ScVI Models

The Model

Input — As previous presenters have explained, scVI takes in a gene expression matrix with annotated batches as input

Output — An embedding into a latent space

In between, scVI tries to learn an embedding (that is non-linear) of the cells that can be used for clustering, getting differential expressions etc.. One can specify the number of hidden layers in between.

Training the model on the cortex dataset

The model we are going to use here is the scvi model. There are other models (like TotalVI) that are in the same package (scvi-tools). We will see its applications on the cortex dataset.

At first we set up the annotated data for the model to use. As we talked about during pre-processing, we can do different types of feature selection.

```
sc.pp.highly_variable_genes(
    adata,
    n_top_genes=2000,
    subset=True,
    layer="counts",
    flavor="seurat_v3"
)
```

After we have set up the anndata (that is stored in the variable **adata**), we run

```
model = scvi.model.SCVI(adata)
```

```
model.train()
```

The default runs a hidden layer of neural network with some dropout rate for a specified number of epochs (defaults to four hundred). Let's take a look at the parameters of the model. We can specify the number of hidden layers and the number of neurons.

Model Summary

We can print out the details of the model that scVI defaults to.

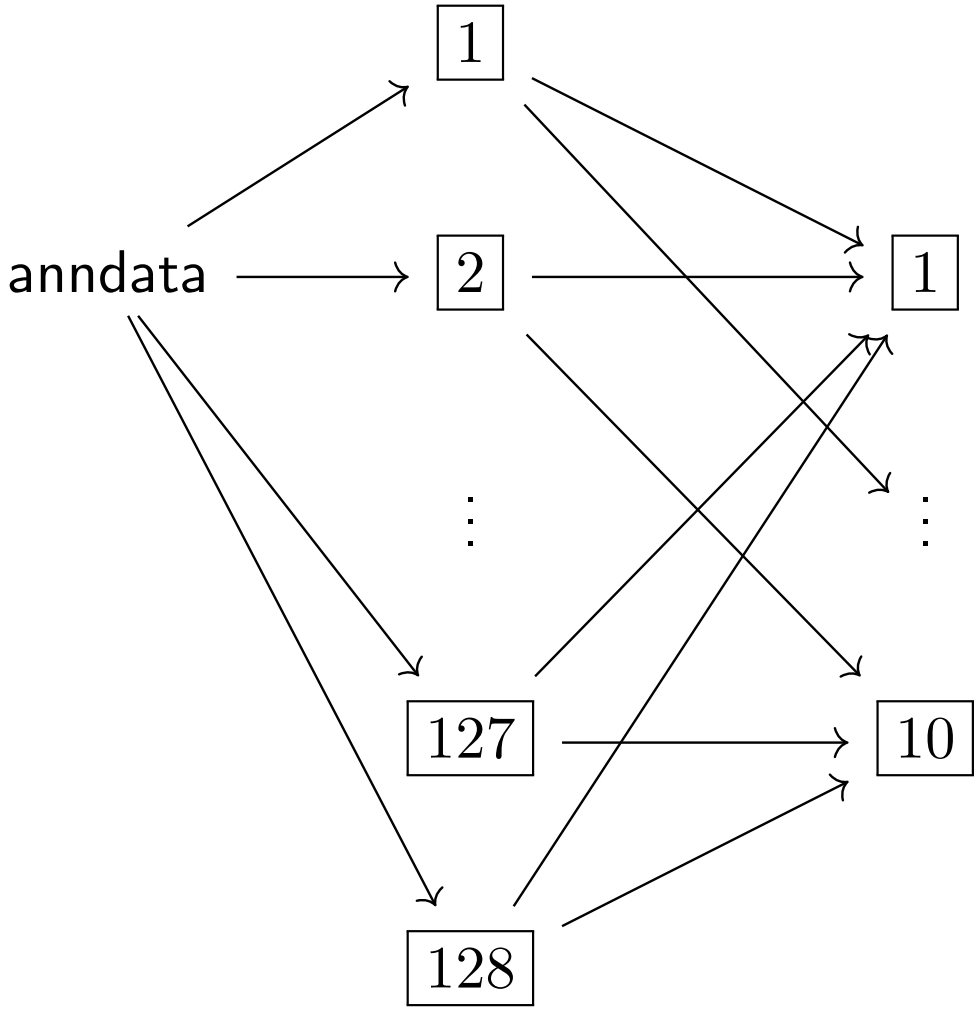
```
Hidden neurons - 128
Latent neurons - 10
Dispersion - genes
Distribution of latent neurons - normal
```

I tried seeing if my understanding of the model was accurate so I printed out the model summary of the TotalVI model (which also comes in with scvi-tools) and got

```
Latent space  - 20
Latent distribution - normal
Gene dispersion - genes
Protein dispersion - protein
```

