A biology-driven deep generative model for cell-type annotation in cytometry

Quentin Blampey, Nadège Bercovici, Charles-Antoine Dutertre, Isabelle Pic, Joana Mourato Ribeiro, Fabrice André, Paul-Henry Cournède

OXFORD

Briefings in Bioinformatics, 2023, 1–10
https://doi.org/10.1093/bib/bbad260
Problem Solving Protocol

A biology-driven deep generative model for cell-type annotation in cytometry

Quentin Blampey, Nadège Bercovici, Charles-Antoine Dutertre, Isabelle Pic, Joana Mourato Ribeiro, Fabrice André and Paul-Henry Cournède

Corresponding author. Paul-Henry Cournède, E-mail: paul-henry.cournede@centralesupelec.fr

Abstract

Cytometry enables precise single-cell phenotyping within heterogeneous populations. These cell types are traditionally annotated via manual gating, but this method lacks reproducibility and sensitivity to batch effect. Also, the most recent cytometers—spectral flow or mass cytometers—create rich and high-dimensional data whose analysis via manual gating becomes challenging and time-consuming. To tackle these limitations, we introduce Scyan https://github.com/MICS-Lab/scyan, a Single-cell Cytometry Annotation Network that automatically annotates cell types using only prior expert knowledge about the cytometry panel. For this, it uses a normalizing flow—a type of deep generative model—that maps protein expressions into a biologically relevant latent space. We demonstrate that Scyan significantly outperforms the related state-of-the-art models on multiple public datasets while being faster and interpretable. In addition, Scyan overcomes several complementary tasks, such as batch-effect correction, debarcoding and population discovery. Overall, this model accelerates and eases cell population characterization, quantification and discovery in cytometry.

Keywords: Cytometry, Deep Learning, Normalizing Flows, Cell-type annotation, Batch-effect correction

INTRODUCTION

The simultaneous detection of several cellular proteins by spectral and mass cytometry opens up an unprecedented way to detect, quantify and monitor the function of highly specific cell populations from complex biological samples [1]. These rich analyses are made possible using large panels of markers, typically more than 30 or 40 markers, which considerably increases the information in the data [2]. They provide key insights to better understand specific diseases, immune cell functions or monitor the response to therapies [3]. To obtain such results, population annotation must be performed to provide each cell with a biologically meaningful cell type. Yet, due to the data's high dimensionality and complexity, manual annotations become challenging and labor-intensive [4]. This process, called gating [5], is highly subjective and sensitive to the batch effect, or non-biological data variability [4]. These drawbacks are amplified as the number of cytometry samples increases, reinforcing the need to develop and use automatic tools in population annotation and data analysis [4, 6].

Many clustering tools [7–9] have been developed for automatic data exploration and population discovery. However, a manual analysis of marker expressions is still required to name each cluster with a meaningful cell type. Indeed, clusters do not necessarily correspond to one specific cell type, and it is up to the investigator to decide to which population each cluster corresponds. Mostly, clustering tools are also not scaling well and are sensible to batch-effect, making this approach less suited for large datasets with a large inter-sample variation. An alternative approach to clustering is to use automatic annotation models. The first category of annotation models are supervised or semi-supervised models [10–13], but they rely on prior manual gating to train the models. Moreover, these models can only annotate populations with predefined types of cells, and they cannot be used to discover new ones. The second category, to which our model belongs, corresponds to unsupervised annotation models that leverage prior biological knowledge about the panel of markers. Although some models have been developed [13–15], they either (i) lack interpretability, (ii) cannot discover new populations, (iii) require the usage of batch-effect correction models before being applied or (iv) scale poorly to large datasets. Surprisingly, deep learning has been underused for cytometry annotations, while proving efficient and flexible for many related applications of single-cell biology [16–18].

In this paper, we introduce a single-cell cytometry annotation network called Scyan that annotates cell types and corrects batch effects concurrently without any label or gating needed. Scyan is a Bayesian probabilistic model composed of a deep invertible neural network called a normalizing flow [19–21]. This flow transforms cell data into a latent space that is used for annotation, does not contain batch effect and is key for population discovery.

We demonstrate Scyan efficiency, scalability and interpretability on three public mass cytometry datasets for which manually annotated cell populations are used to evaluate models. We compare Scyan classification performance with two knowledge-based approaches [14, 15], one clustering method [7, 14] and two supervised models [11, 12]. Additionally, we compare Scyan batch-effect correction with four state-of-the-art batch correction methods [18, 22–24]. We also show that our model can be used for population discovery, as well as for the general task of debarcoding. Overall, these properties make Scyan an end-to-end analysis framework for mass/spectral/flow cytometry.

Received: April 27, 2023. **Revised:** June 20, 2023. **Accepted:** June 27, 2023

METHODS

Problem definition

Let $\mathbf{x_1}, \ldots, \mathbf{x_N} \in \mathbb{R}^M$ represent the vectors of M marker expressions for N cells. We assume these expression levels have already been transformed using the *asinh* or logicle [25] transformation and standardized (see supplementary Section 11). Our objective is to associate each cell to one of the P predefined cell types using a marker-population table $\boldsymbol{\rho} \in \mathbb{R}^{P \times M}$, with $\rho_{z,m}$ summarizing the knowledge about the expression of marker m for population z. If it is known that population z expresses m then $\rho_{z,m} = 1$; if we know that it does not express m then $\rho_{z,m} = -1$. Otherwise, if we have no knowledge or if the expression can vary among the population, then $\rho_{z,m} = \text{NA}$. Note that it is also possible to choose values in $\mathbb{R}$; for instance, for mid or low expressions, we can choose 0 and 0.5, respectively (see supplementary Section 6). In addition, we can add covariates $\mathbf{c_1}, \ldots, \mathbf{c_N} \in \mathbb{R}^{M_c}$ associated with each cell, e.g. information about the batch or which antibody has been used by the cytometer. M_c denotes the number of covariates; it can be zero if no covariate is provided.

Generative process

In this section, we detail the generative process of Scyan (illustrated in Fig. 1b/c). Let $\mathbf{X}$ be the random vector of size M representing one cell by its standardized marker expressions; in other words, $\mathbf{X}$ is the random variable from which $\mathbf{x_1}, \ldots, \mathbf{x_N}$ are sampled. We model $\mathbf{X}$ by the following deep generative process:

$$
\begin{aligned}
Z &\sim \mathit{Categorical}(\boldsymbol{\pi}) \\
\boldsymbol{E} \mid Z = (e_m)_{1 \le m \le M}, \text{where} &\begin{cases} e_m = \rho_{Z,m} & \text{if } \rho_{Z,m} \neq \text{NA} \\ e_m \sim \mathscr{U}([-1,1]) & \text{otherwise,} \end{cases} \\
\boldsymbol{H} &\sim \mathcal{N}(\mathbf{0}, \sigma \mathbf{I_M}) \\
\boldsymbol{U} &= \boldsymbol{E} + \boldsymbol{H} \\
\boldsymbol{X} &= f_{\boldsymbol{\phi}}^{-1}(\boldsymbol{U}).
\end{aligned}
\tag{1}
$$

In the above equations, $\boldsymbol{\pi} = (\pi_z)_{1 \le z \le P}$ represents the weights of each population, with the constraints $\pi_z \geq 0$ and $\sum_z \pi_z = 1$. Z is the random variable corresponding to a cell type among the P possible ones. $\boldsymbol{E}$ is a population-specific variable whose terms are either known according to the expert knowledge table $\boldsymbol{\rho}$ or drawn from a uniform distribution between negative expressions (represented by -1) and positive expressions (represented by +1). $\boldsymbol{H}$ contains cell-specific terms, such as autofluorescence. Finally, $\boldsymbol{U}$ is the cell's latent expressions, summing a population-specific component and a cell-specific one. Note that $\boldsymbol{E}$, $\boldsymbol{H}$ and $\boldsymbol{U}$ have a dimension of M. Also, $\boldsymbol{U}$ can be transformed into a measured cell marker expressions vector $\boldsymbol{X}$ by the inverse of a deep invertible network $f_{\boldsymbol{\phi}}$ detailed below. The normalizing flow aims to learn an invertible mapping between the actual marker expression distribution and the target $\boldsymbol{U}$. By mapping marker expressions to a biologically defined latent space, we force the transformation to provide latent expressions on a scale that is shared for every marker, going from negative (-1) to positive (+1). These latent marker expressions are meant to be free of batch effect or any non-biological factor. By the design of $f_{\boldsymbol{\phi}}$ and of the objective function, the normalizing flow is not allowed to make huge space distortions, which helps preserve the biology. Also, an ablation study shows that both $f_{\boldsymbol{\phi}}$ and the uniform term are key for good performances (see supplementary Section 3).

