BIOINFO 593 Final Project - Single Cell Perturbation

Mingyu Zhong mingyuz@umich.edu **Fang Xie** fangxf@umich.edu **Chenhan Li** chenhanl@umich.edu **Junfeng Yang** yjunfeng@umich.edu **Ruide Li** ruideli@umich.edu

Abstract

This project addresses the challenge of predicting gene differential expression values (DEVs) across various cell types and drugs (perturbagens) in the context of bioinformatics. With a dataset encompassing six distinct cell types and 145 perturbagens, our challenge is to predict 7,817 gene differential expression values (GDEV) based on the name of the tuple pair. To address the challenge, our approach combines feature engineering and probabilistic models, including Variational Autoencoders (VAE), VAE distribution model, Gaussian process (GP) & Denoising VAE, and Conditional diffusion. By integrating these methods, our project offers a comprehensive solution to accurately predict gene DEVs, contributing to bioinformatics research and its applications in drug discovery and disease understanding.

1 Introduction

In bioinfomatics field, a crucial challange is Understanding complex interactions among genes, RNAs, proteins and environmental factor. Earlier efforts have been made towards predicting low-dimensional observations in response to perturbagens, such as viability or genes of interest. A stride machine learning, particularly in deep learning of single-cell RNA sequencing(scRNA-Seq), have revolutionized our approach to studying single-cell omics. The development of models capable of predicting cellular responses to perturbations has become a key endeavor in systems biology. The work by Yuge Ji et al [1], underscores the effectiveness of deep learning models in predicting cellular responses to perturbations, which is a central aspect of our own endeavor.

Furthermore, the integration of single-cell RNA sequencing (scRNA-seq) data has provided unprecedented insights into cellular heterogeneity. The methodologies and frameworks employed by Dixit et al. with Perturb-seq[2] and Peidli et al. with scPerturb [3] provide foundational insights into how cells react to genetic alterations and the importance of harmonized data for understanding diverse cellular responses and paves the way for more sophisticated analyses using deep learning techniques.

Despite the progress, challenges exists in linking perturbations to alterations in cell states in a high-throughput manner due to the resource-intensive nature of experiments and the scarcity of large-scale single-cell perturbation datasets[1]. Predicting gene differential expression values (DEVs) in response to perturbations is therefore a potential solution.

The dataset and problem are from a kaggle competition: Single-Cell Perturbations. To navigate these pitfalls and challenges, Our approach integrates advanced predictive modeling for analyzing differential gene expression across a diverse array of cell types and perturbagens. To further refine our model, we employed feature engineering strategies that utilize a rich dataset of 240,090 cells and 21,255 genes. For this project, we aim to build a robust machine learning framework capable of accurately forecasting gene expression changes in different cell types, which can potentially broaden the horizons of medicinal chemistry or repurposing in drug discovery.

2 Problem

The model's input will consist of the perturbagens and cell types. The output will be 7,817 gene Differential Expression Values (DEVs) for each perturbagen and cell type pairing, pooled and Limma transformed across different donors, replicates and batches as opposed to the DMSO control. Specifically, the dataset includes six cell types: B cell, Myeloid cell, NK cell, CD4+ T cell, CD8+ T cell, and T regulatory cell. It contains information on 145 perturbagens. For four cell types (NK cell, T cell CD4+, T cell CD8+, and T regulatory cell), complete gene DEV data for all combinations with the 145 perturbagens are available. However, for B cells and Myeloid cells, only 15 paired gene DEV data is provided. The primary goal is to predict the gene DEVs for the remaining combinations of these two cell types, utilizing transfer learning from the complete data of the four cell types.

The main challenges are threefold: (1) a limited sample size, (2) the absence of specific features for cell types or perturbagens, and (3) high-dimensional, correlated outputs. The original dataset comprises only

614 samples. After partitioning into holdout and test sets, the training dataset is reduced to 473 samples, posing difficulties for models with large parameters, such as neural networks and non-parametric models that rely heavily on data. Lastly, the challenge is compounded by the 7,822 gene DEVs represented as a high-dimensional vector. These genes are interrelated, exacerbating the issue, as few methods can handle situations where the output dimension greatly exceeds the feature space, especially with the limited sample size for training. This complexity means that errors in predicting one gene DEV could adversely affect the predictions of others.

