KRATOS: Context-Aware Cell Type Classification and Interpretation Using Joint Dimensionality Reduction and Clustering

Zihan Zhou*
zhou1248@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Zijia Du*
shduzijia@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, CHN

Somali Chaterji
schaterji@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Abstract

A common workflow for single-cell RNA-sequencing (sc-RNA-seq) data analysis is to orchestrate a three-step pipeline. First, conduct a dimension reduction of the input cell profile matrix; second, cluster the cells in the latent space; and third, extract the "gene panels" that distinguish a certain cluster from others. This workflow has the primary drawback that the three steps are performed independently, neglecting the dependencies among the steps and among the marker genes or gene panels. In our system, KRATOS, we alter the three-step workflow to a two-step one, where we jointly optimize the first two steps and add the third (interpretability) step to form an integrated sc-RNA-seq analysis pipeline. We show that the more compact workflow of KRATOS extracts marker genes that can better discriminate the target cluster, distilling underlying mechanisms guiding cluster membership. In doing so, KRATOS is significantly better than the two SOTA baselines we compare against, specifically 5.62% superior to Global Counterfactual Explanation (GCE) [ICML-20], and 3.31% better than Adversarial Clustering Explanation (ACE) [ICML-21], measured by the AUROC of a kernel-SVM classifier. We opensource our code and datasets here: https://github.com/icanforce/single-cell-genomics-kratos.

CCS Concepts

• **Computing methodologies**; • **Applied computing**; • **Information systems** → **Information retrieval**;

Keywords

Single-cell RNA analysis, DNN, machine learning explanation, classification, perturbation methods, clustering.

ACM Reference Format:
Zihan Zhou, Zijia Du, and Somali Chaterji. 2022. KRATOS: Context-Aware Cell Type Classification and Interpretation Using Joint Dimensionality Reduction and Clustering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3534678.3539455

1 Introduction

Single-cell RNA-sequencing (sc-RNA-seq) technology has enabled the high-throughput interrogation of many aspects of genome biology, including gene expression, DNA methylation, histone modification, chromatin accessibility, and genome 3D architecture resulting in high-dimensional single-cell omics datasets [37]. All these analyses enable the transcriptome-wide measurement of gene expression in individual cells, essential for identifying cell-type-specific clusters, characterizing cell heterogeneity in temporal stages of disease and development, and highlighting semantic clonal structures. These outputs can be arranged into high-dimensional, albeit sparse, matrices whose rows are different cells, and columns are the cell's feature attributes, *e.g.,* gene expression. Converting these high-dimensional matrices to a low-dimensional latent space with semantic architecture will enable more accurate cell clustering, as observed in our motivating Figure 1, with more interpretable downstream analyses [41]. Identifying gene panels that are important to certain cell types highlight the key differences among cell types in the latent space. In this work, KRATOS, we focus on the analysis of sc-RNA-seq datasets. We aim to answer the following overarching question:

> *Given a sc-RNA-seq dataset and corresponding cell types, how can we find the gene panels (marker genes) that define the different cell clusters?*

A useful methodology in such analysis is instantiating a pipeline starting with dimensionality reduction that learns a latent representation for the input dataset, then clustering the latent embeddings into groups, and finally, post-hoc explanations of the cell types constituting the clusters using ML interpretability techniques. The primary caveat of the traditional three-step workflow is that the steps are performed independently [3, 20, 26]. Optimizing each step separately cannot guarantee optimal clustering of genes and biologically meaningful cluster membership patterns. For example, for the human peripheral blood mononuclear cell (PBMC) dataset [44], orchestrating the dimensionality reduction and clustering steps

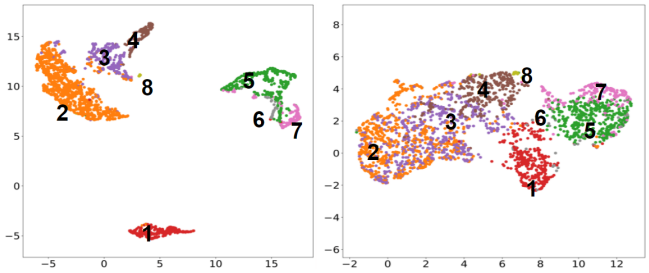


Figure 1: **Visualization of the embedding layer in KRATOS and ACE using UMAP on the human PBMC dataset. Here default parameter settings are used for UMAP. (a) and (b) are latent space embeddings using KRATOS and ACE, respectively. We find that KRATOS performs better on cell clustering, with cells of the same type clustered more tightly.**

*Both authors contributed equally to this research.

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08.
https://doi.org/10.1145/3534678.3539455

separately result in Adjusted Mutual Information (AMI) of 0.6989, while a direct classification step introduced in our system Kratos resulted in a value of 0.8547. Here, AMI is a clustering metric used for clustering performance evaluation, where a higher AMI indicates that the clustering result is closer to the ground truth. Some works aimed at solving this problem through a combination of the three steps in a more integrated pipeline. For example, Adversarial Clustering Explanation (ACE) [19] bundles all three steps into one integrated neural network-based framework, where the last two steps are conducted separately, in the latent space generated in the first step. Nevertheless, the three parts of the network still have their own objective functions and are optimized sequentially, which means that ACE is still solving a separate optimization function for each step. This is akin to solving an optimization problem using greedy steps, which does not guarantee a global optimum. Consequently, we transferred this three-step workflow into a two-step one, in which we fuse the first two steps into a joint single step using a combined optimization function. This optimization function creates the latent embeddings and then clusters in that latent space with higher performance. In the first step, we combine the dimensionality reduction and clustering by solving a single optimization problem. The rationale for doing this is that we are using the datasets with labeled cell types. We assume that, with the labels, the neural network (NN) will find the patterns specific to each cell type, and thus, the clusters can be placed farther from one another in the latent (embedding) space. Besides, in some other works like ACE, the cell type labels are also implicitly used in their clustering step. Furthermore, in the dimensionality reduction of common three-step workflows, the cells with similar gene expressions are assumed to be of the same type. However, this may not be true as the relation between gene expression and cell type is not necessarily continuous. Thus, instead of counting on local similarity between gene expression levels, we directly learn the embedding from the cell type. For the second part of Kratos, we demonstrate the extensibility of Kratos to the third part of the pipeline to be able to act in a plug-and-play mode. We accomplish this by using different interpretability techniques — perturbation methods [19] and gradient-based methods [36, 37] — to explain the classification result (imagine clustering to be a multi-class classification step). Here we use an explanation algorithm inspired by the adversarial machine learning technique [42], first leveraged by ACE. Specifically, we induce small perturbations into the input data, and see how the perturbations influence the cluster assignment given by the Kratos's NN-based classifier. These "adversarial perturbations" enable the NN to find a gene set signature for each class, when contrasted against all the other classes (one-versus-others). We define our explanation as finding out the rank of genes for each class according to the value of the induced perturbation that changes the classification layer's logit. In summary, in contrast with the SOTA sc-RNA-seq analysis integrated pipeline, ACE [19], our system Kratos is novel in the following aspects. First, we jointly optimized the dimensionality reduction and clustering step into one classification step with a single objective function, making better use of the labeled sc-RNA-seq dataset through optimizing over a single categorical objective function. This allows the model to directly search for the global optimum. Second, our NN-based classifier is straightforward compared to other complex models, reducing the number of hyperparameters that need to be tuned (6 for SOTA versus 2 for Kratos) and decreasing the training time by 90% (from 161 seconds to 17 seconds). As for the explanation, most of existing works treat genes independently when analyzing their impact on a cell type [17, 18, 30]. These works tend to neglect the fact that many genes have dependencies in the sense that they may be a part of the same gene regulatory network. In Kratos, we extract a panel of (important) genes rather than individual marker genes to distinguish the cell types, enabling a more holistic approach to classification and downstream analyses.

