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.

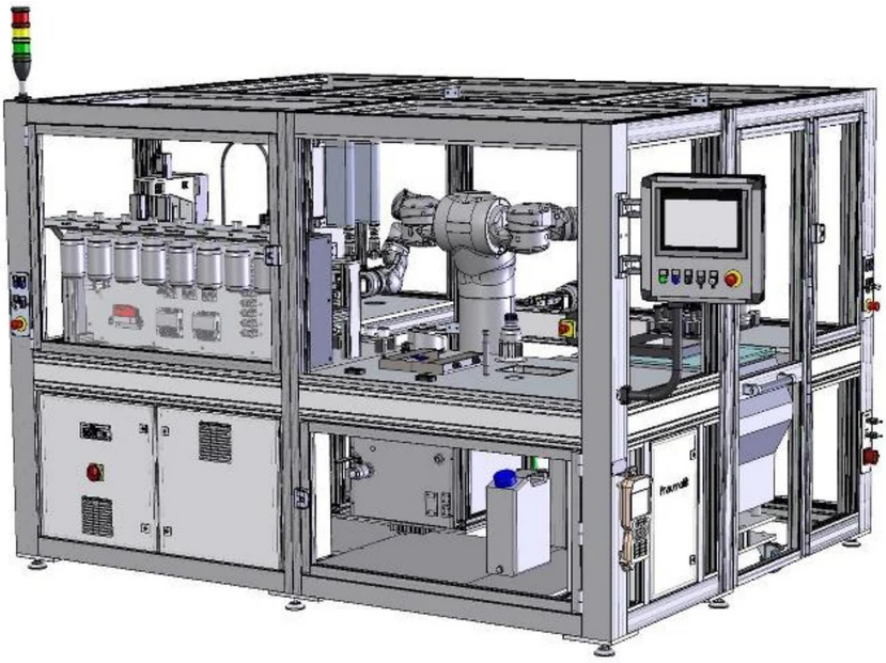

Figure 1.1: Robo platform

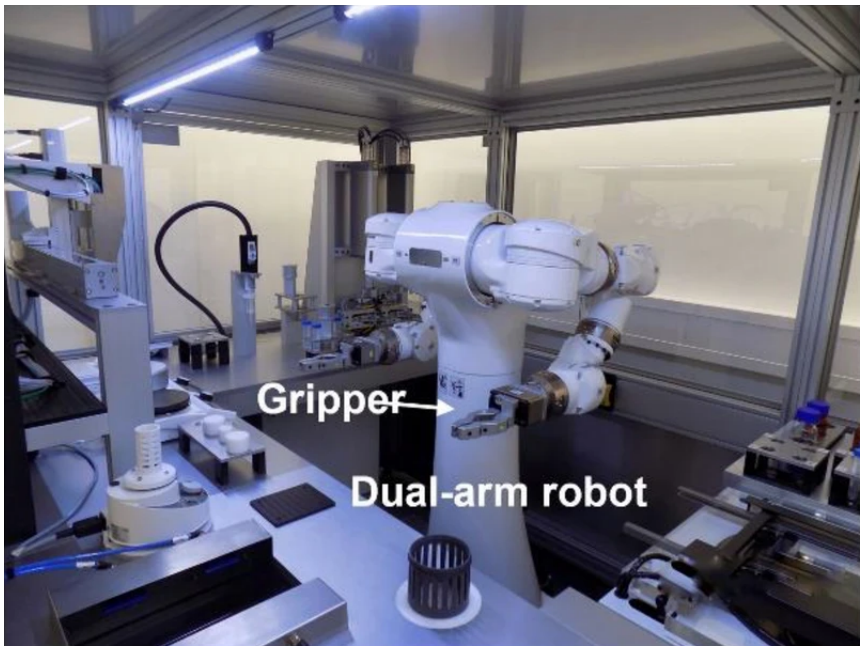

Figure 1.2: Dual-arm robot

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.

