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 Master of Artificial intelligence

BIBIN BABU

Submitted on: 13.05.2024

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.

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 utilized 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.

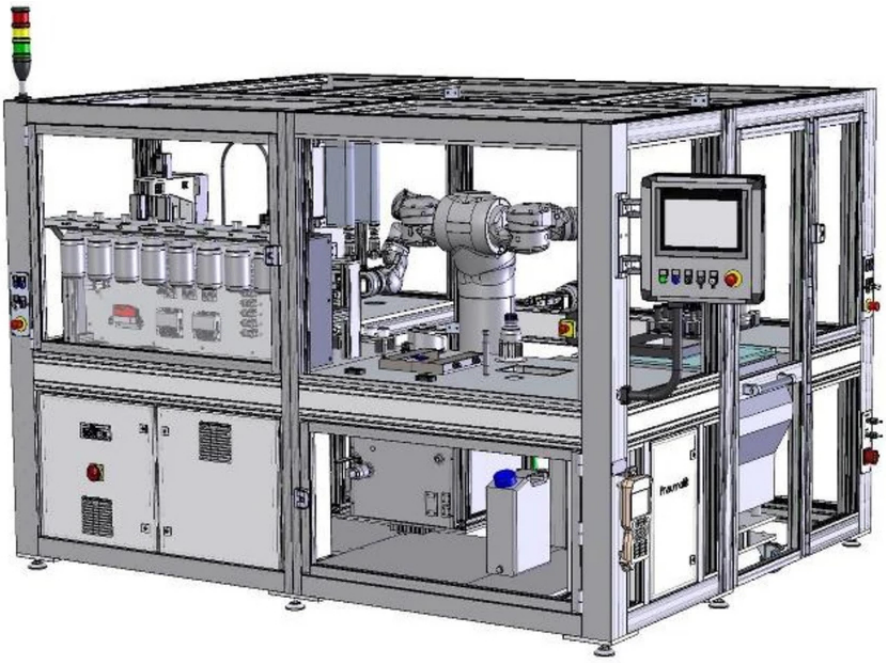

Figure 1.1: Robo platform

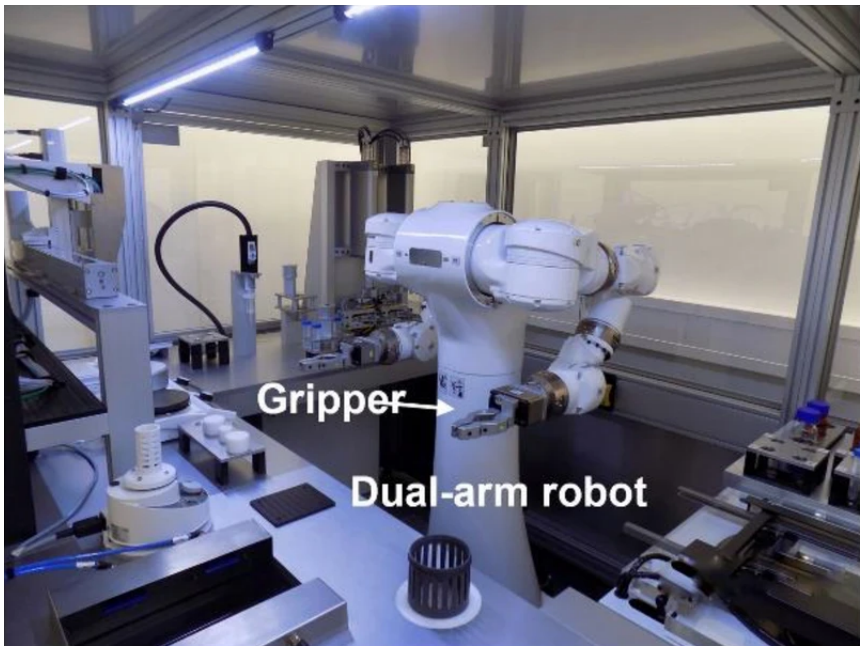

Figure 1.2: Dual-arm robot

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. Despite this limitation, our work serves as a valuable starting point for studying drug efficacy in a controlled environment. Fortunately, if we are able to replicate human body tumor cells in the lab in the future, the techniques currently used to study the 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.

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 were 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.

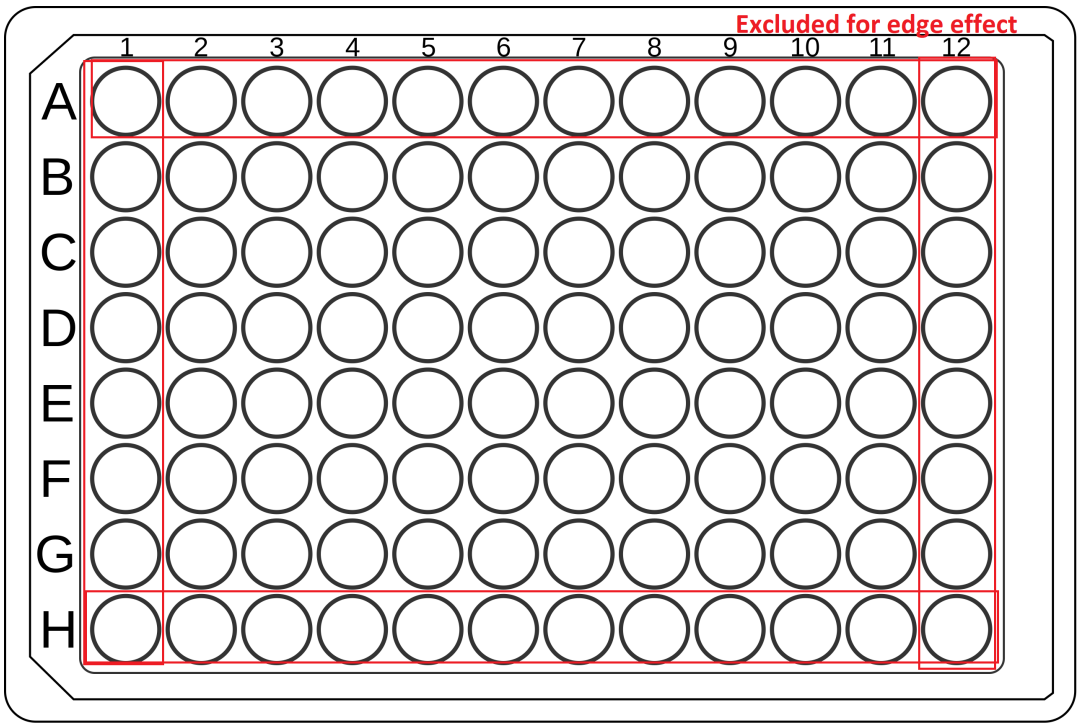

Figure 1.3: A well plate containing 96 wells where rows A, H and columns 1, 2 are excluded due to edge effects.

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

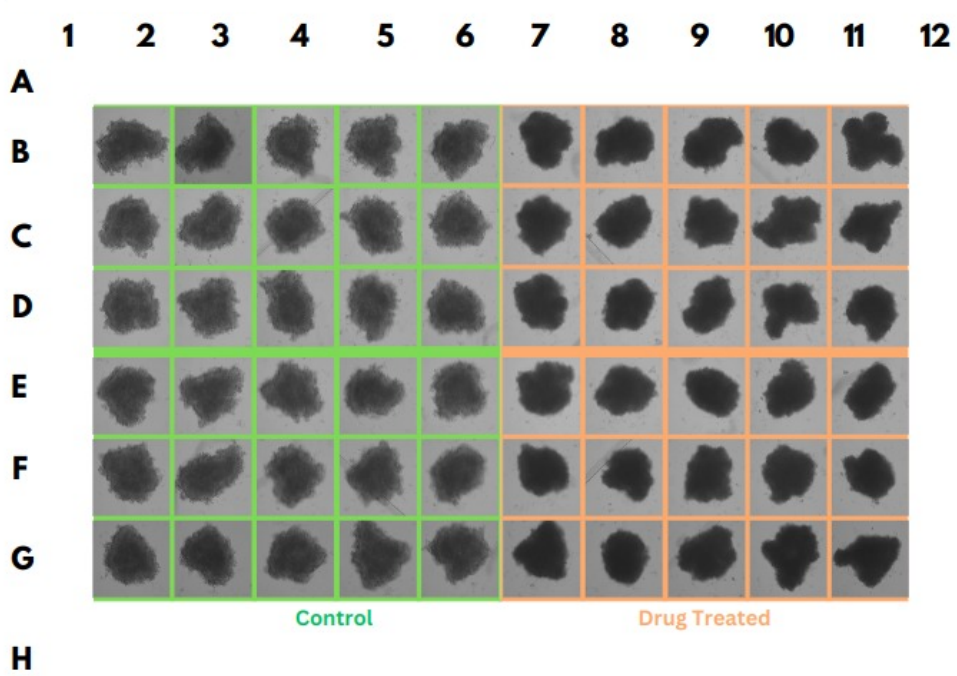

Figure 1.4: Well plate setup for the single-dose experiment where the left half remains untreated and the right half is treated with a single drug concentration. This image was taken three days after drug application, i.e., on day 10.

