Optimal Transport Feature Alignment for Cross-Domain Cell Image Classification

Anonymous authors
Paper under double-blind review

Abstract

Classifying single-cell images across datasets collected under varying imaging conditions remains a fundamental challenge in computational cytology. Differences in microscope hardware, illumination, staining, and media introduce substantial domain shifts, leading to poor generalization of models trained on a single dataset—even when underlying cell types are shared. To address this, we propose a robust pipeline that integrates (i) multi-encoder feature extraction using foundation models, (ii) unsupervised identification of shared cell types between datasets, (iii) alignment of embedding spaces using graph-regularized optimal transport (OT), and (iv) final classification via ensemble-based voting across encoders. By explicitly filtering unmatched classes and transporting only comparable subsets, our method mitigates the impact of non-overlapping cell types and structural misalignment. Experimental results across multiple real-world cell image datasets demonstrate improved accuracy (+20% in average over all datasets considered in the paper when adding class filtering and OT), robustness to domain shift, and reliable performance in both fully unsupervised and low-label regimes—without requiring fine-tuning or retraining on the target domain.

1 Introduction

Cell image analysis is a challenging task, which makes it difficult to develop robust machine learning (ML) algorithms for classifying cells. Indeed, in medical imaging, and especially in cytology, data are often fragmented across sites, cohorts, and acquisition protocols, leading to substantial distribution shifts. This problem is compounded by the fact that data collection frequently scales much faster than expert labeling, so that large volumes of images remain unlabeled. As a result, only small annotated datasets are available, making it necessary to exploit additional unlabeled datasets acquired under varying conditions (e.g., different devices, protocols, or annotators) to enable transfer to a new target dataset. This imbalance between available data and available labels constitutes a major challenge for robust cell analysis.

One possible route to address this challenge is to train domain-specific foundation models (Bommasani et al., 2021). These models are pre-trained on a large variety of images across many domains (not only medical imaging), and, thus, make them generalizable. Yet such models inevitably have a limited scope: they tend to perform well within their (broad) training distribution, but covering the full spectrum of imaging conditions and ensuring that new data remain "in-distribution" at runtime is challenging in practice.

To illustrate this observation, we invite the reader to look at Fig. 1, where we show a typical transfer example. The query cell (first column) comes from the RAABIN white blood cell dataset, assumed to be unlabeled. To annotate it, we use the labeled Peripheral Blood Cells (PBC) dataset, collected under different conditions. The first row shows the predicted cell class using DinoBloom (Koch et al., 2024), a foundation model specifically tailored for cell images and trained on over 380,000 white blood cell images from diverse datasets. Since DinoBloom was trained on RAABIN but not on PBC, the query lies within its training domain, while the reference cells do not. One sees that, in this precise case, the query cell, a lymphocyte, is misclassified as an erythroblast, because in DinoBloom's feature space it appears closer to PBC erythroblasts (last three columns, first row).

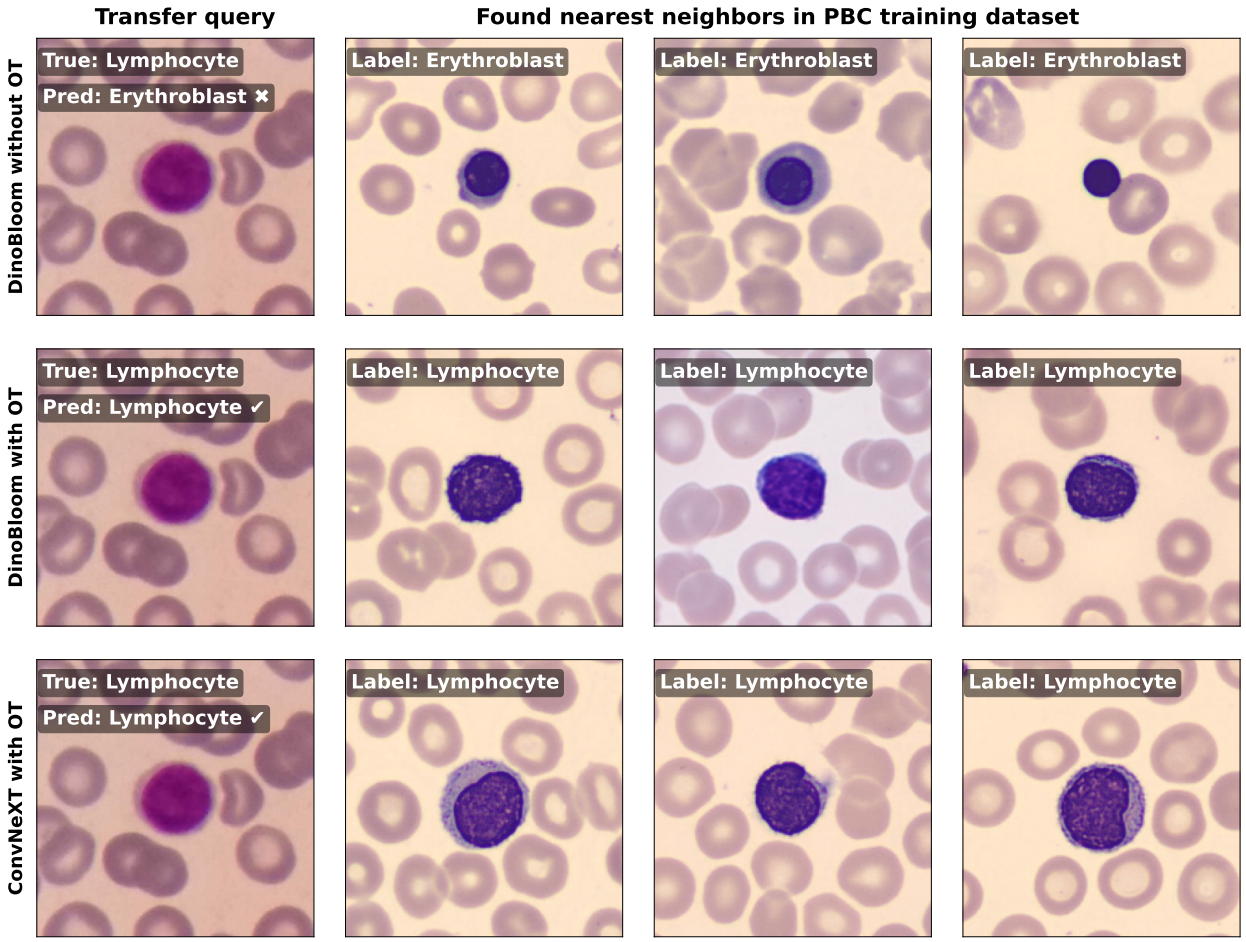

Figure 1: Transfer query for one cell in the RAABIN white blood cell dataset. The targeted cell is a lymphocyte that we assume to be unlabeled. To label it, we use the annotated Peripheral Blood Cells (PBC) dataset to train a k-NN classifier. The nearest neighbors (last 3 columns) found by the k-NN search are used to label the unknown cell. The first row corresponds to the classification process using DinoBloom, a pre-trained foundation model specifically tailored for cell images. As one can see, the label prediction (erythroblast) is not correct in this case. The second row corresponds to the same model but with optimal transport (OT) on top of it, a mathematical tool that we use in our pipeline to align training and transfer data domains (see text). This preprocessing step, together with the same pretrained mode and classifier, now leads to the correct prediction for this cell (i.e., a lymphocyte). The last row corresponds to the same unknown cell but for which we used ConvNeXT (and not DinoBloom) to highlight that our approach also gives the correct prediction using a foundation model not specifically designed for cell imaging but with OT on top of it.

Therefore, while DinoBloom covers a large spectrum of imaging conditions and demonstrates an impressive ability to generalize across many transfer cell imaging tasks, out-of-distribution (OOD) scenarios still remain inevitable: no matter how extensive the training spectrum, there will always be unseen conditions where even the most powerful foundation models fail to transfer reliably. As a consequence, fine-tuning or retraining foundation models for each new dataset can alleviate domain shifts to a certain extent but, these methods will always be constrained by a limited "in-distribution" domain. On top of that, curating and aggregating sufficiently diverse datasets, and training expressive domain-specific models from scratch, requires substantial resources that are rarely available (especially in biomedical research).

In this paper, we propose an orthogonal direction: rather than enlarging training datasets or retraining models to cover an ever-broader spectrum of conditions, we take foundation models as they are and correct training feature representations with transfer-sample embeddings, aligning feature spaces and effectively mapping out-of-distribution samples into the in-distribution transfer space.

More concretely, to correct for domain shifts and align feature representations across datasets we use optimal transport (OT) (Villani, 2003; Li et al., 2022; Gong et al., 2022; Mémoli, 2011a;b). This approach has proven effective in addressing this challenging problem across various domains, and particularly in biological applications (Thual et al., 2022; Breeur et al., 2024; Demetci et al., 2022b;a). Indeed, OT can be used to transport one distribution into another, thus allowing them to share a common ground space and align their data domains. Therefore, we believe that OT is an appealing solution to align medical cell datasets and improve cell classification.

To the best of our knowledge, no studies have applied OT to align data spaces across cell imaging datasets, while domain shifts commonly encountered in biomedical imaging are significant. Our strategy is the following: we use OT to map cell embeddings from a labeled dataset – acquired under different experimental conditions and/or imaging devices – into the embedding space of a transfer dataset. These mapped representations can then be used to train a classifier for labeling the unknown cells. By aligning both training and test datasets into a common representation space, OT mechanically improves classification accuracy and robustness to domain shifts. Unlike other approaches that require large labeled datasets, our method makes it possible to map the feature representations of only a few annotated samples – as it is often the case in medical imaging – onto another, potentially much larger, dataset and reliably scale up annotations.

The results shown in the second and third rows of Fig. 1 are a preliminary illustration that aligning feature representations of foundation models without the need to add more datasets to cover larger domains can be highly beneficial. The second row shows indeed that combining DinoBloom with our OT alignment strategy now yields a correct classification. The third row demonstrates that even more general foundation models, such as ConvNeXT – despite not being trained on medical images – perform better with our approach than cell-specific models that lack cross-dataset feature alignment.

On top of that, we propose to not only align embedding spaces across datasets, but also their cell domain. As a preprocessing step, we identify shared (overlapping) classes between the training and the transfer datasets and filter out unmatched cell types to only transport comparable subsets from the training to the transfer set. By filtering noisy and OOD samples, OT only focuses on common patterns, thereby improving both feature representation alignment and classification.

Finally, to enhance the robustness of our approach, we also propose to rely not only on one foundation model, but, on the contrary, to exploit the potential of diverse foundation models with a multi-encoder-based method (Kalweit et al., 2024). As a post-processing task, we aggregate predictions from multiple foundation models using a weighted voting scheme. This ensemble approach mitigates errors from any single model, leverages varying performance across datasets, and avoids the need to fine-tune model selection. As a result, it achieves more reliable predictions, better adaptability, and improved generalization to new acquisition devices and experimental conditions.

Therefore, to correct for medical cell datasets misalignment, we introduce in this paper a novel strategy centered around three key contributions: (1) we filter out unmatched classes to align training and transfer sample domains, (2) align feature representations across datasets using OT, and (3) apply ensemble strategies for classification. This approach leverages the strength of foundation models while directly addressing the

domain shift problem that limits their out-of-the-box use in cytology through embedding alignment. Our method is fast, works on small and large datasets and does not depend on a specific model or classifier.

2 Related Work

In recent years, considerable efforts have been made to adapt and extend foundation models to medical data. For example, SAM has been specialized into MedSAM for medical image segmentation tasks (Ma et al., 2024) and CellSAM for cell segmentation (Israel et al., 2023). DinoBloom has been trained on large white-blood cell image datasets and is build on a tailored DINO pipeline (Koch et al., 2024); scDINO is also an extension of DINO for histopathology and cellular image analysis (Pfaendler et al., 2023). Ref. Kraus et al. (2024) studied the scalability of ViTMAE models in cellular biology.

There were also great advances to mitigate shift domains and ensure better generalization across datasets using multi-encoder ensemble techniques and finetuning: Kalweit et al. (2024) used combinations of features from different foundation models to fully exploit their potential and increase robustness when transferring to an unseen dataset; Arango et al. (2024) developed an approach to select an accurate pretrained model with optimal hyperparameters for a new dataset. Other methods have been also considered to improve out-of-domain (OOD) performance such as NMTune (Chen et al., 2024), a method designed to mitigate the effect of noise in pre-trained data and lowered its impact on downstream tasks.

While on the right track, previous studies, particularly in the medical domain where experimental environments and acquisition devices often introduce domain shifts, still suffer from dataset misalignment. To circumvent this issue and align data space, Optimal Transport (OT) (Villani, 2003; 2009) and especially its extension via the Gromov-Wasserstein (GW) distance (Memoli, 2007; Mémoli, 2011a;b; Sturm, 2012) proved to be highly efficient. Indeed, GW distance enables comparison of distributions defined on different metric spaces, making it possible to construct a transport map between *a priori* incompatible data space and thereby re-align them. Thanks to its wide-ranging potential, GW has been successfully applied across diverse machine learning tasks, including computer vision applications (Memoli, 2007; Mémoli, 2011a; Schmitzer & Schnörr, 2013), alignment of word embedding spaces (Alvarez-Melis & Jaakkola, 2018), distribution-level alignment for LLMs (Melnyk et al., 2024), learning of generative models across different domains (Arjovsky et al., 2017; Tolstikhin et al., 2018; Bunne et al., 2019), graphs and clustering tasks (Vayer et al., 2018; Xu et al., 2019; 2020; Chowdhury & Needham, 2020a;b; Malki et al., 2023), learning autoencoders (Xu et al., 2020), distribution matching (Benamou et al., 2015).

OT has also found applications in a wide range of biomedical areas, such as neurology (Thual et al., 2022), metabolomics and cancer research (Breeur et al., 2024), single-cell and sequencing alignment (Demetci et al., 2022b;a; Yang et al., 2021; Pao-Huang et al., 2025), chromatogram alignment (Skoraczyński et al., 2022), cell lineage tracking (Yang et al., 2020), temporal changes in gene expression (Schiebinger et al., 2019), gene expression cartography (Nitzan et al., 2019), sender-to-target cell mapping to recover cell signaling relationships (Cang & Nie, 2020).

3 Methods

Let $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{N}$ be a labeled training dataset of cell images, and $\mathcal{D}_{\text{test}} = \{x_j\}_{j=1}^{M}$ be an unlabeled test dataset from a different imaging domain. Our task is to predict the cell type of each x_j in $\mathcal{D}_{\text{test}}$ without any labeled data from the test domain and under the assumption that only some cell types are shared across domains.

Fig. 2 shows the entire pipeline of our model, divided in the following main tasks:

1. We use foundation models to generate embeddings for each dataset (see Sec. 3.1 for feature extraction details); we thus obtain original (training) embeddings and transfer (test) embeddings.

2. We then identify shared cell types across both datasets to align cell domains. This first pre-processing step enables us to filter non-overlapping cell types from $\mathcal{D}_{\text{train}}$ and improve both dataset alignment and cell classification. As shown in step 2 of Fig. 2, to detect overlapping classes, we compare

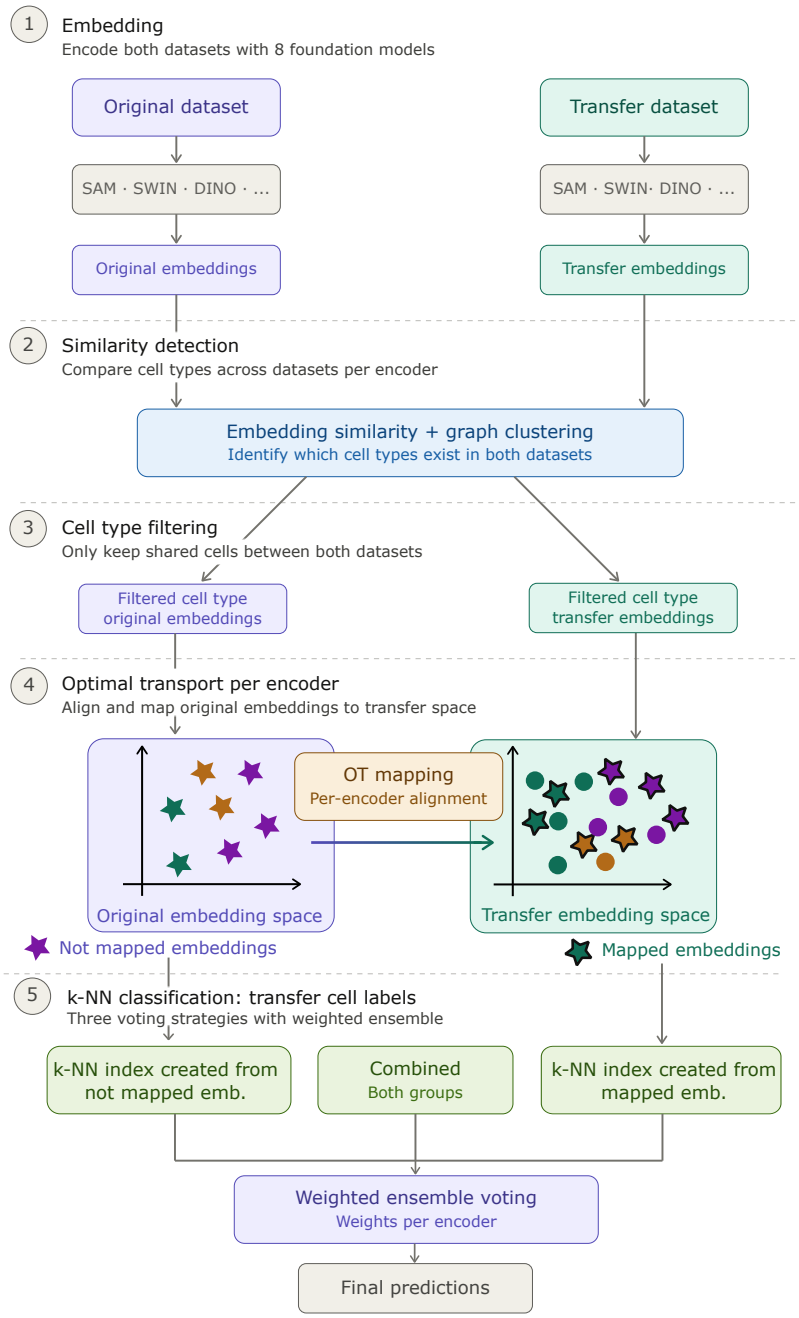

Figure 2: Machine Learning pipeline to enhance classification robustness and generalization across medical datasets: (1) sample feature representations (embeddings) generated from pretrained foundation models (used as encoders); (2) Similarity detection based on graph clustering to detect shared cell types between the original and transfer datasets; (3) cell type filtering to align training and transfer sample domains based on sample embedding similarities; (4) Optimal Transport (OT) to align training and transfer embedding spaces; (5) cell classification using preprocessed embeddings for each foundation model and ensemble voting method to obtain the final cell prediction labels. In our case we use a k-NN classifier based on the EWC-FAISS method described in Ref. Kalweit et al. (2024) and a linear probing classifier as a benchmark. More details can be found in the Method Sec. 3.

similarities between train and transfer embeddings. Indeed, we expect cells of the same type, even when acquired under different conditions, to exhibit similar embedding representations and to cluster together. All details can be found in Sec. 3.2.

3. After retaining only common cell subsets (step 3 of Fig. 2), we move to the central step of our approach detailed in Sec. 3.3: dataset alignment to address domain shifts by mapping the training distribution domain onto the transfer domain using optimal transport (OT). Here, we transport training sample embeddings (features) extracted from multiple foundation models – used as encoders – onto the transfer embedding space. Step 4 of Fig. 2 illustrate this process. New mapped training embeddings now reflect their proximity to transfer samples and are used as input of the next and last step.

4. The last step is the cell classification (of transfer cells) using a multi-encoder ensemble strategy to enhance robustness. As shown in step 5 of Fig. 2, cell classification is split into two parts: we rely both on raw training and unaligned embeddings extracted from various foundation model encoders (left) and on their mapped counterparts in the transfer embedding space (right) to make predictions. We finally combine all predictions with a weighted voting scheme to label transfer cells. More details can be found in Sec. 3.4. Note that our approach is agnostic of the chosen classification model. In our case, we use a k-NN classifier (EWC-FAISS introduced in Ref. Kalweit et al. (2024)) and a linear probing classifier as a benchmark.

3.1 Embedding Extraction with Foundation Models

Foundation models are pretrained on diverse objectives, allowing them to capture a broad spectrum of features across many domains. Each model has its own specificities and can therefore extract complementary and sometimes orthogonal features from data. By combining them and using them as encoders, we can generate more meaningful and robust cell embeddings.

In our paper, we use the following foundation models: DINO (Caron et al., 2021; Oquab et al., 2024) exploits self-distillation, where the model learns without external labels by comparing different augmented views of the same image; Segment Anything (SAM) (Kirillov et al., 2023) is designed for segmentation, and learn to delineate specific objects such as points and bounding boxes within an image based on prompts; SWIN (Liu et al., 2021; 2022a) constructs a hierarchical representation of images via layered feature maps and applies a shifted-window attention mechanism to focus on specific regions; ConvNeXT (Liu et al., 2022b; Woo et al., 2023) a convolutional neural network (CNN) architecture designed to modernize CNNs and bring their performance on par with Vision Transformers (ViTs) on large-scale vision benchmarks like ImageNet; CLIP (Radford et al., 2021) learns joint representations of images and text by training on large-scale image–caption pairs, enabling it to associate visual content with natural language descriptions; ViTMAE (He et al., 2022) uses a masked autoencoder strategy, where portions of the image are hidden and the model learns to reconstruct them, encouraging robust feature learning from incomplete visual inputs; DinoBloom (Koch et al., 2024) is built on a tailored DINO pipeline and trained on over 380,000 white blood cell images, enabling robust cell-type representation learning and strong generalization across blood and bone marrow smears.

Each encoder, from one of the foundation models detailed above, processes the input data x to produce embeddings vectors $\mathcal{E}_i^{\text{train, enc}} = \text{enc}(x_i)$ and $\mathcal{E}_j^{\text{test, enc}} = \text{enc}(x_j)$. Once embeddings are generated, they are used as input for identification of shared cell types across datasets (see Sec. 3.2), optimal transport mapping to align datasets (see Sec. 3.3), and, eventually, cell classification via ensemble voting (see Sec. 3.4).

3.2 Unsupervised Identification of Shared Cell Types

Before using the entire original dataset $\mathcal{D}_{\text{train}}$ to label cells in $\mathcal{D}_{\text{test}}$, we propose to preprocess data by filtering cell types that are not present in the test dataset, i.e., the non-overlapping classes between $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$. The goal of this preprocessing step is two-fold: (i) it removes unwanted noise by only keeping shared cell types leading to a better classification; (ii) it significantly improves the accuracy of the optimal transport mapping. Indeed, as explained in Sec. 3.3, the optimal transport step aligns datasets by mapping the

training cell embeddings ($\mathcal{D}_{\text{train}}$) onto the test cell embeddings ($\mathcal{D}_{\text{test}}$). By filtering out training classes that are not present in the test dataset, we ensure that the mapping is restricted to the shared cell types only, preventing mismatches and making the transfer more reliable.

