Project Checkpoint - Can a Self-Attention Model Capture Biological Context of Sequences to Classify Enhancer Regions?

ICS 675

William Little (30286169) November 26, 2025

Progress Reflection

The first step I took was downloading the large *human_genome_ensembl* dataset [1] that contains 155,000 sequences, split into train and test sets, both with 50/50 positive-negative split. I performed exploratory data analysis to observe aggregate and individual pattern traits such as distribution of sequence lengths, invalid characters, and empty files. Approximately 5% of the data was removed due to really short sequences (1-30 bp) or containing bad characters.

The next step was to decide on a method for embedding the k-mers into fixed-length tokens for input to the attention model. Given the resource and time constraints of this project, I use the pretrained k-mer embeddings from the dna2vec model [3], which applies the word2vec algorithm to learn biologically meaningful vector representations for k-mers of length $k \in \{3, \ldots, 8\}$. This allows to investigate the model performance with varying k-mer length.

Then I had to design the model architecture. My hypothesis aims to test a light-weight attention mechanism with a classification head to make predictions. I implemented the architecture described in Figure 1 in Python with the PyTorch framework. I preprocessed the valid DNA sequences into a PyTorch data file with their associated labels, and then process the embeddings on-the-fly to save memory during training. The model code is accessible in my GitHub repository linked in the following section under EnhancerAttention.py.

The last step was training a small preliminary model for initial benchmarks and ensure all code is functioning properly before training large models.

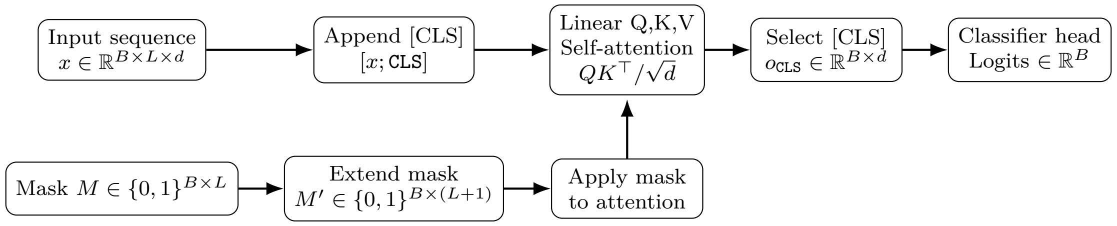
Input sequence $x \in \mathbb{R}^{B \times L \times d}$
Append [CLS] $[x; \texttt{CLS}]$
Linear Q,K,V Self-attention $QK^\top/\sqrt{d}$
Select [CLS] $o_{\texttt{CLS}} \in \mathbb{R}^{B \times d}$
Classifier head Logits $\in \mathbb{R}^B$
Mask $M \in \{0,1\}^{B \times L}$
Extend mask $M' \in \{0,1\}^{B \times (L+1)}$
Apply mask to attention

Figure 1: Model architecture used for enhancer classification. The input sequence is tokenized and embedded, a learnable [CLS] token is appended, and a self-attention layer produces contextualized representations. The final classification is obtained by passing the [CLS] output vector through a fully connected linear head. A corresponding attention mask is extended and applied to ensure padded positions do not contribute to attention scores, allowing for variable sequence lengths.

Code Repository

All code used in this research is located at `https://github.com/williamlittle423/human_enhancer_detection`.

Preliminary Benchmarking

The preliminary benchmarking results were obtained from a trial training run using a single epoch over the full training set. The original dataset consists of thousands of individual text files, totaling 507 MB and introducing substantial storage and I/O overhead. To reduce memory usage and improve data loading efficiency, all sequences and labels were aggregated into PyTorch tensors, reducing the dataset size to 34.6 MB. During training, the peak memory footprint reached 404.8 MB, arising from the dataset, model parameters, library overhead, and other processes. This level of memory usage is well within the capacity of local machines and HPC nodes.

Runtime analysis was performed on a single CPU core of an Apple M2 processor, resulting in a total training time of 77.8 minutes for one epoch. Future experiments will leverage GPU acceleration via CUDA, which is expected to substantially reduce training time for the main experiments spanning 25–250 epochs.

Preliminary Results

To validate the correctness of the data-processing pipeline and training implementation, a preliminary benchmarking experiment was conducted with the attention-based classifier. For this initial evaluation, a lightweight model was trained for a single epoch on the full training split of the `human_enhancers_ensembl` dataset. Each training sample was therefore seen exactly once, providing a minimal but informative check of model behavior.