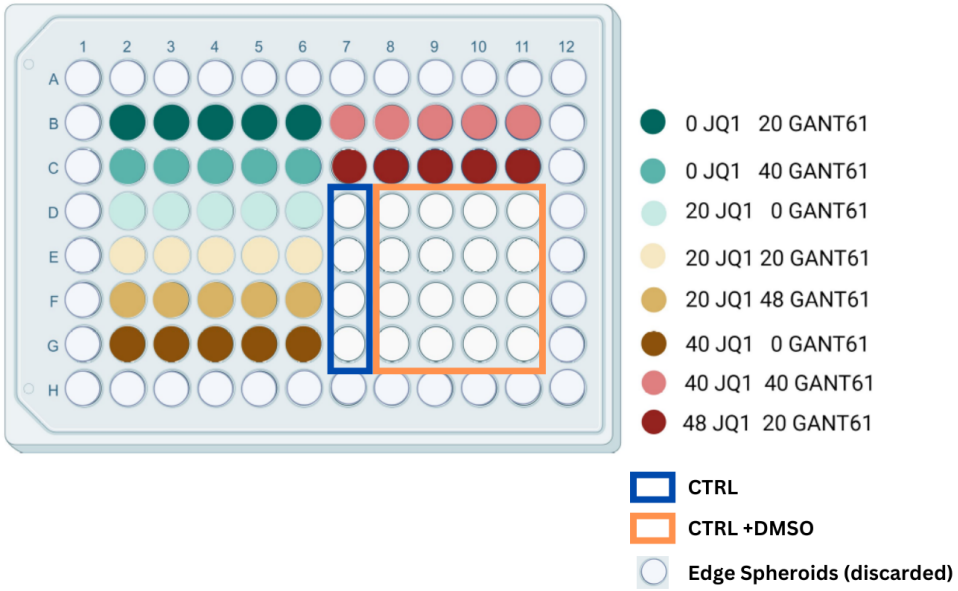

Figure 1.5: Well plate setup for the drug screening experiment where the majority of tumor tissues are treated with different combinations of drug concentrations (multi-colored wells), while some are left untreated (white wells bounded by boxes).

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.

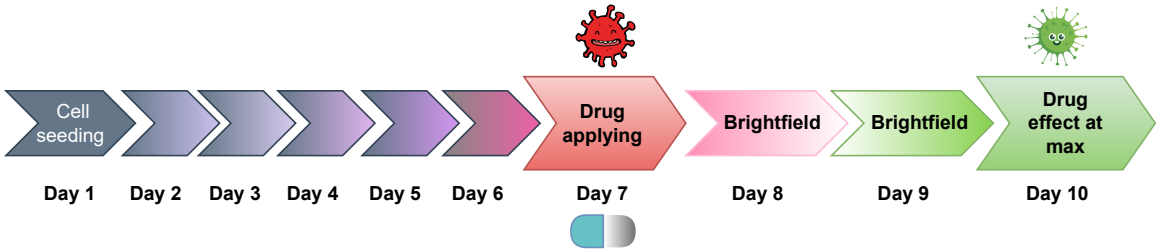

Figure 1.6: Illustrates the flow chart of time evolutoion of 3D tumor tissues.

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 effectively establish a ranking of drug efficacy from bright-field microscopy images, specifically features that capture the alterations induced in three-dimensional tumor tissue models by drug application over a period of time?

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 Related works

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 [10] ; Esteva et al., 2017 [4] ; Irvin et al., 2019 [6]), 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 [12] 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, [11] 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 properties 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.

Coupling engineered features and learned representation. Due to its specificity, fluorescence microscopy has become a quintessential imaging tool in cell biology. However, photobleaching, phototoxicity and related artifacts continue to limit its utility. Recently, it has been shown that artificial intelligence (AI) can transform one form of contrast into another. Mikhail et al.[7] present phase imaging with computational specificity (PICS), a combination of quantitative phase imaging (QPI) and AI, which provides information about unlabeled live cells with high specificity. By applying the computed fluorescence maps back to the QPI data, they measured the growth of both nuclei and cytoplasm independently over many days without loss of viability. This work could provide valuable insights for coupling fluorescent data with learned representations.

5 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.

1. **Step 1:** Create a latent space representation of each image using contrastive learning techniques such as SimCLR, masked autoencoder, or any other self-supervised architectures such as DINO that can effectively help learn the efficient features of alterations induced in three-dimensional tumor tissue models by the impact of drug application over a period of time.

2. **Step 2:** Train a time series prediction model exclusively on the representations of untreated images from Day 7 to Day 10 to predict the representation of the Day 10 image.

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

Since the time series 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.

6 Experiments

6.1 Data set

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. However, for time efficiency, the current data we have collected contains 3 channels per image.

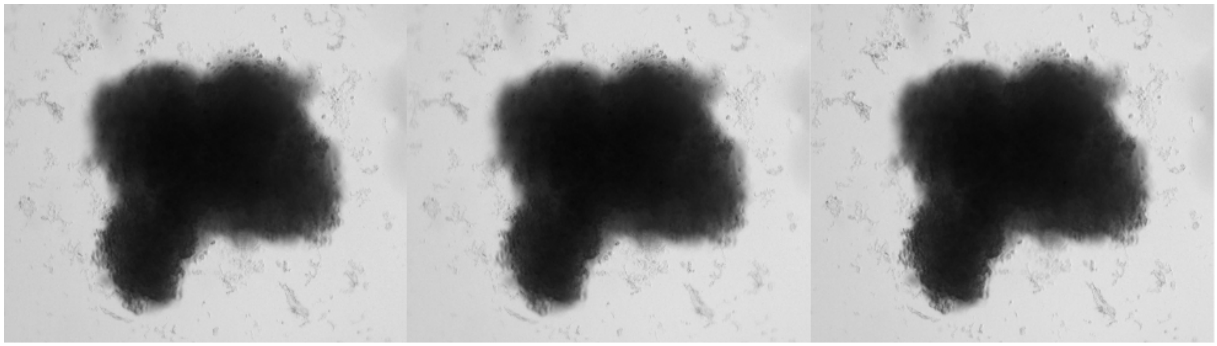

Figure 6.1: Illustration of three layers per image: A, B, and C. The three layers looks visually similar, with slight differences in focal planes. In this figure, A is the sharpest/focused layer.

Figure 6.2 illustrates that, even with the application of the same drug at the same concentration, the morphology of 3D tumor tissues changes differently.

