A Acknowledgments

We would like to acknowledge that Figure 1 was created with BioRender.com. Further, we acknowledge that Large Language Models (LLMs) were used throughout the study.

B PROOFS

B.1 PROOF OF MHA PERMUTATION PROPERTIES

We need to prove three properties of Multi-Head Attention (MHA):

1. Permutation Equivariance: When all inputs are permuted. 2. Permutation Equivariance for Q: When only the query is permuted. 3. Permutation Invariance for K and V: When only keys and values are permuted.

Proof. Let $\mathbf{Q} \in \mathbb{R}^{n_q \times d}$, $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{n_{kv} \times d}$ be the query, key, and value matrices respectively, and $\mathbf{P}_q \in \mathbb{R}^{n_q \times n_q}$, $\mathbf{P}_{kv} \in \mathbb{R}^{n_{kv} \times n_{kv}}$ be permutation matrices.

1. Permutation Equivariance (all inputs): When $n_q = n_{kv}$ and $\mathbf{P} = \mathbf{P}_q = \mathbf{P}_{kv}$:

$$\begin{aligned}
\text{MHA}(\mathbf{PQ}, \mathbf{PK}, \mathbf{PV}) &= \text{softmax}\left(\frac{(\mathbf{PQ})(\mathbf{PK})^\top}{\sqrt{d_k}}\right)\mathbf{PV} \\
&= \text{softmax}\left(\frac{\mathbf{PQK}^\top\mathbf{P}^\top}{\sqrt{d_k}}\right)\mathbf{PV} \\
&= \mathbf{P}\text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_k}}\right)\mathbf{V} \\
&= \mathbf{P}\text{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V})
\end{aligned}$$

2. Permutation Equivariance for Q: When only Q is permuted:

$$\begin{aligned}
\text{MHA}(\mathbf{P}_q\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \text{softmax}\left(\frac{(\mathbf{P}_q\mathbf{Q})\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \\
&= \text{softmax}\left(\frac{\mathbf{P}_q\mathbf{QK}^\top}{\sqrt{d_k}}\right)\mathbf{V} \\
&= \mathbf{P}_q\text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_k}}\right)\mathbf{V} \\
&= \mathbf{P}_q\text{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V})
\end{aligned}$$

3. Permutation Invariance for K and V: When only K and V are permuted:

$$\begin{aligned}
\text{MHA}(\mathbf{Q}, \mathbf{P}_{kv}\mathbf{K}, \mathbf{P}_{kv}\mathbf{V}) &= \text{softmax}\left(\frac{\mathbf{Q}(\mathbf{P}_{kv}\mathbf{K})^\top}{\sqrt{d_k}}\right)\mathbf{P}_{kv}\mathbf{V} \\
&= \text{softmax}\left(\frac{\mathbf{QK}^\top\mathbf{P}_{kv}^\top}{\sqrt{d_k}}\right)\mathbf{P}_{kv}\mathbf{V} \\
&= \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_k}}\right)\mathbf{V} \\
&= \text{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V})
\end{aligned}$$

These properties together demonstrate that: a) MHA is permutation equivariant when all inputs are permuted equally. b) MHA is permutation equivariant with respect to Q when only Q is permuted. c) MHA is permutation invariant with respect to K and V when only K and V are permuted.

□

B.2 Proof of ISAB Permutation Equivariance

To prove that the Induced Set-Attention Block (ISAB) is permutation equivariant, we need to show that for any permutation matrix $\mathbf{P}$, $\text{ISAB}_m(\mathbf{P}\mathcal{S}) = \mathbf{P}\,\text{ISAB}_m(\mathcal{S})$.

Proof. Let $\mathcal{S} \in \mathbb{R}^{n\times d}$ be the input set, $\mathbf{I} \in \mathbb{R}^{m\times d}$ be the set of learnable inducing points, and $\mathbf{P} \in \mathbb{R}^{n\times n}$ be any permutation matrix.

Recall that ISAB is defined as a two-step process:

$$\begin{aligned}\mathbf{H} &= \text{MAB}(\mathbf{I}, \mathcal{S}, \mathcal{S}) \\ \text{ISAB}_m(\mathcal{S}) &= \text{MAB}(\mathcal{S}, \mathbf{H}, \mathbf{H})\end{aligned}$$

1) First, we examine how permutation affects $\mathbf{H}$:

$$\begin{aligned}\mathbf{H}' &= \text{MAB}(\mathbf{I}, \mathbf{P}\mathcal{S}, \mathbf{P}\mathcal{S}) \\ &= \text{LayerNorm}(\mathbf{I} + \text{MHA}(\mathbf{I}, \mathbf{P}\mathcal{S}, \mathbf{P}\mathcal{S}))\end{aligned}$$

From our above MHA proof, we know that MHA is invariant when only K and V are permuted. Therefore:

$$\mathbf{H}' = \mathbf{H}$$

2) Now, we examine the second step:

$$\begin{aligned}\text{ISAB}_m(\mathbf{P}\mathcal{S}) &= \text{MAB}(\mathbf{P}\mathcal{S}, \mathbf{H}, \mathbf{H}) \\ &= \text{LayerNorm}(\mathbf{P}\mathcal{S} + \text{MHA}(\mathbf{P}\mathcal{S}, \mathbf{H}, \mathbf{H}))\end{aligned}$$

