Universiteit Leiden

Master Computer Science

Identifying promising regions of differentiated cardiomyocytes in brightfield images through Machine Learning

Name:	Lucas de Wolff
Student ID:	s3672980
Date:	31/10/2024
Specialisation:	Artificial Intelligence
1st supervisor:	Lu Cao
2nd supervisor:	Sylvia Le Dévédec

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Differentiating stem cells into cardiomyocytes is crucial for regenerative medicine and drug discovery for heart deceases. Assessing the quality of this differentiation is challenging and usually relies on manual microscopy, which is time-consuming and resource-intensive. Provided by an image dataset in which human induced pluripotent stem cells (hiPSCs) were differentiated into cardiomyocytes, we developed a two-stage machine learning pipeline to automate the evaluation of these differentiated cardiomyocytes relying only on images acquired from brightfield microscopy. In the first stage, we trained the EfficientNetV2 model on alpha-actinin-2 (ACTN2) fluorescent images annotated by experts. This model classified regions as well-differentiated or poorly differentiated cardiomyocytes with high accuracy. Using this trained model, we generated annotations for corresponding brightfield images, creating a new dataset with quality labels. In the second stage, we trained a similar CNN on these generated annotations to assess cardiomyocyte quality directly from brightfield images. While our approach shows promise, it has limitations. Relying on generated annotations may introduce errors, potentially propagating inaccuracies from the first model to the second. Additionally, the inherent noisiness and lower detail of brightfield images make accurate classification challenging. Time constraints also prevented the complete optimization of the brightfield model. Our results suggest that combining brightfield microscopy with machine learning can automate parts of the cardiomyocyte quality assessment process. However, further work is needed to improve data quality and model performance. This approach can reduce reliance on manual evaluations and accelerate advancements in stem cell therapies and regenerative medicine.

Contents

Acknowledgements

I want to express my deepest gratitude to my supervisor, Dr. L. Cao, for her invaluable guidance and expertise in both Computer Science and the Biomedical fields. I am also thankful to Dr. S.E. Le Dévédec for providing this project and her support throughout its development. I sincerely thank PhD student S.C. Davidse for his assistance and always being available to answer my questions. Lastly, I sincerely appreciate my girlfriend, friends, and family for their support and encouragement. Without the contributions of these individuals, this work would not have been possible.

I would like to thank the Allen Institute for providing their high-quality dataset and their readiness to offer additional support and resources throughout my research.

Chapter 1

Introduction

The differentiation of stem cells into specific cell types in vitro has enormous potential for regenerative medicine and drug discovery. For example, producing functional cardiomyocytes in the laboratory is crucial for developing treatments for heart disease, a leading cause of death globally. This capability could enable the repair or replacement of damaged cells or even entire organs. However, directing stem cells to differentiate into their intended cell types is challenging and prone to errors. Developing efficient and cost-effective differentiation protocols remains a significant hurdle in biochemistry.

A critical step in refining these differentiation protocols is accurately evaluating stem cell quality. Traditionally, this evaluation has been conducted through manual microscopy, where differentiated stem cells are examined using brightfield images. This manual method is time-consuming and demands significant human effort. Automating this evaluation with machine learning could greatly reduce reliance on human expertise, accelerating research and development in this area. While existing research has explored similar applications, none have specifically automated the assessment of cardiomyocytes using brightfield microscopy. This research aims to create a pipeline that takes a brightfield image as input and outputs a region map indicating well- and poorly differentiated cardiomyocytes.

To our knowledge, no dataset exists for brightfield microscopy that provides location-specific information on sarcomere quality. However, a recent study by the Allen Institute for Cell Science published a dataset in which experts manually labeled the sarcomere organization of human induced pluripotent stem cell (hiPSC)-derived cardiomyocytes in 18 microscopic images. They assigned class labels to regions with clearly defined structures, categorizing them into five classes ranging from no structure to highly organized. However, these annotations were based on ACTN2-tagged images rather than brightfield images. To overcome this limitation, we designed a two-stage classification pipeline that generates brightfield data using the provided ACTN2 pattern dataset. This generated data was then used to train a classifier for brightfield images. We also simplified the classification task to a binary evaluation of "good" and "bad" cells by merging the original five classes into two broader categories based on a visual interpretation of the Allen Institute's classification scheme.

In this study, we first train a convolutional neural network (CNN) on the expert-annotated ACTN2 data. Next, we use this trained model to generate predictions on ACTN2 images with corresponding brightfield counterparts. We then filter these predictions based on their confidence to ensure that only the highest-quality annotations are kept. Finally, we train another CNN on the generated data to create region maps for brightfield images of cardiomyocytes.

In this study, we aim to achieve the following goals:

1. Determine the accuracy of a machine learning model in detecting well-differentiated cardiomyocyte regions in ACTN2 images, using expert annotations as a reference.

2. Assess the effectiveness of the ACTN2 classifier in generating annotations for brightfield images.

3. Explore the feasibility of training a machine learning model to accurately assess cardiomyocyte quality from brightfield images, using the annotations generated in the previous step as ground truth.

This thesis is organized as follows: Chapter 2 provides essential background knowledge on relevant biomedical concepts and technical modeling details. Chapter 3 discusses related work, giving context and motivation for this study. Chapter 4 describes the dataset provided by the Allen Institute and used for training the classifiers. Chapter 5 outlines the methods for developing the ACTN2 model, including preprocessing, tuning, and model design choices. Chapter 6 presents the experimental results of the ACTN2 classifier. Chapter 7 details the methods for generating the brightfield dataset and creating the brightfield classifier. This is followed by Chapter 8, which presents the results of our final classifier. In Chapter 9, we discuss the results of both classifiers and the limitations of our work. Finally, we conclude the thesis in Chapter 10 and give suggestions for future research.

Chapter 2

Background

To fully understand the methods and results presented in this report, it is essential to grasp the relevant concepts from both biomedical and computer science. This section explains important concepts about stem cell differentiation and machine learning techniques used in this study.

2.1 Biomedical Background

2.1.1 Stem Cell Differentiation

Stem cells are unspecialized cells that have the potential to develop into many different cell types. There are two main categories: pluripotent stem cells and somatic stem cells. Pluripotent stem cells can differentiate into all adult cell types. On the other hand, somatic stem cells are located in adult organs or tissue and can only differentiate into cells of that respective organ or tissue. However, human induced pluripotent stem cells (hiPSCs) are unique; these somatic cells have been reprogrammed back to their pluripotent state, allowing them to change into cells other than their original lineage [24]. In this study, hiPSCs are differentiated into cardiomyocytes, the heart's muscle cells. For this, precise biochemical protocols are followed.

2.1.2 Cardiomyocytes: Structure and Function

Cardiomyocytes are the muscle cells responsible for the heart's contraction. As hiPSCs mature into cardiomyocytes, they undergo gene expression changes and structural developments. A key aspect of this maturation is the organization of sarcomeres, the essential building blocks of muscle contraction. When viewed under a microscope, the proteins within the sarcomeres initially appear diffused, which indicates an immature state. As the cells mature, the proteins form distinct, dot-like (punctate) patterns, showing increasing structural organization. This eventually leads to the formation of Z-disks, which anchor the thin filaments and are necessary for muscle contraction. Understanding sarcomere organization is necessary for assessing cardiomyocyte functionality and quality. The assessment of sarcomere organization is commonly done using microscopy [4].

2.1.3 Brightfield Imaging

Brightfield imaging is one of the most basic forms of microscopy. As a sample is illuminated from underneath, light passes through the sample, making denser areas appear darker, as opposed to transparent areas. This method allows us to capture live images of the cells while noninvasive. However, while brightfield microscopy captures important cell features, it focuses more on general cell morphology. It would, for example, be less suited for capturing sarcomere structure in cardiomyocytes than other more sophisticated techniques. As a result, distinguishing between well-differentiated and poorly differentiated cells based solely on brightfield images is challenging.

2.1.4 Fluorescent Imaging: ACTN2-mEGFP Labeling

Fluorescent imaging techniques allow us to study cellular physiology in more detail. Alpha-actinin-2 (ACTN2) is a protein located at the Z-disks of sarcomeres. This makes it a good marker for assessing sarcomere organization in cardiomyocytes. By genetically modifying hiPSCs to express the ACTN2 protein with a modified enhanced Green Fluorescent Protein (mEGFP), researchers can visualize sarcomere structures in live cardiomyocytes without applying external dyes or fixing procedures [18]. As the cells grow, they will naturally express this fluorescent marker. A light of a specific wavelength is then shone onto the sample, illuminating the fluorescent ACTN2 protein.

2.2 Computer Science Background

Interpreting cell microscopy results requires human expertise and is time-consuming. Machine learning can help alleviate this need by automatically detecting important patterns in the images. This section explains the relevant ideas behind such an intelligent system.

2.2.1 Neural networks

Neural networks are fundamental in the field of artificial intelligence. Inspired by the structure and function of the human brain, neural networks consist of layers of nodes, or neurons, that are wired together to form a network. A neural network has an input layer, one or more hidden layers, and an output layer. Typically, each neuron in a layer is connected to every neuron in the following layer, called a fully connected layer. The connections between neurons have an associated weight that determines the strength and direction of their influence. By changing these weights, the network can (semi-)independently 'learn' complex functions. Every neuron also has an activation function that determines whether or not the neuron will fire or the intensity of firing. The activation functions introduce non-linearity into the system, allowing the modeling of non-linear functions. In the case of a classification problem, the output layer of a neural network usually consists of a softmax activation function. This function produces a vector of probabilities summing up to 1, representing the class probabilities.

Training Neural Networks

Training a neural network involves adjusting its weights to minimize the loss function, which measures the error between the network's predictions and the ground truth. This adjustment is performed using an optimization technique called gradient descent. Gradient descent is an algorithm that seeks the (local) minimum of a function by iteratively moving in the direction of the steepest decrease, that is, opposite to the gradient of the function. The network updates its weights by calculating the gradient of the loss function with respect to each weight and then adjusting the weights in the opposite direction of this gradient. This backward computation through the network to update weights is known as backpropagation. The magnitude of these weight updates is controlled by a parameter called the learning rate. Formally, gradient descent is expressed as:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L(\theta_t) \tag{2.1}$$

Where θ_t represents the parameters (weights) at iteration t, η is the learning rate and $\nabla_\theta L(\theta_t)$ is the gradient of the loss function L with respect to the parameters θ at iteration t.

Due to loss landscapes' often noisy and irregular nature, standard gradient descent may not always reach the global minimum. Many improved optimization algorithms have been proposed to enhance the performance of gradient descent. The optimization algorithm used in this study is RMSprop

(Root Mean Square Propagation) [15]. RMSprop improves gradient descent by adapting the learning rate for each weight individually. It keeps track of a moving average of the squared gradients and adjusts the learning rates accordingly. For example, when the past gradients have been large values, the learning rate will be divided by a larger value and thus will result in a smaller update. This helps stabilize and accelerate training. The RMSprop algorithm is defined by:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2 \tag{2.2}$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{2.3}$$

Where $E[g^2]_t$ is the exponentially decaying average of past squared gradients at time t, γ is the decay rate that determines the weight given to past gradients, $g_t = \nabla_\theta L(\theta_t)$ is the current gradient of the loss function with respect to the parameters, and ϵ is a small constant to prevent division by zero.

Furthermore, a popular addition that can be applied to many optimization algorithms is *momentum*. Incorporating momentum into RMSprop accelerates convergence by considering the past velocity of parameter updates. RMSprop with momentum is defined as:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2 \tag{2.4}$$
$$v_t = \beta v_{t-1} + \frac{g_t}{\sqrt{E[g^2]_t + \epsilon}} \tag{2.5}$$
$$\theta_{t+1} = \theta_t - \eta v_t \tag{2.6}$$

Where v_t is the velocity (momentum) term at time t, and β is the momentum coefficient, which determines the contribution of the previous velocity.

Neural networks are susceptible to overfitting. This is when the model learns the training data too well, including its noise and outliers, leading to poor performance on unseen data. Many techniques have been proposed to combat overfitting. Dropout [22] is widely used due to its simplicity and proven effectiveness [28]. Dropout can be applied to specific layers, randomly turning off a certain proportion of neurons based on a dropout rate. This ensures that the network does not become overly dependent on any subset of neurons, improving its generalization ability.

Other techniques, such as batch normalization [9], are also commonly used in training neural networks. Data is usually processed in batches to save computational resources (and introduce some necessary noise). The gradient is computed on each batch; however, batches can vary significantly in their mean and standard deviation, which can hinder learning. By centering and rescaling each batch, batch normalization stabilizes the training process [20].

