PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à MINES ParisTech

Deep learning for computational phenotyping in cell-based assays
Apprentissage profond pour le phénotypage numérique aux essais cellulaires

Soutenue par
Joseph BOYD
Le jj mois aaaa

École doctorale n°621
Ingénierie des Systèmes, Matériaux, Mécanique, Énergétique

Spécialité
Bio-informatique

Composition du jury :

Prénom NOM Titre, Établissement	*Président*
Prénom NOM Titre, Établissement	*Rapporteur*
Prénom NOM Titre, Établissement	*Rapporteur*
Prénom NOM Titre, Établissement	*Examinateur*
Prénom NOM Titre, Établissement	*Examinateur*
Prénom NOM Titre, Établissement	*Examinateur*
Prénom NOM Titre, Établissement	*Examinateur*
Prénom NOM Titre, Établissement	*Directeur de thèse*

MINES ParisTech | PSL

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Summary: *This thesis*

Résumé: *Ce chapitre...*

1.1 Computational phenotyping

The success of the Human Genome Project (Lander et al. [2001]) in mapping the totality of human genes has inspired similar efforts for the enumeration of biological phenotypes. One useful relation views a *phenotype* as, in conjunction with environmental factors, the manifestation of the genetic code. Natural selection has been said to operate in the "P-space" of all possible phenotypes, with selection propagating to the "G-space" of all possible genotypes. Hence, phenetic information bears directly on important biological questions regarding disease and mortality (Houle et al. [2010]).

Unlike a genotype, which is constrained to the configuration of a fixed number of genes, an organism's phenotype spans the space of its observable characteristics. This notion is somewhat problematic, as what is observable has broadened in time with advancements in technology. As a result, an organism's phenotype today may span its molecular, cellular, tissular, morphological, and behavioural traits, none of which are necessarily fixed in time. For this reason, it is often convenient to refer to a tractable subset of an organism's phenotype, for example that pertaining to an organismal subsystem, such as the properties of its cells. As a result, the term *phenotype* and the related *phenome* are in practice want of semantic hygiene (Mahner and Kary [1997]).

Phenomics, by analogy to genomics, refers to the study of high-dimensional readouts across the full spectrum of an organism's phenotype (Houle et al.

[2010]). The interest in phenomics coincides with the rise of bioimage informatics (Myers [2012]). Bioimages, deriving from the various forms of microscopy, contain rich information on the morphological aspects of cells, cell populations in culture or tissue, and complete organisms, that is lost in other biological readouts. Bioimages in time-lapse can additionally reveal behaviour and track phenotypic changes in motion. Thus, bioimages are a favourable medium for phenotypic information.

1.1.1 High content screening

High content screening (HCS) is a methodology for the systematic discovery of phenotypes from image data in cellular assays (Haney [2008]). HCS may be viewed as extending the methodology of high throughput screening (HTS) to the medium of images. HTS is an experimental setup to test many experimental conditions in a systematic way, typically with a very simple readout, for example cell viability or a univariate measure of cytotoxicity. HCS performs such screens with a more complex and informative readout by leveraging, in particular, multiple channels of fluorescence microscopy. HCS thus increases the "content" of the readout to a large number of features (perhaps hundreds), while maintaining the throughput. HCS can be used for fundamental biological research, where gene expression can be modulated via techniques such as RNA interference, or otherwise knocked out entirely, inducing phenotypic effects in cultured cells. HCS has been instrumental in deciphering the molecular basis of a number of diverse biological processes, such as cell division (Neumann et al. [2010]), protein secretion (Simpson et al. [2012]), and endocytosis Collinet et al. [2010]. HCS has also been used to systematically screen for the localisation of biomolecules inside cells Boland et al. [1998]).

HCS also plays a role in the early "hit-to-lead" stages of the drug discovery process (Haney et al. [2006], Pepperkok and Ellenberg [2006a]). In this case, a cell line population is exposed to *small molecule* drug compounds. Thus, the cell line is the model for a disease, and the screening process aims to identify the drugs that are active thereupon. For instance, we may be interested in identifying drugs that specifically kill cancer cells. HCS complements other techniques for drug identification such as biochemical assays. For instance, Swinney and Anthony [2011] differentiate target-based and phenotypic screens. The study looks at 257 drugs published between 1999 and 2008. Despite the preeminence of target-based approaches, they find the most common mode of first-in-class drug discovery is phenotypic screening. This is most true of infectious and central nervous system diseases. Cancer treatments are most frequently discovered by biologics[1], which also

[1] Biologics are genetically-engineered proteins that target the immune system.

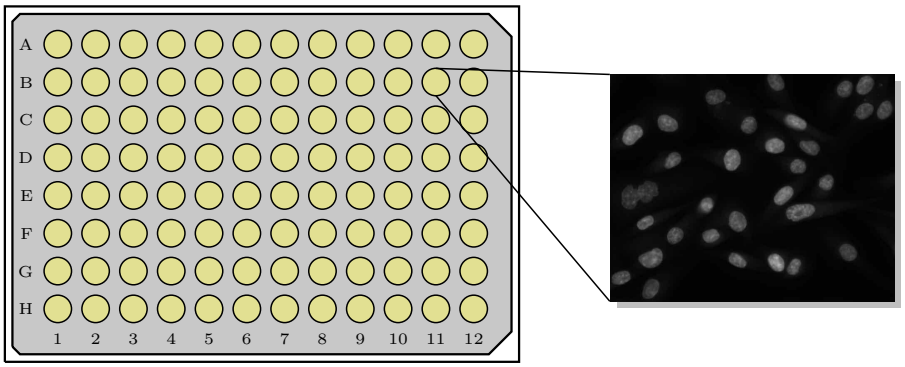


Figure 1.1: The basis of a systematic high content screen. A microplate from which is registerd a fluorescence (single-channel) microscopy image highlighting nuclei of cell population.

predominate for diseases of the immune system. Target-based approaches succeed for the discovery of half of the follower drugs.

1.1.2 The elements of high content screening

In order to perform experiments in high throughput, HCS relies to a significant degree on experimental automatisation. Experiments are conducted in wells arranged in a grid structure on a microtiter plate[2] as depicted in Figure 1.1. The wells are seeded to *confluence*[3] with cells of a chosen cell line. Within each well, a different perturbation experiment is conducted. In all biological experiments it is important to rely on proper controls against which the tested perturbations are compared. Screens compare populations of negative controls (unperturbed) with others exposed to perturbation. Positive controls take the form of perturbations with a known effect, such as small molecule cytotoxicity in a drug screen. Some number of images corresponding to non-overlapping *fields of view* (we henceforth use the term *fields*) are taken from each well, typically producing from hundreds to thousands of unique images per plate. In the following we briefly describe the key elements of a screen.

Cell lines

The object of analysis of an HCS assay is an immortalised cell line (hereafter referred to as a *cell line*), which consists in a population of cells, sustained

[2]Microtiter plates (hereafter *plates*) are small (usually polystyrene) trays divided into a rectangular grid of wells functioning as individual test tubes.

[3]Such that the well surface is covered with cells

by division without senescence. As a result, a cell line continues replicating indefinitely from a common ancestor. Cell lines are the biological model acting as a representation of the disease. The first and most well-known cell line generated is the HeLa cell line Scherer et al. [1953]. Since HeLa, many cell lines have been developed. Cell lines are useful biological models due to their longevity in cell culture, and are used extensively in biomedical research, for example in assessing the cytotoxicity of a drug treatment. However, their accuracy as biological models can be compromised by their essence as mutated cells, and the effects of repeated passages, cloning, and biochemical contaminants can lead to significant genetic drift from their *in vivo* ancestors (Marx [2014]). Despite these limitations, cell lines remain the most widely used model system used in screening, and many scientific and pharmacological discoveries have been made from screens on cell lines.

Microscopy and fluorescence

In high content screening, imaging data derives from one of another of the many types of optical microscopy. A broad range of techniques for performing microscopy exists, each leveraging the principles of optics in different ways, and the technique will be tailored to the experimental objectives. *Bright field* microscopy passes visible light through a sample from below, producing a picture in which light is attenuated according to the varying densities of the imaged specimen. *Phase contrast* is a more sophisticated variant of transmitted light microscopy, measuring the phase shift of the visible light traveling through the specimen, producing an image with a greater degree of contrast.

Fundamental to high content screening is *fluorescence microscopy*, which uses a laser to excite fluorescent molecules in organic matter. These molecules are known as *fluorophores* and emit light at a unique wavelength (think colour) upon excitation. As such, localised fluorophores can be utilised to highlight key cellular regions. The predominant technique used in HCS is immuno-fluorescence, which relies on fluorescently-labeled antibodies. Other widely used techniques include live dyes, stable expression or fluorescence in situ hybridization. In most screening applications, the nuclei are stained with one of these techniques, allowing for the subsequent identification of individual cells. The other fluorescent markers are selected in accordance with the research questions, for example, microtubules, Golgi apparatus, or plasma membrane. Thus, the fluorescence markers of a screen, responding to distinct wavelengths of light, yield a set of multiplexed images painting a composite picture of the cell in its key sub-cellular structures. Image analysis can then be used to extract the high content of the screen.

Project workflow

A typical HTS project workflow commences with a pilot screen that validates the pipeline from experimental protocol through to image analysis (see Terjung et al. [2010]). This is followed by a large scale screen, increasing the number of perturbations tested. It is here that candidate *hits* (drug perturbations registering a significant effect) are identified by automatic analysis (Section 1.1.3. Finally, candidate hits are carried forward to secondary screens, involving a greater degree of detail.

1.1.3 High content analysis

With a large image dataset in hand (Section 1.1.2), so begins the automated image analysis, or *high content analysis*. Each image depicts a population of cells subject to perturbation or else representing a control case. The aim is to attribute a phenotype to the population in terms of a specific or vector of measurements. Analysis of the pixel values across the various fluorescent channels yield a set of features or *readouts*. Cell phenotypes are compiled through feature extraction of each measured unit of the image. The distribution of perturbed readouts can be compared to control cases in a statistical framework so as to establish screen hits. In order of complexity, the readouts may be categorised according to the following:

I **Univariate**: In the ideal case, biological functions may be quantitatively described by a single feature. For example, the nuclear area might increase dramatically under certain treatments. Thus, it would be sufficient to measure the corresponding feature (nuclear size) and statistically analyse its distribution for the different experimental conditions. Such a scenario would likely be easier to explain in biological terms.

II **Multivariate**: A more complex case arises when we analyse different phenotypic descriptors (biologically meaningful features) and their interdependencies. Then we would contend with the multi-variate distribution of these features, and our analysis would be necessarily more sophisticated.

III **Machine learning**: In a final case, phenotypes might not be discernible in such basic terms, and would rather require the tools of statistical learning to elucidate the patterns in the cell population. In this case, we would rather extract a large number of features without clear biological meaning. Learning can then be unsupervised or supervised. In the supervised case, the biological meaning could be injected (by the analyst) through use of biologically meaningful classes.

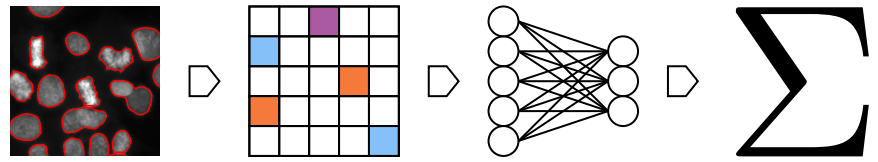


Figure 1.2: A conventional high content analysis follows four ordered stages. Each stage may be accomplished by a variety of algorithms, and some stages may be omitted in certain pipelines, or subsumed to a common framework.

These readouts may be taken at various scales, however a logical (and indeed, conventional) starting point is at the level of individual cells (Perlman et al. [2004], Adams et al. [2006]), entailing an initial segmentation of the image. As a side note, it is for this reason that it is favourable to seed cells at a density that will minimise cell overlap. The seeding density must therefore be chosen according to the unique morphological properties of the cell line (Bray et al. [2016]). However, *segmentation-free* approaches directly analysing the full image (Orlov et al. [2008], Uhlmann et al. [2016]), or image segments, have proven successful, in particular through application of deep learning (Kraus et al. [2016]). The tradeoff is between a fine-grained analysis at the cellular level, where careful consideration of cell structure and fluorescent colocalisation is a focus (Slack et al. [2008]), and a coarse analysis of the cell population, where population densities and dynamics can be measured. Attempts to benefit from both scales have been made (Godinez et al. [2017]). After these early stages of image analysis, a screen dataset is usually subject to a range of *quality control* procedures, which may use automatic techniques to detect artifacts such as image blur or saturation, or else detect outliers among cells that have been under- or over-segmented Caicedo et al. [2017].

After these initial stages, in particular a type III analysis may proceed towards a *phenotypic profiling* of the cell population. Figure 1.2 encapsulates a conventional approach to profiling. Dimensionality reduction is used variously to eliminate redundant features, as well as to compress the data into its essential components (for example, the set of principal components derived over the population of cell feature vectors). Here, the options abound and the choice of approach is determined by the analytical objectives. A simple, motivating example is the case of counting cell class among a population. Here, a classifier such as in Neumann et al. [2010], performs the role of dimensionality reduction: the classifier maps the feature vector of each cell to a scalar or one-hot encoding representing cell class,

$$f : \mathbf{x} \to \{0, 1\}^K, \tag{1.1}$$

for K classes of cells. The population phenotype is then summarised in $\mathbf{p} \in \mathbf{R}^K$, obtained by aggregation as a simple summation or average, giving the number or proportion of cells per cell class respectively,

$$\mathbf{p} = \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}_i) \tag{1.2}$$

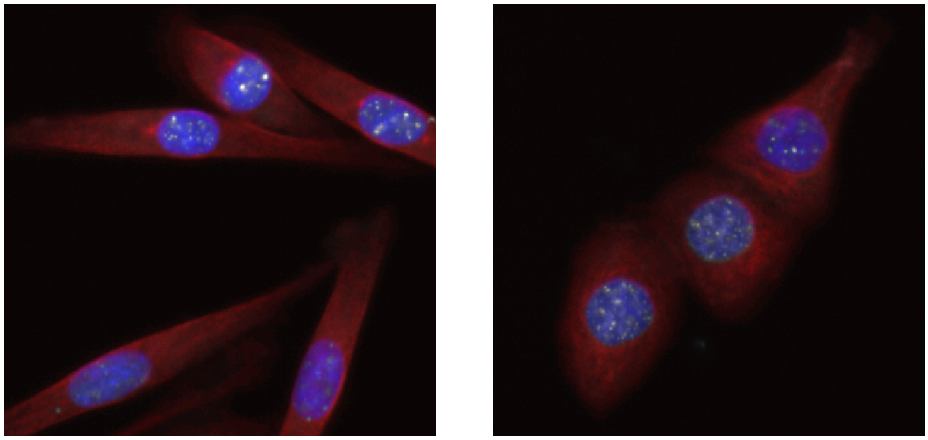
for the N cells in the population.

1.2 Challenges for high content analysis

High content analysis offers many interesting research directions. Those most relevant to this dissertation are described in the following.

1.2.1 Multi-cell-line data

As mentioned above, the use of a single cell line limits HCS as a drug screening approach. Diseases and, in particular, cancer, are often heterogeneous on a molecular level. This translates therapeutically to cases where a treatment effective against one molecular subtype is ineffective (or even promotes genetic instability) against another subtype. Hence, a single cell line introduces a bias towards a particular disease subtype. This motivates the validation of discoveries against multiple cell lines, representing different subtypes of the same disease. A screen based on multiple cell lines, representing multiple subtypes can help in formulating hypotheses on, for example, the mechanism of resistance to disease. This is furthermore a step in the direction of the emerging paradigm of precision medicine (Ashley [2016]), where machine learning will play a decisive role (see, for example, Krittanawong et al. [2017]). Genomics has led the way so far, but other data sources including images are expected to become increasingly part of the picture Hulsen et al. [2019]. This has motivated a large consortium of research groups to propose multi-cell-line drug screens, and to build models capable of predicting the efficiency of a drug from the transcriptomic and genetic data of the cell-line Costello et al. [2014]. However, the success of these approaches has been rather modest. One of the reasons for this was the measurement of drug efficiency, that reduced the drug effect to a single number. Consequently, drug effect similarities cannot be reasonably



(a) MDA231 (b) MDA468

Figure 1.3: Comparison of cells sampled from negative control wells containing a) TNBC cell line MDA231 and b) TNBC cell line MDA468. Cell nuclei appear in blue, cell microtubules appear in red.

calculated from this data. We therefore stand to gain from characterising multi-cell-line drug effects rather in high content phenotypic profiles.

A key use case for HCS in drug screens is the related problem of target prediction, that is, the prediction of the pathway or protein whose function is altered by the drug. This invokes a classification task on the image data, namely, to determine what mechanism of action (MOA) is responsible for inducing the effects visible in the perturbed populations, often adjudicated with reference to the control populations. Screening several cell lines with different genetic and transcriptomic profiles allows one to test more pathways (as not all genes are expressed in all cell lines) and, in so doing, to get a richer description of the MOA of the drug. Quantifying drug effects with respect to multiple cell lines allows to distinguish cell-line-specific drug effects from unilateral effects. A multi-cell line screen, in which several cell lines representative of a common disease are subjected to the same set of perturbations, is therefore well motivated. However, such a screen creates the challenge of a heterogeneity of data. For example, TNBC cell lines MDA231 and MDA468 manifest different morphologies in their unperturbed state. Figure 1.3 shows samples of cells from these two cell lines, taken from microplate wells containing negative control dimethyl sulfoxide (DMSO), with the cells imaged with the same set of fluorescent markers. One observes clear differences in these archetypal morphologies, with MDA231 cells exhibiting elongated nuclear and cellular shape, compared to the isotropic MDA468 cells. This has additional, second-order effects such as the degree of geometric tessellation between groups of cells. Note that

in single-cell line data already, population phenotypes aggregated from the single cell constituents abandon distributional information (Altschuler and Wu [2010]), as multi-modal populations are often reduced to unrepresentative centroids. This problem is potentially exacerbated in multi-cell-line data and remains a challenge. Combining multi-cell-line image data has so far been addressed by few approaches, for example Rose et al. [2018] and Warchal et al. [2016].

1.2.2 The emergent role of deep learning in high content screening

Deep learning is touted as a panacea for computer vision problems, and high content data ought not to be an exception. In applications of deep learning, a neural network may perform several of the Figure 1.2 stages simultaneously, as neural networks naturally incorporate feature extraction and dimensionality reduction components (Sommer et al. [2017]). However, the successful deployment of deep neural networks relies crucially on vast volumes of data to enable effective generalisation. Large annotated datasets such as ImageNet (Russakovsky et al. [2015]) were one of a few key preconditions that fostered the rise of deep learning for object classification in 2012 (we give a brief history in Chapter 2), and extensions of deep learning to other problem domains were likewise accompanied by the curation of large, special-purpose datasets, for example Lin et al. [2014] for object detection. However, annotated data for supervised training is expensive, requiring manual effort, often by domain experts. ImageNet leverages online crowdsourcing platforms (quality is ensured by the consensus of multiple annotators). This is a bottleneck for all deep learning research, and therefore extends to computational phenotyping. The Broad Institute benchmark collection (BBBC) datasets (Ljosa et al. [2012] and, specifically, Caie et al. [2010]) have become a sort of benchmark for developing drug response phenotyping algorithms (for example, Kraus et al. [2016] or Kandaswamy et al. [2016]). However, while benchmark data sets are of course useful and have had an enormous impact on the field, we still face the problem that for new imaging projects, we do not have enough data to train neural networks. This relates to the fact that bio-images tend to be extremely variable between different projects: the visual aspect is heavily influenced by the choice of markers and the mode of microscopy. Indeed, by selecting different markers, we are effectively looking at different objects. For this reason, it seems unlikely that large scale datasets will definitively solve the problem of annotated data, except for the most widely used markers and imaging modalities.

Nevertheless, in recent years, a significant trend in deep learning research has been on making better use of available data. After all, even if annotated data

is hard to come by, unlabeled or *weakly labeled* is available in abundance. For example, Mahajan et al. [2018] used 3.5 billion social media images "weakly annotated" with hashtags to pretrain a state-of-the-art system for object classification. Indeed, notable applications of deep learning in HCS thus far have relied on weakly supervised learning Kraus et al. [2016], Godinez et al. [2017]. Elsewhere, contrastive learning (for example, Chen et al. [2020]) may yet revolutionise the training of deep learning systems by making more efficient use of data, achieving parity with state-of-the-art systems with only a small faction of data.

A recent trend in bioimage analysis has been the prediction of one mode of microscopy from another. Microscopes with the capacity of registering multiple modes of microscopy simultaneously, for example transmitted light and fluorescence images, automatically create an image-to-image translation dataset. Fluorescence labeling as a pixel-wise regression problem has been successfully demonstrated by Christiansen et al. [2018] and Ounkomol et al. [2018], where deep multi-task neural networks are furthermore capable of labeling multiple independent fluorescent channels simultaneously. Earlier, Sadanandan et al. [2017] used fluorescence to construct cell segmentation datasets automatically. These works have shown how one may exploit imaging protocols to bypass the manual annotation bottleneck for deep learning. We explore this possibility in Chapter 5, contrasting two approaches for leveraging fluorescence as an automatic annotator.

Generative models represent another trend in computer vision (Kingma and Welling [2013], Goodfellow et al. [2014a], Oord et al. [2016]), modeling the marginal distribution on data, allowing for data synthesis, in particular image synthesis, as well as unsupervised representation learning. Generative adversarial networks (GANs) (Goodfellow et al. [2014a]) are the most highly developed of deep generative models, and have already found use in high content image data for example, in Osokin et al. [2017]). Elsewhere, generative models such as variational autoencoders (Kingma and Welling [2013]) have found use in phenotyping drug effects for MOA prediction. Image-to-image translation (see above) is also addressed by generative models capable of deep style transfer (Isola et al. [2017], Zhu et al. [2017]). Data synthesis is additionally a form of data augmentation, which in turn is an attempt to make more effective use of a scarce data supply. Moreover, it is one possible route towards the simulation of image data (see Ihle et al. [2019]). Unlike more standard deep learning models, however, generative models are more difficult to train, and may require creative and non-standard solutions. Therefore, as much as the role of deep learning in high content analysis is not yet fully realised, the role of generative models is ever more so, and their application represents a fertile research direction.

1.3 Contributions

This dissertation encompasses work completed on two unique projects in high content analysis. It is therefore organised into two parts, each containing two chapters. The two parts can be read in any order, but they internally follow a progression of ideas that are intended to be read in order of appearance. Each chapter is oriented around a published paper, with the exception of the final chapter, which follows a work in progress.

Part I, comprising of Chapters 3 and 4 covers our high content drug screen of multiple TNBC cell lines. TNBC is a molecularly heterogeneous type of breast cancer with poor prognosis and limited treatment options. Its molecular heterogeneity make finding treatments difficult, and a screen of multiple cell lines is therefore promising. Chapter 3 is introductory and aims to describe the workflows of high content screening and its modes of analysis, while following examples from our drug screen, including the multivariate analysis on *double strand break* detection, published in the proceedings of ISBI 2018 (Boyd et al. [2018]) (which we include in Appendix D. It concludes with a pair of case studies on *phenotypic profiling*, that are intended to dovetail into *Chapter 3*, which serves as a review of profiling methodologies, and develops a new profiling approach for multiple cell lines. This approach is to combine heterogeneous data from different cell lines using domain adaptation. Cells from the divergent domains are mapped to a *domain-invariant* feature space by an adversarial neural network training strategy (Ajakan et al. [2014]). The chapter is an extended version of our paper published in the journal Bioinformatics (Boyd et al. [2020]). We additionally released the dataset for this project, to our knowledge, the first of publicly available dataset of its kind (Boyd et al. [2019a]) alongside an open source code repository including worked, reproducible workflows[4]. In spite of the idealised workflow presented in Section 1.1.2, we are restricted in Part I to the pilot phase of a planned larger screen. We nevertheless find ample phenotypes to observe and on which to develop new methods.

Part II, comprising of Chapters 5 and 6 covers our CAR-T experiments, where we study the Raji cell line as a model for lymphoma. Though not strictly based on a screen, Part II employs high content analysis and shares many characteristics with Part I. Chapter 5 compares two approaches to utilising fluorescence microscopy as an automatic annotator of paired phase contrast microscopy. The second of the two approaches, based on a customised object detection system, was published in the proceedings of ISBI 2020. We again made our datasets public (Boyd et al. [2019b]) as well as source code and scripts for reproducible experiments[5]. A modified version

[4] https://github.com/jcboyd/multi-cell-line
[5] https://github.com/jcboyd/detecting-lymphocytes

of this paper is included inline. Finally, the shortcomings of the two approaches are reconsidered in Chapter 6, and a creative alternative solution is proposed, based on data augmentation by image synthesis using generative models. The final section of the chapter is intended as a template for a future publication.

Not included in this dissertation are minor contributions made as a second author to a study on predicting residual cancer burden from TNBC histopathology images, published in the proceedings of ISBI 2019 (Naylor et al. [2019]), and an as-of-yet unpublished paper on a new structured dropout algorithm for regularising neural networks Khalfaoui et al. [2019].

Chapter 2

Deep learning fundamentals

Summary: *This disseration makes extensive use of artificial neural networks and deep learning. In this chapter we detail the basic properties of neural networks and extend show how these ideas extend to the deep, convolutional variety of networks, key to modern image processing. We further trace the history of the development of deep learning, and its extension into the various problem domains of computer vision*

Résumé: *Ce chapitre...*

2.1 The building blocks of neural networks

A neural network is a collection of computational units known as *neurons*, organised into a sequence of *layers*. An input passes *forward* through a neural network, undergoing a series of transformations through combination with a set of *weights* at each layer. During a training procedure, the neural network is shown examples of input data paired with target outputs. After each round of training, the neural network adjusts its (randomly initialised) weights so as to make it a little more likely to emit the target values given future appearances of the input data.

A simple neural network invokes one or more *hidden layers* between input and output, for example,

$$\begin{aligned} \mathbf{h}(\mathbf{x}) &= \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) && \text{(hidden layer)} \\ \mathbf{f}(\mathbf{x}) &= \mathcal{S}(\mathbf{W}^{(2)}\mathbf{h} + \mathbf{b}^{(2)}) && \text{(output layer)} \end{aligned} \tag{2.1}$$

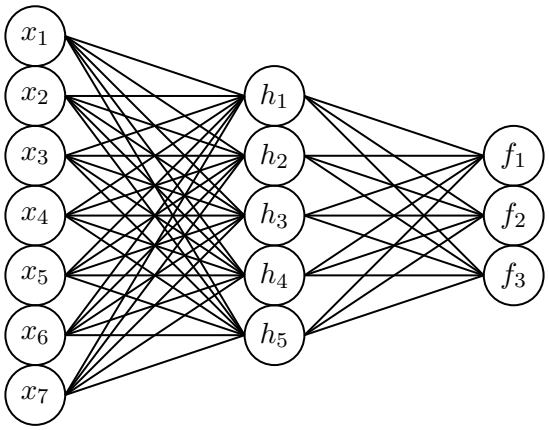


Figure 2.1: Fully-connected neural network with 7 input neurons, a single hidden layer with 5 neurons, and an output layer with 3 neurons.

for input vector $\mathbf{x}$, first layer weights and biases matrix $\mathbf{W}^{(1)}$ and vector $\mathbf{b}^{(1)}$, and second layer weights and biases $\mathbf{W}^{(2)}$ and vector $\mathbf{b}^{(2)}$. The function σ is a non-linear *activation function*[1] and $\mathcal{S}$ is the softmax function at the output layer. The softmax is optional but often used for classification problems as it maps its inputs to categorical probabilities. In this case, the network is normally trained against a cross entropy loss function,

$$\mathcal{L} = \sum_i -\log(p_{y_i}), \tag{2.2}$$

where $p_{y_i} = f(\mathbf{x}_i)_{y_i}$ is the network output (probability) for the ith training input $\mathbf{x}_i$, indexed at the (paired) ith target label y_i . Equation 2.2 is a simplification of the cross entropy formula for the case where the y_i are "one-hot" vectors. Note that neural networks can be trained for classification, regression, or unsupervised feature extraction, depending on the requirements of the data. Each entails a different output layer and loss function for the network. Figure 2.1 depicts an examle of our simple neural network *architecture*.

The neural network framework gives us freedom over the number of layers and the number of neurons in each layer. The more neurons, the more expressive the network, yet the more likely overfitting becomes[2]. We may

[1] Historically, this was the logistic function, $\sigma(x) = 1/(1 + \exp\{-x\})$, a continuous approximation to the Heaviside step function (think on/off), however, in recent years, it has been superseded by the ramp function, more commonly known as the rectified linear unit, $\text{ReLU}(x) = max(0, x)$, which provides various training benefits.

[2] It is rare to require more than three layers, however, and exceeding this amount will be done to exploit deep hierarchical structures in the data.

extend to a multi-layer network simply by stacking the desired number of layers,

$$\mathbf{f}(\mathbf{x}) = \mathcal{S}(\mathbf{W}^{(M)}\sigma(\mathbf{W}^{(M-1)}(\cdots\sigma(\mathbf{W}^{(1)}\mathbf{x}+\mathbf{b}^{(1)})\cdots)+\mathbf{b}^{(M-1)})+\mathbf{b}^{(M)}) \quad (2.3)$$

Thus, the network layers are linear transformations interspersed with non-linear functions. When the inputs correlate positively with a vector of weights, the corresponding hidden layer neuron tends to become active via the non-linearity. In this way, each neuron of a hidden layer amounts to a feature detector for the prior layer. The hidden layer activations are then recombined in the next layer, and so on, allowing for ever more complex aggregations of features. This is the essence of deep learning. However, this tends not to work very well without certain inductive biases, (and, indeed, an amenable dataset) which we will encounter in Section 2.2. Despite their inherent non-linearity, and non-convexity, neural networks can be trained with standard gradient descent,

$$\theta_{t+1} \leftarrow \theta_t - \alpha \cdot \nabla_\theta \mathcal{L} \quad (2.4)$$

for the full set of model parameters θ and *learning rate* α. More sophisticated update rules than vanilla gradient descent exist such as RMSprop (Tieleman and Hinton [2012]), and Adam (Kingma and Ba [2014]). These methods improve over the Equation 2.4 by adapting a learning rate for each parameter (rather than one-size-fits-all). Whatever the chosen method, the gradients are always computed using the backpropagation algorithm, which we describe presently.

2.1.1 Backpropagation

The procedure for computing the gradients at each iteration of gradient descent is called backpropagation. Backpropagation is an application of the chain rule to the graphical structure of the neural network. The crucial formula for backpropagation, as presented in Rumelhart et al. [1985] is,

$$\frac{\partial L}{\partial x_i} = \sum_j \frac{\partial L}{\partial y_{ij}} \cdot \frac{\partial y_{ij}}{\partial x_i}, \quad (2.5)$$

that is, the gradient of neuron x_i at a given layer l is the sum product of the gradients of the neurons emanating from it y_{ij} and the gradient of the connection between them. From a practical standpoint, each layer l over which we perform backpropagation requires three computations:

1. $\frac{\partial L}{\partial \mathbf{s}^{(l)}}$, the gradient of the *scores*, $\mathbf{s}^{(l)} = \mathbf{W}^{(l)}\mathbf{x}^{(l)} + \mathbf{b}^{(l)}$. Note this is a "pseudo-layer" between the linear transformation and the activation.

