LEARNING THE "GAME" OF LIFE: RECONSTRUCTION OF CELL LINEAGES THROUGH CRISPR-INDUCED DNA MUTATIONS

JIAXIAO CAI, DA KUANG, ARUN KIRUBARAJAN, MUKUND VENKATESWARAN

ABSTRACT. Multi-cellular organisms are composed of many billions of individuals cells that exist by the mutation of a single stem cell. The CRISPR-based molecular-tools induce DNA mutations, which allows us to study the mutation lineages of various multi-ceulluar organisms. However, to date no lineage reconstruction algorithms have been examined for their performance/robustness across various molecular tools, datasets and sizes of lineage trees. In addition, it is unclear whether classical machine learning algorithms, deep neural networks or some combination of the two are the best approach for this task. In this paper, we introduce 1) a simulation framework for the zygote development process to achieve a dataset size required by deep learning models, and 2) various supervised and unsupervised approaches for cell mutation tree reconstruction. In particular, we show how deep generative models (Autoencoders and Variational Autoencoders), unsupervised clustering methods (K-means), and classical tree reconstruction algorithms (UPGMA) can all be used to trace cell mutation lineages.

1. INTRODUCTION

Multi-cellular organisms are composed of billions or trillions of different interconnected cells that derive from a single cell through repeated rounds of cell division — knowing the cell lineage that produces a fully developed organism from a single cell provides the framework for understanding when, where, and how cell fate decisions are made. The recent advent of new CRISPR-based molecular tools synthetically induced DNA mutations and made it possible to solve lineage of complex model organisms at single-cell resolution.[1] Starting in early embryogenesis, CRISPR-induced mutations occur stochastically at introduced target sites, and these mutations are stably inherited by the offspring of these cells and immune to further change. At the end of development, only the recording array sequence has to be read rather than the whole genome; the accumulated mutations can then be used as phylogenetic characters allowing the reconstruction of a tree of relationships between all cells. The objective of the project is to simulate the cell developing process and reconstruct the lineage tree of a cell, given the possible ending states. To be more specific, a cell, start from the unmutated ground state, develops into 128 cells by seven divisions in silico. Similar with the late developmental stage of *Drosophila* embryo, there is a one-hour interval between divisions, and some irreversible and inherent mutations happen within the intermission.

Computational complexity of representing each state poses a problem since each entry of the target can take on 30 distinct values, in addition to the empty token for a total of 31^{200} possibilities. To solve this problem, a Variational Autoencoder was trained to learn a meaningful latent representation for developed cells in $\mathbb{R}^{16}$, given the task of reconstruction. Once obtaining representations of the cells, different tree construction strategies were explored in the latent space (which we expand upon in Section 4): (1) Pair-wise distance and unweighted pair group method with arithmetic mean **(UPGMA)** and (2) K-means (Equal Parition) on Latent Representations.

1.1. Contributions.

In this paper, we introduce a zygote developing process simulation and demonstrate its encoding into a lineage tree (a vector with 200 mutable targets represents the each cell). Next, we introduce a Variational Auto-encoder architecture for encoding CRISPR targets, that is trained to learn a meaningful latent representation for developed cells. Finally, several strategies were explored for tree reconstruction, and we hypothesize different criteria for the evaluation of tree reconstruction.

2. BACKGROUNDS

2.1. Phylogenetic of Species vs. Lineage of Cells.

Classic phylogenetics typically analyzes the evolutionary relationships among a small number of species by observing genome sequence. Conversely, a lineage is a cellular level temporal series of development connected by a continuous line of descent from ancestor to descendant. It tracks hundreds of cells and their corresponding nucleic acid sequence. Cellular sequencing data gives detailed genetic information but introduces more challenges to analysis at the same time. The difficulties are from the random process during cell division, namely mutation and deletion. Mutations cause variations in genetic information, while deletions lead to dropouts of nucleic acids. These differences make it challenging to apply phylogenetic methods to lineage-tracing data directly. To solve the problem, a new representation of cells can be learned while fitting the stochastic process.

