UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Pavlin G. Poličar

**Dodajanje primerov v referenčno
vložitev t-SNE odstrani razlike med
različnimi podatkovnimi viri**

MAGISTRSKO DELO
MAGISTRSKI PROGRAM DRUGE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: prof. dr. Blaž Zupan

Ljubljana, 2019

UNIVERSITY OF LJUBLJANA
FACULTY OF COMPUTER AND INFORMATION SCIENCE

Pavlin G. Poličar

**Embedding to Reference t-SNE Space
Addresses Batch Effects in Single-Cell
Classification**

MASTER'S THESIS
THE 2ND CYCLE MASTER'S STUDY PROGRAMME
COMPUTER AND INFORMATION SCIENCE

SUPERVISOR: prof. dr. Blaž Zupan

Ljubljana, 2019

Povzetek

Naslov: Dodajanje primerov v referenčno vložitev t-SNE odstrani razlike med različnimi podatkovnimi viri

Tehnike zmanjševanja dimenzij, kot je t-SNE, nam omogočajo gradnjo informativnih vizualizacij visokorazsežnih naborov podatkov. Pri analizi več naborov podatkov hkrati te metode pogosto ne uspejo odkriti pomenljive skupine, temveč izpostavijo nezaželene razlike med podatkovnimi viri. Da bi odstranili vplive posameznih podatkovnih virov in odkrili strukture skupne vsem podatkom, predlagamo teoretično utemeljeno metodo za dodajanje novih primerov v obstoječo vložitev t-SNE. Metodo vključimo v našo odprtokodno implementacijo metode t-SNE in pokažemo na uporabnost predlagane metode na analizi šestih nedavno objavljenih podatkovnih naborov genskih izrazov posameznih celic. Rezultati so presenetljivi; predlagana metoda namreč povsem odstrani vplive različnih podatkovnih virov, ki so eden temeljnih izzivov pri analizi podatkov s področja molekularne biologije. Predlagana tehnika poleg tega tudi omogoča uporabo vnaprej zgrajenih vložitev t-SNE, kar odpira nove možnosti uporabe interpretabilnih vizualizacij visokorazsežnih naborov podatkov.

Ključne besede

razlike med različnimi podatkovnimi viri, vložitev, t-SNE, vizualizacija, transkriptomika posameznih celic, integracija podatkov, domenska adaptacija

Abstract

Title: Embedding to Reference t-SNE Space Addresses Batch Effects in Single-Cell Classification

Dimensionality reduction techniques, such as t-SNE, can construct informative visualizations of high-dimensional data. When working with multiple data sets, a straightforward application of these methods often fails; instead of revealing underlying classes, the resulting visualizations expose data set-specific clusters. To circumvent these batch effects, we propose a principled embedding procedure that enables the addition of new data points into existing t-SNE embeddings. We provide an open-source implementation of the proposed method and demonstrate the utility of our approach with an analysis of six recently published single-cell gene expression data sets containing up to tens of thousands of cells and thousands of genes. We present surprising evidence that our computationally more direct procedure solves the batch effect problem, one of the core challenges in the analysis of gene expression data, and enables the reuse of t-SNE embeddings, paving the way for interpretable visualizations of high-dimensional data sets.

Keywords

batch effects, embedding, t-SNE, visualization, single-cell transcriptomics, data integration, domain adaptation

COPYRIGHT. The results of this master's thesis are the intellectual property of the author and the Faculty of Computer and Information Science, University of Ljubljana. For the publication or exploitation of the master's thesis results, a written consent of the author, the Faculty of Computer and Information Science, and the supervisor is necessary.

©2019 PAVLIN G. POLIČAR

ACKNOWLEDGMENTS

I would like to thank my colleagues from the Bioinformatics Laboratory at the University of Ljubljana, especially dr. Martin Stražar and prof. dr. Blaž Zupan, without whom this work would not have been possible. I would also like to thank George Linderman and Dmitry Kobak for many helpful discussions on t-SNE. Finally, I would like to thank my friends and family for all their continuous support throughout the years.

Pavlin G. Poličar, 2019

Contents

Povzetek

Abstract

Razširjeni povzetek	i
I Sorodna dela	ii
II Predlagane metode	iv
III Preizkusi in rezultati	v
IV Sklep	vii
 1 Introduction	 1
 2 Related Work	 7
2.1 Scalability	10
2.2 Data Set Bias	14
 3 Methods and Implementation	 17
3.1 Performance Tricks	17
3.2 Adding Data to a Reference Embedding	22
3.3 Fast Mapping to an Embedding Space	24
3.4 openTSNE	25
3.5 Integration into Orange	30
 4 Data and Experimental Setup	 35
4.1 Data	36

CONTENTS

4.2	Single-cell data pipeline	38
4.3	Gene selection	38
5	Results and Discussion	41
6	Collection of Reference Embeddings	49
6.1	Construction of reference embeddings	50
6.2	Reference Embedding for Mouse Retina	51
6.3	Reference Embedding for Human Pancreas	53
6.4	Reference Embedding for Mouse Brain	55
7	Conclusion	57

Razširjeni povzetek

Pri analizi visoko-razsežnih naborov podatkov si večkrat pomagamo z različnimi projekcijami oz. vložitvami v dvodimenzionalni prostor. Intuitivno želimo, da sta si dve točki v vizualizaciji blizu, če sta si blizu tudi v visoko-razsežnem prostoru. V ta namen je bilo razvitih več metod, med katerimi je tudi t-porazdeljena stohastična vložitev sosedov (*ang. t-distributed stochastic neighbor embedding*, tj. t-SNE) [1]. Te metode so tipično zelo uspešne, ko imamo opravka s podatki, ki izvirajo iz enega samega vira, vendar lahko naletijo na težave, ko podatki izvirajo iz različnih naborov podatkov. V tem primeru te metode izpostavijo razlike med različnimi podatkovnimi viri, namesto da bi odkrile pomenljive skupine, ki bi bile skupne vsem podatkom. Te sistematske razlike se na področju bioinformatike imenujejo *skupinske razlike* (*ang. batch effects*) [2, 3, 4, 5].

Na področju molekularne biologije predstavljajo skupinske razlike eno ključnih ovir pri sočasni analizi podatkov iz različnih poskusov. Slika 1.1 prikazuje dve vložitvi podatkov genskih izrazov posameznih celic, kjer smo v vsakem primeru združili dve podatkovni zbirki in direktno uporabili metodo t-SNE. Kot je razvidno iz obeh slik, je metoda znotraj podatkov našla skupine specifične za posamezno podatkovno zbirko. To je v nasprotju z našim pričakovanjem, saj vemo, da se v obeh naborih podatkov nahajajo celice istega celičnega tipa. Pričakovali bi, da nam metoda odkrije različne celične skupine, ne glede na izvor podatkov.

V nalogi smo razvili razširitev metode t-SNE, ki omogoča dodajanje novih primerov v obstoječe vložitve in hkrati tudi odstrani sistematske skupinske

razlike med viroma podatkov. Naš pristop obravnava en nabor podatkov kot referenčno množico, nad katero zgradimo vložitev t-SNE. V to vložitev nato vstavljamo primere iz druge, sekundarne množice, kjer vsak primer obravnavamo neodvisno od drugih. Na ta način odstranimo modeliranje interakcij med točkami v sekundarni množici in tako odstranimo skupinske razlike. Za primer vzemimo podatkovna nabora genskih izrazov, kjer smo v sliki 1.2.a vizualizirali nabor celic vidne možganske skorje [6] za pripravo referenčne vložitve, nato pa na sliki 1.2.b v to vizualizacijo dodali celice hipotalamusa [7]. Za razliko od slike 1.1.a, kjer smo podatkovna nabora enostavno združili, so na sliki 1.2.b celice istega tipa razvrščajo v istovrstne skupine referenčnih podatkov.

V prvem delu magistrske naloge predstavimo metodo t-SNE in njene nedavne izboljšave. Nato izpeljemo lastne izboljšave namenjene učinkovitejši implementaciji algoritma in predlagamo teoretično utemeljeno razširitev algoritma, ki omogoča dodajanje novih primerov v obstoječe vložitve. Naše razširitve in izboljšave implementiramo v hitri, odprtokodni knjižnici v programskem jeziku Python. Knjižnica omogoča uporabo metode t-SNE na velikih naborih podatkov, ki lahko vsebujejo več milijonov primerov. V drugem delu naloge uporabimo predlagano metodo na šestih naborih podatkov s področja molekularne biologije iz raziskav posameznih celic. Rezultati presenetljivo pokažejo, da naš pristop povsem odpravi nezaželene sistematske skupinske razlike med podatki.

I Sorodna dela

Metoda t-SNE je nelinearna tehnika zmanjševanja dimenzij, ki se osredotoča na ohranitev lokalnih struktur v podatkih [1]. Metoda se je uveljavila na področju analize genskih izrazov, saj se veliko nedavnih študij prične z vizualizacijo t-SNE, kjer so prikazani različni tipi analiziranih celic [8, 9, 10]. Kljub široki uporabi so večkrat izpostavljene njene slabosti, ki vključujejo slabo ohranitev globalne organizacije odkritih skupin, počasnost pri obdelavi

večjih naborov podatkov in pomanjkanje teoretično utemeljenega pristopa za dodajanje primerov v obstoječe vložitve [11, 12].

V prizadevanju odpravljanja pomanjklivosti t-SNE, sta Kobak in Berens [13] predlagala več trikov, ki bolje ohranjajo globalno organizacijo skupin v končni vizualizaciji. V delu avtorjev Linderman *et al.* [14] je bila predlagana nova aproksimacijska shema, ki zniža časovno kompleksnost metode iz kvadratne na linearno v odvisnosti od števila primerov.

V zadnjih letih se je pojavilo tudi veliko raziskav na področju sočasne analize več naborov podatkov. Tovrstne analize so tipično otežene zaradi sistematskih razlik znotraj podatkov. Na področju analize slik se na primer te razlike lahko pokažejo kot različne osvetlitve istih subjektov [15] ali pa različne teme besedil na področju obdelave naravnega jezika [16].

Problem razlik med nabori podatkov se ponavadi rešuje neposredno: podatke najprej obdelamo z izbrano metodo, ki odstrani skupinske razlike. Področje *domenske adaptacije* (*ang. domain adaptation*) [16] uporablja samokodirnike (*ang. autoencoders*) z večopravilnostnimi ciljnim funkcijami (*multi-task learning*), ki tipično spodbujajo robustne predstavitve podatkov, tj. predstavitve, kjer ni moč ugotoviti, iz katerega nabora podatkov prihaja posamezni primer [17, 18, 19, 20].

Na področju analiz podatkov iz molekularne biologije sta se uveljavili dve veji metod. Prva veja vsebuje metode, ki popravijo podatke tako, da nadaljnji modeli ne ločijo med referenčnimi in novimi podatki. Nekaj primerov tovrstnih metod so metoda ujemanja sosedov (*ang. mutual nearest neighbor matching*) [3], ter iskanje sidrnih točk (*ang. anchor-based mutual nearest neighbor matching*) [4]. Druga veja metod skuša najti skupen, nižjerazsežen prostor, kjer so razlike med nabori podatkov izničene. Slednje vključujejo kanonično korelacijsko analizo (*ang. canonical correlation analysis*) [2] in *scanorama* [5].

II Predlagane metode

Predlagamo teoretično utemeljeno razširitev metode t-SNE, ki omogoča dodajanje novih primerov v obstoječe vložitve. Za začetne koordinate novih točk najprej poiščemo deset njim najbližjih točk v referenčni množici in uporabimo mediano njihovih koordinat. Podobno kot v navadnem t-SNE-ju optimiziramo koordinate novih točk z gradientnim sestopom. Vsaka nova točka je vstavljena v obstoječi prostor, neodvisno od drugih novih točk. Tako onemogočimo vsakršne interakcije med novimi točkami, in jih prisilimo, da se poravnajo zgolj s pomočjo referenčnih točk. Časovna kompleksnost našega pristopa je $\mathcal{O}(N \cdot M)$, kjer je N število točk v referenčni podatkovni množici in M število točk v novi podatkovni množici. S prilagoditvijo aproksimacijske sheme Linderman *et al.* [14] zmanjšamo časovno kompleksnost našega pristopa na $\mathcal{O}(\max\{N, M\})$.

Pri vložitvi novih primerov moramo biti posebej pazljivi, da zmanjšamo učni korak (*ang. learning rate*), saj je privzeta vrednost pri večini implementacij ($\eta = 200$) previsoka in povzroči, da so nekatere točke “izstreljene” proč od ostalih točk. Priporočamo uporabo učnega koraka $\eta \sim 0.1$, za katerega smo ugotovili, da ustvari dobre vizualizacije. To je še posebej pomembno pri uporabi aproksimacijske sheme, saj je časovna kompleksnost primarno odvisna od velikosti prostora, ki ga razpenjajo točke v vložitvi.