To demonstrate the advantages of Kratos, we conduct experiments on real sc-RNA-seq datasets (*e.g.,* the human PBMC dataset and the bigger human pancreas dataset (Baron), with higher class imbalance [6]), and compare our clustering and interpretability results with SOTA. We use the area under the receiver operating characteristic curve (AUROC) of the support vector machine (SVM) classifier and Pearson Correlation Coefficient (PCC) among the top marker genes to demonstrate the discriminative power and redundancy, respectively. We also use clustering metrics (Silhouette Score, AMI, and Adjusted Rand Index (ARI)), and visualization methods (t-Distributed Stochastic Neighbor embedding (t-SNE)) [39] and the newer method, Uniform Manifold Approximation and Projection (UMAP) [25], given their stochasticity and dependence on the initialization and hyperparameters) to evaluate Kratos against SOTA.

Contributions. We summarize Kratos's contributions as follows.

(1) Based on the current SOTA sc-RNA-seq explanation workflows, our system combines the first two steps, and reaches a superior performance, which is 5.62% superior to Global Counterfactual Explanation (GCE) [27] and 3.31% superior to ACE [19], measured by the AUROC of the SVM classifier used to compare the target cluster with the rest of the clusters. The intuition is that we leverage a joint optimization and transform the dimensionality reduction and clustering problems into a classification problem.
(2) For evaluation of integrated dimensionality reduction and clustering, we use the extracted top-k genes to train a radial basis function (RBF) kernel SVM-based binary classifier that identifies the target class against others. Additionally, we included additional validation metrics, such as clustering metrics and visualizations, to demonstrate the performance of Kratos's learned model and the significance of the interpretability of the gene panels that constitute the different clusters.
(3) We evaluate Kratos on one simulated dataset, two real sc-RNA-seq datasets , and compare against other baselines. Our system outperforms in terms of its discriminative power of identifying key genes on the PBMC dataset (around 3% using all the intepretability methods), convergence time of training Kratos's classification layer (∼ 90% lower), and lower number of hyperparameters that require tuning (from 6 for ACE [19] to 2 for Kratos). We also extend our system to a non-genomics dataset, specifically the MNIST dataset of handwritten digits [16], to show Kratos's generalization power. In particular, for the MNIST dataset, we had to tune the iterations, a gradient-related coefficient, and the margin to produce meaningful perturbations.

The rest of the paper is organized as follows. We provide pertinent preliminaries in Sec 2, related works in Sec 3, Kratos's design in Sec 4, the SOTA, baselines, and datasets in Sec 5, and end-to-end

evaluation of KRATOS in Sec 6. Finally, we conclude and discuss future directions in Sec 7.

2 Background

2.1 Single-cell RNA datasets

Single-cell RNA datasets often use samples that span locations, physiological profiles, multiple laboratories, and a plethora of different sc-RNA-seq protocols, resulting in high cell-to-cell heterogeneity. sc-RNA-seq counts the gene expression levels (or other feature attributes) in different cells, at an unprecedented single-cell granularity, and records them into cell profile matrices. These matrices are typically used to reveal the relation between single-cell RNA expression levels and the corresponding cell types. However, the analysis of the cell profile matrix requires reliable data integration, which is challenging due to the noise from the heterogeneity in clonal cell populations. This heterogeneity could stem from nested batch effects — systematic noise that is due to the noise generated from every batch of cells extracted. Usually, from an organism, single-cell RNA matrices contain thousands of cells and tens of thousands of genes, while the actual functioning genes that differentiate the cell types are far lower, making the matrices sparse.

2.2 Analysis pipeline

To handle these difficulties, a common three-step workflow has been used in the domain, the most recent examples being [3, 20, 26]. First, a dimensionality reduction algorithm constructs the latent space, projecting the features into a compact latent space where it is easier to cluster the data, while also denoising it. Autoencoders [2], Multidimensional scaling (MDS) [32], Principal Component Analysis (PCA) [28], and Factor Analysis (FA) [4] have been used in this step. Then, clustering algorithms like k-means [22], Hierarchical clustering [40], or Louvain clustering [13], are used to group the cells that are similar to each other in this lower-dimensional manifold. Finally, the differences between groups are traced that contain the information of the underlying cell mechanisms.

2.3 Clustering Metrics

To evaluate KRATOS, we have used the following clustering metrics: Silhouette score, AMI, and ARI, described next.

2.3.1 Adjusted Rand Index: Rand index (RI) is a similarity measure between two clusterings, and is defined as the ratio of the number of agreements, assigned to the same or different labels by two clusterings, between two clusters to the total number of pairs of data points, C(n, 2). Rand index ranges between 0 and 1, where a higher score indicate a higher similarity between the two data clusterings. However, the expected value of Rand index for two random partitions is not constant. The ARI is the correctness-for-chance version of the Rand index. ARI generally ranges between 0 and 1 with $E[ARI] = 0$ for two random clusterings and 1 for two identical clusterings. The larger range and fixed baseline increases the sensitivity of the index. No assumption is made on the structure of the dataset and hence ARI can be used to compare clusterings obtained using two entirely different clustering algorithms. However, ARI may produce poor results in the case of high class imbalance [31]. The expression for ARI is:

$$\frac{\mathrm{RI} - E[\mathrm{RI}]}{\max(\mathrm{RI}) - E[\mathrm{RI}]} \tag{1}$$

which can be expressed as:

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \frac{\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}}{\binom{n}{2}}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \frac{\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}}{\binom{n}{2}}} \tag{2}$$

Where X and Y are two clusterings, a_i is the size of cluster X_i, b_j is the size of cluster Y_j, n_{ij} is the number of data points common to clusters X_i and Y_j. Despite this drawback, ARI is a popular clustering metric, usually combined with other metrics like AMI.

2.3.2 Adjusted Mutual Information: AMI computes the similarity between two clusterings by measuring the agreements between two assignments ignoring permutations. It is also adjusted for chance so that two random clusterings (predicted and actual) produce a score of zero. AMI is bounded between -1 and 1, where a score of 1 means perfect matching between the clusterings (with or without permutation). AMI between the assignments U and V is defined as:

$$\mathrm{MI}(U, V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N} \log\left(\frac{N|U_i \cap V_j|}{|U_i||V_j|}\right) \tag{3}$$

$$\mathrm{AMI} = \frac{\mathrm{MI} - E[\mathrm{MI}]}{\mathrm{mean}(H(U), H(V)) - E[\mathrm{MI}]} \tag{4}$$

Where H denotes entropy. AMI has no preference on data structure; thus, the similarity stems from the statistics of two clusters.