3) To show equivariance, we need to prove that this is equal to $\mathbf{P}\text{ISAB}_m(\mathcal{S})$:

$$\begin{aligned}\mathbf{P}\text{ISAB}_m(\mathcal{S}) &= \mathbf{P}\text{MAB}(\mathcal{S}, \mathbf{H}, \mathbf{H}) \\ &= \mathbf{P}\text{LayerNorm}(\mathcal{S} + \text{MHA}(\mathcal{S}, \mathbf{H}, \mathbf{H}))\end{aligned}$$

4) The equality holds because:

$$\begin{aligned}\text{LayerNorm}(\mathbf{P}\mathcal{S} + \text{MHA}(\mathbf{P}\mathcal{S}, \mathbf{H}, \mathbf{H})) &= \text{LayerNorm}(\mathbf{P}\mathcal{S} + \mathbf{P}\text{MHA}(\mathcal{S}, \mathbf{H}, \mathbf{H})) \\ &= \mathbf{P}\text{LayerNorm}(\mathcal{S} + \text{MHA}(\mathcal{S}, \mathbf{H}, \mathbf{H}))\end{aligned}$$

This equality holds because: a) LayerNorm is permutation equivariant b) MHA is permutation equivariant in its first argument (as shown in the revised MHA proof) c) $\mathbf{H}$ remains unchanged under permutation of $\mathcal{S}$

Therefore, we have shown that $\text{ISAB}_m(\mathbf{P}\mathcal{S}) = \mathbf{P}\,\text{ISAB}_m(\mathcal{S})$ for any permutation matrix $\mathbf{P}$, proving that ISAB is permutation equivariant. ☐

B.3 Proof of PMA Permutation Invariance

To prove that the Pooling by Multihead Attention (PMA) operation is permutation invariant, we need to show that for any permutation matrix $\mathbf{P}$ applied to the input set $\mathcal{S}$, PMA($\mathbf{P}\mathcal{S}$) = PMA($\mathcal{S}$).

Proof. Let $\mathcal{S} \in \mathbb{R}^{n\times d}$ be the input set, $\mathbf{s} \in \mathbb{R}^{1\times d}$ be the learnable token, and $\mathbf{P} \in \mathbb{R}^{n\times n}$ be any permutation matrix.

Recall that PMA is defined as:

$$\text{PMA}(\mathcal{S}) = \text{MAB}(\mathbf{s}, \mathcal{S}, \mathcal{S})$$

1) First, let's consider PMA applied to the permuted input:

$$\text{PMA}(\mathbf{P}\mathcal{S}) = \text{MAB}(\mathbf{s}, \mathbf{P}\mathcal{S}, \mathbf{P}\mathcal{S})$$

2) Expanding the MAB operation:

$$\text{MAB}(\mathbf{s}, \mathbf{P}\mathcal{S}, \mathbf{P}\mathcal{S}) = \text{LayerNorm}(\mathbf{s} + \text{MHA}(\mathbf{s}, \mathbf{P}\mathcal{S}, \mathbf{P}\mathcal{S}))$$

3) From our above MHA proof, we know that MHA is invariant when only K and V are permuted, which is the case here since $\mathbf{s}$ is not permuted. Therefore:

$$\text{MHA}(\mathbf{s}, \mathbf{P}\mathcal{S}, \mathbf{P}\mathcal{S}) = \text{MHA}(\mathbf{s}, \mathcal{S}, \mathcal{S})$$

4) Consequently:

$$\begin{aligned}\text{MAB}(\mathbf{s}, \mathbf{P}\mathcal{S}, \mathbf{P}\mathcal{S}) &= \text{LayerNorm}(\mathbf{s} + \text{MHA}(\mathbf{s}, \mathbf{P}\mathcal{S}, \mathbf{P}\mathcal{S})) \\ &= \text{LayerNorm}(\mathbf{s} + \text{MHA}(\mathbf{s}, \mathcal{S}, \mathcal{S})) \\ &= \text{MAB}(\mathbf{s}, \mathcal{S}, \mathcal{S})\end{aligned}$$

5) This shows that:

$$\text{PMA}(\mathbf{P}\mathcal{S}) = \text{PMA}(\mathcal{S})$$

Therefore, we have shown that PMA($\mathbf{P}\mathcal{S}$) = PMA($\mathcal{S}$) for any permutation matrix $\mathbf{P}$, proving that PMA is permutation invariant.

$\square$

B.4 Proof of SAB Permutation Equivariance

To prove that the Self-Attention Block (SAB) is permutation equivariant, we need to show that for any permutation matrix $\mathbf{P}$, $\text{SAB}(\mathbf{P}\mathcal{S}) = \mathbf{P}\,\text{SAB}(\mathcal{S})$.

