Technical University of Applied Sciences Würzburg-Schweinfurt (THWS)
Faculty of Computer Science and Business Information Systems

Master Thesis

Determination of Drug Efficacy on Pancreatic Tumor 3D Spheroidal Tissues

submitted to the Technical University of Applied Sciences Würzburg-Schweinfurt in the Faculty of Computer Science and Business Information Systems to complete a course of studies in MAI

Bibin Babu

Submitted on: February 17th 2025

Initial examiner: Prof. Dr. Magda Gregorová
Secondary examiner: Prof. Dr. Jan Hansmann

Abstract (en)

Pancreatic tumor treatment is hindered by the intricate nature of tumors and their diverse microenvironments. This complexity necessitates an exploration into identifying optimal drug combinations and concentrations tailored to each patient's specific tumor characteristics. This thesis aims to assess drug efficacy by ranking these various drug combinations and concentrations. The ranking is based on features extracted from bright-field microscopy images of three-dimensional tumor tissue models using SimCLR self supervised learning. The core challenge is to learn robust features that accurately characterize alterations in these tumor tissue models induced by drug application over time. This research seeks to develop a standardized and effective approach for evaluating drug efficacy, potentially improving treatment outcomes for pancreatic tumor patients.

Acknowledgment

Thank all who believed in me.

Thank Prof. Magda Gregorova for guiding me to how to think scientifically and also for the SimCLR and autoencoder prediction ranking strategy.

Thank Prof. Jan Hansmann for the softmax approach idea.

Thank Dalia Mahdy for the code for sharp layer calculation and thanks for the biology explanation of tumor tissue model,and the once in a time opportunity.

Thank Phillip Ockermann for the cell viability test idea. Thank stefan Zinggrebe for the microscope focal length explanation. Thank Philipp vaeth for the resize and crop augmentation idea and general explanation for the data augmentation.

Thank Dad and Mom for this life. Without you guys I won't be able to write this. Thank you to my brother for being my backbone. Finally thanks to chatgpt, Perplexity, Claude sonnet, Deepseek, Grammerly. without those there will be only implementations.

Danke.

Würzburg, on 17.02.2025

_____

Bibin Babu

List of Abbreviations

PCA Principal Component Analysis

t-SNE t-distributed Stochastic Neighbor Embedding

UMAP Uniform Manifold Approximation and Projection

CAE Convolutional Autoencoder

H Hight

W Width

Contents

List of Figures

List of Tables

1 Introduction

Pancreatic tumor presents a significant challenge in terms of treatment due to its heterogeneous nature and the mutations that occur during its progression within the human body. Clinicians rely on case studies, human trials, and their own expertise gained from past patient treatments to select drugs for new patients. However, this approach is often based on trial and error, with varying outcomes. Patients may experience either successful treatment or severe side effects such as hair loss and damage to other organs. Since each patient's tumor cells exhibit unique characteristics influenced by factors such as age and genetics, treatments that have worked for one patient may not be effective for another. Consequently, clinicians may need to change the prescribed drugs or try different combinations, which can lead to delays and increased risks for the patient, including mortality.

In light of these challenges researchers at Fraunhofer Translational Center for Regenerative Therapy TLZ-RT Wuerzburg, propose a vision for the future: cultivating multiple three-dimensional tumor tissue models for each patients in the lab using biopsy samples and studying the efficacy of drugs on these three-dimensional tumor tissue models first.*(Note: In this thesis, "3D tumor tissue models or tumor tissue models" refers to physical, lab-grown tissues and not computational or AI models.)* By conducting drug development experiments and analyses on these tissue models, they aim to find the optimum or best drug combination tailored to each patient's specific tumor characteristics.This approach can minimize direct side effects on human patients and reduce the time needed to select the most effective personalized treatment, thereby decreasing the risk of that patient's mortality. Additionally, it can significantly reduce the cost and time of preclinical testing in the drug development process. Ultimately, these information obtained from drug efficacy assessment experiments can inform clinicians' decisions, enabling them to select the most effective drug combination before administering it to the patient.

As a proof of concept, The Fraunhofer TLZ-RT Würzburg laboratory utilizes a modular dual-arm robot-based system [4], equipped with incubators and bioreactors (see Figure 1.1) under physiological conditions to study drug efficacy for the long-term culture of these three-dimensional tumor tissue models. One advantage of this platform is its ability to capture bright-

field microscopy images (detailed explanation in Chapter 5) of 3D tumor tissue models using a customized microscope setup integrated into a robotic platform, providing flexibility in image acquisition to meet experimental requirements.

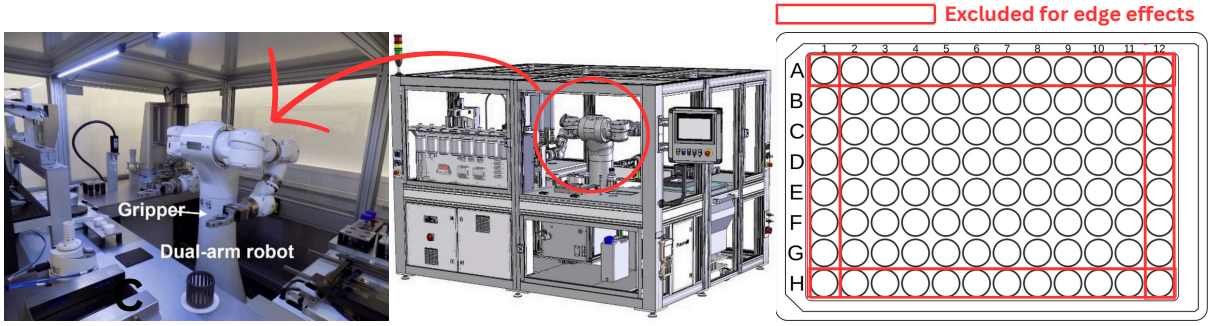

Figure 1.1: First and second images are the roboplatform that cultivates the 3d tumor tissue models. Third imags shows the well plate used to contain these tissue models

Although the vision for the future is to simulate the identical interaction environment of drugs with tumor cells as it occurs in the human body, current technology has not yet achieved this. The current three-dimensional tumor tissue models developed in the lab do not fully resemble real pancreatic tumor cells found in the human body. These 3D tumor tissue models only contain pure tumor tissues, whereas real human pancreatic tumor cells exist within a complex microenvironment comprising tumor cells, blood vessels, other tissues, and various cell types. Fortunately, if human body tumor cells can be replicated in the lab in the future, the techniques currently used to study bright-field microscopy images will still be applicable. However, the fact that bright-field microscopy images are two-dimensional limits the ability to perform a comprehensive analysis of the drug's impact on the entire 3D structure of the cultivated tumor tissue models. Despite this limitation, this research serves as a valuable starting point for studying drug efficacy in a controlled environment.

Alternatives to bright-field microscopy images include 3D fluorescence microscopy and luminescent cytotoxicity assays. However, both methods are invasive. Fluorescent molecules tend to generate reactive chemical species under illumination, enhancing phototoxic effects. This chemical reaction with the 3D tumor tissue model may alter its structure, making it not suitable to isolate the drug's effect over time. Similarly, luminescent cytotoxicity assays result in a dead culture, rendering them unsuitable for longitudinal studies. Additionally, both methods require removing the well plate from the isolated culture environment for extended periods,

making the samples susceptible to external environmental factors. For instance, in fluorescence microscopy, cells are particularly vulnerable to phototoxicity from short wavelength light. In contrast, bright-field microscopy images are non-invasive, allowing continuous culture and the possibility of creating time series of images to study dynamic changes. Therefore, we rely on bright-field microscopy images to study the time-evolutionary effects of drugs.

1.1 Laboratory Setup

3D tumor tissue models are cultured in well plates containing 96 wells, each providing a nutrient medium that allows them to maintain their tissue-specific functions in vitro. Although each plate can yield 96 pure 3D tumor tissue models, the edge effect is accounted for, where outer wells may be exposed to variable conditions such as temperature fluctuations, increased evaporation rates, and other environmental factors. Consequently, we restrict our analysis to the 60 inner wells per plate as in figure 1.1, adhering to standard procedures to ensure consistent and reliable experimental data. Based on the drug concentration applied to 3D tumor tissue models, the bright-field microscopy images we capture can be categorized into:

1. Control (0 percentage drug applied). Samples of these images are shown in the figure 1.2 in the second row.
2. Single concentration (theoretically recommended single concentration of drug treatment) For easiness, we refer to this category as “Single dose”. Samples of these images are shown in the figure 1.2 in the first row.
3. Control Day 10: The control images show a morphological change where the center of the tumor tissue becomes darker, resembling the darkness observed when drugs are applied on Day 10. This effect occurs due to the natural death of tumor cells within the tissue, caused by a lack of nutrients to sustain the cancer cells. Samples of these images are shown in the figure 1.2 in the third row.
4. Drug Screening: Different drug combinations and concentrations are used in this experimental study to evaluate drug efficacy. Some images may resemble single-dose images, while others may resemble Control Day 10 if the drug is ineffective. Some tissue mod-

els may have different morphological changes and may end up in nothing similar to any of the above-mentioned types. Additionally, particular drugs may or may not result in the destruction of surrounding non-tumor cells in the human body, potentially causing side effects. They form debris around the treated spheroid; for simplicity, we refer to this category as ”Explode.” A sample of this type is shown in figure 1.2 in the last row, middle. Above mentioned all, belong to the category ”Drug screening.” These are the images that require relative assessment through ranking. The figure 1.2 in the last row shows additional samples of these images.

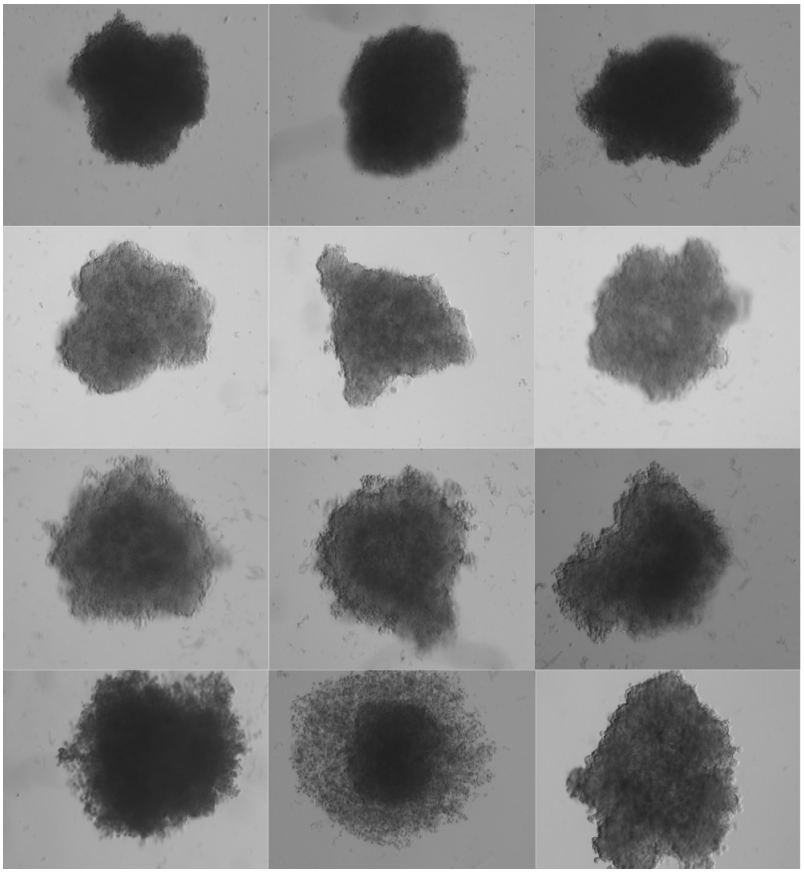

Figure 1.2: Different types of images: Single dose, Control, control day 10 and drug screening as mentioned in section 1.1.

A single dose and drug-screened combinations were applied in different well plate settings, as illustrated in Figure 1.3. The left image shows the well plate setup for the single-dose experiment, where the left half was left untreated, and the right half was treated with a single dose concentration. This image was taken three days after drug application. The right image shows the well plate setup for the drug screening experiment, where most tumor tissues were treated with different combinations of drug concentrations (multi-colored wells). At the same

time, some were left untreated (white wells bounded by an orange box). A total of 8 specific drug concentration combinations were experimented with in the drug screening. For the single-dose experiment, a combination of GAN61 μM and JQ1 1.25 μM concentrations was applied, which differed from the drug screening drug concentration combinations.

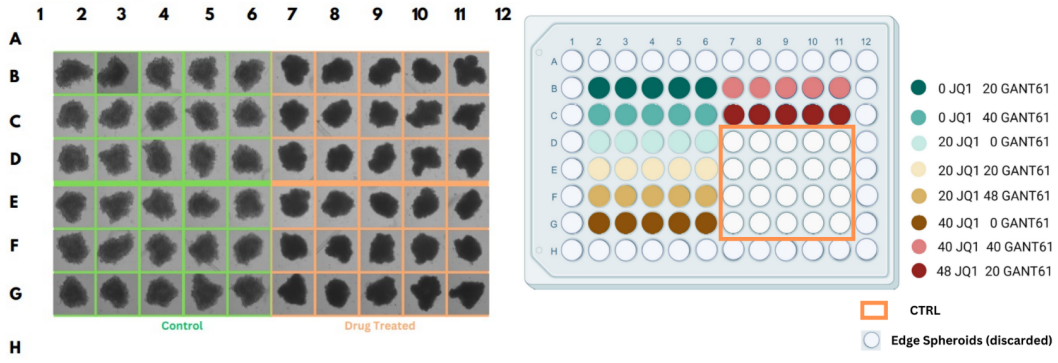

Figure 1.3: Left image shows well plate setup for the single-dose experiment. Right image shows well plate setup for the drug screening experiment

1.4 Illustrates the flow chart of time evolutoion of 3D tumor tissues. The 3D tumor tissue models develop in the wellplate progressively from day 1 reaching their maximum cancerous state by day 7, at which point the drug is administered. By day 10, the drug's effect on the cancerous tissue is expected to peak as nutrient availability gradually decreases, causing the tumor to diminish. Therefore, to isolate the drug's effects, changes in tumor tissue deterioration are assessed on the peak effect day, i.e., day 10, in accordance with established medical protocols and previous research findings.

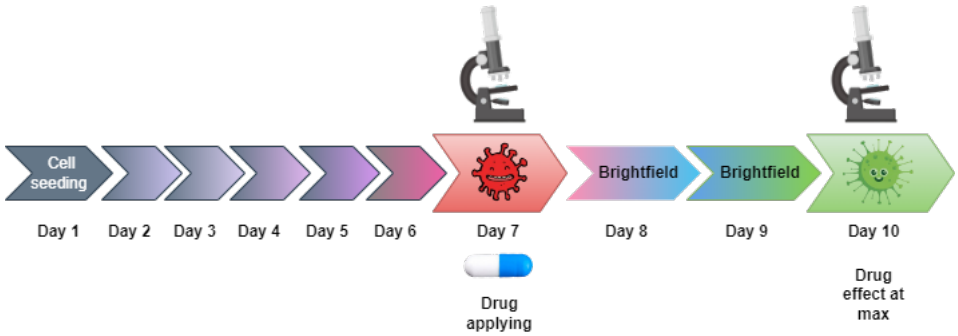

Figure 1.4: Illustrates the flow chart of time evolutoion of 3D tumor tissues.

2 Motivation

We assess the efficacy of the drug by comparing the changes it induces in the bright-field microscopy images over a period of time. The current methods to differentiate these changes involve studying the alterations from day 7 to day 10. These changes are typically observed in three main parameters:

1. Size/Area
2. Circularity/Diameter/Perimeter
3. Pixel intensity or color change

These parameters serve as human-interpretable metrics for assessing the efficacy of the drug. However, these manual methods are inherently limited to observable features, which may overlook subtle or complex patterns within the data. Representation learning techniques, such as those based on deep learning, can be employed to extract features that are not immediately interpretable by humans. These methods leverage neural networks to learn high-dimensional feature representations directly from the images, capturing intricate patterns, relationships, and variations that may correspond to biological phenomena. Furthermore, these learned representations enable a standardized approach to analysis. Unlike manual assessments, which can vary due to human subjectivity, representation learning models produce consistent results once trained. This is the focus of my master thesis, where I take on the crucial role of developing and applying representation learning techniques to uncover these hidden patterns and standardize the analysis process.

3 Research Objective and Questions

Research Objective

This thesis aims to assess drug efficacy by ranking different drug combinations and concentrations on lab-cultivated pancreatic 3D tumor tissue models. The ranking is based on features extracted from bright-field microscopy images of these tumor tissue models. The primary challenge lies in efficiently learning the features of alterations induced in these tumor tissue models by the impact of drug application over a period of time.

Another significant challenge in the ranking strategy is the absence of ground truth labels. The only available information is that drug efficacy is zero for untreated (control) images. Consequently, supervised learning is not possible, and the limited amount of data further complicates the task. Therefore, SimCLR, a self-supervised learning method [2], is used to learn latent representations of the images. Due to the absence of ground truth labels, determining an exact efficacy value through ranking will be the out of focus. Instead, the focus will be placed on formulating ranking strategies to order the drug screening images relative to the various possible transitions between classes, such as untreated to single dose to exploded images, or alternative sequences like single dose to untreated to exploded, or exploded to untreated to single dose, etc as explained in the chapter 9. However, the lack of ground truth labels also complicates the problem, as a standard evaluation method to assess the effectiveness of the ranking strategy is unavailable.

Research Questions

1. Can we learn latent features that capture the alterations induced in tumor tissue models by drug application over a period of time from bright-field microscopy images?

2. Will these features effectively establish a relative ranking assessment of drug efficacy?

3. What methodologies and frameworks can be employed to extract and learn these hidden representations efficiently?

4. What could be reasonable metrics, such as L2 loss or cosine similarity, to support the relative assessment of drug efficacy?

4 Literature Review

Base neural network architecture for representation learning. Learning visual representations of medical images, such as X-rays (radiographic images) and bright-field microscopy images, is crucial for medical image understanding. However, progress in this area has been hindered by the heterogeneity and complexity of subtle features in these images, especially when they don't have labels. Existing work often relies on fine-tuning weights transferred from ImageNet pretraining (Wang et al., 2017 [12] ; Esteva et al., 2017 [5] ; Irvin et al., 2019 [6]), which is suboptimal due to the drastically different characteristics of medical images.

To address these challenges, researchers have proposed various innovative approaches. ConVIRT [**zhang2022contrastive**] offers an alternative unsupervised strategy for learning medical visual representations by exploiting naturally occurring paired descriptive text. This method introduces a new approach to pretraining medical image encoders using paired text data via a bidirectional contrastive objective between the two modalities. It is domain-agnostic and requires no additional expert input. However, given the absence of specific paired text data for our image dataset, ConVIRT does not offer a solution tailored to our specific problem.

The contrastive loss used in ConVIRT is derived from the SimCLR [2] self supervised learning framework. SimCLR learns representations by maximizing agreement between differently augmented views of the same data example via a contrastive loss in the latent space. The framework consists of a neural network base encoder that extracts representation vectors from augmented

data examples. The framework allows for various choices of network architecture without any constraints. The authors opt for simplicity and adopt ResNet [**he2015deepresiduallearningimage**], introducing a learnable nonlinear transformation between the representation and the contrastive loss to substantially improve the quality of the learned representations. However, these methods require careful treatment of negative pairs, typically relying on large batch sizes to retrieve them. Additionally, their performance is highly dependent on the choice of image augmentations.

After the ranking strategies outlined in the methodologies chapter 9 were applied, reasonable metrics needed to be considered to support the relative assessment of drug efficacy. We refer this as 'inference metric'. Cosine distance was selected as the first choice because the SimCLR loss is designed to produce higher similarity between similar images. In the study by [10], Euclidean distance was employed to identify nearby cells for unstained cell detection in bright-field microscope images. Notably, alternative metrics have been proven useful in related applications; for example, [3] employed the Pearson correlation coefficient to measure similarity between predicted images (or their extracted features) and ground truth images in the context of label-free Cell Painting, where five fluorescent Cell Painting channels were predicted from brightfield input. In the study [8] titled *DeepHCS: Bright-Field to Fluorescence Microscopy Image Conversion*, L1 distance (Mean Absolute Error, MAE) was used as a loss function to measure the pixel-wise error between predicted and ground truth fluorescence images, thereby ensuring accurate translation of brightfield images into synthetic fluorescence images. Furthermore, the study [11] on bacterial cell tracking (assessing the overlap of cell regions between consecutive frames) utilized the Jaccard index to compare the performance of different tracking tools, including on bright-field images of *P. protegens* and *S. pneumoniae*. These findings collectively support the use of cosine distance, Euclidean distance, Pearson correlation coefficient, L1 distance, and the Jaccard index as reasonable metrics for the relative assessment of drug efficacy.

Due to the problem of limited data, the exploration of ranking strategies that do not require specific training has been deemed essential. Dimensionality reduction techniques, such as PCA, t-SNE, and UMAP, have therefore been selected. In [7], PCA, t-SNE, and UMAP were applied to high-dimensional cytometry datasets in the study of human cancer, specifically for the analysis of bone marrow aspirates from acute myeloid leukemia patients, with the aim of reducing

the data to two or three dimensions for analysis and interpretation of cytometry data.

In another study [13], PCA, UMAP, and t-SNE were applied to reduce the dimensionality of bottleneck features from a Swin UNETR trained for liver segmentation on T1-weighted MRIs. It was found that either PCA or UMAP improved performance over average pooling for all models tested. Since studies have demonstrated that PCA, t-SNE, and UMAP perform well on medical images, further exploration of these techniques has been conducted to reduce the higher dimensionality of the image data to a single dimension for ranking purposes.

5 Data Description

5.1 Data set

As explained in the chapter 1, the dataset consists of images taken by bright-field microscopy, which operate as follows:

The tumor 3D spheroidal tissue model is illuminated by transmitted white light (i.e, light is projected from below and observed from above). The contrast in the image is created due to the reduction in light intensity as it passes through denser regions of the sample, where more light is absorbed or scattered. This results in a typical bright-field microscopy image where the sample appears darker against a bright background, giving the technique its name. In our case, the 3D tumor model appears as a dark grayish structure on a bright background. For simplicity, bright-field microscopy images will be referred to as 'images'.

Since the tissue models are cultivated using robotic arms, the process sometimes fails to replicate the natural shapes and patterns that typically occur when laboratory personnel manually cultivate the samples. To ensure the dataset's quality and consistency, Dalia Mahdy, Phd student who works in the lab, pre-processed the images through a machine learning model to filter out

invalid ones. By 'invalid images' , we mean those that variates from the expected morphology of the tumor tissues as cultivated by lab personnel. These images are typically elongated either in height or width as shown in second image of figure 5.1. I collected these filtered images via USB and Google Drive in their original TIFF format.

