**Stacked Autoencoder for Multi-Omics Data Integration for Breast Cancer Biomarker Discovery**

This project aims to use a stacked autoencoder to integrate multiple types of omics data and identify potential biomarkers for breast invasive carcinoma.

**Goal:** Design a multi-omic integration model that can transform more than 2 high - dimensional modalities into an interpretable latent space of low dimensions by performing better model validation tests for model functionality.

**Some Pre-requisites**

To get started, you will need to have the following installed on your machine:

* Python 3 and above
* NumPy
* Pandas
* Scipy
* TensorFlow
* Keras
* Scikit-learn

**Data**

The data used to train this SAE network is publicly available and was downloaded from <https://artyomovlab.wustl.edu/phantasus/> which has TCGA data for each of the cancers. We downloaded three datasets – RNAseq gene expression, Proteomics Data, and DNA Methylation Data for Breast Invasive Carcinoma

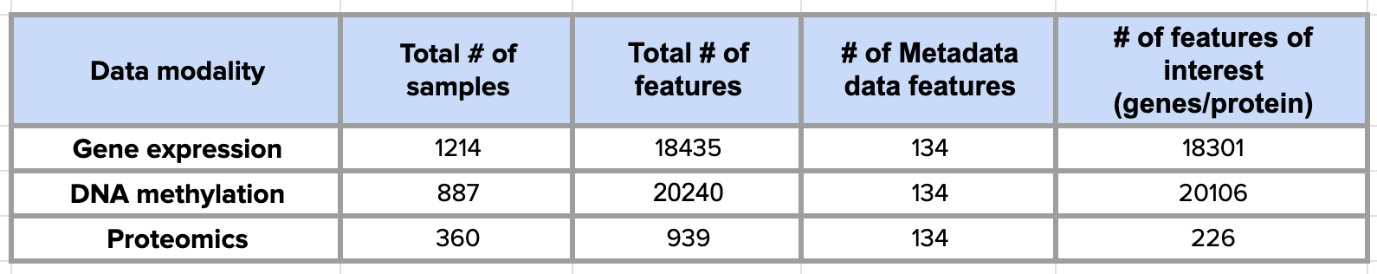
**Introduction**

We are proposing a method to integrate three types of omics data - RNAseq data for gene expression, Proteomics expression data, and DNA methylation data for bulk data that we have downloaded from The Cancer Genome Atlas (TCGA) consortium. In this project, we have leveraged a key advantage of SAEs to understand complex relationships between different omic datasets, which are generally difficult to find using traditional methods. Our SAE set-up has a layer that compresses the features of each data modality by preserving the most critical information from each dataset into a smaller, comprehensive latent space of its own. Then the latent space is concatenated and fed into another layer of stacked encoder so that it identifies non-linear relationships between the latent spaces of different data modalities.

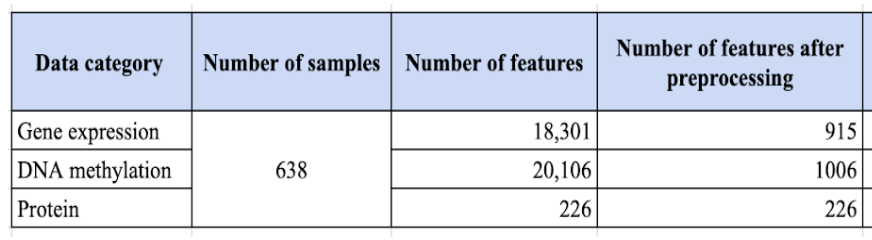
**Pre-processing of Data**

The pre-processed data is available in the *preprocessed data* folder (in the zipped folder). All 3 types of Metadata was cleaned to remove 134 features that were not appropriate for our analysis. We then found common patient ID’s across the three omic datasets, and kept these to have same rows across all datasets. To pre-process the data in the same way as we did, access the **Preprocessing.ipnyb** file in the Notebooks folder.

These were the number of features for each of the datasets before processing:



These were the dimensions of the data after pre-processing:



Feature engineering/selection was performed by filtering out the highly variable genes (>95th percentile), for gene expression and DNA methylation datasets. We retained the proteomics data’s features as is, as it had only 226 features.