Proof. Let $\mathcal{S} \in \mathbb{R}^{n\times d}$ be the input set and $\mathbf{P} \in \mathbb{R}^{n\times n}$ be any permutation matrix.

Recall that SAB is defined as:

$$\text{SAB}(\mathcal{S}) := \text{MAB}(\mathcal{S}, \mathcal{S}, \mathcal{S})$$

1) We start by expanding the MAB operation:

$$\begin{aligned} \text{SAB}(\mathcal{S}) &= \text{MAB}(\mathcal{S}, \mathcal{S}, \mathcal{S}) \\ &= \text{LayerNorm}(H + \text{FF}(H)) \\ \text{where } H &= \text{LayerNorm}(\mathcal{S} + \text{MHA}(\mathcal{S}, \mathcal{S}, \mathcal{S})) \end{aligned}$$

2) Now, let's apply a permutation $\mathbf{P}$ to the input:

$$\begin{aligned} \text{SAB}(\mathbf{P}\mathcal{S}) &= \text{MAB}(\mathbf{P}\mathcal{S}, \mathbf{P}\mathcal{S}, \mathbf{P}\mathcal{S}) \\ &= \text{LayerNorm}(H' + \text{FF}(H')) \\ \text{where } H' &= \text{LayerNorm}(\mathbf{P}\mathcal{S} + \text{MHA}(\mathbf{P}\mathcal{S}, \mathbf{P}\mathcal{S}, \mathbf{P}\mathcal{S})) \end{aligned}$$

3) Using the permutation equivariance of MHA when all inputs are permuted (proven earlier):

$$\begin{aligned} H' &= \text{LayerNorm}(\mathbf{P}\mathcal{S} + \mathbf{P}\text{MHA}(\mathcal{S}, \mathcal{S}, \mathcal{S})) \\ &= \text{LayerNorm}(\mathbf{P}(\mathcal{S} + \text{MHA}(\mathcal{S}, \mathcal{S}, \mathcal{S}))) \\ &= \mathbf{P}\text{LayerNorm}(\mathcal{S} + \text{MHA}(\mathcal{S}, \mathcal{S}, \mathcal{S})) \\ &= \mathbf{P}H \end{aligned}$$

4) The feedforward network FF is applied elementwise, so it's permutation equivariant:

$$\text{FF}(\mathbf{P}H) = \mathbf{P}\text{FF}(H)$$

5) Combining these results:

$$\begin{aligned} \text{SAB}(\mathbf{P}\mathcal{S}) &= \text{LayerNorm}(\mathbf{P}H + \text{FF}(\mathbf{P}H)) \\ &= \text{LayerNorm}(\mathbf{P}H + \mathbf{P}\text{FF}(H)) \\ &= \text{LayerNorm}(\mathbf{P}(H + \text{FF}(H))) \\ &= \mathbf{P}\text{LayerNorm}(H + \text{FF}(H)) \\ &= \mathbf{P}\text{SAB}(\mathcal{S}) \end{aligned}$$

Therefore, we have shown that $\text{SAB}(\mathbf{P}\mathcal{S}) = \mathbf{P}\,\text{SAB}(\mathcal{S})$ for any permutation matrix $\mathbf{P}$, proving that SAB is permutation equivariant. □

C Formal Definitions

C.1 Non-Random Block Masking

Let $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be an input set where each $\mathbf{x}_i \in \mathbb{R}^d$, and let $\rho \in [0, 1]$ be the desired mask ratio. The Non-Random Block Masking (NRBM) process consists of the following steps:

1. **Random Element Selection:** Select a random element $\mathbf{r}$ from the input set:

   $$\mathbf{r} = \mathbf{x}_k, \quad k \sim \text{Uniform}\{1, \ldots, n\}$$

   This element serves as a reference for similarity calculations.
2. **Similarity Calculation:** Compute the cosine similarity between each element and the randomly selected element:

   $$s_i = \text{cosine\_similarity}(\mathbf{x}_i, \mathbf{r}) = \frac{\mathbf{x}_i \cdot \mathbf{r}}{\|\mathbf{x}_i\|\|\mathbf{r}\|}, \quad \forall i \in \{1, \ldots, n\}$$

3. **Similarity-based Ordering:** Define a permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ that sorts elements based on their similarity to $\mathbf{r}$:

   $$s_{\pi(i)} \geq s_{\pi(j)} \quad \text{for all } i < j$$

   This ordering ensures that similar elements are grouped together.
4. **Block Mask Creation:** Create a binary mask $\mathbf{M} = (M_1, \ldots, M_n)$ with the following parameters:
   - Number of elements to mask: $m = \lfloor n\rho \rfloor$
   - Block size: $b = \max(1, \lfloor \frac{m}{n/m} \rfloor)$

   Define the mask as:

   $$M_i = \begin{cases} 1, & \text{if } \pi^{-1}(i) \bmod (2b) < b \text{ and } \sum_{j=1}^{i-1} M_j < m \\ 0, & \text{otherwise} \end{cases}$$