3 Approach

We aim to address the three main challenges individually. We first implemented customized feature engineering using databases and pretrained models to have the DL model learn latent representations of both perturbations and cell types. We next used probabilistic models that are efficient in modeling the data generation process to overcome the limited sample size. The chosen probabilistic models include the Variational Auto Encoder, Gaussian Process Regression, and conditional Diffusion model. These models are capable of handling multidimensional outputs and modeling the interdependencies among the genes.

3.1 Baseline - Dimensional Reduction with Name to Index Prediction

Our first strategy involves transforming the prediction task into a dimensionality reduction problem. Here, an encoder compresses the top 7,817 highly variable genes into a two-dimensional latent space, and then maps this latent representation back to the 7,817 gene DEVs. This approach bases its gene DEV predictions solely on the input cell type and perturbagen categories and provides a baseline for future optimization.

Since the cell type and perturbagen inputs are strings, they are converted into numerical representations using a word-to-index method. This involves assigning a unique, incrementing positive integer to each category. The resultant two-dimensional latent space do not follow a standard normal distribution. Instead, it is a Gaussian distribution with the means of each dimension corresponding to the indices of cell types and perturbagens, and with a unit variance. Once the model is adequately trained, providing the 2D index vector of (cell type, perturbagen) tuple pairs to the decoder yields the corresponding final gene DEV.

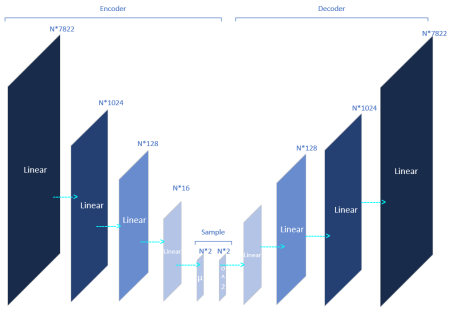

Figure 1: Baseline Variational Auto Encoder for Cell Type and Perturbagen.

3.2 Feature Engineering

3.2.1 Cell Type

We implemented single-cell variational inference (scVI), a deep generative probabilistic modeling approach based on the architecture of VAEs [4] [**?**]. The approach encodes scRNA-seq data into a low-dimensional latent representation where the underlying biological signals are effectively captured. We subsetted the dataset by 1/6, filtered cells and transformed the raw count data into normalized data. We used the top 8k genes ordered by variance. We trained scVI and scANVI models with raw counts. scANVI was used because it is able to model cells without label information, more accurately address the annotation problem and benchmarking probabilistically. The model succeeded in separating out myeloid cells and B cells from T cells (k-means clustering). NK cells and CD8+ T cells were in close proximity as expected because of their similar cytolytic programs (SI Fig.2). The computed key indices from scANVI of these clusterings showed higher levels than those of scVI, indicating its better capturing of biological structure in latent space.

3.2.2 Perturbagen

We used DeepChem that make the use of deep neural networks for chemistry to extract features of each perturbagens. We transformed the 145 molecules (perturbagens) into Simplified Molecular-Input Line-Entry System (SMILES) representation as by RDKit, and then loaded the inputs for the Deep Molecule library to extract various features using pretrained featurizers. Specifically, we used libraries including ”MACCSKeysFingerprint”, ”MordredDescriptors”, ”PubChemFingerprint”, ”Mol2VecFingerprint”, ”CircularFingerprint”, and ”RDKitDescriptors”. For instance, RDKitDescriptors compute values for a list of physicochemical properties, such as molecular weight, polar surface area and hydrogen bond donors/acceptors. These computational methods can effectively extract the high-level properties inherent in the perturbagens for prediction-based approaches. Given the excessive

dimensions generated by featurizers, we further applied Principal Component Analysis (PCA) to reduce the dimensionality with the threshold set as 97% total explained (SI Fig 1).

