Self-Supervised Representation Learning on RxRx1: A Systematic Study of Batch Effect Mitigation

Matteo Lugli Nicola Morelli Omar Carpentiero

June 10, 2025

Abstract

Self-supervised learning offers a promising approach for biomedical imaging where labeled data is scarce, but cellular imaging datasets like RxRx1 present unique challenges due to severe batch effects that can dominate learned representations. We conduct a systematic study of self-supervised methods on RxRx1, focusing on mitigating batch effects from experimental variations across plates, time points, and microscopes. We investigate SimCLR and DINO frameworks with novel batch-aware adaptations. For SimCLR, we introduce Contrastive Batch Decorrelation Loss and adversarial batch invariance to reduce batch-specific information. For DINO, we explore cross-domain learning strategies and simplified augmentation schemes tailored to multi-channel fluorescence microscopy. Our experiments show that standard self-supervised approaches cluster samples by experimental batch rather than biological perturbation. Through careful adaptation of training objectives, we achieve improved downstream classification performance while maintaining unsupervised learning principles. We demonstrate that simplified augmentations enable more stable training and that combining DINO with Barlow Twins loss provides additional regularization benefits, offering practical solutions for biomedical imaging applications with strong batch effects.

1 Introduction

Cellular imaging has become a vital tool in biomedical research, allowing large-scale studies of the cellular response to a wide range of conditions. Traditional supervised learning methods require large amounts of labeled data, which is expensive and time consuming to obtain. Self-supervision frameworks emerge as a powerful alternative, providing good pretrain embeddings in the absence of labeled data.

The RxRx1 dataset[1] is one of the largest collections of fluorescence microscopy images, capturing cell morphologies under thousands of perturbations. However, the presence of strong batch effects, variations introduced by differences in imaging conditions, presents a significant obstacle for machine learning models, especially in the case of unsupervised or self-supervised learning (SSL). Those effects often overshadow biologically relevant signals, making traditional self-supervision and clustering techniques ineffective. In this work, we propose several self-supervised learning frameworks based on SimCLR and DINO, with a specific focus on mitigating batch effects. We conduct an in-depth investigation into which standard training practices within these frameworks should be adapted to better address batch-related variability.

2 Related work

Self-supervised learning (SSL) has recently gained significant traction in the biomedical imaging domain, particularly in cellular imaging, where the scarcity of annotated data poses a major challenge. By learning representations without relying on manual labels, SSL offers

a promising alternative to traditional supervised approaches. However, a common limitation of SSL methods in this context is their vulnerability to batch effects—systematic variations introduced by differences in experimental conditions, such as changes in imaging equipment, reagents, or sample preparation. These batch effects can dominate the learned representations, leading the model to capture experimental artifacts rather than biologically meaningful variations, thereby severely limiting generalization across datasets or experimental runs.

A notable recent study that attempts to address this problem is Metadata-Guided Consistency Learning [5]. In this work, the authors apply popular SSL methods, including DINO [6] and BYOL [4], to the RxRx1-HUVEC dataset, a challenging cellular imaging benchmark known for its pronounced batch effects. Their analysis reveals that standard SSL techniques often produce feature embeddings that are strongly influenced by batch-specific characteristics, instead of relying on the biological signal of interest. To address this, they proposed Cross-Domain Consistency Learning (CDCL), a framework that enforces consistency between features extracted from different batches by utilizing metadata annotations such as experiment ID, plate number, or treatment type. Although CDCL demonstrates notable improvements in representation quality and model performance, it introduces a degree of supervision by relying on label-based metadata when constructing training batches. This reliance partially compromises the foundational principle of self-supervised learning, which is to learn representations without using any form of annotation.

3 Dataset

Each image in RxRx1 is a multi-channel fluorescent microscopy snapshot containing six channels, each corresponding to different cellular components or structures, such as the nucleus, cytoplasm, endoplasmic reticulum, and mitochondria. These fluorescence stains collectively provide rich morphological detail, enabling nuanced discrimination between subtle phenotypic changes. A key challenge posed by RxRx1 lies in its significant batch effects, introduced by variability in experimental conditions across plates, time points, microscopes, and handling procedures. These effects often obscure the true biological signal and can lead to models overfitting to nuisance variations rather than generalizing to novel perturbations. For example, the visual similarity between samples from the same experimental batch may dominate the learned representations, even when the underlying perturbations differ (see Figure 1). To better understand and mitigate the impact of batch effects, we explored two versions of the dataset:

- **3-channel version**: A compressed variant retaining only the three most informative fluorescence channels, typically excluding the last three which contribute minimally to classification accuracy. This version facilitates faster training and reduces computational cost while retaining sufficient discriminative power.
- **6-channel version**: The full-resolution dataset preserving all six fluorescence channels, offering the richest morphological detail and serving as the reference baseline for model evaluation.

Both versions are publicly accessible through the dataset's official website. In this work, we introduce a set of terms to refer to specific subsets derived from the original datasets, based on different organizational criteria:

- **Full**: refers to the complete dataset, using the original train, validation, and test splits.
- **Stratified**: Consists of the 24 experiments associated with the HUVEC cell type, from which we create an 80/20 stratified train-validation split. In this setting, the training set includes samples from all the 24 experiments.
- **Cross validation**: Includes only HUVEC experiments, split into 16 for training, 4 for validation, and 4 for testing, following a 6-fold cross-validation approach.

- **F0**: Denotes the first fold of the Cross Validation setup. Due to computational constraints, this subset was used for our ablation studies.