   This creates contiguous blocks of masked elements in the similarity-ordered sequence.
5. **Masked Set Creation:** Let $\mathbf{t} \in \mathbb{R}^d$ be the mask token. Create the masked set $\mathcal{S}'$ by replacing masked elements with $\mathbf{t}$:

   $$\mathcal{S}' = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\}, \quad \text{where } \mathbf{y}_i = \begin{cases} \mathbf{t}, & \text{if } M_i = 1 \\ \mathbf{x}_i, & \text{if } M_i = 0 \end{cases}$$

The complete NRBM function is then defined as:

$$\text{NRBM} : \mathcal{P}(\mathbb{R}^d) \times [0, 1] \to \mathcal{P}(\mathbb{R}^d) \times \{0, 1\}^n \times \mathcal{S}_n$$

$$\text{NRBM}(\mathcal{S}, \rho) = (\mathcal{S}', \mathbf{M}, \pi)$$

where $\mathcal{S}'$ is the masked set, $\mathbf{M}$ is the binary mask, and π is the permutation used for sorting.

This formulation ensures that:

- The masking is based on similarity to a randomly chosen element, promoting semantic coherence.
- Masks are applied in contiguous blocks, encouraging the model to learn robust patterns.
- The original ordering can be restored using the returned permutation π.

C.2 Sinkhorn Optimal Transport

To ensure our reconstruction loss is permutation-invariant, we employ Sinkhorn Optimal Transport. This method allows us to compare the original and reconstructed sets without relying on a specific ordering of elements.

The key idea is to find an optimal 'matching" between the original set $\mathcal{S} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ and the reconstructed set $\hat{\mathcal{S}} = \{\hat{\mathbf{x}}_1, ..., \hat{\mathbf{x}}_n\}$. This matching is represented by a transport plan $\mathbf{P}$.

The transport plan $\mathbf{P}$ is an $n \times n$ matrix where each element P_{ij} represents how much of element $\mathbf{x}_i$ from the original set is 'transported" to element $\hat{\mathbf{x}}_j$ in the reconstructed set. Formally:

$$\mathbf{P} = [P_{ij}] \in \mathbb{R}^{n \times n}, \quad \text{where } 0 \leq P_{ij} \leq 1$$

To ensure that $\mathbf{P}$ represents a valid matching, we constrain it to be a doubly stochastic matrix:

$$\sum_{j=1}^{n} P_{ij} = 1 \quad \text{for all } i, \quad \text{and} \quad \sum_{i=1}^{n} P_{ij} = 1 \quad \text{for all } j$$

These constraints ensure that each original element is fully 'distributed" across the reconstructed elements, and each reconstructed element is fully 'accounted for" by the original elements.

We define a cost matrix $\mathbf{C}$, where each element C_{ij} represents the 'cost" of transporting $\mathbf{x}_i$ to $\hat{\mathbf{x}}_j$:

$$C_{ij} = \|\mathbf{x}_i - \hat{\mathbf{x}}_j\|_2^2$$

The goal is to find the optimal transport plan $\mathbf{P}^*$ that minimizes the total transport cost:

$$\mathbf{P}^* = \arg\min_{\mathbf{P}} \sum_{i,j} P_{ij} C_{ij} \quad \text{subject to the constraints on } \mathbf{P}$$

To make this optimization problem tractable, we add an entropy regularization term:

$$\mathbf{P}^* = \arg\min_{\mathbf{P}} \sum_{i,j} P_{ij} C_{ij} - \epsilon \sum_{i,j} P_{ij} \log P_{ij}$$

where $\epsilon > 0$ is the regularization strength.

This problem can be solved efficiently using the Sinkhorn-Knopp algorithm, which iteratively updates $\mathbf{P}$ until convergence.

Finally, we define our reconstruction loss as:

$$\mathcal{L}_{\text{rec}} = \sum_{i,j} P^*_{ij} C_{ij}$$

This loss function ensures that the reconstruction quality is evaluated in a permutation-invariant manner, as it only depends on the optimal matching between the original and reconstructed sets, not their order.

D Appendix Tables

D.1 Diagnosis

Model	**Accuracy**	**AUROC**	**F1**
Manual Gating	0.381 ± 0.021	0.781 ± 0.027	0.190 ± 0.015
KMeans (k = 1024)	0.563 ± 0.019	0.938 ± 0.007	0.334 ± 0.019
OTKE (k = 1024)	0.878 ± 0.015	0.988 ± 0.003	0.838 ± 0.066
Deep Sets (supervised)	0.836 ± 0.016	0.981 ± 0.002	0.842 ± 0.016
Set Transformer (supervised)	0.884 ± 0.019	0.989 ± 0.001	0.851 ± 0.015
MAESTRO (ours)	$\mathbf{0.923 \pm 0.014}$	$\mathbf{0.992 \pm 0.003}$	$\mathbf{0.897 \pm 0.029}$

Table 2: **Diagnosis Classification Benchmark.** MAESTRO outperforms all benchmarked methods across Accuracy, AUROC, and F1 metrics for diagnosis classification.

