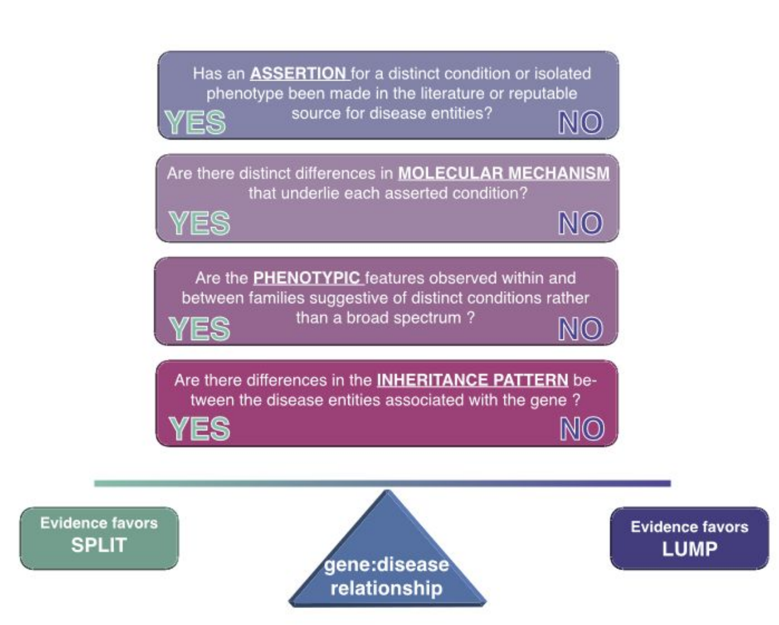
Dimensionality Reduction of RNA-seq Data by Variational Autoencoder

G.B. Wiersum
Northwestern University

To Lump?
Or Split?



Has an **ASSERTION** for a distinct condition or isolated phenotype been made in the literature or reputable source for disease entities?

YES NO

Are there distinct differences in **MOLECULAR MECHANISM** that underlie each asserted condition?

YES NO

Are the **PHENOTYPIC** features observed within and between families suggestive of distinct conditions rather than a broad spectrum ?

YES NO

Are there differences in the **INHERITANCE PATTERN** between the disease entities associated with the gene ?

YES NO

Evidence favors SPLIT

gene:disease relationship

Evidence favors LUMP

Thaxton et al. 2022

- RNA seq has emerged as an extremely powerful tool.
- With higher resolution, come more opportunities to identify more distinct cell types.
- How do we decide when “different” is different enough?
- To make these decisions, rigorous, reproducible methods are needed to reduce data dimensionality.

Beyond MSA

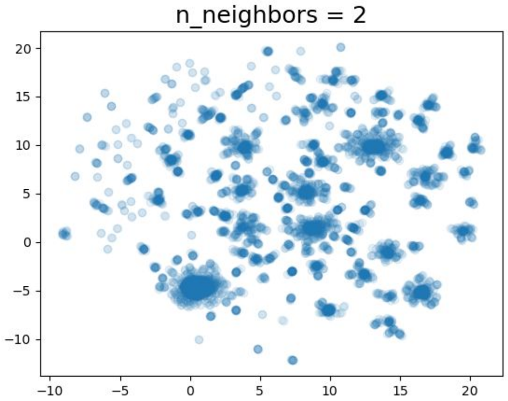
The traditional workflow tools:

- Multi Sequence Alignment
- Distance Metrics
- Principal Component Analysis
- K-means clustering
- t-Distributed Stochastic Neighbor Embedding (t-SNE)
- UMAP

With the exception of the MSA itself these methods are all destructive. The original data cannot be reconstructed from the embeddings.

These methods are highly susceptible to bias - choosing values for K, or a number of principal components, can easily change the outcome.