Izpeljemo tudi tri manjše izboljšave metode t-SNE, ki omogočajo bolj učinkovite programske implementacije, in vključujejo učinkovitejšo računanje ciljne funkcije t-SNE in modela soseščin, ter popravek ciljne funkcije v prvih fazah optimizacije.

Pri obdelavi visokorazsežnih podatkov priporočamo, da se pred vsakršno analizo naredi izbor informativnih značilk, saj je v takšnih podatkih ponavadi veliko število odvečnih atributov. Priporočamo tudi, da se pred analizo visokorazsežnih podatkov izlušči nekaj vodilnih komponent (*ang. principal components*), saj na tak način zmanjšamo šum v podatkih in pospešimo poznejše metode.

Predlagane pristope smo implementirali v odprtokodni knjižnico `openTSNE`,

ki nudi vse izboljšave, ki jih omenjamo v tem delu. Knjižnica je združljiva z obstoječimi knjižnicami za programski jezik Python in omogoča enostavno integracijo s priljubljenimi paketi, namenjeni strojnemu učenju, kot so **numpy**, **sklearn**, **scanpy**. **openTSNE** je prva knjižnica, ki omogoča dodajanje novih primerov v obstoječe vložitve. Knjižnica je implementirana z učinkovito aproksimacijsko shemo, kar omogoča njeno uporabo na podatkovnih naborih z velikim številom primerov. Za primer vzemimo sliko 3.1, kjer smo s knjižnico **openTSNE** vložili 2.058.652 točk v 56 minutah. Knjižnica je prosto dostopna na GitHubu, kjer je nabrala že čez 500 GitHub zvezdic.

Našo odprtokodno knjižnico smo vključili tudi v programsko okolje Orange, namenjeno interaktivni raziskavi podatkov z uporabo delokrogov. Delokroge sestavljajo preprosti gradniki, skozi katere lahko sestavimo raznovrstne analize. Razvili smo nov gradnik **t-SNE**, ki vsebuje vse omenjene izboljšave.

III Preizkusi in rezultati

Da bi ovrednotili koristnost naše metode, izpeljemo analizo treh parov podatkovnih naborov genskih izrazov posameznih celic. V naših preizkusih izberemo en nabor podatkov za referenco, v katerega nato dodamo celice drugega podatkovnega nabora. Naš cilj je zgraditi vložitev obeh podatkovnih naborov, tako da odstranimo razlike med viroma tj., skupine, ki jih odkrijemo naj odražajo biološko smiselne enote, kot so različni tipi celic.

V naših preizkusih uporabimo tri pare podatkovnih naborov, kjer vsak par izvira iz podobnih tkiv. Uporabimo celice mišje retine, ki izhajajo iz študij Macosko *et al.* [8] ter Shekhar *et al.* [9], mišjih možganov, kjer celice iz podatkov Hrvatin *et al.* [6] izhajajo iz vidne možganske skorje, ter iz hipotalamusa [7]. V tretjem paru izberemo dva nabora podatkov iz človeške trebušne slinavke [10, 21], kjer se študiji osredotočita na raziskavo sladkorne bolezni.

V naših preizkusih izvedemo tudi predobdelavo podatkov, ki je standardna za analizo genskih izrazov posameznih celic. Vsak nabor podatkov naj-

prej normaliziramo tako, da je vsota vseh vseh genskih izrazov za vsako celico enaka. Nato uporabimo specializirano metodo za izbor informativnih genov, kjer izberemo 3.000 genov. Preostale gene zavržemo. Nato z metodo vodilnih komponent zmanjšamo prostor na 50 dimenzij. Pridobljeno predstavitev nato vložimo v dve dimenziji z uporabo metode t-SNE. Pri vložitvi uporabimo inicializacijo s koordinatami dveh vodilnih komponent. Ker imamo opravka z visokorazsežnim prostorom zamenjamo privzeto uporabljeno evklidsko razdaljo s kosinusno. Vložitev novih primerov poteka na podoben način. Najprej izberemo 1.000 genov prisotnih v obeh naborih podatkov in te uporabimo za oceno podobnosti. Znova uporabimo kosinusno razdaljo, saj imamo opravka z visokorazsežnim prostorom.

Slike 1.2, 5.1 in 5.2 prikazujejo pridobljene vložitve. V vseh slikah se točke iz drugega, novega nabora podatkov, ujemajo s skupinami iz referenčne vložitve, kar je močan dokaz o uspešnosti naše metode. V slikah je zaznati nekaj napak, vendar so te redke in v večini primerov smiselne. Na sliki 1.2 je nekaj rjavih celic postavljenih na modro skupino; rjave celice predstavljajo celice predhodnice oligodendrocitov, modre pa oligodendrocite. Napaka v postavitvi je morda zgolj posledica nekonsistentnih anotacij avtorjev študij, saj podatki izhajajo iz dveh različnih laboratorijev, ki sta morda imela različne kriterije pri določanju celičnega tipa.

V naših preizkusih uporabimo tudi t. i. *multiscale t-SNE*, kar omogoča boljšo ohranitev globalne organizacije odkritih skupin. Razlika med navadnim t-SNE-jem in multiscale t-SNE-jem je prikazana na sliki 5.3. Na sliki 5.3.a je bil uporabljen navaden t-SNE z naključno začetno postavitvijo točk. Iz slike je razvidno, da se različne skupine istovrstnih celic razpršijo čez celoten prostor, medtem ko so na sliki 5.3.b skupine istih barv postavljene v iste regije prostora.

Predlagana metoda predpostavlja, da so vse skupine iz nove podatkovne množice prisotne tudi v referenčni množici, sicer so lahko primeri teh skupin naključno postavljeni v prostoru. To pomankljivost lahko rešujemo z gradnjo t. i. atlasov – celovitih zbirk podatkov posameznih tkiv, ki vsebujejo vse

znane tipe celic. Zgradili smo tri takšne referenčne vložitve t-SNE za mišjo retino (slika 6.1), človeško trebušno slinavko (slika 6.2) in mišje možgane (slika 6.3). Te referenčne vložitve omogočajo raziskovalcem hitro in interpretabilno anotacijo lastnih eksperimentov, kar se trenutno počne lastnoročno in vsakič znova.

IV Sklep

Predstavili smo nov, teoretično utemeljen pristop za dodajanje novih primerov v obstoječe vložitve metode t-SNE. Metodo uporabimo na različnih naborih podatkov genskih izrazov, kjer sistematske razlike med podatki predstavljajo temeljni izziv pri primerjalni analizi podatkov. Rezultati kažejo na presenetljiv zaključek, da naša metoda povsem odstrani razlike med nabori, kar omogoča uporabo celovitih, vnaprej pripravljenih vložitev za hitro anotacijo in interpretacijo novih bioloških eksperimentov. Pripravimo tudi tri takšne referenčne vložitve za tri različna človeška in mišja tkiva, ki so pogosto predmet raziskav.

Predlagano metodo in zadnje izboljšave objavimo v odprtokodno Python knjižnico `openTSNE`, ki je prosto dostopna na portalu GitHub. Našo implementacijo vključimo tudi v programsko okolje Orange, ki omogoča hitro, interaktivno raziskavo podatkov. V naši skupini razvijamo tudi nove metode za avtomatsko anotacijo celic, ki temeljijo na tehnikah, predstavljene v tem delu. Slika 7.1 tako prikazuje nov gradnik v programskem okolju Orange, ki uporabi opisane metode za gradnjo referenčne vložitve, nato pa vstavi nove primere v obstoječo vizualizacijo. Gradnik avtomatsko ugotovi tipe celic in jih primerno označi na vizualizaciji. Naše delo spodbuja gradnjo in večkratno rabo celovitih, referenčnih vložitev, kar omogoča avtomatsko anotacijo primerov in hkrati nudi njeno interpretacijo.

Chapter 1

Introduction

Two-dimensional embeddings and their visualizations are often used to assist the analysis and interpretation of high-dimensional data. Intuitively, two data instances should be co-located in the resulting visualization if their multi-dimensional profiles are similar. For this task, non-linear embedding techniques such as t-distributed stochastic neighbor embedding (t-SNE) [1] or uniform manifold approximation and projection [22] have recently complemented traditional data transformation and embedding approaches such as principal component analysis (PCA) and multi-dimensional scaling [23, 12].

While useful for visualizing data from a single coherent source, two-dimensional embeddings may encounter problems if the data comes from multiple sources. When performing dimensionality reduction on a merged data set, the resulting visualizations typically reveal source-specific clusters instead of grouping data instances of the same class-type, regardless of data source. This source-specific confounding is often referred to as *domain shift* [24], *covariate shift* [25] or *dataset shift* [26]. In bioinformatics, domain-specific differences are more commonly referred to as *batch effects* [2, 3, 4, 5].

Massive, multi-variate biological data sets often suffer from these source-specific biases. Consider an example from single-cell genomics, the domain we will focus on in this manuscript, which was selected due to high biomedical relevance and abundance of recently published data. Single-cell RNA

sequencing (scRNA-seq) data sets are the result of isolating RNA molecules from individual cells, which serve as an estimate of the expression of each cell’s genes. Single-cell studies, which can range from thousands to millions of individual cells and contain tens of thousands of genes, typically start with the analysis of cell types. It is generally expected that cells of the same type would cluster together in two-dimensional data visualization [4]. For instance, Fig. 1.1.a shows t-SNE embedded data from mouse brain cells originating from the visual cortex [6] and the hypothalamus [7]. The figure reveals several distinct clusters but also separates the data from the two brain regions. These two regions share the same cell types and, contrary to the depiction in Fig. 1.1.a, we would expect the data points from the two studies to overlap. Batch effects similarly limit the utility of t-SNE in the exploration of pancreatic cells in Fig. 1.1.b, which renders the data from a pancreatic cell atlas [10] and similarly-typed cells from diabetic patients [21]. Just like with data from brain cells, pancreatic cells cluster primarily by data source, again resulting in an uninformative visualization primarily driven by batch effects.

Current solutions to embedding the data from various data sources address the batch effect problems up-front. The data is typically preprocessed and transformed such that the batch effects are explicitly removed. In the single-cell literature, t-SNE plots are typically used to evaluate the effectiveness of a batch-effect removal method [3, 2, 5]. If the resulting t-SNE visualization of multiple data sets exhibits good mixing, then the method is deemed to be effective. The elimination of batch effects may require aggressive data preprocessing which may blur the boundaries between cell types and may also introduce unwanted method-specific biases into the data. Another problem is the inclusion of new data, for which the entire data analysis pipeline must be rerun, usually resulting in a different embedding layout and clusters that have little resemblance to original visualization and thus require reinterpretation.

We propose a direct solution of rendering t-SNE visualizations that ad-

Figure 1.1: Batch effects are a driving factor of variation between the data sets. We depict a t-SNE visualization of two pairs of data sets. In each pair, the data sets share cell types, so it would be expected that the cells from the reference data (blue) would mix with the cells in a secondary data sets (orange). Instead, t-SNE visualization clusters data according to the data source.

dresses batch effects. Our approach treats one of the data sets as a *reference* and aims to embed the samples from another, *secondary data set* to a reference-defined, low-dimensional space. We construct a t-SNE embedding using the reference data set, and then use it as a scaffold for the embedding of data points from the secondary data. The key idea underpinning our approach is that each data point is embedded independently of one another. This causes the clustering landscape to depend only on the reference scaffold, thus removing data source-driven variation. In other words, when including new data, the scaffold inferred from the reference data set is kept unchanged and defines a “gravitational field”, independently driving the embedding of each new instance. For example, in Fig. 1.2, the cells from the visual cortex define the scaffold (Fig. 1.2.a) into which we embed the cells from the hypothalamus (Fig. 1.2.b). Unlike in their joint t-SNE visualization (Fig. 1.1.a), the hypothalamic cells are dispersed across the entire embedding space and their cell type correctly matches the prevailing type in the reference clusters. While this work will focus primarily on single-cell data sets, the proposed methods are general and are widely-applicable to any domain.

Our proposed solution defines a mapping of new data points into an existing t-SNE visualization. While the utility of such an algorithm for single-cell data was already hinted at by Kobak and Berens [13], we here provide its practical and theoretically-grounded implementation. Considering the abundance of recent publications on batch effect removal, we present surprising evidence that a computationally more direct and principled embedding procedure solves the batch effect problem when constructing interpretable visualizations from different data sources.