D.2 Sex

Model	**Accuracy**	**AUROC**	**F1**
Manual Gating	0.528 ± 0.066	0.540 ± 0.074	0.515 ± 0.078
KMeans (k = 1024)	0.631 ± 0.029	0.687 ± 0.036	0.601 ± 0.052
OTKE (k = 1024)	0.513 ± 0.053	0.521 ± 0.072	0.510 ± 0.083
Deep Sets (supervised)	0.682 ± 0.058	0.721 ± 0.065	0.665 ± 0.071
Set Transformer (supervised)	0.698 ± 0.062	0.756 ± 0.059	0.685 ± 0.067
MAESTRO (ours)	$\mathbf{0.723 \pm 0.061}$	$\mathbf{0.800 \pm 0.051}$	$\mathbf{0.726 \pm 0.062}$

Table 3: **Sex Prediction Benchmark.** MAESTRO achieves superior performance across Accuracy, AUROC, and F1 metrics for sex prediction tasks compared to other methods.

D.3 Age

Model	**MAE**	**R^2**
Manual Gating	19.96 ±39.98	-0.670 ± 0.128
KMeans (k = 1024)	18.74 ±34.58	-0.370 ± 0.111
OTKE (k = 1024)	8.46 ±18.69	0.647 ± 0.099
Deep Sets (supervised)	7.83 ± 1.21	0.662 ± 0.082
Set Transformer (supervised)	7.46 ± 1.06	0.687 ± 0.054
MAESTRO (ours)	$\mathbf{6.70 \pm 0.99}$	$\mathbf{0.711 \pm 0.088}$

Table 4: **Age Prediction Benchmark.** MAESTRO outperforms all benchmarked methods with the lowest Mean Absolute Error (MAE) and the highest R^2 score in age prediction tasks.

E Appendix Figures

E.1 Data

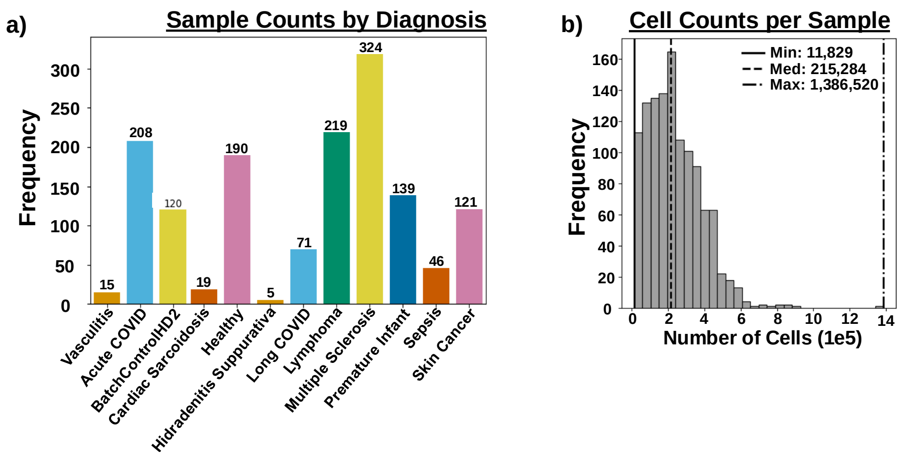

Figure 6: a) We show the counts of samples for each of the 11 different phenotypes (BatchControlHD2 is Healthy) used in our analysis. In total, there are 1,514 samples used. b) We show the distribution of number of cells per sample (the set size). The minimum set size is 11,829 cells and the maximum set size is 1,386,520 cells.

E.2 Data Generation: Time and Place

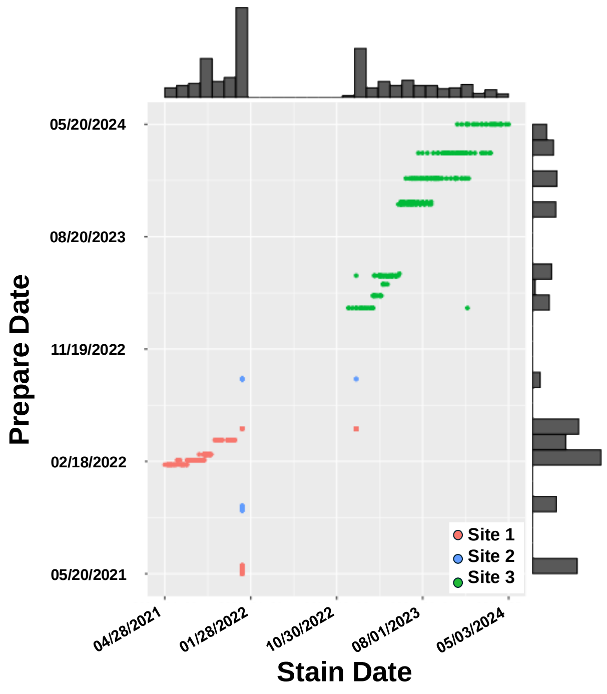

Figure 7: The x-axis denotes the date that the sample was stained with antibodies for protein marker detection, the y-axis denotes the preparation date, with histograms showing the distribution across these dates. The color denotes the location that the sample was processed which we keep anonymous for double blind review.

