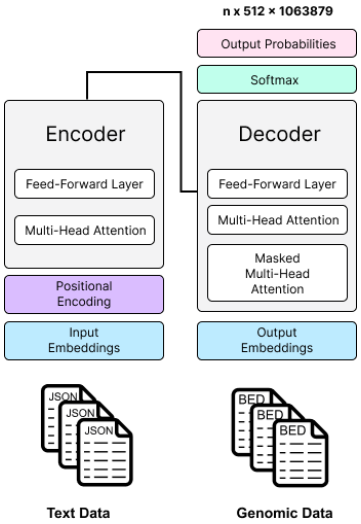


Figure 4. Transformer

Attention Mechanism

The primary innovation of the transformer model is its use of an attention mechanism to determine the relative context for each token in a sequence. Self-attention is used so the model can capture the context of each word it receives as input. To do this, the tokenized input is converted into three separate vectors: a query, key, and value[19]. The query vector is the word whose attention is being calculated, the key is used to represent every other word in a sequence to match against the query, and the value is the result of calculating the attention between query and key vectors. These calculations can be parallelized. Using self-attention preserves the meaning of each word throughout a sequence, whereas RNNs cannot preserve this meaning across a lengthy sequence.

Positional Encoding

The attention mechanism does not include positional information, meaning different tokens can be scrambled and the output of the model would be the same. Sinusoidal positional encodings are added to the original tokens to the input embeddings to reflect relative distances between tokens. A sinusoidal function is used to give higher values for nearby tokens and a smooth decay of values for tokens that are further away[20].

In a typical transformer architecture, positional encoding is present below both the encoder and decoder blocks to preserve sequence information. For this application, only the natural language text input positional encoding was included. No positional encoding was used for the BED files because the order of regions does not matter, only their presence does.

This transformer model is designed to encode a text query and decode it into a series of genomic regions in the form of a BED file. BED files and associated JSON metadata files were retrieved from the ENCODE project, a publicly-available repository that holds ATAC-seq and ChIP-seq data[21]. JSON files were parsed to retrieve the description field, which contained a paragraph about the study the data was generated from.

Both BED and JSON files were tokenized separately. The BED files were tokenized using the ITTokenizer method available in geniml[22]. This tokenizer uses a universe file, or a set of all possible genomic regions, to assign an index to each genomic region in the BED file. The focus of this project is epigenomic signals, therefore the universe file used here was the set of all human cCREs available through the ENCODE project[15]. This contained all possible sites involved in histone modification or CTCF-binding. Unknown tokens were filtered out by excluding the token number "1063878" to further reduce dimensionality.

JSON files were tokenized using the pretrained BERT base-cased model available on HuggingFace[23]. This model was pretrained on a large corpus of English text data and uses WordPiece tokenization to break up natural language into piecewise components[24] (Figure 5).

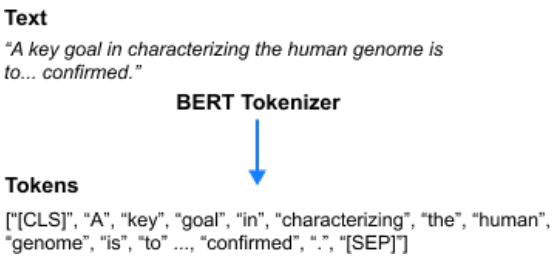


Figure 5: Example of WordPiece tokenization from BERT tokenizer.

Both the BED and text input tokens were padded to a fixed value of 512 so input size would be consistent. Transformer models often use dimensions of either 512 or 1024 because of performance issues; the time complexity of the self-attention mechanism is quadratic based on input length, creating a bottleneck in training[25]. Because of this, only 512 regions from each BED file were sampled to begin preliminary training. The transformer was trained using the Adam optimizer and cross entropy loss.

Methods and Materials

All models were implemented using libraries provided by PyTorch, HuggingFace, and the geniml package developed by the DataBio lab at the University of Virginia[26–28]. The code for this project is available in the following repository: https://github.com/databio/bme-capstone-2023.

Results

Each of the four models generated different output data relevant to specific genomic regions based on a text query. Specifics are detailed below.