2.3.3 Silhouette Score: Silhouette Score quantifies how close a data point is to the assigned cluster compared to other clusters, and well-separated clusters are rewarded by a high Silhouette Score. To calculate the Silhouette Score, the Silhouette Coefficient has to be computed for each data point in the dataset. The Silhouette Coefficient is mathematically defined as:

$$s = \frac{b - a}{max(a, b)} \tag{5}$$

Where a is the mean distance between a sample and all other points in the same class. b is the mean distance between a sample and all other points in the next nearest cluster. The average Silhouette Coefficient of all the samples is defined as the Silhouette score of the entire clustering. The value of Silhouette score is bounded between -1 and 1, where a high value indicates that the data point is well matched to its own cluster and poorly matched to the other clusters (which is ideal). A value close to zero is indicative of the presence of overlapping clusters. A negative value might mean that the clustering has either too many or too few clusters or that the clustering is incorrect. In this way, Silhouette index can be used as a consensus index to compute the optimal number of clusters in a dataset. Silhouette index does not favor a very large number of clusters. Thus, trivial clustering in which every data sample is an individual cluster does not produce a perfect score of 1.

2.4 Interpretability methods

We evaluate two kinds of ML interpretability methods for the explanation step, which is KRATOS's last step — Carlini-Wagner (CW) attack-based methods and Gradient-based methods.

2.4.1 Carlini-Wagner (CW) attack based-Method: This has its roots in the adversarial ML literature [9, 42], where models are exploited to generate attacks to trigger malfunction in the target model. Specifically, in [9], the authors added a perturbation to the given

image to alter the class assignment. CW's objective function is:

$$\begin{aligned} \min \quad & \|x - x^{'}\|_p \\ \text{s.t.} \quad & C(x^{'}) = t \\ & x^{'} \in [0,1]^n \end{aligned} \tag{6}$$

Where x and $x^{'}$ are the original object from class s and the perturbed object. $C(x)$ is the classification function. t is a class label other than s. This problem's solution can be approximated as:

$$x^{'} = x - \epsilon sign(\nabla loss_{F,t}(x)) \tag{7}$$

Where F is the categorical activation (usually $softmax$), $loss$ is the categorical loss function (cross entropy), and ϵ is a small coefficient like learning rate. In ACE [19], CW-like perturbations are used for the explanation part in they can identify the genes that are more related to the cell type.

2.4.2 Gradient-based Methods: We also equip KRATOS with two gradient-based methods — Integrated Gradient [37] and SmoothGrad [36]. Both methods consider the gradient of the objective function relative to the input feature as a measure of the feature's contribution. In SmoothGrad, random noise is added to the input to average out the gradients to make the feature score more robust. The methods are defined as:

$$SmoothGrad_c(x) = \frac{1}{n}\sum_{1}^{n} \frac{\partial S_c(x + N(0,\sigma^2))}{\partial x} \tag{8}$$

Where $S_c(x)$ is the activation of the input x of the $c-th$ class. n is the number of random noises sampled from the distribution $N(0,\sigma^2)$.

In Integrated Gradient, the gradients along the path from a baseline (usually 0's) to an input are computed and summed up as each feature's score. The Integrated Gradient for feature i of input x is defined as:

$$IntegratedGrads_i = (x_i - x_i^{'}) \times \sum_{k=1}^{m} \frac{\partial F(x^{'} + \frac{k}{m} \times (x - x^{'}))}{\partial x_i} \times \frac{1}{m} \tag{9}$$

Here m is the number of steps along the path to approximate an integral. Integrated Gradients uses the Aumann-Shapley method [5] from cooperative game theory, aimed at allocating credit in a cooperative game, when an ML model is imagined to be a game [12]. The Aumann-Shapley value is applicable to infinitesimal games (games with a continuum of moves, each of which in their own right have an infinitesimal contribution on the game's outcome, yet, collectively, they make a more noticeable difference), as opposed to atomic games, in which there are discrete moves in the game.

3 Related work

Workflow for sc-RNA-seq data anlysis: Our work aims at finding out the marker genes for each cell type using a NN-based workflow. The common workflow for scRNA-seq data analysis [26] is to: #1) project the cell-gene expression data into a lower-dimensional space; #2) identify the groups of cells that are similar to each other in the lower-dimensional space and clustering them; #3) adding explanations to these cell clusters. To improve the integration between the three steps, ACE [19] "neuralizes" the clustering step and concatenates it with a deep autoencoder, on one side (step #1), and with adversarial perturbations, on the other end (step #3), to test the robustness of the clustering step. For the first step — dimensionality reduction — ACE uses the autoencoder architecture from SAUCIE [2], which encodes features as it combines denoising, clustering, batch correction, and visualization. Specifically, ACE and GCE [27], another related work that focuses on the model explanation step, both use the latent space embedding to reduce dimensions. Further, in ACE, a concrete autoencoder [1] is added before SAUCIE to select input features, which improves its performance. For the second step — clustering — ACE uses a neuralized k-means algorithm [14] that is connected to the latent embedding layer of SAUCIE. This part of the network takes in the latent embeddings, and learns the positions of the centroids of the data points in the latent space. As for the explanation part, ACE uses a perturbation technique, which draws inspiration from adversarial ML literature [9, 42]. Perturbations are indicative of malicious attacks on learned ML models with the objective to alter the model outcomes. Here, in ACE, the perturbations are used to construct the marker genes, more aptly referred to as gene panels to capture the dependencies between marker genes in the form of a panel of genes with potential dependencies. Although ACE's workflow is more compact and improved the performance, in terms of the AUROC of the SVM classifier, of selected markers genes by ~ 2.8% over its baseline, GCE, the problem of independent optimization (which would be served well by a single optimization function) is not fully realized. This is what motivates us to design KRATOS's joint optimization function for the first two steps of these scRNA-seq integrated analysis pipelines.

Explanation of classification problems: Our work is also related to the explanations of classification problems. Many methods have been proposed to solve this problem. These methods can be roughly categorized into three types: feature attribution methods, counterfactual-based methods, and model-agnostic approximation methods. Feature attribution methods calculate a significance score for each input feature such that a higher score indicates that the prediction is more sensitive to the corresponding feature [21, 34, 35]. SHAPLEY value is a canonical example of this category [33]. Counterfactual-based methods learn an important subregion within the input sample space by inducing alterations in the input samples, such as perturbation, blurring, or inpainting, and then, analyzing the resultant changes in the predictions [10, 11, 38]. Model-agnostic approximation methods approximate the model being explained by using a simpler, surrogate function, affording qualitative mapping between the input and the results of the model [29].

4 Design

4.1 Problem Setup

Assume the input dataset can be expressed as $X = \{x_1, x_2, ..., x_n\}^T \in \mathbb{R}^{n \times p}$ meaning that it contains n cells and p genes as features, with labels $y_1, y_2, ..., y_n$, where $y_i \in C = \{c_1, c_2, ..., c_K\}$ as K classes. Our approach orchestrated by KRATOS, can be framed as a workflow of three steps, with the first two steps merged into one. First, we build a NN to classify the given single-cell RNA matrix. For each input sample x_i, the network will return K activations $a_1, ..., a_K$, showing the probability that the input cell belongs to a certain class at the (final) classification layer. Then, we apply an adversarial perturbation to the input samples and see the change of output, say activations, to learn the significance

score for each gene as the explanation component of KRATOS. The whole workflow is expressed in the Figure 2, and the pseudocode is presented in Algorithm 1.