E.3 Batch Effects

E.3.1 All Samples

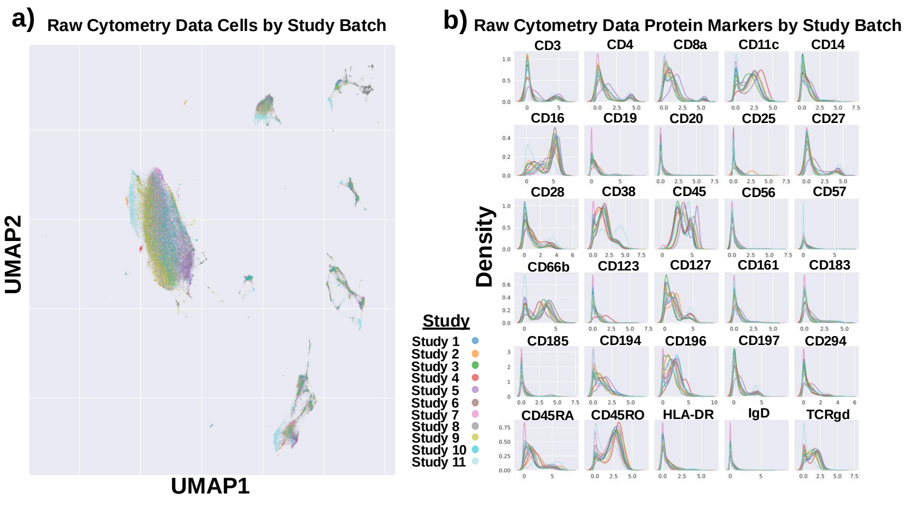

Figure 8: We randomly sample 1000 cells from all of our samples. a) We plot all of the cells on a UMAP colored by their study cohort. While the cells globally cluster by cell-type, we show that there are clear batch effects across the cohorts. b) We take the marker intensity distributions for each study and plot the density. This plot clearly demonstrates batch effects across the studies.

E.3.2 BatchControlHD2 Technical Control Only

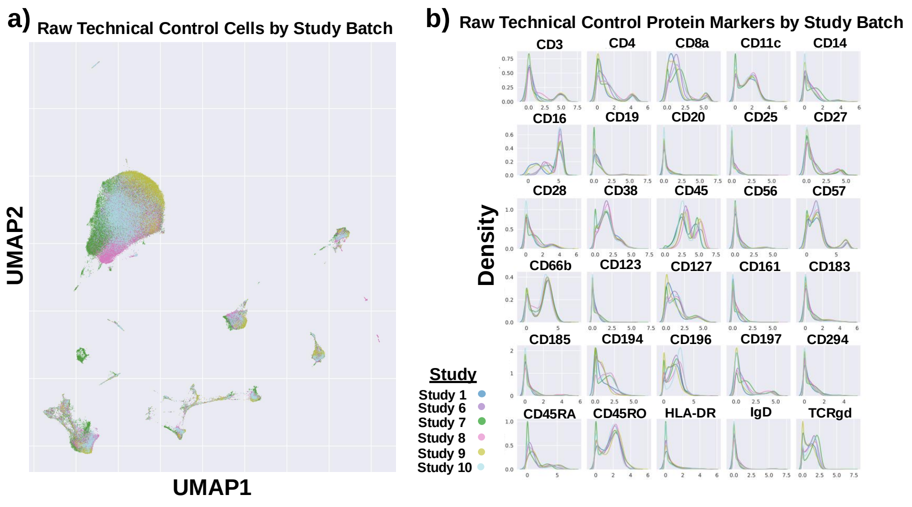

Figure 9: We use BatchControlHD2 as a technical control across the denoted studies. a) Here we randomly sample 1000 cells from each of these samples and plot them using on a UMAP which clearly show batch effects. b) We plot the distribution of protein marker expression for each study that BatchControlHD2 is in. Lastly, in Figure 3 we show that BatchControlHD2 clusters together demonstrating that there are less batch effects after processing with MAESTRO.

F Appendix Information

F.1 Data

We use CyTOF (Cytometry by Time of Flight) to generate our cytometry dataset. CyTOF captures high-dimensional data through mass cytometry, a technique that simultaneously measures up to 50 or more protein markers at the single-cell level. Unlike traditional flow cytometry, CyTOF uses metal isotopes conjugated to antibodies to bind specific cellular proteins. These metal-tagged antibodies are then detected using time-of-flight mass spectrometry, which minimizes signal overlap and allows for precise and comprehensive profiling of cellular phenotypes. This method facilitates in-depth analysis of complex biological systems, offering a detailed snapshot of cellular states and functions across large cell populations.