Fig. 3 summarizes the steps for unsupervised identification of shared cell types. We begin by using embeddings from DinoBloom (Koch et al., 2024) to construct a weighted graph for the unlabeled transfer (test) dataset. Each cell sample is represented as a node, and edges are weighted according to the geodesic distance between cell embeddings – the geodesic distance is the shortest distance between each pair of nodes in a graph. We also apply a threshold to bound the maximal distance below which samples are connected through edges. We refer to Appendix A.1 for details. As shown in the left gray box of Fig. 3, cells of the same type (indicated by the same color) naturally cluster together. To detect clusters, we used two community detection approaches: the Louvain algorithm (Blondel et al., 2008) as a benchmark, and our own cluster detection algorithm. The motivation for developing our own method was to better adapt community detection to our problem and to gain finer control over hyperparameters, thereby enabling more precise cluster detection (see Appendix A.1). An interesting aspect of our approach is the use of a k-NN classifier (EWC-FAISS introduced in Ref. (Kalweit et al., 2024)) in a fully unsupervised manner to assign cluster labels to each cell sample in $\mathcal{D}_{\text{test}}$. This method proves effective not only for cell classification but also for identifying shared cell types across datasets.

Once detected, to determine the cell type of each test cluster, we compute mean cosine similarity matrices. More precisely, these matrices have dimensions $\mathcal{N}_C^{\text{test}} \times \mathcal{N}_C^{\text{train}}$, where $\mathcal{N}_C^{\text{test}}$ is the number of detected clusters in $\mathcal{D}_{\text{test}}$, and $\mathcal{N}_C^{\text{train}}$ denotes the number of cell type subsets in $\mathcal{D}_{\text{train}}$. Cell embeddings from each encoder (i.e., each foundation model) are used to calculate cosine similarity scores between the test and train samples. Then, the results are grouped such that each coefficient of cosine similarity matrices correspond to the average score between one test cluster and one training cell type subset (see the right gray box of Fig. 3). Therefore, each column of these matrices is a $\mathcal{N}_C^{\text{train}}$-dimensional vector and contains the cosine similarity scores between one test cluster and all training cell types. These vectors are then aggregated across encoders to obtain the final similarity representation that we denote as $\mathbf{cs}^{(c_i)}$, where c_i refers to the test cluster label.

Clusters that correspond to shared cell types are expected to yield high similarity scores, enabling cluster identification. Concretely, for each test cluster c_i, the maximum entry of $\mathbf{cs}^{(c_i)}$ indicates the most likely matching cell type of the training dataset, and this label is assigned to cluster c_i. Cell types from the original (train) dataset that do not match any test cluster can then be filtered out. We refer to Appendix A.2 where we detail all steps and the computation of vectors $\mathbf{cs}^{(c_i)}$.

When possible, the cluster identification of transfer cells can be refined in a fully unsupervised manner by enriching the database with cells of the same category. In this extended setup, the original dataset $\mathcal{D}_{\text{train}}$ remains the reference and the only labeled set. On top of the test dataset $\mathcal{D}_{\text{test}}$, we incorporate additional datasets containing similar cell types (e.g., white blood cells). For each new dataset, we apply the steps described above to detect and label clusters in an unsupervised fashion.

Next, using the cell types in $\mathcal{D}_{\text{train}}$ as a reference, we compute the $\mathcal{N}_C^{\text{train}}$-dimensional similarity score vectors $\mathbf{cs}^{(c_i)}$ for all clusters identified across datasets. Clusters corresponding to the same cell type are expected to share similar vectors and group together, even when they belong to different datasets. To reveal these cluster communities, we apply t-SNE and we obtain two-dimensional vectors that naturally group together by cell type. We then use this new vector representation to refine our cluster identification: for a given test cluster, instead of assigning the label corresponding to the highest similarity score among the $\mathcal{N}_C^{\text{train}}$ possible reference cell types, we combine two complementary criteria: (i) the similarity scores themselves, and (ii) the relative position of the cluster in the two-dimensional t-SNE space with respect to all reference cell type vectors, including those from additional (and unlabeled) reference datasets.

It is true that there is no guarantee that the relative positions of clusters in the multi-dimensional space will be preserved in the two-dimensional t-SNE, precisely because t-SNE does not preserve pairwise distances in a strict quantitative sense. However, t-SNE is specifically designed to capture and reveal local neighborhoods. In this context, we can interpret and use the 'distance' of a cluster sample to all other samples in the two dimensional t-SNE space as an indication of the proximity of this cluster to a given cluster community. If

a cluster lies in the vicinity of a community composed of clusters with a different cell-type label, it likely indicates that the cluster was misclassified when using only a single reference dataset.

The fact that this approach performs well in our setting is therefore not surprising: unlike direct comparisons in the similarity-score space, t-SNE tends to enhance the grouping of cluster samples from the same cell type, even when they come from distinct datasets.

We refer to Fig. 4 in the result section 5.1 where such a misclassified cluster has been detected in the case of the LISC dataset when using two other white-blood cell datasets as reference.

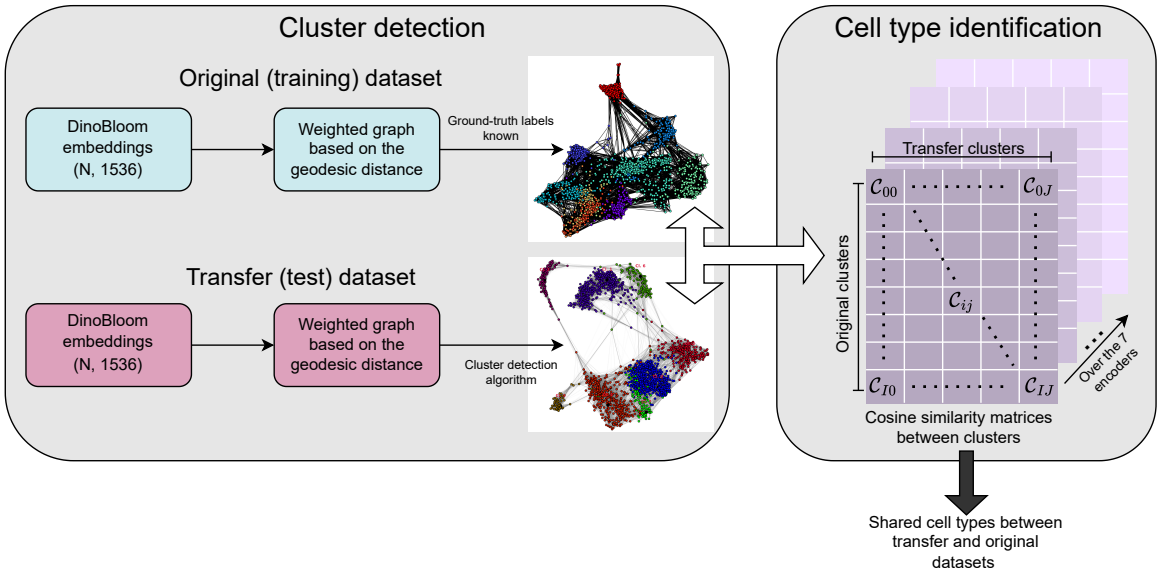

Figure 3: Scheme of the method used for shared cell type identification between the original (training) and the transfer (test) datasets.

3.3 Graph-Based Optimal Transport for Alignment

Optimal Transport (OT) (Villani, 2003; 2009) studies how to transform one probability distribution (or a histogram in the finite dimensional case) into another in the most efficient way. In other words, the goal of OT is to find a mapping that moves mass from a source to a target distribution with minimal cost.

In the context of machine learning, masses p_i can be assigned to each sample of a dataset to account for their relative importance within the dataset – constrained by the condition $\sum_{i=1}^{N} p_i = 1$, where N is the number of samples; if all samples are treated equally, we simply assign equal weights $p_i = 1/N$ to each sample. Then, given two sets of samples $A = \{a_i\}_{i=1}^{N}$ and $B = \{b_j\}_{j=1}^{M}$, OT aims to transform distribution $\{p_i\}_{i=1}^{N}$ of A onto the mass distribution $\{q_j\}_{j=1}^{M}$ of dataset B by finding a coupling matrix T_{ij} between samples a_i of A and samples b_j of B. However, most of the time, samples from different datasets leave in different or incompatible spaces with different number of samples making it difficult to directly compare their probability distributions. To bypass this issue, one can use the Gromov-Wasserstein (GW) distance (Mémoli, 2011a; Sturm, 2012; Peyré et al., 2016) to study how the relative distances of *all* samples within each dataset compare to each other, instead of comparing absolute sample positions across datasets. Therefore, the alignment between datasets is not solely based on individual samples, but relies on the structural information of each dataset, making it a perfect tool for structured data, and in particular for cell image dataset alignment problems. In our case, we will always transport the original dataset $\mathcal{D}_{\text{train}}$ (with known cell labels and used as training dataset) onto the transfer dataset $\mathcal{D}_{\text{test}}$ (the test dataset with unknown labels).

To this aim, (structured) data are expressed in a metric-measure space (C, p), where C is a matrix that represents similarities or distances between samples of the same dataset, and p, as written in the above

paragraph, is the probability (mass) distribution associated to the dataset. In the following we assume no prior information and relative importance between samples; we thus set $p = \frac{1}{N}\mathbb{1}_N$, where $\mathbb{1}_N = (1, \ldots, 1)^T \in \mathbb{R}^N$. We denote by $C^{(A)}$ and $C^{(B)}$ the similarity matrices of two different datasets A and B, and by p and q the probability distributions of A and B.

In our case, matrices C are computed using the geodesic distance, i.e., the shortest distance between each pair of nodes in a graph. To proceed, we treat each sample as a node and we construct a kNN graph (where k represents the number of neighbors of each node) for each dataset. To draw edges between nodes, we use the Euclidean distance between sample embeddings extracted from the different foundation models (encoders) as previously detailed in Sec. 3.1. Then, for each pair of samples (nodes) i and j we can compute the shortest graph distance between them and obtain all C_{ij} coefficients of matrix C – as explained in Sec. 3.2, a similar approach is used to create weighted graphs and to detect shared cell types across datasets. The goal of OT is then to find couplings that associate samples with similar inner structure to minimize the cost $|C^{(A)}_{i,k} - C^{(B)}_{j,l}|$ of transporting pair of nodes (i, k) of dataset A to (j, l) of dataset B.

So far, matrices $C^{(A)}$ and $C^{(B)}$ are only related to the internal structure of their corresponding dataset, or, in other words, how samples are distributed across the graph. Even if embeddings are used to draw the graph, C matrices do not directly compare embedding similarities between both datasets. As discussed in Ref. Vayer et al. (2018), on top of the graph structure, one can also add additional graph properties as node features and define a cost matrix $M_{AB} \in \mathbb{R}^{N\times M}$ to compute the Euclidean distance between these additional features for each sample pair (a_i, b_j) across both datasets A and B. In our case, node features are simply sample embeddings; the cost matrix coefficients thus correspond to $(M_{AB})_{ij} = |\mathcal{E}^A_i - \mathcal{E}^B_j|$. The distance used in the OT framework that incorporates both structure and feature information is called the Fused Gromov-Wasserstein (FGW) distance (Vayer et al., 2018; Thual et al., 2022); this is the distance that we consider in this paper.

Before giving the final formulation of OT using the FGW distance, we introduce $\Pi(p, q)$ the set of all possible couplings T_{ij} between p and q – the probability distributions that we aim to transport from A to B:

$$\Pi(p, q) = \{T \in \mathbb{R}^{N\times M}_+, \quad T\,\mathbb{1}_M = p, \quad T^T\mathbb{1}_N = q\}, \tag{1}$$

where $\mathbb{1}_{N(M)} = (1, \ldots, 1)^T \in \mathbb{R}^{N(M)}$. Couplings T_{ij} thus represent the amount of mass transported from sample a_i in A to sample b_j in B. The above expression ensures that mass fractions from all samples of B that connect to sample a_i in A sum up to p_i (and that all samples of A that connect to sample b_j in B sum up to q_j), such that masses are correctly transported from one dataset to another.

As stated before, OT is an optimization problem that consists of finding the best coupling matrix T_{ij} that minimizes the cost to transport the structure and the features from one data space to another one through the FGW distance defined as follows:

$$d_{\mathrm{FGW}}\left(C^{(A)},\, C^{(B)},\, p,\, q,\, \alpha\right) = \inf_{T\in\Pi(\mathbf{p},\mathbf{q})} \left\langle (1-\alpha)\, M_{AB} + \alpha\, L\left(C^{(A)}, C^{(B)}\right) \otimes T,\, T \right\rangle. \tag{2}$$

In equation 2, $L\left(C^{(A)}, C^{(B)}\right)$ is a loss function to evaluate the misfit between C matrices. We usually consider the quadratic loss resulting in the 4-dimensional tensor $L_{ijkl} \equiv \left|C^{(A)}_{i,k} - C^{(B)}_{j,l}\right|^2/2$. Parameter α is a trade-off between the cost to transport features (through M_{AB}) and the cost to transport pair of samples in each structure (through the loss function L). The symbol $\otimes$ accounts for tensor-matrix multiplication, while $\langle\cdot\rangle$ is the usual matrix scalar product:

$$\begin{aligned} (L\otimes T)_{ij} &= \sum_{kl} L_{ijkl}\, T_{kl}, \\ \langle M, T\rangle &= \sum_{ij} M_{ij}\, T_{ij}. \end{aligned} \tag{3}$$

To solve this optimization problem, we use the POT package (Flamary et al., 2021) which provides all necessary functions. Datasets A and B are the original (training) dataset $\mathcal{D}_{\mathrm{train}}$ and transfer (test) dataset $\mathcal{D}_{\mathrm{test}}$,

respectively. Matrices $C^{(\text{train})}$, $C^{(\text{test})}$ and $M_{\text{train/test}}$ are computed using the original and transfer embeddings $\{\mathcal{E}_i^{\text{train, enc}}\}_{i=1}^N$ and $\{\mathcal{E}_i^{\text{test, enc}}\}_{i=1}^N$ extracted from a given encoder (foundation model) '*enc*'. Therefore, each encoder gives a different OT mapping that can be used independently or concomitantly (see Sec. 3.4).

As a result, the coupling matrix T_{ij} found for each foundation model is used to map the embeddings of the original dataset onto the transfer dataset:

$$\overline{\mathcal{E}_i^{\text{train, enc}}} = N \sum_{j=1}^{M} T_{ij}\, \mathcal{E}_j^{\text{test, enc}}, \tag{4}$$

where $\overline{\mathcal{E}^{\text{train, enc}}}$ are the resulting mapped embeddings, and N is the number of samples of the original dataset $\mathcal{D}_{\text{train}}$, used here to correctly normalize the couplings. By projecting training embeddings onto the test embedding space using OT, training and test data share a common ground space and we thus correct for distribution shifts between them.

3.4 Classification via Ensemble Voting

To classify cells in $\mathcal{D}_{\text{test}}$, we use an adaptive nearest neighbor k-NN search through the *Entropy-guided Weighted Combinational FAISS* (EWC-FAISS) method described in Ref. Kalweit et al. (2024). The novelty of this paper is to filter non-overlapping cell types between the training and test datasets, and combine optimal transport (OT) with an ensemble voting method in the following way: (i) for each encoder, mapped embeddings $\{\overline{\mathcal{E}_i^{\text{train, enc}}}\}_{i=1}^N$ of the filtered original dataset are used to train a FAISS index (with the L2 distance) (Douze et al., 2024). The resulting k-NN classifier trained with the OT mapped embeddings is then used to label and classify cells in $\mathcal{D}_{\text{test}}$; (ii) as a second step, we also train the k-NN index without using OT, i.e., using the original and unmapped embeddings of the training dataset, and use this new classifier to label test cells; (iii) we then concatenate label predictions from both methods (with and without OT), assign weights for each encoder and perform a weighted vote among all label predictions to get the final predicted cell types. The different steps are summarized in the last gray box of Fig. 2.

4 Experimental Setting

4.1 Datasets

We considered in our paper four white-blood cell datasets (all publicly available) and two live cell datasets:

Stained White Blood Cells. The WBC (White-Blood Cells) dataset (Bodzas et al., 2023) includes 16,027 images of stained white blood cells from patients with acute myeloid and lymphoid leukemia, as well as those without leukemic pathology. They are categorized into neutrophil segments, neutrophil bands, eosinophils, basophils, lymphocytes, monocytes, normoblasts, myeloblasts, and lymphoblasts. In this paper, results from the WBC dataset are obtained from a random subset of 14,424 images. The LISC dataset (Rezatofighi & Soltanian-Zadeh, 2011) comprises 257 images of white blood cells from healthy individuals, classified into basophils, eosinophils, lymphocytes, monocytes, and neutrophils. The PBC (Peripheral Blood Cells) dataset (Acevedo et al., 2020) contains 17,092 images of white-blood cells labeled into 11 classes: basophils, eosinophils, lymphocytes, monocytes, neutrophils, Erythroblast, Immature Granulocytes, Metamyelocytes, Promyelocytes, Myelocytes, Platelet. The RAABIN dataset (Kouzehkanan et al., 2021) gathers 16,633 cropped white-blood cells, classified into basophils, eosinophils, lymphocytes, monocytes, and neutrophils. In this paper, results from the RAABIN dataset are obtained from a random subset of 10,175 images.

Live Cell State Imaging. The CELL DEATH NANOLIVE dataset (Kalweit et al., 2025) contains images of treated JIMT-1 breast cancer cells categorized into living, dead, apoptotic, and necrotic cells. It includes 7,420 with the Roboflow software manually annotated and segmented images captured with a high-resolution Nanolive microscope at 60× magnification. The CELL DEATH LIONHEART dataset contains 59 images recorded with the Lionheart automated microscope at 20×.

4.2 Baselines

As a baseline, we compare our results to NMTune approach (Chen et al., 2024), a method designed to mitigate the effect of noise in pre-trained data and lowered its impact on downstream tasks. In this case, we consider the same MLP classifier architecture [(1) Layer D $\times$ 4D, (2) ReLU activation, (3) Layer 4D $\times$ D, (4) Layer D $\times N_{\text{classes}}$, where D is the input feature dimension]. We apply class filtering but no optimal transport since NMTune is already tailored to lower out-of-domain shifts.

We also compare our results to linear probing layer (with/without class filtering and with/without optimal transport)

5 Results

In this section, we first show the results for the identification of shared cell types in white-blood cell datasets. To evaluate our own cluster detection algorithm, we challenge our results with the commonly used Louvain algorithm (Blondel et al., 2008).

Then, we present the main results of our paper for cell classification. Specifically, we evaluate on various white-blood cell and live cell datasets our approach described in the Method section, which incorporates class filtering, optimal transport and ensemble voting from classifier predictions. We also gauge the gain in accuracy using optimal transport mapping and/or ensemble of encoders.

5.1 Shared cell type identification

An overview of the shared cell types identification for white-blood cell datasets is shown in Tables 1 and 2. In the first table, the PBC dataset corresponds to the reference (original) dataset whose cell labels are known. We then use the method described in Sec. 3.2 to detect shared cell types between PBC and three white-blood cell datasets: WBC, RAABIN and LISC. In Table 2, WBC is used as the reference, while PBC, RAABIN and LISC are the test datasets.

To detect cell clusters, we resort to two different cluster detection algorithms. The first row of each table corresponds to the results using the well-known Louvain algorithm (Blondel et al., 2008), while the second row gathers the results using our own algorithm based on weighted node degree and k-NN search (see Appendix A.1 for more details).

To improve the robustness of the results, we randomly select subsets of 2000 samples when the number of cell samples in a test dataset exceeds 2000 (e.g., RAABIN, WBC, and PBC) and predict matching cell types with the reference dataset for all detected clusters in each subset. This procedure is repeated across five random subsets. Results are then aggregated across all subsets (or from a single run if the dataset contains fewer than 2000 samples).

From these aggregated results we report: (i) the predicted shared cell type labels (shown in the first sub-column for each test dataset, with the label-to-cell type correspondence provided in the caption; the ground-truth shared cell types are also indicated in the caption); and (ii) the overall accuracy that corresponds to the mean of two accuracies:

$$\text{Accuracy} = \frac{\text{Acc}_{\text{global}} + \text{Acc}_{\text{clusters}}}{2}, \tag{5}$$

where $\text{Acc}_{\text{global}}$ quantifies the correctly predicted shared cell types between the test and reference (training) datasets, while also penalizing unfiltered non-overlapping classes that remain in the reference dataset. The detailed expression can be found in Appendix B.

The cluster-level accuracy, $\text{Acc}_{\text{clusters}}$, simply corresponds to the fraction of correct cell type predictions across all clusters in the test dataset. Specifically, the predicted cell type of each cluster is compared with its ground-truth cell type, defined as the predominant ground-truth label within that cluster.

Note that the results reported here are obtained after correcting misidentified clusters. As explained in the last paragraph of Method section 3.2, adding new datasets to the reference dataset can help refine the cell type identification of the test dataset. For instance, when PBC is the (labeled) reference dataset (Table

1), predicted cell type clusters from WBC are used as additional reference points to correct for LISC and RAABIN misidentified clusters.

Fig. 4 illustrates this process for the LISC dataset. Each dot in the figure represents a cluster c_i that carries a similarity score vector $\mathbf{cs}^{(c_i)}$, i.e., its 'proximity' to the different cell types in the reference dataset (see Sec. 3.2 and Appendix A.2 for all details). To visualize these multidimensional vectors in a two-dimensional space, we applied t-SNE. Labels above the dots indicate true and known cell types for PBC clusters (the reference set, shown as black dots), and the most likely matching cell types before correction for WBC (blue dots) and LISC (red dots) clusters.

As shown, clusters corresponding to the same cell type tend to group together, even when originating from different datasets, and this property can be exploited to correct wrong labels. For instance, the red LISC cluster labeled '9' is misidentified when PBC alone is used as the reference (see the red inset in the figure, which zooms on this cluster). Indeed, based on similarity scores only, this cluster appears closer to the PBC black cluster labeled '9' than to the black cluster labeled '3' (its true label), and is therefore incorrectly classified as '9'. However, when WBC cells are added, the LISC red cluster is found in close proximity to five WBC clusters labeled '3'. Taking this proximity into account allows us to relabel it correctly as '3', leading to an accuracy of 100% for the LISC dataset – since the other four LISC clusters had already been correctly identified (see Table 1). Importantly, the ground-truth labels of WBC clusters are not required for this correction step. It is also worth noticing that no WBC or LISC clusters gather around non-overlapping cell types (see the PBC black dots 8, 10, 11, 12, 13 and their empty neighborhood).

This correction step is crucial, as it yields a substantial improvement in accuracy (+7.1% in average when WBC is the reference dataset, +15.5% when PBC is the reference dataset).