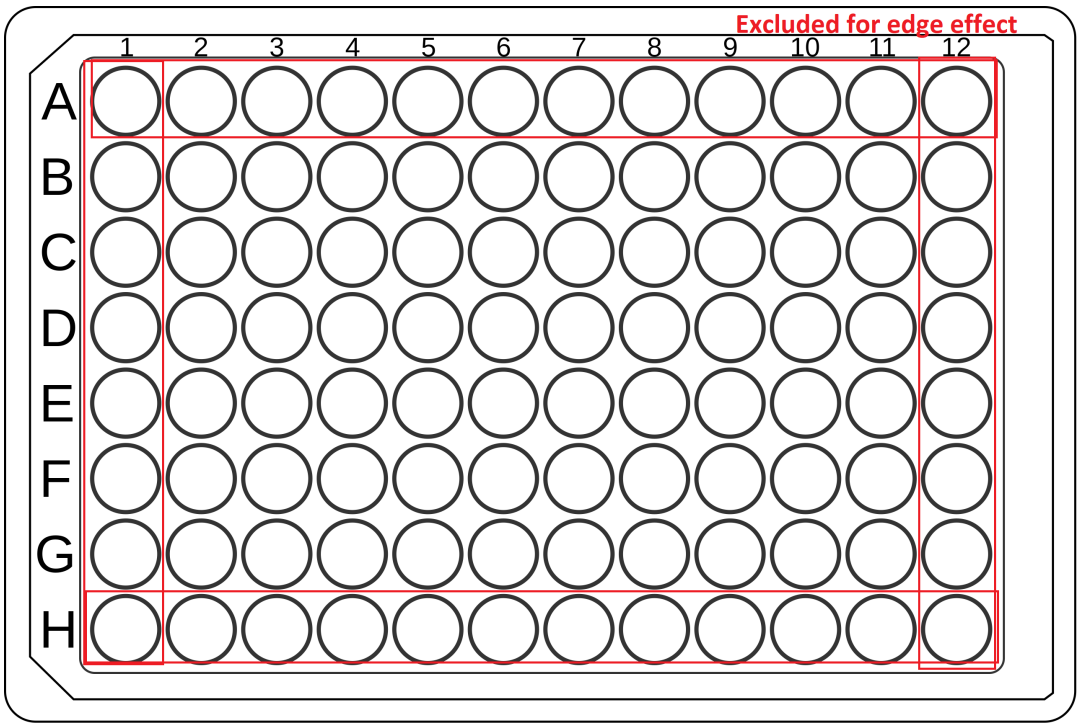

Figure 1.3: A well plate containing 96 wells where rows A, H and columns 1, 2 are excluded due to edge effects.

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.

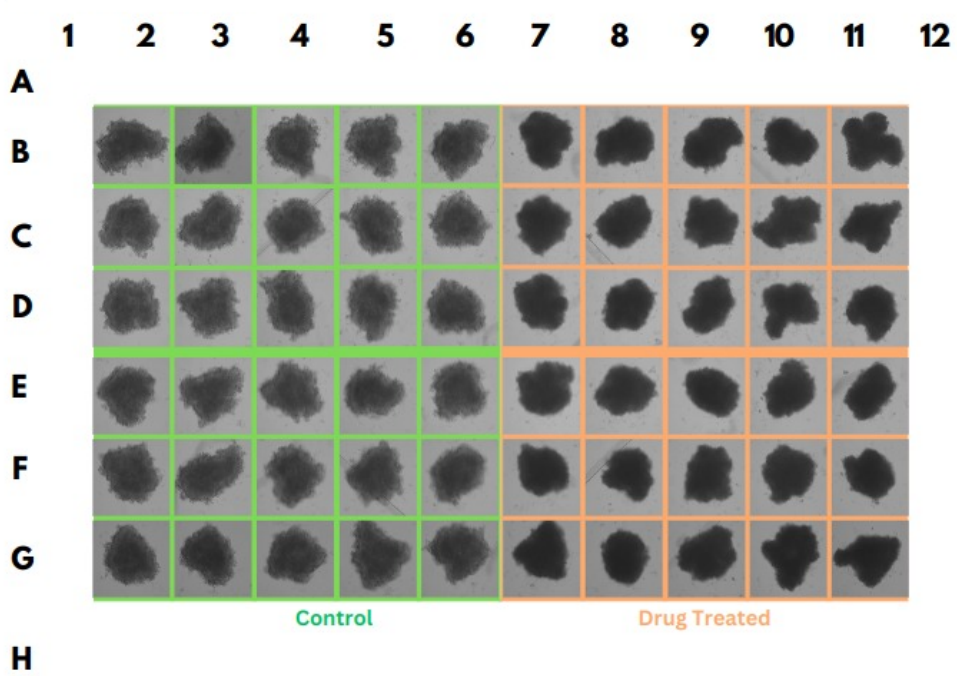

Figure 1.4: Well plate setup for the single-dose experiment where the left half remains untreated and the right half is treated with a single drug concentration. This image was taken three days after drug application, i.e., on day 10.

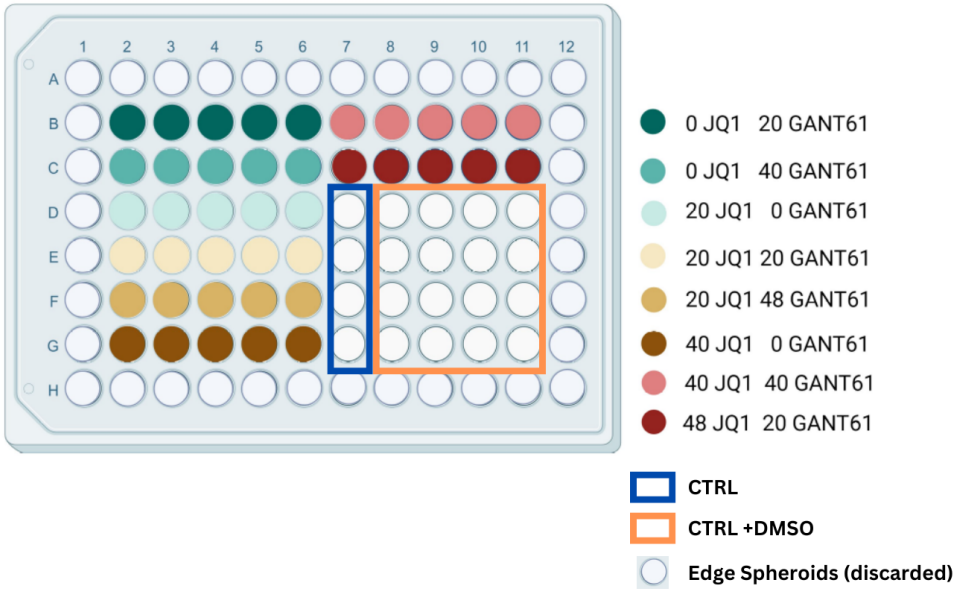

Figure 1.5: Well plate setup for the drug screening experiment where the majority of tumor tissues are treated with different combinations of drug concentrations (multi-colored wells), while some are left untreated (white wells bounded by boxes).