2.2. **Classical Reconstruction Algorithms: UPGMA.** Unweighted Pair Group Method with Arithmetic mean (UPGMA) [3] is one of the most widely used distance based tree reconstruction algorithms. In our project, we constrain the latent space of our auto-encoder to represent distances between cells. To construct a rooted tree, UPGMA finds the nearest nodes/clusters i and j and creates a central node as the new connection. Then updates the distance matrix D and iterates over the above steps until the tree is completely resolved and all branch lengths are known.

3. Related Works

3.1. **Allen Institute DREAM Challenge.** To mobilize for evaluating new optimal tree-building methods, Allen Institute started a current ongoing DREAM challenge about lineage tree reconstruction as there has been minimal rigorous examination of lineage reconstruction algorithms. Inspired by this challenge, the idea of our project is to use a combination of deep learning models and classical machine learning algorithms to solve the linage tree reconstruction task. However, such models excel with large datasets, and the data provided by the challenge is not sufficient for training a deep learning model. Encouraged by Professor Pratik Chaudhari, we decided to generate our own data based on an approach given by a paper written by Dr. Salvador-Martinez, as well as develop our own tree construction algorithm.

3.2. **Variational Auto-encoder in Latent Representation.** In 2018, Yosef Lab introduced single-cell variational inference (scVI) for the probabilistic representation and analysis of gene expression in single cells [2]. It uses a Variational Auto-encoder to aggregate information across similar cells and genes and to approximate the distributions that underlie observed expression values while accounting for batch effects and limited sensitivity.

3.3. **Language Modelling.** The task of lineage tree reconstruction and language modelling (where a model outputs a probability to a given sentence) are extremely related due to their sequential nature and the tree-based structure of both language syntax and genetic mutation. In particular, in theory it is possible to train a language model to perform tree reconstruction, where a walk from the root node to any leaf node is considered a sentence. The state-of-the-art in language modelling involved Transformer-based models such as BERT or Recurrent Neural Networks with other augmentations (e.g. attention, context management). However, the representation of words is not analogous with the representation of cell states in this project, since typical vocabularies (e.g. English) contain around 100,000 tokens whereas our cell targets 31^{200}. This makes both one-hot-encoding and contextual representation extremely difficult for cell states, which causes current language modelling techniques for lineage tree construction to be intractable. We expand on our autoencoder-based latent representation in Section 4.

4. Approaches

4.1. **Simulation.** We introduce a simulation framework for the zygote development process to achieve a dataset size required by deep learning models. In our simulation, every cell is represented as a vector of 200 characters, each character representing one Cas9 target. The simulation begins with one cell, the fertilized egg, that has all its targets in an unmutated state. The unmutated ground state is noted as "0". Mutation events are implemented as following a Poisson distribution. Under the Poisson model, given a mutation rate $\mu_t = 0.0014$ (per unit time), the probability that a site remains unmutated after t minutes is:$e^{-\mu_t t}$. Each unmutated target can mutate to one of 30 possible mutated states with probabilities sampled from a random gamma distribution (shape parameter $k = 0.1$ and scale parameter γ = 2). Once a target is mutated, it can no longer change, either to revert to the unmutated state or to transit to a new state. The mutation process is shown in Figure 1.(a).

Besides mutation, inter-target deletions are possible to happen during the development. In experiment, where 2 or more relatively close mutation will introduce double strands break and lead to a lost of DNA between these breaks. This events have been implemented in the simulation. If two mutations occur in close targets (less than 20 targets apart in the recording array) within a short interval of time during a given cell division, all the targets between them are removed. In our dataset, one simulation records the process that a zygote cell develops into 128 cells by seven divisions with a one-hour interval in-between. Such simulation were processed 3000 times so that we have 2500 lineage trees for training, 200 for validation and 300 for testing. The frequency for each variation has been counted in Table 2.

