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: January 15th 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 representation 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.

Abstract (de)

Die Behandlung von Bauchspeicheldrüsentumoren wird durch die komplexe Natur der Tumore und ihre vielfältigen Mikroumgebungen behindert. Diese Komplexität erfordert eine Untersuchung zur Identifizierung optimaler Medikamentenkombinationen und -konzentrationen, die auf die spezifischen Tumoreigenschaften jedes Patienten zugeschnitten sind. Diese Masterarbeit zielt darauf ab, die Wirksamkeit von Medikamenten zu bewerten, indem sie diese verschiedenen Medikamentenkombinationen und -konzentrationen einstuft. Die Bewertung basiert auf Merkmalen, die aus Helligkeitsmikroskopiebildern dreidimensionaler Tumorgewebsmodelle mittels Repräsentationslernen extrahiert werden. Die zentrale Herausforderung besteht darin, robuste Merkmale zu erlernen, die Veränderungen in diesen Tumorgewebsmodellen genau charakterisieren, die durch die Anwendung von Medikamenten über Zeit induziert werden. Diese Forschung zielt darauf ab, einen standardisierten und effektiven Ansatz zur Bewertung der Medikamenteneffizienz zu entwickeln, der möglicherweise die Behandlungsergebnisse für Patienten mit Bauchspeicheldrüsentumoren verbessert.

Acknowledgment

I would like to express my deepest gratitude to my supervisors, Prof. Dr. Magda Gregorova and Prof. Dr. Jan Hansmann, for their honest feedback and unwavering support throughout the course of this thesis. Their expertise and mentorship have been instrumental in shaping this research.

I would like to express my sincere gratitude to Dalia Mahdy, PhD student at Fraunhofer ISC Würzburg for giving me the opportunity to work on this thesis. Her generosity in sharing her expertise, guidance, and code was invaluable and formed a crucial foundation for my research.

To my family, thank you for reminding me that no matter how much I stumble or fall, I always have a foundation to rebuild from and the strength to start anew.

Würzburg, on 15.01.2025

_____

Bibin Babu

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 not only 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, but also 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 Wuerzburg laboratory utilizes a modular dual-arm robot-based system [3], equipped with incubators and bioreactors (see Figure 1.1 and Figure 1.2) 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 of 3D tumor tissue models using a customized microscope setup integrated into the robotic platform, offering flexibility in image acquisition according to experimental needs.

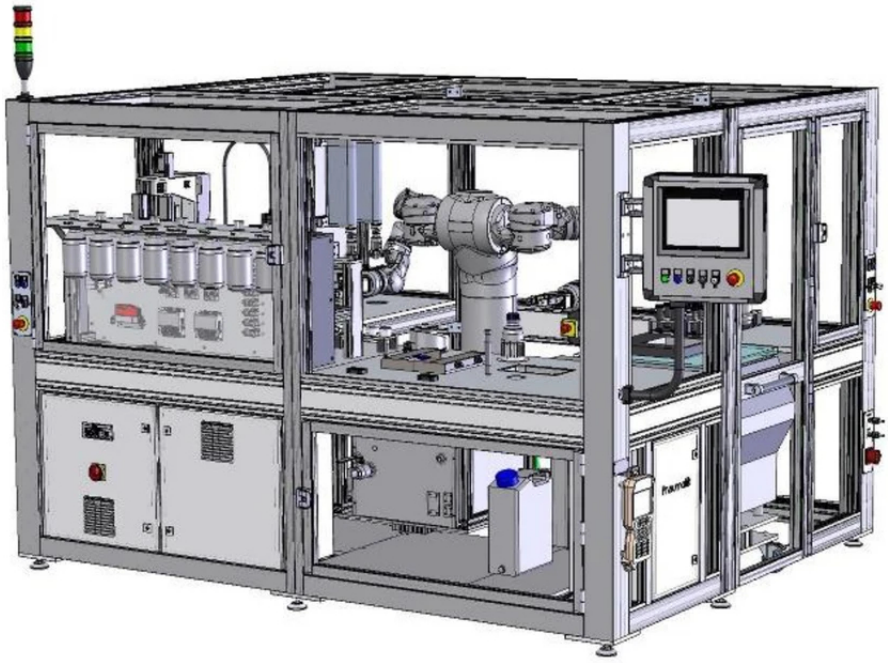

Figure 1.1: Robo platform

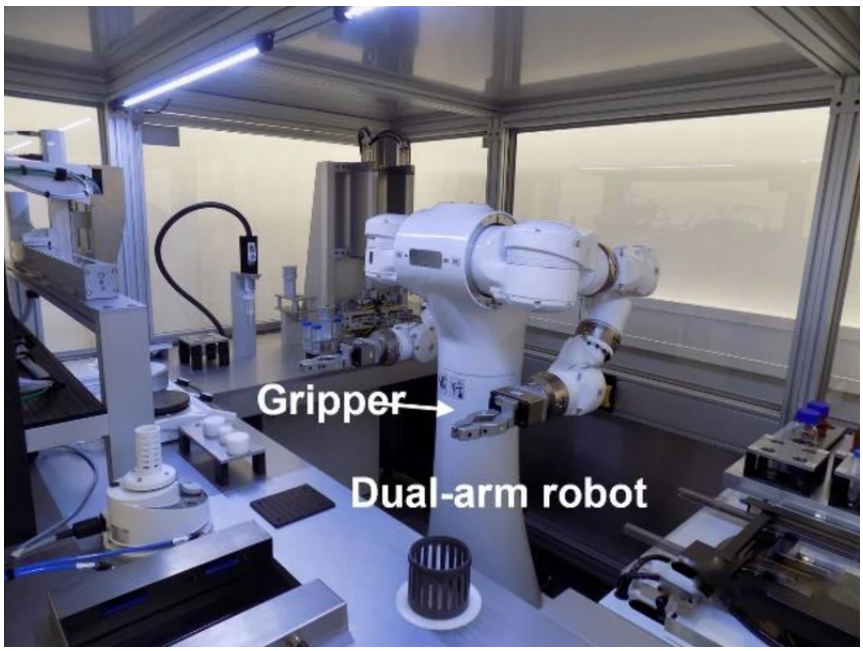

Figure 1.2: Dual-arm robot

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 1.3, adhering to standard procedures to ensure consistent and reliable experimental data.

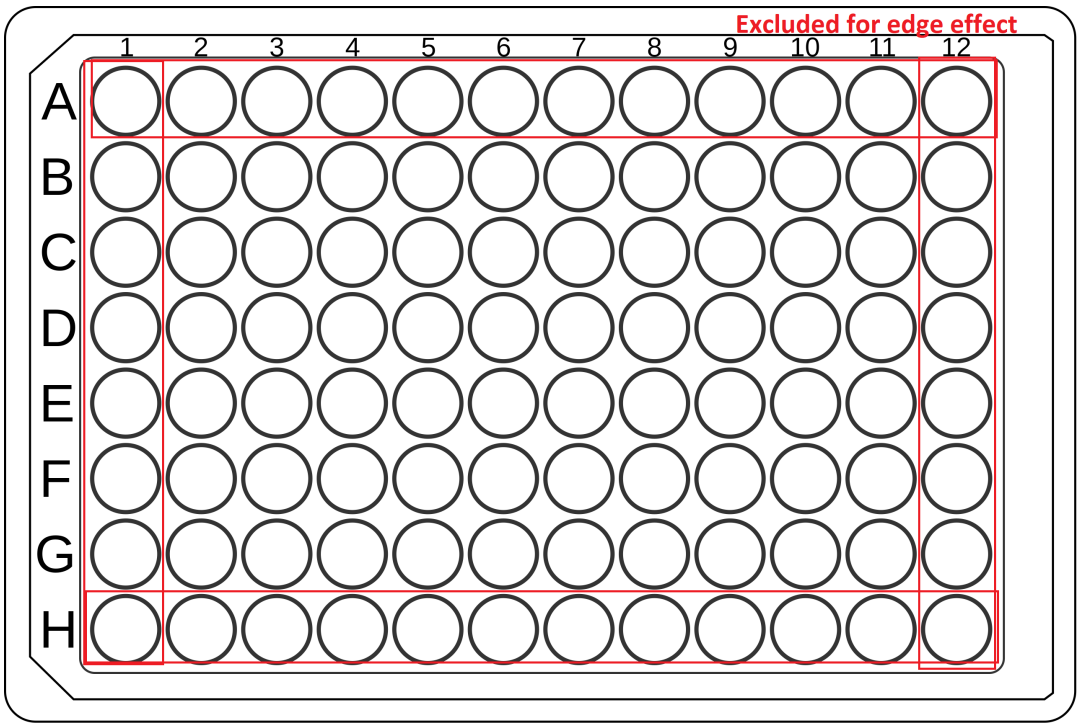

Figure 1.3: A well plate containing 96 wells where rows A, H and columns 1, 2 are excluded due to edge effects.

Based on the drug concentration applied to 3D tumor tissue models, the bright-field microscopy images we capture can be categorized into three:

Images of

1. Control (0 percentage drug applied)
    - For easiness, we refer to this category as “Untreated”
2. Single concentration (theoretically recommended single concentration of drug treatment)
    - For easiness, we refer to this category as “Single dose”
3. Drug screening: different drug combinations and concentrations used for experimental

study of drug efficacy, which may or may not result in the killing of surrounding non-tumor cells in the human body with potential side effects.

- For easiness, we refer to this category as “Drug screened”

The 60 inner wells are divided into sections to provide these three type of tumor tissues shown in figure 1.4 and figure 1.5.