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.

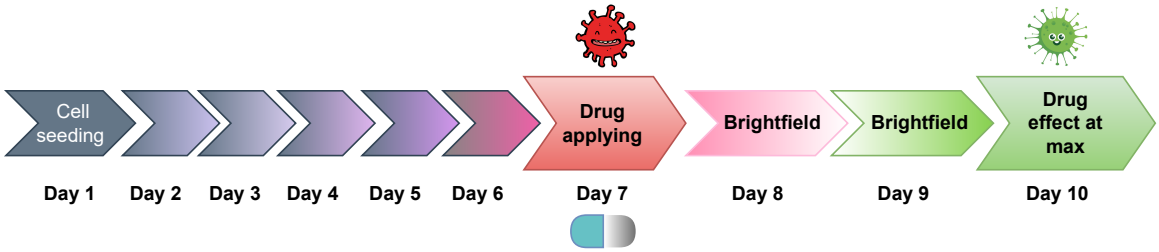

Figure 1.6: Illustrates the flow chart of time evolutoion of 3D tumor tissues.

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 [12] ; 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 [14] 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, [13] 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. [10] 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 [9] 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.[8] 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.

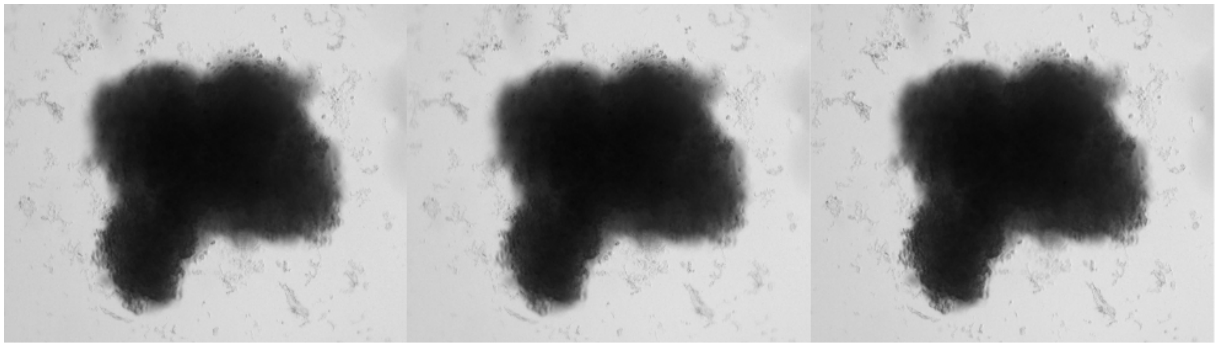

Figure 6.1: Illustration of three layers per image: A, B, and C. The three layers looks visually similar, with slight differences in focal planes. In this figure, A is the sharpest/focused layer.

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.

