Challenges in Large Scale Proteomics Data Analysis: A Survey of Characterization and Correction Solutions for Batch Effects

Amir Alavi*, Harendra Guturu, Mahdi Zamanighomi, Tristan Brown, Jian Wang, Sam Cutler, Biao Li, Khatereh Motamedchaboki, Asim Siddiqui, and Serafim Batzoglou

Batch effects in large-scale proteomics analysis

Introduction

Recent advances in liquid chromatography mass spectrometry (LCMS)-based proteomics analysis have enabled the efficient profiling of thousands of proteins from single LCMS runs. The ability to run untargeted, high throughput LCMS experiments has opened the door to large-scale cohort studies for biomarker and drug target discovery[1,2]. When conducting large-scale cohort studies, technical confounding can be introduced as samples are run across different MS instruments, LC columns, dates, and geographic locations. In order to integrate these samples across datasets for joint analyses, one needs to both diagnose this batch effect and apply methods to correct for it.

Here we compare methods for characterizing batch effects in proteomics data. We have evaluated the presence of a batch effect using multiple batch effect diagnosis methods, including Principal Components Analysis-based approaches, local-neighborhood diversity measures, and machine learning classifier-based methods. Next, we benchmark batch effect correction methods for protein abundance data. These include traditional methods often used in proteomics and genomics as well as a novel deep learning-based batch correction method we developed.