F.2 Pre-Training

Using 1,514 whole blood cytometry samples, we create a train and test set using an 80/20 random split within each diagnosis to ensure that all diagnoses are represented in the test set. This test dataset is never used for training during pre-training, linear probing, ablation, or cell-type proportion retrieval. We pre-train MAESTRO on four NVIDIA A100 GPU's for 156 hours at 29 minutes per epoch (324 epochs). Memory usage varies as each sample is of variable size, but on average the model consumes 58 GB per GPU. We train on our dataset with AdamW optimizer and a batch size of 1. Due to varaible size inputs, we can only use a batch size of one (one for each GPU - four samples at once). The learning rate starts at 1e-4 and using a cosine annealing scheduler to a minimum of 1e-8. We use NRBM to mask four different copies of our samples at masking rates of 20%, 40%, 60%, and 80%. Masking is done on the fly during pre-training and not before-hand. We mask by replacing selected indices with a learnable mask token. MAESTRO is configured with four attention heads, a hidden size of 2,048, and a latent dimension of 1,024. Our input cells start at 30 dimensions (representing 30 different protein markers), which we use a linear layer to transform to 1,024 dimensions before using three ISAB (with 2,500 learned points per ISAB) blocks, followed by PMA. After pooling to a single learnable seed (vector), we decode the vector by copying the pooled input to each index of the unmasked cells and use the masking token to denote indices where the original cell was masked. We follow this with a PMA block with number of seeds equal to the size of the input matrix, and follow this with three SAB blocks. Finally we use a linear layer to transform our output to its original 30 dimensions. All attention blocks use a SwiGLU activation function to learn both linear and non-linear relationships (Shazeer, 2020). Reconstruction is calculated using Sinkhorn Optimal Transport with Euclidean distance for the cost matrix calculation. We align latent representations from the student and teacher model by using a non-linear projection head on the latent representation. Following the projection head we use softmax activation to convert our representations to probability distributions. We use Kullback-Leibler (KL) - Divergence to minimize the difference in distributions. The teacher model is updated with the exponential moving average of the student and a momentum value of 0.999. Student and teacher model temperatures are set to 0.1 and 0.07, respectively.

F.3 Benchmarking method implementation

MAESTRO is pre-trained with an 80/20 train/test set. For linear probing, the same split is used for training and testing the logistic regression model in all benchmarked methods. The logistic regression model is implemented using scikit-learn LogisticRegression(). We repeat five-fold cross-validation 10 times for each method. Due to imbalances of diagnoses we report the Accuracy, AUROC, AUPRC, and F1-Score.

MAESTRO MAESTRO is pre-trained according to the above. Following pre-training, latent representation for all samples are derived using the pre-trained MAESTRO model. Each representation is a vector with 1,024 dimensions. Following, the samples are split into the train and test groups as was used during pre-training to train and test a logistic regression model. The logistic regression model is fit on the training set for 14 various diagnoses, and then inference is done on the held-out test set.

K-means We build two version of the K-means representation. One where we set k = 512, and the other k = 1024. We build these representations using the same 80% of training samples used for all other methods. We perform k-means clustering on all available cells across the samples. Then, for each sample we calculate the proportions of each cluster that exist. This vector of cluster proportions is used as the input to a logistic regression model. The logistic regression model is fit on the training set for 14 various diagnoses, and then inference is done on the held-out test set. Due to imbalances of diagnoses we report the Accuracy, AUROC, AUPRC, and F1-Score.

OTKE The OTKE method is unable to take variable sized sets and using large input sets overloaded the available GPU memory. For each sample, we use a random subset of 10,000 cells (equal to the number of cells input to the student model of MAESTRO). Then, we train the OTKE method, unsupervised, using our training samples. We use the default parameters as described by OTKE (Mialon et al., 2022). We then create representations for all of our samples using the learned kernel. We use the training samples to train a logistic regression model and report metrics on the held-out test set.

Deep Sets As above, Deep Sets is subset to 10,000 cells (same as student model of MAESTRO). Deep Sets is inherently a supervised model so we directly train the model to predict the 14 diagnoses using the 80% training set. We traing using hidden dimension of 2,048 (same as MAESTRO), and train until convergence or max 300 epochs (same as MAESTRO). We report metrics on the held-out test set.

Set Transformer As above, Set Transformer is subset to 10,000 cells (same as student model of MAESTRO). Set Transformer is inherently a supervised model so we directly train the model to predict the 14 diagnoses using the 80% training set. We traing using the same encoder architecture as MAESTRO (linear layer, three ISABs, and a PMA) using hidden dimension of 2,048 (same as MAESTRO), and a latent dimension of 1,024 (same as MAESTRO) and train until convergence or max 300 epochs (same as MAESTRO). We report metrics on the held-out test set.

F.4 Ablation implementation

Unless otherwise denoted, all models in the ablation study use the same training parameters as the above section for MAESTRO pre-training. As above, we use the same 80/20 train/test split for our benchmarking studies.

Random Masking We begin with the most simple model possible. We use only the masked autoencoding module of MAESTRO. We randomly subset each sample to 10,000 cells as we would with the Set Transformer. We randomly mask out elements of the set using a 50% masking rate. We encode and decode each sample as described above in MAESTRO pre-training. After pre-training we derive representations for each sample using the pre-trained model. Using these representations, we use the 80% training set from pre-training to also train a logistic regression model to predict our 14 diagnoses. We report metrics using the held-out test set.