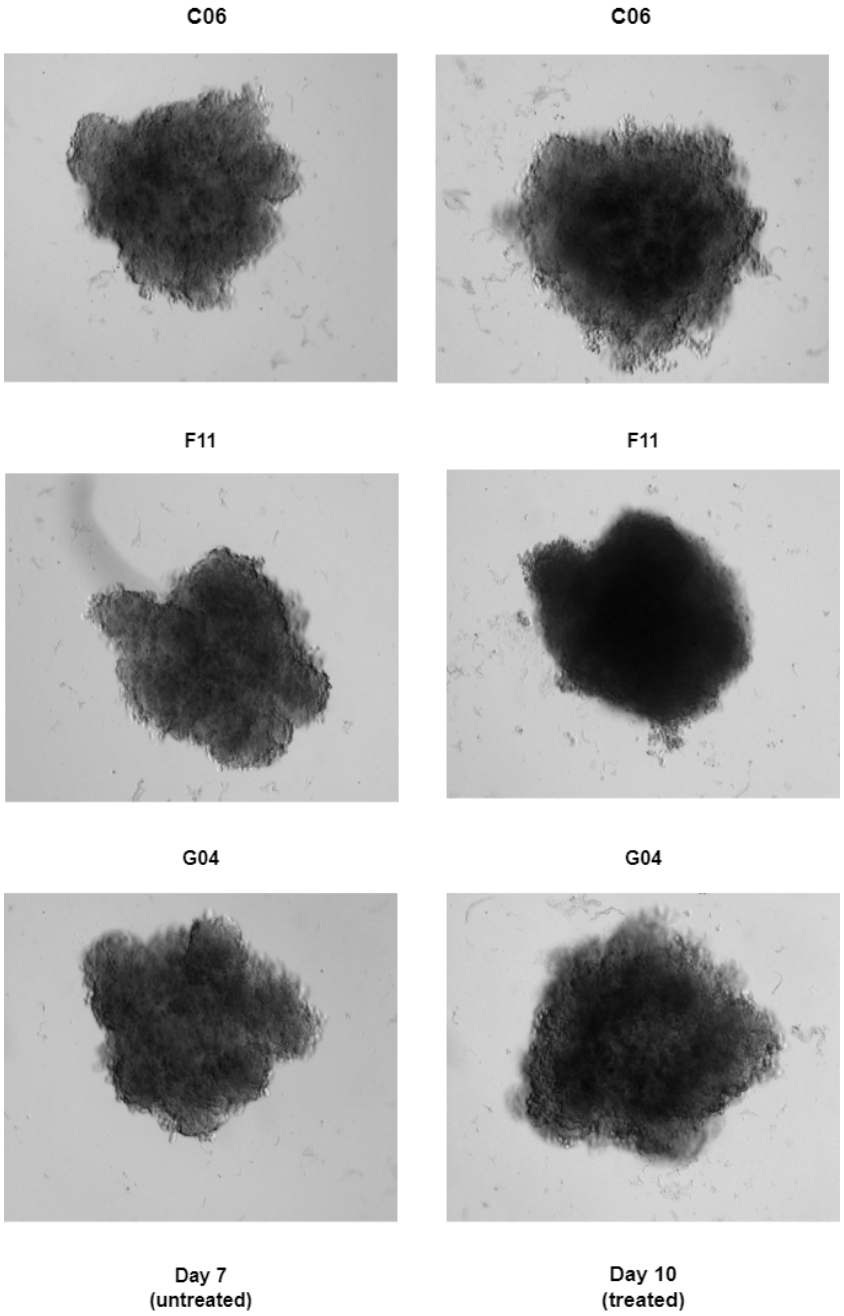

Figure 6.2: C06, F11 and G04 are well names in the well plate.

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

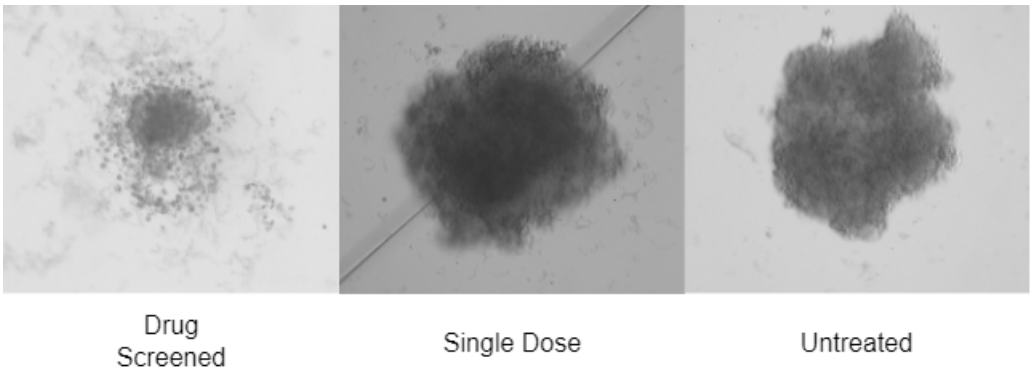

Figure 6.3: Three different types of images: Drug Screened, Single Dose, and Untreated as mentioned in section 1.1.

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 (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 6.9.

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 6.7 and 6.8.