To help clarify the dataset structure, we provide an illustration in Figure 2.

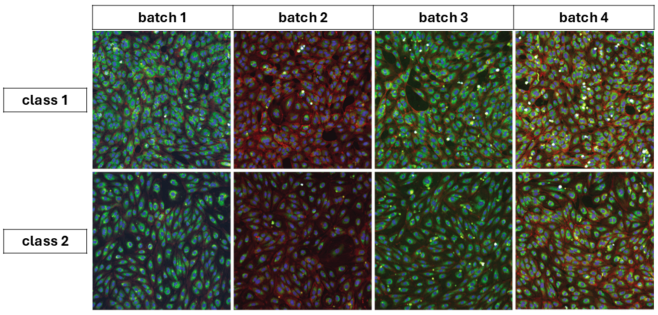


Figure 1: Images of two different genetic conditions (rows) in HUVEC cells across four experimental batches (columns). Notice the visual similarities of images from the same batch.

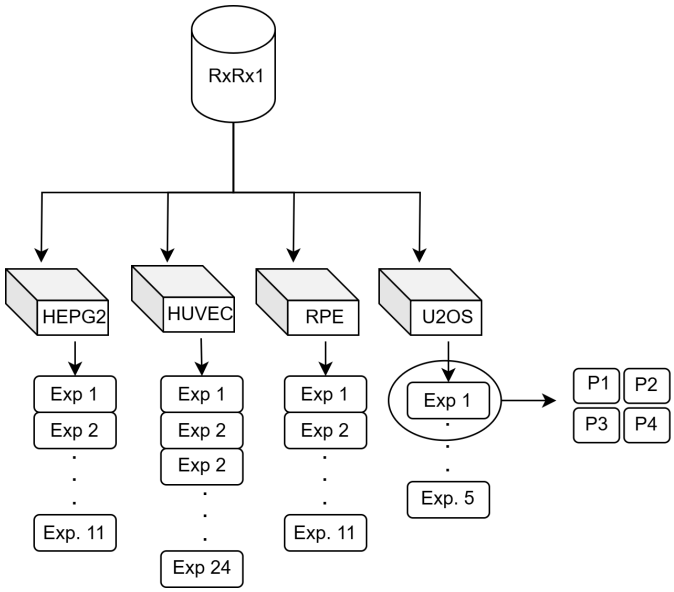


Figure 2: Overview of the RxRx1 dataset structure. The dataset is organized by cell type (HEPG2, HUVEC, RPE, U2OS), each containing multiple experiments. In total, there are 24 experiments for HUVEC and fewer for the other cell types. Each experiment consists of four plates (P1–P4), each containing multiple wells in which siRNAs were introduced to induce specific perturbations. For each well, two non-overlapping image fields are acquired. In total, each experiment comprises 2,464 samples.

3.1 Dataset Limitations

The data distribution in the RxRx1 dataset makes the task unfeasible for standard training pipelines. While models are required to learn and correct for batch effects, the available information is unevenly distributed across experiments. Each of the 1139 classes has between 94 and 492 samples, spread over 51 experiments. However, not every class is present in every

experiment; each one contains between 0 and 10 samples per class, with a global average of 2. This results in many siRNAs being entirely absent from certain batches, complicating batch effect correction. Such imbalance increases the risk of overfitting to specific batches or classes while leaving others unseen, especially the least common, which degrades the model's ability to generalize. Future studies could implement dataset restrictions by removing 31 siRNAs (along with the EMPTY class) to create better balance across batches. Moreover, restricting the number of experiments to only one cell type, HUVEC, the most common in the whole dataset, proved to be necessary in our approach since multiple cell types introduced another severe, and hard to learn (due to the unusual imbalance) batch effect. Since multiple batches of the data essentially act as outliers with respect to the rest of the dataset, self supervision proved very challenging.

4 Methods

Section 4.1 outlines the preprocessing techniques employed to prepare the samples for analysis. Sections 4.2 and 4.3 detail the strategies developed to adapt SimCLR to contexts characterized by pronounced batch effects. Subsequently, Sections 4.4 and 4.5 describe the application of DINO and its cross-domain variant within the proposed framework. Moreover, in sections 4.6 and 4.7 we briefly introduce Barlow Loss and the Multi Centering techniques.

4.1 Preprocessing

Starting from the original Cx512x512, the images were fed to the models after being downsampled to Cx256x256. To partially address batch effects, we applied experiment-wise normalization by processing each experiment separately. For each sample, we subtracted the mean and divided by the standard deviation within its corresponding experimental batch. This approach reduces batch-specific noise and helps ensure that the learned representations captures biological signals. Dataset statistics were calculated beforehand. In contrast, plate-wise normalization—despite its simplicity—had no positive impact on model performance. Table 1 presents the mean and standard deviation of the channel values for two representative experiments.

Table 1: Mean and variance of pixel intensities for each channel in two experiments, showing the diversity of the two distributions.