In the first part of the manuscript, we review the t-SNE algorithm and its recent improvements, introducing the relevant notation where necessary. We reformulate the original t-SNE equations and propose three improvements, which enable more efficient implementations. Then we develop a theoretically-grounded approach to add new data points into existing t-SNE embeddings. We apply our proposed solution to three pairs of single-cell

Figure 1.2: A two-dimensional embedding of a reference containing cells from the visual cortex **(a)** and the corresponding mapping of secondary data containing hypothalamic cells **(b)**. Notice that the majority of hypothalamic cells were mapped to their corresponding reference cluster. For instance, astrocyte cells marked with red on the right were mapped to an oval cluster of same-typed cells denoted with the same color in the visualization on the left.

RNA-seq data sets, where batch effects pose a significant challenge. Surprisingly, our simple and direct approach completely removes batch effects, enabling straightforward, visual classification of cell-types from newly sequenced data sets, paving the way for interpretable visualizations. Finally, we discuss the drawbacks of our approach and offer a possible solution.

Chapter 2

Related Work

From a plethora of approaches, such as principal component analysis, multidimensional scaling, and uniform manifold approximation and projection [22], t-SNE has received much attention as it can address high volumes of data and reveal clustering structure. For instance, many recent prominent reports on single-cell gene expression data start with an overview of the cell landscape, where t-SNE is used to embed high-dimensional expression profiles into a two-dimensional space [8, 9, 10].

Embedding by t-SNE is a local, non-linear dimensionality reduction method, tailored to the visualization of high-dimensional data sets [1]. Given a multi-dimensional data set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \in \mathbb{R}^D$ where N is the number of samples in the reference data set, t-SNE aims to find a low-dimensional embedding $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N\} \in \mathbb{R}^d$ where $d \ll D$, such that if points $\mathbf{x}_i$ and $\mathbf{x}_j$ are close in the multi-dimensional space, their corresponding embeddings $\mathbf{y}_i$ and $\mathbf{y}_j$ are also close. Since t-SNE is primarily used as a visualization tool, d is typically set to two. The similarity between two data points in t-SNE is defined as

$$p_{j|i} = \frac{\exp\left(-\frac{1}{2}\mathcal{D}(\mathbf{x}_i, \mathbf{x}_j)/\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\frac{1}{2}\mathcal{D}(\mathbf{x}_i, \mathbf{x}_k)/\sigma_i^2\right)}, \quad p_{i|i} = 0 \quad (2.1)$$

where $\mathcal{D}$ is some distance measure. This is then symmetrized to

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}. \quad (2.2)$$

The bandwidth of each Gaussian kernel σ_i is selected such that the perplexity of the distribution matches a user-specified parameter value

$$\text{Perplexity} = 2^{H(P_i)} \quad (2.3)$$

where $H(P_i)$ is the Shannon entropy of P_i ,

$$H(P_i) = - \sum_j p_{j|i} \log_2(p_{j|i}). \quad (2.4)$$

Different bandwidths σ_i enable t-SNE to adapt to the varying density of the data in the multi-dimensional space.

The similarity between points $\mathbf{y}_i$ and $\mathbf{y}_j$ in the embedding space is defined using the t -distribution with one degree of freedom

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}, \quad q_{ii} = 0. \quad (2.5)$$

The t-SNE method finds an embedding $\mathbf{Y}$ that minimizes the Kullback-Leibler (KL) divergence between $\mathbf{P}$ and $\mathbf{Q}$,

$$C = \text{KL}(\mathbf{P} \parallel \mathbf{Q}) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}}, \quad (2.6)$$

where the gradients of the objective function C are

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) (\mathbf{y}_i - \mathbf{y}_j) (1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}. \quad (2.7)$$

The resulting embeddings depend heavily on the value of the perplexity parameter. Perplexity can loosely be interpreted as the number of neighbors for which the distances in the embedding space are preserved. Small values of perplexity uncover small, tightly-packed clusters of points and effectively ignore the long-range interactions between clusters. Larger values may result

in more globally consistent visualizations, preserving distances on a large scale and organizing clusters in a more meaningful way. Larger values of perplexity may lead to merging of multiple small clusters, thus obscuring local aspects of the data [13].

Despite its widespread adoption, the t-SNE algorithm suffers from several shortcomings including the lack of global cluster organization in resulting visualizations, poor scalability and the lack of principled methods that enable the addition of new samples into existing embeddings [11, 12]. Recent developments have addressed the problems of global consistency and scalability, while the addition of new data points remains largely unsolved.

To address the lack of global cluster alignment in resulting visualizations, Kobak and Berens [13] proposed several techniques, including estimating similarities with mixtures of Gaussian kernels and various initialization strategies. Using a mixture of Gaussian kernels, also referred to as multiscale similarity estimation, allows us to preserve the long-range interactions between points, while still emphasizing the local structure of the clusters. This leads to better global consistency of the cluster landscape but does not lead to the merging of smaller clusters. The final t-SNE embedding is heavily dependent on the choice of initialization, so Kobak and Berens [13] suggest initializations using dimensionality reduction methods that preserve the global structure. These include PCA, multidimensional scaling, and t-SNE with large perplexity. The standard t-SNE algorithm is then run to reveal local aspects of the data.

Embedding by t-SNE is a non-parametric method and does not provide a direct way to include new samples into an existing embedding. Van der Maaten proposed a parametric mapping based on neural networks using the t-SNE objective function [27]. Unfortunately, this approach relies on slow gradient estimation methods, making it unsuitable for even moderately-sized data sets and introduces many additional parameters related to the network architecture. Adding new samples to a prebuilt embedding largely remains an unsolved problem, and no current popular software library supports adding

new samples into an existing embedding.

2.1 Scalability

Evaluating point similarities in Eqs. 2.1 and 2.5 requires computing the pairwise distances between all data points in the original space and the embedding space, requiring $\mathcal{O}(N^2)$ time. The gradient can be interpreted as the interplay between attractive forces between neighboring points and repulsive forces between all data points. The placement of points forms a kind of “gravitational field”. We show this by decomposing the gradient expression from Eq. 2.7 into the attractive and repulsive forces [28]:

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) (\mathbf{y}_i - \mathbf{y}_j) (1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1} \quad (2.8)$$

$$= 4 \sum_{j \neq i} (p_{ij} - q_{ij}) (\mathbf{y}_i - \mathbf{y}_j) (1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1} \frac{Z}{Z} \quad (2.9)$$

where Z is the normalization term of $\mathbf{Q}$, $Z = \sum_{k \neq l} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}$

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) q_{ij} Z (\mathbf{y}_i - \mathbf{y}_j) \quad (2.10)$$

$$= 4 \left(\sum_{j \neq i} p_{ij} q_{ij} Z (\mathbf{y}_i - \mathbf{y}_j) - \sum_{j \neq i} q_{ij}^2 Z (\mathbf{y}_i - \mathbf{y}_j) \right), \quad (2.11)$$

which can, in turn, be interpreted as the attractive and repulsive forces between data points

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 (\mathbf{F}_{\text{attr}} + \mathbf{F}_{\text{rep}}). \quad (2.12)$$

Since the p_{ij} terms in the attractive forces are obtained through a Gaussian kernel with exponentially decaying tails, many of the values of P_i are near-zero and do not contribute to the overall gradient [28]. Because the bandwidth of each kernel is determined by the user-specified perplexity parameter, it is sufficient to find a fixed number of nearest neighbors for each

data point, and only evaluate p_{ij} at those points. Finding the k nearest neighbors is a well-studied topic and can be done exactly in $\mathcal{O}(N \log N)$ time using vantage-point trees [29]. Linderman *et al.* [14] recently observed that using the approximate nearest neighbors yields visually indistinguishable results, further reducing the time complexity to $\mathcal{O}(N)$.

Linderman *et al.* [14] also proposed a novel interpolation-based approximation of t-SNE gradients, reducing the time complexity needed to evaluate the repulsive forces between data points to be only linearly dependent on the number of data items. The core idea of their approach is to introduce a set of interpolation points spanning the embedding space and compute the repulsive forces between these in place of the actual data points. These new points can then be used to interpolate the repulsive forces back onto the original data points in the embedding space. Because interpolation can be performed in linear time, the bulk of the computational burden lies in computing the repulsive forces between the interpolation points. Note that we have full control of these and if we select the number of interpolation points to be much lower than the number of actual data points, the resulting algorithm may be orders of magnitude faster than the direct evaluation of repulsive forces.

Concretely, the repulsive forces from Eq. 2.12 can be rewritten into the form

$$\mathbf{F}_{\text{rep}} = \sum_{j \neq i} q_{ij}^2 Z(\mathbf{y}_i - \mathbf{y}_j) \quad (2.13)$$

$$= \sum_{j \neq i} \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-2}}{\sum_{k \neq l} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-2}} \frac{(\mathbf{y}_i - \mathbf{y}_j)}{\sum_{k \neq l} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)} \quad (2.14)$$

$$= \left(\sum_{j \neq i} \frac{\mathbf{y}_i - \mathbf{y}_j}{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^2} \right) / \left(\sum_{k \neq l} \frac{1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2}{(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^2} \right) \quad (2.15)$$

$$= \left(\sum_{j \neq i} \frac{\mathbf{y}_i - \mathbf{y}_j}{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^2} \right) / \left(\sum_{k \neq l} \frac{1}{(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)} \right). \quad (2.16)$$

We can also write an expression for each component of $\mathbf{y}_i$ individually

$$\mathbf{F}_{\text{rep},i}(m) = \left(\sum_{j \neq i} \frac{\mathbf{y}_i(m) - \mathbf{y}_j(m)}{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^2} \right) / \left(\sum_{k \neq l} \frac{1}{(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)} \right) \quad (2.17)$$

where $\mathbf{y}_i(m)$ denotes the m^{th} component of $\mathbf{y}$, with $m \in \{1, 2\}$ in the 2-dimensional case.

We observe that the repulsive forces $\mathbf{F}_{\text{rep}}$ can be written as $s + 2$ sums of the form where s is the dimensionality of $\mathbf{Y}$,

$$\phi(\mathbf{y}_i) = \sum_j K(\mathbf{y}_i, \mathbf{y}_j) q_{ij} \quad (2.18)$$

and where $K(\mathbf{y}, \mathbf{z})$ is either the Cauchy kernel or the squared Cauchy kernel

$$K_1(\mathbf{y}, \mathbf{z}) = \frac{1}{(1 + \|\mathbf{y} - \mathbf{z}\|^2)}, \quad \text{or} \quad K_2(\mathbf{y}, \mathbf{z}) = \frac{1}{(1 + \|\mathbf{y} - \mathbf{z}\|^2)^2}. \quad (2.19)$$

To illustrate this point, consider the 2-dimensional case. If we define four new terms ϕ_{1-4} , which all incorporate the kernels from Eq. 2.19

$$\begin{aligned} \phi_{1,i} &= \sum_{j \neq i} \frac{1}{(1 + \|\mathbf{y}_j - \mathbf{y}_i\|^2)} \\ \phi_{2,i} &= \sum_{j \neq i} \frac{\mathbf{y}_j(1)}{(1 + \|\mathbf{y}_j - \mathbf{y}_i\|^2)^2} \\ \phi_{3,i} &= \sum_{j \neq i} \frac{\mathbf{y}_j(2)}{(1 + \|\mathbf{y}_j - \mathbf{y}_i\|^2)^2} \\ \phi_{4,i} &= \sum_{j \neq i} \frac{1}{(1 + \|\mathbf{y}_j - \mathbf{y}_i\|^2)^2}, \end{aligned} \quad (2.20)$$

then the repulsive forces from Eq. 2.17 can be expressed as

$$F_{\text{rep},i}(1) = \left(\sum_{j \neq i} \frac{\mathbf{y}_i(1) - \mathbf{y}_j(1)}{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^2} \right) / \left(\sum_{k \neq l} \frac{1}{(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)} \right)$$

$$= (\phi_{2,i} - \mathbf{y}_i(1)\phi_{4,i})/Z, \quad (2.21)$$

$$F_{\text{rep},i}(2) = \left(\sum_{j \neq i} \frac{\mathbf{y}_i(2) - \mathbf{y}_j(2)}{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^2} \right) / \left(\sum_{k \neq l} \frac{1}{(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)} \right) \\ = (\phi_{3,i} - \mathbf{y}_i(2)\phi_{4,i})/Z, \quad (2.22)$$

where

$$Z = \sum_j \phi_{1,j}. \quad (2.23)$$

Given that both kernels K_1 and K_2 are smooth kernels, these can then be approximated using polynomial interpolation. Consider a single kernel which computes the interaction between two data points y and z . Suppose that $\tilde{z}_i \dots \tilde{z}_p$ and $\tilde{y}_i \dots \tilde{y}_p$ are both collections of p interpolation points. Let $K_p(y, z)$ denote a polynomial interpolant of a kernel $K(y, z)$. Linderman *et al.* [14] show that

$$K_p(y, z) = \sum_{\ell=1}^p \sum_{j=1}^p K(\tilde{y}_j, \tilde{z}_\ell) L_{j,\tilde{y}}(y) L_{\ell,\tilde{z}}(z) \quad (2.24)$$

where $L_{j,\tilde{y}}$ and $L_{\ell,\tilde{z}}$ are the Lagrange polynomials

$$L_{\ell,\tilde{y}}(y) = \prod_{\substack{j=1 \\ j \neq \ell}}^p (y - \tilde{y}_j) / \prod_{j=1}^p (\tilde{y}_\ell - \tilde{y}_j) \quad \text{and} \quad (2.25)$$

$$L_{\ell,\tilde{z}}(z) = \prod_{\substack{j=1 \\ j \neq \ell}}^p (z - \tilde{z}_j) / \prod_{j=1}^p (\tilde{z}_\ell - \tilde{z}_j). \quad (2.26)$$