Invertible transformation network

The core network, $f_{\boldsymbol{\phi}}$ (illustrated in Fig. 1b), is a normalizing flow [19–21]. It transforms the target distribution p_X into the known base distribution p_U, which was described in the previous section. Using a change of variables, we can compute the exact likelihood of a sample $\boldsymbol{x}$ by

$$
p_X(\mathbf{x}; \boldsymbol{\theta}) = p_U(f_{\boldsymbol{\phi}}(\mathbf{x}); \boldsymbol{\pi}) \cdot \log \left| det \frac{\partial f_{\boldsymbol{\phi}}(\mathbf{x})}{\partial \mathbf{x}^T} \right|. \tag{2}
$$

To be able to compute this expression, we need to choose an invertible network with a tractable Jacobian determinant. We have chosen a set of transformations called Real Non-Volume-Preserving (Real NVP [26], we justify this choice in supplementary Section 1) transformations, which are compositions of functions named coupling layers $f_{\boldsymbol{\phi}} := f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}$ with L the number of coupling layers. Each coupling layer $f^{(i)} : (\mathbf{x}, \mathbf{c}) \mapsto \mathbf{y}$ splits both $\boldsymbol{x}$ and $\mathbf{y}$ into two components $(\mathbf{x^{(1)}}, \mathbf{x^{(2)}}), (\mathbf{y^{(1)}}, \mathbf{y^{(2)}})$ on which distinct transformations are applied. We propose below an extension of the traditional coupling layer [26] to integrate covariates $\mathbf{c}$ (illustrated in Fig. 1c):

$$
\begin{cases} \mathbf{y^{(1)}} = \mathbf{x^{(1)}} \\ \mathbf{y^{(2)}} = \mathbf{x^{(2)}} \odot exp\Big(s([\mathbf{x^{(1)}}; \mathbf{c}])\Big) + t([\mathbf{x^{(1)}}; \mathbf{c}]). \end{cases} \tag{3}
$$

In the equations above, $\odot$ stands for the element-wise product, $[.;.]$ is the concatenation operator and (s, t) are functions from $\mathbb{R}^{d+M_c}$ to $\mathbb{R}^{M-d}$, where d is the size of $\mathbf{x^{(1)}}$. These functions can be arbitrarily complex, in our case, multi-layer-perceptrons. Note that the indices used by the coupling layer to split $\boldsymbol{x}$ into $(\mathbf{x^{(1)}}, \mathbf{x^{(2)}})$ are set before training and are different for every coupling layer. This way, we ensure that the flow transforms all the markers. Each coupling layer has an easy-to-compute log Jacobian determinant, which is $\sum_i s([\mathbf{x^{(1)}}; \mathbf{c}])_i$, and is easily invertible as shown in the following equations:

$$
\begin{cases} \mathbf{x^{(1)}} = \mathbf{y^{(1)}} \\ \mathbf{x^{(2)}} = (\mathbf{y^{(2)}} - t([\mathbf{y^{(1)}}; \mathbf{c}])) \odot exp\Big(-s([\mathbf{y^{(1)}}; \mathbf{c}])\Big). \end{cases} \tag{4}
$$

As $f_{\boldsymbol{\phi}}$ is a stack of coupling layers, it is also invertible, and its log Jacobian determinant is obtained by summing each coupling layer log Jacobian determinant. Stacking many coupling layers is essential to learning a rich target distribution and complex variables interdependencies. Overall, the normalizing flow has some interesting properties: (i) the coupling layers preserve order relation for two different expression values, and (ii) penalize huge space distortion (the log determinant term). The two properties are useful to preserve biological variability as much as possible.

Learning process

The model parameters are $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{\phi})$. For computational stability during training, instead of learning $\boldsymbol{\pi}$ itself we actually learn logits $(l_z)_{1 \le z \le P}$ from which we obtain $\pi_z = \frac{e^{l_z}}{\sum_k e^{l_k}}$. By doing this, we ensure the positivity of each weight and guarantee they sum to 1. To train the model, we minimize the Kullback–Leibler (KL) divergence between the cell's empirical marker-expression distribution p_{X^*} and our model distribution p_X. It is equivalent to minimizing the negative log-likelihood of the observed cell expressions $-\mathbb{E}_{\mathbf{x} \sim p_{X^*}}\Big[\log p_X(\mathbf{x}; \boldsymbol{\theta})\Big]$ over $\boldsymbol{\theta}$. Using subsection 2 and adapting it to