2. $\frac{\partial L}{\partial \mathbf{W}^{(l)}}$, the gradient of the weights. These are recorded to ultimately make the descent step (Equation 2.4).

3. $\frac{\partial L}{\partial \mathbf{x}^{(l)}} = \frac{\partial L}{\partial \sigma^{(l-1)}}$, the gradient of the input (previous activation). This is the *error signal* that is passed back to the layer below.

Let us first consider (1), the gradient of the scores. In general, we compute, $\frac{\partial L}{\partial \mathbf{s}^{(l)}} = \frac{\partial L}{\partial \boldsymbol{\sigma}^{(l)}} \cdot \frac{\partial \boldsymbol{\sigma}^{(l)}}{\partial \mathbf{s}^{(l)}}$ where $\frac{\partial L}{\partial \boldsymbol{\sigma}^{(l)}} = \frac{\partial L}{\partial \mathbf{x}^{(l+1)}}$, since the activation of the present layer is the input to the following layer. The activation function σ is applied element-wise. Consequently, in the pseudo-layer between linear transform and activation, each neuron has a single connection. Therefore, with respect to Equation 2.5, the sum reduces to a single element per gradient giving,

$$\frac{\partial L}{\partial \mathbf{s}^{(l)}} = \begin{bmatrix} \frac{\partial L}{\partial \sigma_1^{(l)}} \cdot \frac{\partial \sigma_1^{(l)}}{\partial s_1^{(l)}} \\ \vdots \\ \frac{\partial L}{\partial \sigma_k^{(l)}} \cdot \frac{\partial \sigma_k^{(l)}}{\partial s_k^{(l)}} \end{bmatrix}, \tag{2.6}$$

where for sigmoid, $\frac{\partial \sigma_i^{(l)}}{\partial s_i^{(l)}} = \partial \sigma_i^{(l)}(1-\sigma_i^{(l)})$, and for ReLU, $\frac{\partial \sigma_i^{(l)}}{\partial s_i^{(l)}} = 1_{\{\sigma_i^{(l)}>0\}}$.

For computation (2), the weights gradient, consider, $\frac{\partial L}{\partial w_{kj}^{(l)}} = \frac{\partial L}{\partial s_j} \cdot \frac{\partial s_j}{\partial w_{kj}^{(l)}} = \frac{\partial L}{\partial s_j} \cdot x_j$. In vector form this gives us, $\frac{\partial L}{\partial \mathbf{s}^{(l)}}\mathbf{x}^T$, that is, the outer product of the score gradient and the input vector. Considering that the loss over a batch is the sum of losses for each sample, we have in general, for batch $\mathbf{X} \in \mathbb{R}^{D \times M}$,

$$\frac{\partial L}{\partial \mathbf{W}^{(l)}} = \frac{\partial L}{\partial \mathbf{s}^{(l)}} \cdot \mathbf{X}^T \tag{2.7}$$

For the bias terms, consider that with the bias trick, the gradients $\frac{\partial s_j}{\partial b_k^{(l)}} = 1, \forall k$. We can therefore simply sum the score gradients over the size m batch, $\sum_m \frac{\partial s_{mj}}{\partial b_k^{(l)}}$.

Finally, for computation (3), simply consider the chain rule, $\frac{\partial L}{\partial \mathbf{x}^{(l)}} = \frac{\partial L}{\partial \mathbf{s}^{(l)}} \cdot \frac{\partial \mathbf{s}^{(l)}}{\partial \mathbf{x}^{(l)}}$. It is easy to show by forming the Jacobian matrix that $\frac{\partial \mathbf{s}^{(l)}}{\partial \mathbf{x}^{(l)}} = \mathbf{W}^{(l)}$. Hence,

$$\frac{\partial L}{\partial \mathbf{x}^{(l)}} = \frac{\partial L}{\partial \mathbf{s}^{(l)}} \cdot \mathbf{W}^T \tag{2.8}$$

This final gradient is passed to the previous layer as $\frac{\partial L}{\partial \boldsymbol{\sigma}^{(l-1)}}$ to continue the backpropagation. Upon traversing the network layers, the weight gradients are used to update to weights as in Equation 2.4.

2.2 Convolutional neural networks

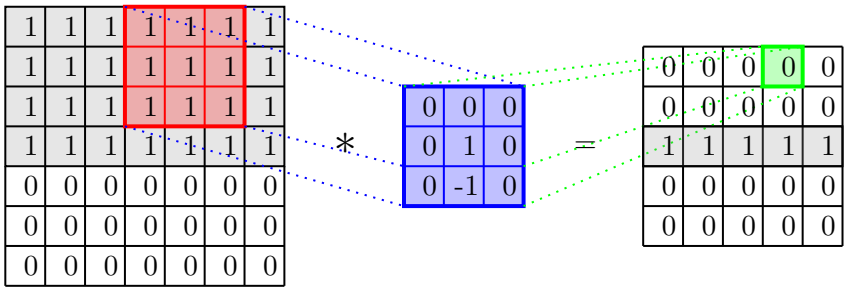


Figure 2.2: Convolution operation consisting of (from left to right) an input image, convolutional kernel, and convolved output image. The *valid* convolution results in the loss of the border pixels. Notice how the output represents the gradient image, obtained by setting the kernel to the finite difference between vertically adjacent pixels.

Convolutional neural networks (CNNs) are a family of neural network architectures having at least one *convolutional layer*. In image processing, a convolution between an image $\mathbf{I}$ and *kernel* $\mathbf{K}$ of size $d \times d$ and centered at a given pixel (x, y) is defined as,

$$(\mathbf{I} * \mathbf{K})(x, y) = \sum_{i=1}^{d} \sum_{j=1}^{d} \mathbf{I}(x + i - d/2, y + j - d/2) \times \mathbf{K}(i, j), \tag{2.9}$$

and is illustrated in Figure 2.2. Convolutions are useful for operations such as feature extraction and edge detection. A CNN is a neural network explicitly wired to perform convolutions[3]. In a convolutional layer, the elements of kernel $\mathbf{K}$ become learnable weights, replacing the large, fully-connected weight matrices from Section 2.1. In effect, the weights are *shared* across the input surface, greatly reducing the model dimensionality.

The earliest recognisable CNN is LeNet (LeCun et al. [1998]), bearing the name of the principal author, Yann LeCun. The destiny of LeNet is forever

[3] A convolutional layer can still be represented as a sparse fully-connected layer.

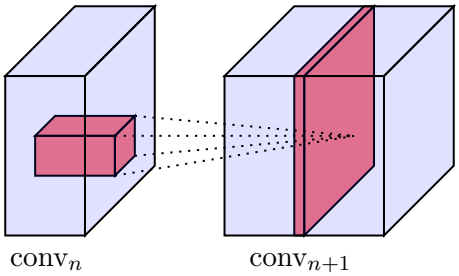


Figure 2.3: Each kernel of a convolutional layer (in red) performs a convolution as a tensor-product at each spatial location of an incoming tensor, to produce a single channel of the output tensor.

entwined with MNIST, the dataset it was developed to classify. MNIST is a 10-class image dataset of small (28×28px) handwritten digits ($0-9$) in greyscale, crowd-sourced from US school students. The LeNet architecture can be written as,

$$\begin{aligned}
\mathbf{H}_1 &= \sigma(\mathbf{X} * \mathbf{K}^{(1)}) && \text{(first convolutional layer)} \\
\mathbf{P}_1 &= \text{maxpool}(\mathbf{H}_1) && \text{(first pooling layer)} \\
\mathbf{H}_2 &= \sigma(\mathbf{P}_1 * \mathbf{K}^{(2)}) && \text{(second convolutional layer)} \\
\mathbf{P}_2 &= \text{maxpool}(\mathbf{H}_2) && \text{(second pooling layer)} \\
\mathbf{F}_1 &= \sigma(\mathbf{W}^{(1)}\mathbf{P}_2 + \mathbf{b}^{(1)}) && \text{(first fully-connected layer)} \\
\mathbf{F}_2 &= \sigma(\mathbf{W}^{(2)}\mathbf{F}_1 + \mathbf{b}^{(2)}) && \text{(second fully-connected layer)} \\
\mathbf{f}(\mathbf{X}) &= \mathcal{S}(\mathbf{W}^{(3)}\mathbf{F}_2 + \mathbf{b}^{(3)}) && \text{(output layer)}
\end{aligned} \tag{2.10}$$

where for MNIST, the $\mathbf{X} \in \mathbb{R}^{28\times28}$ denotes an input image (two-dimensional array), $*$ denotes the convolution operation, and σ and $\mathcal{S}$ denote the activation functions, as described previously.

The first convolutional layers kernels were originally 6 5×5 kernels. The six convolutions are performed on the same input, producing six *activation maps* that are concatenated across the channel dimension. As a result, the following convolutional layer originally had 16 $5 \times 5 \times 6$ kernels. Notice the kernels are now tensors, so as to account for the multi-channel inputs. This represents a generalisation of Equation 2.9 in which the each channel is convolved separately and the outputs are summed channel-wise. A depiction is given in Figure 2.3.

The maxpool are pooling layers (in effect max filter operations), replacing each 2×2 grid of pixels with their maxima. The maxpool operations are usually performed at a *stride* of 2, resulting in an output halved in each

spatial dimension. This ensures the most salient features are routed to the proceeding layers.

After the final pooling operation, the input tensor is implicitly flattened and traverses a series of final, fully-connected layers. The fully-connected layers were originally of size 120, 84, and 10 neurons, for the 10-way classification of MNIST digits. Note the numbers are fairly arbitrary and partially reflect the computational budget of the day. The architecture can be understood as having a feature extraction component in the convolutional and pooling layers, and a classification component in the fully-connected layers, with all layers trained end-to-end in unison using backpropagation and gradient descent. At each convolutional layer, the activations coincide with the motifs embedded in the kernels. These are promoted by the max pooling layers, which simultaneously coarsen the representation. Thus, salient motifs discovered at lower levels, such as edges and corners, can be recombined and aggregated into shapes, and, later, semantic objects. Zeiler and Fergus [2014] confirmed that such feature hierarchies were learned by deep CNNs.

2.2.1 AlexNet and the ConvNet revolution

The emphatic victory of AlexNet (Krizhevsky et al. [2012]) in the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC)[4] heralded the inexorable rise of CNNs. While emulating the same basic design as LeNet, the AlexNet architecture was deeper (more layers) and wider (more neurons/kernels), and processed larger images ($224 \times 224 \times 3$px), featuring in excess of 60 million parameters over 8 learnable layers. The breakthrough of AlexNet is often attributed to the conjunction of three technological developments:

1. The rise of graphical processing units (GPUs)
2. The sourcing of large image datasets such as ImageNet
3. The streamlining of neural network training e.g. more stable activation functions (ReLU), adaptive optimisers, regularisers such as dropout, and better weight initialisation or pre-training strategies.

With academic and industrial vision research in full swing, VGGNet[5] and GoogLeNet were followed as ILSVRC 2014 frontrunners, pushing the state-of-the-art ever further, while achieving lasting fame. The competition winner, GoogLeNet, pioneered a more complex neural network design by engineering the *inception module*, a concatenation of differently-sized convolutions. Laying several inception blocks end-to-end constitutes the 22-

[4] 1000 classes, 1 million images

[5] Visual Geometry Group at Oxford University.

Model	Year	Top-1	Top-5	No. Parameters
AlexNet	2012	40.7%	18.2%	60M
GoogLeNet	2014	-	10.1%	6.8M
VGG-19	2014	25.5%	8.0%	144M
ResNet-152	2015	22.2%	6.2%	60M
EfficientNet	2019	15.6%	2.9%	66M

Table 2.1: Single-model test errors on ImageNet for five groundbreaking CNNs. Note AlexNet was evaluated on the ILSVRC 2012 dataset, the others on ILSVRC 2014. Human performance has been estimated to be 5% Top-5 error (Karpathy [2014]).

learnable-layer GoogLeNet. VGGNet likewise strived for unprecedented network depth (up to 19 learnable layers) while advocating some simple design principles. For example, (padded) 3×3 convolutions are used everywhere, given that a stack of three 3×3 convolutions with C channels has a receptive field of size 7×7 on its inputs, despite having $27C^2$ weights, fewer than a single 7×7 kernel at $49C^2$.

A multitude of network architectures have since been proposed. Among the most influential is ResNet, which introduced the residual block: a skip connection bypassing a stack convolutions. The motivation is to prevent the gradients of deeper layers from overwhelming all others early during training. The residual block thus facilitated the training of extremely deep networks of up to 1202 layers[6]. More recently, EfficientNet proposed a "compound scaling" design strategy to increase network depth, width, and resolution in unison to maximise the performance benefit. As of February 2020, the state-of-the-art in ImageNet is an EfficientNet variant. Table 2.1 displays results from the history of the ILSVRC challenge illustrating the advancement of minimisation of test error over time. Note that AlexNet was evaluated on the ILSVRC 2012 dataset, whereas all others where evaluated on a similar dataset from the 2014 competition. Also, while GoogLeNet was the ILSVRC 2014 challenge winner, it prevailed only with a complex ensemble approach; we provide single-model results only.

2.3 Neural object detection

An object detection system f is a model or automated pipeline that both localises and classifies ontological objects in images. Localisation is usually expressed as a bounding box, which the detector emits along with the class prediction as the tuple,

[6] However, the standard depths are 50, 101, and 152 layers.

$$f : \mathbf{x} \rightarrow \{(x, y, w, h, c)\}, \tag{2.11}$$

for image $\mathbf{x}$, where x, y are the coordinates of the bounding box center, and w, h are its width and height. The value c denotes the class of the detected object. Figure 2.4 illustrates a set of object detections made on a custom image by a state-of-the-art system.

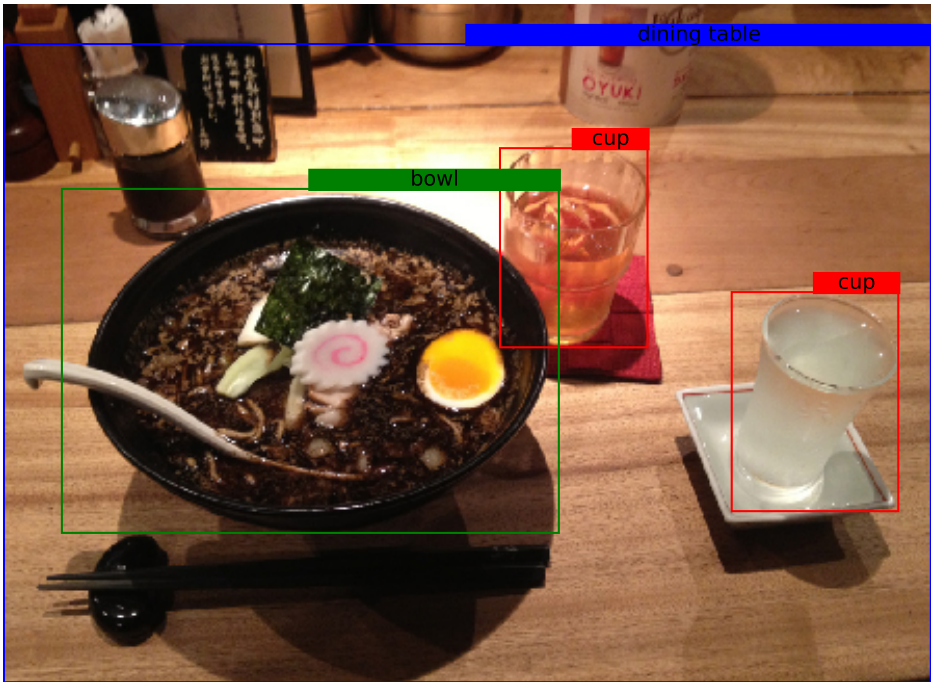


Figure 2.4: Object detections made by pre-trained R-CNN available in `PyTorch` (Paszke et al. [2017]). The image was taken in the Kyoto Gogyo ramen restaurant in Kyoto.

Whereas the ILSVRC datasets came to set the gold standard benchmarks in image classification, similarly rich datasets predominate for object detection. The Microsoft Common Objects in COntext (COCO) (Lin et al. [2014]) (80 classes + background) and the PASCAL[7] Visual Object Classes (VOC) challenge (20 classes + background) datasets (Everingham et al. [2010]) are the two leading datasets against which state-of-the-art object detection systems are tested. Following the explosion of interest in CNNs as image classifiers in 2012, rapid progress was made virtually in parallel for neural object detectors. An early example is *Overfeat* (Sermanet et al. [2013])), appearing in 2013. The basic architecture of Overfeat was an AlexNet,

[7]Pattern Analysis, Statistical Modelling and Computational Learning

refurbished as a fully-convolutional model[8].

No family of object detectors is more celebrated than the R-CNN series, however, the study of which is to be recommended to the pupil of deep learning.

2.3.1 Regions with CNN features

Regions with CNN features (R-CNN) (Girshick et al. [2014]) began as a disjoint pipeline consisting of four ordered stages:

(1) region proposal

(2) feature extraction,

(3a) object classification

(3b) bounding box correction

In the earliest iterations of R-CNN, the "selective search" algorithm played the role of stage (1)[9]. Feature extraction (2) was then performed by a pre-trained CNN, producing a "CNN code" feature vector for each region. Stage (3a) consisted of a set of linear models (typically SVMs) that perform one-vs-the-rest classification for each object class. These models were trained offline on the outputs of stage (2), "warped" to a standard size. Likewise, in stage (3b), a set of per-class (ridge) regression models are trained to correct the region proposal coordinates (x, y, w, h).

In Fast R-CNN (Girshick [2015]) stages (3a) and (3b) were absorbed into the CNN "backbone" of stage (2) as appendages to the neural feature extractor trained end-to-end. Now, entire images were processed in a single forward pass, with the region proposals cropped from the activation maps. The encoded regions were quantised by a generalisation of the max pooling layer, RoIPool. Faster R-CNN Ren et al. [2015] completed the work by integrating the region proposal algorithm (1) into the network. For this purpose, a separate region proposal network (RPN) was trained to predict both object presence and bounding box coordinates. A multi-stage training procedure was used to synchronise the weights between the two networks. Consequently, at prediction time, the feature extraction can be performed for both networks in a single forward pass. Then, the RPN heads produce the region proposals, which are passed to the heads of the R-CNN for detection. The overhead of region proposal is therefore negligible, and Faster

[8] Replacing fully-connected with convolutional layers frees the network from the constraint of a fixed input size.

[9] A greedy algorithm that progressively merges homogeneous regions of an input image, from which derive a large number of bounding box proposals.

R-CNN produces state-of-the-art object detection results more or less at real time.

2.4 Fighting overfitting in deep learning

In machine learning, overfitting is the effect of *fitting the noise instead of the signal*. In practice, all data contains noise obscuring the ground signal, and when a dataset is sufficiently small, a modestly powerful model may interpolate it perfectly, only to then be useless on test data. Much of machine learning is ultimately concerned with striking a balance between overfitting and underfitting. In supervised learning, this balance is evaluated with *generalisation error*, the ability of the model to generalise the data, which is estimated by evaluating a trained model over a test set of independent, unseen data. The *bias-variance decomposition* illustrates the tradeoff between over- and underfitting. Suppose we have $Y = f(x) + \epsilon$ generating training data with $f(x)$ the true signal for data point x, and noise, $\epsilon = \mathcal{N}(0, \sigma^2)$. Let our model estimate be denoted by $\hat{f}(x)$. Then, the MSE (here an arbitrary measure of goodness of fit) between an estimate and the true parameters, averaged over all possible data,

$$\begin{aligned}
\mathbb{E}[(Y - \hat{f}(x))^2] &= \mathbb{E}[((f(x) + \epsilon - \mathbb{E}[\hat{f}(x)]) + (\mathbb{E}[\hat{f}(x)] - \hat{f}(x)))^2] && (2.12)\\
&= \sigma^2 + \mathbb{E}[f(x) - \mathbb{E}[\hat{f}(x)]]^2 + \mathbb{E}[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2] && (2.13)\\
&= \text{noise} + \text{bias}^2 + \text{variance} && (2.14)
\end{aligned}$$

where in step (1.2) some terms are eliminated with zero mean. Note the positive terms means this represents a lower bound on error. The bias term represents underfitting as a simplified model fails to follow a more complex trend (for example a linear model undershooting a higher-order function). This bias will be observed in training error. The variance term represents overfitting as the model varies around its mean greatly to interpolate noisy data. This variance will be observed in test error.

Neural networks, being such powerful "universal approximators" (Hornik et al. [1989]), are especially prone to overfitting. Many strategies to attenuate it have therefore been proposed. A common approach used elsewhere in machine learning is *weight regularisation*, often called *weight decay*. This usually takes the form of a square error penalty term in the loss function. Parametric models that overfit usually require large parameter values (in the extreme, a degree n polynomial with up to $n - 1$ turning points can interpolate $n + 1$ data points), so regularisation curtails this tendency.

Deep learning succeeds because it embodies a different type of bias. Between layers of a vanilla fully-connected network, the number of weights is quadratic in the number of neurons per layer. Deep learning models reduce this dimensionality by spatial weight sharing, in the case of convolutional network, and weight sharing in time, in the case of recurrent neural networks. These may be referred to as *structural biases*.

Other tricks to avoid overfitting include dropout (Srivastava et al. [2014]), whereby neurons are randomly zeroed-out during training so as to curtail complex co-dependencies; batch normalisation (Ioffe and Szegedy [2015]), where a normalisation operation is performed on the inputs of each layer so as to standardise their variance (and avoid "internal covariate shift"); and data augmentation, which aims to increase the data supply. It is data augmentation that has particular relevance to this dissertation, and which we detail in the following.

2.4.1 Data augmentation

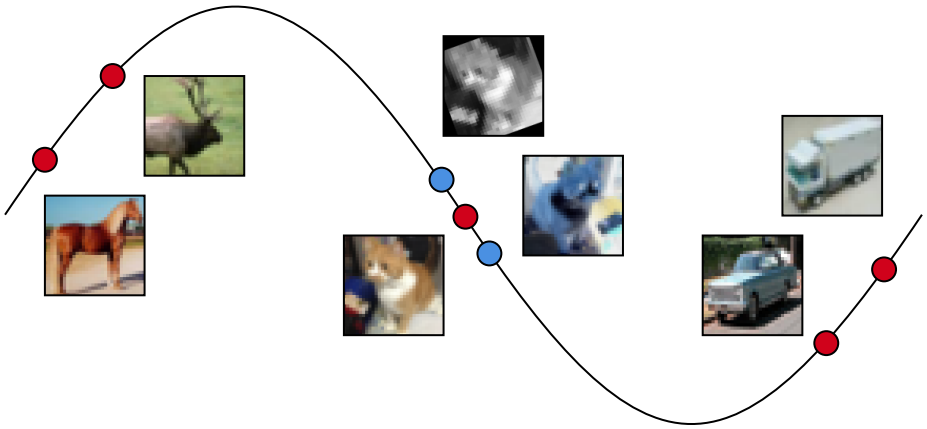


Figure 2.5: Data augmentation can make modest but useful interpolations of the space surrounding real images. CIFAR-10 images marked by red circle; augmented images marked by blue circle; black line represents natural image manifold. Upper augmentation based on conversion to grayscale (an element-wise weighted average of the RGB pixels) and rotation; the lower on colour inversion and horizontal flipping.

In the long run, as more data is added, a model with sufficient capacity improves its predictive power (generalisation error). In an ideal world, our

data is unlimited. However, annotated data for supervised training is expensive, requiring manual effort, often by domain experts. Data augmentation is a technique for obtaining new data fighting overfitting in machine learning models thereby improving generalisation (test time) error (Shorten and Khoshgoftaar [2019]). The idea is to extract additional data *gratis* from the available dataset, by means of bootstrapping or interpolation. A classic algorithm is SMOTE (Synthetic Minority Over-sampling TechniquE) (Chawla et al. [2002]), which interpolates lines between training samples and their nearest neighbours in feature space, to create synthetic points, and as an alternative to over-sampling by replacement.

Data augmentation is especially vital in deep learning, where powerful neural networks require large datasets to train. Due to the geometry of natural images (pixels are ordered and neighbouring pixels are highly correlated), image data is uniquely receptive to data augmentation, and humans are well-equipped for the engineering of it. Images support a multitude of *label-preserving transformations*, that is, transformations of lower-level image properties that nevertheless maintain the high-level semantic content. Examples are cropping, flipping, rotating, noise injection, convolution, and colour space transformations (if colour is available). These may be referred to as basic image manipulations (Shorten and Khoshgoftaar [2019]) and can be "stacked" (combined) endlessly, especially on the fly during training. Classic examples include image warping LeCun et al. [1998], elastic deformations Simard et al. [2003], and PCA-based colour space transformation Krizhevsky et al. [2012]. However, data augmentation must not be mistaken as equivalent to adding authentic and may reinforce existing biases in a dataset (Shorten and Khoshgoftaar [2019]). The augmented images are highly interdependent and will not in general amount to substantially increasing the coverage of space of all images (see Figure 2.5). Additionally, certain transformations will be domain-specific. For example, biomedical imagery can usually be rotated arbitrarily without losing meaning, unlike images of everyday objects or handwritten digits.

Deep learning itself can be used as a powerful, albeit expensive, mode of data augmentation. One good example is *adversarial training*. Goodfellow et al. [2014a] show how with a precise perturbation to a model input, one may "fool" a linear model or neural network alike. Consider perturbing model input $\mathbf{x}$,

$$\hat{\mathbf{x}} = \mathbf{x} + \alpha \cdot \eta, \tag{2.15}$$

where $\eta = \text{sign}(\mathbf{w})$ for $\mathbf{w}$ are the model weights of a softmax regression model and α a tuning parameter. Now, the model will predict $\sigma(\mathbf{w}^T\hat{\mathbf{x}}) = \sigma(\mathbf{w}^T\mathbf{x} + \alpha \cdot ||\mathbf{w}||_1)$. If $\mathbf{x}$ is sufficiently high-dimensional (as image data often is) the

perturbations will accumulate even for $\alpha << 1$ and can change the decision of the model. This produces a paradox especially apparent for image data: one may, for example, add a small, carefully chosen value to each pixel of an image, and the model may suddenly predict the wrong object class with high confidence, even though the perturbation is imperceptible to the human eye. This paradox defies intuitions about how neural networks represent images and "adversarial attacks" remain problematic for deep learning. Goodfellow et al. [2014a] nevertheless showed how adversarial attacks could be harnessed during training to strengthen neural networks against such attacks, in effect a sort of data augmentation.

Generative models, in particular generative adversarial models (GANs) (Goodfellow et al. [2014b]) (loosely related to the above) are a more recent and exciting prospect for data augmentation. Generative models are trained to learn (perhaps implicitly) the data-generating distribution, and thereupon may be used as a sampling system for synthetic data. Ideally, a generative model based on deep neural networks will make for a more powerful interpolation system than any of the above.

One may also consider transfer learning as a form of data augmentation commuted via pretrained model. The information content of a rich source dataset $\mathcal{D}_S$ may be distilled by a powerful neural network, and transferred to augment the available data for learning some task on a target dataset $\mathcal{D}_T$. Indeed, style transfer involving deep, pretrained networks Gatys et al. [2016] is a powerful mode of data augmentation, as we discover in Chapter 6.

2.5 Transfer learning

Transfer learning is a set of solutions for learning problems in which training and test data differ in their marginal distribution, or else that the learning tasks differ between model training and testing (Pan and Yang [2009]). As such, these problems violate the basic assumption for training supervised models. Transfer learning bridges *domains* of learning. A domain $\mathcal{D}$ consists of a feature space $\mathcal{X}$ and marginal probability distribution on that domain $P(X)$. A learning *task* $\mathcal{T}$ is performed on a domain, itself consisting of a label space $\mathcal{Y}$ and some (usually unobserved) objective function $f(\cdot)$. In transfer learning problems we denote source domain and tasks $\mathcal{D}_S$ and $\mathcal{T}_S$ and likewise for target domain $\mathcal{D}_T$ and $\mathcal{T}_T$. The source is the data on which the model is initially trained, and the target is that for which it must be adapted.

Neural networks, with their endless flexibility, are well-equipped for transfer learning. Shortly after the deep learning revolution in 2012, the potential for

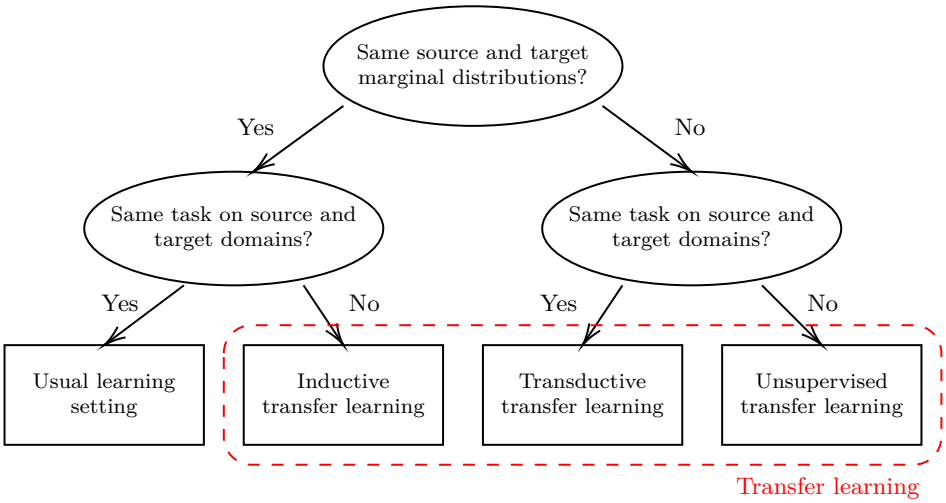


Figure 2.6: Positioning transfer learning with respect to the typical learning setting. Categories of transfer learning differ either with respect to the marginal distribution. Reproduced from https://en.wikipedia.org/wiki/Domain_adaptation

the learning transfer of powerful pretrained neural networks was discovered (see Zeiler and Fergus [2014] and Sharif Razavian et al. [2014]). Pretrained networks are often used in one of two ways:

1. feature extraction: a pretrained network is used to extract features for a new problem by passing images forward through its hidden layers and recording a given layer ("CNN code") as a vector of (obscure) image features.

2. fine-tuning: a pretrained network recommences training on a new problem from the previous point of convergence, usually at a significantly lower learning rate, and often only on the deepest layers.

The pretraining itself is usually done on a large standard image corpus like ImageNet. The success of these approaches illustrates the universality of the filters learned by deep networks, as general feature extractors for images. Neither approach fits easily into the schema of Figure 2.6. However, if the target images are similar in resolution and content, for example, the low-level neural activity of earlier layers may be identical to the source dataset, placing it near inductive transfer learning.

An interesting way to perform transductive transfer with deep learning is with *domain-adversarial neural networks*, which we describe presently.

2.5.1 Domain-adversarial neural networks

The inspiration for domain-adversarial neural networks (DANNs) lies in work published in Ben-David et al. [2010], which defines the $\mathcal{H}$-divergence,

$$d_{\mathcal{H}}(D_S^X, D_T^X) = 2 \sup_{h \in \mathcal{H}} \left| P_{\mathbf{x} \sim D_S^X}(h(\mathbf{x}) = 1) - P_{\mathbf{x} \sim D_T^X}(h(\mathbf{x}) = 1) \right|, \quad (2.16)$$

that is, given a source domain (distribution), D_S^X (marginalised by the input variable), and a target domain D_T^X, and given a hypothesis class[10] $\mathcal{H}$, the divergence between the source and target domains with respect to $\mathcal{H}$ is the classifier (here binary–for simplicity) that, proportionally, most classifies the domains into separate classes.

