Simplified OmiEmbed for Muti-Omics Data on Cancer Classification

Anonymous submission

Paper ID

Abstract

Our project aims to simplify the OmiEmbed framework for multi-omics data analysis, specifically targeting tumor classification task. By leveraging deep learning techniques and integrating various omics data sets, OmiEmbed has shown promising results in analyzing the correlations between multi-omics data and cancers. However, due to the model's complexity, it requires extensive training time and heavy computational power and could have inconsistent performance under various conditions. In this project, we propose a simplified version of OmiEmbed to achieve comparable accuracy while reducing complexity. We utilized the TCGA Pan-Cancer dataset, which consists of multiple omics data types, and employed regular auto encoders for self-supervised learning. Our framework focuses on the tumor classification task, with proper loss functions and training approaches. We anticipated that our simplified OmiEmbed will achieve tumor classification task with less overfitting potential, increased training speed, and reduced computational power requirement.

1. Introduction

Cancer remains a significant global health concern, ranking as the second leading cause of death in the United States (5). With the aim to address this challenge, large-scale analysis and examination of various omics data types have been contributing to improving survival rates and providing opportunities for more effective treatment. Omics science, such as genomics, proteomics, transcriptomics, or metabolomics, are the studies in the areas of biology that end with -omics. These studies evaluate the correlation between cancer-causing tumor types and various biology features, such as gene expression or RNA expression, to identify the potential causes of cancer within human genetic or molecular makeups.

The OmiEmbed framework, as presented in the paper "OmiEmbed: a unified multi-task deep learning framework for multi-omics data" (7), aims to address the challenges in multi-omics data analysis and integration for cancer research. The framework takes multiple omics data types with rows corresponding to samples/patients and columns representing features/identifiers, including gene expression, DNA methylation, and miRNA expression, as inputs. The framework employs a self-supervised learning model that consists of a VAE deep embedding module and one or more downstream task modules. The model outputs the tumor classification or survival prediction results of each sample, depending on the task selected for training and testing. The problem we identified was the extensive training time and heavy computational power the model requires. Therefore, our project aims to explore whether the simplified OmiEmbed framework can achieve comparable accuracy for tumor classification tasks after optimization.

Our project focuses on simplifying the OmiEmbed framework for multi-omics data analysis. Same with the referenced model, the simplified OmiEmbed model takes in multiple omics datasets in the identifiers versus samples format, including gene expression, DNA methylation, and miRNA expression. These data types are represented as two-dimensional matrices, with rows corresponding to samples and columns representing features or variables, or vice versa. Our model focuses on the tumor classification task, and thus the model outputs tumor classification results of each sample in the input data. Our experiments explored models with different embedding designs, various loss functions, and fewer numbers of downstream layers. Our goal is to design a model that achieves comparable or higher testing classification accuracy and improves training and testing speed.

2. Related work

Multi-omics analysis has emerged as a prominent approach in cancer classification, leveraging its capacity to integrate diverse molecular layers and provide a comprehensive understanding of the disease. For example, Yue et al. (2021) devised a multi-omics framework that incorporated genomics, transcriptomics, and proteomics data to categorize lung adenocarcinoma subtypes (6), thereby revealing distinct molecular subgroups and identifying potential targets for personalized treatment strategies. Motivated

by advancements in deep neural networks (DNN), convolutional neural networks (CNN), and semi-supervised learning (SSL), deep learning techniques have exhibited promise in utilizing omics data for improved cancer classification. Notably, Feng et al(2019) developed DeepCC (2), a deep learning-based model for cancer molecular subtype classification, to effectively classify various subtypes of colorectal cancer, surpassing the performance of conventional methods and underscoring the potential of deep learning in multi-omics cancer classification. Similarly, Rhee and others (4) proposed a hybrid model consisting of graph convolutional neural networks (GCN) and relational networks (RN) for the classification of breast tumor subtypes using gene expression data and protein-protein interaction (PPI) networks. The challenge of effectively reducing high-dimensional multi-omics data is often addressed through principal component analysis (PCA), a commonly employed dimensionality reduction technique in multi-omics analysis (3). Furthermore, Xiaoyu et al. (7) introduced a deep learning framework called OmiTrans, which incorporates generative adversarial networks (GANs) to achieve omics-to-omics translation. This model employs unsupervised deep embedding learning for effective dimensionality reduction as a preprocessing step.

