Right, so I think while we’re waiting, I’ll just give a very brief introduction about Jingshu. Jingshu is an Assistant Professor from the Stats Department at University of Chicago. And today she’s going to present some very exciting work on trajectory inference for the single cell data. I’m very excited to hear about her work. I think…

Let’s wait for two minutes and then we’ll start with Jingshu’s work.

So if you have any other related questions about Jingshu before the talk start, you’re free to ask as well. (chuckles)

Hi Lynn. Can you make me a co-host. Oh right. Thanks for reminding me.

Let me see.

So you can see my screen, right? I can see your screen. Looks good to me.

Maybe I’ll hand it over to you now Jingshu I think,

(indistinct) late I think,

they can ask questions if you miss any details.

So thanks everyone for coming and settling for the introduction invitation.

Today I will talk about the Single-Cell RNA Sequencing Data and how we can learn the cell dynamics from the single-cell RNA sequencing.
So the single-cell RNA sequencing is a relatively newly developed technology for measuring the RNA expression levels in the cells. The traditional microarrays or bulk RNA sequencing matches the gene expressions as the average across all cells in a tissue. However, a cell is made of many cells and the cell population is typically not homogenous. So in contrast, in single-cell RNA sequencing, we have measured the transcriptional profile in each individual cell. And this is particularly useful when the cell population is complicated. For example when the cells are experiencing dynamic changes.
And as an application of the method I will introduce today in this lecture, in this talk, I will focus on the study of the mouse neocortex. This is a cartoon showing the migration and generation of the projection neurons in the mouse neocortex. Yeah, you guys see that this is quite a complicated process and there are still a lot of things that are unknown about the neuronal diversity and the mechanism of how the projection neurons are generated. And the goal is that we want to use the single-cell RNA sequencing so that we can have a more complete understanding of this, the neuronal diversity and the neuron development. So you can see that here in this cartoon, this shapes, there are different shapes and colors, to represent different cell types in the neocortex, as the cells are experiencing the continuous dynamic changes actually in the real cell population, it is much complicated than that. There is not clear boundaries between different cell types and there may be... There even, it’s not a clear definition of cell type. So, what we hope, is that we want to use single-cell RNA sequencing to first recover the trajectory of the dynamic changes or the developmental process that the cells are experiencing. So specifically we focus on two datasets. One data set, we name it as data set A.
So this is a data set that is recently collected by my collaborator. And so we have samples... The cells from the mouse neocortex at six different embryonic days. And before our data, there is another dataset we call it, we name it data set B. And this dataset is a smaller dataset than ours but they are... They have also sequenced a very similar brain region of the mouse and they have a sequence of cells from four different embryonic days. So you can see that our, most of the days that are sequenced in our dataset and with the other dataset B, do not overlap. And so it would be beneficial if we can perform a choice analysis of the two datasets and learn a shared developmental trajectory as these two datasets.
are actually sequencing the same mouse brain region. So as you may have imagined, if we don’t do anything, if we just concatenate the cells from two datasets and treat them as datasets from the same lab, then these two datasets actually will not, the cells will not merge because of the bash effects between the two datasets. Because these are from two labs and they have different sequencing machines so the cells become different, though they are coming from the same brain region. And this is a figure called the UMAP which is a two-dimensional projection of the high dimensional, observed single-cell RNA sequencing data so that we can have a visualization so we can have a visualization of the cell population. And using our marker which is called vitae that I will introduce later or we can merge the cells from two different sources. And as I will show later, we can also keep the uniqueness, the unique characteristics that only exist in one of the datasets. So we can keep the biological meaningful differences between the two datasets. And our method is actually not just, data integration approach. So what we can do, is that we can also simultaneously, learn a shared trajectory structure
and we can at the same time do the disintegration or more generally correct for confounding effects such as the data source and other various like cell cycles. And in the... In this figure, the arrows show the direction of the developmental process and the line width represents the score for an edge. So it shows how confident we are in like whether there’s a transition between the two states that the line connects. So our method actually belongs to a larger group of computational tools for single-cell RNA sequencing which is called the trajectory inference. So it is called trajectory inference that is different from statistical inference. In general, the trajectory inference approaches, they will infer or they start with a type of the underlying trajectory structure as what we have already seen in the mouse neocortex, and many more are using single-cell RNA sequencing data.
they will assume a specific type of the trajectory structure for the underlying developmental process, such as a linear structure, a linear topology or a bifurcating, a bifurcation or tree-like trajectory. And as the cell populations that we are trying to understand, become more and more complicated, recent methods also try to infer, the type of the trajectory structure from the observed single-cell RNA sequencing data. And whilst we have learned the trajectory structure, then this trajectory inference approaches, will computationally project and order the cells along the trajectory. And the right order of the cells along the trajectory, are called the pseudotime of the cells. So the trajectory inference, is also called the pseudotime analysis. And since... So the first trajectory inference method is proposed in 2014 and since then it has become a very popular tool that are used in analyzing single-cell RNA sequencing data. And in this study, it calculates, it summarizes the number of single-cell RNA sequences studies that are published per month. And you can see that in recent years, more than half of the published single-cell RNA sequencing studies.
will have some investments of the pseudotime and trajectories in the cell population that they are investigating. And there has also been a lot of methods for trajectory inference. And in this, there is a comprehensive benchmarking paper, recently in “Nature Biotech” and it has summarized 70 different trajectory methods. And in your paper they have compared about 45 different trajectory inference methods from different aspects. So you may wonder, since there are so many trajectory inference methods that are already there, why do we still want to develop, a new trajectory inference method? So the first point is that, although we have 70 different methods, many trajectory inference methods, they are assuming a specific type of the trajectory structure. So many methods only work for a sound developmental process. If you consider the methods that can work for... They have the flexibility if you work for a variety of the trajectory structures then we don’t have that many methods that are available. And another concern that I have is that most methods, these trajectory inference methods, do not have explicit statistical models. So what I mean is that,
though people are kind of clear about what’s the biological signal that we want to find in the trajectory inference, it is actually, many methods are actually pretty vague about from the aspect of like for the single-cell data matrix, what can be the definition of the trajectory that they want to infer.

