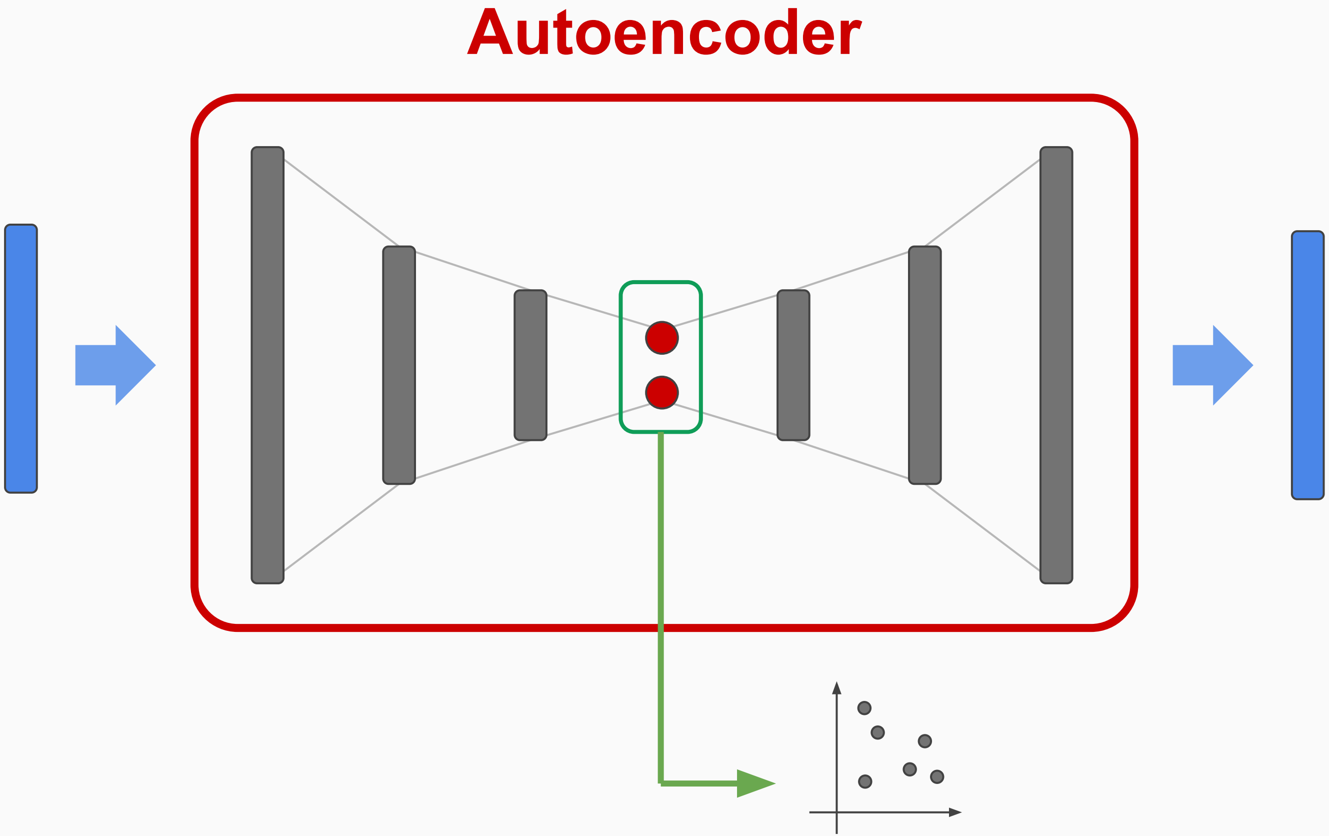
Progress Report

Kuei-Yueh (Clint) Ko

Idea of autoencoder



Preprocess Data
(exact same procedures as approximating UMAP using NN)

```
[0] AMJ_5L_Costim.txt
[1] B6901GFJ-08_Costim.txt
[2] AMJ_5L_CMV_pp65.txt
[3] B6901GFJ-08_CMV_pp65.txt
[4] AMJ_5L_SEB.txt
[5] B6901GFJ-08_SEB.txt
```

