UnitedNet Analysis

Theo

8/2024

Contents

1 Introduction

This is an attempt to do a thorough analysis of the multi-task framework `UnitedNet` [6]. The original source code can be found at https://github.com/LiuLab-Bioelectronics-Harvard/UnitedNet. Forked repositories can be found by the Biodata Analysis Group at https://github.com/BiodataAnalysisGroup/UnitedNet, as well a personal one at https://github.com/thodkatz/UnitedNet.

Ultimately the goal is to dissect the components of the framework, so we can utilize them to a new and improved multi-task framework applied for perturbation modelling.

The paper is examined by integrating the code implementation details along with sketches for a more intuitive understanding. Some fundamental concepts of machine learning are introduced as well to have a coherent view.

2 Architecture

To illustrate the architecture of the model, the analysis will be performed for one of the datasets used in the paper, the ATAC-seq one. But it should be easily understood and generalizable for the rest as well. For the analysis of the data, see 9.1 ATACseq.

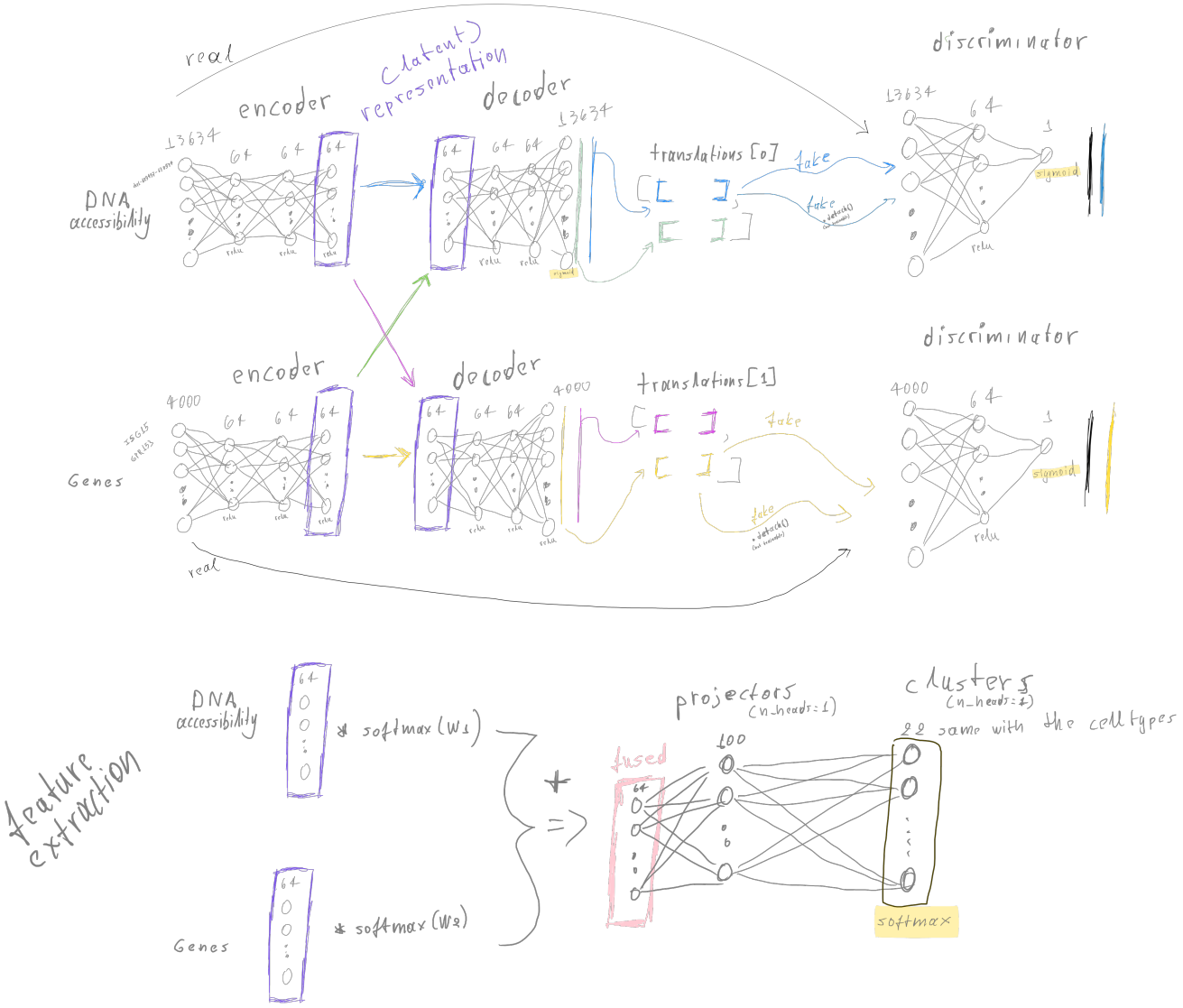

Figure 1: Multi-task architecture refined for the ATAC-seq use case (2 modalities)

For the ATAC-seq, `UnitedNet` has two main tasks: 1) classification of cell types, and 2) cross-modal prediction.

Broadly speaking, to achieve the above, the architecture is based on autoencoders per modality for feature extraction, that can be subsequently utilized for the classification task (the output of the `clusters` network as probabilities for the 22 cell types). The realization of the multi-modality for this classification, is happening with a fusing technique in the latent space.

Simultaneously, the decoding part of the autoencoder assists the generation of not only the within but the other modality as well (cross modal prediction task). The discriminator part is an enhancer for the reconstruction task of the autoencoder, inspired by the Generative Adversarial Networks (GANs).

Let's try to delve deeper into the architecture, and analyze what the above actually mean.

3 Multi-Layer-Perceptron (MLP)

As we can see, all the components referred to as encoders, decoders, discriminators, and projectors are neural networks of the type Multi-Layer Perceptron (MLP), which is considered the most fundamental feedforward neural network. Below, we present a simple definition of an MLP along with its mathematical description.