Table 1: With PBC ground-truth labels (PBC = original dataset; WBC, RAABIN, LISC = Transfer datasets with unknown labels). Shared cell types: 0, 1, 3, 4, 6 for all datasets; cell types in PBC – 0: basophils, 1: eosinophils, 3: lymphocytes, 4: monocytes, 6: neutrophils, 8: Erythroblast: 8, 9: Immature Granulocytes, 10: Metamyelocytes, 11: Promyelocytes, 12: Myelocytes, 13: Platelet.

Datasets	WBC		LISC		RAABIN	
	Shared cell types	Acc.	Shared cell types	Acc.	Shared cell types	Acc.
Louvain	0, 1, 3, 4, 6, 9	86.11	0, 1, 6, 9	45.0	0, 1, 3, 4, 6, 8, 9, 12	66.07
Ours	0, 1, 3, 4, 6, 9	**87.25**	0, 1, 3, 4, 6	**100.0**	0, 1, 3, 4, 6, 8, 10	**70.54**

Table 2: With WBC ground-truth labels (WBC = original dataset; PBC, RAABIN, LISC = Transfer datasets with unknown labels). Shared cell types: 0, 1, 3, 4, 6; cell types in WBC – 0: basophils, 1: eosinophils, 2: lymphoblasts, 3: lymphocytes, 4: monocytes, 5: monoblasts, 6: neutrophils, 7: normoblasts.

Datasets	PBC		LISC		RAABIN	
	Shared cell types	Acc.	Shared cell types	Acc.	Shared cell types	Acc.
Louvain	0, 1, 3, 4, 6	**100.0**	0, 2, 3, 4, 6	56.67	0, 1, 2, 3, 4, 6, 7	57.74
Ours	0, 1, 2, 3, 4, 6	79.49	0, 1, 3, 4, 6	**100.0**	0, 1, 3, 4, 6	**87.21**

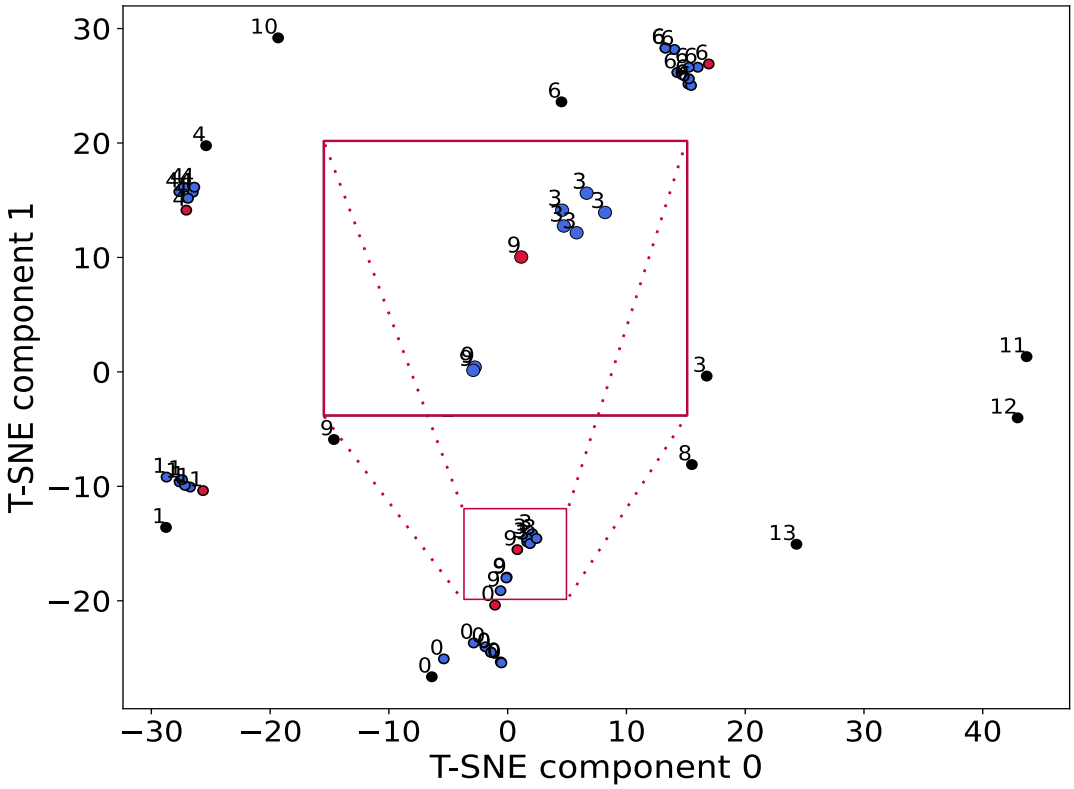

Figure 4: T-SNE 2D components of cluster similarity vectors for PBC (black points), WBC (blue points) and LISC (red points) datasets. The labels on top of each black dot correspond to the ground-truth cell types in PBC, while those on top of the blue and red points are obtained from the shared cell type identification method for each detected cluster (see Sec. 3.2). The four LISC red clusters labeled 0, 1, 4, 6 are correctly identified to their ground-truth class. The LISC cluster labeled 9 has been misidentified, but the red inset shows its close proximity to WBC clusters labeled 3; this can be use to relabel this LISC cluster to its correct class '3'.

5.2 Cell classification

Classification results are presented from Tables 3 to 7. For clarity, not all transfer datasets are reported here; all results can be found in Appendix C. To demonstrate the capability of our approach to generalize to various cell datasets, we show classification accuracies and F1-scores for a live cell dataset (Lionheart) using Nanolive as the training (reference) dataset (see Table 3), and two white-blood cell datasets (RAABIN and LISC) with different combinations of training datasets (see Tables 4, 5 and 6). Indeed, to assess how well our method can generalize to mixture of cells coming from different sources and acquired differently, we trained the classifier with combined WBC–PBC dataset constructed by selecting certain cell types from WBC and the remaining ones from PBC. The precise combination is detailed in the caption of Tables 4 and 5. As a second example, we also trained the index with all cells coming from WBC, PBC and RAABIN (after class filtering) to classify LISC cells (table 6).

To ensure sufficient statistical variability in our results, we proceed as follows: the training datasets are randomly divided into ten subsets of 2,000 samples each. The transfer dataset is also split randomly in ten subsets if it contains more than 2000 cell images. If Optimal Transport (OT) is used, we transport each of the subset embeddings either to the full transfer embeddings (if it contains less than 2000 samples) or to each transfer subset embeddings. We then recombine all training subsets into one set to train the classifiers. Note that due to the random sampling, the number of training samples does not cover the entire initial training set. In the case where the transfer dataset contains more than 2000 samples, we evaluate the cell

classification on each subset and the metric results presented in the corresponding tables (Nanolive – Table 3, and RAABIN – Table 4) correspond to the mean of accuracies and F1-scores over all ten subsets.

The encoders used to generate the embeddings are indicated in the first column. A sum of encoders means that embeddings from each encoder are concatenated to form a new feature representation. All accuracies and F1-scores are obtained using a k-NN classifier after preprocessing data (cell type filtering in the original dataset using results from Sec. 5.1) and with or without OT – indicated by a cross or a checkmark in the result tables. The last two columns of each table indicates the gain or loss in accuracy and F1-score using OT for each encoder. The last three rows indicate: (i) the results following the pipeline described in Fig. 2, i.e., with ensemble voting including OT (see Sec. 3.4); (ii) the ensemble voting method for the linear probing baseline, and (iii) the gain or loss in accuracy and F1-score averaged over all encoders (including the ensemble voting method rows) when using OT.

To better visualize the effect of OT for each encoder, Figs. 7 show the accuracies as histograms for the four cases considered in the tables. On each histogram, the black bars indicate the accuracy for each encoder without OT. White stripes bars shows a decrease in accuracy when combining the k-NN search with OT, while green bars highlight an accuracy gain when applying OT. The blue bar corresponds to the ensemble voting method including OT.

To illustrate how OT improves accuracy, we also refer the reader to Figs. 5 and 6. It shows five LISC query image examples and their corresponding neighbors found for the ensemble voting method in the three training datasets (WBC, PBC, RAABIN) when all combined – thus corresponding to the results shown in Table 4). Fig. 5 is obtained without the use of OT, while Fig. 6 incorporates OT to align all training datasets with LISC.

Table 7 summarizes the results by reporting the average accuracies and F1-scores across all datasets. Results are shown for each encoder individually, for combinations of encoders, and for the ensemble voting approach (penultimate row). The final row indicates the overall improvement achieved with the use of OT.

Table 8 compare the accuracy and F1-score performances when including or not class filtering and OT. The results are averaged over all datasets.

Table 3: Encoder performance for the Nanolive (Lionheart) transfer, with (✓) and without (✗) optimal transport (OT). Classes: (0) live, (1) apoptotic, (2) necrotic cells. Best result per encoder and metric in bold; the Δ columns report the OT gain in percentage points. See Sec. 5.2 for the meaning of the last three rows.

Transfer: **Nanolive – 3 classes** *Train: Lionheart*	Acc. (OT: ✗)	Acc. (OT: ✓)	F1 (OT: ✗)	F1 (OT: ✓)	Δ Acc.	Δ F1
SAM	**31.35**	29.01	2.51	**49.43**	-2.34	46.92
DINO	44.57	**45.89**	**61.73**	57.60	1.32	-4.13
ConvNeXt	50.22	**57.53**	56.51	**59.84**	7.31	3.33
SWIN	48.53	**57.95**	50.50	**67.66**	9.42	17.16
CLIP	**62.71**	61.11	**62.42**	62.22	-1.6	-0.2
ViTMAE	44.80	**51.10**	24.56	**54.17**	6.3	29.61
DinoBloom	60.11	**65.52**	63.35	**66.27**	5.41	2.92
CN + SWIN	49.82	**60.05**	49.33	**63.26**	10.23	13.93
SAM + CN + SWIN + CLIP	52.38	**61.15**	51.43	**64.11**	8.76	12.68
DINO + CN + ViTMAE + SWIN	47.37	**56.98**	61.48	**62.94**	9.61	1.46
CN + SWIN + ViTMAE	49.38	**59.80**	48.72	**63.12**	10.41	14.4
DINO + CN + SWIN	47.35	**56.78**	61.50	**62.95**	9.43	1.45
DINO + CN + ViTMAE	46.72	**52.32**	**61.96**	60.48	5.6	-1.48
DINO + ViTMAE + SWIN	45.69	**51.68**	**61.52**	60.81	5.99	-0.72
Ensemble Voting method	61.52	**65.91**	63.97	**67.43**	4.38	3.46
Ensemble Voting – lin. prob.	55.86	**60.30**	65.51	**66.56**	4.44	1.06
Average Δ_{OT}	-	-	-	-	6.02	9.39

Table 4: Encoder performance for the RAABIN transfer, training set WBC-PBC (types 0, 1, 3 from WBC; 4, 6 from PBC), with (✓) and without (✗) optimal transport (OT). Best result per encoder and metric in bold; the Δ columns report the OT gain in percentage points.

Transfer: **RAABIN** *Train: WBC-PBC*	Acc. (OT: ✗)	Acc. (OT: ✓)	F1 (OT: ✗)	F1 (OT: ✓)	Δ Acc.	Δ F1
SAM	7.94	**25.37**	23.93	**50.36**	17.42	26.43
DINO	27.73	**50.86**	63.84	**63.99**	23.13	0.15
ConvNeXt	33.25	**36.99**	**68.00**	56.27	3.74	-11.73
SWIN	26.62	**55.55**	**66.97**	57.22	28.93	-9.75
CLIP	29.82	**35.21**	**58.86**	48.52	5.39	-10.34
ViTMAE	**31.62**	19.60	10.01	**42.48**	-12.02	32.48
DinoBloom	58.59	**72.13**	**70.10**	56.82	13.54	-13.28
CN + SWIN	30.09	**51.89**	**69.29**	58.09	21.8	-11.2
SAM + CN + SWIN + CLIP	31.67	**51.23**	**70.22**	57.05	19.56	-13.17
DINO + CN + ViTMAE + SWIN	31.47	**56.27**	**70.74**	62.44	24.8	-8.3
CN + SWIN + ViTMAE	31.67	**51.92**	**70.69**	59.21	20.25	-11.48
DINO + CN + SWIN	29.96	**56.13**	**69.58**	61.67	26.17	-7.91
DINO + CN + ViTMAE	34.56	**41.72**	**70.47**	62.33	7.16	-8.14
DINO + ViTMAE + SWIN	28.83	**61.64**	**68.79**	63.78	32.81	-5.01
Ensemble Voting method	44.98	**69.34**	72.46	**79.47**	24.36	7.01
Ensemble Voting – lin. prob.	43.92	**61.73**	73.94	**80.41**	17.81	6.47
Average Δ_{OT}	-	-	-	-	17.14	-2.95

Table 5: Encoder performance for the LISC transfer, training set WBC-PBC (types 0, 1, 3 from WBC; 4, 6 from PBC), with (✓) and without (✗) optimal transport (OT). Best result per encoder and metric in bold; the Δ columns report the OT gain in percentage points.

Transfer: **LISC** *Train: WBC-PBC*	Acc. (OT: ✗)	Acc. (OT: ✓)	F1 (OT: ✗)	F1 (OT: ✓)	Δ Acc.	Δ F1
SAM	**37.91**	29.47	**33.95**	28.50	-8.43	-5.45
DINO	33.68	**48.34**	33.25	**50.57**	14.66	17.32
ConvNeXt	46.02	**53.91**	42.80	**52.10**	7.89	9.3
SWIN	22.19	**61.51**	18.83	**60.89**	39.32	42.05
CLIP	23.54	**41.40**	16.47	**41.09**	17.86	24.62
ViTMAE	**35.94**	25.31	**29.30**	23.39	-10.63	-5.91
DinoBloom	66.54	**73.56**	63.44	**72.72**	7.02	9.29
CN + SWIN	30.54	**61.16**	22.39	**59.97**	30.63	37.58
SAM + CN + SWIN + CLIP	31.71	**62.27**	23.52	**60.38**	30.56	36.86
DINO + CN + ViTMAE + SWIN	33.72	**62.24**	28.94	**61.96**	28.52	33.02
CN + SWIN + ViTMAE	29.37	**61.61**	21.65	**60.22**	32.24	38.57
DINO + CN + SWIN	31.71	**63.53**	25.84	**63.42**	31.82	37.58
DINO + CN + ViTMAE	52.23	**59.20**	54.40	**58.63**	6.97	4.23
DINO + ViTMAE + SWIN	28.72	**59.18**	21.11	**60.59**	30.45	39.48
Ensemble Voting method	52.73	**77.30**	56.24	**77.23**	24.57	20.99
Ensemble Voting – lin. prob.	48.55	**69.48**	48.87	**69.57**	20.94	20.69
Average Δ_{OT}	-	-	-	-	+18.9	+22.63

Table 6: Encoder performance for the LISC transfer, training set WBC+PBC+RAABIN (all cells, before filtering), with (✓) and without (✗) optimal transport (OT). Best result per encoder and metric in bold; the Δ columns report the OT gain in percentage points.

Transfer: **LISC** *Train: WBC+PBC+RAABIN*	Acc. (OT: ✗)	Acc. (OT: ✓)	F1 (OT: ✗)	F1 (OT: ✓)	Δ Acc.	Δ F1
SAM	31.77	**35.52**	25.17	**35.51**	3.75	10.34
DINO	33.12	**51.79**	28.07	**51.45**	18.67	23.38
ConvNeXt	55.59	**56.27**	**57.68**	56.56	0.68	-1.12
SWIN	24.72	**64.21**	16.37	**63.47**	39.49	47.1
CLIP	34.03	**40.97**	32.14	**40.35**	6.94	8.21
ViTMAE	24.83	**26.80**	15.99	**26.93**	1.98	10.95
DinoBloom	70.22	**77.26**	69.80	**76.34**	7.05	6.54
CN + SWIN	25.79	**61.45**	16.56	**62.53**	35.66	45.98
SAM + CN + SWIN + CLIP	27.24	**67.07**	19.79	**67.71**	39.83	47.92
DINO + CN + ViTMAE + SWIN	28.79	**68.64**	23.11	**69.49**	39.85	46.39
CN + SWIN + ViTMAE	24.95	**61.84**	15.25	**62.20**	36.89	46.95
DINO + CN + SWIN	27.95	**67.75**	21.48	**68.55**	39.8	47.07
DINO + CN + ViTMAE	52.57	**68.03**	53.01	**68.83**	15.46	15.83
DINO + ViTMAE + SWIN	23.74	**63.42**	15.31	**63.69**	39.68	48.38
Ensemble Voting method	55.39	**72.45**	52.90	**70.53**	17.06	17.63
Ensemble Voting – lin. prob.	**73.00**	63.43	**70.09**	61.07	-9.57	-9.02
Average Δ_{OT}	-	-	-	-	22.85	28.1

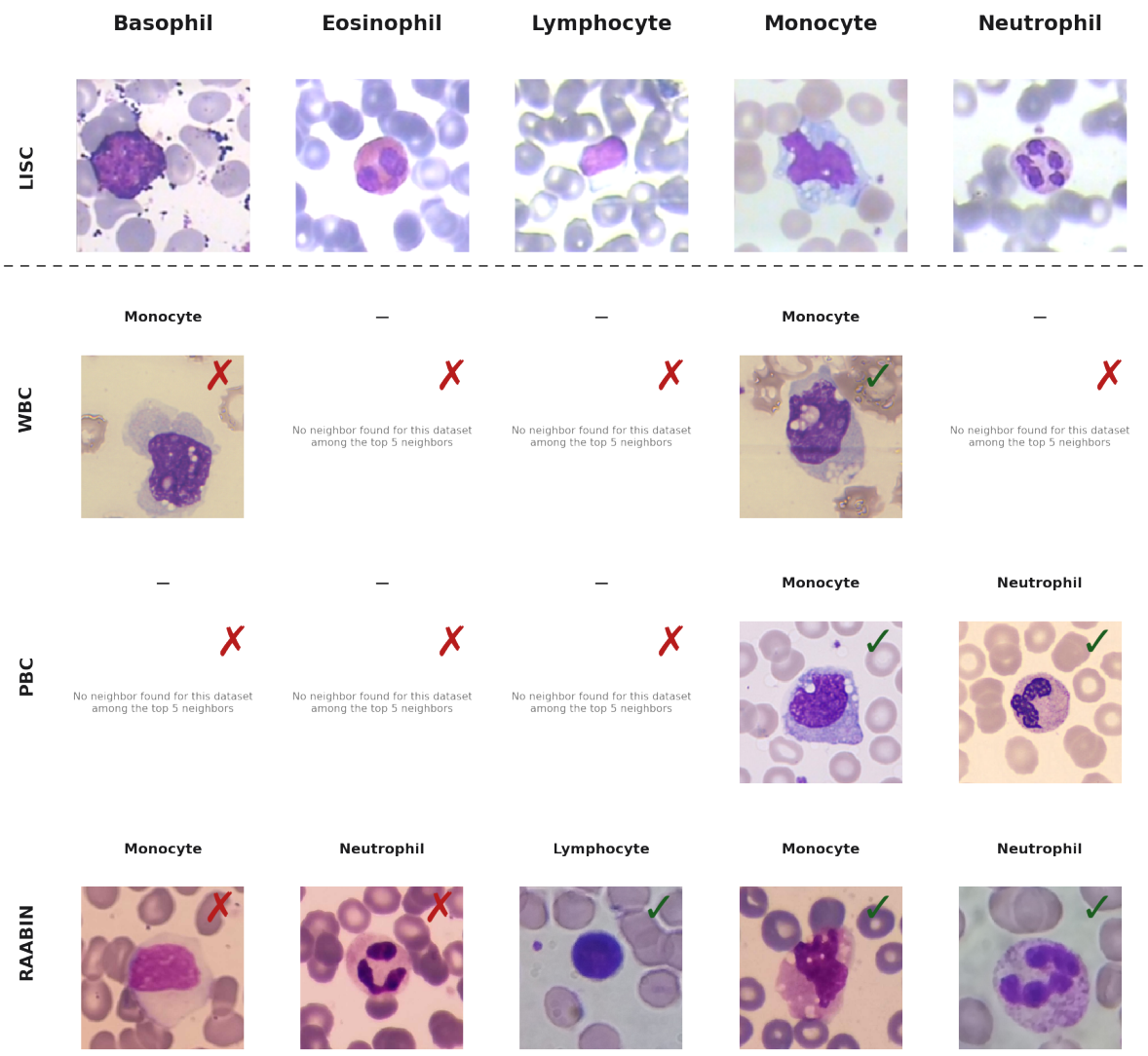

Figure 5: LISC Query image examples and their corresponding closest neighbors in each dataset when the three white-blood datasets WBC, PBC and RAABIN are combined as a large training datasets. The neighbors are found using the ensemble voting method and through a k-NN search, but without the use of OT to align training and LISC transfer embeddings. Checkmarks and cross at the top right of each neihbor image indicates if the image cell label corresponds to the query image label. When no image is displayed for one dataset, it means that the top 5-neighbors does not contain any cell image from this training dataset.

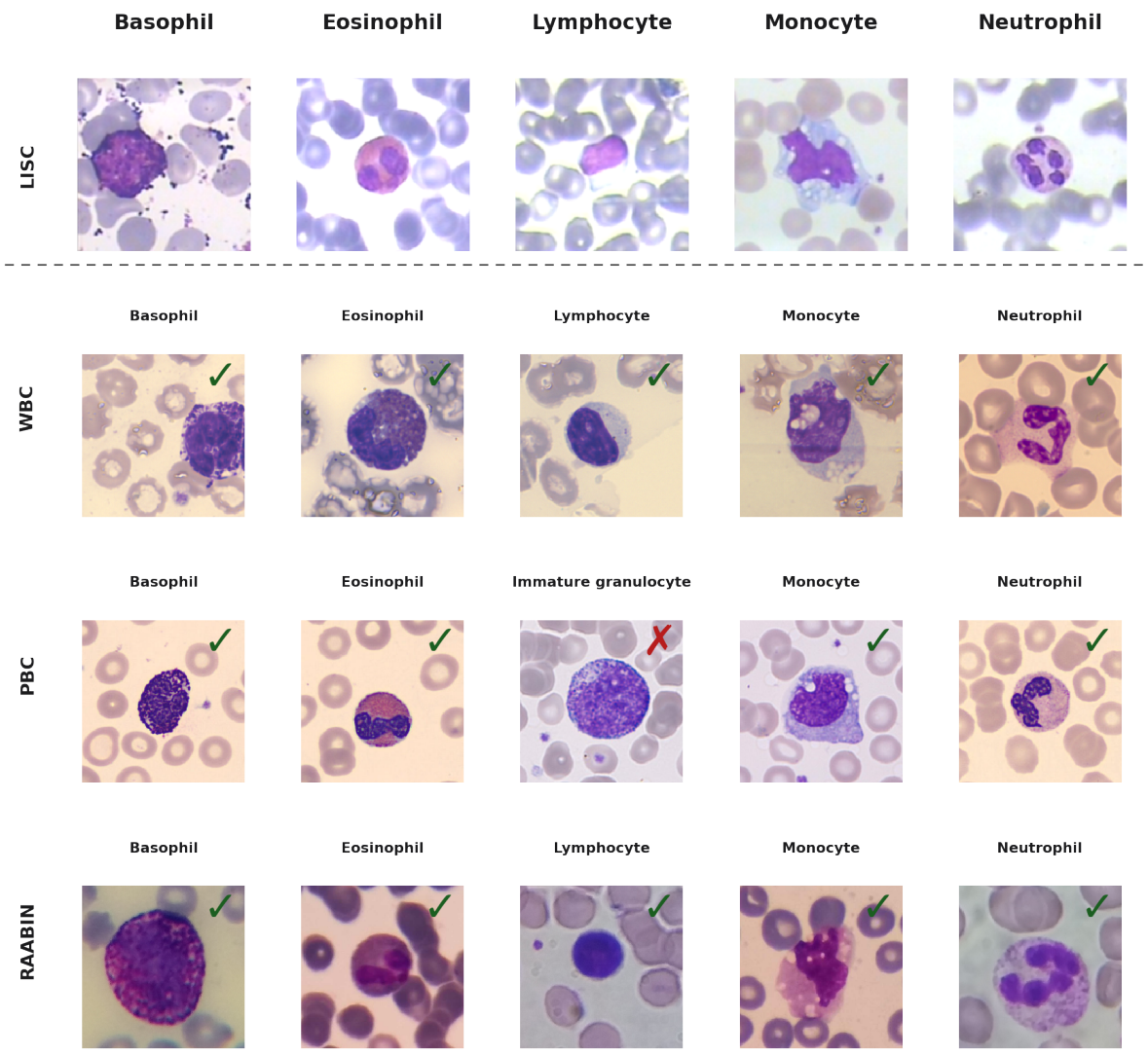

Figure 6: Same transfer query images as Fig. 5 but with the use of OT to align training and transfer embeddings. Checkmarks and cross at the top right of each neighbor image indicates if the image cell label corresponds to the query image label.