Standardization

Sub sample 10^4 events / cells twice for each sample
=> 12 sub samples

Get one sub sample for each group

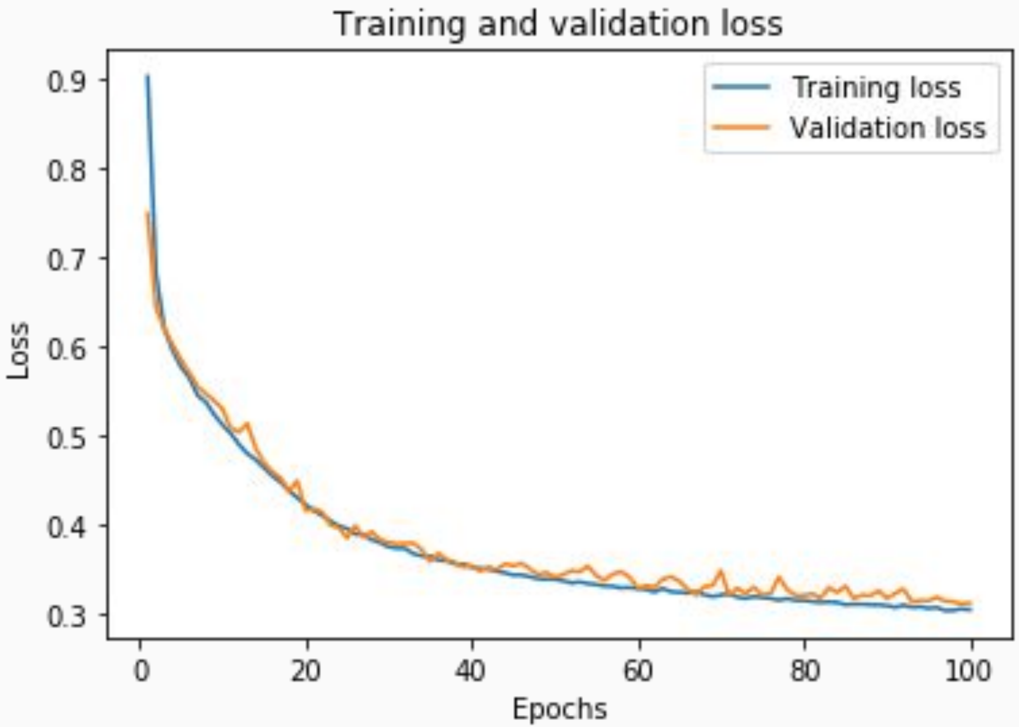
Split to train (90%) and test (10%) dataset

Structure of the autoencoder I used

```
#   InputLayer (None,  14) <--
#        Dense (None, 128)
#        Dense (None,  64)
#        Dense (None,  32)
#        Dense (None,  16)
#        Dense (None,   2) <--
#        Dense (None,  16)
#        Dense (None,  32)
#        Dense (None,  64)
#        Dense (None, 128)
#  OutputLayer (None,  14) <--
```

```
model = Sequential()
model.add(Dense(128, input_shape=(14,), activation='relu'))
model.add(Dense(64,  activation='relu'))
model.add(Dense(32,  activation='relu'))
model.add(Dense(16,  activation='relu'))
model.add(Dense(2,   activation='relu'))
model.add(Dense(32,  activation='relu'))
model.add(Dense(16,  activation='relu'))
model.add(Dense(64,  activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(14))
model.compile(loss='mse', optimizer='rmsprop', metrics=['mse'])
```

Training: Loss over Epochs



Set up the Encoder

```
model.input
```

```
<tf.Tensor 'dense_42_input:0' shape=(?, 14) dtype=float32>
```

```
model.layers
```

```
[<keras.layers.core.Dense at 0x7f9ba405ab70>,
 <keras.layers.core.Dense at 0x7f9ba405aa58>,
 <keras.layers.core.Dense at 0x7f9bcc225f98>,
 <keras.layers.core.Dense at 0x7f9bd22a6ac8>,
 <keras.layers.core.Dense at 0x7f9bcc21ba90>,
 <keras.layers.core.Dense at 0x7f9bcc206c88>,
 <keras.layers.core.Dense at 0x7f9ba413f668>,
 <keras.layers.core.Dense at 0x7f9bc407bc88>,
 <keras.layers.core.Dense at 0x7f9b9443d2e8>,
 <keras.layers.core.Dense at 0x7f9b94458908>]
```

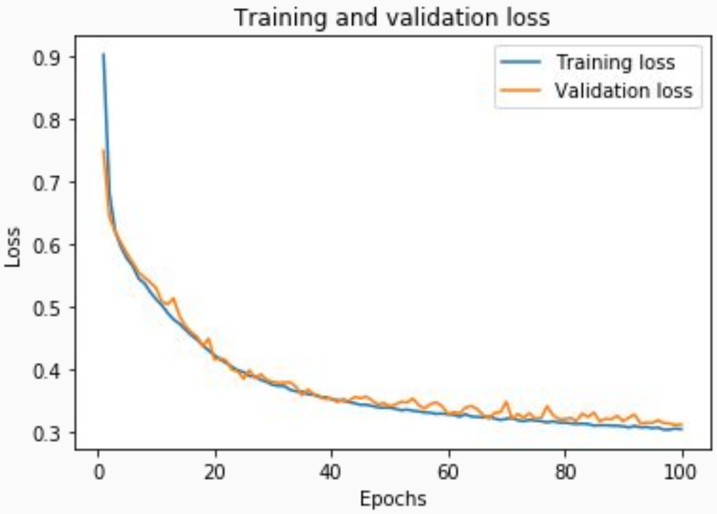
```
model.layers[4].output
```

```
<tf.Tensor 'dense_46/Relu:0' shape=(?, 2) dtype=float32>
```

