UVAE: Integration of Heterogeneous Unpaired Data with Imbalanced Classes

Mike Phuycharoen[1*], Verena Kaestele[2], Thomas Williams[2], Lijing Lin[1], Tracy Hussell[2], John Grainger[2] and Magnus Rattray[1*]

[1]Division of Informatics, Imaging and Data Sciences, Faculty of Biology, Medicine and Health, The University of Manchester, United Kingdom.
[2]Lydia Becker Institute of Immunology and Inflammation, Division of Infection, Immunity and Respiratory Medicine, Faculty of Biology, Medicine and Health, The University of Manchester, United Kingdom.

*Corresponding author(s). E-mail(s): mike.phuycharoen@manchester.ac.uk; magnus.rattray@manchester.ac.uk;

Abstract

We introduce the Unbiasing Variational Autoencoder (UVAE), a novel computational framework developed for the integration of unpaired biomedical data streams, with a particular focus on clinical flow cytometry. UVAE effectively addresses the challenges of batch effect correction and data alignment by training a semi-supervised model on partially labeled datasets. This approach enables the simultaneous normalisation and integration of diverse data within a shared latent space. The framework is implemented in Python with a descriptive interface for the specification and incorporation of multiple, partially overlapping data series. UVAE employs a probabilistic model for batch effect normalisation, with a generative capacity for unbiased data reconstruction and inference from heterogeneous samples. Its training process strategically balances class contents during various stages, ensuring accurate representation in statistical analyses. The model's convergence is achieved through a stable, non-adversarial training mechanism, complemented by an automated selection of hyper-parameters via Bayesian

optimization. We quantitatively validate the performance of UVAE's constituent components and consequently apply it to the real problem of integrating heterogeneous clinical flow cytometry data collected from COVID-19 patients. We show that the alignment process enhances the statistical signal of cell types associated with severity and enables clustering of subpopulations without the impediment of batch effects. Finally, we demonstrate that homogeneous data generated by UVAE can be used to improve the performance of longitudinal regression for predicting peak disease severity from temporal patient samples.

Keywords: unpaired data integration, batch effects, longitudinal samples, variational autoencoder, deep learning, flow cytometry, clinical COVID-19.

Availability: Framework is available at https://github.com/mikephn/UVAE. Benchmarking and clinical data with processing scripts will be made available upon completing peer review.

1 Introduction

High-throughput technologies are generating an abundance of large-scale multi-modal data, offering unprecedented opportunities for the identification of disease-correlated biomarkers. However, leveraging this data for clinical research remains challenging due to difficulties in securing large, homogeneous samples, often resulting in unpaired datasets with disparate readouts. The presence of batch effects, which stem from factors like inter-laboratory variability, reagent inconsistency, and equipment calibration alterations, further adds difficulty to integration by introducing unknown shifts in data distributions, obfuscating the signal of interest under the influence of these confounding variables.

Common data processing strategies such as normalisation, imputation, or data removal, frequently fail to adequately address these challenges, and can potentially introduce further inaccuracy by enforcing incorrect assumptions about the distribution of modelled data. Unknown differences in class contents across batches, such as varying cell-type proportions, are a major challenge, often causing distortions in attempts to align data distributions. Some methods tackle this problem by aligning the data separately within known populations [1] or by iteratively re-weighing the merging function to balance clusters as the integration progresses [2].

Methods for batch effect correction [3] and integration of unpaired data modalities [4] received much recent attention, particularly in context of single cell data. These methods often involve variational autoencoders (VAEs, for details see e.g. [5]) to create latent embeddings by encoding each data stream into a prescribed space with chosen dimensionality and prior distribution. However, this process often requires known mappings of correlated features, making it challenging to integrate unpaired datasets with limited shared markers. For

example, in a multi-panel flow cytometry experiment it is desirable to keep the marker redundancy between panels to a minimum in order to maximise the novel information obtained from each processing batch [6, 7].

In this study, we introduce a VAE-based method capable of integrating and normalising unpaired, partially annotated data streams, thus addressing these challenges. Our model separates the confounding factor variability from the shared latent space, effectively transforming heterogeneous datasets into a unified, homogeneous data stream, while performing simulatenous normalisation, merging, and class inference using stable non-adversarial learning objectives. We provide an easy to use, descriptive python framework, which allows for simultaneous inclusion of an arbitrary number of data streams, with or without supervised annotations for inter-operable normalisation, merging, class prediction, and data generation.

We initially demonstrate the efficacy of our approach using synthetic data, illustrating that simultaneous data integration and batch effect correction, along with the implementation of a class-balancing strategy, result in superior imputation performance and more accurate latent space alignment. Subsequently, we apply our framework to real-world COVID-19 flow cytometry patient data from four hospitals in the Manchester region. The integration results in a robust statistical signal pertaining to cell-type counts and immune state marker levels, showing correlation with disease severity. Additionally, we validate predictive performance when the generated unbiased data is employed for longitudinal regression of patients' peak severity. This work, therefore, presents a significant advancement in the integration and normalisation of unpaired biomedical datasets.

2 Results

2.1 Challenges and solution for integrating clinical flow cytometry data

The development of our method was shaped by challenges encountered during the combined analysis of clinical blood samples from patients admitted to four hospitals in the Manchester region due to COVID-19. The samples were processed using flow cytometry, a technique which operates by introducing protein markers to a sample and assessing their light response across a range of frequencies. Through the analysis of the resulting fluorescence, it is possible to determine the relative abundance of proteins for each cell in the sample, giving insights into the cell type and activity. Such information enables the identification of enriched cell types or shifts in marker levels that correlate with clinical outcomes.

During data collection, we confronted a number of challenges impacting data quality. Given the technical constraints of flow cytometry, where only a select number of markers can be introduced concurrently without interference, we adopted a multi-panel experimental design. Each experiment captures a

subset of the desired markers, thereby producing unpaired data with varying features across different data streams. Further complications emerged due to the extended duration of data collection, diverse sample sources, many processing batches, equipment calibration shifts, and changes in accessible reagents (causing further variances in the acquired markers between batches). Consequently, we obtained a heterogeneous dataset with missing features and many potential batch effects. Conventional flow data analysis methods would necessitate discarding numerous samples lacking certain markers and would demand significant manual effort for precise cell-type population labeling. In the absence of such detailed labeling, sample normalisation risks the introduction of supplementary batch effects given the variable cell-type proportions across batches.

To overcome these challenges, we created a versatile deep learning framework that facilitates data integration while simultaneously rectifying batch effects and imputing features. Prior to deploying our method on real clinical data, we performed comprehensive experiments, which revealed its superior efficacy compared to integration solely reliant on aligning distribution densities across batches, or matching features. The key innovation of our method is that it does not assume that the ground-truth manifolds match between the integrated data series, allowing for automatic integration of diverse blood samples, containing unknown proportions of cell-types.

The workflow our approach is illustrated in Fig. 1. For each unique marker set included for integration, we procure class and doublet labeling wherever available. Subsequently, we train a model encompassing multiple VAEs, which concurrently learn class inference, batch effect correction, and panel integration. By balancing the cell-type proportions in samples used for merging, we safeguard against data over-alignment. Upon completion of the integration, we predict a uniform marker set for each cell, generating a dataset that is more conducive to clustering and statistical evaluation, with more statistical power in highlighting cell-types and markers significantly correlated with disease severity.

Finally, to further demonstrate our method's applicability, we benchmark a range of models to predict peak COVID severity from longitudinal patient samples, using our imputed data as input. We demonstrate that these models hold potential to more precisely track disease severity, additionally offering insights into the predictive ability of the incorporated markers.

2.2 UVAE for integration of disjoint data

The Unbiasing Variational Autoencoder (UVAE) offers a robust solution for the integration of unpaired data streams. It recognises that batch effect correction is crucial to establish an unbiased, common frame of reference for merging disjoint data. The precision of such correction can be influenced by differences in the true underyling distribution of cells between samples (e.g. different numbers of each cell-type) in the data, which is typically not fully known before integration. A semi-supervised classifier, trained on a partially labelled dataset,

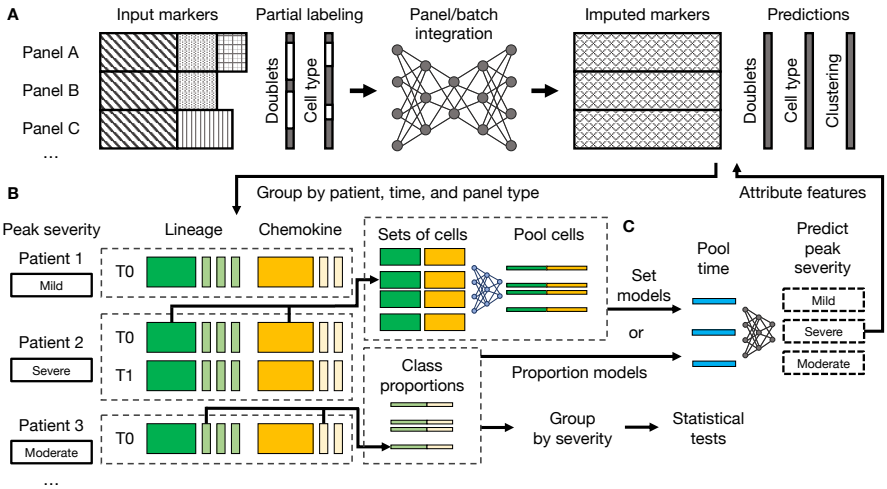

Fig. 1 A) Partially labeled flow cytometry data originating from several panels with partially shared markers is input to a deep learning algorithm performing batch effect correction and panel integration. Entire dataset is regenerated with a uniform marker set and full class predictions, and optionally clustered. B) Cell representations are split into lineage and chemokine panel types, filtered and grouped by patient-timepoint. Proportions of cell-types, clusters, and corrected marker values used to calculate enrichment by peak observed clinical severity. C) Regression models predicting peak severity are trained using either class proportions, or raw marker representations of sets of cells as input, with variable number of timepoints per patient. Gradient attribution is used to identify features which contribute to increase in predicted severity.

can extrapolate the partially known cell-type labels. However, the classifier's generalisation ability across all data is contingent on the alignment of the latent space in which it operates. This cyclical dependency underpins the necessity for simultaneous inference of model parameters, applying progressively improved class inference to latent normalisation and merging.

We have developed an automated Python framework allowing for the specification of multiple, partially overlapping data and labeling series, along with merging and normalisation constraints. A generative model is constructed to incorporate all specified constraints. For each data series with a distinct feature set, a dedicated VAE is created, each sharing the same latent space dimensionality. The latent embeddings are normalised using the specified batch labellings, forming the input to shared classification or regression layers. The probabilistic configuration applies a Gaussian prior to the latent space embedding, learned from data through automatically learned non-linear functions. This facilitates a flexible, non-linear normalisation of batch effects. Furthermore, probabilistic sampling enables the model to capture uncertainty in the learned embedding, bolstering its generative capability for creating unbiased data and inferring from previously unseen samples.

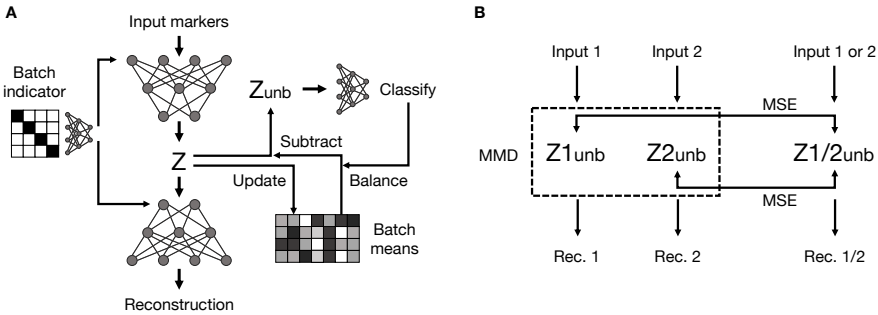

Fig. 2 A) Schematic of a single VAE trained on one set of input markers, encoding input data into latent space Z, and decoding back into data space. Batch effect correction is performed by 1. conditioning encoder and decoder with batch identifier; and 2. offsetting latent embedding to common mean to produce unbiased embedding Z_{unb}. Shared classifiers are trained on unbiased embeddings. Class predictions are used at the end of each epoch to balance the composition of samples for calculating batch mean offsets. B) Data integration applied to two input streams 1 and 2, with a subset of shared features shown as 'Input 1 or 2'. Separate VAEs are trained for streams with distinct feature sets. Additional VAE is trained using a subset of shared features from both streams. Merging is performed by minimising mean squared-error (MSE) loss between pairs of unbiased sample embeddings of each stream encoder, and the unbiased embedding of shared encoder. Additional merging is performed by minimizing MMD loss calculated on class-balanced samples using unbiased embeddings of the two stream encoders.

As training progresses, classifiers predict and balance the class contents of samples utilised for calculating various alignment scores, discussed in subsequent sections. Particularly during latent normalisation (Section 2.3), density matching for merging (Section 2.4), and alignment calculation for hyper-parameter optimisation (Section 2.7.1), class predictions are employed to compute the average class proportions across groups (e.g., batches or panels). This process aids in increasing or decreasing the number of class samples representing each group when determining distribution statistics. Resampling is gradually introduced via a tunable hyper-parameter, facilitating initial convergence and preventing distortion from an untrained model.

Notably, despite potential complexity in constraint combinations, all model elements automatically inter-operate and converge to a common optimisation target, leading to stable, non-adversarial training. Hyper-parameters, including model size, merging and normalisation speed, and learning rates are automatically selected via Bayesian optimisation for optimal performance using various metrics, including arbitrary user-specified alignment scores. The final model can regenerate all data in an unbiased form, including all desired features, as if originating from a single, averaged source. The model serves future applications by allowing the alignment of new data and interpreting it within the existing latent space.

2.3 Batch effect correction

Batch effect correction can be viewed as a special case of integration, where the feature set is identical for all data, allowing for a singular recognition model input. Normalisation is often applied directly to the data, to homogenise it between batches with operations such as mean-centering, scaling of variance, quantile or spline normalisation [8]. In the context of VAEs, normalisation can occur in the latent space, ensuring uniform decoding in the style of a chosen batch [9]. This approach, also referred to as latent arithmetic, involves calculating batch offsets of explicitly drawn sample distributions and may thus be sensitive to underlying class imbalances between batches.

An alternative methodology employs conditional autoencoding, where the batch indicator covariate is deliberately detached from the input or latent space and used to condition the encoder or decoder. Coupled with a factorised Gaussian prior on the latent space, this discourages the latent representation from redundantly capturing batch-specific information and can exhibit less sensitivity to class imbalances, as explicit sample offsetting is not performed. However, it has been observed that the learned representation may still capture information correlated with the batch [10]. To mitigate this, additional loss terms, including a kernel distance metric like MMD between batches, can be introduced to enhance the latent similarity of their distributions. Unlike adversarial learning, MMD utilises a predefined similarity kernel and does not have trainable parameters, contributing to training stability.