**Model Execution**

The models can be found in the *Model* folder. There are separate notebooks for each part of the encoder-decoder architecture that can be called as sklearn objects while compiling the code.

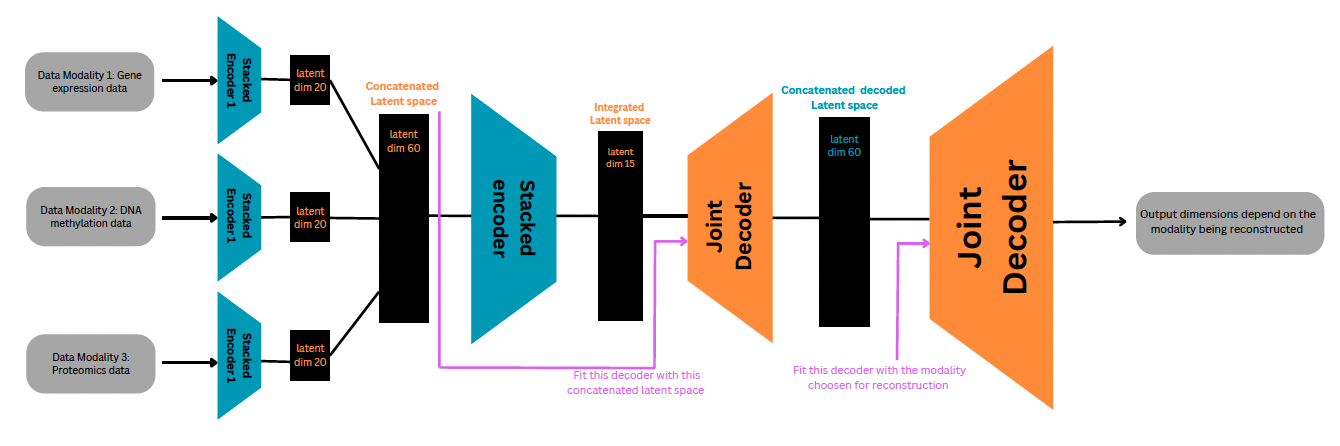


Figure : Model Architecture

Our project proposes **two applications** as implementations:

* Application 1 - **Predictive Model:** To predict whether a given sample based on gene expression, protein expression and DNA methylation data, is a normal or tumor sample for breast invasive carcinoma. This is made possible by our Model architecture as it learns patterns across all datasets for the patient and classifies the patient as normal or tumor. We have a basic Support Vector Machine (SVM) classifier that works as the classifier in this application. Please use **Application\_1.ipnyb** from the Notebooks folder to execute this implementation.
* Application 2 - **Feature Attribution for Biomarker Discovery:** It is very important that the Models employed should have biological interpretability and we set out to understand how our model has learnt biological patterns across datasets and what these can help us discover. We plotted histograms of each of the 15 features (from the output of our Stacked encoder 2), for each sample type (in the same bin) and checked which of these learnt features separate our sample types well using Earth Mover’s Distance (EMD) as a metric to check the separation between the sample types. EMD was used as the sample type representations were different, and each of the histograms had different number of data points. Please use **Application\_2.ipnyb** from the Notebooks folder to execute this implementation.

**Testing for Model Functionality**

We performed three functionality tests:

1. Graphical reconstruction test: We plotted the reconstructed data from our Joint Decoder against our original data modality to see how highly correlated the two dataframes are.
2. Pearson Correlation mean: We also calculated Pearson correlation mean between the original dataset and reconstructed data by our models (not the most accurate method but can be a supportive method to the first one).
3. UMAP: To test for the interpretability of our integrated latent space we used UMAP colored by sample type to observe the clustering pattern of the integrated latent space.

To run these functionality tests, please use the **Model\_functionality\_check.ipnyb** in the Notebooks folder.

**Note:** The choice of activation, the positions of batch normalization and other additional techniques in the encoder and decoder designs were based on many trial and error rounds based on these functionality tests.