After training, the model was evaluated on the held-out test set using the standard metrics for binary classification. Figure 2 shows the resulting ROC curve alongside the diagonal line $y = x$, showing that the model performs substantially better than random guessing even under limited training.

The quantitative metrics are summarized in Table 1. The model achieves an AUC of 0.7834 and an accuracy of 0.7190, with precision, recall, and F1-score all around 0.71–0.72. These results indicate that the attention mechanism is already capturing meaningful sequence features despite minimal training, supporting my hypothesis that lightweight self-attention architectures can extract biologically relevant patterns from DNA sequences. These results already outperform the baseline metrics from the original GenomicBenchmarks paper [1], albeit much lower than the to-date state-of-the-art model HyenaDNA [2], achieving 89.2% accuracy. The objective is come near or surpass that accuracy.

Metric	Value
AUC	0.7834
Accuracy	0.7190
Precision	0.7229
Recall	0.7104
F1-score	0.7166

Table 1: Preliminary evaluation metrics for the single-epoch attention model on the test split.

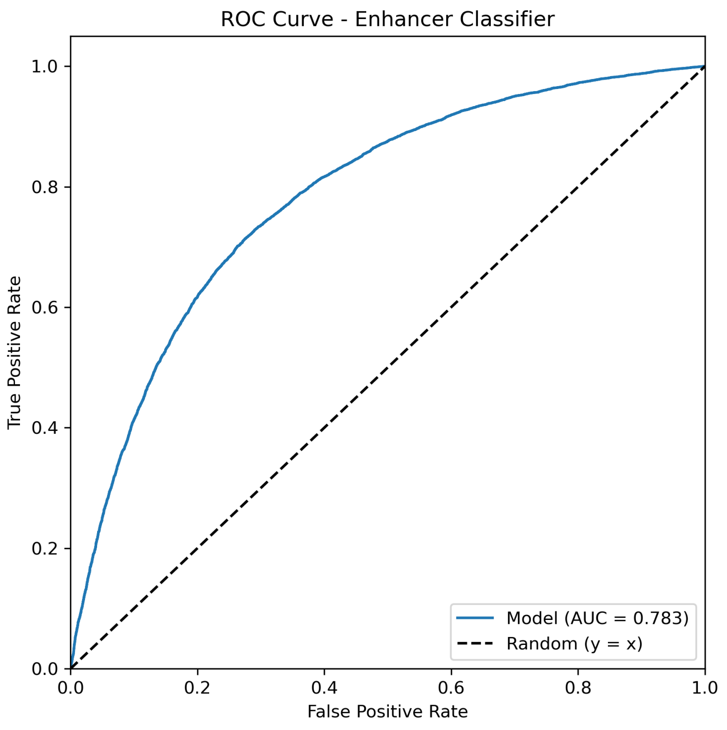

Figure 2: Preliminary ROC curve for the attention-based enhancer classifier, evaluated on the held-out test set. The diagonal line represents random-guessing performance.

Next Steps

The next phase of this project will focus on moving from preliminary benchmarking to full-scale experimentation and model refinement. The objectives include:

1. **Full Training on GPU**: Move the training pipeline to a CUDA-enabled GPU environment to substantially reduce runtime and enable training over 25–250 epochs. This will allow better optimization of the attention-based architecture.
2. **Hyperparameter Tuning:** Explore batch size, learning rate, optimizer variants, dropout, and different pooling or classification strategies (mean pooling vs. CLS token). This tuning is expected to improve generalization and stabilize training.

3. **Model Architecture Improvements:** Experiment with multi-head attention, positional encodings, 1D convolutions before attention, and regularization techniques.
4. **Biological Interpretation** Investigate the learned attention maps and embedding space to determine whether the model captures meaningful sequence motifs or known enhancer-associated patterns.

These steps will transition the project from a functional prototype to a thorough validated sequence classifier.

References

[1] Katarína Grešová, Vojtěch Martinek, Daniel Čechák, et al. Genomic benchmarks: a collection of datasets for genomic sequence classification. *BMC Genomic Data*, 24(1):25, 2023.

[2] Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Callum Birch-Sykes, Michael Wornow, Aman Patel, Clayton Rabideau, Stefano Massaroli, Yoshua Bengio, Stefano Ermon, Stephen A. Baccus, and Chris Ré. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. *arXiv preprint arXiv:2306.15794*, 2023.

[3] Phan Nguyen and Harris Georgiou. dna2vec: Consistent vector representations of variable-length k-mers. In *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM-BCB)*, pages 546–551. ACM, 2017.