Table 7: Classification results with class filtering and with (✓) or without (✗) optimal transport (OT) averaged over all four train-to-transfer examples given from Table 3 to 6. Each Δ value (last two columns) represents the gain or loss in performance (in percentage points) from applying OT and for each encoder. For better clarity of the table, we do not display standard deviations over all datasets for each encoder; we just indicate it in parenthesis for the Ensemble Voting method. When combined to OT, our approach exhibits the lowest standard deviation results showing its robustness across various datasets. We also indicate the results for linear probing in the penultimate row. Last row (Δ_{OT}) shows the increase (positive values) or decrease (negative values) in accuracy and F1-score when using OT (optimal transport) averaged over all encoders.

Average over the 4 examples	Acc. (OT: ✗)	Acc. (OT: ✓)	F1 (OT: ✗)	F1 (OT: ✓)	Δ Acc.	Δ F1
SAM	27.24	**29.84**	21.39	**40.95**	2.6	19.56
DINO	34.77	**49.22**	46.72	**55.90**	14.44	9.18
ConvNeXt	46.27	**51.17**	**56.25**	56.19	4.9	-0.06
SWIN	30.52	**59.81**	38.17	**62.31**	29.29	24.14
CLIP	37.53	**44.67**	42.47	**48.05**	7.15	5.57
ViTMAE	**34.30**	30.70	19.96	**36.75**	-3.6	16.78
DinoBloom	63.86	**72.12**	66.67	**68.04**	8.25	1.37
CN + SWIN	34.06	**58.64**	39.39	**60.96**	24.58	21.57
SAM + CN + SWIN + CLIP	35.75	**60.43**	41.24	**62.31**	24.68	21.07
DINO + CN + ViTMAE + SWIN	35.34	**61.03**	46.07	**64.21**	25.69	18.14
CN + SWIN + ViTMAE	33.84	**58.79**	39.08	**61.19**	24.95	22.11
DINO + CN + SWIN	34.24	**61.05**	44.60	**64.15**	26.81	19.55
DINO + CN + ViTMAE	46.52	**55.32**	59.96	**62.57**	8.8	2.61
DINO + ViTMAE + SWIN	31.74	**58.98**	41.68	**62.22**	27.23	20.53
Ensemble Voting method	53.66 (6.86)	**71.25 (4.84)**	61.39 (8.71)	**73.67 (5.63)**	17.59	12.27
Ensemble Voting – lin. prob.	55.33 (12.76)	**63.74 (4.04)**	64.60 (11.04)	**69.40 (8.14)**	8.4	4.8
Average Δ_{OT}	-	-	-	-	16.23	14.29

Table 8: Effect of cell type filtering and optimal transport (OT) on the global accuracy and F1-score for the ensemble voting method.

Cell type filtering	OT	Acc.	F1 score	Acc. lin. prob	F1 score lin. prob.	Acc. NMTune	F1 score NMTune
✗	✗	54.54	64.91	57.89	70.09	50.59	63.37
✗	✓	64.10	72.52	59.03	68.83	–	–
✓	✗	62.21	68.94	59.84	69.34	54.80	65.28
✓	✓	**73.80**	**76.97**	63.61	70.88	–	–

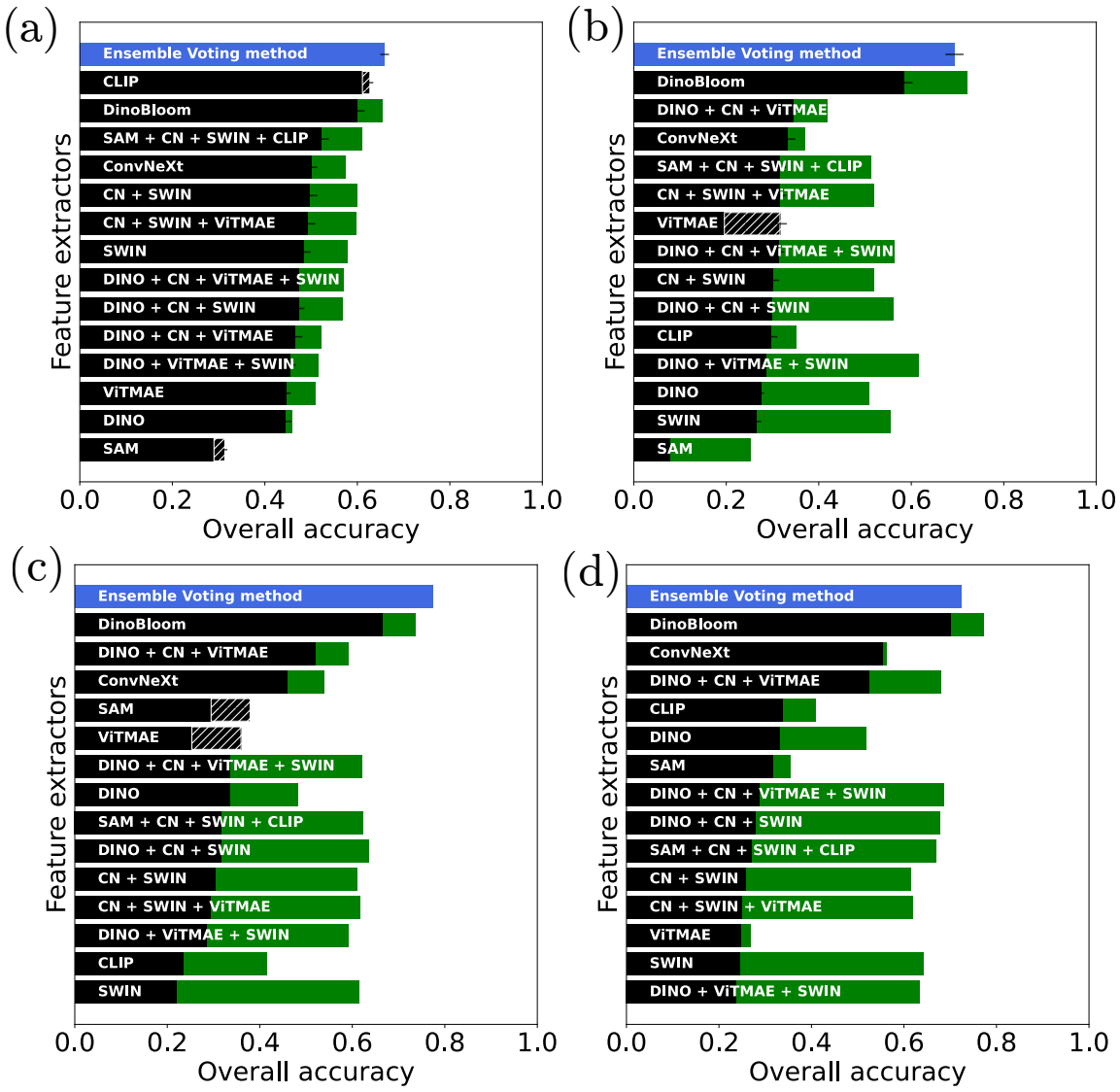

Figure 7: Accuracy results for all encoders and the ensemble voting method (including optimal transport) for the four transfer cases reported from Table 3 to Table 6: (a) Lionheart-to-Nanolive transfer (three classes), (b) PBC-WBC-to-RAABIN transfer, (c) PBC-WBC-to-LISC transfer, (d) PBC-WBC-RAABIN-to-LISC transfer. Black bars indicate the accuracy for each encoder without OT. White stripes bars shows a decrease in accuracy when combining the k-NN classifier with OT, while green bars represent on the contrary an increase in accuracy when applying OT.

6 Discussion

Results show that it is hard to predict which encoder or combination of encoders leads to the best cell classification. For instance, the transfer from Nanolive (NL) to Lionheart (LH) works well with CLIP with an accuracy above 60 % (see Table 3). However the accuracy drops to around 20-30% when using CLIP to classify LISC cells (see Tables 5 and 6). Only DinoBloom demonstrates robustness across datasets, which is expected since it is specifically trained on cell images. Nevertheless, even this foundation model experiences significant drops in accuracy when the classifier is trained with datasets absent from its training domain, such as PBC. For instance, when transferring from WBC to RAABIN, accuracy and F1 score can reach up to 90%; however, the accuracy and F1-score drop to 58.6% and 70.1%, when the training set becomes a mixture between PBC and WBC (see Table 4) and drops even more to 53.3% and 61.7% for PBC alone (see Table 10 in Appendix C).

On the contrary, our approach brings significant improvements. First, the use of optimal transport (OT) proves highly beneficial. By aligning datasets, OT reduces domain shifts caused by transfers across different acquisition devices and experimental conditions, and yields better performance and robustness across encoders. Over the four tables presented in Sec. 5.2, the use of OT results in average in a 16.2% increase in accuracy and 14.3% in F1-score. The best performances over all encoders and methods (underlined in the tables) always occur when OT is turned on. More over, while the performances of encoders significantly decrease when different training sets are mixed (see in particular Table 6), turning on OT greatly improves accuracies and F1-scores. To illustrate this point and the results from Table 6, we refer to Figs. 5 and 6. It shows five query images extracted from LISC and their closest neighbors found via k-NN search in each of the three datasets (WBC, PBC, RAABIN) when they are all combined in a large training dataset. One sees immediately that without OT (see Fig. 5) most of the neighbors are wrong and the richness of the three datasets is not used: the top-5 neighbors often contain neighbors from only one or two different datasets due to large domain shifts with the remaining one. When OT is applied (see Fig. 6), the accuracy is incredibly better: neighbor labels correspond to the query cell labels (except for one image) and neighbors are found in the three datasets since their feature representations are now well aligned. Our results confirm that OT enables the integration of multiple datasets from various sources, enriching the training set with new cell types without suffering from out-of-distribution problems. Beyond mitigating domain shifts, OT even boosts performance when heterogeneous datasets are combined. On LISC, accuracies remain around 70–73% when training on WBC or PBC separately, but rise to 77% when cells from both datasets are mixed (compare Tables 11 and 14 in Appendix C with Table 5).

This is also confirmed by the accuracy histograms shown in Figs. 7 for the different transfer cases considered in the result section 5.2. The green bars represent the accuracy gains obtained with OT, and they clearly dominate the histograms.

At variance to other encoders, the ensemble voting method mitigates accuracy fluctuations across encoders and always stands as one of the best classification results. It shows in particular a high robustness across very different datasets. This method also avoids the need to adapt the combination of encoders to each new dataset and, thus, ensures better generalization. Combined with OT, it demonstrates high stability – and even better accuracy – when training cells originate from multiple datasets. These results are supported by the histograms shown in Figs. 7 where the ensemble voting method combined with OT (blue bar) consistently ranks among the top-performing approaches.

Overall, our ensemble approach with class filtering and OT leads to a mean accuracy of 73.8% and a F1-score of 77% across all datasets (see Table 7). It is by far the best performance and demonstrates the highest stability (assessed by the lowest accuracy and F1-score standard deviations over all datasets). As shown in Table 8, where we compare the ensemble voting method when including or not class filtering and OT, both preprocessing steps increase by almost 16% the overall accuracy and F1-score; OT alone (after class filtering) accounts for roughly a 9% improvement in performance and, thus, highly contributes to a better generalization of cell classification across different devices.

Linear probing without OT leads to similar classification results as the k-NN classifier, but with a higher variance. As shown in Table 7, it also highly benefits from OT, but does not reach the k-NN classifier performances.

Shared cell type identification is also a crucial preprocessing step. As shown in Sec. 5.1, our cluster detection method is always able to identify all shared cell types while filtering out most spurious classes; the node-degree–based cluster detection approach (denoted by 'ours' in Tables 1 and 2) proves to be the most effective. This preprocessing ensures that only the common domain between the training and transfer datasets is retained, which in turn directly improves cell classification. Without OT, class filtering contributes to a performance gain of about 7% overall (see Table 8). The efficiency of OT also depends on this step: by restricting OT to shared classes, the alignment focuses on the relevant common patterns between training and transfer cell embeddings, resulting in a more accurate mapping (+1.2% overall, see Table 8).

7 Conclusion

We introduced a robust, fine-tuning-free pipeline for cross-domain cell image classification that directly addresses the two obstacles pervading computational cytology: domain shift across imaging conditions and the lack of labels in newly acquired datasets. Instead of retraining or adapting foundation models for every new dataset—an option rarely affordable in biomedical settings—our approach operates entirely at the embedding level and combines four ingredients: multi-encoder feature extraction from complementary foundation models, an unsupervised identification of shared cell types based on a dedicated graph cluster-detection algorithm, the filtering of non-overlapping classes, and a graph-regularized (Fused Gromov–Wasserstein) optimal transport (OT) alignment of the training and transfer embedding spaces, followed by an ensemble voting classifier. To the best of our knowledge, this is the first method to couple unsupervised shared-class discovery, structure-aware OT alignment, and multi-encoder ensembling into a single classifier-agnostic and retraining-free pipeline for cell imaging.

Validated on four white-blood-cell datasets (WBC, PBC, RAABIN, LISC) and two live-cell datasets (Nanolive, Lionheart), the method consistently alleviates the domain shifts induced by different microscopes, staining protocols, and acquisition devices. Combining class filtering with OT improves the mean accuracy by roughly 20% over all datasets considered, with OT alone accounting for an average gain of 16.2% in accuracy and 14.3% in F1-score across the four transfer scenarios; for every encoder, the best performance is always obtained with OT enabled. Crucially, our ensemble pipeline attains the highest accuracy and F1-score (73.8% and 77%) while exhibiting the lowest variance across datasets, and its advantage grows when heterogeneous datasets are pooled into a single training corpus—precisely the regime where naive transfer degrades most.

Because the pipeline is agnostic to the underlying classifier and requires neither labels in the target domain nor any fine-tuning of the foundation models, it remains lightweight and readily portable to new imaging conditions. Since each cell is classified by a vote across several encoders, their level of agreement could serve as a per-cell confidence score, automatically flagging low-confidence cells for expert review and concentrating human effort where the model is least reliable.

References

Andrea Acevedo, Anna Merino, Santiago Alférez, Ángel Molina, Laura Boldú, and José Rodellar. A dataset of microscopic peripheral blood cell images for development of automatic recognition systems. *Data in Brief*, 30:105474, 2020. ISSN 2352-3409. doi: https://doi.org/10.1016/j.dib.2020.105474. URL `https://www.sciencedirect.com/science/article/pii/S2352340920303681`.

David Alvarez-Melis and Tommi Jaakkola. Gromov-Wasserstein Alignment of Word Embedding Spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1881–1890, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1214. URL `http://aclweb.org/anthology/D18-1214`.

S. P. Arango, F. Ferreira, A. Kadra, F. Hutter, and J Grabocka. Quick-tune: Quickly learning which pretrained model to finetune and how. In *The Twelfth International Conference on Learning Representations*, 2024.

Martin Arjovsky, Soumith Chintala, and L\'{e}on Bottou. Wasserstein generative adversarial networks. pp. 214–223, 2017.

Jean-David Benamou, Guillaume Carlier, Marco Cuturi, Luca Nenna, and Gabriel Peyré. Iterative Bregman Projections for Regularized Transportation Problems. *SIAM Journal on Scientific Computing*, 37(2): A1111–A1138, January 2015. ISSN 1064-8275, 1095-7197. doi: 10.1137/141000439. URL http://epubs.siam.org/doi/10.1137/141000439.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10): P10008, 2008. ISSN 1742-5468. doi: 10.1088/1742-5468/2008/10/P10008. URL https://iopscience.iop.org/article/10.1088/1742-5468/2008/10/P10008.

Alexandra Bodzas, Pavel Kodytek, and Jan Zidek. A high-resolution large-scale dataset of pathological and normal white blood cells. *Scientific Data*, 10(1):466, July 2023. ISSN 2052-4463. doi: 10.1038/s41597-023-02378-7. URL https://www.nature.com/articles/s41597-023-02378-7.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, S. Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen A. Creel, Jared Davis, Dora Demszky, Chris Donahue, Moussa Koulako Bala Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas F. Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, O. Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Benjamin Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, J. F. Nyarko, Giray Ogut, Laurel J. Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Robert Reich, Hongyu Ren, Frieda Rong, Yusuf H. Roohani, Camilo Ruiz, Jack Ryan, Christopher R'e, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishna Parasuram Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei A. Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models. *ArXiv*, abs/2108.07258, 2021. URL https://api.semanticscholar.org/CorpusID:237091588.

Marie Breeur, George Stepaniants, Pekka Keski-Rahkonen, Philippe Rigollet, and Vivian Viallon. Optimal transport for automatic alignment of untargeted metabolomic data. *eLife*, 12:RP91597, jun 2024. ISSN 2050-084X. doi: 10.7554/eLife.91597. URL https://doi.org/10.7554/eLife.91597.

Charlotte Bunne, David Alvarez-Melis, Andreas Krause, and Stefanie Jegelka. Learning generative models across incomparable spaces. In *International Conference on Machine Learning*, 2019. URL https://api.semanticscholar.org/CorpusID:153312722.

Biao Cai, Lina Zeng, Yanpeng Wang, Hongjun Li, and Yanmei Hu. Community Detection Method Based on Node Density, Degree Centrality, and K-Means Clustering in Complex Network. *Entropy*, 21(12):1145, 2019. ISSN 1099-4300. doi: 10.3390/e21121145. URL https://www.mdpi.com/1099-4300/21/12/1145.

Zixuan Cang and Qing Nie. Inferring spatial and signaling relationships between cells from single cell transcriptomic data. *Nature Communications*, 11(1):2084, April 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-15968-5. URL https://www.nature.com/articles/s41467-020-15968-5.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jegou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9630–9640, 2021. doi: 10.1109/ICCV48922.2021.00951.

Hao Chen, Jindong Wang, Ankit Shah, Ran Tao, Hongxin Wei, Xing Xie, Masashi Sugiyama, and Bhiksha Raj. Understanding and mitigating the label noise in pre-training on downstream tasks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=TjhUtloBZU.

Samir Chowdhury and Tom Needham. Gromov-Wasserstein Averaging in a Riemannian Framework. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 3676–3684, Seattle, WA, USA, June 2020a. IEEE. ISBN 978-1-72819-360-1. doi: 10.1109/CVPRW50498.2020.00429. URL https://ieeexplore.ieee.org/document/9151015/.

Samir Chowdhury and Tom Needham. Generalized Spectral Clustering via Gromov-Wasserstein Learning. *ArXiv*, abs/2006.04163, 2020b.

Pinar Demetci, Rebecca Santorella, Manav Chakravarthy, Bjorn Sandstede, and Ritambhara Singh. SCOTv2: Single-Cell Multiomic Alignment with Disproportionate Cell-Type Representation. *Journal of Computational Biology*, 29(11):1213–1228, November 2022a. ISSN 1557-8666. doi: 10.1089/cmb.2022.0270. URL https://www.liebertpub.com/doi/10.1089/cmb.2022.0270.

Pinar Demetci, Rebecca Santorella, Björn Sandstede, William Stafford Noble, and Ritambhara Singh. SCOT: Single-Cell Multi-Omics Alignment with Optimal Transport. *Journal of Computational Biology*, 29(1):3–18, January 2022b. ISSN 1557-8666. doi: 10.1089/cmb.2021.0446. URL https://www.liebertpub.com/doi/10.1089/cmb.2021.0446.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. 2024.

Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. URL http://jmlr.org/papers/v22/20-451.html.

Fengjiao Gong, Yuzhou Nie, and Hongteng Xu. Gromov-Wasserstein Multi-modal Alignment and Clustering. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 603–613, Atlanta GA USA, October 2022. ACM. ISBN 978-1-4503-9236-5. doi: 10.1145/3511808.3557339. URL https://dl.acm.org/doi/10.1145/3511808.3557339.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15979–15988, 2022. doi: 10.1109/CVPR52688.2022.01553.

Uriah Israel, Markus Marks, Rohit Dilip, Qilin Li, Changhua Yu, Emily Laubscher, Ahamed Iqbal, Elora Pradhan, Ada Ates, Martin Abt, Caitlin Brown, Edward Pao, Shenyi Li, Alexander Pearson-Goulart, Pietro Perona, Georgia Gkioxari, Ross Barnowski, Yisong Yue, and David Van Valen. CellSAM: A Foundation Model for Cell Segmentation, November 2023. URL http://biorxiv.org/lookup/doi/10.1101/2023.11.17.567630.

Gabriel Kalweit, Anusha Klett, Mehdi Naouar, Jens Rahnfeld, Yannick Vogt, Diana Laura Infante Ramirez, Rebecca Berger, Jesus Duque Afonso, Tanja Nicole Hartmann, Marie Follo, Michael Luebbert, Roland Mertelsmann, Evelyn Ullrich, Joschka Boedecker, and Maria Kalweit. Unsupervised feature extraction from a foundation model zoo for cell similarity search in oncological microscopy across devices. In *ICML 2024 Workshop on Foundation Models in the Wild*, 2024. URL https://openreview.net/forum?id=wctboDvy9f.

Gabriel Kalweit, Anusha Klett, Paula Silvestrini, Jens Rahnfeld, Mehdi Naouar, Yannick Vogt, Diana Infante, Rebecca Berger, Jesús Duque-Afonso, Tanja Nicole Hartmann, Marie Follo, Elitsa Bodurova-Spassova, Michael Lübbert, Roland Mertelsmann, Joschka Boedecker, Evelyn Ullrich, and Maria Kalweit. Leveraging a foundation model zoo for cell similarity search in oncological microscopy across devices. *Frontiers in Oncology*, Volume 15 - 2025, 2025. ISSN 2234-943X. doi: 10.3389/fonc.2025.1480384. URL https://www.frontiersin.org/journals/oncology/articles/10.3389/fonc.2025.1480384.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3992–4003, 2023. doi: 10.1109/ICCV51070.2023.00371.

