Progress towards Deep Learning Genome Assembly: de novo ML-based Read-Overlap Graphs and Partial Layout

Anonymous Authors[1]

Abstract

Although deep learning has proven effective on a wide range of problems in computational biology, the influence of deep learning on the central task of genome assembly has been mostly around the edges, mostly in pre- and post-processing. A recent result used graph convolutional networks for finding paths through the assembly graph on HiFi reads, but there is to date no deep learning tool for building an assembly graph *de novo* without any external information. In this extended abstract, we train from scratch a neural network embedding of synthetic short-reads from the SARS-CoV-2 genome into 3D Euclidean space that locally recapitulates the linear sequence distance of the read origins. With respect to the Overlap-Layout-Consensus (OLC) model of assembly, this embedding resolves fully the Overlap stage and part of the Layout stage of OLC, demonstrating for the first time that these tasks are potentially amenable to deep learning methods, though much future work remains in scaling this prototype approach to larger genomes and usable tools.[1]

1. Introduction

There are a wide range of bioinformatics tasks, and choosing the right analysis methodology for each requires carefully navigating the trade-offs (Berger & Yu, 2022). Deep learning has proven effective at a number of these tasks. Downstream prediction and annotation tasks have seen storied successes (e.g. AlphaFold (Jumper et al., 2021)). On the genomic sequencing front, ML has helped improve everything from base calling in sequencers (Wan et al., 2022), to variant calling after alignment (Poplin et al., 2018). Even the tasks of read alignment (Gupta & Saini, 2020; Jung & Han, 2022) or whole sequence alignment (Lall & Tallur, 2023) have seen substantial activity. However, one notable absence on these lists is the genome assembly task (Bonfield et al., 1995; Pevzner et al., 2001; Kececioglu & Myers, 1995; Myers, 1995), which is one of the central problems in genomics. To our knowledge, there exist no deep learning assemblers—a few studies apply deep learning to consensus-sequence determination (Vrček et al., 2022; Padovani de Souza et al., 2019), but nothing for generation of the assembly graph itself—which would seem to suggest that these problems are not amenable to ML. This manuscript explores these difficulties and points out a potential way forward for ML-based assembly.

What is the assembly problem? Our sequencing technology is generally unable to read entire genomes, so instead shotgun sequencing is performed, whereby the genome is randomly fragmented into smaller substrings known as ‘reads’. In a task known as *assembly*, these reads are pieced back together into long superstrings representing in some cases a putative genome or chromosome. Read lengths depends heavily on the sequencing technology. At present, the most popular technology used in practice are Illumina short-reads, which range in length from 200-300 base pairs (so-called 2nd-generation sequencing). More recent third-generation sequencing enables ‘long reads’ on the order of many kilobases, but they are currently still expensive—although many cutting-edge academic studies make use of long reads, the bulk of data generated is still short-reads (Segerman, 2020).

In this manuscript, we focus on *de novo* assembly of short reads, whereby we do not use any outside information—this is in contrast to reference-based mapping/alignment, which assumes we have already sequenced another individual from that species and simply need to locate each read as a substring of the reference, or reference-guided assembly, which uses the reference genome of a related species as side-channel information in the assembly process.

The two main approaches to assembly are the Overlap-Layout-Census (OLC) and de Bruijn Graph (DBG)-based methods (Li et al., 2012). DBG methods first break reads into even smaller k-mers before finding overlaps, whereas OLC operates by finding overlaps on the original reads.

[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.

[1]Code and models available at: https://storage.googleapis.com/progress-towards-assembly/colab.html