3. Method

3.1. Dataset

For our dataset, we used the TCGA pan-cancer (PANCAN) dataset. It is a comprehensive collection of multi-omics data from The Cancer Genome Atlas (TCGA) project. The source of data is the GDC pan-cancer (PANCAN) datasets provided by UCSC Xena (1), which contains two publicly available datasets, TCGA PANCAN and TARGET PANCAN. We preprocessed the data to preserve the data only from TCGA PANCAN. The GDC PANCAN dataset contains various types of omics data, such as gene expression, DNA methylation, and miRNA expression. Four datasets from GDC PANCAN were used in our experiments:

1. Gene expression dataset, in which each value represents the expression value of a transcript/version of a gene from a sample;
2. DNA methylation dataset, in which each value represents the DNA methylation value at a CpG site corresponding to the identifier from a sample;
3. MiRNA expression dataset, in which each value represents the miRNA expression value of a microRNA molecule from a sample;
4. Basic phenotype dataset, which contains the basic phenotypes of each sample (age, gender, tumor type, etc.).

The first three datasets, the omics datasets, serve as input datasets to the model and will be referred to as A, B and C respectively further on in the report. The basic phenotype dataset serves as the labels for the model. All values in the input data are numbers, and the labels were coded as numbers by preprocessing approaches described below.

Dataset info	GDC PANCAN		
Tumor types	33		
Omics types	Gene expression	DNA methylation	miRNA expression
# of sample	11057(TCGA) + 710(TARGET)	9736	11020
# of features	60,483	48,577	1,881

Table 1. Overview information of the GDC PANCAN dataset

The dimensionalities of the three types of omics data before preprocessing were 60,483, 48,577, and 1,881 respectively. Table 1 provides the overview information of the datasets. The GDC PANCAN dataset consists of 33 tumor types plus a normal control samples.

3.2. Data Preprocessing & Data Loading

3.2.1 Down sampling

Due to the design of how the model processes the input data, the input datasets must contain the same samples. Thus we applied the intersection operation on the three input datasets and generated a sample list that contains all samples that exist in all three datasets. The resulting sample list contains 8885 samples. The data loader loads samples only from the sample list when loading input datasets.

3.2.2 Data cleansing

We applied different cleansing approaches to each dataset depending on the data it contains:

- Dataset A originally contained non-TCGA samples (TARGET samples). We excluded the irrelevant samples by removing columns with label `"TARGET-*"`.
- Dataset B originally contained identifiers with missing values. We excluded identifiers with NA values.
- For dataset C, no additional data cleansing was needed.
- For the labeled dataset, we excluded all irrelevant phenotypes by removing all columns except the one labeled `"sample_type"`. We then grouped the samples by their tumor types, and manually coded each tumor type with an integer value.

3.2.3 Handling unbalanced data

The resulting samples list contains two unbalanced tumor types that contain less than 10 samples, while the other

Dataset info	TCGA PANCAN		
Tumor types	31		
Omics types	Gene expression	DNA methylation	miRNA expression
# of sample	8885	8885	8885
# of features	60,483	140,063	1,881

Table 2. Overview information of datasets after preprocessing

31 tumor types have at least 70 samples. Thus we excluded the the samples classified as these two tumor types. The finalized preprocessed datasets contain in total 31 tumor types. Table 2 shows the overview information of the datasets after preprocessing.

3.2.4 Data loading

All downloaded datasets were stored in `.tsv` format. In the input datasets, rows represent the identifiers for the corresponding biomarkers (transcripts/versions of genes, CpG sites or microRNA molecules), and columns represent the samples. In the labeled dataset, rows represent the phenotypes, and columns represent the samples.

Two DataLoader classes were implemented, one for multi-omics analysis and the other for single-omics analysis. In the case of multi-omics analysis, the DataLoader constructor takes in all three omics datasets and the phenotype dataset. After preprocessing, the constructor stores each dataset as a Tensor. The Tensors maintain the format with rows representing identifiers/phenotypes and columns representing samples and in the label Tensor. If the option `use_sample_list` is selected, the preprocess filters the input omics data and keeps only the samples from the sample list.

Each time the DataLoader iterates, it returns a dictionary that contains (1) input_omics, a list of three Tensor objects, each containing the column corresponding to the sample of the index, (2) the label of the sample, and (3) the index.