Valentin Koch, Sophia J. Wagner, Salome Kazeminia, Ece Sancar, Matthias Hehr, Julia A. Schnabel, Tingying Peng, and Carsten Marr. DinoBloom: A Foundation Model for Generalizable Cell Embeddings in Hematology. In Marius George Linguraru, Qi Dou, Aasa Feragen, Stamatia Giannarou, Ben Glocker, Karim Lekadir, and Julia A. Schnabel (eds.), *Medical Image Computing and Computer Assisted Intervention – MICCAI 2024*, volume 15012, pp. 520–530. Springer Nature Switzerland, 2024. ISBN 978-3-031-72389-6 978-3-031-72390-2. doi: 10.1007/978-3-031-72390-2_49. URL https://link.springer.com/10.1007/978-3-031-72390-2_49. Series Title: Lecture Notes in Computer Science.

Seyedeh-Zahra Mousavi Kouzehkanan, Sepehr Saghari, Eslam Tavakoli, Peyman Rostami, Mohammadjavad Abaszadeh, Esmaeil Shahabi Satlsar, Farzaneh Mirzadeh, Maryam Gheidishahran, Fatemeh Gorgi, Saeed Mohammadi, and Reshad Hosseini. Raabin-wbc: a large free access dataset of white blood cells from normal peripheral blood. *bioRxiv*, 2021. doi: 10.1101/2021.05.02.442287. URL https://www.biorxiv.org/content/early/2021/05/04/2021.05.02.442287.

Oren Kraus, Kian Kenyon-Dean, Saber Saberian, Maryam Fallah, Peter McLean, Jess Leung, Vasudev Sharma, Ayla Khan, Jia Balakrishnan, Safiye Celik, Dominique Beaini, Maciej Sypetkowski, Chi Vicky Cheng, Kristen Morse, Maureen Makes, Ben Mabey, and Berton Earnshaw. Masked Autoencoders for Microscopy are Scalable Learners of Cellular Biology. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11757–11768, Seattle, WA, USA, 2024. IEEE. ISBN 9798350353006. doi: 10.1109/CVPR52733.2024.01117. URL https://ieeexplore.ieee.org/document/10657479/.

Xinhang Li, Zhaopeng Qiu, Xiangyu Zhao, Zihao Wang, Yong Zhang, Chunxiao Xing, and Xian Wu. Gromov-Wasserstein Guided Representation Learning for Cross-Domain Recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 1199–1208, Atlanta GA USA, October 2022. ACM. ISBN 978-1-4503-9236-5. doi: 10.1145/3511808.3557338. URL https://dl.acm.org/doi/10.1145/3511808.3557338.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows . In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9992–10002, Los Alamitos, CA, USA, October 2021. IEEE Computer Society. doi: 10.1109/ICCV48922.2021.00986. URL https://doi.ieeecomputersociety.org/10.1109/ICCV48922.2021.00986.

Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11999–12009, 2022a. doi: 10.1109/CVPR52688.2022.01170.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s . In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11966–11976, Los Alamitos, CA, USA, June 2022b. IEEE Computer Society. doi: 10.1109/CVPR52688.2022.01167. URL https://doi.ieeecomputersociety.org/10.1109/CVPR52688.2022.01167.

Jun Ma, Yuting He, Feifei Li, Lin Han, Chenyu You, and Bo Wang. Segment anything in medical images. *Nature Communications*, 15(1):654, January 2024. ISSN 2041-1723. doi: 10.1038/s41467-024-44824-z. URL https://www.nature.com/articles/s41467-024-44824-z.

Nabil El Malki, Robin Cugny, Olivier Teste, and Franck Ravat. DECWA : Density-Based Clustering using Wasserstein Distance. 2023. doi: 10.48550/ARXIV.2310.16552. URL https://arxiv.org/abs/2310.16552. Publisher: arXiv Version Number: 1.

Igor Melnyk, Youssef Mroueh, Brian Belgodere, Mattia Rigotti, Apoorva Nitsure, Mikhail Yurochkin, Kristjan Greenewald, Jiri Navratil, and Jarret Ross. Distributional preference alignment of LLMs via optimal transport. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=2LctgfN6Ty.

Facundo Memoli. On the use of Gromov-Hausdorff Distances for Shape Comparison. In M. Botsch, R. Pajarola, B. Chen, and M. Zwicker (eds.), *Eurographics Symposium on Point-Based Graphics*. The Eurographics Association, 2007. ISBN 978-3-905673-51-7. doi: /10.2312/SPBG/SPBG07/081-090.

Facundo Mémoli. Gromov–Wasserstein Distances and the Metric Approach to Object Matching. *Foundations of Computational Mathematics*, 11(4):417–487, August 2011a. ISSN 1615-3375, 1615-3383. doi: 10.1007/s10208-011-9093-5. URL http://link.springer.com/10.1007/s10208-011-9093-5.

Facundo Mémoli. A spectral notion of gromov–wasserstein distance and related methods. *Applied and Computational Harmonic Analysis*, 30(3):363–401, 2011b. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2010.09.005. URL https://www.sciencedirect.com/science/article/pii/S1063520310001107.

Mor Nitzan, Nikos Karaiskos, Nir Friedman, and Nikolaus Rajewsky. Gene expression cartography. *Nature*, 576(7785):132–137, December 2019. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-019-1773-3. URL https://www.nature.com/articles/s41586-019-1773-3.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=a68SUt6zFt. Featured Certification.

Peter Pao-Huang, Kyra Thrush-Evensen, Daniel Montemayor, Cyril Lagger, and Morgan Levine. Unpaired single-cell dataset alignment with wavelet optimal transport, 2025. URL https://openreview.net/forum?id=BYWVwmbqwK.

Gabriel Peyré, Marco Cuturi, and Justin Solomon. Gromov-wasserstein averaging of kernel and distance matrices. pp. 2664–2672, 2016.

Ramon Pfaendler, Jacob Hanimann, Sohyon Lee, and Berend Snijder. Self-supervised vision transformers accurately decode cellular state heterogeneity, January 2023. URL http://biorxiv.org/lookup/doi/10.1101/2023.01.16.524226.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021. URL https://api.semanticscholar.org/CorpusID:231591445.

Seyed Hamid Rezatofighi and Hamid Soltanian-Zadeh. Automatic recognition of five types of white blood cells in peripheral blood. *Computerized Medical Imaging and Graphics*, 35(4):333–343, 2011. ISSN 0895-6111. doi: https://doi.org/10.1016/j.compmedimag.2011.01.003. URL https://www.sciencedirect.com/science/article/pii/S0895611111000048.

Geoffrey Schiebinger, Jian Shu, Marcin Tabaka, Brian Cleary, Vidya Subramanian, Aryeh Solomon, Joshua Gould, Siyan Liu, Stacie Lin, Peter Berube, Lia Lee, Jenny Chen, Justin Brumbaugh, Philippe Rigollet, Konrad Hochedlinger, Rudolf Jaenisch, Aviv Regev, and Eric S. Lander. Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming. *Cell*, 176(4):928–943.e22, 2019. ISSN 0092-8674. doi: https://doi.org/10.1016/j.cell.2019.01.006. URL https://www.sciencedirect.com/science/article/pii/S009286741930039X.

Bernhard Schmitzer and Christoph Schnörr. Modelling Convex Shape Priors and Matching Based on the Gromov-Wasserstein Distance. *Journal of Mathematical Imaging and Vision*, 46(1):143–159, 2013. ISSN 0924-9907, 1573-7683. doi: 10.1007/s10851-012-0375-6. URL http://link.springer.com/10.1007/s10851-012-0375-6.

Grzegorz Skoraczyński, Anna Gambin, and Błażej Miasojedow. Alignstein: Optimal transport for improved LC-MS retention time alignment. *GigaScience*, 11:giac101, November 2022. ISSN 2047-217X. doi: 10.1093/gigascience/giac101. URL https://academic.oup.com/gigascience/article/doi/10.1093/gigascience/giac101/6795291.

Karl-Theodor Sturm. The space of spaces: curvature bounds and gradient flows on the space of metric measure spaces, 2012. URL https://arxiv.org/abs/1208.0434. Version Number: 2.

Alexis Thual, Huy Tran, Tatiana Zemskova, Nicolas Courty, Rémi Flamary, Stanislas Dehaene, and Bertrand Thirion. Aligning individual brains with Fused Unbalanced Gromov-Wasserstein. 2022. URL https://openreview.net/forum?id=vy7B8z0-4D.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein Auto-Encoders. 2018. URL https://openreview.net/forum?id=HkL7n1-0b.

Titouan Vayer, Laetitia Chapel, Rémi Flamary, Romain Tavenard, and Nicolas Courty. Optimal Transport for structured data with application on graphs, 2018. URL https://arxiv.org/abs/1805.09114. Version Number: 3.

Cedric Villani. *Topics in optimal transportation*. Graduate studies in mathematics ; v. 58. American Mathematical Society, Providence, R.I, 2003. ISBN 082183312X.

Cédric Villani. *Optimal Transport*, volume 338 of *Grundlehren der mathematischen Wissenschaften*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-71049-3 978-3-540-71050-9. doi: 10.1007/978-3-540-71050-9. URL http://link.springer.com/10.1007/978-3-540-71050-9.

Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. Convnext v2: Co-designing and scaling convnets with masked autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16133–16142, June 2023.

Hongteng Xu, Dixin Luo, Hongyuan Zha, and Lawrence Carin. Gromov-wasserstein learning for graph matching and node embedding. *ArXiv*, abs/1901.06003, 2019. URL https://api.semanticscholar.org/CorpusID:58028808.

Hongteng Xu, Dixin Luo, Ricardo Henao, Svati Shah, and Lawrence Carin. Learning autoencoders with relational regularization. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10576–10586. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/xu20e.html.

Karren Dai Yang, Karthik Damodaran, Saradha Venkatachalapathy, Ali C. Soylemezoglu, G. V. Shivashankar, and Caroline Uhler. Predicting cell lineages using autoencoders and optimal transport. *PLOS Computational Biology*, 16(4):e1007828, April 2020. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1007828. URL https://dx.plos.org/10.1371/journal.pcbi.1007828.

Karren Dai Yang, Anastasiya Belyaeva, Saradha Venkatachalapathy, Karthik Damodaran, Abigail Katcoff, Adityanarayanan Radhakrishnan, G. V. Shivashankar, and Caroline Uhler. Multi-domain translation between single-cell imaging and sequencing data using autoencoders. *Nature Communications*, 12(1): 31, January 2021. ISSN 2041-1723. doi: 10.1038/s41467-020-20249-2. URL `https://www.nature.com/articles/s41467-020-20249-2`.

A Cluster detection algorithm and similarity score vectors

A.1 Our cluster detection algorithm

In this appendix, we detail the cluster detection algorithm we used to group cells together in an unsupervised fashion. Let us consider an unlabeled test dataset $\mathcal{D}_{\text{test}}$ containing M cell images. The main idea is to build a weighted graph using cell sample embeddings $\mathcal{E}_j^{\text{test, enc}}$. These embeddings can originate from one foundation model (in the main text we always use DinoBloom – since it is specifically trained on cell images), or can result from the concatenation of several encoders.

A weighted graph $\mathcal{G}$ is always represented by a couple $(\mathcal{V}, e)$, where $\mathcal{V}$ is a set of vertices (the nodes) and e is a set of edges that connect certain nodes together. In a weighted graph, each edge has a weight that represents the strength of the bound between the two connected nodes. Therefore, edge weights can be used as features to capture similarities, correlations or any other meaningful information between vertices; if one wants for instance to obtain a meaningful representation of a subway network, vertices can represent metro stations, and edge weights could be the distance between two connected stations.

In our case, each vertex is a cell sample. Vertices should be connected only if cell samples are similar enough, and edge weights should quantify those similarities. To meet this requirement, we first compute the similarity matrix defined as:

$$W_{ij} \equiv \frac{d_{\text{geo}}(x_i, x_j)}{\max\limits_{i,j} d_{\text{geo}}(x_i, x_j)}, \tag{6}$$

where $d_{\text{geo}}(x_i, x_j)$ is the geodesic distance, i.e., the shortest path between pairs of nodes on the kNN graph created from the M sample embeddings $\mathcal{E}_j^{\text{test, enc}}$. As a result, the similarity between cell embeddings is translated to a distance, that we can then use to create a weighted graph. The normalized weights (between 0 and 1) are then given by:

$$e_{ij} = \max\left(0,\, 1 - \frac{W_{ij} - \min W}{\sigma - \min W}\right), \text{ if } i \neq j, \text{ else } e_{ii} = 0, \tag{7}$$

where σ is a threshold above which $e_{ij} = 0$. In other words, if W_{ij} becomes greater than σ, samples i and j are not connected in the graph. This puts a bound on the maximal geodesic distance, and thus on the maximal dissimilarity that we tolerate between two cells. In our algorithm, we take $\sigma = 0.3$ if $M < 800$, otherwise $\sigma = 0.15$. Note that $e_{ii} = 0$ just means that we do not accept self loops.

We mention that we could have directly used the kNN graph, where in this case edge weights would have been given by the Euclidean distance between embeddings. However, in kNN graphs, the number of neighbors is given by k and is constant over all nodes. Relying on the geodesic distance with a threshold σ, enables the number of neighbors per node to vary: it results in very dense areas with many interconnected nodes, and less dense areas with a low number of neighbors per node – usually between two clusters. Therefore, contrary to a kNN graph where clusters can be visible but are not necessarily distinct from one another, weights given by equation 7 render clusters more distinct, and thus leads to a better detection. As a sanity check and to confirm the above comment, we have tested cluster detection and identification of shared cell types using both graph methods. The graph with edge weights given by equation 7 results in an overall increase of +7.1% compared to the case where we directly use the kNN graph as input of both cluster detection algorithms (ours and Louvain community detection algorithm).

In the spirit of the community detection method detailed in Ref. Cai et al. (2019), our own algorithm is based on the following quantities:

- **The weighted node degree**. This quantity is defined for each node i and corresponds to the sum of edge weights of all neighbors of the node:

$$d_i = \sum_{j \in \mathcal{N}_i} e_{ij}, \tag{8}$$

where $\mathcal{N}_i$ is a the set of neighbors of node i. The more central is a node in a cluster, the higher will be this quantity. This will be the key parameter to identify central nodes of clusters.

- **The node density**. This quantity is defined by the number of edges and nodes in the subgraph created from h-hops from this node – a hop is a jump from one node to a neighbor node:

$$D_i^{(h)} = \frac{|e^{(h)}|}{|\mathcal{V}^{(h)}|(|\mathcal{V}^{(h)}|-1)/2}, \tag{9}$$

where $\mathcal{V}^{(h)}$ is the set of vertices and $e^{(h)}$ the set of weighted edges of the subgraph $\mathcal{G}^{(h)}$ with h hops. This quantity is related to the location of a node in the graph. Nodes at the center of a cluster or only connected to one cluster have a high density, while those sharing connections with several clusters have a low density. This helps identifying boundary (intercluster) nodes between two clusters.

We always normalize the node density column vector $D^{(h)} = \left(D_0^{(h)}, \ldots, D_{M-1}^{(h)}\right)^T$ between 0 and 1, such that the lowest node density is zero, and the maximal node density is bounded by 1.

- **Jaccard matrix.** The Jaccard coefficient of two nodes is proportional to the common neighbors between them:

$$J_{ij} = \frac{|\mathcal{N}_i \cap \mathcal{N}_j|}{|\mathcal{N}_i \cup \mathcal{N}_j|}, \tag{10}$$

where $\mathcal{N}_i$ ($\mathcal{N}_j$) is a the set of neighbors of node i (j). Nodes that belong to the same cluster share many neighbors, and, thus, exhibit high Jaccard coefficients, while boundary nodes between clusters or distant nodes have low or zero Jaccard coefficients. The Jaccard matrix can be thought as a correlation matrix at the level of the graph structure.

- **The node correlation matrix**. This matrix fuses information from node densities and Jaccard coefficients:

$$\mathcal{C}^{\text{nodes}} = (D^{(h)} \cdot D^{(h)\,T}) \cdot J, \tag{11}$$

where the superscript T stands for transpose, and $(\cdot)$ is the usual matrix multiplication. Note that because $D^{(h)}$ is a M column vector (see above), the quantity $(D^{(h)} \cdot D^{(h)\,T})$ is a $M \times M$ matrix.

We also define some useful notations that will be used throughout this section:

- The weighted node degree vector $\mathbf{d} = (d_0, \ldots, d_{M-1})^T$.
- Each detected cluster will be denoted as c_i.
- The central nodes, which will be selected as part of cluster c_i, will be denoted as $n_j^{(c_i)}$ and stored in the set $\mathcal{S}_{\text{cen}}^{(c_i)} = \{n_0^{(c_i)}, \ldots, n_{N_i}^{(c_i)}\}$, where N_i is the total number of nodes in cluster c_i.
- All neighbors nodes of $\mathcal{S}_{\text{cen}}^{(c_i)}$ will be stored in a set $\mathcal{N}_{(c_i)}$.
- The average correlation coefficient for a node k to all nodes in cluster c_i is

$$\overline{\mathcal{C}_k}^{(c_i)} \equiv \underset{l \in \mathcal{S}_{\text{cen}}^{(c_i)}}{\text{mean}} \left[\mathcal{C}_{kl}^{\text{nodes}}\right] \tag{12}$$

- The average similarity weight coefficient for node k to all nodes in cluster c_i is:

$$\overline{W_k}^{(c_i)} \equiv \underset{l \in \mathcal{S}_{\text{cen}}^{(c_i)}}{\text{mean}} [W_{kl}] \tag{13}$$

- The average correlation vector $\overline{\mathcal{C}}^{(h,\, c_i)} = (\overline{\mathcal{C}_0}^{(c_i)}, \ldots)^T$ for all h-hop neighbors of a given central node in $\mathcal{S}_{\text{cen}}^{(c_i)}$.

- The average similarity weight vector $\overline{W}^{(h,\, c_i)} = (\overline{W_0}^{(c_i)}, \ldots)^T$ for all h-hop neighbors of a given central node in $\mathcal{S}_{\text{cen}}^{(c_i)}$.

- A vector $\mathcal{W}_{\mathcal{S}_{\text{cen}}^{(c_i)}}$ which is constructed iteratively as new central nodes are added to $\mathcal{S}_{\text{cen}}^{(c_i)}$. This vector is defined as follows: whenever a new node k is added to $\mathcal{S}_{\text{cen}}^{(c_i)}$, we append to $\mathcal{W}_{\mathcal{S}_{\text{cen}}^{(c_i)}}$ the quantity $\overline{W_k}^{(c_i)}$, where the average is computed over all already present central nodes in $\mathcal{S}_{\text{cen}}^{(c_i)}$. We can also write this vector explicitly after j central nodes have been added to $\mathcal{S}_{\text{cen}}^{(c_i)}$:

$$\mathcal{W}_{\mathcal{S}_{\text{cen}}^{(c_i)}} = \left(\underset{l \in \mathcal{S}_{\text{cen}}^{(c_i)} = \{n_0^{(c_i)}\}}{\text{mean}} [W_{1l}], \ldots, \underset{l \in \mathcal{S}_{\text{cen}}^{(c_i)} = \{n_0^{(c_i)}, \ldots, n_{j-1}^{(c_i)}\}}{\text{mean}} [W_{jl}] \right)^T . \tag{14}$$

  Intuitively we can understand this vector as the following: As we fill $\mathcal{S}_{\text{cen}}^{(c_i)}$ with new nodes, we select nodes further and further away from the initial central nodes. Therefore, we expect that similarity weight coefficients W_{ij} between new central nodes and those already present in $\mathcal{S}_{\text{cen}}^{(c_i)}$ will slowly increase. Indeed, remember that these weights are directly proportional to the similarity between cells. Therefore, as we move far away from the initial nodes, similarity between samples will fade away, which in turn will translate to an increase of the geodesic distance, and thus of the similarity weights. Furthermore, as one moves from one cell type to another, the similarity weight coefficients associated with the original cell-type cluster will exhibit abrupt changes, since samples from the new cell type have low correlation with samples from the original cluster. As explained below, these jumps in weights will be used to detect new clusters.

In short, the algorithm works as follows:

1. For the very first iteration, just locate the node with the maximal weighted node degree and add it to $\mathcal{S}_{\text{cen}}^{(c_1)}$. We denote it as $n_0^{(c_1)}$.

2. Then, set $d_{n_0^{(c_1)}}$ to zero.

3. To select the next central node, consider all 2-hop neighbors of $n_0^{(c_1)}$ and compute $\overline{W}^{(h=2,\, c_1)}$. To only consider the most correlated nodes with $n_0^{(c_1)}$, filter all nodes k for which

$$\overline{W}_k^{(h=2,\, c_1)} \geq \underset{k}{\text{mean}} \left[\overline{W}_k^{(h=2,\, c_1)} \right]. \tag{15}$$

4. Select the node with the maximal weighted node degree among the remaining 2-hop nodes from $n_0^{(c_1)}$. Note that $n_0^{(c_1)}$ is still present at this stage and this is the reason why we previously set its node degree to zero. We could have also excluded this node.

   After several iterations, it may happen that all remaining nodes have already been selected as central nodes and stored in $\mathcal{S}_{\text{cen}}^{(c_i)}$. In this case, every node has a degree equal to zero, so the maximal node degree is also zero and no further node can be selected. To address this, whenever this situation occurs, we restart from step 3 but now include all 3-hop neighbors of the last added central node. If no node is selected at this stage, we progressively enlarge the neighborhood up to 7 hops. If, after this process, no node is still selected and the maximal node degree remains zero among the neighborhood, this indicates that the cluster is likely complete, or that all clusters have already been detected – since, except in very large graphs, a 7-hop neighborhood typically covers nearly the entire graph. In such cases, we can proceed directly to step 9.

5. Add this node to $\mathcal{S}_{\text{cen}}^{(c_1)}$ and set $d_{n_1^{(c_1)}}$ to zero.

6. Then, store in $\mathcal{N}_{(c_1)}$ neighbors k of $n_0^{(c_1)}$ for which $\overline{\mathcal{C}_k}^{(c_i)} > 0$. This condition ensures that only correlated nodes to $\mathcal{S}_{\text{cen}}^{(c_1)}$ are stored. We do not want to store as neighbors of a cluster intercluster nodes (usually with the lowest node density, which is equal to zero after normalization) or distant nodes that do not share any neighbors with the current cluster (which results in a zero Jaccard coefficient). Both cases are witnessed through the node correlation matrix (see equation 11).

7. Until 10 nodes are stored in $\mathcal{S}_{\text{cen}}^{(c_1)}$, iterate over the three last steps (3, 4 and 5). It means that we exclude clusters of less than 10 nodes; of course this parameter can be adjusted for very small datasets.