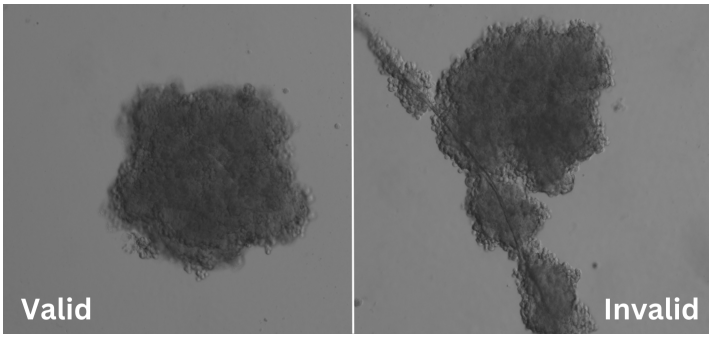

Figure 5.1: Examples of bright-field microscopy images with valid and invalid (unexpectedly elongated) morphology of tumor tissue models.

The original images are 2456x2054 pixels in size, in 16-bit grayscale, and consist of multiple layers. These layers are obtained by capturing images at different focal planes in brightfield microscopy. The number of layers can vary, as images can be taken at any number of focal planes. The sharpness of each layer of an image depends on how well the focal plane of the microscope aligns with the depth of the tumor tissue model. Only one layer of this 3D tumor tissue model will be perfectly in focus, while others may appear blurry because they are slightly above or below the focal point as you can see in figure 5.2. Combining these focal planes later in computational analysis can provide richer data, even if some layers are blurry individually. This is one of the reasons why I decided to use multiple layers. The images I received from the lab mostly have three layers, while a few have one or five layers.

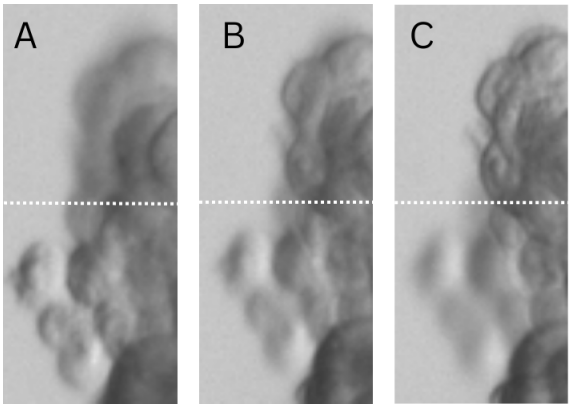

Figure 5.2: Different layers of the same image reveal that, from A to C, the region above the white dotted line becomes sharper, while the region below becomes blurry

Within the same experiment, images are captured using consistent acquisition settings. that means each image in the same experiment consists of different layers captured at same focal lengths. For example, if image 1 has layers corresponding to focal lengths A, B, and C, Image 2 will also have layers corresponding to the same focal lengths A, B, and C. However, for images from different experiments, the focal lengths may differ. For instance, images from experiment 1 (single dose) may have slightly different acquisition settings compared to images from experiment 2 (drug screening), leading to variations in the focal lengths and corresponding layers. Secondly, the layers in each image are slightly misaligned when stacked, as shown in the figure 5.3. This misalignment of layers within the image and variation in focal length inbetween the images may or may not affect the performance of ranking.

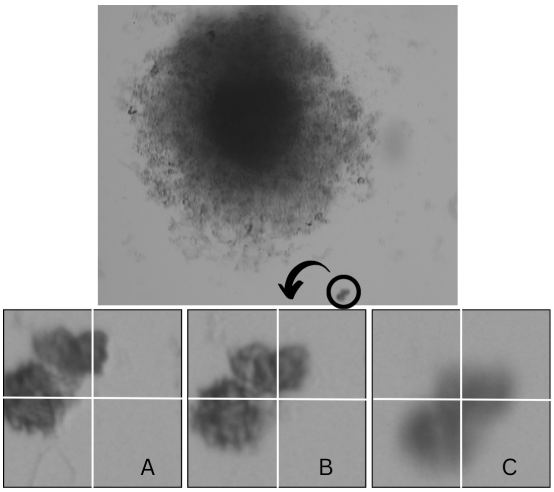

Figure 5.3: A small section of the image was zoomed in and separated to show the misalignment of the three stacked layers.

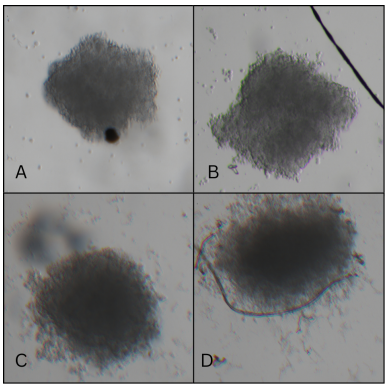

Figure 5.4: Noise

We observed some temporal noise/artifacts, as shown in figure 5.4 in images. The presence of artifacts in the well-plates could be attributed to environmental factors such as temperature,

humidity, or CO_2, or by some inconsistencies in sample handling. Such factors are likely to cause variations in cell behavior and inconsistency in the acquired images. These images were included as-is for SimCLR training, providing an opportunity for the model to learn from realistic conditions. However, I didn't artificially introduce additional noise or artifacts through data augmentation. This approach highlights a potential area for further exploration in future work.

Figure 5.5 illustrates that, even with the same drug concentration applied to the spheroid tissue models under controlled conditions, occasional variations in effects were observed across different experiments. These variations could be attributed to slight differences in spheroid composition or undetected microenvironmental influences that affected drug absorption by the spheroid and cellular response. Initiall idea was to evaluate the relative effect of images using the same drug concentration combination as a group, but due to these differences, the ranking assessment needs to be made relative, based on the individual day 10 images, even though the same drug combination was applied.

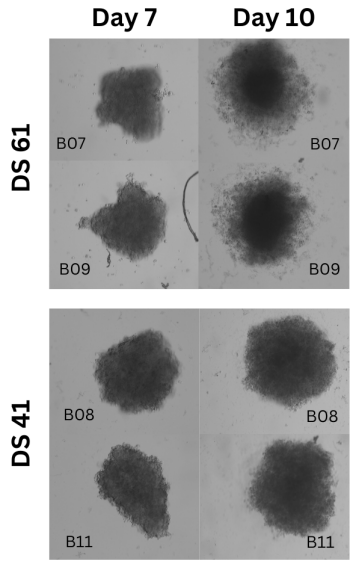

Figure 5.5: Different morphological changes were observed even when the same specific combination of drugs was applied in two different experiments. The same combination of 40 µM JQ1 and 40 µM GANT61 was applied in two different experiments, named DS61 and DS41

The table below 5.1 shows the division of different types of image datasets, as explained in Section 1.1. In this table, Day 8 and Day 9 represent the images obtained during different single-dose and drug-screen experiments. These images are included for SimCLR training.

From the table, it is evident that there is a class imbalance.

Class	Drug Screen	Single Dose	Control day 10	Control	Day 8 & 9	Total
No. of Images (%)	200 (16.05%)	103 (8.26%)	231 (18.54%)	472 (37.89%)	240 (19.26%)	1246

Table 5.1: Dataset Class Overview

An 8-bit image encompasses 256 color tones (ranging from 0 to 255) per channel, while a 16-bit image accommodates 65,536 color tones (ranging from 0 to 65,535) per channel, specifically allowing for 65,536 shades of gray. Retaining the original 16-bit depth is crucial because converting it to an 8-bit image for faster and more efficient computation can lead to significant information loss in intensity details. Since 8-bit images permit only 256 possible values, the finer intensity variations present in 16-bit images become compressed, as illustrated in 5.6. For instance, two distinct 16-bit values (ranging from 30,000 to 30,048) could map to the same 8-bit value (for example, both might be mapped to 117). This results in the loss of subtle intensity differences, which can be vital in our image task, where minute variations in intensity can indicate important features, such as the gradual transition of dark color from the center to the border or the amount of debris surrounding the tumor cell.

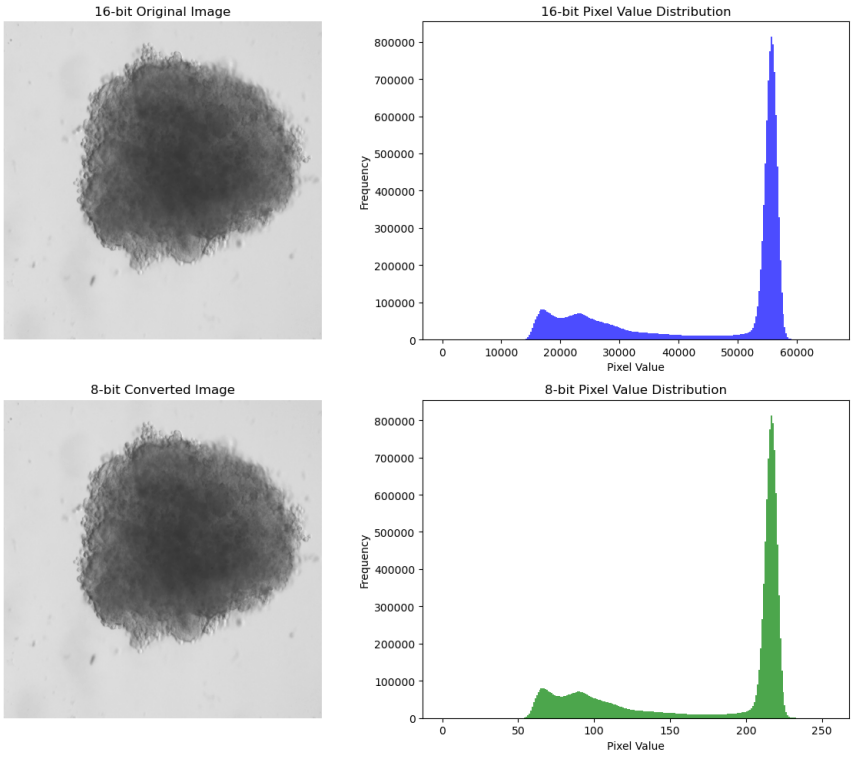

Figure 5.6: 8-bit vs 16 bit data lose comparison. Although the images visually appear similar, 16-bit data has pixel values ranging from 0 to 65535, while 8-bit data has pixel values ranging from 0 to 255

5.2 Challenges

1. **Limited Dataset for Day 7 to Day 10 ranking prediction Model:** The dataset for predicting outcomes between Day 7 and Day 10 is limited, which poses a challenge for training and evaluation.

2. **Issues with Day 10 Images:** Day 10 images can exhibit variations such as flipping, blurring, brightness changes, and position changes from the original position of tumor tissue model in the image. The position change occurs because, when the well plate is brought outside and then placed back under the microscope for taking day 10 image, the position often shifts. This relative position of the tumor cell in the image can shift from the center to other directions. While a 'center crop' explained in in Section 7.1 approach can address this issue to some extent, it fails when the tumor tissue model is located at the edge of the image. To deal this issue for some extend, Applying horizontal and vertical

flips, rotations by 90° and 270°, as well as combinations like horizontal flip + rotation 90° and horizontal flip + rotation 270°, can help make the model invariant to position changes.

3. **Variability in Drug Effects on Day 10:** The same drug can have different effects on Day 10. Hence we can't assess the drug efficacy based on the group of images applied by same drug combination instead we need to make the ranking relative assessment based on each individual the day 10 images as illustrated in 5.5.

A total of 40 images from the drug screen with significant debris were manually selected for intermediate evaluation of SimCLR, as explained in Chapter 8, and ranking strategies, as explained in Chapter 9.

6 Structure of the Thesis

The goal of this thesis is to leverage representation learning of bright-field microscopy images using SimCLR to develop a ranking/ordering scale (1 to n) for all images.

In this chapter, I will explain the framework designed to address this problem. The task is divided into three main pipelines. First, we learn latent representations from the images using SimCLR as a self-supervised learning (SSL) model. The details will be explained in the *Methodology for SimCLR* chapter. Next, we perform an intermediate evaluation to assess whether the features have been effectively learned. The intermediate evaluation aims to classify and cluster the images. The details of this step will be explained in the *Methodology for Intermediate Evaluation* chapter. Subsequently, we use the learned features to establish a ranking scale. To achieve this, the following methods are employed:

1. Prediction model,

2. K-means centroid approach,

3. Softmax approach, and

4. Dimensionality reduction techniques such as PCA, t-SNE, and UMAP.

The details of these methods will be discussed in the *Methodology for Ranking Strategy* chapter. Finally, we address an important question: why do we need to learn feature vectors from the images using SimCLR? Could the ranking task perform better using the raw image vectors directly instead of SimCLR features? To answer this, we conduct a comparative study to evaluate the performance of both approaches in intermediate evaluation and in ranking methods.

The overall structure of the thesis is illustrated in Figure 6.1.

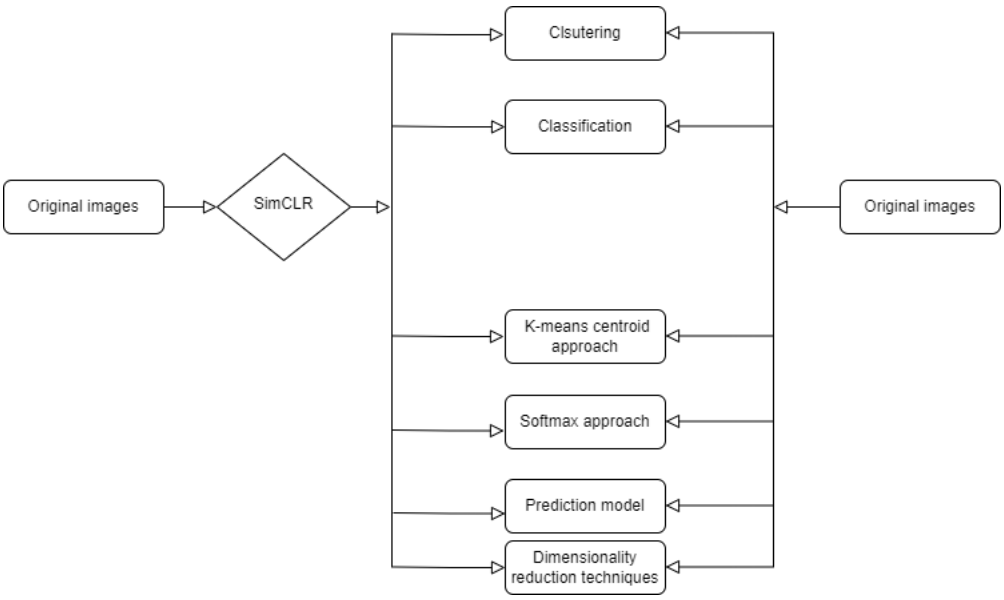

Figure 6.1: Overall framework of the thesis.

7 Methodology for SimCLR

7.1 Data preprocessing

1. **No of channel per image selection**: For 3 reasons, I decided to use 3 layers per image as 3 channels of the image for SimCLR training:

    a) Considering each 3 layers per image may capture depth information at 3 different focal planes of a 3D tumor tissue model, providing richer information compared to only single-channel data.

    b) For ease of integration with our pretrained architecture, ResNet18, which was trained on 3-channel images.

    c) The majority of the images in the dataset have exactly 3 layers, making this a practical choice.

    Images with only 1 channel will have their single channel duplicated to form 3 identical channels. For images with 5 channels, the top 3 sharpest layers are filtered and selected to reduce the image to 3 layers (Process is explained below). This ensures consistency across the dataset while maintaining maximum texture and edge information. The concept of sharpness is central to selecting the most informative layers. Layers with higher sharpness are retained because they carry more texture and edge information, while layers with lower sharpness tend to have less detail and appear blurred. Sharpness is calculated intuitively as follows (Dalia Mahdy's code used for the implementation):

    a) Sharpness is related to how quickly pixel intensities change in an image. These changes are measured using gradients, which calculate how intensity changes between neighboring pixels. The gradient magnitude

$$g_{\text{norm}}$$

is computed as:

$$g_{\text{norm}} = \sqrt{g_x^2 + g_y^2}$$

where g_x and g_y are the gradients in the horizontal and vertical directions, respectively.

b) After calculating the gradients for all pixels in a layer, their magnitudes are averaged:

$$\text{Sharpness Score} = \frac{1}{N} \sum_{i=1}^{N} g_{\text{norm},i}$$

where N is the total number of pixels in the layer. A higher sharpness score corresponds to layers with more edges, transitions, and details, which are likely to be in focus.

c) Layers are ranked based on their sharpness scores, and the top 3 layers are selected. This ensures that the dataset retains layers that are in focus and have the most texture and edge information.

2. **Croping the image to make whole image from rectangular to square ie Hight = Width (H=W) of the iamge**

Advantage of croping to H=W are:

a) Remove unnecessary background which contains no information

b) There will be no shear during resize to 96*96 augmentation like as shown in the figure 7.1

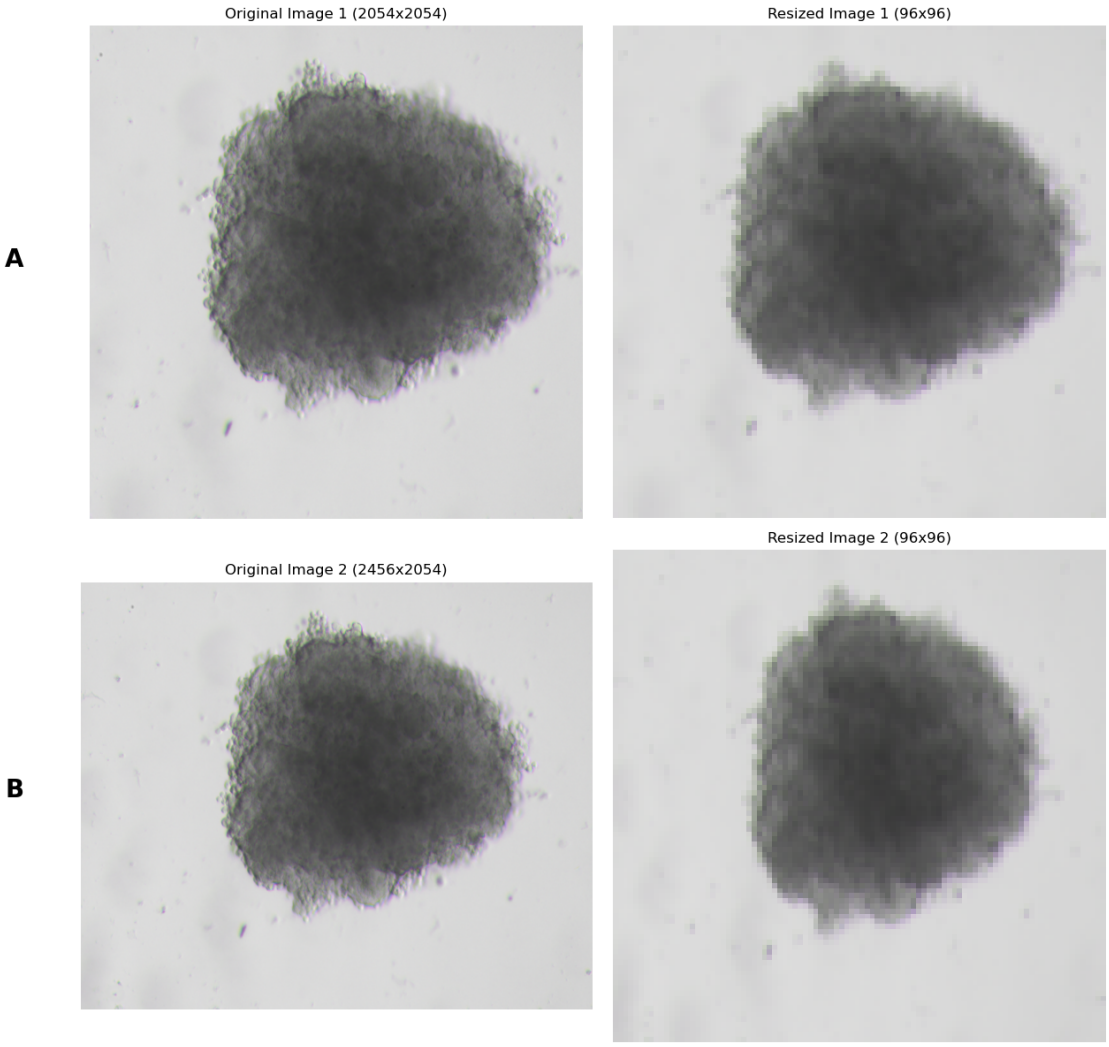

Figure 7.1: First row: Cropped to 2054x2054 and then resized to 96x96, with no shear or elongation in one dimension. Second row: Original image (2456x2054) resized to 96x96, resulting in shear or elongation in one dimension

c) Augmentations such as rotation by 90° and 270°, as well as combinations of horizontal flip and rotations by 90° or 270°, are easily applied when the image is square. If the rotation involves angles other than multiples of 90°, black borders will appear, requiring additional interpolation or other techniques. Therefore, since the image is square, rotations by multiples of 90° can be applied without any additional tasks, as explained in the figure 7.2.

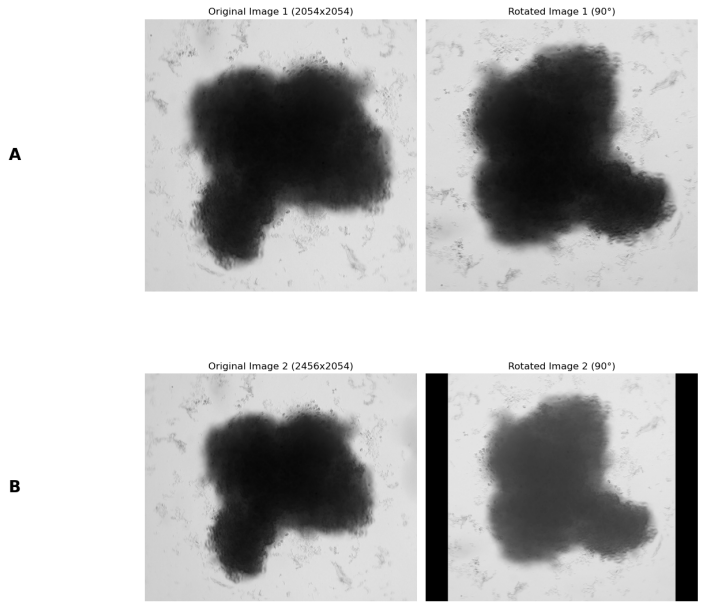

Figure 7.2: A: Cropped to 2054x2054 and then resized to 96x96: No black borders. B: Original image (2456x2054) resized to 96x96: Black borders appeared.