So, and how that they are generating, how the data it can be modeled and generated with the trajectory structure.

So as the statistician, I think it would be beneficial, if we have a model-based trajectory inference approach, so that we can better understand the profit, how good our estimations are and have some certain qualification of the trajectories or slow times that we infer.

And the third point is that as you have shown at the beginning, there is also a growing need, to efficiently align trajectories or do a joint analysis from multiple single-cell RNA sequencing datasets.

As the, as the studies... there has already been a lot of studies for datasets, they are for the same tissue or for the same cell type. And it will be...

And we can learn a better picture of, on this, the biological process in the tissue or for the cell time,
if we can use all available datasets.
And so there’s a strong need,
an increasing need to do this joint trajectory analysis
for multiple datasets.
So because of these reasons,
we develop a new statistical framework and a new
method,
and we call it VITAE,
which is short for variational inference
for trajectory by autoencoders.
And it is a model-based trajectory inference approach.
So our model starts with a definition
of the trajectory backbone.
So we use a graph to define the trajectory backbone.
So we start with a complete graph G,
well the vertices are the distinct cell states and cell type
and an edge denotes a possible transition
between two cell states and or cell types.
And then we can define a cell position on the graph
which is a vector, which is a landscape vector.
And it’s...
A K is the number of vertices on the graph,
in the graph.
So if a cell is exactly,
belongs to one cell state or cell type,
then it is on cell vertex.
And if the cell is experiencing a transition
between two cell states or cell types,
then we denote it as on the edge between two vertices.
And then we can define the trajectory backbone
as a subgraph of G.
So we only include an edge or vertex if we really observe cells that are on the edge. So though there are many possible transitions between the cell types or cell states, We believe, we include a transition only when we do observe cells that are experiencing such transitions. So though there can be many edges in our complete graph, on the sub graph, it’s a sparse success of the other edges that are possible on the graph. And a benefit of the above definition, is that we can allow any types of the trajectory structure. So it can be either a linear structure, a bifurcate chain or a tree-like structure or a cycle, it completely depends on how the data shows. And so we allow, we want the data to automatically determine the other, the trajectory structure or topology of the underlying dynamic process. And we can also define the pseudotime for each cell. I have not written down the exact definition here but the idea is that we first need a root vertex. So a root vertex is the start of this dynamic process. And it can be given by the user, depending on looking at the marker genes or other side biological information. And later we will also... I will also show you that for some datasets, we can automatically determine the root vertex. And with a given root vertex,
then the graph becomes a directed graph. And we had defined the pseudotime of the cell as the shortest path from the root to a specific cell along the trajectory, along the trajectory backbone. So this graph defines the trajectory structure.