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

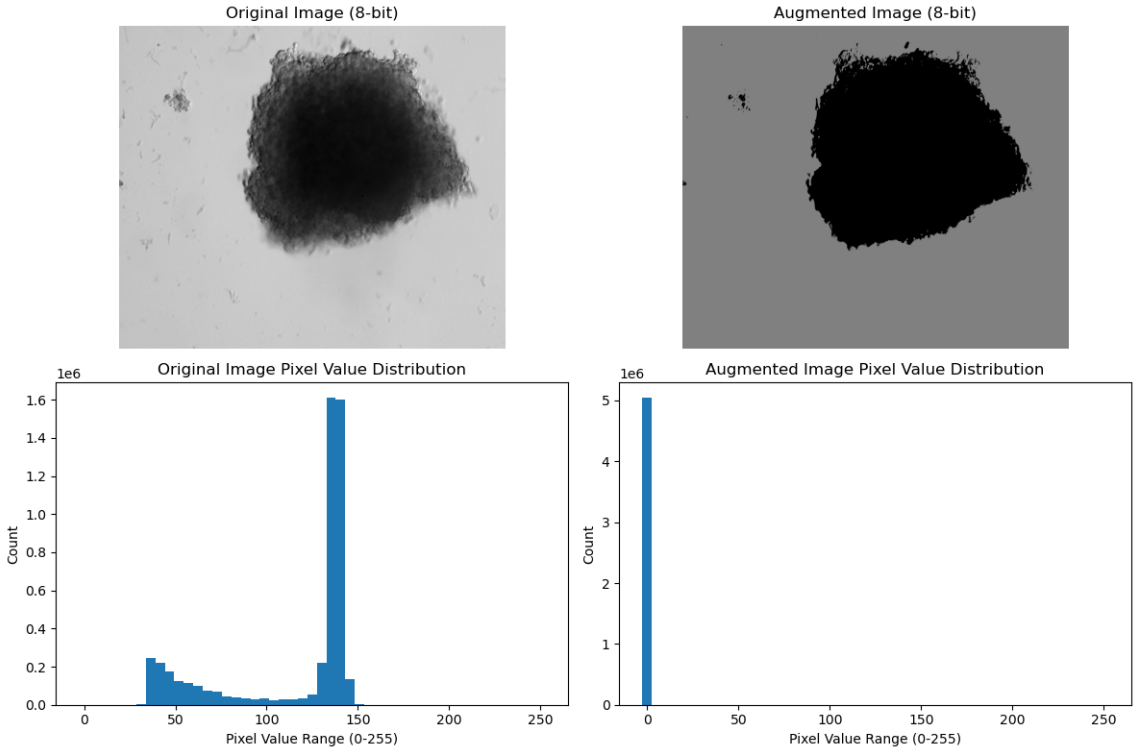

Figure 6.4: 8-bit three-channel image after 3000 epochs of random color jitter applied using PyTorch. Reduction percentage in unique pixel values: 97.81%

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

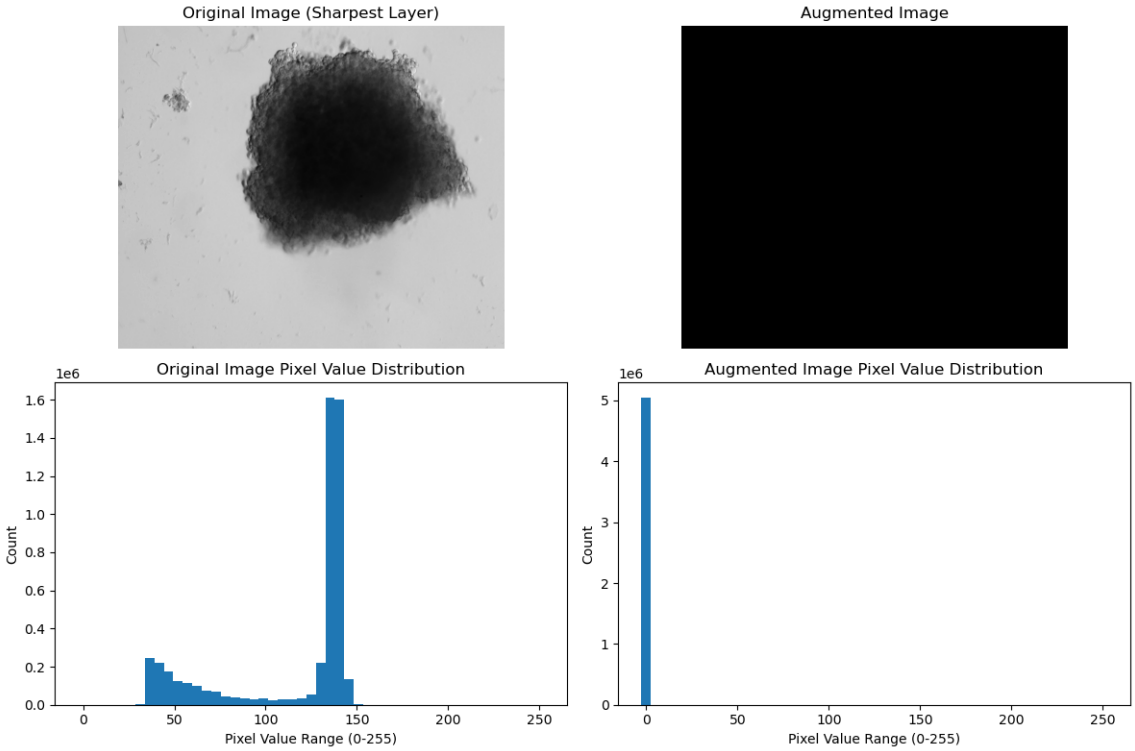

Figure 6.5: 8-bit single-channel image after 3000 epochs of random color jitter applied using PyTorch. Reduction percentage in unique pixel values: 99.27%