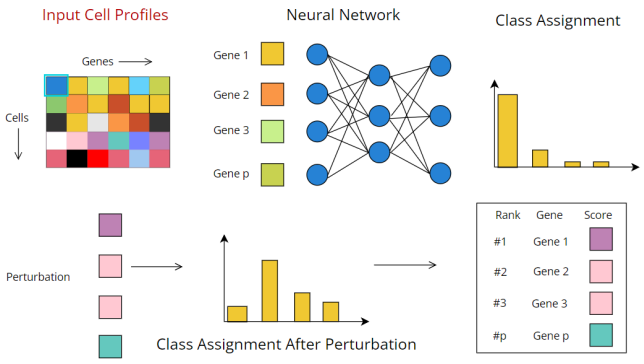


Figure 2: **The architecture of KRATOS's pipeline. The single-cell profile matrix is the input to our NN and the probabilities that a cell belongs to any class will be learned. The NN combines the dimension reduction and clustering steps to optimize them together. With the NN trained, perturbations will be added to the input dataset and sent to the model again to get the updated probability assignment. Then, for each class of cells, we try to find the minimal perturbation, which could lead the NN classifier to output a result, different from the original prediction, and we bound the perturbation within the reasonable range in that useless genes (say "noise" genes) may need infinite perturbation. Finally, the genes will be ranked according to the learned absolute value of perturbation, which translates to the importance score for each gene. Both the neural network and the explanation part can be replaced by other off-the-shelf methods to further refine the pipeline.**

4.2 Classification

As mentioned before, in common sc-RNA-seq analysis workflows, dimension reduction and clustering are conducted separately. The main drawback is that the optimization objectives are not the same and thus the output explanations may not reach their best performance. Our pipeline's final goal is to extract functionally meaningful clusters of cell populations. Further, in many of the other explanation works, the ground truth cell type label is used implicitly to learn the explanations, which means that their workflow is not completely unsupervised. As an improvement, to combine the dimension reduction step and clustering step and make full use of the dataset (with their ground-truth labels), we replace the first two steps with a single, integrated classification neural network. The learning rate is set to 0.001 and the optimizer is Adam [15]. The NN we use for classification is a simple 3-layer fully connected neural network, with two hidden layers and one output layer. The two hidden layers contain 256 and 64 units, respectively, and the output layer contains K units. The activation for each layer is $ReLU$ for first two layers, and $softmax$, for the last layer. According to the definition of softmax, the last layer's output can also be viewed as probabilities that the cell belongs to any type:

$$a_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e_j^z} \tag{10}$$

Here z_i denotes the unit value before the activation. The loss function we are using is binary classification cross entropy:

$$L = -\frac{1}{n}\sum_{i=1}^{n} y_i * log(a_i) + (1 - y_i) * log(1 - a_i) \tag{11}$$

Here $y_i = 1$ if x_i belongs to type i, else $y_i = 0$. In fact, the second layer can be viewed as the low-dimension embedding layer of the three-step workflow, while the third layer is the clustering step.

4.3 Explaining the groups

The second part of the workflow is to learn a significance score for each gene in each cell type. The genes can then be ranked according to the assigned score to show their importance in a certain cell type. Our method is similar to the one in [14]. We calculate the explanations in a one-versus-rest setting, which means that we will compare the probability that a cell type being identified by the NN as class i to the probability that it is identified as any other cell type. We consider this problem as finding a perturbation added to the input cell and lead the classifier to assign a different cell type to the input cell. We denote the induced perturbation as $\delta \in I\!R^p$. The objective function of one-vs-rest setting is:

$$\min_{\delta} \|\delta\|_1 + \lambda \max(0, \alpha + a_i(x+\delta) - \max_{j \neq i} a_j(x+\delta)) \tag{12}$$

Here we assume x belongs to class i according to the classifier, λ is a trade-off coefficient. A smaller λ would decrease the impact of perturbation to the class assignment, while a larger one would encourage the perturbation to a different class. α is a margin coefficient that controls the second term. The second term penalizes the situation that the classifier still assigns x to class i, given the perturbation, up to the margin α. The L1-norm of the perturbation is added to improve the sparsity of the perturbation, since we hope that only top-effective genes are extracted. We can also do a one-vs-one setting. In this case, assume we perturb samples from c_i to c_j. The objective function can be expressed as:

$$\min_{\delta} \|\delta\|_1 + \lambda \max(0, \alpha + a_i(x+\delta) - a_j(x+\delta)) \tag{13}$$

Here the second term penalizes the situation that the classifier assigns a large score to c_i instead of c_j. In other words, it is encouraging the perturbation to alter the input from c_i to c_k.
Finally, the perturbation will be calculated through the optimization over Equation 12 & 13. We quantify the explanation of the i-th gene in a class as the absolute value of i-th term of the perturbation in that class. The larger the score is, the more importance the gene has in that class. Accordingly, the genes will be ranked based on their explanation values.

4.4 Evaluation Strategy

We evaluate KRATOS against 3 baselines on 4 datasets: a simulated dataset, human PBMC dataset [44] and Baron dataset [6], as two real sc-RNA-seq datasets, and the MNIST dataset.
A simulated dataset was generated with known relations among the genes. We then compare the top-ranked genes selected by different explanation pipelines. Good explanations should extract a majority of causal or dependent genes, while less of the "noise" genes.
On the sc-RNA-seq dataset, each algorithm outputs a significance score for each gene in each cell type as explanations. The genes are then ranked by their scores. For each cluster, we use a subset that contains only the top-ranked genes (from top 1% to 100%) to train a SVM classifier with a RBF kernel that separates that cluster from the others. The AUROC of the SVM was calculated and used to demonstrate the discriminative power of the selected genes. A higher AUROC means that the subset of top genes are more representative of the cell type under consideration. We calculated the

Algorithm 1 The description of our system KRATOS's workflow.

Output: For each class c_k, a significance score to each gene, $s_1, s_2, ..., s_p$
Input: Single-cell RNA profile matrix, $X = \{x_1, x_2, ..., x_n\}^T$
Parameters: $epochs$, λ, max_iter, $learning\_rate$, $margin$

0 : Preprocessing: Normalize the dataset. Split the dataset into train set and test set with a ratio of 0.7 : 0.3
1 : Train the classifier NN with $epochs = 100, learningrate = 0.001$, 0.1 of the dataset set will be used as Validation set
2 : For class k, input the X to the trained network, and record the subset for which the classifier makes the correct prediction as $Set_1, ..., Set_K$
3 : For each Set_k, induce a small perturbation $\delta = 0.001$ to each gene and input the perturbed Set_k to the classifier network
4 : Calculate the objective function from Equation 3 and its gradient g w.r.t. each gene
6 : Update the perturbation δ according to
$\delta = \delta - learning\_rate * (signs + \lambda * g)$, where $signs = sgn\delta$
7 : Repeat *Step 3* to *Step 6* for max_iter times to get the δ for c_k as Class k's explanation
8 : Repeat *Step 2* to *Step 7* to get explanations for each class

PCC of the top genes in each class to assess the redundancy, as an auxiliary measure to AUROC. A lower PCC relates to lower redundancy among the genes in the top-ranked genes. Besides, we computed the Silhouette Score, AMI, and ARI as clustering metrics and plot the visualizations of original dataset and latent embeddings output by each algorithm using UMAP and t-SNE, again as two auxiliary measures to visualize the clusterings, especially because these techniques tend to be sensitive to initialization conditions. On the MNIST dataset, we use a one-vs-one setting. For each number, we calculate its discriminatory features relative to other numbers. We plot the values of explanation to show how KRATOS extracts the key differences between different numbers.