And the next step, is that we want to link the trajectory structure with the data generation model. So the single-cell RNA sequencing data matrix, is typically a high dimensional matrix because for each cell, we typically observe tens of thousands of genes and there are also complicated dependency relationships among the genes. And what we assume, is that we assume that these dependencies across genes, can be explained by a latent variables, in a low dimensional space. And we assume that these latent variables, are following our normal distributions and they also have the graph structure. So here, are the positions of the vertices on the graph in this low dimensional space, and the meaning of is a linear combination of these vertices, depending on, of the positions of the vertices, depending on the cell’s graphic position on the graph. And, what I want to emphasize here is one point, that we assume a non-linear marking.
from the latent space to the high dimensional observed data, because we think that though in the low dimensional space, we can represent the trajectory as these linear lines, it is very likely a manifold on the observed data. So this non-linear mapping, can map this linear lines to hertz in the high dimensional space. And now to consider, to account for the confounding covariates, such as the data source or cell cycle, we also allow this non-linear mapping to depend on this covariates. And here we are... Because the observed data count, we assume it follows an active binomial distribution, and L(i) are... Oh, sorry. L(i) here should be known library size. Sorry for the typo. It should be known library sizes. And CRJ, and the CRG, the dispersion parameter switch gene, are unknown parameters. And so in this, in the current model, the unknown parameters we have, these cell, the vertex positions, the cell positions on the graph, the non-linear mapping and this unknown dispersion parameters. So we have a lot of parameters. So to further simplify our estimation, we assume that there are a mixture prior
0:22:18 → 0:22:19.84 on the cell positions.
0:22:19.84 → 0:22:23.23 So it’s a very tactical idea.
0:22:23.23 → 0:22:27.273 So we assume that first the cell,
0:22:27.273 → 0:22:30.275 there are some latent variables, \( C_i \) for each cell.
0:22:30.275 → 0:22:35.275 And so \( C_i \) determines which edge or vertex a cell chooses.
0:22:35.41 → 0:22:39.41 So the cell has some probability to choose a specific edge
0:22:39.41 → 0:22:42.17 or a specific vertex,
0:22:42.17 → 0:22:45.52 and if it chooses an edge,
0:22:45.52 → 0:22:48.903 then, it then eventually choose the location of,
0:22:50.677 → 0:22:53.317 the relative location on the edge..
0:22:55.03 → 0:22:58.883 So that’s what becomes a mixture prior on this diagram two.
0:22:59.84 → 0:23:00.673 And,
0:23:02.375 → 0:23:06.25 for these non-linear mapping functions,
0:23:06.25 → 0:23:10.06 we’re including non-linear mappings.
0:23:10.06 → 0:23:14.533 We model these \( F(G(i)) \) functions by a neural network.
0:23:16.5 → 0:23:18.24 So, Oh, do you?
0:23:18.24 → 0:23:22.48 So our parameters now, our known parameters now are,
0:23:22.48 → 0:23:24.287 this \( U \) the positions of the vertices
0:23:24.287 → 0:23:26.55 on the low-dimensional space,
0:23:26.55 → 0:23:31.55 this \( P_i \), the probability of each vertex and edge,
0:23:34.04 → 0:23:36.727 and the non-linear mappings,
0:23:36.727 → 0:23:39.19 which are the waste in the neural network
0:23:39.19 → 0:23:41.81 and this, this dispersion parameters.
0:23:41.81 → 0:23:42.82 And we...
0:23:42.82 → 0:23:47.27 I space these parameters by combining our mixture model
0:23:47.27 → 0:23:48.733 with a variational autoencoder.
0:23:51.28 → 0:23:52.92 So the variational autoencoder.
0:23:52.92 → 0:23:57.59 So the autoencoder has been a very popular model
for in deep learning.

So what it can do is, is that it can, can have some non-linear mapping of the observed data to a low-dimensional space and we want the low-dimensional space to best, recover our observed time rational data.

And here, we also have such a task. We have the low-dimensional space and we want to best to recover our observed data. And what’s different is that we also have a prior on the latent space, because we have the prior, so we use the variational autoencoder model in deep learning. So the classical variational autoencoder in deep learning, we’ll assume that the latent space, has the standard normal distribution as the prior. And here we just modify it so that we have the...