4.2. **Representation.** For each developed cell, a vector of 200 characters tracks all the mutations during the divisions. Given the ground state as 0 to 30 possible mutations are coded as A to Z and a to z. Moreover, '-' represents a deleted target. Representing variations as unique discrete characters leads to data sparsity and difficulties for training a neural network. Therefore, "mutations embedding" is used to map variations into real number as Table 1 in appendix. Since probabilities of each mutation are obey a gamma distribution, a larger real numbers in this table are used to represent less likely mutations. Based on mutation embedding, cells from one lineage tree are projected in 3D-PCA space and

2D-t-SNE space for visualization in Figure 1. In (a) and (c), cells are colored by their lineage after three divisions. In (b), cells are colored by lineage after two divisions in (b). From (b) and (c), one can observe that graphical-based clustering has its limitation to classify the cell's lineage. For example, Cluster 3 spreads at the two sides of the plot in (b) even if they both belong to the sub-cluster 6 in (c). The 3D-PCA plot supports the observation since cells become mixed after three divisions. It is because cells' diverseness decrease with the increase of division, and deletion introduces uncertainty by hiding variations.

4.3. **Autoencoder Model.** After experimenting with an autoencoder and a variational autoencoder model, we decided to continue with the regular autoencoder model as enforcing two constraints on the latent space with our loss function turned out to be difficult to train. So, in order to constrain our latent encodings to represent pairwise distances between cells, we experimented with the loss function used to train the autoencoder model. Ultimately, we implemented a mean squared error term between the latent representations of two drawn examples in addition to a mean-squared error reconstruction term. Our loss function to minimize for a given set of n cells was then the following:

$$L = \frac{1}{n}\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{1}{2}(||y_i - x_i||_2^2 + ||y_j - x_j||_2^2) + (||z_i - z_j||_2 - p_{ij})^2$$

Where x_i represents an input datum, z_i represents the latent representation of x_i, and y_i represents the generated representation by the decoder from z_i. In order to train the model, we used SGD with mini-batches of size 100. Each update, we would draw a tree at random and randomly select 100 pairs of cells from the tree to train the above loss function. Intuitively, this task is representative of encoding pairwise distances between cells in a tree. This is significant information to encode since it would allow us to construct a pairwise distance matrix between all cells in the set for use in tree reconstruction methods.

4.4. **Model Architecture.** Our autoencoder model was comprised of three fully connected encoder layers and three symmetric fully connected decoder layers. The first layer takes the 200 dimensional input into a size 128 vector with a ReLU activation. The next layer takes this 128 dimensional vector into a size 64 vector with tanh activation. The final encoding layer creates a size 16 latent representation of our cell. As previously stated, the decoder is symmetric to the encoder and takes this latent representation to a generated positive real-valued vector of size 200.

4.5. **Reconstruction.** We propose two different methods of reconstructing the lineage tree. The first is using an Autoencoder to generate latent representations for each cell, and running the same sized 2-means clustering algorithm on the representations to partition our cell space into two equally sized centroids. Then, we separate the two partitions and recurse on each 2-means cluster until we yield leaves (single datum). Each iteration of this will take us a further depth in the tree, with the mean of the centroid being representative of the ancestors of the leaves in that level of the tree. Since K-means takes $O(n^2)$ time to compute two centroids, we yield the recurrence $T(n) = 2T(\frac{n}{2}) + O(n^2)$. By the master theorem, this algorithm runs in $O(n^2)$ time (which is comparable to most tree reconstruction algorithms). In our second method of tree reconstruction, a pairwise distance matrix is calculated directly on counting the different targets between any two mutation embedding and a lineage tree is reconstructed by UPGMA. Figure 3(a) represents the reconstructed tree while (b) is the real lineage tree.