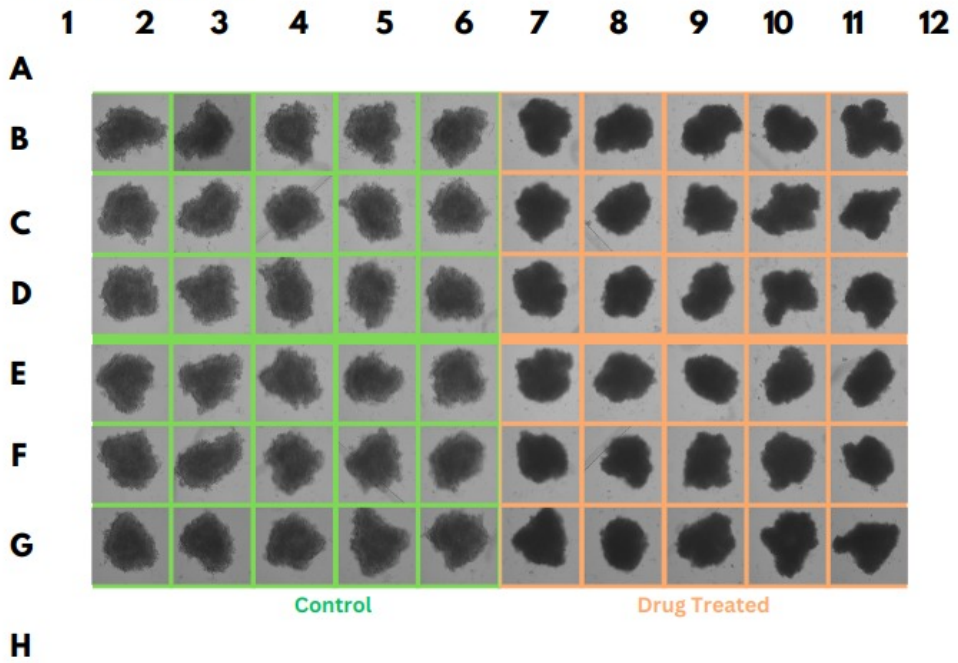

Figure 1.4: Well plate setup for the single-dose experiment where the left half remains untreated and the right half is treated with a single drug concentration. This image was taken three days after drug application, i.e., on day 10.

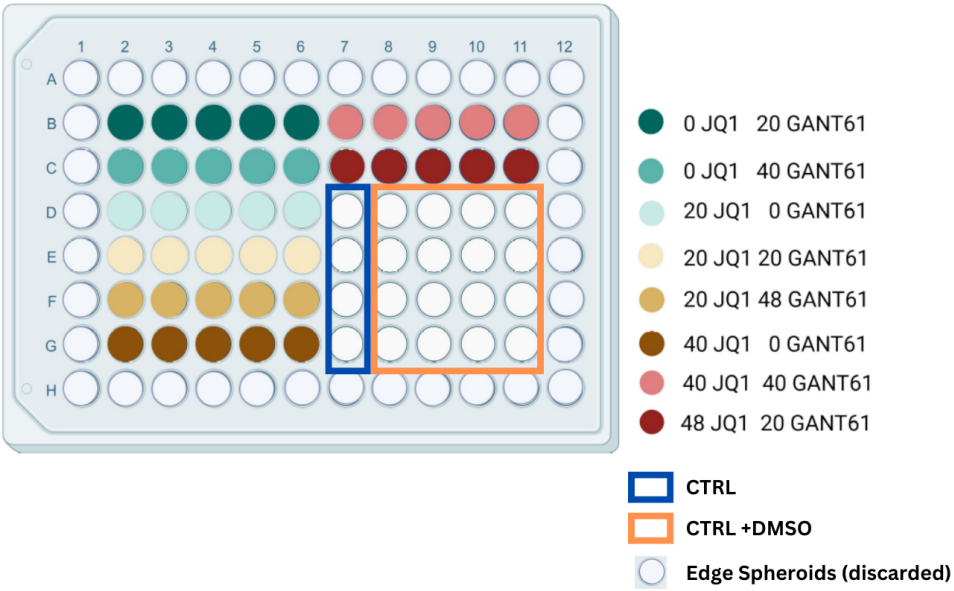

Figure 1.5: Well plate setup for the drug screening experiment where the majority of tumor tissues are treated with different combinations of drug concentrations (multi-colored wells), while some are left untreated (white wells bounded by boxes).

The 3D tumor tissue models develop in the wellplate progressively from day 1 to day 7, 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 and the tumor begins to diminish. To isolate the drug's effects, changes in tumor tissue deterioration are assessed on day 3 post-drug administration (day 10), in accordance with established medical protocols and previous research findings.

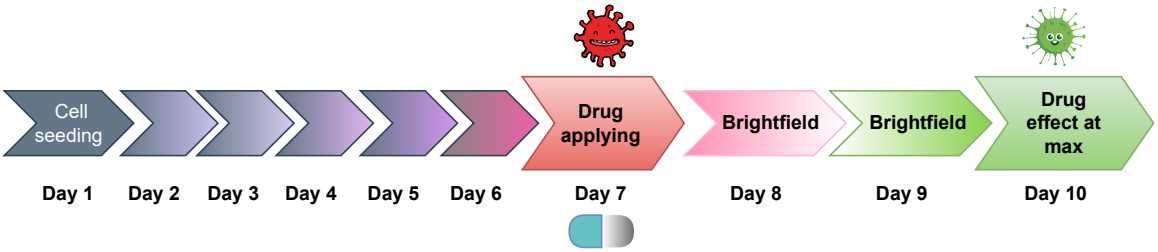

Figure 1.6: Illustrates the flow chart of time evolutoion of 3D tumor tissues.

We assess the efficacy of the drug by comparing the changes it induces in the untreated bright-field microscopy images over a period of time. The current methods to differentiate these changes involve studying the alterations from day 7 (after applying the drug) 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, there may be other hidden information or patterns within these bright-field microscopy images that are not human-interpretable. This potential can be explored using representation learning techniques. Additionally, this method can provide more standardization compared to manual assessment.

2 Objective

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

3 Research questions

1. Can we learn latent features that capture the alterations induced in three-dimensional tumor tissue models by drug application over a period of time?

2. Will these features can effectively establish a ranking of drug efficacy from bright-field microscopy images?

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

3. What could be reasonable metrics, such as L2 loss or cosine similarity, for supporting the relative assessment of drug efficacy?

4. can unsupervised learning be used to tackle this ranking problem since lack of label is our main challenge?

5. How do we deal with the position change in the image collection of day 10 if we want to use prediction model as one of our rankng startegy.

6. How do we incorporate the fact about brightness/blur change in the image collection due to environmental or microscope variations?

7. how do we deal with the insufficient data problem?

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 [11] ; Esteva et al., 2017 [4] ; Irvin et al., 2019 [7]), which is suboptimal due to the drastically different characteristics of medical images. Recent studies have shown promising results using unsupervised contrastive learning on natural images, but these methods have limited effectiveness on medical images because of their high inter-class similarity.

To address these challenges, researchers have proposed various innovative approaches. ConVIRT [13] 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, 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. BYOL (Bootstrap Your Own Latent) [5] addresses these limitations by using an architecture with online and target neural networks, which does not require negative pairs and is more robust to the choice of image augmentations compared to contrastive methods.

While SimCLR has achieved impressive success in the computer vision field, directly applying it to the time series domain often yields poor performance due to its data augmentation and feature extractor not being tailored to the temporal dependencies inherent in time series data. To address this limitation and to obtain high-quality representations of univariate time series, [12] proposed TimeCLR, a framework that combines the strengths of Dynamic Time Warping (DTW) and InceptionTime. Drawing inspiration from the DTW-based k-nearest neighbor classifier, they introduced DTW data augmentation. This technique generates phase shifts and amplitude changes targeted by DTW, preserving the time series structure and feature information. By integrating the advantages of DTW data augmentation and InceptionTime, TimeCLR method extends SimCLR and adapts it effectively to the time series domain. [9] conducted a comprehensive comparative analysis between contrastive and generative self-supervised learning methods for time series data, focusing specifically on SimCLR and MAE (Masked Autoencoder). They observed that, overall, MAE tends to converge more rapidly and delivers impressive performance, particularly when the fine-tuning dataset is relatively small (around 100 samples). However, in scenarios with larger datasets, SimCLR demonstrates a slight but consistent outperformance over its generative counterparts.

Another recent alternative study for self-supervised visual representation is DINO [1], which can be also interpreted as a form of self-distillation with no labels. DINO provides new prop-

erties to Vision Transformers that stand out compared to convolutional networks.

SupCon [8] extends the self-supervised batch contrastive approach to the fully-supervised setting, allowing us to effectively leverage label information. Clusters of points belonging to the same class are pulled together in embedding space, while simultaneously pushing apart clusters of samples from different classes. The drug applied to tumor samples could be used as labels for the bright-field microscopy images.

5 Data Description

5.1 Data set

As we explained in introduction chapter, dataset consist of bright-field microscopy images which basically works by: Sample illumination is transmitted (i.e., illuminated from below and observed from above) white light, and contrast in the sample is caused by attenuation of the transmitted light in dense areas of the sample. The typical appearance of a bright-field microscopy image is a dark sample on a bright background, hence the name. In our case, we have 3d tumor model as dark grayish color in the bright background.