8. Once 10 central nodes have been stored in $\mathcal{S}_{\text{cen}}^{(c_1)}$, we iterate again over steps 3 and 4, but we do not automatically add the new node to the current cluster set $\mathcal{S}_{\text{cen}}^{(c_1)}$. At this stage, we perform a test to either keep filling $\mathcal{S}_{\text{cen}}^{(c_1)}$ or create a new cluster set $\mathcal{S}_{\text{cen}}^{(c_2)}$.

   Let us say that we have already stored $k-1$ nodes in $\mathcal{S}_{\text{cen}}^{(c_1)}$ and we have just selected a new node k among to the 2-hop neighbors of $n_{k-1}^{(c_1)}$. We perform the following test:

$$\overline{W_k}^{(c_i)} - \text{mean}\left(\mathcal{W}_{\mathcal{S}_{\text{cen}}^{(c_i)}}\right) \leq 2.5 \text{ std}\left(\mathcal{W}_{\mathcal{S}_{\text{cen}}^{(c_i)}}\right). \tag{16}$$

   If the test is true, we add node k to $\mathcal{S}_{\text{cen}}^{(c_1)}$, otherwise this node becomes the first central nodes of a new cluster $\mathcal{S}_{\text{cen}}^{(c_2)}$. As explained below equation 14, if node k corresponds to a sample of a new cell type we expect a jump in similarity weights. To witness such a jump, we ask that the average weight coefficients to cluster c_1 for node k jumped by 2.5 standard deviation with respect to the mean value. Note that we only witness positive jumps.

9. If a new cluster is created, set weighted node degree of all nodes stored in $\mathcal{N}_{(c_1)}$ to zero. This step ensures that highly connected neighbors nodes to cluster c_1 – and, thus, which likely belong to this cluster, but have not been classified as central nodes – cannot be detected as part of a new cluster in further iterations; this is the reason why we have excluded from this set all nodes with $\overline{\mathcal{C}_k}^{(c_1)} = 0$, and which potentially belong to another cluster, even if they are neighbors of the current cluster.

10. Before populating $\mathcal{S}_{\text{cen}}^{(c_2)}$ with new nodes, the node degree vector $\mathbf{d}$ is renormalized between 0 and 1. The first central node added to $\mathcal{S}_{\text{cen}}^{(c_2)}$ is then chosen as the node with the highest normalized degree.

11. Then, iterate over the previous steps until a new cluster is again detected.

12. The algorithm stops if one of the following requirements is fulfilled:

    - The number of remaining (not classified) nodes in the graph is lower than 1 % of the total number of nodes already stored as neighbor or central nodes, i.e.,

$$1 - \frac{\sum_i |\mathcal{N}_{(c_i)} \cup \mathcal{S}_{\text{cen}}^{(c_i)}|}{M} < 0.01.$$

    - The maximal value of the initial weighted node degree (before any normalization has occurred) among the remaining nodes is less than 0.01.
    - The maximal value of the weighted node degree (before the last normalization) is lower than 10^{-5}. This condition, together with the previous one, double check that only sparse and marginal nodes which have not been added to the last cluster remain.

When the algorithm stops, we are thus left with several cluster sets $\mathcal{S}_{\text{cen}}^{(c_1)}, \ldots, \mathcal{S}_{\text{cen}}^{(c_{\mathcal{N}_C})}$, where $\mathcal{N}_C$ is the total number of detected clusters. All central nodes contained in these sets are labeled with their corresponding cluster index c_i. All other nodes are still unclassified at this stage (including those in the neighborhood sets

$\mathcal{N}_{(c_i)}$). To label the remaining nodes, we use a k-NN classifier, namely the EWC-FAISS classifier introduced in Ref. Kalweit et al. (2024), which we also consider later for cell classification. This method is well suited to our problem because it performs adaptive nearest-neighbor search on a kNN graph to assign labels – exactly matching our objective of labeling nodes in a graph based on their proximity to central nodes. In this step, the FAISS index is trained on the labeled central nodes and subsequently used to propagate labels to the remaining nodes. Since the central node labels themselves were obtained through a cluster detection algorithm, the entire labeling process remains fully unsupervised.

As a final refinement, we filter out sparse clusters, i.e., clusters that are not well organized around a few representative nodes. For each detected cluster, we identify the most central node (using *closeness centrality*) and require that at least 60% of the cluster's nodes are reachable within two hops from this central node. Nodes belonging to clusters that fail this criterion are reassigned: we retrain the k-NN classifier on the remaining clusters and use it to reclassify the nodes from the eliminated clusters.

To understand and visualize what the algorithms does, let us consider the simplified example shown in Fig. 8 where several iterations of the algorithm are displayed. The graph contains three clusters where nodes accumulate and we only show in detail the first cluster. The weighted node degrees d_i are indicated, on top of each node, labeled from 0 to 8. For this particular example, the normalized node degree vector reads $\mathbf{d} = (0.8, 0.8, 0.8, 0.4, 0.2, 1, 0.8, 0.4, 0.6\ldots)^T$.

For simplicity, we assume that the graph has simple edge weights, i.e., $e_{ij} = 1$ if nodes i and j are connected, otherwise $e_{ij} = 0$. We also assume the following similarity matrix coefficients: for $i \neq j$, $W_{ij} = 1$ if $e_{ij} > 0$, $W_{ij} = 2$ if $e_{ij} = 0$, and $W_{ii} = 0$. Note that here similarity matrix coefficients and edge weights are not related through equation 7. While this example does not precisely match the cell graph structure we consider in the paper, its purpose is simply to illustrate how the algorithm works.

The node with the maximal node degree is selected to be the first central node. In our case, it corresponds to node 5 with $d_5 = 1$ (colored in red in the first top left graph of Fig. 8). Thus, $\mathcal{S}_{\text{cen}}^{(c_1)} = \{5\}$.

To select the next central node, we first aggregate all 2-hop neighbors of the current central node; it corresponds here to all nodes from 0 to 8 (including the central node 5 itself). Then, we proceed with step 3 (see above) and compute $\overline{W}^{(h=2,\,c_1)}$. Here, this vector is $\overline{W}^{(h=2,\,c_1)} = (2, 1, 1, 1, 1, 0, 1, 2, 2)$. We then filter all nodes for which condition 15 is true. In our example the condition is $\overline{W}_k^{(h=2,\,c_1)} \geq 1.22$, and we thus filter nodes 0, 7 and 8. Then, the next selected central node is the one with the highest node degree among the remaining 2-hop nodes (see step 4 of the algorithm detailed above) excluding the current central node (we just set $d_5 = 0$). Here, there are three potential candidates with $d = 0.8$: 1, 2, 6. In real situations, since weighted node degrees directly depend on the edge weights, which in turn depend on the distance between cell samples, it is almost impossible that two nodes inherit from the exact same weighted node degree; if that happens, we just select randomly one of the nodes. Here, we assume that node 2 has been selected – see the second graph in Fig. 8, where node 2 is now colored in red.

We now repeat the above steps for node 2. The filtering condition 15 now reads $\overline{W}_k^{(h=2,\,c_1)} \geq 1.33$. We thus consider as new central node candidates 0, 1 and 6 (excluding 2 and 5, the current seed nodes). Again, all these nodes have the same weighted node degree. We thus select randomly node 1.

One iteration further, node 0 is then selected and we have at this stage $\mathcal{S}_{\text{cen}}^{(c_1)} = \{0, 1, 2, 5\}$. A step further, one can check that the only potential candidate is node 6. At variance to the true algorithm and to illustrate how new clusters are detected, we assume that here we proceed to step 8 and perform test 16 as soon as 4 nodes have been stored in a cluster (instead of 10, see step 7). To determine whether node 6 belongs to the current cluster, we thus need to compute quantity 14, which, as stated above, is constructed iteratively. In our case

$$\mathcal{W}_{\mathcal{S}_{\text{cen}}^{(c_1)}} = \left(\underset{l \in \mathcal{S}_{\text{cen}}^{(c_i)} = \{5\}}{\text{mean}} [W_{2l}] \,, \underset{l \in \mathcal{S}_{\text{cen}}^{(c_i)} = \{2,5\}}{\text{mean}} [W_{1l}] \,, \underset{l \in \mathcal{S}_{\text{cen}}^{(c_i)} = \{1,2,5\}}{\text{mean}} [W_{0l}] \right)^T = (1, 1, 4/3)^T.$$

Node 6 has not been added to the cluster yet; therefore, there are only three coefficients at the moment in $\mathcal{W}_{\mathcal{S}_{\text{cen}}^{(c_1)}}$. One checks easily that condition 16 is still true: $\overline{W_6}^{(c_1)} = 5/4$, $\text{mean}\left(\mathcal{W}_{\mathcal{S}_{\text{cen}}^{(c_1)}}\right) = 10/9$, and

$\text{std}\left(\mathcal{W}_{\mathcal{S}_{\text{cen}}^{(c_1)}}\right) = 0.16$. Node 6 is thus added to $\mathcal{S}_{\text{cen}}^{(c_1)}$ and colored in red; the situation is depicted in the bottom right graph of Fig. 8.

Given the 2-hop neighborhood of node 6, only node 7 remains as a potential central node candidate. We now have $\overline{W_7}^{(c_1)} = 9/5$ and $\mathcal{W}_{\mathcal{S}_{\text{cen}}^{(c_1)}} = (1, 1, 4/3, 5/4)^T$, i.e., $\text{mean}\left(\mathcal{W}_{\mathcal{S}_{\text{cen}}^{(c_1)}}\right) = 1.15$ and $\text{std}\left(\mathcal{W}_{\mathcal{S}_{\text{cen}}^{(c_1)}}\right) = 0.15$. One checks that condition 16 is not respected for node 7; this is not surprising since this node is only connected to cluster $\mathcal{S}_{\text{cen}}^{(c_1)}$ through node 6. Note the jump from $\overline{W_6}^{(c_1)} = 1.25$ to $\overline{W_7}^{(c_1)} = 1.8$, characteristic of the emergence of a new cluster. Therefore, node 7 is not added to $\mathcal{S}_{\text{cen}}^{(c_1)}$ and a new cluster $\mathcal{S}_{\text{cen}}^{(c_2)}$ is created. Before selecting the first central node of $\mathcal{S}_{\text{cen}}^{(c_2)}$ (not necessarily node 7), all weighted node degrees of $\mathcal{S}_{\text{cen}}^{(c_1)}$ and $\mathcal{N}_{(c_1)}$ are put to zero, and the entire node degree vector $\mathbf{d}$ is normalized between 0 and 1. The maximal node degree after normalization is carried by node 8, which becomes the new central node of $\mathcal{S}_{\text{cen}}^{(c_2)}$. The situation is depicted in the last bottom left graph of Fig. 8, where node 8 has been colored in blue as it belongs to a new cluster.

Nodes 3 and 4 will most probably remain unassigned; this is where the k-NN search will prove useful to label these remaining nodes. Given the graph structure, nodes 3 and 4 will be most likely attached to cluster c_1.

A.2 Shared cell type identification through similarity score vectors $\mathbf{cs}^{(c_i)}$

After grouping cells into clusters in the test dataset $\mathcal{D}_{\text{test}}$, the next step is to identify the cell types shared between the test and training datasets. This allows us to filter out training classes that are not present in the test dataset to improve both optimal mapping and cell classification.

In this appendix, we detail the computation of similarity score vectors $\mathbf{cs}^{(c_i)}$ that are used to detect matching cell types across datasets.

Throughout this section, test clusters are denoted by c_i, while cell type subsets in the training dataset are denoted as $\text{ct}^{(i)}$ to avoid confusion between label cell types and unlabeled clusters.

First we start by computing correlations between test and training embeddings for each encoder using *cosine similarity*:

$$\mathcal{C}_{kl}^{\text{enc}} \equiv \frac{\mathcal{E}_k^{\text{test, enc}} \cdot \mathcal{E}_l^{\text{train, enc}}}{||\mathcal{E}_k^{\text{test, enc}}||\,||\mathcal{E}_l^{\text{train, enc}}||}, \tag{17}$$

Then, we group and average correlations over test clusters and training cell types:

$$\overline{\mathcal{C}_{ij}^{\text{enc}}} \equiv \underset{k \in c_i,\, l \in \text{ct}^{(j)}}{\text{mean}} \left(\mathcal{C}_{kl}^{\text{enc}}\right). \tag{18}$$

These results are stored in $\mathcal{N}_C^{\text{test}} \times \mathcal{N}_C^{\text{train}}$ similarity matrices $\overline{\mathcal{C}^{\text{enc}}}$, where $\mathcal{N}_C^{\text{test}}$ is the number of detected clusters in $\mathcal{D}_{\text{test}}$, and $\mathcal{N}_C^{\text{train}}$ denotes the number of cell type subsets in $\mathcal{D}_{\text{train}}$. Shared cell types between the test and training datasets are expected to exhibit strong correlation signals, and these are precisely the signals we aim to identify for each test cluster (see Fig. 9(a) where an example of such a matrix for DinoBloom encoder and test and train datasets WBC and PBC is shown).

To detect these signals, we first need to discriminate signal from noise. Indeed, it might happen that for certain foundation model no signal stands out for any particular training cell types; in that case, it is better to discard this encoder since it will only add noise and randomly identify matching clusters. The absence of out-of-noise signal does not necessarily indicate that the test cluster cell type is not present in the training dataset; most of the time, such an encoder is not able to generate embeddings which group cells into clusters. Therefore, when computing correlations from these embeddings, no particular pattern can be identified and it results in random correlation signals. To remove noisy signals, we simply compute the mean signal-to-noise ratio over all test cluster:

$$\text{SNR} = \underset{i}{\text{mean}} \left[\frac{\sigma_{\overline{\mathcal{C}_i^{\text{enc}}}}}{\overline{\mathcal{C}_i^{\text{enc}}}}\right], \tag{19}$$

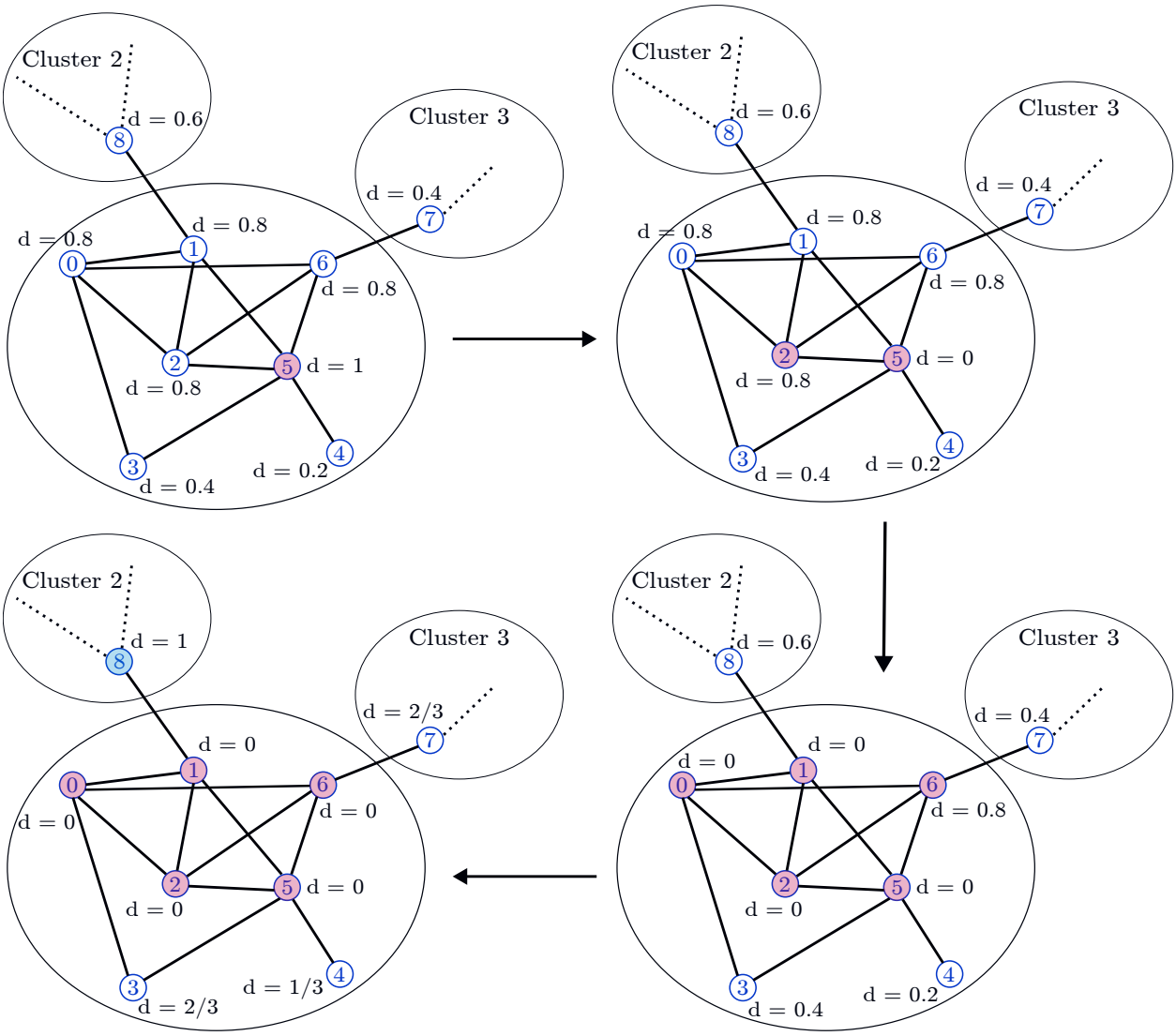

Figure 8: Illustrative run of our cluster detection algorithm on a small toy graph. Starting from the node with the highest weighted node degree (the central node), neighboring nodes are iteratively aggregated into the cluster $\mathcal{S}_{\text{cen}}^{(c_1)}$ until the new-cluster condition is no longer satisfied; a new cluster $\mathcal{S}_{\text{cen}}^{(c_2)}$ (blue node) is then initiated. Remaining unassigned nodes (e.g., nodes 3 and 4) are labeled afterwards using the k-NN search.

where $\overline{\mathcal{C}_i^{\text{enc}}} = \underset{k\in c_i,\, l\in\mathcal{D}_{\text{train}}}{\text{mean}}(\mathcal{C}_{kl}^{\text{enc}})$, and $\sigma_{\overline{\mathcal{C}_i^{\text{enc}}}} = \underset{k\in c_i,\, l\in\mathcal{D}_{\text{train}}}{\text{std}}(\mathcal{C}_{kl}^{\text{enc}})$. If this ratio is below 5%, the associated encoder is discarded. When all encoders but one are removed, we lower the SNR threshold by 1% until at least two encoders are kept.

Once only encoders revealing high enough out-of-noise correlation signals are selected (typically between 2 and 4 encoders), we combine their results to strengthen large correlation signals. Indeed, two clusters sharing high correlations for different encoders most probably belong to the same cell type. Combining encoders also mitigates errors resulting from the use of only one encoder. We proceed as follows:

1. We normalize the coefficients to exhibit signals out of the noise:

$$\mathcal{C}_{ij}^{\mathcal{N},\text{enc}} = \frac{\overline{\mathcal{C}_{ij}^{\text{enc}}} - \overline{\mathcal{C}_i^{\text{enc}}}}{\sigma_{\overline{\mathcal{C}_i^{\text{enc}}}}}, \tag{20}$$

where quantities used in this normalization step are defined below equation 19; here, mean and standard deviation are computed along each matrix row (one test cluster versus all training cell types).

2. To enhance correlation signals beyond one standard deviation to the mean and damp those below half a standard deviation, we apply a nonlinear function $f(\mathcal{C}_{ij}^{\mathcal{N},\text{enc}}) = \tanh\left[3(\mathcal{C}_{ij}^{\mathcal{N},\text{enc}} - 0.5)\right]$ to each matrix coefficient. It also normalized all values between -1 and 1 such that it provides for each test cluster a correlation score to each training cell types. We denote the resulting matrix $\overline{\mathcal{C}^{\text{row,enc}}}$, where the superscript *row* means that normalization 20 is along each matrix row.

3. The final step consists of averaging over all encoders: $\overline{\mathcal{C}^{\text{row}}} = \underset{\text{enc}}{\text{mean}}\left[\overline{\mathcal{C}^{\text{row,enc}}}\right]$. An example of such a normalized matrix for white-blood cell datasets WBC (unlabeled test) vs PBC (labeled train) is shown in Fig. 10(a).

4. We repeat the previous steps, but we now normalize the initial matrix along each column. In this case, we are interested in correlation signals for each cell type subsets versus all test clusters. This step is important because for two clusters of the same cell type a high correlation signal should only peak for these two clusters and no others. By looking row-wise and column-wise we ensure that this is the case. We obtain a new normalized matrix $\overline{\mathcal{C}^{\text{col}}}$.

5. On top of that, the highest correlation signal found for each encoder will correspond to the predicted matching cluster either in the training or the test dataset by looking row-wise or column-wise, respectively. Instead of averaging other all encoders, we can also count the number of times that a matching pair of clusters has been predicted.

   For instance, referring to Fig. 9(a), test cluster 5 in the WBC dataset exhibits a high signal (0.21) with cell type ground-truth label 4 (corresponding to monocytes) in the PBC dataset. If we proceed in the same manner for all selected encoders, we find that WBC test cluster 5 is identified two times to PBC ground-truth label 4 and only once to label 10. Therefore, counting the number of times each PBC cell type is identified as the best matching class for WBC cluster 5 will result in a 11-dimensional vector (here $\mathcal{N}_C^{\text{PBC}} = 11$) with a 2 for label 4, 1 for label 10 and 0 elsewhere. Proceeding in the same manner for each test cluster leads to a new matrix that we also normalize between -1 and 1 and denote as N^{row}. The not normalized and normalized matrices N^{row} are shown in Figs. 9(b) and 10(b), respectively.

6. Of course, the same holds column-wise: how many times a WBC cluster is identified as the best matching group for each PBC cell type? This results in 10-dimensional vectors (here the number of detected clusters in WBC is $\mathcal{N}_C^{\text{WBC}} = 10$) for each PBC cell type that we concatenate all together in a matrix. As a final step, we transpose the matrix such that it has the same dimensions as N^{row} and we denote it as N^{col}.

7. We finally sum row-wise and column-wise matrices together to gather all results in only two matrices $\mathcal{M}^{\text{row}}$ and $\mathcal{M}^{\text{col}}$ defined as

$$\mathcal{M}^{\text{row}} \equiv \frac{\overline{\mathcal{C}^{\text{row}}} + N^{\text{row}}}{2}, \quad \mathcal{M}^{\text{col}} \equiv \frac{\overline{\mathcal{C}^{\text{col}}} + N^{\text{col}}}{2} \tag{21}$$

To summarize the above procedure, final matrices $\mathcal{M}^{\text{row}}$ and $\mathcal{M}^{\text{col}}$ contain normalized matching scores for all pairs of test clusters and training (ground-truth) cell types. Matrix coefficients with values close to 1 indicate a high probability of matching.