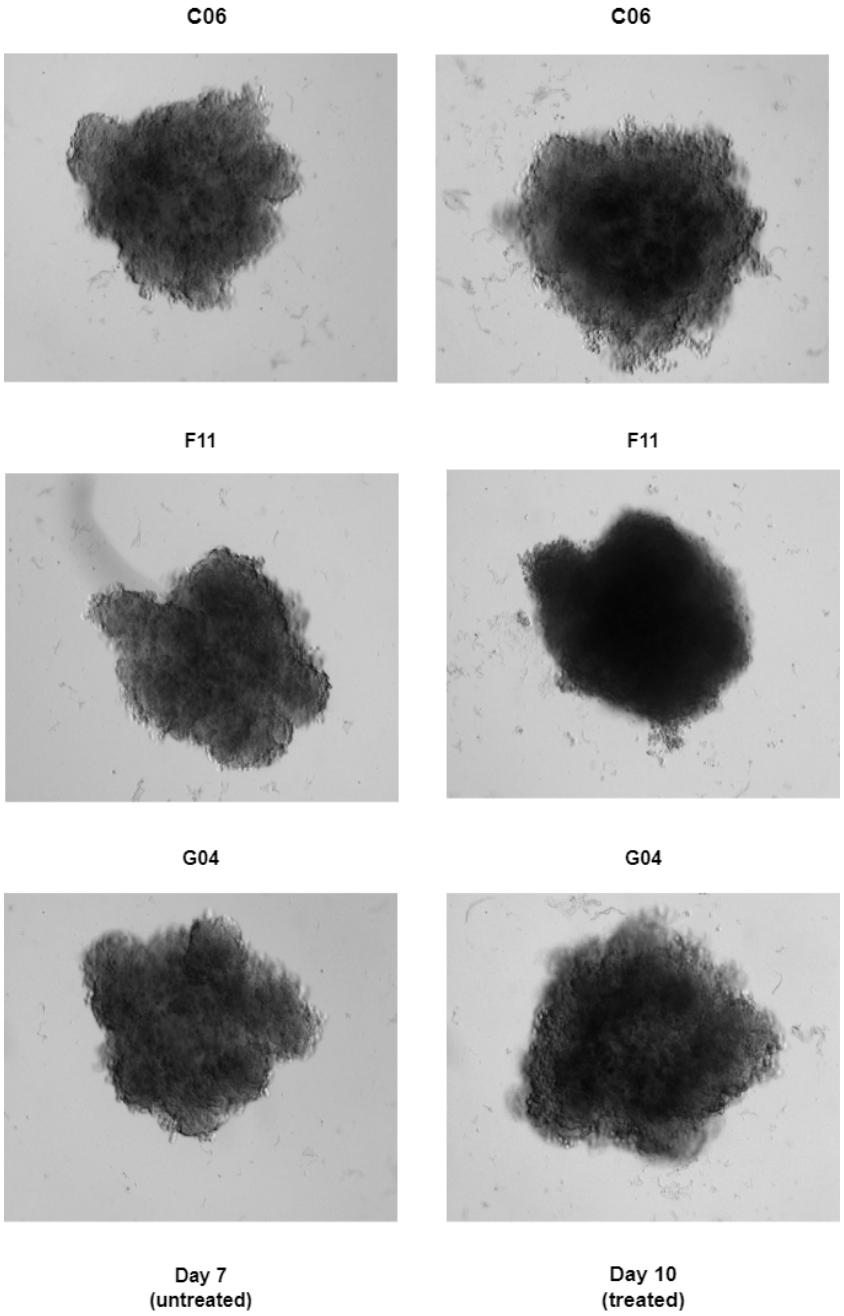

Figure 6.2: C06, F11 and G04 are well names in the well plate.

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 6.1: Dataset Class Overview

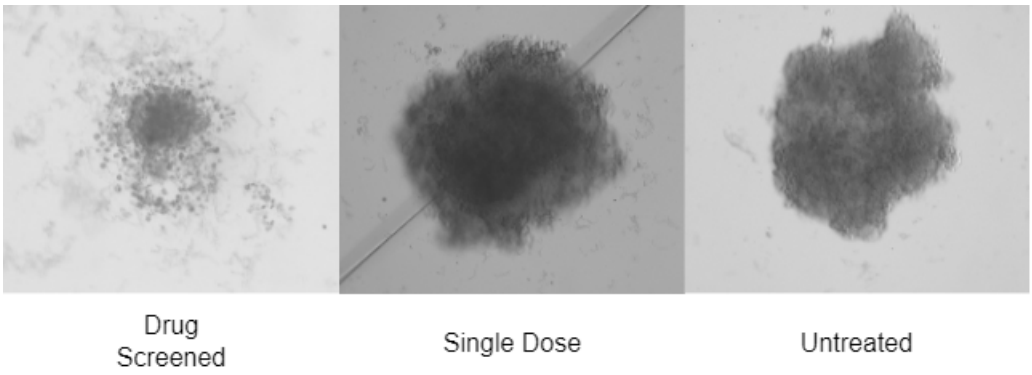

Figure 6.3: Three different types of images: Drug Screened, Single Dose, and Untreated as mentioned in section 1.1.

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. For example, two distinct values in 16-bit (such as 30,000 and 30,001) 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.

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 6.4 and 6.5, whereas for the 16-bit single channel images (extracted one sharp layer from all 3 layer consider it as input to data augemnetation), it was only 49 percentage after 3000 epochs of random color jitter applied as shown in figure 6.8.

Surprisingly 16 bit image but with 3 channels instead of reduction, there was increase in number of unique pixel values maximum by 258757.57 percentage. The problem with the increase is, after data aug, increased pixel values are not distributed similar to the original image instead it shift to two ends for example: 0 or 1 either 0 and 1 which are out of origianl image distribution as shown in figures 6.6 and 6.7.

8 bit three channel image before and after data aug:

Loop 9: Reduction percentage in unique pixel values: 97.81 percentage Original Image - Min: 33, Max: 170 Augmented Image - Min: 0, Max: 2 Number of unique pixel values in the original image: 137 Number of unique pixel values in the augmented image: 3

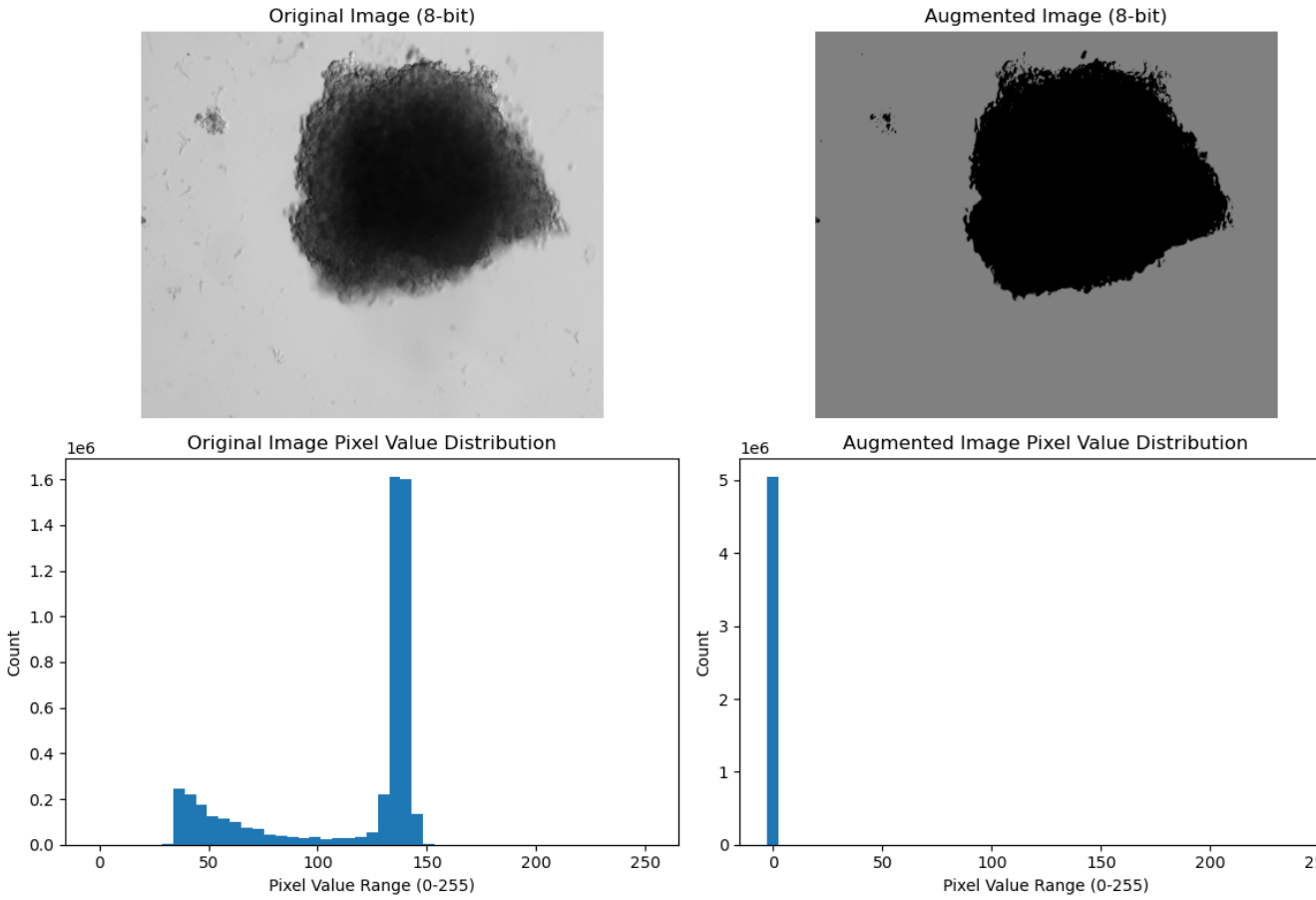

Figure 6.4: eightbit three after 3000 epoch random torch color jitterness apply

8 bit one channel (sharp layer) image before and after data aug:

