Universiteit Leiden

Master Computer Science

Machine Learning on diagnostic imaging data of systemic sclerosis patients based on lower-dimensional image representations

Name:	Andreea Dincu
Student ID:	s2560623
Date:	28/09/2021
Specialisation:	Data Science
1st supervisor:	Prof.dr.ir. F.J. Verbeek
2nd supervisor:	Dr. L. Cao and Dr. V. van Duinen

Master's Thesis in Computer Science

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

Abstract

Systemic sclerosis or Scleroderma is a rheumatic autoimmune disease, characterized by blood vessel abnormalities, and skin and internal organs fibrosis. Due to the complex and varied symptoms that define the disorder, the therapeutic scheme applied to each patient has to be highly personalized. In this work, we explore ways of detecting similarly developed cases, with the potential of helping with the design of individual treatments. Our input data contain high-dimensional fluorescent cellular images of endothelial cells exposed to Systemic sclerosis perturbations. Usually, this kind of analysis starts with segmenting individual cells from the input image and extracting a set of morphological characteristics or measurements for each object. These values are aggregated on an image level to create its signature. The resulting morphological profiles are finally clustered based on similarity. We will deviate from this standard approach and investigate whether compressing the input images into a lower-dimensional space can result in information-rich representations, that could benefit the final patients clustering. We demonstrate that even with a limited amount of data, we can isolate cases of Sceloderma with comparable severity levels.

Contents

Chapter 1

Introduction

Systemic sclerosis or Scleroderma (SSc) is an autoimmune rheumatic disease characterised by vascular damage (vasculopathy) and, skin and internal organs scarring (fibrosis). The interior of the blood vessels is lined by a monolayer of endothelial cells (ECs), called the endothelium. An early pathological signal in the development of Systemic sclerosis is the damage of this layer. Although SSc is not curable at the moment, advancements in the medical field resulted in the possibility of attenuating or treating some of the symptoms [1].

The goal of the current research is to investigate the methodologies of clustering Scleroderma patients. We hypothesize that sub-grouping SSc patients based on the perturbations observed at the ECs level can assist in the development of personalized therapeutic schemes. We specifically curate cellular images to showcase vasculopathy.

Our hypothesis is open-ended as there is no rigorous method to determine the practicality of the identified stratification. A medical specialist can only assess whether a specific association between patients is reasonable based on experience. Additionally, there is a multitude of unobserved factors (possibly unrelated to Systemic sclerosis) that could influence whether a therapeutic scheme would indeed benefit each individual in a group. Nevertheless, we consider the study of our hypothesis relevant, as it has the potential to unveil (subtle) similarities between samples, that could aid the design of therapeutic strategies. Ideally, in case the discovered similarities seem convincing from a medical point of view, such a research project could serve as the enabler for an extensive medical study that attempts to ground the most prominent results into solid medical knowledge.

This research is developed in collaboration with the Leiden University Medical Center (LUMC). The medical domain experts involved in the project provided a data set of high-dimensional cellular images from both Systemic sclerosis (SSc) patients and healthy controls (HC).

1.1 General approach

The analysis of the impact of a disease has on a cellular block is typically conducted at individual cell level [2]. A series of single-cell phenotypic features deemed relevant for the given research are identified and collected, with the assistance of a medical expert. These handcrafted features are further aggregated for creating a morphological fingerprint for an entire specimen. The dimension of the morphological fingerprint is equal to the cardinality of the set of features. This compressed representation is passed to a machine learning algorithm that exploits distinctive phenotypic characteristics to construct clusters of similarly developed cases. Given that there is no ground truth to be employed for guiding the clustering algorithm or evaluating the resulting sub-groups, the study can be formulated in the machine learning domain only as an *unsupervised* problem.

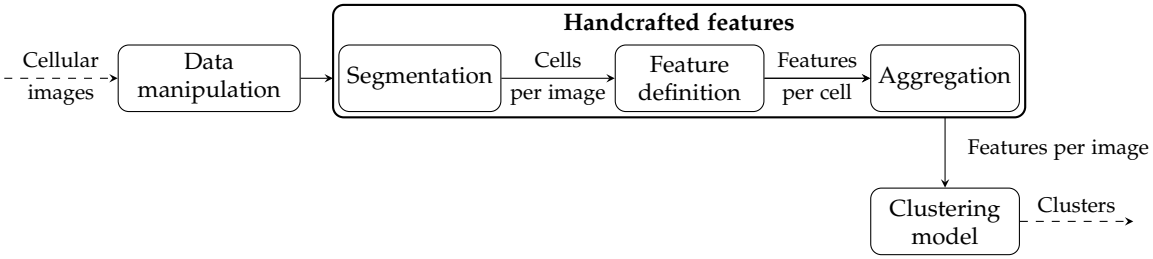


Figure 1.1: Summary of the baseline approach. Cells are segmented from each image i, $c_i = \{c_{i_1}, c_{i_2}, ..., c_{i_k}\}$, with $|c_i|$ different for each image. Feature vectors $\overline{f_{i_k}}$ are defined for each individual cells. All the feature vectors $\{\overline{f_{i_1}}, \overline{f_{i_2}}, ..., \overline{f_{i_k}}\}$ for cells in the same image are aggregated, thus resulting in one feature vector $\overline{f_i}$ per image. This feature vector is used for clustering.

This methodology was shown to yield satisfactory outcomes [3–6]. However, several remarks can be raised regarding the integrity of the derived clustering as a result of the way the morphological fingerprints are built.

Firstly, the identified grouping is naturally highly dependent on the quality and relevance of the handcrafted features. Defining a set of measurements that captures the main characteristics and uniqueness of each sample can be laborious, most often confined by the developer's intuition and knowledge of both medical and machine learning domains. Omitting just a few discriminating features can be detrimental for the results while including a plethora of features introduces the curse of dimensionality [7].

Secondly, prevalent morphological characteristics of the cells can be weakened or even lost due to the aggregation step. By employing a naive dimension-level aggregation, one assumes that the entire feature variance present in a sample can be captured with only one value. Moreover, depending on the chosen aggregation method, the summary statistics would be more or less susceptible to outliers and the overall distribution. Defining a morphological fingerprint that captures the essence of all samples might require a unique aggregation strategy per dimension. This is a non-trivial task that can easily translate into over-engineering.

1.2 Proposed methodology

In the present study, we aim to explore alternative approaches, based on the discovery of relevant patterns or features rather than the employment of a rigid set of measurements. Namely, we replace the curated morphological profiling step with a deep neural network capable of assembling lower-dimensional representations (i.e. embeddings) of the input images. This adjustment intrinsically shifts the attention from single-cell phenotypic features to the characteristics of the entire cell population present in a sample.

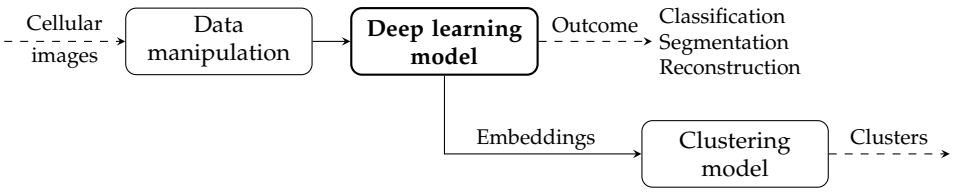


Figure 1.2: Summary of the proposed methodology for defining clusters of Systemic sclerosis patients. The deep learning model can have as objective to either classify, segment or reconstruct the input data.

Autoencoder neural networks are specifically popular for their ability to construct low-dimensional representations of the received input data. The derived embeddings retain essential information for reconstructing [8] the original data (up to a certain degree), or even constructing an improved version of it (e.g. denoising [9]). There are previous attempts trying to make image representations extracted using these networks suitable for data clustering [10]. A closely related methodology is to develop a network that instead of specializing in reconstructing the image, can rather segment desire objects out of it [11]. However, given the objectives of these networks, there is no intrinsic mechanism to ensure that discriminating features between the SSc and HC samples are preserved, as they are not significant for reconstruction. These features are nonetheless instrumental for our application. As the clustering algorithm that processes the embeddings has to be unsupervised, we must ensure that the representations sustain as much as possible the discovery of quality groups. Thus, it is preferred to construct embeddings rich in discriminating features, as it has the potential to aid the stratification.

A promising approach regarding the preservation of discriminating features is to construct a supervised classification model that can accurately distinguish between healthy and diseased samples. As the weights of final layers in the model are particularly trained to heavily influence and support the classification, we can use their activations as our encodings. Unfortunately, this method is more demanding in terms of the amount of data than an autoencoder. Given the limited amount of data that could be collected on the period of the research project, this aspect represents an impediment in the progress of the study, as detailed in Chapter 6. While experimenting with artificially augmenting our data set, we demonstrate that this method is also highly susceptible to overfitting when the variance of data is low.

We implement the segmentation, reconstruction and classification approach, extract the associated

embeddings for our cellular images, and use this representation as input for the clustering algorithm, which has to find sub-groups of the SSc patients. We ultimately assess which one of these deep learning approaches is better suited for deriving appropriate lower-dimensional representations for our study.

In the following chapters, we present both the biological and machine learning background information necessary for a reader to comprehend the entirety of the paper. We further summarise relevant related work and do a deep dive into the particularities of the data and the details of the methods used. We present a series of experiments conducted for developing and assessing the alternative pipelines, discuss their results, formulate the conclusions and future work opportunities.

Chapter 2

Cell culture and data acquisition

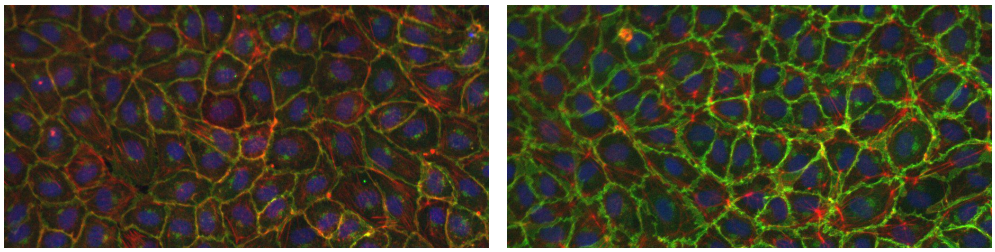
The study of the phenotypic modifications occurred at cell level in the presence of chemical perturbations gives the opportunity to advance the understanding of disease mechanisms [12]. As previously introduced, endothelium injury is a pathological hallmark in the evolution of Systemic sclerosis. We hypothesize that endothelial cell (EC) dysfunction in response to circulating metabolic or inflammatory mediators is likely the main driver of the disease. Thus, treating cultured endothelial cells (ECs) with plasma from confirmed Scleroderma patients offers access to controllable simulation of the disease. Supplementing this with healthy controls treatment enables the possibility of comparative studies. The specific type of cells used in this study are human umbilical vein endothelial cells (HUVEC), which are ECs isolated from veins from the umbilical cord. The patients with increased fibrosis are considered to be more severe cases (SSCHL).

These biological probes can be stained with fluorescent dyes and imaged using a high-throughput microscope for generating cellular images with characteristics of interest highlighted. By visually inspecting the samples, the medical specialists involved in the project have been able to identify two indicators that can be used to detect the SSc samples:

- both the cell-borders and actin fibers have a more chaotic appearance
- the level of actin inside the cell increases, while in healthy cells the actin is mainly distributed around the cell-borders

Based on these findings, we image the *actin*, *borders* and *nuclei* channels. Figure 2.1 offers an illustrative example of extremely visually distinctive confocal images of Systemic sclerosis and healthy controls.

The extracted information is suitable for further investigation on the impact that the SSc perturbation has on cell populations or for developing automated pipelines to identify the disease. The channels can be either introduced into specialized software which produces information-rich singular cell profiles (including features such as shape, intensity, texture features etc.) or into more complex machine learning



(a) Systemic sclerosis. (b) Healthy control.

Figure 2.1: Comparative example of a systemic sclerosis sample and a healthy control. Illustrative 512×1024 crops, with the 3 channels (i.e. *actin*, *borders*, *nuclei*) merged into one image.

or deep learning models which are able to directly unveil meaningful insights from the entire confocal image. An important particularity of this study is that the effects of the applied perturbation are not localized and can be observed over the entire cell population.

An exhaustive protocol on setting up such a screening assay is detailed in [13]; the cell culturing, imaging and profiling involved in our study closely follows the presented methodology.

2.1 Patients and healthy volunteers

Venous blood (K2EDTA) was obtained from Scleroderma patients ($n = 10$), and age-matched healthy volunteers ($n = 5$). SSc patients met the criteria set by ACR/EULAR [14], and were all in the active stage of the disease. Cells were removed by 15 minutes centrifugation at $2500\times$g. Platelet-free plasma was obtained by centrifugation of the supernatant at $2500\times$g for 15 minutes, and stored at -80°C.

2.2 Cell culture

HUVECs were isolated from umbilical cords obtained from the department of obstetrics at the Leiden University Medical Center, anonymized and under full consent of the parents, as previously described [15]. Freshly isolated HUVECs were cultured in EGM-2 (PromoCell C-22111) supplemented with 1% antibiotics for one passage and frozen for later use. For the plasma exposure experiments HUVECs were thawed and cultured in 1% gelatin coated T25 flasks two days prior to seeding in the 96-wells plate (Corning 4580). The glass surface of the 96-wells plate was coated according to [16]. In short, the glass surface was incubated overnight with 0,5% glutaraldehyde (Sigma Aldrich; G 400-4) in "water for injection", WFI (Ampuwa). Afterwards, wells were washed thoroughly with WFI and incubated overnight with WFI. Prior to seeding, the glass surface was incubated with 1% gelatin in PBS for 30 minutes at 37°C. The gelatin solution was aspirated and the coating was cross-linked by incubation with 0,5% glutaraldehyde in PBS for 20 minutes at room temperature. Afterwards the wells were washed repeatedly with warm PBS, and incubated with 1% Glycine in PBS for 7 minutes to block any exposed aldehyde groups. Wells were incubated with PBS for at least 4 hours before seeding the cells. HUVECs were seeded near confluency at a density of 9000 cells/well in 100µL EGM-2 medium. Cells were

cultured for 2 more days before exposure to allow for a quiet uniform monolayer to form.