2.2.2 Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNNs) [11] are a class of deep learning models. A CNN introduces a new type of layer: the convolutional layer. These layers are specialized in extracting spatial features. This makes them ideal for image processing tasks like object detection, image classification, and image segmentation.

Convolutional Layers

A convolutional layer uses filters (kernels) to extract features from images. A kernel is a small matrix of values updated during training, similar to the weights between neurons. The kernel is slid over the image, and the dot-product is taken between the kernel and the image's covered pixel values. This results in a feature map representing the feature intensity at each image point. Usually, CNNs consist of many stacked convolutional layers with activation functions in between to introduce non-linearity.

Most CNN models also contain pooling layers, which are used to reduce dimensionality. While different types of pooling operations exist, global average pooling is used in this study. It is typically placed at the end of a CNN to extract the spatial features and transform them into a 1-D vector. As the name suggests, it takes the average value across all feature maps. The result is a vector containing spatial information that can directly feed into the output layer of the network.

CNN Architecture: EfficientNetV2

There are many design choices to be considered when designing a neural network. Such a design is called a network architecture, which is often created to solve a specific problem. This study uses the EfficientNetV2 CNN architecture by Tan et al.[25], published in 2021. This model family was optimized and evaluated on large image datasets to improve both parameter efficiency and training accuracy. Doing so, they reached much higher accuracy while using almost 7x fewer parameters than state-of-the-art models. This makes it particularly useful in environments with limited computational resources while still wanting high accuracy. Additionally, their models showed great potential for transfer learning, which is when a pre-trained model is fine-tuned on a different problem. They show that the pre-trained models can generalize well to other datasets.

2.2.3 Ensemble Models

In machine learning, an ensemble model combines the predictions of multiple models to improve overall performance. The intuition behind ensembling is that different models may learn complementary patterns from the data, and combining their predictions can help reduce errors and increase robustness. In the context of classification, this typically involves voting or averaging probabilities across the predictions of individual models. There are also more advanced ways to combine models. Three main types are bagging, boosting, and stacking. In short, When bagging, we combine the predictions of models trained on different data subsets. When boosting, we combine models created by iteratively fitting new models trained on the previous model's mistakes. Finally, when stacking, we use a meta-model that learns to combine the model predictions intelligently. Depending on the problem specifications, one method may be more applicable than the others.

2.2.4 Test-Time Augmentation

Test-time augmentation is a technique used to improve the robustness of a model's predictions by applying multiple random transformations (such as rotations, flips, and scaling) to the input data during inference. The model's predictions are averaged across these transformed versions, resulting in a more stable and reliable final prediction. This is particularly useful in scenarios where the input data might contain noise or variation that could lead to inconsistent or uncertain predictions.

2.2.5 Bayesian Optimization

Hyperparameter tuning is an essential step in optimizing machine learning models. Instead of manual tuning, many algorithms exist that automatically find the best hyperparameter configuration without any bias a manual search might have. Bayesian Optimization [21] is an algorithm for hyperparameter tuning that efficiently explores the hyperparameter space by building a probabilistic model of the unknown objective function. This probabilistic model usually uses a Gaussian process as its prior distribution. It will then use this model to select the most promising hyperparameter settings (with an acquisition function) and update its probabilistic model based on the received score. Figure 2.1 gives an example of a probabilistic model with four points selected as the initial samples. The blue line and contour represent the Gaussian process mean and its 95% confidence interval, i.e., the range of values it expects to see under the current probabilistic model for a hyperparameter value. At the sample locations, the uncertainty is zero, i.e. the model knows the exact performance of this value based on a previous observation.

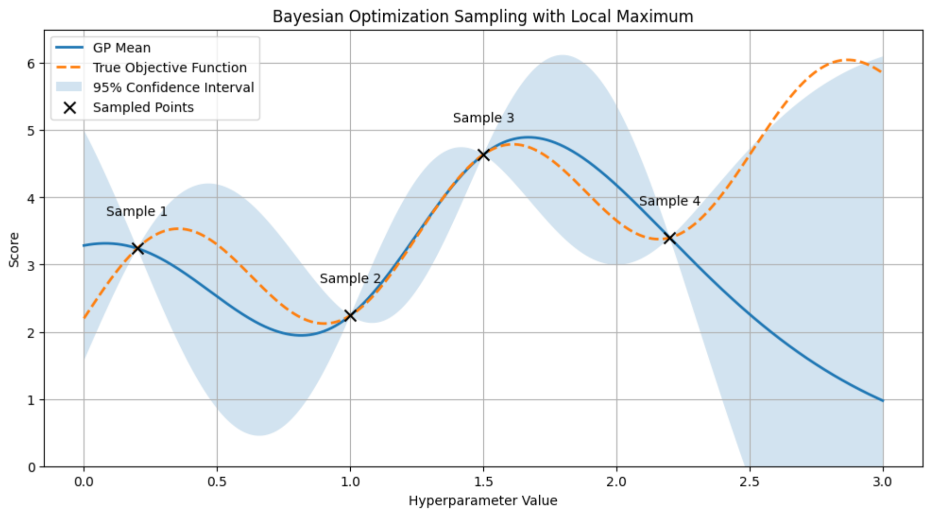

Figure 2.1: Example scenario of hyperparameter sampling using Bayesian optimization with a single hyperparameter. The orange dotted line is the true objective function of which we try to find the optima, which is unknown to us. The black crosses are hyperparameter values that have previously been selected and evaluated. The blue line is our current estimate of the true objective function with the uncertainties illustrated by the blue contour.

Unlike traditional grid search or random search, Bayesian Optimization balances exploration and exploitation, often finding better hyperparameters in fewer iterations.

Chapter 3

Related Work

Much work has been done to automate the assessment of differentiated cells using machine learning and to assist researchers in stem cell culture progress. While many previous studies share overlapping ideas or approaches, this section will highlight why this paper is novel in its application.

Waisman et al. [29] used a convolutional neural network (CNN) to distinguish pluripotent stem cells from early differentiating cells using brightfield microscopy. They achieved over 99% accuracy, showing that brightfield images can be reliable sources for detecting cell differentiation. However, their study focuses on comparing early-stage differentiation with no differentiation. The task becomes significantly more complicated when distinguishing between different stages of differentiation. This paper will limit the classes to poorly and well-differentiated cells.

Lien et al. [13] combined a CNN with a support vector machine (SVM) to classify iPSC-derived cells and evaluate the differentiation efficiency of retinal pigment epithelium cells (RPEs). In their work, they used phase contrast images as the input of their CNN. The classification model achieved 97.8% accuracy, distinguishing between iPSCs, iPSC-MSCs, iPSC-RPEs, and iPSC-RGCs. Through PCA projection of the last layer of the CNN, they manually validated that the trained CNN could be used to recognize different degrees of differentiation in iPSCs-derived cell lineages.

Zhu et al. [30] developed a deep learning model for predicting neural stem cell differentiation from brightfield images. Similarly, Nguyen et al. [16] used brightfield images to predict neural stem cell differentiation. However, their focus lies on single cells and may not apply to cases of cell colonies, which is usually the case when growing stem cells.

As mentioned in the introduction, the work of Gerbin et al. [6] is central to this study. They developed an imaging platform, which they used to create a dataset containing over 30,000 hiPSC-derived cardiomyocytes. They then trained a ResNet18 [7] to classify the local organization of alpha-actinin-2 in subcellular regions into six classes. However, this requires an expensive genetically altered ACTN2-mEGFP. Additionally, their data split might have leaked train data into the test data, which we will elaborate on later in this paper. Ahola et al. [1] used this dataset and fed brightfield images to a U-Net model to estimate cardiomyocyte structure and maturity. However, their study focused mainly on nuclear orientation.

This study builds on these works by focusing on cheap, easy-to-get brightfield microscopy for cardiomyocyte differentiation. Using a two-stage classification pipeline and simplifying the task to binary categories of "good" and "bad" cells, this approach aims to provide a solution for automatically evaluating cell differentiation in cardiomyocytes. The main challenge is that we have to deal with the low-feature and noisy properties of the brightfield images in the context of cardiomyocytes. Overcoming this will be a major step in the automation of cell assessment.

Chapter 4

Data

This chapter describes the datasets used in our study[1], provided by the Allen Institute as part of their research on the relationship between RNA abundance and cellular organization. The datasets include raw and processed images of cardiomyocytes and numerical data capturing various cellular features. Central to our work is the dataset containing expert annotations of sarcomere organization using images capturing ACTN2 fluorescent signals, in this paper referred to as ACTN2 images. Additionally, we used a dataset that includes both brightfield and ACTN2 images. Below, we detail the generation of the images and describe the datasets used.

4.1 Generation of Cardiomyocyte Images

The Allen Institute developed a model system to analyze the relationship between structural organization and cell gene expression. They used a human induced pluripotent stem cell (hiPSC) line expressing ACTN2-mEGFP, a validated approach for studying hiPSC differentiation [19]. This cell line was differentiated into cardiomyocytes over 32 days using an optimized small-molecule protocol to maximize differentiation efficiency.

On day 12, the wells were visually inspected to assess their suitability for further analysis. Flow cytometry determined an efficiency of 78.1% ± 3.7% was achieved. The cells were then replated for high-resolution imaging and assessed for quality using antibody labeling of sarcomeric proteins. Cells that did not meet the quality standards were excluded from further study. On days 18, 25, and 32, the cardiomyocytes were imaged live at high resolution using a 3i spinning-disk microscope. All images in this study have a width of 1776 and a height of 1736 pixels.

[1]Allen Institute imaging dataset

4.2 Expert-Annotated ACTN2 Dataset

Dataset Location: `actn2_pattern_ml_classifier_train` folder.

This dataset consists of 18 high-resolution images of differentiated cardiomyocytes captured using the alpha-actinin-2 marker. These images represent the highest average intensity slice of their original 3D stack. Experts annotated regions within each image based on five distinct patterns of sarcomere organization, as illustrated in Figure 4.1:

1. **Diffuse/Other (Class 1)**: No clear sarcomere structures.
2. **Fibers (Class 2)**: Fibrous structures without clear sarcomere organization.
3. **Disorganized Puncta (Class 3)**: Punctate patterns without regular alignment.
4. **Organized Puncta (Class 4)**: Punctate patterns showing some alignment.
5. **Organized Z-Disks (Class 5)**: Well-organized sarcomere structures with clear alignment.

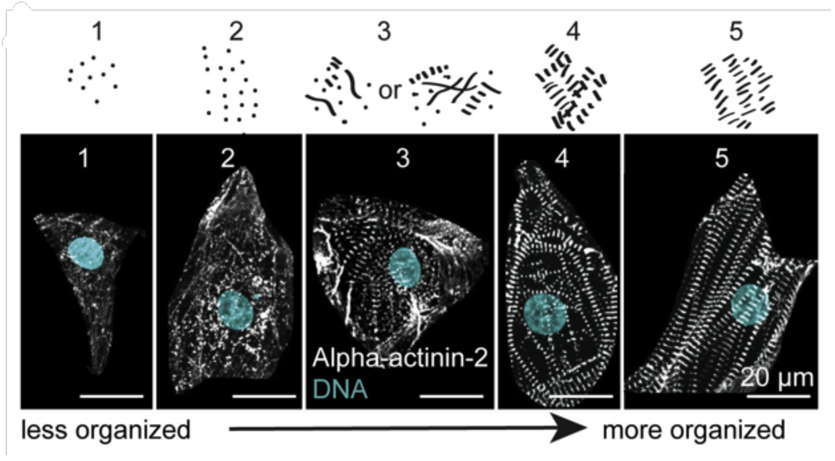

Figure 4.1: Reference images for manual annotation of sarcomere structures across five organization classes. Reproduced from Gerbin et al.[6] under the Creative Commons CC BY-NC-ND license.

Pattern	Number of Samples
Diffuse/Other (Class 1)	596
Fibers (Class 2)	688
Disorganized Puncta (Class 3)	727
Organized Puncta (Class 4)	906
Organized Z-Disks (Class 5)	572
Total	3,489

Table 4.1: Number of annotations per sarcomere organization class in the expert-annotated ACTN2 dataset.

Within each image, regions that clearly represented one of the five classes were identified, resulting in approximately 3,489 annotations, as detailed in Table 4.1.

The annotations were organized into five folders corresponding to each class, each containing the x and y coordinates and their associated image numbers. Figure 4.2 shows an example of an ACTN2 image with its annotations.