So that the latent space have the mixture prior that are assumed in our previous mixture models. And we use the same approach as the variational autoencoder, the variational path which is, though the, to approximate the posteriors of the latent space. So though, because of the complicated priors and non-linear mappings, this prior, the posterior of the latent space conditional on the observed data and the confounding covariates, it can be complicated, we approximate it by our normal distributions and the mean and the variances of the normal distributions which are functions of the observed data Y(i).
and the covariance $X_i$, are also non-linear functions and we model them by the neural network and that’s the encoder.

So the decoder is the nominal mapping function $F(G(i))$, mapping the latent space to the observed data. And encoder are neural networks that are approximating the posteriors of the latent space.

- Hi, Jingshu, can I ask a very quick question?
- If I understand correctly, up to now you have not used the time information, is this true?
- Or you have considered to include the time information in the covariate $X$?
- So we will not use time information in our trajectory inference.

If I have time, I may have a last slide which is a review of the literature into data inference. The most commonly used and best performing methods, they will tend to not use the time information because there is... Though the time information is typically correlated with developmental like, timing of the cells, but because at each collectional time, is a mixture of cells at different environmental time. So it’s a big, complicated relation and some methods use that information but many methods do not use that.
And so our approach is the methods that do not use the collection time information. And, we use it only when we decide which vertex is the root. And I will show that later.

So our... So the last function is, is composed of three parts. The first part is this likelihood based reconstruction loss. So this evaluates how good are our latent spaces to reconstruct the high-dimensional observed data. And the second part is the KL divergence between the posterior distribution and the prior. And you can think of it as a regularization term. And so to regularize if the posterior is very far away from the prior, it can be also viewed as a lower bound of the marginalized data. So here we make the, we add the training parameter beta, and in practice we said, beta are larger than one, so that we can encourage the posterior, the regularization so that the posteriors of the I will, are more likely to tend to align along the edges and vertices. And that’s the idea that has been used in deep learning which is called the beta. And the third term,
A term for adjusting for the covariance is used in this context. The covariance is a measure of how much two random variables change together. In this case, it is used to cover the covariance of the latent space with the covariates. We want our latent space to be correlated with the covariance while maximizing the reconstruction of the data by only using the covariates. Setting the tuning parameter alpha larger than zero helps in decorrelating the data. Another aspect of our training is that we need a good internalization of the graph. Specifically, we need to determine the number of vertices and their positions in the low-dimensional space. This is not an easy task. If we randomly set the total number of vertices, our final graph will depend on their initial values.