By combining Eqs. 2.18 and 2.24 we obtain an approximation to ϕ

$$\tilde{\phi}(y_i) = \sum_{j=1}^N \sum_{\ell=1}^p \sum_{m=1}^p K(\tilde{y}_\ell, \tilde{z}_m) L_{\ell,\tilde{y}}(y_i) L_{m,\tilde{z}}(z_j) q_j, \quad (2.27)$$

which can, in turn, be rearranged into

$$\tilde{\phi}(y_i) = \sum_{\ell=1}^p L_{\ell, \tilde{y}}(y_i) \left(\sum_{m=1}^p K(\tilde{y}_\ell, \tilde{z}_m) \left(\sum_{j=1}^N L_{m, \tilde{z}}(z_j) q_j \right) \right). \quad (2.28)$$

Notice that Eq. 2.28 is a general formulation where we approximate the interaction between two different sets of points $\mathbf{Z}$ and $\mathbf{Y}$, each requiring its own set of interpolation points. In t-SNE, the quantity of interest is the interaction between all pairs of points in $\mathbf{Y}$, requiring a single set of interpolation points. Thus Eq. 2.28 simplifies to

$$\tilde{\phi}(y_i) = \sum_{\ell=1}^p L_{\ell, \tilde{y}}(y_i) \left(\sum_{m=1}^p K(\tilde{y}_\ell, \tilde{y}_m) \left(\sum_{j=1}^N L_{m, \tilde{y}}(y_j) q_j \right) \right). \quad (2.29)$$

Each of the terms in parentheses can be computed once and cached during computation, requiring a single pass over the data, reducing the time complexity to $\mathcal{O}(N)$. These are then used to obtain the repulsive forces in Eqs. 2.21 and 2.22. For a full description of the method as well as several proofs, refer to Linderman *et al.* [14].

2.2 Data Set Bias

With the availability of large data sets such as *ImageNet* [30], *The spoken Wikipedia dataset* [31] and *The Amazon reviews data set* [32] and considering the remarkable results achieved on these vast data collections, many challenges, where the amount of data was difficult to obtain or expensive to gather, were improved through transfer learning [33]. Transfer learning assumes that models trained on large volumes on data will also perform well when applied to a related task. However, it quickly became apparent that subtle differences between data sets *e.g.*, different lighting conditions in image analytics tasks [15] or different topics in natural language processing challenges [16] negatively impact the predictive performance on any such models and that additional steps must be taken to achieve good results When

performing classification or regression tasks using neural networks, a simple but powerful approach to this problem is to retrain the final layers of the network for the task at hand while keeping the earlier layers of the network fixed. This is commonly referred to as *fine-tuning* [34].

Another, more general approach to address these data set-specific biases is through the use of domain adaptation [16]. Domain adaptation has received much attention in recent years as it is typically not dependent on a specific class of models but instead aims to modify new data in a way that it appears to come from the same source as the data the model was originally trained on. Domain adaptation can be achieved through various methods, but most approaches rely on multi-task learning where additional penalties are introduced to the loss function, which aim to maximize domain invariance [17, 18, 19, 20].

Data set-specific biases are also common in computational biology and pose a significant challenge when performing comparative studies between experiments. In the bioinformatics literature, these biases are most commonly referred to as *batch effects*. In RNA-sequencing experiments, batch effects can be the result of many different factors, such as different sequencing protocols, different isolation strategies, different room temperature, or even different air pressure resulting from different weather conditions, all of which affect the transcriptional activity of individual cells [35]. Unfortunately, as shown in Fig. 1.1, batch effects are one of the driving sources of variation, and several approaches have been developed to alleviate this source-specific confounding.

Batch effects are typically addressed either through batch effect removal, where the original data matrices are modified in such a way that the domain-specific biases are removed, or through data integration, where multiple data sets are projected into a joint, lower-dimensional space such that batch effects are removed [5]. A number of approaches have emerged for both families of techniques. These include, most notably, mutual nearest neighbor matching [3] and the use of anchors [4] for batch effect removal, and canonical

correlation analysis [2] and scanorama [5] for data integration.

Chapter 3

Methods and Implementation

We extend the t-SNE algorithm to enable the embedding of new data points to a reference embedding and provide a comprehensive Python t-SNE library, containing the latest developments in as well as our own improvements and extensions. We describe a procedure which enables t-SNE to be applied to multiple data sets, originating from multiple sources while mitigating batch effects.

3.1 Performance Tricks

We reformulate the original t-SNE equations and present three improvements such that, from an implementation standpoint, evaluating t-SNE gradients and error is computationally more efficient.

3.1.1 Fast Kullback-Leibler Divergence

The objective function used in t-SNE is the KL divergence between two probability distributions. The matrix $\mathbf{P}$ models similarities between data points in the original, high-dimensional space, while $\mathbf{Q}$ models the distances between points in the embedding space. These matrices are made to be as similar as possible using the KL divergence, which is a measure of similarity of two probability distributions.

The KL divergence is defined as

$$\text{KL}(\mathbf{P} \parallel \mathbf{Q}) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \quad (3.1)$$

In most implementations the true, normalized values of q_{ij} are only available after one pass over the data set has been completed because the normalization term for q_{ij} is not known in advance and all similarities q_{ij} must be evaluated at least once. To compute the KL divergence, we would need two passes over the data, first to find the normalization factor for all q_{ij} s, and a second pass to normalize the q_{ij} s and evaluate the error.

We can modify this expression such that the KL divergence can be computed in a single pass over the data, by using the unnormalized $\hat{q}_{ij}$ values.

$$\text{KL}(\mathbf{P} \parallel \mathbf{Q}) = \sum_{ij} p_{ij} \log \left(p_{ij} \frac{Z}{\hat{q}_{ij}} \right), \quad (3.2)$$

where $\hat{q}_{ij}$ denotes the unnormalized values q_{ij} and Z denotes the normalization term for $\mathbf{Q}$, i.e., the sum of all q_{ij} values

$$\text{KL}(\mathbf{P} \parallel \mathbf{Q}) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{\hat{q}_{ij}} + \sum_{ij} p_{ij} \log Z. \quad (3.3)$$

Since Z is independent of i and j and $\sum_{ij} p_{ij} = 1$, we obtain

$$\text{KL}(\mathbf{P} \parallel \mathbf{Q}) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{\hat{q}_{ij}} + \log Z. \quad (3.4)$$

Therefore, we can compute the first term in a single pass over the data and accumulate $Z = \sum_{ij} q_{ij}$, which can be added to the error at the end of the pass.

3.1.2 Determining Kernel Bandwidths

Embedding by t-SNE models similarities in the original, high-dimensional space as probability distributions over neighboring data points, assigning

high values to points close to one another and low probabilities to points far away from each other. These distributions are constructed by placing a Gaussian kernel over each data point and evaluating the kernel density at each of the points' neighbors. These values are then normalized to sum to one to form a probability distribution. Each data point is assigned a Gaussian kernel with different bandwidths, which are selected such that the perplexity of the distribution over neighboring points matches a user-specified value. The perplexity parameter in t-SNE can loosely be interpreted as the number of points to which distances are preserved.

The perplexity of the discrete probability distribution over neighboring points P_i is defined as

$$\text{Perplexity}(P_i) = 2^{H(P_i)} \quad (3.5)$$

where $H(P_i)$ is the Shannon entropy of P_i ,

$$H(P_i) = - \sum_j p_{j|i} \log_2(p_{j|i}). \quad (3.6)$$

In most implementations, the bandwidths σ_i are found using iterative binary search and it is more practical to avoid the exponentiation at each step and precompute the $\log(\text{Perplexity}(P_i))$ once and perform binary search using the entropy.

Remember that P_i is governed by a Gaussian kernel centred on point i , given by

$$p_{j|i} = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{d_{ij}^2}{2\sigma_i^2}\right) \quad (3.7)$$

where d_{ij} is the distance between points i and j . However, since P_i is renormalized once at the end of the computation, it is sufficient to skip the normalization terms of the kernel, resulting in

$$p_{j|i} \propto \exp\left(-\frac{d_{ij}^2}{2\sigma_i^2}\right). \quad (3.8)$$

To further reduce the number of operations and speed up computation, most implementations reparameterize Eq. 3.8 using $\beta_i = 1/2\sigma_i^2$ and compute $\exp(-d_{ij}^2\beta_i)$. While this formulation is perfectly valid when using a single Gaussian kernel, Kobak and Berens [13] propose using a mixture of Gaussian kernels e.g.

$$p_{j|i} \propto \frac{1}{\sigma_{1,i}} \exp\left(\frac{d_{ij}^2}{2\sigma_{1,i}^2}\right) + \frac{1}{\sigma_{2,i}} \exp\left(\frac{d_{ij}^2}{2\sigma_{2,i}^2}\right). \quad (3.9)$$

Clearly, we can no longer fully ignore the normalization term as in Eq. 3.8 since all $\sigma_{*,i}$ depend on the perplexity value. We propose an alternative parametrization using the precision of the Gaussian kernel $\tau_i = 1/\sigma_i^2$. Therefore, our probability density for each kernel is given by

$$p_{j|i} \propto \sqrt{\tau_i} \exp\left(-\frac{d_{ij}^2\tau_i}{2}\right). \quad (3.10)$$

Using this parametrization, the entropy of each kernel centered at point i is defined as

$$H_i = - \sum_j \frac{\sqrt{\tau_i} \exp(-d_{ij}^2\tau_i/2)}{\sum_k \sqrt{\tau_i} \exp(-d_{ik}^2\tau_i/2)} \log\left(\frac{\sqrt{\tau_i} \exp(-d_{ij}^2\tau_i/2)}{\sum_k \sqrt{\tau_i} \exp(-d_{ik}^2\tau_i/2)}\right) \quad (3.11)$$

Notice that the first term is just $p_{j|i}$ and we can split the terms in the logarithm, obtaining

$$H_i = - \sum_j p_{j|i} \left[\log(\sqrt{\tau_i} \exp(-d_{ij}^2\tau_i/2)) - \log\left(\sum_k \sqrt{\tau_i} \exp(-d_{ik}^2\tau_i/2)\right) \right]. \quad (3.12)$$

We denote the normalization factor as Z and simplify

$$H_i = - \sum_j \left[p_{j|i} \left(\frac{1}{2} \log \tau_i - d_{ij}^2\tau_i/2 \right) \right] + \sum_j p_{j|i} \log Z \quad (3.13)$$

$$= -\frac{1}{2} \log \tau_i \sum_j p_{j|i} + \frac{\tau_i}{2} \sum_j p_{j|i} d_{ij}^2 + \sum_j p_{j|i} \log Z. \quad (3.14)$$

We notice that $p_{j|i}$ in the first term is a probability distribution so its elements sum up to 1. We can rearrange the terms to end up with

$$H_i = \frac{\tau_i}{2} \sum_j p_{j|i} d_{ij}^2 + \log Z - \frac{1}{2} \log \tau_i. \quad (3.15)$$

While one might argue that it is possible to obtain the kernel bandwidths $\sigma_{*,i}$ using the same formulation as before, by using our formulation it is possible to determine both the kernel bandwidths and compute the exact values of P_i at the same time, reducing the number of operations needed to find the final multiscale probability distributions.