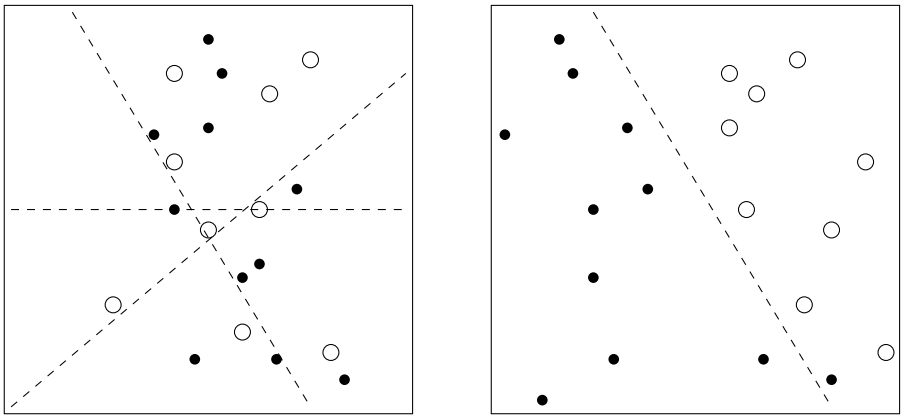
To illustrate, suppose we choose our hypothesis class to be a linear SVM. This class is capable (through choice of the SVM weights) of creating any possible linear hyperplane. Suppose the two domains were linearly separate in the feature space (see Figure 2.7). It would then be possible to train an SVM to perfectly separate the two, putting one domain fully into the positive class, and the other into the negative class. This would maximise the inner expression of the divergence formula, thus describing a high divergence. Should we have, on the other hand, two highly overlapping domains, any hyperplane cut through the middle of the point cloud would give us a divergence close to 0. Should we cut so as to isolate a few outlier points of a particular class, these would represent only a small proportion of the data, and the probability would also be close to 0. Intuitively, the $\mathcal{H}$-divergence captures the effect nicely.

Conveniently, it is possible to compute a consistent estimate of the $\mathcal{H}$-divergence using finite data with the *empirical* $\mathcal{H}$-divergence,

$$\hat{d}_{\mathcal{H}}(S, T) = 2\left(1 - \min_{h \in \mathcal{H}} \left[\frac{1}{n} \sum_{i=1}^{n} \mathbf{1}[h(\mathbf{x}_i) = 0] + \frac{1}{n'} \sum_{i=n+1}^{N} \mathbf{1}[h(\mathbf{x}_i) = 1]\right]\right) \quad (2.17)$$

for samples, $S \sim D_S^X$ of size n and $T \sim D_T^X$ of size n'. For convenience, these are indexed from $1 \to n$ and $(n+1) \to N$, where $N = n + n'$. Though this may be hard to compute precisely in general, we can approximate this simply by training a classifier of the class $\mathcal{H}$ on the constructed dataset, $U = \{(\mathbf{x}, 0) : \mathbf{x} \in S\} \cup \{(\mathbf{x}, 1) : \mathbf{x} \in T\}$, that is, a classifier to differentiate between the domains. The estimated generalisation error, ϵ of this classifier

[10] Category of classifiers, e.g. linear SVMs.



(a) Low divergence (b) High divergence

Figure 2.7: Overlapping domains (a) and divergent domains separated by a linear decision boundary (b). Modified from http://blog.pengyifan.com/tikz-example-kernel-trick/.

could then be used to approximate the empirical domain divergence as $2(1-2\epsilon)$.

Ajakan et al. [2014] first formulate a DANN as a learning model of the form,

$$E(\theta_f, \theta_d, \theta_y) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_y^i(G_y(G_f(\mathbf{x};\theta_f);\theta_y), y_i) + \lambda R(\theta_f, \theta_d) \tag{2.18}$$

where the G_f component acts as a feature extractor, and G_y is a classifier. These may have any neural architecture, shallow, deep, convolutional, etc. Recalling that the target data is unlabelled, they propose to define the regulariser of this model as,

$$R(\theta_f, \theta_d) = -\frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_d^i(\theta_d, d_i) - \frac{1}{n'}\sum_{i=n+1}^{N} \mathcal{L}_d^i(\theta_d, d_i) \tag{2.19}$$

where $\mathcal{L}_d^i(\theta_d, d^i) = \mathcal{L}(G_d(G_f(\theta_f);\theta_d), d_i), d_i \in \{0, 1\}$. Thus formulated, we have a loss maximising the negative of the minimisation term in the divergence formula. Therefore, adding this regulariser to the objective function, and maximising the objective with respect to its parameters, θ_d, give a worse overall loss. At the same time, however, the feature parameters, θ_f, which are trained to minimise the objective, are chosen to minimise this maximisation of the regulariser by the divergence parameters, θ_d, thus maximising

the minimising component of the divergence, thereby minimising the divergence. The feature parameters therefore play a dual role in both minimising classification loss, and minimising divergence (equivalently, maximising discrimination error), *adversarially* to the domain classifier. This promotes domain invariant features. Formally, we have,

$$\begin{aligned}(\theta_f, \theta_y) &= \arg\min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \hat{\theta}_d) \\ \theta_d &= \arg\max_{\theta_d} E(\hat{\theta}_f, \hat{\theta}_y, \theta_d)\end{aligned} \tag{2.20}$$

which can be found as a saddle point of the update steps,

$$\begin{aligned}\theta_f &\leftarrow \theta_f - \mu\left(\frac{\partial \mathcal{L}_y^i}{\partial \theta_f} - \lambda \frac{\partial \mathcal{L}_d^i}{\partial \theta_f}\right) \\ \theta_y &\leftarrow \theta_f - \mu \frac{\partial \mathcal{L}_y^i}{\partial \theta_y} \\ \theta_d &\leftarrow \theta_f - \mu \frac{\partial \mathcal{L}_d^i}{\partial \theta_d}\end{aligned} \tag{2.21}$$

Note that θ_f is updated to minimise the classification loss, but maximise the domain classification loss. This is called the adversarial step. The main contribution of Ganin and Lempitsky [2014] is to apply this to deeper architectures (in particular, convolutional), and to introduce a "gradient reversal layer" to enable a smooth implementation without altering the internal implementations of gradient descent (this still assumes that the local gradients can be specified arbitrarily). This is achieved with the pseudo-function,

$$R_\lambda(\mathbf{x}) = \mathbf{x} \tag{2.22}$$

whose gradient is hard-coded as,

$$\frac{\partial R_\lambda}{\partial \mathbf{x}} = -\lambda \mathbf{I} \tag{2.23}$$

This pseudo-layer is inserted between the feature extractor and the domain classifier in the model formulation. It should be emphasised that this is not a mathematical trick, but rather an implementation hack. The gradient reversal layer is easy to implement in all the major deep learning frameworks.

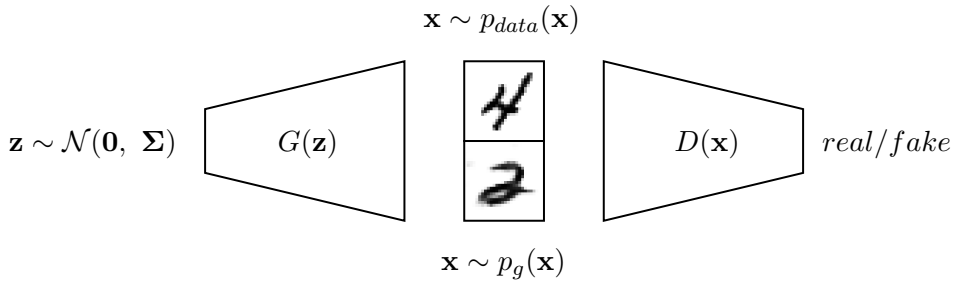


Figure 2.8: A GAN trains a generator network G to fool a discriminator network D in emitting counterfeit images $\mathbf{x}$ by transforming noise input $\mathbf{z}$. In order to do so, G must (implicitly) learn the data-generating distribution, p_{data}.

2.6 Generative adversarial networks

Generative adversarial networks (GANs) (Goodfellow et al. [2014b]) is a learning framework that trains a generator model to be capable of sampling new data from a data distribution (in particular, images), learnt (implicitly) from a training set. GANs compete with variational autoencoders (Kingma and Welling [2013]) and, more recently, autoregressive models (Oord et al. [2016]) for image generation. We select GANs as the primary tool for our work in image synthesis in this dissertation, given their rich literature and versatility (as we shall soon see).

The GAN framework (Figure 2.8) is *adversarial* in that it pits a pair of neural networks in a two-player minimax *game*: a *generator*,

$$G : Z \to X, \tag{2.24}$$

where in the simplest case $\mathbf{z} \sim Z$ is a vector of (uniform or Gaussian) noise and $\mathbf{x} \sim X$ is a synthetic data point (in particular, an image); and a *discriminator*,

$$D : X \to \{0, 1\}, \tag{2.25}$$

that is, a mapping from data point to binary value. D aims to minimise its rate of failure in discerning true data from "fake" data sampled from the the generator, which aims to maximise the error made by the discriminator,

$$\max_G \min_D \mathcal{L}_{GAN}(D, G) = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{2.26}$$

The stationary point of Equation 2.26 is a *Nash equilibrium* (a saddle point) between the two objectives. Goodfellow et al. [2014b] guarantee that the distribution implicitly defined[11] by G, $p_g = p_{\text{data}}$ for G^* at (global) optimality, and

$$D^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})}, \tag{2.27}$$

where p_{data} is the data-generating distribution. In practice, GANs are trained by backpropagation with alternating gradient descent between D and G. The weights of D are frozen while propagating the errors back through D to G.

GAN architecture design must strike a balance between a generator expressive enough to approximate the data distribution, and a discriminator powerful enough to hold it to the task. GANs are chronically afflicted with a phenomenon known as *mode collapse*. Mode collapse is the condition in which the generator "collapses" to generating few or even a unique output, regardless of the noise input $\mathbf{z}$. Metz et al. [2016] differentiate two varieties of mode collapse: *discrete mode collapse*, where the generator collapses to a subset of data modes (easily detected); and the more insidious *manifold collapse*, where the generator collapses to a subspace of the data distribution. They reason that mode collapse originates from the fact that in each iteration of training, the generator is impelled towards a delta function at the mode considered most "real" by the discriminator. The discriminator responds by lowering the probability of this mode, leading to oscillations in generation. The "unrolled" GAN training approach that they advocate mitigates this tendency by informing the generator in advance of the discriminator's response to its prospective weight updates.

Another promising attempt to attenuate the mode collapse problem is to replace the Jensen-Shannon divergence loss function with the Earth Mover (EM) or Wasserstein-1 distance. Such is the strategy of Wasserstein GANs (WGANs). Intuitively, the EM distance measures a distance between two distributions. We can write the EM distance as the quantity $\mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z\sim p(z)}[f_w(g_\theta(z))]$ maximised by choice of f_w (discriminator). The goal is then to minimise this distance through optimisation of the generator. The

[11] GANs are implicit generative models, in contrast to explicit models such as VAEs, which model the probabilities directly. GANs serve as an sample-emitting oracle emulating the true distribution.

training algorithm is very similar to GANs. WGANs are more stable, as the substituted loss function allows the discriminator to be trained to optimality. This greatly mitigates the mode collapse problem. Nevertheless, more subtle forms of partial mode collapse remain a recurrent problem for GANs. As a bonus, the discriminator loss (estimated EM distance) becomes meaningful during training, interpretable as image sample quality, rendering the implementation and fine-tuning of WGANs far easier. Despite the merits of WGANs, we do not have recourse to them in this dissertation.

2.6.1 Deep convolutional GANs

Radford et al. [2015] first succeeded in training more powerful, high-resolution GANs, by identifying a set of design principles: replacing pooling layers with strided convolutions; removing fully-connected layers (apart from the first layer of the generator); using batch normalisation after all but the last layer; using a tanh activation at the end of the generator (and ReLU everywhere else); and using LeakyReLU activations everywhere in the discriminator. Deep convolutional GANs (DC-GANs) represent a vast improvement over the original fully-connected GANs.

2.6.2 Conditional GANs

Mirza and Osindero [2014] present a simple extension to GANs allowing for control over the data generating process. One simply includes an additional input $\mathbf{y} \sim Y$ (for example, class information), to obtain conditional GANs (cGANs). That is, now the generator is conditional,

$$G : Z, Y \to X, \tag{2.28}$$

as is the discriminator,

$$G : X, Y \to \{0, 1\}, \tag{2.29}$$

and the cGAN is trained against the objective,

$$\mathcal{L}_{cGAN}(D, G) = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[\log D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\mathbf{y})))] \tag{2.30}$$

Mirza and Osindero [2014] does not offer so much as an intuition as to why conditional GANs works (seemingly the article was written in a hurry), but it can be instructive to consider. As D learns to associate conditional

information such as object class with the ground truth images, it will declare "fake" any image emitted by G not conforming to the condition, no matter the quality of the image. If G is to succeed, it will be forced to generate images in adherence to the condition. The outcome of successful training is therefore a generator that can, for example, synthesise an authentic-looking image of a desired class on demand.

The implementation of cGANs is a straightforward modification to GANs: the conditional information is concatenated to the typical usual inputs of both D and G and fed to further hidden layers. A DCGAN discriminator may, however, concatenate the conditional information after its convolutional layers.

2.6.3 Assorted GANs

The success of GANs have led to many interesting results in recent years. Image-to-image translation has been achieved with `pix2pix` (Isola et al. [2017]), a GAN conditioned on a full image serving as a blueprint for the generated image. Such models are capable of an endless variety of applications, including segmentation and image colourisation. CycleGANs (Zhu et al. [2017]) achieve *unpaired* image-to-image translation, learning to map between image domains in an unsupervised way. Application include style transfer. RecycleGANs (Bansal et al. [2018]) achieve the same for video-to-video retargeting.

Elsewhere, progressive growing of GANs (Karras et al. [2017]) produce high resolution outputs (up to 1024×1024 pixels). Here, training is done in stages, beginning at low resolutions (4×4), and doubling the resolution at intervals. This is done simply by adding new layers to the generator and discriminator, which remain symmetric throughout training. Thus, the GAN learns global structures first at low resolution, progressively refining its output.

Part I

Triple-negative breast cancer

Chapter 3

High content analysis in drug and wild type screens

Summary: *In this chapter the high content analysis pipeline is put into practice on two image datasets: a drug screen and a wild type screen of triple negative breast cancer cells. The implementation of a conventional pipeline for computational phenotyping is described in phases of cell segmentation and feature extraction. A collection of use cases are then described on each of the screens, covering the full range of univariate, multivariate, and machine learning-based analyses identified in the previous chapter, with the aim of understanding the morphological properties of the cell lines, and the effects of drug perturbation thereupon. We will here lay the groundwork for the*

Résumé: *Cette chapitre...*

3.1 Overview

Triple-negative breast cancer (TNBC) is a variety of breast cancer whose cells do not express proteins *estrogen receptor* (ER), nor *progesterone receptor* (PR), nor do they amplify *Human epidermal growth factor receptor 2* (HER2/neu). As these genetic markers are common targets for cancer therapies, TNBC is more difficult to treat than other breast cancers, being, in particular, unresponsive to hormone therapies (Hudis and Gianni [2011]). As a result, there are limited treatment options for TNBC, and chemotherapy remains the main treatment. TNBC is responsible for 10%-15% of breast cancers (Chavez et al. [2010]) and has a poorer 5-year survival rate than other forms of breast cancer (Gonçalves Jr et al. [2018]). TNBC exhibits molecular heterogeneity at the level of multiple subtypes Hatem et al. [2016].

Two datasets are considered in the following analysis on TNBC cell lines. The first is a pilot drug screen study by fluorescence microscopy on two TNBC cell lines (MDA231 and MDA468) (Filmus et al. [1985]). The second dataset is a wild type screen of 12 cell lines (Chavez et al. [2010]) (11 TNBC cell lines and 1 negative control), that is without perturbation. Part I of this dissertation focusses primarily on the drug screen data, but both datasets feature in the following analyses.

3.1.1 Drug screen dataset

The drug screen pilot data set (Table 3.1) is an assay of two microplates each housing a grid of 384 (18×24) wells, labeled A01-P24, (alphabetic character denoting the row of the well on the plate, numeric denoting the column). One plate tests TNBC cell line MDA231, the other MDA468. Wells were seeded to confluence with a controlled 1000 and 1250 cells for MDA231 and MDA468 cell lines respectively. The drugs are allocated according to a *plate map* (given in full in Appendix B), which is applied for both plates. 36 wells are treated with the neutral agent dimethyl sulfoxide DMSO as a negative control, 2 with positive controls (Olaparib, Cisplatine), 166 with test compounds (concentration $10\mu M$), and 184 untreated (denoted empty). Following hibernation, the cells were fixed, washed, and stained with four fluorescent markers. A final washing procedure is performed, where extraneous dye content is aspirated from the wells, finally preserving only the chemically-bound dye compounds excited during the microscopy, and further evacuating all cells unfastened to the well, as a consequence of perturbation or otherwise. This step has consequences for the analysis, as it may remove cells dead prior to fixation, ostensibly leaving an absence of cells as a proxy for high levels of apoptosis, indicating a cytotoxic compound. The array of perturbations show a range of effects on cell mortality, from no apparent visual effect to a near or complete elimination of cells.

The drugs comprise of a set of panels of kinase, protease and phosphatase inhibitors and can be categorised into 70 mechanism of action (MOA) classes of varying sizes, according to their targets. For our experiments, we take the 8 MOA classes having at least five member drugs. These are CDK inhibitors, cysteine protease inhibitors, EGF receptor kinase inhibitors, MMP inhibitors, DMSO (negative control), PKC inhibitors, protein tyrosine phosphatase inhibitors, and tyrosine kinase inhibitors. In comparison with other datasets, Adams et al. [2006] used 51 drugs in 13 MOA categories, Slack et al. [2008] used 35 drugs in six MOA categories, and the widely studied Broad Institute Benchmark Collection 21 (BBBC21v2) Ljosa et al. [2012] – used, for example, in Kandaswamy et al. [2016] and Godinez et al. [2017] –

Plate/cell line	Well, field	Channel	Acquisition Mode
MDA231, MDA468	Plate row (A-P), column (1-24), well field (1-4) e.g. A01_01	DAPI, Cy3, Cy5, FITC	2D, 2D-Decon, Adv2D-Decon, 2.5D

Table 3.1: The drug screen pilot data, consisting of two cell lines on 384-well plates, with four fields per well. Four fluorescence channels are captured, under four acquisition modes. The full data set of 49152 images is the outer product of each of the table fields.

Stain	Marker
DAPI	A-T regions of DNA
Cyanine 3 (Cy3)	DNA double-strand bbreaks
Cyanine 5 (Cy5)	β-tubulin of microtubules
FITC	Phospholipids in cell membrane

Table 3.2: The stains used in the fluorescence microscopy of the screen and their corresponding biological markers.

consists of 39 drugs in 13 categories. The key difference is that our own MOA classes were not selected *a posteriori* to reflect visually different phenotypes, mounting a greater bioinformatic challenge than the standard benchmark datasets, where even a simple model can be extremely effective. For example, Singh et al. [2014] achieved 90% accuracy with element-wise averaging of hand-crafted features after a simple luminosity correction.

Fluorescence microscopy (20× magnification widefield with deconvolution image restoration) was performed at four non-overlapping fields of view per well produced two experimental datasets of 1536 multiplexed images apiece. The stains used in the screen are listed in Table 3.5. *DAPI* (4',6-diamidino-2-phenylindole) is a bright blue stain that permeates a cell's nuclear membrane and binds to regions of DNA rich in A-T base pairs. DAPI is effective in highlighting the cell nucleus, where the DNA is housed. Cyanine 3 (Cy3) and cyanine 5 (Cy5) are commonly used dyes belonging to a common family. Here, Cy3 is conjugated to a γ-H2AX antibody. γ-H2AX is a protein that naturally binds to double-strand breaks as a marker for the repair of DNA damage. Furthermore Cy5 is conjugated to a β-tubulin antibody, thus highlighting this sub-component of the tubulin polymer that constitutes *microtubules*, which are structure-providing components of cells. Finally, FITC (fluorescein isothiocyanate) highlights phospholipids, molecules in the lipid cellular membrane. Figure 3.1 shows a composite of DAPI, Cy5, and Cy3 fluorescence channels for an indicative field from the drug screen.

Cell line	Well, field	Channel
MDAMB231, MCF10A (negative), Hs578T, MDAMB157, HCC1143, HCC38, MDAMB468, HCC1937, MDAMB436, BT20, BT549, HCC70	Plate row (A-H), column (1-12), well field (1-5). Each cell line occupies one column.	DAPI, Cy3, Cy5, FITC rows A-D, *rhodamine* rows E-H

Table 3.3: The wild type screen data, consisting of 12 cell lines evenly distributed over a 96-well plate (ordered by column). Four fluorescence channels are captured as in the drug screen, albeit with FITC replaced by rhodamine in the final four rows.

The microscopy was performed in various *acquisition modes*, whereby imagery is captured in different ways. These are: 2D (raw camera output), 2D + deconvolution (2D with blur reduction), Advanced 2D + deconvolution (combination of image 3-stack), and 2.5D acquisition and deconvolution (aggregation over 3D image stack). Each mode captured the same cells at virtually the same time. The 2.5D mode was chosen for all analysis contained in Part I as it exhibited the sharpest detail from manual visual inspection.

3.1.2 Wild type screen dataset

In the wild type screen, 12 cell lines were imaged, including 11 TNBC cell lines (see Table 3.3), distributed evenly over a 96-well plate (8 rows, 12 columns). In contrast to the drug screen, the objective was to monitor cell wild type morphologies, hence no chemical compounds were included in the wells for the screen. The cell line MCF10A is immortalised but non-tumorigenic, and serves as a negative control. The dataset is significantly smaller than that of the drug screen, albeit with five fields of view imaged per well. The same fluorescent markers as used in the drug screen are used (Table 3.2), except for the fourth channel, which uses the FITC marker for half the plate, and rhodanine for the other half to monitor expression of tumour-suppressor protein p53.

Table 3.4 collates properties of the 12 cell lines, from which one can begin to appreciate the heterogeneity at play. The MDA family derive from pleural effusions, that is, from a fluid buildup in the patient, while all others derive from primary tumours. The molecular classification Basal A characterises cells that are more luminal while Basal B are more basalDai et al. [2017]. These determine imply different functions of the cells. Not listed are relevant information on common tumour suppressor genes: all cell lines (aside

Cell line	Site[1]	Pathology[2]	Molecular classification
BT-20	PT	IDC	Basal A
BT-549	PT	IDC	Basal B
HS-578T	PT	IDC	Basal B
HCC-38	PT	IDC	Basal A
HCC-70	PT	IDC	Basal A
HCC-1143	PT	IDC	Basal A
HCC-1937	PT	IDC	Basal A
MDA-MB-157	PE	IMC	Basal B
MDA-MB-231	PE	AC	Basal B
MDA-MB-436	PE	IDC	Basal B
MDA-MB-468	PE	AC	Basal A
MCF-10A	NB	Fibrocystic	Basal B

Table 3.4: TNBC cell lines and negative control MCF-10A properties referenced from Chavez et al. [2010]. Drug screen cell lines indicated in bold. [1]Site: PT, primary tumour; PE, pleural effusion; NB, normal breast. [2]Pathology: IDC, infiltrating ductal carcinoma; IMC, infiltrating medullary carcinoma; AC, adenocarcinoma.

from the negative) exhibit a p53 mutation; contrarily, most exhibit wild type BRCA1[1], aside from cell lines HCC-1937 and MDA-MB-436. For the curious reader, the cell lines are prefixed generally according to the research group behind their discovery: BT (Breast Tumour) (Lasfargues and Ozzello [1958]); HS (Hackett + Smith) (Hackett et al. [1977]); HCC (Hamon Center) (Gazdar et al. [1998]); MDA (MD Anderson Cancer Center) (Brinkley et al. [1980]); MCF (Michigan Cancer Foundation) (Soule et al. [1990]).

3.2 Cell measurement pipeline

As mentioned in the introductory Section 1.1.3, the basis of high content analysis are the *features* measured on individual cells. The conventional path to measuring cellular features entails an initial segmentation of individual cells. This section describes the segmentation strategy as well as the features extracted for both the drug and wild type screen datasets described above. These procedures underpin standard preliminary analyses of the data (Section 3.3), as well as the developments described in later chapters.

[1]Breast cancer type 1 susceptibility protein

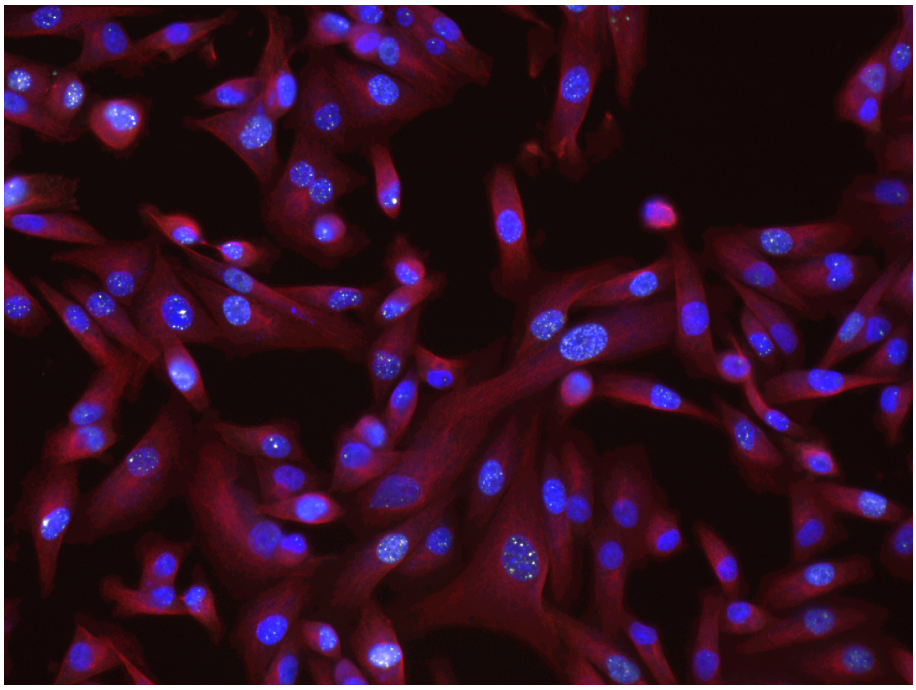


Figure 3.1: Fluorescence image of MDA231 cells. DAPI highlighting the nuclei in blue, cyanine 5 highlighting the microtubules in red, and cyanine 3 highlighting the double-strand breaks as white spots on the cell nuclei.

3.2.1 Nuclei segmentation

The cell nucleus is the logical starting point for cell segmentation (see Figure 3.2). Roughly speaking, most nuclei are visible, uniformly-sized, and ellipsoidal, and, crucially, seldom overlap, as the cell surrounding cell membrane acts as a buffer to other cells, and as the cell density has been chosen such that cells do not grow on top of each other. As a result, they are typically the easiest thing to accurately segment in the fluorescent stack, and may then be used as the starting point for segmenting other cytological components. The segmentation typically follows three stages: pre-filtering, detection, and refinement.

Pre-filtering

A common approach to pre-filtering is median filtering (Huang et al. [1979]), which can be very effective in eliminating noise. The approach works as a sliding window over the image. Thus, for image $f : (X \times Y) \rightarrow V$ and window $W(x, y)$ centered on (x, y) of size $m \times n$, one creates a pre-filtered image p with,

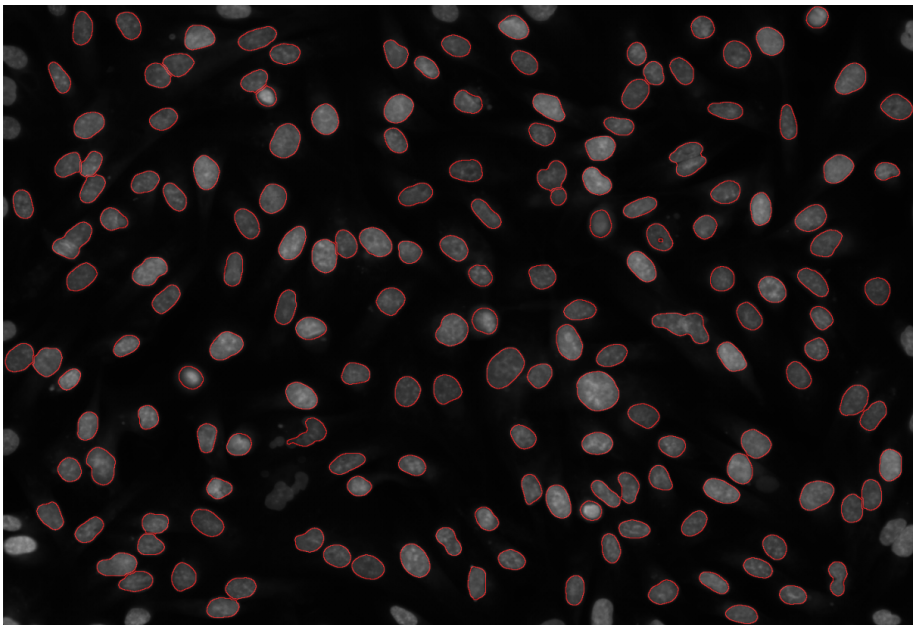


Figure 3.2: Nuclei segmentation on the DAPI channel, with segmentation contours indicated by red bands.

$$p(x, y) = \text{median}\{f(x', y') | (x', y') \in W(x, y)\} \tag{3.1}$$

Note the median is an appropriate operator to use as it is robust to outliers, unlike the mean. Note, however, that the elimination of noise may come at the cost of linking separated objects as, depending on the size of the filter, the pixels between nearby objects can be interpreted as outliers.

Background subtraction

In order to detect the contours of the cell nuclei, one takes the pre-filtered image p and performs a background subtraction. This again works with a sliding window, W, of size $m \times n$, which calculates the local mean intensity for a pixel $b(x, y)$, giving,

$$b(x, y) = \frac{1}{mn} \sum_{i=-\lfloor m/2 \rfloor}^{+\lfloor m/2 \rfloor} \sum_{j=-\lfloor m/2 \rfloor}^{+\lfloor m/2 \rfloor} p(x + i, y + j) \tag{3.2}$$

with which one can perform the subtraction yielding,

$$r(x, y) = \max(0, p(x, y) - b(x, y)) \tag{3.3}$$

Note that values in r will now be close to zero, with non-zeros corresponding to regions of high intensity gradient. One can therefore identify the contours of the cells, $c(x, y)$ by applying a threshold t giving,

$$c(x, y) = \mathbb{1}\{r(x, y) \geq t\} \tag{3.4}$$

The nuclei contours may be filled to produce a final segmentation mask.

Refinement

In the ideal case of well-separated cells, the aforementioned procedures might be sufficient for perfect nuclei segmentation. However, in practice, some additional steps are required to segment overlapping cells. One of the most classical and widely used refinement approaches is the *watershed* algorithm (Beucher and Lantuéjoul [1979]) applied to the inverse distance map of the segmentation result[2]. Informally, the watershed algorithm simulates a flooding of the topography of the intensity levels in an image. The meeting points of the growing regions build the *watershed line*, which is amounts to a separating line between objects. This method partitions the initially identified object into as many regions as there are local minima of the inverse distance map. In order to avoid over-segmentation due to small irregularities in the object boundary potentially leading to local minima, one selects local minima according to their morphological dynamic (Grimaud [1992]).