2.3 Plasma exposure

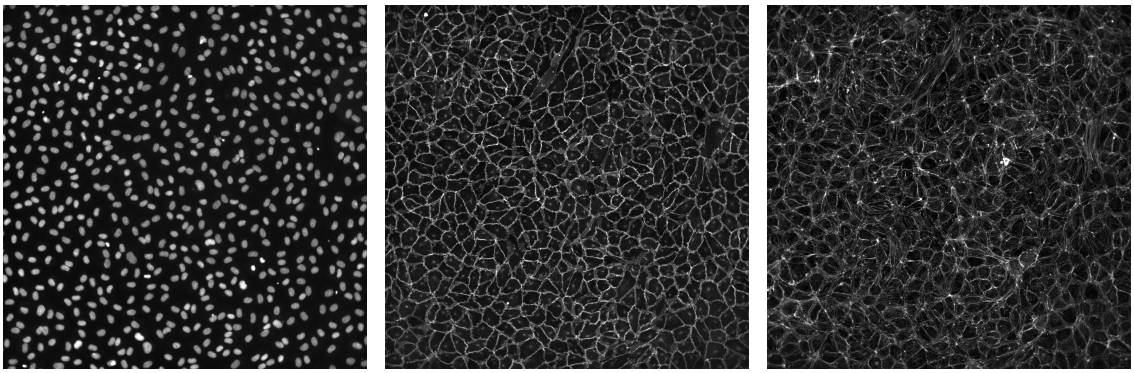
To assess the differences between endothelial cell phenotype induced by healthy and Scleroderma patient plasma, confluent HUVEC monolayers were exposed to 25% recalcified K2EDTA plasma from either SSc patients or healthy controls for 18 hours. K2EDTA plasma was recalcified by adding 0.5 µM recombinant Hirudin (ABCAM ab201396), 25 µg/ml CTI, 1.85 mM $CaCl_2$ to a K2EDTA plasma volume equaling 25% of the total sample volume. Endothelial cell basal medium (PromoCell) containing 1:100 ITS supplement (Gibco 41400) was added to obtain the final volume. 12.5 µM Compstatin was added to prevent complement activation for some experiments. At most 8 technical replicates were made for each plasma exposed HUVECs.

2.4 Immunofluorescent staining and imaging

Prior to fixation, cells were incubated with MitoTracker Deep Red FM (InVitrogen M22426) for 30 minutes. Cells were fixated by incubation with 3,7% PFA + 0,5% BSA in PBS for 15 minutes, washed, and permeabilized by incubation with 0,2% Triton X-100 in HBSS+ for 10 minutes. 10 minute blocking by incubation with 5% BSA in HBSS+ was followed by a two hour long incubation with primary antibody against VE-Cadherin (BD 555661, 2µg/ml in HBSS+ with 0.5% BSA). Cells were washed three times with 5% BSA in HBSS+, and incubated with 488 Alexa-fluorophore labeled Donkey anti-mouse secondary antibody (Invitrogen A-11001, 2µg/ml), 1:200 Rhodamine Phalloidin (Invitrogen R415), and 1:1000 HOECHST 33258 (Molecular Probes) in HBSS+ with 0,5% BSA for one hour. Cells were washed with blocking buffer, and stored under HBSS+. Max-projections of 11 z-steps at 0,8µm intervals were acquired using a high content confocal microscope (Molecular Devices, ImageXpress™ Micro Confocal) at 20× magnification (Super Plan Fluor ELWD DM, NA = 0.45 and Nikon Plan Apo Lambda; NA = 0.75), utilizing full resolution (2048 × 2048) and dynamic range (16-bit). The objective lenses were changed as the latter has an increased contrast and gathers $4\times$ as much light as the other objective.

Dapi, FITC, TRITC, and CY5 channels were used to acquire nuclei, VE-Cadherin, F-actin, and mitochondria, respectively. Four sites were imaged per well in a 2 × 2 configuration spaced 200µm apart.

Due to the low intensity of the mitochondrial staining, mitochondrial structures were indiscernible from the background signal. Therefore, mitochondria were not taken into account for the subsequent analysis. Likely, the thiol-reactive nature of the Mitotracker Deep Red FM probe led to excessive binding to cysteine residues of plasma albumin, lowering the overall availability of the probe for uptake by the mitochondria. Unfortunately, using higher concentrations of Mitotracker Deep Red FM resulted in undesired effect, e.g. mitochondrial stress, due to the high cytotoxicity of this probe.



(a) Nuclei channel. (b) Borders channel. (c) Actin channel.

Figure 2.2: Random sample from data set.

2.5 Image quality control and preprocessing

Images were checked for common artifacts, e.g. out-of-focus images, debris, clipping/saturation artifacts, and clumped cell growth, using in-house routines. A focus-score metric defined as the slope of the radial averaged log-log power spectrum- was obtained for each image, assuming a high ratio of high-frequency components to low-frequency components for in-focus images [17,18]. For each image, median, standard deviation, $\min(Q_{0.01})$, $\max(Q_{0.99})$, $Q_{0.25}$, $Q_{0.75}$, and total intensity values were computed. Results were plotted in histograms comprised of all images for a given channel and plate to assess outliers [18]. Visual inspection of outliers was performed to assess image quality and the presence of artifacts before discarding the image. Sites that contained failed images were discarded to prevent missing values.

To correct for uneven illumination and vignetting, artificial background images were constructed for each channel and plate separately, using in-house routines. To cope with a small number of images and/or the presence of bright structures, an iterative approach was used, loosely based on the approach in [19]. In short, bright areas or debris was iteratively filtered out by first identifying areas with intensity values $3\times$standard deviation above the local median value, and assigning these areas the intensity value of the direct neighborhood. This procedure was repeated for several iterations, after which the median weighted average of the images was computed and smoothed by a Gaussian filter. The correction image was constructed by dividing the mean background value by the smoothed artificial background image. All images in the corresponding channel and plate were multiplied by the correction image to obtain the illumination corrected images.

All these preprocessing steps were applied in the laboratory, before the modelling phase was even initiated.

2.6 Data set composition

After filtering the sites that contained artifacts, the final number of samples that the data set contains is 401. The final distribution of classes is 64% SSc samples and 36% controls.

Each site is represented as an image with shape $3 \times 2048 \times 2048$, with the 3 channels representing the *actin* fibers, *borders* and *nuclei* of the cells. Figure 2.2 illustrates a random example from the final data set.

Based on the opinion of medical experts, our collection includes some SSc samples in which the disease markers are highly distinguishable while in the majority of cases recognizing the examples originated from patients and from the controls is quite challenging.

Chapter 3

Machine Learning background

Machine learning is a branch of Artificial Intelligence that studies data-driven algorithms which try to mimic human reasoning. The performance of these procedures is gradually improved based on the knowledge extracted from input data.

In this chapter, we are going to discuss the machine learning concepts needed for a complete understanding of the methods used in the current study. We start by covering basic techniques of introducing data into the system. Afterwards, we present neural networks, a sub-type of machine learning algorithms that serve as our principal method of constructing image embeddings. Because the data set that we are using is rather small for the requirements of such techniques, we discuss ways to make neural networks applicable in our scenario. We present two model training paradigms used in our pipeline and evaluation tactics. We introduce data compression and the theoretical details behind the three approaches that we used for implementing it. Finally, we discuss some clustering methods relevant to our study.

3.1 Data usage

Typically when training any type of machine learning algorithms, the data has to be divided into three sections:

- a training set - used for training the parameters of the model;
- a validation set - used for assessing the quality of last training epoch;
- a test set - used for defining the actual performance of the model; this set must contain only examples unseen during the training process.

Cross-validation is an additional technique that aims to eliminate any bias that can occur due to a fortunate split of the data set. This procedure is commonly applied for ensuring the correctness and generalization of a model. The idea behind it is to first isolate your test set, and then train your model

with different train and validation splits. The final performance of the model is equal to the average score of all the created instances.

3.2 Training procedures

Machine learning algorithms can be tuned according to two types of learning procedures, supervised and unsupervised learning. **Supervised learning** is a technique in which an objective function is optimized according to data labelled for a specific task. The algorithm is said to have successfully converged when it is able to correctly generalize its knowledge on unseen examples. This methodology is useful in problems such as classification or segmentation. In **unsupervised learning**, the model has to uncover hidden structural components or patterns in unlabeled data, without receiving any ground truth. Examples of tasks that employ this technique are clustering analysis and dimensionality reduction.

3.3 Performance evaluation

Evaluating the trained machine learning model is a vital part of developing a proper solution for any given problem. This evaluation can be done using metric functions. The choice of metric function is dependent on the task that the machine learning algorithm is trained to solve. However, the training procedure should not have an impact, in this case. We introduce several metric functions relevant in the following sections when discussing specific machine learning tasks.

3.4 Neural networks

Neural networks (NNs) are a special type of machine learning algorithm inspired by the connections formed between neurons in the human brain. A general scheme of these networks is illustrated in Figure 3.1.

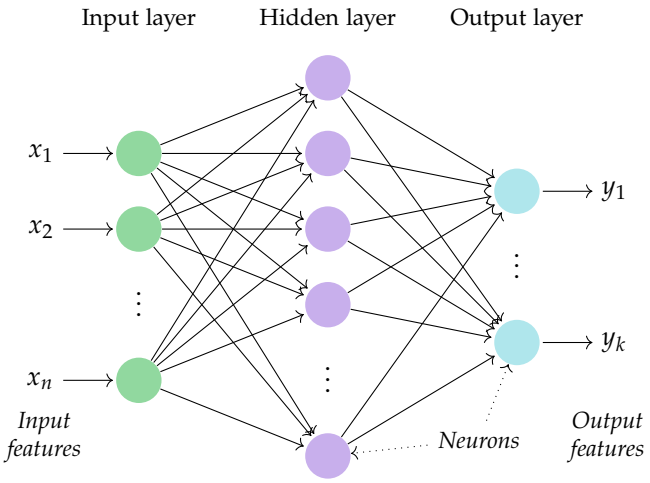


Figure 3.1: General scheme of neural networks.

Neurons are the core elements in a neural network. They are arranged in interconnected layers. Any layer of neurons presents between the first and last layer (input and output layer, respectively) is called a hidden layer. Architectures with more than 1 hidden layer are considered to be deep learning models.

Each neuron applies a non-linear (activation) function on the weighted sum of its inputs. The weights are randomly initiated, and updated during the biphasic training process of the algorithms: forward propagation and backpropagation.

Forward propagation: The data is introduced into the network in batches. In this study, the procedure employed for training neural networks follows the supervised learning approach. Thus, the data is labelled. Each batch of data is propagated through the layers in order to compute the corresponding predictions. An appropriate error is computed between the ground truth and the predicted values, using a loss function. The goal is to minimize the error by gradually moving towards a minimum of the loss function.

Backpropagation: The weights are modified by propagating backwards the error throughout the network. This update is generally conducted using a variant of Gradient Descent. The derivatives of the error with respect to each weight are computed, and the resulting gradients are subtracted from the corresponding weights. The step size with which the optimization algorithm moves is regulated by a learning rate. In the current research project, we use the Adam optimizer [20].

The magnitude of the learning rate can be fixed during the entire training process or it can fluctuate between iterations or epochs, according to a learning rate scheduler. This update is generally defined using parameters such as momentum or decay. In this project, we mainly use schedulers that update the learning rate based on the number of steps taken, when a performance metric is not improving or according to a cosine annealing schedule [21].

Neural networks can be used to model complex functions but, because of their capabilities, are more data-hungry than general machine learning approaches. A complex model trained on a small amount of data will likely **overfit** or learn a direct relationship between a specific sample and an outcome. Building a simplistic model can result in **underfitting** the data or the inability of the model to approximate the target function. Generally, there is considered to be a strong and direct correlation between the dimension (and quality) of the data set and the performance of (deep) neural networks.

3.5 Transfer learning

An important technique that enables the applicability of powerful deep learning networks on small data sets is transfer learning. The fundamental idea of this method is to train a neural network (initialized with random weights) on a large data set, preferably related to the problem you are actually trying to solve. Afterwards, (part of) these pre-trained weights are used as initialization for a model that is trained on the initial (smaller) data set. The architecture of the target model can differ from the pre-trained one,

but it must encapsulate the layers whose weights one wants to preserve. The pre-trained weights can be frozen, thus the backpropagation process will only affect the newly added layers, or they can be further fine-tuned on the small data set.

3.6 Data compression

The goal of the current research is to investigate ways to cluster Systemic sclerosis patients using models able to interpret information from the entire high-dimensional cellular images.

Directly clustering high-dimensional data comes with several challenges, which are generally referred to as the **curse of dimensionality** [7]. The clustering algorithms aim to group the observations based on relevant attributes. As the dimensionality of the data increases, identifying these meaningful attributes becomes more difficult. Furthermore, with a larger number of features the possibility of correlation increases. Thus, a common first step in clustering high-dimensional data is to apply a method for representing this data in a lower-dimensional space, in order to alleviate (part of) these issues.

The most prominent dimensionality reduction technique is principal component analysis (PCA) [22]. PCA reduces the dimensionality of the data by projecting the data points in directions that maximize the retained variance. The particularity of PCA that limits its applicability in our case is that it is agnostic to the locality, thus the positioning of the pixels in the images is completely neglected.

A different approach for encoding the images is to allow a neural network to learn meaningful representations of the data, that can aid the target task (i.e. clustering). Generally, the network learns the required encodings while training to perform a surrogate task, such as image classification, reconstruction or semantic segmentation. Activations of any of the layers in this network can be perceived as eligible embedding, as long as the shape is lower-dimensional than the input data.

We further introduce each of the mentioned surrogate tasks in the context of machine learning and explain their corresponding training details and evaluation methods.

3.7 Image classification

The classification task refers to a problem where a predictive model is trained to assign labels to the input data. This is a supervised learning task and the training of such networks follows the procedure previously detailed, where a loss function has to be employed during the training process. One of the possible candidates for this is cross-entropy:

$$H(P, Q) = -\sum_{x \in X} P(x) * log(Q(x)) \tag{3.1}$$

where X is a set of examples, $P(x)$ is the one-hot encoding of the true label of example x and $Q(x)$ is the probability of example x being each of the considered classes, as returned by the model. The cross-entropy function is easily adaptable to the binary classification case.

In the case of k-class problem, with $k > 2$, trying to maximize the likelihood of the correct class given a sample can result in a longer training time and an overly confident model, less capable of generalization. In order to alleviate this issue, one can use the label smoothing regularization technique [23]. Assuming a small smoothing factor ϵ is set, the hard targets, 1 for the true classes and 0 for the other labels, are replaced with $1 - \epsilon$ and $\epsilon/(k-1)$.

In terms of evaluating a classification model, there are several metrics that can be used. Before defining the metric functions of interest for our study, we need to introduce the following terminology.

Let's assume that we trained a model to recognize samples coming from patients with a particular disease. Thus, the algorithm classifies a sample as either disease, the positive class, or healthy, the negative class. The nature of the data is not relevant for this example. A prediction made by a model on a specific test observation can fall into one of the 4 categories:

True positive (**TP**):	correctly diagnosed positive sample.
True negative (**TN**):	correctly diagnosed negative sample.
False positive (**FP**):	negative sample labeled as positive.
False negative (**FN**):	positive sample labeled as negative.

With this terminology in place, we can now define a series of metrics that can be employed evaluation for binary classifiers. Table 3.1 provides a list of several auxiliary metrics that can be employed in such a binary classification problem.

For a multi-class scenario, the recall, precision and F-score are calculated with respect to each individual class and then averaged in order to compute the overall performance of the model.

Metric name	**Formula**	**Description**
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	Percentage of correctly classified examples.
Recall	$\frac{TP}{TP+FN}$	Percentage of correctly diagnosed samples out of all the positive examples.
Precision	$\frac{TP}{TP+FP}$	Percentage of correctly diagnosed samples out of all the positively diagnosed examples.
F-score	$(1+\beta^2) \times \frac{precision \times recall}{\beta^2 \times precision + recall}$	Weighted harmonic mean of recall and precision. β is a positive real number that indicates the importance of recall relative to precision. The most common value for β is 1, which indicates that both recall and precision are as important.