UVAE allows for batch effect correction to be performed within each VAE, between user-annotated sample groups. Typically, the aim is to eliminate variability between processing batches within each cytometry panel, but numerous unwanted covariates can be concurrently normalised. This correction process employs two simultaneous mechanisms, as illustrated in Fig. 2A.

Firstly, each VAE supports encoder and decoder conditioning with the annotation of covariates, which are to be separated from the shared space. A categorical variable, supplied for each sample, is projected through an embedding layer and input to the decoder, and optionally the encoder, alongside the embedding or data input, respectively. Combined with the factorised Gaussian prior on the latent space and other specified merging constraints, this approach discourages the shared embedding space from redundantly capturing unwanted variability.

Secondly, latent arithmetic can be applied to the remaining part of the latent space (excluding nuisance covariate indicators) to normalise embeddings between categories. Arithmetic means are calculated using class-balanced samples for each category, with samples subsequently offset to match the category mean to either a common mean of all categories or a specified target category. To prevent sudden changes to the embedding, this offsetting is gradually introduced during training, weighted by an ease-in variable. The recalculation of latent offsets is performed at fixed intervals, using classifier prediction to balance the class proportions of samples across the normalised groups. The resulting corrected sample embeddings can be used to interpret the data in an

unbiased space, and serve as inputs for subsequent tasks such as classifiers or merging constraints.

2.4 Integration of unpaired data streams

In more general integration tasks, data series often have disparate channels and therefore exist in distinct data spaces. If key features are consistent across all data, one could consider eliminating the uncommon ones. This, however, reduces the available information, especially as the number of included streams increases, and shared features decrease. Previous studies have proposed zero-padding the missing values for neural network model input [11], though this approach becomes increasingly inefficient as the number of rarely-used features rises. Another alternative is to impute the missing entries across panels. For instance, a regression model could be trained on data where two sets of features are present, predicting one set from the other [12]. Similarly, a nearest neighbour approach could be used for missing value imputation [6]. However, these methods are susceptible to batch effects and other domain shifts between the small training subset and the new batches for which predictions are made.

Experiments can be specifically designed to yield feature sets that are conducive to imputation. In the case of flow or mass cytometry panels, due to the limited number of protein markers that can be read simultaneously, panel design involves selecting sets of desirable markers and combining them with several shared markers that are consistent across all experiments [6, 7]. These shared variables can then serve as reference distributions for comparing readouts obtained from different panels. However, it is beneficial to keep the number of shared markers to a minimum to reduce redundancy and maximize information per experiment. A recent solution for integrating cytometry data with marker overlap was proposed in cyCombine [1], which performs batch effect correction (using ComBat [13]) within cell clusters obtained with FlowSOM [14]. Panel merging is implemented by probabilistically drawing missing marker values from the same cluster in another panel. This method requires that disjoint data streams can be linked through shared markers, and that the batch effects within the shared markers are small enough to not significantly affect the joint clustering.

A multi-modal design including VAEs offers several advantages. Firstly, corrections can be performed in a latent space, a result of a non-linear data transformation, thus allowing for correction of non-linear confounders. Secondly, multiple latent spaces created for each data modality can be merged during training through a combination of loss constraints [15]. This may include a predefined mapping of correlated features shared between modalities and density-matching methods, such as adversarial training [2], which do not require knowing correlations between features.

UVAE allows for the integration of unpaired data series using both explicit data linking through shared features and density matching of arbitrary data streams. The alignment is performed in conjunction with any specified batch

correction and uses class-balanced samples to ensure that the aligned distributions represent proportional amounts of underlying data. In our framework, merging constraints containing the desired labeling of groups to be merged or features to be linked are user-specified over subsets of data. These constraints are used to define optimization objectives that increase alignment. During training, the alignment metrics are back-propagated through VAE encoders and trained to converge simultaneously with other losses, resulting in an aligned latent space. Merging by density matching and alignment through shared features is illustrated in Fig. 2B.

To align embeddings of multiple VAEs that share a subset of similar features, an additional VAE is added during training, which accepts the shared features as its input. Data from all the covered series is subset to the shared columns, and input to the shared VAE. Subsequently, distances are minimised between the embeddings of samples obtained by each series encoder and the embeddings from the shared encoder. Since distances are calculated on paired embeddings for each sample, this process is not susceptible to class imbalance between groups. However, it can be sensitive to batch effects within groups. Therefore, the distance is minimised for the unbiased embeddings, obtained after applying batch correction in all VAEs, including the additional shared channel VAE. The shared VAE applies batch effect correction to the combined batches from all the data series it connects.

To align embeddings without relying on shared features, we apply merging by density-matching, which minimizes the MMD between unbiased latent embeddings of data from each series. We calculate MMD with a multi-scale Gaussian kernel [16]. If more than two conditions are present, pairs are selected randomly in each training mini-batch. The samples used for calculating MMD are class-balanced during training to ensure the same class proportion is present as a representation of each integrated group of data. This type of merging does not require any feature overlap between the aligned data series, so it can be used to merge different data modalities. However, if desired, MMD can also be applied for batch effect correction within a single VAE, to increase similarity of batches in the latent space.

2.5 Generation of homogeneous data

The integrated latent space provides a common frame of reference, enabling data interpretation across different series. This is achieved by utilising the encoder specific to each series to embed the data and apply batch effect correction. However, the latent space is a dense, abstract representation with undefined feature meanings. To facilitate subsequent, interpretable analysis, we leverage the generative capability of UVAE to reconstruct all data back into the data space, where features carry biologically relevant interpretations, such as protein marker levels. This process, referred to as cross-panel translation, imputes all missing markers and predicts classes for all samples, resulting in a homogeneous and interpretable dataset (refer to Fig. 1A). This uniformity

enhances the power of statistical analysis and permits joint clustering of the data, avoiding clusters that segregate by panel or batch.

Because data representation outside of the latent space is batch-specific, a target batch or category must be selected for each nuisance covariate when generating data. All data is then generated in this chosen style. Specifically, when using conditioning and latent normalisation simultaneously, samples are first embedded into the latent space. This embedding process incorporates conditioning of the source batch provided to the encoder and subtraction of the source batch latent offset. The latent representation is subsequently decoded after adding the target batch offset and conditioning the decoder with the target batch indicator.

To enhance homogeneity of the obtained marker levels, we regenerate all samples using a fixed set of multiple decoders, trained on the largest data series. The data is embedded into the latent space using series-specific encoders, and then decoded using a chosen set of decoders, each with fixed target covariates. For features present in multiple series, we average the generated values. Consequently, UVAE provides an effective means of transforming heterogeneous, unpaired data streams into a coherent, interpretable dataset.

2.6 Model selection with external alignment metrics

After specifying the modelled constraints, it is critical to select appropriate hyperparameters, such as model size, latent space dimensionality, and learning rates. Our model provides flexibility in adjusting several hyperparameters that can rapidly enhance alignment (e.g., by weighting the contribution of MMD to the loss, or by manipulating the mutual frequencies of training among added constraints). However, these adjustments can potentially compromise the accuracy of reconstruction by limiting adequate training of the decoders. Consequently, in addition to conventional validation losses, including reconstruction and regularization, we incorporate explicit alignment metrics in our hyperparameter search, executed via the Mango [17] Bayesian optimizer. This approach enables automatic determination of optimal model hyperparameters that balance reconstruction, supervised predictions, and alignment.

Our optimization procedure allows arbitrary, user-defined scoring metrics for inclusion in model selection. In response to the NeurIPS 2021 single-cell integration challenge [4], the authors recommended the iLISI metric [18] as the most robust measure for batch integration. Therefore, we employ LISI to measure batch and cell-type alignment in our optimization process, with the goal of achieving high batch LISI scores (indicating effective batch mixing) and low cell-type or cluster LISI scores (indicating minimal cell-type mixing). Alternatively, for comparison with cyCombine [1], we optimize two analogous metrics used in their study: Earth Mover's Distance (EMD) between data batches, and Mean Absolute Deviation (MAD) within each channel and batch. A high EMD value between data streams before and after integration is desired, while a low MAD score is preferred to avoid oversimplification of the data distribution.

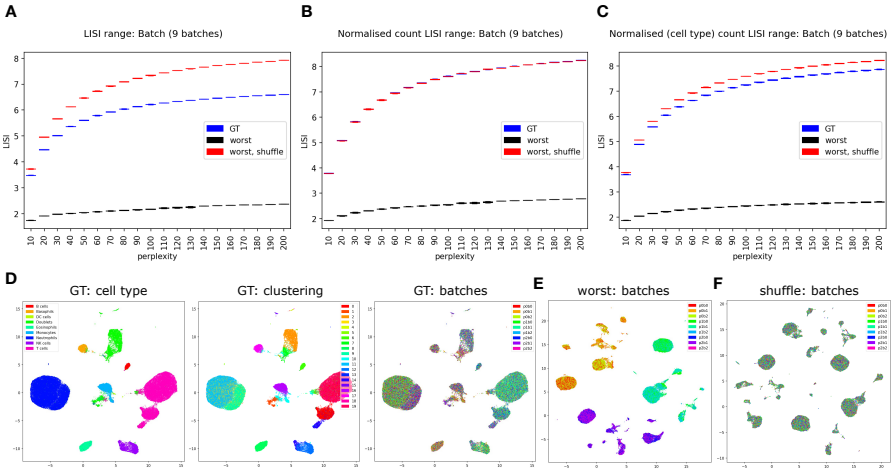

Fig. 3 A-C) LISI scores on Whole Blood synthetic dataset calculated for different perplexities using ground truth model (GT), worst case model of maximum separation (worst), and worst case model of maximum over-mixing (worst, shuffle). Medians across all cells were calculated over 10 repeated model trainings for each configuration. A) Unnormalised LISI scores calculated without accounting for class imbalance between batches. B) Normalised LISI scores calculated by down-sampling each ground truth (GT) cluster in each batch to the lowest count across all batches. C) Normalised LISI scores calculated by down-sampling each manually annotated cell-type in each batch to the lowest count across all batches. D) UMAP of ground-truth data embedded with VAE, showing correct cell-type, clustering, and batch alignment. E) Batch alignment in expected worst-case of over-separation. F) Batch alignment in expected worst case of over-mixing.

2.6.1 Normalising LISI alignment scores for imbalanced data

The LISI score is a measure of class mixing within an embedded neighbourhood of cells, averaging the number of sampling draws until the same class is drawn again [18]. The theoretical range of the LISI score is from 1 (indicating maximum separation) to the total number of unique classes used in the calculation, with higher values indicating increased mixing. Neighbourhood probabilities are evaluated through a Gaussian kernel to emphasise local mixing, with the effective size governed by a perplexity parameter, with higher perplexity leading to increased computational cost.

This scoring approach, however, encounters two primary issues: firstly, LISI tends to yield maximum scores for uniformly mixed data, which is an unrealistic scenario; secondly, the actual range of LISI scores typically does not span the full theoretical range. To address these challenges, we conducted experiments to determine how to effectively use LISI as part of the model selection criteria.

In order to determine the empirical range of scores, we train worst-case models, representing maximum separation and maximum mixing. The maximum-separation model is trained on disjoint data with batch effects, without applying any form of correction or merging. Subsequently, the maximum

mixing model is obtained from this worst-case embedding by randomly permuting the locations for all samples. The scores derived from these models show the limits of expected range of LISI on a given dataset. Additionally, we train a ground-truth (GT) model using perfectly aligned synthetic data which does not contain any artificially introduced batch effects (detailed in 4.8).

We observe that, in practice, LISI scores tend to fall within a narrower range than their theoretical maximum, as illustrated in Fig. 3. This range is influenced by several factors, including the uniformity of the batch and class distribution across the dataset, the overall sample density, and the choice of perplexity. The evaluation of the GT reference reveals that higher LISI scores can erroneously be interpreted as superior in non-uniform datasets. The shuffled model, for instance, obtains a higher batch mixing score than the GT model, despite incorrect alignment (Fig. 3A).

This discrepancy can be mitigated by normalising the sample on which the score is calculated, ensuring an equal number of classes in each tested batch. We down-sample each class to the lowest count present across all batches. If we knew the clusters causing the imbalance we could use them for balancing and erase the undesirable advantage of shuffled models (Fig. 3B). In reality, we likely do not have access to the exact class assignment causing the imbalance. Even so, normalising using cell-type annotation (which, in our test, is coarser than the GT clustering) still results in a narrower gap between the GT and shuffled models than without normalisation (Fig. 3C). For this reason, we adopt the class normalisation approach in our model selection for any LISI metric calculated involving classes present across all batches.

2.7 Performance benchmarks on synthetic data

Benchmarking batch effect correction and alignment methods is often difficult due to the absence of ground-truth alignment data. To address this, we simulate a typical flow cytometry integration problem using a synthetic dataset based on a standardised Whole Blood blood sample from a healthy individual, processed through a flow cytometry protocol. The sample contains 15 cell-type markers and 6 features related to size and complexity, totalling 21 real-valued channels.

We manually annotated this sample for 8 major cell types and doublets, removed debris, and selected a random subset of 100,000 cells. We cluster this data into 20 groups using a Gaussian Mixture Model (GMM). This cluster annotation, more detailed than the manual cell-type annotation, serves as the basis for splitting the dataset unevenly to simulate batch effects. We then divide the dataset into 3 panels, each containing 3 batches. Each split assigns an uneven proportion of each GMM cluster to each batch or panel. Moreover, we remove one marker channel from each panel to obtain incompatible feature sets. Finally, we apply batch standardisation to the data, which causes misalignment in the dataset due to cluster imbalance. Notably, due to underlying population imbalance and missing channels, the correct ground-truth alignment cannot be restored through simple normalisation or concatenation

procedures. Thus, an effective integration method should acknowledge non-uniform density across batches and utilise this density to perform alignment while compensating for information loss from missing channels.

2.7.1 Optimisation strategy and evaluation metrics

In testing various model configurations, we employ fixed model size and computational constraints. For all configurations, we maintain the same latent dimensionality, layer and neuron count, and total epoch and sample count per epoch. As a ground-truth baseline, we train a VAE on the unaltered data, while the worst-case models are trained on data after introducing batch effects, resulting in two extreme cases of batch separation and mixing.

Our model selection process optimises three loss components: batch alignment (iLISI) and category separation (cLISI) scores, and a training/validation loss component reflecting reconstruction, regularisation, and classification performance. We establish the expected range of LISI scores for each metric using the worst-case models. Each LISI component is then clipped and scaled to fit this range and lie between 0 and 1. We employ a multiplier of 3 on the validation loss to allow its contribution to be similar relative to LISI components.