3.3 Feature-Based Variational Auto Encoder

By utilizing the resultant feature vectors described above, we obtained more informative features for each cell type and perturbagen, rather than the simple categorical indices as model inputs. We focused on modeling the underlying data generation process and continued to employ the Variational Autoencoder (VAE) model architecture. To enhance feature vector extraction performance, we increased the complexity of our models by incorporating self-attention and cross-attention blocks in addition to the original dense layers. Additionally, to mitigate the vanishing gradient issue and to accelerate convergence in this deep and complex architecture, we implemented skip connections and cross-attention between the encoder feature extraction results and the decoder hidden states.

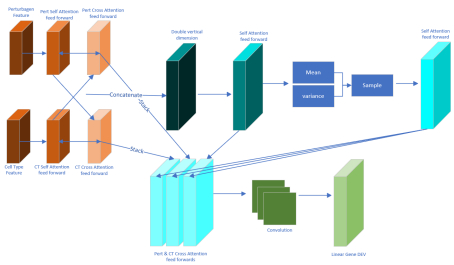

Figure 2: Attention-Enhanced Variational AutoEncoder for Gene Expression Profiling

3.4 Gaussian Process Regression and Denoise Auto Encoder

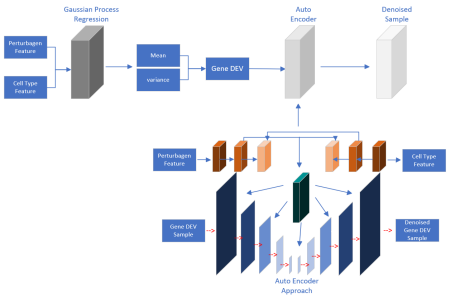

Figure 3: Workflow of Integrated Gaussian Process Regression with Denoising Auto Encoder for Gene DEV Prediction

3.4.1 Gaussian Process Regression

Considering that Gaussian process regression is a non-parametric Bayesian approach capable of automatically adapting model complexity to the data, we employed it to assess whether the parametric VAE, as previously defined, was sufficient for our training samples. Gaussian process regression provided the mean and variance of a Gaussian distribution to estimate the gene DEVs for a specific perturbagen and cell type tuple pair. A higher variance in this context indicates less confidence in the prediction.

3.4.2 Denoising Auto Encoder

We sampled from the Gaussian Process Regression output distribution as a preliminary estimate, followed by an Auto Encoder to denoise this distribution. This process involves mapping all samples from the distribution to the true gene DEV values with the aid of perturbagen features and cell type features. Therefore, the output of the Auto Encoder should represent another distribution, transformed from the Gaussian process regression output. The method can potentially offer a more accurate approximation of the true gene DEVs in terms of prediction accuracy.

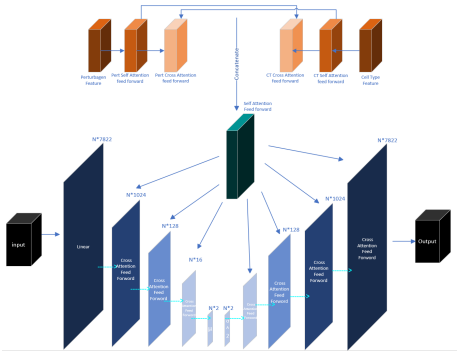

Figure 4: Model Architecture for Denoising A Noisy Gene DEV Sample Input with Perturbagen and Cell Type Features

3.5 Conditional Diffusion Model

Lastly, we modeled the distribution of the data generating process through a generative conditional diffusion model. The diffusion model contains two processes. First, the Forward process that adds noise to the input data at each time step until it follows a distribution that is close enough to the Gaussian distribution with mean 0 and variance 1. Therefore it expects the final output of this forward process be the data that are completely different from the original, or the noised one. The second process is called the Backward process, or Denoising process. It reconstructs the noised version of data from the forward process to the original one with corresponding pertubagen and cell type features as the condition to guide the process. The combination of pertubagen and cell type features is the key part in conditional diffusion model and helps the model to capture more information for better learning. The workflow within the backward process is iteratively denoising the noised gene DEVs. In our case, it starts from the noisiest state. During the denoising process, we used a encoder-decoder structure which is similar to the Denoising Auto Encoder of Gaussian process

regression. We compresses noisy gene DEVs into a latent representation, and the decoder reconstructs the cleaned version of gene DEVs from the noisy input, thus reversing the forward process.