Loop 26: Reduction percentage in unique pixel values: 99.27 percentage Original Image - Min: 33, Max: 170 Augmented Image - Min: 3, Max: 3 Number of unique pixel values in the original image: 137 Number of unique pixel values in the augmented image: 1

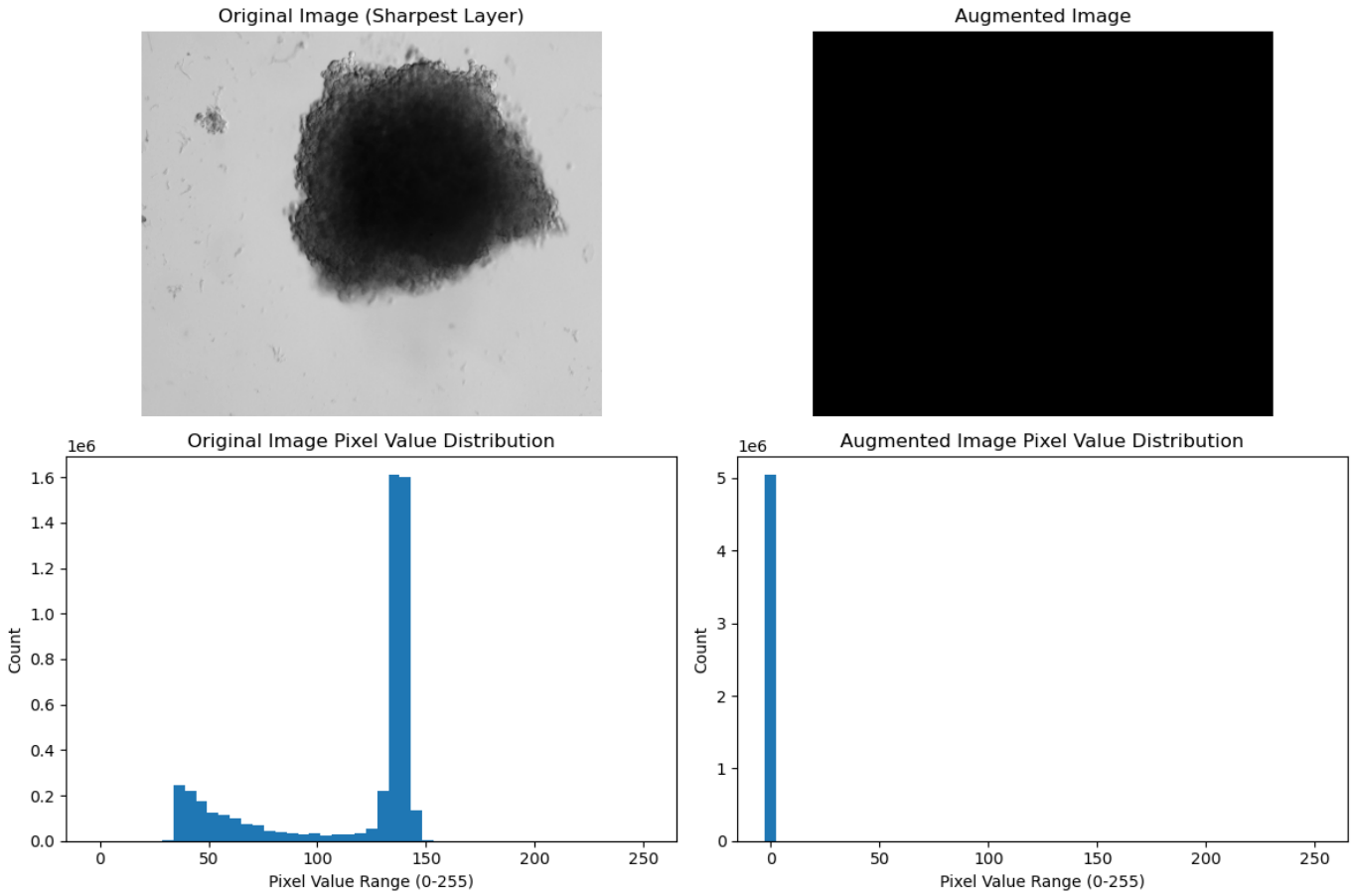

Figure 6.5: eightbit one after 3000 epoch random torch color jitterness apply

16 bit three channel image before and after data aug:

Original Image - Min: 0.13064774870872498, Max: 0.6874189376831055 Augmented Image - Min: 0.022128667682409286, Max: 0.11041323840618134 Number of unique pixel values in the original image: 2111 Number of unique pixel values in the augmented image: 5044624

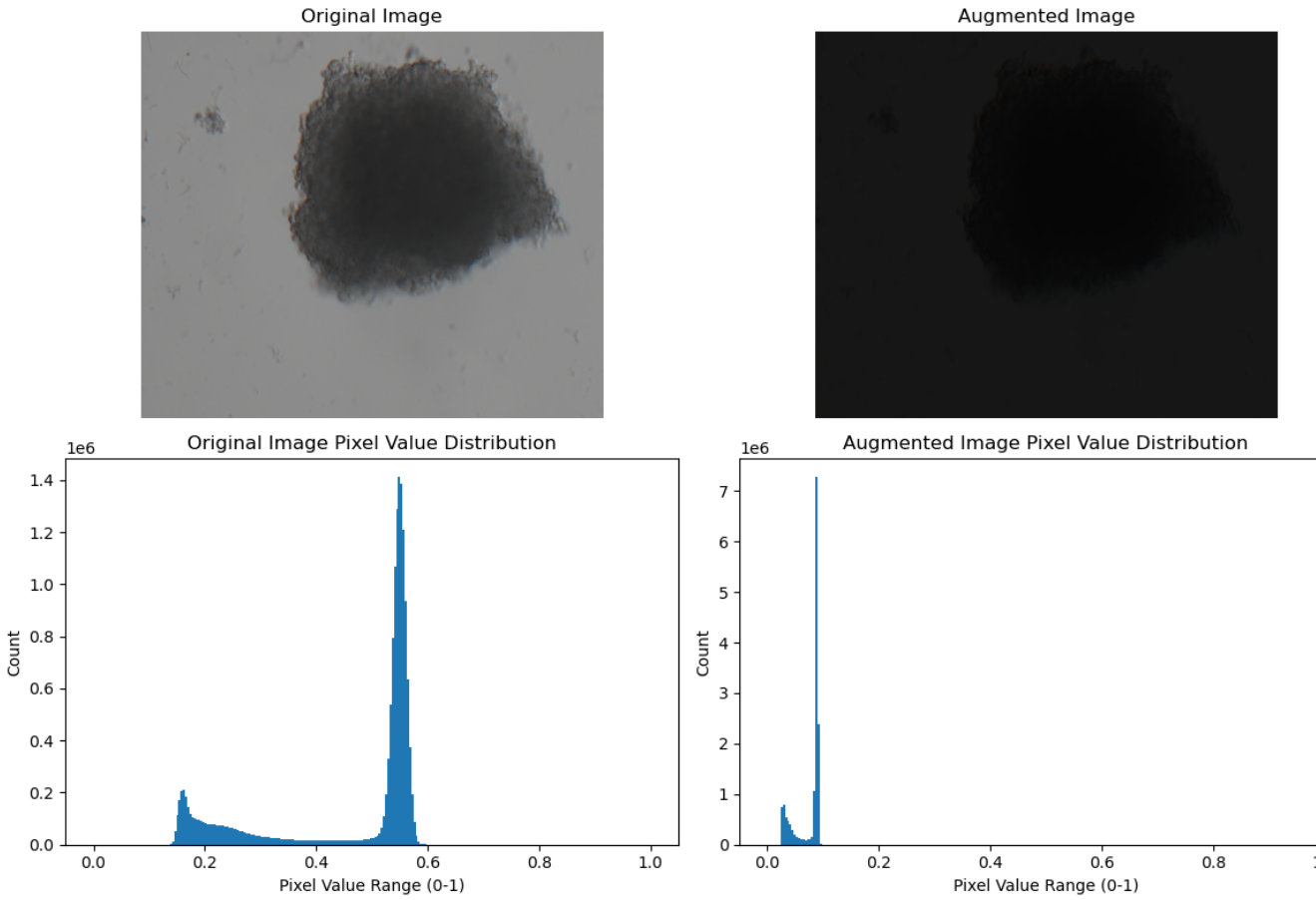

Figure 6.6: sixteen bit three layer after 3000 epoch random torch color jitterness apply

another example for 16 bit three channel image before and after data aug:

Original image shape: torch.Size([3, 2054, 2456]) Augmented image shape: torch.Size([3, 2054, 2456]) Original Image - Min: 0.13064774870872498, Max: 0.6874189376831055 Augmented Image - Min: 0.1969958394765854, Max: 0.3747836649417877 Original Image - Unique pixel counts per channel: Channel 1: 2111 Channel 2: 2100 Channel 3: 2137

Augmented Image - Unique pixel counts per channel: Channel 1: 1672294 Channel 2: 1646188 Channel 3: 1686717

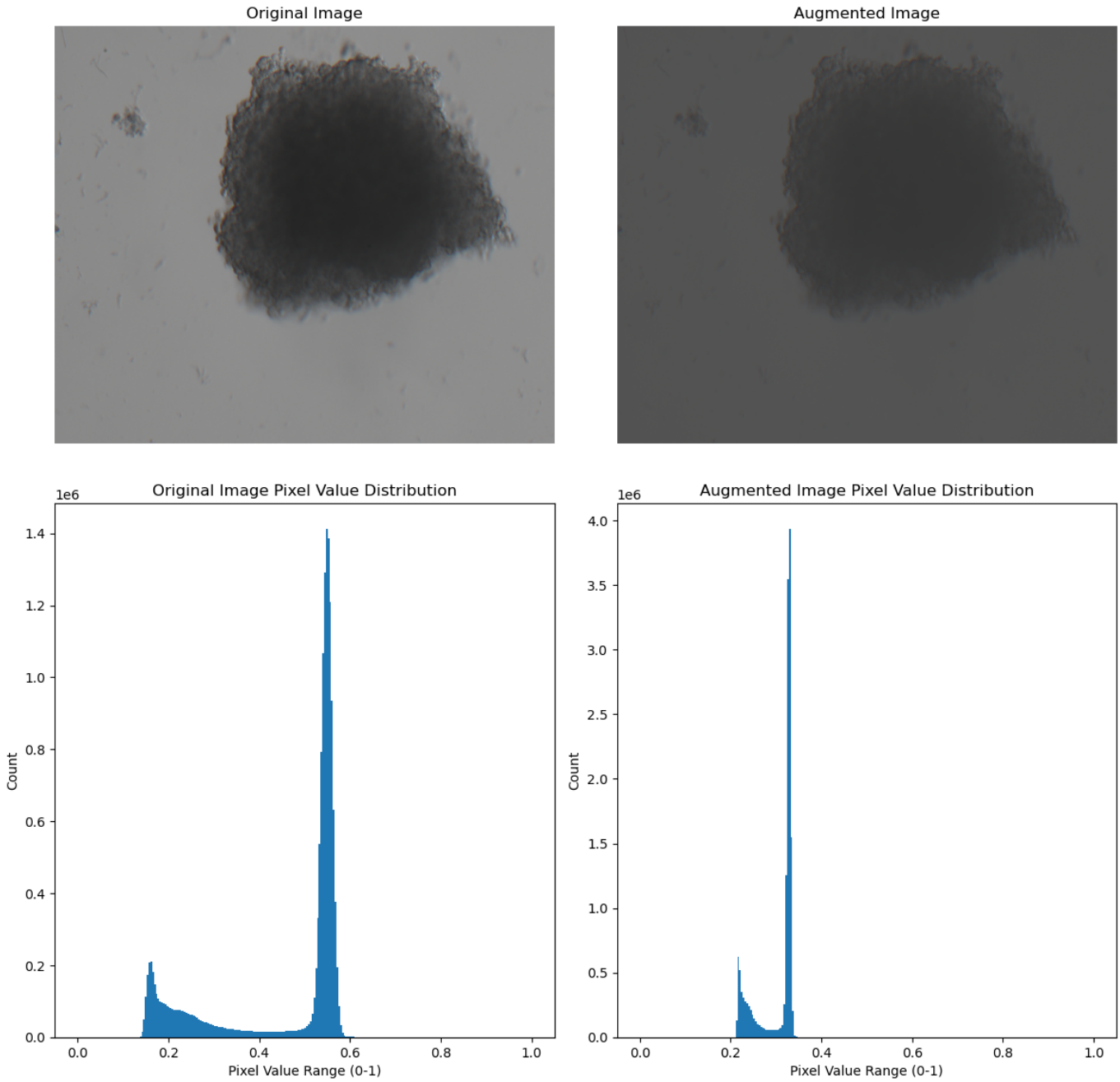

Figure 6.7: sixteen bit three layer after 3000 epoch random torch color jitterness apply

16 bit one channel image before and after data aug:

Loop 125: Reduction percentage in unique pixel values: 49.88 percentage Original Image - Min: 0.13064774870872498, Max: 0.6666666865348816 Augmented Image - Min: 0.0, Max: 1.0 Number of unique pixel values in the original image: 2111 Number of unique pixel values in the augmented image: 1058

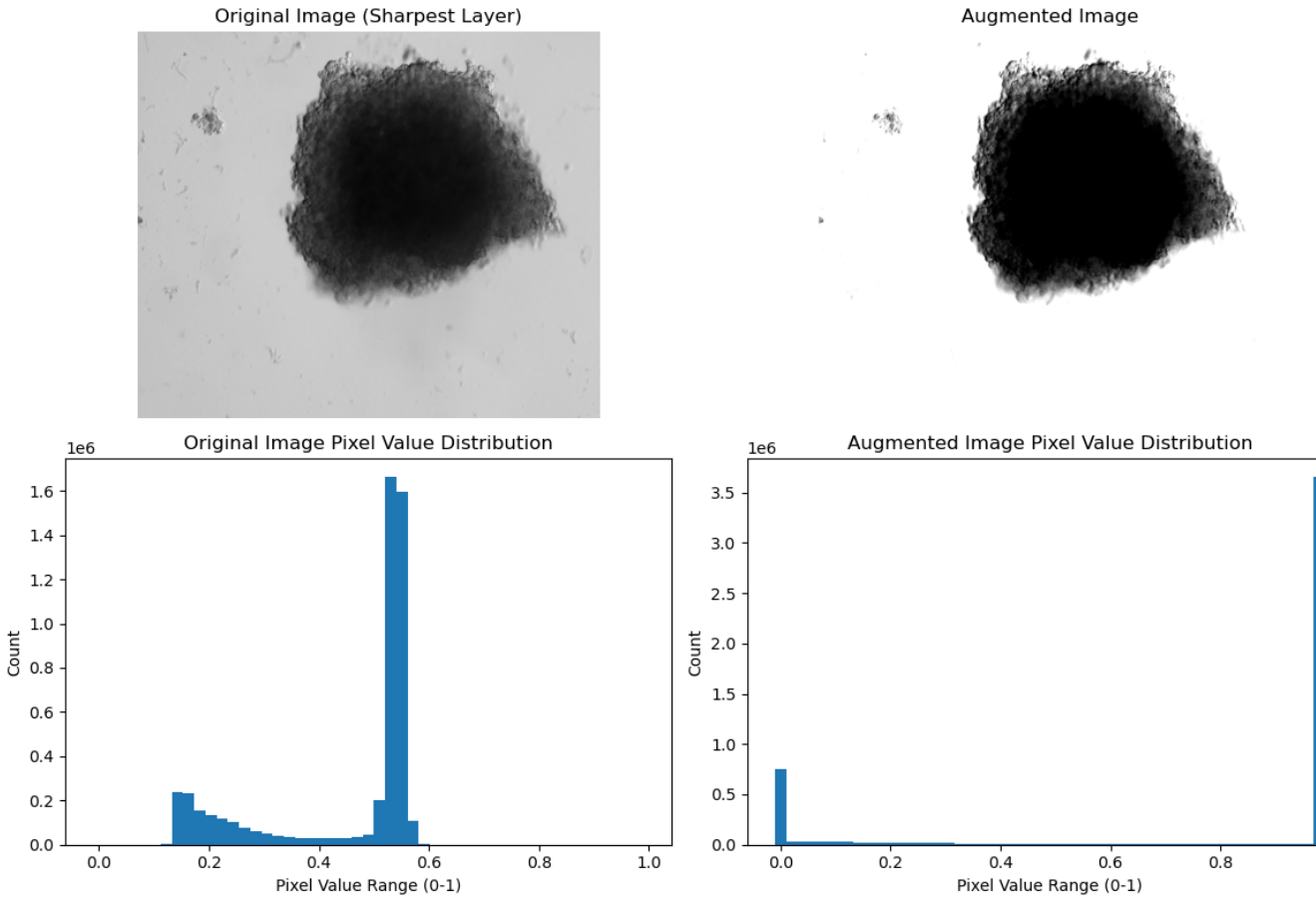

Figure 6.8: sixteen bit one layer after 3000 epoch random torch color jitterness apply

49 percentage max reduction for 16 bit one channel data aug with color jitterness is also not so good in the sense its diminish the gradual spreadness of dark matter as in the above image. so the solution will be instead of using random we experiment with specific parameters inside the color jiterness tranform such a way that it won't exceed lets say 30 percentage of reduction in unique no of pixel values. or another solution will be code own python function or try tensorflow transformation depend on time.

