ACTGap: LSTM and Transformer Decoder-Based DNA Sequence Imputation

Presented by Ashley Xu, Khoi Le, Luke Nguyen, and Romer Miranda
CSCI 1470, Department of Computer Science, Brown University, Providence, RI, USA

Project Purpose

To train and evaluate different deep learning model architectures for predicting missing gap sequences in the human genome.

Background

In biology, **DNA sequencing** refers to determining the exact order of nucleotides in a DNA fragment, using techniques like **Next-Generation Sequencing (NGS)**.

GRCh38, the latest human genome assembly, was generated by fragmenting the genome, sequencing the pieces, and then assembling them computationally.

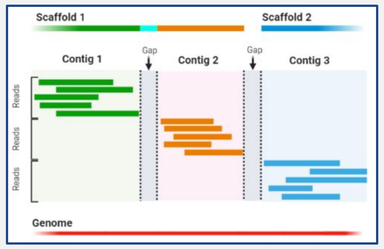


Illustration of reference genome assembly and the existence of gaps.

Despite these efforts, gaps remain in the human reference genome due to **technological limitations, unresolved regions, and highly repetitive sequences** that hinder accurate assembly into a continuous sequence. Previous tools made to fill in these gaps relied on greedy algorithms to solve the problem. However, these methods lack versatility and are unable to fully exploit all the sequencing data available. Some deep learning models are available, such as **DNABert**, but rely on heavy compute, encoder models. **Thus, we propose a LSTM and Transformer Decoder approach to reconstruct missing DNA sequences in incomplete genome assemblies.** The problem we are trying to solve is that of generation: we want our model to learn the underlying distribution of DNA sequence data and generate new sequences.

Data and Preprocessing

K-mer Tokenization

For both of our approaches, we tokenized the original DNA sequence into a sequence of overlapping k-mers, with a k-mer length of 2 and stride of 1. Each k-mer was also mapped to a unique numerical ID after generating a k-mer dictionary. Below is an example tokenization:

Original Sequence:	ACTGGGAACATTA											
K-mer Sequence:	AC	CT	TG	GG	GG	GA	AA	AC	CA	AT	TT	TA
Tokenized Sequence:	1	6	11	15	15	12	0	1	4	2	10	8

LSTM-Based Model

We used sequences from an **isolate of human chromosome 21** in GenBank fasta format, available from NCBI. Chromosome 21 is 45,090,682 bp long, but for our purposes, we used the first ~350,000 bp of the chromosome. The data was split into sequences of a certain window length, with default value of 32. After k-mer tokenization, BEGIN and END tokens were included to provide start and stop context for the model. To simulate gaps in incomplete genome assemblies, we **introduced artificial gaps through masking contiguous sequences of specified length** in either random locations or at the end of each sequence. Masking allowed the model to know exactly where to predict the next token.

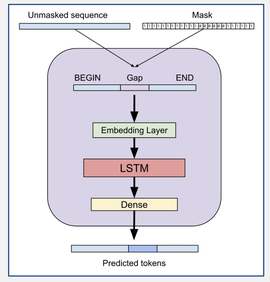
Transformer Decoder-Based Model

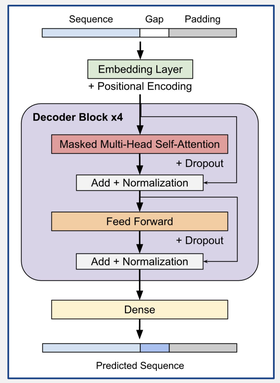
We used protein coding sequences from the CDS fasta downloaded from Ensembl. Sequences longer than 1024 bp were removed from the dataset, and padding tokens were added to the end of shorter sequences such that all input sequences were the same length. Preprocessing outputs were **context and corresponding gap vectors each padded to be the size of the largest gene in the dataset (1024).** The data was shuffled with training, and the batch size was set to 16, a reduction reflecting the Transformer's more compute heavy workflow.

Methodology

LSTM-Based Model

The model itself consists of an **embedding layer, LSTM layer,** and a **dense layer** to act as the classifier. The model was trained using the **Adam optimizer** and sparse categorical cross entropy for the loss function. The data was split into a training and test fasta file, with 80% of the data being allotted for the training data and the remaining 20% for the testing data. We **passed both the tokenized sequence as well as the mask, which consists of 0s and 1s** where 0s indicated a masked or "gap" location and 1 indicating a nucleotide being present, into the model.