3.2.2 Cell membrane segmentation

With the cell nuclei segmented on the DAPI channel, one proceeds to segment the cell cytoskeleton, approximated by the microtubules on the Cy5 channel. Here we follow the approach of Jones et al. [2005], which uses the cell nuclei as seeds for the more difficult cell membrane segmentation. The approach consists of defining a specialised distance metric that accounts for changes in intensity (high gradients), with the matrix,

$$\mathbf{G} = \begin{bmatrix} (\frac{\partial g}{\partial x})^2 & \frac{\partial g}{\partial y} \cdot \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial x} \cdot \frac{\partial g}{\partial y} & (\frac{\partial g}{\partial y})^2 \end{bmatrix} \tag{3.5}$$

where $\mathbf{g}(\mathcal{I})$ is Gaussian blur, whose gradients are simply the finite central difference of the adjacent pixels (these too can be defined by convolutions), and infinitesimal distances are calculated as,

[2]The distance map is the image formed by the distances of the shortest path to the background for every foreground pixel in a segmentation mask.

$$||d\mathbf{x}||^2_{\mathbf{G}} = d\mathbf{x}^T \mathbf{G} d\mathbf{x} \tag{3.6}$$

In reality, we have,

$$G = \frac{\nabla g(f) \nabla^T g(f) + \lambda I}{1 + \lambda}, \tag{3.7}$$

hence the distance between two points is given by the compromise between gradient based distance and Euclidean distance,

$$dx^T G dx = \frac{\|dx^T \nabla g(f)\|^2 + \lambda \|dx\|^2}{1 + \lambda} \tag{3.8}$$

The distance metric is defined for neighbouring pixels. Distances between non-adjacent pixels are computed as the shortest path (for example with Dijkstra's algorithm). Thus, the metric behaves like a spatial distance metric on flat, uniform regions, yet takes intensity into account where necessary. The segmentation is performed by assigning pixels to the nearest seed, that is, cell nucleus, under this Euclidean-regularised metric. One may see from Equation 3.6 that the metric converges to Euclidean distance as $\lambda \to \infty$. Hence, the segmentation becomes a Voronoi tesselation in the limit.

The tunable parameters of the algorithm are: the size (receptive field) of $g\mathcal{I}$; an initial global thresholding step to eliminate the background, for example with an Otsu threshold (Otsu [1979]); and the regularisation λ.

3.2.3 Feature extraction

In order to quantify cellular phenotypes, features are extracted from the segmented cells. These features can be interpretable or general shape, intensity and texture descriptors. In this base line approach, I used the features provided by standard open-source software tools, here CellCognition[3]. Cell Cognition can export many hundreds of features for each fluorescent channel at the user's discretion. These features generally fall into the following categories: basic intensity features, basic shape features (e.g. RoI size, perimeter, circularity), convex hull features, distance map features, granulometry features, Haralick features, moments, and statistical geometric features (see Held et al. [2010], Walter et al. [2010a] for a detailed description).

In these experiments, 238 features were extracted for each of the DAPI and Cy5 channels, and 38 for the Cy3 channel, giving 516 features in total

[3] http://cellcognition-project.org

for each cell. These features are engineered to be plausibly discriminative, yet rarely have an immediate biological sense. Nevertheless, a minority of features can inform about some basic biological phenomena: the size of each segmented object (RoI size) on the DAPI channel directly quantifies nuclear size, from which one may infer about cell growth and aberrant phenomena such as chromosome segregation problems and consequent micronucleation; granulometry and Haralick features measure texture structures, which can be a proxy for mitochondria; average DAPI intensity can indicate an interphase cell prior to or after DNA replication; and spot features directly measure double-strand breaks (DSBs) on the Cy3 channel (see Section 3.3.3).

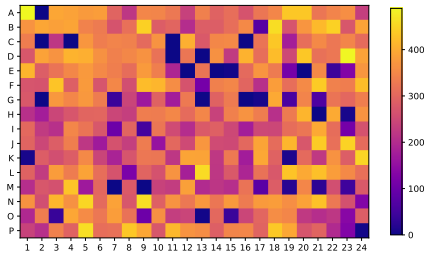
3.3 Use cases in high content analysis

The following section details a series of use cases of explorative analysis in the drug and wild type screens. Section 3.3.1 first investigates the possible presence of spatial biases in the drug screen microplates. Sections 3.3.2 and 3.3.3 in turn describe a univariate and multivariate analysis of the drug screen. Finally, machine learning techniques are used in Section 3.3.4 to make morphological comparisons of TNBC cell lines in their wild type.

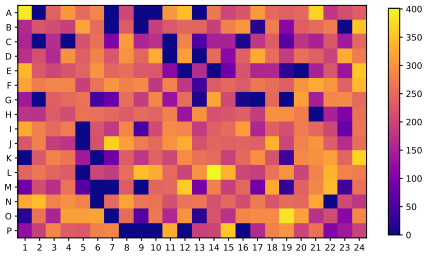
3.3.1 Controlling for spatial effects in the drug screen dataset

One concern in HCS is the presence of systematic bias in the experimental setup. Controlling for this bias is the objective of a series of quality control techniques. In the pilot drug screen dataset, with a single plate per cell cell line, the most relevant concern is the presence of *spatial bias*, that is, a dependency of the phenotypic readout on the position of the well on the plate. A common cause of such a bias is a temperature gradient inside the incubator during hibernation. This can result in an uneven evaporation of well solvents, in particular around the microplate borders, leading to an alteration of the drug concentration. Where spatial biases exist, their effects might be corrected for (Caraus et al., Ljosa and Carpenter). However, the best solution is certainly to reconduct the experiments, as statistical correction only addresses effect strength, not phenotypic alterations (for example, if more cells are dead, one cannot infer what would have happened had the concentration been lower).

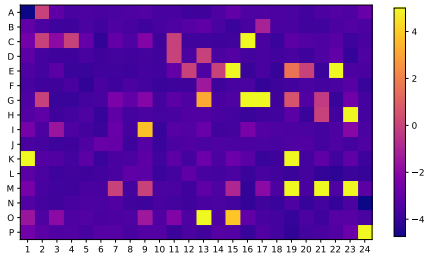
Spatial biases are investigated in Figure 3.3. These figures visualise, for both cell lines, firstly the number of cells in each of the 384 wells of the plate (indexed first by the alphabetic vertical axis, then by the horizontal numeric axis). The cell count relates directly to viability, a key readout for



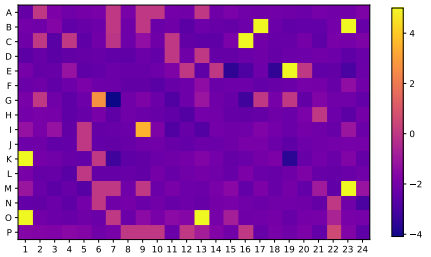
(a) MDA231 cell counts



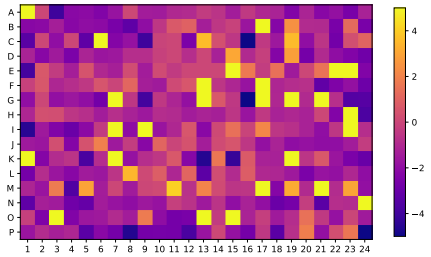
(b) MDA468 cell counts



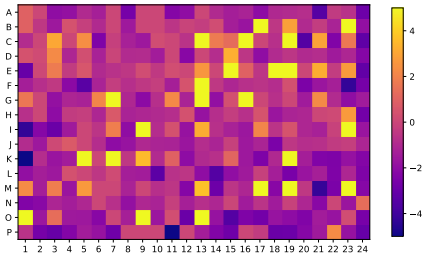
(c) MDA231 PCA1



(d) MDA468 PCA1



(e) MDA231 PCA2



(f) MDA468 PCA2

Figure 3.3: Searching for spatial biases: comparison of cell counts (a), (b); comparison of first principal component of morphological profiles (c), (d); comparison of second principal component (e), (f), arranged according to the spatial . Left column (a), (c), and (e) pertain to cell line MDA231; right column (b), (d), (f) to cell line MDA468.

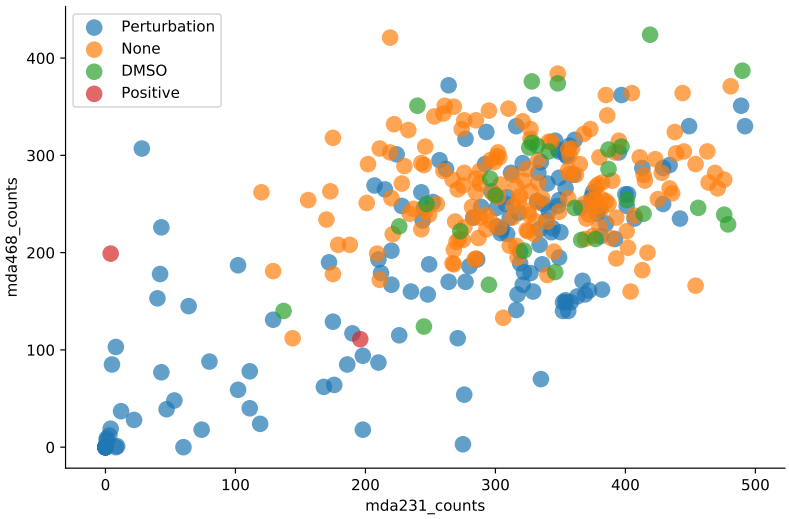


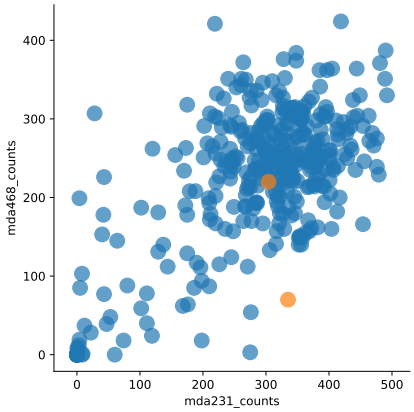
Figure 3.4: Viability comparison between cell lines for a variety of drugs and negative controls. The neutral DMSO and untreated wells strongly overlap, while showing considerable variation in both cell lines. Viability correlates well between cell lines with Pearson correlation coefficient $\rho = 0.6556$

drug potency. Also visualised are, in turn, the first and second principal components of the matrix formed from the vectors of mean feature readouts for the cells of each well. Note that these vectors amount to a simple phenotypic profile of the well condition (a construct that will be explored in great detail in Chapter 4). The principal components of the profile matrix (taken separately) therefore provide a univariate summary of phenotypic information for each well. Small clusters appear in each heat map reflecting the clustering of similar negative controls, however there is no apparent border effect or directional tendency that would indicate systematic bias for any of the three readouts. The validity of the screen data may therefore be concluded.

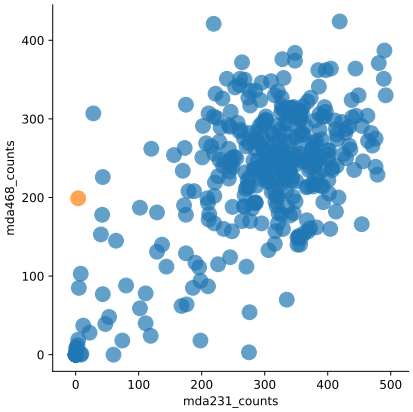
3.3.2 Viabilities of TNBC cell lines correlate

One of the initial aims of this TNBC pilot drug screen is simply to quantify cell *viability*. In a viability assay, viability is conventionally measured in the range [0, 1] (Pegg [1989]). Given that cells are seeded at a fixed density, the change in the number of dead cells with respect to the negative control is an indication of the viability. The viability,

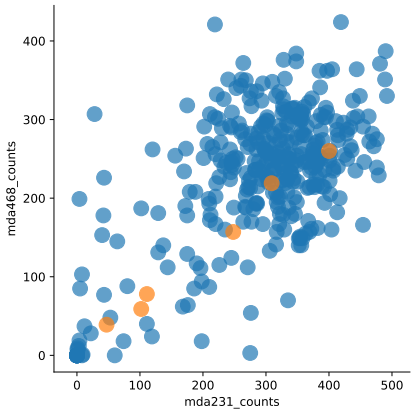
$$v = \frac{n^d}{n^-} \tag{3.9}$$



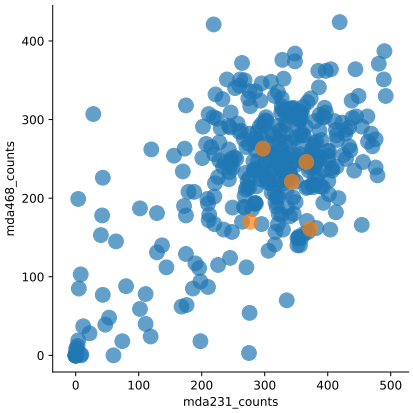
(a) Calcineurin inhibitors



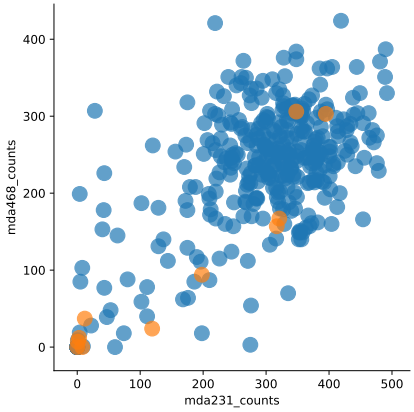
(b) DNA damage (positive)



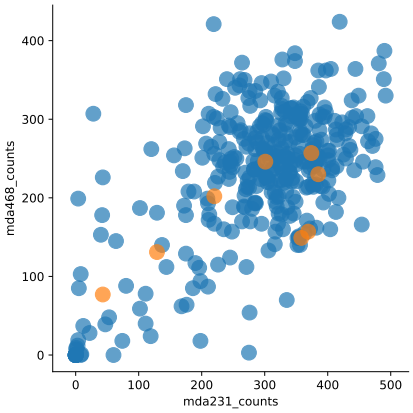
(c) CDK inhibitors



(d) MMP inhibitors



(e) PKC inhibitors



(f) Tyrosine kinase inhibitors

Figure 3.5: Comparison of viability by drug mechanism of action. One may observe differential effects by cell line: (a), (b) show differential effects on viability; (c) shows no clear divergence from the control cluster; (d), (e), (f) show a range of correlated effects.

of cells in response to a drug d refers to the change in the population size n^d relative to that of the negative control n^-. Due to the effects of *lysis* (cellular breakdown) and the washing procedure, the cells remaining at the point of imaging are the "survivors" of the drug perturbation. When cells are segmented, the viability of a drug is given by the cell sample size. When cells are not explicitly segmented, the viability is represented indirectly by how populated the field is. Viability is an important phenotypic characteristic of a drug effect. In Figure 3.4 we are able to compare the number of segmented cells (therefore a proxy viability) between well populations of cell lines MDA231 and MDA468 subjected to the same perturbations. The majority of viabilities belong to a dense cluster of negative controls (*DMSO* and *None*) with an average of 289 cells for cell line MDA231 and 227 cells for MDA468. We additionally observe a small cluster of highly potent cytotoxic drugs near the origin. There is a positive correlation of $\rho = 0.6556$ for the viability of the two cell lines, suggesting that in many cases, one can actually expect a similar viability phenotype for both cell lines. However, a closer look at Figure 3.4 shows that this is by no means true for all drugs. Some drugs act differentially on both cell lines, demonstrating the heterogeneous effect on distinct molecular TNBC subtypes discussed in section Section 1.2.1. These *differential drug effects* correspond to the "L"-structure traced by the data points skirting the horizontal and vertical axes of the plotting area.

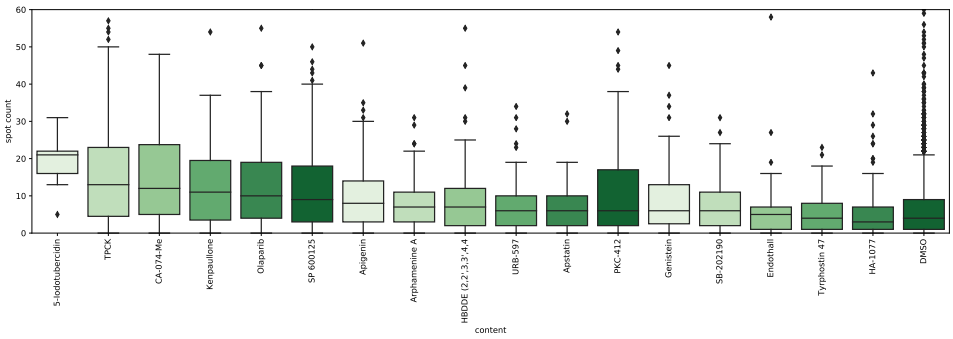
This comparison furthermore allows us to categorise drugs as having no effect, joint effect, or separate effects on the two cell lines. Figure 3.5 visualises the same population data, this time coloured by selected drug mechanism of action (MOA) categories. We see, in particular, the success of protein kinase C (PKC) inhibitors in regularly killing most–if not all–cells in both cell lines, with a small cluster nearby the origin. Note that PKC inhibitors induce interesting phenotypes under other modes of analysis, as we shall see in both Section 3.3.3 and Section 4.3.2.

3.3.3 Cell cycle modulates double-strand break rate

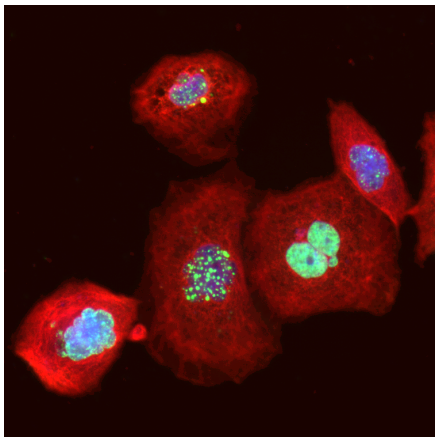
This section contains an excerpt from a paper published in at the International Symposium for Biomedical Imaging in 2018. The paper is reproduced in full in Appendix D.

An example of multivariate analysis in the assay arises in the analysis of double strand breaks (DSBs). Dysfunction of the DNA repair mechanisms is a major hallmark of cancer, also providing therapeutic opportunities. Monitoring DNA damage by the fluorescent labeling of double strand breaks (DSB) in cells is therefore an important readout in drug screening of cancer cell lines. DSBs occur when both strands of the DNA double helix are bro-

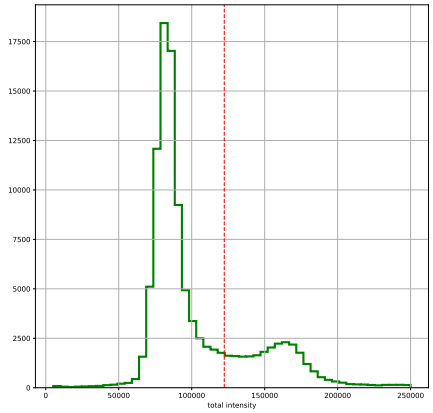
ken. DSBs have various causes, for example, cytotoxic radiation, or DNA replication over an existing single-strand break. Despite the natural DNA repair mechanisms of the cell, DSBs can be irreparable, leading ultimately to apoptosis, or hazardous DNA rearrangements. As such, DSBs are an interesting property to assess when analysing the effects of small compounds upon cancer cells. Three approaches to quantifying DSBs on the Cy3 fluorescence channel are compared, based on detecting and counting spots, granulometric features, and average intensity, which are mutually strongly correlated readouts.



(a)



(b)



(c)

Figure 3.6: Drug perturbations inducing significant changes to DSB distribution (measured by spot density) on cell line MDA231 at $p = 0.01$ with multiple testing correction, ordered by median spot density. Sample of cells perturbed with significant drug PKC-412 (b), DSBs visible as green spots above nucleus. A bimodal distribution is seen on the DAPI channel revealing a growth of mean intensity post DNA replication (c). An Otsu threshold (red vertical line) can be used to stratify the cell population, further revealing distributional differences in DSB rates.

Spot density (spot count normalised by nuclear size) is used as the basis of a DSB comparison between perturbations and negative control. The distribution of spot density from a perturbation is compared with that of the negative control in a Kolmogorov-Smirnov (KS) test. Hits are declared at a p-value of 0.01, with multiple testing controlled for with a Benjamini-Hochberg correction. Significant results are displayed in Figure 3.6a, ranked by median spot density and a sample of cells from a significant drug PKC-412 are shown in 3.6b.

Although some drugs have a clear effect on DSBs, they might also cause intermediate effects, which themselves modulate DSB rates. We see in Figure 3.6c a bimodal distribution in fluorescent density on the DAPI channel, apparently reflecting subpopulations of cells in the G_1 and G_2 phases of the cell cycle. The higher frequency of low intensities corresponds to the longer G_1 phase meaning more cells will be fixed in this phase after hibernation. Because S phase (DNA replication) occurs between the G_1 and G_2 phases, there is more DNA available on which DSBs can occur. Controlling for these phases produces different distributions of DSBs in the G_1 and G_2 subpopulations. This multivariate analysis thus reveals an *effect modifier* of cell cycle phase on DSBs. Stratification is used to establish whether increased rates of DSBs are a direct effect of perturbation or an indirect effect of a modified cell cycle. See Appendix D for further detail.

PKC inhibitors are found to feature prominently among the drugs inducing highest spot densities per cell. However, we know from Section 3.3.2 that drugs in this MOA category often cause low viability. Populations with low sample size tend not to meet the significance criteria. It is therefore drugs such as PKC-412 (Figure 3.6b), which kill fewer cells, that are ultimately declared as hits. Thus, we may identify a potential selection bias: DSBs are a cytotoxic effect that, when induced at a high rate by perturbation, should increase the rate of cell death. Given the experimental protocol, dead cells are not fixed and are lost during the fluorescence washing step. Hence, the more severe the drug effect on DSBs, the fewer the cells available for measurement. It is concluded in Appendix D that this systematic bias cannot be resolved from the available data, and can only be avoided in future assays with an adjusted imaging protocol, but it points to potential problems in the interpretation of the derived hit list.

In this section, it has been shown that even in the case that one is interested only in a single feature (here the number of DSBs or a surrogate feature), it is still beneficial to use a multivariate readout in order to extract a better understanding and a cleaner analysis. In addition, the causal inference framework has been used in order to correct for confounding variables. Given that the analysis of DSBs is very widely used in the field of cancer drug screening, it is important to alert the scientific community to potential

misinterpretations of their screening results.

3.3.4 TNBC cell lines assume distinct wild type morphologies

At a glance, one can see that TNBC cell lines manifest distinct morphologies, even in the absence of perturbation. To better understand the degree of this difference, we turn to the wild type screen on 12 cell lines (11 TNBC + negative control). The segmentation and feature extraction pipeline detailed in Section 3.2 is used. With each cell described by a vector of features, one may proceed to aggregate the readouts so as to describe the cell population in a similar way. Such a phenotypic profile can, for example, summarily characterise the response of the cell population to a given drug perturbation. Profiles, comprising of vectors of aggregated features, can then serve as the basis of comparison between drug effects on a cell line. While this methodology is explored in detail in Chapter 4, here let us consider the special case where individual cell phenotypes are identifiable by an expert annotator. The strategy is thus to classify each cell into one of several meaningful biological classes and to describe the population as a profile proportions of cells in each of the phenotypic categories. This strategy is based on Neumann et al. [2010].

In order to analyse the morphological landscape of these cell lines, seven nuclear morphology classes have been defined. Table 3.5 lists these classes for the wild type screen: *interphase*, the longest part of the cell cycle, where the cell nucleus is typically small and convex; *large interphase*, corresponding to an abnormally large interphase nucleus, potentially the result of a defect in DNA replication or nuclear membrane control; *bright interphase*, where DAPI exhibits higher fluorescence, presumably due to cell cycle deregulation; *prometaphase*, the first observable mitotic state (condensed chromosomes, broken nuclear envelope), which is relatively rare due to its short duration; *metaphase*, the mitotic phase in which the cell's chromosomes are aligned in the metaphase plate prior to chromosome segregation; *apoptosis*, cell death; and *polylobed*, in which the cell nucleus takes on an abnormal shape, due to problems in the division process.

Phenotypic profiles are not only informative about drug perturbations, but also about cell lines themselves. Indeed, a cancer cell line is supposed to have acquired properties that differentiate them from normal cells. Depending on the markers used, such differences can be measured by imaging approaches. One may hypothesise it to be interesting to identify the phenotypic profiles of the 12 cancer cell lines from the wild type screen (see Section 3.1.2), corresponding to different molecular subtypes of TNBC, in order to understand to what extent the molecular subtypes coincide with phenotypic differences.

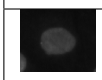
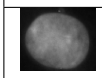
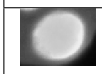
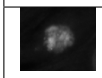
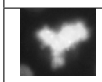
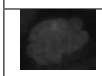
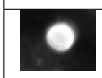
	Phenotype	Description
	Interphase	Longest phase, cell nucleus is typically small and convex
	Large interphase	Large interphase nucleus, possible replication defect
	Bright interphase	DAPI at higher intensity, due to cell cycle deregulation
	Prometaphase	Mitotic phase, prior to division (short $\Longrightarrow$ infrequent)
	Metaphase	Chromosome alignment
	Polylobed	Abnormal shape–mitosis abberation
	Apoptosis	Cell death

Table 3.5: Morphological classes manually annotated on wild type screen data to create a ground truth for training cell classifier.

This would allow us to infer the biological processes which are perturbed in these different cell lines.

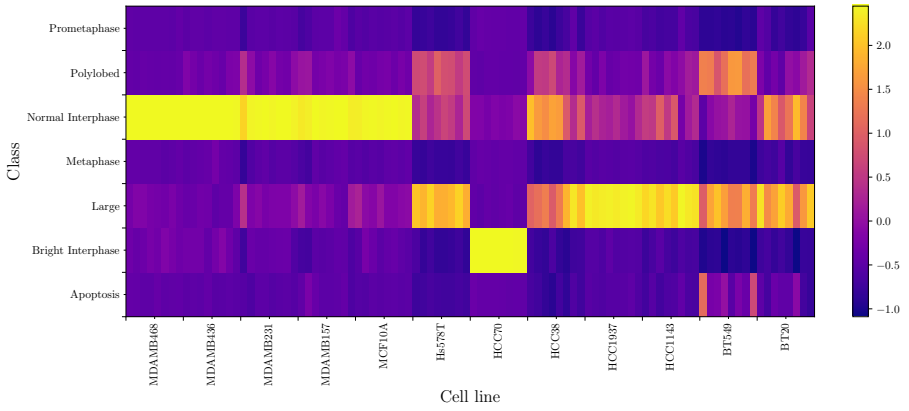


Figure 3.7: Morphological profiles of 12 TNBC cell lines with 8 replicates apiece, based on 7 biologically meaningful classes, derived from manual annotation. One observes the phenotypic similarity between TNBC families.

926 cells were annotated in the wild type screen dataset[4], according to seven morphological classes specified in Table 3.5, from which were extracted the 239 DAPI-channel nuclei features. A SVM with RBF kernel was then trained, with hyperparameters optimised by grid search and 10-fold cross-validation). The trained SVM was used to classify the remaining cells in the assay (some 36338 cells in total). The number of cells in each phenotypic class are counted, and divided by the total to create phenotype profiles for each well, that is, a summary of seven proportions for each of the 96 wells. The profiles were normalised by subtracting the average from the negative control cell line, MCF10A. The 96 profiles (8 per cell line) are visualised in Figure 3.7. We can see how families of TNBC cell lines cluster in their phenotypes. Of note are the cell lines of the drug screen, MDA231 and MDA468, which manifest predominantly normal interphase cells.

Figure 3.8 plots a (two-dimensional) UMAP dimensionality reduction (McInnes and Healy [2018]) of a balanced sample of 150 cells from 4 of the 12 cell lines. We see a clear separation of cell lines in the UMAP embedding, mirroring their distinctive appearance in the microscopy. Note that a subset of cell lines was chosen to avoid clutter, but the same degree of separation was observed across the 12. The negative control cell line (MCF10A) is included, as is one of the cell lines studied in the drug screen (MDA231).

A random forest classifier with 500 trees is also trained to classify cell lines in feature space. The data set is built from a balanced sampling of 150 cells per cell line, totaling 1800 cells, with 25% of the data reserved for testing. The random forest achieves in excess of 70% accuracy on the 450 test samples. A confusion matrix is provided in Appendix A. Repeated experiments reveal that the greatest confusion occurs between two cell lines of the same family: MDA231 and MDA436. Thus, even with a small fraction of the total data available for training (and little to no tuning), an off-the-shelf classifier achieves a strong accuracy. This demonstrates that the cells of distinct cell lines occupy different regions of the feature space and that their wild type morphologies are different.

In this section we have seen two levels of phenotypic distinction between TNBC cell line wild types. The first concerns the regulation of cell states, as in Figure 3.7, where we saw how cell populations over different cell lines could be differentiated in a morphological profile, which appeared to cluster by pathology. Through annotation (first manual, then by classifier), these morphological classes abstracted away from the second distinction, that of the morphology of individual cells within each cell line, as in Figure 3.8. It is overcoming this second distinction that will play a strong role in the motivations of the methods of Chapter 4, albeit in an unsupervised setting

[4]Later, a larger set of 6530 cells were annotated and released for pedagogical purposes at https://github.com/jcboyd/deep-learning-workshop/tree/cell-data

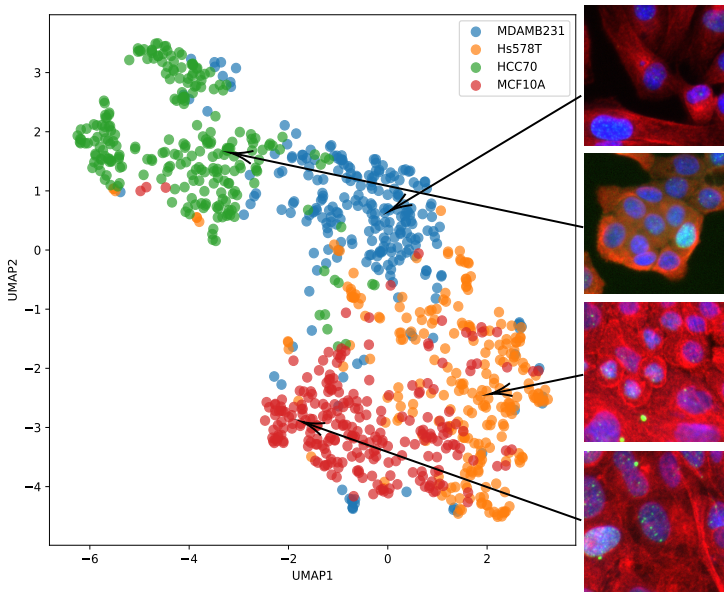


Figure 3.8: UMAP projection from feature space of sample cells from four TNBC cell lines (selected among 12) in a wild type screen. On the right we show a fixed-size (128×128px) crop centered on an indicative cell from each cell line.

where manual annotation is not feasible.