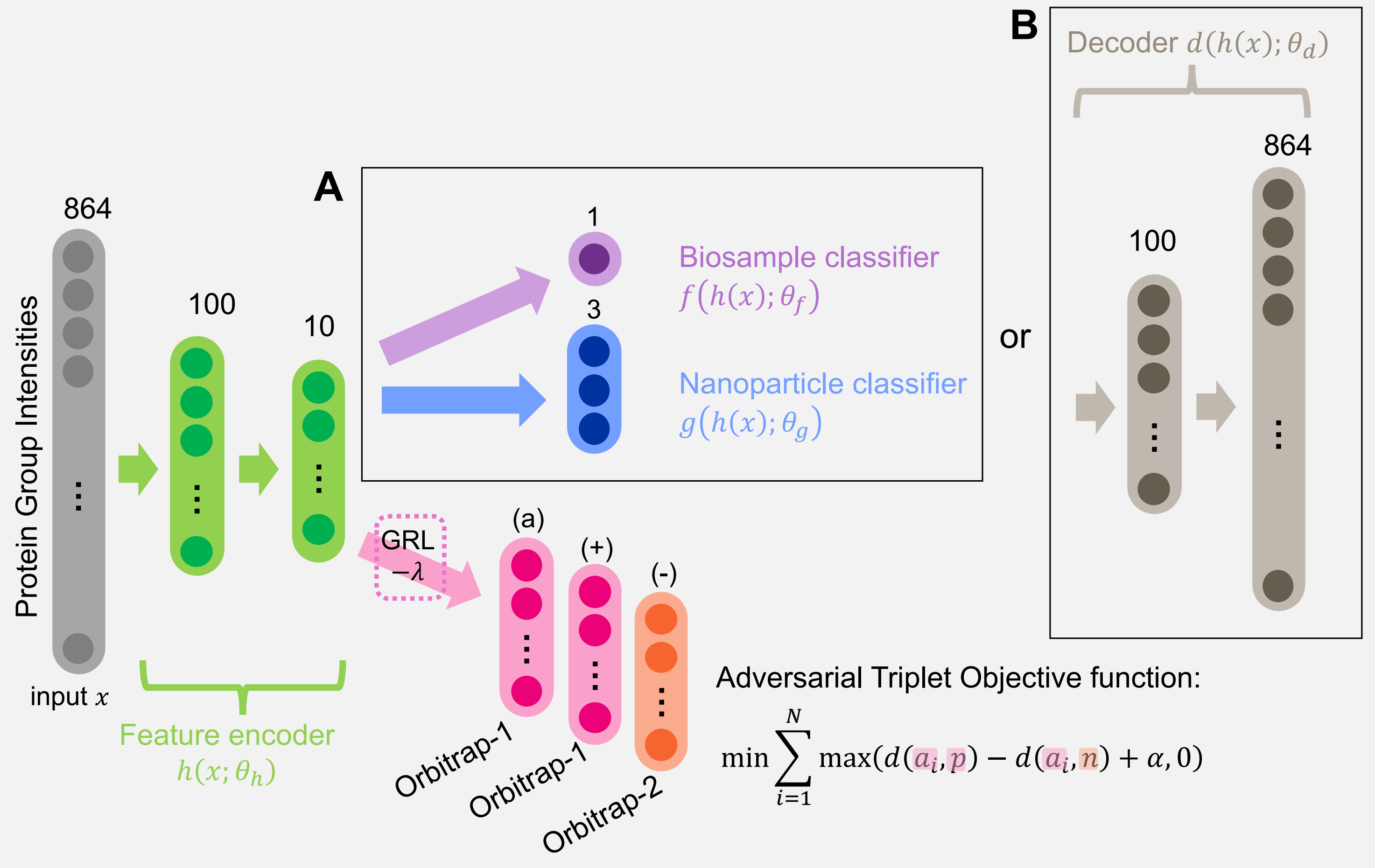
LCMS Data

We collect a batch-diverse dataset using Seer's Proteograph™ Product Suite. Our dataset includes 882 LCMS runs (DDA) across:

- Two types of control plasma samples
- Three Seer Proteograph nanoparticles (NPs)
- Three LCMS instruments
- Eight LC columns.

Methods

We applied each method on our dataset to produce batch-corrected representations, which we then evaluated with batch characterization metrics. We applied baseline methods that are traditionally used in proteomics or other omics data analysis pipelines like PCA, MSStats[3], and ComBat[4]; methods based on nearest neighbor matching like MNN[5] and Scanorama[6];and Harmony[7] which is an iterative clustering + translating algorithm; and deep learning-based approaches like scVI[8]. We then applied our developed, deep learning approach by extending Domain Adversarial Neural Networks[9] (DANN).



Adversarial Triplet Objective function:

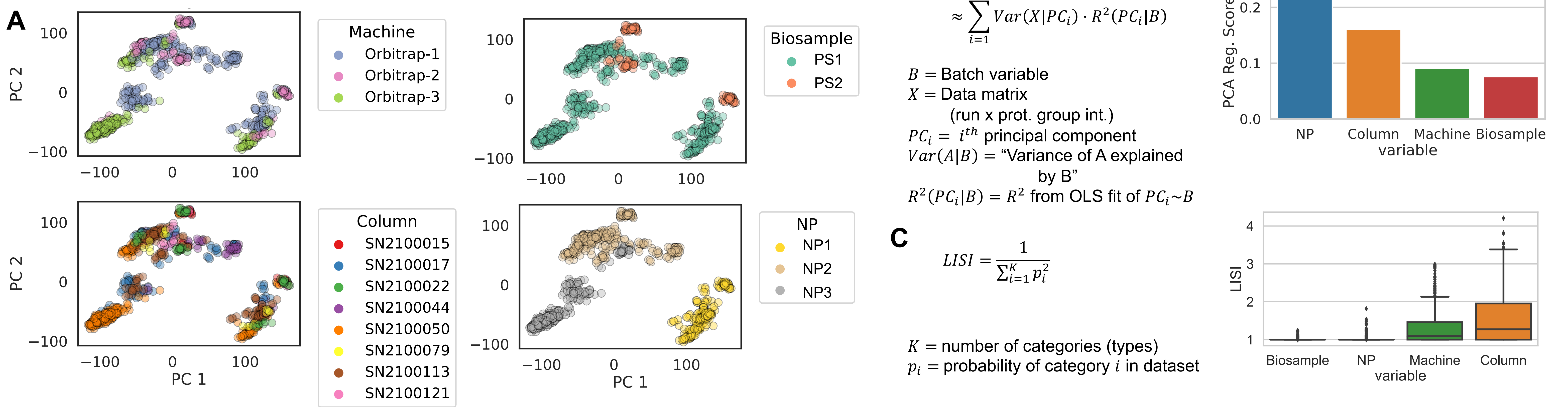
$$\min \sum_{i=1}^{N} \max(d(a_i, p) - d(a_i, n) + \alpha, 0)$$

Figure 1. Adversarial neural network architectures to learn batch-invariant representations.

Protein group intensity data is fed forward through a fully connected ReLU encoder stage (green), that is trained to perform poorly on a Triplet Loss which tries to discriminate technical batches. At the same time, this representation is trained to minimize either (A) two classification tasks as in DannClf, or (B) a reconstruction loss as in DannRecon.