We cannot rely on a simple center crop for our images because some tissue models are not located centrally within the image; instead, they are closer to the edges. To ensure that the tissue models are completely included in the cropped region, we must identify the boundaries of the tissue models, including any debris, obtain the corresponding bounding box, and crop the image accordingly to guarantee that the cropped image contains the full tumor tissue models. The original image dimensions are 2456×2054 (Height × Width). It turns out that setting H = W = 2054 ensures that the tumor tissue model is fully contained in all images processed with computer vision libraries (to be explained soon). Any tumor tissue model that exceeds this size is either elongated in a single dimension (e.g., horizontally), indicating invalid cultivation by the robotic system, and will be discarded by Dalia's machine learning model. For implementing this procedure, the following steps were employed using the `skimage`, `cv2`, and `numpy` libraries:

1. The image is converted to grayscale by averaging across the three layers and then normalized to an 8-bit format using the `cv2.normalize` function.

2. The Otsu thresholding method (`skimage.filters.threshold_otsu`) is used to create a binary mask, followed by inversion to highlight the tumor tissue model region.

3. Contours are extracted from the binary mask using `cv2.findContours`. The largest contour, corresponding to the tumor tissue model or debris, is identified by sorting the contours based on area.

4. A bounding rectangle is determined around the largest contour using `cv2.boundingRect`. The rectangle is then adjusted to ensure the crop is centered around the detected region and fits within the 2054×2054 dimensions. The image is then cropped, ensuring that the tumor tissue model and debris are fully included in the crop. Figure 7.3 demonstrates the whole procedure.

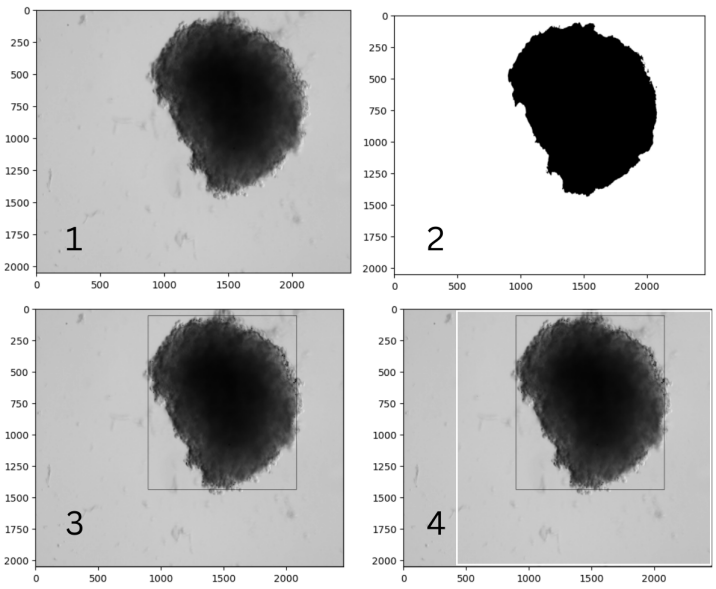

Figure 7.3: Pipeline for careful cropping of the image.

3. **Normalize the 16-bit image to [0, 1]**

4. **Perform data augmentations** which detaily explained in the section 7.2

5. **Perform Z-score normalization** after data augmentation since:

   a) Pretrained models require this preprocessing.

   b) It ensures that the data is still normalized even after data augmentation tweaks, allowing for effective feeding into the neural network.

6. For each original image, repeat step 2 twice to obtain two augmented images.

7.2 Data augmentation

I started with best data augmentation pipeline that proposed by SimCLR [2] paper did, because it gave best performance on natural image dataset such as Imagenet and CIFAR for downstream task like classification.

Which follows a sequential of:

1. Randomly crop and resize to a specific smaller size. In my case, I chose to crop and resize to 96×96. I selected this small height and width for the image size to train our model because achieving good performance with these small image sizes (i.e., fewer pixel details compared to 2054×2056) may indicate that we can improve performance with larger image sizes. Additionally, it's computationally efficient. The image is randomly cropped uniformly from 0.08 to 1.0 in area of the original image and maintains a random aspect ratio (default: from 3/4 to 4/3) of the original aspect ratio. This crop is then resized to its original size.

2. Apply a horizontal flip (the SimCLR paper only used horizontal flip because it doesn't make sense to have vertical flip/rotation in natural images). This is the default policy in SimCLR as it improves 1 percentage accuracy for downstream tasks in the SimCLR paper. We can also apply a vertical flip (as flipping the tumor tissue model vertically still results in valid samples that belong to our image distribution.), combination of horizontal and vertical flip, rotation of (90, 270), or a combination of horizontal flip + rotation (90, 270) with a probability of 50 percent. We include these additional augmentations because they still fit within our distribution samples. Other flip and rotation combinations are restricted due to their repetitive pattern, as shown in figure 7.5, and the black cut problem explained in figure 7.2. We set this as our default for all data augmentation.

3. Randomly change the brightness and contrast up to lower=1-0.8, upper=1+0.8, with a probability of 80 percent. I removed saturation and hue since they do not affect grayscale images. Grayscale images are neutral, and the saturation and hue of a grayscale image are zero. Thus, when we try to adjust the values, we essentially have no color to modify as it is achromatic.

4. Apply Gaussian blur and increase sharpness. Blur augmentation is also the default policy in SimCLR because it has been found to improve performance for downstream linear classification tasks by around 2 percent. Like them, we randomly sample sigma from 0.1 to 2. We also keep the kernel size to be 10 percent of the image height/width, which is 5 in our case. Some images are blurred (probably due to a microscopic error). To mimic the original version, I added an increase in sharpness, which doubles the sharpness factor. We set this as our default for all data augmentation. Figure 7.4 shows the original and corresponding sharpened image.

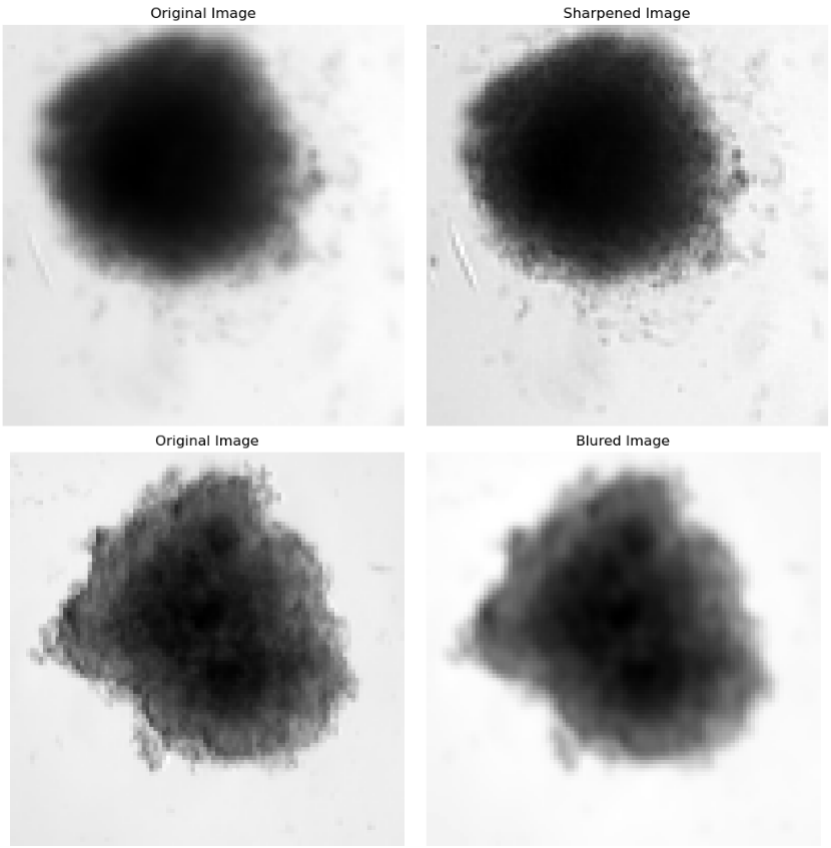

Figure 7.4: Applying blurriness and sharpness to the original images: The higher the sharpness, the more texture and edge details are enhanced, while blurriness results in the opposite effect

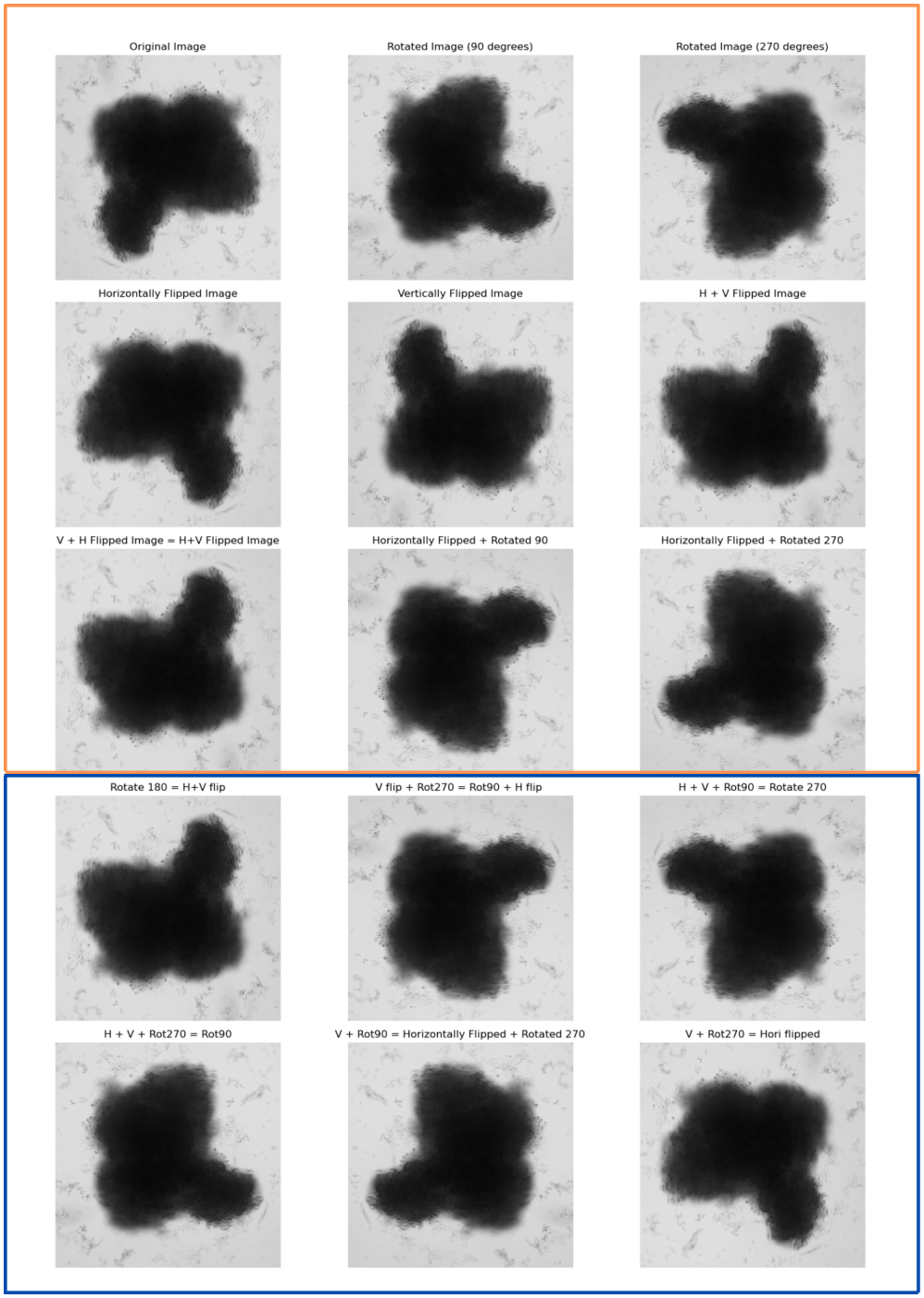

Figure 7.5: Orange box consist of implemented flips and rotations. Blue box consist of avoided flips and rotation combinations because of their repeated pattern to the orange box augmentations.

I refer to the above chain of data augmentation parameters as 'strong' data augmentation, as illustrated in the first row of Figure 7.11.

Since there is a significant intensity change in brightness and contrast for 'strong' augmentation,

I decided to analyze the effect of only the brightness and contrast changes. For 16-bit images with 3 channels, there was an increase in the number of unique pixel values—by a maximum of 258,757 percent in one of the changes in the combination of brightness and contrast. This is due to the fact that after data augmentation, the new pixel values are not distributed similarly to the original image. Instead, they shift toward the two extremes, such as 0 or 65535, deviating significantly from the original image distribution, as shown in figure 7.6.

16-bit three-channel image before and after data augmentation:

- Number of unique pixel values in the original image: 2111
- Number of unique pixel values in the augmented image: 5044624

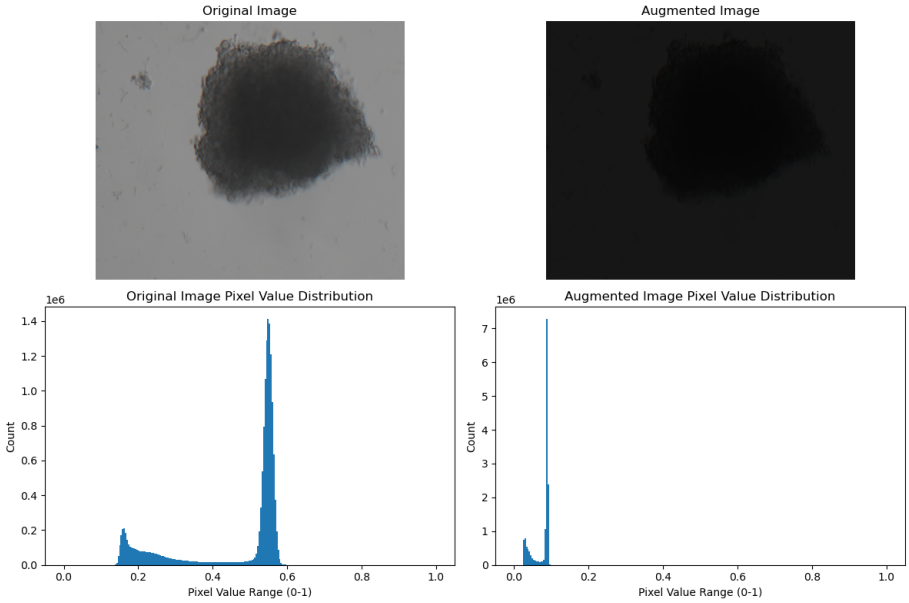

Figure 7.6: 16-bit three-channel image in one of the brightness and contrast changes according to 'strong' augmentation
pipeline parameters parameters

As seen in Figure 7.6, I found that the 'strong' combination of brightness and contrast augmentations resulted in a distribution that deviates from the original distribution. Therefore, it is a good idea to experiment with less intense changes, instead of using the best-performing data augmentation parameters from the SimCLR paper. Additionally, in the SimCLR paper, random crop resizing is first done by randomly cropping a uniform size ranging from 0.08 to 1 of the area of the original image. This means that even very small crops without a tumor tissue model in them are trained to have high cosine similarity to a corresponding smaller crop patch of the

tumor tissue model positioned in the center of the image, due to the design of the loss function in SimCLR. This may doesn't make sense in terms of representation learning, as we have a common background (the inner wellplate) in all of our images, and attaching the background to the tumor tissue model doesn't provide any unique information. (In natural images, at least cars are typically seen on roads, so relating cars or bikes to roads makes sense, and ships float in water, so attaching the background to an object in natural images is meaningful.) Hence, the next data augmentation sequence was designed to have a random crop with a scale of 0.4 to 1 of original image (so that crop patches will always contain at least a small portion of the tumor tissue model), and brightness and contrast variations reduced to lower=1-0.2, upper=1+0.2 and lower=1-0.35,upper=1+0.35 consequently as showed in figures 7.7 and 7.8 to maintain the original distribution after augmentations. I named this augmentation as 'sweet'. which illustrated in 7.11 second row.

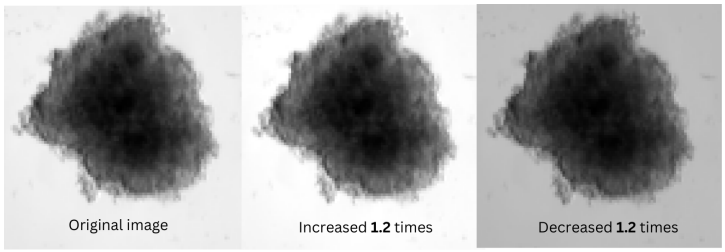

Figure 7.7: Contrast increased or decreased was applied to the original image

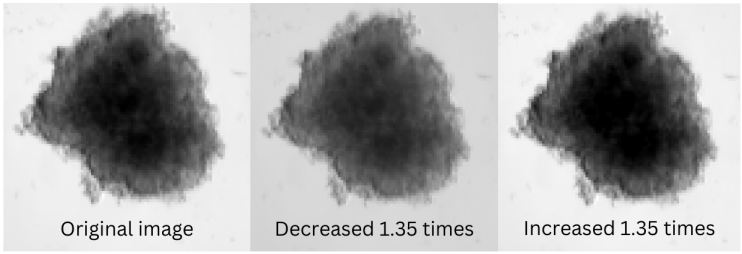

Figure 7.8: Brightness increased or decreased was applied to the original image

If we apply the random crop and resize augmentation on 'explode' images, it's possible that smaller crop patches from the image containing spheroid tumor tissue models without debris will have high cosine similarity with other small crops from the image, which consist only of background without debris, as shown in Figure 7.9. This does not efficiently learn the total spread of debris around the tumor tissue model. It may be better to avoid cropping and resizing to 96x96; instead, we should always resize to 96x96 so that the model learns to map

high cosine similarity by observing the entire image, including all of the debris, assuming it learns something about variations in the amount of debris. This could be beneficial for the time prediction ranking model, as it needs to predict changes from day 7 to day 10 based on the shape and structure of changes induced by drug application. With this assumption in mind, I decided to implement the data augmentation pipeline using only resizing. All other data augmentations remain the same as 'sweet'.

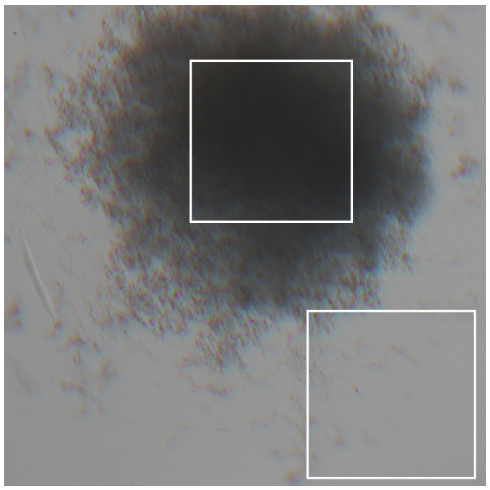

Figure 7.9: 5 Data augmentations we explored

I named this augmentation as 'Resize', which illustrated in 7.11 third row.

All of the above augmentations include contrast changes. With this augmentation, it is possible that the SimCLR model has learned to represent both the darker sections of image and the gray sections of the same image to have similar feature representations (because two data augmentations from the original image may consist of one with increased contrast that results in darker tissue and one with decreased contrast that results in gray tissue, as seen in Figure 7.10). This may not actually be beneficial for our downstream task, since one of our goals is to differentiate between untreated images (gray color) and single-dose images (darker color). Moreover, it will reduce the performance intermediate evaluations and ranking strategies if we have similar latent representations for both single-dose and control images.

Figure 7.10: Contrast was increased or decreased on the original control Day 10 image. The contrast-decreased image resembles the control, while the contrast-increased image resembles the treated image

Hence I removed contrast from 'Resize' and 'Sweet' and named them as 'Resize no contrast' and 'Sweet no contrast' correspondingly. which illustrated in 7.11 third row and fourth row correspondingly. Although we know that 'strong' augmentation is sensitive to our dataset, I speculate it may lead to better-learned latent space representations by capturing features that are invariant to both forms of augmentation of the same original image. For example, when contrast changes in both augmentations derived from the same original image, the model learns to achieve high cosine similarity from the same shape; if cropping occurs, it learns to maintain high cosine similarity for the same pixel-wise values. Hence its worth to experiment with this strong augmentation for downstream tasks.5 Different data augmentation pipe lines explained above illlustrated as nutshell in the table 7.11

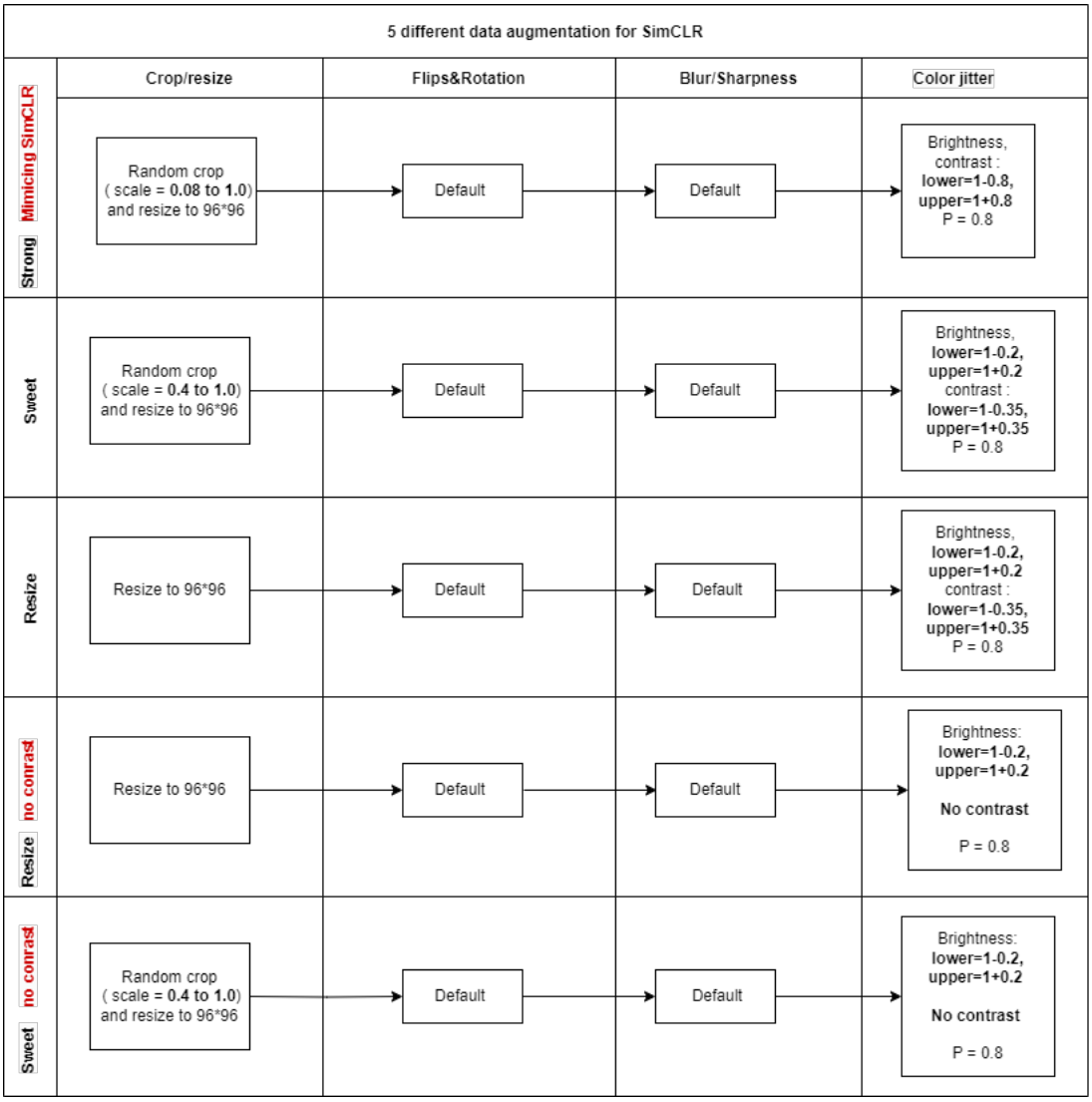

Figure 7.11: 5 different data augmentation pipelines we explored

7.3 Train SimCLR as SSL model

As explained in chapter 7, SimCLR was used as the model for self-supervised learning (SSL).