To get the initial values of the unknown parameters, we first train with beta equal to zero, not using any prior distributions. Then, we only look at the reconstruction loss from the likelihood of the data. This approach helps in getting the initial values of the parameters.
So it’s like the normal, the classical autoencoder. From that we can get some initial estimate of $Z(i)$, the latent space variables. Then we perform clustering on $Z(i)$, and we let the clustering algorithm automatically determine the number of clusters and use that as the number of vertices. We also use the cluster centers as the initialization, as the initial values of $U$. So that’s the main part, the key ideas in our pre-training start, so that we can have a good initial addition to, for the start of the training. Another trick that we have taken is that in practice, the best performing existing trajectory inference methods will attempt... They are typically very fast. And in order to have comparable computational costs we also have accelerated version of our algorithm which is a simply to reduce the input, the dimension of the input space, so we can replace $Y(i)$, the high-dimensional vector of the gene expressions with it’s principal components. Now, principal component, principal scores, which is a low-dimensional vector $L$. We, by default we will take the first 64 dimensions for the principal scores.
And so we replace the elected binomial distribution by a normal gaussian distribution assumption of these principal scores. And as you will see later in our evaluations with real and synthetic data, we see that we actually have comparable performance with our previous likelihood, with our standard likelihood based methods for this accelerated version. So after the final step is that after the training the autoencoder, we have approximated distributions posterior distributions of the latent space and also the cell positions that and which vertex, the vertex or the posterior distribution of $C_i$, well $C_i^*$ is which vertex or edge the cell is from. And we need to use those information, to determine the trajectory backbone and to project each cell on our inferred trajectory backbone. So how we do that is, first we calculate an edge score. So this edge score is... So we have different scores for an edge, and that is determined on looking at the posteriors of cells. How many cells from the posterior distribution? How many cells choose to lie on that specific edge? If there are a lot of cells then that means it's very likely that edge exist. If there are very few cells
then very likely that edge should not be, the edge that is included in the trajectory backbone. And the denominator is that we want to, give a relatively high fair score for the edges of that connecting to small clusters, to small cell types, such as we want to also capture the transition between two rare cell types. So that’s why we have this regularization, waiting by with, in the denominator. And we include an edge in the trajectory backbone if it’s edge score is larger than some*. And when we have an inferred trajectory backbone, then the next step is, we want to project the cells on the inferred trajectory, and we do it by looking at the... Based on the posterior, approximated posterior distributions of the cell positions that *. We want to find the closest point on the inferred trajectory for this cell based on the posterior distributions. And the distance, this expectation we can also use as a evaluation or the cell positions. And the third thing we need to, because our final results is some kind of directed graph. So we need to determine the root vertex. So the root vertex can be either given by the user, or if as I feel I asked,
for some datasets like the data at the beginning of my talk, the cells are collected in the time series, and we can make use of that time series to determine the root vertex. The rough idea is that for each vertex, we can calculate a fraction time score, which is an average of the cells or projected on the edge that connects to the vertex, depending on the distance from the cell to the vertex. And so we can have some vertex collection time, collection time score for each vertex. And we choose the root vertex as the vertex that has the smallest collection time score. And with the roots and with our inferred trajectory, it’s straightforward to calculate the pseudotimes for each score. So that’s the whole process of our model-based methods for trajectory inference. And now let’s look at some benchmarking results. So first we... Our benchmarking includes both some real datasets and some synthetic datasets. And we follow the... Most of the benchmarking follows the same framework as this well known benchmarking paper in the “Nature of Biotech” in 2019. And our datasets are selected as a subset of the datasets that they have tried and our criteria is that these datasets,