Deep learning approaches to integrate large-scale datasets by correcting batch effects

Results



B

$$PCA\ Reg.Score = Var(X|B) \approx \sum_{i=1}^{G} Var(X|PC_i) \cdot R^2(PC_i|B)$$

B = Batch variable
X = Data matrix (run x prot. group int.)
PC_i = i^{th} principal component
$Var(A|B)$ = "Variance of A explained by B"
$R^2(PC_i|B)$ = R^2 from OLS fit of $PC_i \sim B$

C

$$LISI = \frac{1}{\sum_{i=1}^{K} p_i^2}$$

K = number of categories (types)
p_i = probability of category i in dataset

Figure 2. Characterization of batch effects in the data.

A) PCA embeddings of protein group log intensities of each run, colored by four different covariates. **B)** Principal Components Regression[10], showing that batch variables (LC column and MS Instruments) are major contributors to the variance in the data, over analysis of these control plasma samples. **C)** Local Inverse Simpson's Index[7] (LISI) score, measuring effective diversity of a label within small neighborhoods, which shows low levels of integration of batch variables. While **(B)** shows where signal resides in a data matrix, **(C)** shows the level of mixing, and is better suited for comparing batch effect correction methods.

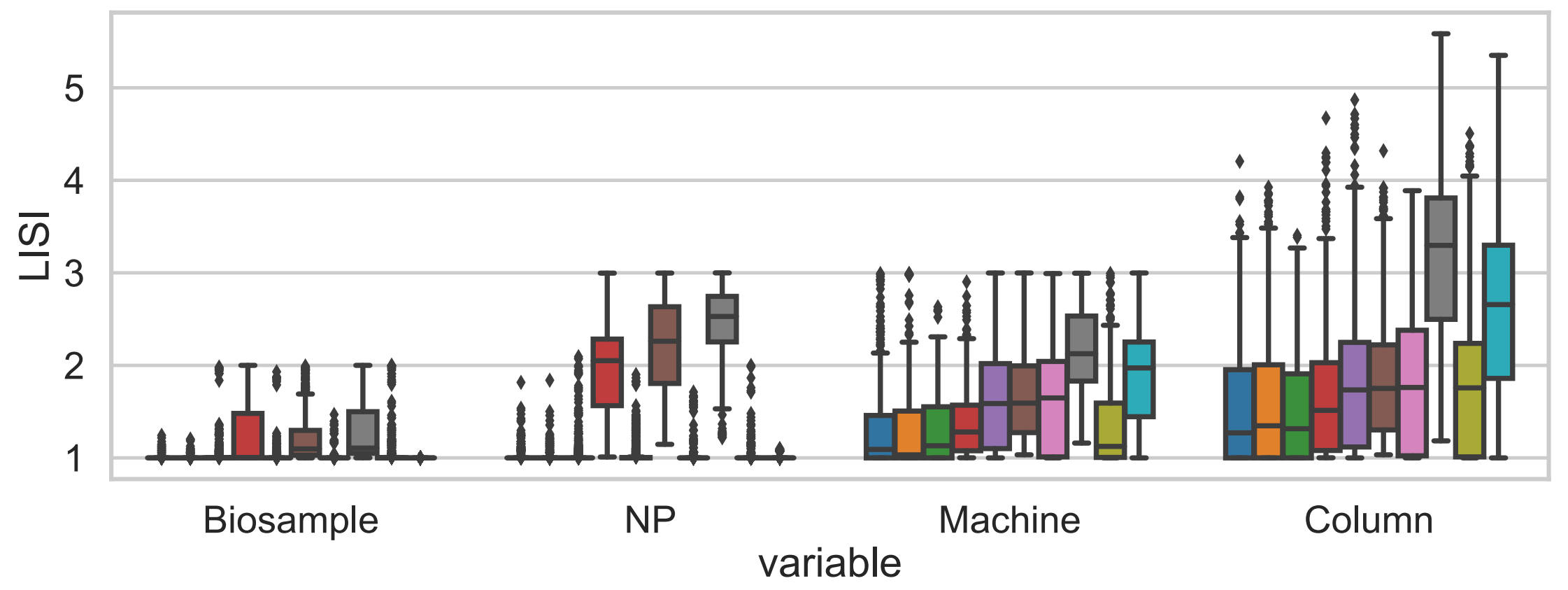


Figure 3. Dataset mixing and biological signal preservation of batch effect correction methods.