3.1.3 Kullback-Leibler Divergence Using Exaggeration

During the t-SNE optimization phase, most implementations report the KL divergence, which can be used to determine whether the embedding has reached a stable solution. The optimization phase consists of two stages: the early exaggeration phase, followed by the regular optimization phase. The early exaggeration phase increases the attractive forces between neighboring points, enabling points to move around more easily through the embedding space to form their respective clusters. In the regular optimization phase, the attractive forces are restored to their true values, such that the true t-SNE objective is optimized.

In most implementations, the early exaggeration phase is achieved by multiplying the affinity matrix $\mathbf{P}$ by some user-specified scalar value α . The standard t-SNE implementation [36] uses early exaggeration with $\alpha = 12$. Scaling $\mathbf{P}$ by some factor α means that $\mathbf{P}$ is no longer a valid probability distribution and the reported KL divergence no longer takes on the correct value of the error.

We derive a correction for the KL divergence, such that the true error can be reported even during the early exaggeration phase when $\mathbf{P}$ is scaled by α .

Since $\mathbf{P}$ is scaled by a constant factor α , we must find an expression for

$$\text{KL}(\mathbf{P} \parallel \mathbf{Q}) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}} \quad (3.16)$$

such that all p_{ij} s are replaced by αp_{ij} .

$$\text{KL}(\mathbf{P} \parallel \mathbf{Q}) = \sum_{ij} \frac{\alpha}{\alpha} p_{ij} \log \frac{\alpha p_{ij}}{\alpha q_{ij}} \quad (3.17)$$

$$= \frac{1}{\alpha} \sum_{ij} \alpha p_{ij} \left(\log \frac{\alpha p_{ij}}{q_{ij}} - \log \alpha \right) \quad (3.18)$$

$$= \frac{1}{\alpha} \left(\sum_{ij} \alpha p_{ij} \log \frac{\alpha p_{ij}}{q_{ij}} \right) - \frac{1}{\alpha} \left(\sum_{ij} \alpha p_{ij} \log \alpha \right) \quad (3.19)$$

Notice that the first term is exactly the KL divergence where p_{ij} s are scaled by α . Notice also in the second term that $\sum_{ij} P_{ij} = 1$ and that α cancels out, leaving us with

$$\text{KL}(\mathbf{P} \parallel \mathbf{Q}) = \frac{1}{\alpha} \left(\sum_{ij} \alpha p_{ij} \log \frac{\alpha p_{ij}}{q_{ij}} \right) - \log \alpha. \quad (3.20)$$

We use this expression to report the true error value during the early exaggeration phase.

3.2 Adding Data to a Reference Embedding

We extend the t-SNE algorithm to enable the embedding of new data points to a reference embedding [37]. Our algorithm consists of estimating similarities between each new point and the reference data and optimizing the position of each new data point in the embedding space. Unlike parametric models such as principal component analysis or autoencoders, t-SNE does not define an explicit mapping to the embedding space, and embeddings need to be found through loss function optimization.

The position of a new data point in the embedding space is initialized to the median reference embedding position of its k nearest neighbors. While

we found the algorithm to be robust to various choices of k , we use $k = 10$ in our experiments.

We adapt the standard t-SNE formulation from Eqs. 2.1 and 2.5 with

$$p_{j|i} = \frac{\exp\left(-\frac{1}{2}\mathcal{D}(\mathbf{x}_i, \mathbf{v}_j)/\sigma_i^2\right)}{\sum_i \exp\left(-\frac{1}{2}\mathcal{D}(\mathbf{x}_i, \mathbf{v}_j)/\sigma_i^2\right)}, \quad (3.21)$$

$$q_{j|i} = \frac{(1 + \|\mathbf{y}_i - \mathbf{w}_j\|^2)^{-1}}{\sum_i (1 + \|\mathbf{y}_i - \mathbf{w}_j\|^2)^{-1}}, \quad (3.22)$$

where $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_M\} \in \mathbb{R}^D$, where M is the number of samples in the new data set and $\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_M\} \in \mathbb{R}^d$. Additionally, we omit the symmetrization step in Eq. 2.2. This enables new points to be inserted into the embedding independently of one another. The gradients of $\mathbf{w}_j$ with respect to the loss (Eq. 2.6) are:

$$\frac{\partial C}{\partial \mathbf{w}_j} = 2 \sum_i (p_{j|i} - q_{j|i}) (\mathbf{y}_i - \mathbf{w}_j) (1 + \|\mathbf{y}_i - \mathbf{w}_j\|^2)^{-1} \quad (3.23)$$

In the optimization step, we refine point positions using batch gradient descent. We use an adaptive learning rate scheme with momentum to speed up the convergence, as proposed by Jacobs [38] and used in the original t-SNE implementation by van der Maaten [1]. We run gradient descent with momentum α set to 0.8 for 250 iterations, where the optimization converged in all our experiments. The time complexity needed to evaluate the gradients in Eq. 3.23 is $\mathcal{O}(N \cdot M)$, however it is straightforward to modify the interpolation-based approximation from Eq. 2.29 to

$$\tilde{\phi}(w_i) = \sum_{\ell=1}^p L_{\ell, \tilde{\mathbf{y}}}(w_i) \left(\sum_{m=1}^p K(\tilde{y}_\ell, \tilde{y}_m) \left(\sum_{j=1}^N L_{m, \tilde{\mathbf{y}}}(y_j) q_j \right) \right), \quad (3.24)$$

where, again, the intermediate results can be cached during computation, reducing the time complexity to $\mathcal{O}(\max\{N, M\})$. We still use a single set of interpolation points $y_1 \dots y_p$ spanning the space of both $\mathbf{Y}$ and $\mathbf{W}$, which are selected in the same manner as in Eq. 2.29.

Special care must be taken to reduce the learning rate η as the default value in most implementations ($\eta = 200$) may cause points to “shoot off” from the reference embedding. This phenomenon is caused due to the embedding to a previously defined t-SNE space, where the distances between data points and corresponding gradients of the optimization function may be quite large. When running standard t-SNE, points are initialized and scaled to have variance 0.0001. The resulting gradients tend to be very small during the initial phase, resulting in stable convergence. When embedding new samples, the span of the embedding space is much larger, resulting in substantially larger gradients, and the default learning rate causes points to move very far from the reference embedding. In our experiments, we found that decreasing the learning rate to $\eta \sim 0.1$ produces stable solutions. Additionally, as a fallback, we use point-wise gradient clipping, which rescales the gradients if their vector norms exceed some predefined threshold.

These two steps are especially important when using the interpolation-based approximation, which places a grid of evenly-spaced interpolation points over the embedding space, where the spacing is a user parameter. The number of grid points is automatically determined by the span of the embedding. Clearly, if even one point “shoots off” far away from the embedding, the number of required grid points may grow exponentially, substantially increasing the runtime. The reduced learning rate and gradient clipping suppress this issue, and do not slow the convergence because of the adaptive learning rate scheme, provided the optimization is run for a sufficient number of steps.

3.3 Fast Mapping to an Embedding Space

High-dimensional data sets with many features often have much lower intrinsic dimensionality as many of the variables tend to be highly correlated with each other or are uninformative for the task at hand. We can exploit this to improve embedding quality and to speed up the construction of visualiza-

tions. To remove uninformative features, we employ existing feature-selection methods that find a smaller number of informative variables which capture interesting aspects of the data. Additionally, high-dimensional data sets often contain noise, which can be partially removed by using principal components analysis (PCA) to extract the driving factors of variation from the data. These two preprocessing steps can have two major benefits for subsequent t-SNE analysis: reducing the noise in the data leads to more informative visualizations, and reducing the number of dimensions can significantly speed up the construction of t-SNE embeddings.

3.4 openTSNE

Following the extensions and improvements proposed above, we introduce `openTSNE`, a comprehensive Python library that implements t-SNE [39]. The library is compatible with the Python data science ecosystem (e.g., `numpy`, `sklearn`, `scanpy`). Its modular design fosters extendibility and ease of experimentation with various settings and changes in the analysis pipeline. For example, the visualization in Fig. 5.3.a constructed with the standard t-SNE algorithm can be obtained by running the following code:

```
adata = anndata.read_h5ad("hrvatin_2018.h5ad")
tsne = openTSNE.TSNE(
    perplexity=30, metric="cosine", initialization="random"
)
embedding = tsne.fit(adata.obsm["pca"])
```

where we first read the data, set the desired t-SNE parameters, and run the algorithm using the `fit` method. Fig. 5.3.b, which uses multiscale similarity kernels, was obtained by executing the following code:

```
adata = anndata.read_h5ad("hrvatin_2018.h5ad")
affinities = openTSNE.affinity.Multiscale(
    adata.obsm["pca"], perplexities=[50, 500], metric="cosine"
)
```

```
init = openTSNE.initialization.pca(adata.obsm["pca"])
embedding = TSNEEmbedding(init, affinities)
embedding.optimize(n_iter=250, exaggeration=12, momentum=0.5,
    inplace=True)
embedding.optimize(n_iter=750, momentum=0.8, inplace=True)
```

where we, again, first read the data, define the affinity model based on two Gaussian kernels with varying perplexity, use PCA-based initialization, and run the typical two-stage t-SNE optimization.

Our `openTSNE` is currently the only Python t-SNE library that supports adding new samples into an existing embedding. For example, we can reuse the `embedding` prepared above to map new data points into the existing embedding space,

```
new_data = anndata.read_h5ad("chen_2017.h5ad")
adata, new_data = find_shared_genes(adata, new_data)
gene_mask = select_genes(adata.X, n=1000)
embedding.affinities = affinity.PerplexityBasedNN(
    adata[:, gene_mask].X, perplexity=30, metric="cosine"
)
new_embedding = embedding.transform(new_data[:, gene_mask].X)
```

to load and prepare the new data, define the affinity model, and obtain new embeddings. The resulting embedding is shown in Fig. 1.2. Our `openTSNE` embeds new data one data instance at the time, without modifying the reference embedding.

The proposed `openTSNE` library combines the fast interpolation-based approximation for gradient descent and approximates nearest neighbors using `pynndescent`, an implementation of nearest-neighbor descent [40]. This allows `openTSNE` to scale to massive data sets. For instance, Fig. 3.1 shows an embedding constructed using `openTSNE` of a scRNA-seq data set, containing transcriptomes of 2,058,652 single cells, where the entire data set was embedded in 56 minutes running on 32 threads on an Intel(R) Xeon(R) Gold 6138 CPU @ 2.00GHz processor.

Figure 3.1: A t-SNE embedding of data containing 2,058,652 single cells characterizing developing cells during organogenesis [41], constructed using openTSNE in using 32 threads in 56 minutes. The colors correspond to clusters identified by the original authors.

`openTSNE` is a pure Python library, and as such, introduces some computational overhead: `openTSNE` is about 25% slower than `FIt-SNE` [14], a recent t-SNE implementation in C++, but it is still orders of magnitude faster than other Python implementations including `scikit-learn` [36] and `MulticoreTSNE` [42]; see Fig. 3.2. `openTSNE` implements controlled execution with callback-based progress monitoring and control, making it suitable for interactive data exploration environments such as Orange [43, 44]. A pure-Python implementation offers distinct advantages over their C++ counterparts that include integration with Python’s rich data science infrastructure and ease of installation through PyPI and *conda*.

`openTSNE` is freely available on GitHub¹ and has, at the time of writing, over 500 GitHub stars. Detailed documentation and usage examples are available online².

¹<https://github.com/pavlin-polcar/openTSNE>

²<https://opentsne.readthedocs.io/>

Figure 3.2: Benchmarks of various Python t-SNE implementations. We ran benchmarks in both single-threaded and multi-threaded mode for all libraries except `scikit-learn`, which only supports single-threaded execution. `MulticoreTSNE` is a C++ library with a Python wrapper and implements a tree-based approximation of gradients as proposed by van der Maaten[28]. Both `openTSNE` and `FIt-SNE` use the recently proposed polynomial interpolation approximation proposed by Linderman *et al.* [14], which leads to better scaling with the number of data points. `openTSNE` is a Python library, which incurs some overhead, making it slightly slower than its C++ counterpart. All benchmarks were run on an Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz processor.

3.5 Integration into Orange

We incorporate our fully-featured Python implementation of t-SNE, **openTSNE**, into Orange, an interactive, data-exploration environment [43, 44]. Orange is based on workflows, where the user can build complex, data-analysis pipelines using combinations of simpler components, called widgets. Interactive environments are suitable for exploratory data analysis and data visualization, for both of which t-SNE is ideally suited. We implemented a new **t-SNE** widget, which is included in Orange v3.21.0 and above and is freely available on Github³. The widgets have also been featured in several blogs on Single-Cell Orange homepage^{4,5,6}.