1. challenges: 1.1 limited data set for day 7 to day 10 prediction model 1.2 day 10 image can be flipped, blured, brightness change, position change (position cahnge is due to because when we take day 10 images we have to bring the well plate outside and sometimes when we put that into microscope the position is already changed). Relativly small position of tumor cell in the image from center to other directions change can be resolved for some extend using center crop approach, but if the cell is in the edge then center crop won't help. i didn't do any data preprocessing specifically for that problem. but I expect the croping and resizing to 96*96 will be a solution. because when we do that data transformation, augmented pairs will be positioned differently.

only state if you can show the proof 1.3 same drug can have different effect on day 10 (example: huge variation in: ds 61 g6 and 41 gp6. less variation: RBTDS 4.1 and 4.2 gp 3) initially

i thought I will be able to use as drug gp to evaluate, because of this difference, the difference can be due to different patient cancer cell? I choosed images from different gps which have huge debris amount visually. it was few so I didn't code my self.

The original images are approximately 2500×2500 pixels in size, in 16-bit grayscale, and consist of multiple channels. These channels come from taking images at different focal planes in brightfield microscopy. The number of channels can vary, as you can take images at any number of focal planes. the sharpness of an image is dependent on how well the focal plane of the microscope aligns with the depth of the sample. Only one focal plane (layer) will be perfectly in focus, while others might appear blurry because they are slightly above or below the focal point. Combining these focal planes later in computational analysis can provide richer data, even if some channels are blurry individually. thats why I initially decided to use multiple channels. Images I got from the lab consisit of majority have 3 channels while minorly as very few have one channel and 5 cahnnels.

add the details of no of channels/image and how does it calcualtes the sharpness intuiton behind it: https://chatgpt.com/share/675f440d-3868-8010-ae82-ad6cdca13c5d last explanation

However, for the ease of use to integrate with our pretarined architecture we use (both resnet18 and unet), I determined to start with 3 channels per image. what matters is most sharped because most sharpest channels have more less texture/edge information information than less sharpness less information (less texture/edge information) how to calculate the sharpness and why does it make sense explained in data preprocess section.

also one thing to point out is that the depth of focus or focal length amoung the channels can be different/same in different mages ? ask Dalia when I observed the channels/layers in each image, there is a slight gradual shift (visually) in the position amoung them. ie, if the first channel have the tumor model in the center, if look at the last channel we can see 0.1 mm shift in random direction.

the images with one channel will duplicate the channels to make upto 3.

investigate the performance of single channel too because maybe thats enough for us too to able to get our expected result. how to choose the focused layer will be explained the data preprocessing section. correct the figure.take small patch and show the difference in edge effect. but neverthless we can't give show the focused one in whole image? check for image like that maybe you will find in 5 layers or maybeyou will find in 3 layers

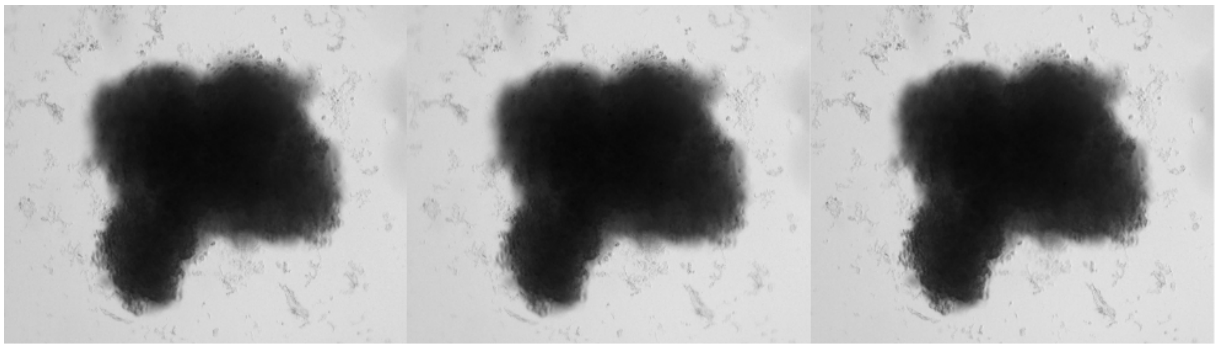

Figure 5.1: Illustration of three layers per image: A, B, and C. The three layers looks visually similar, with slight differences in focal planes. In this figure, A is the sharpest/focused layer.

Figure 5.2 illustrates that, even with the application of the same drug at the same concentration, the morphology of 3D tumor tissues changes differently.

correct the figure. show drug screen with same drug concentration we have diferent effect between different experiment. this maybe due to environmental factors or anything

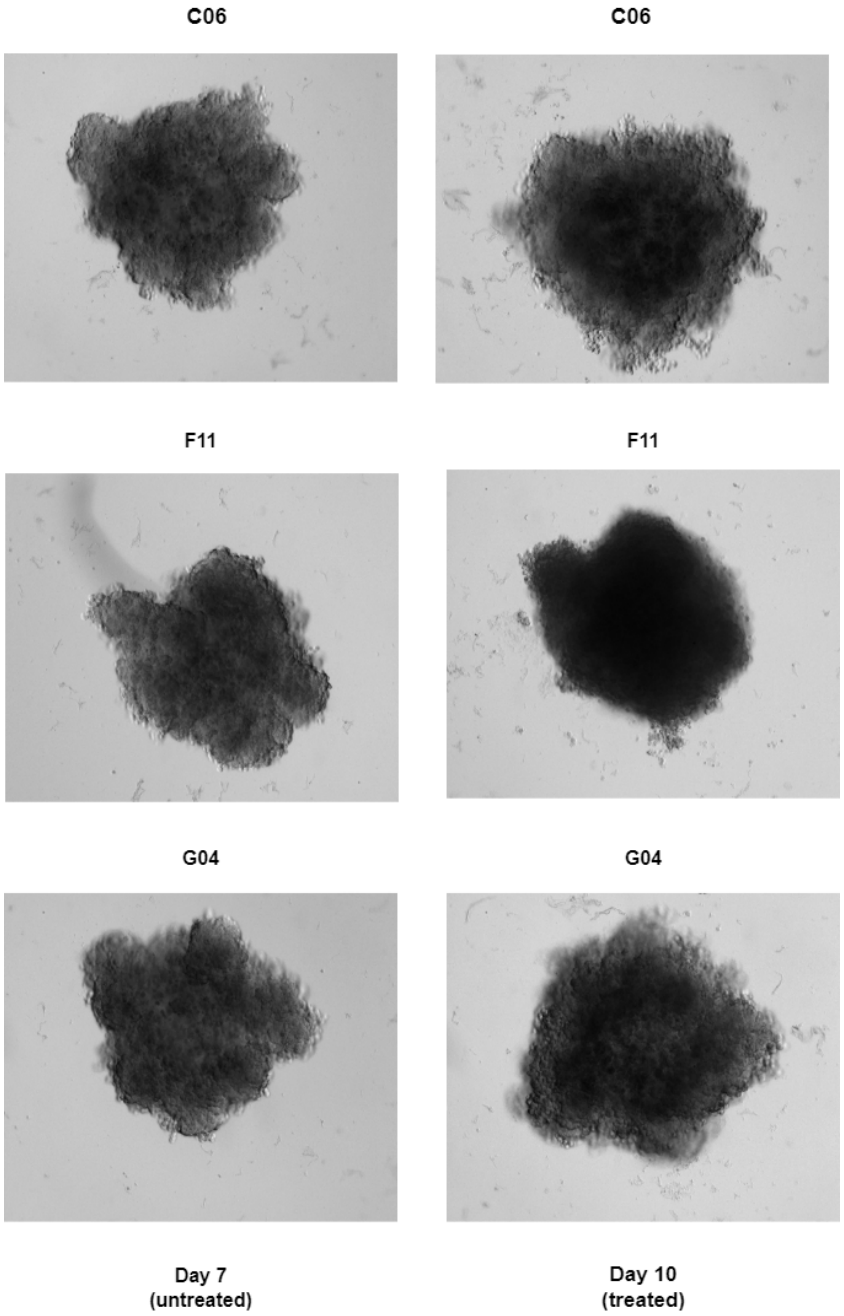

Figure 5.2: C06, F11 and G04 are well names in the well plate.

The table below shows the division of three different types of image datasets, as explained in the section 1.1.

Class	Drug Screened	Single Dose	Untreated	Total
No. of Images (%)	12 (3%)	204 (60%)	150 (37%)	366

Table 5.1: Dataset Class Overview

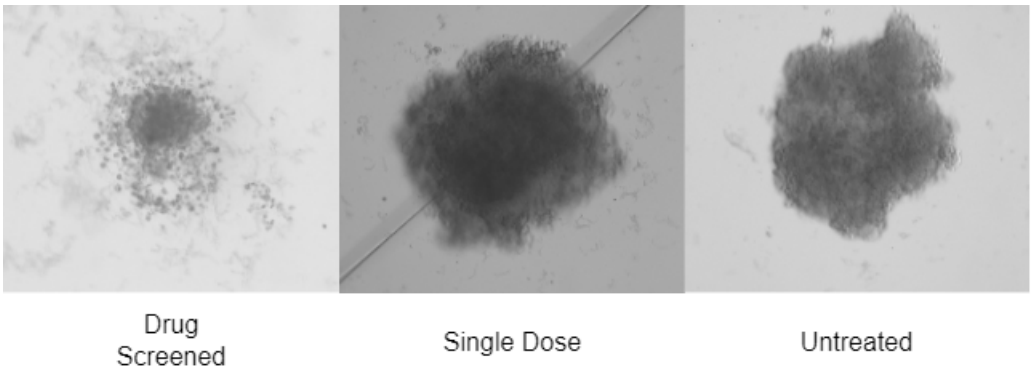

Figure 5.3: Three different types of images: Drug Screened, Single Dose, and Untreated as mentioned in section 1.1.