During the training process, we have considered the time steps and sample sizes as two factors. The selection of the time steps impact both the information captured and the trade-off between model complexity and computational resources. The choices of sample sizes may impact the model stability and convergence.

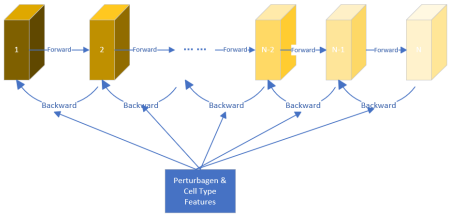

Figure 5: Workflow of Conditional Diffusion Model for Gene DEV Prediction

4 Evaluation

Since the output of all the approaches is expected to be a distribution that approximates the true gene DEVs given a perturbagen and cell type tuple pair input, we evaluated the model's performance using the following methods: (1) the Mean Row-wise Root Mean Squared Error (MRRMSE) between a point estimate derived from the distribution and the true value, and (2) the coverage of a confidence interval created with a predefined probability of containing the true gene DEV value. For simplicity, we assumed that gene DEVs are independent.

To calculate the Mean Row-wise Root Mean Squared Error (MRRMSE), for each sample/row (one cell type treated with one perturbagen) in the dataset, we computed the RMSE and then average these results across the rows/samples to obtain the final metric. To approximate the performance of the model, we passed in an estimation of gene DEV provided by the models to calculate MRRMSE. The estimation derived from the distribution came from different sources. First, we used point estimate statistics (mean or median) to be representative of the distribution for computing MRRMSE against the true gene DEV. Alternatively, we simply took a sample from the distribution. A lower MRRMSE value indicates a more accurate model.

To calculate the coverage of a confidence interval with an independent multivariate distribution, we determined the endpoints of the interval using a predefined probability for each dimension (genes), viewing them as univariate random variables. By averaging the count of whether the interval contains the true value across the sample size, we derived a coverage probability for the constructed confidence interval. Theoretically, the closer the empirical coverage probability is to the nominal predefined probability, the better the model's performance if the independence assumption is satisfied. Additionally, a higher empirical coverage than nominal coverage is preferable to a lower empirical coverage than nominal coverage, in cases where these two differ. If the actual coverage significantly differs from the nominal coverage, it could be due to the violation of the independence assumption.

5 Results

5.1 Baseline - Dimensionality reduction Variational Auto Encoder with Name to Index Prediction

MRRMSE of Deterministic Prediction	2.9254
MRRMSE of All Samples	2.5259
MRRMSE of Mean Vectors	2.3497
MRRMSE with Median Vector	1.72
Average CI Coverage	0.4055

Table 1: Feature Based VAE Model Performance Metrics

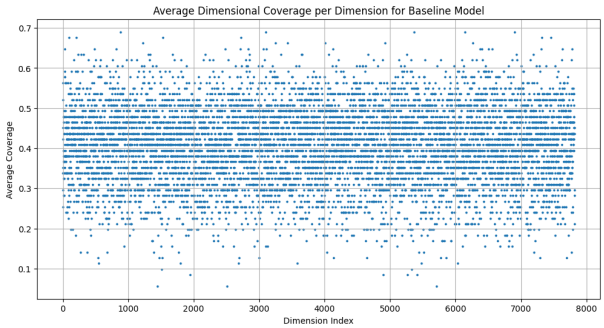

Figure 6: Coverage Probability of 95% Nominal Confidence Interval for Each Dimension of Baseline VAE Prediction