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. At the moment since we are trying to create the full pipe line first we used standard data augmentations from SimCLR paper.

1. Normalize the 16-bit image to [0, 1] (so that torch.transform.color jitterness didn't work without scaled data)

2. Do the following augmentations:

1. Do a horizontal flip.

2. Crop the image randomly, And resize it to 96×96.

3. Randomly change the brightness, contyrast, saturation and hue of the croped patch.

3. For each original image we do the step 2 twice to get two augmentated images.

Model

The model takes one image and produce a latent representation of the input. The aim of the model is to cluster similar images together in higher dimension.

Training

The training follows like this: We take a batch of images with batch size B.

Our dataset class returns two augmented versions for each original image in the batch, resulting in $2B$ images as input.

The model produces $2B$ latent representations, independently for each augmentated image.

For each batch, the two augmentations of the same image are treated as positive pairs, while all others are considered negative samples.

We calculate the cosine similarities between positive pairs and betwee negative pairs. Where Positive pairs are the image augs of same image and negative pairs will be the image augs of different image. Which will give as input to the below loss fn.

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

Zi,Zj are positive pairs and Zi,Zk will be negative pairs.

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

6.2 Training SSL model

For step 1, we used SimCLR as SSL (Self supervised leraning) first model. Later will try other models (Masked auto encoder, Dino) depend on 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.

Ours is not classification problem, so some treated with some specific dosage will have the same effect as untreated one or drug screened one, that's possible. For example, a single dose and a drug screened one may have the same effect, same pixel intensity. Therefore, our goal is not to push or pull because of the label name but because of its natural similarity or dissimilarity. At the moment, SimCLR is kind of forcing similar images to be invariant. Is there any way to do natural pulling and pushing? E.g., Siamese network? Considering the weight effect Prof. Magdas, we need to find a way to use weight but without tweaking too much.

6.2.1 Training SimCLR

$$X = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \cdots & \mathbf{x}^{(m)} \end{bmatrix} \quad \text{where} \quad X \in \mathbb{R}^{n_x \times m}$$

$$x_1 = \begin{bmatrix} 1 \\ 0.8 \\ \vdots \\ \vdots \\ 0.1 \\ 0.9 \\ \vdots \\ 0.0 \end{bmatrix}$$

Suppose we have m training examples $\left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}\right\}$, where $\mathbf{x}^{(i)} \in \mathbb{R}^{n_x}$ for $i = 1, 2, \ldots, m$.

$$X = \begin{bmatrix} x^{(1)} & x^{(2)} & \ldots & x^{(m)} \end{bmatrix}$$

Whats the correct data preprocessing order: resize, dataasug , normalisation

Question: How can you say simclr is not pulling similar images of different dog breed together naturally? We can understand from the loss in a batch, lets say bulldog breed pulled together in the latent space and in another batch german sheperd pulled together

in the latent space. We understood that from the mathematics. but does that mean these 2 different dog breeds are together in the latent space at the end of training? (situation specific to our dataset will be: some specific single dose image and some specific drug screened image can also have sililarity like they belong to same class eventhough they have diffrent drug concentration applied) In that case how does that happened the 2 different dog breed pulling

Ans: Because of the data augmentation. Data augmentation helps to generalise the features/characterics of different breed but in same class images while training. For example, suppose we have images of a black cat and a pink cat in our dataset. Through color augmentations (e.g., changing the hue, brightness, etc.), an augmented version of the pink cat might look visually closer to a black cat. Although the contrastive loss function doesn't explicitly pull the black and pink cats together, the model learns to associate these images due to their shared visual characteristics after augmentation. As a result, the representations of the black and pink cat may end up closer in the latent space because the model has learned invariances to augmentations such as color changes, textures, or lighting conditions.

Why do we need to do Data augmentation?

It force to have invariance (to get same feature vector regardless its transformation which helps to have a latent space with positive samples together or more aligned. Also it helps to deal with invariance to microscope differrent sunlight exposure or environment factor or focal plane so it aligns with the interest of biological reason.

The choice of the data augmentation to use is the most crucial hyperparameter in SimCLR since it directly affects how the latent space is structured, and what patterns might be learned from the data.

Two augmentations stand out for imagenet classification in their importance: crop-and-resize, and color distortion. Interestingly, however, they only lead to strong performance if they have been used together as discussed by Ting Chen et al. in their SimCLR paper. When performing randomly cropping and resizing, we can distinguish between two situations: (a) cropped image A provides a local view of cropped image B, or (b) cropped images C and D show neighboring views of the same image While situation (a) requires the model to learn some sort of scale invariance to make crops A and B similar in latent space, situation (b) is more challenging since the model needs to recognize an object beyond its limited view. However, without color distortion, there is a loophole that the model can exploit, namely that different crops of the same image usually look very similar in color space. Consider the picture of the dog above. Simply from the color of the fur and the green color tone of the background, you can reason that two patches belong to the same image without actually recognizing the dog in the picture. In this case, the model might end up focusing only on the color histograms of the images, and ignore other more generalizable features. If, however, we distort the colors in the

two patches randomly and independently of each other, the model cannot rely on this simple feature anymore. Hence, by combining random cropping and color distortions, the model can only match two patches by learning generalizable representations.

Lesson: Data augmentation should be beneficial such a way that model can't rely/exploit on common/unwanted pattern to distinguish between 2 images?

Question: Why don't we cutout/remove white well plate background? careful for drug screening because incase of drug screened images its debris are spread around so we need to be careful when cutting out.

Question: Pretrained resnet is on 8 bit, is there any concept like then the custom image should be also 8 bit to match it? or need to normlaise the same way that resnet pretrained normalise? what else factors may effect like this other than normalisation?

☐ Find pretrained architectures for medical gray scale images/ Unet better for medical images? (Unet is better for segmanetation - comment from MG)

Question: What happens data augmentation randomly change after each epoch? My answer: then learning won't happen properly because each epoch gradient calculation will be different to each epochs where we can't take average. Wrong: gradient calculation works independently so there will be no problem with gradient calc. What about loss? because loss is averaging for the bacth.

Question: Does Ntxent loss denominator act as pushing dissimilar images apart? No. Denominator only act as normalisor to scale the probability (output of softmax) so that the total probability distribution will sum upto 1.

Question: Why don't we calculate the loss of divergence of similar image to dissimilar image?(At the moment ntxent loss is only calculating loss of divergence of similar images(positive sample)) Because we don't know how much dissimilar they are or if its dissimilar at all. For example: Suppose we have 2 different breed dogs which will be in same class. positive samples will be bulldog aug1 and bulldog aug2. negative sample contains german sheperd aug1 and sheperd aug2. we don't want to push away the positive sample and negative sample here, because eventhough they are in 2 different breed they belongs to same class. likewise in our dataset, some specific single dose image and some specific drug screened image can also have sililarity like they belong to same class eventhough they have diffrent drug concentration applied. hence we don't want to push them away by including this loss. Thats why current ntxent loss is better than Binary cross entroppy loss (BCE) where BCE considers loss of divergence in probability distribution of ground truth dissimlar image to predicted dissimilar image probability. Question:Is there alternative for BCE where it doesn't include loss of divergence of

dissimilar to dissimilar? one idea is use just modified just CE and give ground truth labels equal probability like if we have (1,0,1,0,1,1,1) then probability matching should be (20,0,20,0,20,20,20) and this should match like 0.2log(predicted prob)

Question: Whats model collapse, and in which situation it happens? Mg comment: Model collapse meaning differ depend on the model or context. so understand first whats generally model collapse and then read original paper to find out whats the specific model collapse problem we have with simclr.

variation ideas:

1. Each image considered as rgb since we have 3 channel + 2 std augs

2. 1 channel considered as anchor(most sharped layer) others as 2 anchors.