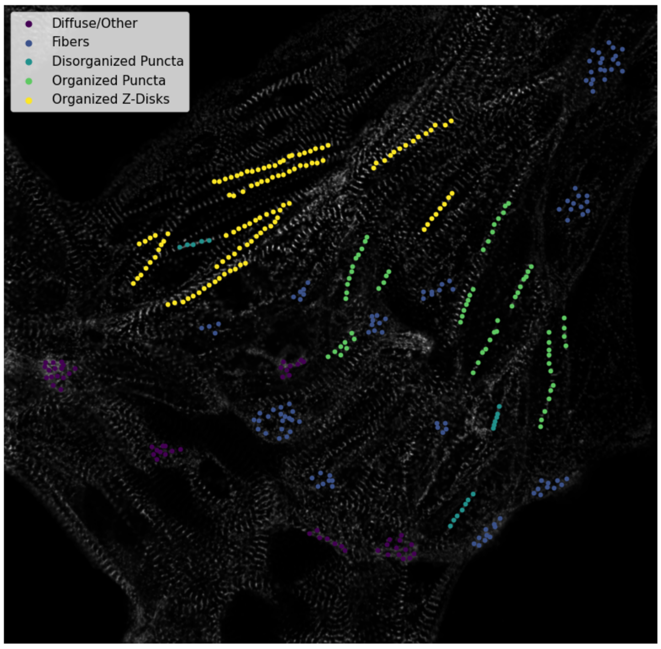

Figure 4.2: An example of an ACTN2 image of differentiated cardiomyocytes with overlaid expert annotations. The annotations highlight regions corresponding to the five classes of sarcomere organization used for classification.

4.3 Brightfield and ACTN2 Images Dataset

Dataset Location: `2d_segmented_fields_fish_1/2D_fov_tiff_path` folder.

This dataset consists of 478 images, each containing ten channels corresponding to different imaging techniques, including brightfield and ACTN2 channels. The purpose of the dataset was to collect all images used in their research (not including the annotated ACTN2 images). The brightfield images were acquired using an LED light source with a peak emission of 740 nm and a bandpass filter of 706/95 nm, capturing the cells without any fluorescent markers. The ACTN2 images are similar to those described in the previous section but without expert annotations.

The images in this dataset provide the brightfield counterparts to the ACTN2 images, allowing for analysis that includes both structural protein markers and standard microscopy images. Figure 4.3 shows the first two channels of a 10-channel image, corresponding to the brightfield and ACTN2 channel.

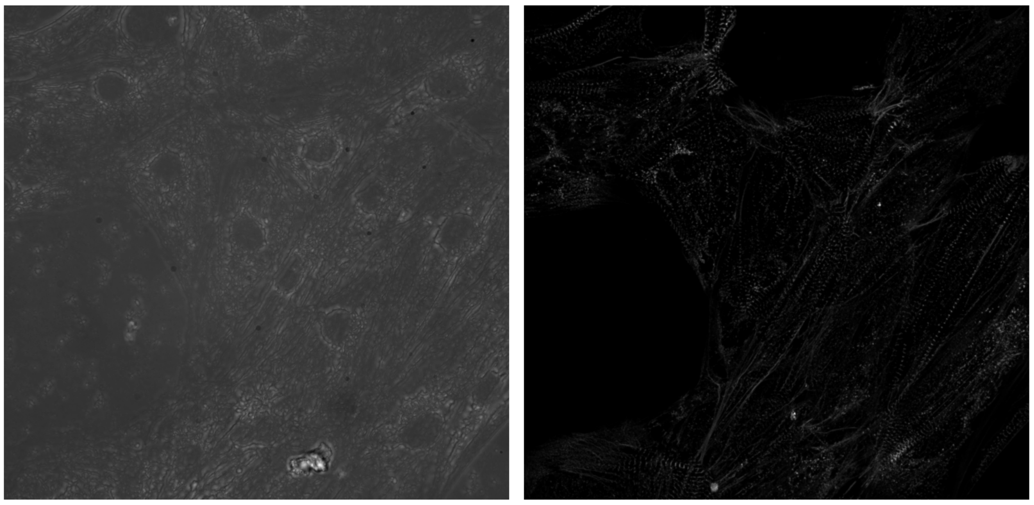

Figure 4.3: First two channels of the 10-channel image, as provided by the Allen Institute. The two channels are the brightfield and ACTN2 channels.

Chapter 5

Methods: ACTN2 Model

This chapter details the development of the ACTN2 pattern classifier. First, we describe the preprocessing steps to prepare the data for training. Then, we discuss the modeling process, including hyperparameter tuning and the training of the classifiers. Finally, we explain how the models are combined into an ensemble to generate the final predictions.[1]

5.1 Preprocessing ACTN2 Pattern Dataset

The ACTN2 pattern dataset had been split into five subfolders, each holding the data for the specific organization class. Within these folders, coordinates were stored alongside a number between 1 and 18, indicating the corresponding image ID. A central task was to transform the annotations into small parts of the image, called *patches*, which the computer vision model can use to learn class-specific features. The patches underwent some preprocessing steps to mold them into usable training data.

5.1.1 Binary Classification

This research aims to identify well-differentiated regions rather than specific sarcomere organization patterns. Therefore, the first and most crucial preprocessing step involved assigning all five classes to one of two categories: well-differentiated and not-well-differentiated cardiomyocytes. This simplification of the classification task is expected to result in higher accuracy compared to using five classes.

[1]All work in this paper was carried out using the hardware and software specifications listed in Table C

Because the five classes can be seen as ordinal data (where the quality of cardiomyocytes increases from left to right), splitting the classes at the midpoint seems reasonable. Following expert opinion, the decision was made to classify 'Diffuse/other' and 'Fibers' as the *bad* class and 'Disorganized puncta,' 'Organized puncta,' and 'Organized z-disks' as the *good* class. The primary rationale was that the formation of puncta patterns is considered positive, thus including the 'Disorganized puncta' class in the *good* category. After binarizing the five classes, the bad class consisted of 1284 samples, and the good class consisted of 2205 samples.

5.1.2 Splitting the Data

The dataset was divided into training, validation, and test sets with a ratio of 80%, 10%, and 10%, respectively. Due to the spatial proximity of annotations (depending on the patch size), a simple random split would likely result in data leakage from the test and validation sets into the training set. To be specific, annotations that are located within the distance of the patch size will share overlapping pixels. Therefore, by splitting on the annotation level, it is likely that the test and validation sets contain patterns seen by the model in the training set. To address this issue, the data was split at the image level, ensuring that each image was assigned to only one of the splits. This method also aimed to maintain class balance across all splits. An overview of the class distribution in each split is provided in Table 5.1.

Split	Image	Bad Count	Good Count	Split Proportion
Train	{2,3,4,5,6,7,8,9,10, 11,12,13,15,16,17}	987	1773	0.791
Validation	{0}	168	203	0.106
Test	{1,14}	129	229	0.103

Table 5.1: Distribution of class samples across the training, validation, and test sets of the annotated ACTN2 data. The dataset was split at the image level to avoid data leakage and maintain class balance. The table shows the number of samples for each class and the proportion of the total dataset represented by each split.

5.1.3 Data Augmentation

Neural networks typically require a large number of samples to perform well. However, the current dataset is relatively small, increasing the risk of overfitting, especially when using a complex model. Additionally, the imbalance in the training set might cause the model to favor the majority class during predictions. Data augmentation effectively addresses both issues by artificially expanding the dataset and creating new, unique instances from the original samples.

A set of possible transformations was defined, including rotations of 90, 180, or 270 degrees and vertical or horizontal mirroring. When augmenting each patch, transformations were sampled without replacement from this set, ensuring no transformation was reused for the same patch. To address the class imbalance, the bad class was augmented five times, while the good class was augmented three times. This resulted in 4,935 bad samples and 5,319 good samples, totaling 10,254 patches and increasing the training set size by a factor of 3.71. The validation and test sets were not augmented.

5.1.4 Tuned Preprocessing Steps

Selecting appropriate preprocessing methods is crucial for optimizing machine learning models. To avoid extensive manual testing of various methods and their combinations, we used the automatic tuning algorithm Bayesian Optimization. This algorithm efficiently samples hyperparameters to find optimal (or near-optimal) configurations within a given time limit. Both the data preprocessing steps and the model hyperparameters underwent automatic tuning. This section will cover the tuned preprocessing steps.

Patch Sizes

The coordinates were converted into square patches with sizes of 24, 48, or 96 pixels, with the coordinate positioned at the center pixel. Gerbin et al. used a patch size of 96 pixels. Their study did not specify their approach to handling edge cases where annotations were within 96 pixels of the image boundary. A potential solution would be to pad the regions outside the image with zeros. However, because these annotations would likely act as noise and were limited in their number, we chose to exclude them.

Histogram Normalization

Increasing the contrast in images can aid neural network learning. Histogram normalization redistributes pixel values so that the cumulative distribution of the histogram approximates a straight line, thus spreading pixel intensities more evenly. A Gaussian smoothing filter (kernel size of 5 by 5 pixels) is applied *before* histogram normalization to avoid enlarging any noise in the original image. The Gaussian blur is only applied when histogram normalization is enabled.

Scaling Methods

Following optional histogram normalization, a scaling method was applied to further aid model convergence and avoid issues like vanishing or exploding gradients. Three popular scaling techniques were considered, where each can outperform the other depending on the problem: centering, standardization, and Normalization.

- *Centering* involves subtracting the mean pixel intensity of the training images from all images, centering them around zero.

  $$X_{\text{centered}} = X - \mu \tag{5.1}$$

  where X represents the original pixel intensity and μ is the mean pixel intensity of the training dataset.

- *Standardization* extends centering by also dividing pixel values by the standard deviation, ensuring images have zero mean and a standard deviation of one.

  $$X_{\text{standardized}} = \frac{X - \mu}{\sigma} \tag{5.2}$$

  where σ is the standard deviation of the pixel intensities in the training dataset.

- *Normalization* scales pixel values to a range between 0 and 1 by subtracting the minimum pixel value and dividing by the range of pixel values.

  $$X_{\text{normalized}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{5.3}$$

  where X_{min} and X_{max} are the minimum and maximum pixel intensities in the training dataset, respectively.

5.1.5 Additional Preprocessing

To ensure the data was fully processed and ready for training, all patches were upscaled using OpenCV's [3] linear interpolation method to match the models' input shape of 128×128 pixels. Additionally, the models in this study expect RGB input images, while our data consists of grayscale images. A common practice is to stack each grayscale image three times along the channel dimension, creating a three-channel RGB image with identical channels. As a final step, the binary class labels were one-hot encoded into two-element vectors to be compatible with the categorical loss functions used during training.

5.2 ACTN2 Model

5.2.1 Architecture

Since everything was run on a single local GPU, computational resources had to be spent wisely. Therefore, one main architecture was exploited for its efficient learning using a small number of trainable model parameters. This family of models is called EfficientNetV2. Through the high-level keras API, multiple versions of the EfficientNetV2 model were available, where the main difference lay in their number of parameters. In this study, we experimented with (EfficientNet)V2B1, V2B2, V2B3, and V2S. The models have roughly 8 million, 9 million, 14 million, and 24 million parameters, respectively. We discovered in early experimentation that for the ACTN2 classifier, larger models (V2B3 and V2S) performed better compared to their smaller siblings (V2B1 and V2B2). Therefore, only these were considered for this subproblem. Both models had been pre-trained on the ImageNet dataset, as Gerbin et al. did, with the difference being that they used a ResNet18 model.

Modifying the Models

The pre-trained EfficientNet models on ImageNet-1000 have an output layer designed for 1,000-class classification. We replaced the original output layer with a new, randomly initialized classification head to adapt these models for our binary classification task. Specifically, we removed the original output layer and added the following layers:

- A global average pooling Layer
- A batch normalization Layer
- A dropout layer
- A Fully Connected Layer with two neurons and a softmax activation function

The new classification head was initialized randomly and trained from scratch, whereas the existing layers were fine-tuned.

5.2.2 Tuning

Not all hyperparameters were tuned automatically; some were set to certain values based on previous research. The batch size was set to 32, as larger batch sizes can lead to sharp minima, which tend to generalize less well compared to flat minima [10]. The recommended values from Tan et al. were used for the optimizer hyperparameters, except for the RMSprop momentum hyperparameter. This hyperparameter controls the accumulation of past gradient updates to smooth and accelerate the optimization process during gradient descent. We also experimented with different patch sizes to explore whether varying context sizes would improve model performance. Additionally, experiments were conducted using histogram equalization and different scaling methods. Finally, the learning rate was tuned within the range of $1e^{-7}$ to $1e^{-3}$ using doubling steps to simplify the search process.