An 8-bit image encompasses 256 color tones (ranging from 0 to 255) per channel, whereas a 16-bit image accommodates 65,536 color tones (ranging from 0 to 65,535) per channel, in our case 65,536 shades of gray. Retaining the original 16-bit depth is crucial for two primary reasons:

1. 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 only allow 256 possible values, the finer variations in intensity that are present in 16-bit images become compressed as illusstrated in 5.4. For example, two distinct values in 16-bit (such as 30,000 to 30,048) could map to the same 8-bit value (for instance, both might be mapped to 117). This results in the loss of subtle intensity differences, which can be crucial in our image task, where minute variations in intensity can be indicative of important features such as gradual trasnsition of dark color frfom center to border or amount of debris surrounded to the tumor cell.

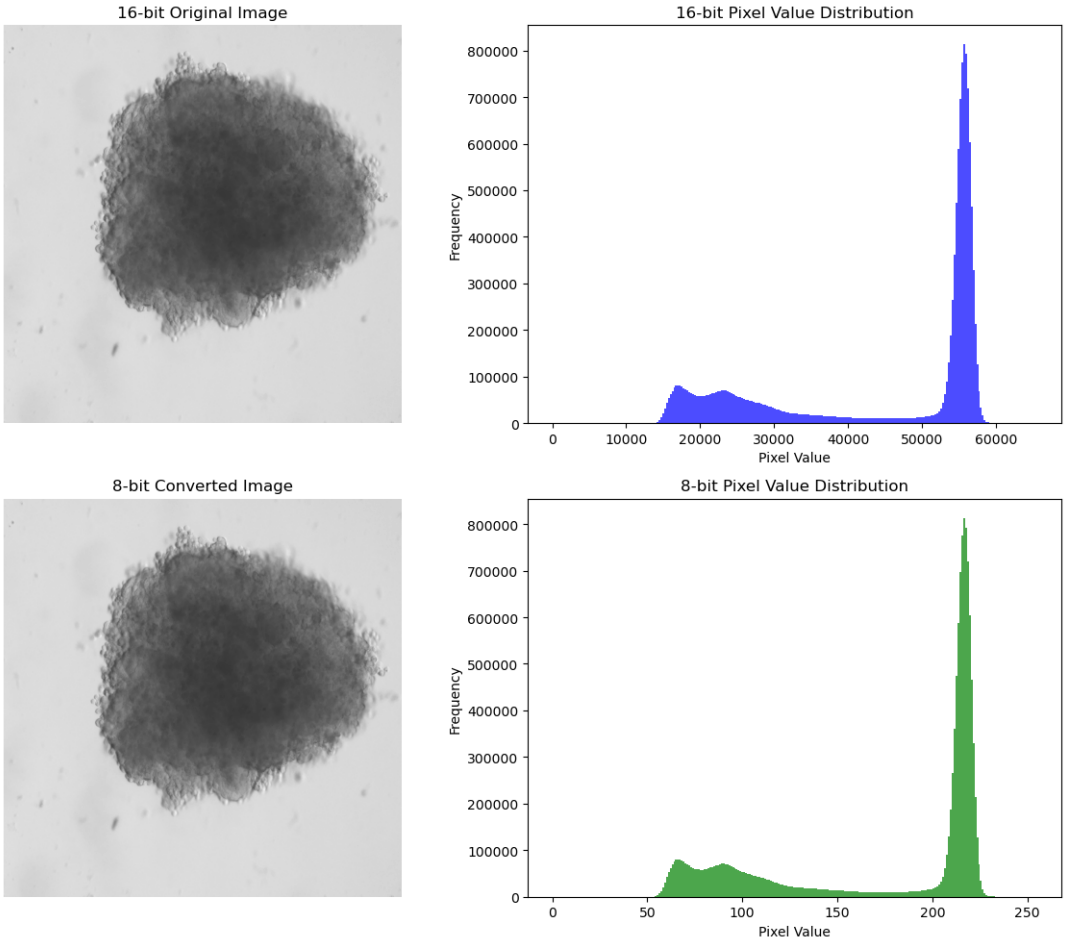

Figure 5.4: 8-bit vs 16 bit data lose comparison

2. During data augmentation processes that involve substantial alterations in brightness, contrast, or color, an 8-bit image—already limited to 256 tones—could lose up to 50 percentage of these tones, leaving only 128 levels of color and tone. This reduction can lead to ”banding,” where areas with smooth transitions in tone exhibit visible stripes with jagged edges. In contrast, a 16-bit image, even with a 50 percentage reduction in tones, would retain over 32,000 levels. This higher tonal range allows for smoother transitions, better edge preservation, and enhanced accuracy in color and hue representation. As a result, the dynamic range—the difference between the lightest and darkest areas of the image—remains much more effectively preserved in 16-bit images than in 8-bit images.

In our case, the maximum reduction in unique pixel values for the 8-bit images, regardless number of channels was found to be 99.27 percentage after 3000 epochs of random color jitter applied using `torch.transforms.RandomApply([transform.ColorJitter(brightness=1, contrast=1, saturation=1, hue=0)], p=1)` as shown in figure 5.5 and 5.6, whereas for the 16-bit single-channel images (where one sharp layer was extracted from all three layers and considered as input for data augmentation), the reduction in unique pixel values was only 49 percentage after 3,000 epochs of random color jitter, as shown in Figure 5.7.

8-bit three-channel image example before and after data augmentation:

- Number of unique pixel values in the original image: 137
- Number of unique pixel values in the augmented image: 3
- Original Image - Minimum pixel value: 33, Maximum pixel value: 170
- Augmented Image - Minimum pixel value: 0, Maximum pixel value: 2

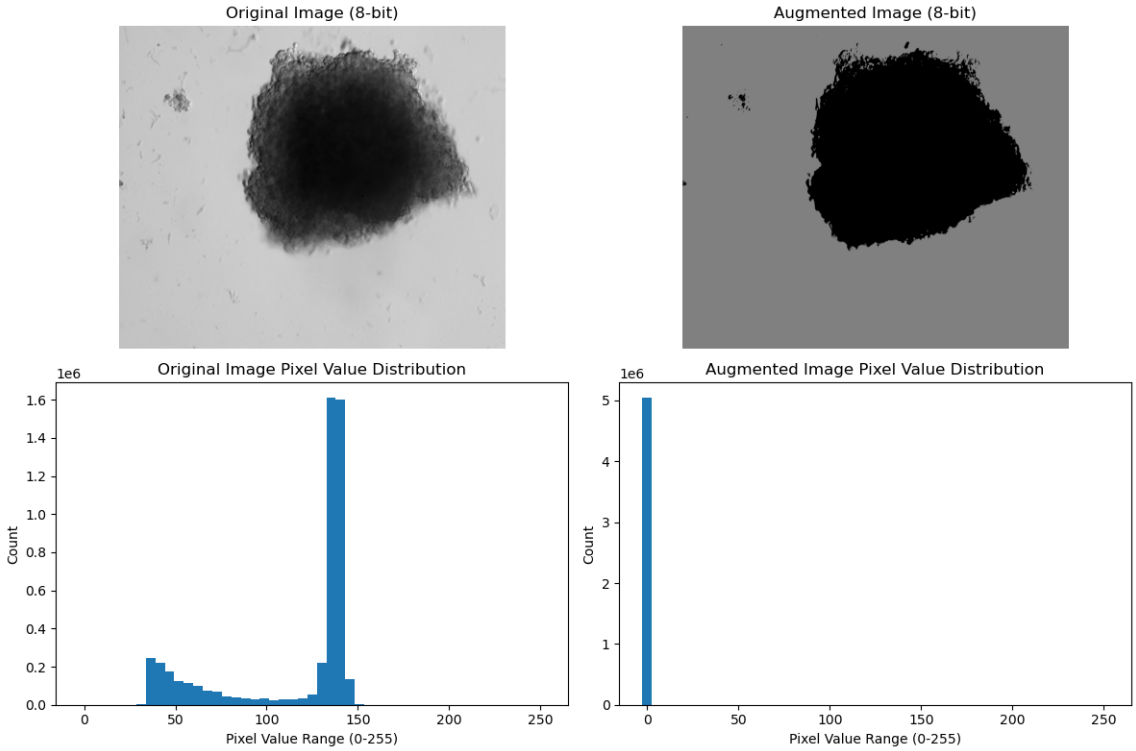

Figure 5.5: 8-bit three-channel image after 3000 epochs of random color jitter applied using PyTorch. Reduction percentage in unique pixel values: 97.81%

8-bit single-channel (sharp layer) image before and after data augmentation:

- Number of unique pixel values in the original image: 137
- Number of unique pixel values in the augmented image: 3
- Original Image - Minimum pixel value: 33, Maximum pixel value: 170
- Augmented Image - Minimum pixel value: 3, Maximum pixel value: 3