Set up the Encoder

```
XX = model.input
YY = model.layers[4].output
F = K.function([XX], [YY])
encoder = lambda X: F([X])[0]
```

Run Encoder on Train and Test

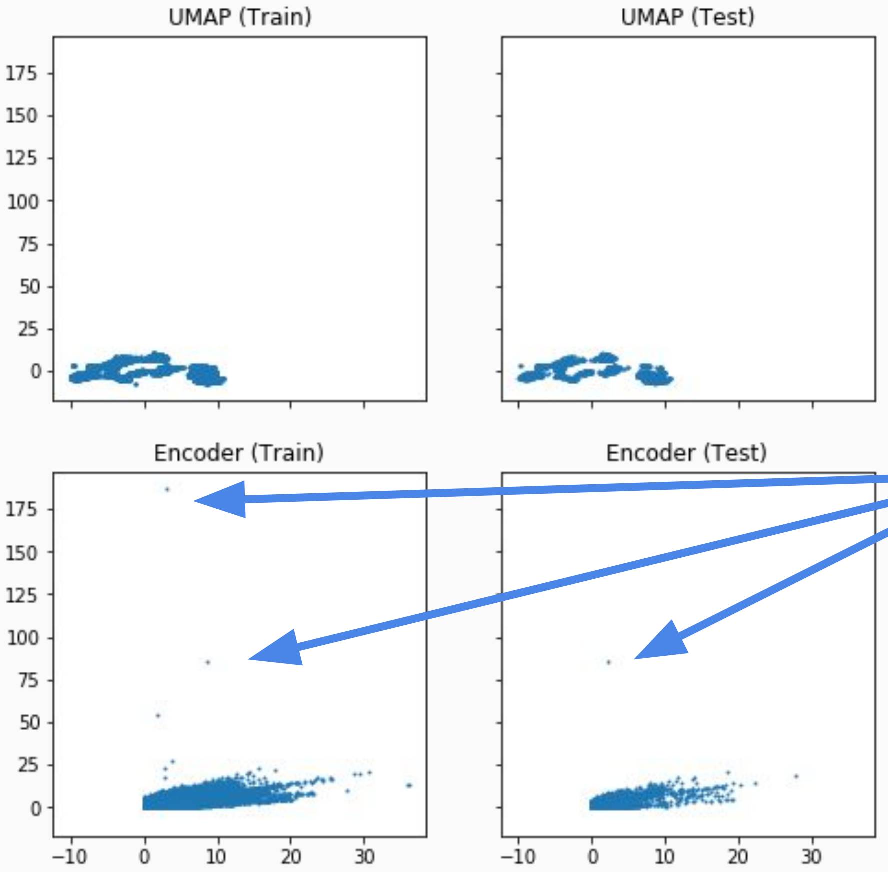



```
encoded_out_train = encoder(X_train)
encoded_out_test = encoder(X_test)

print(encoded_out_train.shape)
print(encoded_out_test.shape)
```

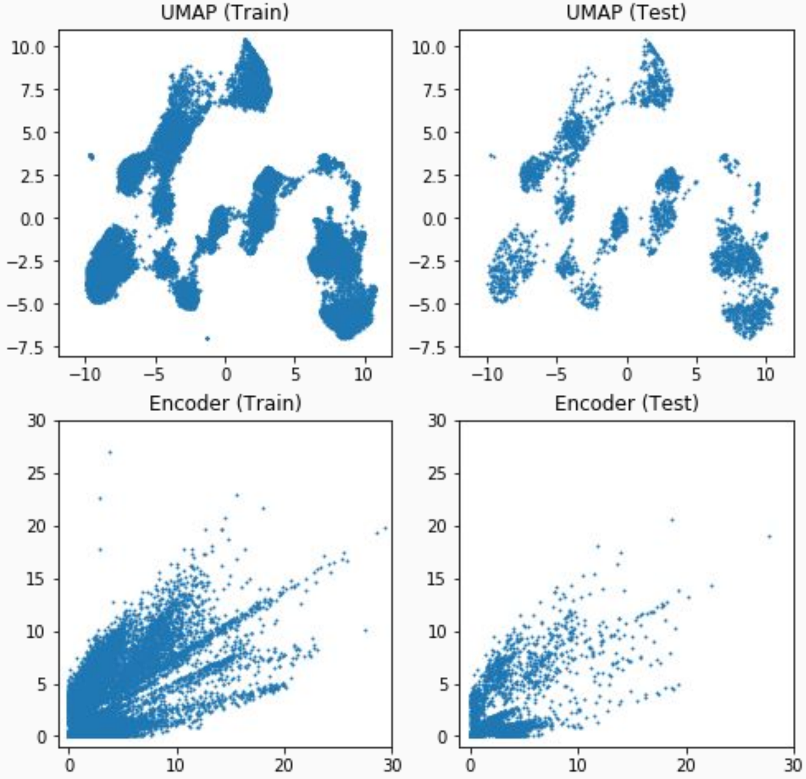
```
(27000, 2)
(3000, 2)
```

Visualize the “Encoded” Train and Test



There are some points with extreme values

Visualize the “Encoded” Train and Test



UMAP

Autoencoder

Train (90%)