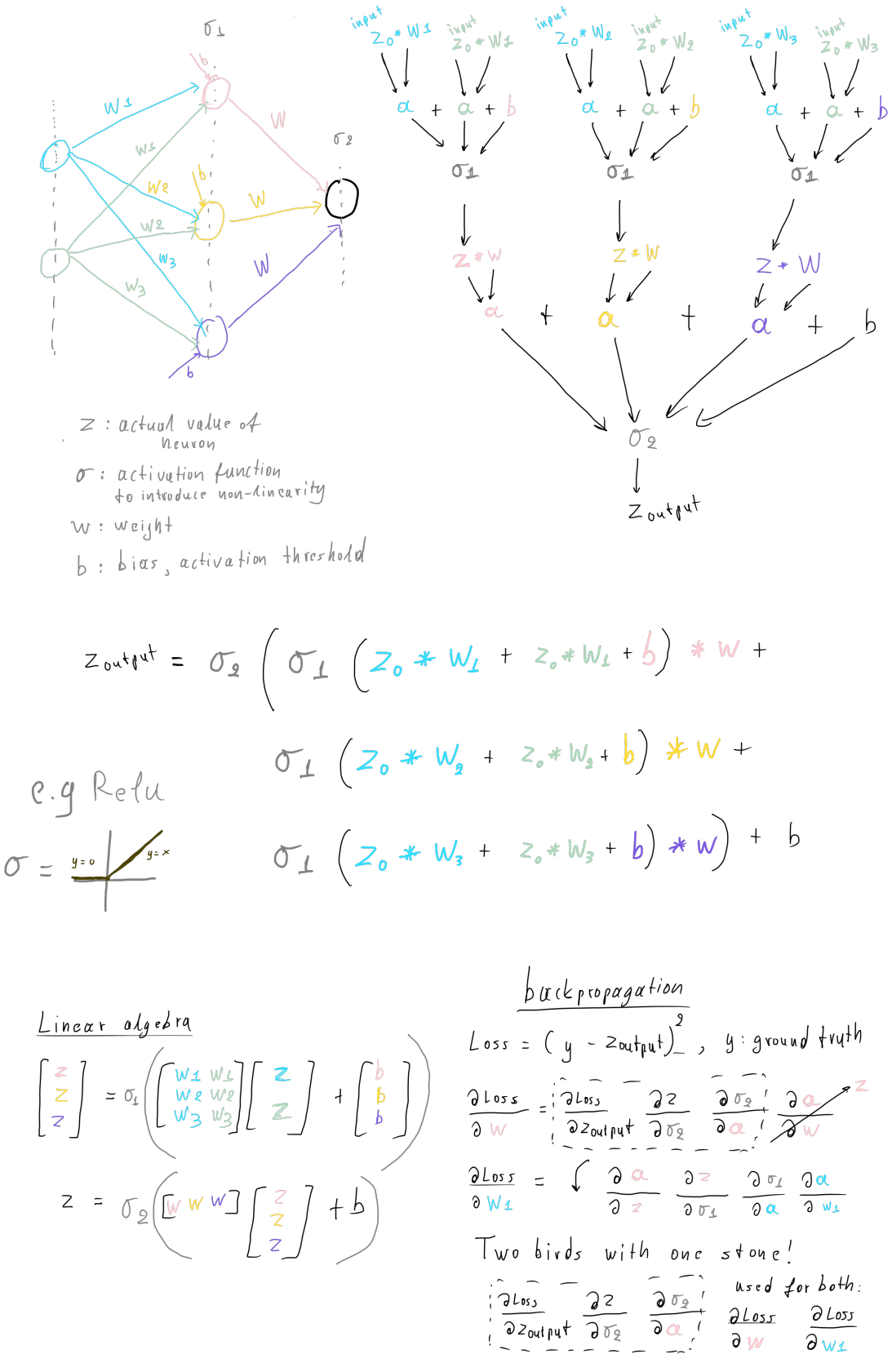

Figure 2: MLP with one hidden layer of 3 neurons, input layer with 2 neurons, and an output of 1 neuron

Practically speaking, a neural network is yet another mathematical model that can approximate ANY function, as stated by the universal approximation theorem. The same applies to other mathematical models, such as polynomials (via Taylor series) and Fourier series, which can also approximate any function. The major advantage of neural networks, and the reason they are flourishing today, is the mitigation of the **curse of dimensionality**. Specifically, for a given number of samples, the number of parameters required to approximate a function scales polynomial in neural networks, as opposed to the exponential in the polynomial model.

Backpropagation is another key reason for the success of neural networks, as it is a highly efficient method for calculating gradients, following the simple idea of the chain rule ($\frac{dy}{du} = \frac{dy}{dx}\frac{dx}{dx}$). Gradients is the cornerstone of training a neural network, because they can inform us on the direction of the minimization of a function. So, in our case, we just define a loss function, and then the goal is to solve the $\nabla f_{loss}(W) = 0$. But, it is computationally infeasible to calculate this in the high dimensional non-linear space, so we try to use **gradient descent**. Gradient descent is an iterative algorithm, that tries to find the next weight that minimizes the loss function, $w_{n+1} = w_n - \gamma \nabla f_{loss}(w_n)$, and γ is the learning rate.

The interesting thing is that all we are trying to do is just simply fitting a function. For example, the classification problem of recognizing human digits given a fixed image, is to find a function that maps a grid of a certain amount of pixels, to a vector of length equal to the number of digits, where each one of the values illustrates the probability of the digit. The challenge is to find the function, or actually in our case to learn it.

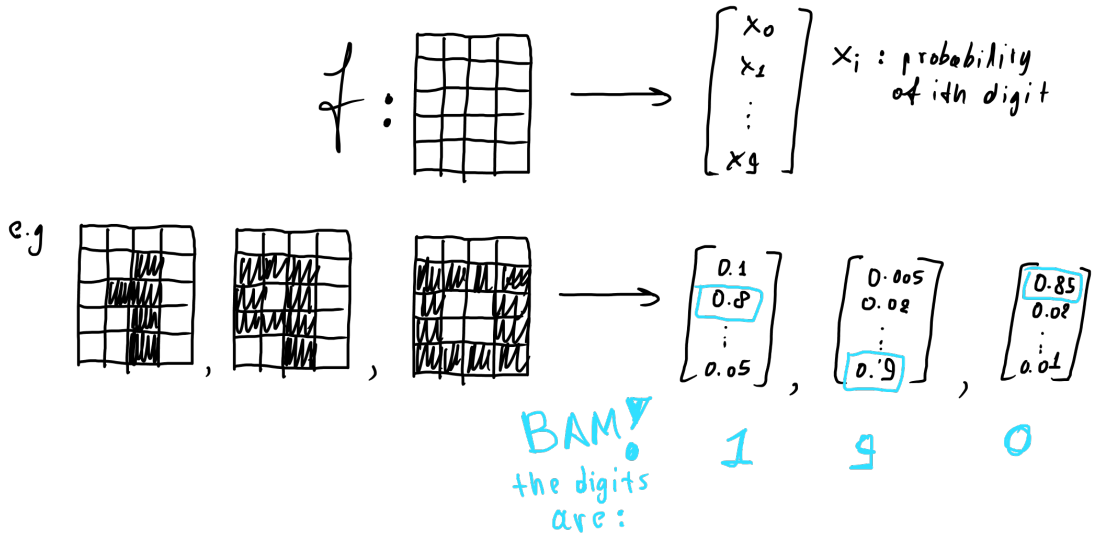

Figure 3: Fitting a function

4 Autoencoder

`UnitedNet`, for the ATAC-seq scenario, aims to solve two main tasks: 1) the classification of cell types and 2) the cross-modal prediction. A well-established method, especially for unlabeled data, is the use of unsupervised learning algorithms for representation learning. Representation learning is particularly useful for classification tasks, or any type of task, due to its feature extraction properties. To illustrate this, consider a simple autoencoder, which is one of the most common unsupervised architectures for representation learning.

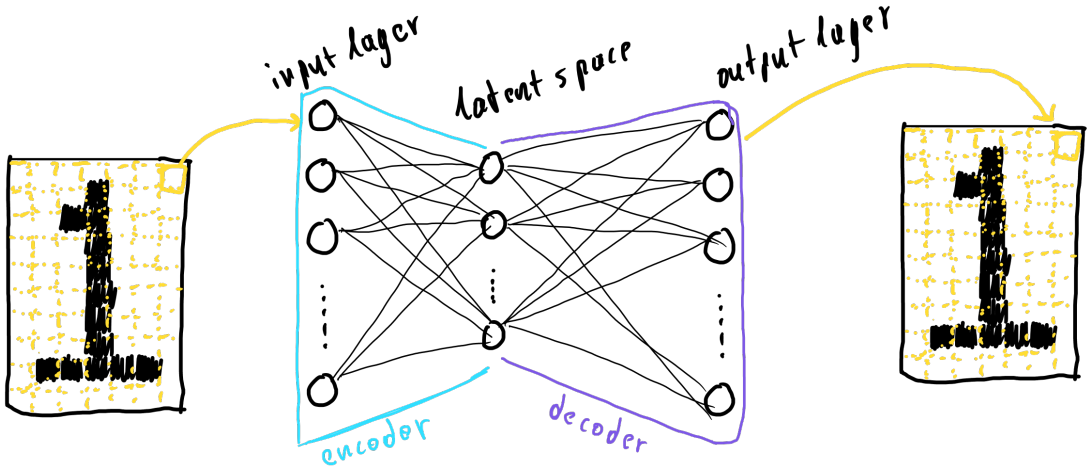

Figure 4: Autoencoder

An autoencoder simply learns to reconstruct the input. Although reconstructing an input may seem a useless task, we are interested not necessarily in the reconstruction, but what the model was able to learn to perform this reconstruction. Based on the hypothesis that the reconstruction task, will lead to the model to understand the data, we can extract what the model has learned.

The architecture of an autoencoder is centered around two key processes: compression (encoding) and decompression (decoding) of data dimensions. When a trained encoder successfully reconstructs an image, it indicates that the encoder has effectively distilled meaningful features into a compressed representation. This compressed representation, which is the output of the encoder and a reduced-dimensionality version of the input, is known as the latent space.

For example, considering the Figure 4. Having as input only an image of displaying the number one, without any labels that this number displays the number one, a trained model in the latent layer, could eventually learn to distinguish shapes, edges, texture, that can direct the decoder to reconstruct the output. In a subsequent task of classifying the digits with a small set of labeling data, the latent layer could be used as input.

Another way to think of autoencoders, and the latent space, is an attempt of moving from pixel-wise information to high level features sufficient to understand a sample. To illustrate this, consider the experiment of trying to draw a dollar bill from memory (Figure 5). As we can see, although we don't retain all the information of a dollar bill, we can still recognize it in our daily lives due to our vision-feature space.