Experiment	**Ch 1**	**Ch 2**	**Ch 3**	**Ch 4**	**Ch 5**	**Ch 6**
			Mean			
HEPG2-01	38.15	26.58	35.02	40.29	44.27	31.06
HEPG2-05	3.75	9.55	6.50	6.06	2.44	5.12
			Standard Deviation			
HEPG2-01	39.86	15.30	25.56	24.06	41.10	20.23
HEPG2-05	7.04	10.93	5.84	6.97	3.06	4.68

4.2 SimCLR Baseline

As a baseline for our self-supervised learning experiments on the RxRx1 dataset[1], we adopted the SimCLR framework [2], a contrastive learning approach that learns image representations without the need for explicit labels. The core idea of SimCLR is to train an encoder network to maximize agreement between different augmented views of the same image, using a contrastive loss function. Given an image, SimCLR applies two independent stochastic data augmentations to generate two correlated views, forming a positive pair. A large number of such pairs are sampled per mini-batch. Each image is passed through a

shared encoder network (in our case, various ResNet[7] and ViT[3] architectures), followed by a projection head—typically a small multilayer perceptron (MLP)—which maps the encoded features into a latent space where contrastive loss is applied. The model is trained using the Normalized Temperature-scaled Cross Entropy Loss (NT-Xent), which encourages representations of positive pairs (augmented views of the same image) to be similar, while simultaneously pushing apart representations of all other images in the batch (considered negative samples). Formally, given a batch of N image pairs (resulting in $2N$ images), for each positive pair (i, j), the loss for a single pair as is in Eq. (1).

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(z_i, z_k)/\tau)} \tag{1}$$

where z_i and z_j are the ℓ_2-normalized projection vectors, $\text{sim}(\cdot, \cdot)$ is the cosine similarity, and τ is a temperature hyperparameter controlling the sharpness of the distribution. To adapt SimCLR to the RxRx1 dataset, which consists of high-resolution multi-channel fluorescent microscopy images, we designed a set of domain-aware augmentations. These included random cropping and resizing, horizontal flipping, color jitter, and Gaussian noise, which served to generate sufficiently distinct views of the same underlying biological sample. Such augmentations are crucial in self-supervised contrastive learning, as they define the invariances the model will learn. We experimented with a range of encoder architectures, including ResNet-18, ResNet-50, and ResNet-152 [7], as well as ViT-Tiny and ViT-Small [3], to explore how model capacity and architecture influence representation learning in this domain. Due to computational constraints, we were limited to a maximum batch size of 256 images, which slightly reduced the number of negative samples per update step, a factor known to influence contrastive learning performance [2]. Overall, SimCLR provided the baseline for evaluating the effectiveness of more advanced or tailored self-supervised learning techniques applied to the RxRx1 dataset.

4.3 Batch Invariance Experiments

Building upon the SimCLR baseline, we designed two experiments aimed at reducing the influence of experimental batch effects, one of the major confounding factors in the RxRx1 dataset. These experiments sought to enforce batch-invariant representations, either through a modified loss function or an adversarial training scheme. In Section 4.3.1, we introduce a Contrastive Batch Decorrelation Loss to explicitly reduce batch-specific information by altering the contrastive objective to treat inter-batch pairs as positives. Then, in Section 4.3.2, we propose an Adversarial Batch Invariance approach, in which a batch classifier adversary is trained to identify the experimental batch, while the encoder is simultaneously trained to fool it, effectively removing batch-discriminative features from the learned representations.

4.3.1 Contrastive Batch Decorrelation Loss

To reduce the influence of batch-specific artifacts in the learned representations, we introduced an additional loss term that encourages the model to group together samples from different experimental batches while separating those from the same batch. This objective was designed to complement the standard SimCLR contrastive loss, and was applied jointly during training. Concretely, while the standard SimCLR framework defines positive pairs as different augmented views of the same image and negative pairs as all other samples in the batch, we constructed an auxiliary contrastive loss by inverting this idea with respect to batch identity. In our formulation, samples from *different batches* were treated as positives, and those from the *same batch* as negatives. The aim was to train the encoder to ignore batch effects and focus on the underlying signal. Formally, for a given anchor z_i and a positive z_j such that $B(i) \neq B(j)$, we defined the batch contrastive loss as for the former SimCLR loss (1) where z_i and z_j are the output projections from the encoder, $\text{sim}(\cdot, \cdot)$ denotes cosine similarity, τ is the temperature parameter, and $B(i)$ indicates the batch label of

sample i. The indicator function ensures that only samples from different batches contribute to the denominator. The final training objective combined this batch decorrelation loss with the standard SimCLR NT-Xent loss (2).

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{SimCLR}} + \lambda \cdot \mathcal{L}_{\text{batch}} \tag{2}$$

where λ is a tunable hyperparameter that balances the trade-off between learning meaningful features and removing batch-related information. This hybrid loss aimed to allow the model to preserve the instance-level discriminative power of SimCLR while gradually suppressing batch-identifying features, leading to more robust and generalizable embeddings.

4.3.2 Adversarial Batch Invariance

We also explored an adversarial strategy to remove batch-specific information from the learned representations. The core idea was to discourage the encoder from retaining any signal that could be used to predict the experimental batch from which a sample originated. To implement this, we introduced a batch classifier network trained to predict the batch label of a sample based on its encoded representation. The encoder, in turn, was trained to fool this classifier, thereby enforcing invariance to batch-related variation. The training was structured in an alternating fashion. First, the batch classifier was trained using standard cross-entropy loss while keeping the encoder frozen. Then, in the next phase, the encoder was updated while keeping the classifier frozen, with the goal of minimizing the standard SimCLR loss and maximizing the cross-entropy loss of the classifier. This adversarial signal was implemented by taking the negative of the cross-entropy, effectively pushing the encoder to remove features useful for batch discrimination. The total loss used to train the encoder was defined as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{SimCLR}} - \lambda \cdot \mathcal{L}_{\text{batch\_CE}} \tag{3}$$

where $\mathcal{L}_{\text{SimCLR}}$ is the standard contrastive loss as in Eq. (1), $\mathcal{L}_{\text{batch\_CE}}$ is the cross-entropy loss of the batch classifier, and λ is a hyperparameter controlling the strength of the adversarial signal. This adversarial training setup encouraged the encoder to discard features predictive of the batch, promoting the learning of embeddings that generalize across experimental variations.