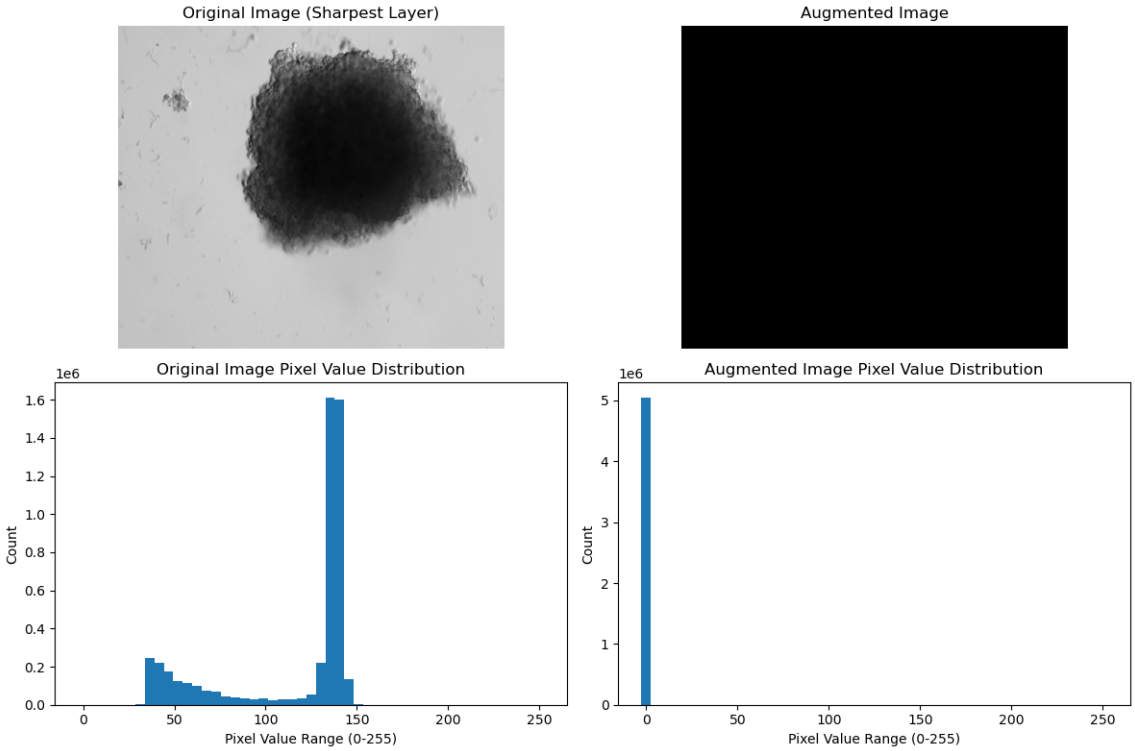

Figure 5.6: 8-bit single-channel image after 3000 epochs of random color jitter applied using PyTorch. Reduction percentage in unique pixel values: 99.27%

16-bit single-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: 1058
- Original Image - Minimum pixel value: 0.13064774870872498, Maximum pixel value: 0.6666666865348816
- Augmented Image - Minimum pixel value: 0, Maximum pixel value: 1

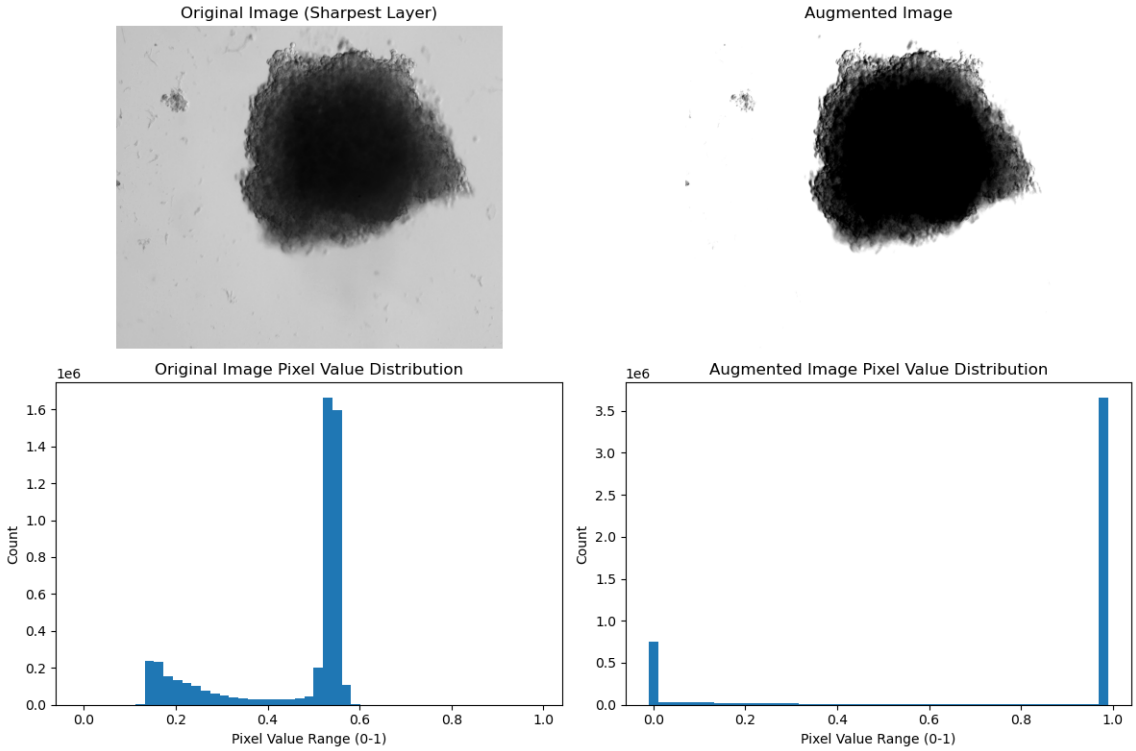

Figure 5.7: 8-bit single-channel image after 3000 epochs of random color jitter applied using PyTorch. Reduction percentage in unique pixel values: 49.88%

6 Methodology

The goal is to leverage representation learning of bright-field microscopy images to develop a ranking/ordering scale (1 to n) for these images. since we don't have ground truth labels to rank the images except control images. my strategy was to simplify the current ranking problem to only scale/order/rank images using only control, single dose and exploded images. Thereason to pick these groups 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 is clinically recommended at the moment (eventhough we don't know how much drug effected or howmmuch it killed the cancer) and exploded are visually exploded from the original cancer cell meansing we can visually see debris around the cancer cell potentially which may potentially harm the surrounding goof cells. once if we can order those small subset of entire images, we can add the other image as inference to see where they plotted relative to control or single dose or exloded in this scale.

6.1 Data preprocessing

Detailed study/research/experiments on data augmentation and image preprocessing techniques sepcifically for our 16 bit gray scale image are still need to be done. Currently, as the focus is on creating the complete pipeline, the standard data augmentation combination (which showed high performance for SimCLR downstream tasks) from the SimCLR [2] paper is being used, as shown below.

1. We can't center crop it because for some of cancer cells are not centered in the image instead they are close to egdes. so we have to make sure that when we crop it it should include the cacne cell fully. the solution is find the boundary of cancer cell and get the bounding box of cancer cell then make crop to required size. original image width size is: 2456*2054 (H*W). crop the original image to have H = W. since the cancer cell debris spread across the width, we didn't change the width to include the debris, instead we reduced the hight to same size of width. hence we get H = W = 2054. if cancer cell is formed for instance, the cell it self spread acroos one diemsion that means its not valid cultvation by robot so we can digard it. ie maximum area of cancer cell should be included in this square 2054*2054 size image. advantage of croping like this are:

   a) remove unnecessary background which contains no information

   b) there will be no shear during resize to 96*96 augmentation like in the figure.

   c) give proof for above statement

2. Normalize the 16-bit image to [0, 1] for the following reasons:

   a) Ideally, normalization should be done at the end after augmentations to ensure scaled input to the neural network, but in our case, we have to normalize first since the augmentation with `torch.transform.ColorJitter` didn't work without scaled data.

   b) `torch.transform.ToTensor()` didn't scale the data points to the [0, 1] range.

3. Perform the following augmentations:

   a) Apply a horizontal flip.

   b) Randomly crop the image and resize it to 96×96. i choosed this small H and W as image size to feed train our model because of two reasons: 1. computational fast,

so that we can complete the pipeline in time. 2. if we can get good performance in ranking with this small image sizes (ie less pixel details compared to 2054 * 2056) that means we can improve the performance with larger image sizes.

c) Randomly change the brightness, contrast, saturation, and hue of the cropped patch.

4. Perform Z-score normalization after data augmentation for the following reasons:

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.

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

Visualisation of before and after preprocessing of image shown in figures 6.5 to 6.11.

6.2 Data augmentation

I started with data augmentation just like simclr paper did that is strong data augmentation, just to get to play around , beacuse it was easy to do , since I can directly adapt code from tutorial just to have complete pipeline.

I found that those strong augmentations transformed to distribution out of the original distribution like we see in the figures below. specifically color jitterness including brightness, contrast, hue, saturation.

Interestingly, for 16-bit images with 3 channels, instead of a reduction, there was an increase in the number of unique pixel values—by a maximum of 258,757 percentage. The issue with this increase is that after data augmentation, the new pixel values are not distributed similarly to the

original image. Instead, they shift to the two extremes, such as 0 or 1, or sometimes pushing values to both 0 and 1, which deviate significantly from the original image distribution, as shown in Figures 6.1 and 6.2.