Table 5.2 showcases the tuned hyperparameters. These included both model hyperparameters as well as those affecting the data preprocessing. The Bayesian Optimization implementation by Keras Tuner [17] was used as the automatic tuning algorithm. The number of trials was set to 100, and each trial was set to last for at most 25 epochs. For both the tuning process and the final model training, an early stopping procedure was used with a patience of 10 epochs, monitoring the validation loss. EfficientNetV2S and EfficientNetV2B3 models were tuned separately, resulting in distinct hyperparameter recommendations.

Table 5.2: Hyperparameter Setup for Automatic Model Tuning

Hyperparameter	**Type**	**Value / Range**
Batch Size	Fixed	32
Rho (RMSprop)	Fixed	0.9
Learning rate decay	Fixed	0.9
Patch Size	Tuned	{24, 48, 96}
Histogram Equalization	Tuned	{True, False}
Scaling	Tuned	{Center, Normalize, Standardize}
Learning Rate	Tuned	1e-3 to 1e-7 with halving steps
Dropout Rate	Tuned	0.1 to 0.5 with steps of 0.05
Momentum (RMSprop)	Tuned	0.8 to 1.0 with steps of 0.01

5.2.3 Training

Binary cross-entropy, also known as the log loss, was used for the loss function. Since the classes are well-balanced, accuracy was used as our evaluation metric.

As the authors of EfficientNetV2 recommended, we used the RMSprop optimizer in combination with a linear warm-up of the learning rate. In the warming-up episode, the learning rate is linearly increased from 0 to a target learning rate. Doing this can push the model away from 'bad' loss landscapes and reduce variance during learning [14]. This episode lasts for five epochs. After this, the learning rate is decayed using exponential decay with a decay rate of 0.9. The warm-up episode length and the decay rate were not tuned during the hyperparameter search to save computational resources. In contrast to the tuning phase, during training, the models were given 100 epochs instead of 25 epochs, with a patience of 10 epochs for early stopping.

Following the hyperparameter tuning, the top hyperparameter settings were selected for both models based on their tuning results. This selection was based on whether the tuning processes achieved good accuracy and whether the hyperparameters were distinct enough to cause model variety. Each configuration was then used to train a new model, following the above-described training settings while allowing the models more time to converge.

5.2.4 Creating Final Predictions

Several steps were taken to take full advantage of the patterns learned by the models. Since most models already achieved high accuracy on the training set (>98%), using this to evaluate the ensembles would not effectively differentiate the performances. Therefore, the validation set was used to test and compare the ensembles, while the test set was reserved for evaluating the final performance.

Ensemble

The final ensemble predictions were made by aggregating the outputs of the seven trained models. Each model produced a probability vector for the two classes, which were then averaged. The class with the highest combined probability was selected as the final prediction. To optimize the ensemble, a brute-force search tested all possible subsets of the models, identifying the combination that minimized validation error. It was hypothesized that the different hyperparameter settings would cause enough diversity in our models to form effective ensemble models.

Test-time augmentation

We applied multiple random transformations to each data patch and averaged the model predictions across these transformations. This method, called test-time augmentation, can reduce sensitivity to input variations and increase model robustness. The same 11 transformations used during data augmentation (Section 5.1.3)—including rotations, flips, and scaling—were applied here. The transformations were sampled randomly with no replacement from this set.

Different numbers of transformations were tested, ranging from none to several, with the final prediction being the average of all transformed outputs.

Ensemble + Test-time Augmentation Experiment

All possible ensembles were evaluated across a range of test-time augmentations, varying the number of transformations from 0 to 11. Because the augmentations are sampled randomly, the experiment was repeated 100 times, after which the accuracy was averaged. The ensemble and number of transformations that achieved the highest validation accuracy were selected for generating brightfield data and evaluated on the test set for reference.

Chapter 6

ACTN2 Classifier Results

This chapter will cover the results of the ACTN2 classifier, which is the first step in moving towards a pipeline that can detect well-differentiated cardiomyocytes using brightfield images. First, the hyperparameter tuning results will be presented. Then, the training curves, which use the best-found hyperparameters, will be shown. Finally, the results for finding the best ensemble combination and number of alterations in test-time augmentation will be displayed.

6.1 Tuning Results

After performing Bayesian Optimization, the hyperparameter configurations were ranked based on their lowest achieved validation loss. For EfficientNetV2B3, the top three configurations were selected to potentially contribute to the final model ensemble. For EfficientNetV2S, we initially also selected the top three configurations. However, due to its slightly different hyperparameter settings, the fourth-best configuration was also chosen to introduce more diversity in the ensemble's predictions. Table 6.1 displays the identified hyperparameter configurations and their corresponding best validation accuracy.

From the table, it is clear that most configurations share similar hyperparameter values. The patch size is consistently set to 96 pixels, and histogram equalization is disabled for all models. A learning rate between 5e-05 and 1e-4 proves effective, and dropout is generally maintained at around 0.3. Two hyperparameters show more variation: scaling and momentum. The V2B3 models typically use centering, while the V2S models prefer standardization. Although the original EfficientNetV2 authors used momentum values around 0.9, lower values near 0.8 also performed well.

	Model names						
Hyperparameters	V2B3 1	V2B3 2	V2B3 3	V2S 1	V2S 2	V2S 3	V2S 4
Patch Size	96	96	96	96	96	96	96
Histogram Equalization	False	False	False	False	False	False	False
Scaling	center	center	center	stand.	stand.	stand.	norm.
Learning Rate	5e-05	5e-05	1e-4	1e-4	1e-4	1e-4	5e-05
Dropout Rate	0.4	0.35	0.3	0.35	0.35	0.3	0.3
Momentum (RMSprop)	0.91	0.92	0.95	0.83	0.82	0.81	0.81
Accuracy (validation)	0.9312	0.9252	0.9243	0.9398	0.9262	0.9349	0.9249

Table 6.1: **Hyperparameter configurations for the seven best models.** The number following the model type indicates the rank of the hyperparameter configuration in terms of performance (validation loss) during tuning. 'Stand.' refers to standardize, and 'norm.' refers to normalize.

6.2 Training results

All seven models were trained for up to 100 epochs. Figure 6.1 shows the training and validation accuracies observed during training. For simplicity, each model is numbered, and from here on, models are referenced by their respective numbers.

Despite the increased training time compared to the tuning process, most models were early-stopped before reaching 25 epochs. Only V2B3 Model 1 was trained until epoch 27. Most models achieved nearly perfect scores of 1.0 on the training set. For some models, the validation accuracy fluctuated significantly between epochs. Model 7 exhibited highly oscillating behavior, with accuracy dropping to around 50% at certain points. Models 3 and 4 showed a very gradual increase in validation accuracy before being early-stopped.

	Model names						
	V2B3 1	V2B3 2	V2B3 3	V2S 1	V2S 2	V2S 3	V2S 4
Accuracy (validation)	0.9326	0.9084	0.9272	0.9407	0.9299	0.938	0.9218

Table 6.2: **Validation accuracies for the seven trained models.** The number following the model type indicates the rank of the hyperparameter configuration in terms of performance during tuning.

Figure 6.2 shows that not all models retained or improved their performance relative to the tuning phase. V2B3 Model 2, for instance, showed a decline in validation accuracy, dropping from 0.9252 during tuning to 0.9084. V2S Models 3 and 4 also experienced slight decreases in accuracy. In contrast, V2B3 Models 1 and 3, along with V2S Models 1 and 2, demonstrated slight improvements, with V2S Model 1 achieving the highest validation accuracy of 0.9407.

6.3 Ensemble & Test-Time Augmentation

We evaluated all possible combinations of the seven models. For each combination, we also tested different augmentation levels, ranging from no augmentations to 11 augmentations per image (using all augmentations). The results are shown in figure 6.2. We see that, in general, increasing the number of augmentations correlates positively with accuracy, with the highest accuracy typically achieved using 11 augmentations. The ensemble of models 6 and 7 achieved the best validation accuracy of **0.9596**, which is an improvement over their individual accuracy scores of 0.938 (6) and 0.9218 (7). On the test set, the same ensemble reached an accuracy of **0.9469**, though other ensembles reached a higher accuracy on the test set. It can also be seen that the confidence interval for the best-performing model decreases in size when the number of augmentations is increased, both for the validation and the test set.

6.4 Confusion Matrix

In figure 6.3, we see the predictions on the test set (using the ensemble of models 6 and 7 with 11 augmentations) compared with their ground truth labels presented in a confusion matrix. The ensemble model makes a few mistakes overall, getting slightly fewer predictions wrong in the good category.

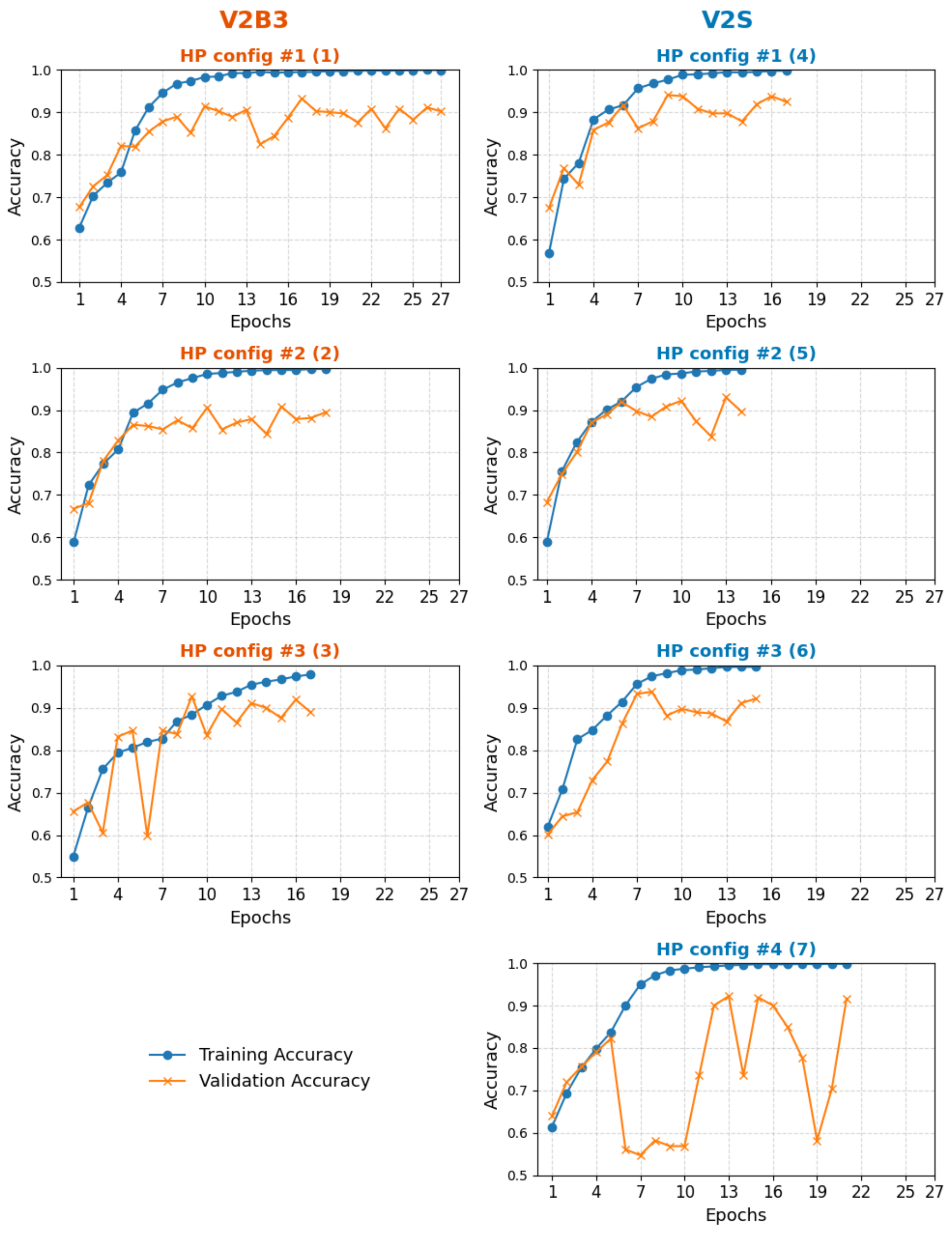

Figure 6.1: **Training and validation accuracies over epochs for the seven trained models.** For convenience, the models are labeled 1 to 7. Models 1–3 represent V2B3 configurations, and models 4–7 represent V2S configurations.