maybe have enough number of cells not too few cells. And we wanted to cover different types of topologies. And this is the benchmarking results. So the columns, sorry. So the rows are the different datasets that I have mentioned. And we compare five different methods. So we have, would come first compare two versions of our approach. Vitae one as the original elected binomial likelihood base. Vitae and accelerated version, replacing the gene expression vectors by principal scores. Then we compare it with three different, state of the arch trajectory inference methods. The monocle PAGA. The monocle series is from the lab that, who developed the first trajectory inference methods, and there, and then they further, the first take monocle for one and now they have monocle three. So the monocle series are always commonly used in these single-cell RNA sequencing papers. And two, I expect the performing, trajectory inference methods in the benchmarking paper, the PAGA and Slingshot. And all these methods, do not use this time information explicitly. So it’s a fair comparison between these methods. And for all the methods they are given to by those.
We give them the two numbers of clusters or the vertices to start from, and the two root vertices. And we, and each column is, we compare its measurement, its metric for the evaluation of the performance of each method. So the first two columns, are the matrix for recovery of the trajectory topology or the trajectory structure. And next two columns are the evaluation of the cell position, estimation accuracy. And the last metric is for evaluating the pseudotime accuracy. And a larger score means a better performance, and a lower score means a worse performance. So you can see that, our approach has much better performance in recovery of the trajectory topology. We also have some benefits of the cell position estimates, and because of both, we have a better performance in the pseudotime accuracy. And the other thing you can see is that our, our accelerated version have comparable, slightly worse but comparable performance, compared to the, our likelihood based vitae. And though it has a much quicker, much less computational cost. So finally, let’s come back to the case study on mouse neocortex. So this is the,
the visualization of merging the raw data.
And this is the performance of our methods.
And for comparison, we compare is, another very popular use,
data integration method called Seurat.
So Seurat is the software, the most often used software,
for single-cell RNA sequencing analysis.
Their lab have developed a series of computational tools
for analyzing the single-cell RNA sequencing data.
And this is from their integration methods.
So you can see that both methods can,
is able to integrate the both two datasets,
but for some details, I think,
because we are assuming this trajectory structure,
because we are assuming this trajectory structure,
we have a slightly better performance.
For example, this group of cells are the layer one neurons,
where the group of here, are here in Seurat.
So you can see that because they come from,
because the outer layer parts and the layer parts,
come from two datasets.
Because as I mentioned earlier in dataset B,
they have collected cells from, at day 11.
So this are, we can take a look
of the collection days of each cell.
So you can see that these cells, they are,
the layer one parts come from day 11,
And the rest parts is a mixture
of cells from both two datasets.
And by the way they all belong to the layer one.
So we know that they belong to layer one
0:42:45.21 –> 0:42:48.21 by looking at the marker genes expression
0:42:48.21 –> 0:42:49.92 which I did not show here.
0:42:49.92 –> 0:42:54.92 So it’s because we encourage the cells to align together
0:42:55.01 –> 0:43:00.01 if they are along the address, if they are similar cells.
0:43:02.61 –> 0:43:04.603 And you can see here, that’s,
0:43:06.494 –> 0:43:10.48 so the two datasets, they have this interpolation
0:43:10.48 –> 0:43:14.27 of the pseudo, of the collection time.
0:43:14.27 –> 0:43:16.05 And you can see for example,
0:43:16.05 –> 0:43:18.32 for these projected cells,
0:43:18.32 –> 0:43:21.503 we can see this continuous positions,
0:43:23.73 –> 0:43:26.93 like alignments of the cells of different days
0:43:26.93 –> 0:43:31.93 from so the most the dark is the cells from day ten.
0:43:33.656 –> 0:43:36.248 And the red ones are the cells from day 18
0:43:36.248 –> 0:43:39.79 and even days are, are from dataset A,
0:43:39.79 –> 0:43:41.66 and odd days are from dataset B.
0:43:41.66 –> 0:43:43.15 So you can see that though they’re coming
0:43:43.15 –> 0:43:46.31 from two different sources,
0:43:46.31 –> 0:43:50.733 we can, we are able to align them in the right order.
0:43:54.24 –> 0:43:55.073 And,
0:43:56.17 –> 0:43:58.27 and as another comparison.
0:43:58.27 –> 0:44:03.16 So we compare our estimation of shared trajectory,
0:44:03.16 –> 0:44:05.55 with another partisan approach
0:44:05.55 –> 0:44:10.55 which is we’re first to do data integration with Seurat
0:44:11.5 –> 0:44:13.95 and then we can use Slingshots,
0:44:13.95 –> 0:44:18.147 to perform trajectory inference on the integrated data.
0:44:19.375 –> 0:44:20.63 And you can see that this,
0:44:20.63 –> 0:44:23.83 we have a much cleaner trajectory structure.
0:44:23.83 –> 0:44:28.83 And we also have a comparable computational cost.
0:44:29.09 –> 0:44:30.72 So Seurat and Slingshots,
0:44:30.72 –> 0:44:33.54 they cannot be, they do not need regularization.
0:44:33.54 –> 0:44:38.18 And with one CPU, they, it takes about 12 minutes.
And for our accelerated VITAE, generating this figure, we have, we can, we take about three minutes on one GPU port at about 10 minutes on eight CPU cores. Which is, the eight CPU cores are like currently, like most of our laptops, but we'll have such computational resources. So we have comparable computation cost with this state of our methods.