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
- Original Image - Minimum pixel value: 0.13064774870872498, Maximum pixel value: 0.6874189376831055
- Augmented Image - Minimum pixel value: 0.022128667682409286, Maximum pixel value: 0.11041323840618134

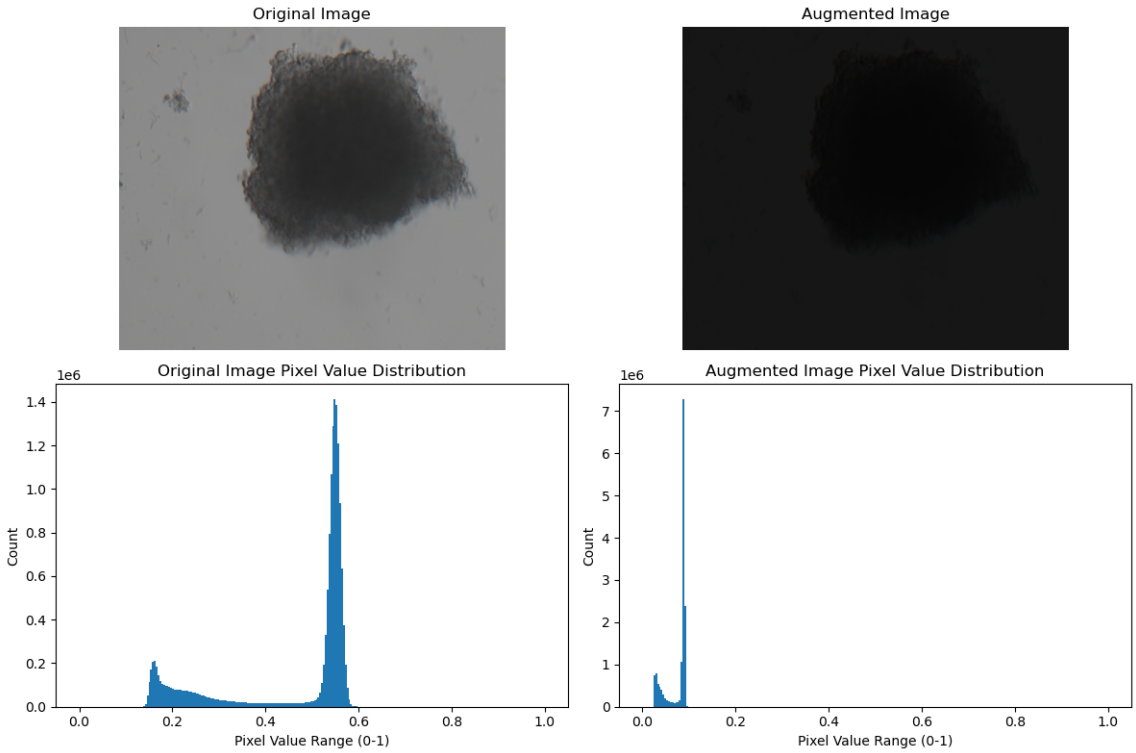

Figure 6.1: 16-bit three-channel image after 3000 epochs of random color jitter applied using PyTorch.

Another example of a 16-bit three-channel image before and after data augmentation:

- Original Image - Unique pixel counts per channel: 2137
- Augmented Image - Unique pixel counts per channel: 1686717
- Original Image - Minimum pixel value: 0.1306, Maximum pixel value: 0.6874
- Augmented Image - Minimum pixel value: 0.1970, Maximum pixel value: 0.3748

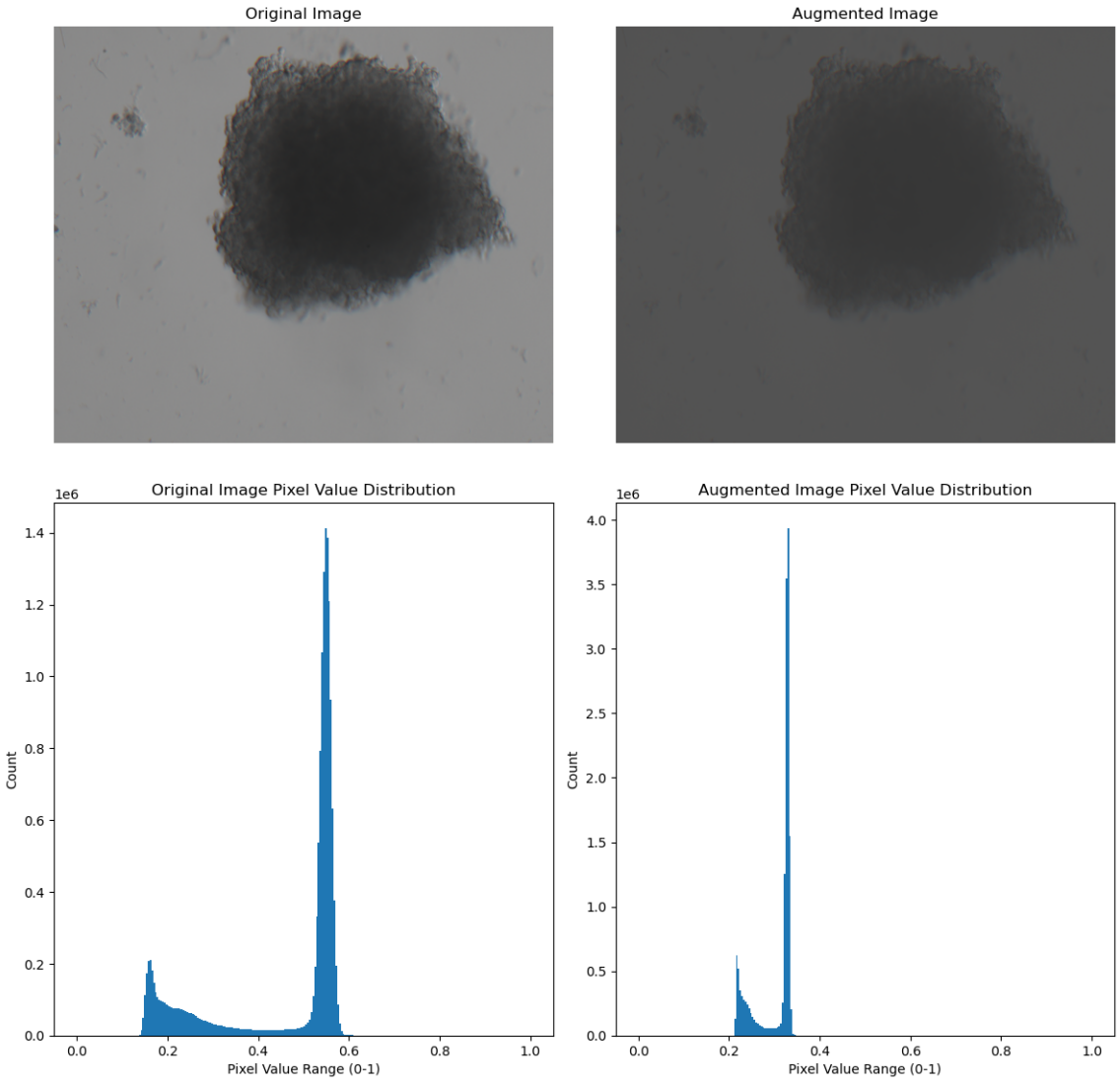

Figure 6.2: 16-bit three-channel image after 3000 epochs of random color jitter applied using PyTorch.

16-bit single-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: 1058
- Original Image - Minimum pixel value: 0.13064774870872498, Maximum pixel value: 0.6666666865348816
- Augmented Image - Minimum pixel value: 0, Maximum pixel value: 1

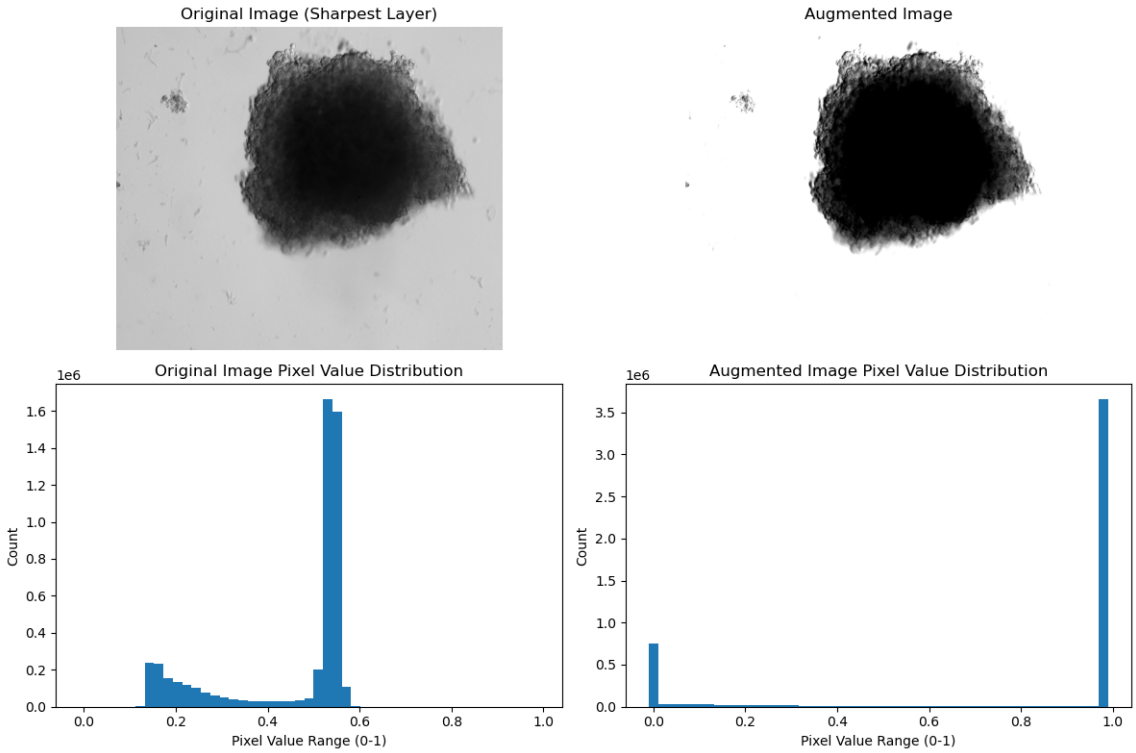

Figure 6.3: 8-bit single-channel image after 3000 epochs of random color jitter applied using PyTorch. Reduction percentage in unique pixel values: 49.88%