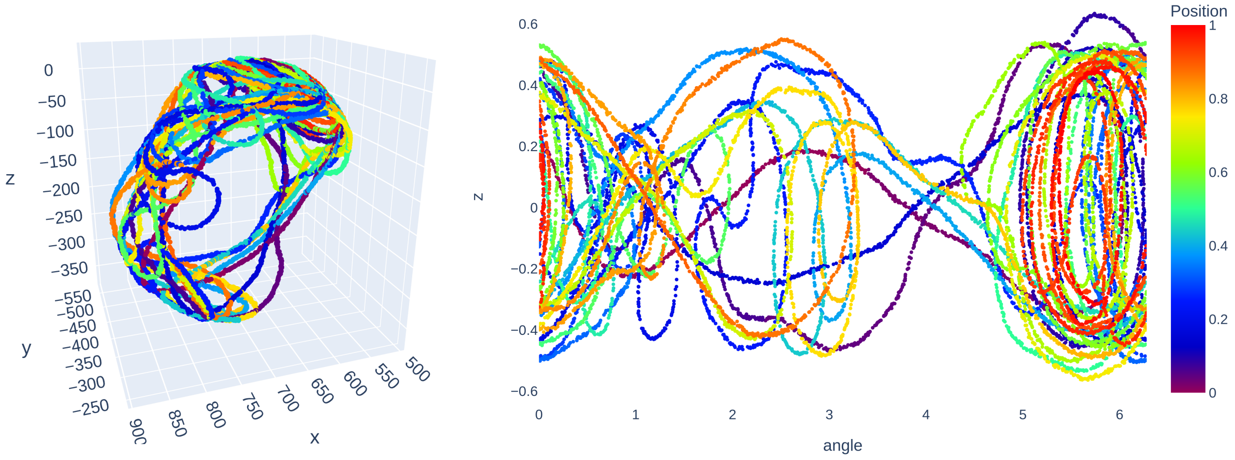


Figure 1. Euclidean embedding of synthetic short-reads (length $l = 256$) from the SARS-CoV-2 genome as a pseudo-read-overlap assembly graph. (**left**) Actual embedding was into 3D, attempting to preserve local distances along the sequence. (**right**) For ease of visualization, we further transformed the data onto a plane to better show the trajectories. Note that the embedding was not given any of the positional (**color**) information, but instead managed to recover that information implicitly from just the raw reads.

Here, we choose to completely eschew preprocessing of reads to demonstrate as proof-of-concept the extent to which neural networks can (and cannot) replace algorithmic design, so we follow the OLC approach in spirit. OLC methods first find overlaps in the prefix of one read with the suffix of another read; based on those overlaps, stretches of reads are bundled into contigs; finally, the most likely consensus sequence is chosen for each contig. In this paper, we show that a deep neural network embedding can perform the Overlap step, as well as part of the Layout step.

2. Methods

The input into an assembly algorithm is a corpus of reads $\{R_i\} \subseteq \Sigma^l$, where $\Sigma = \{0, 1, 2, 3\}$ (or alternately, A, C, G, T) and length l (which in our synthetic data will be $l = 256$, but is typically between 150 and 300). Each read R_i is assumed to come from some unknown true position p_i on the genome, which is a string of length L. Our strategy is to learn an embedding ϕ into $\mathbb{R}^n$ ($n = 3$ in this paper) of those reads such that $||\phi(R_i) - \phi(R_j)|| \approx |p_i - p_j|$ for small distances. If we were performing reference-based alignment/mapping, this would be easy—we could simply train a location prediction network directly outputting a prediction for p_i. We would have to take into account sequencing errors in the training distribution, but that is straightforward. However, the difficulty with assembly graphs is that we do not know the underlying locations of reads on the genome, but still need to approximate pairwise distances.

One approach would be to directly use edit distances. Given a mapping ϕ that approximately preserves edit distances for all possible reads in Σ^l, we could immediately use that embedding to create an assembly graph. There is work on neural edit distance embeddings for faster nearest neighbor search (Zhang et al., 2019), but the distortion rate is high when n is small (e.g. $n = 3$). Luckily, we can do better if we only care about embedding only reads that are similar to the reads in our corpus.

However, computing edit distances takes quadratic time (Backurs & Indyk, 2015), which is an expensive operation to use within a training loop. Thus, instead, for each read R_i, we will generate an 'adjacent' read $\hat{R}_i$ with known distance δ_i, where $\delta_i < l$. The adjacent read will be constructed by first shifting R_i either to the right or left by δ_i positions, filling in the empty positions randomly, and then applying random substitutions at a rate $\theta = 0.01$ (corresponding roughly to Illumina error rates, though the astute reader will note that this simplified adjacent read model ignores indels). We can then train our embedding ϕ on these pairs $(R_i, \hat{R}_i)$, which now have a known label δ_i. For the loss function, we will start with a Huber loss comparing δ_i to $||\phi(R_i) - \phi(\hat{R}_i)||_2$. However, note that $delta_i \in [0, l]$, but $|p_i - p_j| \in [0, L]$ and $l \ll L$. Thus, we modify the Huber loss by a reversed sigmoid multiplicative factor $S(x) = 1 - \frac{1}{1+e^{-m}}$, where $m = \frac{\min(\delta_i, ||\phi(R_i) - \phi(\hat{R}_i)||) - l/2}{l/16}$ so that the optimization focuses on distances $< l/2$.