In the case of single-omics analysis, the Dataset constructor does the same tasks except loading and processing only one omics data, and the returned dictionary contains input_omics of length 1.

3.3. Overall Architecture

We took the original OmiEmbed model as a reference and implemented the modified model. The model consists of two major components, the embedding network, which can be further separated into the encoder and the decoder, and the downstream network. Figure 1 visualizes the model architecture when taking in three input datasets.

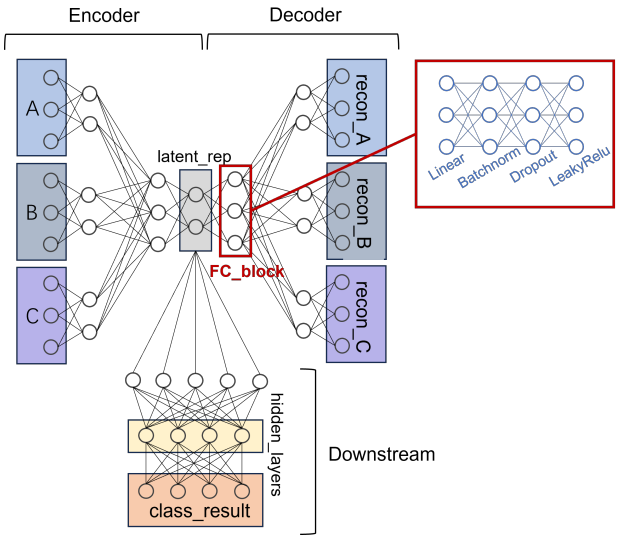

Figure 1. Modified OmiEmbed Model

3.4. Embedding Network

The embedding network consists of an encoder structure and a decoder structure. The embedding structure differs for models that take in different numbers of input datasets. In the case of three input datasets, the encoder takes in all three datasets, A, B, and C, from the data loader, and then processes them individually in the first two layers. Note that each "layer" visualized in Figure 1 and referred to in this report is a block of layers, which contains a Linear layer, followed by a normalization layer, followed by a dropout layer, and finally followed by an activation layer. At the third layer, the three output tensors from the second layer are concatenated together for forward passing. The fourth layer, which is the last layer of the encoder, produces a single latent representation. The decoder processes the task to reconstruct the input datasets from the latent representation. Similar to the encoder, the second layer of the decoder splits the single output from the last layer and produces three output tensors, each representing the data from an input dataset.

The major difference between the modified model and the original OmiEmbed model is the choice of encoders. The OmiEmbed model utilized variational autoencoders, while our modified model uses regular autoencoders.

3.4.1 Embedding loss:

The embedding loss the embedding network optimizes on is defined to be the weighted average of the three reconstruction losses for each input dataset. The loss

function we selected for the reconstruction losses is the Mean Squared Error (MSE) of the reconstructed dataset compared to the original input dataset.

$$\mathcal{L}_{embed} = \text{mean}\left(\omega_a \mathcal{L}_{recon\_A} + \omega_b \mathcal{L}_{recon\_B} + \omega_c \mathcal{L}_{recon\_C}\right) \quad (1)$$

$$\mathcal{L}_{recon\_N} = \text{MSE}(\text{recon\_N}) \quad (2)$$

The loss function was designed to assign higher weightage to the reconstruction of gene expression features (from A), DNA methylation values (from B), and miRNA expression features (from C). By adjusting the weight coefficients in the loss function, we aimed to emphasize the reconstruction accuracy of the targeted omics types during the training process.

3.5. Downstream Network: Tumor Classification

The downstream network takes in the latent representation produced by the encoder in the embedding network as the input. The network then passes the input through multiple FC_block layers, and eventually produces a classification result for each sample from the model's input. We utilized the Cross-Entropy loss function as the downstream loss to measure the discrepancy between the predicted class probabilities and the true labels:

$$\mathcal{L}_{downstream} = \text{CE}(\text{pred\_probs}) \quad (3)$$

Thus, the overall loss function of our modified OmiEmbed model is defined as the sum of the embedding loss and the classification loss, weighted by user-defined weight coefficients. Note that the weight on the downstream loss should be significantly greater than the embedding loss since the focus of the model is the classification task in the downstream network. We sought to learn the effect of the proportion between the two weights and find the optimal combination that produces the best result:

$$\mathcal{L}_{total} = \alpha \mathcal{L}_{class} + \beta \mathcal{L}_{embed} \quad (4)$$

The weight coefficients serve as hyperparameters and play a crucial role in balancing the contribution of the embedding loss and the classification loss, ensuring that both stages are appropriately accounted for during training and optimization.