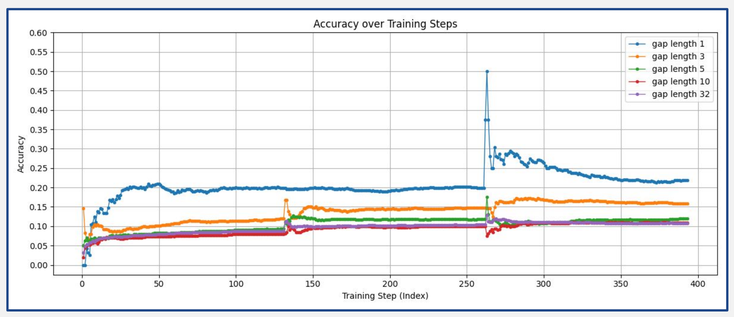


Transformer Decoder-Based Model

Our Transformer Decoder-Based Model includes a **positional encoding layer, 4 Transformer Decoder Blocks, a normalization layer, and a classifier (dense) layer** that produces logit values across the possible k-mer tokens. The Transformer Decoder Blocks each consist of a **masked multi-headed self-attention layer, 2 feed-forward layers, 2 normalization layers, and a dropout layer.** For hyperparameters, there are 8 attention heads, a dropout rate of 0.3, and an embedding size of 256. The model was trained using the **Adam optimizer** with learning rate = 0.0003 and sparse categorical cross entropy for the loss function. Loss and accuracy were calculated based on only the gap sequence.

Results

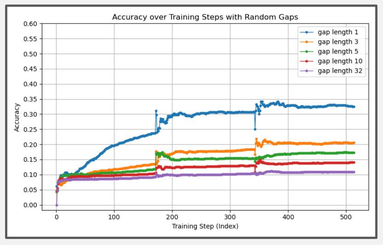
Transformer Decoder-Based Model

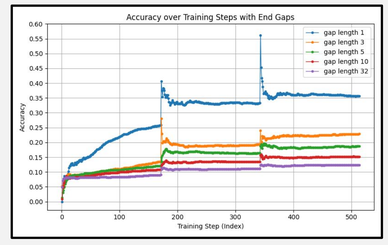


We recognized that LSTM model limitations include its inability to capture long-range dependencies, and thus we implemented a transformer decoder based model. However, compared to the accuracy in both LSTM models, we get **significantly worse performance across all gap lengths**. In the LSTM, accuracy varied noticeably between gap lengths of 5, 10, and 32, but the decoder's accuracy collapsed onto each other—indicating it did not learn the gap-specific patterns. We believe this performance stems from **insufficient training, where compute heavy decoders were penalized more heavily than LSTMs**.

Results Cont.

LSTM-Based Model





Within the LSTM model, we wanted to explore whether or not the location of the gap, as well as the gap length, affects the accuracy of our model. With random insertion of a contiguous gap, the model only examined the sequence up to the start of the gap, omitting any information after the gap. Therefore, we hypothesized that the model would better perform on data with gaps inserted at the end, as the model would have the most prior information available. The figures revealed that we were correct, as **accuracies were slightly higher when adding gaps to the end** compared to when they were randomly inserted. Additionally, **smaller gaps were expected to have higher accuracy** as the model has less to predict, and this was supported by the figures.

Example Prediction

```
Predicted: 5  4  5  15 15 15 15 15 15 15
Actual:    4  3  13 6  9  4  0  3  15 12
```

Discussion

Limitations

Overall, the accuracy for both the LSTM-Based Model and Transformer Decoder-Based Model were **lower than ideal.** In the context of our research problem, we would want a deep learning model for inferring gaps in incomplete genome assemblies to have nearly 100% accuracy, especially since the standard for sequence fidelity is extremely high for use in clinical applications. Because of **limited time and compute power**, we only trained our model on a small portion of the human genome. This especially hinders the performance of a transformer decoder-based model which benefits from greater compute and larger datasets. In addition, our models **periodically defaults to predicting a gap sequence of repeated tokens rather than a sequence of varying tokens**, which suggests that it is not fully learning sequence patterns or long-range dependencies in the human genome.

Future Work

To improve the learning and performance of our models, it would be beneficial to train them on **larger datasets and/or longer sequences**. Further, training on a **HPC cluster** such as OSCAR would allow for further accuracy gains, especially from a transformer-based model. One of our more ambitious goals was to evaluate the two models' performance on **genomic data of other species**, such as mouse and sea urchin. We hypothesize that a more intensively trained model would have performed worse on non-human data, highlighting that **species level differences may be discerned** by appropriate models.

1. Ji, Y., Zhou, Z., Liu, H., & Davuluri, R. V. (2021). DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. Bioinformatics, 37(15), 2112-2120.
2. Chen, E., Chu, J., Zhang, J., Warren, R. L., & Birol, I. (2021). GapPredict - A Language Model for Resolving Gaps in Draft Genome Assemblies. IEEE/ACM transactions on computational biology and bioinformatics, 18(6), 2802–2808. https://doi.org/10.1109/TCBB.2021.3109557
3. Wikimedia Foundation. (2025a, March 30). Reference genome. Wikipedia. https://en.wikipedia.org/wiki/Reference_genome