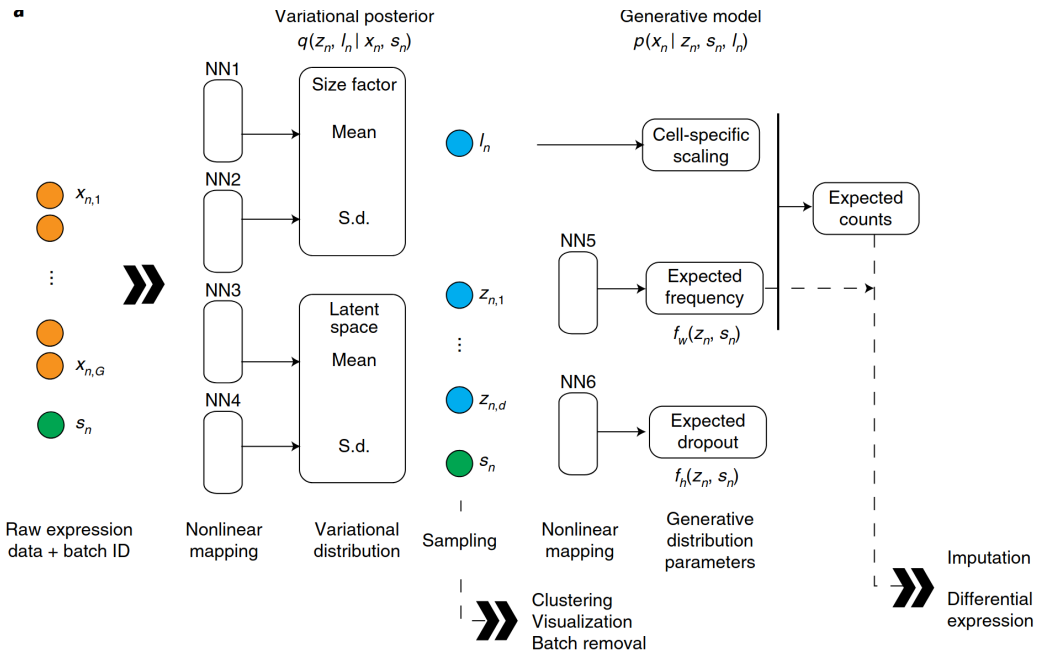
and it doesn't mention the number of hidden layers. I am a little confused.

What I thought it did



What I was missing was the steps before and after the hidden layer.

What it actually does

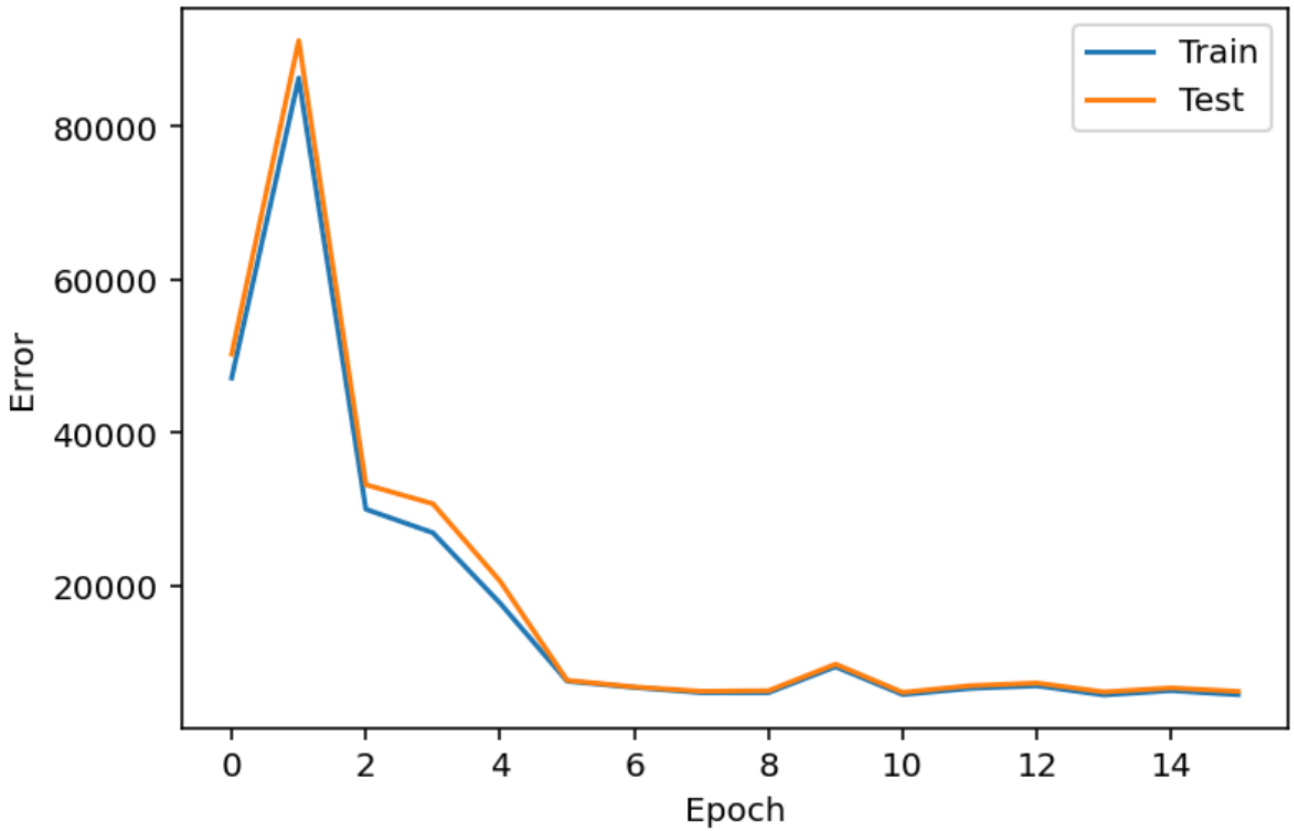


Output

We can find the latent representation of the anndata from the model through

```
latent = model.get_latent_representation()
```

What we can do with this (computing tSNE for instance) will be explained by another team member.



Convergence of loss function on the cortex dataset while training. What loss is it using here(??)