Table 3.1: Metric functions used for evaluating classification model.

To benchmark neural network classifiers, one can standardize machine learning algorithms such as support vector machines (SVMs) [24]. In the context of linear binary classification, the algorithm aims to find the hyperplane, named decision boundary, that maximizes the distances between the points associated with each class. Nonlinear classification problems can be solved using a kernel trick, that translates the data points into a feature space where the linear approach is applicable.

3.8 Image reconstruction

Image reconstruction involves compressing the input into a lower-dimensional space, and then trying to build up a higher-dimensional representation from this restricted encoding, that resembles as closely as possible the original data (see Figure 3.2). Details are lost in accordance with how restrictive the encoding space is defined to be. The neural networks that implement such an algorithm are called autoencoders. As this is a standard way to compress information using deep learning, we are interested in its suitability for embedding our cellular images.

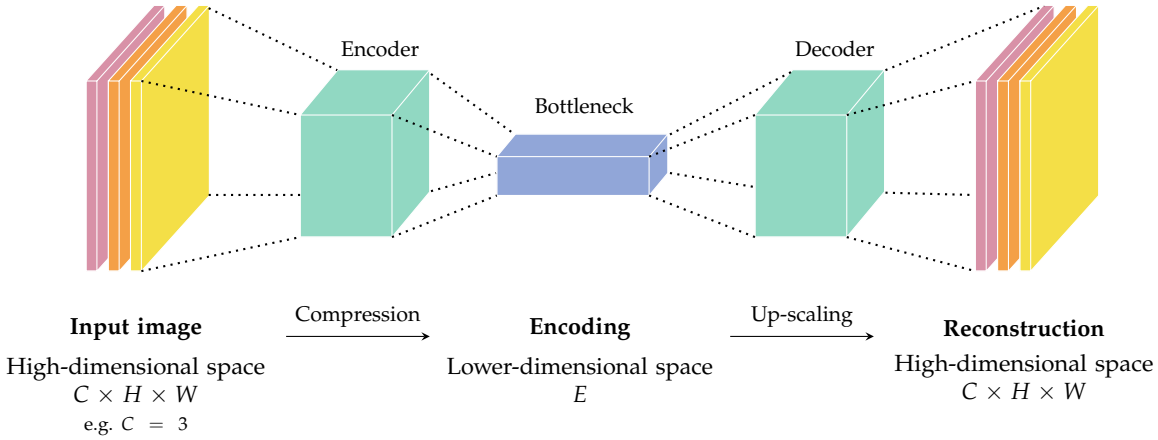


Figure 3.2: Summary of the reconstruction pipeline and general autoencoder architecture.

Even though this class of models are considered to fit into the unsupervised learning paradigm, as they do not require labelled data, their optimization process still follows the supervised learning approach. During the training of autoencoders, the distance between the original image and the reconstruction has to be minimized. A loss function that can be used is a pixel-level version of the mean-squared error (MSE):

$$MSE = \frac{1}{N} \times \sum_{n}^{N} \sum_{i}^{H} \sum_{j}^{W} (y_{n_{i,j}} - \hat{y}_{n_{i,j}})^2 \tag{3.2}$$

where N is the number of samples in the data set, H and W are the height and width of one sample, y_n is the ground truth of the n-th sample and $\hat{y}_n$ is the predicted value.

3.9 Semantic segmentation

Semantic segmentation is the process of classifying individual pixels in an input image as belonging to a specific class. We are interested in applying this technique for segmenting the actin fibers, such that the data representations computed throughout the model will intrinsically encapsulate information about the positing of the actin. This information is of interest as it exposes one of the indicators mentioned in Chapter 2 for identifying samples from SSc patients.

The output of a neural network that performs such a task is a pixel-wise probability distribution over the classes that a pixel can be a part of. In the nuclei segmentation example, one pixel can only take two values, as it can be part of either the background (0) or nuclei (1). The ground truth in this example needs to be a binary map, with a value of 1 on the positions where nuclei are placed.

Cross-entropy can be again used for calculating the segmentation performance. Another popular choice is the dice loss.

An evaluation metric that can be applied to measure the accuracy of a segmentation model is the Jaccard index or Intersection-over-Union (IoU). The standard accuracy is not a useful metric in this case as it is not suitable for problems that involve class imbalance, such as semantic segmentation. IoU has generally been defined as the intersection of the percentage of overlap between the ground truth and the prediction. In the context of semantic segmentation, IoU can be redefined at a class level as shown in eq. (3.3).

$$IoU_k = \frac{TP_k}{TP_k + FP_k + FN_k}, \quad \text{where } k \text{ is a class.} \tag{3.3}$$

The overall performance of the model is then calculated as the average of the class-level IoUs. However, there are cases, such as the binary semantic segmentation of nuclei, in which the IoU for the nuclei class might be sufficiently informative.

3.10 Clustering techniques

Clustering analysis is a machine learning technique with the goal of discovering legitimate groupings of the data. There are numerous types of clustering approaches, but, due to the specific of our study, we focus specifically on unsupervised variants such as connectivity-based and centroid-based methods.

3.10.1 Connectivity-based clustering

Connectivity-based clustering or hierarchical clustering is a class method that provides a hierarchy of samples based on their distance to each other, instead of definitive clusters. This family of algorithms is divided into two types:

1. **Divisive approach**: the algorithm starts with the complete set of data and iteratively connects samples based on their distance.

2. **Agglomerative clustering**: a single element is used as a starting point and the rest of the samples are aggregated to it, on a distance-based, to form clusters.

One particularity of hierarchical clustering is that the number of clusters is not required for training. A cut has to be made at the desired level in the hierarchy in order to obtain samples grouping. This is especially useful in more exploratory applications, where there is no strict requirement on the number of clusters.

The algorithms naturally require the definition of a distance metric, but also the choice of a linkage criterion. This criterion specifies the rule based on which connections between samples are formed. Common linkage criteria are:

1. Single-linkage criterion which combines at each step the clusters with the minimum distance between all elements in the two sets.
2. Maximum or complete linkage criterion aggregates the clusters with the maximum distance between all the elements.
3. Average linkage criterion which combines clusters based on the average distance between the elements in the two sets.

3.10.2 Centroid-based clustering

Centroid-based clustering assumes that clusters can be represented by a central vector. The observations in data that are closest to this vector are considered to be part of the corresponding cluster. This approach requires as parameters the desired number of clusters k.

K-means clustering is an algorithm representative of this approach. Firstly, k random cluster centre points (centroids) are defined. The centroids do not have to be observations in the data set. Then, the centroids are iteratively updated such that the squared distances between elements assigned to the cluster and its centre are minimized. The algorithm converges when the centroids are not updating anymore.

3.10.3 Clustering evaluations

The evaluation of clustering algorithms focuses on the degree of separation between clusters or the similarity of the samples that are grouped together.

The Silhouette Coefficient can be used to assess how well the clusters are defined when there are no ground truth labels. It is calculated using the equation $(b - a)/max(a, b)$, where a is the mean intra-cluster distance (i.e. distance between the points within a cluster) and b is the mean inter-cluster distance (i.e. distance between clusters). The Silhouette score ranges between $[-1, 1]$ with 1 defining well-separated groups, 0 indicating coinciding clusters and -1 describing wrongly assigned clusters.

Davie-Bouldin score [25] is another cluster separation metric which computes the average similarity

scores between pairs of clusters. The similarity is the ratio of intra-cluster and inter-cluster distances. The lowest the scores, the better the performance.

Homogeneity and completeness are two metrics that requires a labelled data set. The maximum homogeneity score is obtained when each cluster is formed only from elements with the same label, while completeness score is maximal when examples from a specific class are associated to the same cluster. Both metrics are based on the normalized conditional entropy measures.

Finally, V-Measure is the harmonic mean between the homogeneity and completeness scores.

Chapter 4

Related work

Morphological profiles of high-throughput fluorescent cellular images are regularly defined using aggregated values of handpicked features or measurements of segmented cells [13]. These signatures were used for various analyses from annotating genes [26], identifying relationships between genes [27, 28], discovering similarities between different compounds [29], etc. These phenotype profiles are nonetheless a collection of features that summarize, in a restrictive manner, the relevant information in the input images in a compressed space.

Automatically constructing lower-dimensional representations of images using neural networks is an established technique for dimensionality reduction [30, 31], which starts to be embraced by the biological imaging field. The encodings can be tailored towards specific goals [32, 33], according to the underlying mathematical understanding of the objective functions used to train the deep learning models. In previous work, multiple ways of computing lower-dimensional representations of different medical images were explored.

Embeddings were previously extracted using models such as recurrent neural networks [34] or siamese convolutional networks [35]. The authors used these representations as a way of retrieving images of similarly developed medical cases [36, 37]. These approaches demonstrate the similarity property that the encodings can encapsulate. Being able to define similarity measures between embeddings is vital for successfully applying clustering algorithms on these lower-dimensional representations.

As such, autoencoders are a popular option for learning feature representations of high-content fluorescent screening. Protein expression patterns of single cells were previously extracted using convolutional autoencoders. Based on them, human proteins could have been examined using hierarchical clustering [38]. The neural network was trained using a pixel-wise mean-squared objective function. Moreover, HeLa cells exposed to various RNAi genes depletion were clustered for observing similarities between the induced abnormal phenotypes. The features necessary for clustering were defined on a single-cell level using an autoencoder architecture [39], with the sum of residuals between the input and

output data as the loss function.

Morphological profiles of entire high-throughput cellular images depicting drug inducted genetic perturbations of cancer cells were extracted using an Inception-v3 classifier [40] pre-trained on ImageNet [41]. The segmentation step of singular cells was omitted. Thus, the solution considered the entire confocal images. The neural network was not fine-tuned on the target data set. The final linear layer of the model, in charge of classification, was removed, and the classification task was eventually performed using a 1-nearest neighbour approach on the activations of the penultimate layer of the network, for comparison to previous results [42]. Additionally, in [43] transfer learning approaches were tested on several neural network architectures for classifying mechanisms of action (MoAs) of breast cancer cells. However, differences in the acquisition and staining processes of confocal images can deter the pre-trained models' generalization [44]. Adapting the weights to a different domain may lead to an undesired drop in performance or to limited improvements.

Convolutional neural networks classifiers were successfully used to learn discriminative feature representations of generic visual recognition tasks [45]. Thus, applying such architectures to medical images is a promising avenue. Lower-dimensional encodings of genetic perturbations on various human cells (including HUVEC) were created using an adapted version of the DenseNet-161 classifier [46]. These embeddings were further used for varied types of analysis including hierarchical clustering of similarities between endothelial cells treated with different growth factors [47].

Chapter 5

Methods

In this chapter, we explore the exact methodology used for developing a deep learning solution for clustering Systemic sclerosis patients. Figure 5.1 illustrates the complete pipeline. Each component is elaborated on in the following sections.

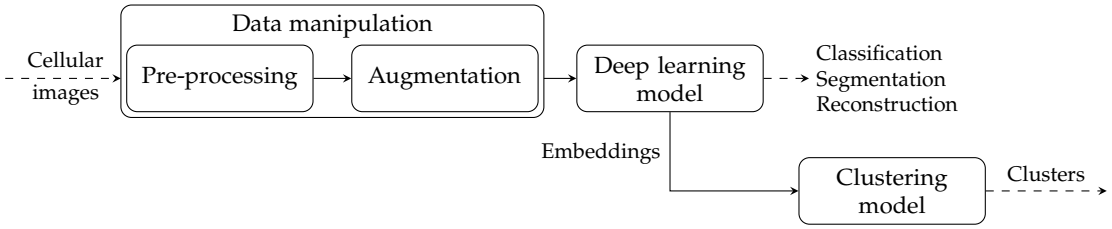


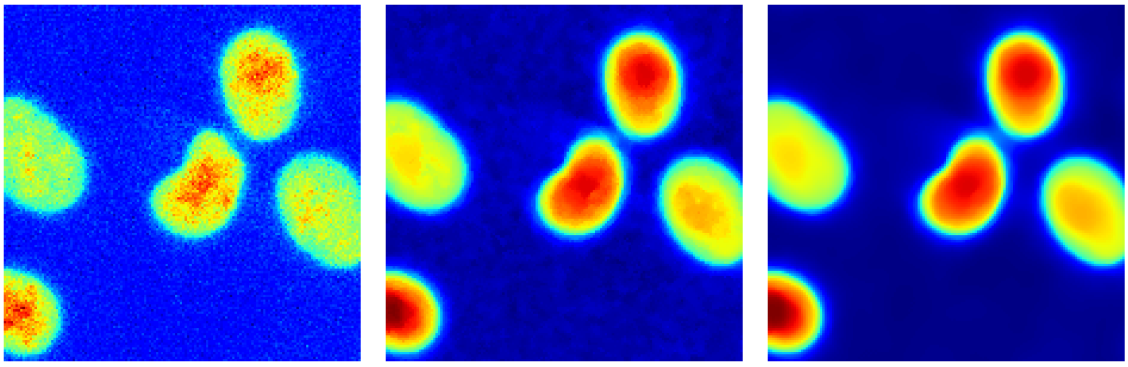
Figure 5.1: Complete scheme for the proposed solution for defining clusters of Systemic sclerosis patients. The deep learning model can have as objective to either classify, segment or reconstruct the input data.

5.1 Data preprocessing

The first step in our pipeline is data preprocessing, which consists of 5 transformations, as depicted in Figure 5.2. The preprocessing is meant to clear out as much as possible from the artifacts and noise in the confocal images.

Figure 5.2: Pipeline used for data preprocessing. Part of the pipeline was implemented in Jython scripts using ImageJ pre-defined functions, and part in Python.

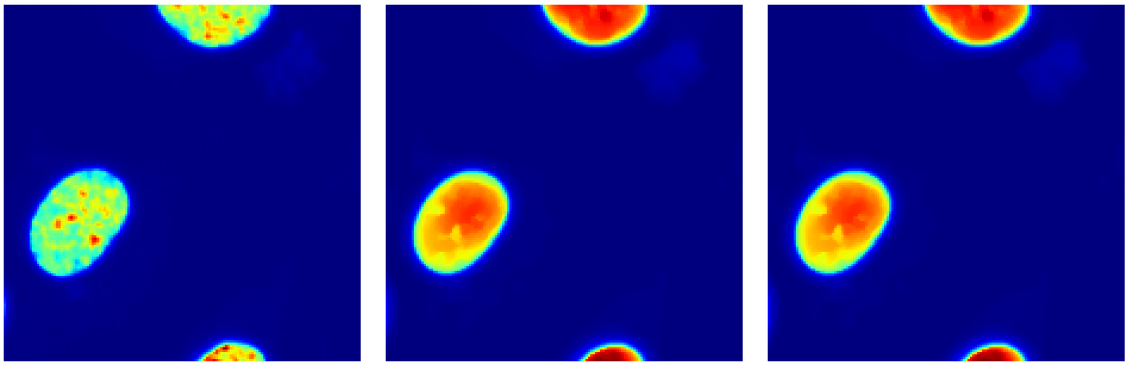
The outlier extraction helps us eliminate the debris that can be interpreted as useful information (small nuclei or cells). We use a selective median filter for this step. For each pixel, the median of



(a) (Objective lenses 1) 150 × 150 patch of the nuclei channel of a raw image.