4.4 DINO

DINO (Distillation with NO labels) is a self-supervised learning framework that leverages knowledge distillation to train powerful vision transformers (ViTs) without requiring labeled data. By encouraging consistency between two networks—a student and a teacher—on differently augmented views of the same image, DINO learns rich and transferable visual representations. Its architecture and training paradigm have demonstrated strong performance on various downstream tasks such as image classification, object detection, and semantic segmentation, making it a significant advancement in the field of unsupervised representation learning. To train a backbone using DINO, multiple views are generated for each sample in the minibatch—typically 2 global views and 8 local views, although these values can be treated as hyperparameters. The local views are smaller crops of the original image, while the global views cover larger areas of the same image. Both local and global views undergo a series of augmentations like Gaussian blur or color jitter and a final normalization; more details are provided in 5.2. After that, the global crops are forwarded through the teacher network, while all of the views are given to the student, obtaining G teacher features $\{t_1, \ldots, t_G\}$ and $G+L$ student features $\{s_1, \ldots, s_G, s_{G+1}, \ldots, s_{G+L}\}$. To avoid collapse, the teacher features are centered using a vector C defined as the exponential moving average of the teacher's mean activations over each mini-batch. Lastly, to obtain a probability distribution, softmax is applied: $p_i^t = \text{softmax}(t_i - C)$, $p_i^s = \text{softmax}(s_i)$. The loss is defined as

in (4):

$$\mathcal{L}_{\text{DINO}} = \sum_{i=1}^{G} \sum_{j=1, j\neq i}^{G+L} -p_i^t \log p_j^s \tag{4}$$

Teacher and student share the same architecture -a feature extractor and a projection head. During training, the learning signal is back-propagated only through the student, while the teacher is updated using exponential moving average. Unlike other contrastive methods such as SimCLR, DINO's loss function does not require computing pairwise similarities across all samples in a batch, making it more efficient in resource-constrained environments.

4.5 Cross Domain Learning

Even someone without expertise can easily notice that in images from the RxRx1 dataset (see Figure 1), the signal the model needs to learn to distinguish is much weaker than the batch effects. This is a major issue for the original DINO pipeline, which will quickly teach the model to simply cluster the samples based on batch effects. To the best of our knowledge the only way of effectively overcoming this problem is proposed in [5], in which the authors make use of the classification label during the training procedure. Specifically, during training, two images (instead of a single one) that have undergone the same treatment (and therefore share the same label), but come from different experiments (and thus exhibit different batch effects), are used to extract global and local views. This strategy (Cross Domain Learning or CDL) helps the model focus on the underlying biological signal rather than being misled by batch-specific variations. Figure 3 is a simplified graphical representation of the modified pipeline. Since CDL involves loading a new image for each sample in the mini-batch, the training incurs a significant memory and compute overhead.

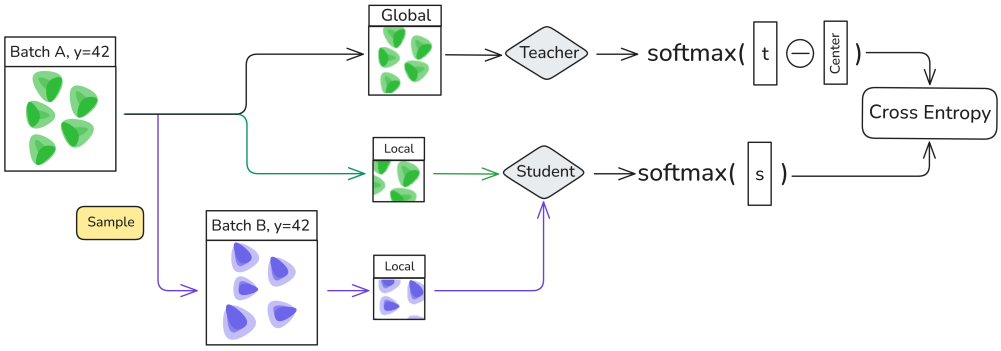


Figure 3: Simplification of the DINO training pipeline. Green arrows represent the standard DINO procedure, while the blue arrow is the variation introduced with Cross Domain Learning. The latter includes a extra sampling step, where a new image with the same label but from a different experimental batch is extracted.

4.6 Barlow Loss

Following the findings of [5], we paired the standard DINO Loss with the Barlow Twin Loss [10], defined in (5):

$$\mathcal{L}_{\text{B}} = \sum_i (1 - C_{ii}) + \lambda \sum_i \sum_{j\neq i} C_{ij} \tag{5}$$

where C is the cross correlation matrix computed across the mini-batch samples. Intuitively, since the loss encourages reducing the values off the main diagonal, the overall goal of the model becomes increasing the independence between the learned features. The cross-correlation matrix C is computed via the dot product of features extracted from each sample

in the mini-batch, necessitating a synchronization step when training across multiple GPUs. Our experiments indicate that this loss function is effective when applied to the HUVEC subset of the RxRx1 dataset with a relatively small batch size. However, it exhibited significant instability when the dataset was scaled, while the mini-batch size remained unchanged.