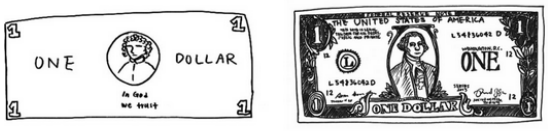

Figure 5: Fig. Left: Drawing of a dollar bill from memory. Right: Drawing subsequently made with a dollar bill present. Image source: Epstein 16

There are a lot of variations of these type of autoencoders, that aim to improve the feature extraction process and make the neurons of the latent space more discriminative, as described on this variation of spectral clustering using dual autoencoders [8]. It is worth mentioning that feature extraction is usually evaluated with the quality of clusters formed in the latent space. For example, in the ATAC-seq relevance analysis of the `UnitedNet`, the latent space is visualized in a 2D plot, with the dimensionality reduction algorithm of UMAP (Figure 6). As we can see there is great quality of clustering, the points are not mingled with each other.

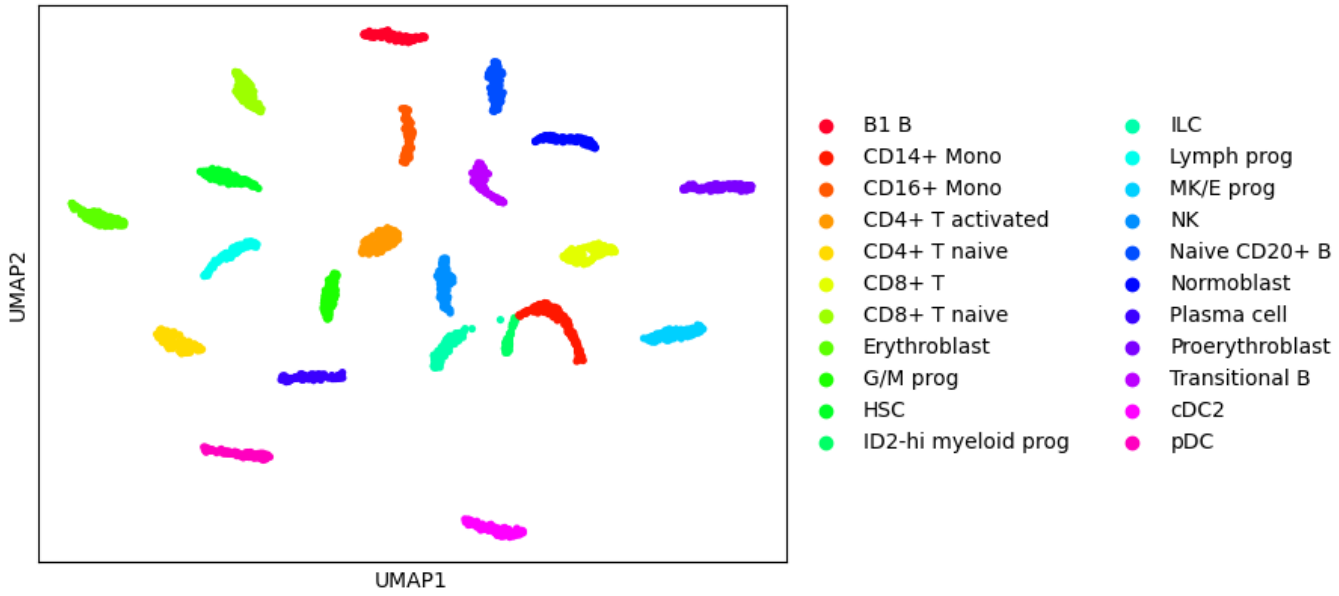

Figure 6: Latent space visualized with UMAP for the ATAC-seq scenario (test batch: 's1d1')

Other well known type of autoencodes that could be explored are the Variational Autoencoders [4], or the Masked [2]. See 13 Ideas to improve UnitedNet.

So with this core idea of representation learning, we can extract the latent space to perform the classification task. But at the same time, the decoder part of the autoencoder acts as a generative machine. So, in our case, it is used for the cross modal prediction task as well. That means that the decoder of the gene expression modality learns not only to reconstruct the gene expression modality from the latent space produced by the encoder of the gene expression modality, but attempts to generate/construct the DNA accessibility modality by the latent space produced by the encoder of the DNA accessibility modality. The exact same applies for the DNA accessibility autoencoder, that its decoder learns to generate the gene expression, along with the DNA accessibility. This is also illustrated by the cross arrows in the architecture (1 Multi-task architecture refined for the ATAC-seq use case (2 modalities)).

Now let's try to understand in the next section, how the losses are computed and what is the flow of tensors.

5 Losses

There are quite a few losses going on in the model. These are:

- Cross entropy loss (Classification)
- Reconstruction loss
- Translation loss (Cross-modal prediction)
- Generator loss
- Discriminative loss
- Contrastive loss

Let's try to see one by one, how they are being used, and what part of the model's parameters are updated for each of this loss.

For the generator, and discriminative loss, we have dedicated a new section, where first we introduce the GANs (6 Generative Adversarial Networks (GANs)). The same applies for contrastive learning, which is also a very crucial component of the modern self-supervised approach (7 Contrastive learning).

The cross-entropy loss is associated with the objective of the 1) classification of the cell types, and all the rest (reconstruction, translation, generator, discriminative, contrastive), are summed to get the total loss for the cross-modal prediction.

Of course, since we are dealing with a multi-task architecture, common parts of the networks are used, for knowledge transfer to happen. So a loss being associated with an objective isn't necessarily valid. But in our case, we make this distinction, because the authors use alternating training of these two objectives, and with the two losses associations mentioned above. More about the training at 8 Training.

It is worth mentioning that gradient calculation is linear with the loss function ($\frac{d\text{total loss}}{d\theta} = \frac{d\text{loss1}}{d\theta} + \frac{d\text{loss2}}{d\theta}$). So we could think each loss individually, compute the gradients, sum all of them, and do the weights update $w_{n+1} = w_n - \frac{d\text{total loss}}{d\theta} * \gamma$. To know which weights needs to be updated, we could ask, which weights have contributed to calculate the loss, as we have seen in the backpropagation as well 3 Multi-Layer-Perceptron (MLP).

Simply adding the losses, can lead for one of them to dominate the others. So effectively some losses may not have any influence [3]. This is something that we could also explore to improve (13 Ideas to improve UnitedNet).

5.1 Cross-entropy loss

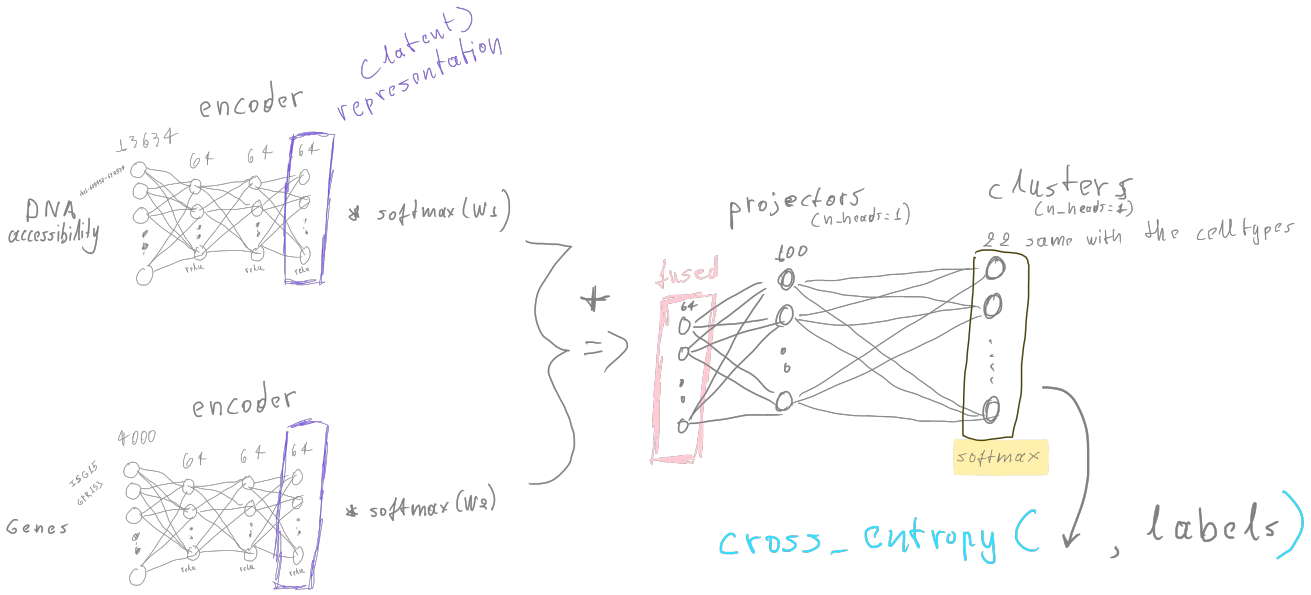

Figure 7: Model's parameters update based on cross entropy loss

Isolating the networks updated for the classification from 1 Multi-task architecture refined for the ATAC-seq use case (2 modalities), the cross entropy loss is responsible to update the parameters of the encoders, the weights for the fusioning, the projector, and the final cluster network.