Chapter 4

Domain-invariant features for mechanism of action prediction in a multi-cell-line drug screen

Summary: *We have seen previously that high content screening is an important tool in drug discovery and characterisation. Often, high content drug screens are performed on one single cell line. Yet, a single cell line cannot be thought of as a perfect disease model. Many diseases feature an important molecular heterogeneity. Consequently, a drug may be effective against one molecular subtype of a disease, but less so against another. To characterise drugs with respect to their effect not only on one cell line but on a panel of cell lines is therefore a promising strategy to streamline the drug discovery process. The contribution of this chapter is twofold. First, we investigate whether we can predict drug mechanism of action (MOA) at the molecular level without optimisation of the MOA classes to the screen specificities. To this end, we benchmark a set of algorithms within a conventional pipeline, and evaluate their MOA prediction performance according to a statistically rigorous framework. Second, we extend this conventional pipeline to the simultaneous analysis of multiple cell lines, each manifesting potentially different morphological baselines. For this, we propose multitask autoencoders, including a domain-adaptive model used to construct domain-invariant feature representations across cell lines. We apply these methods to a pilot screen of two triple negative breast cancer cell lines as models for two different molecular subtypes of the disease.*

Résumé: *Ce chapitre...*

4.1 Overview

As discussed in Chapters 1 and 3 high content screening (HCS) is a powerful tool for identifying potential drugs effective against a particular disease. The idea is to expose a cell line representative of some disease to a large panel of drugs. For each drug, one obtains a set of images informative of its phenotypic effect and hence on the biological pathways undergoing perturbation. Various advances in microscopy automation and image analysis have pushed HCS to the early *hit-to-lead* stages of the drug discovery process Haney et al. [2006].

The discovery of new drugs may be guided by a reference set of drugs of known mechanism of action (MOA). The MOA of a drug is the particular cellular pathway it perturbs to achieve its effect. Through application of image analysis, one may attempt to infer the MOA of an unknown drug from HCS image data. Note that MOA can be defined at different levels and with different degrees of specificity: MOA might concern the exact protein that is targeted (e.g. AURKA inhibition), or a specific effect on cellular components (e.g. stabilisation of microtubuli) or perturbation of a more general cellular pathway (e.g. DNA repair). HCS is usually optimised with respect to particular pathways by the choice of the fluorescent markers and readouts Pepperkok and Ellenberg [2006b]. Consequently, MOA prediction might be reasonably straightforward if MOA classes are chosen in accordance with the phenotypic readout Ljosa et al. [2013], but it is challenging in general, in particular if we aim at predicting specific MOAs the assay has not been optimised for.

A second difficulty concerns the cellular model that is used. As a proxy for diseased cells, a cell line cannot be thought of as a perfect model. Many diseases feature a significant molecular heterogeneity. Consequently, a drug may be effective against one molecular subtype of a disease, but less so against another. Furthermore, immortalised cell lines may diverge over time due to genetic drift. For example, HeLa, the quintessential cell line, is famously the cause of great scientific confusion due to difficulties in cell line identification Horbach and Halffman [2017] and significant molecular and phenotypic variability Liu et al. [2019]. To characterise drugs with respect to their effect not only on one cell line but on a consensus of several is therefore a promising strategy to streamline the drug discovery process. Nevertheless, this is not an easy task in morphological screening, as different cell lines usually have distinct archetypal morphologies even prior to perturbation, as shown in section **??**. It is therefore conceptually difficult to characterise and compare drug effects across cell lines.

In this Chapter, we investigate whether we can predict MOA at the molecular level without optimisation of the MOA classes to the screen specificities.

To this end, we benchmark a set of algorithms within a conventional pipeline, and evaluate their MOA prediction performance according to a statistically rigorous framework.

Second, we extend this conventional pipeline to the simultaneous analysis of multiple cell lines, each with potentially different morphological baselines. For this, we propose multitask autoencoders, including an adaptive model used to construct domain-invariant feature representations across cell lines. We apply these methods to a pilot screen of two triple negative breast cancer (TNBC) cell lines as models for two different molecular subtypes of the disease.

In Section 4.2 we formalise a range of profiling approaches from the literature according to four key properties, and extend this to a multi-cell-line analysis. In Section 4.3 we illustrate the benefit of multi-task models for our dataset through extensive cross-validation and provide an exploratory analysis of differential drug effects between the two cell lines. In Section 4.4 we discuss our methods and the obtained results.

4.2 Phenotypic profiling for mechanism of action prediction

This section describes the approaches for phenotypic profiling we have benchmarked. We embed these descriptions in a formalised overview of phenotypic profiling strategies to motivate the different setups. In Section 4.2.3 we describe methods for a joint analysis of multiple cell lines.

4.2.1 MOA prediction

Drugs are assigned a class based on their *mechanism of action* (MOA), the cellular pathway perturbed by the drug, as depicted in Figure 4.1. Given a set of drug profiles annotated with MOA classes, we can simulate reference and discovery drug sets in a *leave-one-compound-out* cross-validation (LOCOCV) scheme. At each fold of the cross-validation, we hold out a drug and predict its MOA class using a classifier trained on the remaining "reference" drugs. The prediction is made as the nearest neighbour (1-NN) in cosine distance between drug profiles, $d(\mathbf{p}, \mathbf{p}') = 1 - \cos\theta_{\mathbf{p},\mathbf{p}'}$. This was proposed in Ljosa et al. [2013] as an equitable way of comparing profiling algorithms. We settle for this lightweight approach as our focus here is on the discriminative power of the profiles.

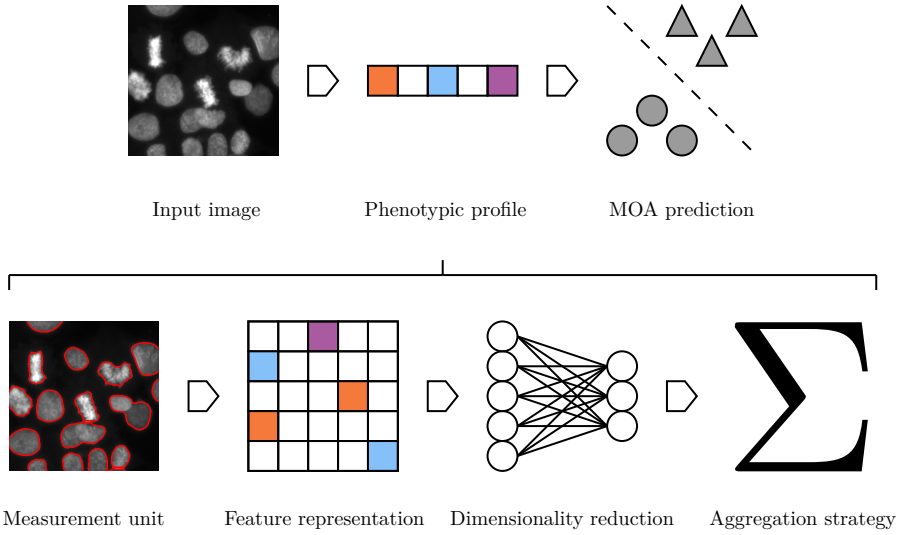


Figure 4.1: MOA prediction is performed on an image via a phenotypic profile. The development of such a profile spans four ordered stages. Each stage may be accomplished by a variety of algorithms, the combination of which define a unique pipeline. Some stages may be omitted in certain pipelines, or subsumed to a common framework.

4.2.2 Phenotypic profiling

As stated in Section 1.1.3, the conventional approach to HCS analysis is a multi-stage pipeline, consisting of a sequence of modules of image and statistical analysis, including cell segmentation and hand-crafted feature extraction Caicedo et al. [2017]. The aim is to ascribe a phenotypic profile to each cell population to serve as the basis of comparison between drugs. Each profile will take the form of a vector $\mathbf{p} \in \mathbb{R}^D$ of some dimensionality D and is constructed according to four ordered methodological stages: measurement unit, feature representation, dimensionality reduction, and aggregation strategy (Figure 4.1). Certain properties may be omitted by some approaches, or subsumed to a common framework, such as a neural network, that may perform each task simultaneously Kraus et al. [2016], Godinez et al. [2017]. In the following sections we detail each property in turn, providing references to the relevant literature and describing the concrete setup that was retained for the benchmarking.

Measurement unit

The most common measurement unit is the cell itself, constituting a *per-cell* analysis. This entails an initial segmentation of the cells (their nuclei

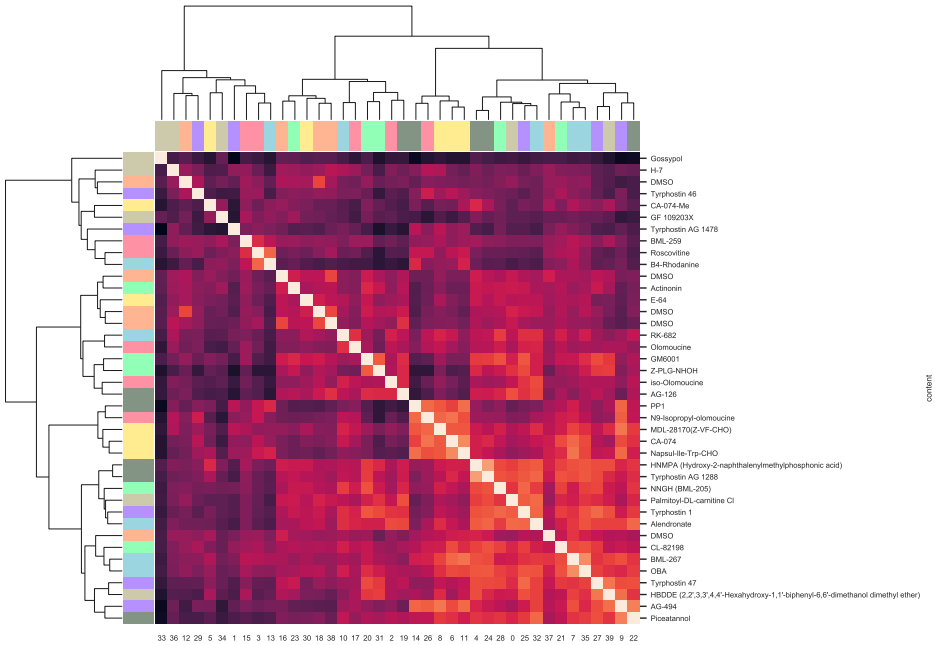


Figure 4.2: Example of how phenotypic profiles may cluster with a hierarchical model and Ward linkage for 40 drugs in 8 mechanism of action classes (including negative control)s from our drug screen data set. Heat map colour indicates distance between profiles, and dendrogram leaf colours indicate mechanism of action class.

and other organelles). We segmented cell nuclei on the DAPI channel by subtracting a background image formed with a mean filter, before clipping to zero. Touching nuclei were further separated by applying the watershed transform on the inverse distance map of the foreground image. The cytoplasm was segmented from the microtubule channel (Cy5) following Jones et al. [2005].

Alternatively, one might analyse the image field directly in a *per-image* analysis, such as in Orlov et al. [2008], Uhlmann et al. [2016], or Godinez et al. [2017]. Such approaches are referred to as *segmentation-free*, as they obviate the segmentation phase of the conventional pipeline. In this study, we deliberately choose to focus on the cell as unit of measurement.

Feature representation

For a given choice of measurement unit, one further chooses a feature representation. This yields a matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ for each well where N is

the number of samples for that well and D is the number of features measured. For each segmented cell we extracted a previously published set of features (Walter et al. [2010b]) across the three fluorescent channels, as well as spot features informative on DNA double-strand breaks (Boyd et al. [2018]). These features, hereafter referred to as *handcrafted features*, thus retain a degree of biological interpretability. In contrast, in Orlov et al. [2008] and Uhlmann et al. [2016] a large number of handcrafted features are extracted over each image as a whole.

More recently, features are extracted within the layers of a convolutional neural network (CNN) trained directly on image pixels. We benchmarked a convolutional autoencoder (CAE) following the design of Sommer et al. [2017], trained on $40\times40\times3$ inputs, formed by extracting 100×100px padded bounding boxes of segmented cells, rescaling, and stacking the fluorescent channels. The central hidden layer of the trained CAE is then used as a feature representation.

Dimensionality reduction

Dimensionality reduction requires some function $enc : \mathbf{X} \to \mathbf{Z}$ where $\mathbf{Z} \in \mathbb{R}^{N\times M}$, with reduced dimensionality $M < D$. The objective is to capture the essential information in lower dimension or to cast the high-dimensional feature vector to an interpretable representation. Supervised classification of individual cells Neumann et al. [2010] is one way of achieving this, as each cell is represented either by a one-hot binary vector $\mathbf{z}_i \in \{0,1\}^M$ or by a vector of probabilities $\mathbf{z}_i \in [0,1]^M$ where $\sum_j z_{ij} = 1$ and M is the number of classes, in effect, the new dimensionality. With multiple-instance learning (MIL) Kraus et al. [2016] one can circumvent the manual effort involved in creating a phenotypic ontology and a manually curated training set. Here, one labels each cell with the MOA of the drug of the population, thus creating a weakly supervised ground truth. As individual cells may respond differentially to perturbation, not all regions of an image will bear the hallmarks of a particular drug, but the cellular landscape can be viewed as a multiple instance bag of objects. Godinez et al. [2017] make this assumption implicitly. We benchmarked a random forest tuned to 500 trees, trained on cells weakly labeled by MOA class of their well ($M = 8$). Necessarily, we partition wells into separate train and test sets, where the test data alone is used to build profiles for the MOA prediction downstream.

Another popular option is to use unsupervised learning. We benchmarked hard clustering methods k-means and hierarchical clustering in Euclidean space with Ward linkage. These were tuned to $M = 80$ and $M = 100$ clusters respectively (by cross-validation, on the training set). K-means is fast to fit approximately, in particular using mini-batch training. On the other hand,

even using optimised software Müllner et al. [2013], hierarchical clustering is not scalable. We also performed soft clustering with Gaussian mixture models (GMM) Slack et al. [2008], tuned to $M = 100$ Gaussians.

Feature selection Loo et al. [2007] and principal components analysis (PCA) are other popular options. Here, we applied PCA on the handcrafted features, selecting 40 of the 516 components, retaining $\sim 90\%$ of the energy on average. We further whitened the latent features.

Autoencoders, as used by Kandaswamy et al. [2016], formulate a function $f(\mathbf{x}) = dec(enc(\mathbf{x}))$, where $enc(\cdot)$ and $dec(\cdot)$ correspond to the encoder and decoder parts of the neural network. This model can be trained with a mean square error (MSE) loss function,

$$\mathcal{L}(\mathbf{X}; \boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} ||\mathbf{x}_i - f(\mathbf{x}_i)||_2^2 + \lambda ||\boldsymbol{\theta}||_F^2 \tag{4.1}$$

for the N samples in the dataset and where λ is a tunable hyperparameter for the regulariser. The hidden representation corresponds to the output of the encoder, the central layer of the neural network, i.e. our reduced sample is $\mathbf{z}_i = enc(\mathbf{x}_i)$. We train shallow affine autoencoders–with a single hidden layer (tuned to $M = 100$ neurons)–on handcrafted features. We also train deep convolutional autoencoders directly on image pixels, as described in Section 4.2.2. Note that such models perform both feature extraction and dimensionality reduction simultaneously. Here, the encoder consists of 5×5 and 3×3 convolutional layers, with 16 and 8 kernels respectively, and a fully connected layer ($M = 128$), each alternating with max pooling layers. The decoder mirrors this, albeit replacing pooling with upsampling.

Aggregation strategy

Once all cells are endowed with a representation, one needs some means of reducing the population to a single profile, $\mathbf{p}$. A variable number of cells per well requires an aggregation strategy yielding a profile of fixed size. The most straightforward approach is an element-wise averaging as in Adams et al. [2006] where $\mathbf{p} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{z}_i$. This amounts to replacing the cell population cluster with its own centroid, and for classification or clustering approaches (Section 5.2.1), this simply corresponds to the percentage of cells that fall into each category.

Alternatively, Perlman et al. [2004] apply an element-wise Kolmogorov-Smirnov test and Loo et al. [2007] use the vector normal to the SVM decision boundary between perturbed and control populations. As we are modeling

the negative control as one of our ground truth classes, we aggregate exclusively with element-wise averaging in our analysis.

4.2.3 Multi-cell-line analysis

One can extend the above MOA prediction framework for multiple cell lines either by pooling data or by ensembling models. In a pooling analysis such as Warchal et al. [2016], the cells of the respective cell lines are first normalised and then grouped across drugs to increase the amount of available data. An ensemble approach such as in Rose et al. [2018] creates models for each cell line and aggregates their individual predictions. This approach has the additional advantage of allowing different imaging modalities of fluorescent markers.

We adopted a pooling approach to predict MOA from multiple cell lines. The challenge of this approach is to reconcile the inherent differences between the cell lines in feature space, which derives from the fundamental morphological differences of the cell lines. For this purpose, we tested multi-task autoencoders (Figure 4.3), extensions of both our affine and convolutional autoencoders.

Multitask autoencoders for multi-cell-line analysis

Multi-task models learn to predict multiple targets simultaneously and multitask neural nets often build more generalised internal representations Caruana [1997]. We propose multitask autoencoders as an approach to reconcile the divergent nature of our multi-cell-line data.

One obvious design is to have separate decoders for each cell line with a shared encoder. During training, minibatches can be split after the shared layers with samples routed to the decoder corresponding to their cell line. We thus minimise,

$$\mathcal{L}_{MTA}(\mathbf{X};\boldsymbol{\theta}) = \sum_{i:d_i=0} ||\mathbf{x}_i - dec_s(enc(\mathbf{x}_i))||_2^2 + \sum_{i:d_i=1} ||\mathbf{x}_i - dec_t(enc(\mathbf{x}_i))||_2^2 \tag{4.2}$$

where d_i identifies the cell line of $\mathbf{x}_i$. We test multitask variants of both our affine and convolutional autoencoders described in Section 5.2.1.

The fundamental morphological differences between the cell lines can be quantified in feature space by a $\mathcal{H}$-divergence, first proposed by Ben-David

et al. [2010], where $\mathcal{H}$ is some hypothesis class (such as the space of linear classifiers). This is expressed as $d_{\mathcal{H}}(D_S^X, D_T^X) = 2\sup_{h\in\mathcal{H}} \left| P_{\mathbf{x}\sim D_S^X}(h(\mathbf{x}) = 1) - P_{\mathbf{x}\sim D_T^X}(h(\mathbf{x}) = 1) \right|$, where the domains D_S^X and D_T^X are marginal probability distributions on $\mathbf{x}$. That is, given source and target domains, and given a hypothesis class $\mathcal{H}$, the divergence between the source and target domains is the best performance among that class of classifiers trained to distinguish them. In practice, we can approximate this by training a classifier of the class $\mathcal{H}$ on the constructed dataset, $U = \{(\mathbf{x}, 0) : \mathbf{x} \in S\} \cup \{(\mathbf{x}, 1) : \mathbf{x} \in T\}$, that is, a classifier trained to distinguish between the domains. Ajakan et al. [2014] proposed multi-task classifiers involving a domain discriminator trained against a classifier adversarially. As the classifier was trained to minimise one loss, the competing domain discriminator was trained to maximise another loss, such that data from either domain could not be distinguished, promoting *domain-invariant features* in the earlier, shared layers of the network.

Thus, we propose domain-adversarial autoencoders (DAA), to promote domain-invariant representations between the cell lines. This consists of attaching a domain discriminator $g(\mathbf{x})$ to the encoding layer. This can be thought of as a dynamic regularisation function. In a bias-variance tradeoff, we expect this to on average *increase* the reconstruction error of the autoencoder. However, we hypothesise that the domain-invariant features learned will be more useful to the downstream MOA prediction when combining heterogeneous cell line data. For example, with a single additional affine layer, $g(\mathbf{x}) = \mathcal{S}(\mathbf{W}_d enc(\mathbf{x}) + \mathbf{b}_d)$, where $\mathbf{W}_d$ and $\mathbf{b}_d$ are the weights and biases of the layer, and $\mathcal{S}$ is the softmax function producing posterior probabilities $p(d = 0|\mathbf{x})$ and $p(d = 1|\mathbf{x})$. The loss function then becomes,

$$\mathcal{L}_{DAA}(\mathbf{X}, \mathbf{d}; \boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{x}_i - f(\mathbf{x}_i)||_2^2 - \frac{\omega}{N}\sum_{i=1}^{N} d_i g(\mathbf{x}_i) - \log[1 + \exp(g(\mathbf{x}_i))] \tag{4.3}$$

that is, the difference of a mean square error (MSE) loss and a log loss, where $f(\mathbf{x})$ is defined as before, and ω is a modulating hyperparameter. However, now the parameters of $g(\mathbf{x})$ are updated to *maximise* $\mathcal{L}_{DAA}$, so as to improve domain discrimination. At the same time, the parameters of $f(\mathbf{x})$ are updated to *minimise* $\mathcal{L}_{DAA}$. This has the dual effect of minimising the MSE (as usual) but also maximising the log loss. This is known as an adversarial step, and aims at converging to a saddle point between the two objectives. In practice, this is implemented with a *gradient reversal* pseudo-

layer Ganin et al. [2016], which is readily programmable in standard deep learning frameworks.

We test multitask versions of both our affine and convolutional autoencoders, and compare them directly in Section 4.3.2. The domain discriminator of our DAAs are linear in terms of the encoding (domain invariant features) and the weight of the log loss was tuned to $\omega = 1.5$. For each affine model we tried the same range of hidden units in a grid search $M \in \{100, 125, 150, 175, 200\}$, and trained for 20 epochs using the RMSprop gradient descent algorithm Tieleman and Hinton [2012]. For the convolutional autoencoders we kept the architecture defined in Section 5.2.1. We further used weight decay ($\lambda = 10^{-3}$) for all models as well as batch normalisation Ioffe and Szegedy [2015], which we found stabilised the training, in particular the adversarial training.

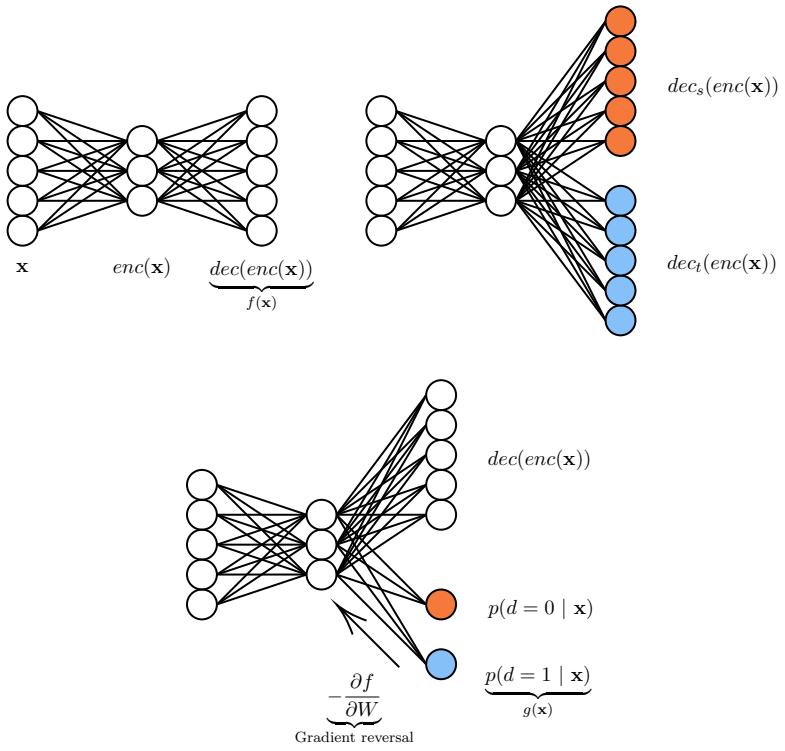


Figure 4.3: Multitask autoencoders used for dimensionality reduction over multi-cell-line data. Clockwise from top left: vanilla autoencoder, multitask autoencoder, and domain-adversarial autoencoder. Colouring indicates separate treatment of each domain (cell line).

4.2.4 Model evaluation

We compared different profiling settings by evaluating performance on a MOA prediction task. For this, we balanced our datasets by randomly sampling five drugs from each of the 8 classes specified in Section **??**, analysing 40 drugs at a time. Applying the LOCOCV scheme described in Section 4.2.1, we note that random accuracy is 12.5%. To account for random variability, we repeated LOCOCV 60 times with different sets of randomly sampled drugs and in Section 4.3 report average top-1 accuracy and standard deviation as the percentage of MOAs correctly predicted by the 1-NN classifier. We consider this to be a more rigorous approach in a comparative study, as while a given method often fit one drug set well, it was harder to find hyperparameter choices that worked well across all sets. We used a Wilcoxon signed-rank test to establish significance against baselines over the 60 rounds.

4.2.5 Software

We use Cell Cognition Held et al. [2010] to perform the first stages of the classical analysis pipeline, namely, image preprocessing, cell segmentation and feature extraction.

All models were coded using the `scikit-learn` Pedregosa et al. [2011] and `Keras` Chollet et al. [2015] frameworks for Python, unless otherwise noted[1]. Basic image processing was performed with `scikit-image` van der Walt et al. [2014].

4.3 Results

In Section 4.3.1 we evaluate a range of approaches to dimensionality reduction–as described in Section 5.2.1–on their utility in creating cell representations that aggregate into discriminative phenotypic profiles for MOA prediction. This we do in separate single-cell-line experiments. In Section 4.3.2 we show how our best performing model on single-cell-line data–the autoencoder–may be extended for multi-cell-line analysis, providing comparisons for learning on handcrafted features, as well as raw pixels. We then illustrate how our optimised phenotypic profile design can be used to identify differential drug effects between cell lines across our entire drug panel in Section 4.3.2. In Section 4.3.2 we explore how the effect of adding cell lines to an analysis effects MOA predictability.

[1] Worked examples of code and feature data available at `https://github.com/jcboyd/multi-cell-line`

Approach	MDA231 accuracy (μ, σ)	MDA468 accuracy (μ, σ)
Handcrafted features	$18.58, 5.62$	$20.08, 4.49$
PCA + whitening	$21.33, 6.54^{*}$	$19.58, 6.43$
Hierarchical clustering	$17.83, 6.46$	$20.13, 6.46$
K-means	$19.38, 7.56$	$19.50, 5.86$
GMM	$20.21, 6.88$	$21.29, 7.37$
Autoencoder	$\mathbf{22.13}, \mathbf{6.48}^{**}$	$\mathbf{23.92}, \mathbf{6.23}^{**}$
Random forest (MIL)	$19.51, 9.95$	$16.81, 8.16$
Conv. autoencoder	$19.96, 6.23$	$13.79, 5.51$

Table 4.1: Comparison of dimensionality reduction approaches against unreduced baseline for cell lines treated separately. We show mean and standard deviation of accuracies over 60 runs with (*) indicating significant results at the $p = 0.05$ level; (**) at the $p = 0.01$ level.

4.3.1 Single cell line analysis

In Table 4.1 we evaluate a range of approaches to dimensionality reduction on cell lines taken separately. The baseline for this comparison are the hand-crafted features averaged element-wise from segmented cells in each well. Note that even such a simple baseline proved to be highly competitive in earlier comparative studies such as Ljosa et al. [2013]. The models are used to create a reduced representation of cells prior to aggregation by element-wise averaging (Section 4.2.2). The one exception is the convolutional autoencoder, which learns cell representations directly from image pixels.

We observe dimensionality reduction techniques register broad improvement over the baseline, with PCA ($W = 385.0, p < 0.05$) and autoencoders ($W = 281.5, p < 0.01$) significant for the MDA231 cell line. Autoencoders also registered significant improvement ($W = 356.0, p < 0.01$) for the MDA468 cell line. This further motivates autoencoders as the benchmark in our multi-cell-line analysis (Section 4.3.2).

The deep convolutional autoencoder fails to stand out from the group. However, this may rather testify to the effectiveness of handcrafted features on cell line data–at least at this resolution–over learning representations from scratch.

The sole weakly supervised method, multiple instance learning (MIL) with random forests, shows promise on cell line MDA321, but falls short on MDA468. This may stem from the necessary splitting of data into train and test sets prior to LOCOCV, reducing the available training data. Approaches based on weakly supervised MIL are popular, particularly for deep learning approaches, but we do not see any benefit for them on our

Approach	Pooled cell line accuracy (μ, σ)
Autoencoder	$31.67, 6.43$
Multitask autoencoder	$32.04, 6.88$
Domain-adversarial autoencoder	$\mathbf{35.67}, \mathbf{6.94}^{**}$

Table 4.2: MOA prediction on multiple cell lines (pooled) with autoencoders trained on handcrafted features. From top to bottom: vanilla autoencoders (baseline), multitask autoencoders and domain-adversarial autoencoders. We compare with the vanilla autoencoder (top row) ($(^{**}) : p < 0.01$)

dataset.

4.3.2 Analysis on multiple cell lines

So far, we have considered the analysis of several cell lines as independent problems to inform model selection. We now turn to a joint analysis on multiple cell lines.

Prediction of MOA from multiple cell lines

With their different transcriptional programs multiple cell lines potentially bear complementary information on the mechanism of action of a drug. We pool cells in corresponding wells across our two cell lines, thus enlarging the available data for each drug. In each case, the data from each cell line were standardised separately to have zero mean and unit variance for all features. Our multitask autoencoders are compared with their single-task counterparts, the best performing models from Section 4.3.1.

We observe in both Tables 4.2 and 4.3 that we obtain a higher degree of accuracy in MOA prediction for our multitask autoencoders compared with their baselines, particularly the domain-adversarial autoencoders, which achieve a statistically superior average accuracy ($W = 283.5, p < 0.01$) for the shallow variant, based on handcrafted features, as well as for the deep learning variant ($W = 438.5, p < 0.01$). The former constitutes our best overall accuracy in MOA prediction on this dataset. This supports our hypothesis that promoting domain invariant features facilitates the mixing of heterogeneous data from multiple cell lines. As anticipated in Section 4.2.3, adversarial training did not improve the reconstruction error of our autoencoders, but the resultant features performed better downstream in the MOA prediction pipeline.

Inspired by Ganin et al. [2016], we use t-SNE Maaten and Hinton [2008] to project a sample of learned cell features into two dimensions. We typi-

Approach	Pooled cell line accuracy (μ, σ)
Conv. autoencoder	19.58, 6.98
Multitask conv. autoencoder	20.42, 6.14
Domain-adversarial conv. autoencoder	**22.38**, **5.91****

Table 4.3: MOA prediction on multiple cell lines (pooled) with convolutional autoencoders. From top to bottom: vanilla convolutional autoencoders (baseline), multitask convolutional autoencoders and domain-adversarial convolutional autoencoders. We compare with the vanilla convolutional autoencoder (top row) ($(^{**}) : p < 0.01$).

cally observe a higher degree of alignment between the feature distributions of the two domains as produced by the domain-adversarial model, as illustrated in Figure 4.4. To quantitatively confirm this domain overlap, we compute the mean silhouette score over all points where the cluster identity of each point is simply its domain class. The scores given in Figure 4.4 of 0.11 (lower overlap) and 0.01 (higher overlap) for unadapted and adapted features respectively are typical.Altschuler and Wu [2010] wrote that multiple modalities render aggregation over a cell population problematic, as a centroid may be a bad representative of the overall population. Computing domain invariant features appears to be a partial remedy to this when pooling heterogeneous data in a multi-cell-line analysis.