For optimisation, we use Mango over 30 iterations to train each configuration over 50 epochs, with 100,000 samples per epoch. We keep the majority of hyper-parameters fixed across all configurations, while optimising learning rates for unsupervised, supervised, and merge losses, frequencies and pull rates for each merging constraint, and the ease-in rate for normalisation and resampling. This approach constrains the size of the optimisation space while maintaining a high degree of freedom for training speed of all model constraints.

We evaluate each model configuration using 10 models trained with the best hyper-parameter settings. We calculate three metrics across the model instances and determine the median score:

- Normalised LISI batch score: A higher score indicates increased batch mixing.
- Normalised LISI GT cluster score: A lower score indicates improved cluster separation.
- Imputation error: the mean of channel mean-squared errors, comparing ground-truth marker values (standardised) to imputed values (also standardised), with values reconstructed from the remaining panel decoders.

These metrics utilise data inaccessible during model selection and training: the ground-truth (GT) clustering and pre-normalised marker input values. Two LISI metrics are computed on the trained latent embedding, normalised by down-sampling cells to match the lowest cell number from each batch in each GT cluster.

2.7.2 Combination of latent space normalisation, panel merging, and resampling results in most accurate alignment

Table 1 Best performing combinations of batch correction and merging constraints, with or without resampling on Whole Blood synthetic dataset

Configuration[1]					Metrics						
					Overall	MSE imputed		iLISI		cLISI	
Conditioning[2]	Latent norm.	MMD[3]	Subspace	Resampling[4]	rank	median	rank	median	rank	median	rank
✓	✓	✓	✓	✓	**1**	.29	**3**	6.37	**1**	1.27	**1**
	✓	✓	✓	✓	**2**	.29	**2**	6.26	**2**	1.28	**2**
	✓		✓	✓	**3**	.29	**4**	5.7	**4**	1.29	**4**
		✓	✓	✓	**4**	.32	7	5.74	**3**	1.29	**5**
✓	✓		✓	✓	**5**	.28	**1**	4.69	17	1.28	**3**
✓		✓	✓	✓	6	.32	8	5.54	6	1.3	9
	✓	✓	✓		7	.33	10	5.46	8	1.3	7
			✓		8	.32	6	5.26	12	1.31	12
✓		✓	✓		9	.33	11	5.19	13	1.3	8
	✓		✓		10	.33	9	5.3	11	1.31	13
✓	✓	✓	✓		11	.35	14	4.88	14	1.3	6
		✓	✓		12	.35	12	5.46	7	1.32	18
		✓		✓	13	.37	16	5.45	9	1.31	14
✓			✓		14	.3	**5**	3.95	21	1.31	15
	✓	✓		✓	15	.37	17	4.75	16	1.3	10
✓		✓		✓	16	.36	15	5.35	10	1.34	20
✓	✓	✓		✓	17	.39	19	5.61	**5**	1.41	22
	✓	✓			18	.39	18	4.36	19	1.3	11
		✓			19	.44	22	4.87	15	1.32	17
✓	✓		✓		20	.35	13	3.11	22	1.35	21
✓	✓	✓			21	.4	20	4.35	20	1.31	16
✓		✓			22	.41	21	4.68	18	1.33	19

[1]Shared settings: VAE (50 dim, 2 hidden layers, 256 neurons each), supervised LISI target, classifier fitted. [2]Conditioning both encoder and decoder. [3]MMD between panels only. [4]Resampling normalisation/MMD using predicted cell-type.

In this test, we trained a series of models including various combinations of constraints associated with batch correction (conditional autoencoding, latent space normalisation), and panel merging (shared subspace merging, MMD). For configurations incorporating at least one constraint potentially sensitive to class imbalance (MMD or latent normalisation), we trained models with and without resampling. The resampling approach balanced the samples for the sensitive constraints, equalising predicted cell-type counts across batches to the mean count of all batches. All configurations included a classifier, trained to predict cell-types from a subset of manually annotated labels. In order to

maintain consistency, the models were otherwise identical and their hyperparameters were optimised for the supervised LISI score (see Methods 4.6). If conditioning was incorporated, we applied it to both the encoder and decoder, following the Conditional Variational Autoencoder (CVAE) model [19]. If MMD was employed, it was applied between panels. The results are shown in Table 1. Additional results using unsupervised LISI optimisation target are shown in Supplementary Table 6.

We observe that the inclusion of resampling with MMD and latent normalisation consistently increases model performance, even though the cell-type prediction only approximates the ground truth (GT) cluster split that initially caused the class imbalance. Remarkably, the optimal performing configuration includes two types of batch effect correction and two types of panel merging, applied concurrently. When comparing batch correction constraints, the normalisation of the latent space to a common mean between batches holds more significance in our experiments than encoder and decoder conditioning. For merging the data streams, subspace merging using known features is present in all best performing configurations, and appears to be a more important component than MMD minimised between panels. We observe a trade-off between merging and reconstruction accuracy. Although the best alignment is obtained by a combination of all constraints, the best imputation performance is achieved by a model excluding MMD merging.

These findings suggest that it is advantageous to build models utilising a combination of multiple merging and batch effect correction components, provided feature and covariate assignments are available. Hyperparameter optimisation can then be leveraged to automatically calibrate their mutual contributions to enhance model performance.

2.7.3 Encoder and decoder conditioning mitigates over-alignment

The original CVAE appends a one-hot conditioning variable to both the encoder's input (data vector) and the decoder's input (latent embedding vector). However, certain methods forego encoder conditioning [2]. We speculate that in the absence of encoder conditioning, increased batch similarity of the latent embedding may be achievable in the event of many-to-one mapping (where multiple disjoint batches map to the same distribution in the latent space). However, it may not be possible to differentiate between various types of samples in the case of one-to-many mapping. For instance, if batch effects offset the values for a particular cell subtype, this subtype may be represented by a vector identical to another subtype in a different batch. Without providing additional batch information, the encoder has no information to distinguish between these subtypes, mapping them to identical locations in the latent space and resulting in sub-optimal alignment. Consequently, we experimented with models conditioning either both the encoder and decoder, or only the decoder.

Furthermore, we wanted to test the effects of concatenation of the one-hot conditioning variable after a linear embedding. Typically this is not required,

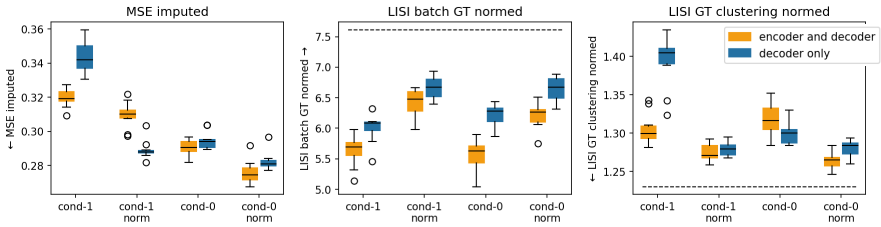

Fig. 4 Results of embedding the conditioning variable through a linear layer (cond-1) or directly concatenating (cond-0), while conditioning both encoder and decoder, or decoder only. Results with or without latent space normalisation (norm). All models shared settings: VAE (50 dim, 2 hidden layers, 256 neurons each), MMD loss between panels, subspace merging, resampling with cell-type annotation, unsupervised LISI target, classifier fitted. Dotted lines indicate scores obtained by ground-truth model trained on unaltered data.

as most models use only binary conditioning. While our test dataset has 3 conditions per panel, in the real datasets we may have tens of batches per panel, and with varying number between panels. This may make direct concatenation of such large, variable sized vector problematic. We therefore compare the results of direct concatenation, or concatenation of linear embedding, in both panel autoencoders (one-of-3 vector projected to 2 dimensions), and Subspace autoencoder (one-of-9 vector projected to 2 dimensions).

Experiments were conducted both with and without latent space normalisation, which can be applied concurrently to correct batch effects within each data stream. The results are shown in Fig. 4. Primarily, we noticed that the linear embedding of the conditioning variable can significantly degrade imputation accuracy in the absence of latent normalisation, and potentially lead to increased class mixing without encoder conditioning. Secondly, we observed that models that condition only the decoder generally exhibit higher batch mixing. However, this might come with a trade-off of undesirable class mixing and lower imputation accuracy, particularly for our overall top-performing configuration (direct concatenation of conditioning with latent normalisation included). Given that real-world data will likely necessitate the use of linear embedding of categories due to the large number of batches, we deduce that conditioning both the encoder and decoder can reduce the risk of over-alignment and loss of imputation accuracy, while posing a less substantial risk of reduced batch alignment.

2.7.4 Batch clustering can be used to prevent over-alignment of unlabelled data

Our previous analyses leveraged partial class annotation, which we employed to quantify over-alignment during optimisation and to balance classes. However, certain tasks require data alignment in the absence of known class annotations. Under such circumstances, when the presumptive batch effects are minimal, we can jointly cluster the data and use these cluster assignments as class assignments. This method relies on the existence of shared clusters, distributed

approximately evenly across batches. In more general settings, we cannot make this assumption as our aim is to allow for arbitrary misalignment of data. Joint clustering for data with significant batch effects tends to split data by batch and class, making the clustering unsuitable for class assignment, as classes are not shared between batches.

Fundamentally, our method does not require class annotation for alignment, nor does it assume that batch effects are small. However, without class balancing and a class-separation metric, we risk overly mixed embedding results. To tackle these challenges, we propose two mechanisms that utilise clustering as a stand-in for class assignment, making them suitable for aligning arbitrarily misaligned, unlabelled data.

In cases where a dataset has no known class assignments, we independently cluster each batch of data into a fixed number of clusters. We then treat these clusterings as distinct class assignments. Resampling is conducted not by balancing classes between batches (as in the supervised case), but by sampling an equal number of instances from each cluster. This approach can be applied when categorical assignments are not shared between batches. Fig. 5B-D compares the performance of this approach to training without resampling. We found that unsupervised balancing enhances imputation performance for all configurations, similar to resampling in the supervised cases. A minor decrease in batch alignment was observed, with a potential substantial increase in class separation for models utilising conditioning. We infer that in certain circumstances, our unsupervised balancing method can enhance embedding accuracy by preventing over-alignment.

With the aim to improve the stability of our approach, given the often unpredictable nature of clustering methods, we allow multiple overlapping categories to be used for balancing, with each automatically resampling a random portion of the data. For example, we cluster each batch three times, using 20, 15, and 10 clusters, then use each assignment to balance one third of the data. The results are illustrated in Supplementary Fig. 13. While we observe some increase in imputation performance for some configurations the results are variable, and we did not see consistent improvements of using multiple cluster assignments on our toy dataset.

Secondly, we introduce cluster-based unsupervised LISI metric to score alignment on datasets lacking shared class assignment. In such a scenario, we expect that the optimal class score should be roughly equivalent to the number of batches. This is because in a perfectly aligned uniform dataset, each location will feature one distinct cluster identity from each batch. Contrary to the supervised scenario, class normalisation cannot be applied here, as class identities are not shared across batches. We verify that for a ground-truth model, the number of aligned clusters approaches the number of batches at high perplexity levels (see Fig. 5A). We then test multiple configurations optimised using either the unsupervised (batch clustering) or supervised (cell-type) LISI target.

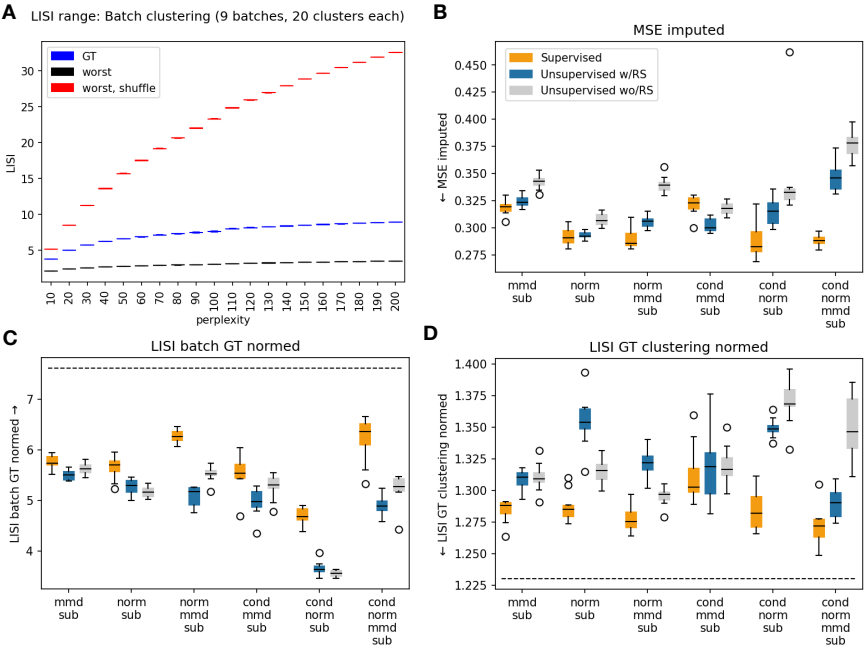

Fig. 5 A) LISI scores on Whole Blood synthetic dataset calculated using independent clustering of 9 overlapping batches to 20 clusters each. Scores were calculated for different perplexities using ground truth model (GT), worst case model of maximum separation (worst), and worst case model of maximum over-mixing (worst, shuffle). Medians across all cells were calculated over 10 repeated model trainings. B-D) Imputation and alignment comparison for supervised models (using predicted cell-type LISI optimisation target and cell-type resampling), or unsupervised models using independent clustering of each batch. Unsupervised w/RS indicates the batch clustering was used to balance latent normalisation (norm) or between-panel MMD. Unsupervised wo/RS indicates no resampling was performed. All models shared settings: VAE with 50 dim, 2 hidden layers, 256 neurons each. 'cond': encoder and decoder conditioning of batch. 'norm': latent space normalisation. 'sub': subspace merging. Dotted lines indicate scores obtained by ground-truth model trained on unaltered data. B) Imputation mean-squared errors. C) LISI scores of batch alignment normalised with ground-truth cluster split (higher is better). D) Normalised LISI scores of ground-truth cluster separation (lower is better).

We aimed to assess our unsupervised optimisation method by isolating the impact of LISI optimisation metric used to score class separation. Access to class prediction can enhance alignment through resampling, but training the classifier and backpropagating the training signal through encoders can also affect alignment. To make the results in this test consistent, we trained a classifier in both supervised and unsupervised cases, but did not use its predictions during training in either case. In the supervised case, the classifier is used after training to calculate class-normalised LISI scores. The results of this isolated test are shown in Supplementary Table 7 and Supplementary Fig. 12. We observed that using the unsupervised LISI metric does not clearly impact either imputation or alignment performance, which remains similar to

the supervised case. The fact that the optimal cluster LISI score is not equal to the lower bound (of one) and is not precisely known (see Fig. 5A) does not seem to impair performance, likely due to the balancing act between merging and separation during optimisation, which is unlikely to operate at the extremes of either score.