3.6. Baseline Architecture

The baseline model of our experiments is the original OmiEmbed model. It shares a similar overall architecture with our modified model, which consists of a embedding network and a downstream network. The major difference of the original model from our modified model, except the multi-tasking downstream design, is that the original model uses a variational autoencoder (VAE) in the embedding network, which calculates the mean and the logarithm of the variantion of the latent representation, and passes the reparamterization results to the decoder and passes the mean to the downstream network, when forwarding.

3.7. Training Approaches

In the experiments, each input dataset was split into a training set, a validation set and a testing set. Three Dataloaders were created, each for the training set(s), the validation set(s), and the testing set(s) respectively.

Instead of updating the entire model in every epoch, we experimented with different training approaches and concluded with the following approach, which produces the optimal classification results in general. When training, the number of epochs is split into three phases, each with a different optimizer that optimizes on a loss and updates a part of the model:

Phase 1: in each epoch, the optimizer back-propagates with the embedding loss and updates the parameters of the layers in the embedding network only. The classification accuracy during Phase 1 stays constant and low.

Phase 2: the optimizer back-propagates with the downstream loss and updates the parameters of the layers in the downstream network only. Phase 2 should bring a boost to classification accuracy.

Phase 3: the optimizer back-propagates with the total loss and updates the parameters in the entire model. Phase 3 should bring another boost to classification accuracy.

We experimented with different numbers of epochs allocated to each phase and concluded with the optimal allocation, elaborated in the Results.

The primary metric on which we evaluated the model performance is the classification accuracy, plus the macro-average precision, recall and AUC scores on the classes/tumor types as sanity checks.

4. Experiments and Results

4.1. Implementation:

You can find the code for our project on GitHub https://github.com/Wuj1Zhang/SimpleOmi.

4.2. The effects of learning-rate scheduler

The learning-rate scheduler, acting as a wrapper class for the optimizer, plays a crucial role in determining the decay function of the learning rate based on the number of epochs passed. We explored the effectiveness of different learning-rate schedulers on our modified model.

To assess the impact of various learning-rate schedulers, namely Cosine, Step, and Linear, we conducted a comparative analysis. The outcomes of this analysis are depicted in [Figure 2].

Upon analyzing the results, we observed that the cosine learning rate scheduler exhibited the most favorable performance, effectively adapting to the dynamics of our model. Conversely, the step and linear schedulers demonstrated suboptimal performance when compared to the cosine scheduler.

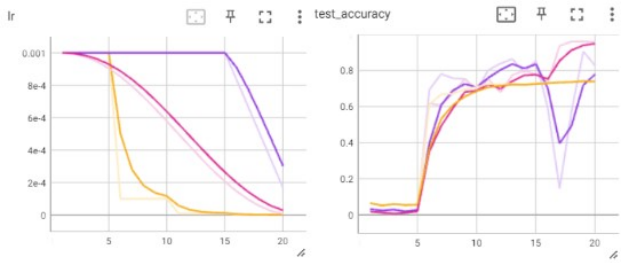

Figure 2. The relationship between test accuracy and the number of epochs for different learning-rate schedulers. The purple line corresponds to the Linear learning-rate scheduler, where the learning rate decays to zero over 30 epochs. The Rose line represents the cosine learning-rate scheduler, while the Yellow line represents the step learning-rate scheduler with a step size of 5 epochs.

4.3. Loss Coefficients

4.3.1 Embedding and downstream loss coefficients

To examine the influence of weighting the embedding and downstream losses in our model, denoted as α and β respectively in Equation 4, we conducted comprehensive experiments to evaluate their impact on performance. In the baseline model, we initially assigned default weights of $\alpha = 0.01$ and $\beta = 1.0$ to prioritize the downstream classification task.

To streamline the experimentation process and obtain prompt initial results, we employed the Bayesian search method for hyperparameter tuning. Remarkably, as illustrated in Figure 3, we observed that setting the weights around $\alpha = 1.0$ and $\beta = 1.0$ resulted in a testing accuracy of approximately 96.5% within fewer than 40 epochs. This indicates that the optimizer attributed equal significance to both the classification and embedding losses during backpropagation, thereby updating the model parameters accordingly. However, despite this encouraging result, we observed some instability in the optimization process during Phase 3.