3. Each image considered as rgb since we have 3 channel + more than 2 std augs (research for medical gray scale images)

4. Include anchor as positive , ie 3 augs toal (1 anchor as aug other 2 layers as augs)

5. Remove positive sample j from denominator of loss fn. Since j is the only one image as positive sample in the sum of denominator softmax its contribution will be less.

6. Supervised Simclr: Make sure no same breed/class images in the negative samples. Does this increase the confidence score for 1's in the cross entropy loss? or does this suppress confidence score for dissimilar one forcly? because in our case we get similar images from 2 different class

7. Try another loss fn like Triplet loss this already shown in the orig paper.

8. Use Unet architecture

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.

(need edit from here) story should start from here: In both cases, as per the SimCLR architecture, the augmented versions are then fed into a pretrained ResNet18 model.

Variation 1:

Input to model (train loader dimensions) :

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): [32, 512] (Batch size, standard resnet18 output dimension after avg pooling) 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) 2*Batch size is due to the concatenation of aug1 images and aug2 images for the efficient/smart way to calculate the loss. No of values in feature vector is a variable which we can change and experiment which will give better accuracy. This output feed to loss function to train model to reduce the loss there by increase the cosine similarity between positive samples in the batch.

Below images after data augmentation and normalisation of 3 channel version:

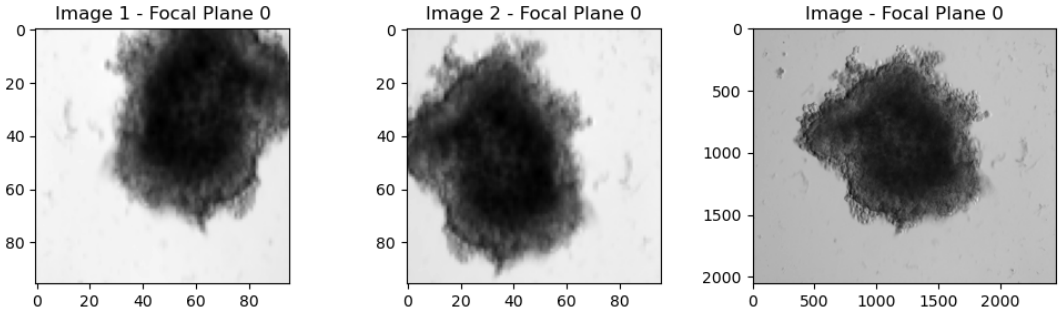

Figure 6.9: op1

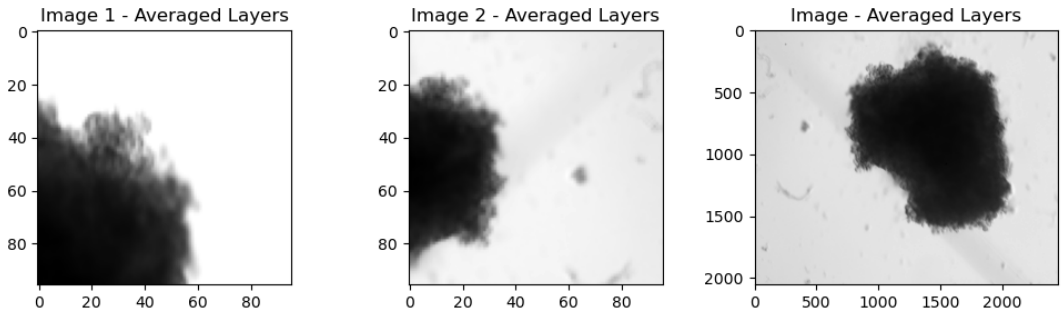

Figure 6.10: op3

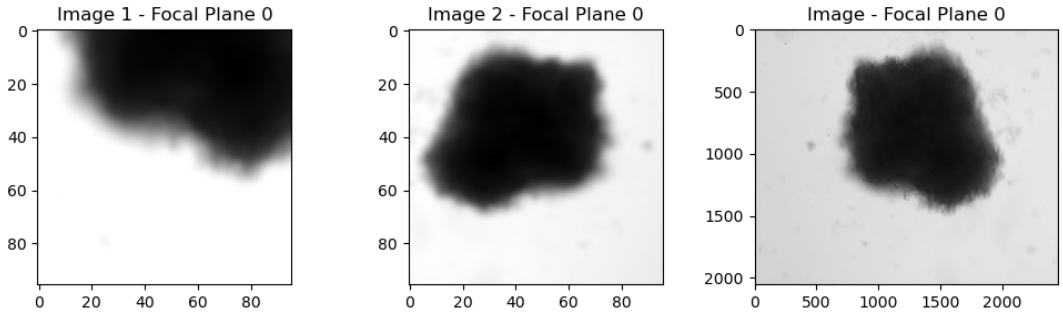

Figure 6.11: Blured augs

Results: Task of the model traiining is to reduce the loss thereby increase the cosine similarity of similar/positives images in the batch.

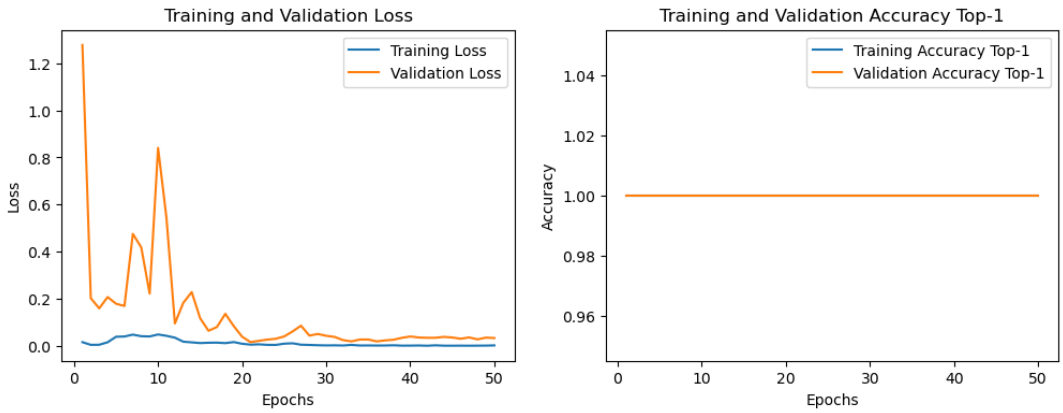

Figure 6.12: batchsize: 16

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): [32, 512] (Batch size, standard resnet18 output dimension after avg pooling) 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) 2*Batch size is due to the concatenation of aug1 images and aug2 images for the efficient/smart way to calculate the loss. No of values in feature vector is a variable which we can change and experiment which will give better accuracy. This output feed to loss function to train model to reduce the loss there by increase the cosine similarity between positive samples in the batch. Below images after data augmentation and normalisation of 1 channel version:

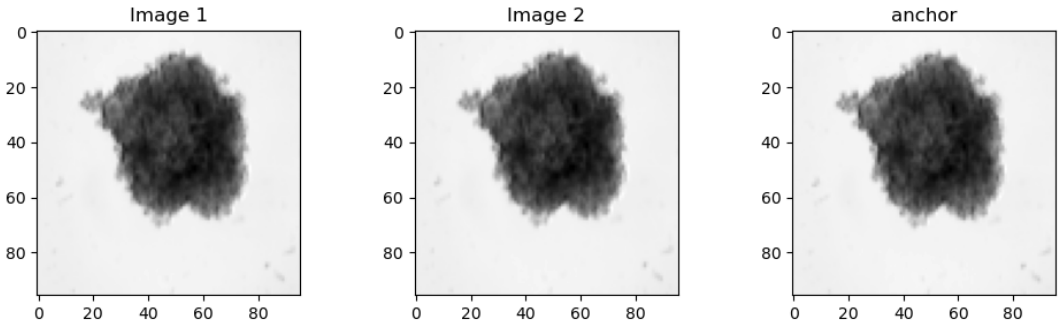

Figure 6.13: 1dop1

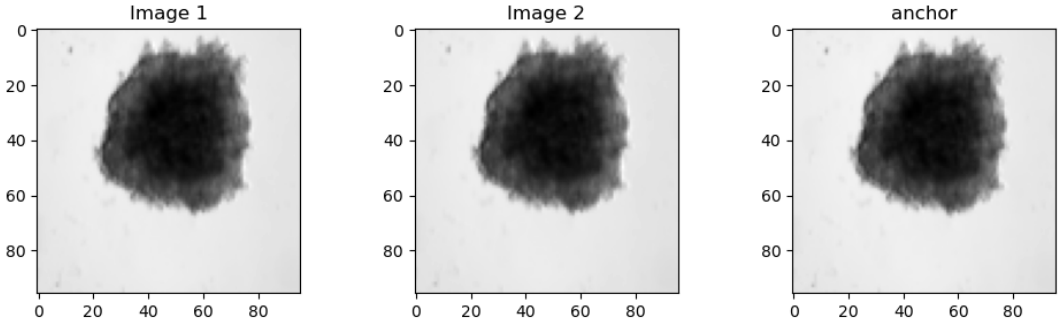

Figure 6.14: 1dop2

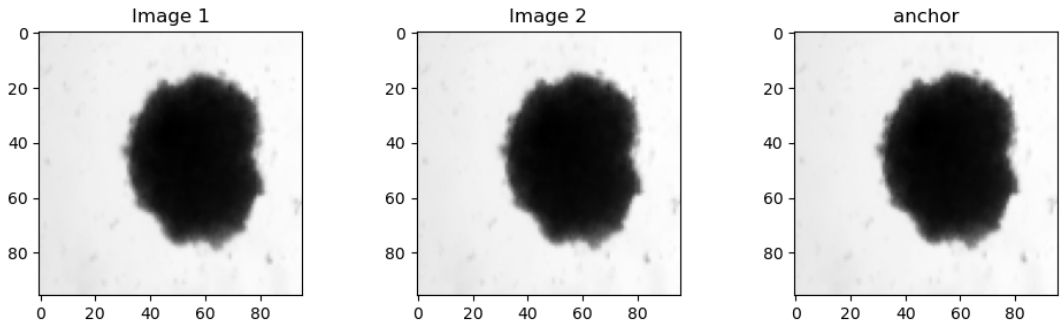

Figure 6.15: 1dop3

6.3 Intermediate evaluation of SSL model

Evaluation of the SSL model depend on the time series inference loss/metric, neverthless we can use other evaluation metrics such as downstream task like fine tuning classification loss trained upon the SSL. A common setup, which also verifies whether the model has learned generalized representations, is to perform Logistic Regression on the features. In other words, we learn a single, linear layer that maps the representations to a class prediction, where untreated, single dose, drug screened these 3 categories will be our classes. Since the base network is not changed during the training process, the model can only perform well if the representations of describe all features that might be necessary for the task. Further, we do not have to worry too much about overfitting since we have very few parameters that are trained. Hence, we might expect that the model can perform well even with very little data. We implemented a simple pipeline for a Logistic Regression setup where the images have been encoded in their feature vectors.

Note from SiCLR tutorial: If very little data is available, it might be beneficial to dynamically encode the images during training so that we can also apply data augmentations. However, the way we implement it here is much more efficient and can be trained within a few seconds. Further, using data augmentations did not show any significant gain in this simple setup.

Question: in the tutorial simclr trained on different distribution and logistic regression trained on different distribution, should we do the same? Does it matter?

$$\text{cos\_sim} = \begin{bmatrix} -9e15 & 0.2 & 0.3 & 0.4 & 0.9 & 0.5 & 0.1 & 0.6 \\ 0.2 & -9e15 & 0.1 & 0.4 & 0.5 & 0.3 & 0.9 & 0.2 \\ 0.3 & 0.1 & -9e15 & 0.7 & 0.6 & 0.2 & 0.1 & 0.8 \\ 0.4 & 0.4 & 0.7 & -9e15 & 0.9 & 0.5 & 0.2 & 0.1 \\ 0.9 & 0.5 & 0.6 & 0.9 & -9e15 & 0.7 & 0.3 & 0.2 \\ 0.5 & 0.3 & 0.2 & 0.5 & 0.7 & -9e15 & 0.6 & 0.4 \\ 0.1 & 0.9 & 0.1 & 0.2 & 0.3 & 0.6 & -9e15 & 0.5 \\ 0.6 & 0.2 & 0.8 & 0.1 & 0.2 & 0.4 & 0.5 & -9e15 \end{bmatrix}$$

$$\text{pos\_mask} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\text{comb\_sim}[0] = \begin{bmatrix} 0.9, & 0.2, & 0.3, & 0.4, & -9e15, & 0.5, & 0.1, & 0.6 \end{bmatrix}$$

$$\text{comb\_sim} = \begin{bmatrix} 0.9 & 0.2 & 0.3 & 0.4 & -9e15 & 0.5 & 0.1 & 0.6 \\ 0.5 & 0.3 & 0.1 & 0.4 & -9e15 & 0.5 & 0.9 & 0.2 \\ 0.6 & 0.1 & 0.7 & 0.4 & -9e15 & 0.3 & 0.2 & 0.8 \\ 0.9 & 0.4 & 0.7 & 0.8 & -9e15 & 0.6 & 0.1 & 0.4 \\ 0.7 & 0.5 & 0.9 & 0.1 & -9e15 & 0.6 & 0.2 & 0.4 \end{bmatrix}$$

6.3.1 Further variations

Further variations that we can try:

1. train a Logistic Regression model for datasets with only 10, 20, 30, and all examples per class. This gives us an intuition on how well the representations learned by contrastive learning can be transfered to a image recognition task like this classification.

□ think about another model other than to evaluate the trained simclr (example siamese netwrok) maybe not with classification problem.

2. As a baseline to our results above, we can train a standard ResNet-18 with random

initialization on the labeled training set. The results will give us an indication of the advantages that contrastive learning on unlabeled data has compared to using only supervised training.

Question: Does this 2 variations really helps to evaluate SSL model in our case?

6.3.2 Baseline comparison

As a baseline to our results above, we will train a standard ResNet-18 with random initialization on the labeled training set of untreated, single dose,drug screened. The results will give us an indication of the advantages that contrastive learning on unlabeled data has compared to using only supervised training.

It is clear that the ResNet easily overfits on the training data since its parameter count is more than 1000 times larger than the dataset size. To make the comparison to the contrastive learning models fair, we apply data augmentations similar to the ones we used before: horizontal flip, crop-and-resize, grayscale, and gaussian blur.

Note: in MG tutorial, Color distortions as before are not used because the color distribution of an image showed to be an important feature for the classification. Hence, in mg tutorial they observed no noticeable performance gains when adding color distortions to the set of augmentations.

Question: Does this color distortion matters in our gray scale image, need to look what exactly color distortion do to the cell grains (we like to add color distortion for the background changes but it should not effect the cell grains color?

Similarly, we restrict the scale is restricted to between 80 percentage and 100 percentage of the original image size. This is because, for classification, the model needs to recognize the full object, while in contrastive learning, we only want to check whether two patches belong to the same image/object. contrast transforms might lead to more aggressive cropping, possibly capturing smaller parts of the image, which could increase the difficulty for a model to learn specific features. Hence, the chosen augmentations below are overall weaker than in the contrastive learning case. The training function for the ResNet is almost identical to the Logistic Regression setup. Note that we allow the ResNet to perform validation every 2 epochs to also check whether the model overfits strongly in the first iterations or not.

6.4 Time series prediction model

6.5 Integrated/ensembled model: SSL+Time series model

7 To Do

☐ Check untreated images in treated image folder (Data preperation) check for mismatch

✓ Checked box:

8 Conclusion

These methodological steps collectively form the framework for addressing critical research questions and challenges throughout the course of this thesis. By exploring the dataset, assessing models, and developing custom architecture, we aim to to assess drug efficacy by ranking different drug combinations and concentrations.

9 Proposed timeline

Figure 9.1: Proposed Master Thesis timeline

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 Lear 2020.

[6] Jeremy Irvin et al. CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels a 2019.

[7] Mikhail E. Kandel et al. "Phase imaging with computational specificity (PICS) for measuring dry mass changes in sub-cellular compartments". In: Nature Communications 11.1 (Dec. 2020).

[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] 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 2017, pp. 3462–3471.

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

[12] Yuhao Zhang et al. Contrastive Learning of Medical Visual Representations from Paired Im 2022.

BIBIN BABU, 13.05.2024