4.6. **Evaluation.** In a rooted tree, a clade is a group of leaves that have a common ancestor that is not a common ancestor for any other leaf in the tree. Percentage of clade similarity, in other words, the number of branches in one tree that is present in another), is a measurement of the topological distance between two trees [4]. In our project, the percentage of clade similarity is calculated by an R package: Analyses of Phylogenetics and Evolution (APE). For instance, regarding to the lineage trees in Figure 3, there are 127 clades in each tree, and 57 clades in the reference tree are not identical to the reconstructed tree. Therefore the percentage of clades similarity is $\frac{127-57}{127} = 62.99\%$ for the reconstruction.

5. Experimental Results

5.1. **Autoencoder.** Recall that to follow the DREAM challenge setup, we set the length of vector as 200 representing for mutable CRISPR targets and to simulate the late developmental stage of *Drosophila* embryo, we set a one-hour interval between divisions. However, based the frequency counts in Table 2, this set up may introduces more deletions than expected. There are 50% targets are effected by deletions. These deletions force all targets between the two mutations to be replaced with the deleted character drop some of the inherent information. It makes the lineage tree difficult to reconstruct.

In our training period, we noticed that the MSE reconstruction loss would go to zero quite quickly, while the MSE pairwise constraint term would only slowly decrease. To combat this, we performed weighting of those terms in the loss function, similar to beta-variational autoencoder. However, this yielded quite similar results. One reason for this might be that the reconstruction and latent distance terms conflict as a result of the deletions, or that the reconstruction term introduces some noise into our latent space that it is not relevant to the lineage distance between cells. Because of this reconstruction term, the auto-encoder may have had difficulty learning the importance of the deletion character in the cell lineage. In the latent space, the representation for unmutated and deletion characters may have been too close together, which would lead to confusion in partitions of the tree in our latent space when a large deletion occurred.

Our autoencoder still extracted some useful information on distances between leaves, as initial tree partitions seemed to cluster in our PCA of the latent representations, as shown in Appendix Figure 2. Notice that though it is somewhat partitioned, the distance between the partitions is still quite small, posing difficulties for our reconstruction algorithms. These partitions were not organized in n-spherical clusters, but rather were spread out in hyperplanes and other sorts of shapes in the projected 3-dimensional space, which posed difficulties for our even 2-means tree reconstruction method.

It is quite possible that due to the high dimensionality of the task, that a longer training period and greater variance and number in simulated trees would've allowed the autoencoder model to learn the distribution of deletions and thus better enforce our latent space constraint.

5.2. **UPGMA Tree Reconstruction.** When first dealing with the data, we thought of edit distance to be a useful metric in the tree reconstruction process, and in our training dataset, the real valued representation of the cell data seemed to cluster decently according to the partitions of the tree. In test dataset, UPGMA was used to reconstruct lineage tree based on distance matrix from embedding space and the latent space. The histogram in Fig 4 shows the percentages of Clade similarity of each tree reconstruction. In general, both embedding and latent representation are not able to reconstruct the lineage tree. Our real valued embeddings ended up yielding a relatively better similarity score, and we believe this was due to the latent representation being confused between reconstruction and enforcing our distance constraint, and thus not being able to model either well. Fig 5 shows that auto-encoder is learning to reconstruct a balanced tree with while updating the weights. Fig 5 shows the reconstructed tree from embedding space and latent space. The latent space is getting a as good representation as embedding space.

5.3. **Even 2-Means Tree Reconstruction.** The difficulty that arises with using the even 2-means reconstruction algorithm was that it is unlikely that our latent representations can exist in n-spheres due to the sizeable ambient space of our targets (this scenario would be the ideal since it would make the tree partitions clear for the reconstruction algorithm to use). Despite our latent representations preserving relative semantic relationships when projected onto lower dimensions, our K-means clustering rarely offers improvements to our similarity scores, most likely as a result of the curse of dimensionality mentioned above. As a result, our reconstructions of initial partitions of the leaf set seemed to be only 67% accurate. We concluded that the even 2-means reconstruction algorithm was not a good fit for the task due to our distance constraints in the latent space.