Text2BED

The original goal of this model is achieved, a NL text query is inputted into the model and a new, generated BED file is returned. The validation of this model proved to be a challenge, as there are no current methods

for a numerical or more empirical method of proving the validity of the model's results. What was decided on, however, was to test out a few example queries. The queries: “kidney, human, cancer”, “pediatric, lung, inflammation”, “lymphoma, human, prognosis” were passed through the model and the resulting BED files were assessed by then entering some of the top regions in the file into IGV. The gene corresponding to the region outputted in the BED file was then searched in PubMed for relevant literature associated with the gene. Important findings and associations with the gene were noted in a table, indicating whether or not they were relevant to the original NL query (Table 1). This gives a foundational indication of the model’s ability to create a BED file that actually is related to a text query but future steps for this model primarily include finding a less brute-force and more computational way of assessing the model’s performance.

Table 1. Validation of regions outputted in generative BED file created from the NL query: “human, kidney, cancer”

Region from Generative BED File	Corresponding Gene	Notes
chr8 70054387-70054650	PRDM14	Promotes malignant phenotype in cells, Tumorigenicity, cancer initiation (lung, testis, kidney, breast)
chr19 16894195-16894531	CPAMD8	Glaucoma
chr16 1981418-1981735	TBL3	Polycystic kidney disease, Chronic kidney disease
	NOXO1	Tumorigenesis, Cancer progression
chr18 13279033-13279208	LDLRAD4	Gastrointestinal stromal tumors, Polycystic kidney disease, Chronic kidney disease
chr17 63694673-63695005	MAP3K3	Promotes tumor growth, Carcinoma progression
chr19 6662730-6663080	TNFSF14	Reversing immunosuppressive tumor microenvironment

Direct Encoder

The model generates numerical representations of genomic regions based on text input. Quantitative results showing the model's accuracy is shown through the progression of the loss function outputs. There was a general downward trend in loss, indicating that the model performance got better through the parameter adjustments made by the optimizer. The more epochs run through the training loop, the better the model performed. This is also represented by the loss curve seen in Figure 6.

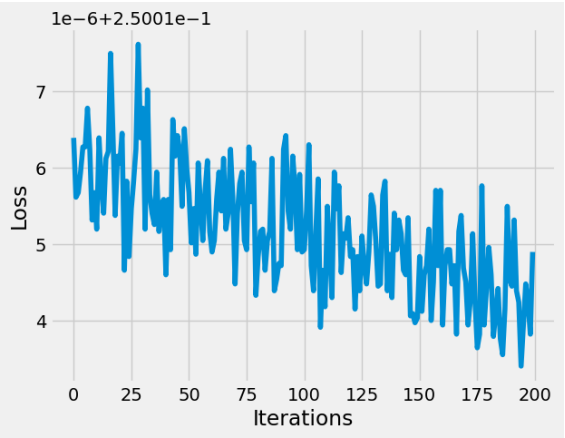


Figure 6. Loss curve of direct encoder mode

When run with ten epochs and a batch size of 5, the model expressed a loss closer to 0.25001. This MSE score indicates that the average squared difference between the predicted and actual values is relatively low,

indicating a fairly accurate model. This also means it has further potential for improvement with further training.

Diffusion Model

After training, the diffusion model experienced a large reduction in loss as seen in the loss curve shown in Figure 7 below. The loss leveled out at a value of around 0.1 which means the model was accurately able to reproduce training data to a relatively high level.