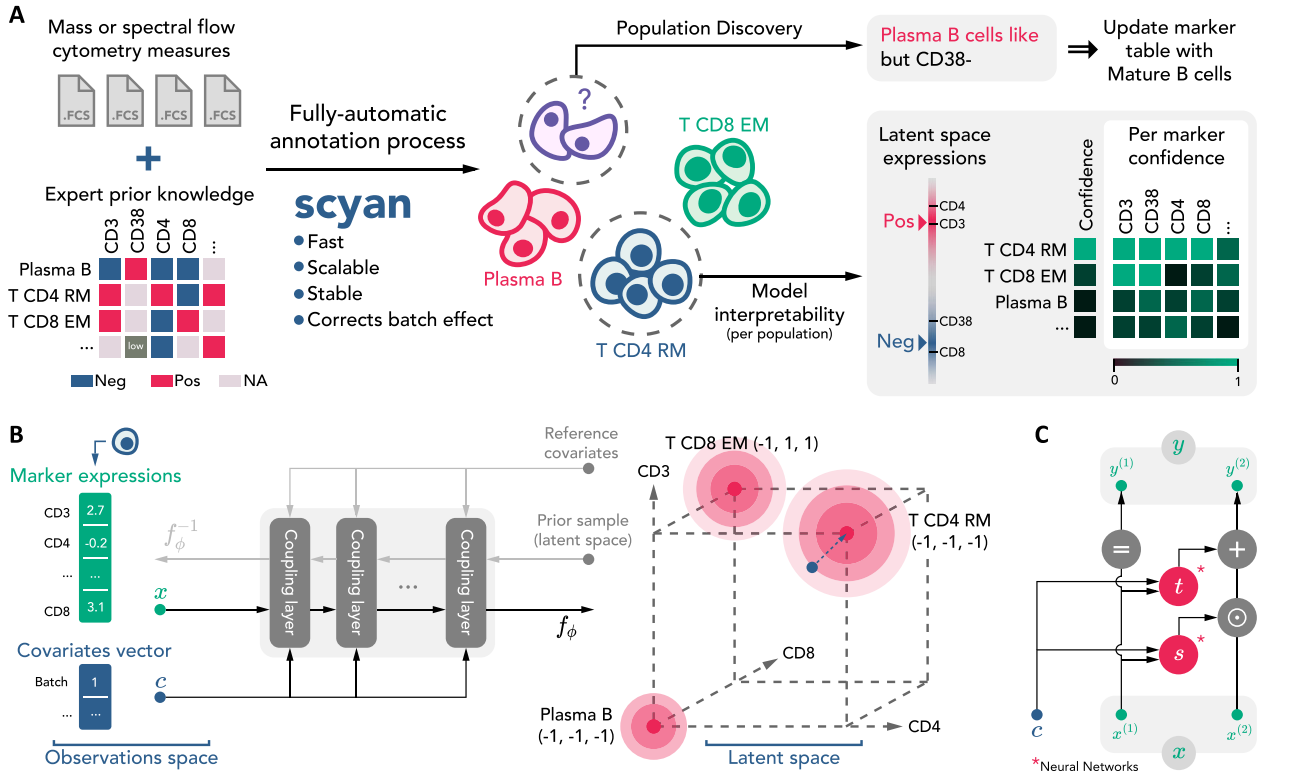

Figure 1. Overview of Scyan usage and architecture. A, Illustration of Scyan typical use case. It requires (i) one or multiple cytometry acquisitions and (ii) a knowledge table that details which population is expected to express which markers: Then, Scyan annotates cells in a fast and unsupervised (or fully automatic) manner while removing batch effect (if any). After training, we provide interpretability tools to understand Scyan annotations and discover new populations that can eventually be added to the table afterward. **B**, Illustration of Scyan architecture: it is composed of two core components: (i) f_{ϕ}, a neural network called normalizing flow, and (ii) a latent space on which a target distribution $\mathbf{U}$ is defined (cube on the right). The table from (a) is used to define this target distribution $\mathbf{U}$ mathematically. Also, the latent space (on which $\mathbf{U}$ is defined) has the same dimension as the original space; therefore, each marker has its corresponding latent expression. Finally, one cell is represented by its marker expressions vector and eventual covariates. Once a cell is mapped into the latent space, annotation can be made by choosing the highest probable population, whose distribution is Gaussian-like and on a hypercube vertex. **C**, One coupling layer, the elementary unit that composes the transformation f_{ϕ}, contains two multi-layer perceptrons (s and t) and uses cell covariates such as the batch information.

integrate covariates leads to minimizing the following quantity:

$$\mathscr{L}_{\mathrm{KL}}(\boldsymbol{\theta}) = -\sum_{1\le i\le N}\left[\log\left(p_U(f_{\boldsymbol{\phi}}(\mathbf{x_i},\mathbf{c_i});\boldsymbol{\pi})\right) + \log\left|det\,\frac{\partial f_{\boldsymbol{\phi}}(\mathbf{x_i},\mathbf{c_i})}{\partial \mathbf{x}^{\mathrm{T}}}\right|\right]. \quad (5)$$

In the above equation, the term $p_U(f_{\boldsymbol{\phi}}(\mathbf{x_i},\mathbf{c_i});\boldsymbol{\pi}) = \sum_{z=1}^{p}\pi_z \cdot p_{U|Z=z}(f_{\boldsymbol{\phi}}(\mathbf{x_i},\mathbf{c_i}))$, which is not computationally tractable because the presence of NA in $\boldsymbol{\rho}$ leads to the summation of a uniform and a normal random variable. We approximate the density of the sum of the two random variables by a piecewise density function that is constant on $[-1+\sigma, 1-\sigma]$ with Gaussian queues outside of this interval. In practice, we choose a normal law with a low standard deviation, which leads to a good piecewise approximation (see supplementary Section 2). If we consider the KL-divergence as described above, some modes may collapse; that is, one small population may not be predicted. Indeed, a small population z that has a small weight π_z leads to smaller gradients towards this population. To solve this issue, we favor small populations once every two epochs. For that, for all z, we replace π_z by $\pi_z^{(-\mathrm{T})} = \frac{e^{-l_z/T}}{\sum_k e^{-l_k/T}}$, where T is called temperature [27, 28] as it increases the entropy of $\boldsymbol{\pi}^{(-\mathrm{T})}$. Note that here we added the minus signs to reverse the weights of the populations so that it favors small ones. A temperature close to 0 leads to high weights for small populations, while an infinite temperature leads to equal population weights, i.e. the maximum entropy. Alternating between $\boldsymbol{\pi}$ and $\boldsymbol{\pi}^{(-\mathrm{T})}$ allows for a better balance of population sizes at the end of the training.

We optimize the loss on mini-batches of cells using the Adam optimizer [29]. Once finished training, the annotation process $\mathscr{A}_{\boldsymbol{\theta}}$ consists in choosing the most likely population according to the data using Bayes's rule. So, for a cell $\boldsymbol{x}$ with covariates $\mathbf{c}$, we have

$$\mathscr{A}_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{c}) = argmax_{1\le z\le p}\ \pi_z \cdot p_{U|Z=z}(f_{\boldsymbol{\phi}}(\mathbf{x},\mathbf{c})). \quad (6)$$

We also define a log threshold t_{min} to decide whether or not to label a cell (see supplementary Section 4 to determine its value). That is, we do not label a cell if

$$max_{1\le z\le p}\ p_{U|Z=z}(f_{\boldsymbol{\phi}}(\mathbf{x},\mathbf{c})) \le e^{t_{min}}.$$

Batch-effect correction

Batch information is one-hot-encoded and added to the covariates (see covariates usage in Eq. 4). Minimizing $\mathscr{L}_{\mathrm{KL}}(\boldsymbol{\theta})$ (as in Eq. 5) will naturally reduce inter-batch variability in the latent space. Note that we do not need to add any additional loss terms, and Scyan will naturally use the batch information inside the covariates to better align the distribution of the different batches on the target distribution $\mathbf{U}$. Since the normalizing flow is invertible, we

can conserve this batch alignment when mapping back in the original space using a simple trick: we map all cells back with the same reference covariates $\mathbf{c_{ref}}$. Note that $\mathbf{c_{ref}}$ is simply one of the existing covariates vectors; usually, we choose the covariates of the patient with the most cells. More formally, to correct the batch effect of a sample $\boldsymbol{x}$ with covariates $\mathbf{c} \neq \mathbf{c_{ref}}$, we first transform $\boldsymbol{x}$ into its latent expressions via $f_{\boldsymbol{\phi}}$. Since the latent space is batch-effect free, latent expressions can then be transformed back into the original space using the covariates of the reference batch and $f_{\boldsymbol{\phi}}^{-1}$. Formally, we denote by $\tilde{\mathbf{x}}$ the batch-effect-corrected cell associated with $\boldsymbol{x}$, that is, $\tilde{\mathbf{x}} = f_{\boldsymbol{\phi}}^{-1}\left(f_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{c}), \mathbf{c_{ref}}\right)$. In this manner, we get expressions $\tilde{\mathbf{x}}$ as if $\boldsymbol{x}$ were cell expressions from the reference batch. After training, we can also infer the missing values (NA) from the table using the mean latent expressions for all populations: it allows to refine batch correction for all markers.

Interpretability and population discovery

Understanding Scyan predictions

One important thing to notice is that $U_1 \mid (Z = \mathrm{z}), \ldots, U_M \mid (Z = \mathrm{z})$ are independent for every population z. It means that we can decompose $log\ p_{\mathbf{U}|Z=\mathrm{z}}(\boldsymbol{u}) = \sum_m log\ p_{U_m|Z=\mathrm{z}}(u_m)$, and we can gather all these terms into a matrix of scores $\left(log\ p_{U_m|Z=\mathrm{z}}(u_m)\right)_{z,m}$. The term $log\ p_{U_m|Z=\mathrm{z}}(u_m)$ can be interpreted as the impact of marker m towards the prediction of the population z for the latent cell expression $\boldsymbol{u}$. Based on that, we can interpret Scyan predictions for a group of cells $(\mathbf{x_i}, \mathbf{c_i})_i$ by transforming the cells into their latent expressions and then averaging the score matrices. The resulting matrix is typically displayed on a heatmap (Fig. 4d), and populations are sorted by their score (sum over a score matrix row). Note that, in the figure, each population score is scaled to make it easier to read.

Latent expressions

Considering a cell $\boldsymbol{x}$ and its covariates $\mathbf{c}$, its latent representation is $\boldsymbol{u} = f_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{c})$. The information of which marker is positive or negative is contained in $\boldsymbol{u}$. Indeed, $u_m \approx 1$ corresponds to a positive expression, while $u_m \approx -1$ represents a negative expression, whatever the marker m (i.e. expression levels for all markers are unified). Similarly, $u_m \approx 0$ is a mid-expression, and so on. We average the latent cell expressions over one population to obtain a latent expression at the population level. These population-level latent expressions can be displayed for one population (Fig. 4c) or for all of them at once (Fig. 4b).