6. Discussions

In our approach, we attempted to encode absolute distances in our pairwise distance matrix. However, the task of tree reconstruction involves a fair amount of context since we have to consider the "branch" that we are currently on and the set of possible cells that we can choose as leaves. Because deletion frequently happens in our data set, there might be a co-relationship between each mutation stage. Future work for our approach would be to implement some kind of automatic generation, for example, using an algorithm from NLP such as Beam Search, which maintains a set of the most likely outputs and updates the set at each iteration, thus maintaining context.

Next, the dimension of our dataset is quite large, consisting of 200 possible mutation targets for each cell. We choose the length based on the dataset of the DREAM challenge, but to solve the lineage tree problem, the length of targets is a hyperparameter. As such, we should be able to extract more information by adjusting it and, for example, increasing the number of targets to construct more complex models or decreasing the number of targets to construct simpler mutation processes. Besides, one consideration to make regarding the size of each cell state is the curse of dimensionality. In particular, it could be not very easy for our model to both represent and generate such large 200-dimensional vectors during training time. It also makes the act of representing the encoding as a vector complicated since one-hot-encoding results in a combinatorial explosion.

Last, in order for the autoencoder to represent a proper model in the latent space, we may consider adopting batch normalization for some layers or the input distances matrix. It could help to mitigate the problem of the internal covariate shift so that the latent space could have a better representation of the distance.

7. REFERENCES

[1] I. Salvador-Martınez, M. Grillo, M. Averof, and M. J. Telford, "Is it possible to reconstruct an accurate cell lineage using CRISPR recorders?," Developmental Biology, p. 23.

[2] R. Lopez, J. Regier, M. B. Cole, M. I. Jordan, and N. Yosef, "Deep generative modeling for single-cell transcriptomics," Nat Methods, vol. 15, no. 12, pp. 1053–1058, Dec. 2018.

[3] C. D. Michener and R. R. Sokal. A quantitative approach to a problem in classification. Evolution, vol.11, pp.130-162, 1957.

[4] Jaime Huerta-Cepas, Francois Serra and Peer Bork. ETE 3: Reconstruction, analysis and visualization of phylogenomic data. Mol Biol Evol 2016; doi: 10.1093/molbev/msw046

8. APPENDICES

TABLE 1. Mutation Embedding Table

0	-	A	B	C	...	Z	a	b	c	d
0	1	2	3	4	...	27	28	29	30	31

TABLE 2. Variation Frequency in Simulated Data

Variation	0	-	A	B	C	D	E	F	G	...
Freq	0.352	0.535	0.0329	0.023	0.0148	0.0171	0.0069	0.004206	0.003204	...