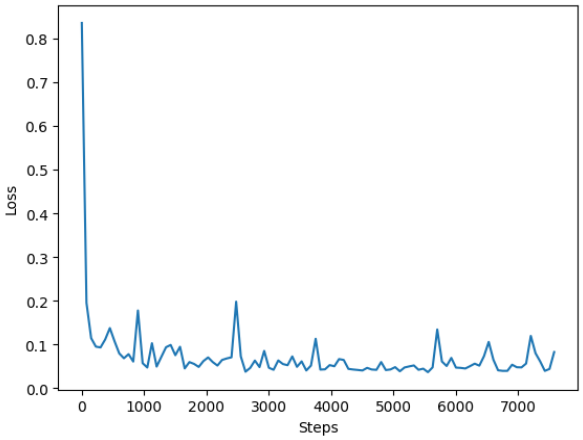


Figure 7: Loss curve for the diffusion model

The model is able to generate novel images from user text input that resemble the original training images. These images are converted into region sets contained in BED files. Through the process of training it can be qualitatively observed that the quality of the output images is becoming closer to the original training data as seen in Figure 8 below. The accuracy and biological relevance of these new region sets is yet to be further evaluated. Due to the large computational complexity of the model, only a small number of training examples were able to be used in this project. For further, more improved work, the number of training instances could be increased which would likely result in higher quality data generation. It would also be beneficial to determine a numerical metric to determine the accuracy and biological relevance of the generated region sets.

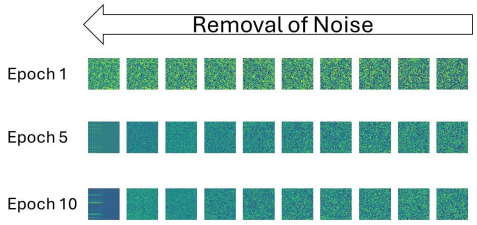


Figure 8: Visualization of the removal of noise for epoch 1, epoch 5, and epoch 10.

Transformer

Each component of the model architecture was unit tested to verify whether data was being passed correctly from the encoder to the decoder portion. The encoded output resulted in a tensor of size 100 by 512 by 512 (sample number by input dimension by model dimension). Initially, entire BED files were passed to the decoder model, however resulted in a dimensionality mismatch between the encoder and decoder portions.

After passing the first training example, the output of the model resulted in a set of output probabilities in the shape 100 by 512 by 1063879 (sample number by model dimension by total number of genomic regions), where the probabilities represent how likely a certain genomic region is present. Future work is required to train the model on Rivanna using more training examples.

Discussion

The goal of this study was to create four distinct deep-learning models that generate relevant genomic region sets to a user-entered search that can then be used in experiments for novel biomedical analysis. While four models have been developed, there are limitations and future work that can be done on them. The main limitation of the text2BED model is although it is functioning properly and can produce a novel BED file, there is no quantitative way of validating if the resulting BED file is clinically associated with the initial user NL query. On the other hand, the main limitation of the other three models is limited training data. Only 100 BED files were used in model training, so future training with more text-BED pairs would increase their accuracy. While the direct encoder showed promising results through its small loss metric, the model currently outputs a number representation of genomic regions. Future work should focus on translating these numeric outputs into newly generated BED files. The diffusion model appears to work efficiently, but this model type is not typically used for data types other than images. This limits the model as its architecture may not efficiently capture patterns on data that are not images. The transformer decoder is greatly limited by the length of the input sequence due to the computational complexity of the self-attention mechanism. This is an active area of research, and transformer-based models are currently being optimized for memory usage. One such approach is called Flash Attention, which restructures how query, key, and value vectors are calculated[5].

Though preliminary training has been conducted on these models, more rigorous validation is needed. The next step after increasing the amount of training data is to verify whether generated genomic regions are valid. We propose the usage of a Jaccard index for preliminary verification to determine similarities probabilistically between an actual and a generated BED file[29]. To conduct this analysis, a BED file will be chosen as the ground truth set. A query matching this BED file will be given to the model, and a Jaccard score will be found by taking the ratio of the number of genomic regions in both files to the total regions in the union of both files.

Conclusion

All model architectures were successfully implemented and some generated BED files, however more training and optimization is needed. Future directions include expanding the training data to all data on the ENCODE database and validating generated BED files using a Jaccard score or other quantitative metric. The development of these models could have a significant impact on biomedical research. Each model

offers a tool for analyzing epigenomic data with respect to text-entered queries. This allows for the investigation of specific regions that have proven to be associated with specific biological processes. Using tools such as this can perpetuate the advancement of biomedical research by leading to more targeted studies and treatment options.

Acknowledgments

We thank Dr. Nathan Sheffield, Nathan LeRoy, and Claude Hu for their mentorship during this project.