Cross entropy is a widely used loss for classification tasks. Measumerements have been taken as well for balancing the classes with adaptive weighting for unrepresented ones (named as `class_weights`).

5.2 Reconstruction loss

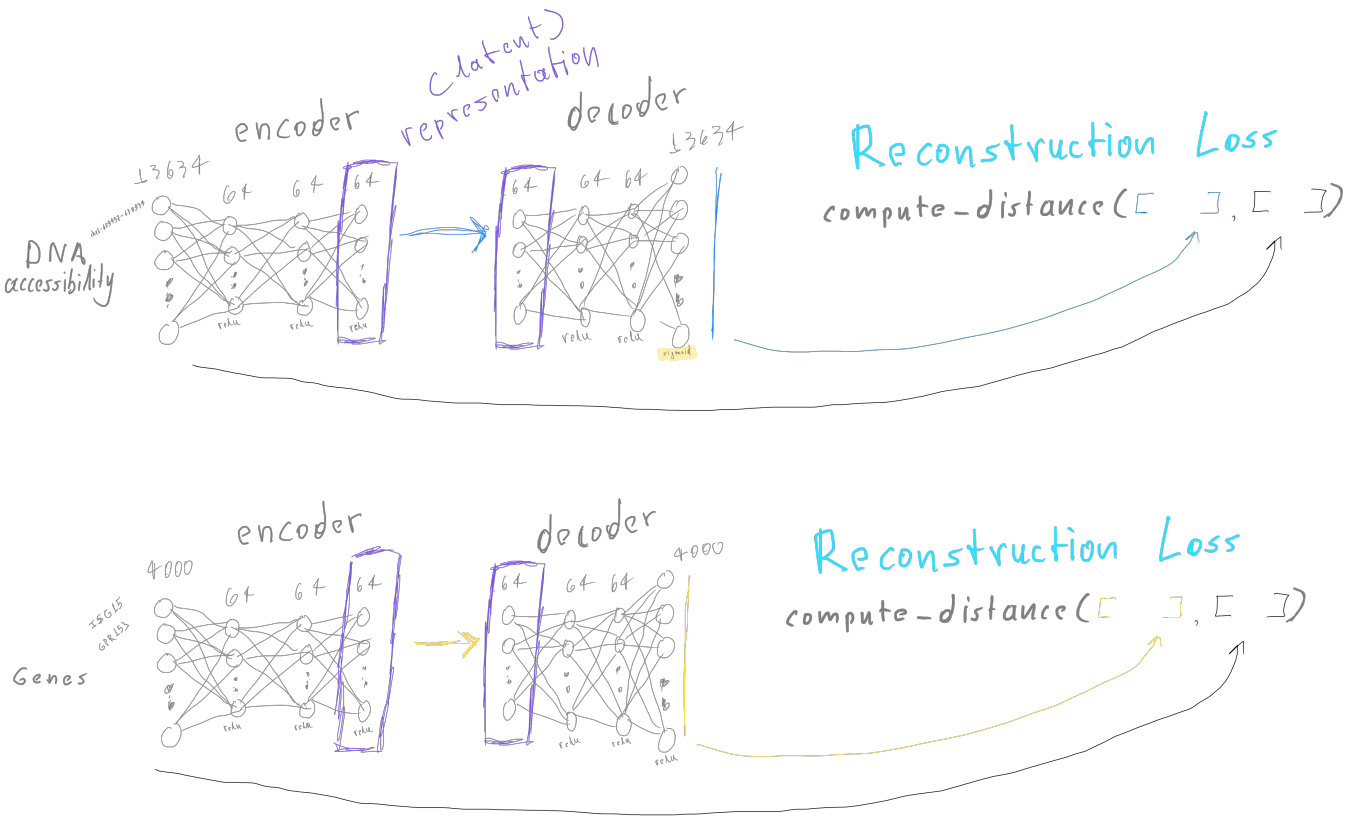

Figure 8: Model's parameters update based on reconstruction loss

The reconstruction loss is calculated for each modality. The loss is based on the similarity of the input and output tensor (`compute_distance`). Then the losses of each modality are summed to get the total reconstruction loss.

5.3 Translation loss

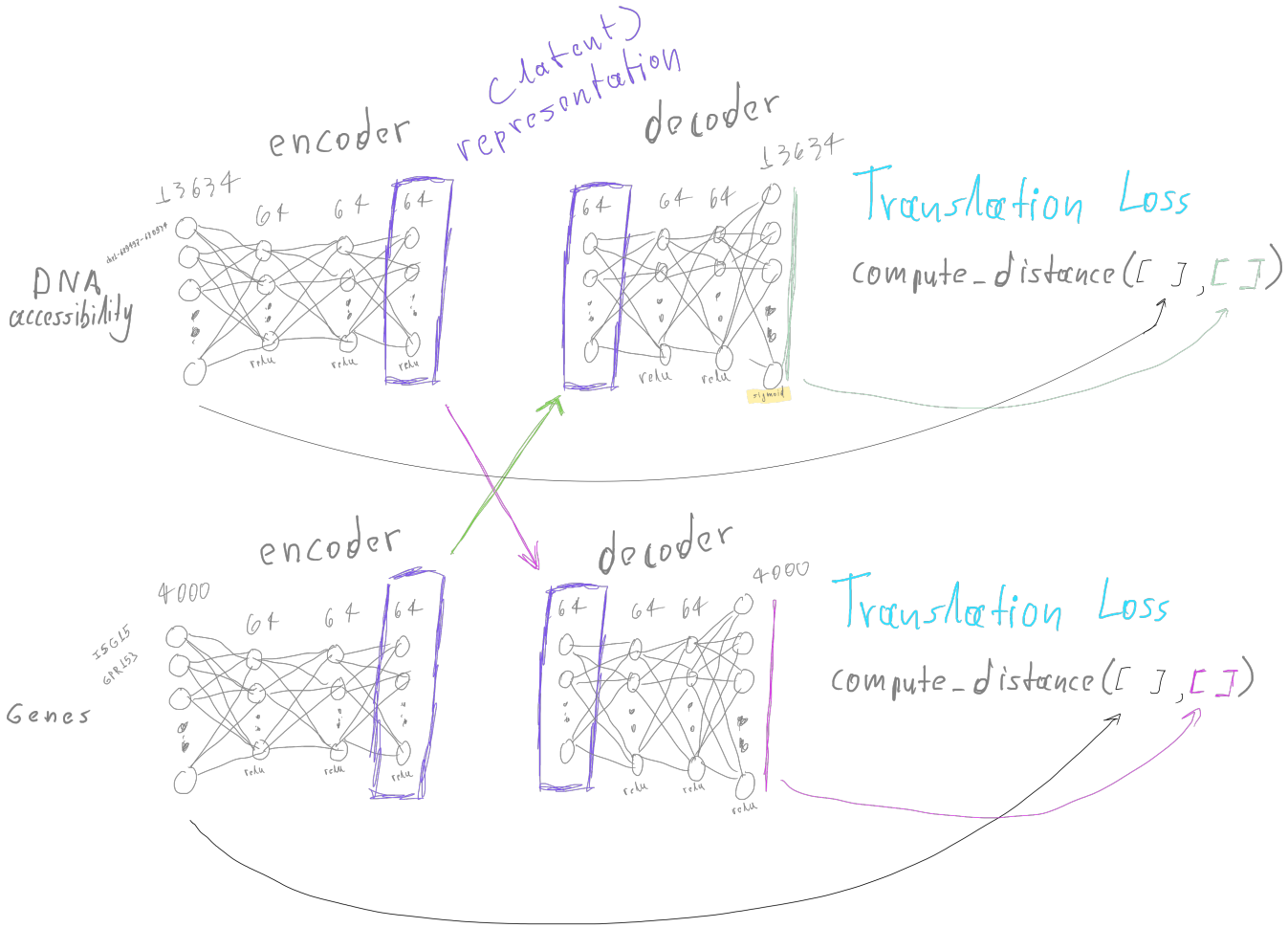

Figure 9: Model's parameters update based on translation loss

Similarly to the reconstruction loss, we have a translation loss, which measures the similarity between the prediction of each modality when using the other modality as input. These translation losses are computed separately for each modality and then summed to obtain the total translation loss.