These findings highlight the importance of carefully balancing the weights assigned to the embedding and downstream losses. The equal weighting approach led to remarkable accuracy gains, emphasizing the significance of considering both aspects during model optimization. Nevertheless, further exploration and fine-tuning are warranted to improve the stability and convergence of the optimization process in Phase 3.

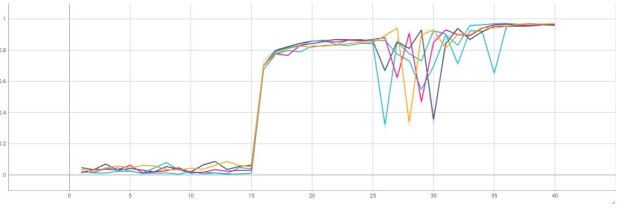

Figure 3. The relationship between test accuracy and the number of epochs for different weights coefficients on embed loss. Five lines in different colors with .

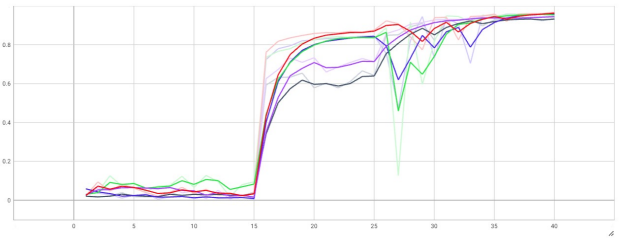

Figure 4. The relationship between test accuracy and the number of epochs for different combinations of reconstruction loss coefficients. Five lines in different colors represent different combinations of reconstruction loss coefficients. All other hyperparameters are the same(P1=15, P2=10, P3=15). The red one is the best one with($\omega_a = 0.85, \omega_b = 1.05, \omega_c = 1.05$)

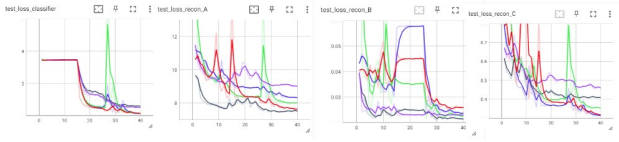

Figure 5. Corresponding reconstruction loss and classification loss v.s epoch with Figure 4

4.3.2 Separately reconstruction loss coefficients

After incorporating the weighting of the embedding loss in our experiments, we proceeded to investigate the potential performance improvements by individually assigning weights to each reconstruction loss (referred to as A, B, and C) as shown in Equation 1. Through hyperparameter tuning of these three loss coefficients, we initially employed a multiple random search approach. In each iteration, we randomly generated ten sets of coefficients for the reconstruction losses A, B, and C, ranging from 0 to 1.2, and assessed the resulting accuracies.

Following each round of ten sets, we examined the accuracy outcomes and progressively narrowed down the coefficient ranges for subsequent iterations. This iterative process allowed us to select five representative combinations of reconstruction loss coefficients that yielded the most promising performance within the minimal coefficient range (as illustrated in Figure 4, and 5).

Although the final differences in accuracy among the selected combinations were minimal, we observed a notable trend. Specifically, the coefficient assigned to the reconstruction loss of A was slightly lower than those assigned to the reconstruction losses of B and C. This particular combination facilitated a more significant improvement in accuracy during Phase 2 and ensured a more stable convergence during Phase 3.

In summary, while the overall variation in accuracy was marginal, the combination of lower coefficients for the reconstruction loss of A and higher coefficients for the reconstruction losses of B and C proved to be optimal. This combination enhanced the accuracy curve during Phase 2 and achieved a more stable convergence during Phase 3.

4.4. Effects of Phase arrangement on the performance of model

The original OmiEmbed model utilized a three-phase training approach. In our Simplified OmiEmbed model, we conducted experiments to explore the impact of phase allocation on the model's performance. Employing a controlled variable approach, we used the normal phase length combination (p1 = 15; p2 = 10; p3 = 15) as the control group and performed three distinct comparative experiments. These experiments involved setting the lengths of Phase 1, Phase 2, and Phase 3 to 0, effectively simulating the absence of each respective phase. Subsequently, we repeated each experiment 10 times and averaged the testing accuracy for visualization purposes. Notably, we observed varying degrees of influence on the model's performance due to the different phases showing in Figure 6.