(b) (Objective lenses 1) The nuclei channel after applying outliers extraction, background subtraction and median filter.

(c) (Objective lenses 1) The nuclei channel after total variation denoising.



(d) (Objective lenses 2) 150 × 150 patch of the nuclei channel of a raw image.

(e) (Objective lenses 2) The nuclei channel after applying outliers extraction, background subtraction and median filter.

(f) (Objective lenses 2) The nuclei channel after total variation denoising.

Figure 5.3: Example of results of different pre-processing steps on samples images with each objective lenses. Color map added to emphasize the differences.

its surroundings is calculated. If the deviation of the intensity of a pixel from the computed median value exceeds a pre-set threshold, the pixel is replaced with the localized median. We apply the outlier extraction filter only on the *nuclei* and *borders* channels. On the *nuclei* channel, the localized median is computed on a 10 pixels radius and brighter pixels, with a deviation of at least 50, are modified. For the *borders* channel, the radius is increased to 50 and the intensity threshold to 2 000.

The background subtraction and median filter transformations are used to further eliminate granular noise. We use background subtraction based on the "rolling ball" algorithm introduced in [48] with a rolling of 50. The median filter is applied with a small radius of 2, as we do not want the details to be blurred too much.

As noted in Chapter 2, two objectives were used during the data acquisition phase. While using the 20× Super Plan Fluor ELWD DM with $NA = 0.45$, problems with condensation were encountered. Thus, part of the data set ended up being noisier. In the comparative example in Figure 5.3, it can be seen that the differences between the two subsets are not fully attenuated after applying the first three transformations. In order to bring the images closer, a final denoising step is applied. After testing

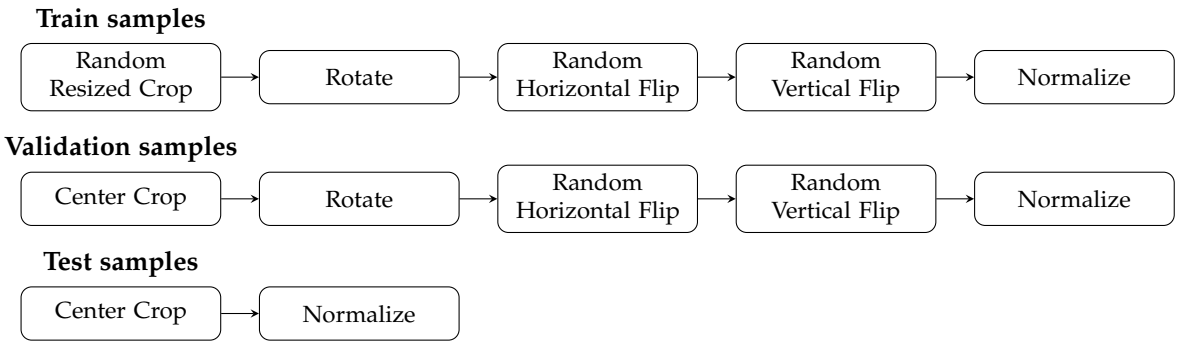


Figure 5.4: Pipeline used for data augmentation. The augmentations applied on validation and test samples have to preserve more from the original characteristics of images. The augmentations applied on test examples are restricted to just simple cropping and normalization.

various methods, we settled on a total variation denoising approach, as implemented in skimage[1].

The last preprocessing transformation applied is simple cropping at the centre, to shape 1536×1536. Because of the acquisition equipment and procedure, the raw samples have the tendency to be more out-of-focus towards the margins. By centre cropping, we ensure that the information passed to the models is more uniform in this sense.

All the hyper-parameters used during the preprocessing phase were determined empirically.

5.2 Data augmentation

In Section 2.6 we detailed the composition of the data set, with the final size of 401 3-channel confocal images. Because the number of samples is rather small for properly training a neural network, we have to apply several augmentation steps (see Figure 5.4). While augmenting, we also ensure that the final shape of the images (H_{in}, W_{in}) matches the neural network's definition, with H_{in} being the height of the image and W_{in}, the width.

For the training subset, we crop a random segment of the image, with the aspect ratio preserved at a scale $\sim \mathcal{U}(0.5, 1)$, using nearest-neighbour interpolation. The crop is further resized to the necessary shape (H_{in}, W_{in}). Next, we apply a rotation at a random angle, multiple of 90°. Horizontal and vertical flips are performed with a probability of 0.5 each. The images are then normalized. This step is commonly used in machine learning, as it helps with the convergence of the model [49]. We normalize each channel in an observation to $\mathcal{N}(0, 1)$. Considering c to be a channel in a random sample I, the normalization can be formalized as $(c - \bar{c})/\sigma(c)$, with $\bar{c}$ being the mean value in c and $\sigma(c)$, the standard deviation.

The validation images are cropped at the centre, randomly rotated, flipped and normalized. In this manner, we still increase the number of samples, while preserving the original aspect ratio. Increasing

[1]Total variance denoising function implementation in skimage: https://scikit-image.org/docs/dev/api/skimage.restoration.html#skimage.restoration.denoise_tv_chambolle.

the size of the validation set helps the gradient descent algorithm to perform smoother updates, as we can create more and larger batches.

The images reserved for testing are the only centre cropped and normalized, as we want to calculate the final performance of the model only on original samples.

5.3 Neural networks for extracting embeddings

At this stage, the shape of the samples is equal to $3 \times 1536 \times 1536$. This dimension is too high for the data to be directly served as input into a clustering model, as explained in Section 3.6. Thus, we need to construct a model which creates lower-dimensional representations of the images, while preserving relevant features. The goal of this research project is to cluster Scleroderma patients based on the similarities in the abnormalities observed in ECs. In this context, the term "relevant" refers to the information that can help with grouping similar samples.

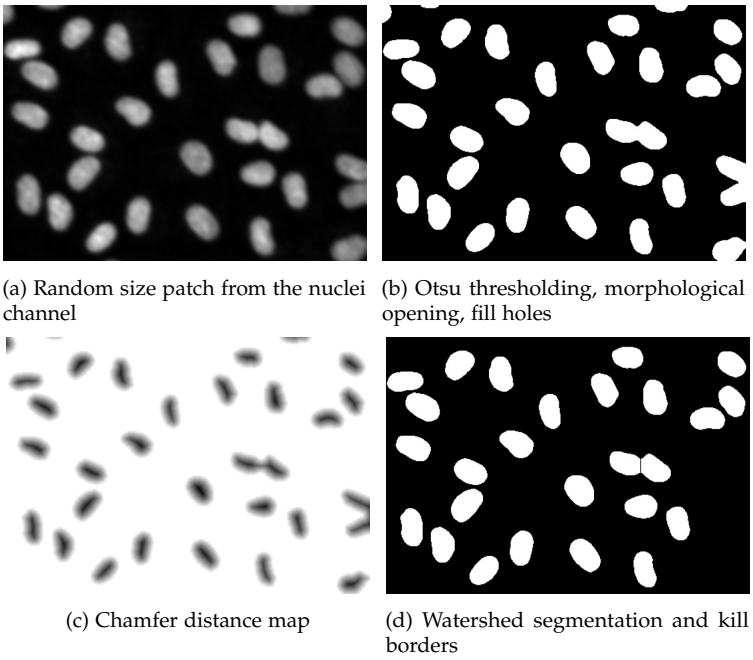
As previously mentioned, the simplest approach that can be followed for creating these encodings is to reshape each sample into a single-dimensional array and apply the PCA algorithm on the flattened forms. The dimensions that explain a sufficient fraction of the variance in the samples can be concatenated into lower-dimensional embeddings. We consider the clusters generated using this technique as one of the benchmarks in our experiments.

A more advanced procedure is to build a neural network whose activations of an internal layer can be extracted as embeddings. This introduces a restriction on the architecture: to include at least one block that constructs a lower-dimensional representation of the data. The neural network is optimized to perform a surrogate goal and the corresponding encodings are used as input for a clustering model. We investigate the effectiveness of the embeddings extracted using three different surrogate tasks: *actin* segmentation, image reconstruction, and binary classification. We implement each of these options and conduct experiments to determine which of them are appropriate solutions for our case study.

5.3.1 Segmentation approach

Developing a segmentation network is equivalent to training a model to separate areas of interest from an image. Following the insights received from the medical professionals at LUMC (see Chapter 2) on the characteristics that can be used to visually identify a Systemic sclerosis sample, we develop a model for segmenting the actin present inside the cells. We aim to disregard the actin present on the borders of the cells.

Training the model requires both the confocal images and the corresponding expected outputs, as it is a supervised process (see Section 3.9). In our case, the input data is composed of the *borders* and *actin* channels. The intuition behind using also the *borders* channel is that we want to provide the necessary information for identifying the borders and neglecting the actin present on them.



(a) Random size patch from the nuclei channel

(b) Otsu thresholding, morphological opening, fill holes

(c) Chamfer distance map

(d) Watershed segmentation and kill borders

Figure 5.5: Steps in manual nuclei semantic segmentation.

Naturally, we adopt the U-Net as our choice of architecture for segmentation. As the originally proposed model was developed to segment only nuclei, we need to adapt the solution to accommodate our problem. The implementation that we opted for is a multi-channel adaptation of the original U-Net architecture. Basically, the only modification is to change the input layer to accept multiple channels. We tile the input channels to 512×512. As explained in Chapter 2, the cellular modifications introduced by the Systemic sclerosis perturbations are distributed in the entire sample. Thus, cropping a smaller region from the original confocal image does no result in crucial information loss.

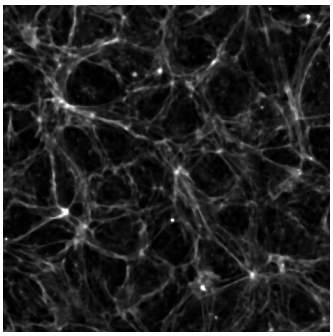
Ground truth

The U-Net computes the probability distribution of pixels to be part of the background/border (0) or to be actin (1). The output has the same shape as the input channels. The definition of a proper ground truth requires the development of a deterministic segmentation pipeline for both the *borders* and *actin* channels. The computation of each binary mask in this section starts from the channels pre-processed as detailed in Section 5.1.

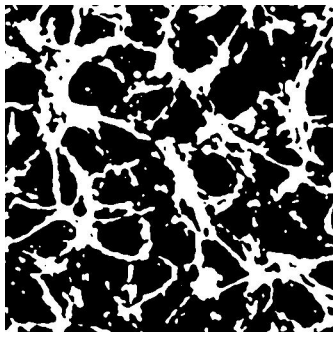
For detecting the borders of the cells, we guide the identification of the individual cells by nuclei positioning. A Voronoi-based segmentation [50] that uses already identified nuclei seed[2] can be applied in this regard. The algorithm divides an image into adjacent areas around each seed (i.e. Voronoi cells), which contain all the points closest to particular nuclei. The lines that separate the Voronoi cells coincide with the cell borders and are interpreted as the borders mask.

The steps in nuclei semantic segmentation are illustrated in Figure 5.5. We first apply a standard Otsu

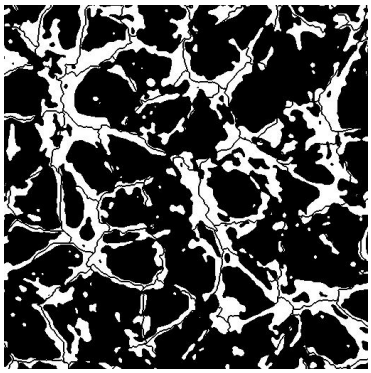
[2]Voronoi-based segmentation implementation in R: https://rdrr.io/bioc/EBImage/man/propagate.html.



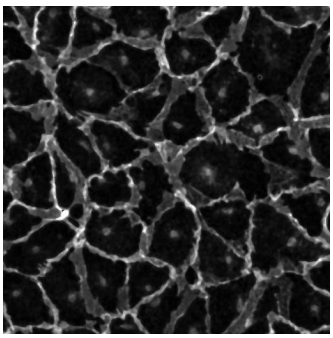
(a) 512 × 512 patch of actin channel.



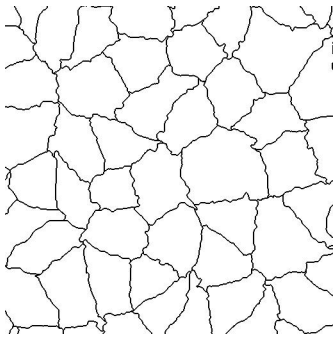
(b) Actin binary map.



(e) Actin inside the cells binary map. Binary 'and' operation between the actin and borders binary maps.



(c) 512 × 512 patch of borders channel.



(d) Borders binary map.

Figure 5.6: Steps in defining the segmentation ground truth (actin inside the cells binary map) for a random 512 × 512 patch, and its associated weight map.

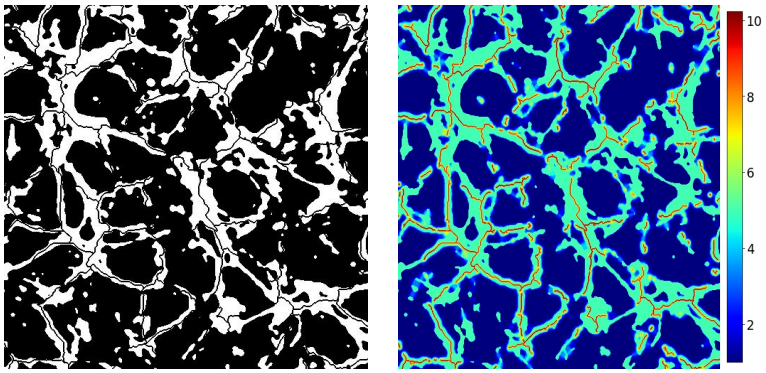
thresholding [51] on the nuclei channel. The method returns one value that separates the intensities in the actin channel into two classes (actin fibres and background) such that the inter-class variance is maximized or the intra-class variance is minimized. This threshold is then used to define a binary mask. To filter out granular impurities we apply a morphological opening, with a circle structural element with a radius of 3. The holes that can occur inside the isolated nuclei are removed using a flood-fill approach. As there is a possibility for touching nuclei, we apply the watershed segmentation [52] with dynamic 1 and 4-pixel connectivity on the inverse Chamfer distance map transform of the binary map[3]. The inverse Chamfer distance map is computed with normalized Chessknight weights and 32-bit output format. The individual nuclei are labelled and we clear the incomplete nuclei visible on the borders.

We use the resulting nuclei binary masks as seeds for the Voronoi-based segmentation. The output is a second binary mask with information only on the regions that correspond to the borders of the cells (Figure 5.6c).

The actin information is extracted by simply applying the Otsu thresholding on the nuclei channel (Figure 5.6b). The ground truth definition for the semantic segmentation of the actin inside the cells is finalized by applying a binary 'and' operation between the actin and borders binary masks (Figure 5.6e).

Following the methodology in [11], we define a weight map that assigns an importance score to each pixel in the ground truth map (Figure 5.7). This score is taken into account for loss computation. In our

[3]Distance map watershed segmentation documentation: https://imagej.net/plugins/distance-transform-watershed



(a) Actin inside the cells binary map. (b) Pixel-level weight map.