4.7 Multi centering

The multi-centering technique, as proposed in [5], extends the standard DINO framework by maintaining a distinct center for each batch or experimental group. This approach aims to improve the stability of the training process in settings characterized by high inter-batch variability, such as those affected by strong batch effects. Instead of computing a single global center across all samples, the method computes and updates a separate center for each batch, allowing the model to better accommodate distributional shifts across experiments. In our implementation, we followed the procedure described in [5] and applied multi-centering in combination with our simplified augmentation pipeline. However, our empirical results indicate that this technique does not lead to a consistent improvement in downstream task performance. Specifically, we observed that while training dynamics appeared more stable, the final representations did not yield higher accuracy compared to the baseline DINO model without multi-centering. More details are presented in section 5.

5 Experiments

The training pipeline is divided into two stages. In the first stage, we independently pretrain a feature extractor using either SimCLR or DINO, exploring both self-supervised learning approaches separately. In the second stage, a classification head is trained for the downstream task of perturbation classification. This task is particularly challenging due to the presence of 1,139 distinct classes with minimal inter-class variation. Although not entirely accurate, we refer to this second stage as 'fine-tuning' for simplicity. In Sections 5.1 and 5.2 we present the technical details and training configurations that proved most effective for training our models within constrained computational budgets. These settings were identified through targeted experimentation rather than systematic grid search. As our setup operates with significantly fewer capabilities than those used in [6] and [5], we do not aim to match their downstream performance. Nonetheless, we believe this work provides valuable insight by isolating and analyzing the key components of contrastive and self-supervised training frameworks, clarifying their individual contributions to overall model behavior.

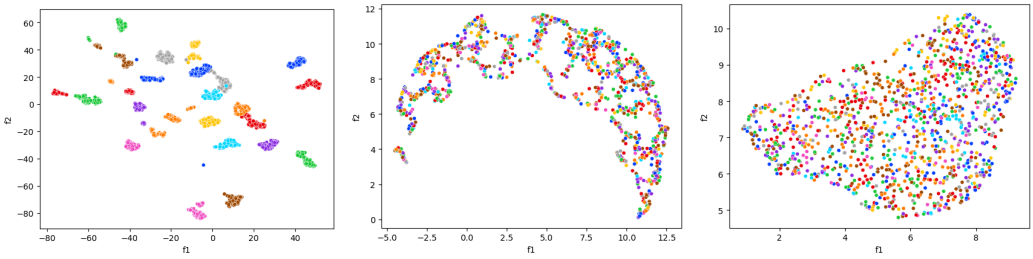
5.1 SimCLR

In this section, we evaluate the performance of the SimCLR framework on the RxRx1 dataset, following the approaches outlined in Sections 4.2 and 4.3. The backbone employed is a ViT-S/16 architecture, pretrained using a contrastive learning objective on domain-specific augmentations. These augmentations include random resized cropping, horizontal and vertical flipping, channel shuffle, grayscale conversion and random color jittering operation applied across the six input channels. All images are resized to 224×224 resolution. To evaluate the learned representations, we train a linear classification head using perturbation labels, while keeping the backbone weights frozen. The evaluation is performed on the F0 evaluation set containing unseen batch effects during training phase. Table 2 summarizes the downstream classification accuracy across various SimCLR configurations. In addition to the baseline model, we report results for two variants that integrate explicit mechanisms to reduce batch sensitivity: one using the Contrastive Batch Decorrelation Loss (CBDL), and another employing an adversarial batch-invariance training strategy.

Table 2: Downstream accuracy of different SimCLR configurations on the RxRx1 dataset. All results are computed on the F0 validation set.

Backbone	CBDL	Adversarial	Accuracy
ViT-S/16	×	×	0.035
ViT-S/16	✓	×	0.04
ViT-S/16	×	✓	0.042

UMAP Visualizations. To qualitatively assess the representations, we visualize the embeddings via UMAP. Figure 4 displays the 2D projections of features extracted from each configuration. In the baseline case, embeddings show clear clustering aligned with experimental batches, indicating that batch-specific artifacts dominate the representation space. In contrast, both the CBDL and adversarial variants result in more entangled clusters, suggesting improved invariance to batch effects. Due to the large number of classes (1,139), no distinct class-wise structures are visually separable.



(A) **(B)** **(C)**

Figure 4: UMAP projections of SimCLR-trained embeddings on the F0 validation set: (A) Baseline, (B) with CBDL, (C) with Adversarial Invariance.

These results confirm that contrastive self-supervised learning with standard SimCLR is highly sensitive to batch effects in the RxRx1 dataset. Incorporating strategies to enforce batch invariance improves generalization, but the model still underperforms compared to transformer-based methods such as DINO.

5.2 DINO

We use a dual-network architecture comprising a student and a teacher network, both initialized with a ViT-S/16 (21M parameters) backbone and identical weights. The teacher is updated using an exponential moving average (EMA) of the student parameters:

$$\theta_t \leftarrow \tau\theta_t + (1-\tau)\theta_s \tag{6}$$

with τ following a cosine schedule from 0.996 to 1.0. Local and global views are obtained by first downsampling the original 512×512 images to 256×256, followed by random cropping to generate 224×224 global views and 96×96 local views. Training proved highly sensitive to the choice of view augmentations, with suboptimal configurations often leading to representation collapse. To stabilize learning, we adopt a minimal augmentation strategy consisting of random cropping, horizontal flipping, and Gaussian blur. Unlike the original DINO setup, we exclude random grayscale, color jittering, and solarization. Given the inherent complexity and low-level variability of our images, we hypothesize that overly aggressive augmentations may degrade the quality of the learned representations rather than improve them. In this context, the augmentation pipeline should be treated as a critical set of hyperparameters;

however, its impact on pretraining performance remains relatively underexplored in the literature that has been reviewed for this work. Table 3 shows the differences between the augmentations used in original DINO, in [5] and in our work. Additionally, we experimented with two input configurations: one using only the first three channels and another using all six channels. In the latter case, non-geometric augmentations were applied independently to the first three and the last three channels. Optimization is performed with AdamW [8],