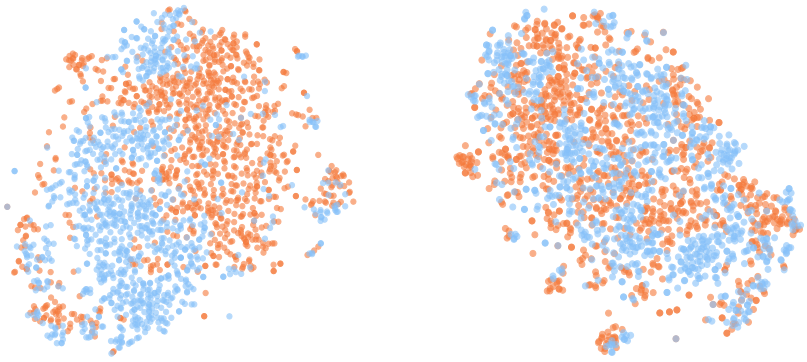


Figure 4.4: t-SNE embeddings of encodings from autoencoder (left) and domain-adversarial autoencoder (right), with cell lines distinguished by colour, and mean silhouette scores of 0.11 and 0.01 respectively.

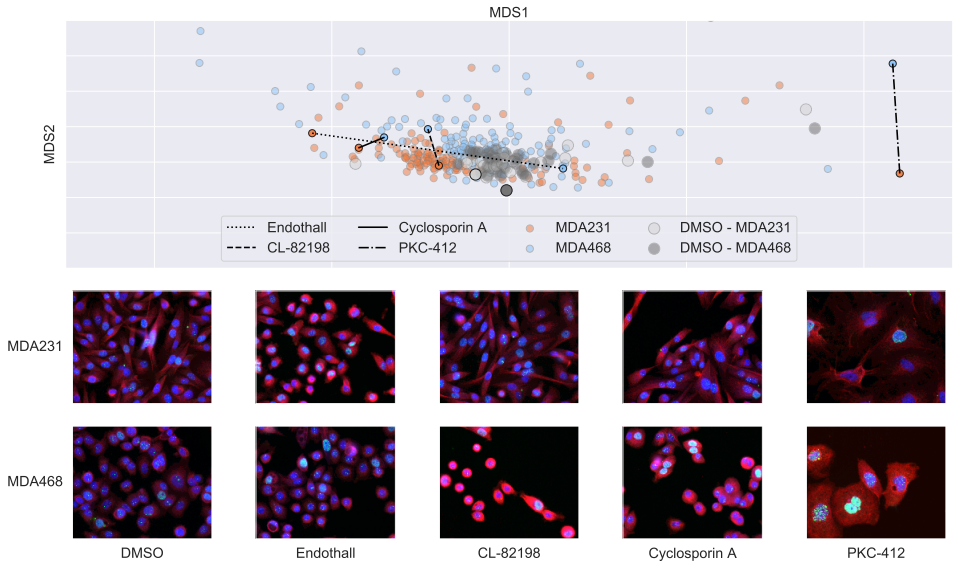


Figure 4.5: MDS embedding of drug effect profiles for MDA231 and MDA468 cell lines with DMSO centroid centered on origin. Detection of differential drug effects between cell lines with examples for each category below (MDA231 top, MDA468 bottom). From left to right: no drug effect in either cell line (negative control); drug effect in MDA231 cell line only; drug effect in MDA468 cell line only; similar drug effects in both cell lines; differentiated drug effects in both cell lines. Shown are example images, blue: DAPI, red: microtubules, green: DSB.

Differential drug effects across cell lines

Our DAA approach provides us with a representation that is optimised with respect to both MOA prediction accuracy and domain invariance between the cell two lines. This can assist us in producing profiles for all drugs in our pilot screen and investigate the differential effects of drugs across cell lines. For this, we trained our network on all data, producing phenotypic profiles for all drugs in the screen. We zero-centered each cell line by subtraction of their respective DMSO centroids and compared distances of drug profiles both from the DMSO centroid and between cell lines. By ranking these distances, we can identify four drug effect cases:

- no drug effect in either cell line;
- drug effect in one cell line only;
- differentiated drug effects in both cell lines;
- similar drug effects in both cell lines.

We visualise the relative distances between drug profiles using multi-dimensional scaling (MDS) on Euclidean distance in Figure 4.5 and identify examples of each of these cases. We include a comparison of DMSO populations that illustrate the unperturbed morphological differences between the two cell lines. We first show an indicative sample of DMSO cells from each cell line. Among the drugs, Endothall has a phenotypic effect on MDA231, but no visible effect on MDA468 (MDA231 cells are rounded up and smaller than in DMSO). Conversely, CL-82198 has an effect on MDA468 cells (cells are smaller and display cytoskeletal changes) and no visual effect on MDA231 cells. Cyclosporin A has a similar effect on both cell lines; the cell lines actually preserve many of their morphological baseline differences, but have a higher fraction of binucleated cells. PKC-412 has a differential effect on both cell lines. While the cell size is increased, the morphological properties as well as the number of DSBs seem to be very different between cell lines.

Effects of accumulating cell lines

Rose et al. [2018] demonstrated an increasing accuracy in MOA prediction as data from cell lines are added to create a growing ensemble of predictive models. This illustrates the value of drawing upon several biological sources to guide a drug discovery process. Nevertheless, predictive models will tend to perform better when supplied with greater volumes of data anyway. Any attribution of a model's success to a richer biological foundation must first correct for the confounding effect of an increasing sample size.

We ran a separate experiment controlling for the aforementioned bias to attempt to measure the effective power of heterogeneous cell line data. To do this we created equally sized samples: 10000 randomly subsampled cells from the MDA231 cell line; 10000 randomly subsampled cells from the MDA468 cell line; and 5000 cells sampled from each cell line and pooled into a multi-cell-line dataset. We did this for the handcrafted features of segmented cells, averaged element-wise, again repeated over the 60 experimental folds. We found the pooled samples yielded an average accuracy of 20.89, significantly improving over the pure MDA231 sample at 14.94 ($W = 240.0, p < 0.01$) and the pure MDA468 sample at 19.42 ($W = 516.0, p < 0.1$). This therefore supports the hypothesis that a multi-cell-line analysis can be advantageous in and of itself, even before accounting for any increased sample size.

Generalising to further cell lines

To test how our model behaves with increasing numbers of cell lines, we acquired image data without drug perturbation of a third TNBC cell line

(MDA-MB-157) under the same protocol as the pilot screen. In order to apply domain adaptation to K cell lines with $K > 2$, the log loss in equation 4.3 was replaced with a cross entropy loss,

$$\mathcal{L}_{DAA}(\mathbf{X}, \mathbf{d}; \boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{x}_i - f(\mathbf{x}_i)||_2^2 + \frac{\omega}{N}\sum_{i=1}^{N} \log p_{d_i} \tag{4.4}$$

where p_{d_i} is the softmax probability produced by the domain discriminator, indexed by the domain of ith sample, and all other terms are defined as before. We found that this simple modification sufficed to train effectively on the new dataset, provided ω was reduced as the number of cell lines increased. We produce a t-SNE plot in Figure 4.6 and observe a similar tendency of distributional overlap for three cell lines. Interestingly, there is only a moderate number of additional parameters when we add a new cell line, which contrasts to the multi-task autoencoders where a whole new decoder is required for each new cell line.

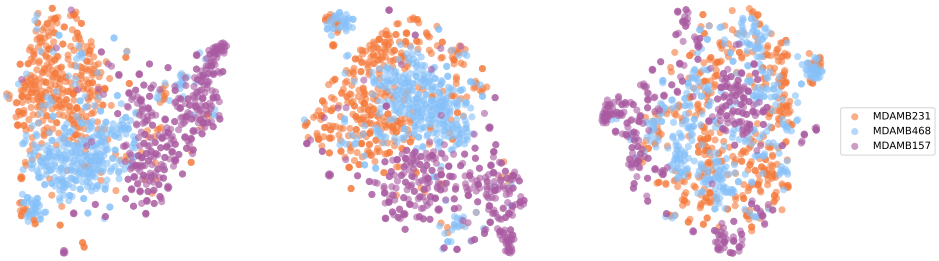


Figure 4.6: t-SNE embeddings of encodings from handcrafted features (left), autoencoder (center) and domain-adversarial autoencoder (right), with cell lines distinguished by colour. Respective silhouette scores of 0.22 and 0.14 and -0.02 confirm the reduced divergence in the adapted domains.

As stated above, this new dataset was not a drug screen and therefore we could not evaluate our model in the same way as before. We were, however, able to pretrain our model on this morphological screen and transfer it to our drug screen to be used as a feature encoder directly. Compared with an equivalent vanilla autoencoder our model performed marginally better for three cell lines (32.08 compared with 31.17) ($W = 366, p < 0.1$), following our evaluation strategy. Though not the exact intended application of our model, we again see an improvement over baselines, suggesting an aptitude of our method for analysing multi-cell-line data. It will be the subject of future work to test our method on a full drug screen in greater numbers of cell lines.

4.4 Discussion

In this chapter we address prediction of mechanism of action (MOA) at a molecular level. Importantly, we have not optimised the MOA classes with respect to the readout of the screen, as is common in many benchmarking studies. We have studied a number of different approaches, including traditional approaches based on hand-crafted features, and deep learning approaches, allowing us to learn suitable representations.

A major gain can be achieved by using multiple cell lines, but the choice of algorithm is important to most benefit from the data heterogeneity. We investigated several approaches, and obtained the best results for an autoencoder with a domain discriminative component to promote domain-invariant features across multiple cell lines. This approach requires the same set of markers to be used and ideally the same set of drugs to be tested.

In addition of improving MOA prediction accuracy, this method further produces a representation that allows us to compare effects of drugs on different cell lines. We use the representation in order to make comparisons between (drug, cell line) pairs. This is one of the most important use cases if the cell lines represent different molecular subtypes of a disease. Importantly, it allows one to identify highly specific drugs that only act on one particular subtype–the paradigm of precision medicine–and to distinguish them from drugs that are generally effective across different subtypes, as well as from drugs that lead to different phenotypic effects, which in turn suggest a target of different pathways depending on the transcriptional program.

While these approaches have only been applied to a small-scale pilot study, they provide an interesting starting point for larger multi-cell-line screens. A larger screen would, of course, provide greater opportunities to explore the strengths and weaknesses of our proposed method. It may be, for example, that our method is again advantageous for higher numbers of cell lines, yet it may happen that in the limit, performance converges to that of simpler methods, such as the vanilla autoencoders, as the variety of cell lines forces the autoencoder to learn domain-invariant features as a matter of course. In this case, our method will have provided a useful regularisation in the use case of lower numbers of cell lines.

With the exception of batch normalisation, we have neglected to borrow tricks of the trade from the GAN literature on adversarial learning. It is possible that our reported results underestimate the quality of our method as result. A future study could include an assessment of the effects of varying activation function (for example, LeakyReLU), alternating gradient descent steps, and learning rate scheduling, to name a few.

Finally, the domain-invariant features themselves remain mysterious. It

would be fitting to analyse the properties of those arising from our experiments and understand how they could be improved. Consider Information Maximising GANs (InfoGANs) (Chen et al. [2016]), a GAN with an information maximising regularisation function. InfoGANs allocate a subset c of the generator input noise vector z to act as a latent code, which is optimised to have maximum mutual information with the generator output. That is, one maximises $I(c; G(z, c))$. In practice, the discriminator predicts the distribution on c used in generation, given the generated image. If the discriminator is easily able to discern c given generated image x, then the mutual information is high, and the content of c has been preserved during generation. The only way to maximise the mutual information is, for each latent variable c_i, for the generator and discriminator to "agree" on a salient and independently-varying property of image objects (such as size and rotation), such that the auxiliary network can infer the original, independently-varying latent variable. In principle, the outcome is a generator that will generate images that are to some extent controllable by semantically meaningful properties. Information maximising is therefore a powerful prior for learning disentangled representations. Some similar mechanism could be attempted in our autoencoders to address the interpretability of the latent codes.

Part II

Chimeric antigen receptor T-cell therapy

Chapter 5

Experimentally-generated ground truth for detecting cell types in an image-based immunotherapy screen

Summary: *Chimeric antigen receptor (CAR) is an immunotherapy whereby T lymphocytes are engineered to selectively attack cancer cells. Image-based screens of CAR-T cells, combining phase contrast and fluorescence microscopy, suffer from the gradual quenching of the fluorescent signal, making the reliable tracking of cell populations across time-lapse movies difficult. We propose to leverage the available fluorescent markers as an experimentally-generated ground truth for phenotyping the cell population in time. From there we compare two learning strategies. The first, based on predicting fluorescent markers directly from phase contrast microscopy is, in the first instance, a powerful visualisation system. The second, an object detection system learns from an automatically annotated training set, acquiring with some simple image processing of the image set. Depending on the experimental objectives, either approach has scope for potentially eliminating the need for the cumbersome fluorescent markers. This approach will underpin the development of cheap and robust microscope-based protocols to quantify CAR-T activity against tumor cell in vitro.*

Résumé: *Ce chapitre...*

5.1 Overview

Chimeric antigen receptor T-cell (CAR-T) therapy is an immunotherapy whereby T lymphocytes (a subtype of white blood cells) are engineered to selectively attack cancer cells. Generally speaking, CARs are engineered or *recombinant* receptors designed to target a specific protein, in practice a tumour antigen. When a CAR allows the CAR-T cell to latch onto an abnormal antigen, and deliver cytotoxic chemicals to induce *lysis* (membrane breakdown) of the target cell (Benmebarek et al. [2019]). CAR technology has 30 years of development encompassing several design generations (Maude et al. [2015]). As a therapy, T cells are extracted from a patient or healthy donor, modified (such as with viral transduction) to express the CAR in culture, and infused back into the patient. The *in vivo* CAR-T cells then target tumours as a "living drug" treatment. The main engineering challenge is ensuring safety and efficacy, in particular the target specificity of the T cells (Sadelain et al. [2013]). CD19[1], a B-cell surface protein is expressed by almost all B-cell cancers, such as acute lymphoblastic leukemia, one of the most common and fatal forms (primarily from relapse) of pediatric cancer worldwide (Hunger and Mullighan [2015]). In the latter disease, CD19 CAR-T has achieved up to 90% complete remission rates in clinical studies such as Maude et al. [2014]. Ideally, the antigen will be cancer-cell-specific, however CD19 CAR-T leads to B-cell *aplasia*, that is, the depletion of B cells both cancerous and normal. Nevertheless, such side effects have been shown to be manageable (Bonifant et al. [2016]). The pursuit of new forms of CAR is the subject of rapid and enthusiastic research (Wang et al. [2017]).

In a microscope setting where both transmitted light and fluorescence microscopy can be taken for the same cells simultaneously, interesting opportunities arise. Recently, successful attempts have been made (Christiansen et al. [2018], Ounkomol et al. [2018]) to predict fluorescent signals from transmitted light images, demonstrating that for certain biological experiments, the relevant information is wholly contained in the phase contrast signal. This is an attractive prospect because fluorescence microscopy, despite its power, has various drawbacks, with several experimental complications (such as fading dyes), in particular when imaging assays are performed over several days. In addition, the fluorescent marking of cells is expensive, time-consuming, and potentially invasive to the experiment. In this Chapter, we aim to leverage the fluorescent markers to quantify the cultured cells from the phase contrast alone. We compare two approaches to this quantification: the first, based on fluorescent labeling, is described in Section 5.4; the second, an object detection system, in Section 5.4. In the part of the

[1]Not to be confused with COVID-19.

Chapter, we compare the two approaches.

5.1.1 CAR-T dataset

We obtained a set of time-lapse microscopy images from CAR-T experiments performed on an IncuCyte machine. In these experiments, the disease is modelled by Raji, an immortalised cell line of B lymphocytes from a 1963 Burkitt's lymphoma patient. We study Raji cell populations in isolation, as well as cocultured with CAR-T cells. The setup is detailed in Table 5.1. The row A wells study isolated Raji cells; the row B wells study the coculture. Each well is imaged at 5× magnification in four fields of view of size 1408 × 1040 pixels. Images were taken every two hours over a 220 hour period (110 frames apiece) producing a time lapse movie. Each row of the microplate studied consists of two groups of two replicates, where each group has a different seeding quota. In our analysis, however, we consider the groups to be interchangeable and, where convenient, we pool data within each plate row. Each frame of each time lapse movie pairs a phase contrast image with green fluorescent protein (GFP) and mCherry fluorescent images, depicting the same scene. A sample is given in Figure ??. The mCherry fluorescence is present in all Raji cells while the GFP only appears in dead cells. The T cells, where present are only visible on the phase contrast channel. Thus, fluorescent markers combine for a quasi-annotation of the cells, which we may express with the (fuzzy) logic,

$$\begin{aligned} \neg\text{object} &\implies \text{background} \\ \text{object} \wedge \text{GFP} &\implies \text{dead Raji} \\ \text{object} \wedge \text{mCherry} \wedge \neg\text{GFP} &\implies \text{Raji} \\ \text{object} \wedge \neg\text{mCherry} \wedge \neg\text{GFP} &\implies \text{CAR-T} \end{aligned} \tag{5.1}$$

Note the high volume of data: the cells are seeded to a total of up to 60000 per well, with four wells per experiment type, each with four fields and 110 time lapse frames. Coupled with the fluorescence annotations, this is, in principle, a veritable treasure trove of data for deep learning. Annotation by experiment is a promising strategy: we can easily collect a large ground truth and, in addition, the "experimental ground truth" is much more objective than a manual one. A similar strategy has already been applied to image segmentation (Sadanandan et al. [2017]). This seemingly overcomes the main bottleneck in leveraging deep learning models. However, this abundance of training data comes at a cost in resolution: most cells fit inside a 14 × 14 pixel window (contrast this with the much higher resolutions of

	1	2	5	6
A	raji-target cells (1) 30K cells / well		raji-target cells (1) 30K cells / well	
B	CAR June 1:2 (1) 30K cells / well raji-target cells (1) 30K cells / well		CAR June 1:2 (1) 15K cells / well raji-target cells (1) 30K cells / well	

Table 5.1: Characteristics of CAR-T experiments studied. The . Row A studies RAJI cells in isolation; row B studies cocultured Raji and CAR-T cells.

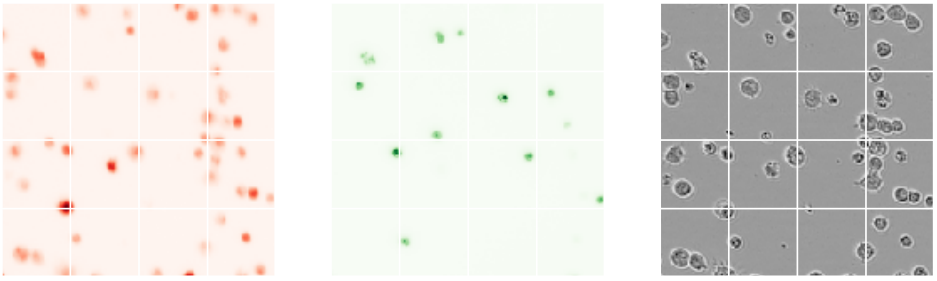


Figure 5.1: Aligned image channel crops (200 × 200px) marking living Raji cells in mCherry (left), dead cells in GFP (center), and phase contrast (right).

our cells from Part I). This low resolution proves to be a constraint in the analyses to come, and an engineering challenge for our models.

5.1.2 Experimental phenomena

Even at the best of times it is highly advisable to perform an exploratory analysis of a dataset. This principle served us well in Chapter 3, as we obtained important insights that we carried through to Chapter 4. With the temporal dimension now thrown in, the CAR-T dataset proves to be highly dynamic, and becomes increasingly chaotic in time, in particular as the Raji B-cells undergo mitosis. We presently detail some notable phenomena observable in our experimental data. These prove to be influential factors in our methodological designs.

Apoptotic cells acquire GFP

In Figure A.4 we trace the death of a Raji cell in full fluorescence. One may observe some subtle morphological changes such as a reduction in size, as well as a change of texture, which remain visible on the phase contrast channel. The fluorescence, however, shows a clear accumulation of GFP fluorophores. We measure the average intensity of each fluorescent channel in the cell region of interest and plot these over time in Figure 5.2. The GFP signal rises for at least 10 frames, corresponding to a 20 hour period. The mCherry signal declines in time, seemingly in accordance with the quenching effect. We will see this latter phenomenon repeated elsewhere.

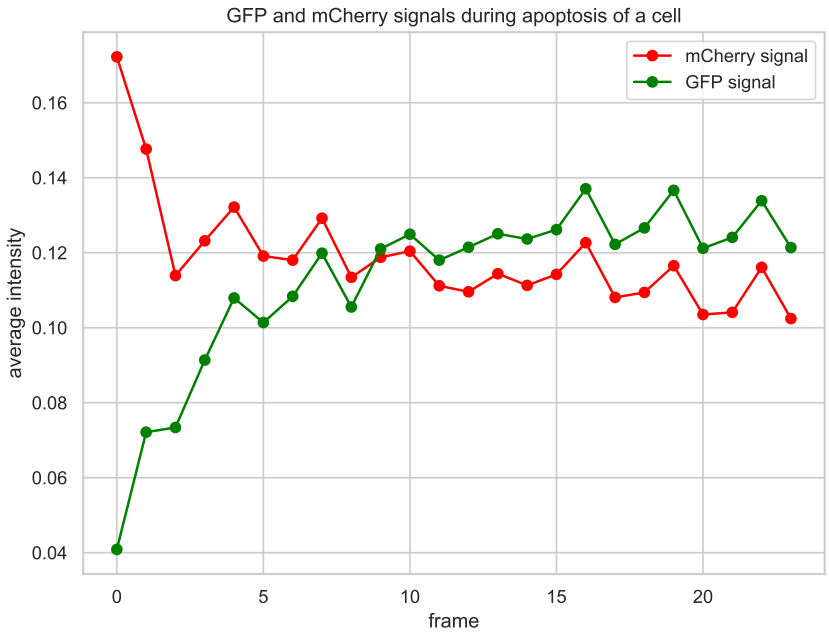
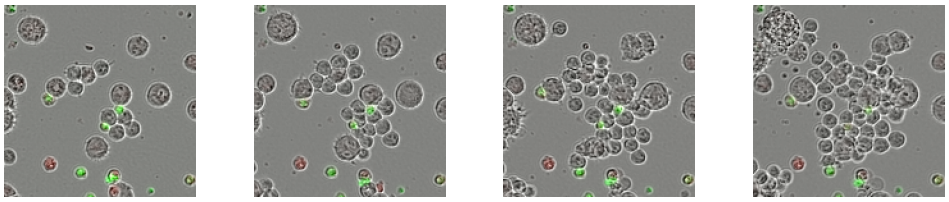


Figure 5.2: Tracking a mitotic event. The fluorescent turns green as a B cell undergoes morphological changes (a)-(f). This is reflected in a plot of intensity profiles (g).

Raji mitosis creates cell clumps

The phenomenon of mitosis or cell division greatly complicates the task of counting cells, and influences our methods in the current chapter as well as the next. It is well known that Raji cells group to form clusters or "clumps" (Epstein et al. [1966]). Figure 5.3 shows a sequence of cropped

frames centered on a significant mitosis event. From a handful of dispersed cells in the initial frame ($t = 50$), cell division occurs in time ($t = 70$), ($t = 80$), ($t = 90$), creating clusters of closely touching, or even overlapping, cells. Whatever methodological approach we take, this complexifies the detection and segmentation individual cells as we can no longer utilise the image background to these ends. Whereas at first the cell contours are well-defined, the more the cells multiply, the less clear their separation, as cells begin to overlap.



(a) $t = 50$ (b) $t = 70$ (c) $t = 80$ (d) $t = 90$

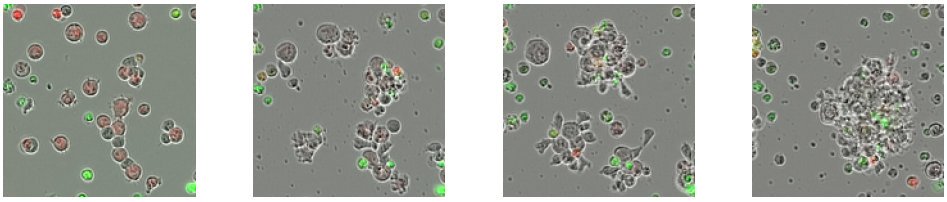
Figure 5.3: Chronicling a mitotic event. Cells accumulate in clusters due to mitosis.

CAR-T cells create clusters of lysed target cells

With the inclusion of T cells, the picture becomes even more complicated. In Figure 5.4 we show a series of crops taken at 30 hour intervals from well $B6$. T cells can now be seen as non-fluorescent objects with an elongated morphology. Their activities involve the killing of B cells, the lysed (fragmented) remains of which are shepherded into tight clusters of cellular matter. The formation and ultimate merging of several such clusters is observable in Figure 5.4. We also give an example of a CAR-T cell latching onto a Raji antigen, with the subsequent death of the Raji cell in Appendix A.3. From a modeling perspective, individual cells are impossible to discern from such entropic clusters, as the membrane that provides each cell with its identity has been lost. We offer perspectives for handling these cases in Section 5.4.

5.1.3 Coping with fluorescent quenching

Image-based screens of CAR-T cells, combining phase contrast and fluorescence microscopy, suffer from the gradual quenching of the fluorescent signal, making the reliable tracking of cell populations across time-lapse imagery difficult. We saw throughout Section 5.1.2 the diminished mCherry (red) fluorescent signal in latter time steps. In Figure 5.5 we see how the distribution



(a) $t = 0$ (b) $t = 30$ (c) $t = 60$ (d) $t = 90$

Figure 5.4: CAR-T cells (devoid of fluorescence) attack Raji B cells by latching onto Raji cell surface antigens and delivering cytotoxic chemicals. The induced lysis of the target Raji cells yields growing clusters of cellular matter.

of total image fluorescence diminishes across time, regardless of well. Already, the quenching effect obfuscates the interpretation of the experiments by lab technicians. It is furthermore a complication for learning algorithms, as fluorescence becomes inversely correlated with the various cellular phenomena occurring in the latter stages of the experiment, for example: the dissemination of large cells due to cell growth, and the appearance of cell clusters due to mitosis, as illustrated above. Learning only on earlier frames, where the fluorescence is strongest, is unlikely to suffice. To mitigate the quenching effect, we normalise by,

$$x' = \min(1, \frac{\bar{x_0}}{\bar{x}} \cdot \frac{x - \min(x)}{\text{perc}_{99}(x) - \min(x)}) \tag{5.2}$$

where $\hat{x}_0$ and $\hat{x}$ are the means of the initial image and the image to be normalised, and perc_{99} returns the 99th percentile intensity. We then threshold by some small value τ to remove background noise.

5.2 Fluorescent labeling

In biological experiments pairing transmitted light and fluorescent signals, the fluorescence may often be largely inferred from the transmitted light image alone, as in Christiansen et al. [2018], Ounkomol et al. [2018], and Lee et al. [2018]. That is, given the pair X, Y of transmitted light and fluorescent image, the fluorescent labeler F learns a mapping such that,

$$F(X) \approx Y \tag{5.3}$$

Note that the prospects for *fluorescent labeling* will always depend on, among other things, the available image resolution. While Ounkomol et al. [2018]

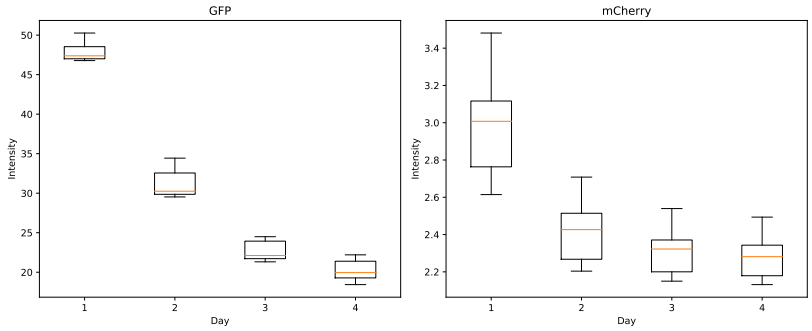


Figure 5.5: Comparison of average GFP (left) and mCherry (right) fluorescence measured across at different fields of view and compared 24 hour intervals. One may observe the quenching effect of fluorescence over time, which occurs most rapidly in the first increment.

predicted fluorescent images of nucleoli, nuclear envelope, and microtubules from transmitted light images with a median Pearson correlation coefficient $r \geq 0.8$, "painting" other organelles such as desmosomes and Golgi apparatus worked far less well $r \leq 0.2$. In our experiments the relatively low resolution of cells must therefore by compensated by high-level morphological cues associating phenotype with fluorescence.

Here we present a family of models for performing fluorescence labeling of our phase contrast images automatically. The motivations of the system are twofold: firstly, it can be used as a visualisation tool; secondly, it may serve as an intermediate step in a cell counting system. Naturally, the quality of the latter function depends on that of the former.

5.2.1 Image-to-image translation models

Though not the only fluorescence labeler in the literature, we posit the "F-Net", proposed by Ounkomol et al. [2018], to be the proper baseline for our problem, on the grounds that it is essentially a repurposed U-Net (Ronneberger et al. [2015]), a widely studied architecture. This architecture has a deep encoder-decoder structure, such as the autoencoders studied in Section , albeit fully convolutional, and with *skip connections* concatenating each upsampling layer with its opposite downsampling layers. We give provide the full architecture specification in Appendix C.3. The network is trained against the objective function,

$$\min_F \mathcal{L}_{L_2}(F) = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[||\mathbf{y} - F(\mathbf{x})||_2^2] \tag{5.4}$$

that is, to minimise the expectation of mean square error over the distribution of input images $\mathbf{x}$ and target images $\mathbf{y}$, where $F : \mathbf{x} \rightarrow \mathbf{y}$ is the labeler network parameterised by a set of weights θ. In our use cases, the input images are the phase contrast images, and the target images are the fluorescent channels. By default, we train the network to predict a single fluorescent channel (as in Ounkomol et al. [2018]), although evidently, one could train a multi-task labeler to predict a multiplexed fluorescent output. Our experiments with this approach did not yield a significant improvement, however.

The F-Net approach is more or less generalised by the image-to-image translation models proposed in Isola et al. [2017]. Thus, we formulate the family of `pix2pix` models,

$$\min_G \max_D \mathcal{L}_{PIX}(D_{patch}, G) = \mathcal{L}_{cGAN}(D_{patch}, G) + \lambda \mathcal{L}_1(G) \tag{5.5}$$

where $\mathcal{L}_1(G) = \mathbb{E}[||G(\mathbf{x}) - \mathbf{y}||_1]$ is mean absolute error with tuning hyperparameter λ and $\mathcal{L}_{cGAN}$ is the conditional GAN (cGAN) objective,

$$\mathcal{L}_{cGAN}(D, G) = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})}[\log D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\mathbf{y})))] \tag{5.6}$$

where D is the discriminator and G is the generator, equivalent to F in all ways aside from it using a hyperbolic tangent (tanh) activation function, as is conventional for DCGANs, rather than a linear one. The variable $\mathbf{z}$ refers to the generator's noise input. In the image-to-image setting, the discriminator network is a fully-convolutional network denoted "PatchGAN" (see Appendix C.4),

$$d_{patch} : X, Y \rightarrow [0, 1]^{h \times w} \tag{5.7}$$

producing an $h \times w$ grid of outputs rather than a single neuron, with each grid element critiquing an overlapping region of the input image. The grid of outputs are then averaged to give the discriminator,

$$D_{patch}(\mathbf{x}, \mathbf{y}) = \frac{1}{hw} \sum_{i=0}^{h} \sum_{j=0}^{w} d_{patch}(\mathbf{x}, \mathbf{y})_{ij} \tag{5.8}$$

In practice, the conditioning means inputs are concatenated channel-wise. Isola et al. [2017] make the extremely insightful observation, however, that a GAN learns its own loss function via the discriminator, which penalises "unrealistic" images. As such, it is an infinitely flexible methodology and

the discriminator D can be understood to act as an adaptive loss function, analagous to the way features are learned and optimised for the task of classification within the layers of a deep classifier. The choice of L_1 over an L_2 auxiliary loss is based on the empirical observation that L_2 leads to blurry results, something we observe presently. The `pix2pix` system has been demonstrated to be effective on a myriad of image-to-image translation problems. Just as with U-Net, they can achieve impressive results on a relatively small number of images.