Figure 5.7: Segmentation ground truth for a random 512 × 512 patch, and its associated weight map.

case, we assign increased importance to the pixels that depict the cell borders and the actin.

In Section 6.1 analyze our results from applying this segmentation approach and discuss how suitable this class of models is for image encoding construction.

5.3.2 Reconstruction approach

In Chapter 3 we introduce the concept of image reconstruction task and present a state-of-the-art class of neural networks for solving it in Chapter 4. The reconstruction problem can be summarized in two phases: the encoding step, where the input data is compressed into a lower-dimensional space (generally one dimensional), and the decoder step, within which the network tries to expand the information in the latent space to an image that resembles as closes as possible the original one. The output image is directly compared to the corresponding input channel.

Although generally, such networks can comfortably handle 3-channel input images, the nature of the information depicted in our samples made it challenging to find an autoencoder architecture that could properly reconstruct the entire sample at once. In our case, each channel represents the complete characteristics of a biological sample. When proving all 3 channels as input for the autoencoder, we observed that the network would direct its focus on a single channel (i.e. the *nuclei* one) and optimize the reconstruction of it. Thus, we opted for training one autoencoder per channel and define the final embedding of the sample as $e_i = e_{actin_i} \parallel e_{borders_i} \parallel e_{nuclei_i}$, where operator $\parallel$ denotes the vectors concatenation function, e_i is the embedding for the entire sample and $e_{channel_i}$ are the encodings constructed for each individual channel in the current sample. The specific order in which one chooses to concatenate the vectors (in our example: $actin \rightarrow borders \rightarrow nuclei$) does not influence the clustering performance. However, the ordering has to be preserved during the whole experimentation for consistency reasons.

The architecture we adopted follows the encoder-decoder paradigm, and it is illustrated in Figure 5.8. We opted for input image patches of 256 × 256, softmax as the output activation function and MSE as

the reconstruction loss.

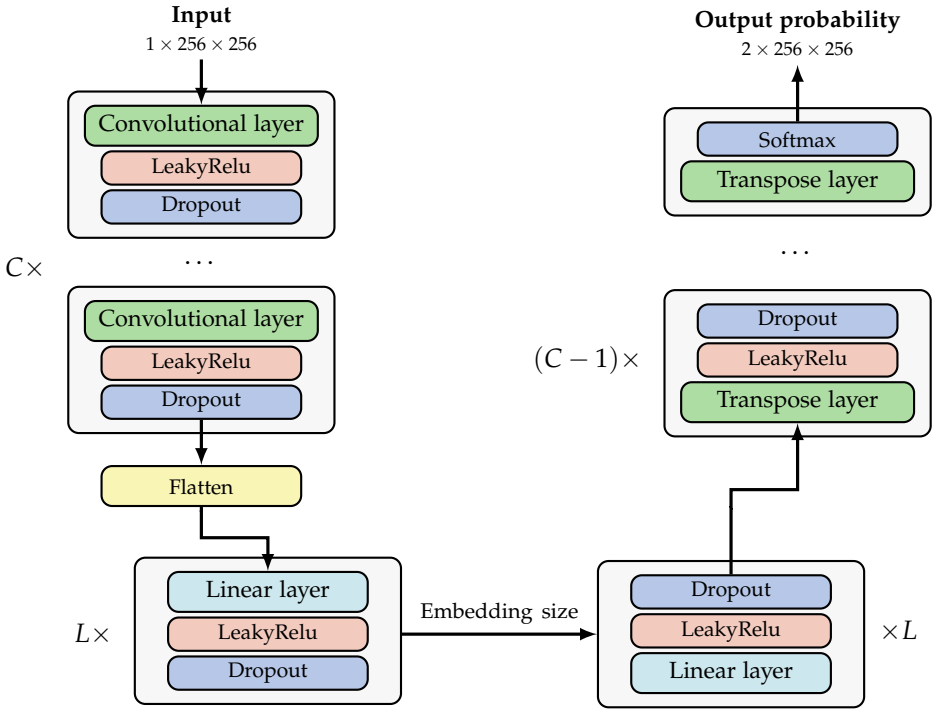


Figure 5.8: Architecture of the autoencoder used for reconstructing all the channels. Softmax outputs the probability distribution of each pixel being either background or object of interest. The transpose layer implements the transpose convolution operation [53]. We use $C = 4$ and $L = 2$. The last linear block in the encoder must output a feature vector with the dimensionality equal to the desired embedding size. The dropout rate is 0.4.

5.3.3 Classification approach

The last procedure that we employ for constructing the necessary sample embeddings is a classification approach. The neural network is designed to solve a binary classification task, namely to predict whether a sample is treated with plasma from a Systemic sclerosis patient or healthy control. The architecture has to contain a block of layers whose output features have the shape of the desired embedding. This block helps the classification, as its weights are still trained during backpropagation, and it also serves as a proxy for us to extract the embeddings.

Custom classifier

In order to get a sense of the behaviour of our data set as the input for a classifier, we started by constructing a small custom architecture, shown in Figure 5.9.

The network starts with a series of convolutional blocks, designed to gradually reduce the dimensionality of the data by a factor of 2. The first convolutional layer receives 3 input channels and outputs 256. The following convolutional layers have the number of output channels equal to half of the number of the input channel. All convolutional layers have kernel size $(3, 3)$, stride $(1, 1)$, padding $(1, 1)$ and use Leaky ReLU [54, 55] as activation. They are accompanied by both batch normalization operations [56] and

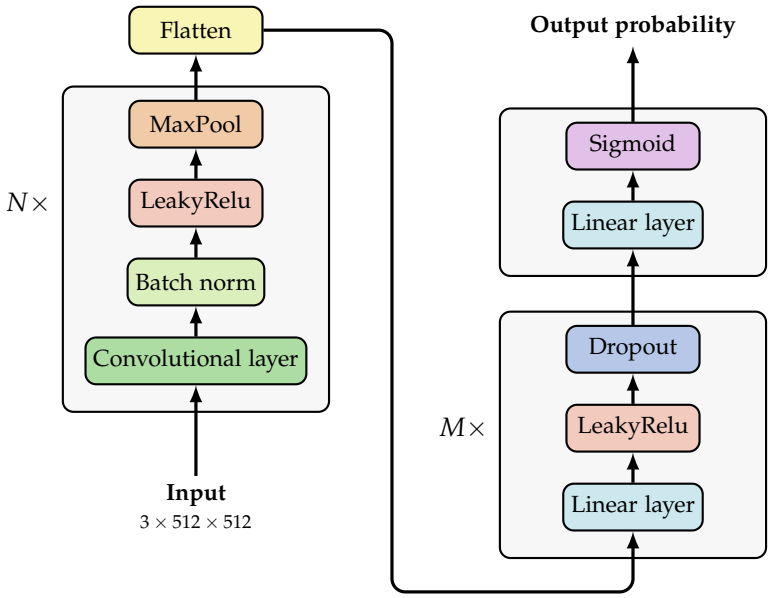


Figure 5.9: General scheme of the classifier architecture. In our experiments $N = 8$ and $M = 2$. The dropout rate is 0.4.

dropout regularization [57], due to the restrictive size of the data set. The max-pooling layers shrink the dimensionality of the input by a factor of 2. The following two linear layers output 64 and *embedding size* features, respectively. They use Leaky ReLU as activation. The final layer is a linear layer with sigmoid activation, as the classifier is trained to perform binary classification.

Pre-trained variants

There are a variety of powerful state-of-the-art classification models that hold impressive results in a variety of problems, such as DenseNet [46], ResNet [58], EfficientNet [59], to name a few. We expect using these networks to favour a better separation of our classes and, in turn, a higher classification performance. However, applying such complex models directly to our samples is quite challenging, as they are not developed for small data collections: the number of weights to be tuned is high and the regularization is generally quite modest for our requirements. Augmenting the samples to a quantity appropriate for training these neural networks results in creating a low variance data set, which in turn prompts overfitting. In order to be able to use such powerful networks, we need to make use of transfer learning (see Section 3.5). Figure 5.10 showcases the general architecture of the classifier model which includes a pre-trained model.

The fastest way to integrate transfer learning into the pipeline is to employ available pre-trained models on ImageNet [41]. Although this data set does not resemble the nature of our cellular images, using the weights from pre-trained models on it as a starting point could offer a better initialization than random. The overfitting encountered during fine-tuning shrank the set of architectures that were capable to handle our data set. After testing multiple networks, we settled on the DenseNet-121 architecture, as the overfitting was more controllable than in any other case. We use binary-cross entropy as the loss

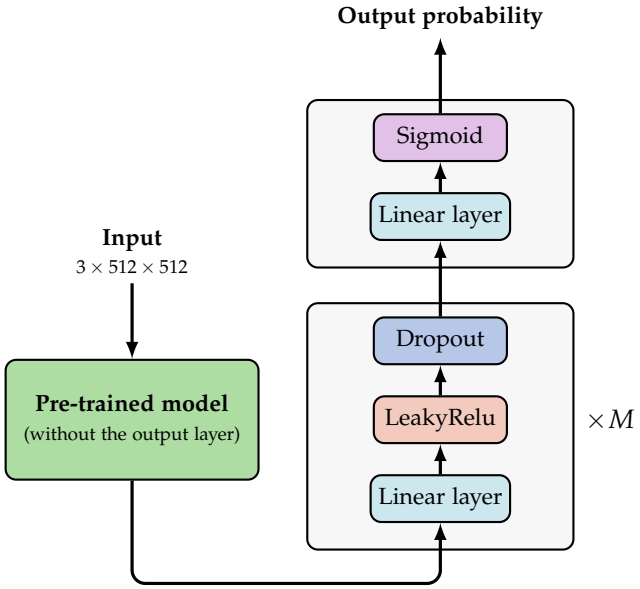


Figure 5.10: General scheme of classifier including pre-trained model. One of the M Leaky ReLU activated linear layers must have the dimension of the output features equal to the desired embedding size.

function. We set the number of linear blocks following the pre-trained model $M = 1$.

Nevertheless, a more rigorous approach is to pre-train a model on a large collection of images that relates at some level to our data. As our acquisition process, samples and end-goal problem are highly specialized, we consider as good enough candidates any high-resolution confocal images collections, that contain, besides the *nuclei* channel, at least another channel resembling the *actin* or *borders* ones. The closest data set that we have been able to find is the RxRx1 CellSignal 2019 data set[4], which depicts drug inducted genetic perturbations on human cells. The linked task is the detection of the type of perturbation applied at the sample level. There are 1 139 possible classes (including controls). We tried to pre-train different state-of-the-art architectures on this data set and then fine-tune the weights on our smaller collection. We used cross-entropy with label smoothing regularization (see Chapter 3). Although the loss function seemed to have a less extreme variation, the final performance of the target model did not improve. An important note here is that due to time constraints, we did not invest a lot in optimizing the pre-trained model, which might have in turn hindered the potential of this experiment.

5.4 Clustering algorithms

Lower-dimensional representations of the samples are obtained using each of the presented methodologies. The embeddings are defined at a well-level. As we mentioned in 2, a biological sample is cultured in a well and for each well, we imaged at most 4 non-overlapping sites. This means that one biological sample corresponds to at most 4 unique observations in our data set (thus, at most 4 encoding representations). We apply the median over the embeddings associated with different sites of the same well to obtain the

[4]Original source of the data set and complete description: https://www.rxrx.ai/rxrx1.

final sample embedding.

We first analyse whether the information encapsulated in the embeddings retain sufficient discriminating features to be able to form two clusters with high ground truth class homogeneity. Thus, we check if we can construct, based on the encodings, one cluster with the class majority being Systemic sclerosis samples and another one, with the control observations. We implement this using the K-means algorithm with $k = 2$.

We then apply hierarchical clustering to finally try to check whether we can identifying pertinent subgroups in our data. We assess the resulting clusters based on the homogeneity, completeness, Silhouette and Davies-Bouldin scores.

Chapter 6

Experiments

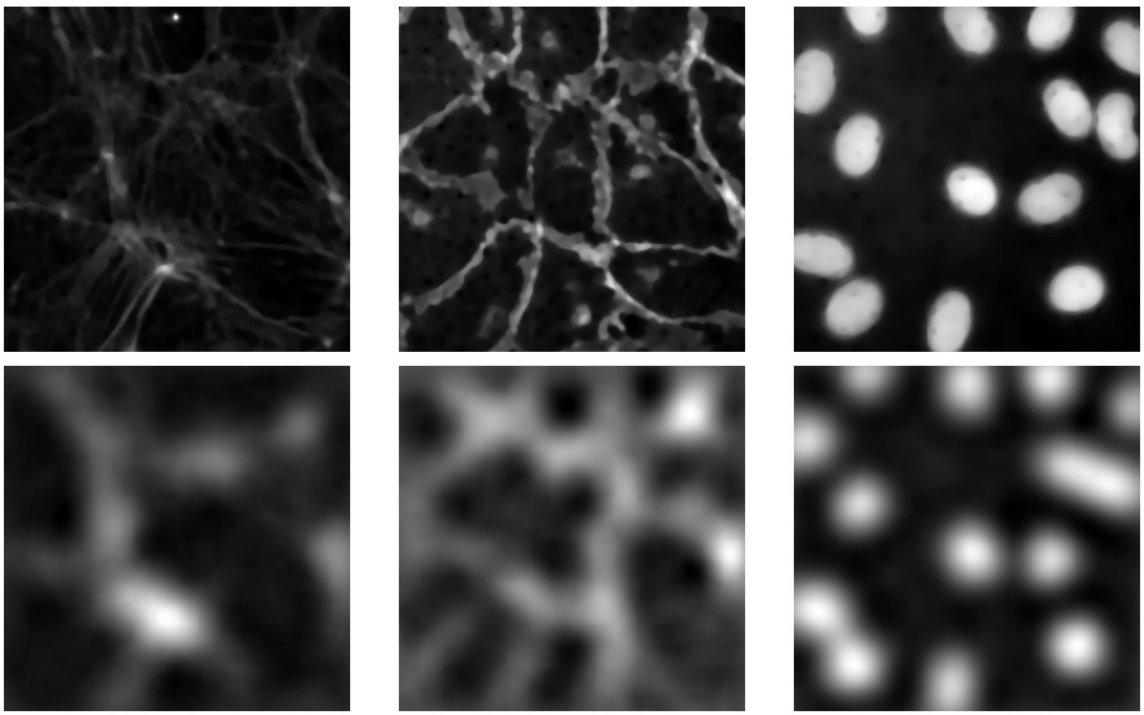
In this section, we describe the experimental settings tested for defining lower-dimensional representation of the cellular images and sub-groups of patients and controls. We start by investigating how suitable models performing segmentation are for constructing image embeddings. Then we experiment with both reconstruction and classification architectures. We train neural networks for both tasks on data sets augmented at different degrees and then define the image embeddings for our unaltered input images. We develop baseline embeddings by directly applying PCA. We verify that the encodings still contain features that discriminate between SSc and HC samples by training a simple SVM classifier and constructing a two-class clustering using K-means. Finally, we develop dendrograms using the image representations extracted from the neural networks.