2.1. Architecture

There have been several recent papers connecting convolutional filters with both edit distance (Dai et al., 2020) and minimizers (Yu, 2021), two primitives used in modern as-

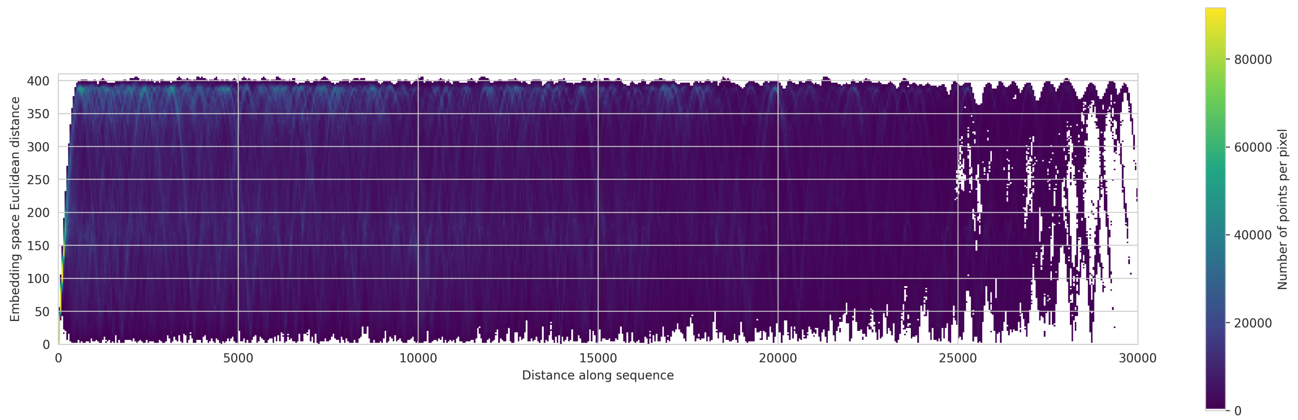


Figure 2. Heat map of embedding space Euclidean distance versus actual distance along the sequence. A perfect global embedding would depict a diagonal line here, but we of course only trained a local embedding. To visualize the local embedding properties, we need to zoom into the low-distance regime (Figure 3).

sembly algorithms. As such, we designed our architecture with a single convolutional layer with max-pooling to focus the model on k-mer features of interest, roughly corresponding to the use of minimizers in standard assembly algorithms. We chose filters to pick k-mers for $k \in \{1, \dots, 10\}$, with a total of 8218 total convolutional filters, with increasing numbers of filter for larger k. After max-pooling (with windows and strides proportional to k), this resulted in 253824 neurons after max-pooling. We then followed up with eight fully-connected layers (size 512, 256, 128, 64, 32, 16, 8, and then 3) with GELU activations between fully-connected layers to reduce the dimensionality down to 3. The total number of trainable parameters of the network was 130,437,175. We chose an embedding dimension of 3 for two reasons: (1) to visually inspect the results, and (2) because assembly graphs are known to be non-planar. A practical deep learning assembler would likely use bigger embedding dimension.

2.2. Training

In each epoch, we randomly generate one adjacent read (as defined in Methods) for each read in our corpus. Thus, the network learns embeddings for pairs of reads similar to our reads, without the danger of overfitting on any particular pairs. Indeed, because we are interested in finding the best embedding for a given set of reads, overfitting is no more a danger than it would be for an autoencoder, as the very point is to find the best low-dimensional embedding/compression of the original data with respect.

3. Results and Discussion

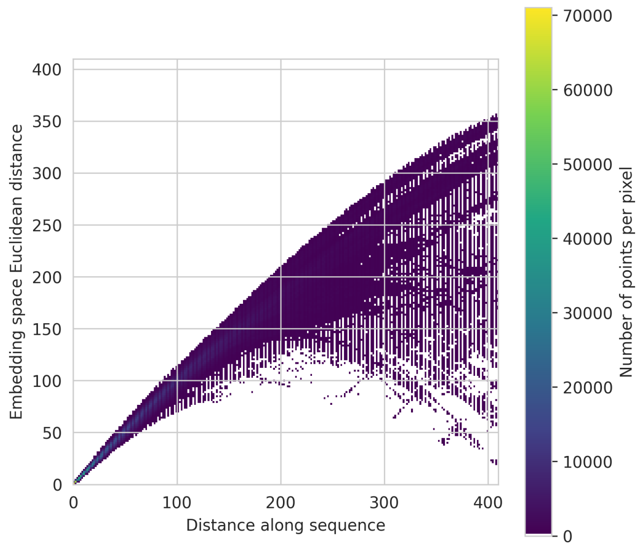
We started with the SARS-CoV-2 genome `NC_045512.2`, which is just under 30 kilobases long. A corpus of synthetic short-reads of length 256bp were generated to a depth of coverage of 200. The synthetic reads had an error rate of 1%, evenly divided three-ways among insertions, deletions, and substitutions. Reads with deletions were padded randomly to ensure constant length; insertions were uniformly chosen among the four nucleotides, and substitutions were uniformly chosen among the other three nucleotides.