Benchmark-related methods

Datasets used

We compare Scyan with the related works on three public mass cytometry datasets. One is from patients with acute myeloid leukemia [7] (AML, N = 104 184 cells, mass cytometry), one from bone marrow mononuclear cells [30] (BMMC, N = 61 725 cells, mass cytometry) and the last one from peripheral blood mononuclear cells (PBMCs) samples of peanut-allergic individuals [31] (POISED, N = 4178 320 cells, mass cytometry). The latter contains 30 samples, divided among seven batches, and under two different conditions (peanut stimulated or unstimulated). Finally, one flow cytometry dataset [32] has been used for debarcoding (N=100 000 cells). Manual gating has been performed in previous studies [7, 30, 31], providing ground truth labels to evaluate annotation models. Note that the unsupervised models listed below do not use these labels during training.

Compared models

We compared Scyan with six other annotation models: three knowledge-based models (ACDC [14], a baseline model defined by the authors of ACDC and MP [15]), one clustering method (Phenograph [7]) and two supervised models (LDA [11] and CyAnno [12]). We also benchmarked our model ability to correct batch effect to four models: Cydar [23], Combat [24], SAUCIE [18] and Harmony [22]. We used the POISED dataset on which we had seven biological batches, and we amplified the batch effect to complex the batch correction (see subsection 2.7.4).

Evaluation

We evaluated the models for the classification task using accuracy, macro-averaged F1-score and balanced accuracy [33]. The results are detailed in Fig. 2a. For the debarcoding task, the Silhouette score and the Calinski Harabasz Score were used. All the above metrics were implemented in Scikit-learn [34]. Concerning the batch-effect-correction task, we provide two metrics: the cell-type LISI (cLISI), which measures if the biological variability is kept, and the integration LISI (iLISI), which measures how well the batches overlap, i.e. if the batch-effect was corrected. We used the implementation from Harmony [22]. For more details, see supplementary Section 10.

Batch effect amplification

On the POISED dataset, we amplified the batch effect so that the benchmark becomes more complex. Let $\sigma_{BE} > 0$ a scale factor, and $b_1, \ldots, b_N \in [1 \ldots 7]$ the batch number associated with each of the N cells. Then, we sample seven matrices $\mathbf{S}_1, \ldots, \mathbf{S}_7 \in \mathbb{R}^{M \times M}$, whose elements are drawn from $\mathcal{N}(0, \sigma_{BE})$. For a cell i of expression $\boldsymbol{x}_i$, the batch-effect-amplified expression $\boldsymbol{x}_i'$ is multiplied by some batch-relative term: $\boldsymbol{x}_i' = (\boldsymbol{I}_M + \mathbf{S}_{b_i})\boldsymbol{x}_i$. In this equation, $\boldsymbol{I}_M$ is the identity matrix of size $M \times M$, and the multiplication operation is matrix multiplication. In practice, we use $\sigma_{BE} = 0.01$. Note that the UMAPs were computed on the cell-type-related markers and did not use cell-state markers.

Implementation details and hyperoptimization

We implemented our model using Python and the Deep Learning framework Pytorch [35]. We used between six and eight coupling layers whose multi-layer-perceptrons (s, t) have each between six and eight hidden layers depending on hyperparameter optimization. The hidden layer size can vary between 16 and 32. Model hyperoptimization can be performed using an unsupervised heuristic (see supplementary Section 9), but Scyan is robust to small changes of the main hyperparameter σ (see supplementary Section 5).

RESULTS

Scyan model

Scyan is composed of two core components: (i) $f_{\boldsymbol{\phi}}$, a neural network called normalizing flow, and (ii) a latent space on which a target distribution $\boldsymbol{U}$ is defined (Fig. 1b). This target distribution is a mixture of distributions—one per population—built using prior biological knowledge about the cell types. This knowledge is provided as a table: for all populations, each expected marker expression is given or left unknown (more details in supplementary Section 6). This table is then used to mathematically define the target distribution $\boldsymbol{U}$. Also, the latent space (on which $\boldsymbol{U}$ is defined) has the same dimension as the original space; therefore, each marker has its corresponding latent expression.

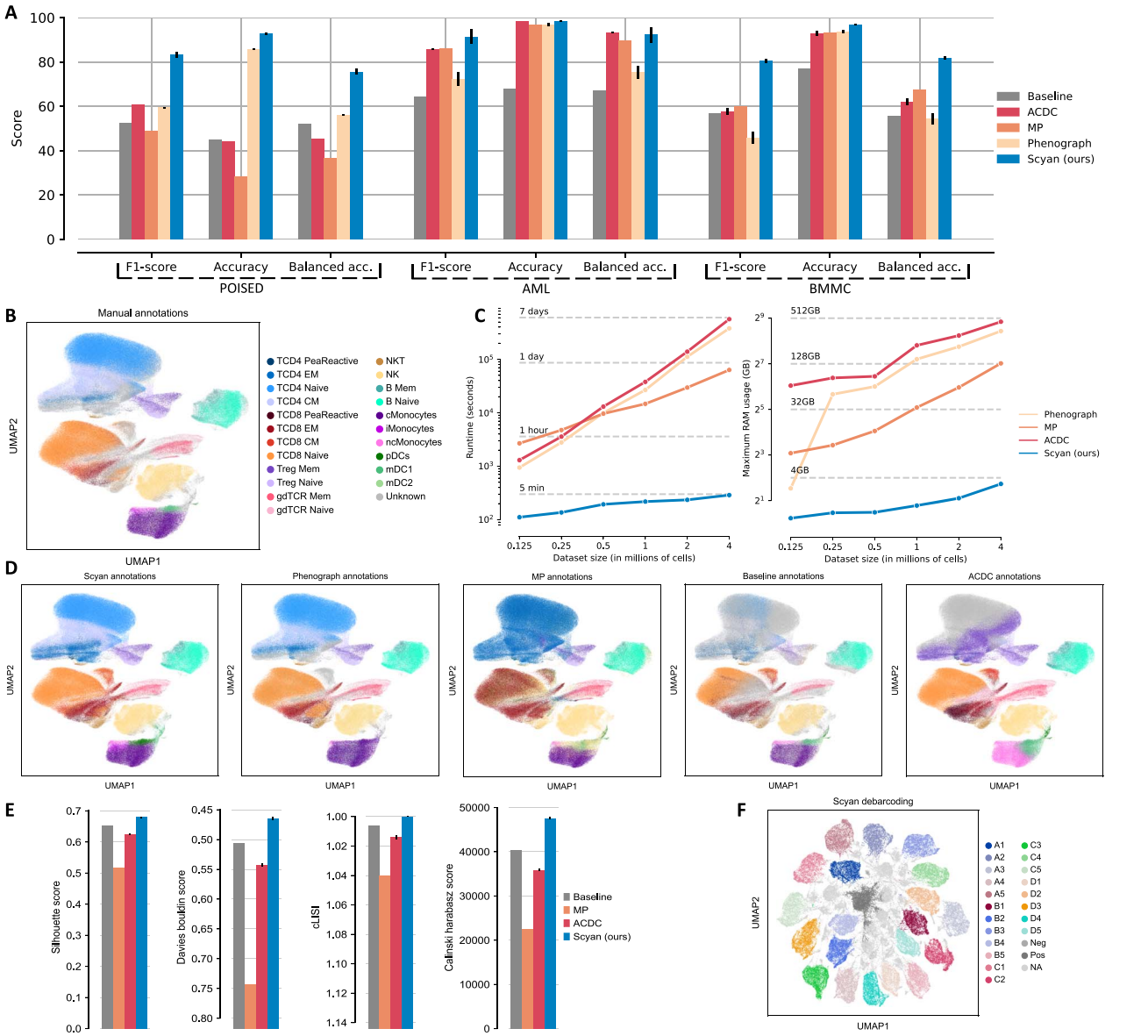

Figure 2. Comparison with state-of-the-art unsupervised methods. A, Performance comparison of Scyan and four other unsupervised methods on three datasets (POISED, AML, BMMC) using three metrics for each. **B**, UMAP [39] representing the manually annotated populations on the POISED dataset. **C**, Models runtime comparison (left) and RAM usage comparison (right) over multiple dataset sizes. **D**, UMAPs representations of the annotations of all five models on the POISED dataset. **E**, Unsupervised metrics for the debarcoding task. **F**, UMAP representing Scyan debarcoding. Cells that did not correspond to any desired barcode were left unclassified (NA).