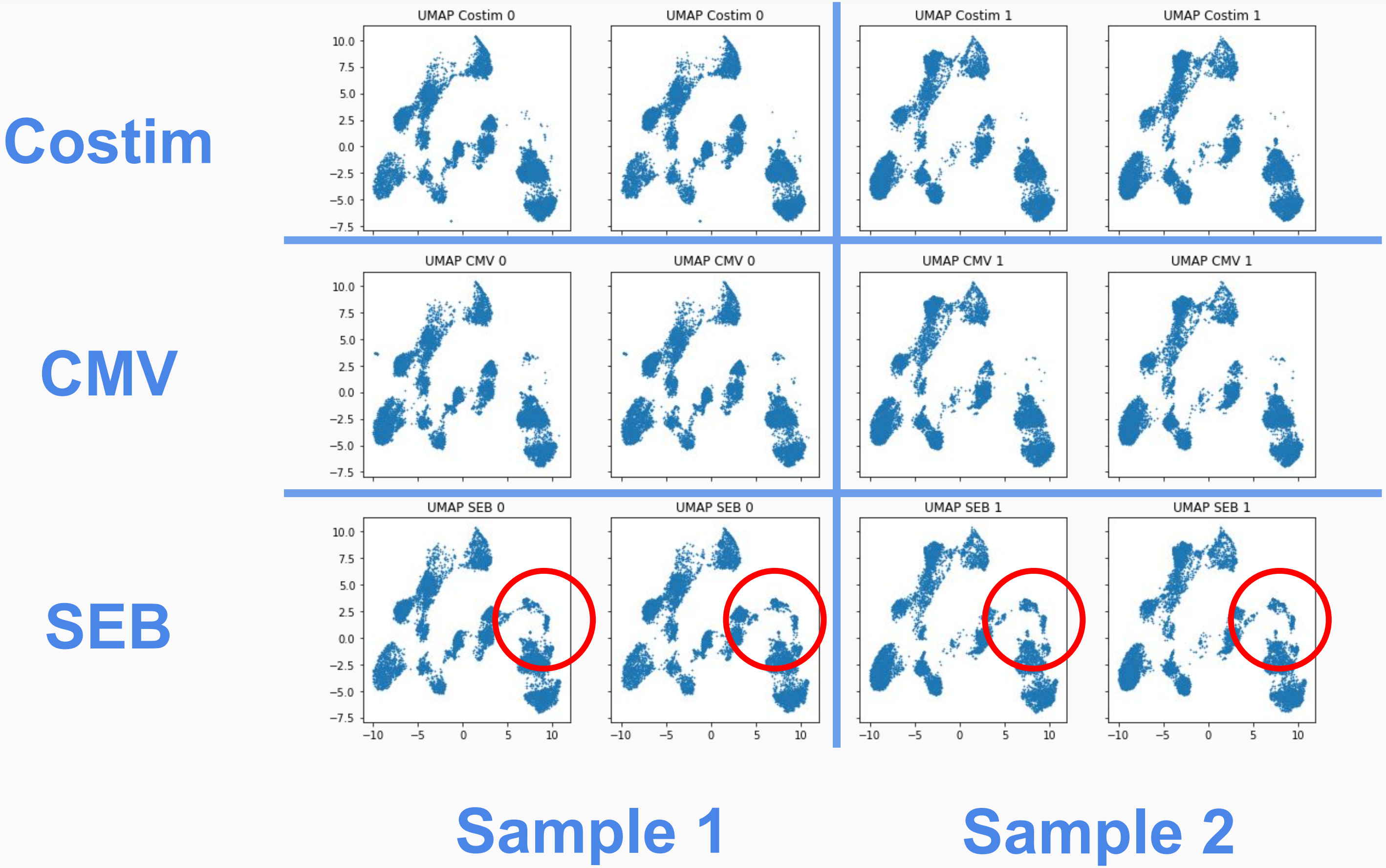
Test (10%)

Note:

Here I zoom in to where the most points are

The train and test are mixture of Costim, CMV, and SEB dataset

Apply the Encoder function to other sub samples and compare with UMAP



Apply the Encoder function to other sub samples and compare with UMAP