In addition to the above, we also train the F-Net F against a mean absolute error loss,

$$\min_F \mathcal{L}_{L_1}(F) = \lambda \mathcal{L}_1(F) \tag{5.9}$$

We trained all models (Equations 5.4, 5.5, and 5.9) on 1120 non-overlapping 256×256px phase contrast crops selected from the first 14 fields of view from the Raji-only experiments (row A of the plate). The cropping strategy sampled from the first frame of each 24-hour window, for the first 96 hours of each time lapse movie. This strategy was adopted to maximise the available variation in the training data, without losing too much to the fluorescent quenching effect of later frames, while at the same time retaining the convenience of keeping all images in memory. Validation and testing was performed according to the same strategy on independent fields A06_03 and A06_04. Batches of size 1 were sampled randomly and all models were trained with the Adam optimiser with $(\beta_0, \beta_1) = (0.5, 0.999)$ and maximum learning rate 2×10^{-4}.

5.2.2 Results

In practice, a fluorescence prediction can be considered accurate if the placement of its predictions are accurate, even if it does not match the ground truth intensities accurately. For this reason Ounkomol et al. [2018] propose evaluating based on the Pearson correlation coefficient (PCC),

$$r = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sqrt{\sum(x - \bar{x})^2 \sum(y - \bar{y})^2}} \tag{5.10}$$

over all pixels x in the prediction and y in the test image. We compare the average PCC correlation across all test images for each of the three models and each mode (GFP, mCherry) (see Figure 5.6). The best overall performance comes from the L_2 model with mean 0.7617 and standard deviation 0.1049 for mCherry labeling, and mean 0.6163 and standard deviation 0.0931

for GFP. The pix2pix system performs at mean 0.7573 and standard deviation 0.0814 for mCherry, and mean 0.5848 and standard deviation 0.0972 for GFP. Finally, the L_1 models performs at mean 0.7720 and standard deviation 0.0727 for mCherry, and 0.5115 and standard deviation 0.1099 for GFP. Paradoxically, the L_2 model appears, by inspection, to generate worse results, with a significant amount of fluorescent "spillage" and blur effects. However, this noise can be removed with a simple thresholding operation. The pix2pix outputs look immediately sharper, but ultimately fail more often to apply fluorescence where it counts, leading to more false negatives. On the other hand, it is also clear that the L_2 and L_1 models exhibit a greater degree of mode collapse than the pix2pix model, which necessarily varies its fluorescent labeling to continuously fool the discriminator.

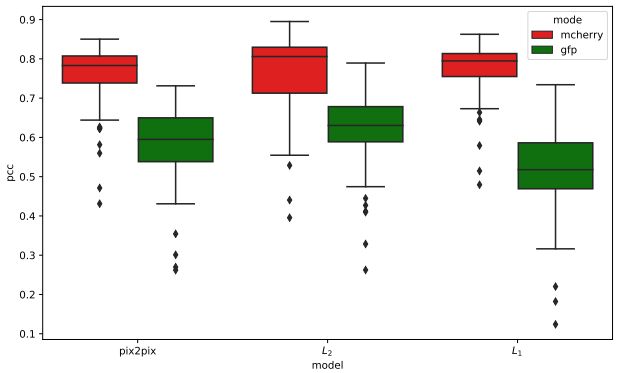


Figure 5.6: Pearson correlation coefficients for outputs of three fluorescent labelers, for both mCherry and GFP fluorescence prediction, measured over 80 test images.

Full fluorescence outputs

An important qualitative test is to visualise the joint model outputs in full fluorescent colour, as well as in time lapse. To produce an appealing output, we first correct for a "camera shake" effect of the plate moving inside the microscope. Between successive frames, this is usually no more than $10\mu m$, corresponding to about 7px, in any direction. We therefore find the necessary correction $(\delta x, \delta y)$ frame by frame by brute force over a suitable range of offsets. Fortunately, the Raji cells do not move much over time, and we can therefore find the right offset as the one that minimises the squared error between the frames. The images are each cropped slightly to account for the extents of the cumulative displacements in each direction caused by the shake. This tends to be a small amount, apparently with the camera displacements occurring around a fixed center of mass. Lastly, we correct

the frames of the fluorescent channels by the same offsets discovered in the phase contrast.

We set RGB pixel at position (i, j) of each full fluorescence image I as,

$$I_{ij} = [M_{ij} + P_{ij}, G_{ij} + P_{ij}, P_{ij}] \tag{5.11}$$

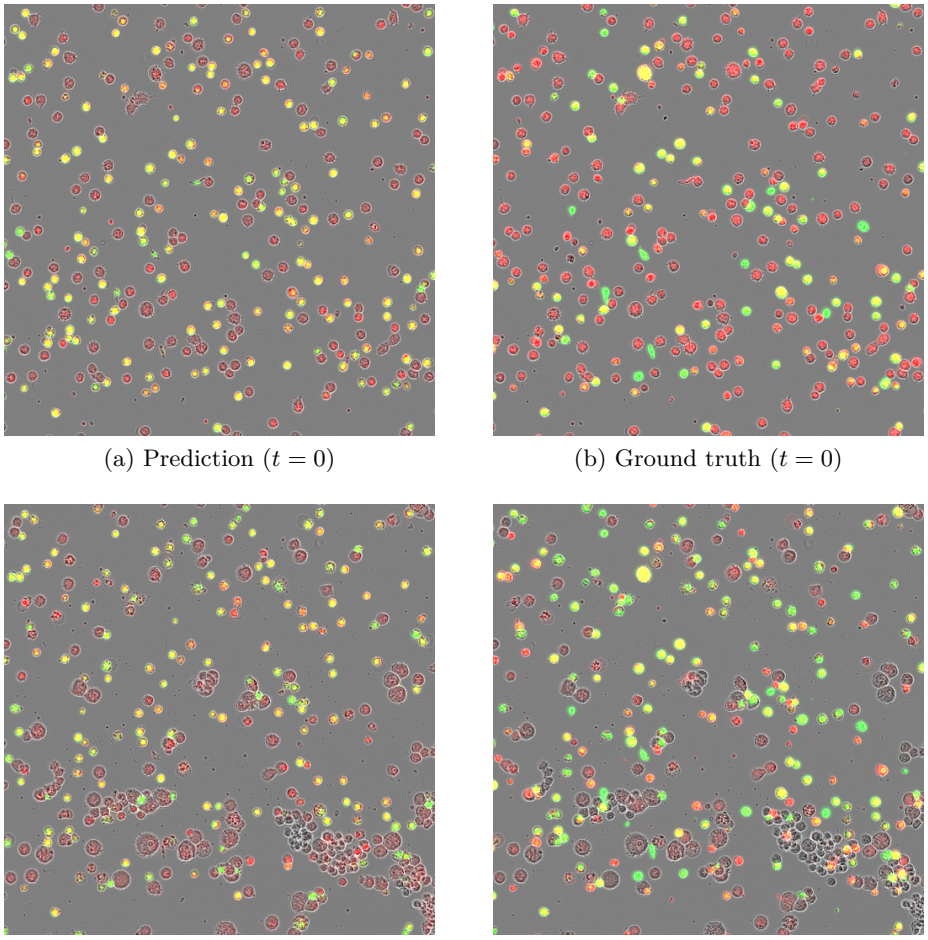
where M is the mCherry channel, G is the GFP channel, and P is the phase contrast image. The values are further clipped to the range $[0, 1]$. In Figures 5.7 and 5.8 we compare synthesised images combined from the respective fluorescent labelers to the corresponding ground truth fluorescence. The errors are manifest, yet the `pix2pix` models do a fine job of discerning cell types, implicit in its application of GFP over dead Raji cells in the Raji-only experiment (Figure 5.7) and its leaving T cells unlabelled in the full CAR-T experiment 5.8, as per the logic of Equation 5.1. The predicted fluorescence intensity levels also closely approximate those of the ground truth. Nevertheless, it should be noted that the the mutual information between fluorescence intensity and phase contrast morphology may ultimately be too small to always guarantee an accurate prediction of intensity. For this reason, we must accept, for example, that certain dead cells appear more yellow than green, and vice versa. One possible extension of our work would be to incorporate time information into the prediction, which might alleviate this effect. Ultimately, however, we can confirm that for the broader picture, the fluorescent labeling is a success as a visualisation tool, at least for the earlier frames of the video, and we should ought now to consider how to incorporate it into a practicable tool for the experimental setting.

We release full 110-frame time lapse movies comparing prediction (left) and ground truth full fluorescence (right) online: Raji-only 256×256px[2], Raji-only 512×512px[3], full CAR-T 256×256px[4], and full CAR-T 512×512px[5].

5.2.3 Bridging the gap to cell detection

In our problem context, we would like to derive a quantitative profile from live cell imaging data. While effective as a visualisation tool, fluorescent labeling only goes partway towards a full quantification of the phase contrast contents. Consider sample outputs for densely clustering cells, given in Figure 5.9. Here we can see how the task of counting cells is far from

[2]https://jcboyd.github.io/assets/car-t-videos/raji_target/256.mp4
[3]https://jcboyd.github.io/assets/car-t-videos/raji_target/512.mp4
[4]https://jcboyd.github.io/assets/car-t-videos/CAR_June/256.mp4
[5]https://jcboyd.github.io/assets/car-t-videos/CAR_June/512.mp4



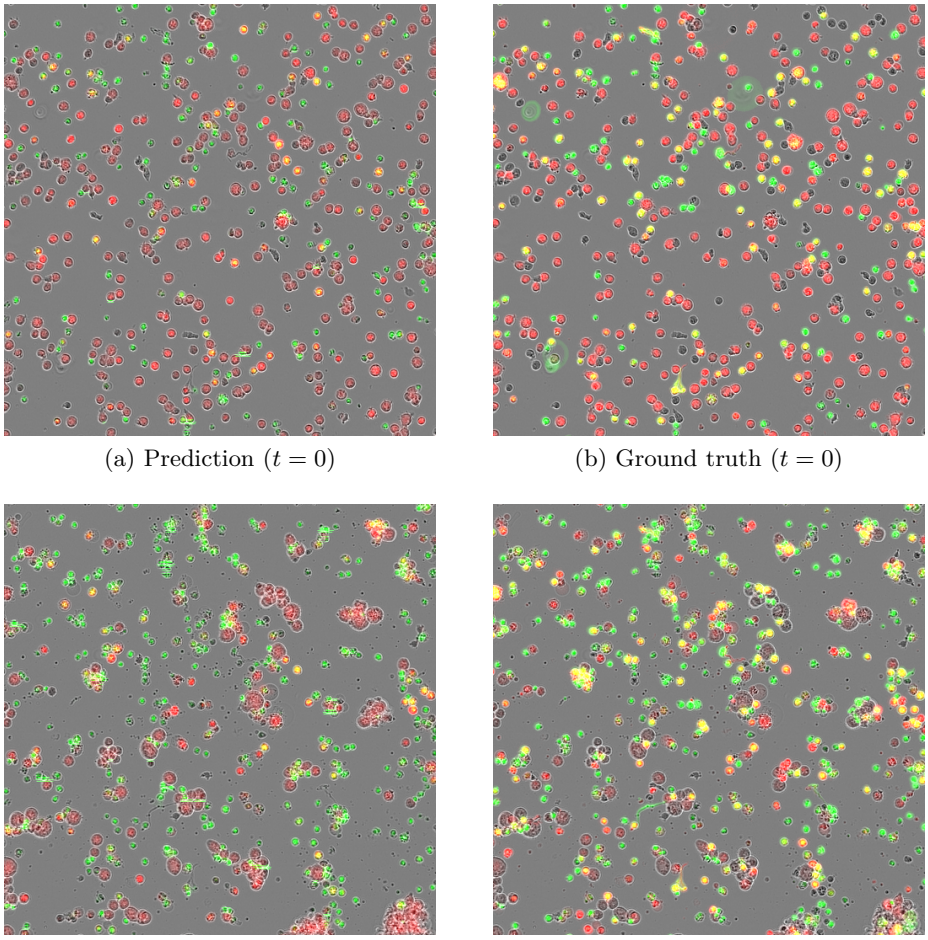
(a) Prediction ($t = 0$)

(b) Ground truth ($t = 0$)

(c) Prediction ($t = 48$)

(d) Ground truth ($t = 48$)

Figure 5.7: Full fluorescence for indicative (512×512px) crop from an Raji-only experiment. Columns distinguish fluorescent labeler predictions (left) and ground truth (right); rows distinguish times an early frame ($t = 0$) (top) and a later one ($t = 48$) (bottom).



(a) Prediction ($t = 0$)

(b) Ground truth ($t = 0$)

(c) Prediction ($t = 40$)

(d) Ground truth ($t = 40$)

Figure 5.8: Full fluorescence for indicative (512×512px) crop from a CAR-T experiment. Columns distinguish fluorescent labeler predictions (left) and ground truth (right); rows distinguish times an early frame ($t = 0$) (top) and a later one ($t = 40$) (bottom).

over following fluorescent labeling, with dense "clouds" of fluorescence corresponding to masses of cells. One can imagine all sorts of *post hoc* solutions for disambiguating these fluorescent clouds.

Blob detection for disambiguating fluorescent labels

As Raji cells have a consistent size and simply, circular geometry, one approach to counting individual cells from the fluorescent label outputs is with *blob detection*. A simple approach to blob detection is with a Laplacian of Gaussian (LoG) filter. That is,

$$\Delta G_\sigma(x,y) * f(x,y) = [\nabla_{xx} G_\sigma(x,y) + \nabla_{yy} G_\sigma(x,y)] * f(x,y)$$

where Δ is the Laplacian operator (sum of second partial derivatives), G_σ is a Gaussian filter with standard deviation σ and f is an image. Blobs distribute intensities in a Gaussian-like way, and the Gaussian pre-filter smooths the image. The second derivative of a Gaussian has a minimum in the center of bell curve, thus marking the center of the blob. The maxima detected in the resultant image are the detected blobs. Usually the filter is performed over a range of scales, with the maxima taken over the z-stack.

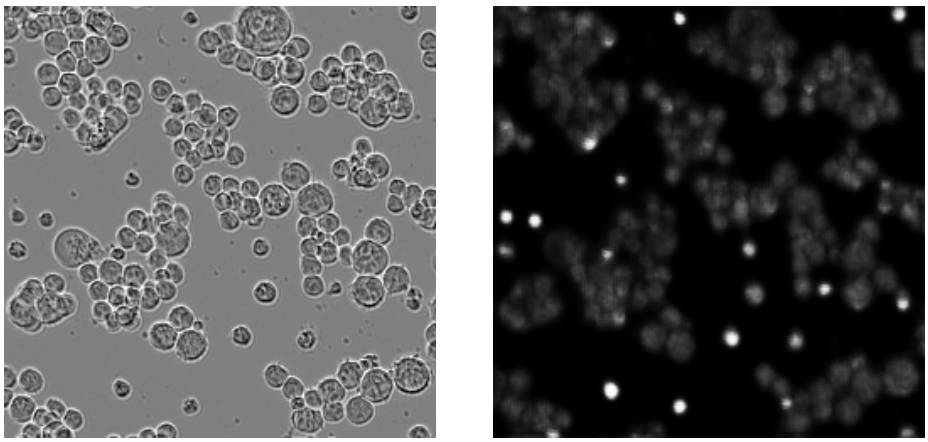
In practice, the tunable parameters for LoG blob detection are the range of standard deviations $\{\sigma_i\}$ to test for (corresponding to the anticipated size range) and a threshold θ on the minimum permissible intensity. These we tune by hand. We provide an example of this approach to blob detection of fluorescent outputs in Figure A.5.

We find, however, that this simple approach to blob detection does not suffice for cases such as in in Figure 5.9. While some system could surely be devised to infer the number of cells from a mass of flouroscence, in the following section we study an alternative approach based on the fundamentally distinct methodology of object detection, which makes counting cells a primary objective.

5.3 Object detection system

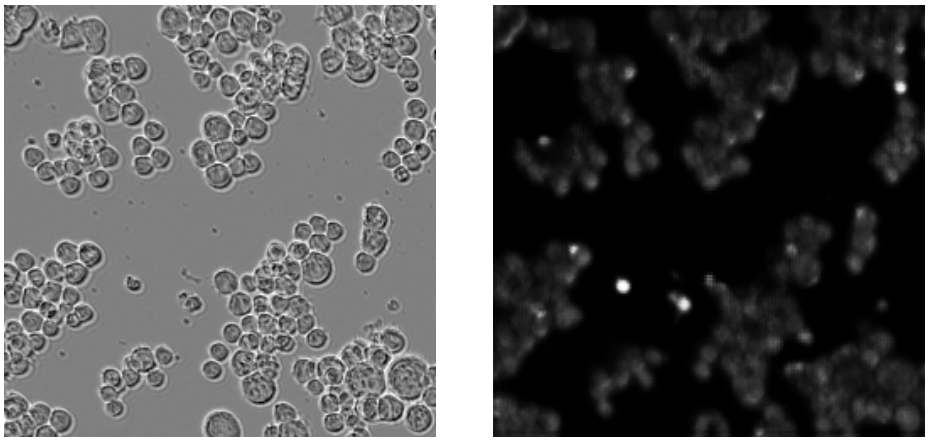
This section contains an extended version of a paper published in at the International Symposium for Biomedical Imaging in 2020.

We found in the previous section a limitation on the scope of fluorescent labeling for counting cells. This brings us back to a more conventional high content analysis (HCA). Whereas in Chapter 4 we were concerned with generating a phenotypic profile that characterised a drug effect effectively,



(a) Phase contrast input

(b) mCherry fluorescent output



(c) Phase contrast input

(d) mCherry fluorescent output

Figure 5.9: Fluorescent labeler outputs produce dense clouds of mCherry fluorescence for two manually selected phase contrast inputs. These outputs may be difficult to disambiguate.

regardless of how obscure the meaning of the profile elements. In the present context, we are concerned with the more comprehensible task of counting cell types in time. Note that this task conforms to the conventional pipeline (see Section **??**). In particular, the dimensionality reduction step will be fulfilled by the attribution of a cell type,

$$\mathbf{z} = f(\mathbf{x}) \tag{5.12}$$

where $\mathbf{z} \in \{0,1\}^K$ and $\sum_{k=0}^{K} z_k = 1$ for K the number of cell classes, that is, $\mathbf{z}$ is one-hot, for extracted cell features $\mathbf{x}$, and cell classifier f. The cell population is then summarised by aggregation as a simple summation or average, giving the number or proportion of cells per cell class respectively in the phenotypic profile,

$$\mathbf{p} = \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}_i) \tag{5.13}$$

As we shall see, the first three stages of the HCA pipeline are achieved by a single neural network.

A phenotypic profile of this nature summarises a single image frame. However, we are interested in summarising a full time lapse movie. Therefore, our aim is to create a set of such count profiles,

$$\mathbb{P} = \{\mathbf{p}_i\}_{i=0}^{T}, \tag{5.14}$$

where $\mathbf{p}_i \in \mathbb{R}^K$ for K classes is the ith profile in a series of T frames. This ordered set amounts to a set of K time series, one per class, of length T, and is our final output in Section 5.3.3.

In Section 6.6.1 we describe an acquisition and preprocessing pipeline for an experimentally-generated object detection ground truth with which to train our object system. In Section 5.3.2 we specify this system, and in Section 5.3.3 we describe our evaluation methodology and report model performance on two manually annotated datasets. In this Section we consider only the Raji-only experimental setting. Note that our methodology should naturally extend to other the full CAR-T setting also, something we intend to address in future work.

5.3.1 Experimentally-generated ground truth

Our pipeline (see Figure 5.10) begins by applying a Gaussian filter (tuned to $\sigma = 2$) to the phase contrast image. We then segment cells by subtracting

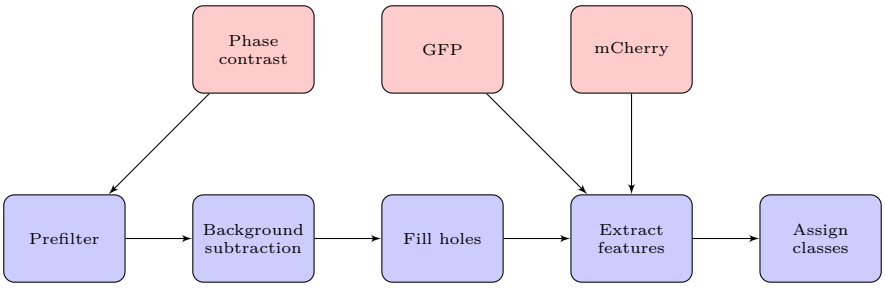


Figure 5.10: Pipeline for automatic construction of ground truth for training object detection system. Basic image processing steps indicated in blue; image inputs indicated in red.

a background image formed with a mean filter of diameter 19px, before clipping to zero as in Walter et al. [2010a]. We fill object holes with a morphological reconstruction by erosion and use a morphological opening to remove small details. An example of this pipeline depicted in Figure A.6. We further filter objects outside a reasonable size range ($< 6 \times 6$px $\approx 50\mu m^2$, determined by ranking cell areas), as these tend to be dust and other non-cellular particles on the well surface. An Otsu threshold on the distribution of averaged GFP signal per cell is then used to allocate a class (living/dead) for each connected component (see Figure 5.11. To train our object detector (Section 5.3.2), we also randomly sample background crops from the images, allowing for partial overlaps with cells.

5.3.2 Object detection system

In order to track cell phenotype populations over time, we require a robust object detection system to identify individual cells. The core of our system is a convolutional neural network and is detailed below.

Training as a classifier

Our preprocessing pipeline is imperfect and does not give a complete annotation of the cell populations as would be required by state-of-the-art detection systems such as Redmon et al. [2016]. We therefore opt for crop-wise training, where the bounding boxes of successfully segmented cells are padded, to create 24×24px crops, centered on the cells. Due to the low image resolution, we found this sizing provided sufficient contextual information to the network. Combined with background crops, this amounts to approximately 100,000 training examples in three classes. Samples are given in Figure 5.13.

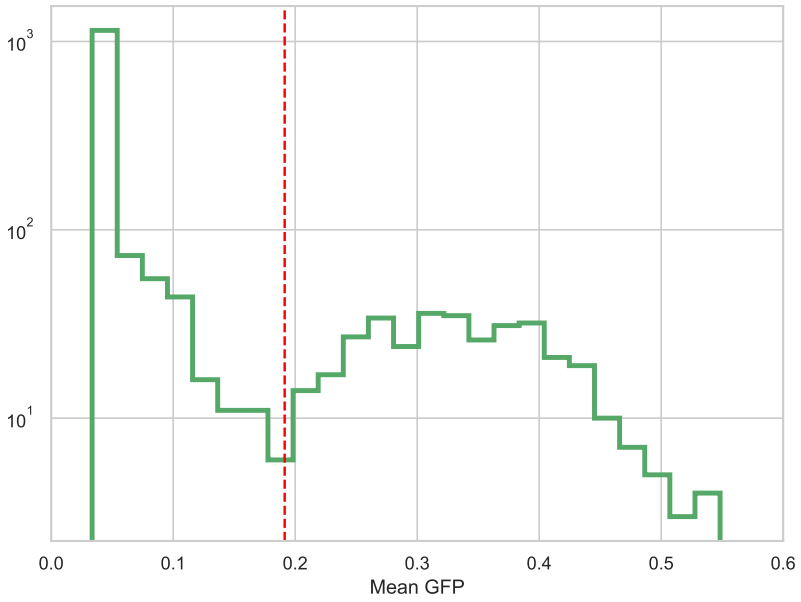


Figure 5.11: Living and dead Raji cells revealed as distinct modes of a bimodal distribution on mean GFP fluorescence intensity per connected component of cell segmentation output.

Our network architecture is detailed in Table 5.2. All weighted layers have a ReLU activation, except Output_o and Output_c, which have softmax activations, and Output_b, which remains linear. The convolutions are all valid, and a 24×24px input image is reduced to 1×1px by the final layer. We implement this network in the `Keras` deep learning frameworkChollet et al. [2015] and all code for our system is publicly available[6]. We train the network with stochastic gradient descent with learning rate 5×10^{-3} and Nesterov momentum ($\mu = 0.9$). Mini-batches of size 128 are sampled stochastically and simple data augmentation (horizontal and vertical flipping) is performed on the fly. We regularise the network with batch normalisation Ioffe and Szegedy [2015] and weight decay ($\lambda = 3 \times 10^{-5}$).

Inspired by Redmon et al. [2016], we formulate a multi-task prediction in which we predict $Pr(o)$, where o indicates the presence of an object in the center of the receptive field and, separately, $Pr(c|o)$, that is, the probability of cell phenotype class c given the presence of an object. These probabilities are combined at inference time (Section 5.3.2). In addition, our network performs regression on the height h and width w of the bounding box of the cell, measured as a fraction of the crop size from the crop centre. This information is readily available when generating the training set. Note that

[6] `https://github.com/jcboyd/detecting-lymphocytes`

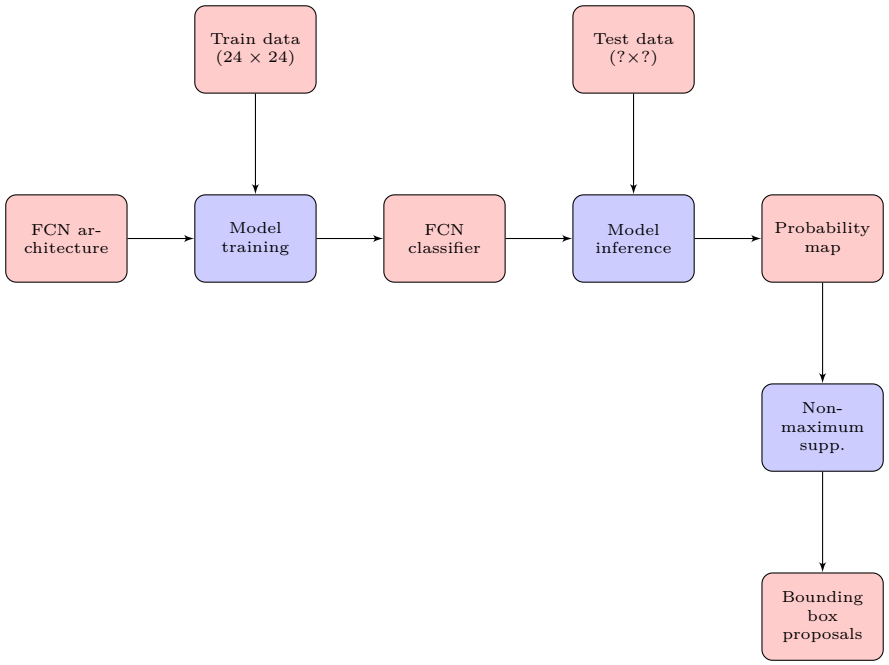


Figure 5.12: Pipeline for training a deployment of a fully convolutional classifier. Training may occur on fixed-sized input (24×24)px, but convolutions permit variable-sized inference.

Layer	Connection	Size	Output $(w \times h \times d)$
Input	-	-	$24 \times 24 \times 1$
Conv_1	Input	3×3	$22 \times 22 \times 16$
Conv_2	Conv_1	3×3	$20 \times 20 \times 16$
MaxPool_1	Conv_2	2×2	$10 \times 10 \times 16$
Conv_3	MaxPool_1	3×3	$8 \times 8 \times 64$
Conv_4	Conv_3	3×3	$6 \times 6 \times 64$
MaxPool_2	Conv_4	2×2	$3 \times 3 \times 64$
Conv_5	MaxPool_2	1×1	$1 \times 1 \times 128$
Conv_6	Conv_5	1×1	$1 \times 1 \times 128$
Output_o	Conv_6	1×1	$1 \times 1 \times 1$
Output_c	Conv_6	1×1	$1 \times 1 \times 1$
Output_b	Conv_6	1×1	$1 \times 1 \times 2$

Table 5.2: Specification of the network architecture. We distinguish three multitask outputs.

we make the assumption that our chosen crop size represents a hard maximum on the size of a cell's bounding box, a reasonable simplification for our dataset. Our network is therefore trained to minimise the loss func-

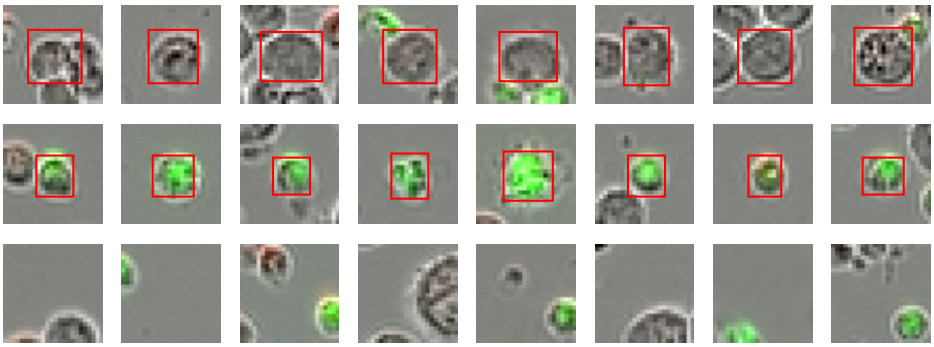


Figure 5.13: Samples of living Raji cells (top), dead cells (middle), background (bottom) annotated with bounding boxes. Fluorescence is included for clarity only and is not used in training.

tion,

$$\mathcal{L}(\mathbf{x}_i, o_i, c_i, w_i, h_i; \theta) = l_o(\hat{o}_i, o_i) + \mathbb{1}_o^i[l_c(\hat{c}_i, c_i)] + \mathbb{1}_o^i[l_b(\hat{w}_i, w_i, \hat{h}_i, h_i)] \tag{5.15}$$

with respect to model parameters θ, where l_o and l_c are each a standard cross entropy and $l_b(\hat{w}_i, w_i, \hat{h}_i, h_i) = (\hat{w}_i - w_i)^2 + (\hat{h}_i - h_i)^2$. The estimates $\hat{o}_i, \hat{c}_i$ $\hat{w}_i$, and $\hat{h}_i$ are the network outputs for object presence, object class, and bounding box width and height. The indicator function $\mathbb{1}_o^i = 1$ when training example $\mathbf{x}_i$ contains an object and $\mathbb{1}_o^i = 0$ otherwise.

We benchmarked our network as a classifier of cropped cells against a logistic regression trained on features extracted from a pre-trained 50-layer ResNetHe et al. [2016]. In order to do this, we resized our cropped cells to 32×32px and recorded the final convolutional layer of the ResNet, a vector of dimension 2048. This baseline achieved an accuracy of 0.83 on balanced test data, whereas our own network achieved 0.96. Though deep pre-trained networks are known to be powerful general-purpose feature extractorsSharif Razavian et al. [2014], they may also be over-parameterised for many problemsRaghu et al. [2019].