Our t-SNE widget acts as a wrapper around **openTSNE** and includes all the improvements and tricks described in this manuscript. For example, Fig. 3.3 shows a how we can obtain a t-SNE embedding using the multiscale similarities trick proposed by Kobak and Berens [13]. The widget allows the user to change several parameters connected to the construction of t-SNE visualizations.

Perplexity. Perplexity can loosely be interpreted as the number of neighbors to which distances are preserved from each data point. Lower values reveal smaller, tightly packed clusters. Larger values better reveal the global organization of the resulting landscape, but may lead to the merging of smaller clusters, obscuring local aspects of the data.

Preserve global structure. The checkbox allows the user to switch from the standard t-SNE affinity model, using a single Gaussian kernel to estimate similarities between data points in the original space, to a mixture of Gaussian kernels as proposed by Kobak and Berens [13]. Ticking the checkbox often results in more globally-consistent visualizations, but also leads to longer optimization time.

³<https://github.com/biolab/orange3>

⁴<https://singlecell.biolab.si/blog/pancreas/>

⁵<https://singlecell.biolab.si/blog/x-and-y-chromosome-in-embryos/>

⁶https://singlecell.biolab.si/blog/aml_projection/

Exaggeration. The exaggeration parameter determines the strength of the attractive forces described in Chapter 2. This parameter is useful when dealing with large data sets, where the boundaries between clusters may be blurred.

PCA components. Extracting the top PCA components is a common preprocessing step when using t-SNE as it can remove noise in the data, often leading to clearer visualizations.

Normalize data. The user can decide to enable or disable zero-mean centering and variance scaling, another common preprocessing step in machine learning.

We also enable the addition of new data points into an existing embedding, as shown in Fig. 3.4 where we build upon the workflow from Fig. 3.3 and embed the remainder of the data set into the prebuilt embedding.

Our widget makes use of Qt threads, offloading all computation from the main GUI thread, keeping the user interface fully responsive and leading to a better overall user-experience. This is especially useful for long-running tasks since the user can view the construction of the embedding in real-time or decide to cancel the task if the computation is taking too long.

Figure 3.3: The new Orange t-SNE widget includes the latest improvements to t-SNE. The workflow in the upper panel shows the workflow, where we first load single-cell data, split it into a training and test set, and construct a t-SNE visualization on the training data. The lower panel shows the new t-SNE widget. The data contains bone marrow mononuclear cells from a patient with acute myeloid leukemia and two healthy donors as a control [45]. We can ensure global cluster alignment by checking the **Preserve global structure** checkbox, which indicates that the similarities between data points should be computed using a mixture of Gaussian kernels.

Figure 3.4: With our implementation, Orange supports embedding new samples into an existing embedding. Building on the workflow from Fig. 3.3, we consider the remaining test data and add it into the previous t-SNE embedding using the **Apply Domain** widget. The results are visualized in a **Scatter Plot**. Notice that the shape and color of the clusters largely agree with the embedding in Fig. 3.3.

Chapter 4

Data and Experimental Setup

We apply the proposed approach to t-SNE visualizations of single-cell data. Data in this realm include a variety of cells from specific tissues and are characterized through gene expressions. An example of one such data set, after removing poorly expressed genes, is shown in Fig. 4.1.

Genes	CCDC153	CCL4	CLDN11	CTSS	ERMN	LY6C1	MAG	PF4	SPP2	VIP
Cell ID										
B1TTTCAGGCTACA	1	1	1	7	0	0	0	17	0	0
B1AGATACCGTAAT	0	13	0	12	0	1	0	0	0	0
B0GCGTAGCACGCA	0	0	23	0	24	0	21	0	0	0
B1CAGTTTCTAGTC	0	0	1	0	0	0	0	0	0	66
B1TCACATCAGCAA	0	0	27	0	14	0	11	0	2	0

Figure 4.1: A sample of a single-cell gene expression data set [7]. Each row represents data belonging to a single cell and is uniquely identified using molecular barcodes. Columns represents different genes. During the sequencing process, each captured molecule is matched to a cell barcode and gene. The number of molecule detections is then aggregated into a count matrix, as shown on the figure. For instance, consider the entry in the second row and second column. The entry indicates that the molecule corresponding to gene **CCL4** was detected 13 times in cell **B1AGATACCGTAAT**.

In our experiments, we considered several recently published data sets where cells were annotated with the cell type. Our aim was to construct

t-SNE visualizations where similarly-typed cells would cluster together, despite systematic differences between data sources. Below, we list the data sets and describe single-cell specific data preprocessing procedures.

4.1 Data

To demonstrate the effectiveness of our approach, we use three pairs of reference and secondary single-cell data sets originating from different organisms and tissues. The data in each pair were chosen so that the majority of cell types from the secondary data set were included in the reference set (Table 4.1). The cells in the data sets originates from the following three tissues:

Mouse brain. The data set from Hrvatin *et al.* [6] contains cells from the visual cortex exploring transcriptional changes after exposure to light. This was used as a reference for the data from Chen *et al.* [7], containing various cells from the mouse hypothalamus and their reaction to food deprivation. From the secondary data, we removed cells with no corresponding types in the reference, namely ependymal cells, epithelial cells, tanycytes, and unlabelled cells.

Human pancreas. The data set from Baron *et al.* [10] was created as an atlas of pancreatic cell types. We used this set as a reference for data from Xin *et al.* [21], who examined transcriptional differences between healthy and type 2 diabetic patients.

Mouse retina. The data set from Macosko *et al.* [8] was created as an atlas of mouse retinal cell types. We used this as a reference for the data from Shekhar *et al.* [9], who built an atlas for different types of retinal bipolar cells.

Study	Organism/Tissue	Protocol	Cells	Genes	Cell Types	Sparsity (%)
Hrvatin <i>et al.</i>	mouse brain	inDrop	48,266	20,394	9	94
Chen <i>et al.</i>		Drop-seq	14,437	19,514	6	93
Baron <i>et al.</i>	human pancreas	inDrop	8,569	17,499	9	91
Xin <i>et al.</i>		SMARTer	1,492	31,588	4	86
Macosko <i>et al.</i>	mouse retina	Drop-seq	44,808	23,743	12	97
Shekhar <i>et al.</i>		Drop-seq	27,499	23,651	5	96

Table 4.1: Data sets used in our experiments. In each pair, the first data set (Hrvatin *et al.* [6], Baron *et al.* [10], and Macosko *et al.* [8]) was used as a reference. In all cases, we relied on the quality control and annotations from the original publication. To facilitate comparisons, the cell annotations were harmonized using cell type annotations from the cell ontology [48]. Notice that different RNA sequencing protocols were used to estimate gene expressions. We report the number of cell types from each data set retained after preprocessing. Single-cell data is sparse, typically containing less than 10% expressed genes per cell.

4.2 Single-cell data pipeline

Due to the specific nature of single-cell data, additional steps must be taken to properly apply t-SNE . We use a standard single-cell preprocessing pipeline, consisting of the selection of 3,000 representative genes (see Sec. 4.3), library size normalization, log-transformation, standardization, and PCA-based representation that retains 50 principal components [4, 46]. To obtain the reference embedding, we apply multi-scale t-SNE using PCA initialization [13]. Due to high-dimensionality of the preprocessed input data we use cosine distance to estimate similarities between reference data points [47]. When adding new data points from the secondary data set to the reference embedding, we select 1,000 informative genes present in both data sets and use these to estimate the similarities between the secondary data item and reference data points. The similarities are estimated using cosine similarity. We note that similarities are computed using the raw count matrices. The preprocessing stages are detailed in accompanying Python notebooks¹.

4.3 Gene selection

Single-cell data sets suffer from high levels of technical noise and low capture efficiency, resulting in sparse expression matrices [49]. To address this problem, we use a specialized feature-selection method, which exploits the mean-dropout relationship of expression counts as recently proposed by Kobak and Berens [13]. Here, genes with higher than expected dropout rate are regarded as potential markers for cell subpopulations and are retained in the data.

Given an expression matrix $\mathbf{X} \in \mathbb{R}^{N \times G}$ where N is the number of samples and G is the number of genes in the data set, we compute the fraction of cells where a gene g was not expressed as

$$d_g = \frac{1}{N} \sum_i I(X_{ig} = 0). \quad (4.1)$$

¹<https://github.com/biolab/tsne-embedding>

The mean $\log_2$ expression of gene g considers only cells expressing g :

$$m_g = \langle \log_2 X_{ig} \mid X_{ig} > 0 \rangle. \quad (4.2)$$

All genes expressed in less than ten cells are discarded. In order to select a specific number of $\hat{G}$ genes, we use a binary search to find a value b such that

$$\sum_g I(d_g > \exp[-(m_g - b)] + 0.02) = \hat{G}. \quad (4.3)$$

Chapter 5

Results and Discussion

Figs. 1.2, 5.1, and 5.2 show the embeddings of the reference data sets and their corresponding embeddings of the secondary data sets. In all the figures, the cells from the secondary data sets were positioned in the cluster of same-typed reference cells, providing strong evidence of the success of the proposed approach. There are some deviations to these observations; for instance, in Fig. 1.2 several oligodendrocyte precursor cells (OPCs) were mapped to oligodendrocytes. As the name would suggest, these two cell types are closely related: OPCs are cells that most commonly differentiate into oligodendrocytes. This may be due to differences in annotation criteria by different authors, or due to inherent similarities of these types of cells. Examples of such erroneous placements can be found in other figures as well, but they are not common and constitute less than 5% of the cells (less than 5% in brain, 1% in pancreas and 2% in retina secondary data sets).

Notice that we could simulate the split between reference and secondary data sets using one data set only and perform cross-validation, however this type of experiment would not incorporate batch effects. We want to remind the reader that handling batch effects were central to our endeavour and that the disregard of this effect could lead to overly-optimistic results and data visualizations strikingly different from ours. For example, compare the visualizations from Fig. 1.1.a and Fig. 1.2.b, or Figs. 1.1.b and 5.1.b.

Figure 5.1: Embedding of pancreatic cells from Baron *et al.* [10] and cells from the same tissue from Xin *et al.* [21]. Just like in Fig. 1.2 the vast majority of the cells from the secondary data set were correctly mapped to the same-typed cluster of reference cells.

There are additional modifications that we use in the embedding of the secondary data set that were recently proposed and enhance the original t-SNE visualization. One important extension is the use of multi-scale similarities that, besides local ordering of the data points, includes global optimization of cluster placement. For illustration, consider visualizations with standard and multi-scale t-SNE in Fig. 5.3. Notice, for instance, that in multi-scale t-SNE (Fig. 5.3.b) the clusters with neuronal cells are clumped together, while their placement in standard t-SNE is arbitrary (Fig. 5.3.a).

We also observed the important role of gene selection in crafting the reference embedding spaces. We found that when selecting an insufficient number of genes, the resulting visualizations display overly-fragmented clusters. When the selection is too broad and includes lowly expressed genes, the subclusters tend to overlap. These effects can all be attributed to sparse-

Figure 5.2: An embedding of a large reference of retinal cells from Macosko *et al.* [8] **(a)** and mapping of cells from a smaller study that focuses on bipolar cells from Shekhar *et al.* [9] **(b)**. We use colors consistent with the study by Macosko *et al.* [8].

ness of the data sets and may be intrinsic to single-cell data. In our studies, we found that selection of 3,000 genes yields most informative visualizations (Fig. 5.4).

In principle, our theoretically-grounded embedding of secondary data into the scaffold defined by the reference embedding could be simplified with the application of the nearest neighbors-based procedure. For example, while describing a set of tricks for t-SNE, Kobak and Berens [13] proposed positioning new points into a known embedding by placing them in the median position of their 10 nearest neighbors, where the neighborhood was estimated in the original data space. Notice that we use this approach as well, but only to initialize the positions of new data instances that are further optimized. In Fig. 5.5 we demonstrate that nearest neighbor-based positioning is insuf-

Figure 5.3: A comparison of standard and multi-scale t-SNE on data from the mouse visual cortex Hrvatin *et al.* [6]. **(a)** Standard t-SNE places clusters arbitrarily. **(b)** Augmenting t-SNE with multi-scale similarities and using proper initialization provides a more meaningful layout of the clusters. Neuronal types occupy one region of the space. Oligodendrocyte precursor cells (OPCs) are mainly progenitors to oligodendrocytes, but may also differentiate into neurons or astrocytes.

Figure 5.4: Gene selection plays an important role when constructing the reference embedding. **(a)** Using too few genes results in fragmented clusters. **(b)** Using an intermediate number of genes reveals clustering mostly consistent with cell annotations. **(c)** Including all the genes may lead to under-clustering of the more specialized cell types.

ficient and may yield clumped visualizations where the optimal positioning using the t-SNE loss function is much more dispersed and rightfully shows a more considerable variation in the secondary data. Some data points may also fall into the empty regions between different clusters, while after optimization, they typically move closer to same-typed groups.