Lastly, we compare the results of our unsupervised approaches to the method in a supervised context. In this experiment, the supervised case fully utilises class knowledge, by optimising a normalised LISI score to judge cell-type separation, fitting a cell-type classifier, and using it during training to class-balance the sensitive constraints. This is compared to unsupervised cases, where no classifier is trained, and batch clustering is used to judge separation. Results in Fig. 5 illustrate that while supervised approaches generally perform best across imputation and alignment metrics, the unsupervised case does benefit from cluster resampling, especially in terms of imputation performance, and in the alignment of best-performing model configurations.

2.7.5 Comparison with cyCombine

Table 2 Comparison with cyCombine on Dana-Farber Cancer Institute data.

Dataset	EMD		MAD	
DFCI1	0.66	0.77	0.02	0.06
DFCI2	0.66	0.73	0.02	0.09
DFCIb3	0.86	0.79	0.01	0.09
Method	cyCombine[1]	UVAE (ours)	cyCombine[1]	UVAE (ours)

[1]Scores from the original paper.

We compared our method against cyCombine [1], a state-of-the-art approach designed for the integration of flow cytometry data. cyCombine utilises FlowSOM clustering on the data, premised on the assumption that batch effects are minimal and similar cells converge in the same cluster. It then applies batch effect normalisation using ComBat, a linear factor analysis method. cyCombine can then align different data streams from separate panels using shared channels, carrying out imputation by sampling from other panels within the same cluster and introducing noise to enhance sample diversity within each cluster.

cyCombine was evaluated by its authors on various datasets, including one sourced from the Dana-Farber Cancer Institute. This dataset consists of peripheral blood mononuclear cell (PBMC) samples from 20 healthy donors and 56 patients with chronic lymphocytic leukemia. This dataset includes two panels with unique markers; the first panel incorporates 36 markers, and the second panel consists of 34 markers, with a shared set of 15 markers. The authors included two types of experiments on this dataset: batch correction within each panel, and cross-panel integration via batch correction on shared markers from the third batch in both panels. However, given that cyCombine

exclusively utilises shared channels for cross-panel integration, all three scenarios essentially represent batch effect correction, as only the shared markers are incorporated. The EMD and MAD results were reported for these scenarios.

Using our model, we conducted batch correction within each panel, and between two batches from distinct panels (utilising only shared channels). In the absence of access to the cell type labelling used by the authors, we treated this dataset as unsupervised: we clustered the data together to facilitate resampling, giving six clusters that we treated as cell types. Our model incorporated conditioning of the batch variable, conditioning of the health status variable, latent space normalisation of the batch, and MMD applied between the batches. Our hyperparameters were optimised to yield the lowest weighted combination of model validation loss, EMD, and MAD scores. Noting that our model demonstrates strength on the EMD metric but underperforms on the MAD metric, we increased the contribution of the MAD score by a factor of three. Furthermore, we introduced the capacity to adjust the beta [20] parameter of a VAE, which scales KL divergence, into our hyperparameter optimisation. We then employed Mango for 50 iterations to optimise the combined objective.

The outcomes, shown in Table 2, indicate that our model surpasses cyCombine on the EMD metric on two datasets; however, we observed poorer MAD scores. This suggests that our model may lose some information from the original batches during the integration process. We attribute this to the nature of our method, which generates data from the latent space subjected to KL divergence regularisation, while cyCombine adjusts the original marker values with less regularisation, preserving variance more effectively. This test, however, presents a limited application of our model, as non-overlapping channels are disregarded in the alignment process and scoring.

2.8 Application to COVID-19 CIRCO data

Our clinical data consists of flow cytometry samples obtained between March 2020 and May 2021 from 4 hospitals in Manchester (UK) region. Multiple blood samples were taken from patients throughout their hospitalisation (between 1 and 6 temporal samples), and were labelled with peak observed clinical severity. Additionally, we processed control samples from healthy volunteers. The details of our patient cohort can be found in our previous publication [21].

Table 3 lists the total number of patients grouped by panel type and severity, as well as the total number of obtained flow samples. Flow cytometry was performed with two main panel types. Whole blood lineage panels, with markers selected to identify cell-types, can be used to track the overall immune cell-type proportions through disease trajectories and across severities. Chemokine panels, with markers targeting immune cell activity, enable the tracking of activation marker levels, primarily in neutrophils. Due to changes in panel design and available reagents, we obtain a number of distinct marker sets within each panel, 13 sets for lineage, and 19 sets for chemokine panels. Next sections describe how we used our method to integrate the data, identify

Table 3 Patient and flow sample counts by panel type and severity

	Patients peak severity			*Controls*	*Total*	*Marker*
Panel type	**Mild**	**Moderate**	**Severe**	**Healthy**	*samples*	*sets*
Lineage	73	78	76	31	1033	13
Chemokine	73	82	76	29	1040	19
Both	62	68	65	25	662	

[1]Combined count of all time-points of all patients.

[2]Indicates the same blood sample was processed in both types of panels.

significant variability of cell populations and markers , and train a regression model to predict peak severity from patient time-series.

2.8.1 Flow cytometry patient data acquisition

Typical methods of processing flow cytometry data involve manual cell-type gating (classification) of samples with packages such as FlowJo [22], and clustering with FlowSOM [14]. Batch effect correction may be performed using quantile or spline normalisation [8]. The first published analysis of flow cytometry samples of patients with COVID-19 used viSNE [23] to create a low-dimensional embedding, followed by clustering of the reduced space [24]. This approach was not suitable for our purposes, as it requires consistent marker sets. It may also be argued that viSNE reduction to 2 dimensional representation does not accurately preserve distances between samples.

In order to integrate data with disjoint marker sets we first processed each batch individually using standard methods to normalise and sub-sample the cells for integration. To minimise the number of distinct marker sets we created a mapping of marker names which we encounter in the data to a common set of names, treating differences in reagents as arbitrary batch-to-batch variability which will become aligned in our integration process. For each processed data series we used FlowJo [22] for manual compensation of channel spillover, and partial annotation of cell-types. We then performed semi-automated annotation of doublets and debris with OpenCyto [25]. For this we developed an R [26] pipeline involving automated reading of *.fcs* files from FlowJo workspaces, compensation, normalisation, and application of Logicle transform [27]. We used the openCyto *singletGate* to annotate doublets, and create a two-stage *flowClust* gate (first $SSC^{-}FSC^{-}$, then $SSC^{-}CD45^{-}$) to annotate debris. Each file was subsampled and values were stored in an HDF5 [28] data structure for easy interoperability with Python [29]. We included additional samples of rare populations in order to augment the model training process, but did not include these extra samples in statistic calculation or longitudinal analysis.

For batch annotation we used a concatenation of acquisition date and processing date, that is for samples to be in one batch they must have the same acquisition date, and the same processing date as other samples in this batch. This annotation is used for the batch effect correction mechanisms, which operate within each marker set. It is worth noting that we could use different batch assignment (for example, cells processed in a week, or treating patients as batches), or use multiple simultaneous assignments. We decided that the

combination of acquisition and processing date is broad enough to contain controls and multiple severity levels in each batch, while at the same time narrow enough that it is unlikely that additional technical variability will divide those batches further.

2.8.2 Integration approach

Our integration methodology starts by splitting the data into distinct marker sets, each optionally labelled for cell-types, debris, and doublets, as well as batch assignment for each cell. We first identify the primary marker sets, which span majority of the data. For the lineage panel, three primary marker sets span 87% of the samples, while for the chemokine panel, four span 77% of samples. These sets are selected for the initial parallel integration. During this phase, we train a VAE for each marker set, holding a random 30% of the data for validation of model selection. Each VAE includes latent space normalisation and conditioning with the provided batch assignment (refer to Section 2.3). During training we include additional samples of infrequent populations in the lineage panel.

To merge the latent spaces of the VAEs, a Subspace constraint is added, applying merging through shared channels. This is achieved by minimising embedding distance with a shared channel VAE, which performs its own batch correction using a combined set of batches for latent space normalisation and conditioning. Secondly, an MMD constraint is added between the embeddings of all VAEs (detailed in Section 2.4).

Two classifiers are defined to operate from the shared latent space of all VAEs. The first classifier predicts cell-types, while the second predicts doublets. We use the cell-type classifier to balance samples for the latent space normalisation and MMD. In contrast to the whole blood lineage panel, the chemokine panel predominantly comprises neutrophils, with small number of monocytes and eosinophils. Here, besides the cell-type balancing, we apply unsupervised balancing, for which GMM clustering is independently applied to each batch, resulting in 10 clusters each. This cluster assignment is subsequently used during resampling in conjunction with cell-type predictions, where each balancing is applied to a random half of the resampled data (as detailed in Section 2.7.4).

Having defined the model's architecture, hyper-parameter optimization is performed for optimal balance between reconstruction, merging, and separation criteria (further discussed in Section 2.6). Specifically, we optimise the frequency and pull strength of MMD and Subspace constraints, unsupervised and merging learning rates, and the ease-in epochs parameter for resampling. For the lineage panel, the normalised LISI score based on cell-types is used to measure separation (refer to Section 2.6.1). In contrast, for the chemokine panel, an unnormalized score defined by batch clustering is used. We experiment with different ranges and weights for these scores to achieve satisfactory results (additional details in Supplementary 4.7).

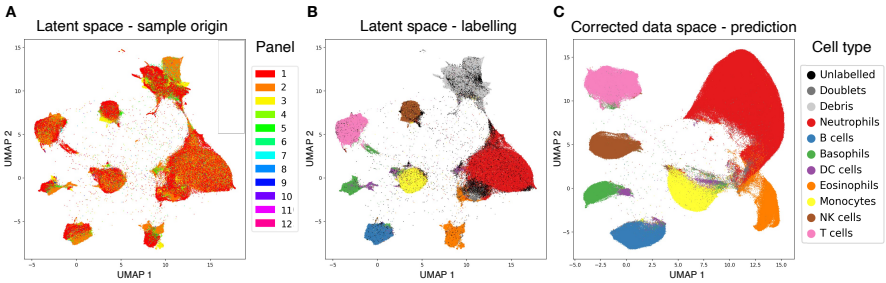

Fig. 6 UMAP visualizations of the latent space (A, B) and reconstructed data space (C) for whole blood lineage panels post-integration. A) Latent space coloured by panel overlap. B) Latent space coloured by manual cell-type annotation. C) Reconstructed data space after batch effect correction, showing predicted cell types and excluding debris.

The optimization process results in hyper-parameters which we use to train the model integrating the primary marker sets in parallel. We then proceed with the inclusion of subsequent, smaller sets into the model. During this phase, the model trained using the primary sets remains unchanged. For each newly added marker set, an additional VAE is incorporated. To merge its latent space with the primary sets, the most similar set among the primary VAEs is identified (with the largest feature overlap), and a Subspace merging is performed between the two. An MMD constraint is added between the new set and all the primary sets. Resampling is performed analogously with the primary sets, resampling both lineage and chemokine using cell-types, and additionally resampling the chemokine model with batch clustering. The newly added VAE is trained with the hyper-parameters selected during initial model selection, mapping the data onto the latent space of the primary model.

After the integration process is complete we predict the cell-types for all data, excluding the extra rare populations added for training. We also perform uniform marker reconstruction for all cells. This involves defining a marker superset, combining all features from the primary marker sets. We restrict ourselves to regenerating only these markers, avoiding the inclusion of uncommon markers found in a smaller subset of samples, which would likely result in imprecise interpolation. For batch uniformity, one batch is arbitrarily selected as the target for latent space normalisation and conditioning within each VAE. The regenerated data subsequently serves to analyze marker levels correlated with disease severity.

2.8.3 Whole blood lineage panels

For the whole-blood lineage panel, we integrated 12 distinct marker sets into a unified 50-dimensional latent space. A 2D UMAP representation of this space is shown in Fig. 6A, illustrating data alignment across marker sets, while Fig. 6B shows the manual cell-type labeling. Cell-type and doublet classifiers were used for the prediction of class assignments for individual cells. Using the three largest panels, we regenerated markers while correcting for batch effects. This

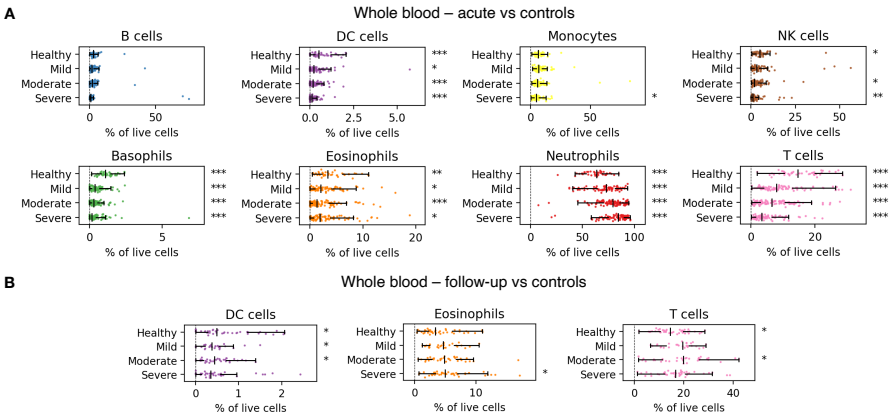

Fig. 7 Predicted proportions of cell types in whole blood lineage samples. Vertical bars denote the minimum, median, and maximum cell-type proportion, with outliers excluded. Significance indicators alongside each row are derived from Welch's t-test. Stars adjacent to the *Healthy* row mark significance comparing healthy controls against combined patient samples across all severity levels; stars alongside other rows denote significance between specified severity samples and controls. A) Comparing controls with acute samples. B) Comparing controls with follow-up samples (only significantly differing cell-types shown).

regenerated data space is visualized in the 2D UMAP in Fig. 6C, with relative values of markers shown in Supplementary Fig. 14.

After obtaining class predictions we computed cell-type proportions, excluding doublets and debris, for all patient time-point samples. This resulted in vectors of proportions between the eight remaining cell types. The distribution of these proportions is shown in Fig. 7A. Statistical significance was obtained using Welch's t-test, with indicators provided for each severity level. Our data reveals an increase in proportion of neutrophils with escalating COVID severity, concomitant with a decline in T-cells, DC-cells, NK-cells, and basophils. Neutrophilia and lymphopenia are established characteristics of the disease reported in other studies [30–32]. We also observe a decrease in eosinophils (which has been associated with poor prognosis [33]), more pronounced for samples labelled as moderate, but nonetheless observed for all acute severity groups, and a subtle decrease in monocytes for severe cases relative to controls.