The purpose of the normalizing flow is to learn an invertible mapping between the actual marker expression distribution and the target $\boldsymbol{U}$. By mapping marker expressions to a biologically defined latent space, we force the transformation to provide latent expressions on a scale that is shared for every marker, going from negative (-1) to positive (+1). These latent marker expressions are meant to be free of batch effect or any non-biological factor. By the design of $f_{\boldsymbol{\phi}}$ and of the objective function, the normalizing flow is not allowed to make huge space distortions, which helps preserve the biology. After learning the model parameters $\boldsymbol{\phi}$, annotations are performed on the latent space. We annotate a cell by choosing the population distribution whose likelihood is the highest for the cell latent representation. If a cell latent representation does not correspond to any component of the mixture, then the cell remains unlabelled, but population discovery can be run afterward to annotate it eventually (see Fig. 4).

Scyan provides a better and faster annotation than unsupervised methods

Classification metrics comparison

We evaluated Scyan to four unsupervised or semi-supervised models for the classification task (ACDC and its baseline [14], MP [15] and Phenograph [7]), on three public datasets, and over three different metrics (see subsection 2.7 for more details). The tests show that Scyan outperforms the other models. In particular, Scyan is about 20 points higher than the other models on POISED and BMMC for the F1-score and the balanced accuracy, which is explained by the capacity of Scyan to detect better small populations (Fig. 2a). On these datasets, multiple populations represent less than 1% of the total number of cells, making these populations more difficult to detect and label. Yet, small population annotations can still be essential, and thus so is Scyan's capacity to detect them. Also, the gap between Scyan and the

other models is more stringent for POISED, showing our model's ability to better annotate large, complex datasets with batch effect.

Computational speed, scalability and memory usage

To demonstrate the scalability of Scyan on large datasets, we compare the execution times and the random access memory (RAM) usage of the different algorithms over multiple dataset sizes (Fig. 2c). The different sizes were obtained by sub-sampling the POISED dataset, for various sample sizes from 125 000 to 4 million cells. All experiments were run using the same hardware; in our case, CPUs only (i.e. no GPU acceleration, even though Scyan can use GPUs, supplementary Section 12). On $N = 4$ million cells, Scyan runs in 5 min, while ACDC/MP/Phenograph need between 1 and 7 days. Scyan scales well to large datasets, as shown by the low slope on Fig. 2c. Concerning RAM consumption, Scyan uses less than 4GB of RAM, which means it can be run on any standard laptop. In comparison, ACDC, MP and Phenograph all require between 128 and 512GB of RAM, which is only available on large computer clusters. See supplementary Section 8 for comments about supervised models runtime.

Comparison for barcoding deconvolution

Barcoding is a method that reduces the batch effect and data variability by allowing the processing of multiple cell samples together, each cell sample being labeled—or barcoded—with a unique combination of antibodies. This protocol requires (i) the dedication of a few markers to make barcodes and (ii) the identification of each cell sample based on its barcode. The latter task, called debarcoding [32], can also be expressed as a knowledge-based annotation task. In this situation, we annotate samples instead of populations, and the expert knowledge required for this task simply corresponds to the known barcodes. Figure 2e shows that Scyan outperforms ACDC, MP and the baseline on a public dataset with 20 barcodes and six markers [32]. The UMAP on Fig. 2f shows a clear separation of the different barcodes, with some small residual clusters (not to be considered) corresponding to non-existing barcodes. The UMAPs corresponding to the debarcoding of the other methods can be found in supplementary figure 7.

Scyan corrects batch effect

A batch effect is a phenomenon that induces data variability due to non-biological factors such as the use of a different antibody or slightly different cytometer settings. In practice, these factors may introduce variability that interferes with the analysis and can lead to confusion, over-interpretation and difficulties in annotating populations. To tackle this issue, Scyan can align the distribution of different batches (see methods subsection 2.5). Classically, batch effect correction is performed before annotation, but our method allows for correcting it at the same time as the annotation. Taking into account the batch helps Scyan to annotate the populations better. Figure 3a shows the batch effect we want to correct (from the amplified POISED dataset, see subsection 2.7.4). The next figures, i.e. Fig. 3b/c, show that Cydar's and Combat's corrections are very limited, even though they keep the biological variability. SAUCIE provides the best iLISI, so it mixes well the different batches, but it also removes most of the biological variability (high cLISI). On the opposite, Scyan and Harmony successfully remove the batch effect while preserving the biological variability. Another benchmark on POISED without amplification can be found in supplementary Section 7.

Scyan latent space provides interpretability and helps population discovery

Scyan's latent space (see subsection 2.6.2) is key for interpretability. Specifically, it enables the understanding of the Scyan annotation process, and also helps to quickly characterize new populations of cells to improve the annotation. We illustrate population discovery on the POISED dataset. For this purpose, we show that we could annotate six populations that were missed during manual gating, such as differentiated effector T cells [36] (TCD8 TEM) and $\gamma\delta$TCR CD16+ cells (Fig. 4a shows the populations we had before running population discovery). To demonstrate two different ways of discovering new populations, we show that we can (i) annotate more precise populations among known ones using Leiden [8] sub-clustering and Scyan latent space (see subsection 2.6.2 and Fig. 4b), or (ii) discover a population that was missing from the table (see Fig. 4c/d). For the latter case, we show that cells corresponding to a population that is not in the table will be annotated as 'unknown' by Scyan, and its interpretability (subsection 2.6.1) will help characterize this missing population. One advantage of Scyan is its table flexibility: the new populations, once characterized, can be added to the knowledge table, and Scyan will then be able to annotate them. This is shown by Fig. 4e that summarizes all the populations we annotated.

Understanding Scyan annotation process

Scyan annotation process can be interpreted on one cell or a group of similar cells (see methods subsection 2.6.1). Typically, we can select one population and interpret Scyan's annotation process on this group of cells. First, we can display all the latent marker expressions corresponding to these cells (Fig. 4c). It opens up a new simple way to understand which marker is positive or negative at a glance. Indeed, the latent space has a shared scale for all markers, and a simple scale indicates expression levels between Negative (-1) and Positive (+1). But mostly, we can provide a confidence measure (or log-probability) to belong to each cell type (Fig. 4d, left column). Then, for each population, we can decompose the population probability as a sum of marker impact (expressed as log-probabilities). It allows explaining which marker contributed the most to the probability of each of the populations. This interpretability property can be used to discover new populations (see subsection 3.4.2).

Annotating unknown populations

Sometimes, users may forget some populations in the table given to Scyan, and the corresponding cells will be left unclassified. Since every population from the POISED dataset was already described, we decided to remove two populations from the table provided to Scyan (non-classical and intermediate monocytes) to see if we could retrieve them. As shown in Fig. 4a on the red magnifying glass, cells corresponding to these populations were annotated as being 'Unknown' (light gray color). We can further investigate these 'Unknown' cells to retrieve their corresponding population, see Fig. 4c/d. To summarize, the process is the following: (i) we choose a group of cells that were unclassified by Scyan, (ii) we quickly characterize these cells using Scyan latent space and (iii) we update the table to annotate them. Combining Fig. 4c and Fig. 4d provides a description of the Scyan annotation process that is understandable by humans, through decomposition into confidence measures by marker and by population.