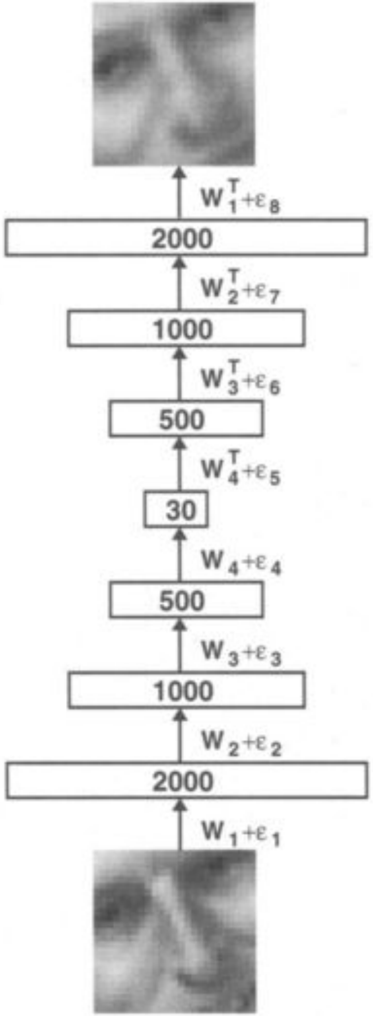
The number of clusters often depends on the number you expect to see.



How many clusters do you see?

Variational AutoEncoders

Hinton and Salakhutidinov, 2006



Variational autoencoders take advantage of several of the strengths of neural networks.

By being “Embarassingly Parallel” they can be easily GPU accelerated.

VAEs do not require alignment. (They learn from context, not position)

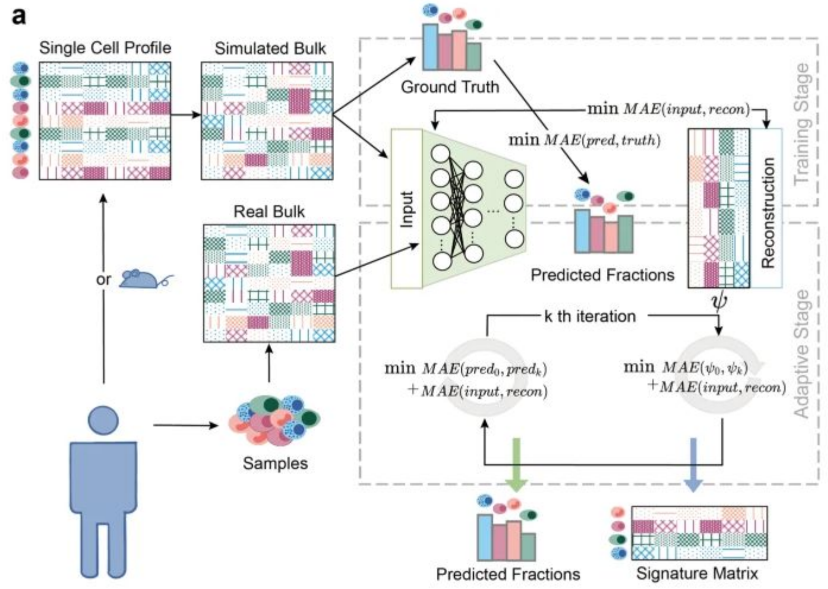
Data is ingested, compressed and reconstructed from the compressed encoding.

Loss is defined as the total difference between the input and reconstruction.

The goal is to faithfully represent the variation of input data in as few neurons (weights and biases) as possible.

Autoencoders for scRNA-seq

Fig. 1: TAPE workflow and clarification of adaptive stage.



Chen at al, 2022

Training takes advantage of the enormous bulk of information publicly available.

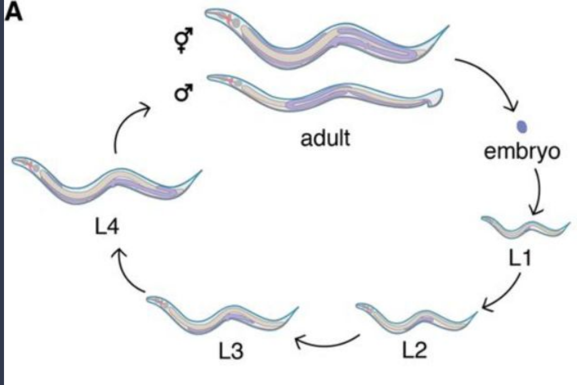
RNA profiles are fed into the autoencoder model

Output is dimensionally reduced signatures.

Fine-tuning can be used to provide more fine-grained detail of cell subtypes (transfer learning).

Noisy data is OK!

Methods



Roach et al

Training Data:

The full-length transcriptome of C. elegans using direct RNA sequencing - Roach et al. 2020

Larval stage 2, 3, 4 and young adult C. elegans were separately collected and underwent RNA extraction (whole organism).