A comparative analysis was then performed using follow-up samples from patients, which were collected after their recovery from COVID. We similarly grouped the follow-up samples by peak severity exhibited during the acute stage of the disease. Our data shows much smaller differences from controls in cell-type proportions compared to acute samples. The predicted ratios for cell-types with significant correlation with severity are shown in Fig. 7B. We observe eosinophil proportion that is larger in the follow-up samples from patients who recovered from severe COVID. T-cell proportion also appears to be increased, particularly in moderate severity, while DC-cell proportion appears to be reduced, particularly for moderate and mild severity samples.

Table 4 Significance of differences in cell-type proportions (via Welch's t-test) between patient samples and controls. Cell-types predicted by the model are compared to manually labelled. P-values correspond to controls compared against pooled samples for all severities. Bold values indicate p<0.05.

	P-value acute vs controls		**P-value follow-up vs controls**	
Cell-type	Predicted	Labelled	Predicted	Labelled
B cells	1.45E-01	2.54E-01	1.21E-01	2.01E-01
Basophils	**1.55E-10**	9.35E-02	2.38E-01	3.73E-01
DC cells	**5.91E-05**	**4.84E-02**	**2.77E-02**	8.22E-01
Eosinophils	**3.50E-03**	**4.06E-02**	5.39E-02	5.05E-01
Monocytes	6.08E-01	4.92E-01	3.18E-01	4.49E-01
Neutrophils	**1.51E-09**	**1.52E-04**	1.98E-01	6.71E-01
NK cells	**4.04E-02**	**9.83E-03**	4.86E-01	**3.96E-03**
T cells	**1.26E-09**	**5.89E-07**	**3.41E-02**	6.61E-01

We compared the significance of ratios after imputation and batch effect correction against values obtained on the same set of samples by only using manual labelling. Table 8 compares the p-values for each cell-type between controls and combined severity groups for acute and follow-up samples. We observe higher statistical significance by using our model predictions for nearly all significant cell-types, with the exception of NK-cells in both acute and follow-up samples, where our model predicts smaller significance than manual labelling. The largest increase in significance is observed in basophil and neutrophil proportions, where the p-value is 8 and 5 orders of magnitude smaller, respectively.

2.8.4 Neutrophil chemokine panels

Chemokine panel integration involved 18 distinct marker sets, which we projected into a common 50-dimensional latent space using the approach described in Section 2.8.2. The methodology is similar to that applied to the whole blood panel (section 2.8.3), with some key differences: no upsampling of rare cell-types in training, batch clustering added as a second resampling constraint alongside cell-type prediction, and unsupervised LISI alignment metrics used for hyper-parameter optimisation. We used four largest marker sets for the primary parallel integration. A 2D UMAP of the latent space is shown in Fig. 8A, illustrating dataset alignment, and Fig. 8B, illustrating cell-type prediction.

We then used chemokine data to investigate variability of activity markers expressed on neutrophils. Using cell-type and doublet predictions we filtered out neutrophils, and used the four largest marker sets to impute markers with batch effects corrected. We then used these values to compute statistical significance between controls and patient samples. We performed tests using acute and follow-up samples, comparing controls to all patient samples pooled together, and also separately for each severity group. The distribution of values for the most significant markers is illustrated in Fig. 9A for the acute samples, and Fig. 9B for the follow-up samples, with significance indicators obtained from Welch's t-test. We observe that a number of activity markers is significantly correlated with increasing disease severity in both acute and follow-up

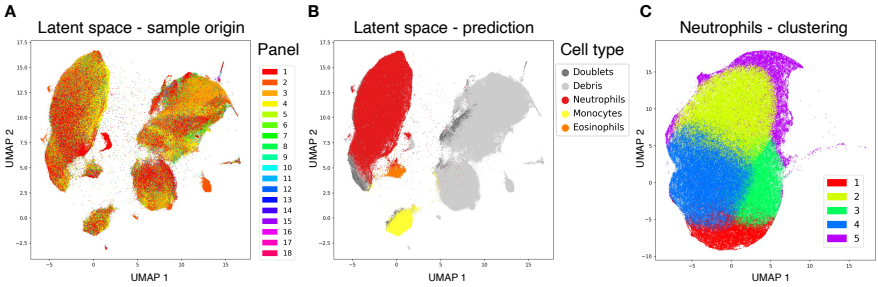

Fig. 8 UMAP visualisations of latent space (A, B) and reconstructed data space (C) of chemokine panels after integration. A) Latent space with colours showing panel overlap. B) Latent space showing cell-type prediction. C) Reconstructed data space of neutrophils only, after batch effect correction was applied. Colours show GMM clustering of combined neutrophil samples.

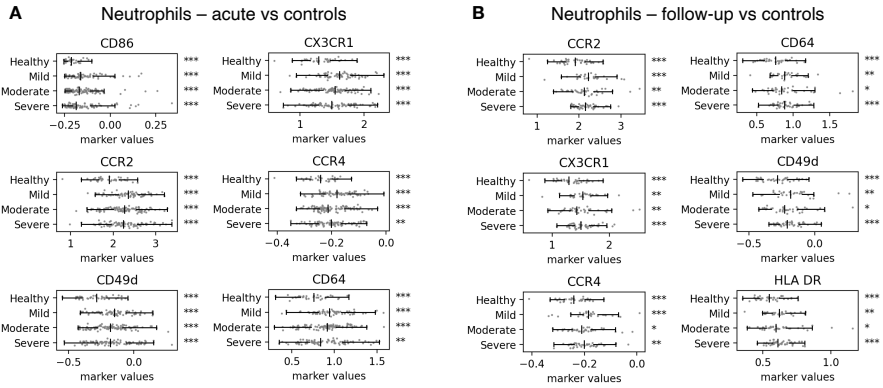

Fig. 9 Most significant differences in neutrophil marker values between patient samples and controls in chemokine panel, after batch effect correction and imputation of missing markers. Vertical lines indicate minimum, median, and maximum normalised marker value, excluding outliers. Significance indicators shown next to each row were obtained by Welch's t-test; stars next to Healthy row indicate significance between healthy controls and combined patient samples of all severities; stars next to other rows indicate significance between particular severity samples and controls. A) Comparison between controls and acute samples. B) Comparison between controls and follow-up samples.

samples, including CCR2, CD49d, CX3CR1, CCR4, CD64, and HLA-DR, which suggests a complex immune response. CD64 has been shown to be up-regulated in infections [34], including COVID-19 [35]. In our data, CD86 is the strongest marker associated with acute disease stages, with smaller albeit still significant enrichment in follow-up samples. We compared the significance levels of our regenerated markers to the values obtained from manual gating and noticed increase in statistical significance for many markers, such as CD86 or CCR4, but also a decrease for some, such as CD64. A full list of p-values for regenerated and manually labelled values can be seen in Supplementary Table 10.

In contrast to the original data with missing values and batch effects, the corrected and regenerated values are more suitable for clustering, as the clustering algorithm will no longer tend to split the samples by batch. Rather than analysing individual marker values separately, this approach allows for identifying enriched populations where activity markers concurrently vary. We performed GMM clustering of neutrophils as illustrated in Fig. 8C, dividing the space into 5 regions. The relative marker values in this space are shown in Supplementary Fig. 16. Although we did not observe separation between cluster populations, we can define boundaries in a continuous space of marker levels, and analyse how proportions of cells between those regions vary with severity. Supplementary Fig. 17 illustrates the relative differences in markers between five GMM clusters, while Supplementary Fig. 18 shows the relative proportions of clusters varying with severity. We observe that neutrophil cluster most significantly associated with severe samples, cluster 3, is characterised by relatively low levels of many activity markers, including CCR10^{lo}, CCR3^{lo}, CD10^{lo}, CD101^{lo}, CD117^{lo}, CD16^{lo}, CD33^{lo}, CD41^{lo}, CD62L^{lo}, CD63^{lo}, CD66b^{lo}, CD86^{lo}, CXCR2^{lo}, CXCR4^{lo}, HLA-DR^{lo} and PDL1^{lo}. For mild acute samples, as well as mild and severe follow-up samples, we observe an increase in cluster 2 proportions, which shows higher expression of CCR2^{hi} and CX3CR1^{hi}, and low CD86^{lo}. Simultaneously we see a decrease in cluster 4 (CCR2^{lo}, CCR4^{lo}, CD33^{lo}, CD49d^{lo}, and CD86^{lo}), which is inversely associated with severity for both acute and follow-up samples.

2.8.5 Longitudinal severity regression

Through imputation of full marker sets we addressed the problem of missing features, while batch effect correction alleviated the nuisance covariate shifts present in our data. As a result, it became easier to construct models that use longitudinal time series inputs to predict disease severity. We trained a number of such models and used them to further investigate disease progression and the predictive importance of the lineage and chemokine features included in our panels.

Using imputed data from both panels we constructed models predicting peak observed severity from a time-series of samples, and tested them using cross-validation. Patients together with controls were randomly split into four folds, and for each test fold three models were trained on combinations of the remaining folds, where two were used for training, and one for validation with early stopping. Average predictions of three models were used to predict the peak severity of samples in each test fold.

Note that we obtain the regenerated markers from a single model integrating all the data. This does not cause data leakage, as the severity labels were never used in the integration process. A stricter approach might be argued, in which one would create a new integration model for each of the test folds, and project the test input data on top of it afterwards. However, since data integration requires both new and existing samples, the difference here would only highlight the relative efficacy of parallel versus sequential integration. For

efficiency and due to relatively small sample size we did not investigate this problem further.

The peak severity categories, Healthy, Mild, Moderate, and Severe, were encoded with target ordinal regression values of 0.5, 2.0, 3.0, and 4.5 respectively, and the model was optimised for mean squared error (MSE) loss. For calculating classification scores, the predictions were rounded to the nearest integer, and compared to mapping between 1 and 4. The expansion of range for the lowest and highest category ensures that all four categories target approximately the same span of values (that is, Healthy samples can be predicted between 0.5 and 1.5, Mild between 1.5 and 2.5, etc.), which improved our accuracy for the Healthy and Severe categories. To enable comparison between models which use one or both panel inputs, we only used samples which contained both types of panel, resulting in 26 controls, and 41 mild, 47 moderate, 34 severe acute patients.

Our annotation of severity does not vary between individual patient time-points, and refers to the peak severity observed throughout the course of hospital admission. The data includes variable number of time-points per patient, ranging from 1 to 6. We therefore construct differentiable models that first predict the severity of individual time-point samples for each patient (using shared parameters), and consecutively pool all the time-point outputs to predict peak severity. We tested max-pooling as well as simple recurrent neural network (RNN) as the pooling functions, to investigate whether additional information can be obtained from the temporal order of patient samples.

The classification or clustering of cells is used to reduce the number of features and can improve generalisation by obscuring batch effects otherwise present in the original marker values. However, we speculated that it could also hinder prediction, as important features about sub-populations within each class could be discarded. We therefore investigated two types of inputs to our models. Firstly, we created models using cell-type proportion vectors as inputs for each temporal sample. These included cell-types from the lineage panel, as well as cluster proportions of neutrophils in the chemokine panel. In the latter, to evaluate coarser and finer granularity, we tested clusters obtained from GMM (5 total) as well as Leiden algorithm (31 total). The visualisation of these clusters is shown in Supplementary Fig. 15.

Secondly, we created models which accept marker values directly, without applying cell-type prediction or clustering. These models accept a random set of cells, each cell represented as a vector of marker levels, and process the set through a permutation invariant function (a span-1 convolution, followed by parallel max and average-pooling, and a fully-interconnected layer). These set models allow for learning directly from continuous marker levels, and avoid defining boundaries between cell-types altogether. Similarly to proportion models, we investigated combining the temporal information by max-pooling or an RNN layer. For making predictions with these models, we repeat the cell sampling and return an average score for multiple random sets.

Table 5 Cross-validation performance in regression of peak severity for longitudinal samples. MSE shows validation and test losses for ordinal regression. F1 scores are calculated by rounding regression output to assign a severity class. Underlined scores are best in each model category, bold scores are best overall. WB - whole blood lineage input; Chemo. - neutrophil input from chemokine panel; MLP - multilayer perceptron.

Input included		MSE		Classification F1				
WB	Chemo.	Val.	Test	Healthy	Mild	Mod.	Severe	Avg.
Proportions input, linear, max-pooling								
✓	-	1.324	1.298	0.308	0.318	0.556	0.111	0.323
-	GMM	1.549	1.568	0.000	0.243	0.394	0.056	0.173
-	Leiden	1.435	1.438	0.323	0.349	0.483	0.000	0.289
✓	GMM	1.379	1.386	0.182	0.343	0.419	0.100	0.261
✓	Leiden	1.309	1.338	0.462	0.275	0.454	0.000	0.298
Proportions input, MLP, max-pooling								
✓	-	1.313	1.293	0.333	0.352	0.496	0.273	0.363
-	GMM	1.574	1.548	0.000	0.283	0.400	0.000	0.171
-	Leiden	1.427	1.438	0.323	0.353	0.479	0.000	0.289
✓	GMM	1.409	1.421	0.235	0.364	0.468	0.103	0.292
✓	Leiden	1.319	1.427	0.462	0.325	0.489	0.053	0.332
Proportions input, MLP, RNN pooling								
✓	-	1.275	1.236	0.308	0.364	0.446	0.261	0.345
-	GMM	1.521	1.520	0.000	0.355	0.330	0.279	0.241
-	Leiden	1.377	1.432	0.323	0.392	0.437	0.318	0.367
✓	GMM	1.365	1.360	0.069	0.342	0.414	0.311	0.284
✓	Leiden	1.335	1.302	0.303	0.358	0.460	0.273	0.349
Set input, convolution, max-pooling								
✓	-	1.168	1.159	**0.488**	0.454	0.556	0.360	**0.464**
-	✓	1.250	1.290	0.486	0.325	0.493	0.195	0.375
✓	✓	1.137	1.173	0.474	0.360	0.508	0.340	0.420
Set input, convolution, RNN pooling								
✓	-	1.222	1.251	0.378	**0.455**	**0.569**	0.367	0.442
-	✓	1.268	1.290	0.432	0.311	0.441	0.431	0.404
✓	✓	1.158	**1.151**	0.486	0.383	0.527	**0.436**	0.458

Table 5 shows the cross-validated performance in predicting held-out peak severities for our samples. Test performance is shown as MSE loss for the regression output, as well as F1 classification score, obtained by rounding the output to obtain categorical prediction. We observe, that for linear models using cell-type or cluster proportions as input, the best performance is achieved by only using proportions of major populations from the whole-blood panel. This performance increases after introducing a hidden non-linear layer to the neural network. In this multi-layer perceptron (MLP) setting, the best test loss is similarly obtained from models using just the lineage panel data, with the RNN pooling model reaching lower MSE loss, but the max-pooling model reaching higher F1 score. For the models using neutrophil cluster proportions we see that using more numerous Leiden clusters is more beneficial than training with fewer GMM clusters. The MLP model with RNN pooling, trained only on Leiden proportions achieves the best overall F1 score among all models trained on proportions.