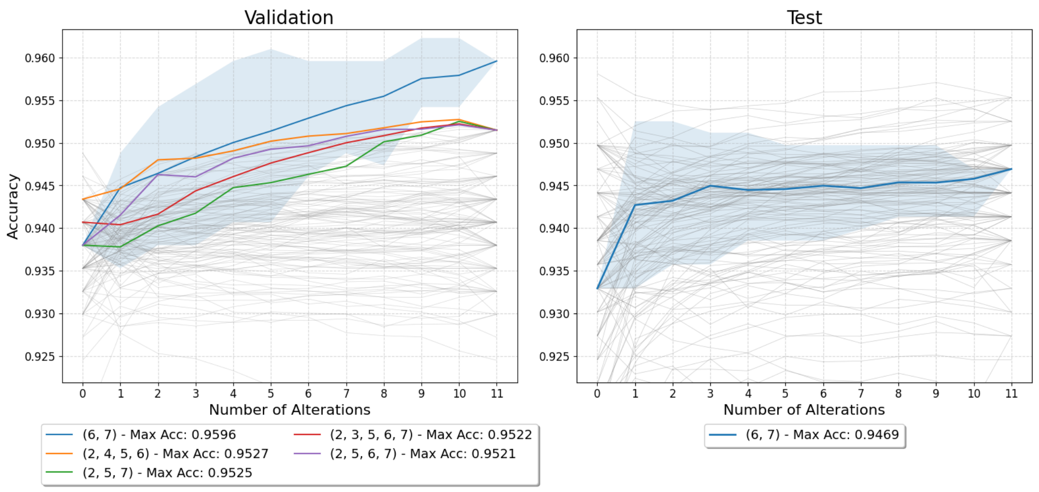

Figure 6.2: **Validation and Test Accuracy of Various Ensembles Across Alterations. Left:** Validation accuracies for all ensembles, highlighting the top five performers. **Right:** Test accuracy of the best-performing validation ensemble. Accuracies are averaged over 100 runs.

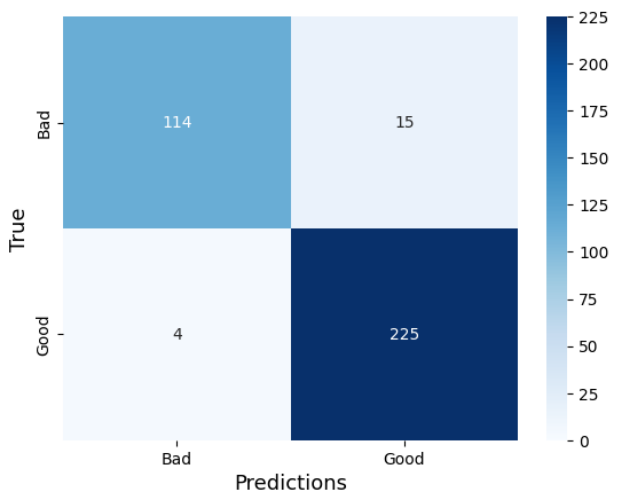

Figure 6.3: **Confusion matrix of the best ensemble model (ACTN2)** on the total test set, using test-time augmentation. Single run, since all augmentations were used and the model output is deterministic.

Chapter 7

Methods: Brightfield Model

This chapter outlines the methods used to develop the Brightfield model, which classifies brightfield images based on annotations generated by the ACTN2 ensemble. The process includes generating annotated brightfield data, splitting the data, and training the Brightfield model. The goal is to create a model that can take a brightfield image and produce a region map identifying 'good' and 'bad' differentiated cardiomyocytes. Due to time constraints, specific optimizations, such as automatic hyperparameter tuning, were not performed for the Brightfield model.

7.1 Generation of Annotated Brightfield Data

To create annotations for the brightfield images, we used the best-performing ACTN2 ensemble, which combined models 6 and 7 and applied 11 test-time augmentations. We randomly selected 100 images from the combined brightfield and ACTN2 datasets. Since the ACTN2 images in this subset were slightly darker than those used to train the ACTN2 model, we adjusted their histograms to match the average histogram of the training images from the ACTN2 model. This adjustment ensures consistency in the image distributions.

Next, we isolated the foreground pixels in each image to avoid feeding the trained ensemble patches considered background. We achieved this separation using Li thresholding [12], an automatic method determining the optimal pixel intensity to distinguish between foreground and background. Figure 7.1 shows an example of the foreground extraction process. Because a patch size of 96×96 pixels was used in our models, pixels within 48 pixels of the border were excluded.

We randomly selected 500 pixels per image from the foreground pixels

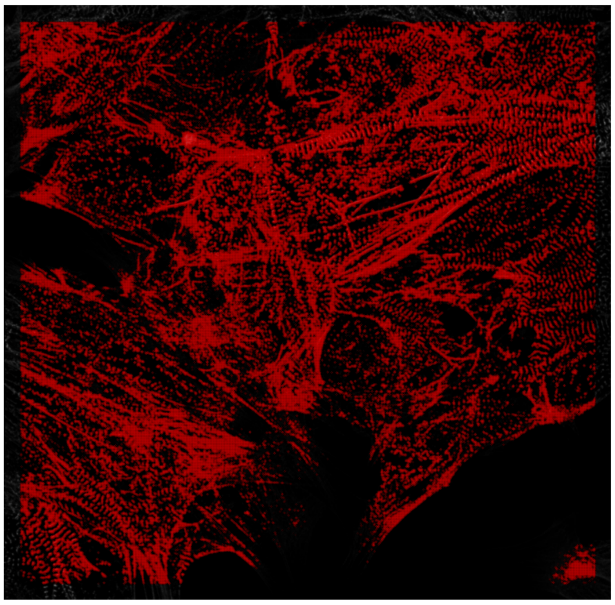

Figure 7.1: **An example of the selected foreground pixels (red) in an ACTN2 image after applying Li thresholding** to separate foreground from background pixels. No pixels within 48 pixels of the border were selected.

to serve as the center of the new image patches. Test-time augmentation was applied to each patch (11 random augmentations) and then fed to the ACTN2 ensemble model, which provided class probabilities for each patch. To ensure high-quality annotations, we applied a confidence threshold. By analyzing the ACTN2 validation set predictions, we determined that a confidence threshold of roughly 0.97 (0.9739) maintained a validation accuracy of 99%. This means that only predictions where the difference between class probabilities exceeded 0.97 were considered reliable. As shown in Figure 7.2, the green dotted line, which represents the 99% threshold, intersects the red line at around 0.72. This means this threshold allowed us to retain approximately 72% of the patches while having near-perfect annotation accuracy. The reason for selecting a 99% accuracy threshold instead of 100% is that 100% accuracy was never achieved on the validation set through increasing the threshold.

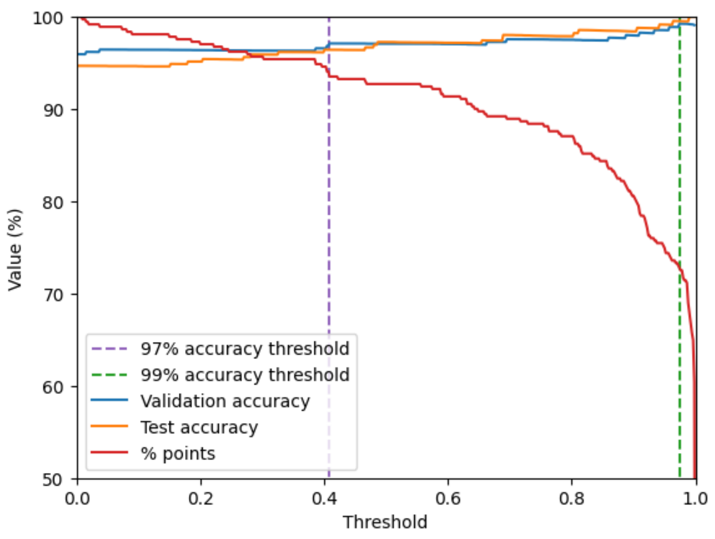

Figure 7.2: **Accuracy plotted against the confidence threshold**. The red line represents the proportion of predictions that satisfy the threshold value. The blue and orange lines represent the validation and test accuracy, respectively, for the thresholded subset of predictions. The dotted lines indicate the threshold value at which a certain validation accuracy is reached. A 99% accuracy threshold was used in this study.

Using this method, we generated an extensive collection of coordinates. The ACTN2 patches were discarded after being used for label prediction, and the coordinates were used to grab the corresponding brightfield patches. The brightfield images, however, followed a particular pattern (Appendix B).

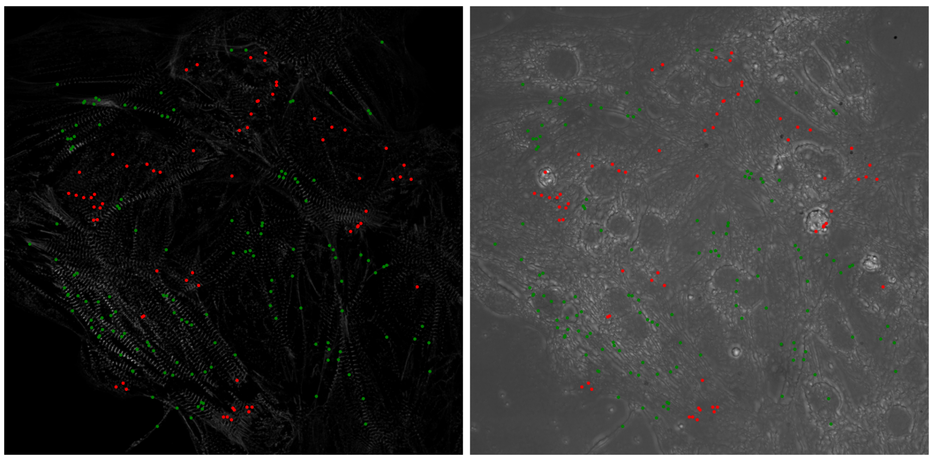

Figure 7.3: **Example of automatically generated annotations.** Red and green points indicate the sampled locations, where red represents 'bad' regions and green represents 'good' regions. **Left**: ACTN2 image with annotations generated by the trained ACTN2 classifier. **Right**: Corresponding brightfield image with the same annotations mapped from the ACTN2 image.

Among the 100 selected brightfield images, 57 had an average pixel intensity of around 4,000 with a standard deviation of approximately 300, whereas 43 had an average intensity of around 40,000 and a standard deviation of roughly 1,000. It was essential to address this disparity to effectively train on the brightfield images. We achieved this by selecting the low-intensity images from the training set as references since they formed the majority. We then calculated their average histogram and used it to match all images, ensuring that the images maintained a consistent intensity distribution.

After mapping the labeled ACTN2 coordinates to the brightfield data, we obtained 14,824 patches. Of these, 4,817 were classified as 'bad' patches, and 10,007 were classified as 'good' patches. An example of how predictions on an ACTN2 image translate to brightfield annotations is depicted in Figure 7.3.

7.2 Splitting the Data

To prepare the dataset for training, we divided the data into training, validation, and test sets with an 80%, 10%, and 10% split, respectively. We assigned all patches from a single image to the same split to prevent data leakage and maintain class balance across splits. Given the larger number of images (100) compared to the ACTN2 dataset (18), we used the `differential_evolution` function from SciPy [27] to optimize the split. This approach ensured that each subset closely matched the target proportions and maintained class balance. The split was performed before data augmentation to avoid data leakage. The distribution of classes in each split is displayed in Table 7.1.

Split	Nr. Images	Bad Count	Good Count	Split Proportion
Train	79	4817	10007	0.8
Validation	10	751	909	0.09
Test	11	1103	945	0.11

Table 7.1: **Distribution of class samples across the training, validation, and test sets of the generated Brightfield data.** The dataset was split at the image level to avoid data leakage and maintain class balance. The table shows the number of samples for each class and the proportion of the total dataset represented by each split rounded to 3 digits.

7.3 Augmentation

Since there was an inherent bias towards sampling patches containing successfully differentiated cardiomyocytes (protocol has 78% efficiency), data augmentation was used to balance the classes. The 'bad' class was augmented three times, using the same transformations used for the ACTN2 model to sample randomly from. The 'good' class was augmented once, effectively doubling this class. After augmentation, we had 19,268 bad patches and 20,014 good patches, which is close to a 50% class distribution and comes to a total of 39,282 patches.

7.4 Preprocessing, Tuning & Training

The same preprocessing steps were applied to the brightfield patches as with the ACNT2 model. Each patch was upscaled and stacked three times along its channel dimension to meet the EfficientNetV2 input shape.

We used the optimized hyperparameters from the ACTN2 models as a guideline for the brightfield model, expecting it to perform well given the similarity of their tasks. We manually experimented with the learning rate, dropout rate, and model architecture (also considering V2B1 and V2B2).