Table 3: Comparison of View Augmentations: Original DINO vs. Haslum et al. vs. Ours

Augmentation	**Original DINO**	**Haslum et al.**	**Ours**
Global view scale	$[0.4, 1.0]$	$[0.9, 1.0]$	$[0.9, 1.0]$
Local view scale	$[0.05, 0.4]$	$[0.3, 0.5]$	$[0.3, 0.5]$
Random Rotate (90°)	×	✓	✓
Random Vertical Flip	×	✓	×
Random Horizontal Flip	✓	✓	✓
Color Jittering	✓	✓	×
Random Grayscale	✓	×	×
Gaussian Blur	✓	×	✓
Solarization	✓	×	×

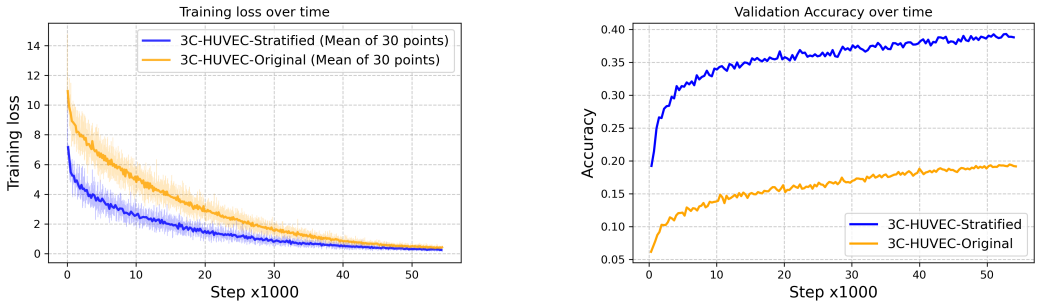
with a maximum learning rate of $5e^{-4}$ which scales linearly for 10 epochs then is decreased with cosine annealing, and a weight decay of $4e^{-3}$. We employ fine-grained learning rate scheduling and perform a step at every iteration. Due to resource limitations, the majority of experiments are performed on a single node with a single GPU. To approximate the performance of larger-scale setups, we employ gradient accumulation over 8 steps with a per-GPU batch size of 64, resulting in an effective batch size of 512. Employing gradient accumulation in place of a larger batch size per GPU reduces the stability of the Barlow Loss, as the cross-correlation matrix is computed on only 64 samples. We found that 100 pretraining epochs are enough to achieve convergence with our setup. The loss function is a weighted average of the standard dino loss (4) and the barlow loss (5):

$$\mathcal{L} = 0.25 \times \mathcal{L}_{\text{DINO}} + 0.75 \times \mathcal{L}_{BARLOW} \tag{7}$$

After the pretraining stage, we train a classification head in a supervised setting using treatment labels as ground truth, leaving the backbone weights frozen. Following the original DINO design, we remove the DINO projection head and attach our classification head directly to the ViT encoder. This way, the `[cls]` token output by the ViT can be forwarded as input to the new classification head. We use a batch size of 256, AdamW as the optimizer with a maximum learning rate of $5e^{-4}$ with no warmup and default polynomial decay fine-grained scheduling. The loss function in this case is a simple Cross Entropy Loss between the output probability distribution and the ground truth.

As an initial step, we simplified the classification task by performing a stratified 80/20 train-validation split. In this setting, the models are exposed to all batch effects present in the original dataset during both pretraining and finetuning. We compare the downstream accuracy of this approach to that obtained using the official RxRx1 training split (which reserves entire experiments for testing and validation), with both configurations relying on the same backbone and classification head architecture. In both cases, training is limited to the 3 channel version of the HUVEC subset. The training loss curve and the validation accuracy are presented in Figure 5. It is straightforward to observe that accounting for all batch effects during both pre-training and fine-tuning substantially improves downstream performance compared to the results reported in [5]. To ensure consistency with that study, we choose not to further investigate this direction but instead we proceed by constructing dataset splits that allocate entire experiments — and consequently their associated batch

effects — exclusively to the validation and test sets. Therefore, when accuracy values are reported from this point onward, it is important to note that they are always computed on samples exhibiting batch effects not observed during model training.

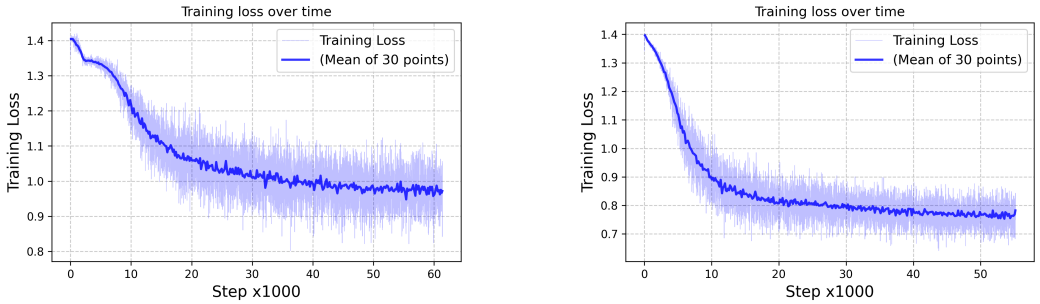


(A) Training loss **(B)** Accuracy

Figure 5: (A) Training loss and (B) validation accuracy for the classification task using a ViT backbone trained with DINO. We compare a model trained on a stratified dataset to one evaluated on validation data containing batch effects not encountered during either pretraining or finetuning.