We then compared the proportion models to set models, which use marker values directly. The set models achieve higher overall performance in both regression loss and classification scores. The lowest regression loss is achieved

by the model using RNN pooling, which simultaneously accepts two panel inputs. However, the best overall classification score is obtained by the model using max-pooling, and only using whole-blood panel input. Overall, those two models show similar performance, with a notable difference in predicting mild (where the max-pooling model is better) and severe classes (where the RNN model is better). We conclude that the whole-blood data alone can be a good predictor of COVID severity, however it is worth noting that using the chemokine data only, both set models still outperform all proportion models in classification, and all but one in regression.

To visualise disease trajectories for individual patients we used the best performing max-pooling model to predict the severity of each patient time-point separately. Since the models were trained with a shared network structure for processing each time-step, and max-pooling at the end, we can effectively visualise the predicted outputs before the maximum is taken. This allows us to interpret the severity of each longitudinal sample with detail beyond the overall label of peak severity. Fig. 10 shows the individual time-points for a subset of controls and patients. We see that with the varying prediction of severity, the proportion of cell-types changes, with particularly strong increase in neutrophils. While the immune cell-type proportions vary significantly across controls, for severe COVID samples we see large decreases of all other populations. The variable level of markers, also illustrated by cluster proportions, highlights a complex and variable immune response across individuals.

After training, we used gradient attribution to identify most predictive regression features learned by the chemokine-only RNN pooling model. The reason for selecting this model, despite its weaker predictive performance compared to the dual-input version, is that using only one panel input results in stronger gradients for that input. We then grouped the obtained scores by the severity predicted by the model for each sample. The markers with the strongest sensitivity on neutrophils are shown in Fig. 11, and include CCR2, PDL1, CX3CR1, CD64, and CD117. These features are consistent with myelopoiesis resulting in immature and dysfunctional neutrophils [36], which has been observed in severe COVID-19 [37, 38]. Presence of FSC-W as a strong feature indicates sensitivity to neutrophil size, associated with activated cells [39]. All of the above mentioned features show decreasing importance as the predicted disease severity increases. A full list of gradients is given in Supplementary Fig. 19.

3 Discussion

Integration of heterogeneous data brings along many challenges, including disparate features and difficult to control batch effects. We presented UVAE, a variational autoencoder framework addressing these issues. Our framework allows for automatic training and optimisation of models tailored for user-specified data inputs and annotations, together with merging and normalisation. A key innovation of our method is the ability to use supervised or

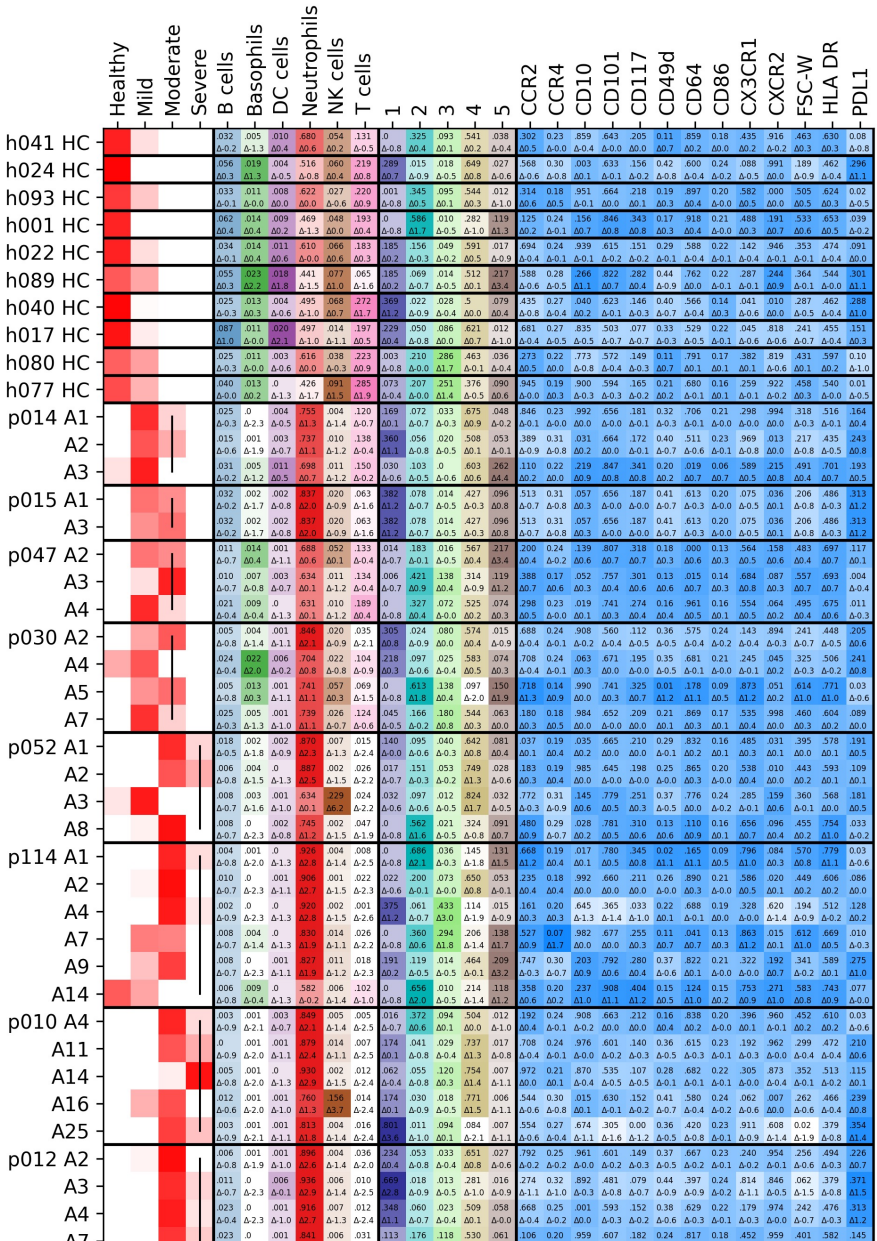

Fig. 10 Visualisation comparing a subset of healthy controls (HC) and longitudinal patient samples (p...), columns showing left to right: predicted severity of each time-point (A...) in separation (vertical black bar indicates peak severity label), whole-blood cell-type proportions, neutrophil GMM cluster proportions (1-5), selected chemokine marker values on neutrophils. Patient rows are ordered by disease progression time. Top number in each field is the cell-type/cluster proportion, or marker value. Bottom number is the number of standard deviations from mean values observed in controls, also indicated by colour intensity.

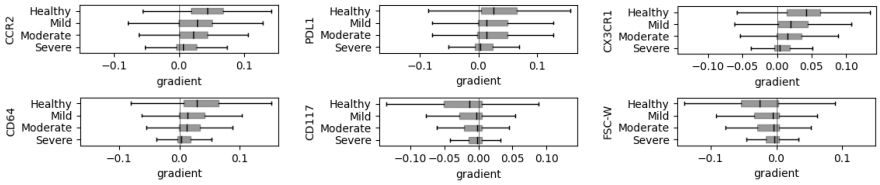

Fig. 11 Markers showing highest average gradient of regression output predicting increasing COVID severity w.r.t. neutrophils marker input. Gradients were obtained from a model trained using only chemokine panel input, and are split by predicted severity of the sample.

unsupervised annotations to balance the integration process, increasing performance on datasets where ground-truth manifolds of disparate data streams do not exactly match.

We applied our framework to clinical flow cytometry samples of COVID-19 patients, which included multiple panel designs and marker sets. Thanks to data imputation and correction we are able to perform uniform analysis on all available samples, overcoming batch effects and missing features. We highlighted features of disease consistent with previous knowledge, including cell-type proportions, and chemokine markers. The uniformity of the regenerated data allowed for easy application of clustering methods, showing novel associations of neutrophil populations with COVID-19 progression. We additionally obtained associations in the follow-up data, uncovering footprints of disease severity present after patients' recovery.

The regenerated data enabled us to train models predicting outcomes from a variable number of longitudinal samples. We show that using randomly sampled sets of cells without cell-type annotation results in superior predictive performance to more traditional approaches involving inference on cell-type proportions. We also showed how such regression models can be used to gain further insights to disease trajectories, and used feature attribution to identify most predictive features associated with disease severity.

We believe that this work introduces significant method improvements in the analysis of clinical flow cytometry datasets, with its principles extending to the analysis of any heterogeneous or unpaired data. Our Python framework is applicable to all real-valued data, and is easily extensible and freely available.

4 Methods

4.1 Processing of flow cytometry samples

Flow cytometry COVID-19 data was reported in our previous study [21]. The samples were manually processed using FlowJo [22] for channel spillover compensation, and annotation of cell-types. The samples were then extracted from workspaces with CytoML [40] using an R [26] pipeline. Logicle transform [27] was applied with flowCore [41], with default parameters (w=0.5, t=262144, m=4.5, a=0). Manual labelling, where available, was first read from samples using flowWorkspace [42]. Each *.fcs* file was then subsampled to 100000 events

for application of openCyto [25] *singletGate* to identify doublets, and a two-stage *flowClust* gate (first SSC$^-$FSC$^-$, then SSC$^-$CD45$^-$) to identify debris. These automatically generated labels were saved as separate annotation from the manual labelling. For each *.fcs* file, a random subsample of 10000 events was taken. In addition, 1000 cells from each manually defined class were taken in the whole-blood lineage panel in order to up-sample rare populations for training. Channel values together with annotations were then stored in an HDF5 [28] data structure. This structure preserved the panel and sample hierarchy, while separately storing randomly sampled and up-sampled data for each original *.fcs* file. Python [29] class was created for reading the HDF5 structure in a simplified way, allowing to aggregate all samples which share the same marker set. This interface was used to generate numpy [43] arrays of marker values and annotations for input to UVAE models.

Clinical data was obtained from the original study [21], which provided labels of peak disease severity for each patient. Batch identifiers were obtained by concatenating the acquisition date and processing date for each sample.

4.2 UVAE algorithm

UVAE is a framework for automatic optimisation of VAE-based semi-supervised architectures, implemented in Python [29] using Keras [44] for neural network training. UVAE model consists of multiple network components, created to match data constraints specified by the user. The model can be saved and loaded, creating persistence of parameters across runs of the encompassing Python script. This allows, for example, to train a model on one set of data and later predict another, or train a model and anchor additional data on top of it.

Fundamentally, UVAE trains a β-VAE-like [20] autoencoder for each input data series with a specific feature set. Supervised annotations can be defined over arbitrary subsets of the data (using masking to pad the samples where labels are unavailable), which are then used to train classification or regression functions operating on latent space shared between all VAEs. Besides supervised prediction targets, annotations can be specified for merging or normalisation constraints, such as batch effect correction or MMD. In a UVAE model, encoders are used to project from the data space to the latent space while removing batch effect biases. All remaining constraints use the latent embedding from encoders to make further predictions, such as classifying samples, or calculating MMD distance. These serial constraints can backpropagate their loss through the encoders in order to converge them.

In training, mini-batches are defined for each trainable constraint, all mini-batches are combined, and trained in a random order. Gradient clipping is applied to increase stability. The combined loss is the average of all batch losses in each epoch, including autoencoders evidence lower-bound (consisting of MSE reconstruction loss and scaled KL-divergence), classification, regression, and merging losses. Three *Adam* optimisers are used in parallel, *unsupervised* - used to train the autoencoders, *supervised* - used for classification and

regression constraints, and *merge* - used for merging constraints. The mutual learning rates of these optimisers can be used to balance the speed of training to achieve optimal alignment, reconstructions, and predictions. At the end of an epoch, resampling and normalisation are applied if present. Supplementary Algorithm 1 outlines the training process, which encompasses classifier training outlined by Supplementary Algorithm 2.

To manage multiple series of partly annotated data, UVAE uses indexing to store mapping to discontinuous samples as continuous arrays, which are then used during training to define stochastic mini-batches. These batches can be optionally up-sampled with replacement, or down-sampled by setting the *frequency* parameter. If *frequency* is set to one, the amount of training for each constraint is proportional to the number of datapoints it includes. Independently to overall training frequncy, each constraint, e.g. autoencoder, classifier, or merging constraint, defines an index array, which by default maps to every input sample included in this constraint once. When resampling is applied to a constraint, its index array is randomly modified to match underlying sample proportions based on the source of resampling (manual labelling or classifier prediction), and the chosen rule (sampling equally from each class, or sampling to match average proportion of classes across all batches). Resampling retains the overall number of samples per constraint. The *ease-in epochs* hyperparameter defines the proportion of the data to be resampled, starting from zero, and linearly increasing to all included data when training epochs reach this number. The resampling process using a classifier to match the average proportion of classes across batches is outlined in Supplementary Algorithm 3. If a validation split is defined, a randomly selected proportion of data is selected to be held out, including all supervised or merge constraints defined over those samples. Resampling is not applied to validation samples.

4.3 Batch effect correction

UVAE defines two methods of batch effect correction, which are simultaneously supported by each autoencoder to separate the influence of nuisance covariates from the latent space. These operate in addition to any MMD-based constraints, which can also be used to minimise batch differences.

Conditioning of VAE decoder and encoder concatenates a vector specifying the nuisance covariate to encoder and decoder input (encoder conditioning is optional). This covariate is a categorical variable, and can be projected to a fixed dimensionality through a linear layer. The linear embedding is trained with backpropagation during training of each VAE and is the same for encoder and decoder. More than one covariate can be concatenated simultaneously. The nusiance covariates are not concatenated to encoder output when the embedding is used as input to other constraints, or when latent normalisation offsets are calculated.

Latent space normalisation involves offsetting each sample embedding in the latent space by a vector specific to the batch from which the sample originates. This vector is calculated to offset whole batch embedding (mean

embedding of a random set of inputs drawn from a given batch) to a common reference vector, which can be either the mean of a specific target batch, or the average of all batch means. The vectors are calculated at the end of each epoch after resampling, so that included samples are class-balanced between batches. The offsets are only updated during training, and are saved as part of the model, linked to their batch identifiers. If more than one normalisation constraint is specified simultaneously, the offsets for each constraint are calculated with the remaining normalisations applied. During prediction, the offsets are applied before the embedding is passed to consecutive constraints, with the exception of VAE decoders, which receive the un-normalised embeddings as input. Supplementary Algorithm 4 outlines the latent normalisation process.