6.1 Segmentation experiment

Training a deep learning model for segmenting the actin inside the cells proved to be a quite difficult problem, due to the delicate details in the input channels (i.e. actin and borders). Constructing a larger data set, through acquisition and/or augmentation could lead to better convergence. However, the method would still not be suitable for our use case, due to some architectural technicalities in the U-Net.

The initial reason for training the deep learning model was to be able to encode the initial confocal images into a lower-dimensional space. Given the general architecture of the U-Net follows the standard autoencoder structure (see Figure 3.2), the final layer in the encoder could be perceived as a viable source for image encodings. However, by simply computing the output shape of this layer, one can determine that the actual representation at that level is inherently higher-dimensional than the input.

The complete input shape for our U-Net model is $2 \times 512 \times 512$. By simply serializing this representation we obtain a vector of $2 \times 130\,000$ features. The encoder's output has a shape of $1024 \times 32 \times 32$, which converts into a vector with over 1 million features. Adding more blocks to the encoder further increases



(a) Actin reconstruction. (b) Borders reconstruction. (c) Nuclei reconstruction.

Figure 6.1: Example of the channels reconstruction achieved with

this number. Due to this dimensionality expansion, we conclude that the U-Net architecture is not suitable for extracting lower-dimensional representations of our data.

6.2 Reconstruction experiment

We train three individual autoencoders for each channel: *nuclei*, *actin* and *borders*. We use random crops from the augmented samples with a shape of 256×256. The reconstruction of the *borders* and especially the *actin* channels are more challenging, due to the high amount of fine details. The size of the bottleneck is tuned to 256 such that the reconstruction for all channels is possible and the dimensionality of the encoding is kept relatively low. The channel embeddings extract for different sites of the same well are averaged. We finally concatenate the channel encodings into a single one-dimensional embedding. This process results in 103 sample embeddings with a size of 768. Figure 6.1 showcases an example of sample reconstruction.

We first apply SVM with a 5-fold cross-validation scheme to check whether we can discriminate between the two classes in our data set using the embeddings generated with autoencoders. The accuracy of the model is 63.5%.

We use PCA to extract the first two principal components and be able to visualize the results of the clustering. We apply two clusters K-means Figure 6.2a, based on the euclidean distance, and a dendrogram Figure 6.3a, with the cosine distance as metric. The associated metrics are included in

		Image embeddings from			
		PCA	**Autoencoder**	**Custom classifier**	**DenseNet classifier**
Cluster size	Cluster 1	37	16	12	41
	Cluster 2	66	87	91	62
Homogeneity score		0.0024	0.0037	0.2171	**0.2499**
Completeness score		0.0024	0.0055	**0.2464**	0.2427
V-Measure		0.0024	0.0044	0.2309	**0.2463**
Davies-Bouldin index		5.49	4.94	**0.409**	0.846
Silhouette score		0.324	0.469	**0.738**	0.516

Table 6.1: Performance metrics for the two clusters K-means algorithm applied on embeddings defined using deep learning models trained for image reconstruction and classification.

Table 6.1.

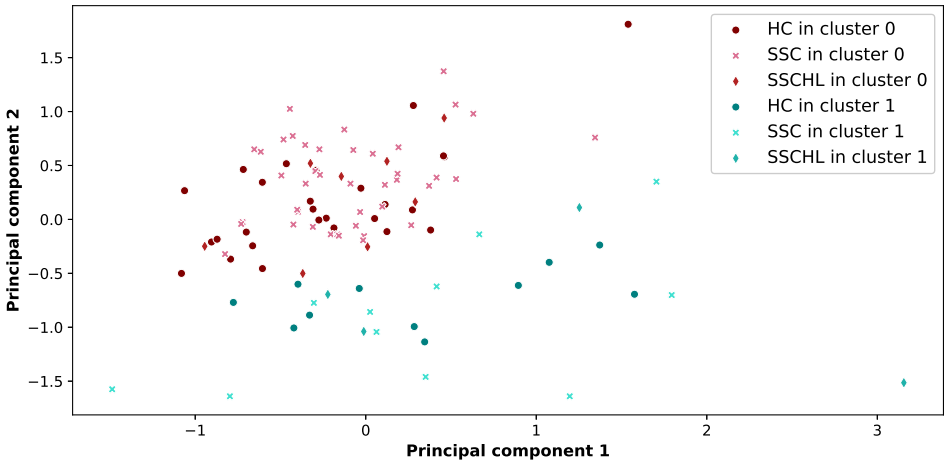
6.3 Classification experiment

Firstly, we define a classifier benchmark. We directly embed the images in a 64-dimensional representation using PCA and train an SVM classifier with a 5-fold cross-validation scheme. The performance metrics are included in Table B.2.1.

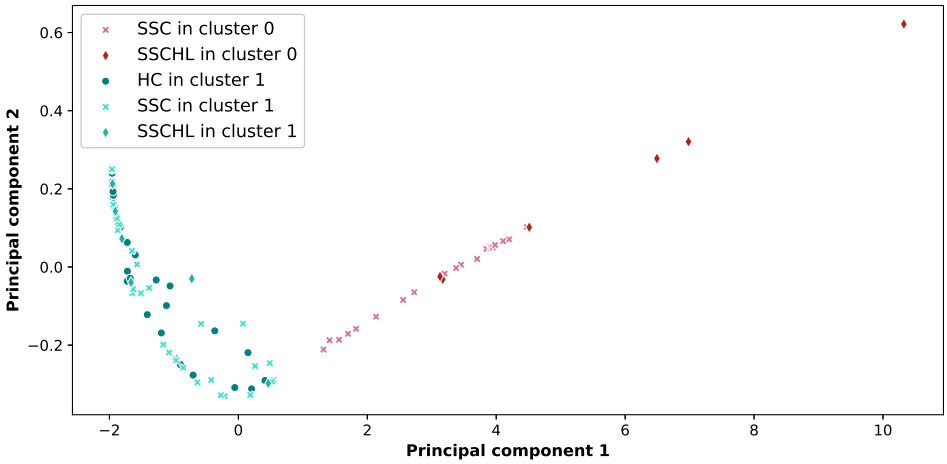
6.3.1 Custom classifier

We train several custom models, as presented in Section 5.3.3, on 512×512 random crops of augmented samples. We use an embedding size of 8. Due to the conservative size of the data set, we experiment with several multiplication ratios of the data, in order to determine the amount of augmentation accepted by the classifier before it starts to overfit. The augmentation levels that we consider extend the data set $1\times$ (only original data), $5\times$, $10\times$, $20\times$, $30\times$ and $50\times$. We use a *batch size* of 8. The train, validation and test set represent 60%, 20% and respectively 20% of the augmented data. We ensure that all the input data originated from one biological sample are included in only one of these subsets. The data is shuffled each epoch such that the batches have a random composition every time. For each run, we store the weights of the classifier with the lowest validation error. The Adam optimizer has a starting *learning rate* of $5e-5$, $1e-5$ *weight decay*, *beta*0 and *beta*1 equal to 0.9 and 0.999 respectively, and an *epsilon* of $1e-08$. The learning rate is annealed using a cosine scheduler with a maximum of 10 iterations and an accepted minimum learning rate of $5e-10$.

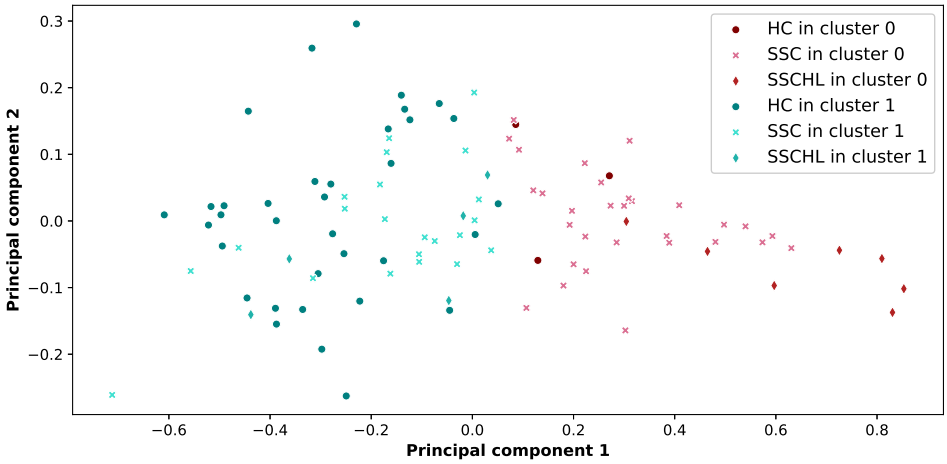
Figure C.1.1 shows the train and validation learning curves resulted in the data augmentation experiment. Based on these figures, we conclude that using a data set augmented less than $5\times$ results in underfitting, while augmenting with a larger factor makes the model overfit. Thus, we select the custom classifier



(a) Based on embeddings extracted from autoencoder.



(b) Based on embeddings extracted from custom classifier.



(c) Based on embeddings extracted from the ImageNet pre-trained DenseNet-121.

Figure 6.2: Output of K-means algorithm with $k = 2$ and euclidean distance for embeddings extracted from the image reconstruction and classification deep learning models.

trained on a 5× data set for performing the remaining experiments. The performance metrics of this model are listed in Table B.2.1. When extracting the embeddings, we use the unaltered original samples and serve a 512×512 central crop for each of them to the classifier. The median is applied over the embeddings associated with different sites of the same well. Thus, we construct 103 embeddings with size 8.

Again, we start by applying an SVM classifier with a 5-fold cross-validation scheme, to check if this low-dimensional representation can still be used to discriminate between classes. The accuracy of the model is 67.7%.

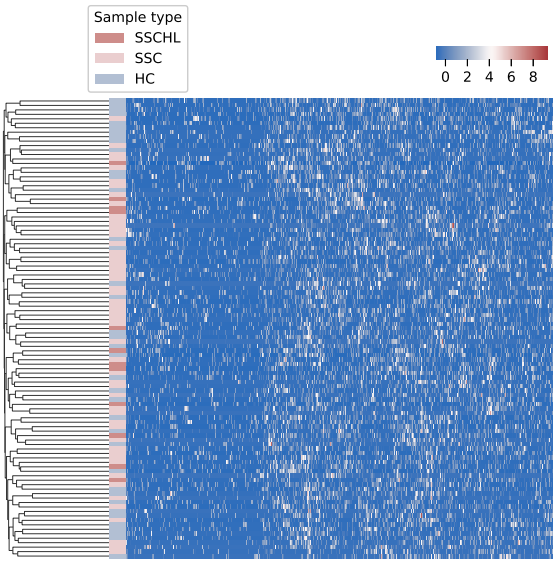
We compute the first two principal components and serve them as input for the two clusters K-means clustering Figure 6.2b and dendrogram construction Figure 6.3b. The K-means clustering uses the Euclidean distance, while the dendrogram is computed using the cosine distance as the metric. The performance metrics for the K-means clustering are included in Table 6.1.

6.3.2 Pre-trained classification network on ImageNet experiments

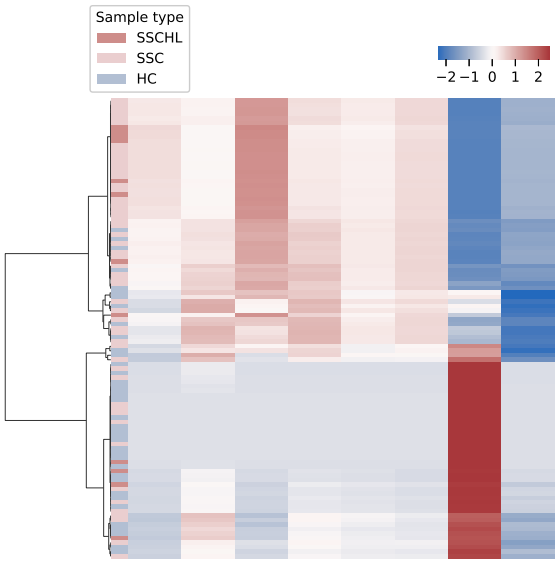
We apply the exact same steps as described in the previous subsection on a classifier pre-trained on the ImageNet data set. We experimented with multiple state-of-the-art architectures pre-trained on ImageNet and based on our trials, we settle on the DenseNet-121 architecture. All the weights in the pre-trained network are fine-tuned using our data set.

We run the same data augmentation experiment on the chosen architecture. The associated learning curves are shown in Figure C.1.2. We again select the classifier fine-tuned on the data set augmented to 5× its size and extract the embeddings. The performance descriptors of the classification model are included in Table B.2.1.

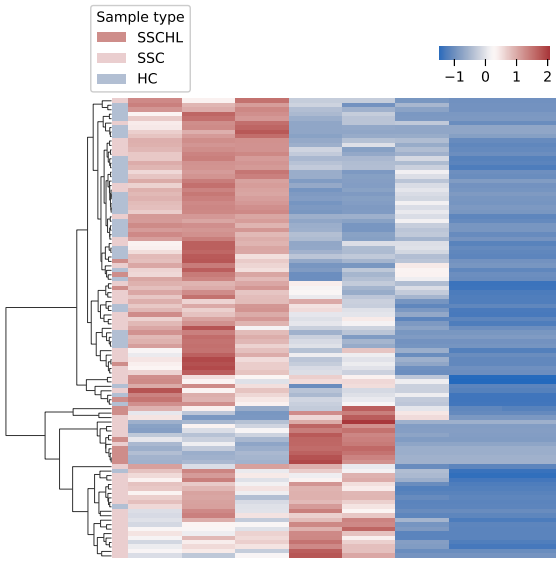
As the accuracy of an SVM model with 5-fold cross-validation trained on these lower-dimensional representations is 73.5%, we can conclude that the embeddings capture more of the discriminative features between the two input classes, HC and SSc, than in the other cases. We define the corresponding principal components using PCA and train clustering models on them. The K-means clusters are visualized in Figure 6.2c and the associated performance metrics are shown in Table 6.1. Figure 6.3c illustrates the dendrogram.



(a) Based on embeddings extracted from autoencoder.



(b) Based on embeddings extracted from custom classifier.



(c) Based on embeddings extracted from the ImageNet pre-trained DenseNet-121.

Figure 6.3: dendrograms generated on lower-dimensional representations of unaugmented images extracted from the image reconstruction and classification deep learning models.

Chapter 7

Conclusion and Discussion

7.1 Conclusion

Sub-grouping patients based on the ECs abnormalities is a complex task, with the potential of advancing the understanding of Systemic sclerosis and unveiling personalized therapeutic schemes. The unsupervised nature of this problem results in the necessity of an interdisciplinary approach and the final assessment of the discovered clustering needs to be made by medical experts.

The definition of descriptive features for a sample has been previously done by computing a large quantity of handcrafted features on a singular cell level. We explore ways to adapt the pipeline to operate directly on the entire images, such that the context is not lost. We consider deep learning models that perform image segmentation, reconstruction or classification, while intrinsically compressing the data into lower-dimensional representations. The resulting embeddings of the input samples are further used for our clustering goal. Although some of the methods demonstrate high potential for being able to construct suitable embeddings, the lack of data hinders their performance. Nevertheless, we show that even in this scenario, the created embeddings already contain features that favor the unsupervised separation of Systemic sclerosis samples from healthy controls and even the unsupervised discovery of groups of severe cases.