49% maximum reduction for 16-bit single-channel data augmentation with color jitter is also not ideal, as it diminishes the gradual spread of darker regions as happened in original image, as as observed in figure 6.3. One potential solution is to experiment with specific parameters within the color jitter transform instead of using random values, ensuring that the reduction in

the number of unique pixel values does not exceed, for example, 30%. Another option would be to write a custom Python function, depending on the available time. Other augmentations from PyTorch work fine in this experiment.

then I removed contrast, saturation and hue. because if I tried to make those augs as invariant that means I'm making control and treated as similar because if I increase or decrease contrast or saturation or hue it increase/decrease the darkness of cancer cell to have similarity. which doesn't make sense because we want exact opposite of it.

but if I use the same croping percentage as the original simclr does, that means I crop 0.1 to 1 percentage which doesn't make sense because that menas we make smilarity to small background to cancer which is unnecessary. so i chaneg the percentage to 0.4-1.

Questionable below one:

lasty it may be possible to learn efficient feature extraction by not applying cropping, ie learn by seeing full picture (big picture). I understand this is not good argument because then dog and cat have same color and maybe similar shape but still learns to differentiate using simclr. But maybe for time predictoin ranking model it helps. because model have to predict the change from day 7 to day 10. and most of the ime the size/shape changes. especially to the category which are exploded, they produce debris. so maybe data augmentation seeing the time change.

6.3 Train SimCLR as SSL model

For step 1, as explained in chapter 6, SimCLR was used as the first model for self-supervised learning (SSL). Future work: Later, other models such as masked autoencoders and DINO will be explored, depending on the available time. Why we would like to try other models? Because SimCLR demands larger batch size and more data for better performance which we

don't have.

Model

The Resnet18 [6] 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 6.1 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 6.2

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

$$\ell_{i,j} = -\log \frac{\exp\left(\operatorname{sim}\left(z_i, z_j\right)/\tau\right)}{\sum_{k=1}^{2N} \mathbf{1}_{[k\neq i]} \exp\left(\operatorname{sim}\left(z_i, z_k\right)/\tau\right)} \tag{6.1}$$

which we can reformualte as:

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

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

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

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

Substituting that back into the equation:

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

This gives us:

$$\ell_{i,j} = -\frac{\text{sim}\left(z_i, z_j\right)}{\tau} + \log\left[\sum_{k=1}^{2N} \mathbf{1}_{[k\neq i]} \exp\left(\text{sim}\left(z_i, z_k\right)/\tau\right)\right] \tag{6.2}$$

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.

Equation 6.2 implemented as the loss function in our experiments.

The above standard SimCLR loss function and data augmentation combination with the ResNet18 model will be used as the initial benchmark for experiments, and in the future, the following variations will be explored.

Variation ideas:

1. Explore different data augmentation combinations by researching the best augmentations for medical grayscale images, and applying intuitive approaches beyond the standard SimCLR [2] data augmentation combinations as explained in section 6.1.

2. Each image is treated as an RGB image with 3 channels, and two of the best-performing data augmentations, which yielded high performance for our downstream task.

3. One channel is considered as the anchor (the most sharpened layer), and the others are treated as the two augmentations.

4. One channel is considered as the anchor (the most sharpened layer), and two of the best-performing data augmentations, which yielded high performance for the our downstream task.

5. Supervised SimCLR: Ensure that no images from the same breed/class are included in the negative samples.

6. Since SimCLR architecture [2] allows for flexibility in model selection, and explore other pretrained models than Resnet18 suitable for medical grayscale images. For example pretrained U-Net [10] model for MRI brain images from PyTorch.

7. Include the anchor as a positive sample, i.e., 3 augmentations in total (1 anchor as augmentation and the other 2 layers as augmentations). This resembles to triplet loss (not sure, need to be studied)

For variations 5 and 7 we need to modify the loss function since it includes more than 2 augmentations.

Variations implementations:

The two variations tried so far differ only in how they handle the image for data augmentation.

In the first variation, we take a 3-channel image and treat it like a standard RGB image, applying SimCLR-style augmentations to create two augmented versions.

In the second variation, we take a 3-channel image and compute the sharpness of each layer by calculating the magnitude of the gradient of pixel intensities in the x and y directions, which indicates edge strength and provides a measure of how sharp the transitions between pixel values are. The sharpest layer is used as the anchor, while the other two layers are treated as augmentations.

Variation 1:

Input to model (train loader dimension) :

- aug1: torch.Size([16, 3, 96, 96]) (batch size, no of channels, H, W)
- aug2: torch.Size([16, 3, 96, 96]) (batch size, no of channels, H, W)

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

- torch.Size([16, 512, 1, 1]) (Batch size, standard resnet18 output dimension after avg pooling, H,W)
- This output feature will be used for further downstream task.

Model output after projection head:

- torch.Size([16, 20]) (Batch size, no of values in feature vector)

- No of values in feature vector is a variable which we can change and experiment which will give better accuracy.

The figure below shows the anchor and its two augmented versions as explained in section 6.1.

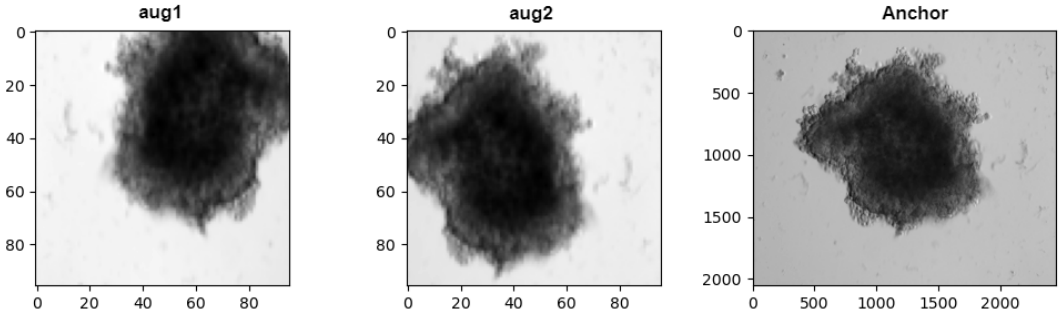

Figure 6.4: Sample 1: Anchor (the preprocessed original image) with 3 channels and its augmentations

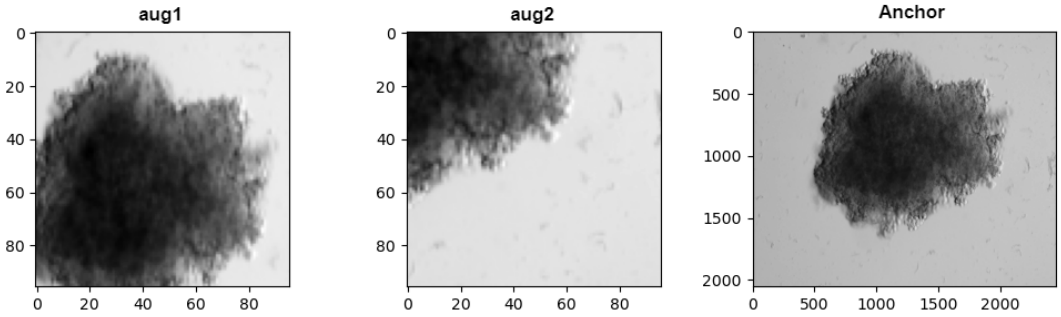

Figure 6.5: Sample 2: Anchor (the preprocessed original image) with 3 channels and its augmentations

Variation 2:

Input to model (train loader dimensions) :

- aug1: torch.Size([16, 1, 96, 96]) (batch size, no of channels, H, W)
- aug2: torch.Size([16, 1, 96, 96]) (batch size, no of channels, H, W)

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

- torch.Size([16, 512, 1, 1]) (Batch size, standard resnet18 output dimension after avg pooling, H, W)

- This output feature will be used for further downstream task.

Model output after projection head:

- torch.Size([16, 20]) (Batch size, no of values in feature vector)

- No of values in feature vector is a variable which we can change and experiment which will give better accuracy.

The figure below shows the anchor and its two augmented versions as explained in section 6.1.