Multi-rate Masking We hypothesize that better representations can be learned using various masking rates, forcing the model to reconstruct missing cells using different amounts of information. Again, we randomly subset each sample to 10,000 cells as we would with the Set Transformer. We randomly mask out elements of the set at rates of 20%, 40%, 60%, and 80%. These values are arbitrarily chosen to cover the spectrum of 0% to 100%. We encode and decode each sample as described above in MAESTRO pre-training. After pre-training we derive representations for each sample using the pre-trained model. Using these representations, we use the 80% training set from pre-training to also train a logistic regression model to predict our 14 diagnoses. We report metrics using the held-out test set.

Block Masking We hypothesize that our model is able to reconstruct the masked cells because they are randomly masked, and it's able to learn the reundancy of cells in an entire set. We implement NRBM to prevent our model from learning these relationships. Using the same implementation as the above multi-rate masking, we swap out random masking for NRBM. We encode and decode each sample as described above in MAESTRO pre-training. After pre-training we derive representations for each sample using the pre-trained model. Using these representations, we use the 80% training set from pre-training to also train a logistic regression model to predict our 14 diagnoses. We report metrics using the held-out test set.

Self-Distillation From the above we see minimal, although, improving performances for each added module. Here, we introduce our online tokenizer via self-distillation which allows for the encoding of the full set of cells. This is our full model and details on implementation can be found in the above section on MAESTRO pre-training.

without Random Masking Finally, to test that the self-distillation method itself is not what allows for rich representations in MAESTRO, we test performance using only the self-distillation module. We encode sets using a student teacher model and align the latents of the full set and subset of cells. Notably, there is a large decrease in performance when we remove the masked modelling section.

F.5 Cell-type proportion retrieval

For each sample, 16 cell-type proportions were derived using manual gating, the gold-standard in immunology. We use these 16 cell-type proportions as the target for retrieval for each representation learning method. Since manual gating was used to derive these values, it is not benchmarked against in this section.

MAESTRO We pre-train MAESTRO using the above setting. After pre-training we derive each of the latent representations of each of our samples. Using the same 80/20 train/test split we train a one-layer neural network to simultaneously predict all 16 cell-type proportions. This is important because each value depends on the others. To demonstrate that this information is already encoded into the latent representation we only train for three epochs.

K-means K-means clustering is an alternative method to identifying cell-types, but is not considered the gold-standard like manual gating is. Because of this, k-means clustering serves as a strong baseline for other methods to outperform. Although we use k values of 512 and 1024 (well above the expected number of cell-types), we still expect this representation to be predictive of cell-type proportions. Using the same 80/20 train/test split we train a one-layer neural network to simultaneously predict all 16 cell-type proportions. This is important because each value depends on the others. To test whether information is already encoded into the latent representation we only train for three epochs.

OTKE We derive OTKE sample latent representations as described above. Using the same 80/20 train/test split we train a one-layer neural network to simultaneously predict all 16 cell-type proportions. This is important because each value depends on the others. To test whether this information is already encoded into the latent representation we only train for three epochs.

Deep Sets Deep Sets is supervised and therefore we do not use the one-layer neural network as described above. Instead, we use the default Deep Sets model with the 16 cell-type proportions as the target values. For each sample we subset their matrix randomly to 10,000 cells due to computational constraints. Using default parameters we train the model to predict 16 cell-type proportions. As above, because the information is inherently within the input we only use three epochs to train the model to test whether it is able to encode the 16 cell-types.

Set Transformer Set Transformer is supervised and therefore we do not use the one-layer neural network as described above. Instead, we use the Set Transformer encoder as described above, with the 16 cell-type proportions as the target values. For each sample we subset their matrix randomly to 10,000 cells due to computational constraints. Using default parameters we train the model to predict 16 cell-type proportions. As above, because the information is inherently within the input we only use three epochs to train the model to test whether it is able to encode the 16 cell-types.

F.6 LIMITATIONS

MAESTRO faces several limitations that span computational, methodological, and interpretive challenges. The model has high computational demands, including long training times and reliance on advanced hardware such as multiple GPUs with high memory bandwidth, make it inaccessible to research groups with limited resources. MAESTRO requires single-sample batch processing due to variable input sizes limits optimization strategies, resulting in longer convergence times. Masking and reconstruction strategies like Non-Random Block Masking (NRBM) depend on specific assumptions about data redundancy, which may not generalize well to diverse datasets. The model's focus on global representations risks underrepresenting rare cell populations and outliers, which are biologically significant in many studies. Interpretability is another major concern, as the latent representations encode diagnostically relevant information but lack transparency in how specific features are prioritized, complicating biological insights. MAESTRO's dependency on fixed protein panels restricts its application to datasets with variable panels, while its computational intensity renders it unsuitable for real-time analysis. The absence of multi-modal data integration limits its adaptability to more complex biological studies involving diverse data types. Additionally, its reliance on manual gating for validation—a labor-intensive and operator-dependent process—hinders scalability for larger datasets. These limitations collectively underscore the need for further refinement of the model, increased validation across heterogeneous datasets, and improved accessibility and interpretability to expand its utility in biomedical research.