We consider this study to be a compelling proof of the capabilities of deep learning models for developing lower-dimensional image representations, that encapsulate the information from the entire sample and can be successfully used as input for clustering methods.

7.2 Discussion

The augmentation experiment performed on the autoencoder, and two classification networks, proves that the amount of acquired data is insufficient for successfully training deep learning models. All three

neural networks underfit when the collection is not augmented and start to showcase signs of overfitting when the size of the data set is increased as little as 10 times, with the classifiers being slightly more susceptible to this issue. The overfitting is due to the limited variance in the images, despite increasing the data collection. The neural networks are sufficiently complex to create a straightforward mapping between the input images and the expected output. The high magnitude spikes in the epoch loss are induced by the amount of data and the necessity of using a small batch size. However, the learning curves associated with the training processes demonstrate that models with decent reconstruction and classification performance are achievable, with an appropriate number of samples.

We observe that the autoencoders are able to separate the objects of interests from background in most cases. However, due to the limited embedding space and the fine details present in both the *actin* and *borders* channels, the reconstructions depict only the most pronounced features and have an observable blurriness. The latter issue can be alleviated by increasing the embedding space, while the reconstruction of the fine details can only be achieved with a larger data set and more complex architecture.

For the classification models, the DenseNet-121 based architecture generally outperforms the custom one. However, the baseline SVM model seems to have a better recall than any of the neural networks. By also considering the low accuracy and precision scores of the SVM, we can conclude that the baseline model tends to classify most of the samples as being from patients, which indeed increases the recall score. Thus, this isolated score should not be considered as a reliable indicator of the model's performance.

The performance of the SVMs trained using the deep learning generated embeddings confirms the intuition that, in our study, the lower-dimensional representations should be constructed to retain features that can still discriminate between the SSc and HC classes. Due to the complex information encapsulated in each sample (i.e. irregular actin fibres and borders), the features required for image reconstruction do not necessarily overlap with the ones for class discrimination. This idea is even more apparent when analysing the clusters generated by the K-means algorithm. The grouping produced using the embeddings originated from the autoencoder architecture do not separate samples from patients and controls. This division exists in the clusters defined based on the classifier networks. Both the custom classifier and pre-trained DenseNet-121 based architecture are able to isolate a sub-set of SSc samples. All the performance metrics increase when using the classifier as an embedding model. The increased homogeneity and completeness scores indicate that we can identify clusters with less class variety. As both the Silhouette and Davies-Bouldin improve, the discovered groups are more compact while the distance between them increases. Generally, the SSc samples separated from HCs have a cellular structure visibly distinct from the controls. These results provide compelling evidence that the classification neural networks are suitable for constructing embeddings for our data set.

The dendrograms reinforce the previous findings. The representation vectors generated using the autoencoder are strikingly similar. The strong resembles confirms that features associated with the nature of the samples (i.e. HC or SSc) are not valuable for reconstruction. On the other hand, the embeddings

generated using the classifiers are visibly better at separating the two classes and thus support the discovery of more interesting sub-groupings. In both cases the features are able to encapsulate differences depending on the origin of the samples. Clusters of mainly Systemic sclerosis cases are formed. Sever SSc cases (SSCHL) are grouped together even if the classifiers were not served any explicit information in this regard. Increasing the performance of the classifiers and the confidence of their predictions (assigning classification scores closer to 0.0 to HC samples and 1.0 to SSc samples) will strengthen the discriminative features in the encodings and further encourage an increased inter-cluster distance and class homogeneity.

7.3 Future work

In terms of future efforts, the primarily focus should be to gather a larger collection of samples, such that artificial augmentation becomes less of a necessity for training a decent model and more of an improvement option. It is preferred for the additional data to be balanced in terms of class distribution.

The next set is to observe the performance of the models proposed in the current work on the new data set. Re-training or at least fine-tuning the DenseNet-121 based model should result in an increased confidence in the prediction and higher performance. Having extreme scores associated with the predictions generates an improved separation of the classes in the embedding space and a better support for a clustering approach. If the model is fine-tuned, the additional data has to be pre-processed and especially normalized using the same scheme as for the original data.

If the number of samples allows, replacing the DenseNet-121 pre-trained model with a more complex neural networks (DenseNet-161, DenseNet-201, various ResNet models, etc.) might further increase the classification performance.

Finally, by investing time in defining a performance metric for the clustering algorithm, one can rephrase the entire study into a supervised problem. The metric can even be encapsulated in the loss function for both the autoencoder architecture and the classifiers, such that the trained weights can be specialised towards the construction of embeddings useful for clustering, besides a good performance of the surrogate task.

Acknowledgements

I would like to thank Prof. Fons Verbeek for offering me the opportunity to work on this project and Dr. Vincent van Duinen for proposing the project and for the patience to understand my needs, as a computer scientist, for properly integrating the biological knowledge in the AI solution.

Furthermore, I would like to extend my deepest appreciations to Dr. Lu Cao for the insightful conversations and constant guidance throughout the entire research period.

Finally, I want to acknowledge the tremendous support received from Rudmer Postma, from the extensive responses to all of my questions, to the availability to guide me in my first cell culturing experiment. The entire section describing the cell culturing and data acquisition (Chapter 2) process was kindly documented by Rudmer.

Bibliography

[1] Christopher P Denton and Dinesh Khanna. Systemic sclerosis. *The Lancet*, 390(10103):1685–1699, 2017.

[2] Juan C Caicedo, Sam Cooper, Florian Heigwer, Scott Warchal, Peng Qiu, Csaba Molnar, Aliaksei S Vasilevich, Joseph D Barry, Harmanjit Singh Bansal, Oren Kraus, et al. Data-analysis strategies for image-based cell profiling. *Nature methods*, 14(9):849–863, 2017.

[3] Vebjorn Ljosa, Peter D Caie, Rob Ter Horst, Katherine L Sokolnicki, Emma L Jenkins, Sandeep Daya, Mark E Roberts, Thouis R Jones, Shantanu Singh, Auguste Genovesio, et al. Comparison of methods for image-based profiling of cellular morphological responses to small-molecule treatment. *Journal of biomolecular screening*, 18(10):1321–1329, 2013.

[4] Ashley A Powell, AmirAli H Talasaz, Haiyu Zhang, Marc A Coram, Anupama Reddy, Glenn Deng, Melinda L Telli, Ranjana H Advani, Robert W Carlson, Joseph A Mollick, et al. Single cell profiling of circulating tumor cells: transcriptional heterogeneity and diversity from breast cancer cell lines. *PloS one*, 7(5):e33788, 2012.

[5] Darren A Cusanovich, Riza Daza, Andrew Adey, Hannah A Pliner, Lena Christiansen, Kevin L Gunderson, Frank J Steemers, Cole Trapnell, and Jay Shendure. Multiplex single-cell profiling of chromatin accessibility by combinatorial cellular indexing. *Science*, 348(6237):910–914, 2015.

[6] Bushra Raj, Daniel E Wagner, Aaron McKenna, Shristi Pandey, Allon M Klein, Jay Shendure, James A Gagnon, and Alexander F Schier. Simultaneous single-cell profiling of lineages and cell types in the vertebrate brain. *Nature biotechnology*, 36(5):442–450, 2018.

[7] Michael Steinbach, Levent Ertöz, and Vipin Kumar. The challenges of clustering high dimensional data. In *New directions in statistical physics*, pages 273–309. Springer, 2004.

[8] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[9] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[10] Iordania Constantinou, Michael Jendrusch, Théo Aspert, Frederik Görlitz, André Schulze, Gilles Charvin, and Michael Knop. Self-learning microfluidic platform for single-cell imaging and classification in flow. *Micromachines*, 10(5):311, 2019.

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[12] Jaak Simm, Günter Klambauer, Adam Arany, Marvin Steijaert, Jörg Kurt Wegner, Emmanuel Gustin, Vladimir Chupakhin, Yolanda T Chong, Jorge Vialard, Peter Buijnsters, et al. Repurposing high-throughput image assays enables biological activity prediction for drug discovery. *Cell chemical biology*, 25(5):611–618, 2018.

[13] Mark-Anthony Bray, Shantanu Singh, Han Han, Chadwick T Davis, Blake Borgeson, Cathy Hartland, Maria Kost-Alimova, Sigrun M Gustafsdottir, Christopher C Gibson, and Anne E Carpenter. Cell painting, a high-content image-based assay for morphological profiling using multiplexed fluorescent dyes. *Nature protocols*, 11(9):1757, 2016.

[14] Frank Van Den Hoogen, Dinesh Khanna, Jaap Fransen, Sindhu R Johnson, Murray Baron, Alan Tyndall, Marco Matucci-Cerinic, Raymond P Naden, Thomas A Medsger Jr, Patricia E Carreira, et al. 2013 classification criteria for systemic sclerosis: an american college of rheumatology/european league against rheumatism collaborative initiative. *Arthritis & Rheumatism*, 65(11):2737–2747, 2013.

[15] Dianne Vreeken, Caroline Suzanne Bruikman, Stefan Martinus Leonardus Cox, Huayu Zhang, Reshma Lalai, Angela Koudijs, Anton Jan van Zonneveld, Gerard Kornelis Hovingh, and Janine Maria van Gils. Eph receptor b2 stimulates human monocyte adhesion and migration independently of its ephrinb ligands. *Journal of leukocyte biology*, 108(3):999–1011, 2020.

[16] Edgar F Smeets, Eckhardt JU von Asmuth, Cees J van der Linden, Jet FM Leeuwenberg, and Wim A Buurman. A comparison of substrates for human umbilical vein endothelial cell culture. *Biotechnic & histochemistry*, 67(4):241–250, 1992.

[17] David J Field and Nuala Brady. Visual sensitivity, blur and the sources of variability in the amplitude spectra of natural scenes. *Vision research*, 37(23):3367–3383, 1997.

[18] Mark-Anthony Bray, Adam N Fraser, Thomas P Hasaka, and Anne E Carpenter. Workflow and metrics for image quality control in large-scale high-content screens. *Journal of biomolecular screening*, 17(2):266–274, 2012.

[19] MJ Currie, DS Berry, T Jenness, AG Gibb, GS Bell, and PW Draper. Starlink software in 2013. *Astronomical Data Analysis Software and Systems XXIII*, 485:391, 2014.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[22] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

[23] Rafael Müller, Simon Kornblith, and Geoffrey Hinton. When does label smoothing help? *arXiv preprint arXiv:1906.02629*, 2019.

[24] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[25] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

[26] Mohammad Hossein Rohban, Shantanu Singh, Xiaoyun Wu, Julia B Berthet, Mark-Anthony Bray, Yashaswi Shrestha, Xaralabos Varelas, Jesse S Boehm, and Anne E Carpenter. Systematic morphological profiling of human gene and allele function via cell painting. *Elife*, 6:e24060, 2017.

[27] Varadharajan Sundaramurthy, Rico Barsacchi, Nikolay Samusik, Giovanni Marsico, Jerome Gilleron, Inna Kalaidzidis, Felix Meyenhofer, Marc Bickle, Yannis Kalaidzidis, and Marino Zerial. Integration of chemical and rnai multiparametric profiles identifies triggers of intracellular mycobacterial killing. *Cell host & microbe*, 13(2):129–142, 2013.

[28] Adam B Castoreno, Yegor Smurnyy, Angelica D Torres, Martha S Vokes, Thouis R Jones, Anne E Carpenter, and Ulrike S Eggert. Small molecules discovered in a pathway screen target the rho pathway in cytokinesis. *Nature chemical biology*, 6(6):457–463, 2010.

[29] Cynthia L Adams, Vadim Kutsyy, Daniel A Coleman, Ge Cong, Anne Moon Crompton, Kathleen A Elias, Donald R Oestreicher, Jay K Trautman, and Eugeni Vaisberg. Compound classification using image-based cellular phenotypes. *Methods in enzymology*, 414:440–468, 2006.

[30] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[31] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[32] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487. PMLR, 2016.

[33] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5147–5156, 2016.

[34] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751, 2013.

[35] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688, 1993.

[36] Carrie J Cai, Emily Reif, Narayan Hegde, Jason Hipp, Been Kim, Daniel Smilkov, Martin Wattenberg, Fernanda Viegas, Greg S Corrado, Martin C Stumpe, et al. Human-centered tools for coping with imperfect algorithms during medical decision-making. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2019.

[37] Yu-An Chung and Wei-Hung Weng. Learning deep representations of medical images using siamese cnns with application to content-based image retrieval. *arXiv preprint arXiv:1711.08490*, 2017.

[38] Alex X Lu, Oren Z Kraus, Sam Cooper, and Alan M Moses. Learning unsupervised feature representations for single cell microscopy images with paired cell inpainting. *PLoS computational biology*, 15(9):e1007348, 2019.

[39] Christoph Sommer, Rudolf Hoefler, Matthias Samwer, and Daniel W Gerlich. A deep learning and novelty detection framework for rapid phenotyping in high-content screening. *Molecular biology of the cell*, 28(23):3428–3436, 2017.

[40] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[41] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[42] Nick Pawlowski, Juan C Caicedo, Shantanu Singh, Anne E Carpenter, and Amos Storkey. Automating morphological profiling with generic deep convolutional networks. *BioRxiv*, page 085118, 2016.

[43] Alexander Kensert, Philip J Harrison, and Ola Spjuth. Transfer learning with deep convolutional neural networks for classifying cellular morphological changes. *SLAS Discovery: Advancing Life Sciences R&D*, 24(4):466–475, 2019.

[44] Shai Ben-David and Ruth Urner. On the hardness of domain adaptation and the utility of unlabeled target samples. In *International Conference on Algorithmic Learning Theory*, pages 139–153. Springer, 2012.

[45] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. *Advances in neural information processing systems*, 27:766–774, 2014.

[46] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[47] Michael F Cuccarese, Berton A Earnshaw, Katie Heiser, Ben Fogelson, Chadwick T Davis, Peter F McLean, Hannah B Gordon, Kathleen-Rose Skelly, Fiona L Weathersby, Vlad Rodic, et al. Functional immune mapping with deep-learning enabled phenomics applied to immunomodulatory and covid-19 drug discovery. *bioRxiv*, 2020.

[48] Stanley R Sternberg. Biomedical image processing. *Computer*, 16(01):22–34, 1983.

[49] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[50] Thouis R Jones, Anne Carpenter, and Polina Golland. Voronoi-based segmentation of cells on image manifolds. In *International Workshop on Computer Vision for Biomedical Image Applications*, pages 535–543. Springer, 2005.

[51] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[52] Luc Vincent and Pierre Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(06):583–598, 1991.