Based on this approximated posterior distributions we also have some kind of uncertainty quantifications on the cell positions. For example, here, it shall say some parts of the cells, these cell positions along the trajectory are not very reliable. And that will help us to evaluate our, the estimate, how we think our estimate in pseudotime.

And finally, this is showing some gene expression change along the pseudotime order. And, and we look at some top markers that are changing along the pseudotime order for some trajectories in the whole trajectory structure. And you can see, here we separately fish the curve for two datasets, but you can see that they overlap with each other quite well. And so that's also an evidence showing that we can have a good,
0:46:23.032 -> 0:46:26.14 a good performance in aligning the two datasets.
0:46:28.16 -> 0:46:30.753 So the take home message is,
0:46:30.753 -> 0:46:35.753 first we perform this model-based trajectory inference,
0:46:36.17 -> 0:46:38.25 to understand cell dynamics.
0:46:38.25 -> 0:46:41.2 And our, the second is our methods.
0:46:41.2 -> 0:46:43.46 So our method is a model-based approach.
0:46:43.46 -> 0:46:48.46 We can combine the mixture prior model, Oh, sorry.
0:46:49.99 -> 0:46:53.556 We can combine the collected mixture structure
0:46:53.556 -> 0:46:57.6 for defining the trajectory structure
0:46:57.6 -> 0:47:00.41 with the variational autoencoders
0:47:00.41 -> 0:47:03.333 so that we can efficiently,
0:47:04.71 -> 0:47:07.94 efficiently solve the mixture model
0:47:07.94 -> 0:47:11.91 and have enough flexibility to fit the data well.
0:47:11.91 -> 0:47:14.06 And so our,
0:47:14.06 -> 0:47:16.97 trajectory inference approach features,
0:47:16.97 -> 0:47:19.22 the analysis of integrating
0:47:21.9 -> 0:47:25.59 And if you are anxious to know more details,
0:47:25.59 -> 0:47:29.31 we have our paper, a manuscript,
0:47:29.31 -> 0:47:32.74 a manuscript already available on bio archives
0:47:32.74 -> 0:47:37.74 and we also have our package codes on VITAE.
0:47:38.05 -> 0:47:39.64 And that’s all. Thank you.
0:47:39.64 -> 0:47:44.401 And if you have any questions, I’m happy to answer
them.
0:47:44.401 -> 0:47:47.353 - Thanks Jingshu for this excellent, excellent talk.
0:47:49.09 -> 0:47:50.34 I wonder whether the audience,
0:47:50.34 -> 0:47:52.24 have any questions for Jingshu.
0:47:58.112 -> 0:48:01.55 Okay. So Jingshu I have some maybe minor questions.
0:48:01.55 -> 0:48:03.65 I recall that in the model,
0:48:03.65 -> 0:48:07.41 you have this actual term encouraging \( X \),
0:48:07.41 -> 0:48:11.68 the data explained by \( X \) the covariates,
the confounding covariates, to be orthogonal to the leading factor, right? And then there is a penalty term alpha. So I wonder, what’s the motivation for you to including this term instead of first removing the effect from the i directly. And in practice, how should we have set alpha? So we want to remove these confounding effects, right? But the other fact is that, these confounding effects, the X, are not exactly orthogonal with Z, because for instance for the two datasets that I have, we cannot say that the signal is completely orthogonal to each dataset they have come from because there are two biological differences between the two datasets. So, here, the problem is not completely identifiable but people do it in practice a lot. So we want to kind of decorrelate Z and X to some extent so that we can remove, remove the batch effects that we do not want but keep the two biological differences. So I think the underlying assumption is that we are assuming that the two biological differences are large enough so that compared to the... So we remove the smaller differences, the batch effects
but we can still pick the two biological differences, to some extent.
So there's no guarantee that it will work, but in practice, it work on a lot of datasets.
I think that's...
So we will inherit this idea from this paper by Nancy Huang and her students.
So I think in removing the batch effects.
So I think the idea is that, we hope that it can work for a lot of datasets.
And the reason that we want to have this penalty, is that if we don’t add any penalty,
 then, because this autoencoder is trained by this, by this stochastic gradient descent.
So sometime it may not find the optimal global solution.
So if we don’t encourage it, the X and Z to be decorrelated,
it sometimes may not be able, it may give you a solution that is not,
that’s the Z still are highly correlated with X and the batch effects are still there.
So then this alpha, I think in practice we set it to be 0.0,
so it’s a very small penalty so that we can put some, some kind of penalty to regularize that.
Small amount, I guess you mentioned.
Yes, yes.
And it may be that does not work, and then in practice you can choose another alpha and try it
and see if it gives you the best results that you want.
Right. So, so I want...
I guess, right.

So I guess my question is like, since it’s not entirely a supervised problem,

like how, right.

So I’m not sure how to check,

what is a good alpha in the sense,

but if you tell me like a small alpha,

well you’re going to be fine

then I just take it to be small alpha.

Yeah. I think the way you check it is that,

for example the way we check it here is that,

so sometimes we have some referenced cell types,

so that, you know roughly what you are doing.

So here, for example,

here this reference cell types, these are,

these are not used in the modeling approach,

so these are for evaluation the performance.

And for some datasets, you can’t,

we can mark our genes and do some class points,

so, you know, roughly like which though,

(indistinct)

and for two datasets, you can see,

like whether you can correctly merge the cell types

that are shared among the datasets

but keep the cell types

that are unique to different cells.

Then for our trajectory inference is slightly complicated

because these cell types are not well separated.

So another way that we can check our performance,

is that you can see here that we can correctly,

for these projected cells we can correctly like,
order the wrong days in the right order.

So that we know that we keep some biological meaningful signals there.

I think there still can be some bias. Yeah.

Okay, great. Thanks.

So, thanks Jingshu again for this excellent talk.

And if you have any question you want to ask Jingshu that you cannot think about for now,

you can always email her offline and if you want to use her software,

I think she’ll be more than happy to answer your question.

Yes. Yes.

(chuckles)

So I guess that’s all for today.

Thank you everyone joining.

Thank you Jingshu for being here, and have a nice remaining day.