5.2 Feature-Based Variational Auto Encoder

MRRMSE of Deterministic Prediction	1.4741
MRRMSE of All Samples	1.5012
MRRMSE of Mean Vectors	1.4805
MRRMSE with Median Vector	1.4838
Average CI Coverage	0.2467

Table 2: Baseline Model Performance Metrics

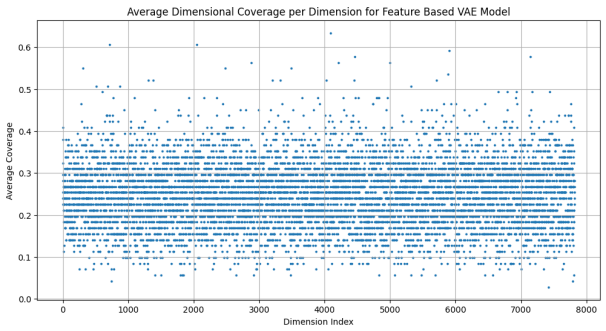

Figure 7: Coverage Probability of 95% Nominal Confidence Interval for Each Dimension of Feature Based VAE Prediction

5.3 Gaussian Process and Denoising Auto Encoder

MRRMSE of All Samples	1.1604
MRRMSE of Mean Vectors	1.0342
MRRMSE with Median Vector	1.0342
Average CI Coverage	0.0024

Table 3: GP denoising VAE Model Performance Metrics

5.4 Conditional Diffusion Model

Model Metrics	Ts=20	Ts=20, Ss=3x	Ts=100
MRRMSE of Deterministic Prediction	1.7791	1.8235	2.0768
MRRMSE of All Samples	1.2419	1.2323	1.3598
MRRMSE of Mean Vectors	1.2271	1.2177	1.3594
MRRMSE with Median Vector	1.2344	1.2230	1.3597
Average CI Coverage	0.27449	0.30019	0.00447

Table 4: Conditional Diffusion Model Performance Metrics (Ts = Time steps, Ss = Sample size)

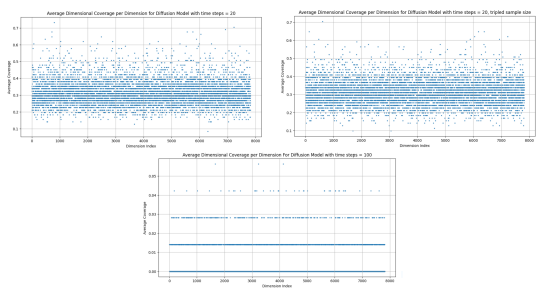

Figure 8: Coverage Probability of 95% Nominal Confidence Interval for Each Dimension of three Diffusion Models

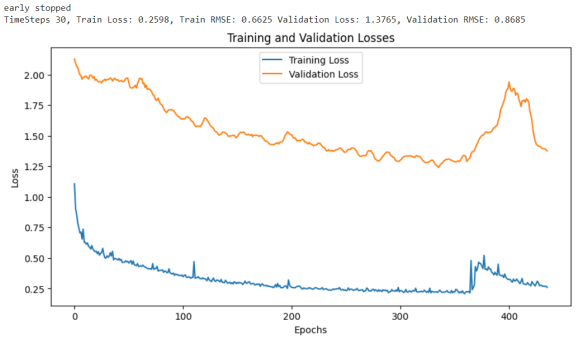

Figure 9: Loss Graph of Time Step 30 with Total Time Steps = 100

6 Discussion & Analysis

The deterministic prediction was produced by skipping the sampling process and passing a fixed value for the given perturbagen and cell type tuple pair, instead of simply taking a sample from the distribution. The MRRMSE of all samples (about 8888 samples in total) represented an unbiased estimate of MRRMSE by simply taking a sample from the modeled distribution to make a prediction. Alternatively, we took the mean or median vector of each distribution as the representation, assuming the gene DEVs are an independent multivariate distribution. Admittedly, the assumption was made for simplicity and for implementation of our VAE model. Incorporating information from interrelated genes modules would be helpful in refining the model.