4.5 Further design considerations

The classifier in KRATOS is currently a simple neural network. We can replace it with more specialized network architectures, guided by the task or domain. For the explanation part, apart from the adversarial techniques we use, we can also incorporate other off-the-shelf classification explanation algorithms like Smoothgrad, Integrated Gradients, and so on, as we have done in our assessment to compare the two kinds of explanation techniques in the context of our target tasks.

5 Implementation

5.1 Datasets: Real and Simulated

We first use SymSim [43] to generate a simulated sc-RNA-seq dataset, which contains 500 cells and 140 genes, with 5 cell types, and check how many of the marker genes identified by KRATOS are real causal genes. After that, we applied our method to the PBMC dataset [44], which contains 2,638 cells and 1,838 genes, and represented this input as a cell-by-gene log-normalized expression matrix. The cells in the dataset are annotated in eight cell types based on differentially expressed marker genes. Then, we use the Baron dataset [6], another sc-RNA-seq dataset, with 1,886 cells and 14,878 genes, and 13 cell types. The Baron dataset is more challenging to interpret than the PBMC dataset due to the following reasons: 1) It contains far more genes (14,878 genes), ~ 8 times the number of genes than in the PBMC dataset; 2) It is more unbalanced, where 5 out of 13 cell types contain less than 15 samples (less than 0.1% of all genes), while PBMC only has 2 out of 8 unbalanced cell types. Finally, to test the generalization of our results across domains, we evaluated KRATOS using the MNIST database [16], where handwritten digits have been size-normalized and centered in a fixed-size image.

5.2 Baseline methods

To compare KRATOS with the ACE pipeline for evaluating the clustering step. Then, we select three methodologically different explanation methods, which can generate the ranking of genes as pertains to their discriminative power in distinguishing each cell type. First, we adapt the explanation part of the ACE pipeline, which is based on the CW attack-based method [9, 42]. This technique finds the minimal perturbation to alter the group assignment. This variant of the pipeline with the CW attack-method as the explanatory step serves as the main variant of our KRATOS, which we call KRATOS _cw to distinguish from the other variant, as in Figure 5a. Then, we use KRATOS's two other variants — KRATOS _sg, with SmoothGrad [36]; and KRATOS _ig, with Integrated Gradient [37] — as representative feature attribution methods, to compute the importance score for each gene's contribution to the classification result. SmoothGrad differs from Integrated Gradients in that SmoothGrad directly takes gradients as contributions while Integrated Gradients is an extension of SHAPLEY value method in deep learning.

6 Evaluation

6.1 Simulated dataset

To evaluate the performance of KRATOS, we used the simulated sc-RNA-seq dataset, which enables us to specify the causal genes with ground truth, and check how many marker genes identified by KRATOS are real causal genes. To generate the simulated sc-RNA-seq dataset, we use the SymSim (Synthetic model of multiple variability factors for Simulation) simulator that explicitly models the data generating processes observed in sc-RNA-seq experiments [43], also used by [19, 27] for their evaluation. The SymSim pipeline explicitly captures three main sources of variation that regulate single cell expression patterns: noise intrinsic to the process of transcription, extrinsic variation, and technical variation due to low sensitivity and measurement noise or bias, further providing control knobs to vary these types of parameters. This simulation tool generates the basic cell-gene data matrix with 500 cells, 2000 genes, and 5 cell types. We then filter out the causal genes, following SymSim's criteria ($nDiff - EVFgene > 0 and |log2fold - change| > 0$) between at least one pair of cell types, and select 20 causal genes with highest fold-change among all pairs of cell types. We next simulated 20 dependent genes as the linear summation of 1-10 random causal genes selected in the previous steps, with added Gaussian noise from $N(0, 1)$, and weights of summation follows the uniform distribution of $U(0.01, 0.8)$. Finally, the remaining 100 noise genes are randomly selected. Thus, our simulated dataset consists of 140 genes in total. We then applied KRATOS and ACE to this simulated dataset, and compared the identified marker genes

with causal genes for each cell type. We expect that the marker genes, with the highest importance score, of all cell types should be completely overlapped with the causal genes. Thus, we check the top-20 marker genes for each cell type, and count how many causal genes are identified as the marker genes. As the Table 1 describes, KRATOS performs well on the simulated dataset, where almost half of the marker genes identified by KRATOS are causal genes and the others are dependent genes, while only one noise gene is identified as the marker gene. However, Table 1 also shows that while half of the marker genes identified by ACE are real causal genes, almost quarter of the marker genes (5 in average) are noise genes. Both KRATOS and ACE could successfully identify almost half of causal genes within top-20 marker genes. However, ACE identifies much higher number of noise genes, which is more deleterious than the higher number of dependent genes identified by KRATOS. Note, also, that the higher number of dependent genes in KRATOS is at par with the higher redundancy observed in KRATOS relative to ACE, as shown in the Figure 4. Table 1 suggests that KRATOS outperforms ACE in identifying the relevant genes, validated by ground truth.

Table 1: **The distribution of marker genes using KRATOS and ACE. Here "C", "D", and "N" stand for the number of causal genes, dependent genes, and noise genes, respectively, among the top-20 identified marker genes. We see that KRATOS can extract comparable causal genes as ACE, but the noise genes chosen by KRATOS is far lower than ACE (for the most part, 0, vs ~ 5 for SOTA).**

Clust ID	C_{Kratos}	D_{Kratos}	N_{Kratos}	C_{ACE}	D_{ACE}	N_{ACE}
1	8	12	0	9	7	4
2	8	11	1	8	6	6
3	10	10	0	10	5	5
4	10	10	0	8	7	5
5	9	11	0	9	5	6

6.2 Performance on the human PBMC dataset

Evaluation of KRATOS on the PBMC dataset includes: visualization of the embedding and importance scores for marker genes.

First, we focus on the visualization of the embedding generated by KRATOS and the ACE pipeline using the PBMC dataset. In Figure 3a, 3b & 3c, we show the UMAP [7, 25] for the original cell-gene dataset, and the embedding layer, in KRATOS and ACE, respectively. We observe that the cells of the same type tend to cluster more tightly in the embedding layer of KRATOS, superior relative to the clustering in ACE. Also, the t-SNE [39] in Figure 3d, 3e & 3f shows a similar pattern of clustering as visualized using UMAP, with KRATOS outperforming ACE. Moreover, we calculate the Silhouette Coefficients, ARI, and AMI, to measure the goodness-of-clustering. In Table 2, the ARI & AMI of KRATOS can be observed to be larger than those of ACE & GCE, indicating that clusters are best separated in KRATOS's embedding. Further, the Silhouette Coefficient of KRATOS is larger than ACE, which is 0.37 and 0.07 for KRATOS and ACE, respectively, meaning 53% superior for KRATOS. Notice that GCE has the highest Silhouette Score meaning that it has the best structures for clusters. Nevertheless, we consider AMI sa the first priority here as it is comparing to the ground truth labels and is less affected by the imbalance of the clusters. To wrap up, these results indicate that KRATOS outshines ACE and GCE in creating a good low-dimension embedding and best cluster assignment.