LISI scores of all correction methods we applied. Though Scanorama mixed the best with respect to Machine and Column, it overmixed the biological variables. DannClf and DannRecon do not have this issue and are able to mix the technical variables while preserving the distinction between biological variables.

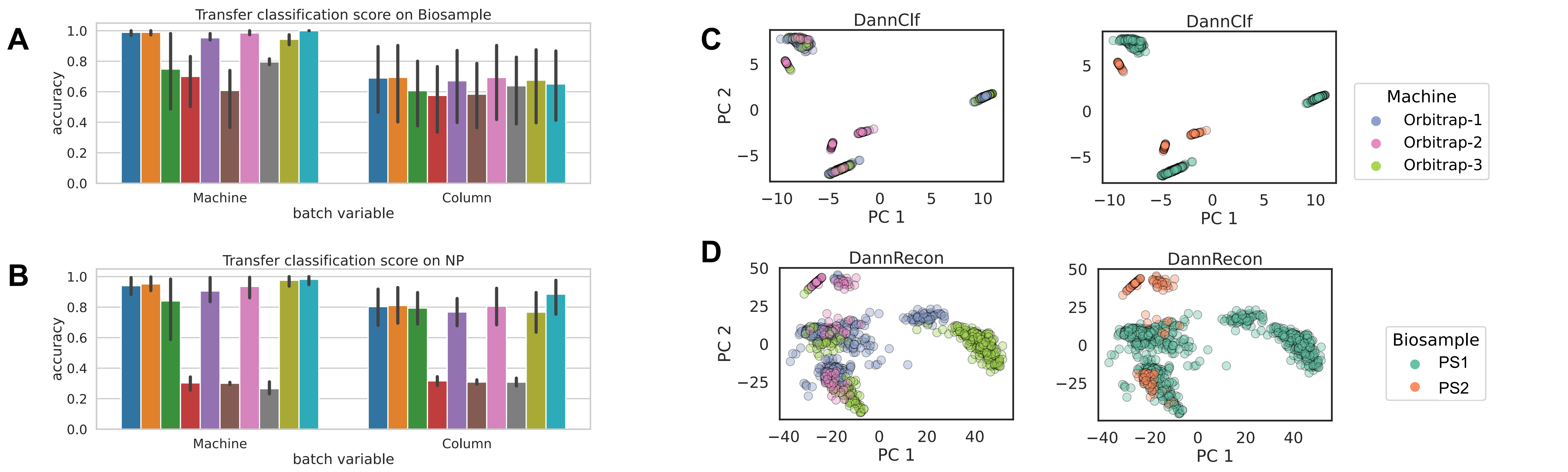


Figure 4. Quantitative and qualitative assessment of batch corrected representations for downstream tasks.

A) Comparison of using batch corrected representations for classifying biological phenotypes across batches. For example: for MS Instruments (Machine), a KNN classifier is trained on Orbitrap-1 and Orbitrap-2 data to classify between the two control plasma samples (PS1 or PS2), and the test accuracy is assessed on Orbitrap-3. This is repeated for testing on Orbitrap-1 and Orbitrap-2. Note, this same procedure is repeated for Column. **B)** The same but the prediction task is to classify amongst the three Nanoparticles. **C)** PCA embeddings of the learned features from our DannClf and D) DannRecon models.

Conclusion

- Batch effects can contribute to a large amount of the noise in large-scale proteomics datasets, compared to biological variations. We have observed a large batch effect attributed to mass spectrometers and LC columns in our dataset.
- Deep learning-based approaches can learn to integrate diverse proteomics datasets well. Our extension of DANN can harmonize data across technical factors, while maintaining the fidelity of the biological signal in the data.
- While DannClf can harmonize the data well, the representations it learns are most useful for classification. Future work such as our unsupervised variant, DannRecon, may learn more general-purpose batch corrected representations.

References

[1] Blume et al. Nat. Comm. (2020)
[2] Ferdosi et al. PNAS (2022)
[3] Choi et al. Bioinformatics (2014)
[4] Johnson et al. Biostatistics. (2007)
[5] Haghverdi et al. Nat. Biotechnol. (2018)
[6] Hie et al. Nat. Biotechnol. (2019)
[7] Korsunsky et al. Nat. Methods (2019)
[8] Lopez et al. Nat. Methods (2018)
[9] Ganin et al. J. Mach. Learn. Res. (2016)
[10] Büttner et al. Nat. Methods (2019)

Publications