References

1. Gharavi, E. *et al.* Joint representation learning for retrieval and annotation of genomic interval sets. 2023.08.21.554131 Preprint at https://doi.org/10.1101/2023.08.21.554131 (2023).

2. Qi, L. & Teschendorff, A. E. Cell-type heterogeneity: Why we should adjust for it in epigenome and biomarker studies. *Clin. Epigenetics* **14**, 31 (2022).

3. Manzoor, M. A. *et al.* Multimodality Representation Learning: A Survey on Evolution, Pretraining and Its Applications. *ACM Trans. Multimed. Comput. Commun. Appl.* **20**, 1–34 (2024).

4. Bengio, Y., Courville, A. & Vincent, P. Representation Learning: A Review and New Perspectives. Preprint at http://arxiv.org/abs/1206.5538 (2014).

5. Dao, T., Fu, D. Y., Ermon, S., Rudra, A. & Ré, C. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. (2022) doi:10.48550/ARXIV.2205.14135.

6. Gharavi, E. *et al.* Embeddings of genomic region sets capture rich biological associations in lower dimensions. *Bioinformatics* **37**, 4299–4306 (2021).

7. Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models. https://arxiv.org/abs/2103.04922.

8. Hierarchical Navigable Small Worlds (HNSW) | Pinecone. https://www.pinecone.io/learn/series/faiss/hnsw/.

9. Simha, A. Understanding TF-IDF for Machine Learning. *Capital One* https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/ (2021).

10. sklearn.feature_extraction.text.TfidfVectorizer. *scikit-learn* https://scikit-learn/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html.

11. Autoencoders & Variational Autoencoders. https://www.andrew.cmu.edu/user/xihuang/blog/ae_vae/ae_vae.html.

12. Dhariwal, P. & Nichol, A. Diffusion Models Beat GANs on Image Synthesis. Preprint at http://arxiv.org/abs/2105.05233 (2021).

13. Rombach, R., Blattmann, A., Lorenz, D., Esser, P. & Ommer, B. High-Resolution Image Synthesis with Latent Diffusion Models. Preprint at http://arxiv.org/abs/2112.10752 (2022).

14. Rymuza, J. *et al.* Methods for constructing and evaluating consensus genomic interval sets. Preprint at https://doi.org/10.1101/2023.08.03.551899 (2023).

15. The ENCODE Project Consortium *et al.* Expanded encyclopaedias of DNA elements in the human and mouse genomes. *Nature* **583**, 699–710 (2020).

16. sentence-transformers/all-MiniLM-L6-v2 · Hugging Face. https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2 (2024).

17. Vaswani, A. *et al.* Attention Is All You Need. Preprint at http://arxiv.org/abs/1706.03762 (2023).

18. Radford, A. *et al.* Language Models are Unsupervised Multitask Learners.

19. Helene_k. Attention and Transformer Models. *Medium* https://towardsdatascience.com/attention-and-transformer-models-fe667f958378 (2022).

20. Ellmen, I. Understanding positional encoding in Transformers | Oxford Protein Informatics Group. https://www.blopig.com/blog/2023/10/understanding-positional-encoding-in-transformers/ (2023).

21. Luo, Y. *et al.* New developments on the Encyclopedia of DNA Elements (ENCODE) data portal. *Nucleic Acids Res.* **48**, D882–D889 (2020).

22. Tokenization - BEDbase: a unified platform for genomic regions. https://docs.bedbase.org/geniml/tutorials/tokenization/.

23. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (2018) doi:10.48550/ARXIV.1810.04805.

24. google-bert/bert-base-cased · Hugging Face. https://huggingface.co/google-bert/bert-base-cased (2024).

25. Keles, F. D., Wijewardena, P. M. & Hegde, C. On The Computational Complexity of Self-Attention. (2022) doi:10.48550/ARXIV.2209.04881.