Processed by nanopore RNA seq

Fastq data downloaded from WormBase

Data Preparation

For nucleic acid data, one-hot encoding is standard practice: A is coded as [1,0,0,0], T/U as [0,1,0,0] etc.

Each file is about 500MB - 200k to 1M reads, which poses a challenge: Processing a 4GB dataset in RAM becomes impractical with commonly available hardware.

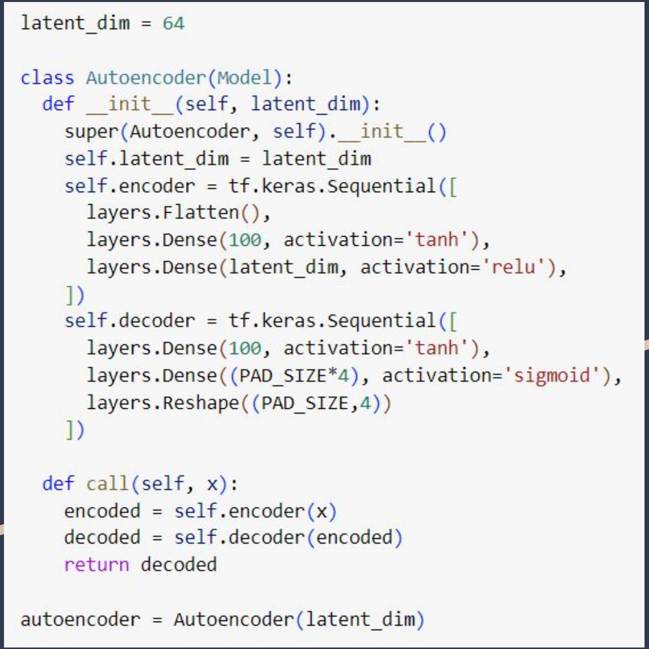
Fastq data is read line-by-line and one-hot encoded.

These one-hot encodings are “padded” to a uniform size.

Sequences that are larger than the pad-size are randomly cropped to size.

Because of memory constraints, files have to be read in fractions

Building a “toy” model



```
latent_dim = 64

class Autoencoder(Model):
  def __init__(self, latent_dim):
    super(Autoencoder, self).__init__()
    self.latent_dim = latent_dim
    self.encoder = tf.keras.Sequential([
      layers.Flatten(),
      layers.Dense(100, activation='tanh'),
      layers.Dense(latent_dim, activation='relu'),
    ])
    self.decoder = tf.keras.Sequential([
      layers.Dense(100, activation='tanh'),
      layers.Dense((PAD_SIZE*4), activation='sigmoid'),
      layers.Reshape((PAD_SIZE,4))
    ])

  def call(self, x):
    encoded = self.encoder(x)
    decoded = self.decoder(encoded)
    return decoded

autoencoder = Autoencoder(latent_dim)
```

Autoencoder model consists of two modules: Encoder and Decoder

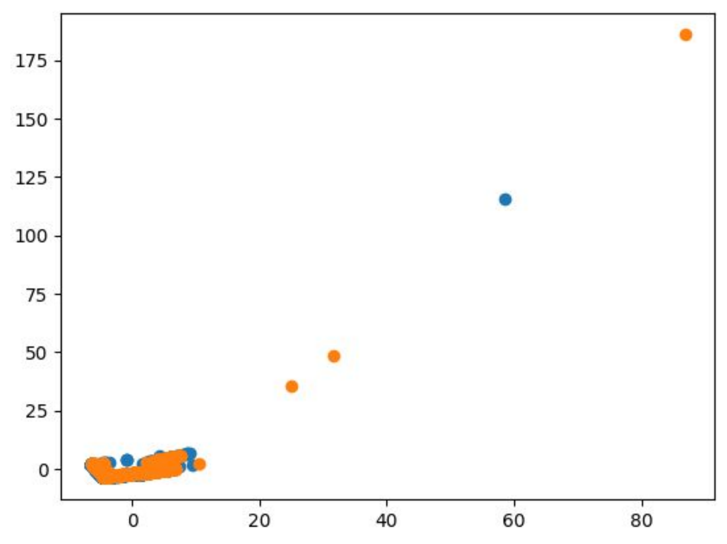
Encoder flattens sequence and feeds into 100 neurons with tanh activation. These neurons feed into the central “code layer”

Decoder reads from the “code layer” and reverses the process. Encoding is read to 100 neurons by tanh activation. These are densely connected to a layer the original size of the input.