To quantitatively evaluate the predictive accuracy of the described procedure, we fit k -nearest neighbors classifiers on each reference t-SNE embedding from Figs. 1.2.a, 5.1.a and 5.2.a and use them to predict the cell types for the secondary data set embeddings from Figs. 1.2.b, 5.1.b and 5.2.b. The accuracy measures are reported in Table 5.1.

We compare our approach to two machine learning techniques, namely a k -nearest neighbor classifier (KNN) and a random forest ensemble, and scMap-Cluster [50], a k -nearest neighbor based method, tailored explicitly to scRNA-seq data. scMap-Cluster uses three correlation-based distance measures and uses a voting scheme to perform classification. A threshold may also be set so that new cell types may be labeled as **unrecognized**, but since

Figure 5.5: Comparison of data placement using the nearest neighbors approach from Kobak and Berens [13] and the optimized placement using our algorithm. **(a)** Data points are placed to the median position of their 10 nearest neighbors in the reference set. **(b)** Point positions are optimized, revealing a different, more dispersed placement that better reflects the variety of cells in the secondary data set.

Tissue	Method	Accuracy	ARI
mouse brain	t-SNE	0.96	0.93
	KNN	0.96	0.93
	Random Forest	0.98	0.96
	scMap-Cluster	0.66	0.70
human pancreas	t-SNE	0.99	0.99
	KNN	0.99	0.98
	Random Forest	0.96	0.89
	scMap-Cluster	0.95	0.93
mouse retina	t-SNE	0.99	0.94
	KNN	0.99	0.96
	Random Forest	0.99	0.99
	scMap-Cluster	0.88	0.59

Table 5.1: We compare our approach to three other methods, evaluating performance using classification accuracy and the adjusted rand index (ARI). The proposed approach performs comparably to other methods.

our secondary data sets were chosen such that this does not happen, this was disabled. For the two machine learning approaches, we apply the typical single-cell preprocessing pipeline described in Sec. 4.2, i.e., library-size normalization, log-transformation, and select 1,000 most informative genes. Similarly to scMap-Cluster, we use the cosine distance to find the 5 nearest neighbors in the KNN model. We used 100 trees in the random forest ensemble. The models were fit on the reference data set, and no hyper-parameter tuning was performed.

Surprisingly, both the random forest and k -nearest neighbor models outperform scMap-Cluster, which is specifically tailored to scRNA-seq data. However, these results may be skewed, as, in our examples, all the cell-types from the secondary data set were present in the reference data set. One

of the core features of scMap-Cluster is the detection of novel cell types, which none of the other methods support. In other words, the other three methods would always assign a cell-type to a given cell, regardless of cell origin. Additionally, scMap-Cluster was primarily designed and tested on data sets produced by full-length sequencing protocols, which tend to detect a much higher number of molecules than other, sequencing protocols based on unique molecular identifiers (UMI). These two classes of sequencing protocols produce data sets with different sparsity and variance characteristics. This is consistent with the results in Table 5.1, as only the data sets from the human pancreas, were produced using a full-length sequencing protocol, where scMap-Cluster achieves reasonably high accuracy.

The aim of t-SNE is to construct embeddings, in which neighborhoods are preserved, therefore it is unsurprising that the accuracy of our t-SNE based approach is largely consistent with the k -nearest neighbors model. Our approach is comparable to the other models in terms of accuracy, but we would note that the goal of t-SNE embeddings is to serve as visual aids in exploratory data analysis. However, our procedure, in addition to providing the end-user with a cell-type prediction, allows the user to examine the low-dimensional embedding space, which may provide richer insight and interpretation of the resulting predictions.

Chapter 6

Collection of Reference Embeddings

Our method assumes that all cell types from the secondary data set are present in the reference embedding. The proposed method would fail to reveal novel cell types in the secondary data set, possibly positioning them arbitrarily close to unrelated clusters. Procedures such as scmap were recently proposed to cope with such cases and identify the cells whose type is new and not included in the reference [50]. Our procedure does not address such cases, and for scaling-up to a broader collection of cell types relies on the emerging availability of extensive collections of the reference data such as those managed by Human Cell Atlas initiative [51].

To address the issue of broad cell type coverage, we constructed three comprehensive reference embeddings containing the major cell types from regions of the human pancreas, mouse retina, and mouse brain. These are publicly available¹ and can be used to quickly get an overview of the cell-type composition of newly sequenced data.

¹<http://file.biolab.si/tsne-embedding/references/>

6.1 Construction of reference embeddings

The reference embeddings are all constructed using the following procedure:

1. We identify several suitable data sets originating from the same organism and tissue, each containing a reasonable number of cells. Cells in the data sets must also be annotated with their respective cell types.
2. We harmonize cell-type annotations to the cell ontology [48]. Most studies adopt their own form of labeling, which are largely inconsistent with one another.
3. We remove low-quality cells and doublets (errors in the sequencing protocol which assign molecules from multiple cells to a single cell identifier), cells with a total read count lower/larger than a threshold are removed. Different thresholds are selected for each data set due to varying sequencing depths in different protocols.
4. Similarly, we remove low-quality genes i.e., genes that were detected only a handful of times. These reads tend to be unreliable and are removed from subsequent analysis. We remove all genes with less than ten total read counts. Spike-ins – synthetic molecules added during sequencing to account for sequencing depth – are also removed.
5. We perform library size normalization to account for different cell sequencing depths and apply a variance-stabilizing log-transformation.
6. For each of the selected data sets, we find the 1,500 most informative genes. Non-selected genes and genes that are not present in all the data sets are discarded.
7. Next, we use scanorama [5] to project all the data sets into a joint, 100-dimensional space, where batch effects are removed.
8. Finally, we construct the t-SNE embedding. We use PCA-based initialization in combination with multiscale similarities using perplexities 30

and 500 to preserve the global organization of clusters. Given the high-dimensionality of the shared space, we use cosine distance to estimate similarities between data points.

6.2 Reference Embedding for Mouse Retina

We select two data sets from studies by Macosko *et al.* [8] and Shekhar *et al.* [9]. The data from Macosko *et al.* [8] contains major cell groups from the different layers of the mouse retina, including photoreceptor-rod, photoreceptor-cone, ganglion, horizontal, amacrine, Mueller and bipolar cells as well as some glial cells including astrocytes and microglia. The data set also includes a small number of endothelial, pericyte cells, and fibroblasts. Shekhar *et al.* [9] characterize different groups of bipolar cells from the mouse retina, revealing nine distinct subtypes of cone bipolar cells and a single population of rod bipolar cells. The final embedding is shown in Fig. 6.1 and contains 51,617 cells.

Figure 6.1: A comprehensive reference t-SNE embedding containing all the major cell types from the mouse retina. The cells originate from studies done by Macosko *et al.* [8] and Shekhar *et al.* [9].

6.3 Reference Embedding for Human Pancreas

We select four data sets containing human pancreatic cells. The data originate from studies by Baron *et al.* [10], Xin *et al.* [21], Muraro *et al.* [52], Segerstolpe *et al.* [53] and Enge *et al.* [54].

We removed Schwann cells, T cells, mast cells and macrophages from the Baron *et al.* [10] and Enge *et al.* [54] data set due to their low abundance and pancreatic endocrine cells from Enge *et al.* [54] due to poor annotation (alpha, beta, delta, pancreatic polypeptide and epsilon cells are all endocrine cells). The resulting embedding is shown in Fig. 6.2 and contains 16,192 cells.

Figure 6.2: We combine five pancreatic data sets containing the major pancreatic cell types including all endocrine cells (alpha, beta, delta, pancreatic polypeptide (PP) and epsilon cells) and exocrine cells (acinar cells).

6.4 Reference Embedding for Mouse Brain

We select seven data sets from studies of the mouse brain, originating from Hrvatin *et al.* [6], Chen *et al.* [7], Gokce *et al.* [55], Zeisel *et al.* [56], Zeisel *et al.* [57], Tasic *et al.* [58] and Harris *et al.* [59]. These studies cover a number of different brain regions such as the visual cortex [6], hypothalamus [7], cortex [58, 56], hippocampus [56, 59], stratum [55]. Additionally, Zeisel *et al.* [57] contains cells from the entire mouse nervous system. The resulting embedding is shown in Fig. 6.3 and contains 221,557 cells of 41 different cell types.

Figure 6.3: We construct a t-SNE embedding with cells originating from different regions of the mouse brain, originating from seven different single-cell studies. The data set contains 41 different cell types including various neuronal types such as excitatory and inhibitory neurons as well as different neurons from different brain regions, different types of glial cells including astrocytes, oligodendrocytes, microglia and others, vascular cells, endothelial and epithelial cells, and muscle cells.

Chapter 7

Conclusion

We propose a new approach to add new samples into existing t-SNE embeddings in a principled manner. We apply our approach to single-cell RNA-seq data where, surprisingly, the proposed procedure solves the batch effect problem, one of the most significant challenges in biological studies. We bundle our approach into a comprehensive Python implementation of the t-SNE algorithm, making it the first Python library capable of embedding new samples into existing embeddings. Our implementation makes use of the latest developments in t-SNE, allowing the method to be applied to data sets containing up to millions of data points.

While we have provided strong evidence that our approach provides an effective means of removing batch-effects, our method assumes that all newly inserted data points have a corresponding cluster in the original data set. If the newly inserted data contains cell-types that are not present in the reference embedding, these cells may be placed arbitrarily into the embedding. This calls for the construction of comprehensive cell atlases and their embeddings. We provide three such embeddings containing all the major cell types from the mouse retina, mouse brain and human pancreas. These reference embeddings are made publicly available¹ and can be used to facilitate cell-type annotation in novel scRNA-seq studies, leading to interpretable vi-

¹<http://file.biolab.si/tsne-embedding/references/>

sualizations.

Our approach allows for and encourages the creation and reuse of comprehensive reference embeddings, making comparative studies possible in fields where the data suffer from high levels of source-specific biases. While we have prepared a number of such reference embeddings for single-cell data, we anticipate that more will become available with the release of new single-cell data sets. In this way we pave the way for explainable AI through the use of visual classification.

Our group has already begun the development of new methods based on the ideas described in this manuscript, which enable the automatic annotation of reference embeddings, and the subsequent addition of new, unannotated samples. Prototypes of these methods have already been integrated into Orange [44], enabling the automatic annotation of single-cell data sets. Currently, most single-cell studies perform cell-type annotation by hand, which is both tedious and requires much domain knowledge. Our approach aims to streamline the single-cell analysis pipeline, both speeding up annotation and enabling comparative analyses between studies. Moreover, our (unpublished) approach finds regions in the visualization space and automatically annotates the plots with explanations for each discovered cluster and paves the way for reusable and interpretable embeddings (Fig. 7.1).

Figure 7.1: Our proposed approach has already been incorporated into Orange, and our group has been developing automatic cell-type annotation methods. The figure shows a workflow which first constructs a t-SNE embedding using the described procedure and automatically finds and annotates regions of different cell types. The resulting reference t-SNE visualization is shown in the lower-left widget and contains healthy bone marrow cells mononuclear cells [60]. In the lower-right region, new data points originating from a patient with acute myeloid leukemia were added to the existing embedding. The original data points in the reference data set are hidden, but the regions remain in-tact.

References

- [1] L. van der Maaten, G. Hinton, Visualizing data using t-SNE, *Journal of Machine Learning Research* 9 (Nov) (2008) 2579–2605.
- [2] A. Butler, P. Hoffman, P. Smibert, E. Papalexi, R. Satija, Integrating single-cell transcriptomic data across different conditions, technologies, and species, *Nature Biotechnology* 36 (5) (2018) 411.
- [3] L. Haghverdi, A. T. Lun, M. D. Morgan, J. C. Marioni, Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors, *Nature Biotechnology* 36 (5) (2018) 421.
- [4] T. Stuart, A. Butler, P. Hoffman, C. Hafemeister, E. Papalexi, W. M. Mauck III, Y. Hao, M. Stoeckius, P. Smibert, R. Satija, Comprehensive Integration of Single-Cell Data, *Cell* 177 (7) (2019) 1888–1902.
- [5] B. Hie, B. Bryson, B. Berger, Efficient integration of heterogeneous single-cell transcriptomes using Scanorama, *Nature Biotechnology* 37 (6) (2019) 685.
- [6] S. Hrvatin, D. R. Hochbaum, M. A. Nagy, M. Cicconet, K. Robertson, L. Cheadle, R. Zilionis, A. Ratner, R. Borges-Monroy, A. M. Klein, et al., Single-cell analysis of experience-dependent transcriptomic states in the mouse visual cortex, *Nature Neuroscience* 21 (1) (2018) 120.
- [7] R. Chen, X. Wu, L. Jiang, Y. Zhang, Single-cell RNA-seq reveals hypothalamic cell diversity, *Cell Reports* 18 (13) (2017) 3227–3241.