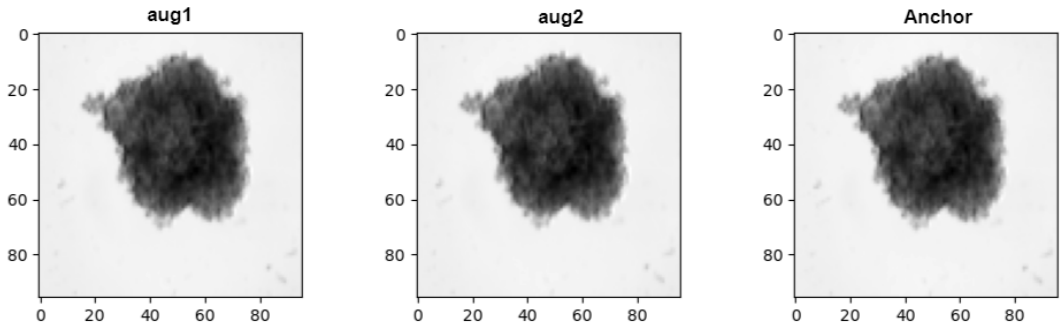

Figure 6.6: Sample 1: Anchor (the preprocessed sharpest layer amoung all 3 layers) with one channel and its augmentations

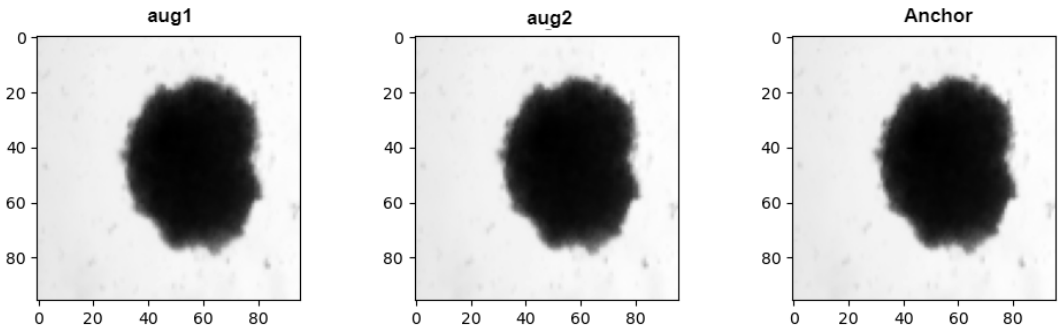

Figure 6.7: Sample 2: Anchor (the preprocessed sharpest layer amoung all 3 layers) with one channel and its augmentations

6.4 Ranking strategy 1: Using CAE

6.4.1 Day7 to day7 reconstruction

I start with classical anomaly detection approach that we train a model to reconstruct/predict day 10 image from day 10 image exclusively on the untreated images. Since the day 10 prediction model is trained solely on untreated images, we expect the the inference loss/metric (i.e., the difference between the predicted and actual Day 10 image) will be very small for untreated images. Conversely, the inference loss/metric will increase for treated images as their predictions deviate from those of untreated images. This inference loss/metric will be used as the feature for the ranking/order scale, where the initial images will start with untreated images that have very small inference loss/metric, and the scale will end with images having high inference loss/metric in ascending order.

6.4.2 Day7 to day10 predcition

day10 predcition

1. **Step 1:** Create a latent space representation of all images, including untreated, clinically recommended, and drug screening 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. In other words, feature vectors of similar images will be more linearly separable.

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

3. **Step 3:** Perform inference on the representations of test images, which include untreated, clinically recommended, and drug screening images.

4. **Step 4:** Perform step 2 and step 3 on images instead of simclr feature vectors for comparitive study.

Since the day 10 prediction model is trained solely on the representations of untreated images, the inference loss/metric (i.e., the difference between the predicted and actual Day 10 image representations) will be very small for untreated images. Conversely, the inference loss/metric will increase for treated images as their representations deviate from those of untreated images. This inference loss/metric will be used as the feature for the ranking/order scale, where the initial images will start with untreated images that have very small inference loss/metric, and the scale will end with images having high inference loss/metric in ascending order. Determining a reasonable inference loss/metric will be one of the research problems to tackle.

Delta predcition

1. **Step 1:** Calculate the difference/delta between the day 7 and day 10 simclr feature vectors exclusively on the untreated images.

2. **Step 2:** Train a prediction model where input is day 7 feature vector and target will be Delta.

3. **Step 3:** Perform inference on the test set where input will be day 7 feature vector of all images which include untreated, clinically recommended, and drug screening images to predict the delta between the day 7 feature vector and day 10 feature vector.

Since the delta prediction model is trained solely on the representations of day 7 untreated images to predict the delta between the day 7 feature vector and day 10 feature vector of untreated images, the inference loss/metric (i.e., the difference between the predicted delta and actual delta) will be very small for untreated images. Conversely, the inference loss/metric will increase for treated images as their representations deviate from those of untreated images. This inference loss/metric will be used as the feature for the ranking/order scale, where the initial images will start with untreated images that have very small inference loss/metric, and the scale will end with images having high inference loss/metric in ascending order. Determining a reasonable inference loss/metric will be one of the research problems to tackle.

6.5 Ranking strategy 2: K means centroid approach

1. **Step 1:** Feed control images into k means and find the centriod of control (untreated) cluster based on both cosine distance and the euclidean distance. (we can choose the distacne metric if we have time)

2. **Step 2:** calculate the euclidean/cosine distance from this centroid to every images.

3. **Step 3:** Perform the same operation for simlcr features and on original images.

6.6 Ranking strategy 3: Softmax approach

1. Train classification model to classify untreated and treated. 2. Then do inference on them and take softmax probability as metric for ranking.

6.7 Intermediate evaluation of SSL model

if we do kmeans withcosine distance distance then it only compare cosine sim of whole image thats why we need to do normal kmeans with euclidean dist to show the magnitude similarity

Evaluation of the SSL model depends on the time series inference loss/accuracy metric, neverthless we can use other evaluation metrics, such as downstream task like classification.

suppose I have a vector of image. if I normalise it unit length, willl I loose critical information? **https://chatgpt.com/share/671a6a63-ca7c-8010-92a0-23b6bd25ef05**

classification

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 two categories, 'untreated' and 'single dose,' 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. We implemented a simple pipeline for a Logistic Regression setup, where the images are encoded into their feature vectors.

2. Baseline comparison: As a baseline for comparison to our results above in the section 6.3, we will train a standard ResNet-18 with random initialization on the labeled training set, consisting of the 'untreated' and 'single dose' categories. The results will help us assess the advantages of contrastive learning on unlabeled data compared to purely supervised training. It is evident that ResNet-18 easily overfits the training data since its parameter count is over 1,000 times larger than the dataset size. To ensure a fair comparison with the contrastive learning models, we apply similar data augmentations as before, including crop-and-resize and color jittering.

kmeans clustering

Idea is whether the learned representation from simlcr outperforms the original images in clustering the images (unsupervised manner) For that we use simple kmeans clusetring. We will cluster them based on both euclidean distance as well as cosine distance.

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:

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

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.

Direct day 7 to day 10 distance evaluation

calculate distance between day 7 untreated and its corresponding day 10 treated one. Idea is: 1. it should give same distance between day 7 and its corresponding day 10 even its changed in position/flipped. ie checking whether simlcr learned to be invariant to position error in microscope error because of manual handling/transporting. 2. it sould give same distance even if its blured/sharpened 3. it should give sma edistance even if its changed in brightness 4. main test for k means centroid approach. it should give different distance to single dose and exploded.

7 Experimental Setup

8 Experimental Results and Discussion

8.1 3 channel vs 1 channel

8.2 Unet vs resnet

8.3 96 vs 256

8.4 bacthsize 16 vs 64 vs 128 vs 256

8.5 Intermediate evaluation

8.5.1 Classification

8.5.2 Clustering

8.5.3 Data augmentation day 7 to day 10 distance evaluation

basic evaluations for ranking: 1. whether it learned to be invariant the position change: do flipps and calculate the cosine distance from control to treated flipped versions. (i don't expect it learn to the change in center of position, if it learns good we can say center crop have some effect maybe? but not for the edge one?) 2. shape invariant:control to all single dose should be almost same cosine distance.

it also applicable to time prediction and reconstruction ranking evaluation and also from kmeans centriod approach.

8.6 Ranking Evaluation

8.6.1 Using CAE

Day 7 to 7 reconstruction

Original images: Unfortunately, the inference loss/metric will ve same beacuse control day 10 and treatd looks same . I was dumb enough to do that. thats why i need day 7 to day 7 reconstruction model.

Day 7 to 10 prediction

Day 10 prediction model

Delta prediction model

8.6.2 K means centroid approach

8.6.3 Softmax approach

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat

a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

A	...	...	...	...	
B	...	...	...	...	
C	...	...	...	...	...
D	...	...	...	...	...
E	...	...	...	...	...
F	-	-	-	$\surd$	$\surd$
G	-	-	-	$l \leq x$	$11 \leq l \leq x$

Table 8.1: Cool table

Appendix

Bibliography

[1] Mathilde Caron et al. *Emerging Properties in Self-Supervised Vision Transformers*. 2021.

[2] Ting Chen et al. *A Simple Framework for Contrastive Learning of Visual Representations*. 2020.

[3] 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).

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

[5] Jean-Bastien Grill et al. *Bootstrap your own latent: A new approach to self-supervised Learning*. 2020.

[6] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015.

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

[8] Prannay Khosla et al. *Supervised Contrastive Learning*. 2021.

[9] Ziyu Liu et al. *Self-Supervised Learning for Time Series: Contrastive or Generative?* 2024.

[10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015.

[11] 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.

[12] Xinyu Yang, Zhenguo Zhang, and Rongyi Cui. “TimeCLR: A self-supervised contrastive learning framework for univariate time series representation”. In: *Knowledge-Based Systems* 245 (2022), p. 108606.

[13] Yuhao Zhang et al. *Contrastive Learning of Medical Visual Representations from Paired Images and Text*. 2022.

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, January 15th 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) stored in the database maintained by PlagScan as well as personal data which are part of this work may be stored there.

Consent is voluntary. Without this consent, the plagiarism check cannot be prevented by removing all personal data and protecting the copyright requirements. Consent to the storage and use of personal data may be revoked at any time by notifying the faculty.

Bibin Babu, January 15th 2025