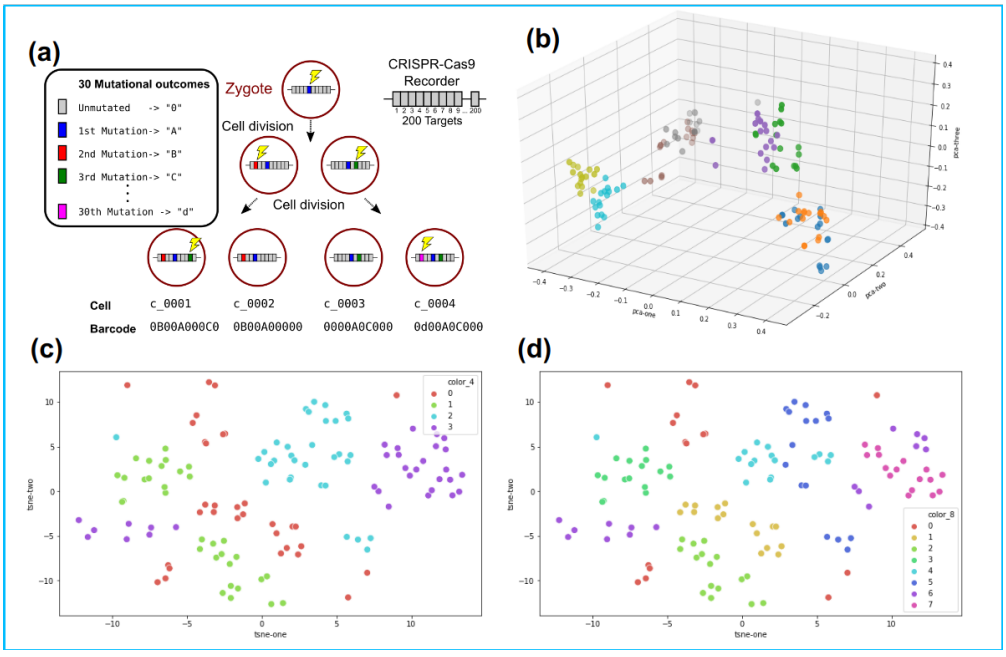

FIGURE 1. Visualization of Mutation Embedding

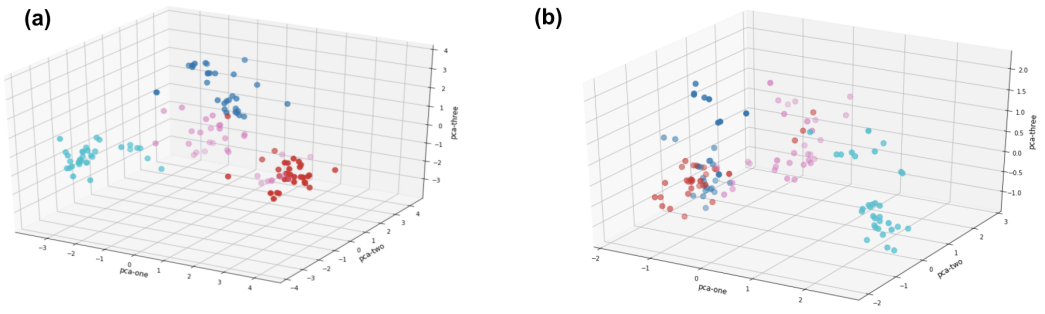

FIGURE 2. Example trees in PCA dimension of autoencoded latent space

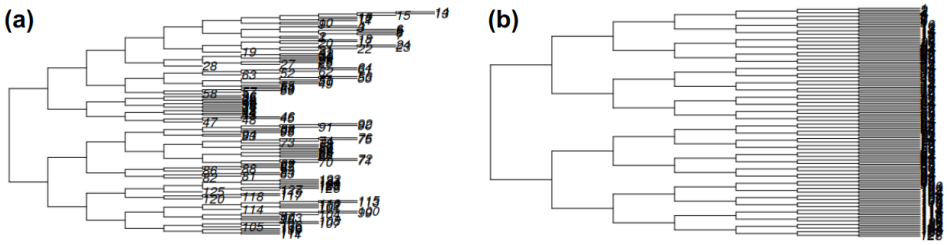

FIGURE 3. UPGMA Based on Mutation Embedding

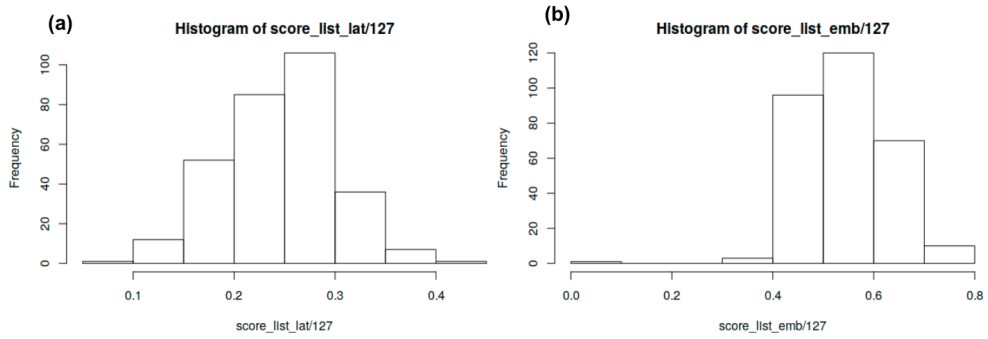

FIGURE 4. Histgram of Clade similarity in latent embedding (a) and real-valued embedding (b) space