5.4 Generator loss

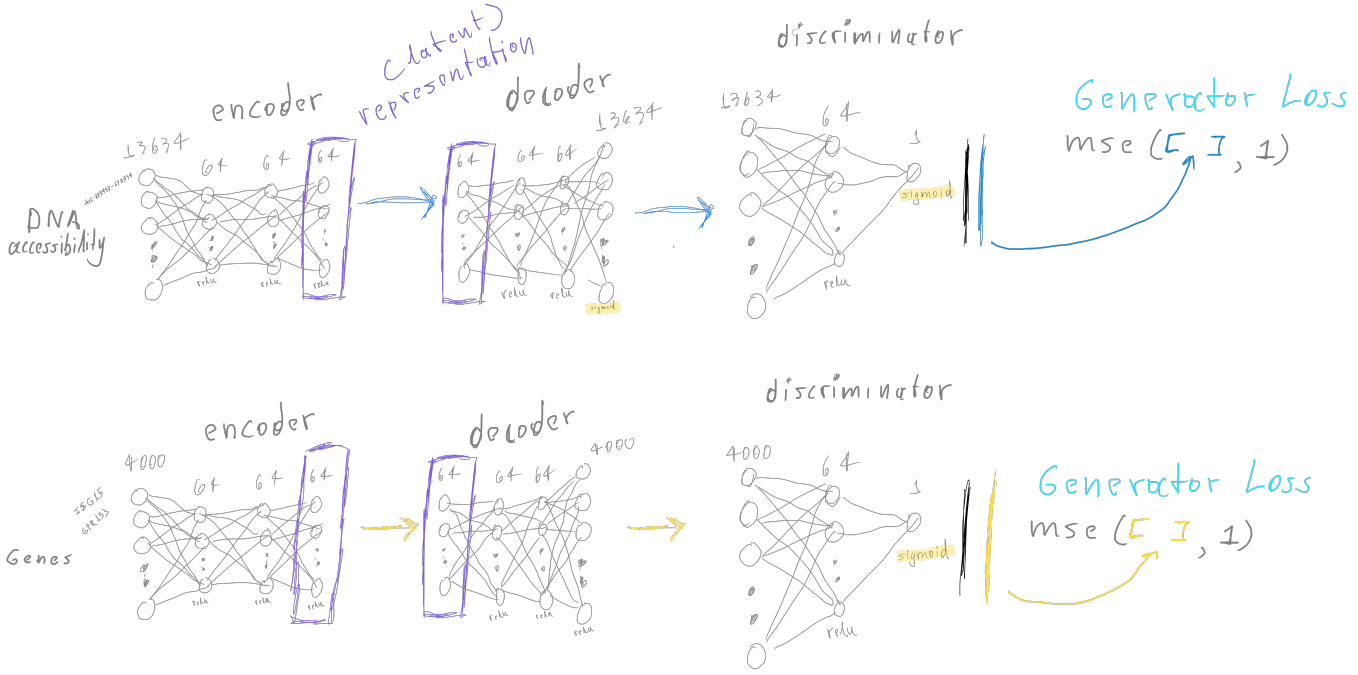

Figure 10: Model's parameters update based on generator loss

The generator loss is part of the enhancement of the autoencoder. The generator loss aims to fool the discriminator that its generated data is the real one. That's why we can see, that in the mean squared error loss, the expected value is 1, which means true.

The total generator loss is calculated by summing the generator loss of each modality.

5.5 Discriminative loss

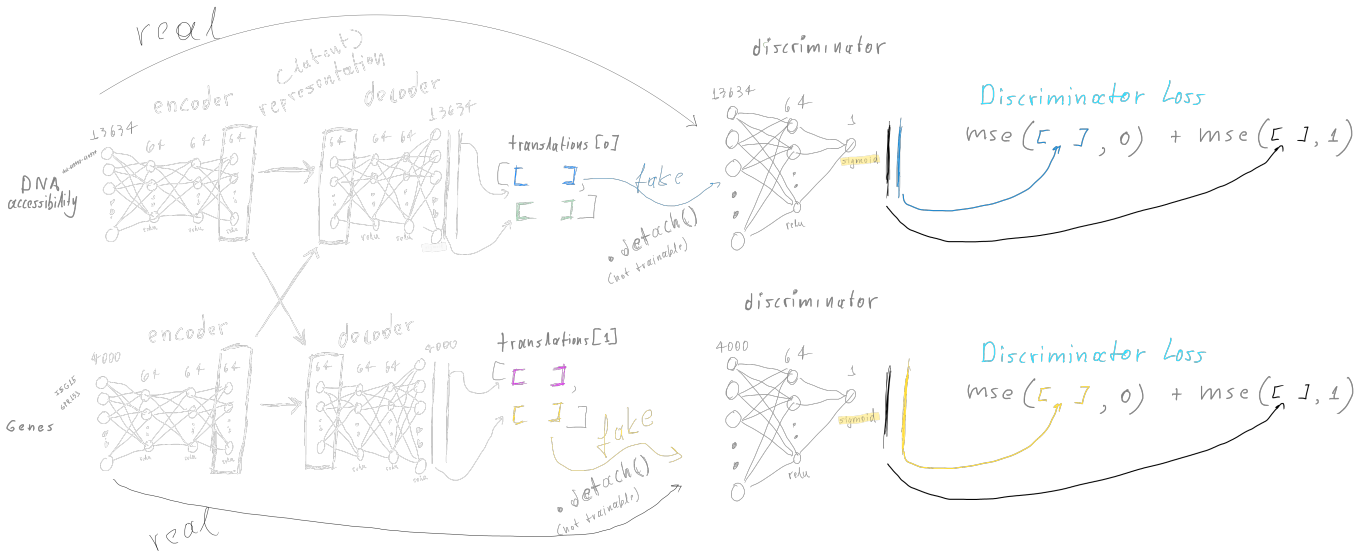

Figure 11: Model's parameters update based on discriminative loss

The discriminative loss is also part of the enhancement of the autoencoder. The goal is to identify which data comes from the generator (fake), and which is the original one (real). In the mean squared error loss, the data coming from the generator should be labelled as 0, which means fake, and the original data should be labelled as 1, which means real.

When calculating the discriminative loss, to avoid updating the weights of the rest of the network, focusing only on the weights of the discriminator network, we "forgot" the rest of the network, that the fake data is associated with. Thus, the pytorch function `detach()` used.

Now the discriminator loss should influence only the weights of the discriminator work, without the loss being propagated to the rest of the network. The total discriminator loss is calculated by summing up each of these losses per modality.