6.1 MRRMSE Analysis

From the metric results in the tables, we can see that deterministic prediction is much worse than the expected error of randomly drawing from a distribution. Additionally, using the mean and median of the samples drawn from the distribution would further improve the estimation. In particular, the sample median estimation has the best performance as it is the most robust estimator against outliers in the sample.

Feature-based models outperformed baseline model. All three VAEs, feature-based VAE, GP denoising VAE and diffusion VAE, showed overall lower values of MRRMSE compared to baselines (Table1-3). The result indicates that the models that feature engineering is crucial for obtaining accurate predictions. An example strong correlation between real and predicted means was shown in Fig8 (TGRV8 expression across different samples). Indeed, RMSE values obtained from this approach were much lower than those from baseline and feature-based approach, indicating a more successful prediction of perturbagen response across different cell types. Taken together, prediction accuracy could benefit more from a set of high-quality features than from a complex model to conduct feature extractions.

6.2 Confidence Interval Analysis

As shown in Average Dimensional Coverage plots, the actual coverage for all dimensions is much less than the nominal 95% coverage. Also, averaging across all dimensions to get the Average CI Coverage metric in the tables, the mean coverage is also far from the nominal coverage of 95% of the confidence interval constructed under the assumption of independent multivariate distribution. Therefore, the results of the CI coverage would suggest that the gene DEVs are indeed dependent, so obtaining a confidence interval by treating the multivariate distribution as independent univariate distributions is over simplified. Given that the gene DEVs are highly correlated, which violates the assumption of the independent multivariate distribution, we expect that correct modeling of dependency between the gene DEVs, such as incorporating gene module information, will be critical in constructing a valid confidence interval.

6.3 Diffusion Models Parameter Analysis

To determine the effect of sample size and number of time steps on the performance of model prediction, we have tried time steps of 20, 50, and 100, and sample sizes of 3 times, 5 times, and 9 times. For simplicity, we included 3 outcomes: a model with 20 time steps, a model with 20 time steps and a tripled sample size, and a model with 100 time steps. The trend indicates that with 20 time steps, the model performs better, and with a tripled sample size, the model is better than the other two options.

Since the Diffusion model starts with samples from an independent multivariate standard normal distribution, the mean and median are the same, which means the MRRMSEs are the same for both mean and median.

In addition, during the training process of the Diffusion Model, we implemented an early stopping mechanism which could lead weights to converge to a local minimum instead of the global minimum. Thus, as the number of time steps increases, the error accumulates and propagates through the later iterations, leading to a suboptimal solution. Therefore, our model performs better when we set time steps to be 20 than 100. Figure 9 shows the increasing validation error and sign of double descent during the training process. The weight of that iteration would be a suboptimal solution such that the bias/error propagates to the next time step. Therefore, if we remove the early stopping rate, it's possible that the model with higher total time steps perform better.

7 Conclusion and Outlook

We applied several modified VAEs to human PBMCs treated with various perturbagens, from learning latent embeddings to predicting DEVs. Specifically, we illustrated the modeling of perturbations across different cell types, and have demonstrated the potential of these methods in revealing potential transcriptional regulations. While GP denoising VAEs performed better than baselines in our experiments in terms of RMSE, our CI evaluation needs more improvement. The current heuristic orignates from several assumptions including independence of genes. Modeling dependency with approaches like copula might benefit the evaluation. In this regard, establishing alternative evaluations, such as pathway enrichment analysis from predicted vs real DEVs in each cell type, can provide more insights into the biological processes we captured. Benchmarking our model with established method such as scGEN, scVAE will also be helpful. Nonetheless, we expect our project facilitate in sillico, feature-based perturbation screening or predictions across various cell types.

References

[1] F. Alexander Wolf Yuge Ji, Mohammad Lotfollahi and Fabian J. Theis. A call for deep-learning models to forecast effects of perturbations. *Cell Systems*, 12(6):495–505, 2021.

[2] Biyu Li Jenny Chen Charles P. Fulco Livnat Jerby-Arnon Nemanja D. Marjanovic Danielle Dionne Atray Dixit, Oren Parnas and Tyler Burks et al. Perturb-seq: Dissecting molecular circuits with scalable single-cell rna profiling of pooled genetic screens. *Cell*, 167(7):1853–1866.e17, 2016.