The training process followed the same protocol as the ACTN2 model. Binary cross-entropy was used as the loss function, which is appropriate for the binary classification task. The RMSprop optimizer was configured with a linear warm-up of the learning rate over the first five epochs, followed by an exponential decay at a rate of 0.9. The models were trained for up to 100 epochs to allow enough time for full convergence. Early stopping was applied with a patience of 10 epochs, monitoring validation loss.

7.5 Ensemble and Test-Time augmentation

Identical to the ACTN2 classifier, final predictions were created by combining the trained models and applying test-time augmentation with a predetermined value of 11 augmentations. We trained multiple models intending to all contribute equally to the ensemble model by averaging their predictions. These models all used the same manually tuned hyperparameters since only a very limited range of values was found that resulted in stable, effective learning of the networks. We hoped that the random initialization of the networks and the random data shuffles during training would result in diverse models that benefit from the ensemble.

7.6 Creating a Region map

Using the trained model ensemble with test-time augmentation, an entire microscopic brightfield image could be predicted at once by taking the patches around every pixel and predicting its class, i.e., a region map. However, predicting every pixel is very time-consuming; therefore, a resolution was chosen to create predictions for every nth pixel. This resolution was set to $n = 20$, meaning that for each image (1776x1736 pixels minus the edge of 48 pixels), 6888 predictions should be performed. However, we could decrease this number even more by only predicting foreground pixels. These included the coordinates that were within an Euclidean distance of 5 pixels of a foreground pixel as determined by Li thresholding. This was done to get a more dense region map, assuming that the predictions of these non-overlapping pixels would affect model performance minimally.

Chapter 8

Brightfield Classifier Results

This section showcases the results of the brightfield classifier. Eight classifiers were trained on the brightfield data, sharing the same manually tuned hyperparameters. Their predictions were ensembled to hopefully increase their performance. However, it turned out that the accuracy and loss of one model outperformed the combined models. Therefore, the results of that particular model are given solely. First, the hyperparameters used for training the models are presented. After, the training process is shown. Then, the confusion matrix for the test set is given, giving more insight into the class-specific performances. Finally, a region map is shown for one of the brightfield test images, with a region map for the ACTN2 image to act as the ground truth.

8.1 Tuning Results

Table 8.1 shows the hyperparameters after manual tuning for stability and accuracy. The tuned hyperparameters closely follow the configuration of

Hyperparameter	Value
Patch Size	96
Histogram Equalization	False
Scaling	normalize
Learning Rate	1e-06
Dropout Rate	0.4
Momentum (RMSprop)	0.81
Model Architecture	V2B1

Table 8.1: Hyperparameter configuration used for training the brightfield models.

model 7 (V2S 4) from the ACTN2 models. They differ in the learning rate, dropout rate, and model architecture. The brightfield model uses a lower learning rate of 1e-6 and an increased dropout rate of 0.4. Most notably, it uses the V2B1 architecture instead of the V2S architecture.

8.2 Training Results

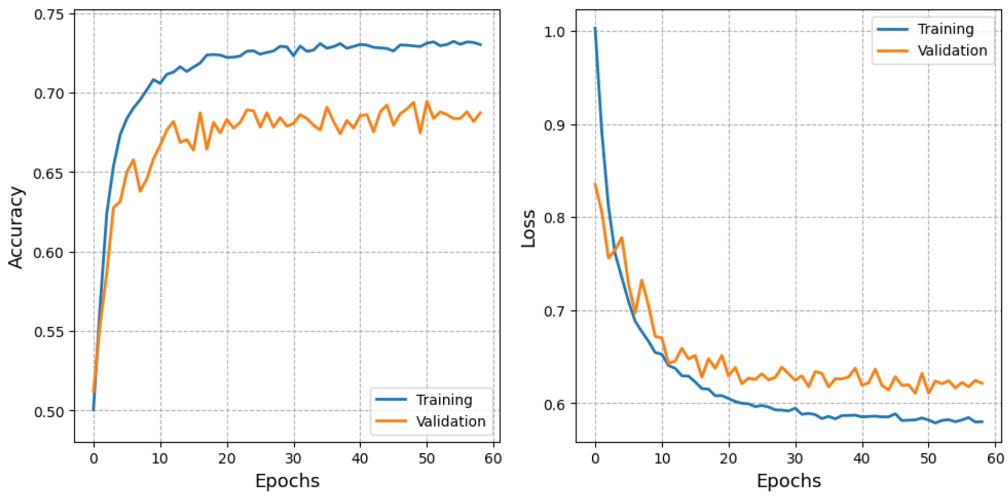

Figure 8.1: Training and validation accuracy/loss curves of the brightfield models.

In figure 8.1, the training progress of the best-performing model is displayed. A much lower accuracy is achieved of approximately 69%, compared to the ACTN2 model (95%). After 20 epochs, almost no increase/decrease is seen in the accuracy/loss. The training and validation curves roughly follow the same shape, with the training set achieving a higher final accuracy.

8.3 Confusion matrix

The best-performing model was applied to the test set for final evaluation. As before, 11 test-time augmentations were applied to the classifier. Figure 8.2 shows the confusion matrix. It can be seen that the brightfield model tends to have more trouble assigning the proper class when presented with 'good' regions. To be more specific, the model is correct on only 56% of the 'good' patches, while it achieves roughly 91% accuracy on the 'bad' patches.

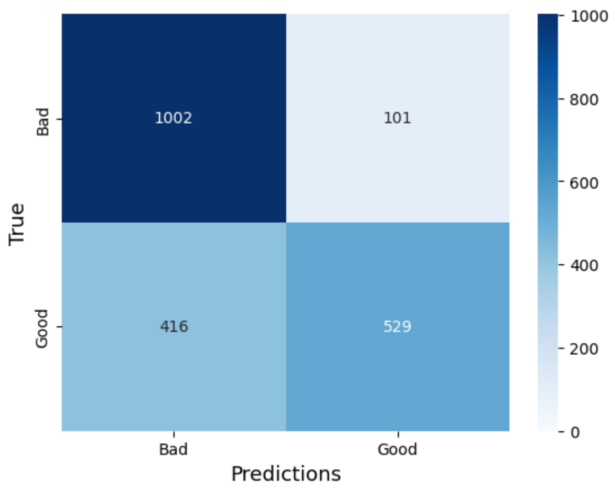

Figure 8.2: **Confusion matrix of the best-performing brightfield model** on the total test set, using test-time augmentation.

On the test set, the final brightfield model achieved an accuracy of **74.756%** using test-time augmentation.

8.4 Region Map Results

As described in section 7.6, a region map was created using the brightfield classifier. Only foreground pixels of the brightfield image were predicted, which were determined through Li thresholding. To better visualize the results, two additional region maps were created: the ACTN2 ground truth region map and a plot showing the agreement between the two region maps.

Figure 8.3 shows the region maps for a single case example brightfield image. The brightfield region map is shown in the top left with the ACTN2 region map in the top right[1], which serves as the ground truth in this case. The bottom row shows the blank brightfield image on the left for reference, with a region map showing the agreement between the two predictions on the right. The brightfield region map contains many 'bad' regions, whereas the ACTN2 model (ground truth) predicts considerably more portions of 'good' regions. This aligns with the observations made in the confusion matrix, where it became apparent that many good areas are confused with bad areas. The agreement plot is primarily green and yellow regions. The green regions

[1]Appendix A, shows a region map for the ACTN2 ensemble compared with its ground truth (expert annotations)

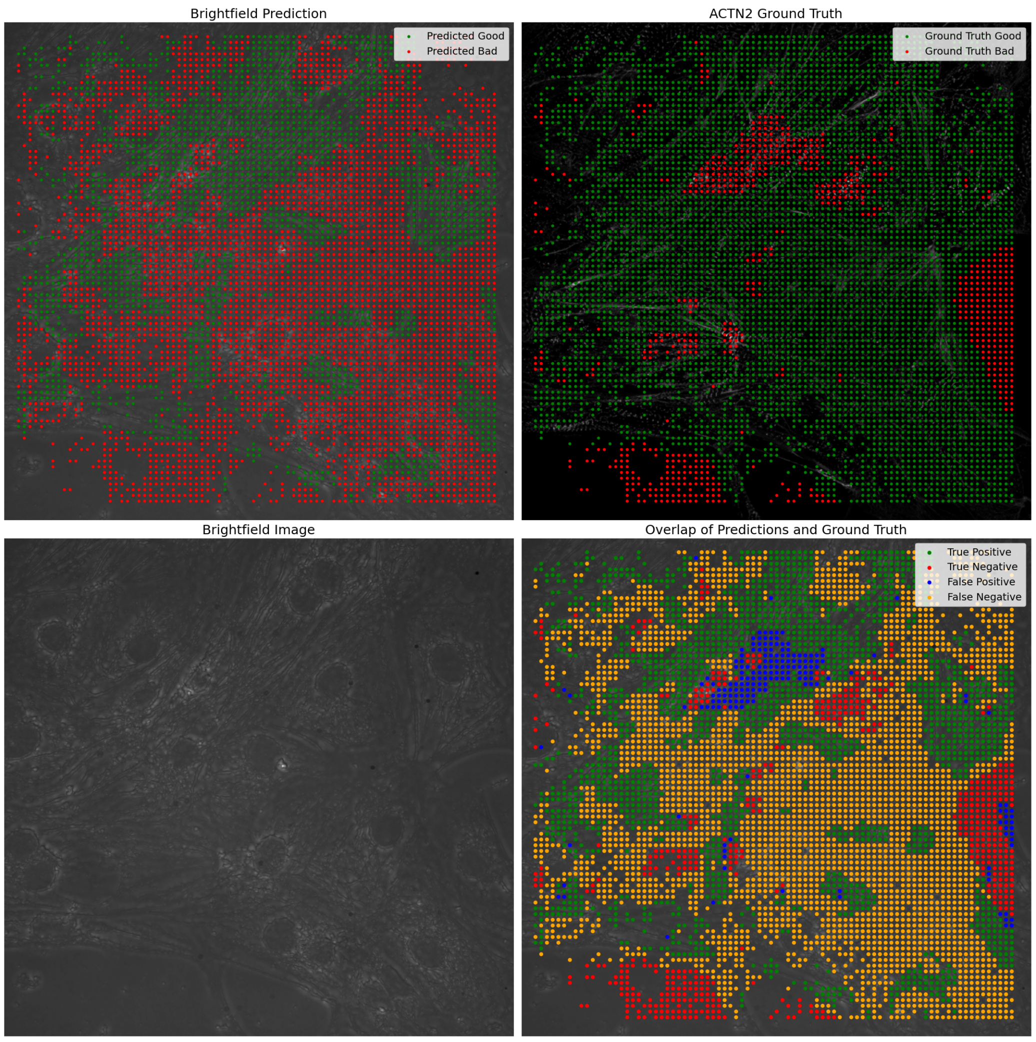

Figure 8.3: **Example of a region map generated by the Brightfield model.** The map highlights 'good' (green) and 'bad' (red) regions of differentiated cardiomyocytes in a test image.

denote where the brightfield model correctly predicts the 'good' class. The yellow region consists of the predictions that are predicted as 'bad' but are, in fact, 'good.'

Chapter 9

Discussion

9.1 Discussing The ACTN2 Model

9.1.1 Tuning Process

The top hyperparameter settings were chosen from the Bayesian optimization process. However, these tend to have very similar values as the Bayesian Optimization algorithm narrows the sampling range as it finds better settings. The values are especially similar when the initial samples don't effectively cover the entire search space. Therefore, choosing the top configurations might not be the best method of selecting models that form the final ensemble, as ensembles generally thrive through models with different strengths and weaknesses. Instead of picking only the top-performing hyperparameter settings, we could consider selecting configurations that perform reasonably well but are located in other regions of the hyperparameter space. This was done to a certain extent by including the fourth-best hyperparameter configuration for the V2S model. This proved useful as this configuration contributed to the best-performing ensemble, even though its performance was worse than the other configurations.

Interestingly, histogram equalization was not used in any of the configurations. As histogram equalization is known to improve contrast, it was expected to improve feature extraction by the CNN. Also, a patch size of 96 pixels was used in most configurations. Logically, the larger the context, the better the prediction; however, the problem also increases in complexity. Considering that 96 pixels is a rather large patch size, we expected the smaller patch sizes to perform better. Surprisingly, this was not the case. Section 9.1.2 gives our speculated reason for the tuning process to result in choosing these hyperparameters.