We implemented the neural embedding architecture and training described above using Jax, Flax, and Optax. For this experiment, we trained for 16000 epochs, though the training and validation loss both started plateauing at around 3000 epochs. All experiments were performed in Google Colab on an A100 GPU high-RAM instance. Total wall clock time for 16000 epochs was around 27 hours (around 5 hours for 3000 epochs).

After training, we embedded all the original reads into $\mathbb{R}^3$ (Figure 1 left). Observing that the reads clustered near the surface of a cylinder, we unrolled that cylinder into 2D to allow us to visually observe that nearby reads do indeed form paths in the embedding that could correspond to contigs (Figure 1 right).

We then quantified in Figures 2 and 3 how well-aligned embedding distance is with both distance along the genomic sequence and the edit distance between reads. We observed that local distances tracked well when the embedding distance is below around 50 base pairs. At larger distances, there is little correspondence between embedding distance and either distance along sequence or edit distance.

However, most importantly, we are basically guaranteed that if the actual distances are small, so is the embedding distance. Additionally, it is relatively rare for embedding distances to be small but the actual distances large. Thus, the



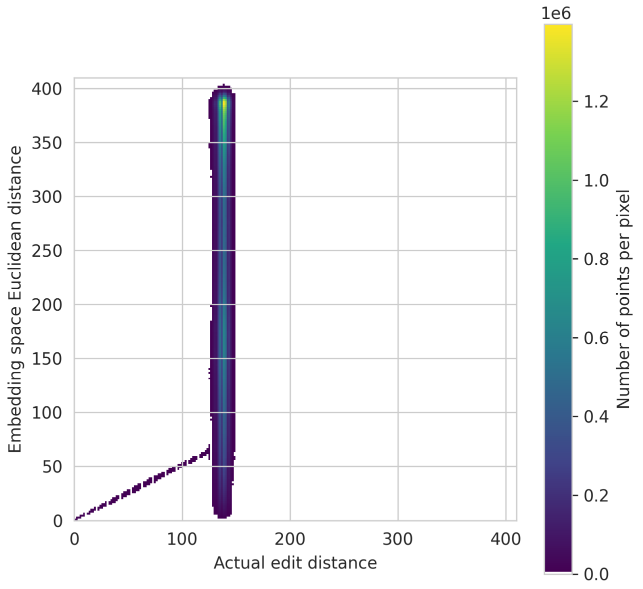


Figure 3. Heat maps comparing embedded distance with distance along sequence and actual edit distance (low distance regime).
(**left**) Zoomed-in version of Figure 2. The embedding space distance generally increases with distance along sequence, until the reads are too far apart along the sequence; note that the reads were only trained with a maximum distance of $l = 256$, so it is not surprising that distant reads have closer embedded distances.
(**right**) Actual Levenshtein edit distances between reads of length $l = 256$ obviously cannot exceed 256, and in practice for synthetic reads from the SARS-CoV-2 genome do not exceed 150. Note that edit distance corresponds roughly to twice the embedding distance in the small-distance regime; this is because an overlap distance of 1 requires two edits, an insertion and a deletion. Although it does occasionally happen that reads far apart in edit distance are close on the embedding, this is relatively rare. More importantly, if the distance along sequence or edit distance are small, then the the embedding distance is also small.

embedding is a good proxy for an Overlap read-assembly graph, as visually confirmed in Figure 1. Furthermore, not only do we have an Overlap graph, but it appears that the embedding does part of the work of the Layout step of OLC assemblers, in that we get a number of long paths of reads that do not intersect with other paths, which can be easily turned into contigs.

4. Conclusion

We have demonstrated progress towards building a *de novo* deep learning genome assembler, by training a neural embedding to $\mathbb{R}^3$ that maps a corpus of reads to points that can be connected into a read-overlap assembly graph. This is to our knowledge the first deep learning approach to build the assembly graph itself, as opposed to operating on other parts of the assembly process, such as graph traversal.

However, our prototype method is still very inefficient, requiring many hours of training time on even a small viral genome to just get the assembly graph when standard assemblers like Velvet (Zerbino & Birney, 2008) or SOAPdenovo2 (Luo et al., 2012) can complete the full assembly task in a matter of minutes; much future work remains before a deep learning approach would be competitive with standard assemblers. One potential for future optimization is relaxing the constraint that the embedding should be human-visualizable in 3D; that may allow for using a smaller network with fewer parameters while also helping avoid spurious path intersections. Additionally, there is some trade-off that can be made by stopping the training earlier: a more self-intersecting assembly graph will make finding contigs harder, but at least empirically this network basically converged within 3000 epochs, though we kept training until 16000 to be sure. Indeed, analyzing the resulting assembly graph may also lead to a better stopping criteria, as that is the only critical piece for downstream layout and consensus.