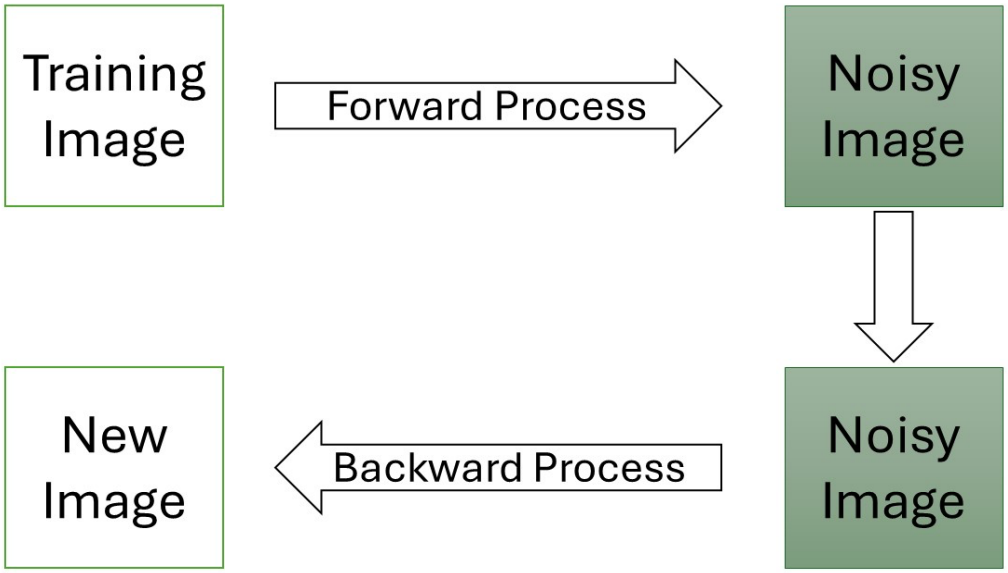
26. Paszke, A. *et al.* PyTorch: An Imperative Style, High-Performance Deep Learning Library. (2019) doi:10.48550/ARXIV.1912.01703.

27. Wolf, T. *et al.* HuggingFace's Transformers: State-of-the-art Natural Language Processing. (2019) doi:10.48550/ARXIV.1910.03771.

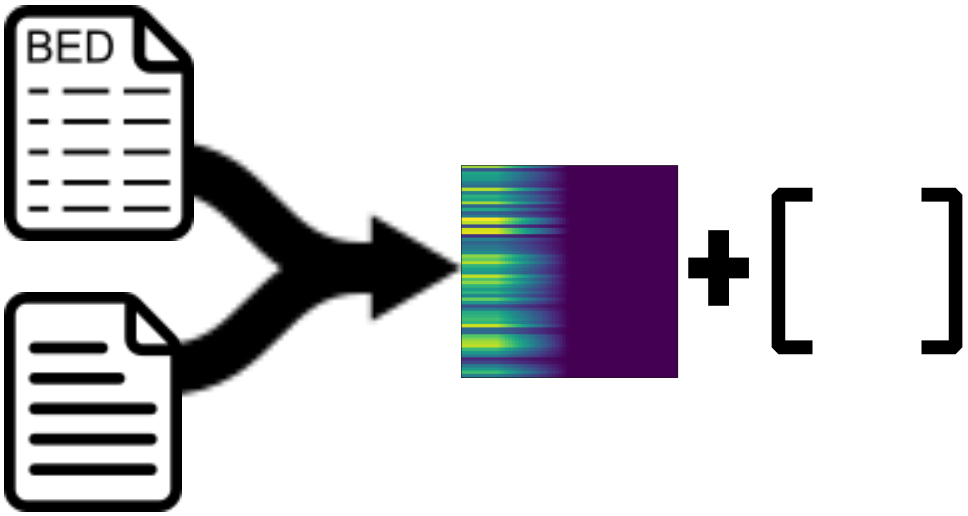
28. Geniml - BEDbase: a unified platform for genomic regions. https://docs.bedbase.org/geniml/.

29. Chandra, N. K. & Bhattacharya, S. Dependent Bayesian multiple hypothesis testing. in *Handbook of Statistics* vol. 47 67–81 (Elsevier, 2022).

Supplementary Materials



Supplementary Figure 1: General overview of diffusion model architecture.



Supplementary Figure 2: Diagram showing the data transformation from BED and text files to image and embeddings