4.4 Merging of data with disjoint feature sets

Disparate input features require training multiple VAEs, one per feature set. Their latent space embeddings can be aligned by merging corresponding data series (or chosen subsets) using feature matching, or density matching. Both types of constraints implement a *pull* parameter, which can be adjusted to increase or decrease their loss contribution.

Feature matching is implemented as a Subspace constraint, which is defined by the user over subsets of data with some overlap in annotated input features (channels). Subspace constraint identifies the common channels present across all included data, and implements a VAE accepting this subset as input. During training, random samples are encoded through their specific encoders, and simultaneously through the subspace VAE encoder (in the latter case, subset to shared channels). Both VAEs can implement batch effect correction, which in the shared VAE is performed using a union of all batches present in the included data. Mean squared distance is calculated between the two embeddings for each sample, and encoders are trained to minimise this distance. Supplementary Algorithm 5 outlines the process of merging using shared features.

Density matching is implemented through minimisation of MMD loss between specified groups of data. These groups can span samples encoded by one or more VAEs. For each mini-batch of training, a random sample of inputs is drawn from two groups (randomly selected in each mini-batch if more than two groups are being merged). The samples are drawn from class-balanced data if resampling is defined. MMD loss is calculated between the two groups of samples through multi-scale Gaussian kernel, as defined in trVAE [16]. All encoders used to embed the samples are then trained to minimise the MMD loss. Supplementary Algorithm 6 outlines the MMD merging process.

4.5 Regeneration of uniform dataset

Regeneration of the entire dataset is performed to obtain data space representation of each sample without batch or marker-set specific variability. To achieve this, all desired samples are generated using a fixed set of decoders,

with fixed targets selected for each of the nuisance covariates such as batch. The process involves multiple instances of translation across encoder-decoder pairs trained on different marker sets, and therefore requires that the latent space is well-aligned across batches and panels.

In the simplest case, for each sample, the sample-specific encoder is used to embed it to a latent space representation. This embedding is then input to a chosen decoder, even if it was originally trained on samples containing different features. If multiple decoders are used, the same embedding is input to all of them, and for features that repeat an arithmetic mean is taken.

If batch effect correction mechanisms are used, the process requires additional consideration of source and target batches (or other nuisance covariates). If encoder and decoder conditioning is used, the sample is first encoded while providing the source batch to the encoder. The latent representation is then decoded while providing the target batch to the decoder. Similarly, if latent normalisation is used, the source batch offset is subtracted after the encoding, and the target batch offset is added before decoding. The process of feature imputation is outlined in Supplementary Algorithm 7.

4.6 Normalised or unsupervised LISI scoring

LISI calculation is performed using the implementation provided in Harmony [18]. LISI scores are calculated given an embedding, categorical annotation of this embedding, and perplexity parameter defining effective neighbourhood size. Score is returned for each sample (each cell), where higher score indicates a higher level of mixing between specified categories in the given neighbourhood. A median is taken across samples to obtain a single value. Typically, two scores are calculated using batch and class annotation (in the original paper referred to as iLISI and cLISI, respectively), and high batch mixing scores and low class mixing scores are desirable.

We define a normalised LISI score in order to avoid assigning high scores to embeddings with too high level of mixing. To calculate the normalised score, given an embedding with batch and class annotations, where classes are shared across batches (supervised case), the lowest count of each class across all batches is determined, and classes in each batch are randomly down-sampled to match this lowest count. The down-sampled embedding, alongside the corresponding batch and class values, is used to perform the LISI calculation, and median is taken across samples.

We define an unsupervised LISI score for situations where no class assignment is available, and it cannot be reliably assumed that a clustering of data will encompass all batches in each cluster (in heterogeneous datasets the clustering might tend to split the batches due to batch effects). To calculate unsupervised LISI score, clustering is performed on each of the batches independently. Cluster identities are used in place of classes for LISI calculation. In this situation, the ideal separation score for sufficiently high perplexity values will not be equal to one, but instead will be equal to the number of batches,

as one unique cluster identity from each batch is expected to appear in each neighbourhood.

4.7 Hyper-parameter optimisation

Hyper-parameter search is performed with *mango* [17] Bayesian optimiser. Validation loss is calculated using a held-out portion of the data, and includes model loss (consisting of autoencoder, supervised, and merging constraint losses), and alignment losses. Each constraint in the model has an associated *weight* parameter, which can be used to scale the contribution of its loss during model selection process, without affecting backpropagation. Alignment losses are defined as two LISI scores, measuring batch alignment (higher is better) and category alignment (lower is better), calculated for a given perplexity. Because the range of LISI scores is difficult to predict *a priori*, when training models on real data we measure and adjust the expected range and multiplier for each score over the course of several runs to achieve satisfactory alignment. Any custom scoring function can be defined instead of LISI for model selection, such as a combination of EMD and MAD scores.

UVAE defines a number of hyper-parameters which can be adjusted (all or a subset can be included in the search). Optimisation process will automatically discard hyper-parameters that do not contribute to the architecture, i.e. if no conditioning is used, conditioning parameters will be excluded from the search. The hyper-parameters consist of: latent space dimensionality; number of hidden layers in VAE encoder and decoder; number of neurons in a hidden layer; slope of Leaky ReLU activation; amount of dropout; pull strength of merging constraints; dimensionality, depth, and number of neurons in a conditioning embedding layer; unsupervised, supervised, and merge learning rate multipliers; gradient clipping coefficient; ease-in epochs; frequency of training for each constraint; batch size; beta parameter multiplying KL-divergence. Optionally, depth and number of neurons can be optimised independently for each VAE. Supplementary Table 11 lists the hyper-parameter values used to train the COVID-19 models, and their optimisation ranges.

4.8 Generation of synthetic benchmark data

Synthetic data was generated to investigate the ability of our algorithm to correct for batch effects and impute missing channels in a situation where integrated data streams show an uneven distribution of underlying classes. A single flow cytometry sample was selected from fully annotated controls to serve as the ground-truth, with debris and unlabelled events removed. From the remaining sample, containing eight valid immune cell-types, a random 100000 cells were taken, and standardisation to mean=0 and standard deviation=1 was applied. This data was clustered with GMM to 20 clusters, which were used to split the sample into panels and batches, so that each panel and batch holds an uneven proportion of clusters. The sample was split into 3 panels, and each panel was split into 3 batches. For each split, 40% of the data was

split at random. To perform the uneven split, 3 samples were drawn from a random exponential, and used to proportionately randomly assign the remaining 60% to each panel or batch. After the split, the data in each batch was independently standardised again, which caused misalignment due to varying cluster proportions. During panel split, two channels were removed from each panel: CD45 and CD11c from panel 1, CD16 and CD11b from panel 2, and CD10 and CD14 from panel 3. These values were then used to calculate the imputation error during benchmarking.

4.9 Synthetic testing procedure

Tests involved training various configurations of UVAE on the synthetic sample, with the goal of aligning batches and panels, and using two panels to impute the missing values of the third one. The alignment results were assessed using batch and ground-truth cluster assignments to calculate a normalised LISI scores (where each cluster in each batch was down-sampled to the lowest count across batches, performed without replacement to avoid duplicate cells in LISI nearest-neighbour calculation). LISI was calculated with perplexity=100, using official R implementation [18]. The imputation was performed for each panel by encoding its values using panel encoder, and decoding them using the remaining two decoders, while setting an arbitrary but constant target batch. The average of the two outputs was then standardised, and MSE loss calculated between the result and the original removed values.

A ground-truth model was trained on the unaltered synthetic sample (before it was split and offset) to obtain idealised reference performance. A single VAE was trained for 50 epochs with batch size of 512, with 50-dimensional latent space, two hidden layers of 256 neurons each with leaky *ReLU* activation (0.2 slope). Training used MSE reconstruction loss with standard β-VAE ELBO, where $\beta = 1$. The same hyper-parameters were used for the remaining models.

Hyper-parameter optimisation was performed for each tested configuration using mango optimiser [17] for 30 iterations. The optimised objective was a sum of model loss (involving ELBO, and optionally classification and merging losses) multiplied by a factor of 3.0, and two LISI scores. The ranges of LISI scores were defined by training two worst-case models to identify the top and bottom of expected score range. During optimisation, the actual score was placed in this range to be normalised between 0 and 1. Worst-case scores for maximum separation were obtained by training a model on the corrupted data, without applying any merging or batch effect correction. The scores for worst-case of maximum mixing were obtained by then shuffling the positions of cells obtained from the maximum separation model.

The hyper-parameter search included *frequency* (1.0x to 10.0x) and *pull* (1.0 to 20.0) of MMD and Subspace constraints (if present), the mutual *supervised*, *unsupervised* and *merge* learning rates (between 0.2x to 1.0x of default

Adam learning rate of 0.001), and the *ease-in epochs* (1 to 10) for resampling and normalisation. After optimisation, 10 models were trained for each configuration and scores of all were reported.

For each tested configuration, a UVAE was created with three marker sets added, and 20% of the data held-out for validation. The tests included or excluded the batch effect correction constraints (latent space normalisation, or VAE conditioning), and merging constraints (shared channel Subspace, or MMD), with or without resampling applied to the sensitive constraints (latent space normaliation, and MMD). To separate the influence of training a classifier, resampling with cell-types was performed using manually annotated labels, by balancing the labelled cell-types in each batch. Unsupervised resampling was performed by using an independent GMM cluster assignment for each batch (20 clusters per batch), where balancing was performed by taking an equal number of samples from each cluster. In experiments with multiple clusterings applied in parallel, independent clusterings to 20, 15 and 10 clusters per batch were used.

4.10 Comparison with cyCombine

For comparison with cyCombine the data was downloaded from the original repository [1], and included *.fcs* files for panels 1 and 2 of Chronic Lymphocytic Leukemia (CLL) CyTOF dataset, alongside metadata designating sample status. The samples were extracted using the *prepare_data* function provided by the authors, with settings *cofactor*=5, *derand* and subsampling enabled, and with columns subset to valid flow markers (36 in panel 1, 34 in panel 2, 15 shared between panels). This data was used to train UVAE models, including one for each panel, and one for combining the third batch of each panel, with input channels subset to the shared markers. In each case, UVAE models had hyper-parameters optimised with *mango* for 50 iterations, using a loss function consisting of a sum of model loss, negative EMD score, and MAD score multiplied by 3.0. EMD and MAD were calculated with cyCombine implementation. Optimisation involved all UVAE hyper-parameters, including β multiplier of KL-divergence. Because of this, the KL loss was excluded from the model selection score. The models included VAE conditioning of batch and disease status (healthy or CLL), latent space normalisation of batch, and MMD applied between batches. Resampling was applied to normalisation and MMD using 6 clusters obtained by GMM clustering of the whole dataset. After optimisation and training, samples were regenerated by setting the target batch to *5* in each individual panel model, and the first panel (*p1b3*) in the shared channel model.

4.11 Severity regression models

For input to models predicting patient disease severity a common set of markers was generated using WB and chemokine models. For the WB panel, three largest marker sets were selected, and random 500000 cells were regenerated

as an average of those sets. Cell-type prediction was performed by combining cell-type and doublets classifier outputs. For the chemokine panel, four marker sets were used to regenerate a random sample of 500000 cells, similarly annotated with cell-type prediction. Additionally, the marker values of predicted neutrophils were clustered by GMM to 5 clusters, and by Leiden algorithm to automatically detected 31 clusters. Predicted debris and doublets were removed from further analysis. The remaining cells were used for input to the models either directly, as sets of cell marker vectors, or after calculating proportions of categories in each patient time-point sample. These proportions were obtained separately for WB cell-types, and for GMM or Leiden clusters.

In order to quantify the predictive importance of WB and chemokine panels, time-points which did not contain both WB and chemokine data were excluded from training of regression models. This allowed all model configurations to be trained on the same set of samples. Only acute patient samples were used, resulting in 26 controls, 41 mild, 47 moderate, 34 severe patients, with a total of 246 time-points, and between 1 to 6 time-points per patient. For each longitudinal time-series, the peak severity regression target was encoded as Healthy: 0.5, Mild: 2.0, Moderate: 3.0, Severe: 4.5. After prediction, the output was rounded to nearest integer to be assigned a class: Healthy (1), Mild (2), Moderate (3), Severe (4). To equalise sample count for each severity group, up-sampling was applied with replacement for each severity class so that samples occur ten-fold more frequently per epoch than the most frequent class (that is, 470 times each). Zero-padding with masking was used to allow for variable number of time-points per sample.

To train models accepting proportions as input, proportion vectors were obtained for each temporal sample, and if both panels were used the two vectors were concatenated. In linear models, the input was directly followed by a single fully-interconnected neuron with linear activation, applied independently to all non-zero inputs. The prediction was then max-pooled across all time-points to result in peak severity. In MLP models, a time-distributed dense layer with 50 neurons and $ReLU$ activation was independently applied to each time-point. Consecutively, the max-pooling models applied a single linear neuron, followed by global maximum. The RNN pooling models applied an RNN layer with 50 units and *tanh* activation, followed by a single linear neuron.

To train set models, a random sample of 20 cells was taken from either panel input, or both simultaneously. Each set input was then processed for each time-step by a shared network arm consisting of length-1 convolution (effectively a time-distributed dense layer applied to each cell independently) with 50 filters and $ReLU$ activation, followed by global max-pooling and average-pooling, outputs of which were concatenated. The output of this network arm is invariant to the permutation of cells given as input. A separate arm was trained for WB and chemokine inputs, and if both were used, their outputs were concatenated for each time-point. For a max-pooling model, an additional time-distributed dense layer with 50 neurons and $ReLU$ activation was applied before a single linear neuron and global max-pooling. The RNN model

did not use the extra dense layer, but instead applied an RNN layer (with 50 units and *tanh* activation), followed by a linear neuron.

All models were trained using 4-fold cross-validation, randomly splitting patient and control samples. For each fold, the remaining three folds were used to train models on combinations of two concatenated training folds, and one validation fold for a maximum of 2000 epochs with early stopping applied if the validation loss did not improve for 50 epochs. After three models were obtained, they were used in an ensemble to predict on the held-out fold, with their outputs averaged. The proportion models were trained with a batch size of 256. The set models were trained with a batch size of 64. For each training sample of a set model, a new random set of cells was sampled. To predict a validation loss, this sampling was repeated 10 times, with the losses averaged. For final prediction, the sampling was repeated 100 times for each sample, with the predictions averaged.

Gradient-based attribution was applied to identify most sensitive features of the set models. Using a trained model, a random sample of cells was input to the model, and gradient of the output w.r.t. the input was calculated. This process was repeated 10 times for a new random set of cells. The gradients were then split by the predicted severity of each sample, and by their annotated cell-type.