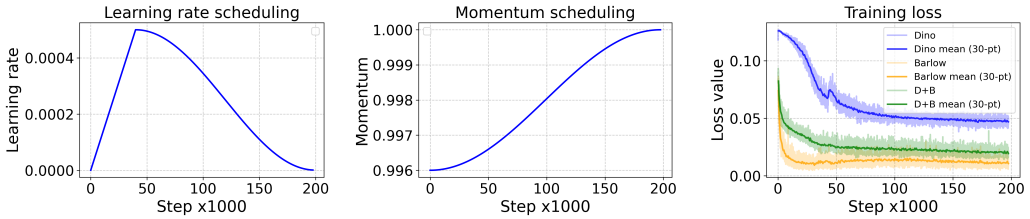
We further evaluated the effect of our simplified augmentation scheme in comparison to the original DINO augmentations (see Table 3). This analysis was essential to determine whether reducing augmentation complexity could support stable model convergence without triggering collapse. Accordingly, we trained two backbone networks using the previously described standard set of hyperparameters, with the exception of the Barlow Loss signal, which was disabled in both models. The only variable between the two networks was the augmentation strategy employed to generate the local and global views. Figure 6 presents the DINO loss curves for both backbones, indicating that the simplified augmentations result in overall more stable convergence. In addition to improved stability, the training loss drops below 0.8 when using simplified augmentations, whereas it remains above 0.9 with the original setup. We then trained a classification head on each backbone: the model trained with the original augmentations achieved a downstream accuracy of **19%**, whereas the model using our simplified augmentations reached **21%**. These results support our hypothesis that, for challenging tasks, employing simpler augmentations can make the learning problem more tractable.
After identifying a set of transformations that stabilized training, we further improved robustness and reinforced our baseline by performing pretraining without the Barlow loss, followed by fine-tuning on the cross-validation dataset (6 folds), yielding an accuracy of **19.5%** $\pm$ **2%**.

We evaluate the effect of integrating the Barlow Twins loss into the training objective. The backbone and projection head were jointly trained on the F0 split. As this configuration yields the most effective embeddings—based on performance in the downstream classification task—we present the corresponding learning rate and teacher momentum schedules, along with the loss curves, in Figure 7.



(A) Original DINO augmentations **(B)** Simplified set of augmentations

Figure 6: DINO training loss for a ViT-S/16 backbone trained with the original set of augmentations designed in [6](A) and our own set of simple augmentations (B). Darker and lighter blue represent the aggregated values (mean every 30 steps) and the original data respectively.



(A) Learning rate **(B)** Momentum **(C)** Training loss

Figure 7: Learning rate values (A), teacher momentum values (B), training loss curves (C).

We observed that the Barlow Twins loss rapidly decreases below 0.03 and contributes to stabilizing the fluctuations of the DINO loss. The head trained on top of this backbone achieves a downstream accuracy of **24%**, outperforming the 21% obtained on the same fold with the previous configuration.
To visualize the high-dimensional embeddings produced by the backbones, we employ Uniform Manifold Approximation and Projection (UMAP) [9]. UMAP is a dimensionality reduction technique that preserves both local and global structure in the data, making it well-suited for revealing clustering patterns and the overall geometry of the learned representations.

Table 4: ViT-S/16 backbones trained with different DINO configurations.

Backbone	Barlow Loss	Multi-Centering	Augmentations	Accuracy
A (ViT-S/16)	✓	×	Simple	**0.24**
B (ViT-S/16)	×	×	Simple	0.21
C (ViT-S/16)	✓	×	Original	0.19
D (ViT-S/16)	✓	✓	Simple	0.23
E (ViT-S/16)	✓	✓	Original	0.19

Figure 8 presents UMAP visualizations of features extracted by the models listed in Table 4. All models were trained on the same subset of the RxRx1 dataset (F0) but with different training configurations. The features were extracted from samples in the F0 training set. For instance, plots A and B illustrate the impact of different training objectives: plot A used