16-bit 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

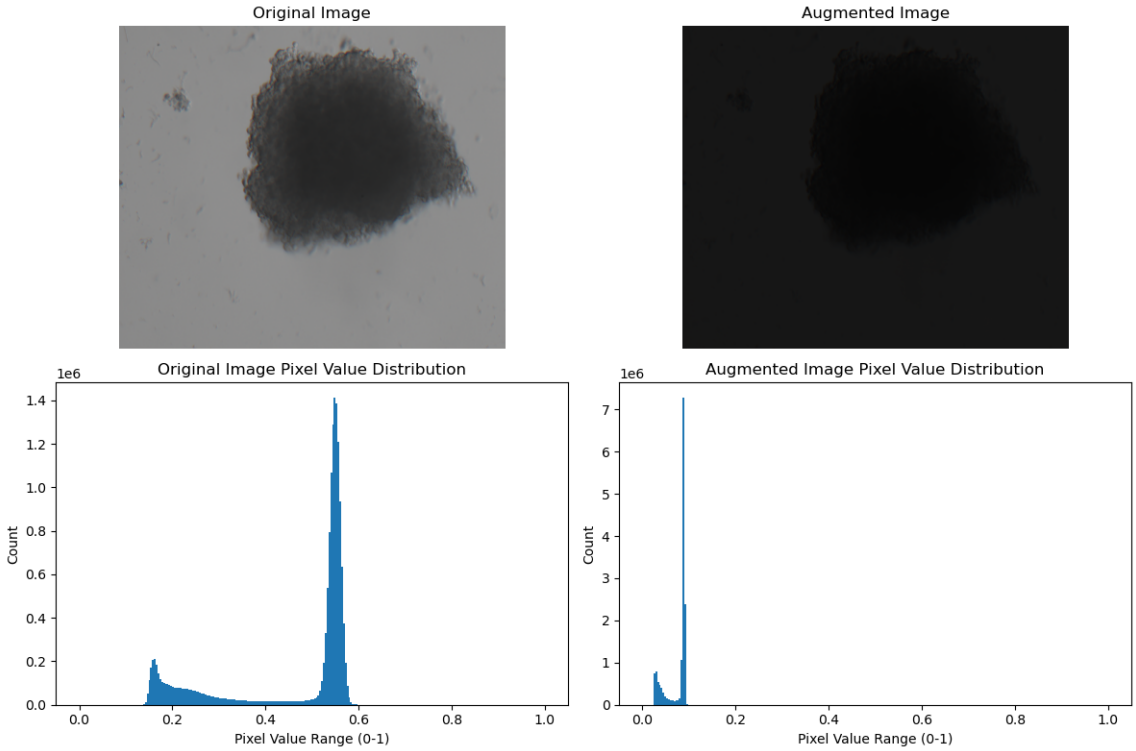

Figure 6.6: sixteen bit three layer after 3000 epoch random torch color jitterness apply

Figure 6.7: 16-bit three-channel image after 3000 epochs of random color jitter applied using PyTorch. Increase in percentage of unique pixel values: 2388%

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

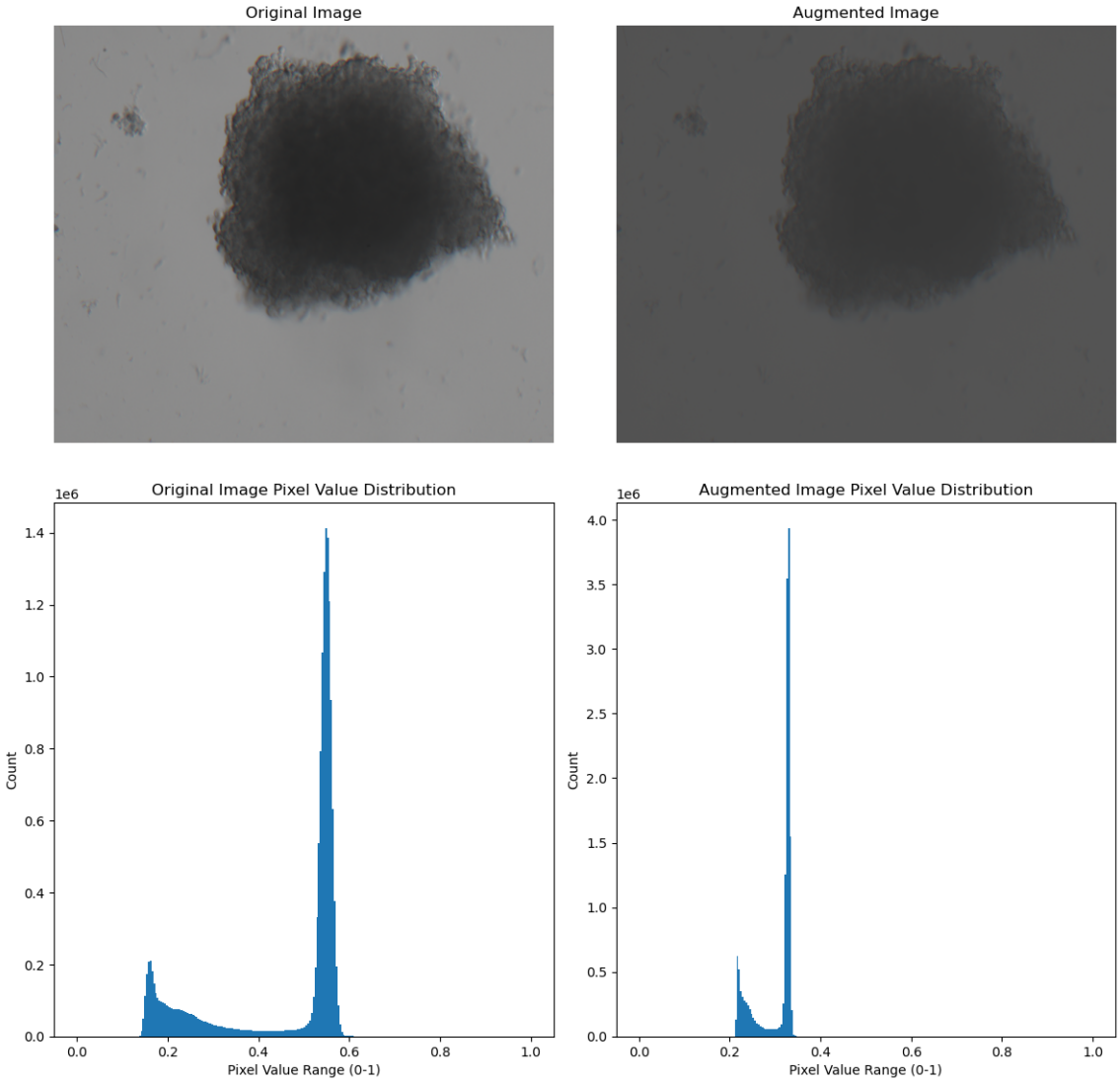

Figure 6.8: 16-bit three-channel image after 3000 epochs of random color jitter applied using PyTorch. Increase in percentage of unique pixel values: 78%