References

[1] Pedersen, C.B., Dam, S.H., Barnkob, M.B., Leipold, M.D., Purroy, N., Rassenti, L.Z., Kipps, T.J., Nguyen, J., Lederer, J.A., Gohil, S.H., Wu, C.J., Olsen, L.R.: cycombine allows for robust integration of single-cell cytometry datasets within and across technologies. Nature Communications **13**, 1698 (2022). https://doi.org/10.1038/s41467-022-29383-5

[2] Cao, Z.-J., Gao, G.: Multi-omics single-cell data integration and regulatory inference with graph-linked embedding. Nature Biotechnology (2022). https://doi.org/10.1038/s41587-022-01284-4

[3] Tran, H.T.N., Ang, K.S., Chevrier, M., Zhang, X., Lee, N.Y.S., Goh, M., Chen, J.: A benchmark of batch-effect correction methods for single-cell rna sequencing data. Genome Biology **21** (2020). https://doi.org/10.1186/s13059-019-1850-9

[4] Luecken, M.D., Büttner, M., Chaichoompu, K., Danese, A., Interlandi, M., Mueller, M.F., Strobl, D.C., Zappia, L., Dugas, M., Colomé-Tatché, M., Theis, F.J.: Benchmarking atlas-level data integration in single-cell genomics. Nature Methods **19**, 41–50 (2021). https://doi.org/10.1038/s41592-021-01336-8. Accessed 2022-12-01

[5] Tomczak, J.M.: Deep Generative Modeling. Springer, ??? (2022). https://books.google.co.uk/books?id=uidgEAAAQBAJ

[6] Abdelaal, T., Höllt, T., van Unen, V., Lelieveldt, B.P.F., Koning, F., Reinders, M.J.T., Mahfouz, A.: Cytofmerge: integrating mass cytometry data across multiple panels. Bioinformatics **35**, 4063–4071 (2019). https://doi.org/10.1093/bioinformatics/btz180

[7] Becht, E., Tolstrup, D., Dutertre, C.-A., Morawski, P.A., Campbell, D.J., Ginhoux, F., Newell, E.W., Gottardo, R., Headley, M.B.: High-throughput single-cell quantification of hundreds of proteins using conventional flow cytometry and machine learning. Science Advances **7**(39), 0505 (2021) https://www.science.org/doi/pdf/10.1126/sciadv.abg0505. https://doi.org/10.1126/sciadv.abg0505

[8] Gassen, S.V., Gaudilliere, B., Angst, M.S., Saeys, Y., Aghaeepour, N.: Cytonorm: A normalization algorithm for cytometry data. Cytometry Part A **97**, 268–278 (2020). https://doi.org/10.1002/cyto.a.23904

[9] Lotfollahi, M., Wolf, F.A., Theis, F.J.: scgen predicts single-cell perturbation responses. Nature Methods **16**, 715–721 (2019). https://doi.org/10.1038/s41592-019-0494-8

[10] Louizos, C., Swersky, K., Li, Y., Welling, M., Zemel, R.: The Variational Fair Autoencoder. arXiv (2015). https://doi.org/10.48550/ARXIV.1511.00830. https://arxiv.org/abs/1511.00830

[11] Nazábal, A., Olmos, P.M., Ghahramani, Z., Valera, I.: Handling incomplete heterogeneous data using vaes. Pattern Recognition **107**, 107501 (2020). https://doi.org/10.1016/j.patcog.2020.107501

[12] Becht, E., Tolstrup, D., Dutertre, C.-A., Morawski, P.A., Campbell, D.J., Ginhoux, F., Newell, E.W., Gottardo, R., Headley, M.B.: High-throughput single-cell quantification of hundreds of proteins using conventional flow cytometry and machine learning. Science Advances **7**(39), 0505 (2021) https://www.science.org/doi/pdf/10.1126/sciadv.abg0505. https://doi.org/10.1126/sciadv.abg0505

[13] Johnson, W.E., Li, C., Rabinovic, A.: Adjusting batch effects in microarray expression data using empirical Bayes methods. Biostatistics **8**(1), 118–127 (2006) https://academic.oup.com/biostatistics/article-pdf/8/1/118/25435561/kxj037.pdf. https://doi.org/10.1093/biostatistics/kxj037

[14] Van Gassen, S., Callebaut, B., Van Helden, M.J., Lambrecht, B.N., Demeester, P., Dhaene, T., Saeys, Y.: Flowsom: Using self-organizing maps for visualization and interpretation of cytometry data. Cytometry Part A **87**, 636–645 (2015). https://doi.org/10.1002/cyto.a.22625

[15] Yang, K.D., Belyaeva, A., Venkatachalapathy, S., Damodaran, K., Katcoff, A., Radhakrishnan, A., Shivashankar, G.V., Uhler, C.: Multi-domain translation between single-cell imaging and sequencing data using autoencoders. Nature Communications 2021 12:1 **12**, 1–10 (2021). https://doi.org/10.1038/s41467-020-20249-2

[16] Lotfollahi, M., Naghipourfar, M., Theis, F.J., Wolf, F.A.: Conditional out-of-distribution generation for unpaired data using transfer vae. Bioinformatics **36**, 610–617 (2020). https://doi.org/10.1093/bioinformatics/btaa800

[17] Sandha, S.S., Aggarwal, M., Fedorov, I., Srivastava, M.: Mango: A python library for parallel hyperparameter tuning. In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3987–3991 (2020). IEEE

[18] Korsunsky, I., Millard, N., Fan, J., Slowikowski, K., Zhang, F., Wei, K., Baglaenko, Y., Brenner, M., Loh, P.-r., Raychaudhuri, S.: Fast, sensitive and accurate integration of single-cell data with harmony. Nature Methods **16**, 1289–1296 (2019). https://doi.org/10.1038/s41592-019-0619-0

[19] Sohn, K., Lee, H., Yan, X.: Learning structured output representation using deep conditional generative models. In: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 28. Curran Associates, Inc., ??? (2015). https://proceedings.neurips.cc/paper$_{f}iles/paper/2015/file/8d55a249e6baa5c0677229$ $Paper.pdf$

[20] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., Lerchner, A.: beta-VAE: Learning basic visual concepts with a constrained variational framework. In: International Conference on Learning Representations (2017). https://openreview.net/forum?id=Sy2fzU9gl

[21] Mann, E.R., Menon, M., Knight, S.B., Konkel, J.E., Jagger, C., Shaw, T.N., Krishnan, S., Rattray, M., Ustianowski, A., Bakerly, N.D., Dark, P., Lord, G.M., Simpson, A., Felton, T., Ho, L.-P., TRC, N.R., Feldmann, M., CIRCO, Grainger, J.R., Hussell, T.: Longitudinal immune profiling reveals key myeloid signatures associated with covid-19. Science Immunology **5**(51), 6197 (2020) https://www.science.org/doi/pdf/10.1126/sciimmunol.abd6197. https://doi.org/10.1126/sciimmunol.abd6197

[22] Ashland, D. OR: Becton, Company: Flowjo™ software (2021)

[23] Amir, E.-a.D., Davis, K.L., Tadmor, M.D., Simonds, E.F., Levine, J.H., Bendall, S.C., Shenfeld, D.K., Krishnaswamy, S., Nolan, G.P., Pe'er, D.:

visne enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia. Nature Biotechnology **31**, 545–552 (2013). https://doi.org/10.1038/nbt.2594

[24] Mathew, D., Giles, J.R., Baxter, A.E., Oldridge, D.A., Greenplate, A.R., Wu, J.E., Alanio, C., Kuri-Cervantes, L., Pampena, M.B., D'Andrea, K., Manne, S., Chen, Z., Huang, Y.J., Reilly, J.P., Weisman, A.R., Ittner, C.A.G., Kuthuru, O., Dougherty, J., Nzingha, K., Han, N., Kim, J., Pattekar, A., Goodwin, E.C., Anderson, E.M., Weirick, M.E., Gouma, S., Arevalo, C.P., Bolton, M.J., Chen, F., Lacey, S.F., Ramage, H., Cherry, S., Hensley, S.E., Apostolidis, S.A., Huang, A.C., Vella, L.A., null null, Betts, M.R., Meyer, N.J., Wherry, E.J., Alam, Z., Addison, M.M., Byrne, K.T., Chandra, A., Descamps, H.C., Kaminskiy, Y., Hamilton, J.T., Noll, J.H., Omran, D.K., Perkey, E., Prager, E.M., Pueschl, D., Shah, J.B., Shilan, J.S., Vanderbeck, A.N.: Deep immune profiling of covid-19 patients reveals distinct immunotypes with therapeutic implications. Science **369**(6508), 8511 (2020) https://www.science.org/doi/pdf/10.1126/science.abc8511. https://doi.org/10.1126/science.abc8511

[25] Finak, G., Frelinger, J., Jiang, W., Newell, E.W., Ramey, J., Davis, M.M., Kalams, S.A., De Rosa, S.C., Gottardo, R.: Opencyto: An open source infrastructure for scalable, robust, reproducible, and automated, end-to-end flow cytometry data analysis. PLoS Computational Biology **10**, 1003806 (2014). https://doi.org/10.1371/journal.pcbi.1003806

[26] R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2022). R Foundation for Statistical Computing. https://www.R-project.org/

[27] Parks, D.R., Roederer, M., Moore, W.A.: A new "logicle" display method avoids deceptive effects of logarithmic scaling for low signals and compensated data. Cytometry Part A **69A**, 541–551 (2006). https://doi.org/10.1002/cyto.a.20258

[28] The HDF Group: Hierarchical Data Format Version 5. http://www.hdfgroup.org/HDF5

[29] Van Rossum, G., Drake, F.L.: Python 3 Reference Manual. CreateSpace, Scotts Valley, CA (2009)

[30] McKenna, E., Wubben, R., Isaza-Correa, J.M., Melo, A.M., Mhaonaigh, A.U., Conlon, N., O'Donnell, J.S., Ní Cheallaigh, C., Hurley, T., Stevenson, N.J., Little, M.A., Molloy, E.J.: Neutrophils in COVID-19: Not innocent bystanders. Front. Immunol. **13**, 864387 (2022)

[31] Wang, F., Nie, J., Wang, H., Zhao, Q., Xiong, Y., Deng, L., Song, S., Ma, Z., Mo, P., Zhang, Y.: Characteristics of Peripheral Lymphocyte Subset

Alteration in COVID-19 Pneumonia. The Journal of Infectious Diseases **221**(11), 1762–1769 (2020) https://academic.oup.com/jid/article-pdf/221/11/1762/33202327/jiaa150.pdf. https://doi.org/10.1093/infdis/jiaa150

[32] Diao, B., Wang, C., Tan, Y., Chen, X., Liu, Y., Ning, L., Chen, L., Li, M., Liu, Y., Wang, G., Yuan, Z., Feng, Z., Zhang, Y., Wu, Y., Chen, Y.: Reduction and functional exhaustion of t cells in patients with coronavirus disease 2019 (covid-19). Frontiers in Immunology **11** (2020). https://doi.org/10.3389/fimmu.2020.00827

[33] Du, Y., Tu, L., Zhu, P., Mu, M., Wang, R., Yang, P., Wang, X., Hu, C., Ping, R., Hu, P., et al.: Clinical features of 85 fatal cases of covid-19 from wuhan. a retrospective observational study. American Journal of Respiratory and Critical Care Medicine **201**(11), 1372–1379 (2020). https://doi.org/10.1164/rccm.202003-0543oc

[34] Hoffmann, J.J.M.L.: Neutrophil cd64: a diagnostic marker for infection and sepsis. Clinical Chemistry and Laboratory Medicine **47**(8), 903–916 (2009). https://doi.org/10.1515/CCLM.2009.224

[35] Karawajczyk, M., Douhan Håkansson, L., Lipcsey, M., Hultström, M., Pauksens, K., Frithiof, R., Larsson, A.: High expression of neutrophil and monocyte CD64 with simultaneous lack of upregulation of adhesion receptors CD11b, CD162, CD15, CD65 on neutrophils in severe COVID-19. Ther. Adv. Infect. Dis. **8**, 20499361211034065 (2021)

[36] Meghraoui-Kheddar, A., Chousterman, B.G., Guillou, N., Barone, S.M., Granjeaud, S., Vallet, H., Corneau, A., Guessous, K., de Roquetaillade, C., Boissonnas, A., Irish, J.M., Combadière, C.: Two new neutrophil subsets define a discriminating sepsis signature. Am. J. Respir. Crit. Care Med. **205**(1), 46–59 (2022)

[37] Rice, C.M., Lewis, P., Ponce-Garcia, F.M., Gibbs, W., Groves, S., Cela, D., Hamilton, F., Arnold, D., Hyams, C., Oliver, E., Barr, R., Goenka, A., Davidson, A., Wooldridge, L., Finn, A., Rivino, L., Amulic, B.: Hyperactive immature state and differential cxcr2 expression of neutrophils in severe covid-19. Life Science Alliance **6**(2) (2023) https://www.life-science-alliance.org/content/6/2/e202201658.full.pdf. https://doi.org/10.26508/lsa.202201658

[38] Schulte-Schrepping, J., Reusch, N., Paclik, D., Baßler, K., Schlickeiser, S., Zhang, B., Krämer, B., Krammer, T., Brumhard, S., Bonaguro, L., et al.: Severe covid-19 is marked by a dysregulated myeloid cell compartment. Cell **182**(6) (2020). https://doi.org/10.1016/j.cell.2020.08.001

[39] Hug, S., Bernhard, S., Stratmann, A.E.P., Erber, M., Wohlgemuth, L.,

Knapp, C.L., Bauer, J.M., Vidoni, L., Fauler, M., Föhr, K.J., Radermacher, P., Hoffmann, A., Huber-Lang, M., Messerer, D.A.C.: Activation of neutrophil granulocytes by platelet-activating factor is impaired during experimental sepsis. Front. Immunol. **12**, 642867 (2021)

[40] Finak, G., Jiang, W., Gottardo, R.: Cytoml for cross-platform cytometry data sharing. Cytometry Part A **93** (2018). https://doi.org/10.1002/cyto.a.23663

[41] Hahne, F., LeMeur, N., Brinkman, R.R., Ellis, B., Haaland, P., Sarkar, D., Spidlen, J., Strain, E., Gentleman, R.: flowcore: a bioconductor package for high throughput flow cytometry. BMC Bioinformatics **10** (2009). https://doi.org/10.1186/1471-2105-10-106

[42] Finak, G., Jiang, M.: flowworkspace: Infrastructure for representing and interacting with gated and ungated cytometry data sets. R package version 4.12.1 (2023)

[43] Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E.: Array programming with NumPy. Nature **585**(7825), 357–362 (2020). https://doi.org/10.1038/s41586-020-2649-2

[44] Chollet, F., et al.: Keras. https://keras.io (2015)