The presence of Phase 1 significantly impacted the model's ability to learn from input features, as the embed model employed an AutoEncoder to reduce the dimensionality of high-dimensional omics data. Therefore, the pretext problem-solving performed during the first phase directly influenced the stability and convergence speed of subsequent phases. However, it was noteworthy that the model still achieved relatively high accuracy in the classification task, indicating that the primary impact was on the convergence speed of accuracy.

Remarkably, even when Phase 2 was omitted, the model was still able to achieve high accuracy in the classification task. This can be attributed to the fact that in Phase 3, we optimized and updated the downstream network. However, the fluctuating accuracy curve highlighted the importance of Phase 2 for stable model optimization.

The absence of Phase 3 demonstrated that the model had already started converging during Phase 2. The lack of Phase 3 resulted in the absence of a secondary accuracy improvement. Since Phase 3 involved optimizing both the embed network and the downstream task simultaneously, its presence was helpful in achieving a secondary improvement in model performance (accuracy).

Overall, these findings underscore the importance of each phase in the model's optimization and convergence characteristics. Phase 1 primarily influenced the convergence speed, Phase 2 played a crucial role in stable optimization, and Phase 3 contributed to a secondary accuracy improvement by jointly optimizing the embed network and downstream tasks.

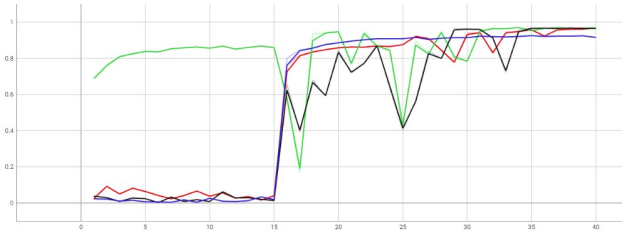

Figure 6. Test-Accuracy v.s epoch. The red line is the best phase length combination(p1 = 15; p2 = 10, p3 = 15); The green line lacks Phase 1; The black line lacks Phase 2; The blue line lacks Phase 3. All other hyper-parameters are the best after hyper-parameters tuning.

4.5. Comparison accuracy and other metrics

Our ultimate objective in the experiments was to compare the performance of the Simplified Omiembed model with the traditional Omiembed model. To achieve this, we conducted controlled pairwise comparisons between different models and epochs while focusing on accuracy, as depicted in Figure 7 (Figure 7). The x-axis of the figure represents the types of Omics datasets processed by the models, including separate Omics datasets (A, B, C) and multi-omics datasets (ABC). Through the analysis, we observed that the Simplified Omiembed model achieves comparable accuracy to the traditional Omiembed model across all types of Omics dataset classification tasks. Notably, the Simplified Omiembed model demonstrates superior accuracy within a limited training time (Epoch limit). This indicates that the Simplified Omiembed model effectively reduces the computational time during the training process.

Furthermore, in order to comprehensively evaluate and compare the performance, we employed additional metrics, as shown in Figure 8 (Figure 8). The results demonstrate that the Simplified Omiembed model achieves similar performance to the traditional Omiembed model across these metrics.

Overall, the experimental results support the notion that the Simplified Omiembed model delivers comparable or even better performance compared to the traditional Omiembed model in terms of accuracy. Additionally, it significantly reduces the computational time required for training. These findings highlight the effectiveness and ef-

ficiency of the Simplified Omiembed model in omics data analysis.

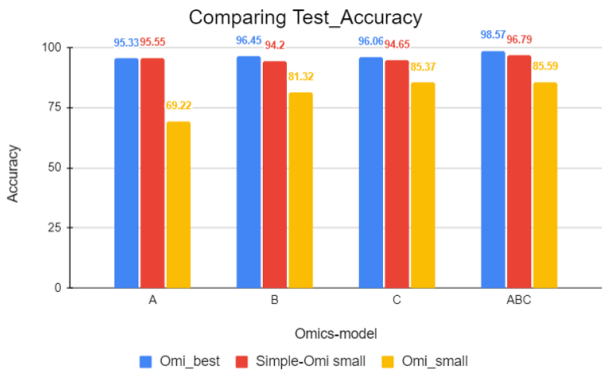

Figure 7. Testing accuracy v.s. model subtypes for specific models and epochs limit. Omi_best represents the original model over 200 epochs (50 + 50 + 100); Simple-Omi_small represents modified model after 40 epochs (15 + 10 + 15); Omi_small represents original model after 40 epochs (15 + 10 + 15)