5.6 Contrastive loss

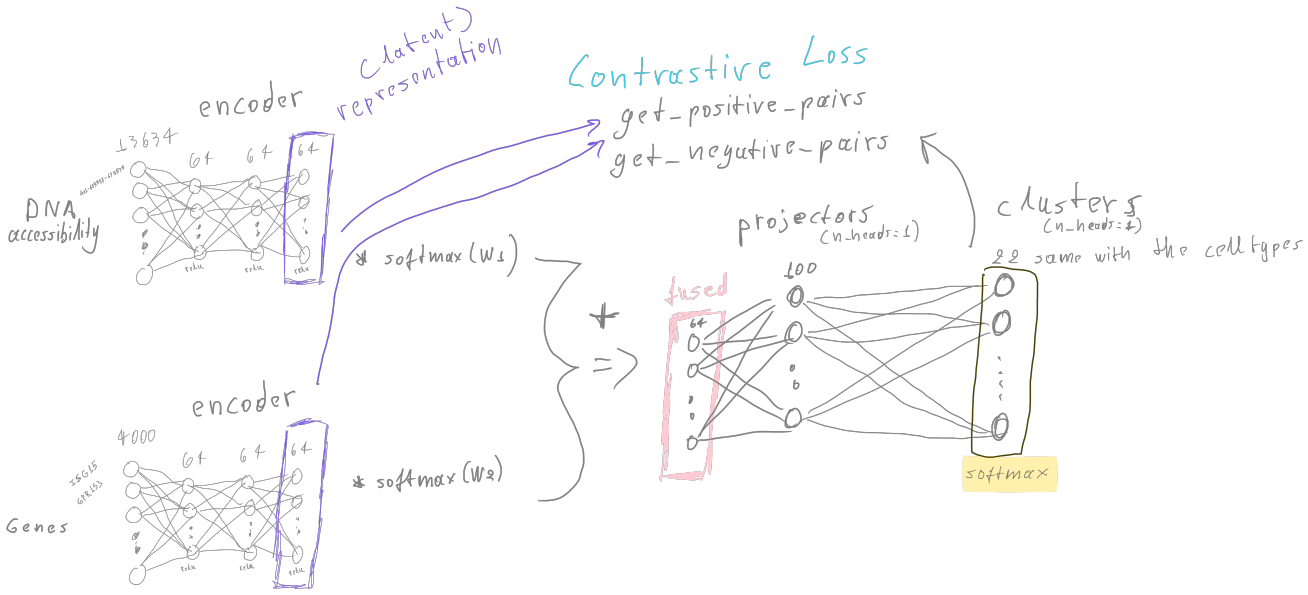

Figure 12: Model's parameters update based on contrastive loss

The idea of contrastive loss is to introduce the concept of positive and negative pairs, and to measure the similarity between them. Positive ones should be similar and negative ones not.

The way that positive pairs are defined in `UnitedNet` is based per instance. So the latent codes of the modalities per instance are positive pairs. The negative pairs are defined based on the prediction of the model. The latent codes that made different predictions are negative ones.

Although the contrastive loss utilized information from the predictions, the optimizer responsible to update the corresponding parameters is associated only with the encoders part.

6 Generative Adversarial Networks (GANs)

As we have have seen in the losses 5.5 Discriminative loss, and 5.4 Generator loss, the generator is responsible for fooling the discriminator, and the discriminator is responsible for identifying which data comes from the generator, and which is the original one. Thus, their tasks are adversarial and they compete with each other. This is an idea presented originally by [1].

7 Contrastive learning

We could explore how to improve the representations of the network by contrastive learning. Contrastive learning is part of the self-supervised paradigm, that flourishes lately.

- Representation Learning with Contrastive Predictive Coding
- Bootstrap Your Own Latent (BYOL)

8 Training

The two tasks as mentioned above, the 1) classification, and the 2) cross-modal prediction, they are being trained alternating, instead of jointly. This is claimed to have better results based on the authors.

Training alternating means that the losses defined for these two objectives (as mentioned in 5 Losses) are calculated first for the one → weights are updated, and then similarly for the other. More specificially for the `model.train`, the first task is the cross-modal prediction and the second the classification. This is happening for every epoch. The common part of the model between these two objectives are the encoders.

```
for epoch in tqdm(range(model.config[str_train_epochs])):
  epoch += 1
  model.cur_epoch = epoch
  model.train()
  for schedule in schedules:
      run_through_dataloader(model, dataloader_train, schedule, train_model=True)
```

As we can see in the above code snippet, the ‘schedule‘ represents the objectives.

Another interesting aspect of the training is what the authors call as `finetuning`, and `transfer`. More specifically in the code we have:

```
for test_batch in test_batches:
  adatas_train, adatas_test = split_data(test_batch)
  model = UnitedNet(f"{root_save_path}/{test_batch}", device=device, technique=atacseq_c
  model.train(adatas_train, verbose=True)
  model.finetune(adatas_train, verbose=True)
  model.transfer(adatas_train, adatas_transfer = adatas_test, verbose=True)
```

The `model.finetune` is exactly the same with the `model.train`, but instead of running first the `schedule` responsible for the cross-modal prediction task and then the classification, it runs first the classification and then the cross-modal prediction.

Regarding `model.transfer`, there is an actual leakage of the test set to the training of the model. We could assume that in the unsupervised task of the reconstruction, there is no harm for the model to see the test data, since our goal could be just the feature extraction for the subsequent classification task. Although this could be sensible since unsupervised is unsupervised meaning it requires no resources (e.g. labels), the evaluation of the model at the end is tested both on the classification and the cross modal task, which is incorrect in terms of fairly evaluating a model.

9 Preprocessing

9.1 ATACseq

2 modalities, 1) gene expression levels, and 2) DNA accessibility sites.

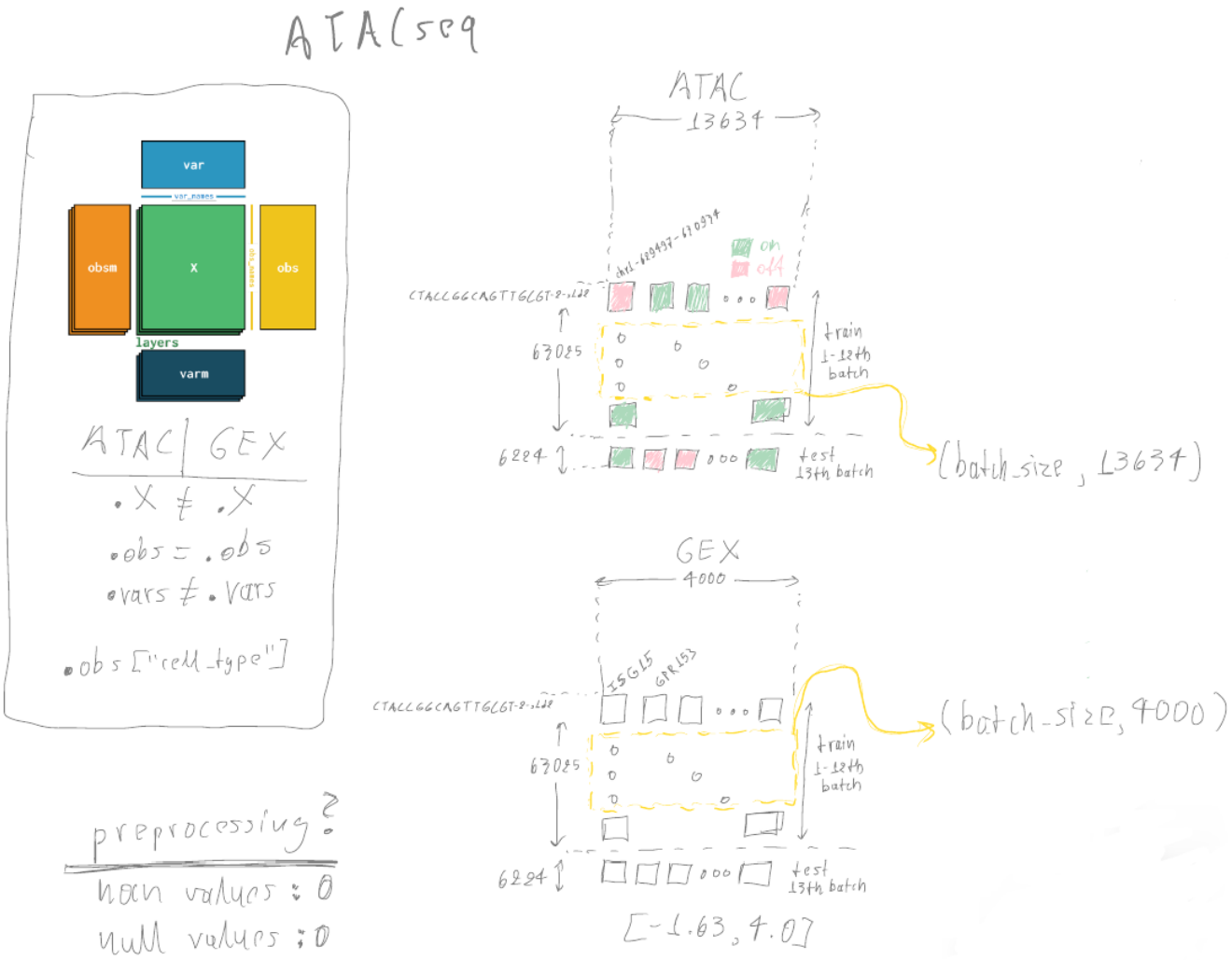

Figure 13

10 Relevance analysis

> Correlation doesn't imply causation, but causation implies statistical association
>
> Internet

Neural network models are associative models. Using explainable techniques, shed light on how the model make the prediction, but that doesn't mean that if for example the model used a particular gene to make the prediction of the type of the cell, that there is a **causal** relationship between this gene and the type of the cell.