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

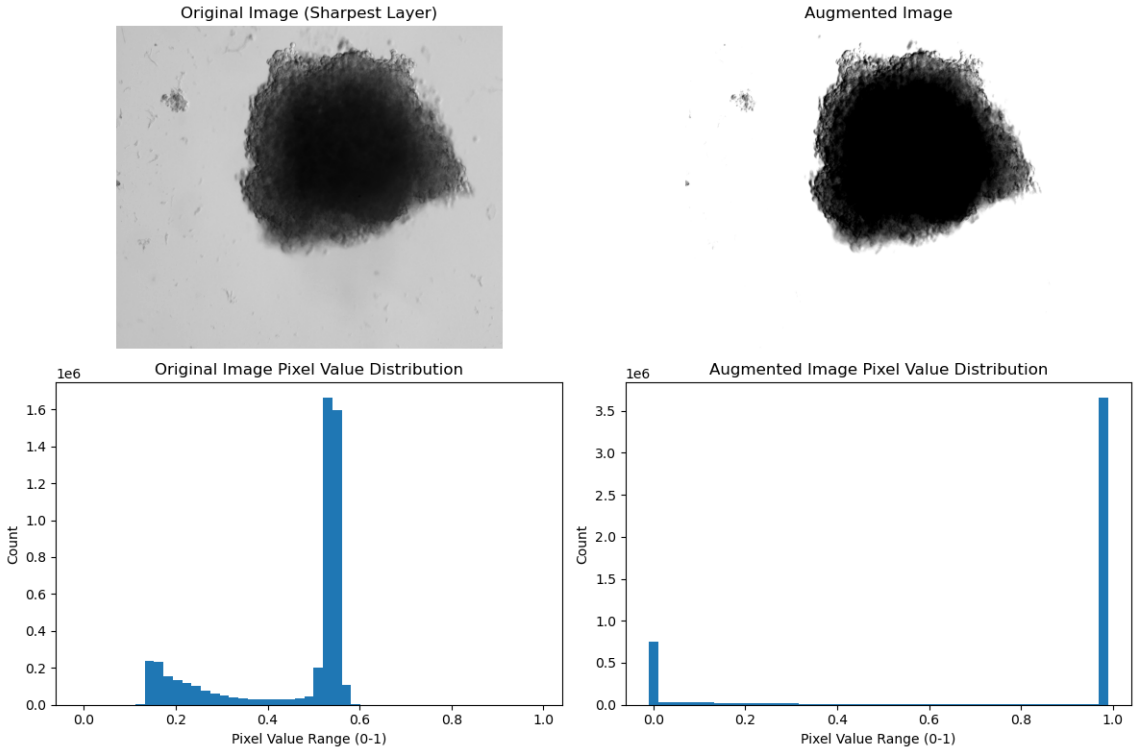

Figure 6.9: 8-bit single-channel image after 3000 epochs of random color jitter applied using PyTorch. Reduction percentage in unique pixel values: 49.88%

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 6.9. 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.

6.2 Training SSL model

For step 1, as explained in chapter 5, SimCLR was used as the first model for self-supervised learning (SSL). 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.

6.2.1 Using SimCLR as SSL model

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. 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.

2. Perform the following augmentations:

    a) Apply a horizontal flip.

    b) Randomly crop the image and resize it to 96×96.

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

3. 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.

4. 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.

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 6.2.1 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:

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

which we can reformualte as:

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

$$\ell_{i,j} = -\left(\log\left(\exp\left(\operatorname{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_j\right)/\tau\right)\right) - \log\left(\sum_{k=1}^{2N} \mathbf{1}_{[k\neq i]} \exp\left(\operatorname{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_k\right)/\tau\right)\right)\right)$$

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

follows:

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

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 6.2.1.

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. Apply more than two standard augmentations to meet the large batch size requirement of SimCLR [2].

6. Remove the positive sample j from the denominator of the loss function. Since j is the only image as a positive sample in the sum of the denominator softmax, its contribution will be less.

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

8. 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 [11] model for MRI brain images from PyTorch.

9. 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 9 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 6.2.1.