Model

The Resnet18 [**he2015deepresiduallearningimage**] model processes a single image to produce a latent representation of the input, aiming to cluster similar images together in a latent space.

Training

The training process follows these steps:

1. We take a batch of images with batch size N.
2. Our dataset class returns two augmented versions for each original image as explained in section 7.1 in the batch, resulting in $2N$ images as input.
3. The model produces $2N$ latent representations, independently for each augmented image.
4. For each batch, the two augmentations of the same image are treated as positive pairs, while all others are considered negative pairs.
5. We calculate the cosine similarities between the positive and negative pairs. These cosine similarities are then used as input to the loss function described below equation 7.2

The original loss function for each pair from SimCLR paper [2] is defined as:

$$\ell_{i,j} = -\log \frac{\exp\left(\operatorname{sim}\left(z_i, z_j\right)/\tau\right)}{\sum_{k=1}^{2N} \mathbf{1}_{[k\neq i]} \exp\left(\operatorname{sim}\left(z_i, z_k\right)/\tau\right)} \tag{7.1}$$

where $\mathbf{1}_{[k\neq i]} \in \{0,1\}$ is an indicator function evaluating to 1 iff $k \neq i$, and τ denotes a temperature parameter. The final loss is computed across all positive pairs, both (i,j) and (j,i), in a mini-batch with z_i, z_k representing negative pairs.

Intituivly given a set of $\{z_k\}$ with the batch size 2N, that includes a positive pair of examples z_i and z_j, the loss function aims to identify z_j in $\{z_k\}_{k\neq i}$ for a given z_i.

Above loss function we can reformualte as:

1. Apply the logarithm: The negative log of a fraction can be separated into the difference of the logarithms:

$$\ell_{i,j} = -\left(\log\left(\exp\left(\operatorname{sim}\left(z_i,z_j\right)/\tau\right)\right) - \log\left(\sum_{k=1}^{2N}\mathbf{1}_{[k\neq i]}\exp\left(\operatorname{sim}\left(z_i,z_k\right)/\tau\right)\right)\right)$$

$$\ell_{i,j} = -\log\left(\exp\left(\operatorname{sim}\left(z_i,z_j\right)/\tau\right)\right) + \log\left(\sum_{k=1}^{2N}\mathbf{1}_{[k\neq i]}\exp\left(\operatorname{sim}\left(z_i,z_k\right)/\tau\right)\right)$$

2. Simplifying the first term: The logarithm of an exponential function simplifies as follows:

$$-\log\left(\exp\left(\operatorname{sim}\left(z_i,z_j\right)/\tau\right)\right) = -\frac{\operatorname{sim}\left(z_i,z_j\right)}{\tau}$$

Substituting that back into the equation gives us:

$$\ell_{i,j} = -\frac{\operatorname{sim}\left(z_i,z_j\right)}{\tau} + \log\left(\sum_{k=1}^{2N}\mathbf{1}_{[k\neq i]}\exp\left(\operatorname{sim}\left(z_i,z_k\right)/\tau\right)\right) \tag{7.2}$$

Equation 7.2 is implemented as the loss function in our experiments.

The SimCLR framework was originally implemented in TensorFlow by the authors. In this work, I adopted the PyTorch implementation of SimCLR and utilized the parameters provided in the PyTorch SimCLR tutorial [9] , as they demonstrated optimal performance for the STL10 dataset. The STL10 dataset is still natural image dataset, with an image resolution of 96×96, matches the dimensionality of the our images that feeded into the model. I used the following

parameters for training from the tutorial: a learning rate of 5×10^{-4}, a temperature of 0.07, a weight decay of 1×10^{-4}, a cosine learning rate schedule with a minimum learning rate of $\frac{\text{lr}}{50}$ and T_{max} set to the maximum number of epochs where T_{max} refers to the maximum number of iterations or epochs for which the cosine learning rate schedule is defined. It determines the period of the cosine function used to anneal the learning rate. the ADAM optimizer, and 4 hidden layers with a hidden dimension of 128 in the projection head also used. Additionally I used max epoch as 245 and Batch size as 64.

The following explains how we process a 3-channel image as if it were a standard RGB image and apply augmentations to generate two augmented versions.

Input to model (train loader dimension) :

- aug1: torch.Size([64, 3, 96, 96]) (batch size, no of channels, H, W)
- aug2: torch.Size([64, 3, 96, 96]) (batch size, no of channels, H, W)

Model output just after convolution layers: (before applying projetion head)

- torch.Size([64, 512, 1, 1]) (Batch size, standard resnet18 output dimension after avg pooling, H,W)

Model output after projection head:

- torch.Size([64, 20]) (Batch size, no of values in feature vector)

Both output feature before and after projection head will be used for further downstream task to check the performance.

7.4 Evaluation of SimCLR training

Top-1 Accuracy:

Top-1 accuracy measures how often the loss function correctly identifies z_j in $\{z_k\}_{k\neq i}$ for a given z_i. Specifically, it evaluates how often the model ranks z_j and z_i as having the highest cosine similarity in a given set $\{z_k\}$, where the batch size is $2N$ while the set includes a positive pair of examples z_i and z_j. If the positive pair z_j and z_i is ranked first (i.e., has the highest cosine similarity), it is counted as correct. Top-1 accuracy is calculated as the mean of these correct counts over all samples in the batch.

Purpose: Indicates how often the model correctly identifies the positive example as the most similar, reflecting the model's precision at a fine-grained level.

Top-5 Accuracy:

Top-5 accuracy measures how often the loss function identifies z_j in $\{z_k\}_{k\neq i}$ for a given z_i within the top 5 ranked pairs. Specifically, it evaluates whether z_j is ranked among the 5 highest cosine similarity scores in $\{z_k\}$, where the batch size is $2N$, and the set includes a positive pair z_i and z_j. If the positive pair z_j and z_i is ranked within the top 5, it is counted as correct. Top-5 accuracy is the mean of these correct counts over all samples in the batch.

Purpose: Evaluates the model's ability to identify the true positive within a broader range of candidates, providing a softer measure of precision compared to Top-1 accuracy.

Mean Position: For each pair, the ranking is calculated based on its cosine similarity relative to all other z_k in $\{z_k\}_{k\neq i}$. Calculate the ranking position of the positive pair z_j and z_i across all sample pairs in the batch. The mean position is the average of these ranking positions.

Purpose: Provides a quantitative metric for how far down the ranked list the positive pairs typically appears, offering insight into the quality of the model's ranking performance. Figures 7.12, 7.13, 7.14, 7.15, and 7.16 illustrate the top-1 accuracy, top-5 accuracy, and mean position for the training, validation, and inference datasets, respectively, across the 'strong,' 'sweet,' 'sweet no contrast,' 'resize no contrast,' and 'resize' data augmentation pipelines. In the figures: red, blue and green curves are corresponding to validation, train, inference dataset. From these figures, the following insights can be concluded:

While training and validation images are processed using data augmentation pipelines such

as 'strong', 'sweet', 'resize', 'sweet no contrast', and 'resize no contrast', inference images are only resized to 96x96, meaning there are no additional data augmentations. This means that when calculating the ranking position, top 1 accuracy, top 5 accuracy, or loss, the model performs inference on pairs of images that are always full images without any cropping. When comparing strong augmentation to the others, we can observe that the loss, top accuracies, or mean position for the training and validation datasets perform less than others. This is because 'strong' has to learn harder than other augmentations since two different augs of the same image in strong may have completely different image distributions between them. For example, one might have a highly increased contrast, while the other has a highly decreased contrast. Another example: one has a very small portion of the background, and the other has a small portion of a tumor tissue model that is just pure black, which doesn't have any certain pattern or feature relation to the same image. These small portions of black can also be present in the batch from another image as an augmentation of a different image. Therefore, it has to learn to map high cosine similarities for these two different augmentations. Specifically, the top accuracy struggles to get close to 100 compared to others. Another interesting thing to note is the top accuracy and mean position for the inference dataset. From the beginning of the epoch, inference top accuracies or mean positions are close to 100 or 1, respectively. This is because inference images are just resized and do not have any kind of brightness, contrast, blur/sharpness, or flip/rotation applied as augmentation; they are just the original images but with a reduced size of 96x96. While ranking or calculating the similarity between these pairs, the model gets to see the full picture to calculate the cosine similarity. That also reflects why 'Resize' and 'resize no contrast' data augmentation pipelines which doesn't have cropping have better performance in terms of training and validation loss, as well as top accuracies or mean positions than data augmentation pipelines that include cropping. However, there remains an open question about whether the model learns any effective features for downstream tasks.

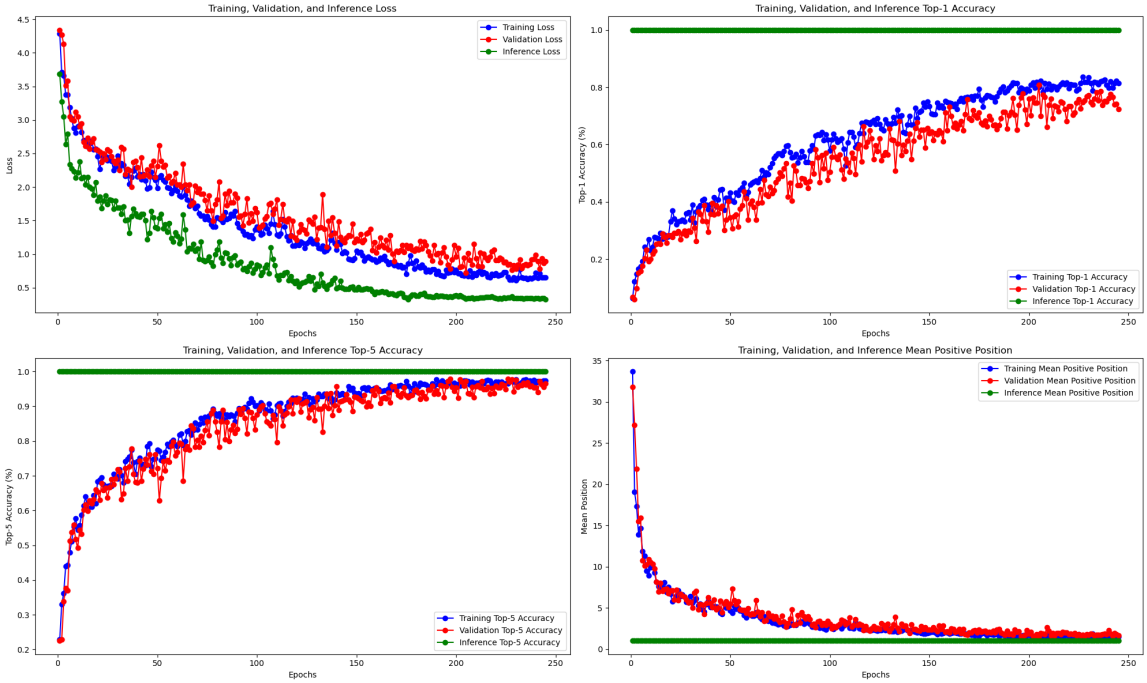

Figure 7.12: strong: loss curve on top left, top-1 accuracy on top right, top-5 accuracy on bottom left, and mean position on bottom right

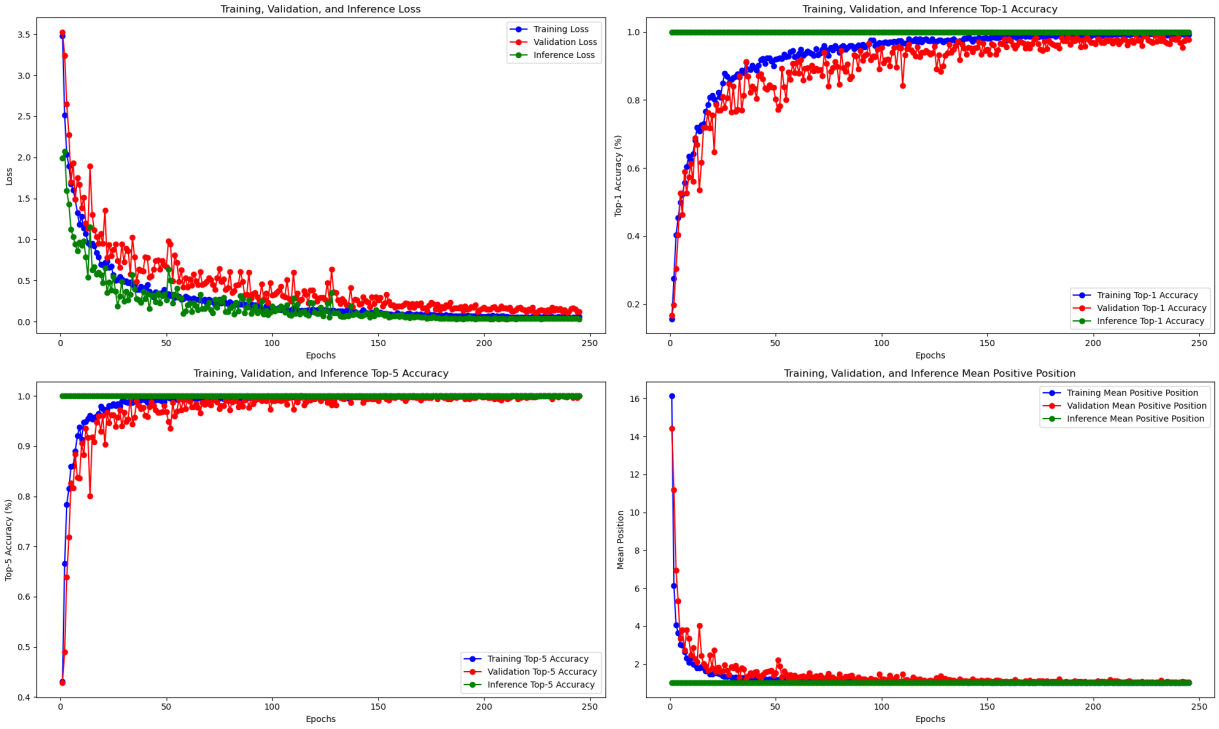

Figure 7.13: sweet: loss curve on top left, top-1 accuracy on top right, top-5 accuracy on bottom left, and mean position on bottom right

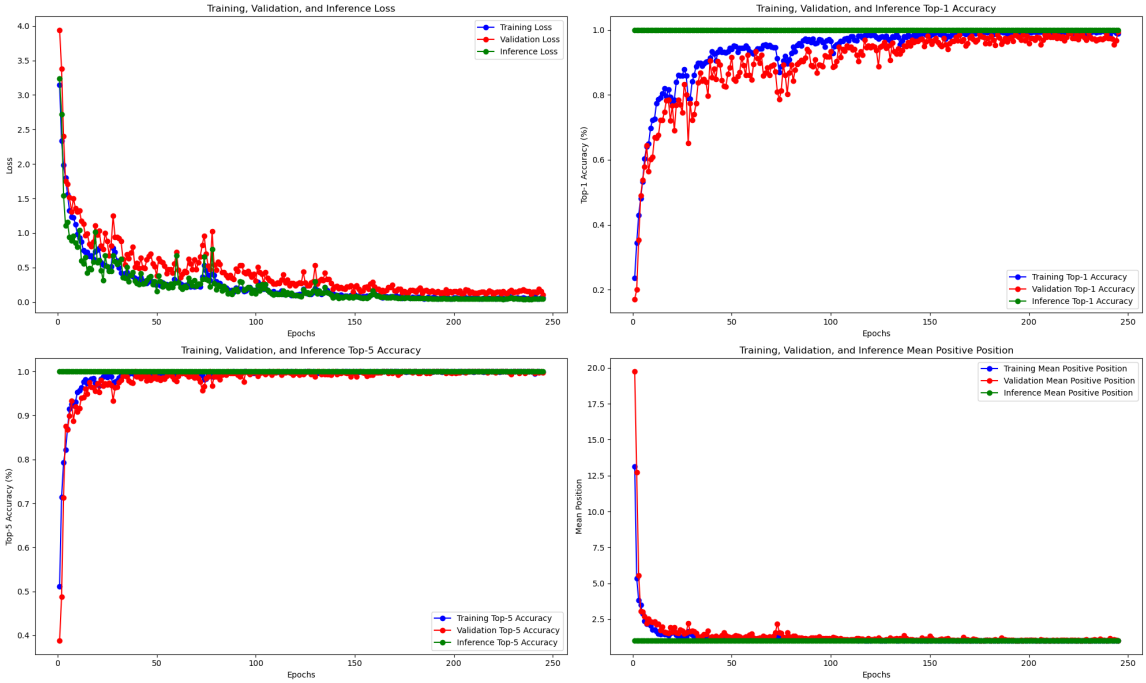

Figure 7.14: sweet no contrast: : loss curve on top left, top-1 accuracy on top right, top-5 accuracy on bottom left, and mean position on bottom right

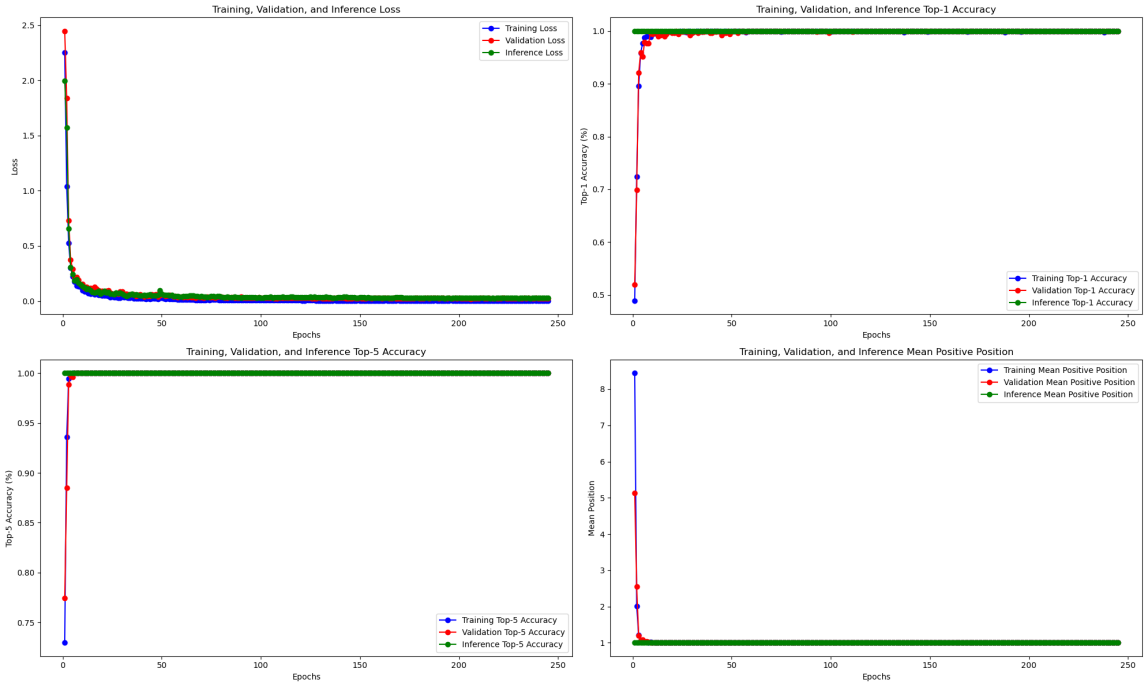

Figure 7.15: Resize no contrast: : loss curve on top left, top-1 accuracy on top right, top-5 accuracy on bottom left, and mean position on bottom right

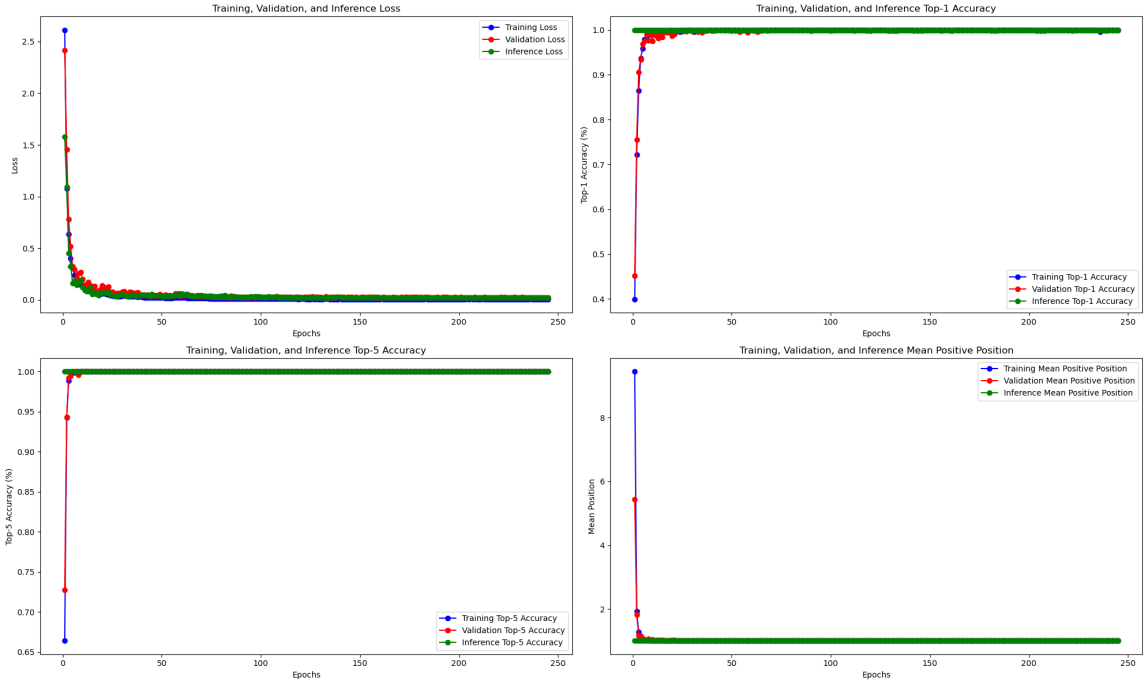

Figure 7.16: Resize: : loss curve on top left, top-1 accuracy on top right, top-5 accuracy on bottom left, and mean position on bottom right

In Figure 7.17, we compare the PCA visualizations of SimCLR feature vectors and original image vectors for the classes: control, single dose, and explode. From the figure, we observe that, except for the 'resize' and 'resize no contrast' pipelines, all other data augmentation pipelines using SimCLR vectors before the projection head are able to clearly separate these classes better compared to SimCLR vectors after the projection head or the original image vectors. Among these, the 'sweet no contrast' pipeline demonstrates the most distinct separation of the classes.