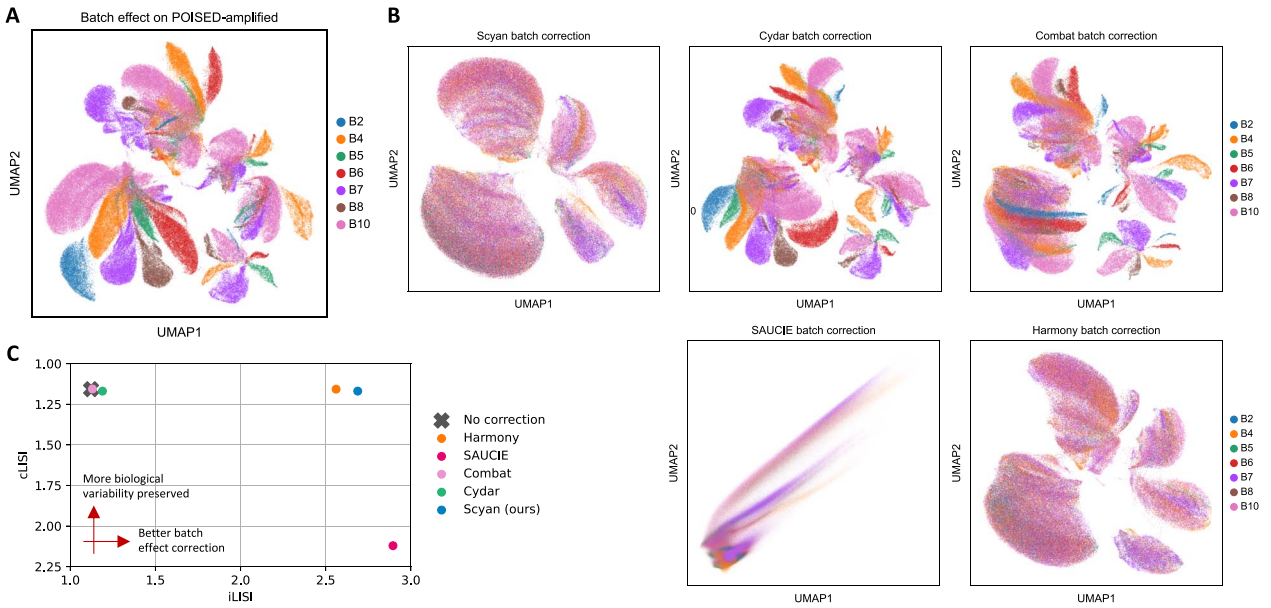

Figure 3. Batch-effect correction on the POISED dataset with batch-effect amplification. A, UMAP showing the seven different batches (before batch effect correction). The batch effect is visible since different batches form separated clusters. **B**, Batch-effect correction of Scyan, Cydar, Combat, SAUCIE and Harmony. A superposition of all batch distributions can show a good batch effect correction. **C** Batch-effect correction metrics. A low cLISI (at the top of the figure) denotes good cell-type variability preservation, while a high iLISI (on the right of the figure) denotes better batch mixing.

Comparison with supervised models

Performances on POISED

Two supervised models (LDA [11] and CyAnno [12]) were compared with Scyan on the POISED dataset. Figure 5a shows the performances of the three models on POISED. Even though the benchmark is in favor of supervised models (since they use labels), Scyan still has a higher performance. Also, since we are comparing an unsupervised method with supervised methods, comparing results with manual gating is biased. For this reason, we additionally compare the models' agreement (in a pairwise manner) using Cohen's Kappa score (Fig. 5b). We see that LDA and Scyan are the two models whose agreement is the highest, even higher than LDA and CyAnno (while they are two supervised methods trained for the same task). Concerning the disagreement between Scyan and the manual gating, we show in the supplementary figure 9a/b that most disagreement is partly due to the subjective delimitation boundaries between non-classical/intermediate/classical monocytes. As shown by supplementary figure 9a/b, although Scyan is also properly annotating these populations, it has slightly different decision boundaries than manual gating, which still significantly decreases F1-score or balanced accuracy. It emphasizes again the importance of comparing the agreement between all models instead of only comparing with manual gating.

Annotations of the ungated cells

One key aspect of annotation models is whether they annotate more cells than traditional manual gating. This can enhance the biomarker discovery and provides higher statistical significance during post-annotation analyses. We compared the number of cells annotated by Scyan, LDA [11] and CyAnno [12] on POISED, and demonstrated that Scyan annotates more cells than CyAnno, and a similar amount of cells to LDA (Fig. 5d). Indeed, CyAnno annotated 15% of the ungated cells, Scyan 97% and LDA was set up to annotate all cells. Moreover, Scyan annotated six more populations compared with CyAnno and LDA. Most importantly, we show by back gating that the annotated cells were properly classified (see supplementary figure 9c). Indeed, one limitation of supervised models such as LDA or CyAnno is that they cannot annotate new populations, i.e. they are limited to manually gated populations. Although knowledge-based annotation models (like ours) are limited to populations from the provided table, the table can be easily extended. This property is, therefore, crucial for population discovery with Scyan.

Usage for biomarker discovery

The POISED dataset is decomposed into two conditions: peanut-stimulated samples, and unstimulated ones. We try to find biomarkers that are differentially expressed on peanut-stimulated samples. For that, for all models, biomarkers were extracted, and we ran Wilcoxon signed-rank tests [37] between the two conditions (we assume patients are independent). On Fig. 5c, we sorted the biomarkers by P-value for all models, and we display the $-log_{10}(P - value)$ of the first 400 biomarkers. We show that Scyan extracts more biomarkers of higher significance. Note that a similar process could be run for other clinical conditions, such as the patient response to treatment. Having more significant biomarkers means it will be easier to predict such an outcome.

DISCUSSION

We have introduced Scyan, a multi-purpose neural network for cytometry annotation, batch-effect removal, debarcoding, and population discovery. It provides a robust and broad pipeline to analyze cytometry cell populations, monitor their dynamics over time and compare the populations' proportions among patients. Such analyses can help discover biomarkers or specific populations characteristic of response to treatment, for example. Scyan can perform fast and automatic annotations for these large datasets and correct potential batch effects. Some studies use barcoding to reduce the batch effect, hence requiring a debarcoding step that Scyan can also perform. Thus, Scyan is suitable for various types of cytometry projects and does not rely on any extra cytometry analysis library.