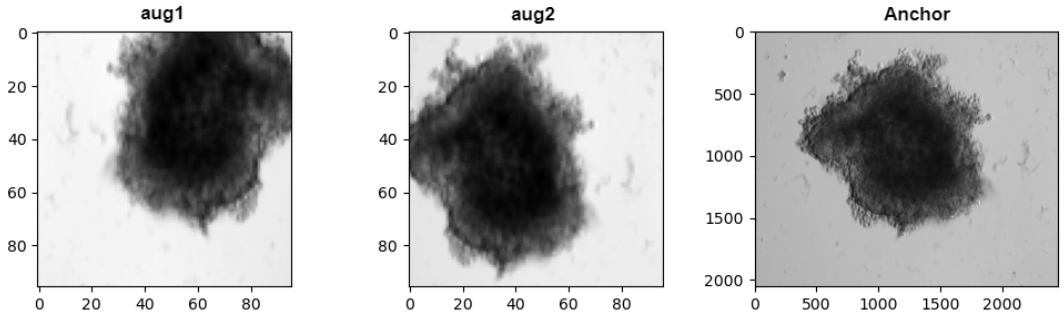

Figure 6.10: Sample 1: Anchor (the preprocessed original image) with 3 channels and its augmentations

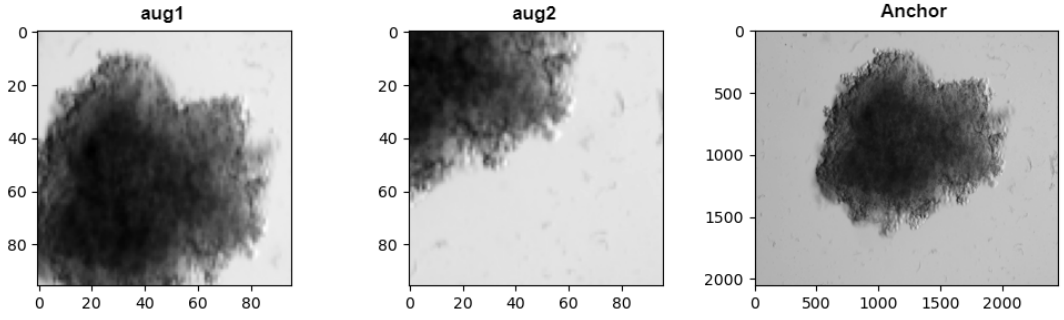

Figure 6.11: Sample 2: Anchor (the preprocessed original image) with 3 channels and its augmentations

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 6.2.1.

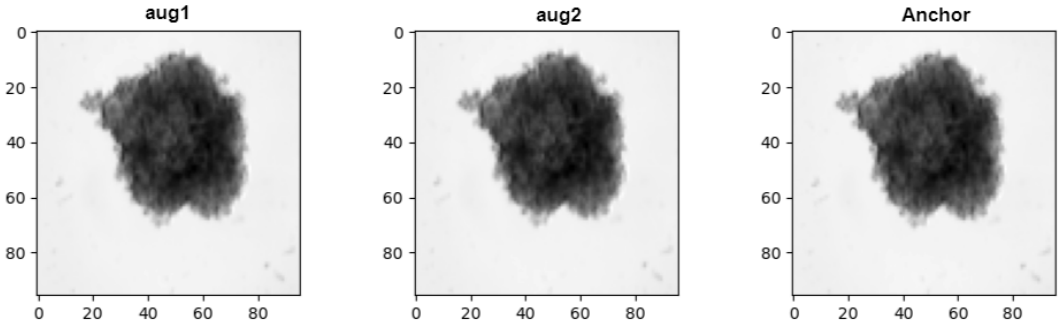

Figure 6.12: Sample 1: Anchor (the preprocessed sharpest layer amoung all 3 layers) with one channel and its augmentations

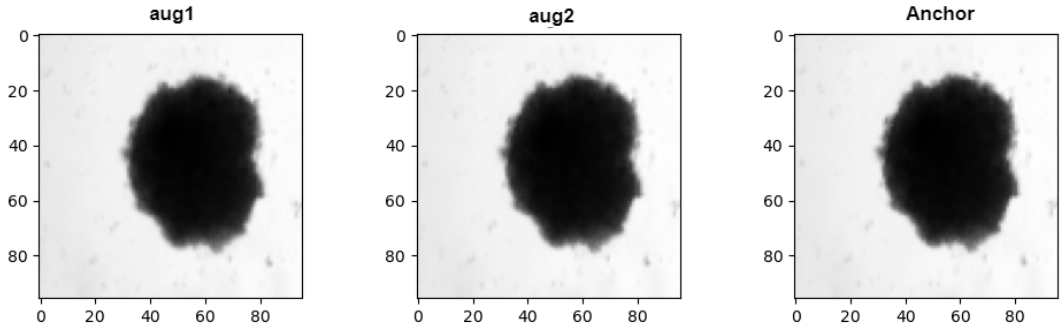

Figure 6.13: Sample 2: Anchor (the preprocessed sharpest layer amoung all 3 layers) with one channel and its augmentations

6.3 Intermediate evaluation of SSL model

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

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

thats why we need to do instance segmantation and classifiacation to see if it able to learn more than magnitude similarity and cosine similarity like as texture, contrast, and brightness

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.

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 6.2.1, 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.

7 Furture works

7.1 Time series prediction model

7.2 Integrated/ensembled model: SSL+Time series model

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] 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 a 2019.

[8] 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).

[9] Prannay Khosla et al. Supervised Contrastive Learning. 2021.

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

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Bi 2015.

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

[13] 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.

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

BIBIN BABU, 13.05.2024