To exploit the results stored in these two matrices and visualize them, we can interpret each row of these matrices as a vector containing the matching scores between one test cluster and all cell types in the training dataset. Therefore, for each test cluster c_i, we can create multidimensional vectors as follows:

$$\mathbf{cs}^{(c_i)} = (\vec{cs}_1^{(c_i)}, \ldots, \vec{cs}_{\mathcal{N}_C^{\text{train}}}^{(c_i)})^T, \quad \text{with} \quad \vec{cs}_j^{(c_i)} = (\mathcal{M}_{ij}^{\text{row}}, \mathcal{M}_{ij}^{\text{col}}). \tag{22}$$

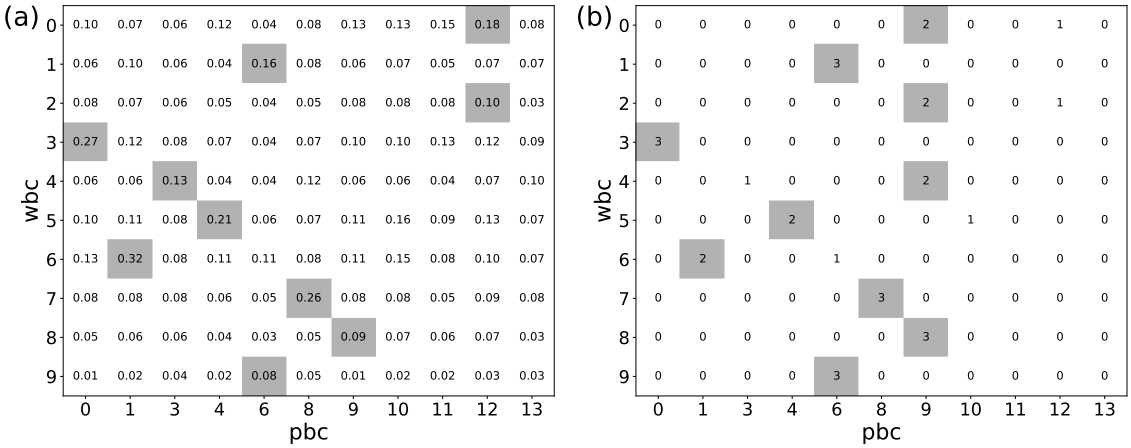

Figure 9: (a) Matrices $\overline{\mathcal{C}^{\text{row, BLOOM}}}$ that gathers correlations between pair of detected test WBC clusters and training PBC cell types computed from cosine similarity between DinoBloom embeddings. The gray boxes highlight the maximal (and not normalized) correlation signal. The x-labels correspond to the ground-truth PBC cell types, while the y-labels just correspond to labels given by the cluster detection algorithm as new clusters are detected. (b) Matrices N^{row} that gathers the number of times that a matching pair of test WBC and training PBC cell types has been predicted by an encoder. Here, three encoders were used; whenever a matrix coefficient is equal to 3, it means that all encoders have predicted a high correlation between the same pair of clusters.

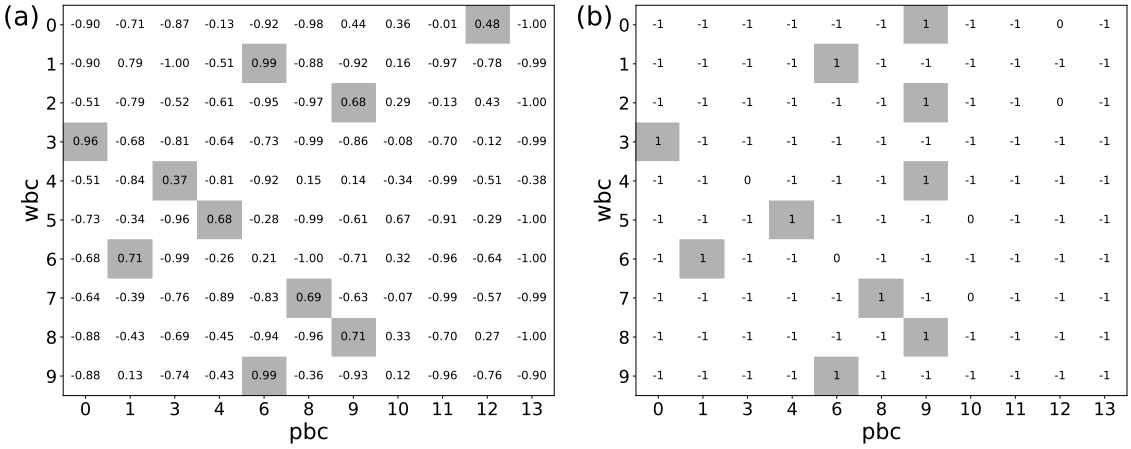

Figure 10: Same as Fig. 9, but with normalized coefficients and averaged over all encoders. The normalization is performed row-wise: for each row we constrain values between -1 and 1.

The vectors $\mathbf{cs}^{(c_i)}$ contain normalized similarity scores and are composed of $\mathcal{N}C^{\text{train}}$ two-dimensional vectors $\vec{cs}j^{(c_i)}$. Each of these two-dimensional vectors can be interpreted as a coordinate pair, making it possible to visualize them directly in a plot.

Such plots can be observed in Figs. 11 for the LISC dataset (with WBC as the reference labeled dataset). Five clusters have been detected for LISC using the algorithm detailed in Appendix A.1, thus resulting in five plots (one plot for each $\mathbf{cs}^{(c_i)}$, i ranging from 1 to 5). On each plot, dots correspond to the 2-dimensional vectors $\vec{cs}_j^{(c_i)}$. The rightmost point corresponds to the most probable matching cell type. Indeed, we recall that each point carries the row-wise and column-wise matching scores as coordinates (see equation 22). Therefore, the rightmost part of the plot corresponds to high matching scores (the closer to 1, the better).

In particular, if one of the vectors $\vec{cs}_j^{(c_i)}$ has coordinates $(1,1)$, it means that the highest correlation signal has been *always* found between cell type $\mathrm{ct}^{(j)}$ and the corresponding test cluster c_i for *all* selected encoders.

Next to each dot, one finds two numbers: the corresponding WBC cell type (labels between 0 and 7, see the caption of Fig. 11 for label-to-cell type correspondence), as well as its normalized distance to the rightmost point of coordinates $(1,1)$ computed as follows:

$$p_j^{(c_i)} = 1 - \frac{\left[1 + 2\min\left(1 - \mathcal{M}_{ij}^{\mathrm{row}},\, 1 - \mathcal{M}_{ij}^{\mathrm{col}}\right)\right]\sqrt{\left(1 - \mathcal{M}_{ij}^{\mathrm{row}}\right)^2 + \left(1 - \mathcal{M}_{ij}^{\mathrm{col}}\right)^2}}{3\sqrt{2}}. \tag{23}$$

Rather than a distance, $p_j^{(c_i)}$ can be better interpreted as a matching probability of cluster test c_i with the corresponding cell type $\mathrm{ct}^{(j)}$. From the plots of Figs. 11, one concludes that LISC detected cluster 0 can be identified to cell type 6, i.e., neutrophils (with the highest possible matching probability $p_6^{(0)} = 1$). Cluster 1, 2, 3 and 4 can be identified to WBC labels 0 (basophils), 3 (lymphocytes), 4 (monocytes) and 1 (eosinophils), respectively. Its a perfect match: one can check from LISC ground-truth labels that clusters 0 to 4 indeed correspond to the found cell types. It will be then possible to eliminate all cell samples of another type in WBC (i.e., lymphoblasts, monoblasts and normoblasts), when using it as the training dataset for LISC cell classification.

B Global Accuracy $\mathrm{Acc}_{\mathrm{global}}$ used in Sec. 5.1

The global accuracy, $\mathrm{Acc}_{\mathrm{global}}$, used to assess the precision of shared cell type identification, is defined as

$$\mathrm{Acc}_{\mathrm{global}} = \frac{\mathcal{S}_{\mathrm{pred}} \cap \mathcal{S}_{\mathrm{ground-truth}}}{|S_{\mathrm{ground-truth}}|} \times \frac{\mathcal{N}_{\mathrm{pred-filtered}}}{\mathcal{N}_{\mathrm{gt-filtered}}}, \tag{24}$$

In the equations above, $\mathcal{S}_{\mathrm{pred}}$ denotes the set of predicted shared cell types between a given test dataset and the reference dataset, while $\mathcal{S}_{\mathrm{ground-truth}}$ is the corresponding set of true shared cell types (based on the test dataset's ground-truth labels).

The global accuracy, $\mathrm{Acc}_{\mathrm{global}}$, combines two factors:

- the fraction of correctly predicted shared cell types

  $$\frac{\mathcal{S}_{\mathrm{pred}} \cap \mathcal{S}_{\mathrm{ground-truth}}}{|S_{\mathrm{ground-truth}}|},$$

  which measures how well the predicted overlap matches the true overlap ($|S_{\mathrm{ground-truth}}|$ is the number of true shared cell types); and

- the quality of class filtering in the reference dataset

  $$\frac{\mathcal{N}_{\mathrm{pred-filtered}}}{\mathcal{N}_{\mathrm{gt-filtered}}},$$

  where $\mathcal{N}_{\mathrm{pred-filtered}}$ is the number of classes correctly predicted as filtered, and $\mathcal{N}_{\mathrm{gt-filtered}}$ is the total number of classes that should be filtered, i.e., all cell types that are not present in the test dataset.

  This second term penalizes cases where all shared cell types are detected (maximizing the first ratio of $\mathrm{Acc}_{\mathrm{global}}$) but spurious classes in the reference dataset are not removed.

C All classification results

In this section, we show the classification results for all datasets (after class filtering). In total, we tested our approach and pipeline on 15 different pairs of training and transfer live and white blood cell datasets. Here are the correspondence between result tables and pairs of dataset:

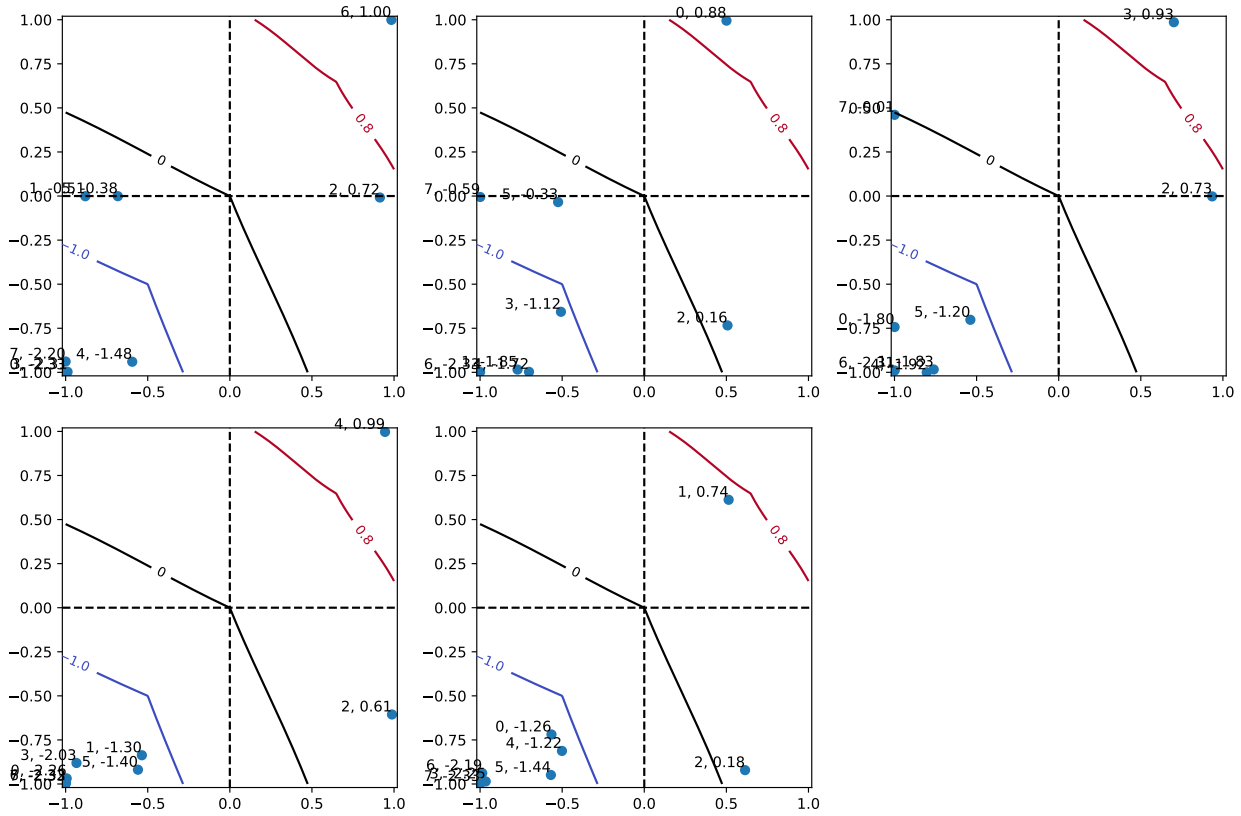

Figure 11: 2-dimensional visualization cluster matching between LISC (unlabeled test dataset) and WBC (labeled training dataset). Each graph shows the 2-dimensional vectors $\vec{cs}_j^{(c_i)}$ contained in $\mathbf{cs}^{(c_i)}$ (see equation 22) for each detected LISC cluster c_i. In the present case, five clusters have been detected by the cluster detection algorithm detailed in Appendix A.1, resulting in five different graphs. Labels on top of each dot contain two numbers: the first integer indicates the corresponding cell group (0: basophils, 1: eosinophils, 2: lymphoblasts, 3: lymphocytes, 4: monocytes, 5: monoblasts, 6: neutrophils, 7: normoblasts); the second number is the overall matching score $p_j^{(c_i)}$ computed from expression equation 23. The closer this number is from 1 (the closer the blue dot is from the upper right corner of the plot), the highest are correlations between the corresponding test clusters c_i and cell type $\text{ct}^{(j)}$. The red, black and blue solid lines are different contours of matching score $p_j^{(c_i)}$.

1. Transfer dataset: WBC, training dataset: PBC – Table 9
2. Transfer dataset: RAABIN, training dataset: PBC – Table 10
3. Transfer dataset: LISC, training dataset: PBC – Table 11
4. Transfer dataset: PBC, training dataset: WBC – Table 12
5. Transfer dataset: RAABIN, training dataset: WBC – Table 13
6. Transfer dataset: LISC, training dataset: WBC – Table 14
7. Transfer dataset: RAABIN, training dataset: WBC-PBC (Combination 1) – Table 15
8. Transfer dataset: RAABIN, training dataset: WBC-PBC (Combination 2) – Table 16
9. Transfer dataset: LISC, training dataset: WBC-PBC (Combination 1) – Table 17

10. Transfer dataset: LISC, training dataset: WBC-PBC (Combination 2) – Table 18
11. Transfer dataset: LISC, training dataset: WBC + PBC + RAABIN – Table 19
12. Transfer dataset: Lionheart (2 classes), training dataset: Nanolive – Table 20
13. Transfer dataset: Lionheart (3 classes), training dataset: Nanolive – Table 21
14. Transfer dataset: Nanolive (2 classes), training dataset: Lionheart – Table 22
15. Transfer dataset: Nanolive (2 classes), training dataset: Lionheart – Table 21

Table 9: Comparison of encoder performance with class filtering and with (✓) or without (✗) optimal transport (OT) – Transfer dataset: WBC, training dataset: PBC. The last row shows the ensemble voting performance (including OT), and the average Δ_{OT} row summarizes the mean gain across all encoders.

Transfer: **WBC** *Train: PBC*	Acc. (OT: ✗)	Acc. (OT: ✓)	F1 (OT: ✗)	F1 (OT: ✓)	Δ Acc.	Δ F1
SAM	39.01	**50.99**	42.12	**54.90**	11.98	12.78
DINO	57.48	**72.37**	68.75	**77.22**	14.9	8.47
ConvNeXt	59.10	**72.89**	59.82	**72.73**	13.8	12.91
SWIN	74.91	**81.31**	77.95	**80.55**	6.39	2.59
CLIP	50.98	**74.90**	55.99	**78.39**	23.91	22.4
ViTMAE	30.19	**32.98**	**32.84**	31.25	2.79	-1.59
DinoBloom	**95.96**	84.84	**95.65**	82.81	-11.11	-12.83
CN + SWIN	69.34	**81.72**	71.62	**81.80**	12.38	10.17
SAM + CN + SWIN + CLIP	73.28	**84.48**	75.33	**84.51**	11.21	9.18
DINO + CN + ViTMAE + SWIN	72.09	**86.03**	75.24	**86.22**	13.94	10.98
CN + SWIN + ViTMAE	70.11	**82.35**	72.37	**82.36**	12.24	9.99
DINO + CN + SWIN	71.70	**86.17**	74.58	**86.44**	14.47	11.86
DINO + CN + ViTMAE	64.00	**80.00**	66.33	**81.38**	16.0	15.05
DINO + ViTMAE + SWIN	78.44	**85.86**	82.25	**86.08**	7.42	3.83
Ensemble Voting method	86.11	**92.54**	87.82	**92.14**	6.43	4.32
Average Δ_{OT}	-	-	-	-	10.45	8.01

Table 10: Comparison of encoder performance with class filtering and with (✓) or without (✗) optimal transport (OT) – Transfer dataset: RAABIN, training dataset: PBC. The last row shows the ensemble voting performance (including OT), and the average Δ_{OT} row summarizes the mean gain across all encoders.

Transfer: **RAABIN** *Train: PBC*	Acc. (OT: ✗)	Acc. (OT: ✓)	F1 (OT: ✗)	F1 (OT: ✓)	Δ Acc.	Δ F1
SAM	19.76	**25.54**	29.55	**51.30**	5.77	21.75
DINO	35.73	**55.49**	**71.49**	61.66	19.77	-9.82
ConvNeXt	30.59	**37.89**	**69.16**	59.11	7.29	-10.05
SWIN	29.42	**51.78**	**66.70**	60.17	22.35	-6.53
CLIP	31.54	**36.72**	**57.51**	48.97	5.19	-8.55
ViTMAE	20.37	**25.98**	15.68	**35.90**	5.62	20.22
DinoBloom	53.34	**67.53**	**61.72**	48.57	14.18	-13.15
CN + SWIN	35.38	**49.63**	**72.64**	63.55	14.25	-9.09
SAM + CN + SWIN + CLIP	37.77	**51.57**	**73.57**	63.66	13.81	-9.91
DINO + CN + ViTMAE + SWIN	36.74	**57.57**	**73.48**	67.54	20.83	-5.94
CN + SWIN + ViTMAE	34.53	**52.17**	**72.25**	64.27	17.64	-7.98
DINO + CN + SWIN	36.93	**56.59**	**73.91**	67.25	19.67	-6.66
DINO + CN + ViTMAE	31.79	**46.38**	**70.46**	65.58	14.58	-4.88
DINO + ViTMAE + SWIN	32.30	**67.83**	**68.79**	65.05	35.53	-3.75
Ensemble Voting method	45.64	**64.35**	70.00	**77.30**	18.71	7.3
Average Δ_{OT}	-	-	-	-	15.68	-3.14

Table 11: Comparison of encoder performance with class filtering and with (✓) or without (✗) optimal transport (OT) – Transfer dataset: LISC, training dataset: PBC. The last row shows the ensemble voting performance (including OT), and the average Δ_{OT} row summarizes the mean gain across all encoders.

Transfer: **LISC** *Train: PBC*	Acc. (OT: ✗)	Acc. (OT: ✓)	F1 (OT: ✗)	F1 (OT: ✓)	Δ Acc.	Δ F1
SAM	37.03	**37.14**	34.95	**36.65**	0.11	1.7
DINO	46.70	**58.45**	43.07	**59.13**	11.75	16.06
ConvNeXt	41.56	**46.43**	35.02	**44.75**	4.87	9.72
SWIN	25.65	**52.82**	17.66	**48.22**	27.17	30.56
CLIP	26.89	**28.00**	16.13	**27.19**	1.11	11.06
ViTMAE	19.51	**25.82**	10.56	**23.99**	6.31	13.44
DinoBloom	**66.38**	54.19	**68.65**	48.95	-12.19	-19.71
CN + SWIN	32.11	**45.77**	24.57	**44.51**	13.66	19.93
SAM + CN + SWIN + CLIP	30.94	**48.30**	23.36	**47.11**	17.37	23.75
DINO + CN + ViTMAE + SWIN	31.55	**62.72**	26.32	**61.76**	31.17	35.44
CN + SWIN + ViTMAE	29.75	**45.38**	21.83	**44.40**	15.63	22.57
DINO + CN + SWIN	33.12	**62.60**	27.78	**62.02**	29.48	34.23
DINO + CN + ViTMAE	42.60	**62.32**	37.48	**62.47**	19.72	24.98
DINO + ViTMAE + SWIN	31.91	**63.42**	24.44	**63.52**	31.5	39.08
Ensemble Voting method	53.89	**71.23**	54.64	**70.34**	17.34	15.69
Average Δ_{OT}	-	-	-	-	14.33	18.57

Table 12: Comparison of encoder performance with class filtering and with (✓) or without (✗) optimal transport (OT) – Transfer dataset: PBC, training dataset: WBC. The last row shows the ensemble voting performance (including OT), and the average Δ_{OT} row summarizes the mean gain across all encoders.

Transfer: **PBC** *Train: WBC*	Acc. (OT: ✗)	Acc. (OT: ✓)	F1 (OT: ✗)	F1 (OT: ✓)	Δ Acc.	Δ F1
SAM	**33.73**	32.87	35.50	**43.64**	-0.86	8.14
DINO	66.08	**71.56**	67.83	**76.72**	5.48	8.89
ConvNeXt	**69.00**	60.83	61.35	**65.38**	-8.17	4.03
SWIN	**65.11**	58.63	**70.74**	66.32	-6.48	-4.42
CLIP	**63.00**	59.98	**67.12**	65.70	-3.02	-1.42
ViTMAE	21.54	**35.18**	4.97	**23.01**	13.64	18.04
DinoBloom	**93.84**	85.99	**92.42**	85.88	-7.85	-6.54
CN + SWIN	**72.67**	64.05	**70.24**	69.32	-8.62	-0.92
SAM + CN + SWIN + CLIP	**76.36**	66.58	**75.71**	72.16	-9.78	-3.55
DINO + CN + ViTMAE + SWIN	**75.74**	65.98	**73.88**	70.98	-9.76	-2.9
CN + SWIN + ViTMAE	**73.53**	63.68	**70.38**	68.81	-9.85	-1.57
DINO + CN + SWIN	**75.20**	66.14	**74.27**	71.33	-9.06	-2.94
DINO + CN + ViTMAE	**75.17**	64.82	70.10	**70.33**	-10.35	0.24
DINO + ViTMAE + SWIN	**71.43**	65.88	**75.46**	72.10	-5.54	-3.36
Ensemble Voting method	**88.68**	88.67	89.30	**90.71**	-0.01	1.41
Average Δ_{OT}	-	-	-	-	-4.68	0.88

Table 13: Comparison of encoder performance with class filtering and with (✓) or without (✗) optimal transport (OT) – Transfer dataset: RAABIN, training dataset: WBC. The last row shows the ensemble voting performance (including OT), and the average Δ_{OT} row summarizes the mean gain across all encoders.