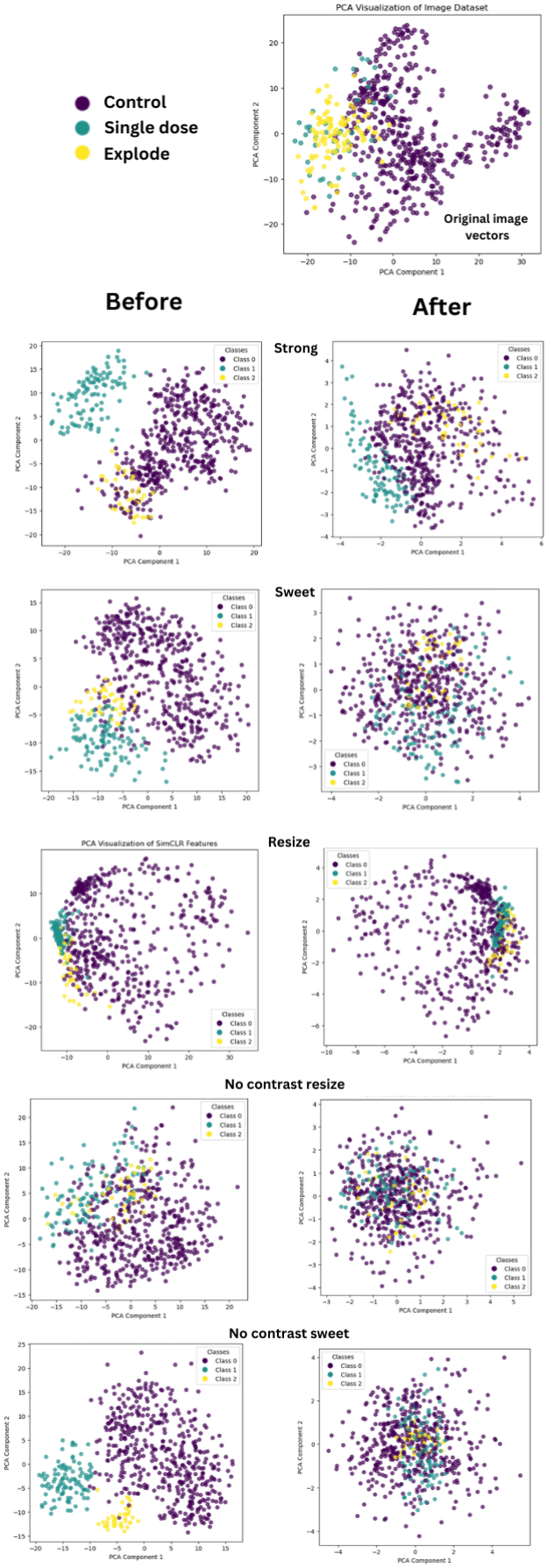

Figure 7.17: PCA visualisation of both after and before projection of the SimCLR feature vector for the classes control, single dose and explode images

8 Methodology for Intermediate evaluation of SimCLR model

The final evaluation of the SimCLR model depends on the ranking task. Nevertheless, other evaluation metrics, such as downstream tasks like classification and clustering, can be used to verify if SimCLR has effectively learned to differentiate between control, single dose, and explode images.

8.1 Classification using Logistic Regression on the SimCLR features and original image vectors

1. A common approach to verify whether the SSL model has learned generalized representations is to perform Logistic Regression on the learned features. In other words, we use a single linear layer that maps these representations to class predictions, where the three categories, 'control' and 'single dose', 'explode' serve as our classes. The Logistic regression model can only perform well if the learned representations capture all the relevant features necessary for the task. Moreover, we don't need to worry much about overfitting since only a few parameters are trained. Therefore, we expect the model to perform well even with limited data.

2. Baseline comparison to original images. We classify the original image vectors to see if SimCLR feature vectors are better or worse than original image to classify.

The logistic regression model is implemented as a single linear layer, where the input dimension corresponds to the feature dimension of feature spits out from SimCLR, and the output

dimension corresponds to the number of classes. Specifically, the model uses a feature dimension of 512 if the feature is before projection head or 20 if the feature is after projection head and outputs predictions for 3 classes. The mathematical representation of the model is as follows:

$$\hat{y} = xW^T + b$$

where:

- $x \in \mathbb{R}^{N \times d}$ is the input feature vector, where $d = 512$ if the feature is extracted before the projection head, or $d = 20$ if the feature is extracted after the projection head. - $W \in \mathbb{R}^{3 \times d}$ represents the learnable weights. - $b \in \mathbb{R}^3$ is the bias term, - $\hat{y} \in \mathbb{R}^{N \times 3}$ are the logits representing the unnormalized class scores for N samples.

This model is trained to minimize the cross-entropy loss for multi-class classification. 3 classes trained for 250 epochs. The dataset was divided into a training set (80%) and a validation set (20%). Batch size = 8. A learning rate of 5×10^{-4}.

The learning rate scheduler used is the `MultiStepLR`, which reduces the learning rate at specific milestones during training. In this case, the learning rate is reduced by a factor of 0.1 at the epochs corresponding to 60% and 80% of the total training epochs. These milestones are defined as:

$$\text{milestones} = [\text{int}(T_{\max} \times 0.6), \text{int}(T_{\max} \times 0.8)]$$

where $T_{\max}$ represents the total number of training epochs. The `gamma` parameter specifies the factor by which the learning rate is multiplied at each milestone, which in this case is 0.1.

Table 8.1: Dataset Summary from Drug Screening Experiments

Dataset	**Total Number of Images**
Control Dataset	472
Single Dose Dataset	103
Explode dataset from Drug Screening Experiments	40

Table 8.7 shows the total number of images in each dataset used for classification. Additional images from drug screening experiments were not included, as their class labels are uncertain.

Some images exhibit low to medium resemblance to the three defined classes, while others differ significantly, making it difficult to use them for classification task with confidence.

8.1.1 Comparison of Classification Accuracy and Epochs for Different Data Augmentations

As explained in the data augmentation section, we use 'strong,' 'sweet,' 'resize,' 'sweet no contrast,' and 'resize no contrast' SimCLR features to compare their performance against each other and the original images in classifying the three defined classes. We calculate the training accuracy and testing accuracy for each data augmentation pipeline as well as for the original image vectors. Train epoch in the table: The number of epochs required to reach the best training accuracy. Test epoch in the table: The number of epochs required to reach the best test accuracy. To use original images directly first we resized to 96*96 since we did that for simclr so that it could be fair comparison. Then we normalised each by dividing it by 65535 (16 bit). Each image flatten into vectors of ([1,96*96*3]where 96 =H=W and 3 = no of channels) suitable for logistic regression model. and we use same parameters that we used for simclr feats classification lke learning rate , batch size etc for fair comparison. From table 8.2 and table 8.3 we show that when using the features Before Projection Head all augmentation methods reached 100 train and test accuracy. And when using the features original image vectors both accuracy reached upto 99 percentage. So comparing before projection head SimCLR features to original image vectors there is only subtle difference. Their almost indistinguishable performance leads to the conclusion that it is difficult to determine whether SimCLR features have been learned efficiently from classification task. But it is clear that the features extracted before the projection head perform far better than those extracted after the projection head, as seen in the SimCLR [2]paper. Also, the number of training and test epochs required to reach the best accuracy is lower for the features extracted before the projection head than those extracted after the projection head as we seen in figures 8.1 and 8.2. Interestingly, original image vectors performed better than the after projection head SimCLR features.

Augmentation Type	Metric	Strong	Sweet	Resize	Resize No Contrast	Sweet No Contrast
After Projection Head	Train Accuracy (%)	66.67	47.76	89.63	63.41	54.67
	Train Epoch	246	249	250	250	245
	Test Accuracy (%)	68.29	55.28	90.24	63.41	56.10
	Validation Epoch	228	246	223	202	244
Before Projection Head	Train Accuracy (%)	100	100	100	100	100
	Train Epoch	190	3	13	12	21
	Validation Accuracy (%)	100	100	100	100	100
	Validation Epoch	1	1	1	2	2

Table 8.2: Classification performance metrics for different augmentation strategies before and after the projection head using SimCLR

Metric	Train Accuracy (%)	Train Epoch	Validation Accuracy (%)	Validation Epoch
Original Image	99.39	249	99.19	17

Table 8.3: Classification performance metrics for original image vectors

Since the train and test accuracy are both 100 percent for before projection head features, which may indicate overfitting, I performed a 10-fold cross-validation as an additional check to ensure the model's performance generalizes well. The results are shown in Tables 8.4 and 8.5. The train and test accuracy for the features before the projection head drops from 100 percent to around 99 percent, and for the original image vectors, it drops from 99 percent to approximately 96 percent. This shows that the trend of the features before the projection head being slightly superior, as observed in earlier results, is also reflected here. Additionally, the after-projection head features still perform considerably worse than the before-projection head and original image vectors. Comparing this with classification using the ResNet18 architecture shows that the before-projection head SimCLR feature vectors and ResNet18 feature vectors achieve similar performance. However, between the data augmentation pipelines when using the features before the projection head, it is difficult to evaluate which one is better, as all of them show subtle differences in accuracy.

Augmentation Type	Metric	Strong	Sweet	Resize	Resize No Contrast	Sweet No Contrast
After Projection Head	Average train Accuracy (%)	35.17	36.60	49.44	35.28	34.57
	Average test Accuracy (%)	33.48	34.62	54.43	33.77	34.65
Before Projection Head	Average train Accuracy (%)	99.81	99.90	99.86	99.90	99.89
	Average validation Accuracy (%)	99.71	99.77	99.49	99.72	99.61

Table 8.4: Classification performance metrics for different augmentation strategies before and after the projection head using cross validation with 10 k folds

Metric	Logistic Reg Average Accuracy (%)	ResNet18 Average Accuracy (%)
Train	96.73	99.86
Validation	96.50	99.93

Table 8.5: Classification performance metrics for original image vectors using cross-validation with 10 k folds

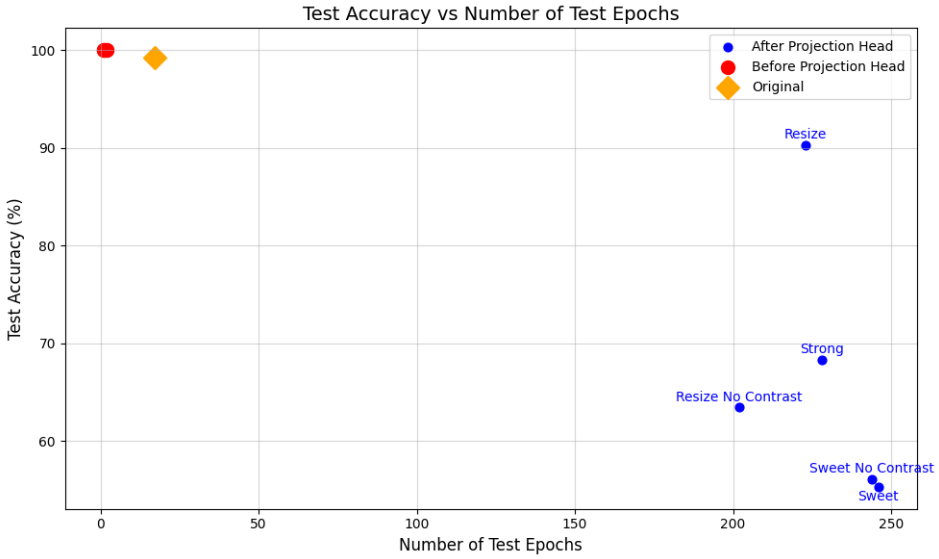

Figure 8.1: Test accuracy vs Test epochs of original image vectors and Simclr features

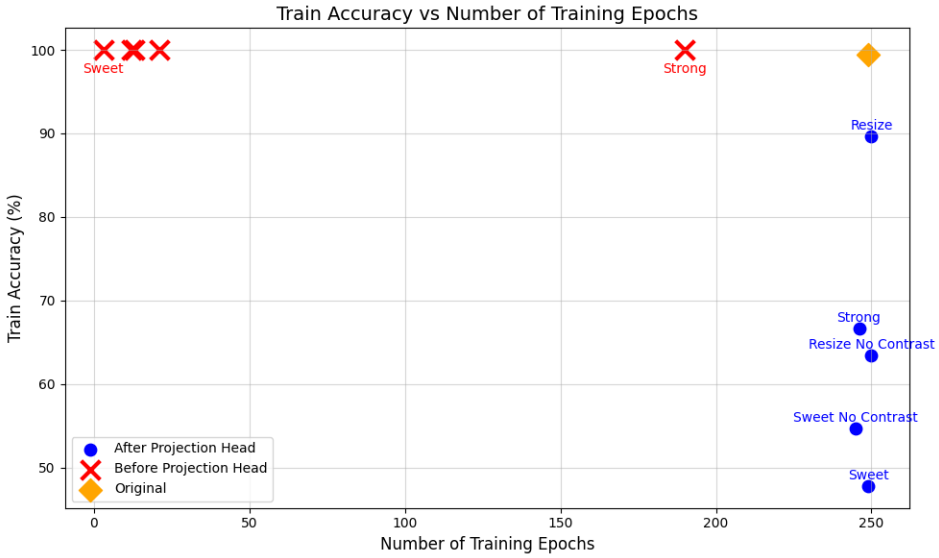

Figure 8.2: Train accuracy vs Train epochs of original image vectors and Simclr features

8.1.2 Inference

Twenty-two images that closely resemble single-dose images from drug screening were manually selected for inference and refer them as 'inference images'. These images were not labeled by biology experts, and therefore, any inference results do not correspond to the actual ground truth. Note that this inference is from the first experiment of classification where cross validation training didn't used. For the inference process, features before the projection head were utilized, as they achieved 100 percent accuracy during training and validation.

Type	**Strong**	**Sweet**	**Resize**	**Resize no Contrast**	**Sweet no Contrast**
Before Projection Head	90.91	100.00	100.00	100.00	95.45
Original Images				68.18	

Table 8.6: Performance Metrics Before and After the Projection Head

From the inference experiment table 8.6, it can be seen that, except for 'strong' and 'sweet no contrast', all other augmentations were able to achieve 100 percent accuracy in classifying the inference images as belonging to the single-dose image class. Using the original image features, the accuracy was 68.18 percent, while SimCLR features before the projection head achieved at least 90 percent accuracy.

8.2 K-means clustering

The idea is to determine whether the learned representation from SimCLR outperforms the original images in clustering the images in an unsupervised manner. To achieve this, we use simple K-means clustering as explained in [1]. We use both euclidean distance and cosine distance to find the nearest data point to the cluster centroid as two different approach. Derivation of Objective function for both cosine distance based and euclidean distance based approach explained in appendix 10.1. When performing K-means with cosine distance, it compares only the direction of the vector. Therefore, we also perform standard K-means with Euclidean distance to evaluate the magnitude similarity.

8.2.1 Evaluation

Since there is a class imbalance in the control, single dose and explode classes. Its also useful to cluster balanced subset. Additionally, there are some images in control, where debris is formulated, it can be attributed due to the environmental factors like cell medium. Which is noticed to be significantly less than the amount of debris formed around the treated spheroids as we seen in explode images. Assuming that these images can be misclustered as exploded images or vice versa, it is beneficial to cluster a dataset containing only control images without any debris. Concluding above assumptions, for clustering, we use below dataset categories.

Imbalance full dataset: contains all control: 472, all single dose: 103, all explode: 40 images.

Balance uncured 40 subset (Balanced to the minimum number of set in the class which is explode): control contains 20 explode look alike controls + 20 normal control images without any debris, all 40 explode image and 40 single dose images.

Balance curated 40 subset (Balanced to the minimum number of set in the class which is explode): contains randomly selected control 40 images without and debris, all 40 explode, and 40 single dose images. These category of dataset showed in the table 8.7.

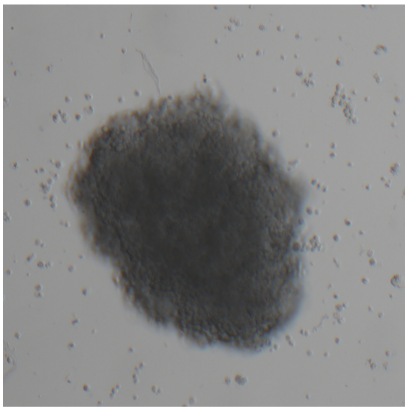

Figure 8.3: Example of control images which contains debris which may miscluster with explode or viseversa

Table 8.7: Summary of Datasets

Dataset	**Control (C)**	**Single Dose (SD)**	**Exploded (E)**
(Imbalance) uncurated Full Dataset	472	103	40
Uncurated 40 Subset (Balanced)	(20 with debris + 20 without debris)	40	40
Curated 40 Subset (Balanced)	40 (all without debris)	40	40

K-means was run 400 times with different random initializations.

8.3 Evaluation of cosine distance based K-means:

Table 8.8: Evaluation Results on different datasets and augmentation pipeline with cosine distance

Projection Head	Augmentation Type	Full Dataset (Unbalanced)	40 Subset (Balanced)	Curated 40 Subset
Before	Strong	**83.90**	**100**	**100**
	Sweet	**99.18**	**100**	**100**
	Resize	65.7	97.5	100
	Resize No Contrast	69.92	90	100
	Sweet No Contrast	**98.21**	**99.17**	**99.17**
After	Strong	72.68	95.83	100
	Sweet	47.32	74.17	78.33
	Resize	60.32	92.50	100
	Resize No Contrast	44.72	64.17	66.67
	Sweet No Contrast	52.20	86.67	89.17

The results in table 8.8 show a clear trend of superior clustering performance before the projection head compared to after the projection head.

Among before projection head evaluation:

Data augmentation ‘sweet’ performs the best over other data augmentations and achieved consistent accuracy over all different data set types. Since sweet and strong performed 100 percentage accuracy across cured and uncured balanced dataset its evident that the problem for having less accuracy is nothing to do with explode look alike images (with debris) in control instead it could be imbalance class problem. Comparing ’sweet’ and ’strong’ they both achieved 100 percentage accuracy ie (same performance) over all different data set types except the full dataset. for full dataset sweet performed far better than strong , sweet even achieved 99.18 where strong reached only 83.90. Comparing all data augmentations ’resize’ and ’resize no contrast performed the least giving the insight that cropping is essential to get better cluster performance using cosine distance as distance metric. As a brief conclusion its pretty clear that cropping 0.35 to 1 percentage of whole image is crucial especially cropping have huge effect for cluster performance using cosine distance as distance metric. One thing that need to keep in mind that all above explanations or achievements by different data augmentations are

corresponds to clustering performance doesn't mean that it could work for ranking strategies too.

8.4 Evaluation of euclidean distance based K-means:

The results in table 8.9 shows just like cosine distance based K-means, with euclidean distance as distance metric also established a superior performance trend for before the projection head compared to after the projection head SimCLR features. Just like cosine distance based K-means, between the before projection head, data augmentation pipeline with cropping have far better performance than without cropping data augmentations such as resize and resize no contrast. Even though 'strong' achieved 100 percentage accuracy across all dataset types except the full dataset imbalanced. when it comes to full dataset, sweet and sweet no contrast performs better than all others. And among all data augmentation we can see 'sweet no contrast' maintain the constance in performance across all dataset types with more than 97 percentage accuracy and also performed better than just 'sweet' across all dataset. Interestingly after projection head 'strong' achieved higher clustering performance 99 and above across all dataset types except the full dataset. which is a good indication that we may also consider the after projection head features for ranking task.

Table 8.9: Evaluation Results on Different Datasets and Augmentations with Euclidean Distance

Projection Head	Augmentation Type	Full Dataset (Unbalanced)	40 Subset (Balanced)	Curated 40 Subset
Before	Strong	**88.45**	**100**	**100**
	Sweet	**98.21**	**91.67**	**95.83**
	Resize	70.40	78.33	88.33
	Resize No Contrast	68.62	90.00	100
	Sweet No Contrast	**99.02**	**97.50**	**99.17**
After	Strong	**75.45**	**100**	**100**
	Sweet	51.54	74.17	79.17
	Resize	43.25	83.33	87.5
	Resize No Contrast	47.8	63.33	65.83
	Sweet No Contrast	51.87	85.0	88.33

8.5 Evaluation of original images:

From the table 8.10, comparing original images features to SimCLR features, we can see that entire data augmentation for before head features outperformed entirely for both cosine distance and euclidean distance based K-means. While comparing original image vectors to after projection head SimCLR features the performance is comparable except for the strong data augmentations where after head still have better performance. Below figure 8.4 demonstrates the above findings in a nutshell. From the figure its evident that Before projection head SimCLR features perform better than after projections head SimCLR features.