We next applied the differentiation analysis part in ACE toward identifying top-ranked genes for different cell types based on the results of KRATOS and of the ACE and GCE pipeline. We also replace

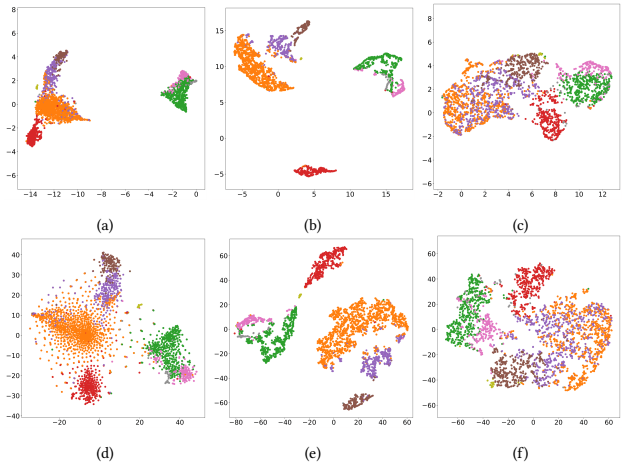


Figure 3: **Visualization of the embedding layer in KRATOS & ACE using UMAP and t-SNE for the human PBMC dataset. (a), (b), and (c) are the UMAP embeddings, using default parameter settings. (d), (e), and (f) are the t-SNE embeddings using default settings, except $perplexity = 30$. (a) and (d) are embeddings of the original input dataset, others are for latent space embeddings. We see that the different classes are more separated from each other with the low-dimension representation. KRATOS outperforms on the cell clustering performance, where cells of the same type tend to cluster more tightly. UMAP outperforms t-SNE with more compact clusters for different classes, possibly reflecting a more accurate global structure.**

Table 2: **Silhouette coefficients, ARI, and AMI for different clustering methods. Note that the ARI & AMI of KRATOS are much larger than those of ACE & GCE, also, the Silhouette Coefficients of KRATOS are larger than ACE. The result indicates that KRATOS shines in its clustering performance.**

Clustering Methods	Silhouette	ARI	AMI
KRATOS	0.3745	0.8495	0.8547
ACE	0.0762	0.5215	0.6219
GCE	0.5083	0.7619	0.6989

the explanation part with the two gradient-based methods. For each cell type, the panels of top-k genes with the highest significance scores are identified by the explanation methods, where k ranges from 1% to 100%.

It is desirable that the top-k genes show minimum redundancy, which is captured using the PCC between all gene pairs among top-k genes. We calculate the PCC within each cell type, then compute the group-averaged PCC for each k. Figure 4a, 4b , & 4c plots the PCC among top-k genes generated by KRATOS and ACE, where k ranges from 1% to 40%, computed using the Carlini Wagner (cw) algorithm, Integrated Gradients (ig), or Smooth Grad (sg) as the explanation part respectively. Figure 4d shows that GCE has higher redundancy among the top-k genes than KRATOS. As discussed earlier in the simulation part of the evaluation, the higher redundancy is at par with the higher number of dependent genes seen in KRATOS relative to ACE. Thus, this property is less harmful to the clustering objective than the AUROC metric, as seen in Figure 5, in all its comparisons with ACE.

We then trained an SVM, within each given cell type and top-k genes, to evaluate the discriminative power of top-k genes for distinguishing from the remaining groups. Our SVM classifier uses the radial bias function (RBF) kernel, with two hyperparameters — the regularization coefficient C and the bandwidth parameter σ. We implement the 3-fold stratified cross-validation to evaluate the performance, in terms of AUROC, of SVM classifier, and random

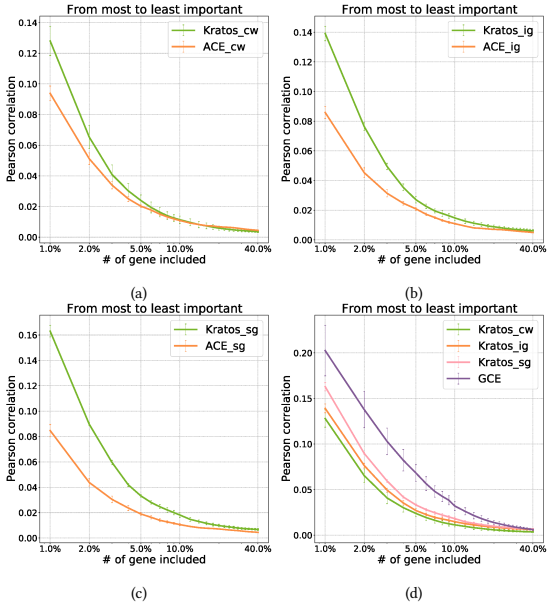


Figure 4: **Redundancy plot. Redundancy among the top k genes in each group, as measured by PCC, as a function of k. Error bars correspond to the standard deviation from the mean, calculated from the group-specific correlations. A lower PCC implies lower redundancy among the top genes, extracted by the explanations. We see that when the selected number of genes is small, ACE has a lower redundancy than our work, while GCE has a higher redundancy. However, according to the results from the simulated dataset, the low correlation of ACE might be because it is selecting noise genes as opposed to marker genes, while KRATOS is selecting a greater number of dependent genes, indicating a more balanced performance of KRATOS.**

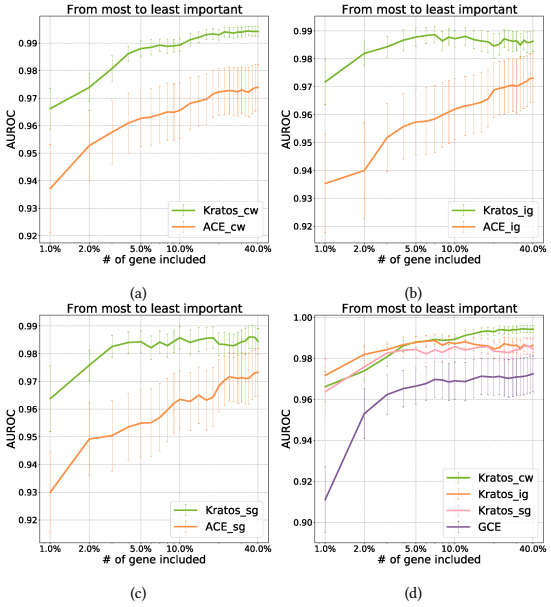


Figure 5: **Discriminative power plot. The plot compares the classification performance using different methods. We calculated the AUROC of a binary SVM classifier using only the top genes as features, across each cell type. A higher AUROC corresponds to a better discriminative power of the selected marker genes. From this plot, we see that KRATOS surpasses other baselines especially when the number of included genes is small. Specifically, the accuracy of KRATOS, on average, is 3.31% superior to ACE, and 5.62% superior to GCE within the top-1% marker genes.**