Causal models can infer causal relationships, and with associative models, it is no more than just an association. This is very well explained in the SHAP library as well, here.

So, it should be noted, that SHAP values can only be helpful for guidance, but not for causal statements such as `the gene X seems that is a marker gene for the Y cell type`. A correct sentence would be `the gene X is associated with the Y cell type`.

Of course, if we assume that the model has learned biological rules, an explainable model could lead to biological insights. So if the model is able to test our hypothesis, can we use it a baseline for other set of hypotheses?

A quick way to understand how false a correlation is, let's have an example:

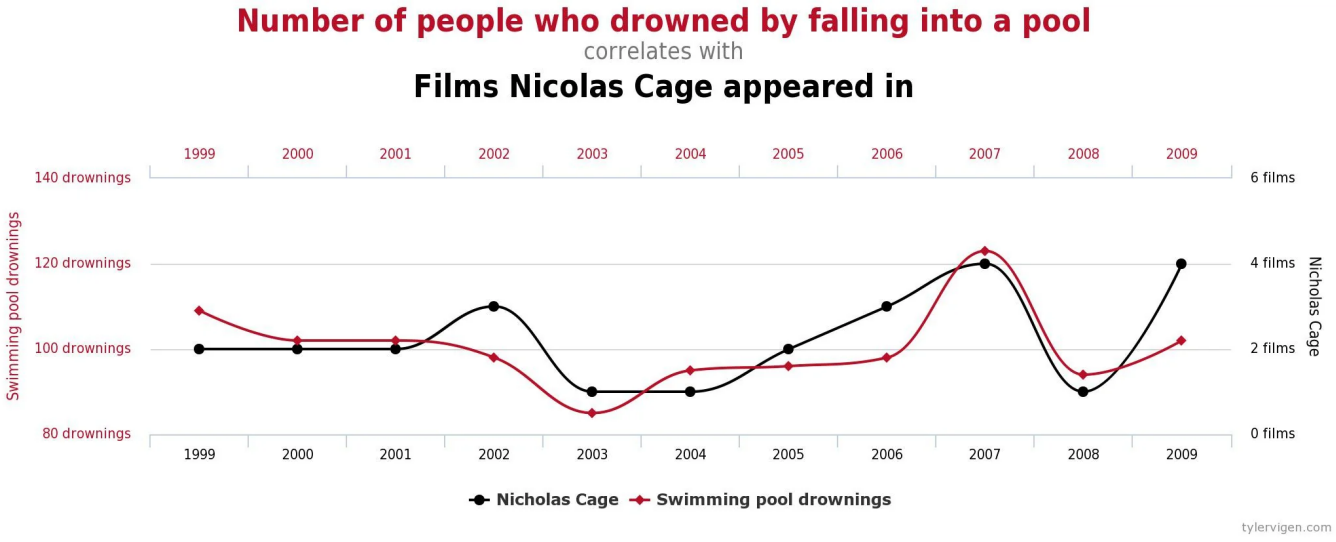

Figure 14: Correlation doesn't imply causation. Nicolas Cage films shouldn't be a causal factor for the swimming pool drownings.

This whole field is called causal inference, and causal AI or causal machine learning. A recent study of causality in single cell can be found by the Fabian Theis lab [7]. In a nutshell, the state of the art is not there yet to make any causal statements, and there isn't an established method for causality, since it could be assumed an almost impossible project, if there are no prior biological ground truths. The following section of the paper summarized this greatly:

"Causal approaches offer the hope to model the mechanisms that rule the behaviours and dynamics of complex systems such as cells. However, this is the very early age of causal machine learning applied to single-cell biology, and while many methods have been presented in this perspective, a large effort remains to be made to experimentally validate them, which has been challenging so far due to the relatively lack of perturbational data, a key pivot for causal discovery. Fortunately, continuous advances in experimental

technologies for single-cell sequencing should keep increasing the availability and quality of data."

Causality is a step beyond machine learning explainability, and it could be assumed as another improvement factor, for a causal analysis to be conducted, or a more thorough analysis of the shap values, see (13 Ideas to improve UnitedNet).

10.1 SHAP values

As we have mentioned above, shap values are used to help us understand what features the model considers important to perform a certain task.

In the ATAC-seq scenario, for the cell type classification task, we have two modalities, so we can make hypothesis about the genes and the DNA accessibility sites, being markers for cell type categories.

On the other hand, with the task of cross-modal prediction, we could make hypotheses for how important are the features of one modalitiy to make predictions for the other.

It is worth mentioning, that the authors calculate the shap values not for the whole multi-task architecture, but for sub-networks that are responsible for each task. We need to keep in mind, to make use of shap values, what is the input and what is the output of the model. A classification task of cell types couldn't for example provide insights on feature-to-feature relevance, if the task is not a feature-to-feature (cross-modal prediction) task.

Thus, in the code, there are two sub-networs, called `submodel_trans` (translation subnetwork, translation means cross-modal prediction) and `submodel_clas` (classification subnetwork).

```
class submodel_trans(torch.nn.Module):
    def __init__(self, bigmodel, enc_dec_id):
        super(submodel_trans, self).__init__()
        self.encoder = bigmodel.encoders[enc_dec_id[0]]
        self.decoder = bigmodel.decoders[enc_dec_id[1]]

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x


class submodel_clus(torch.nn.Module):
    def __init__(self, bigmodel):
        super(submodel_clus, self).__init__()
        self.encoders = bigmodel.encoders
        self.fusers = bigmodel.fusers[bigmodel.best_head].weights
        self.projectors = bigmodel.projectors[bigmodel.best_head]
        self.n_head = bigmodel.n_head
        self.clusters = bigmodel.clusters[bigmodel.best_head]
        self.prob_layer = bigmodel.prob_layer
        self.best_head = bigmodel.best_head

    def fusing(self, fuser, latents):
        weights = nn.functional.softmax(fuser, dim=-1)
        weighted_latents = torch.sum(weights[None, :] * torch.stack(latents, dim=-1), dim=
        return weighted_latents
```

```
def forward(self, *modalities):
    self.latents = [
        encoder(modality)
        for (encoder, modality) in zip(self.encoders, modalities)
    ]

    self.fused_latents = self.fusing(self.fusers, self.latents)

    self.hiddens = self.projectors(self.fused_latents)

    with torch.no_grad():
        w = getattr(self.clusters, "layers")[0].weight.data.clone()
        w = nn.functional.normalize(w, dim=1, p=2)
        getattr(self.clusters, "layers")[0].weight.copy_(w)

    self.cluster_outputs = self.clusters(self.hiddens)

    self.predictions = self.prob_layer(self.cluster_outputs)

    return self.predictions
```

Having as reference the multi-task architecture (1 Multi-task architecture refined for the ATAC-seq use case (2 modalities)), these two networks split it to single-task networks. These single-task networks are used only for forward passes, and not for training. The training is performed on the whole mulit-task architecture.

Thus, feature-to-feature analysis should be based on the `submodel_trans`. More specifically the shap values should be calculated for every modality. So in the ATAC-Seq scenario, we need a subnetwok that has as input the genes, and outputs the DNA accessibility, and one more network, that has as input the DNA accessibility, and outputs the genes (two instances of `submodel_trans`). In total two networks, to assess for predicting DNA accessibility what genes are the most important, and the other way around, for predicting genes, what DNA accessibility are the most important.

For class type relevance analysis, we just need one network to use, the `submodel_clas`.

11 Perturbation modeling

As we have mentioned in 1 Introduction, the goal is to design a new multi-modal multi-task framework for perturbation modelling, leveraging multi-omics, and potentially spatial information.

Let's examine the tasks of perturbation modelling.

But what is a perturbation? What type? How the model can learn different types of perturbations? They can be encoded as directions in the latent space.

- Out of distribution prediction of perturbed gene expression profile. That could mean either predict for a different cell type, the perturbed gene expression profile for a particular perturbation, or it can be to predict for the same cell type, but for a different perturbation.
- Identify the type of perturbation.

Our architecture can be based on transformers or simpler models such as variations of autoencoders (e.g. masked variational autoencoder).

Regarding, transformers, there is a thorough survey ("Transformers in single-cell omics: a review and new perspectives") with the main takeaway that is a very promising but not yet mature research direction. There are many flaws such as lack of an evaluation framework, instabilities, lack of diversifying data, and lack of the sequential property (equivalent of positional embeddings). On the other hand, order can be induced by spatial transcriptomics, and then potentially visual transformer architectures can be leveraged.