[53] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[54] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[56] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[57] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[58] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[59] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

Appendix

A.1 Reconstruction architecture

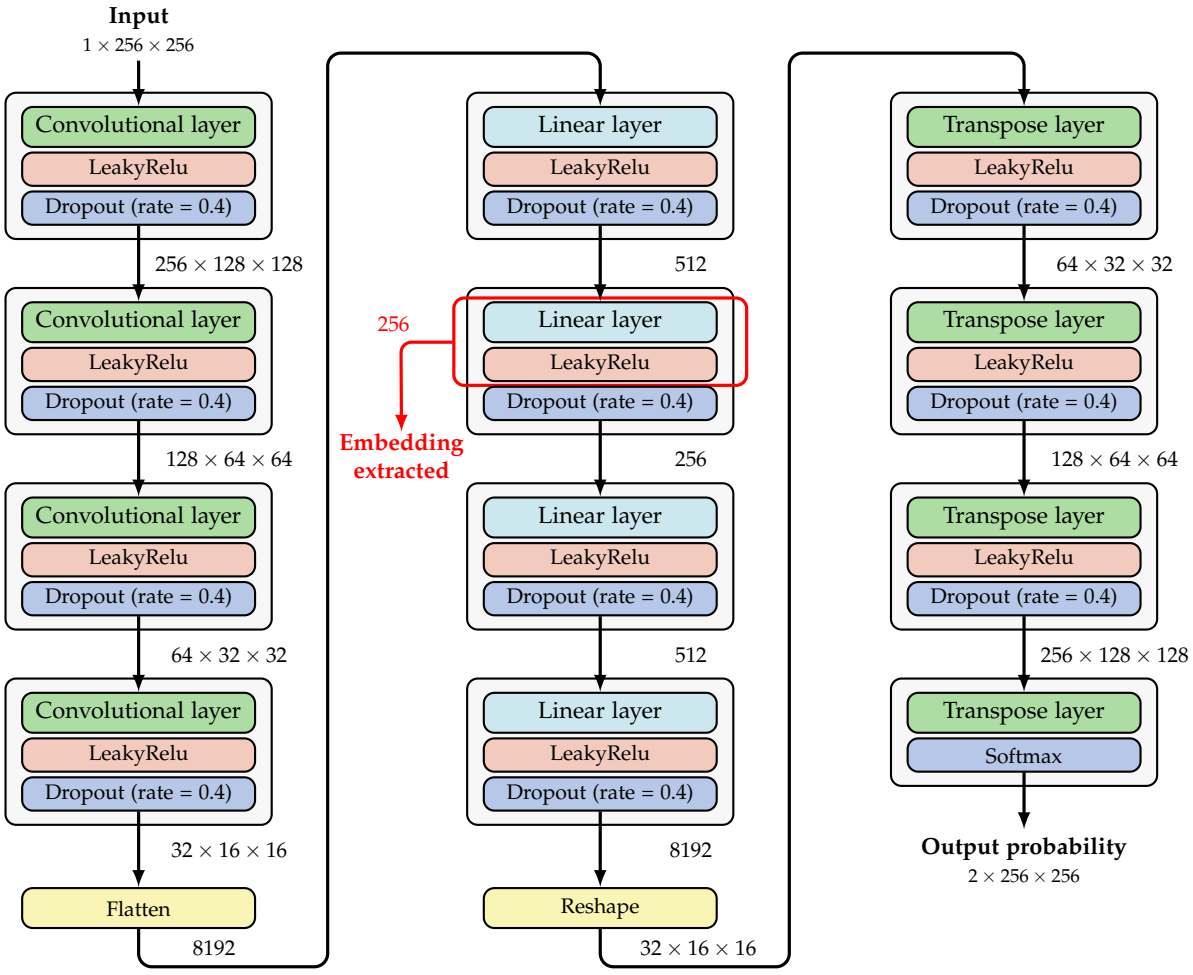


Figure A.1.1: Complete reconstruction architecture. Only one channel is used as input. The output represents a probability distribution for each pixel to be the object of interest (i.e. actin, nuclei or borders) or background.

A.2 Custom classification architecture

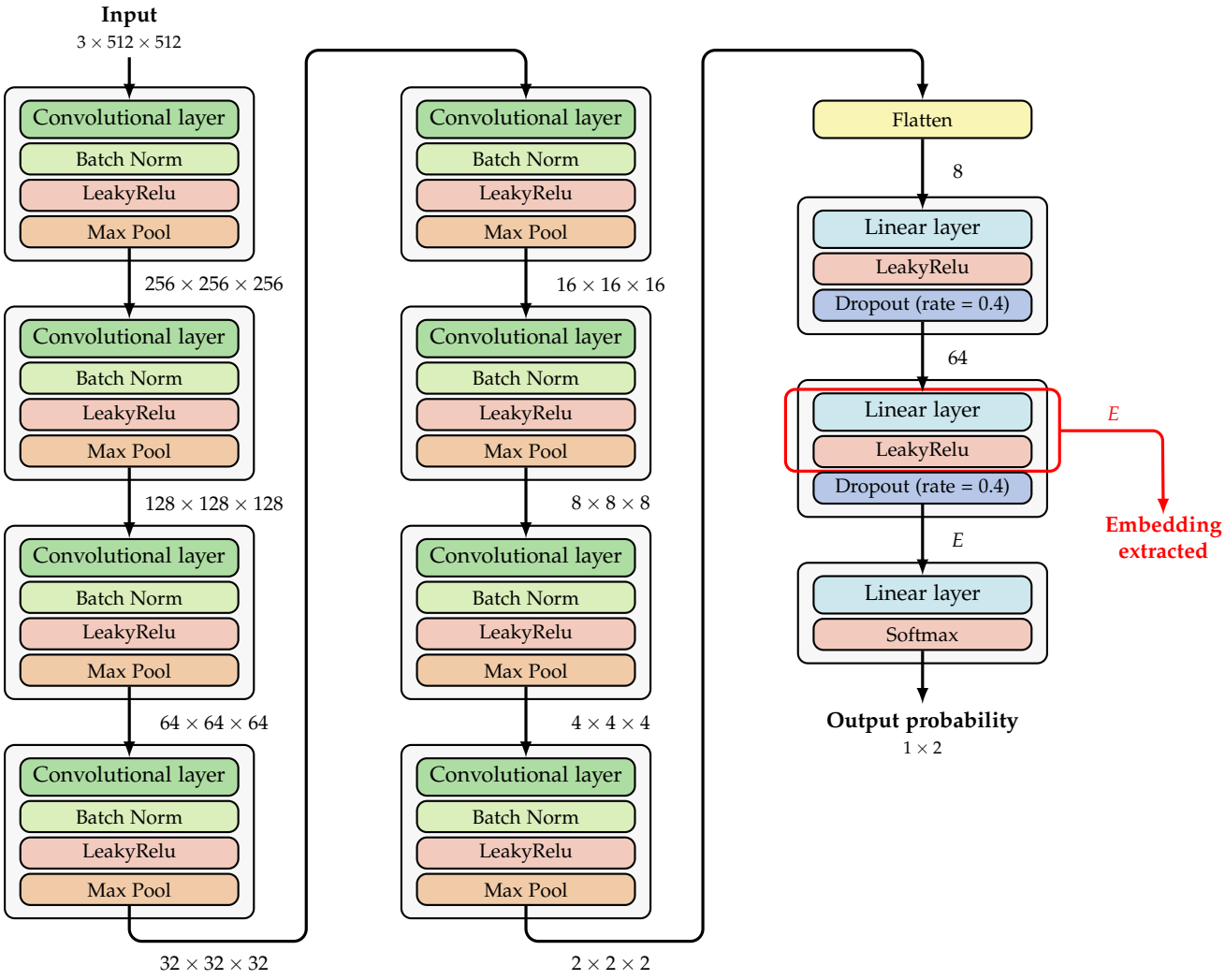


Figure A.2.1: Complete custom classification architecture, with E being the embedding size. In our case, $E = 8$.

A.3 DenseNet-121 based architecture

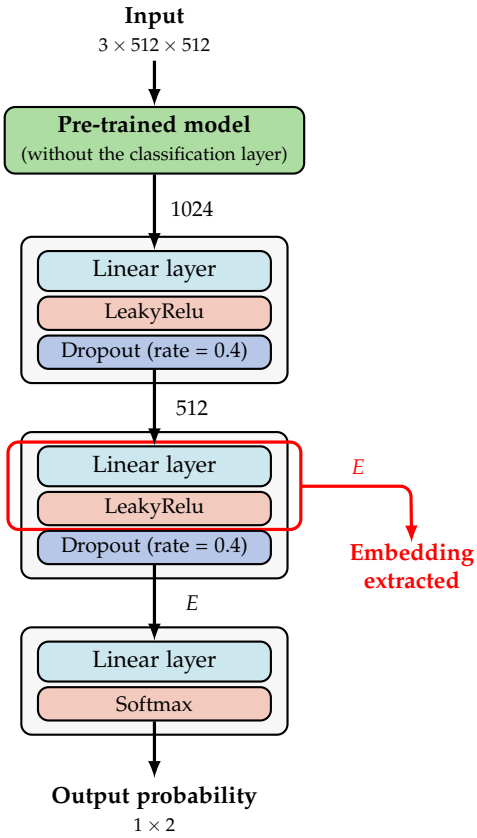


Figure A.3.1: Architecture of the classifier based on DenseNet-121 model, with E being the embedding size. In our case, $E = 8$.

B.1 Performance reconstruction model

Table B.1.1 includes the reconstruction loss on validation and test sets for each of the autoencoders trained on individual channels.

Channel	Loss		
	Train	Validation	Test
Nuclei	0.396	0.225	0.221
Borders	0.788	0.679	0.703
Actin	0.744	0.664	0.642

Table B.1.1: Loss values for the nuclei, borders and actin reconstruction autoencoders.

B.2 Performance classification model

Metrics		Baseline classifier (SVM)	Custom classifier	DenseNet-121 based
Loss	Train	-	0.553	**0.550**
	Validation		**0.550**	0.581
	Test		**0.537**	0.638
Accuracy	Train	0.603	0.742	**0.767**
	Validation		0.686	**0.752**
	Test		0.673	**0.696**
Recall	Train	**0.860**	0.646	0.782
	Validation		0.640	0.724
	Test		0.607	0.621
Precision	Train	0.637	**0.829**	0.776
	Validation		0.710	**0.810**
	Test		0.755	**0.842**
F1	Train	0.732	0.691	**0.744**
	Validation		0.650	**0.736**
	Test		0.636	**0.698**

Table B.2.1: Performance metrics for the two classifiers used in the project: the custom architecture and the neural network based on the DenseNet-121 architecture.

C.1 Learning curves for the classification model

Figure C.1.1 and Figure C.1.2 include the training and validation learning curves for the neural networks classifiers resulted from the data augmentation experiment.

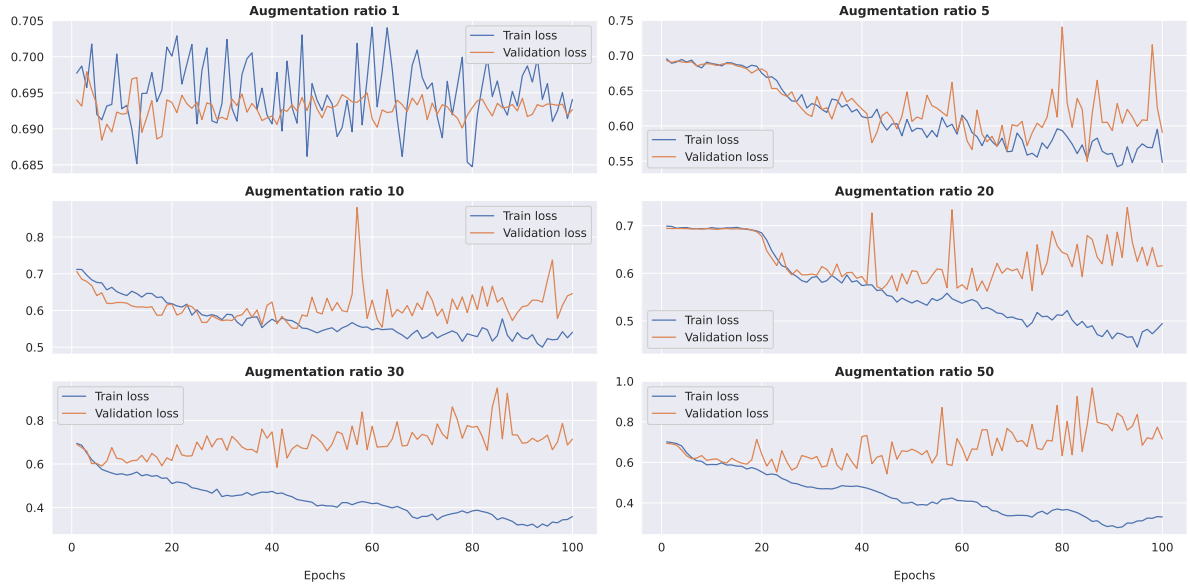


Figure C.1.1: Train and validation loss curves for the custom architecture using data sets augmented 1, 5, 10, 20, 30 and 50 times. The loss function is binary cross entropy.

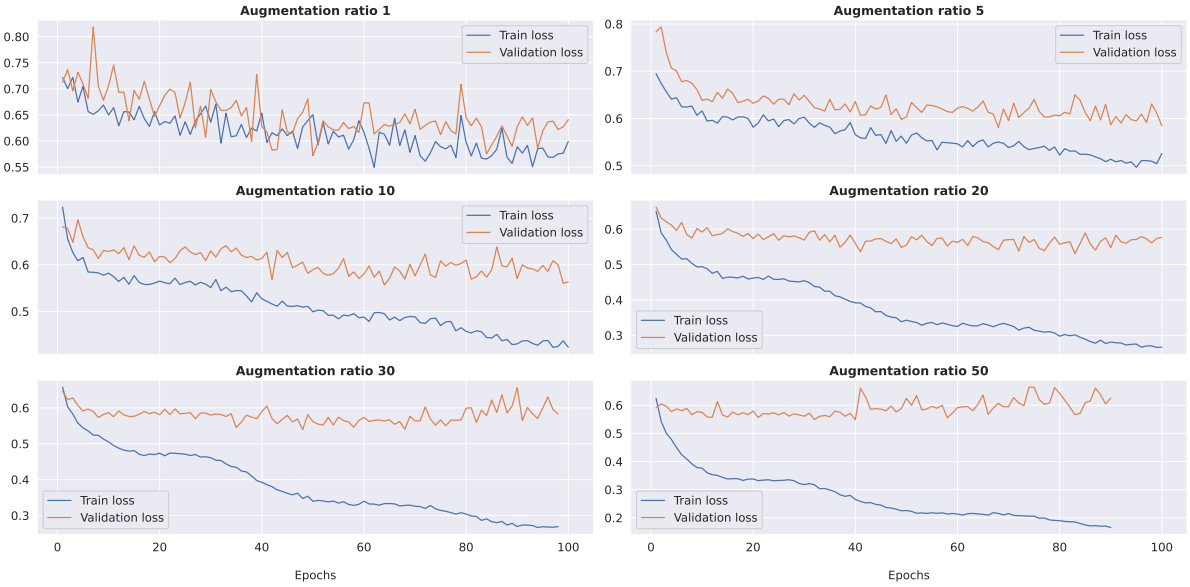


Figure C.1.2: Train and validation loss curves for the DenseNet-121 based architecture using data sets augmented 1, 5, 10, 20, 30 and 50 times. The loss function is binary cross entropy.