-
- [8] E. Z. Macosko, A. Basu, R. Satija, J. Nemesh, K. Shekhar, M. Goldman, I. Tirosh, A. R. Bialas, N. Kamitaki, E. M. Martersteck, et al., Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets, *Cell* 161 (5) (2015) 1202–1214.
 - [9] K. Shekhar, S. W. Lapan, I. E. Whitney, N. M. Tran, E. Z. Macosko, M. Kowalczyk, X. Adiconis, J. Z. Levin, J. Nemesh, M. Goldman, et al., Comprehensive classification of retinal bipolar neurons by single-cell transcriptomics, *Cell* 166 (5) (2016) 1308–1323.
 - [10] M. Baron, A. Veres, S. L. Wolock, A. L. Faust, R. Gaujoux, A. Vetere, J. H. Ryu, B. K. Wagner, S. S. Shen-Orr, A. M. Klein, et al., 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.
 - [11] J. Ding, A. Condon, S. P. Shah, Interpretable dimensionality reduction of single cell transcriptome data with deep generative models, *Nature Communications* 9 (1) (2018) 2002.
 - [12] E. Becht, L. McInnes, J. Healy, C.-A. Dutertre, I. W. Kwok, L. G. Ng, F. Ginhoux, E. W. Newell, Dimensionality reduction for visualizing single-cell data using UMAP, *Nature Biotechnology* 37 (1) (2019) 38.
 - [13] D. Kobak, P. Berens, The art of using t-SNE for single-cell transcriptomics, *bioRxiv* (2018) 453449.
 - [14] G. C. Linderman, M. Rachh, J. G. Hoskins, S. Steinerberger, Y. Kluger, Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data, *Nature Methods* 16 (3) (2019) 243.
 - [15] K. Saenko, B. Kulis, M. Fritz, T. Darrell, Adapting visual category models to new domains, in: *European Conference on Computer Vision*, Springer, 2010, pp. 213–226.
 - [16] X. Glorot, A. Bordes, Y. Bengio, Domain adaptation for large-scale sentiment classification: A deep learning approach, in: *Proceedings of the*

- 28th International Conference on Machine Learning (ICML-11), 2011, pp. 513–520.
- [17] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, M. Salzmann, Unsupervised domain adaptation by domain invariant projection, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 769–776.
- [18] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, T. Darrell, Deep domain confusion: Maximizing for domain invariance, arXiv preprint arXiv:1412.3474 (2014).
- [19] M. Long, H. Zhu, J. Wang, M. I. Jordan, Unsupervised domain adaptation with residual transfer networks, in: Advances in Neural Information Processing Systems, 2016, pp. 136–144.
- [20] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, D. Erhan, Domain separation networks, in: Advances in Neural information Processing Systems, 2016, pp. 343–351.
- [21] Y. Xin, J. Kim, H. Okamoto, M. Ni, Y. Wei, C. Adler, A. J. Murphy, G. D. Yancopoulos, C. Lin, J. Gromada, RNA sequencing of single human islet cells reveals type 2 diabetes genes, *Cell Metabolism* 24 (4) (2016) 608–615.
- [22] L. McInnes, J. Healy, J. Melville, UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, arXiv preprint arXiv:1802.03426 (2018).
- [23] M. Wattenberg, F. Viégas, I. Johnson, How to use t-SNE effectively, *Distill* 1 (10) (2016) e2.
- [24] R. Gopalan, R. Li, R. Chellappa, Domain adaptation for object recognition: An unsupervised approach, in: 2011 International Conference on Computer Vision, IEEE, 2011, pp. 999–1006.

-
- [25] S. Bickel, M. Brückner, T. Scheffer, Discriminative learning under co-variate shift, *Journal of Machine Learning Research* 10 (Sep) (2009) 2137–2155.
 - [26] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, N. D. Lawrence, *Dataset shift in machine learning*, The MIT Press, 2009.
 - [27] L. van der Maaten, Learning a parametric embedding by preserving local structure, in: *Proceedings of the Twelfth International Conference on Artificial Intelligence & Statistics (AI-STATS)*, JMLR W&CP, Vol. 5, 2009, pp. 384–391.
 - [28] L. van der Maaten, Accelerating t-SNE using tree-based algorithms., *Journal of Machine Learning Research* 15 (1) (2014) 3221–3245.
 - [29] P. N. Yianilos, Data structures and algorithms for nearest neighbor search in general metric spaces, in: *Symposium on Discrete Algorithms*, Vol. 93, 1993, pp. 311–321.
 - [30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, pp. 248–255.
 - [31] A. Köhn, F. Stegen, T. Baumann, Mining the Spoken Wikipedia for Speech Data and Beyond, in: N. C. C. Chair), K. Choukri, T. Declerck, M. Grobelnik, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, S. Piperidis (Eds.), *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, European Language Resources Association (ELRA), Paris, France, 2016.
 - [32] J. McAuley, J. Leskovec, Hidden factors and hidden topics: understanding rating dimensions with review text, in: *Proceedings of the 7th ACM Conference on Recommender Systems*, ACM, 2013, pp. 165–172.
 - [33] S. J. Pan, Q. Yang, A survey on transfer learning, *IEEE Transactions on Knowledge and Data Engineering* 22 (10) (2009) 1345–1359.

-
- [34] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, J. Liang, Convolutional neural networks for medical image analysis: Full training or fine tuning?, *IEEE Transactions on Medical Imaging* 35 (5) (2016) 1299–1312.
 - [35] J. T. Leek, R. B. Scharpf, H. C. Bravo, D. Simcha, B. Langmead, W. E. Johnson, D. Geman, K. Baggerly, R. A. Irizarry, Tackling the widespread and critical impact of batch effects in high-throughput data, *Nature Reviews Genetics* 11 (10) (2010) 733.
 - [36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research* 12 (2011) 2825–2830.
 - [37] P. G. Poličar, M. Stražar, B. Zupan, Embedding to Reference t-SNE Space Addresses Batch Effects in Single-Cell Classification, in: *Proceedings of the 22th International Conference on Discovery Science*, Springer, 2019.
 - [38] R. A. Jacobs, Increased rates of convergence through learning rate adaptation, *Neural Networks* 1 (4) (1988) 295–307.
 - [39] P. G. Poličar, M. Stražar, B. Zupan, openTSNE: a modular Python library for t-SNE dimensionality reduction and embedding, (submitted) (2019).
 - [40] W. Dong, C. Moses, K. Li, Efficient k-nearest neighbor graph construction for generic similarity measures, in: *Proceedings of the 20th international Conference on World Wide Web*, ACM, 2011, pp. 577–586.
 - [41] J. Cao, M. Spielmann, X. Qiu, X. Huang, D. M. Ibrahim, A. J. Hill, F. Zhang, S. Mundlos, L. Christiansen, F. J. Steemers, et al., The

- single-cell transcriptional landscape of mammalian organogenesis, *Nature* 566 (7745) (2019) 496.
- [42] D. Ulyanov, Multicore-TSNE, <https://github.com/DmitryUlyanov/Multicore-TSNE> (2016).
- [43] J. Demšar, T. Curk, A. Erjavec, Č. Gorup, T. Hočevár, M. Milutinovič, M. Možina, M. Polajnar, M. Toplak, A. Starič, et al., Orange: data mining toolbox in Python, *The Journal of Machine Learning Research* 14 (1) (2013) 2349–2353.
- [44] M. Stražar, L. Žagar, J. Kokošar, V. Tanko, A. Erjavec, P. G. Poličar, A. Starič, J. Demšar, G. Shaulsky, V. Menon, et al., scOrange—a tool for hands-on training of concepts from single-cell data analytics, *Bioinformatics* 35 (14) (2019) i4–i12.
- [45] G. X. Zheng, J. M. Terry, P. Belgrader, P. Ryvkin, Z. W. Bent, R. Wilson, S. B. Ziraldo, T. D. Wheeler, G. P. McDermott, J. Zhu, et al., Massively parallel digital transcriptional profiling of single cells, *Nature Communications* 8 (2017) 14049.
- [46] F. A. Wolf, P. Angerer, F. J. Theis, SCANPY: large-scale single-cell gene expression data analysis, *Genome Biology* 19 (1) (2018) 15.
- [47] P. M. Domingos, A few useful things to know about machine learning., *Communications for the ACM* 55 (10) (2012) 78–87.
- [48] J. Bard, S. Y. Rhee, M. Ashburner, An ontology for cell types, *Genome Biology* 6 (2) (2005) R21.
- [49] S. Islam, A. Zeisel, S. Joost, G. La Manno, P. Zajac, M. Kasper, P. Lönnerberg, S. Linnarsson, Quantitative single-cell RNA-seq with unique molecular identifiers, *Nature Methods* 11 (2) (2014) 163.
- [50] V. Y. Kiselev, A. Yiu, M. Hemberg, scmap: projection of single-cell RNA-seq data across data sets, *Nature Methods* 15 (5) (2018) 359.

-
- [51] O. Rozenblatt-Rosen, M. J. Stubbington, A. Regev, S. A. Teichmann, The Human Cell Atlas: from vision to reality, *Nature News* 550 (7677) (2017) 451.
- [52] M. J. Muraro, G. Dharmadhikari, D. Grün, N. Groen, T. Dielen, E. Jansen, L. van Gorp, M. A. Engelse, F. Carlotti, E. J. de Koning, et al., A single-cell transcriptome atlas of the human pancreas, *Cell systems* 3 (4) (2016) 385–394.
- [53] Å. Segerstolpe, A. Palasantza, P. Eliasson, E.-M. Andersson, A.-C. Andréasson, X. Sun, S. Picelli, A. Sabirsh, M. Clausen, M. K. Bjursell, et al., Single-cell transcriptome profiling of human pancreatic islets in health and type 2 diabetes, *Cell Metabolism* 24 (4) (2016) 593–607.
- [54] M. Enge, H. E. Arda, M. Mignardi, J. Beausang, R. Bottino, S. K. Kim, S. R. Quake, Single-cell analysis of human pancreas reveals transcriptional signatures of aging and somatic mutation patterns, *Cell* 171 (2) (2017) 321–330.
- [55] O. Gokce, G. M. Stanley, B. Treutlein, N. F. Neff, J. G. Camp, R. C. Malenka, P. E. Rothwell, M. V. Fuccillo, T. C. Südhof, S. R. Quake, Cellular taxonomy of the mouse striatum as revealed by single-cell RNA-seq, *Cell Reports* 16 (4) (2016) 1126–1137.
- [56] A. Zeisel, A. B. Muñoz-Manchado, S. Codeluppi, P. Lönnerberg, G. La Manno, A. Juréus, S. Marques, H. Munguba, L. He, C. Betsholtz, et al., Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq, *Science* 347 (6226) (2015) 1138–1142.
- [57] A. Zeisel, H. Hochgerner, P. Lönnerberg, A. Johnsson, F. Memic, J. Van Der Zwan, M. Häring, E. Braun, L. E. Borm, G. La Manno, et al., Molecular architecture of the mouse nervous system, *Cell* 174 (4) (2018) 999–1014.

-
- [58] B. Tasic, V. Menon, T. N. Nguyen, T. K. Kim, T. Jarsky, Z. Yao, B. Levi, L. T. Gray, S. A. Sorensen, T. Dolbeare, et al., Adult mouse cortical cell taxonomy revealed by single cell transcriptomics, *Nature Neuroscience* 19 (2) (2016) 335.
- [59] K. D. Harris, H. Hochgerner, N. G. Skene, L. Magno, L. Katona, C. B. Gonzales, P. Somogyi, N. Kessaris, S. Linnarsson, J. Hjerling-Leffler, Classes and continua of hippocampal CA1 inhibitory neurons revealed by single-cell transcriptomics, *PLoS Biology* 16 (6) (2018) e2006387.
- [60] P. van Galen, V. Hovestadt, M. H. Wadsworth II, T. K. Hughes, G. K. Griffin, S. Battaglia, J. A. Verga, J. Stephansky, T. J. Pastika, J. L. Story, et al., Single-cell RNA-seq reveals AML hierarchies relevant to disease progression and immunity, *Cell* 176 (6) (2019) 1265–1281.