Among the foundational models, the one with complete documentation, interactive tutorials, and perturbation modelling case studies is scGPT. On the other, it is claimed that a perturbation foundational model is lacking [5].

Some things that we could potentially investigate would be to use a pretrained foundational model such as scGPT, and then fine-tuning it with a multi-task architecture for a series of downstream tasks. It is worth noting, that this is the equivalent of leveraging self-supervised, representation learning with the multi-task paradigm. This approach has been already tested for NLP tasks ("Multi-Task Deep Neural Networks for Natural Language Understanding"), and seems a well established approach.

Thus, to have a more robust baseline to further our studies in perturbation modeling, we will use simpler models that are well established in the field. Specifically, the butterfly framework claims that.

Alignment of variational autoencoders. There is a lot of work for two modalities, private and shared latent spaces, and promises that it can be extended for more modalities, but without any studies testing or verifying this. Thus, our first step could be to focus on two modalities, proteins, and RNA, including perturbation information. Afterwards, we could explore how can we integrate more modalities, e.g. niche, to take into account spatial information.

The main key point of our research would be to check if a multi-task framework can benefit all the tasks described one by one by the butterfly framework. We could also leverage the latest optimal transport research on the perturbation perdiction task. We can also analyze any potential positive and negative transfer, and the importance of losses coefficients.

Regarding an attempt to standardize the evaluation of transformers in single cell-data, we have scEval[9], that showed that scGPT is probably the best overall one.

On the other hand, there is a very new spatial foundational model, the Nicheformer, and there is no comparison data.

Model	Open Source?	Batch Effect Correction	Cell-type Annotation	Multi-omics Data Integration	Imputation	Gene Function Prediction	Perturbation Prediction	Gene Network Analysis	Simulation
scGPT	✓	✓	✓	✓	✓	✓	✓	✓	✓
Geneformer	✓		✓			✓			
scBERT	✓		✓						
CellLM	✓		✓						
tGPT	✓	✓							
scFoundation			✓		✓		✓		
SCimilarity		✓	✓						

More versatile

✓ Functions from the original design ✓ Functions added by scEval

Data for this table were collected until **August 1st, 2023**.

11.1 Datasets

There is a tool as well to explore perturbation datasets http://projects.sanderlab.org/scperturb/. There is pertpy as well.

- ECCITE-seq

11.2 Existing Methods

- Nicheformer.

  Transformer for cellular representation, trained on spatial transcriptomics data. Fine tuned for downstream spatial tasks.

- scGEN
- GenKI
- SCENIC+
- CellOT/CINEMA-OT

11.3 Spatial tasks

- Spatial density prediction or niche
- Region label prediction
- Inferring niche from gene expression profile
- Transfer spatial labels to disassociated cells

source: Nicheformer

12 Requirements

- Explainability, causality is crucial for single cell
- Multi-task learning has potential, either with an architecture of simpler models e.g. autoencoder variations, or more sophisticated approaches with transformers, and foundational models.
- Multi-omics
- Leverage spatial transcriptomics
- Integrate perturbation modeling tasks

Ideally, we would like to tackle all the above.

13 Ideas to improve UnitedNet

“ Moreover, there are some remaining questions such as why multi-task learning can improve multi-modal data analysis, how to reduce randomness in the neural network training, how to design other loss functions to integrate more tasks (e.g., single-cell trajectory inferencing) ”[6]

- Transformer
- Monitor the losses. Currently, there are no coefficients for each loss. Thus, the scale of one loss could shadow the other [3].
- Multi-task common loss
- Improve representations e.g. Variational autoencoder
- Resnet for encoder
- Causal inference ?
- Why they don’t use a validation set? It seems they have only training, testing split.

“ Preliminary results suggest that, while single-cell transformers may be able to generalize across datasets, state-of-the-art task-specific models as well as simpler models such as logistic regression often outperform them, even on tasks with few annotations, and that zero-shot performance of the current transformers is questionable” (https://www.nature.com/articles/s41592-024-02353-z)

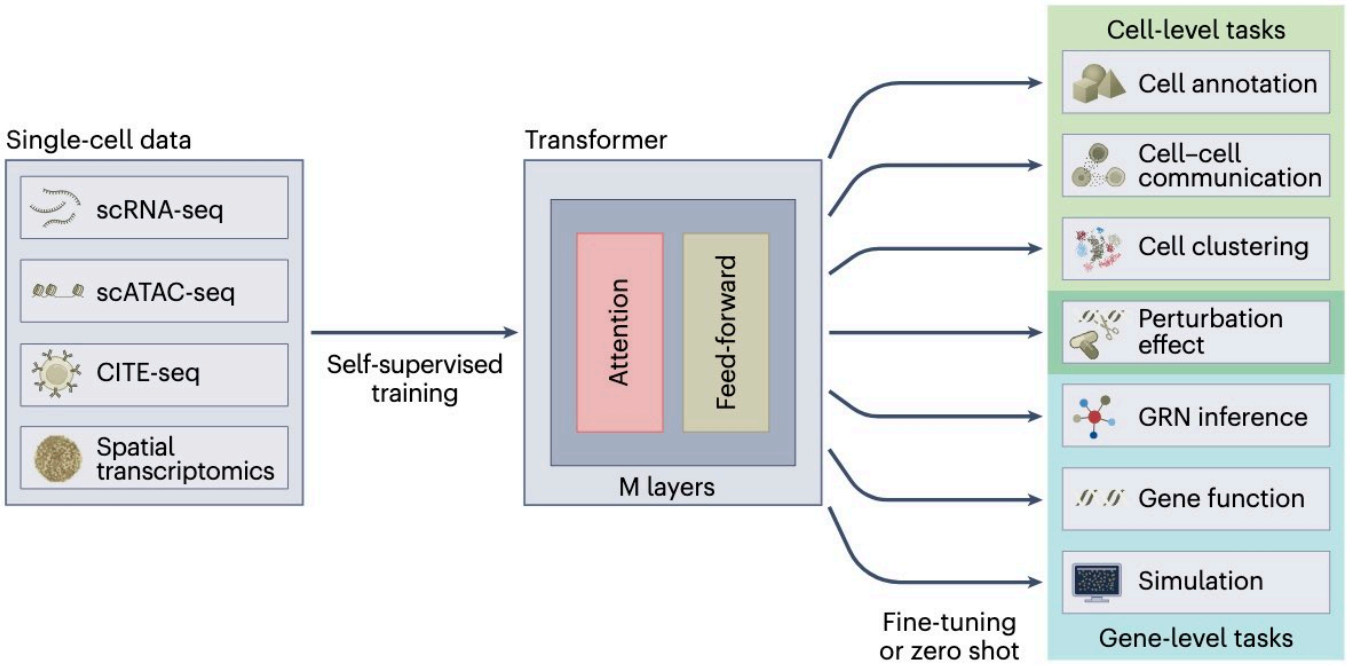

Figure 15: (https://www.nature.com/articles/s41592-024-02353-z)

References

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks.

[2] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked Autoencoders Are Scalable Vision Learners.

[3] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics.

[4] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes.

[5] Artur Szałata, Karin Hrovatin, Sören Becker, Alejandro Tejada-Lapuerta, Haotian Cui, Bo Wang, and Fabian J. Theis. Transformers in single-cell omics: A review and new perspectives. 21(8):1430–1443.

[6] Xin Tang, Jiawei Zhang, Yichun He, Xinhe Zhang, Zuwan Lin, Sebastian Partarrieu, Emma Bou Hanna, Zhaolin Ren, Hao Shen, Yuhong Yang, Xiao Wang, Na Li, Jie Ding, and Jia Liu. Explainable multi-task learning for multi-modality biological data analysis. 14(1):2546.

[7] Alejandro Tejada-Lapuerta, Paul Bertin, Stefan Bauer, Hananeh Aliee, and Fabian J Theis. Causal machine learning for single-cell genomics.

[8] Xu Yang, Cheng Deng, Feng Zheng, Junchi Yan, and Wei Liu. Deep Spectral Clustering Using Dual Autoencoder Network. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4061–4070. IEEE.

[9] Hongyu Zhao, Tianyu Liu, Kexing Li, Yuge Wang, and Hongyu Li. Evaluating the Utilities of Large Language Models in Single-cell Data Analysis.