Table 8.10: Evaluation Results Using Different Distance Metrics for original images

	Metric	Full Dataset (Unbalanced)	40 Subset (Balanced)	Curated Full Dataset (Unbalanced)	Curated 40 Subset (Balanced)
Original Images	Cosine Distance	62.76	72.5	58.15	77..50
	Euclidean Distance	55.28	76.67	53.66	77.50

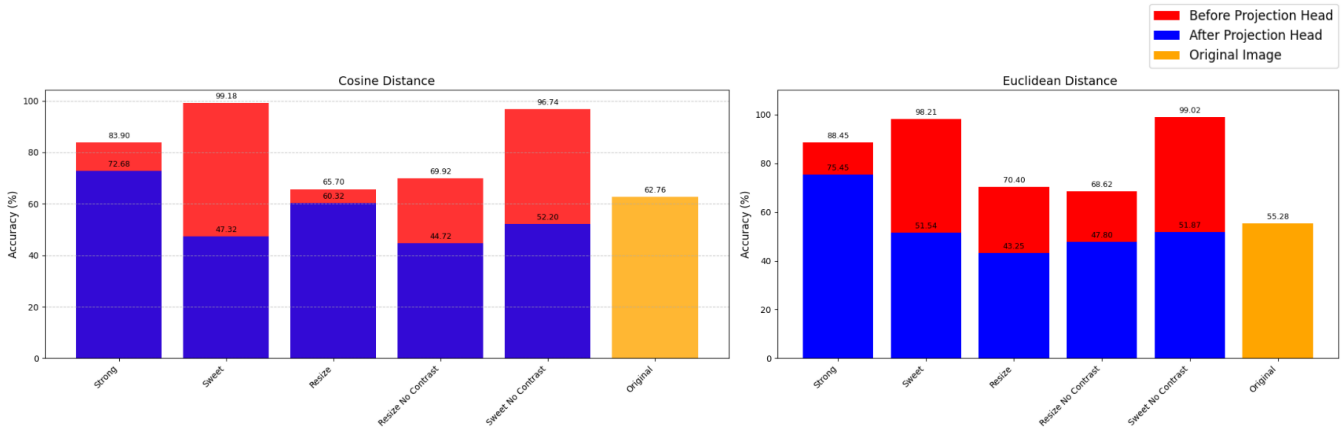

Figure 8.4: Comparison of clustering performance between SimCLR features and original image features

8.6 Inference for cosine distance based K-means:

Since before projection head SimCLR features have far better performance, I do inference on them. I choose 22 images that have close visual resemblance to single dose images from drug screen for inference referred as 'inference image' just like we did in classification inference 8.1.2. (I repeat these are not biology expert labeled drug screen image. so any inference result is irrelevant in the sense to actual ground truth). We used all inference images for full dataset

along with the 3 classes refer as Full dataset (imbalanced). But for balanced datasets except control and explode we changed the dataset setup as follows: for keeping the dataset balanced: 20 single dose and 20 inference images. so that it will still be 40 as other classes.

So during the 3 class (for the uncurated imbalance full dataset set up explained in table 8.7) clustering 'sweet' (99.18 percent accuracy) performed better than sweet no contrast (98.21 percent accuracy) for all full dataset imbalanced. and in other dataset types, 'sweet' achieved 100 percentage accuracy which makes clear that for 3 class problem sweet have higher performance comparing to other data augmentations. but during inference from the table 8.11 its clear that sweet accuracy reduced to 95.76 with -3.42 change in reduction while sweet no contrast reduced to 97.33 with -0.88 reduction change. and sweet no contrast kept the highest accuracy during inference. figure 8.5 shows that with 'sweet no contrast' data augmentation (in left side confusion matrix) it classified all 103 single dose images correctly and during inference figure 8.5 (right side confusion matrix) it clustered all inference images in the cluster of single dose images.

Table 8.11: Inference evaluation results on cosine distance based K-means

Type	Augmentation	Full Dataset (Imbalanced)	Uncured subset balanced	Curated subset balanced
	Strong	85.09 (+1.19)	100 (0)	100 (0)
	Sweet	95.76 (-3.42)	100 (0)	100 (0)
Before Projection Head	Resize	67.03 (+1.33)	97.50 (0)	100 (0)
	Resize No Contrast	71.89 (+1.97)	90.83 (+0.83)	100 (0)
	Sweet No Contrast	**97.33 (-0.88)**	**98.33 (-0.84)**	**99.17 (0)**

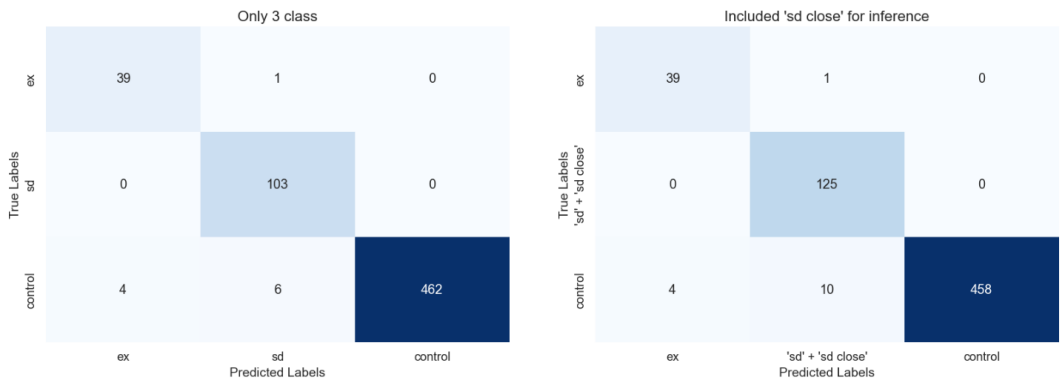

Figure 8.5: Confusion matrix for K-means clustering based on cosine distance for both 3 class clustering (left side) and inference (right side) for uncurated imbalance full data set.

8.7 Inference for euclidean distance based K-means:

For euclidean distance, during 3 class (for the uncurated imbalance full dataset set up explained in table 8.7) clustering 'sweet no contrast' have the highest performance with 99.02 accuracy without any misclassification for single dose images as we seen in figure 8.6 (left side confusion matrix). and during inference it clustered all inference images and single dose images together in the cluster of single dose images correctly as we seen in figure 8.6 (right side confusion matrix). And from the table 8.12 its also shows that sweet no contrast have consistent performance across all dataset types.

From this we can conclude that 'sweet no contrast' data augmentation before projection head is the best data augmentation for clustering task for both cosine distance and euclidean distance as distance metric. Also it able to cluster the inference images in the cluster of single dose images. which basically means that sweet cropping of 0.35 to 1 area of original image (less intense cropping is essential for clustering performance.)

Table 8.12: Inference evaluation results on euclidean distance based on K-means

Type	Augmentation	Full Dataset (Imbalanced)	Uncured subset balanced	Curated subset balanced
	Strong	84.61 (-3.84)	100 (0)	100 (0)
	Sweet	98.27 (+0.06)	91.67(0)	95.83 (0)
Before Projection Head	Resize	72.06 (+1.66)	79.17 (+0.84)	86.67(-1.66)
	Resize No Contrast	71.11 (+2.49)	100 (+10)	100 (0)
	Sweet No Contrast	**99.06 (+.04)**	**97.50 (0)**	**99.17 (0)**
After Projection Head	Strong	75.51 (+0.06)	92.5 (-7.5)	95 (-5)

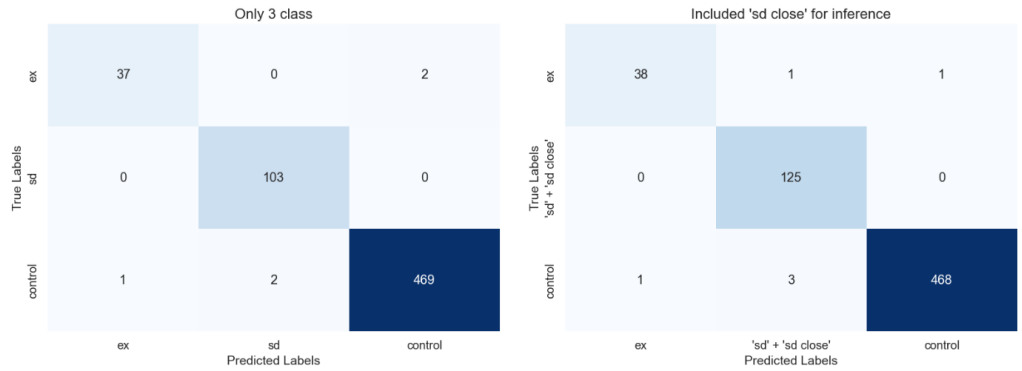

Figure 8.6: Confusion matrix for K-means clustering based on euclidean distance for both 3 class clustering (left side) and inference (right side)

9 Methodology for Ranking

Since we don't have ground truth labels to rank the images except control images (since control images doesn't applied by drug, drug efficacy is basically zero). My strategy was to simplify the current ranking problem to only scale/order/rank images relative to only control, single dose and explode images. What I mean by ordering relative to these specified classes means for example, while ordering drug screen images it shows maybe first control look alike drug screen images then single dose look alike drug screen images then comes explode images. The reason to pick these specific classes for relative positioning is that controls: we know that there is no drug applied which means that there is no effect of drug at all. single dose images are the category which we apply specific drug concentration and its clinically recommended at the moment as the most effective generally (even though we don't know how much drug effected or how much it killed the tumor tissue model) and explode are visually see debris around the tumor tissue model which may potentially harm the surrounding good tissues. once if we can order those control, single dose and explode image using ranking strategy, we can add the inference images to see where they positioned in the scale of control or single dose or explode images. Finally if we can order control , single dose and exploded images in an scale using any of the ranking methodologies thats gonna explain below, we use the same methodology to order all of the drug screening images, we expect it will follow an order of images where drug screening images which are similar to control comes first in the scale then single dose and then exploded images comes next in the scale.

Ranking customized accuracy calculation

Accuracy formulation: Goal is to check whether we have clear separation of each class by the inference metric as explained in literature review **??** while using as it as a scale order. Accuracy below formulated to check whether the first and third classes are well-separated from the middle class. there by it also penalizes if first and third classes are overladed each other. If its clearly separated from the middle class means all 3 classes are well separated and accuracy will be

100.

Detailed explanation of the accuracy calculation:

Once we obtain the inference metric for each individual feature vector in each class, the next steps involve calculating the mean inference metric for each class, identifying the middle class based on the mean values of inference metric, and proceeding with the remaining computations as described below.

Task 1: Mean of Each Class

The mean inference metric for each class C_i is calculated as:

$$\mu_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} M(\mathbf{x}, \mathbf{c})$$

where $|C_i|$ represents the number of points in class C_i, and $M(\mathbf{x}, \mathbf{c})$ is the inference metric between a point $\mathbf{x}$ and the centroid $\mathbf{c}$.

Task 2: Middle Class Based on Mean

To determine the middle class, sort the mean values μ_1, μ_2, μ_3 in ascending order:

$$\mu_{\text{sorted}} = \{\mu_{\text{min}}, \mu_{\text{mid}}, \mu_{\text{max}}\}$$

The middle class is the class corresponding to μ_{mid}.

Task 3: Minimum and Maximum of Middle Class

For the middle class (denoted as C_{mid}), compute the minimum and maximum inference metric values:

$$\text{middle\_class\_min} = \min_{\mathbf{x} \in C_{\text{mid}}} M(\mathbf{x}, \mathbf{c})$$

$$\text{middle\_class\_max} = \max_{\mathbf{x} \in C_{\text{mid}}} M(\mathbf{x}, \mathbf{c})$$

Task 4: Error and Accuracy Calculation

1. **Error Calculation**: Count the number of points in the first class C_1 that exceed the middle class's minimum, and the number of points in the third class C_3 that are less than the middle class's maximum. The total error is given by:

$$\text{error}_1 = |\{\mathbf{x} \in C_1 : M(\mathbf{x}, \mathbf{c}) > \text{middle\_class\_min}\}|$$

$$\text{error}_3 = |\{\mathbf{x} \in C_3 : M(\mathbf{x}, \mathbf{c}) < \text{middle\_class\_max}\}|$$

$$\text{total\_errors} = \text{error}_1 + \text{error}_3$$

2. **Accuracy Calculation**: Subtract the total errors from the total number of points across all classes to compute the number of non-errors. Divide this by the total number of points to calculate the accuracy:

$$\text{accuracy} = \frac{\text{total\_points} - \text{total\_errors}}{\text{total\_points}}$$

where:

$$\text{total\_points} = |C_1| + |C_2| + |C_3|$$

Summary of Steps 1. Compute μ_1, μ_2, μ_3. 2. Identify the middle class using μ_{mid}. 3. Compute middle_class_min and middle_class_max. 4. Count the errors error_1 and error_3. 5. Calculate accuracy using the formula above.

9.1 Day 7 to day 10 prediction using Convolutional Autoencoder

1. **Step 1:** Create a latent space representation of all images, including control, single dose, and drug screen images, using SimCLR. The idea is that SimCLR effectively learns efficient features of similar images that are not captured by human-interpretable metrics. We expect the SimCLR feature vectors of similar images will be closer in the latent space.

2. **Step 2:** Train a prediction model exclusively on the representations of control images from Day 7 to Day 10 using convolutional autoencoder. (Input: Day 7 SimCLR feature vector and target: Day 10 SimCLR feature vector)

3. **Step 3:** Perform inference on the representations of test images, which include control, single dose, and drug screen images. Since the day 10 prediction model is trained solely on the representations of control images, the inference loss/metric (i.e., the difference between the predicted and actual Day 10 image representations) will be very small for control images. Conversely, the inference loss/metric will increase for treated images as their representations deviate from those of control images. This inference loss/metric will be used as the feature for the ranking/order scale, where the initial images will start with control images that have very small inference loss/metric, and the scale will end with images having high inference loss/metric in ascending order.

4. **Step 4:** so in the above methods we train first solely on control images then we did inference on all images just like classical anomaly detection approach. with that same idea/concept, but now we considered other classes as normal and will try to find the deviation from that transition. that is for instance we train solely on day 7 untreated to day 10 single dose images then we do the inference on all images so that inference loss will be how much it deviated from the single dose images. repeating the same concept, we train solely on day 7 untreated to day 10 explode images then we do the inference on all images so that inference loss will be how much it deviated from the explode.

5. Perform the above steps on original image features instead of SimCLR feature vectors for comparative study.

Lack of dataset problem for day 7 to day 10 images: Due to environmental factors sometimes we may get images of day 7 but we won't able to acquire day 10 images or sometimes we may have day 10 images but not corresponding day 7 images. Hence for the images for prediction from day 7 to day 10, we only have 130 control images of pair transitions. 29 images of pair for single dose and 40 image pairs for explode.

CAE Prediction model for original images

Data preparation: I decided to train with day 7 control to day 10 control images.Normalized the images by dividing them by 65535. reduced the size to 96*96. Added data augmentations such as horizontal/vertical flips and rotations and random brightness/sharpness/blur as explained in the sweet augmentation section except contrast and cropping. We can't change contrast as I explained in the augmentation chapter,and also i didn't use cropping because of the idea that in order to learn whole change or transformation to day 10, it needs to see the whole day 7 image which is already explained in the data augmentation resize chapter 7.2. I made sure that:

1. Augmentation to be coupled, meaning the same data augmentation type parameter value is used for both day 7 and corresponding day 10 images.

2. Flips or rotation will be unique to each image with random brightness/sharpness/blur change. This means that in the dataset, the original image can have different augmentations with the same parameter value of brightness/sharpness/blur since it's random, but it won't have the same geometrical transformation.

The augmentation helps to increase the possible sample distribution that could happen in real life. By that we are trying to find a solution for data deficiency and also helps to prevent overfilling. also it helps to be invariant of the brightness/position/blur/sharp changes that can happen due to the microscope mishandling. So now final dataset consist of all class having 650 pair of transitions.

Training parameters and archiecture:

Convolutional Autoencoder Architecture: We developed an autoencoder architecture with an

encoder-decoder structure, utilizing convolutional layers, batch normalization, dropout, and activation functions to learn feature representations and predict day 10 image from day 7. The number of feature maps is progressively increased and then symmetrically decreased to balance feature extraction and construction of day 10 same size image as day 7. **Encoder** The encoder consists of three convolutional layers with a kernel size of 3×3 and `same` padding to maintain spatial resolution.

- Input channels begin with 3, progressively increasing to 16, 32, and 64 to extract deeper hierarchical features.
- Each convolutional layer is followed by `BatchNorm2d` for stable learning, `ReLU` activation for non-linearity, and dropout with probabilities of 0.2, 0.3, and 0.4, respectively, to mitigate overfitting.
- Downsampling is performed using `MaxPool2d` with a kernel size of 2×2, stride of 2, and padding to reduce spatial dimensions at each step.

Decoder The decoder mirrors the encoder, reconstructing the input from the learned features.

- It begins with 64 channels and symmetrically decreases the number of channels to 32, 16, and finally 3 to match the input dimensions.
- Each convolutional layer retains a kernel size of 3×3 with `same` padding, followed by `BatchNorm2d`, `ReLU` activation, and dropout with probabilities of 0.3 and 0.2, respectively.
- Upsampling is performed using `Upsample(scale_factor=2, mode='nearest')` to restore spatial dimensions step by step.
- The final layer includes a `Sigmoid` activation function to ensure output pixel values are in the range $[0, 1]$, making the model suitable for day 10 image construction.

The overall structure employs progressive channel expansion in the encoder ($3 \rightarrow 16 \rightarrow 32 \rightarrow 64$) to capture detailed feature hierarchies and symmetric channel reduction in the decoder ($64 \rightarrow 32 \rightarrow 16 \rightarrow 3$) for day 10 image construction. training parameters: bacth size = 32.

learning rate = 0.001. optimizer = adam. loss = mse. epochs = 500.

Result: Unfortunately, the model didn't learn the features of transition from day 7 to day 10 well. the loss didn't decrease in the 500 epochs as we see in the figure 9.1. Figure reflects that model struggles to predict the day 10 images from day 7 image, the predicted image is more resemblance visually to day 7 than day 10. For the model its hard to learn, probably because in most cases as we see in figure 9.3 the day 10 image tumor tissue model is totally changed in shape or size or pixel intensity. Also from the same figure shows, it could be the positional change due to the microscope handling position. (We used different flip and rotation augmentation but it seems like it mayn't be effective as we think it is.) Since the same problem we may have with the day 7 to day 10 single dose or explode images, I decided to move to SimCLR features to see if it can learn the transition better than the original images.

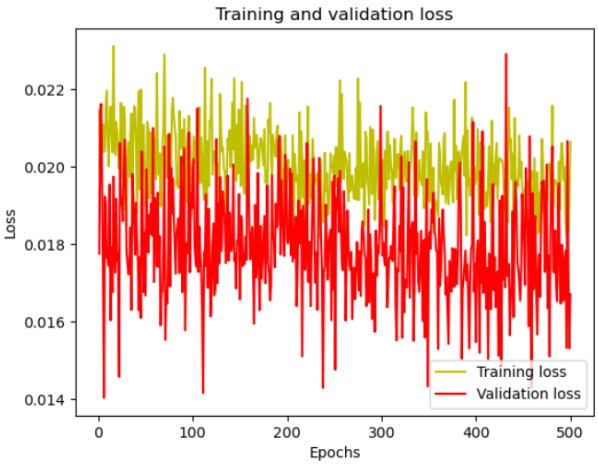

Figure 9.1: Train vs test loss for original image vectors using CAE prediction model from day 7 to day 10

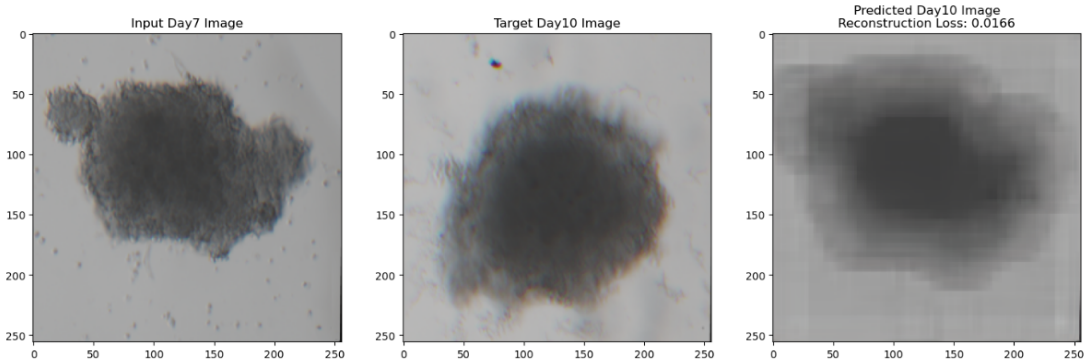

Figure 9.2: Train vs test loss on original image vectors using CAE prediction model from day 7 to day 10

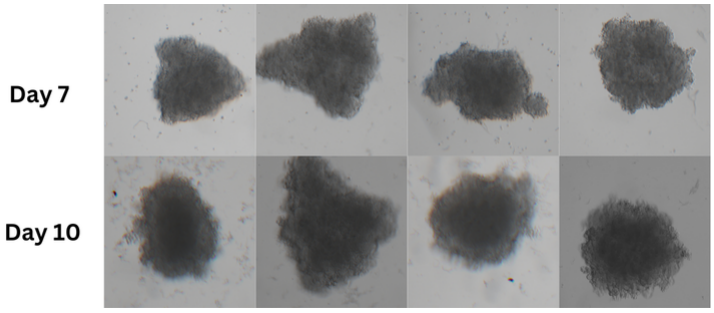

Figure 9.3: Each pair of day 7 to day 10 transition shows changes in size/shape etc. last pair of transition shows mis position of the cancer cell. In the day 7 it was above the middle line of the image. and in day 10 its below of the middle line of the image

Simclr features

Set up of data processing same as above original setup. first I started with the pair of day 7 untreated to predict the day 10 control images setup. Normalized images by dividing them by 65535. resized to 96*96. then I increased the dataset using same data augmentation technics that I did to original image setup as explained in the above section but with made sure that flips or rotation will be unique to each image with random brightness/sharpness/blur change within a range as above. then I feed to simclr model to get features after and before projection head to train the CAE model.

Training setup: First to ensure the simclr features are scaled I did featurewise minmax scaling. Then I feed those into FeaturePredictor model which is a fully connected neural network. It consists of a series of 12 linear layers, where the input size reduces from 512 to 8 and then symmetrically increases back to 512 to match the original feature dimensions for before projection head features. For after projection head features,consists of a series of 6 linear layers, where the input size reduces from 20 to 4 and then symmetrically increases back to 20.