[3] Ciyue Shen Torsten Gross Joseph Min Stefan Peidli, Tessa D. Green and Samuele Garda. scperturb: Harmonized single-cell perturbation data. *bioRxiv*, 2022.

[4] Regier J. Cole M.B. et al. Lopez, R. Deep generative modeling for single-cell transcriptomics. *Nature Methods*, 15(12):1053–1058, 2018.

A Appendix

This is a section in the appendix.

A.1 Feature Engineering

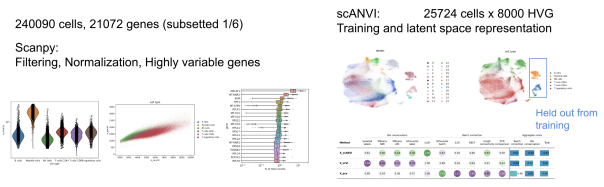

Figure 10: Feature Engineering - Cell Type

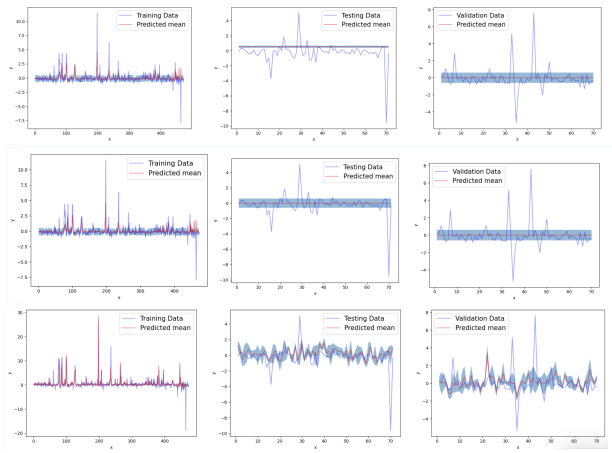

Figure 11: Gaussian Process & Denoising

	RMSE_train	RMSE_test	RMSE_valid
Baseline			
Pert & CT	1.2727	1.2755	1.0947
Pert_CT paired	1.2963	1.0725	1.2767
w/o Pert	1.2967	1.2917	1.0926

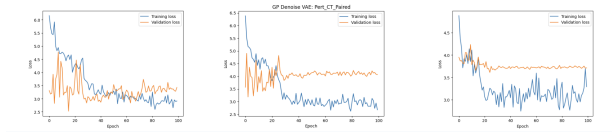

Figure 12: GP Denoising VAE

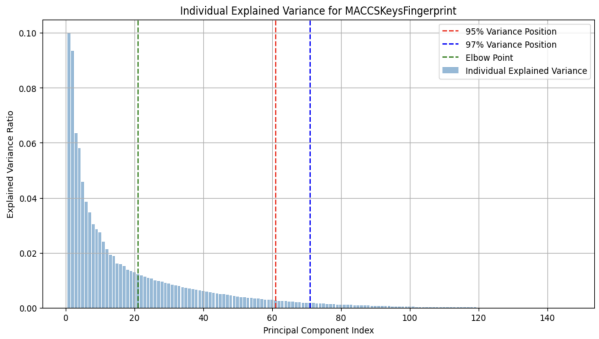

Figure 13: Example Feature Engineering of Perturbagen by MACCSKeysFingerprint

MRRMSE of train (ct + pert)	1.2184
MRRMSE of test (ct + pert)	1.2782
MRRMSE of valid (ct + pert)	1.0940
MRRMSE of train (ct-pert paired)	1.2160
MRRMSE of test (ct-pert paired)	1.2779
MRRMSE of valid (ct-pert paired)	1.0954
MRRMSE of train (w/o pert)	0.9925
MRRMSE of test (w/o pert)	1.3450
MRRMSE of valid (w/o pert)	1.1735

Table 5: Gaussian Process Regression Performance Metrics