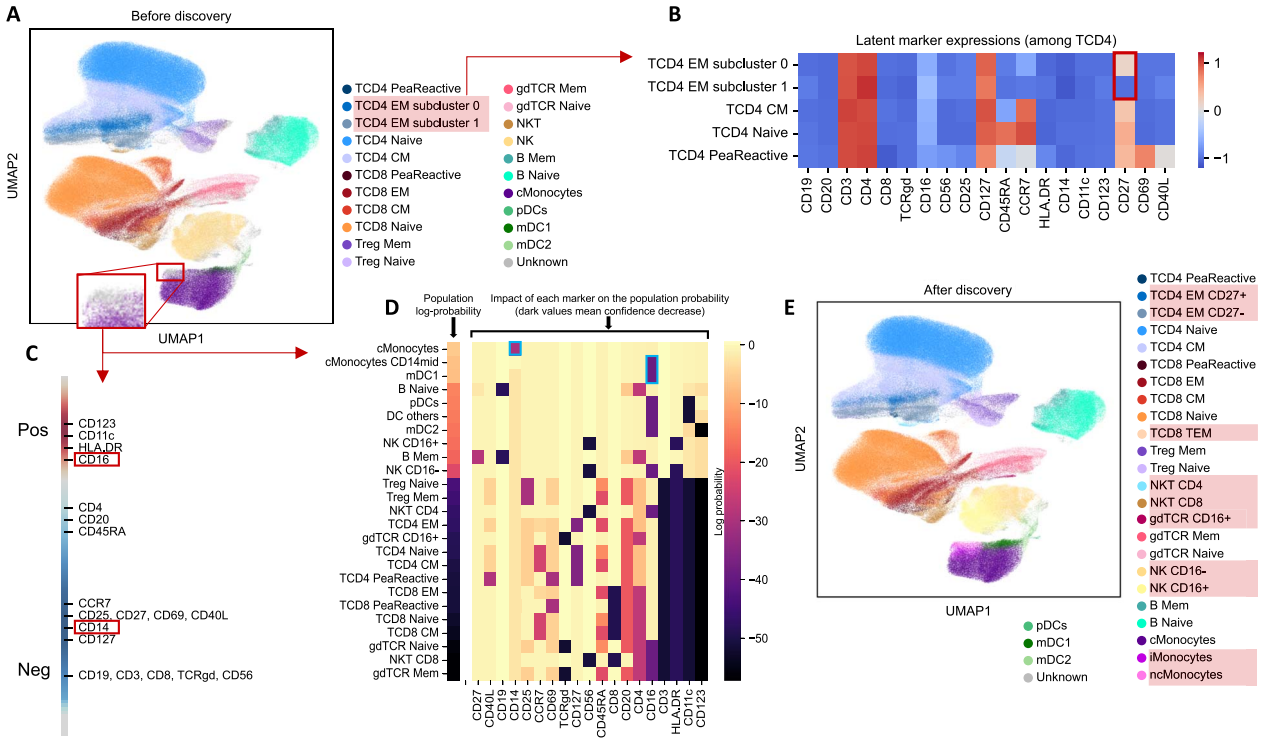

Figure 4. Interpretability and population discovery with Scyan. A, UMAP on POISED before population discovery. Two subclusters of TCD4 EM cells have been defined and characterized in (b). Also, intermediate and non-classical monocytes were removed from the knowledge table to show that we can retrieve existing populations that are missing from the table: as shown by the red magnifying window, Scyan annotated these cells as unknown, and (c/d) helped to characterize them as intermediate and non-classical monocytes. **B**, Latent space expressions for subsets of TCD4 cells, displayed on a heatmap. We can easily see the difference between the two clusters: one is CD27+, the other CD27-. The subclusters were obtained by running Leiden [8] clustering on the TCD4 EM populations. **C**, Scyan helps characterize the unknown cells defined in (a) by showing its latent marker expressions, displayed on a shared scale going from Negative to Positive expressions. We can see, for instance, that these 'Unknown' cells are CD16 positive and CD14 negative. **D**, Scyan provides soft predictions for all populations (first column), i.e. a log probability is associated with each population. Then, each population probability is decomposed into a sum of marker impact on the probability (one row). The dark colors indicate that the corresponding marker expression decreased the population probability of the corresponding row. According to the first column, we see that the first guesses of Scyan are classical monocytes (both CD14 high and mid) and mDC1. Then, we look at the three corresponding rows: they correspond to the confidence of Scyan for these populations decomposed by markers. For instance, we see that the expression of CD14 (which is negative, according to (c)) decreased Scyan's confidence toward the prediction of classical monocytes. Thus, based on the first row, we can conclude that the 'Unknown' cells are similar to classical monocytes but are CD14- instead of CD14+, and that these cells are non-classical or intermediate monocytes. Similarly, the third row shows that they look like mDC1 cells but with a CD16+ expression instead of negative (again, (c) was needed to see the expression of CD16). Once more, we indeed conclude that these cells are non-classical/intermediate monocytes, and they can be added back to the table for the annotation. **E**, UMAP of Scyan annotations after population discovery. The red boxes denote new populations compared with (a): such populations were found using subclustering and analyzing Scyan's latent space.

Scyan annotates populations without needing labels and, therefore, can fully replace manual gating. It uses a marker-population table containing expert knowledge. The literature offers many resources and existing knowledge to construct such tables, but some marker expressions remain unexplored. For this reason, we offer the possibility to handle 'not applicable' values inside the table and, to improve flexibility, intermediate expressions such as 'mid' or 'low'. In the case where the panel remains not well known enough to build the input table, Scyan can help discover new populations: analyses start by annotating large populations and then gradually target smaller and smaller cell types. Also, with the increasing usage of cytometry, we expect the marker knowledge to improve over time, reinforcing Scyan performance and ease of use.

In terms of model architecture, normalizing flows are a recent and promising field of research in generative models. They benefit from interesting mathematical properties such as (i) exact likelihood computation and (ii) invertibility. We show that normalizing flows can be used to leverage marker knowledge in a biologically natural way, providing interpretability. Indeed, the network invertibility allows switching between the measured marker expressions and their latent expressions. In this space, all latent markers have unified expression ranges, which is convenient for human analysis, especially for population discovery. It also makes the model reliable and transparent to biologists, which can help build trust toward the model annotations and validate them. Moreover, normalizing flows are smooth transformations that control how the space is deformed, ensuring that we do not alter the biological meaning behind marker expressions. At the same time, it benefits from the expressiveness and flexibility of deep neural networks. In fact, the usage of neural networks allows adding additional terms in the loss function to handle the batch effect, which is naturally corrected with the network invertibility. Eventually, we can further push the usage of these convenient mathematical properties for other tasks in single-cell analysis, for instance, single-cell RNA sequencing data or imaging mass cytometry data [38].

Overall, Scyan promises to be powerful in several ways. Scyan is robust for identifying unique cell populations which, like dendritic cells or stromal cells, for instance, are underrepresented in

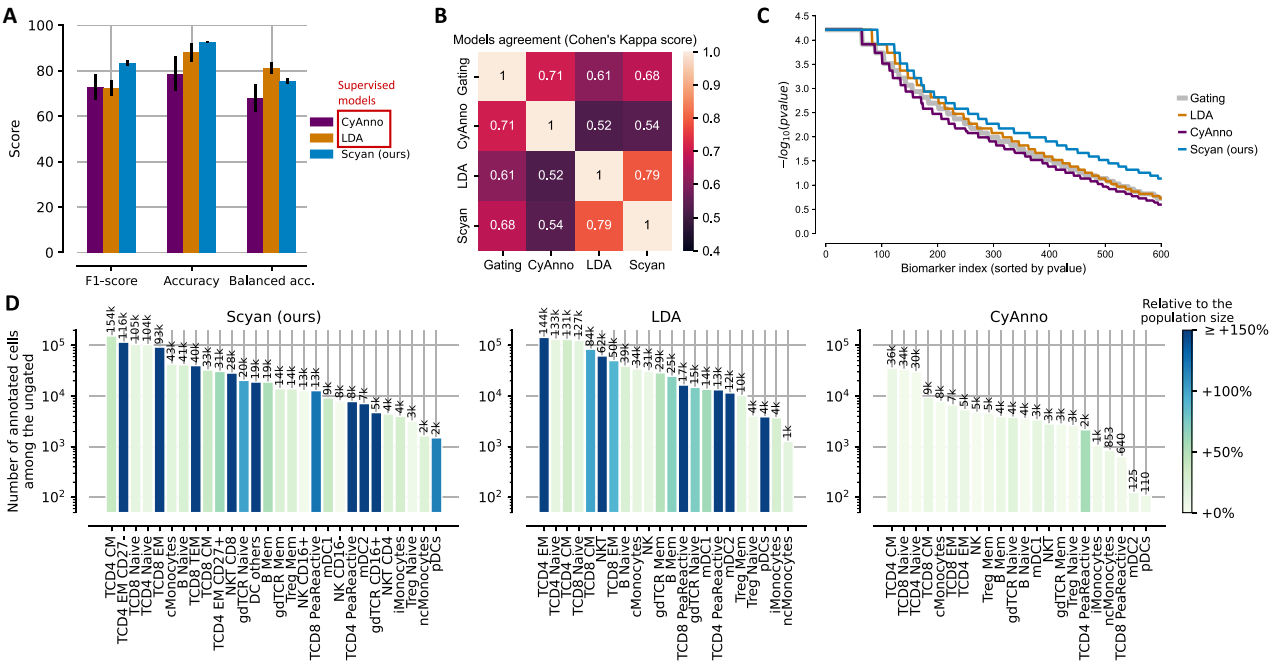

Figure 5. Comparison with supervised models. The last two figures were done after Scyan's population discovery. **A**, Metrics on POISED. Note that among the three methods tested, CyAnno and LDA are supervised methods (i.e. using training labels from manual gating). **B** Heatmap representing pairwise models agreement using Cohen's Kappa score. A high value indicates a better agreement (the highest value is 1). **C**, After annotation, one can extract biomarkers and run differential expression relative to a clinical condition. Here, we show the significance of the biomarkers for all methods (higher is more significant). **D**, Number and percentage of cell types that were annotated by the models among the ungated ones.

complex biological samples although they play essential roles in shaping disease resolution or progression. The ability of Scyan to analyze large datasets, in a fast and accurate manner, will be essential for this task and open up the possibility of unraveling the heterogeneity of such rare populations of cells. Mostly, these analyses can be made on all types of cytometers, and whatever the presence or not of batch effect.

Key Points

- We introduce Scyan, an automatic and unsupervised method for annotating cell populations in cytometry; it replaces manual gating, which is time-consuming, challenging, non-reproducible and sensitive to batch effect.
- Scyan annotates cell types using only prior expert knowledge about the cytometry panel (i.e. no label needed) and uses a deep generative model called normalizing flow.
- Scyan is fast, scalable, interpretable and significatively more accurate than existing state-of-the-art methods.
- It also has an integrated batch-effect correction and can run debarcoding and population discovery.
- The Scyan package is open-source, well-documented, simple to use and integrates with core frameworks.