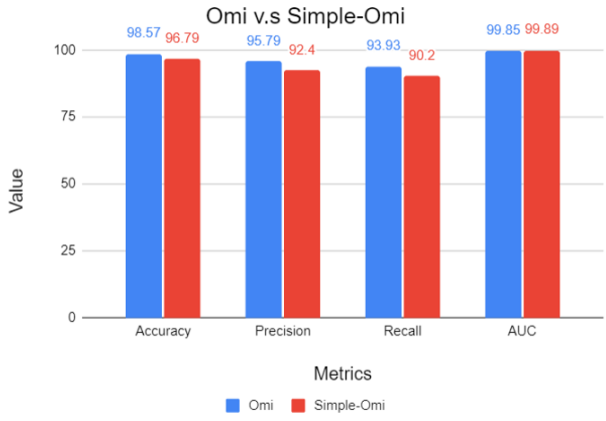

Figure 8. Metrics results of the original and modified models

5. Conlusions and disscussion

In conclusion, the Simplified Omiembed model demonstrates a better alignment with the specific requirements of our selected Omics datasets compared to the traditional Omiembed model. While maintaining a similar level of accuracy to the original model, the Simplified Omiembed model significantly reduces the training time by requiring fewer epochs, thereby effectively decreasing the computational time during model optimization and updates. The incorporation of weighted loss functions effectively addresses the challenge of significant reconstruction loss encountered in the Phase 3 stage, particularly in the gene expression and DNA methylation networks.

However, it is important to note that the Simplified Omiembed model also sacrifices a portion of its stability during training. Furthermore, compared to the traditional Omiembed model, it attenuates the contributions of different phases to the model's training accuracy, redirecting them towards enhancing model stability. As a result, the Simplified Omiembed model can serve as a foundational choice for classroom settings or student projects, especially in Omics data classification tasks. Moreover, it offers significant scope for adjustment and enhancement in terms of model complexity and stability for future researchers.

The approach of utilizing self-supervised learning for dimension reduction showcased by the Simplified Omiembed model can be widely applicable to other high-dimensional models. In future work, we aim to explore the use of Convolutional Neural Networks (ConvNets) as basic neural network blocks for model improvements and extend the application of the model to the analysis of other Omics datasets, such as ATAC-Seq.

Overall, the Simplified Omiembed model presents a promising framework that balances accuracy and efficiency for Omics data analysis, while also offering opportunities for further advancements and adaptations.

References

[1] Xenabrowser. https://xenabrowser.net. Accessed: [5, 2023]. 2

[2] Feng Gao et al. Deepcc: a novel deep learning-based framework for cancer molecular subtype classification. *Oncogenesis*, 8(9):44, 2019. 2

[3] Chen Meng et al. Dimension reduction techniques for the integrative analysis of multi-omics data. *Briefings in bioinformatics*, 17(4):628–41, 2016. 2

[4] Sungmin Rhee, Seokjun Seo, and Sun Kim. Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification. *arXiv preprint arXiv:1711.05859*, 2017. 2

[5] Hyuna Sung, Jacques Ferlay, Rebecca L Siegel, Mathieu Laversanne, Isabelle Soerjomataram, Ahmedin Jemal, and Freddie Bray. Global cancer statistics 2020: Globocan estimates of incidence and mortality worldwide for 36 cancers in 185 countries. *CA: A Cancer Journal for Clinicians*, 71(3):209–249, 2021. 1

[6] Dongsheng Yue, Weiran Liu, Liuwei Gao, Lianmin Zhang, Tao Wang, Shanshan Xiao, Yingxue Fu, Nan Li, Rui Lin, Yao Hu, et al. Integrated multiomics analyses revealing different molecular profiles between early-and late-stage lung adenocarcinoma. *Frontiers in Oncology*, 11:746943, 2021. 1

[7] Xiaoyu Zhang, Yuting Xing, Kai Sun, and Yike Guo. Omiembed: A unified multi-task deep learning framework for multi-omics data. *Cancers*, 13(12), 2021. 1, 2