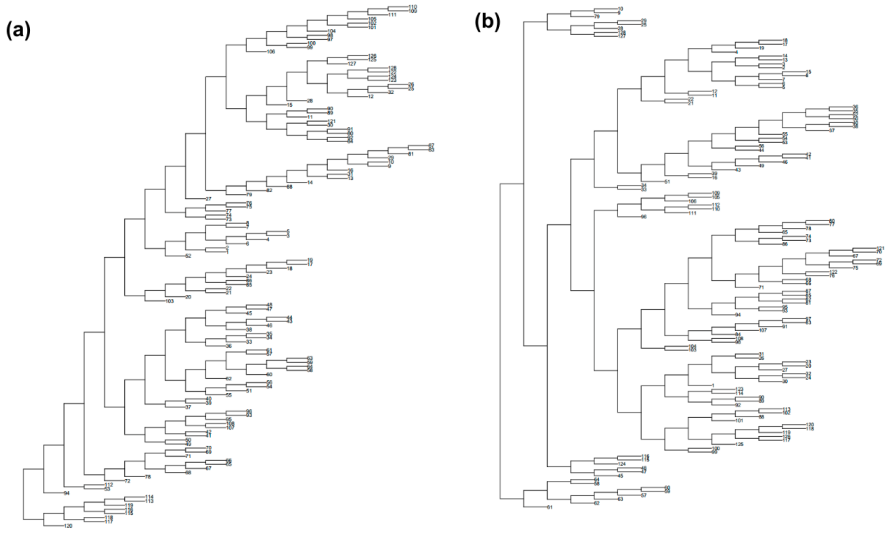

FIGURE 5. Tree becomes more balanced with longer training period. Updates = (a) 16300 , (b) 32600

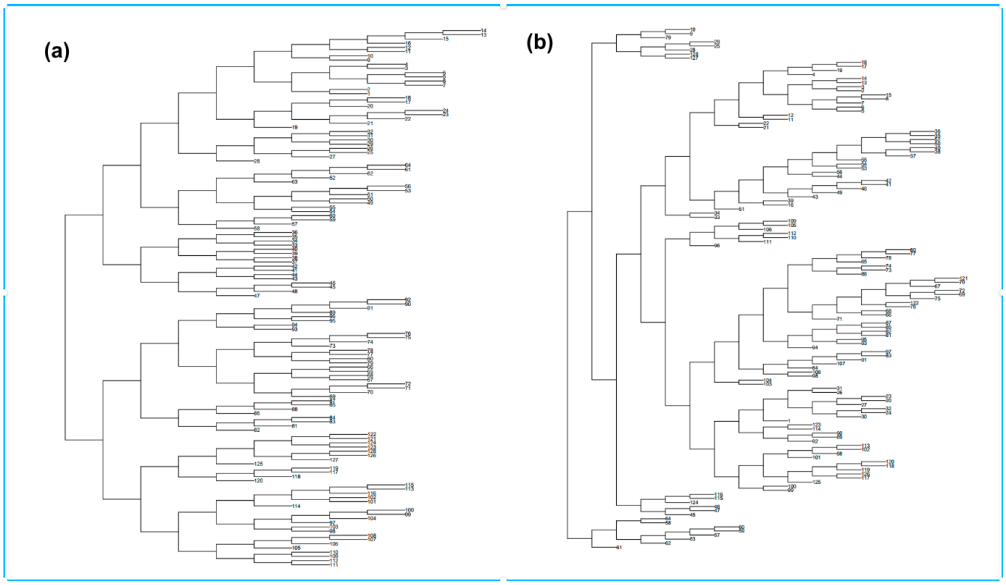

FIGURE 6. Reconstructed Tree from (a) Embedding vs. (b) Latent space