All hyperparameter configurations get a score reflecting how good the configuration is. Based on the received scores, tuning models, like Bayesian Optimization, choose their next configuration. By default, the scoring function takes the smallest (largest when maximizing) validation loss found thus far in the tuning process as its scoring of the hyperparameter configuration. This can incorrectly assign high scores to highly unstable configurations, and that got lucky with one low validation loss. It's hypothesized that these models typically find sharp local minima and thus generalize poorly to new data (as briefly mentioned in section 5.2.2).

A solution could be to design a custom scoring function that considers the network's stability. However, we speculate that this problem could be rooted in a noisy validation estimate and thus can be prevented by focusing on fixing that instead.

9.1.2 Training Process

The retraining of the models resulted in noticeably different performances compared to the tuning results. There is expected to be some variance in performance after retraining using the same hyperparameters since each network is initialized randomly. However, in this case, the difference is quite large (sometimes up to 2% drop in accuracy), and the models were less restrained in their training time. Perhaps we should have relaxed the early-stopping behavior, now set to 10 epochs. Since most models were already close to an accuracy of 1, the gap between validation and training accuracy could not increase anymore by getting better on the training set. As long as the model would not overfit and cause an increase in validation loss, training for longer was fine and might, in some cases, be desirable. Aside from this, typically, the retraining does not have a performance difference this large. Therefore, we suspected something else might be the root problem.

A significant limitation in our experiments was the lack of diversity in the ACTN2 validation and test sets. We split the dataset at the image level to prevent data leakage from overlapping patches. This resulted in the validation set containing only one image and the test set containing just two. Likely, such small sets do not capture the variability in the data, making it difficult to evaluate the model's actual performance reliably. This is problematic because a small validation set can skew the hyperparameter tuning process. This was observed during tuning, where the Bayesian Optimization algorithm consistently sampled configurations that resulted in unstable training dynamics characterized by fluctuating loss curves. This noise misled the algorithm into favoring hyperparameter settings that appeared optimal in a noisy context but were suboptimal. Similarly, using only two images for the test set may lead to an over- or underestimation of the model's true accuracy. The same pattern could be observed during retraining, where a few models suffered from significant drops in performance in between epochs.

We propose three solutions. The first solution would be to increase the number of images in the validation and test sets, even if it would be at the expense of some training data. Another option would be to increase the validation and test sets through the same augmentation techniques used to increase the training size. Lastly, one could use k-fold validation. In k-fold cross-validation, the total dataset is split into k subsets of equal size. Then, each subset is used once as the validation set, while the rest serves as the training set. The performance is averaged between all validation sets to provide a more robust estimate for the tuning algorithm.

9.1.3 Ensemble & Test-Time Augmentation

Implementing test-time augmentation in our experiments resulted in a noticeable decrease in the confidence intervals of the model predictions and an increase in average accuracy as the number of augmentations grew. This trend suggests that test-time augmentation effectively enhances the robustness of the model's predictions. By evaluating multiple augmented versions of each test image and aggregating the results, the model becomes less sensitive to anomalies that could affect a single prediction.

The best-performing model ensemble consisted of two V2S models. However, this success did not translate to the test set, where the ensemble's performance was significantly lower than other ensembles (grey lines in the background). This discrepancy indicates that the validation set did not serve as a reliable indicator of the ensemble's effectiveness on new data. It highlights the risk of overfitting to a small validation set, which can mislead the model selection process. Nonetheless, it still performs better than not using an ensemble, assuming the combined validation and test sets are representative of the data distribution.

9.1.4 Confusion Matrix

The confusion matrix in Figure 6.3 displays the performance of the final ACTN2 model on the test set using 11 test-time augmentations. While the model performs well overall, it appears to struggle slightly more with correctly classifying 'bad' samples, misclassifying some of them as 'good'. One possible explanation is that 'bad' patches may have less distinctive or more variable features, making them more challenging for the model to classify accurately. However, given the small and limited validation and test sets, it is difficult to determine if this issue is consistent or specific to these images.

Further investigation with a larger and more diverse dataset is needed to confirm whether the model generally has difficulty with 'bad' patches.

Additionally, since human experts made the annotations, there may be some noise or subjective bias in the labels. Human annotations can vary between annotators, where even the Allen Institute's research demonstrated disagreement between experts on similar tasks. This inconsistency might contribute to the misclassifications observed. It is also possible that the model's errors occur mainly on samples where even human experts might disagree on the classification, making it inherently challenging to achieve higher accuracy without simply modeling annotator bias.

9.2 Discussing The Brightfield Model

9.2.1 Generation of Brightfield Annotations

Only regions clearly belonging to one class were annotated for the ACTN2 pattern dataset. However, when generating data points by randomly sampling foreground pixels, we could have selected unclear points that do not distinctly belong to a particular class. Therefore, the 99% accuracy achieved after thresholding does not necessarily translate to this new data. Given that brightfield images are inherently noisier, uncertainty about data quality is undesirable.

Another issue with the current data generation method relates to the confidence threshold. Applying the threshold assumes that the model's confidence values are meaningful; the model's probabilities accurately reflect the likelihood of correctness. While thresholding resulted in 99% accuracy on the original validation set, there is no guarantee that this level of accuracy holds for the newly generated samples. Additionally, the threshold might reduce the diversity of the samples, thereby affecting the quality of the training data.

Finally, although we attempted to predict only foreground pixels using Li thresholding, the technique is imperfect. It might have included patches that should have been classified as background. We hoped the model would still classify these patches as 'bad', but this remains an assumption.

A simple solution to these problems could be to have experts validate some of the generated samples.

Early in experimentation, an observation was made that there were two distinct groups of brightfield images. One group had a very low mean with a small standard deviation, and the other group had a high mean with a large standard deviation (Appendix B). This was expected to not hinder the model's performance as long as these statistics were the sole differentiators between these two groups. However, the model did not perform well using this data, even after preprocessing steps and manual tuning. It was discovered that matching the histograms to the average histogram of one of the two groups in the training set fixed this issue and enabled the model to reach better performance. This suggests that the difference between the two groups is not solely the statistics but may be sampled from different distributions, possibly due to imaging conditions. Further analysis of the images could reveal the origin of this dissimilarity. An alternative solution could have been to use data from only one of the two groups, but this was avoided to prevent data exclusion.

9.2.2 Tuning Process

We initially used the hyperparameters from the ACTN2 ensemble as a guideline for manually tuning the brightfield classifiers. However, many configurations led to unstable learning, where the model heavily focused on the training set. This would cause the validation loss to fluctuate heavily and eventually flatline on a high loss value. We manually changed the ACTN2 guideline hyperparameters until we found stable settings, as shown in Table 8.1. We reduced the learning rate from 5e-5 (used in the ACTN2 models) to 1e-6. Normalization was the only method that yielded good results among the scaling methods. We also noticed that a relatively high dropout rate of 0.4 most successfully prevented overfitting. Higher dropout rates hindered the training too much. Ultimately, we found that the smallest model architecture, EfficientNetV2B1, performed the best. This is likely due to the noisy nature of the brightfield images. We suspect that the larger models were overfitting to the noise and outliers. In contrast, simpler models are naturally better at generalization, with the risk of the models underfitting.

9.2.3 Ensemble Performance

We repeatedly trained a classifier on the same set of hyperparameters (found after manual tuning) eight times. After conducting a brute-force search to explore different model combinations, we found no ensemble improved the predictive performance. We speculate that the random initialization of model parameters did not introduce sufficient diversity among the classifiers. This is likely because all models start their training using the same pre-trained parameters for a large portion of their network. Only the last manually added layers introduce randomness, which is roughly 7000 parameters.

9.2.4 Training Process

The training curves show that the training and validation accuracies follow a similar pattern, with the validation accuracy slightly lower than the training accuracy. This indicates no apparent signs of overfitting, as the validation loss does not increase while the training loss decreases. After training, the model's performance without test-time augmentation achieved approximately 69% accuracy. Although the learning was stable and did not show signs of overfitting, the model might be underfitted. While an effective hyperparameter configuration was not found, a slightly more complex model, such as V2B2, could potentially achieve better accuracy with the proper configuration. This could be confirmed through automatic hyperparameter tuning.

9.2.5 Confusion Matrix

For a binary classification problem with balanced classes, random guessing would yield approximately 50% accuracy. In our case, with an accuracy of 74.756% on the test set, the model can distinguish the two classes to some extent. However, in Figure 8.2, the model performed considerably worse on the 'good' class, with near-random guessing performance for this particular class. Nonetheless, from a practical viewpoint, the low number of false positives is a valuable property of the model. Whenever the model predicts a region to be positive, according to the test set results, the prediction is correct with a probability of 80.9%. This means that when the pipeline is extended to move the microscope to good regions automatically, it will most likely correctly identify a good region. The only drawback is that it may potentially miss areas that are even better differentiated.

9.2.6 Region Map

We observe two interesting things in figure 8.3. First, the cardiomyocyte nuclei—visible as darker, circular-shaped objects in the bottom-left of the brightfield image—are frequently misclassified as 'bad' regions by the brightfield model. In contrast, the ACTN2 model correctly identifies these nuclei as healthy components of the cells. While this discrepancy might initially suggest a shortcoming of the brightfield model, it is important to consider that these nuclei are not visible in the ACTN2-tagged images. Consequently, the brightfield model faces a classification challenge that the ACTN2 model does not encounter. Since the sampling of data points for generating the training set was random and the nuclei occupy a relatively small portion of the images, the brightfield model likely had few examples of nuclei regions to learn from. It is also possible that the ACTN2 model assigned lower confidence scores to these regions, causing them to fall below the confidence threshold and thus be excluded from the generated dataset.

Second, applying Li thresholding results in a different selection of foreground pixels in the brightfield image compared to the ACTN2 image. Some regions that appear as background in the ACTN2 image present distinct features in the brightfield image. This discrepancy leads the brightfield model to make predictions on areas for which it has not been trained, increasing the likelihood of misclassification. Considering that, during our experiments, we noticed that the ACTN2 model consistently successfully predicted 'bad' for dark (background) regions, using the brightfield image as the primary reference for foreground pixel selection may be beneficial. By doing so, pixels that appear bright in the brightfield image but dark in the ACTN2 image would be labeled as 'bad' during data generation. This adjustment could enable the brightfield model to effectively learn which patterns to ignore in the image, improving its classification performance.

Chapter 10

Conclusion & Future Work

This thesis presented a machine learning pipeline designed to automate the assessment of cardiomyocyte differentiation quality using brightfield microscopy images. The primary goal was to reduce the reliance on time-consuming manual evaluations and to assist researchers in quickly identifying the quality of cardiomyocytes.

Our approach involved a two-stage classification pipeline. First, to accommodate the absence of brightfield data, we trained a classifier on ACTN2-tagged images with expert annotations made available by the Allen Institute. By training the ACTN2 classifier, we could generate reliable annotations for ACTN2 images for which corresponding brightfield images were available. These generated annotations served as the ground truth for training the brightfield classifier, which was used to map regions of well- and poorly differentiated cardiomyocytes in the brightfield images.

The ACTN2 classifier achieved high accuracy (roughly 95% on the test set), demonstrating that machine learning models can effectively replicate expert evaluations of cardiomyocyte quality using ACTN2-stained images. After applying a dynamic confidence threshold, an accuracy of 99% was reached on the test set by excluding only 28% of the annotations. In contrast, the brightfield classifier faced challenges due to the noisy nature of brightfield images and potential biases in data generation. It showed moderate accuracy (roughly 75% on the test set) and tended to misclassify well-differentiated regions as poorly differentiated.

Several limitations affected the performance of our final model. The small and limited validation and test sets for the ACTN2 classifier most likely formed the most significant performance bottleneck. It may have prevented the classifier from reaching a much higher accuracy and thus affected the quality of the generated brightfield data. Additionally, time constraints limited our ability to further experiment with the brightfield model, including implementing automatic hyperparameter tuning.

Despite these challenges, our work provides a foundation for future efforts to automate cardiomyocyte quality assessment using readily accessible imaging techniques. By reducing reliance on manual evaluations and expensive fluorescence methods, this approach has the potential to accelerate research in stem cell therapies and regenerative medicine.

Future work should address the limitations caused by the small validation and test set sizes of the ACTN2 pattern dataset. Implementing k-fold cross-validation could provide a more reliable assessment of the model's performance by using the entire dataset for validation across different folds. Additionally, more advanced data augmentation techniques [5] can increase the dataset's diversity. Methods such as erasing transformations, generative adversarial networks (GANs) [2], and feature mixing strategies [23] could help expand the dataset, making it robust to input variations and adding entirely new samples. The same could be done for the brightfield dataset.