ACKNOWLEDGMENTS

This work is supported by Prism – National Precision Medicine Center in Oncology funded by the France 2030 program and the French National Research Agency (ANR) under grant number ANR-18-IBHU-0002.

AUTHOR CONTRIBUTIONS STATEMENT

I.P. conducted experiments for a clinical study (private data), which J.R.M. and F.A. are in charge of. N.B., F.A. and C.A.D. detailed the need for a new methodological method to analyze these data. Q.B. conceived the mathematical model, wrote the source code, published the library and wrote the paper. All authors reviewed the paper. P.H.C. supervised the study and reviewed the mathematical method. F.A. and P.H.C acquired the grant to fund the project.

CODE AVAILABILITY

The code developed in this article is available as an open-source Python package, accessible on Github at https://github.com/MICS-Lab/scyan. The code used to run and compare the other algorithms is also available at https://github.com/quentinblampey/cytometry_benchmark.

DATA AVAILABILITY

The four datasets and knowledge tables considered in this paper are public and accessible at https://github.com/MICS-Lab/scyan_data

REFERENCES

1. Behbehani GK. Immunophenotyping by mass cytometry. *Methods Mol Biol* 2019; **2032**:31–51.
2. Spitzer MH, Nolan GP. Mass cytometry: single cells. *Cell* 2016; **165**:780–91.
3. McKinnon KM. Flow cytometry: an overview. *Curr Protoc Immunol* 2018; **120**:5.1.1–11.

4. Newell EW, Cheng Y. Mass cytometry: blessed with the curse of dimensionality. *Nat Immunol* 2016; **17**:890–5.
5. Staats J, Divekar A, McCoy J, et al. Guidelines for Gating Flow Cytometry Data for Immunological Assays. In: McCoy, Jr, J. (ed) *Methods in molecular biology*, Vol. **2032**. Clifton, NJ: Humana, New York, NY, 2019, 81–104.
6. Aghaeepour N, Finak G, FlowCAP Consortium, et al. Critical assessment of automated flow cytometry data analysis techniques. *Nat Methods* 2013; **10**:228–38.
7. Levine JH, Simonds EF, Bendall SC, et al. Data-driven phenotypic dissection of AML reveals progenitor-like cells that correlate with prognosis. *Cell* 2015; **162**:184–97.
8. Traag VA, Waltman L, Eck NJ VAN. From Louvain to Leiden: guaranteeing well-connected communities. *Sci Rep* 2019; **9**: 5233.
9. Qiu P, Simonds EF, Bendall SC, et al. Extracting a cellular hierarchy from high-dimensional cytometry data with SPADE. *Nat Biotechnol* 2011; **29**:886–91.
10. Li H, Shaham U, Stanton KP, et al. Gating mass cytometry data by deep learning. *Bioinformatics* 2017; **33**:3423–30.
11. Abdelaal T, Unen V, Höllt T, et al. Predicting cell populations in single cell mass cytometry data. *Cytometry A* 2019; **95**: 769–81.
12. Kaushik A, Dunham D, He Z, et al. CyAnno: a semi-automated approach for cell type annotation of mass cytometry datasets. *Bioinformatics* 2021; **37**:4164–71.
13. Liu P, Liu S, Fang Y, et al. Recent advances in computer-assisted algorithms for cell subtype identification of cytometry data. *Front Cell Dev Biol* 2020; **8**:234.
14. Lee H-C, Kosoy R, Becker CE, et al. Automated cell type discovery and classification through knowledge transfer. *Bioinformatics* 2017; **33**:1689–95.
15. Ji D, Nalisnick E, Qian Y, et al. Bayesian Trees for Automated Cytometry Data Analysis. In: Doshi-Velez F, Fackler J, Jung K, Kale D, Ranganath R, Wallace B, Wiens J (eds) *Proceedings of the 3rd Machine Learning for Healthcare Conference*. Cambridge MA, USA: PMLR, 2018. p. 465–83.
16. Lopez R, Regier J, Cole MB, et al. Deep generative modeling for single-cell transcriptomics. *Nat Methods* 2018; **15**:1053–8.
17. Zhang AW, O'Flanagan C, Chavez EA, et al. Probabilistic cell-type assignment of single-cell RNA-seq for tumor microenvironment profiling. *Nat Methods* 2019; **16**:1007–15.
18. Amodio M, van Dijk D, Srinivasan K, et al. Exploring single-cell data with deep multitasking neural networks. *Nat Methods* 2019; **16**:1139–45.
19. Rezende DJ, Mohamed S. Variational inference with normalizing flows. *International conference on machine learning* 2015;1530–38.
20. Papamakarios G, Nalisnick E, Rezende DJ, et al. Normalizing flows for probabilistic Modeling and inference. *The Journal of Machine Learning Research* 2021;**20**(1):2617–80.
21. Izmailov P, Kirichenko P, Finzi M, Wilson AG. Semi-supervised learning with normalizing flows. *International Conference on Machine Learning*. Cambridge, MA, USA: PMLR, 2020. p. 4615–30.
22. Korsunsky I, Millard N, Fan J, et al. Fast, sensitive and accurate integration of single-cell data with harmony. *Nat Methods* 2019; **16**:1289–96.
23. Lun ATL, Richard AC, Marioni JC. Testing for differential abundance in mass cytometry data. *Nat Methods* 2017; **14**:707–9.
24. Johnson WE, Li C, Rabinovic A. Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics* 2007; **8**:118–27.
25. Parks DR, Roederer M, Moore WA. A new "Logicle" display method avoids deceptive effects of logarithmic scaling for low signals and compensated data. *Cytometry A* 2006; **69A**: 541–51.
26. Dinh L, Sohl-Dickstein J, Bengio S. Density estimation using real NVP arXiv:1605.08803 [cs, stat]. 2017.
27. Ackley DH, Hinton GE, Sejnowski TJ. A learning algorithm for boltzmann machines. *Cognit Sci* 1985; **9**:147–69.
28. Ficler J, Goldberg Y. Controlling linguistic style aspects in neural language generation. In: Brooke Julian, Thamar Solorio Thamar, Koppel Moshe (eds) *Proceedings of the Workshop on Stylistic Variation*. Copenhagen, Denmark: Association for Computational Linguistics, 2017. p. 94–104.
29. Kingma DP, Ba J. Adam: a method for stochastic optimization http://arxiv.org/abs/1412.6980. 2017. https://doi.org/10.48550/arXiv.1412.6980.
30. Bendall SC, Simonds EF, Qiu P, et al. Single-cell mass cytometry of differential immune and drug responses across a human hematopoietic continuum. *Science* 2011; **332**:687–96.
31. Chinthrajah RS, Purington N, Andorf S, et al. Sustained outcomes in oral immunotherapy for peanut allergy (POISED study): a large, randomised, double-blind, placebo-controlled, phase 2 study. *The Lancet* 2019; **394**:1437–49.
32. Zunder ER, Finck R, Behbehani GK, et al. Palladium-based mass tag cell barcoding with a doublet-filtering scheme and single-cell deconvolution algorithm. *Nat Protoc* 2015; **10**:316–33.
33. Jiao Y, Du P. Performance measures in evaluating machine learning based bioinformatics predictors for classifications. *Quant Biol* 2016; **4**:320–30.
34. Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: machine learning in python. *J Mach Learn Res* 2011; **12**:2825–30.
35. Paszke A, Gross S, Massa F, et al. PyTorch: an imperative style, high-performance deep learning library http://arxiv.org/abs/1912.01703. 2019. https://doi.org/10.48550/arXiv.1912.01703.
36. Sallusto F, Lenig D, Förster R, et al. Two subsets of memory T lymphocytes with distinct homing potentials and effector functions. *Nature* 1999; **401**:708–12.
37. Wilcoxon F. Individual comparisons by ranking methods. *Biometrics* 1945; **1**:80–3.
38. Chang Q, Ornatsky OI, Siddiqui I, et al. Imaging mass cytometry. *Cytometry A* 2017; **91**:160–9.
39. McInnes L, Healy J, Melville J. UMAP: uniform manifold approximation and projection for dimension reduction arXiv:1802.03426 [cs, stat]. 2020.