Each intermediate layer is followed by:

Batch normalization (BatchNorm1d): Normalizes activations to stabilize training and accelerate convergence. ReLU activation: Introduces non-linearity to model complex feature relationships. Dropout (Dropout): Applied with probabilities ranging from 0.2 to 0.4 to prevent overfitting, with deeper layers using higher dropout rates for regularization. The final layer is a linear transformation to ensure the output feature size matches the day 10 feature vector

dimensions which is 512 for before and 20 for after.

batch size = 32. loss = mse loss. optimizer = adam optimizer. learning rate = 0.0001. no of epochs = 2000. used cross validation with 3 fold and early stop with the parameters patience = 500 and delta = 0.0001. training data and validation data divided as 80: 20 ratio.

As we see in the figure below 9.4, for the training of day 7 to day 10 control images and single dose, the train loss and validation loss decreased gradually in the 2000 epochs. But for the category of considering single dose as normal ie training day 7 to day 10 single dose images we can see overfitting behavior in the loss. This is probably due to the fact that for training day 7 to day 10 single dose images we have only 29 images as original. This is the problem of data deficiency. We have tried to mitigate this problem by increasing the dataset size by data augmentation but nevertheless for the explode images training didn't quite worked out.

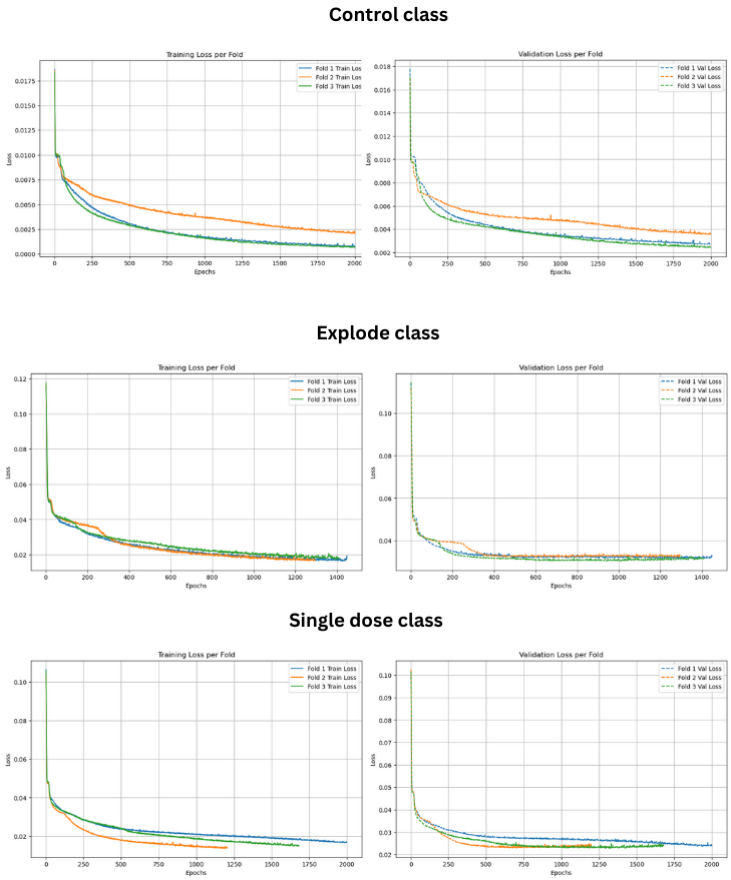

Figure 9.4: Training loss Vs test loss for each class as normal (ie the class is only used in training). Single dose dataset shows overfitting

We calculated inference loss/metric for both before and after projection head feats and used

following metrics as inference loss between target feature vector and predicted feature vector: mse, cosine distance, L2 (euclidean distance), L1 distance, Pearson correlation, dot product,jaccard similarity, hamming distance.

The reason to choose these metrics are the metrics mainly used in medical images as explained in literature review.

Results: As we can see the figure 9.5. Some of the inference metric are able to make a clear separation of the training class to others ie talent one class is have clear separation from other classes as we seen in figure. and some of the metric couldn't make a clear separation for atlas one class ie all of the classes are mixed as we seen in the figure for dot product and jaccard similarity. We observed this same trend for both before and after projection head features and for all transition category. Ideal situation that we would like to have is all of the classes are separated from each other. so that when we do inference on drug screen images there will be look alike smooth transition from class to other.

Projection Head	Normal as	Strong	Sweet	Resize	No Contrast Resize	No Contrast Sweet
Before	Single Dose	65.93	67.22	66.55	66.39	66.60
	Control	50.41	68.10	53.15	66.29	**77.51**
	Explod	65.93	75.59	67.58	67.48	68.61
After	Single Dose	62.15	65.41	61.38	51.81	60.39
	Control	35.42	38.16	33.82	35.26	39.66
	Explod	48.40	64.32	61.01	36.40	51.96

Table 9.1: Cosine distance as inference metric

Projection Head	Normal as	Strong	Sweet	Resize	No Contrast Resize	No Contrast Sweet
Before	Single Dose	66.24	66.60	66.39	66.13	66.60
	Control	52.95	67.11	55.79	66.03	**77.61**
	Explod	70.84	74.35	64.74	69.70	71.98
After	Single Dose	64.12	68.87	64.01	47.57	55.12
	Control	35.68	36.61	33.66	35.32	36.71
	Explod	51.50	56.41	37.80	44.98	59.98

Table 9.2: Euclidean distance as inference metric

Projection Head	Normal as	Strong	Sweet	Resize	No Contrast Resize	No Contrast Sweet
Before	Single Dose	66.24	66.60	66.39	66.13	66.60
	Control	52.95	67.11	55.79	66.03	**77.61**
	Explod	70.84	74.35	64.74	69.70	71.98
After	Single Dose	64.12	68.87	64.01	47.57	55.12
	Control	35.68	36.61	43.80	35.32	36.71
	Explod	51.50	56.41	61.48	44.98	59.98

Table 9.3: MSE distance as inference metric

Projection Head	Normal as	Strong	Sweet	Resize	No Contrast Resize	No Contrast Sweet
Before	Single Dose	65.01	66.60	66.39	65.05	66.60
	Control	53.83	66.75	51.34	66.03	72.29
	Explod	65.15	68.20	62.56	64.79	**76.53**
After	Single Dose	62.36	66.13	57.91	42.24	51.55
	Control	35.52	37.44	33.61	34.64	36.81
	Explod	48.71	52.12	59.10	41.68	56.10

Table 9.4: L1 distance as inference metric

Projection Head	Normal as	Strong	Sweet	Resize	No Contrast Resize	No Contrast Sweet
Before	Single Dose	66.34	66.39	66.55	66.13	66.44
	Control	50.26	68.51	60.34	66.24	**75.65**
	Explod	64.68	67.01	65.67	65.51	67.01
After	Single Dose	48.50	64.68	48.24	50.52	62.15
	Control	47.78	43.02	44.26	34.49	51.14
	Explod	45.04	67.11	59.26	47.21	53.15

Table 9.5: Pearson correlation as inference metric

Projection Head	Normal as	Strong	Sweet	Resize	No Contrast Resize	No Contrast Sweet
Before	Single Dose	33.61	34.13	33.40	34.23	**53.93**
	Control	39.97	49.28	34.90	40.12	47.10
	Explode	37.75	35.32	34.85	44.93	32.99
After	Single Dose	36.81	46.38	47.00	39.71	34.75
	Control	33.30	38.99	34.23	34.28	33.66
	Explode	34.18	45.45	33.87	36.19	33.61

Table 9.6: Dot product as inference metric

Projection Head	Normal as	Strong	Sweet	Resize	No Contrast Resize	No Contrast Sweet
Before	Single Dose	55.69	66.65	48.55	62.72	66.44
	Control	49.95	68.46	61.89	68.67	**69.29**
	Explod	54.34	66.18	53.83	67.32	66.91
After	Single Dose	41.78	51.50	34.33	37.59	50.47
	Control	42.14	37.18	34.33	33.09	36.30
	Explod	39.14	50.78	42.55	35.47	41.47

Table 9.7: Jaccard similarity as inference metric

Projection Head	Normal as	Strong	Sweet	Resize	No Contrast Resize	No Contrast Sweet
Before	Single Dose	61.84	66.60	62.67	59.20	66.44
	Control	34.80	60.24	48.60	64.84	63.70
	Explod	62.82	72.60	56.36	64.06	**67.22**
After	Single Dose	41.73	46.74	35.94	48.55	56.77
	Control	45.76	42.97	34.33	33.92	36.76
	Explod	45.24	55.17	43.07	43.07	45.04

Table 9.8: Hamming distance as inference metric

From the above tables, 9.8, 9.7 , 9.6 , 9.5 , 9.4 , 9.2 , 9.3 and 9.1, we can see that cosine distance, euclidean distance, mse, L1 distance and Pearson correlation coefficient are able to separate the classes moderately comparing to dot product as inference metric is separating the classes least among all metrics which make it as the worst metric for ranking.

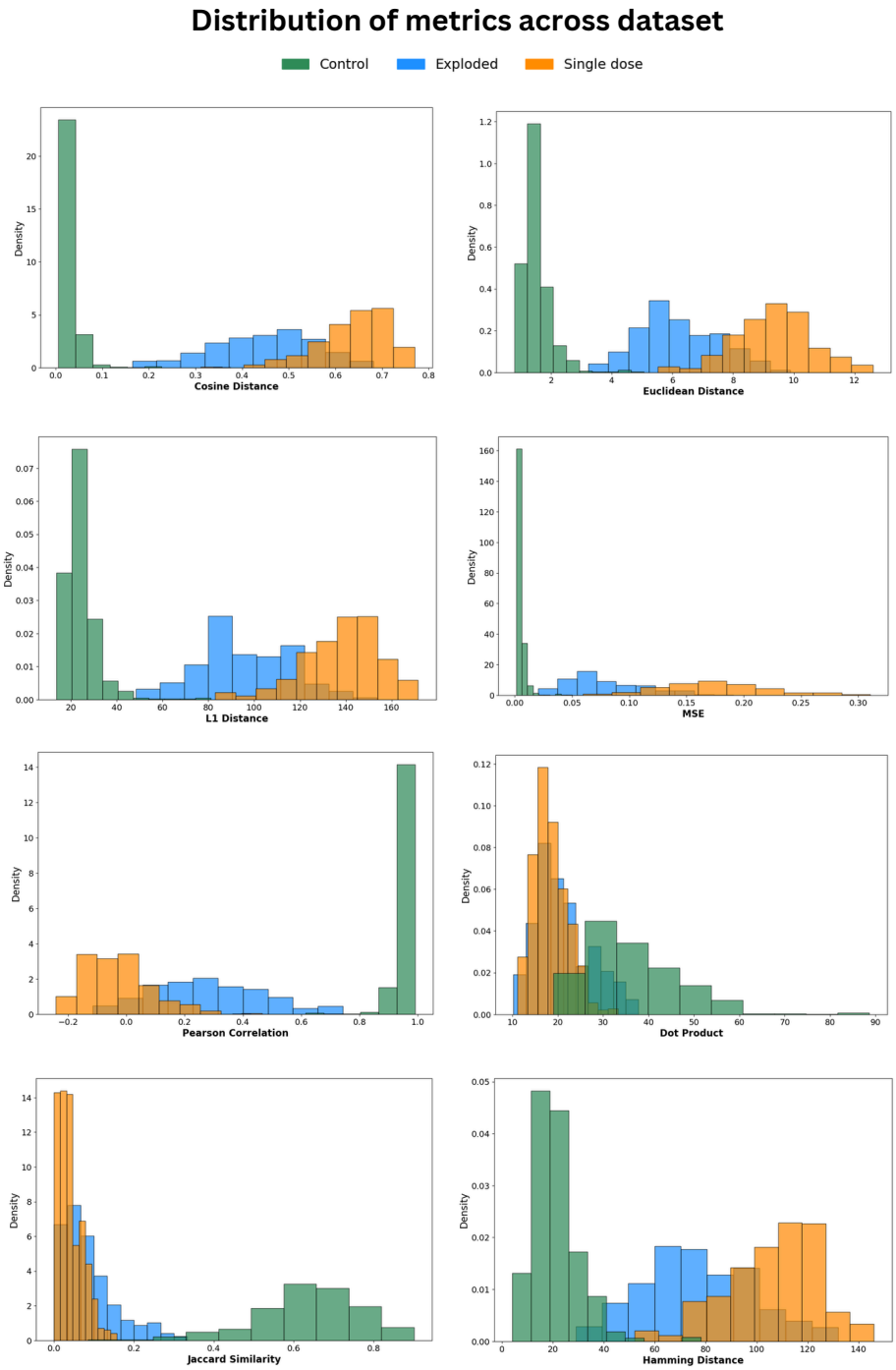

Figure 9.5: Ranking scale for before projection head SimCLR features while control class used as normal ie control class only used for training

Since we couldn't separate those 3 classes well that means we won't have good ranking of drug screen images since sd close images and explode images will be mixed while ranking. this calls the need for other ranking strategy such as softmax approach, K means centroid approach, dimensionality reduction technics. where for K means centroid , dimensionality reduction technics, small no of data will not effect that much for the performance of ranking. since there

is no training required.

9.2 Ranking strategy 2: Softmax approach

This strategy utilizes the before projection head SimCLR vectors since its linearly separable for downstream task classification as suggested in original SimCLR paper [2].

We are using only before projection head SimCLR features since original images or after projection head SimCLR features are not able to classify 100 percentage as we seen in intermediate evaluation chapter 8

1. Train a classification model to classify control and single dose images.

2. Since we attained 100% accuracy for the train and test datasets for this binary classification, if we look at the softmax score of all the single dose feature vectors to classify as the single dose class, the values will be equal to or really close to 1. Similarly, if we look at the softmax score of all the control feature vectors to classify as the single dose class, the values will be equal to or really close to 0.

3. Now, when we do inference with these learned weights and parameters on the same control and single dose features, we will get the softmax value of all the single dose feature vectors to classify as the single dose class. The values will be equal to or really close to 1. Similarly, if we look at the softmax score of all the control feature vectors to classify as the single dose class, the values will be equal to or really close to 0.

4. Now, we add an explode class in addition to the control and single dose classes in inference. The idea is that since we didn't use the explode class in training, the softmax score of all the explode feature vectors to classify as the single dose class will not be exactly or closer to 1. It can be below the values of single dose class features, which means it can overlap with the values of control features. But what if those explode feature softmax values did not overlap with the values of control? If we use the softmax score to predict the single dose class as an inference metric for ranking, in the range of our ranking scale, control class features come first as the values are closer or equal to 0, then explode class

features come next in the scale in between 0 and 1, and single dose class features come last as the values are closer to 1. This is the ideal situation that we would like to have. If we get this kind of ranking scale, we can say that our softmax score is able to separate control, explode, and single dose classes, and we can use the same customized ranking accuracy calculation as we used in the previous ranking strategies.

5. Use these same steps to classify between control and explode, so that we expect those class softmax values will be at both ends (0 and 1), and the single dose class softmax values will be in between control and explode class values. Likewise, classification between explode and single dose, where control feature softmax values will be in between explode and single dose class softmax values.

Initialy I used original dataset of control (472 images) and single dose images (103 images) as explained in the dataset table and we used explode (40 images) I train with different parameters for number of epochs and batch size. following parameters gave higher accuracy: 750 epochs, batch:16. Every other parameter is same as explained in the classification section in the chapter 8.

Classification	**Strong**	**Sweet**	**Resize**	**No Contrast Resize**	**No Contrast Sweet**
Control vs SD	25.04	49.75	76.91	85.85	90
Control vs Ex	83.41	49.27	95.93	60.49	71.54
Ex vs SD	97.07	96.75	93.5	87.64	87.31

Table 9.9: Ranking accuracy on inference using all the dataset we have for each class using SimCLR features and

Classification	**Control vs SD**	**Control vs Ex**	**Ex vs SD**
Accuracy (%)	45.53	83.41	21.62

Table 9.10: Ranking accuracy on inference using all the dataset we have for each class using original image features

As we seen in the above tables 9.9 and 9.10, none of the data augmentation types are not able to give clear separation between 3 classes. No contrast sweet, resize and sweet data augmentation data types achieved highest accuracy 90 and 95.12 and 97.07 in the category of Control vs SD

and Control vs Ex and Ex vs Sd categories correspondingly. Before head projection SimCLR features still performed comparatively well comparing to original image vectors.

The main limitation of this approach is we can't guarantee when we get new images it will sail separate three classes 100 percentage like we seen above, there can be overlaps, since the classification only promise us that the softmax score of the control will be closer or equal to zero and single dose will be closer or equal to 1 in control vs sd case. that means classification only promise us that we get clear separation between 2 classes not three classes based on the softmax value.

9.3 Ranking strategy 3: K Means approach

Unlike other 2 strategies this strategy doesn't effect the class imbalance problem since there is no specific training required.

1. **Step 1:** Feed control images SimCLR features into K-means and find the centroid of control cluster based on both cosine distance and the euclidean distance using k-means clustering.

2. **Step 2:** Calculate the euclidean/cosine distance from this centroid to every simclr features.

3. **Step 3:** Rank the images based on the distance from the centroid of control simclr features.

4. **Step 4:** Perform the above ranking procedure from centroid of single dose image simclr features to other image features and from centroid of explode image simclr features to other image features correspondingly.

5. **Step 5:** Perform the same operation on original images.

Dataset we used : We used all of the control images (472), single dose image (103) and exploded images (40).

Projection Head	Distance From	Strong	Sweet	Resize	No Contrast Resize	No Contrast Sweet
Before	Single Dose Mean	93.33	93.50	48.23	92.52	93.82
	Control Mean	78.86	87.64	91.49	85.53	92.20
	Explode Mean	83.25	25.53	82.74	29.43	82.76
After	Single Dose Mean	**95.12**	85.69	28.29	77.72	83.74
	Control Mean	71.54	76.75	82.76	14.47	20.16
	Explode Mean	85.37	80.00	37.72	78.21	80.81

Table 9.11: Performance of K-means centroid approach based on cosine distance as inference metric using SimCLR features

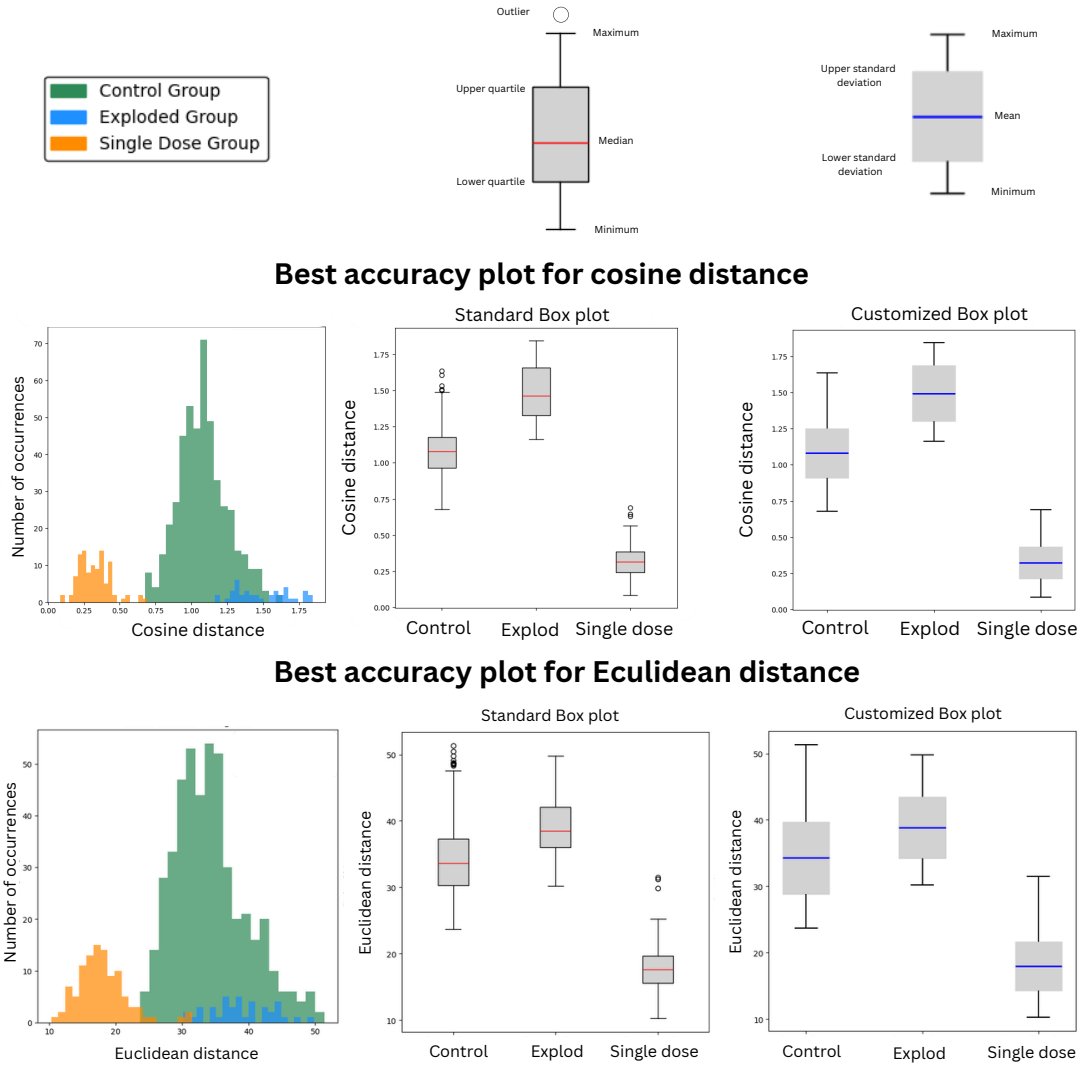

Figure 9.6: Selected inference metric scales for ranking accuracy using SimCLR features. Strong after projection head feature, using single dose as centroid , got best for ranking accuracy while cosine distance as inference metric showed in the first plot and Strong before projection head feature, using single dose as centroid , got best for ranking accuracy while euclidean distance as inference metric as shown in the second plot

Projection Head	Distance From	Strong	Sweet	Resize	No Contrast Resize	No Contrast Sweet
Before	Single Dose Mean	**92.52**	90.41	31.71	83.58	91.38
	Control Mean	69.11	37.24	76.75	39.84	57.89
	Explode Mean	82.44	82.60	67.97	23.90	82.28
After	Single Dose Mean	90.57	80.33	36.42	76.75	21.63
	Control Mean	55.70	76.75	76.75	76.75	77.07
	Explode Mean	83.09	77.40	55.93	19.84	81.63

Table 9.12: Performance of K-means centroid approach based on euclidean distance as inference metric using SimCLR features

Distance From	Single Dose Mean	Control Mean	Explod Mean
Cosine	17.56	46.50	28.45
Euclidean	24.227	50.08	34.47

Table 9.13: Performance of K-means centroid approach on original images

Insights from the above tables 9.6, **??** and **??** and figure 9.6: Cosine distance of each after projection head strong simclr feature from the single dose mean have the most separation of classes with accuracy of 95.12. It doesn't separate three classes fully but its something to hope for. With the euclidean distance, before projection head strong simclr feature from the single dose mean have the most separation of classes with accuracy of 92.52. It clearly separates one class from other but still far away from the result of 3 class separation that we hoped for. All of the above inference metric scaling I did it on original image vector and as you can see below performance of the ranking accuracy using cosine and euclidean distance were quite low comparing to the simclr feature vectors.Which basically tells us for this ranking strategy also simclr features are better than original image vectors.