A potential improvement is to apply a similar tuning process to the brightfield classifier as was used for the ACTN2 classifier. This also allows us to determine if larger models can achieve stable learning on the brightfield dataset. If successful, it would enhance model complexity and address potential underfitting.

Since the brightfield ensemble model did not outperform the best single model, incorporating different neural network architectures into the ensemble could be beneficial. Architectures such as ResNet, DenseNet [8], or Transformer-based models [26] might extract complementary features, improving the ensemble's overall performance.

Regarding the ensemble, one could experiment with more advanced ensemble techniques. Due to the relatively small validation set size for the first classifier, this was not feasible in our case. Using a previously proposed solution like better data augmentation methods, a technique like model stacking might significantly improve the results. This might already apply to the brightfield models, where previously, no ensemble combination existed that improved relative to the individual model performances.

Lastly, integrating the brightfield classifier with the microscope's control system to automatically identify optimal imaging regions could further reduce the need for manual intervention. This integration would enable real-time guidance of the imaging process, focusing on areas most likely to contain well-differentiated cardiomyocytes. Implementing such a system would require overcoming challenges related to real-time image processing, which still needs to be fully explored.

Bibliography

[1] Ahola, A., Belay, B., Wählby, C., Hyttinen, J.: Label-free estimation of sarcomere orientation from brightfield microscopy images of induced pluripotent stem cell derived cardiomyocyte nuclei. In: 2022 Computing in Cardiology (CinC). vol. 498, pp. 1–4 (2022). https://doi.org/10.22489/CinC.2022.359

[2] Biswas, A., Md Abdullah Al, N., Imran, A., Sejuty, A.T., Fairooz, F., Puppala, S., Talukder, S.: Generative adversarial networks for data augmentation. In: Data Driven Approaches on Medical Imaging, pp. 159–177. Springer (2023)

[3] Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)

[4] Cao, L., Schoenmaker, L., Ten Den, S.A., Passier, R., Schwach, V., Verbeek, F.J.: Automated Sarcomere Structure Analysis for Studying Cardiotoxicity in Human Pluripotent Stem Cell-Derived Cardiomyocytes. Microscopy and Microanalysis **29**(1), 254–264 (12 2022). https://doi.org/10.1093/micmic/ozac016, `https://doi.org/10.1093/micmic/ozac016`

[5] Garcea, F., Serra, A., Lamberti, F., Morra, L.: Data augmentation for medical imaging: A systematic literature review. Computers in Biology and Medicine **152**, 106391 (2023). https://doi.org/https://doi.org/10.1016/j.compbiomed.2022.106391, `https://www.sciencedirect.com/science/article/pii/S001048252201099X`

[6] Gerbin, K.A., Grancharova, T., Donovan-Maiye, R.M., Hendershott, M.C., Anderson, H.G., Brown, J.M., Chen, J., Dinh, S.Q., Gehring, J.L., Johnson, G.R., Lee, H., Nath, A., Nelson, A.M., Sluzewski, M.F., Viana, M.P., Yan, C., Zaunbrecher, R.J., Cordes Metzler, K.R., Gaudreault, N., Knijnenburg, T.A., Rafelski, S.M.,

Theriot, J.A., Gunawardane, R.N.: Cell states beyond transcriptomics: Integrating structural organization and gene expression in hipsc-derived cardiomyocytes. Cell Systems **12**(6), 670–687.e10 (2021). https://doi.org/https://doi.org/10.1016/j.cels.2021.05.001, `https://www.sciencedirect.com/science/article/pii/S2405471221001563`

[7] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR **abs/1512.03385** (2015), `http://arxiv.org/abs/1512.03385`

[8] Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)

[9] Ioffe, S.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)

[10] Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: Generalization gap and sharp minima. CoRR **abs/1609.04836** (2016), `http://arxiv.org/abs/1609.04836`

[11] LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L.: Handwritten digit recognition with a back-propagation network. Advances in neural information processing systems **2** (1989)

[12] Li, C., Lee, C.: Minimum cross entropy thresholding. Pattern Recognition **26**(4), 617–625 (1993). https://doi.org/https://doi.org/10.1016/0031-3203(93)90115-D, `https://www.sciencedirect.com/science/article/pii/003132039390115D`

[13] Lien, C.Y., Chen, T.T., Tsai, E.T., Hsiao, Y.J., Lee, N., Gao, C.E., Yang, Y.P., Chen, S.J., Yarmishyn, A.A., Hwang, D.K., Chou, S.J., Chu, W.C., Chiou, S.H., Chien, Y.: Recognizing the differentiation degree of human induced pluripotent stem cell-derived retinal pigment epithelium cells using machine learning and deep learning-based approaches. Cells **12**(2) (2023). https://doi.org/10.3390/cells12020211, `https://www.mdpi.com/2073-4409/12/2/211`

[14] Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., Han, J.: On the variance of the adaptive learning rate and beyond. arXiv preprint arXiv:1908.03265 (2019)

[15] McMahan, B., Streeter, M.: Delay-tolerant algorithms for asynchronous distributed online learning. Advances in Neural Information Processing Systems **27** (2014)

[16] Nguyen, D.H., Van Khuat, D., Nguyen, T.H., Tran, T.A.: Predictive neural stem cell differentiation using single-cell images based on deep learning. Journal of Science and Technology on Information and Communications **1**(1), 4–10 (2024)

[17] O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L., et al.: Keras Tuner. `https://github.com/keras-team/keras-tuner` (2019)

[18] Roberts, B., Arakaki, J., Gerbin, K.A., Malik, H., Nelson, A., Hendershott, M.C., Hookway, C., Ludmann, S.A., Mueller, I.A., Yang, R., et al.: Scarless gene tagging of transcriptionally silent genes in hipscs to visualize cardiomyocyte sarcomeres in live cells. bioRxiv p. 342881 (2018)

[19] Roberts, B., Hendershott, M.C., Arakaki, J., Gerbin, K.A., Malik, H., Nelson, A., Gehring, J., Hookway, C., Ludmann, S.A., Yang, R., et al.: Fluorescent gene tagging of transcriptionally silent genes in hipscs. Stem Cell Reports **12**(5), 1145–1158 (2019)

[20] Santurkar, S., Tsipras, D., Ilyas, A., Madry, A.: How does batch normalization help optimization? Advances in neural information processing systems **31** (2018)

[21] Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems **25** (2012)

[22] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research **15**(1), 1929–1958 (2014)

[23] Summers, C., Dinneen, M.J.: Improved mixed-example data augmentation. In: 2019 IEEE winter conference on applications of computer vision (WACV). pp. 1262–1270. IEEE (2019)

[24] Takahashi, K., Tanabe, K., Ohnuki, M., Narita, M., Ichisaka, T., Tomoda, K., Yamanaka, S.: Induction of pluripotent stem cells from adult human fibroblasts by defined factors. Cell **131**(5), 861–872 (2007).

https://doi.org/https://doi.org/10.1016/j.cell.2007.11.019, https://www.sciencedirect.com/science/article/pii/S0092867407014717

[25] Tan, M., Le, Q.V.: Efficientnetv2: Smaller models and faster training. CoRR **abs/2104.00298** (2021), https://arxiv.org/abs/2104.00298

[26] Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International conference on machine learning. pp. 10347–10357. PMLR (2021)

[27] Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods **17**, 261–272 (2020). https://doi.org/10.1038/s41592-019-0686-2

[28] Wager, S., Wang, S., Liang, P.S.: Dropout training as adaptive regularization. Advances in neural information processing systems **26** (2013)

[29] Waisman, A., La Greca, A., Möbbs, A.M., Scarafía, M.A., Santín Velazque, N.L., Neiman, G., Moro, L.N., Luzzani, C., Sevlever, G.E., Guberman, A.S., Miriuka, S.G.: Deep learning neural networks highly predict very early onset of pluripotent stem cell differentiation. Stem Cell Reports **12**(4), 845–859 (2019). https://doi.org/https://doi.org/10.1016/j.stemcr.2019.02.004, https://www.sciencedirect.com/science/article/pii/S2213671119300529

[30] Zhu, Y., Huang, R., Wu, Z., Song, S., Cheng, L., Zhu, R.: Deep learning-based predictive identification of neural stem cell differentiation. Nature communications **12**(1), 2614 (2021)

Appendix A

ACTN2 Region Map versus True Annotations

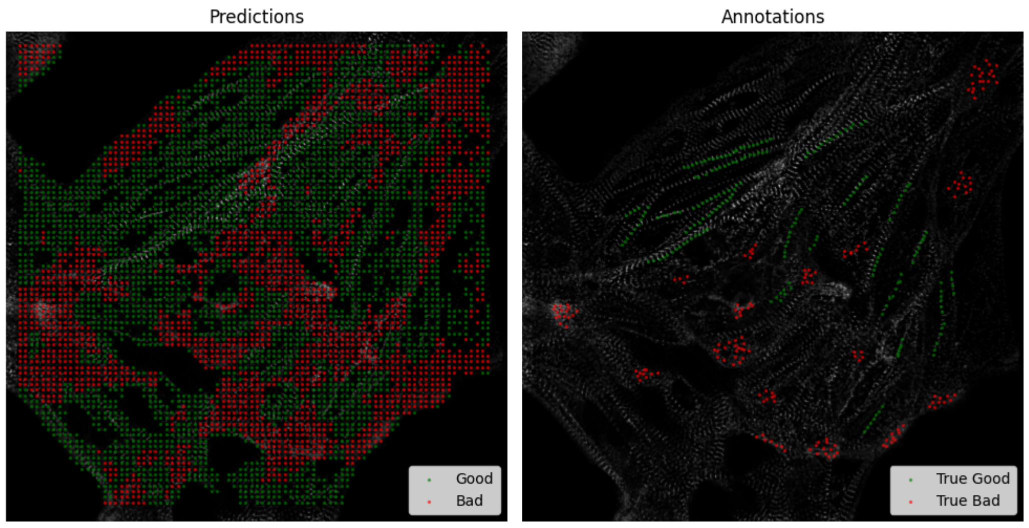

Figure A.1: **Example of ACTN2 Region Map with Compared with True Annotations.** Foreground pixels are extracted using Li thresholding.

Appendix B

Brightfield Pixel Intensities

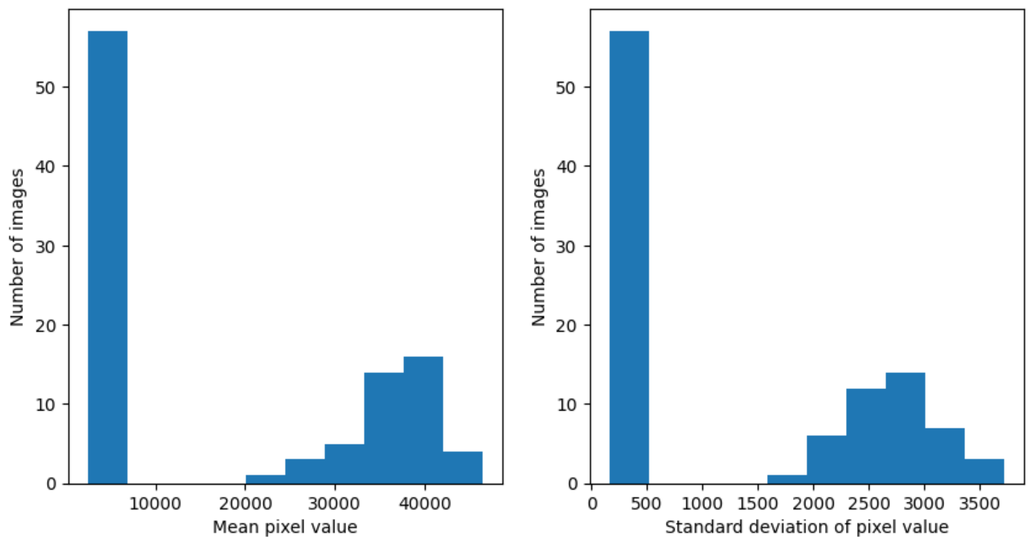

Figure B.1: **Brightfield image intensities.** There is a big disparity between some brightfield images in their brightness and contrast.

Appendix C

Hardware and Software Specifications

Specification	**Details**
Hardware	
GPU	NVIDIA RTX 2060 with 6GB GDDR6 memory
CPU	AMD Ryzen 5 3600, 6 cores, 3.6GHz
RAM	16 GB
Operating System	Windows 11
Software and Package Versions	
Programming Language	Python 3.10.11
Deep Learning Framework	TensorFlow 2.10.0
Extra libraries	Scikit-learn 1.5.1
	Scikit-image 0.24
	NumPy 1.26.4
	OpenCV 4.10.0

Table C.1: Hardware and software specifications used in this study