Decoded output is compared against input to determine gradient for backpropagation.

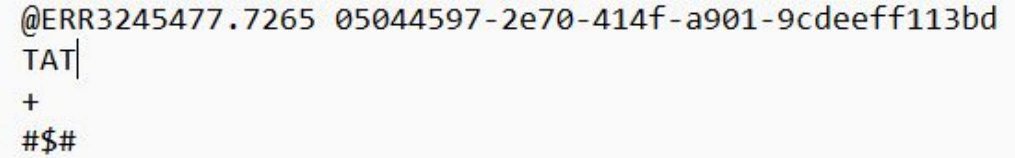
Training iterates over batches which are read sequentially from the files.

PCA-enabled QC

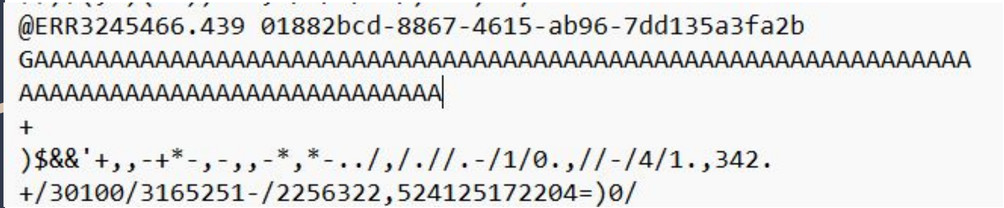


The result is a dimensionally reduced set of sample features. Rather than a sequence of 1000+ nucleotides, we have 64 eigenvalues

Extreme outliers become obvious -

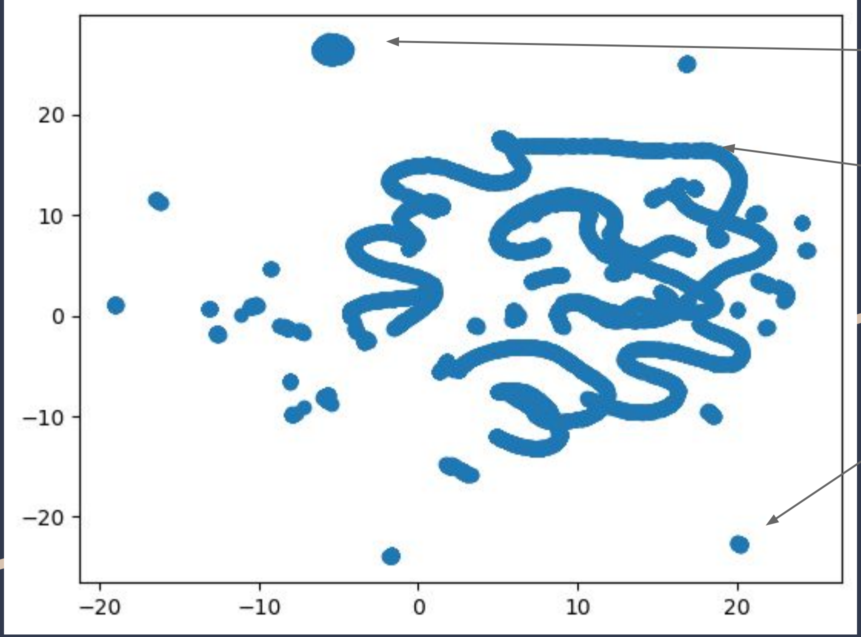


```
@ERR3245477.7265 05044597-2e70-414f-a901-9cdeeff113bd
TAT
+
#$#
```



```
@ERR3245466.439 01882bcd-8867-4615-ab96-7dd135a3fa2b
GAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
+
)$&&'+,,-+*-,-,,-*,*-../,/.//.-/1/0.,//-/4/1.,342.
+/30100/3165251-/2256322,524125172204=)0/
```

UMAP-based deconvolution



By passing encodings through UMAP, we can begin to deconvolve RNA reads.

Running examples from each subgroup through BLAST:

Island at top contains: Ribosomal protein L4, Nucleobindin, ATP synthase subunit gamma, S-adenosylmethionine synthase-3

Line contains ribosomal proteins

Bottom right contains mitochondrial reads

Thank You