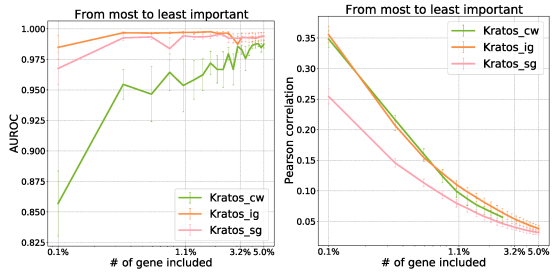
search [8] with 3-fold cross-validation to optimize two hyperparameters within each training set, where the C and σ were assumed to be exponentially distributed, respectively, with scaling factors of 100 and 0.1. We reported the average performance, in terms of area under the receiver operating characteristic curve (AUROC), among different target cell types. Specifically, we excluded two cell types, megakaryocytes and dendritic cells, from our SVM evaluation part due to their relatively small sample size (17 and 36 correspondingly), while we still include them in training KRATOS. In Figure 5a, 5b, & 5c, we plot the classification performance based on KRATOS and the ACE pipeline, and we found that KRATOS outperforms the ACE pipeline for all three variants using the three explanation methods, on average, 3.31% superior to the accuracy of ACE, and ranging from 2.91% (for CW attack-based methods) to 3.64% (for the Integrated Gradients variant), within top-1% marker genes. Also, we compare GCE to KRATOS, combined with the different explanation methods, and the Figure 5d indicates that KRATOS outperforms, 5.62% on average, relative to GCE within top-1% marker genes.

6.3 Performance on the human pancreas dataset

To further evaluate the performance of KRATOS, we applied KRATOS to the Baron [6], a more complex cell-gene dataset with a higher number of genes. We replicate the method for the identification for marker genes as in the previous part, evaluating KRATOS combined with different explanation methods. As the Figure 6a shows, even with much higher dimensionality, KRATOS can still achieve quite good performance with any explanation method (higher than 95% accuracy for the top 1% genes). Moreover, KRATOS achieves decent performance (from 86% to 98%) with even top-1‰marker genes, around 13 genes, in Baron dataset. Also, the superior performance of KRATOS combined with Integrated Gradients (98% accuracy with only 1‰genes) suggests that the Integrated Gradients variant of KRATOS may be the most suitable variant for this dataset, a testimony to KRATOS's ability to be used as a plug-and-play integrated scRNA-seq analysis framework, based on the domain-specific nuances. It is noticeable that CW does not perform as well here. This may be due to the imbalance in the dataset that leads to a biased model or the lack of tuning on the CW hyperparameters, as the tradeoff coefficient λ and margin α can significantly affect the results. In addition, we plot the PCC among the identified marker genes in Figure 6b, where KRATOS combined with SmoothGrad generates the minimum correlation among identified marker genes within any quantile of genes.

6.4 MNIST dataset

We tested the potential of KRATOS to be applied to other domains to evaluate its generalization using the MNIST dataset [16], where handwritten digits have already been size-normalized in [0, 1]. Similar to the identification of marker genes in the sc-RNA-seq data analysis task, our task was to find out the key pixels that could explain the correct classification of handwritten digits. However, the key pixels could be different for the same digit, based on the different handwriting styles, while the marker genes are applicable for all cells in the dataset. We reshape the MNIST data from the matrix $28 * 28$ into the array $784 * 1$ to make the input of the same form as in sc-RNA-seq data, though some of the information may have been lost during the transformation. We use the same NN



(a) (b)

Figure 6: **Discriminative power and redundancy plot. The classification performance and redundancy of KRATOS with different explanation methods. We find that KRATOS performs well (from 95% to 99% accuracy) on the Baron dataset within 1‰of genes, while the redundancy is similar to the one in the PBMC dataset (all range from 10% to 15%). Moreover, KRATOS achieves decent performance (from 86% to 98%) even within 1‰of genes.**

architecture and perturbation analysis. Only, the one-versus-rest loss function in perturbation analysis is replaced with the one-versus-one loss function so that we can measure the importance of each pixel and identify the key pixels during the transformation from a certain digit to another digit. To plot the results more clearly, we only plot the pixels with top-25 & bottom-25 importance scores, while other pixels' importance scores are replaced as 0. Further, the importance scores are normalized within each image. The Figure 7 shows the key pixels for 12 pairs of digit transformations, and we find that most of the identified pixels are reasonable. For example, when we transform "3" to "8", the digit has to add pixels to the left side of "3", while removing pixels to the left side of "8" when transforming "8" to "3". Also, the key pixels for the transformation between "0" and "8" are the center pixels of "8".

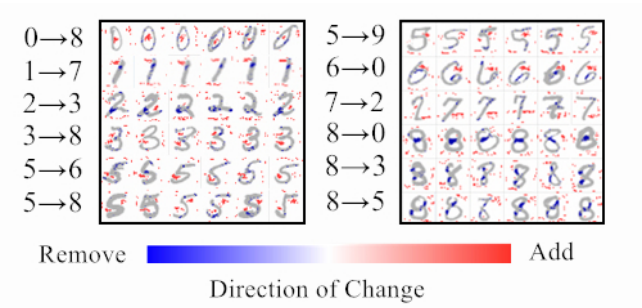


Figure 7: **The key pixels for the digits' transformations. We show the key pixels for 12 pairs of digits transformations, where the backgrounds in grey indicate the original digits. The pixels in blue (red) indicate that these pixels should be removed (added) to convert to the target digit. Most of the pixels in blue and red are consistent with our expectation.**

7 Discussion

In this paper, we proposed a single-cell RNA analysis pipeline, KRATOS, which is based on neural network and ML interpretability. The most elegant part of KRATOS is the combination of the first two steps — dimensionality reduction and clustering — within one neural network classifier instead of conducting each step separately. This joint optimization of the two steps significantly improves the performance of the ML explanation part. Our experiments on real and simulated single-cell RNA datasets demonstrate that KRATOS's selected top genes have the best discriminative power for clustering cell types. Beyond genomics, we also demonstrated that KRATOS can be used in computer vision tasks such as image recognition using the MNIST dataset. Our approach here of using modular pipelines (of three blocks here) derives inspiration from our prior work in building and using reusable blocks or "kernels" of code in our Domain Specific Language (DSL) Sarvavid [24], specifically designed for computational genomics. Sarvavid allows users to naturally express their genomics application using the DSL kernels. Sarvavid expedites development of new computational genomics tools, such as in ScalaDBG, to swiftly assemble high-quality complex genomes [23]. Similarly, here our hope is that KRATOS will enable modular developments improving the accuracy and scalability of single-cell genome annotation algorithms.

There are several ways to extend this work. First, KRATOS's first two parts can be replaced by other specialized neural network classifiers. Modified CW optimization in the perturbation step can make it less sensitive to cluster imbalance. In addition, better modeling of sc-RNA-seq data to perform improved batch correction will also benefit KRATOS. Finally, merging the explanation step to the neural network architecture representing the merged first two steps would create an integrated trinity of steps for sc-RNA-seq data analyses.