Transfer: **RAABIN** *Train: WBC*	Acc. (OT: ✗)	Acc. (OT: ✓)	F1 (OT: ✗)	F1 (OT: ✓)	Δ Acc.	Δ F1
SAM	24.42	**36.77**	28.16	**56.73**	12.35	28.57
DINO	52.51	**62.28**	**65.50**	62.69	9.77	-2.81
ConvNeXt	56.96	**58.57**	**73.88**	65.36	1.61	-8.52
SWIN	44.93	**68.64**	**79.03**	68.79	23.71	-10.24
CLIP	45.31	**53.11**	**78.58**	58.62	7.8	-19.96
ViTMAE	21.18	**29.19**	24.69	**43.49**	8.02	18.8
DinoBloom	**88.65**	85.07	**91.95**	65.58	-3.58	-26.37
CN + SWIN	53.54	**66.42**	**82.43**	68.70	12.88	-13.73
SAM + CN + SWIN + CLIP	57.34	**64.52**	**84.54**	64.11	7.18	-20.44
DINO + CN + ViTMAE + SWIN	54.57	**65.96**	**82.42**	66.32	11.39	-16.1
CN + SWIN + ViTMAE	53.48	**66.36**	**82.39**	68.37	12.88	-14.03
DINO + CN + SWIN	54.59	**66.27**	**82.26**	66.74	11.68	-15.51
DINO + CN + ViTMAE	59.31	**62.65**	**77.54**	65.97	3.34	-11.56
DINO + ViTMAE + SWIN	47.02	**68.97**	**79.92**	67.85	21.96	-12.07
Ensemble Voting method	73.45	**84.74**	**88.38**	88.29	11.29	-0.08
Average Δ_{OT}	-	-	-	-	10.15	-8.27

Table 14: Comparison of encoder performance with class filtering and with (✓) or without (✗) optimal transport (OT) – Transfer dataset: LISC, training dataset: WBC. The last row shows the ensemble voting performance (including OT), and the average Δ_{OT} row summarizes the mean gain across all encoders.

Transfer: **LISC** *Train: WBC*	Acc. (OT: ✗)	Acc. (OT: ✓)	F1 (OT: ✗)	F1 (OT: ✓)	Δ Acc.	Δ F1
SAM	27.93	**30.86**	20.24	**30.10**	2.93	9.86
DINO	50.49	**51.75**	51.92	**53.79**	1.26	1.87
ConvNeXt	55.07	**55.73**	**55.45**	54.94	0.65	-0.52
SWIN	31.48	**66.38**	24.87	**66.63**	34.9	41.75
CLIP	22.35	**51.94**	11.06	**52.24**	29.58	41.18
ViTMAE	11.74	**26.31**	11.70	**25.91**	14.57	14.21
DinoBloom	76.79	**80.54**	77.59	**79.83**	3.75	2.25
CN + SWIN	47.34	**60.40**	42.44	**59.99**	13.06	17.55
SAM + CN + SWIN + CLIP	46.63	**62.53**	42.08	**61.44**	15.9	19.36
DINO + CN + ViTMAE + SWIN	48.43	**62.19**	42.52	**63.11**	13.76	20.59
CN + SWIN + ViTMAE	47.72	**60.40**	42.60	**60.09**	12.68	17.49
DINO + CN + SWIN	47.70	**62.99**	41.92	**64.04**	15.29	22.12
DINO + CN + ViTMAE	**60.66**	57.85	**60.81**	58.19	-2.81	-2.62
DINO + ViTMAE + SWIN	36.24	**62.83**	31.45	**64.32**	26.59	32.87
Ensemble Voting method	59.98	**72.01**	61.18	**73.13**	12.03	11.95
Average Δ_{OT}	-	-	-	-	12.94	16.66

Table 15: Comparison of encoder performance with class filtering and with (✓) or without (✗) optimal transport (OT) – Transfer dataset: RAABIN, training dataset: WBC-PBC (COMBINATION 1). The training dataset WBC-PBC (COMBINATION 1) contains cells from WBC and PBC as follows: cell types 0 (basophils), 1 (eosinophils), 3 (lymphocytes) from WBC and 4 (monocytes), 6 (neutrophils) from PBC. The last row shows the ensemble voting performance (including OT), and the average Δ_{OT} row summarizes the mean gain across all encoders.

Transfer: **RAABIN** *Train: WBC-PBC (COMB. 1)*	Acc. (OT: ✗)	Acc. (OT: ✓)	F1 (OT: ✗)	F1 (OT: ✓)	Δ Acc.	Δ F1
SAM	7.94	**25.37**	23.93	**50.36**	17.42	26.43
DINO	27.73	**50.86**	63.84	**63.99**	23.13	0.15
ConvNeXt	33.25	**36.99**	**68.00**	56.27	3.74	-11.73
SWIN	26.62	**55.55**	**66.97**	57.22	28.93	-9.75
CLIP	29.82	**35.21**	**58.86**	48.52	5.39	-10.34
ViTMAE	**31.62**	19.60	10.01	**42.48**	-12.02	32.48
DinoBloom	58.59	**72.13**	**70.10**	56.82	13.54	-13.28
CN + SWIN	30.09	**51.89**	**69.29**	58.09	21.8	-11.2
SAM + CN + SWIN + CLIP	31.67	**51.23**	**70.22**	57.05	19.56	-13.17
DINO + CN + ViTMAE + SWIN	31.47	**56.27**	**70.74**	62.44	24.8	-8.3
CN + SWIN + ViTMAE	31.67	**51.92**	**70.69**	59.21	20.25	-11.48
DINO + CN + SWIN	29.96	**56.13**	**69.58**	61.67	26.17	-7.91
DINO + CN + ViTMAE	34.56	**41.72**	**70.47**	62.33	7.16	-8.14
DINO + ViTMAE + SWIN	28.83	**61.64**	**68.79**	63.78	32.81	-5.01
Ensemble Voting method	44.98	**69.34**	72.46	**79.47**	24.36	7.01
Average Δ_{OT}	-	-	-	-	17.14	-2.95

Table 16: Comparison of encoder performance with class filtering and with (✓) or without (✗) optimal transport (OT) – Transfer dataset: RAABIN, training dataset: WBC-PBC (COMBINATION 2). The training dataset WBC-PBC (COMBINATION 2) contains cells from WBC and PBC as follows: 4 (monocytes), 6 (neutrophils) from WBC and cell types 0 (basophils), 1 (eosinophils), 3 (lymphocytes) from PBC. The last row shows the ensemble voting performance (including OT), and the average Δ_{OT} row summarizes the mean gain across all encoders.

Transfer: **RAABIN** *Train: WBC-PBC (COMB. 2)*	Acc. (OT: ✗)	Acc. (OT: ✓)	F1 (OT: ✗)	F1 (OT: ✓)	Δ Acc.	Δ F1
SAM	17.49	**26.84**	19.01	**53.04**	9.35	34.03
DINO	32.25	**51.07**	36.85	**52.57**	18.82	15.72
ConvNeXt	27.37	**37.80**	54.38	**58.39**	10.43	4.01
SWIN	26.13	**56.02**	47.04	**58.99**	29.89	11.94
CLIP	22.78	**39.96**	**52.12**	47.98	17.19	-4.14
ViTMAE	19.37	**27.32**	**27.98**	23.37	7.94	-4.61
DinoBloom	60.61	**68.44**	**65.17**	46.85	7.83	-18.32
CN + SWIN	29.47	**49.32**	57.22	**60.68**	19.85	3.46
SAM + CN + SWIN + CLIP	31.40	**51.24**	**60.20**	59.50	19.83	-0.7
DINO + CN + ViTMAE + SWIN	31.87	**55.08**	56.97	**61.08**	23.21	4.11
CN + SWIN + ViTMAE	31.58	**50.74**	**61.39**	59.78	19.16	-1.61
DINO + CN + SWIN	29.61	**55.18**	52.50	**62.26**	25.57	9.76
DINO + CN + ViTMAE	28.16	**44.44**	53.11	**61.37**	16.29	8.26
DINO + ViTMAE + SWIN	32.43	**62.27**	50.40	**57.87**	29.84	7.47
Ensemble Voting method	48.50	**68.71**	63.79	**70.13**	20.2	6.34
Average Δ_{OT}	-	-	-	-	18.36	5.05

Table 17: Comparison of encoder performance with class filtering and with (✓) or without (✗) optimal transport (OT) – Transfer dataset: LISC, training dataset: WBC-PBC (COMBINATION 1). The training dataset WBC-PBC (COMBINATION 1) contains cells from WBC and PBC as follows: cell types 0 (basophils), 1 (eosinophils), 3 (lymphocytes) from WBC and 4 (monocytes), 6 (neutrophils) from PBC. The last row shows the ensemble voting performance (including OT), and the average Δ_{OT} row summarizes the mean gain across all encoders.

Transfer: **LISC** *Train: WBC-PBC (COMB. 1)*	Acc. (OT: ✗)	Acc. (OT: ✓)	F1 (OT: ✗)	F1 (OT: ✓)	Δ Acc.	Δ F1
SAM	**37.91**	29.47	**33.95**	28.50	-8.43	-5.45
DINO	33.68	**48.34**	33.25	**50.57**	14.66	17.32
ConvNeXt	46.02	**53.91**	42.80	**52.10**	7.89	9.3
SWIN	22.19	**61.51**	18.83	**60.89**	39.32	42.05
CLIP	23.54	**41.40**	16.47	**41.09**	17.86	24.62
ViTMAE	**35.94**	25.31	**29.30**	23.39	-10.63	-5.91
DinoBloom	66.54	**73.56**	63.44	**72.72**	7.02	9.29
CN + SWIN	30.54	**61.16**	22.39	**59.97**	30.63	37.58
SAM + CN + SWIN + CLIP	31.71	**62.27**	23.52	**60.38**	30.56	36.86
DINO + CN + ViTMAE + SWIN	33.72	**62.24**	28.94	**61.96**	28.52	33.02
CN + SWIN + ViTMAE	29.37	**61.61**	21.65	**60.22**	32.24	38.57
DINO + CN + SWIN	31.71	**63.53**	25.84	**63.42**	31.82	37.58
DINO + CN + ViTMAE	52.23	**59.20**	54.40	**58.63**	6.97	4.23
DINO + ViTMAE + SWIN	28.72	**59.18**	21.11	**60.59**	30.45	39.48
Ensemble Voting method	52.73	**77.30**	56.24	**77.23**	24.57	20.99
Average Δ_{OT}	-	-	-	-	18.9	22.63

Table 18: Comparison of encoder performance with class filtering and with (✓) or without (✗) optimal transport (OT) – Transfer dataset: LISC, training dataset: WBC-PBC (COMBINATION 2). The training dataset WBC-PBC (COMBINATION 2) contains cells from WBC and PBC as follows: 4 (monocytes), 6 (neutrophils) from WBC and cell types 0 (basophils), 1 (eosinophils), 3 (lymphocytes) from PBC. The last row shows the ensemble voting performance (including OT), and the average Δ_{OT} row summarizes the mean gain across all encoders.

Transfer: **LISC** *Train: WBC-PBC (COMB. 2)*	Acc. (OT: ✗)	Acc. (OT: ✓)	F1 (OT: ✗)	F1 (OT: ✓)	Δ Acc.	Δ F1
SAM	30.88	**34.12**	26.84	**35.14**	3.24	8.3
DINO	40.37	**63.11**	36.80	**62.64**	22.75	25.83
ConvNeXt	38.21	**50.05**	37.44	**50.05**	11.84	12.61
SWIN	37.38	**53.78**	26.61	**51.24**	16.4	24.63
CLIP	21.98	**48.70**	10.69	**49.01**	26.73	38.32
ViTMAE	14.07	**25.80**	6.36	**26.62**	11.74	20.26
DinoBloom	**72.86**	60.82	**73.29**	58.75	-12.04	-14.54
CN + SWIN	39.04	**51.59**	32.78	**51.58**	12.54	18.81
SAM + CN + SWIN + CLIP	42.14	**49.03**	36.47	**49.57**	6.89	13.1
DINO + CN + ViTMAE + SWIN	40.25	**63.82**	37.46	**64.35**	23.57	26.88
CN + SWIN + ViTMAE	38.46	**51.77**	34.39	**52.01**	13.31	17.62
DINO + CN + SWIN	39.52	**64.14**	35.29	**64.89**	24.62	29.6
DINO + CN + ViTMAE	44.21	**67.70**	40.83	**69.06**	23.5	28.23
DINO + ViTMAE + SWIN	35.33	**68.27**	29.69	**69.35**	32.94	39.66
Ensemble Voting method	48.31	**73.50**	43.70	**72.29**	25.19	28.59
Average Δ_{OT}	-	-	-	-	16.21	21.19

Table 19: Comparison of encoder performance with class filtering and with (✓) or without (✗) optimal transport (OT) – Transfer dataset: LISC, training dataset: WBC + PBC + RAABIN. Before filtering, the training dataset WBC+PBC+RAABIN contains all cells from WBC, PBC and RAABIN, regardless of their cell type. Best results per encoder (and per metric) are shown in bold. Each Δ value represents the gain or loss in performance (in percentage points) from applying OT. The last row shows the ensemble voting performance (including OT), and the average Δ_{OT} row summarizes the mean gain across all encoders.

Transfer: **LISC** *Train: WBC+PBC+RAABIN*	Acc. (OT: ✗)	Acc. (OT: ✓)	F1 (OT: ✗)	F1 (OT: ✓)	Δ Acc.	Δ F1
SAM	31.77	**35.52**	25.17	**35.51**	3.75	10.34
DINO	33.12	**51.79**	28.07	**51.45**	18.67	23.38
ConvNeXt	55.59	**56.27**	**57.68**	56.56	0.68	-1.12
SWIN	24.72	**64.21**	16.37	**63.47**	39.49	47.1
CLIP	34.03	**40.97**	32.14	**40.35**	6.94	8.21
ViTMAE	24.83	**26.80**	15.99	**26.93**	1.98	10.95
DinoBloom	70.22	**77.26**	69.80	**76.34**	7.05	6.54
CN + SWIN	25.79	**61.45**	16.56	**62.53**	35.66	45.98
SAM + CN + SWIN + CLIP	27.24	**67.07**	19.79	**67.71**	39.83	47.92
DINO + CN + ViTMAE + SWIN	28.79	**68.64**	23.11	**69.49**	39.85	46.39
CN + SWIN + ViTMAE	24.95	**61.84**	15.25	**62.20**	36.89	46.95
DINO + CN + SWIN	27.95	**67.75**	21.48	**68.55**	39.8	47.07
DINO + CN + ViTMAE	52.57	**68.03**	53.01	**68.83**	15.46	15.83
DINO + ViTMAE + SWIN	23.74	**63.42**	15.31	**63.69**	39.68	48.38
Ensemble Voting method	55.39	**72.45**	52.90	**70.53**	17.06	17.63
Average Δ_{OT}	-	-	-	-	22.85	28.1

Table 20: Comparison of encoder performance with class filtering and with (✓) or without (✗) optimal transport (OT) – Transfer dataset: Lionheart (2 classes), training dataset: Nanolive. The two classes are: (0) Live cells, (1) Dead cells. The last row shows the ensemble voting performance (including OT), and the average Δ_{OT} row summarizes the mean gain across all encoders.

Transfer: **Lionheart (2 classes)** *Train: Nanolive*	Acc. (OT: ✗)	Acc. (OT: ✓)	F1 (OT: ✗)	F1 (OT: ✓)	Δ Acc.	Δ F1
SAM	34.48	**43.00**	37.51	**45.76**	8.51	8.25
DINO	**77.66**	64.98	**80.51**	67.32	-12.68	-13.19
ConvNeXt	54.41	**76.45**	47.08	**77.87**	22.04	30.79
SWIN	56.16	**75.66**	39.90	**77.64**	19.5	37.74
CLIP	**79.41**	72.89	**79.79**	74.47	-6.52	-5.33
ViTMAE	72.83	**77.23**	67.16	**78.05**	4.41	10.88
DinoBloom	**76.27**	64.98	**78.92**	67.32	-11.29	-11.6
CN + SWIN	47.64	**77.23**	29.23	**78.05**	29.59	48.81
SAM + CN + SWIN + CLIP	51.21	**80.80**	33.17	**81.42**	29.59	48.26
DINO + CN + ViTMAE + SWIN	**78.20**	67.75	**76.55**	70.37	-10.45	-6.18
CN + SWIN + ViTMAE	49.03	**77.05**	32.14	**79.23**	28.02	47.08
DINO + CN + SWIN	**80.98**	67.75	**79.91**	70.37	-13.22	-9.54
DINO + CN + ViTMAE	**71.32**	68.54	**73.85**	70.76	-2.78	-3.09
DINO + ViTMAE + SWIN	75.66	**80.80**	77.64	**81.42**	5.13	3.78
Ensemble Voting method	**78.62**	74.88	**79.66**	77.34	-3.74	-2.32
Average Δ_{OT}	-	-	-	-	5.74	12.29

Table 21: Comparison of encoder performance with class filtering and with (✓) or without (✗) optimal transport (OT) – Transfer dataset: Lionheart (3 classes), training dataset: Nanolive. The two classes are: (0) Live cells, (1) Apoptotic cells, (2) Necrotic cells. The last row shows the ensemble voting performance (including OT), and the average Δ_{OT} row summarizes the mean gain across all encoders.

Transfer: **Lionheart (3 classes)** *Train: Nanolive*	Acc. (OT: ✗)	Acc. (OT: ✓)	F1 (OT: ✗)	F1 (OT: ✓)	Δ Acc.	Δ F1
SAM	**32.28**	27.45	12.97	**41.25**	-4.83	28.27
DINO	54.40	**54.46**	**72.44**	62.32	0.07	-10.11
ConvNeXt	36.11	**51.52**	41.48	**68.76**	15.41	27.29
SWIN	50.76	**61.97**	53.44	**73.49**	11.21	20.05
CLIP	**72.52**	57.71	**73.16**	64.63	-14.81	-8.52
ViTMAE	51.39	**55.26**	43.30	**61.77**	3.87	18.48
DinoBloom	**61.24**	49.11	**78.95**	63.21	-12.14	-15.75
CN + SWIN	32.77	**55.46**	36.50	**72.22**	22.69	35.72
SAM + CN + SWIN + CLIP	39.95	**54.53**	41.83	**70.52**	14.58	28.7
DINO + CN + ViTMAE + SWIN	**57.87**	47.95	**74.16**	63.67	-9.92	-10.49
CN + SWIN + ViTMAE	34.85	**54.53**	37.72	**71.76**	19.68	34.04
DINO + CN + SWIN	**58.80**	43.19	**75.78**	62.03	-15.61	-13.75
DINO + CN + ViTMAE	**52.55**	46.46	**70.64**	57.79	-6.08	-12.85
DINO + ViTMAE + SWIN	52.78	**53.97**	69.15	**69.33**	1.19	0.17
Ensemble Voting method	**62.07**	58.83	**77.13**	74.79	-3.24	-2.34
Average Δ_{OT}	-	-	-	-	1.47	7.93

Table 22: Comparison of encoder performance with class filtering and with (✓) or without (✗) optimal transport (OT) – Transfer dataset: Nanolive (2 classes), training dataset: Lionheart. The two classes are: (0) Live cells, (1) Dead cells. The last row shows the ensemble voting performance (including OT), and the average Δ_{OT} row summarizes the mean gain across all encoders.

Transfer: **Nanolive (2 classes)** *Train: Lionheart*	Acc. (OT: ✗)	Acc. (OT: ✓)	F1 (OT: ✗)	F1 (OT: ✓)	Δ Acc.	Δ F1
SAM	**50.00**	44.37	20.70	**48.14**	-5.63	27.44
DINO	63.45	**68.14**	67.02	**70.26**	4.69	3.24
ConvNeXt	74.78	**75.49**	76.25	**77.99**	0.71	1.74
SWIN	63.52	**69.58**	57.62	**70.02**	6.06	12.4
CLIP	68.53	**70.70**	68.48	**70.81**	2.17	2.33
ViTMAE	56.97	**58.54**	41.61	**60.04**	1.57	18.43
DinoBloom	**71.53**	68.38	**72.00**	71.02	-3.15	-0.98
CN + SWIN	71.54	**71.62**	68.86	**72.77**	0.08	3.9
SAM + CN + SWIN + CLIP	71.76	**71.88**	69.20	**72.83**	0.12	3.62
DINO + CN + ViTMAE + SWIN	68.66	**71.57**	71.05	**72.95**	2.9	1.9
CN + SWIN + ViTMAE	71.25	**71.65**	68.43	**72.72**	0.4	4.3
DINO + CN + SWIN	68.55	**71.59**	70.99	**72.99**	3.04	2.0
DINO + CN + ViTMAE	66.95	**70.57**	70.04	**72.40**	3.62	2.36
DINO + ViTMAE + SWIN	64.84	**69.62**	67.88	**71.11**	4.78	3.23
Ensemble Voting method	**73.28**	72.59	72.88	**73.44**	-0.69	0.55
Average Δ_{OT}	-	-	-	-	1.38	5.77

Table 23: Comparison of encoder performance with class filtering and with (✓) or without (✗) optimal transport (OT) – Transfer dataset: Nanolive (3 classes), training dataset: Lionheart. The two classes are: (0) Live cells, (1) Apoptotic cells, (2) Necrotic cells. The last row shows the ensemble voting performance (including OT), and the average Δ_{OT} row summarizes the mean gain across all encoders.

Transfer: **Nanolive (3 classes)** *Train: Lionheart*	Acc. (OT: ✗)	Acc. (OT: ✓)	F1 (OT: ✗)	F1 (OT: ✓)	Δ Acc.	Δ F1
SAM	**31.35**	29.01	2.51	**49.43**	-2.34	46.92
DINO	44.57	**45.89**	**61.73**	57.60	1.32	-4.13
ConvNeXt	50.22	**57.53**	56.51	**59.84**	7.31	3.33
SWIN	48.53	**57.95**	50.50	**67.66**	9.42	17.16
CLIP	**62.71**	61.11	**62.42**	62.22	-1.6	-0.2
ViTMAE	44.80	**51.10**	24.56	**54.17**	6.3	29.61
DinoBloom	60.11	**65.52**	63.35	**66.27**	5.41	2.92
CN + SWIN	49.82	**60.05**	49.33	**63.26**	10.23	13.93
SAM + CN + SWIN + CLIP	52.38	**61.15**	51.43	**64.11**	8.76	12.68
DINO + CN + ViTMAE + SWIN	47.37	**56.98**	61.48	**62.94**	9.61	1.46
CN + SWIN + ViTMAE	49.38	**59.80**	48.72	**63.12**	10.41	14.4
DINO + CN + SWIN	47.35	**56.78**	61.50	**62.95**	9.43	1.45
DINO + CN + ViTMAE	46.72	**52.32**	**61.96**	60.48	5.6	-1.48
DINO + ViTMAE + SWIN	45.69	**51.68**	**61.52**	60.81	5.99	-0.72
Ensemble Voting method	61.52	**65.91**	63.97	**67.43**	4.38	3.46
Average Δ_{OT}	-	-	-	-	6.02	9.39