9.4 Ranking strategy 4: Dimensionality reduction technics

PCA aka Principal component analysis can be defined as the orthogonal projection of the data onto lower dimensional linear space, known as the principle subspace, such that variance of the

projected data is maximized, equally it can defined as the linear projection that minimizes the average projection cost,defines as the mean squared distance between the datapoint and their projection [1]. PCA is less effective when the intrinsic data structure is highly nonlinear.

t-SNE: In contrast, t-distributed Stochastic Neighbor Embedding (t-SNE) is a nonlinear technique designed specifically for visualizing high-dimensional data by preserving local neighborhood structures. t-SNE works by converting pairwise distances in high-dimensional space into probabilities that represent similarities, then optimizing a low-dimensional embedding to reflect these local relationships accurately. While t-SNE excels at revealing clusters and subtle local structures that PCA might miss, it is computationally intensive and sometimes struggles to preserve the global structure of the data. These characteristics can limit its effectiveness as a ranking tool when global relationships are important, even though it provides superior visualization of complex patterns.

UMAP: Uniform Manifold Approximation and Projection (UMAP) is another nonlinear dimensionality reduction method that builds on principles from manifold learning and topology to capture both local and global data structures. UMAP tends to be faster than t-SNE and often preserves more of the global structure, making it a promising alternative when a balanced view of local detail and overall data distribution is needed. However, like t-SNE, UMAP is sensitive to hyperparameter tuning and the inherent structure of the data, which can influence its performance.

We will use the PCA technique to identify the direction of maximum variance in the high dimensional feature space in our case 512 D for before projection head feature simclr feature vector, and 20 D for after projection head simclr feature vector and 27648 D for original images and projects data onto this single axis (1D). The resulting First PCA scores (positions along this axis) aim to preserve maximum variance as possible from the original data while reducing redundancy and noise. Idea is that whether this 1D PCA value can be used as as our ranking metric or not. Similarly we use other dimensionality reduction techniques such as t-SNE (t-Distributed Stochastic Neighbor Embedding) and UMAP (Uniform Manifold Approximation and Projection) to reduce the higher dimension of data to single dimension to check whether these single dimension data of t-sne or UMAP values are useful for ranking the images.

Metric	DR Technique	Strong	Sweet	Resize	No Contrast Resize	No Contrast Sweet
Before Projection Head	PCA	73.33	51.87	49.59	34.80	77.56
	UMAP	93.33	99.51	99.35	93.50	99.67
	t-SNE	99.84	**100**	**100**	**100**	99.19
After Projection Head	PCA	80.49	21.95	76.75	76.75	31.71
	UMAP	93.50	34.47	56.26	25.53	93.50
	t-SNE	74.63	55.61	98.70	76.75	31.71

Table 9.14: Comparison of dimensionality reduction techniques (PCA, UMAP, t-SNE) on before and after the projection head SimCLR features for ranking strategy.

PCA	t-SNE	UMAP
54.47	54.80	17.56

Table 9.15: Comparison of dimensionality reduction techniques (PCA, UMAP, t-SNE) on original image vectors for ranking strategy.

From the above table 9.14 we see that before projection head SimCLR features out performs generally after projection head SimCLR features and the original image feature vectors in among all dimensionally reduction technic. While comparing among before projection head: It shows that PCA reducing data dimension to 1 D doesn't separate 3 classes well compare to the t-SNE and UMAP. While umap achieved closer to 100 percent ranking accuracy, t-SNE performed better among all dimensionality reduction technic with the 100 percentage ranking accuracy to separate 3 classes in sweet, resize, No contrast data augmentation pipelines. Limitation is that we are not sure that whether these clear separation between classes ensure there is perfect transition of drug efficacy within the class. And in this ranking methodology also SimCLR features outperformed original image vectors.

10 Conclusion

In this chapter, we aim to determine which of the four methodologies is more reliable and consistent in terms of relative positioning. To do this, we will evaluate the best-performing strategy from each of the four methodologies that showed promising results in ranking accuracy, as explained in the Methodology for Ranking chapter (9).

The evaluation will involve checking whether the 22 inference images still fall within the range (minimum and maximum values) of the single dose metric, based on the best-performing data augmentation type/category from each methodology. The goal is to see if these inference images are still positioned within the range of the single dose images.

It is important to note that this final evaluation is not entirely reliable. The inference images have not been verified by a biology expert to confirm that they are the most similar images to the single dose images visually or by drug efficacy from the drug screening. Therefore, while this evaluation may not be scientifically precise, it provides a useful comparison, especially in the absence of expert validation.

Ranking Strategy	**Inference Images Outside Single Dose Range**	**Ranking Accuracy**
Prediction model Euclidean distance Before	0	76.93
tsne: before sweet	0	100
tsne: resize before	0	100
tsne: before no contrast resize	0	100
Softmax: cond7 vs Ex Resize	1	97.33
Softmax: Ex vs sd Strong	7	92.46
K-means: After cosine strong using single dose mean	12	93.17

Table 10.1: Inference ranking performance across best-performing data augmentation type/category from each methodology. Ranking accuracy refers to the inclusion of inference images in the ranking of images, along with control, single dose, and explod. ”Inference Images Outside Single Dose Range” indicates the number of inference images positioned outside the minimum and maximum range of the single dose images.

From the table 10.1, we observe that some of the inference images are outside the single dose image ranking metric range for the Softmax and K-means strategies. This makes these strate-

gies less reliable compared to the others. In fact, in the K-means approach, the 12 inference images that fall outside the single dose image range are positioned among the control class, making the K-means strategy the most unreliable. Interestingly, the Prediction model and t-SNE strategies have zero inference images that fall outside the single dose range. While the Prediction model still struggles to clearly separate the three classes (with accuracy of 76.93) (likely due to the limited data available for predicting transitions from day 7 to day 10, with only 29, 40, and 130 pairs for the single dose, explode, and control classes, respectively), the t-SNE strategy maintained a clear separation between the three classes, achieving 100 percent ranking accuracy. However, a limitation remains in t-SNE: we are still unsure whether this clear separation between the classes ensures perfect drug efficacy transition within and between the classes, as we expect.

One consistent finding across all results, including ranking strategy and intermediate evaluation methods, is that SimCLR features before the projection head generally outperform those after the projection head and the original image features, except in the K-means: cosine strong strategy, where the features after the projection head performed better. However, this strategy can be disregarded since it turned out to be unreliable.

While using before projection head features, between different data augmentation pipeline methods such as strong, sweet, resize, no contrast sweet, no contrast resize. which one performs better? If we look at different ranking strategies and intermediate evaluation, each of them performed best in different ranking and intermediate evaluation strategies. point is, there is no consistency in terms of one of the data augmentation pipelines showed best performant overall. In another words, they actually seem to perform randomly and which raises the question whether we can rely on them. This may be due to not enough data. If we have more data maybe the same methodology reveals which one better. At the moment they give some results but I'm not actually sure, whether its useful for the final objective or ranking. because even though simclr before projection head features with different data aug pipelines gives clear separation between classes with 100 percentage ranking accuracy using t-SNE, that doesn't mean there is ideal transition of drug efficacy within the class or between classes we are looking for our objective. Hence I'm not convinced the final evaluation that we get in the above table for relative positioning of drug screen images to three classes make sense at the end.

One key takeaway is that, for intermediate evaluations, clustering using SimCLR feature vectors before the projection head performed very well, achieving around 99 percent accuracy. This is an interesting result because, using completely unsupervised learning, we were able to cluster the three classes—control (untreated), single dose, and explode images—without relying on any manually handcrafted feature extraction methods, such as Size/Area, Circularity/Diameter/Perimeter, or Pixel intensity changes observed in the bright-field microscopy images from day 7 to day 10. In the classification task, using SimCLR before the projection head achieved around 99 percent accuracy, which is comparable to the performance of supervised ResNet18, which also achieved around 99 percent classification accuracy, as seen in the classification chapter.

Perhaps, the ranking approach is not the best formulation of the problem. Instead of pursuing this method, a better approach might be to focus on which group or cluster of specific drug screening concentrations is closer to the cluster of single dose images. Despite my best efforts, the current methods do not seem to yield useful results.

10.1 Future research direction

1. Hybrid: Combine human-interpretable features with unsupervised learning features for a combined effect.

2. Weakly supervised DINO transformer-based and SimCLR approach.

3. Experiment with different image sizes for SimCLR: 96 vs. 256 vs. 512 vs 2054. since more the image size more information it contain.

4. Experimenting different Batch size: 16 vs. 64 vs. 128 vs. 256 with SimCLR. Why explore other models? SimCLR requires a larger batch size and more data for optimal performance, which we currently lack.

Appendix

Derivation of K-Means Clustering using Euclidean Distance and Mean

Objective Function The k-means algorithm aims to minimize the total squared Euclidean distance between data points and their assigned cluster centroids. The objective function is:

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \|\mathbf{x}_i - \mu_k\|^2$$

where:

- n: Number of data points,
- K: Number of clusters,
- $\mathbf{x}_i$: The i-th data point,
- μ_k: The centroid of the k-th cluster,
- r_{ik}: Binary indicator; $r_{ik} = 1$ if $\mathbf{x}_i$ belongs to cluster k, otherwise $r_{ik} = 0$.

Cluster Assignment Step

For a fixed set of centroids $\{\mu_k\}_{k=1}^{K}$, assign each data point $\mathbf{x}_i$ to the nearest centroid. This minimizes:

$$r_{ik} = \begin{cases} 1, & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2, \\ 0, & \text{otherwise.} \end{cases}$$

Centroid Update Step

For a fixed cluster assignment $\{r_{ik}\}$, minimize J with respect to the centroids $\{\mu_k\}$:

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \|\mathbf{x}_i - \mu_k\|^2$$

Focus on a single cluster k. The term involving μ_k is:

$$\sum_{i=1}^{n} r_{ik} \|\mathbf{x}_i - \mu_k\|^2 = \sum_{i=1}^{n} r_{ik} \left(\mathbf{x}_i^\top \mathbf{x}_i - 2\mathbf{x}_i^\top \mu_k + \mu_k^\top \mu_k \right)$$

Take the derivative with respect to μ_k and set it to zero:

$$\frac{\partial}{\partial \mu_k} \sum_{i=1}^{n} r_{ik} \|\mathbf{x}_i - \mu_k\|^2 = -2 \sum_{i=1}^{n} r_{ik} \mathbf{x}_i + 2 \sum_{i=1}^{n} r_{ik} \mu_k = 0$$

Simplify:

$$\sum_{i=1}^{n} r_{ik} \mathbf{x}_i = \sum_{i=1}^{n} r_{ik} \mu_k$$

Factor out μ_k:

$$\mu_k = \frac{\sum_{i=1}^{n} r_{ik} \mathbf{x}_i}{\sum_{i=1}^{n} r_{ik}}$$

This is the mean of the points in cluster k.

Algorithm Summary

The k-means algorithm alternates between the following two steps until convergence:

1. **Cluster Assignment Step**: Assign each point $\mathbf{x}_i$ to the nearest cluster:

$$r_{ik} = \begin{cases} 1, & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2, \\ 0, & \text{otherwise.} \end{cases}$$

2. **Centroid Update Step**: Update the centroid of each cluster as the mean of its assigned points:

$$\mu_k = \frac{\sum_{i=1}^{n} r_{ik}\mathbf{x}_i}{\sum_{i=1}^{n} r_{ik}}$$

This iterative process continues until the assignments r_{ik} and centroids μ_k no longer change or the change is below a threshold.

Cosine distance

Normalization

To calculate the cosine distance, we first **normalize** all data points and centroids. Suppose the normalized data points and centroids are $\mathbf{x}_i$ and $\mathbf{c}_k$, respectively, then:

$$\|\mathbf{x}_i\| = 1 \quad \text{and} \quad \|\mathbf{c}_k\| = 1.$$

This ensures all vectors are on the unit sphere. The cosine similarity between two vectors $\mathbf{x}_i$ and $\mathbf{c}_k$ is defined as:

$$\text{cosine similarity} = \frac{\mathbf{x}_i^\top \mathbf{c}_k}{\|\mathbf{x}_i\|\|\mathbf{c}_k\|}$$

Since $\|\mathbf{x}_i\| = 1$ and $\|\mathbf{c}_k\| = 1$, we substitute these values into the equation:

$$\text{cosine similarity} = \frac{\mathbf{x}_i^\top \mathbf{c}_k}{1 \times 1}$$

This simplifies to:

$$\text{cosine similarity} = \mathbf{x}_i^\top \mathbf{c}_k.$$

Thus, the **cosine distance** becomes:

$$\text{cosine distance} = 1 - \mathbf{x}_i^\top \mathbf{c}_k.$$

Relating Cosine Distance to Euclidean Distance

For normalized vectors, we derive the relationship between **Euclidean distance** and **cosine distance**. The squared Euclidean distance between a data point $\mathbf{x}_i$ and a centroid $\mathbf{c}_k$ is:

$$\|\mathbf{x}_i - \mathbf{c}_k\|^2 = \sum_j (x_{ij} - c_{kj})^2.$$

Expanding this:

$$\|\mathbf{x}_i - \mathbf{c}_k\|^2 = \|\mathbf{x}_i\|^2 + \|\mathbf{c}_k\|^2 - 2\mathbf{x}_i^\top \mathbf{c}_k.$$

Since $\|\mathbf{x}_i\| = 1$ and $\|\mathbf{c}_k\| = 1$, we get:

$$\|\mathbf{x}_i - \mathbf{c}_k\|^2 = 1 + 1 - 2\mathbf{x}_i^\top \mathbf{c}_k.$$

Simplify:

$$\|\mathbf{x}_i - \mathbf{c}_k\|^2 = 2(1 - \mathbf{x}_i^\top \mathbf{c}_k).$$

Thus, for normalized vectors, the Euclidean distance is proportional to the cosine distance:

$$\|\mathbf{x}_i - \mathbf{c}_k\|^2 = 2 \cdot \text{cosine distance}.$$

Rearranging to express the cosine distance:

$$\text{cosine distance} = \frac{\|\mathbf{x}_i - \mathbf{c}_k\|^2}{2}.$$

Objective Function

The k-means algorithm with cosine distance aims to minimize the cosine distance between data points $\mathbf{x}_i$ and their assigned cluster centroids $\mathbf{c}_k$. The objective function is:

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \left(1 - \frac{\mathbf{x}_i^\top \mathbf{c}_k}{\|\mathbf{x}_i\| \|\mathbf{c}_k\|}\right),$$

where:

- n: Number of data points,
- K: Number of clusters,
- $\mathbf{x}_i$: i-th data point,
- $\mathbf{c}_k$: Centroid of cluster k (normalized to unit length),
- r_{ik}: Binary indicator; $r_{ik} = 1$ if $\mathbf{x}_i$ belongs to cluster k, otherwise $r_{ik} = 0$.

Objective Function in Terms of Euclidean Distance

Using the above result, the k-means objective function with cosine distance:

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \left(1 - \mathbf{x}_i^\top \mathbf{c}_k\right)$$

can be rewritten in terms of Euclidean distance:

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \frac{\|\mathbf{x}_i - \mathbf{c}_k\|^2}{2}.$$

Here, the factor $\frac{1}{2}$ accounts for the scaling difference.

Cluster Assignment Step

For a fixed set of centroids $\{\mathbf{c}_k\}_{k=1}^K$, assign each data point $\mathbf{x}_i$ to the nearest cluster based on the **cosine similarity** (or equivalently, minimize cosine distance):

$$r_{ik} = \begin{cases} 1, & \text{if } k = \arg\max_j \mathbf{x}_i^\top \mathbf{c}_j, \\ 0, & \text{otherwise.} \end{cases}$$

Centroid Update Step for Cosine Distance

For a fixed cluster assignment $\{r_{ik}\}$, minimize J with respect to the centroids $\{\mathbf{c}_k\}$:

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \frac{\|\mathbf{x}_i - \mathbf{c}_k\|^2}{2}$$

Focus on a single cluster k. The term involving $\mathbf{c}_k$ is:

$$\sum_{i=1}^{n} r_{ik} \frac{\|\mathbf{x}_i - \mathbf{c}_k\|^2}{2} = \sum_{i=1}^{n} r_{ik} \frac{1}{2} \left(\|\mathbf{x}_i\|^2 + \|\mathbf{c}_k\|^2 - 2\mathbf{x}_i^\top \mathbf{c}_k\right)$$

Since the vectors are normalized, $\|\mathbf{x}_i\| = 1$ and $\|\mathbf{c}_k\| = 1$, we have:

$$\sum_{i=1}^{n} r_{ik} \frac{1}{2} \left(1 + 1 - 2\mathbf{x}_i^\top \mathbf{c}_k\right) = \sum_{i=1}^{n} r_{ik} \left(1 - \mathbf{x}_i^\top \mathbf{c}_k\right)$$

Now, take the derivative of the above with respect to $\mathbf{c}_k$ and set it to zero:

$$\frac{\partial}{\partial \mathbf{c}_k} \sum_{i=1}^{n} r_{ik} \left(1 - \mathbf{x}_i^\top \mathbf{c}_k\right) = \sum_{i=1}^{n} r_{ik} \mathbf{x}_i = \sum_{i=1}^{n} r_{ik} \mathbf{c}_k$$

Simplify:

$$\sum_{i=1}^{n} r_{ik} \mathbf{x}_i = \sum_{i=1}^{n} r_{ik} \mathbf{c}_k$$

Factor out $\mathbf{c}_k$:

$$\mathbf{c}_k = \frac{\sum_{i=1}^{n} r_{ik} \mathbf{x}_i}{\sum_{i=1}^{n} r_{ik}}$$

This is the mean of the normalized points in cluster k.

Centroid Update Step

For a fixed cluster assignment $\{r_{ik}\}$, update the centroids $\{\mathbf{c}_k\}$ as the **normalized mean** of all data points assigned to cluster k:

$$\mathbf{c}_k = \frac{\sum_{i=1}^{n} r_{ik} \mathbf{x}_i}{\|\sum_{i=1}^{n} r_{ik} \mathbf{x}_i\|}.$$

Algorithm Summary

The k-means algorithm with cosine distance alternates between two steps until convergence:

1. **Cluster Assignment Step**: Assign each point $\mathbf{x}_i$ to the cluster with the highest cosine similarity:
$$r_{ik} = \begin{cases} 1, & \text{if } k = \arg\max_j \mathbf{x}_i^\top \mathbf{c}_j, \\ 0, & \text{otherwise.} \end{cases}$$

2. **Centroid Update Step**: Update the centroid of each cluster as the normalized mean of its assigned points:
$$\mathbf{c}_k = \frac{\sum_{i=1}^n r_{ik}\mathbf{x}_i}{\|\sum_{i=1}^n r_{ik}\mathbf{x}_i\|}.$$

Conclusion

By normalizing the data points and centroids, the k-means clustering objective can be expressed in terms of both cosine distance and Euclidean distance. The equivalence:

$$\|\mathbf{x}_i - \mathbf{c}_k\|^2 = 2(1 - \mathbf{x}_i^\top \mathbf{c}_k)$$

enables seamless interpretation and implementation in the algorithm.

Bibliography

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Ting Chen et al. “A Simple Framework for Contrastive Learning of Visual Representations”. In: *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*. Vol. 119. Proceedings of Machine Learning Research, 2020, pp. 1597–1607.

[3] J. O. Cross-Zamirski, E. Mouchet, G. Williams, et al. “Label-free prediction of cell painting from brightfield images”. In: *Scientific Reports* 12 (2022), p. 10001.

[4] Sofia Dembski et al. “Establishing and testing a robot-based platform to enable the automated production of nanoparticles in a flexible and modular way”. In: *Scientific Reports* (2023).

[5] Andre Esteva et al. “Dermatologist-level classification of skin cancer with deep neural networks”. In: *Nature* (2017).

[6] Jeremy Irvin et al. *CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison*. 2019.

[7] TJ Keyes et al. “A Cancer Biologist’s Primer on Machine Learning Applications in High-Dimensional Cytometry”. In: *Cytometry Part A* 97.8 (2020). Epub 2020 Jun 30, pp. 782–799.

[8] G. Lee et al. “DeepHCS: Bright-Field to Fluorescence Microscopy Image Conversion Using Deep Learning for Label-Free High-Content Screening”. In: *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*. Ed. by A. Frangi et al. Vol. 11071. Lecture Notes in Computer Science. Springer, Cham, 2018, pp. 297–305.

[9] Phillip Lippe. *UvA Deep Learning Tutorials*. `https://uvadlc-notebooks.readthedocs.io/en/latest/`. 2024.

[10] F. Mualla et al. “Automatic Cell Detection in Bright-Field Microscope Images Using SIFT, Random Forests, and Hierarchical Clustering”. In: *IEEE Transactions on Medical Imaging* 32.12 (2013). Epub 2013 Aug 30, pp. 2274–2286.

[11] Hristo Todorov, Teresa Miguel Trabajo, and Jan Roelof van der Meer. “STrack: A Tool to Simply Track Bacterial Cells in Microscopy Time-Lapse Images”. In: *mSphere* 8.2 (2023), e0065822.

[12] Xiaosong Wang et al. “ChestX-Ray8: Hospital-Scale Chest X-Ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 3462–3471.

[13] McKell Woodland et al. “Dimensionality Reduction and Nearest Neighbors for Improving Out-of-Distribution Detection in Medical Image Segmentation”. In: *Machine Learning for Biomedical Imaging* 2 (UNSURE2023 special issue 2024), pp. 2006–2052.

Declaration on oath

I hereby certify that I have written my master thesis independently and have not yet submitted it for examination purposes elsewhere. All sources and aids used are listed, literal and meaningful quotations have been marked as such.

Bibin Babu, February 17th 2025

Consent to plagiarism check

I hereby agree that my submitted work may be sent to PlagScan (www.plagscan.com) in digital form for the purpose of checking for plagiarism and that it may be temporarily (max. 5 years)