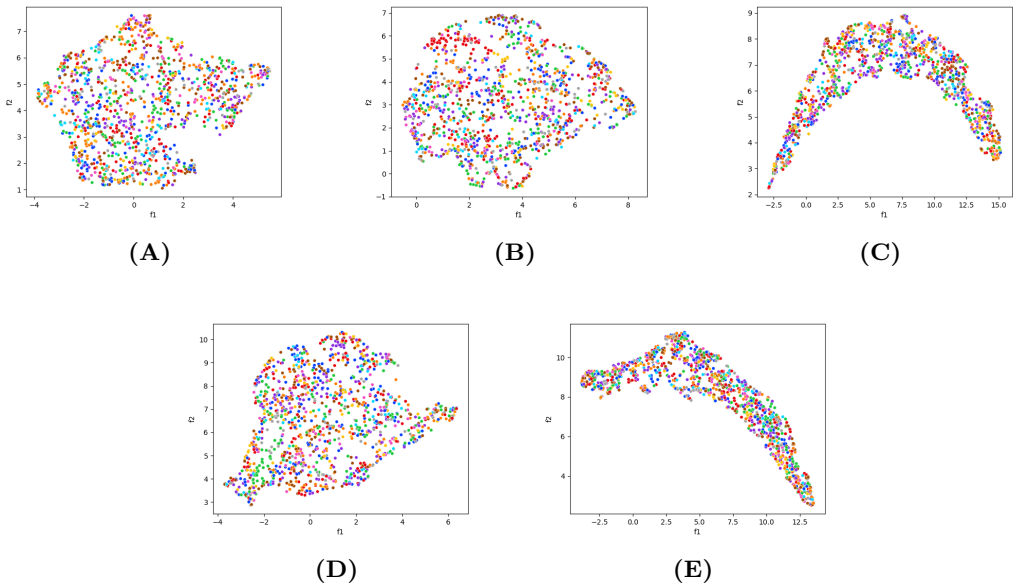
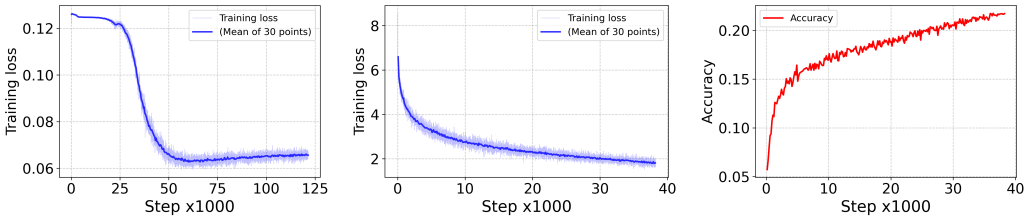


Figure 8: UMAP embeddings of the backbones listed in Table 4

Barlow loss, whereas plot B relied solely on DINO loss. A key difference between them is the presence of a red cluster in the top right corner of plot B, suggesting that the Barlow loss may have mitigated batch effects in that particular experiment. Plots C and E exhibit a notably distinct structure compared to the others. A common characteristic of these two models is the use of multi-centering during training. This suggests that the application of multi-centering substantially alters the extracted features, leading to markedly different clustering patterns. However, this change did not translate into improved downstream accuracy in our experiments. Overall, none of the plots exhibit a clear clustering of samples based on the experiment identifier. In all cases, the distribution appears relatively random, which is desirable. Given the extremely high number of classes (1139), it is challenging to visually discern any additional grouping patterns that might correspond to the classification labels.

As a final experiment, we extended the F0 split by incorporating the rest of the dataset, which also includes samples from additional cell types. We first attempted to train using cross-domain learning while accounting only for batch effects related to the experimental setup. However, training diverged immediately. This suggests that cell type introduces an additional layer of batch effect, which can obscure the signal of interest. We then repeated the training while also accounting for cell type: the additional sample being loaded not only originates from a different experiment but also belongs to a different cell type. In this configuration, training successfully converged. To enable convergence, we also had to scale up the model by switching to a ViT-Base architecture instead of ViT-Small, and increasing the batch size per GPU to 128. Moreover, because of computational constraints, we selected the 3 channel version of the dataset.



(A) DINO **(B)** Head **(C)** Accuracy

Figure 9: Dino (A) and head (B) training loss signals, downstream accuracy over time (C).

We then finetuned the head on F0, thus using only HUVEC cell types. The goal was to see if pretraining using more examples of different cell types could produce a better feature extractor. Figure 9 shows the training loss of both the backbone and the head during training, and the downstream accuracy over time. In plot A we can see a slight increase in the DINO loss after 60k steps, so we early-stopped the training and used that checkpoint to load the weights of the feature extractor during fine-tuning. Plot B and C instead show the head training loss and the accuracy over time. Despite the second showing a steep upward trend and surpassing the 20% threshold, the first falls just below 2. This indicates that with a more refined scheduling strategy, comprehensive hyperparameter tuning via grid search, and potentially extended training across multiple GPUs, it is likely possible to achieve substantially higher accuracy.

6 Conclusions

This study systematically explored self-supervised learning methods to mitigate batch effects in the RxRx1 biomedical imaging dataset. We confirmed that standard self-supervised approaches like SimCLR and DINO are highly susceptible to these effects, causing them to cluster data by experimental batch rather than biological information.

To overcome this, we worked to adapt some existing self-supervised learning frameworks. For SimCLR, we used Contrastive Batch Decorrelation Loss and an adversarial batch invariance strategy. For DINO, we implemented cross-domain learning and simplified augmentation schemes. Our experiments showed these adaptations significantly improved downstream classification performance by reducing batch-specific artifacts. We observed that techniques enforcing batch invariance led to more robust and generalizable embeddings, and that simplified augmentations enabled stable training. Combining DINO with Barlow Twins loss also offered regularization benefits.

While promising, our work has limitations, including computational constraints on hyperparameter tuning and challenges from the RxRx1 dataset's complex data distribution. Future work should explore new self-supervised frameworks, refined augmentation techniques, and broader dataset validation to further improve batch effect mitigation.

References

[1] Rxrx1 dataset. https://www.rxrx.ai/rxrx1. Accessed: 2025-05-02.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR, 2020.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Syl-

vain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[4] Jean-Bastien Grill et al. Bootstrap your own latent a new approach to self-supervised learning. 2020.

[5] Johan Fredin Haslum et al. Metadata-guided consistency learning (cdcl) for high content images. 2022.

[6] Mathilde Caron et al. Emerging properties in self-supervised vision transformers. 2021.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[9] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[10] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pages 12310–12320. PMLR, 2021.