Ultimately, this paper is at its core a feasibility study. Although there may be very reasonable doubt (including among the authors) that deep learning assemblers will ever become competitive with state-of-the-art heuristics and software, we here show that it is at least possible.

References

Backurs, A. and Indyk, P. Edit distance cannot be computed in strongly subquadratic time (unless seth is false). In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pp. 51–58, 2015.

Berger, B. and Yu, Y. W. Navigating bottlenecks and trade-offs in genomic data analysis. *Nature Reviews Genetics*, pp. 1–16, 2022.

Bonfield, J. K., Smith, K. F., and Staden, R. A new dna sequence assembly program. *Nucleic acids research*, 23 (24):4992–4999, 1995.

Dai, X., Yan, X., Zhou, K., Wang, Y., Yang, H., and Cheng, J. Convolutional embedding for edit distance. In *proceedings of the 43rd international ACM SIGIR conference on Research and Development in information retrieval*, pp. 599–608, 2020.

Gupta, G. and Saini, S. Davi: Deep learning-based tool for alignment and single nucleotide variant identification. *Machine Learning: Science and Technology*, 1(2):025013, 2020.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Jung, Y. and Han, D. Bwa-meme: Bwa-mem emulated with a machine learning approach. *Bioinformatics*, 38(9): 2404–2413, 2022.

Kececioglu, J. D. and Myers, E. W. Combinatorial algorithms for dna sequence assembly. *Algorithmica*, 13(1-2): 7, 1995.

Lall, A. and Tallur, S. Deep reinforcement learning-based pairwise dna sequence alignment method compatible with embedded edge devices. *Scientific Reports*, 13(1):2773, 2023.

Li, Z., Chen, Y., Mu, D., Yuan, J., Shi, Y., Zhang, H., Gan, J., Li, N., Hu, X., Liu, B., et al. Comparison of the two major classes of assembly algorithms: overlap–layout–consensus and de-bruijn-graph. *Briefings in functional genomics*, 11(1):25–37, 2012.

Luo, R., Liu, B., Xie, Y., Li, Z., Huang, W., Yuan, J., He, G., Chen, Y., Pan, Q., Liu, Y., et al. Soapdenovo2: an empirically improved memory-efficient short-read de novo assembler. *Gigascience*, 1(1):2047–217X, 2012.

Myers, E. W. Toward simplifying and accurately formulating fragment assembly. *Journal of Computational Biology*, 2(2):275–290, 1995.

Padovani de Souza, K., Setubal, J. C., Ponce de Leon F. de Carvalho, A. C., Oliveira, G., Chateau, A., and Alves, R. Machine learning meets genome assembly. *Briefings in Bioinformatics*, 20(6):2116–2129, 2019.

Pevzner, P. A., Tang, H., and Waterman, M. S. An eulerian path approach to dna fragment assembly. *Proceedings of the national academy of sciences*, 98(17):9748–9753, 2001.

Poplin, R., Chang, P.-C., Alexander, D., Schwartz, S., Colthurst, T., Ku, A., Newburger, D., Dijamco, J., Nguyen, N., Afshar, P. T., et al. A universal snp and small-indel variant caller using deep neural networks. *Nature biotechnology*, 36(10):983–987, 2018.

Segerman, B. The most frequently used sequencing technologies and assembly methods in different time segments of the bacterial surveillance and refseq genome databases. *Frontiers in cellular and infection microbiology*, 10:527102, 2020.

Vrček, L., Bresson, X., Laurent, T., Schmitz, M., and Šikić, M. Learning to untangle genome assembly with graph convolutional networks. *arXiv preprint arXiv:2206.00668*, 2022.

Wan, Y. K., Hendra, C., Pratanwanich, P. N., and Göke, J. Beyond sequencing: machine learning algorithms extract biology hidden in nanopore signal data. *Trends in Genetics*, 38(3):246–257, 2022.

Yu, Y. W. On minimizers and convolutional filters: a partial justification for the unreasonable effectiveness of cnns in categorical sequence analysis. *arXiv preprint arXiv:2111.08452*, 2021.

Zerbino, D. R. and Birney, E. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, 18(5):821–829, 2008.

Zhang, X., Yuan, Y., and Indyk, P. Neural embeddings for nearest neighbor search under edit distance. 2019.