Acknowledgments This material is based in part upon work supported by the National Science Foundation (NSF) under Grant Number CNS-2146449 (NSF CAREER award) and by the National Institutes of Health (NIH) Grant R01AI123037. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] Abubakar Abid, Muhammad Fatih Balin, and James Zou. 2019. Concrete autoencoders for differentiable feature selection and reconstruction. *arXiv preprint arXiv:1901.09346* (2019).
[2] Matthew Amodio, David Van Dijk, Krishnan Srinivasan, William S Chen, Hussein Mohsen, Kevin R Moon, Allison Campbell, Yujiao Zhao, Xiaomei Wang, Manjunatha Venkataswamy, et al. 2019. Exploring single-cell data with deep multitasking neural networks. *Nature methods* 16, 11 (2019), 1139–1145.
[3] Tallulah S Andrews, Vladimir Yu Kiselev, Davis McCarthy, and Martin Hemberg. 2021. Tutorial: guidelines for the computational analysis of single-cell RNA sequencing data. *Nature protocols* 16, 1 (2021), 1–9.
[4] Ricard Argelaguet, Britta Velten, Damien Arnol, Sascha Dietrich, Thorsten Zenz, John C Marioni, Florian Buettner, Wolfgang Huber, and Oliver Stegle. 2018. Multi-Omics Factor Analysis—a framework for unsupervised integration of multi-omics data sets. *Molecular systems biology* 14, 6 (2018), e8124.
[5] Robert J Aumann and Lloyd S Shapley. 1974. *Values of non-atomic games.* Princeton University Press.
[6] Maayan Baron, Adrian Veres, Samuel L Wolock, Aubrey L Faust, Renaud Gaujoux, Amedeo Vetere, Jennifer Hyoje Ryu, Bridget K Wagner, Shai S Shen-Orr, Allon M Klein, et al. 2016. A single-cell transcriptomic map of the human and mouse pancreas reveals inter-and intra-cell population structure. *Cell systems* 3, 4 (2016), 346–360.
[7] Etienne Becht, Leland McInnes, John Healy, Charles-Antoine Dutertre, Immanuel WH Kwok, Lai Guan Ng, Florent Ginhoux, and Evan W Newell. 2019. Dimensionality reduction for visualizing single-cell data using UMAP. *Nature biotechnology* 37, 1 (2019), 38–44.
[8] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research* 13, 2 (2012).
[9] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp).* IEEE, 39–57.
[10] Chun-Hao Chang, Elliot Creager, Anna Goldenberg, and David Duvenaud. 2018. Explaining image classifiers by counterfactual generation. *arXiv preprint arXiv:1807.08024* (2018).
[11] Ruth C Fong and Andrea Vedaldi. 2017. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE international conference*

on computer vision. 3429–3437.
[12] Eric J Friedman. 2004. Paths and consistency in additive cost sharing. *International Journal of Game Theory* 32, 4 (2004), 501–518.
[13] L GUILLAUME. 2008. Fast unfolding of communities in large networks. *Journal Statistical Mechanics: Theory and Experiment* 10 (2008), P1008.
[14] Jacob Kauffmann, Malte Esders, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2019. From clustering to cluster explanations via neural networks. *arXiv preprint arXiv:1906.07633* (2019).
[15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[16] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
[17] Jun Li and Robert Tibshirani. 2013. Finding consistent patterns: a nonparametric approach for identifying differential expression in RNA-Seq data. *Statistical methods in medical research* 22, 5 (2013), 519–536.
[18] Michael I Love, Wolfgang Huber, and Simon Anders. 2014. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome biology* 15, 12 (2014), 1–21.
[19] Yang Young Lu, C Yu Timothy, Giancarlo Bonora, and William Stafford Noble. 2021. ACE: Explaining cluster from an adversarial perspective. In *International Conference on Machine Learning*. PMLR, 7156–7167.
[20] Malte D Luecken and Fabian J Theis. 2019. Current best practices in single-cell RNA-seq analysis: a tutorial. *Molecular systems biology* 15, 6 (2019), e8746.
[21] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*. 4768–4777.
[22] James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Vol. 1. Oakland, CA, USA, 281–297.
[23] Kanak Mahadik, Christopher Wright, Milind Kulkarni, Saurabh Bagchi, and Somali Chaterji. 2019. Scalable genome assembly through parallel de Bruijn graph construction for multiple k-mers. *Scientific reports* 9, 1 (2019), 1–15.
[24] Kanak Mahadik, Christopher Wright, Jinyi Zhang, Milind Kulkarni, Saurabh Bagchi, and Somali Chaterji. 2016. Sarvavid: a domain specific language for developing scalable computational genomics applications. In *Proceedings of the 2016 international conference on supercomputing*. 1–12.
[25] Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018).
[26] Hannah A Pliner, Jay Shendure, and Cole Trapnell. 2019. Supervised classification enables rapid annotation of cell atlases. *Nature methods* 16, 10 (2019), 983–986.
[27] Gregory Plumb, Jonathan Terhorst, Sriram Sankararaman, and Ameet Talwalkar. 2020. Explaining groups of points in low-dimensional representations. In *International Conference on Machine Learning*. PMLR, 7762–7771.
[28] David Reich, Alkes L Price, and Nick Patterson. 2008. Principal component analysis of genetic data. *Nature genetics* 40, 5 (2008), 491–492.
[29] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.
[30] Mark D Robinson, Davis J McCarthy, and Gordon K Smyth. 2010. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* 26, 1 (2010), 139–140.
[31] Simone Romano, Nguyen Xuan Vinh, James Bailey, and Karin Verspoor. 2016. Adjusting for chance clustering comparison measures. *The Journal of Machine Learning Research* 17, 1 (2016), 4635–4666.
[32] Anne Senabouth, Samuel W Lukowski, Jose Alquicira Hernandez, Stacey B Andersen, Xin Mei, Quan H Nguyen, and Joseph E Powell. 2019. ascend: R package for analysis of single-cell RNA-seq data. *GigaScience* 8, 8 (2019), giz087.
[33] LS Shapley. 1953. Quota Solutions of n-Person Games. *Edited by Emil Artin and Marston Morse* (1953), 343.
[34] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *International Conference on Machine Learning*. PMLR, 3145–3153.
[35] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013).
[36] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. 2017. SmoothGrad: removing noise by adding noise. In *Proceedings of the ICML Workshop on Visualization for Deep Learning*. 1700–1706.
[37] Tim Stuart and Rahul Satija. 2019. Integrative single-cell analysis. *Nature Reviews Genetics* 20, 5 (2019), 257–272.
[38] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*. PMLR, 3319–3328.
[39] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).
[40] Joe H Ward Jr. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association* 58, 301 (1963), 236–244.
[41] Joshua D Welch, Alexander J Hartemink, and Jan F Prins. 2017. MATCHER: manifold alignment reveals correspondence between single cell transcriptome and epigenome dynamics. *Genome biology* 18, 1 (2017), 1–19.
[42] Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. 2020. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing* 17, 2 (2020), 151–178.
[43] Xiuwei Zhang, Chenling Xu, and Nir Yosef. 2019. Simulating multiple faceted variability in single cell RNA sequencing. *Nature communications* 10, 1 (2019), 1–16.
[44] Grace XY Zheng, Jessica M Terry, Phillip Belgrader, Paul Ryvkin, Zachary W Bent, Ryan Wilson, Solongo B Ziraldo, Tobias D Wheeler, Geoff P McDermott, Junjie Zhu, et al. 2017. Massively parallel digital transcriptional profiling of single cells. *Nature communications* 8, 1 (2017), 1–12.