Inference as a detector

Following Sermanet et al. [2013] we designed our network to be *fully convolutional* (FCN). A FCN is capable of performing inference on any size of input, and is extended naturally to object detection. Thus, once trained on cell crops as a classifier, inference may be performed on an entire image

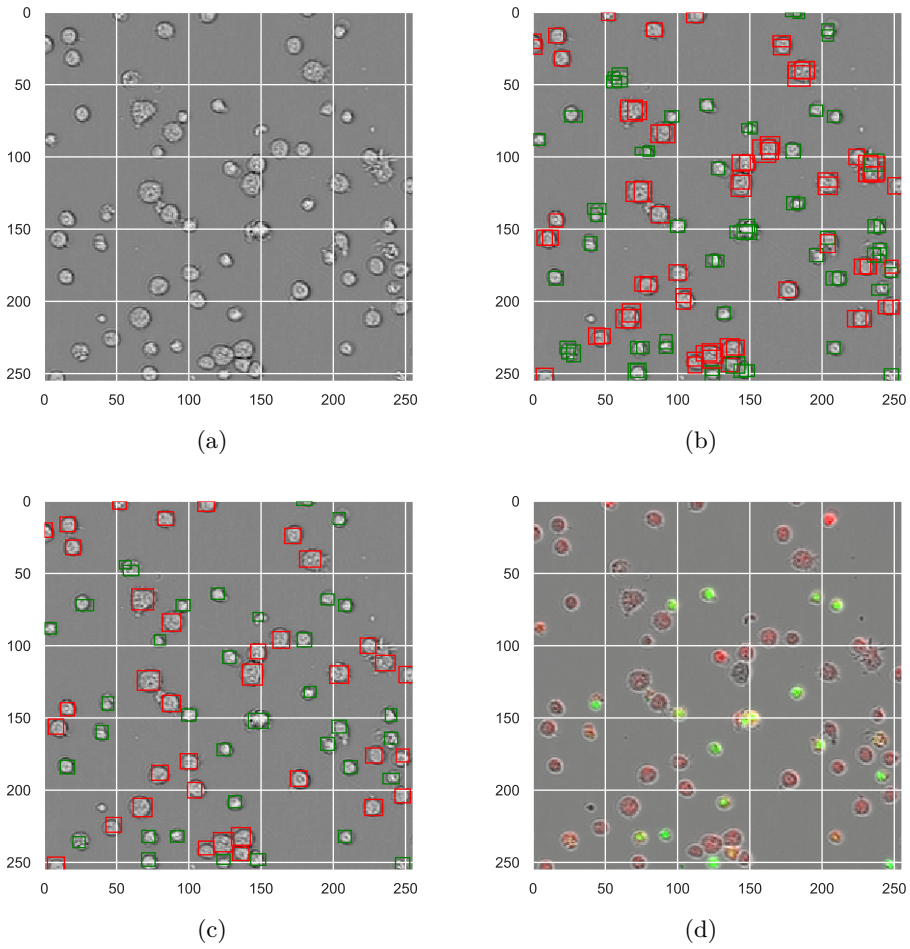


Figure 5.14: The processing of a test image: phase contrast input image (a); raw model bounding box predictions (b); non-maximum suppression post-processing (c); finally, for comparison, the corresponding full fluorescence image (d).

in a single forward pass, producing a map of softmax probabilities at every location in the image. Note that even on CPU, a full 1408×1040px image is processed by the network in about 1s. Fully-convolutional whole-image inference emulates sliding-window detection, albeit without the tremendous inefficiency of executing the model separately at every spatial position. Note that the resolution of the output will depend on the number of pooling layers in the network. For example, our network includes two max pooling layers, hence we make detections at a stride of 4 across the input image domain.

At inference time, the object and conditional class probabilities are combined to give the marginal class probabilities $Pr(c) = Pr(c|o) \cdot Pr(o)$. Note that $Pr(c|\neg o) = 0$. These probabilities are thresholded and pruned with non-maximum suppression (NMS), providing a final detection mask for each class. For the NMS algorithm, we use an intersection over union threshold of 0.35. An example of this procedure is shown in Figure 5.14.

Smoothing probabilities in time

Because the cells are relatively stationary, we can improve the prediction of our system by leveraging information across time. We find a simple weighted average of prediction probabilities from consecutive frames, computed prior to NMS, improves overall performance. We thus define the *smoothed* probability $p_{ij}^{(t)} \leftarrow 1/4 \cdot p_{ij}^{(t-1)} + 1/2 \cdot p_{ij}^{(t)} + 1/4 \cdot p_{ij}^{(t+1)}$ for the probability at image position (i, j) at time t. The weights were tuned manually for both performance and parsimony.

5.3.3 Results

Evaluation strategy

To evaluate our system, we manually annotated three days worth of frames of size 256×256px from each of two independent experimental replicates, totaling 72 images and approximately 7,000 test object detections. The replicates were chosen to represent different population dynamics: the first exhibits higher levels of cell mitosis; the second exhibits higher levels of cell apoptosis. We henceforth refer to these two datasets as Mitosis and Apoptosis respectively. The annotations consist of manually annotated bounding boxes around the cells. We make this dataset publicly available along with the images used to train the network[7]. Note that despite this manually

[7] `https://zenodo.org/record/3515446`

annotated evaluation dataset, our model is still trained on a ground truth that is automatically generated from the experiment.

We score our detections in terms of the distance of the bounding box centers to the ground truth bounding box centers. We define the following metrics:

- True positive (TP) - a cell is detected in the vicinity of a ground truth cell.
- False positive (FP) - a cell is detected outside the vicinity of any ground truth cell.
- False negative (FN) - no cell is detected within the vicinity of a ground truth cell.

Here we define vicinity to be $\leq$ 10px, the maximum distance a predicted cell center may fall from a ground true center while still falling within the typical cell bounding box (14×14px). These metrics are computed per cell class, from which we calculate precision, recall, and F_1 scores. Note the F_1 score prevails over the commonly used Matthews correlation coefficient as it does not require us to define true negatives (a meaningless quantity in our framework). These are displayed in Tables 5.3 and 5.4 for the Mitosis and Apoptosis test sets. We see the effect of smoothing is globally positive, significantly improving the precision of the dead cell class, and giving the highest average F_1 scores of 83.86 for Mitosis and 81.19 for Apoptosis. Note that the results on the dead cell class are markedly worse. We postulate this is due to the class imbalance at test time, as well as the difficulty of discerning individual cells from cell clusters.

Method	Class	Precision	Recall	F_1
Without smoothing	Living	0.8534	0.8636	0.8585
	Dead	0.7179	0.8693	0.7864
With smoothing	Living	0.8466	0.8883	**0.8669**
	Dead	0.7702	0.8549	**0.8103**

Table 5.3: Detection performance on the Mitosis test set, stratified by object class. Best results in bold.

Tracking population numbers over time

Our detection system is ultimately used to enumerate cell phenotypes over the course of CAR-T experiments. In Figure 5.15 we plot ground truth population numbers against the numbers inferred by our system. One can see the increasing number of living cells in Figure 5.15(a), corresponding to

Method	Class	Precision	Recall	F_1
Without smoothing	Living	0.9451	0.7778	0.8533
	Dead	0.6253	0.8935	0.7357
With smoothing	Living	0.9447	0.7957	**0.8638**
	Dead	0.6628	0.8904	**0.7600**

Table 5.4: Detection performance on the Apoptosis test set, stratified by object class. Best results in bold.

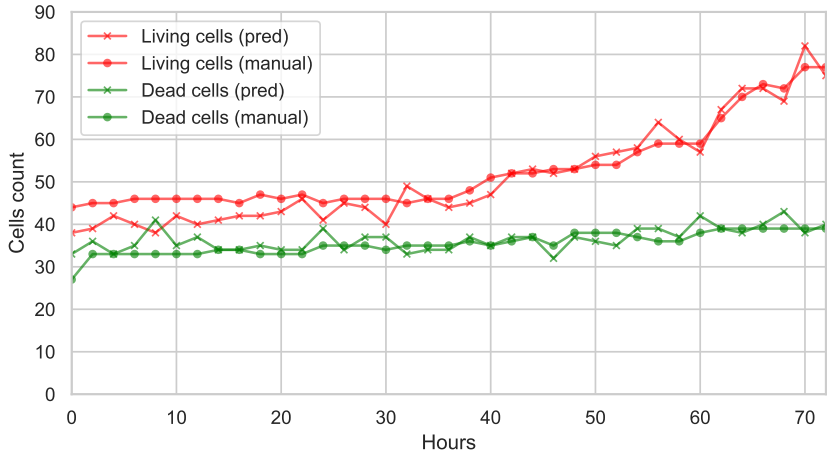
increasing amounts of cell division, whereas in Figure 5.15(b), one can see increasing amounts of apoptosis. In the former, our system achieves a mean relative error percentage of 5.95% and 5.56% (resp. living and dead cells) and 5.81% and 5.37% in the latter.

5.3.4 Perspectives

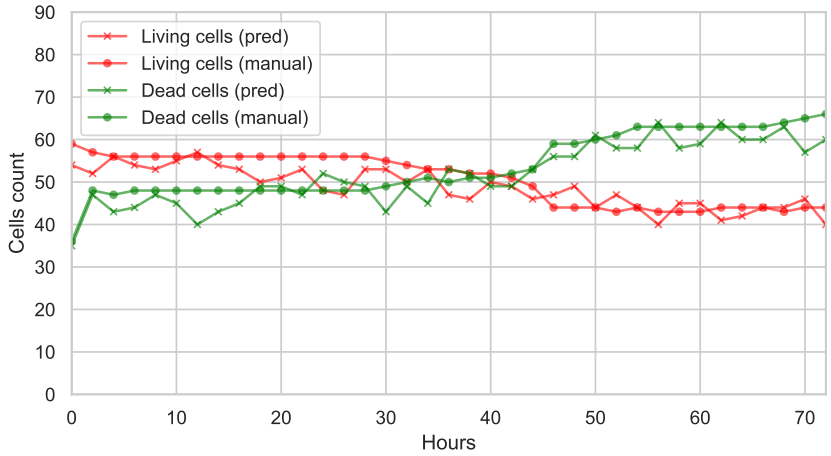
In this paper we have shown the viability of predicting phenotypes in the absence of fluorescence, as well as how fluorescence may be used to generate a robust ground truth for machine learning. We have trained a neural object detection system and tested it on two manually annotated datasets. We have also given an example of how time information can be incorporated into the prediction task. We feel the system can be further improved with a more precise and expanded dataset, something we intend to address in future work.

5.4 Discussion on competing strategies

In this chapter we have compared two strategies for quantifying cells from phase contrast images, by leveraging paired fluorescence images. In Section we showed how image-to-image translation models (including a generative adversarial variant) could be used as fluorescent labelers, and could successfully synthesise the fluorescent channels corresponding to a phase contrast microscopy input. In Section we took a different route, and showed how we could use fluorescence as the quasi-annotation of a training set for cell detector and classifier. In Figure 5.16 we compare the time series derived from applying each method on all fields of the plate, for the first 72 hours of each time lapse movie. The fields are pooled to obtain a total count for each of the four replicate wells at each time point. The time series for the detection system are obtained simply by counting bounding boxes as in Section 5.3.3. For the fluorescence labeler, we require a more ad hoc approach. We cropped the central 1024×1024px of each frame and passed it through our



(a)



(b)

Figure 5.15: Population curves for manually-annotated Mitosis test set (a) and Apoptosis test set (b), compared with detection system outputs.

labeler networks to generate the fluorescent images. We then performed a background subtraction using a mean filter to normalise the intensity levels, before thresholding to obtain a binary mask. We then simply count the number of non-zero pixels to obtain an index acting as proxy to the total number of cells. Figure 5.16 shows a strong correlation between the outputs of the two systems. This is corroborated in Table 5.5.

Well	Alive	Dead
A01	0.9751	0.8935
A02	0.9851	0.8657
A05	0.9888	0.8230
A06	0.9734	0.8840

Table 5.5: Correlations between object detection and fluorescence labeling time series for alive and dead Raji cells in four experimental replicates.

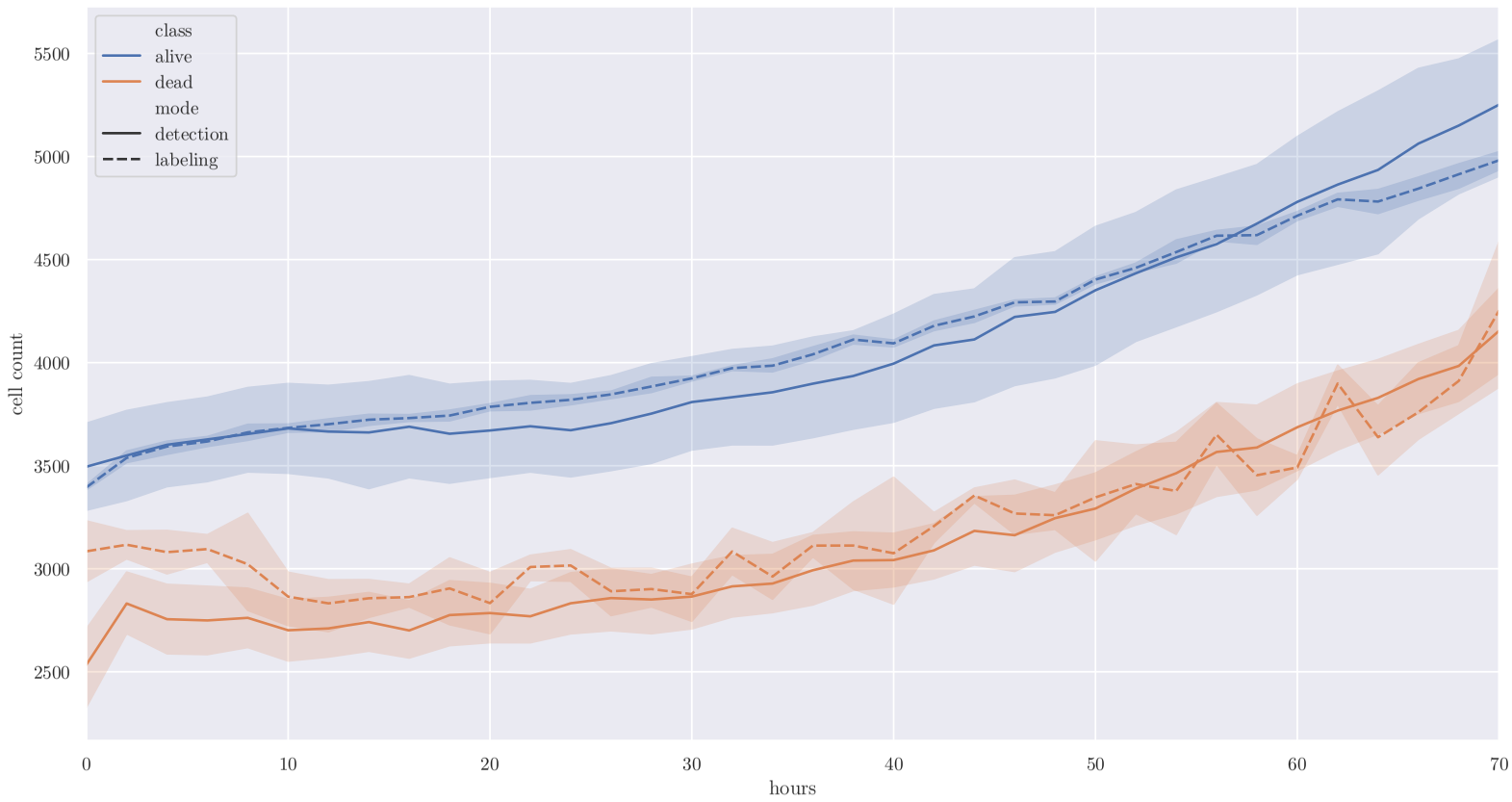


Figure 5.16: Comparing cell quantification strategy by accumulation of cell types over time in four well replicates, aggregating over fields of view. The labeling series are normalised to have the same mean as the detection series. The lines are the means taken over the four replicates and the shaded regions represent their 95% confidence intervals.

Chapter 6

Deep style transfer for synthesis of images of CAR-T cell populations

Summary: *We have seen in the previous chapter the limitations of an automatically-generated training set for phenotyping cells in CAR-T experiments, whose accuracy degrades over time. In this final chapter we propose and test novel ideas for circumventing these limitations. We formulate a style transfer problem for synthesising cell line populations by unsupervised generative models. We succeed in generating realistic phase contrast images with a complete ground truth sufficient to train a state-of-the-art object detection system.*

Résumé: *Ce chapitre...*

6.1 Overview

In Chapter 5 we compared approaches for detecting and classifying cell types from phase contrast images; directly using our object detection system, and indirectly using our fluorescence labeler (which doubles as a visualisation tool). Each approach had its shortcomings: highly clustered cells prevent the harvesting of training data for object detection, and fluorescent quenching in later experimental frames cause problems for both. In this chapter we explore an entirely different approach that aims to circumvent these problems. Our approach is based on generative models

The desired end result is a (infinite) supply of perfectly annotated synthetic images, on which we can train a state-of-the-art object detector. The in-

tention is that such a detector will transfer seamlessly to the true data and outperform our previous. Recall that such a system was infeasible on the weakly supervised dataset we previously had for training.

6.2 Feasibility study: synthesising cell crops

To assess the feasibility of using generative models for cell population generation, we attack a simpler problem: generating crops of individual cells. We have grounds to be optimistic, given the low resolution targeted, coupled with the abundance of training data. Osokin et al. [2017] earlier achieved impressive results synthesising fluorescence readouts of cell crops. Using the automatic cropping strategy detailed in Chapter 5, we assembled a training set of 42241 24×24px cell crops taken from four days of frames from a well $A01$, field 1 (a Raji-only experiment). The natural imbalance of cell states produced a class imbalance of 32507 living to 9734 dead cells. Samples from the training set are given in Figure 6.1.

Figure 6.1: Example cell-centered 24×24px crops from our ground truth training set.

We trained fully-connected and deep convolutional GANs, along with conditional versions of each (see Appendix C.1, C.2 for network specifications). In training the conditional models, we balanced the number of living and dead Raji cells sampled in each mini-batch. Other than this, the training settings were held constant across our experiments: 50 epochs; batch sizes of 120 samples; 100-dimensional Gaussian noise input. All models were trained with the Adam optimiser (Kingma and Ba [2014]) with learning rate $2e^4$), $(\beta_0, \beta_1) = (0.5, 0.999)$, and learning rate decay $1e^{-6}$, with the Jensen-Shannon objective function.

6.2.1 Generative adversarial networks

Generative adversarial networks (GANs) (Goodfellow et al. [2014b]) is a learning framework that trains a generator model to be capable of sampling new data from a data distribution (in particular, images), learnt (implicitly) from a training set. GANs compete with variational autoencoders (Kingma and Welling [2013]) and, more recently, autoregressive models (Oord et al. [2016]) for image generation. We select GANs as the principal weapon for

our work in image synthesis in this chapter, given their rich literature and versatility (as we shall soon see).

The GAN framework is *adversarial* in that it invokes a two-player minimax *game* between a pair of neural networks, a *generator*,

$$G : Z \to X, \tag{6.1}$$

where in the simplest case $\mathbf{z} \sim Z$ is a vector of (uniform or Gaussian) noise and $\mathbf{x} \sim X$ is a synthetic data point (in particular, an image), and a *discriminator*,

$$D : X \to \{0, 1\}, \tag{6.2}$$

that is, a mapping from data point to binary value. D aims to minimise its rate of failure in discerning true data from "fake" data sampled from the the generator, which aims to maximise the error made by the discriminator,

$$\max_G \min_D \mathcal{L}_{GAN}(D, G) = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{6.3}$$

The stationary point of is a *Nash equilibrium* (a saddle point) between the two objectives. Goodfellow et al. [2014b] guarantee that the distribution implicitly defined[1] by G, $p_g = p_{\text{data}}$ at (global) optimality, and $D(\mathbf{x}) = p_{\text{data}}(\mathbf{x})/(p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x}))$ where p_{data} is the true distribution of the data. GANs are trained by backpropagation with alternating gradient descent.

GAN architecture design must strike a balance between a generator expressive enough to approximate the data distribution, and a discriminator powerful enough to hold it to the task. GANs are chronically afflicted with a phenomenon known as *mode collapse*. Mode collapse is the condition in which the generator "collapses" to generating a few or even a unique output, regardless of the noise input $\mathbf{z}$. Metz et al. [2016] differentiate two varieties of mode collapse: *discrete mode collapse*, where the generator collapses to a subset of data modes; and the more insidious *manifold collapse*, where the generator collapses to a subspace of the data distribution. They reason that mode collapse originates from the fact that in each iteration of training,

[1] GANs are implicit generative models, in contrast to explicit models such as VAEs, which model the probabilities directly. GANs serve as an sample-emitting oracle emulating the true distribution.

the generator is impelled towards a delta function at the mode considered most "real" by the discriminator. The discriminator responds by lowering the probability of this mode, leading to oscillations. The "unrolled" GAN training approach that they advocate mitigates this tendency by informing the generator in advance of the discriminator's response to its prospective weight updates.

Another promising attempt to attenuate the mode collapse problem is to replace the Jensen-Shannon divergence loss function with the Earth Mover (EM) or Wasserstein-1 distance. Such is the strategy of Wasserstein GANs (WGANs). Intuitively, the EM distance measures a distance between two distributions. We can write the EM distance as the quantity $\mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z\sim p(z)}[f_w(g_\theta(z))]$ maximised by choice of f_w (discriminator). The goal is then to minimise this distance through optimisation of the generator. The training algorithm is very similar to GANs. WGANs are more stable, as the substituted loss function allows the discriminator to be trained to optimality. This greatly mitigates the mode collapse problem. Nevertheless, more subtle forms of partial mode collapse remain a recurrent problem for GANs. In addition, the discriminator loss (estimated EM distance) becomes meaningful during training, interpretable as image sample quality, rendering the implementation and fine-tuning of WGANs far easier. Despite the merits of WGANs, we do not have recourse to them in this work.

The simplest practicable variant of GANs incorporates D and G as fully-connected networks. Such models, (perhaps cherry-picked) success on generating MNIST digits in Goodfellow et al. [2014b]. Figure 6.2 shows, however, underwhelming results sampled from the trained fully-connected generator, with only rudimentary intensity levels and cellular structure of the ground truth captured. We furthermore encountered difficulties with mode collapse more frequently than when dealing with its convolutional counterparts below. This lends support to our presumption that there is little to recommend fully-connected GANs for computer vision applications.

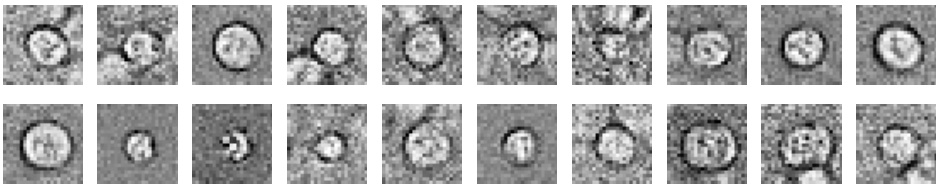


Figure 6.2: Sample 24 × 24px crops from our fully-connected GAN.

6.2.2 Deep convolutional GANs

Radford et al. [2015] first succeeded in training more powerful, high-resolution

GANs, by identifying a set of design principles: replacing pooling layers with strided convolutions; removing fully-connected layers (apart from the first layer of the generator); using batch normalisation after all but the last layer; using a tanh activation at the end of the generator (and ReLU everywhere else); and using LeakyReLU activations everywhere in the discriminator. Deep convolutional GANs (DC-GANs) represent a vast improvement over the MLP-based GANs trained above. Figure 6.3 provides samples from the convolutional generator that are qualitatively close to ground truth samples[2], with recognisable cellular characteristics, as well as convincing neighbouring cells on the crop periphery.

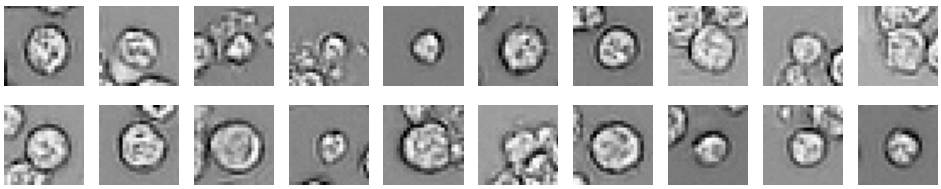


Figure 6.3: Sample 24×24px crops from our DCGAN. One may notice the generator has learned to synthesise convincing peripheral cells.

To establish the uniqueness of these samples, we trained a k-nearest neighbour classifier (k-NN) ($k = 9$) to retrieve the closest samples (in Euclidean distance) within the training data. Such a test is important, as we do not wish the network merely to memorise ground truth samples. A set of comparisons are made in Figure 6.4, illustrating the absence of sample "memorisation". Note, however, that this is neither an altogether reliable nor conclusive test: a Euclidean k-NN could be easily fooled by a "parrot" GAN that simply scaled intensities across memorised samples, but this does not appear to be the case here.

6.2.3 Conditional GANs

Mirza and Osindero [2014] present a simple extension to GANs allowing for control over the data generating process. One simply includes an additional input $\mathbf{y} \sim Y$ (for example, class information), to obtain conditional GANs (cGANs). That is, now the generator is conditional,

$$G : Z, Y \to X, \tag{6.4}$$

as is the discriminator,

[2]Indeed, when swapped with true samples, they succeeded in fooling a room full of experts as to their authenticity.

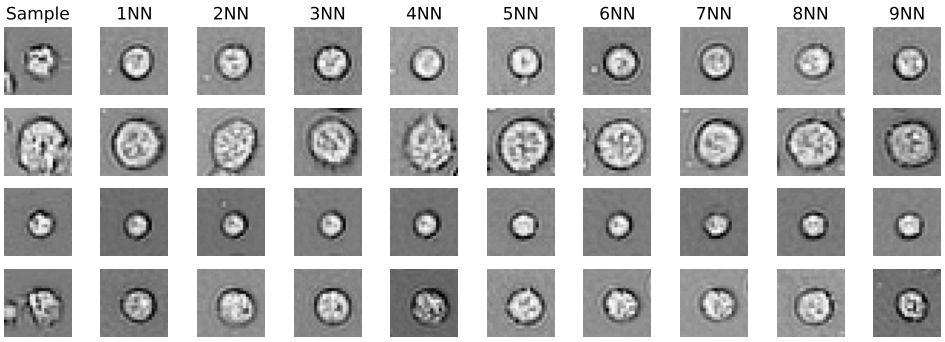


Figure 6.4: Comparison of cells generated with DCGAN (left-most column) against 9 nearest neighbours from training set.

$$G : X, Y \rightarrow \{0, 1\}, \tag{6.5}$$

and the cGAN is trained against the objective,

$$\mathcal{L}_{cGAN}(D, G) = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[\log D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\mathbf{y})))] \tag{6.6}$$

Mirza and Osindero [2014] does not offer so much as an intuition as to why conditional GANs works (seemingly the article was written in a hurry), but it can be instructive to consider. As D learns to associate conditional information such as object class with the ground truth images, it will declare "fake" any image emitted by G not conforming to the condition, no matter the quality of the image. If G is to succeed, it will be forced to generate images in adherence to the condition. The outcome of successful training is therefore a generator that can, for example, synthesise an authentic-looking image of a desired class on demand.

The implementation of cGANs is a straightforward modification to GANs: the conditional information is concatenated to the typical usual inputs of both D and G and fed to further hidden layers. A DCGAN discriminator may, however, concatenate the conditional information after its convolutional layers.

We try both fully-connected and deep convolutional variants of cGANs, where we condition on class information for living and dead Raji cells. This we encode as a two-dimensional one-hot vector. Note that this represents a first attempt to control the data generation process. As with our previous results with GANs, the results with fully-connected networks are underwhelming, and we omit them here. Figure 6.5, however, displays samples from a

conditional DCGAN, which are qualitatively both as convincing as our DCGAN samples, and also obey the conditional class signal, allowing us to be able to generate–on demand–10 samples apiece for the two classes.

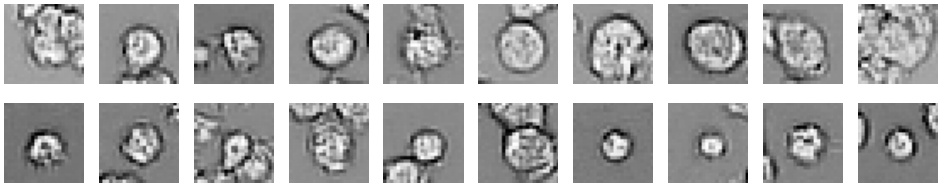


Figure 6.5: Sample 24 × 24px crops from our cDCGAN. The top row are crops produced by conditioning for living Raji cells; the bottom are crops produced by conditioning for dead Raji cells.

6.3 Style transfer for simulating populations of Raji cells

Gatys et al. [2016] proposed a "neural algorithm of artistic style", an algorithm to perform *style transfer* using neural networks, the synthesis of an image combining textures of one style image and another content image. Note that this involved neither the architecting nor the training of a new purpose-built neural network, rather the clever utilisation of (any) pretrained CNN. The work combined two related ideas: content reconstruction and texture reconstruction using CNNs. **Content reconstruction** with deep networks has been explored since the very first wave of CNNs assumed their primacy. Zeiler and Fergus [2014] showed how their earlier "deconvolutional network" (Zeiler et al. [2010]) could be used to reconstruct input images from the activations inside CNNs [3] Simonyan et al. [2013] proposed a gradient-based method for visualising learned features of a pretrained network. Thus, a randomly initialised image was transformed into a structured image by incrementally adding its (regularised) gradient with respect to of a chosen neuron from the network, so as to maximised the activation of that neuron. The deepest neurons, those corresponding to semantic object classes let to the synthesis of recognisable yet ethereal impressions of those class objects. The authors argued their technique roughly generalised that of Zeiler and Fergus [2014]. Mahendran and Vedaldi [2015] took idea of using neuron-to-image gradients to full image construction. This is the idea that later inspired neural style transfer: a target image is fed into a pre-trained

[3]The work is best remembered for proposing "ZF-Net", a very modest modification of AlexNet, as well as using pre-trained CNNs as feature extractors for other models.). In so doing, the paper did a lot to demystify the feature hierarchies learned by deep CNNs.

CNN and the tensor of its activations recorded; a randomly initialised synthesis image is then fed into the network, producing its own activation; the synthesis image is updated using gradient descent so as to minimise the mean square error of its activation to that of the target. The outcome is a near-perfect reconstruction of the image. **Texture reconstruction** with CNNs was achieved by the authors of style transfer in Gatys et al. [2015], in a prelude to that later work. The principle is the same as Mahendran and Vedaldi [2015], yet rather than to reconstruct an image for its activations directly, ones reconstructs for a statisical properties derived from those features, namely their Gram matrix. The principle is that two images sharing such features are identical from a textural perspective. The authors write,

> Textures are per definition stationary, so a texture model needs to be agnostic to spatial information. A summary statistic that discards the spatial information in the feature maps is given by the correlations between the responses of different features. These feature correlations are, up to a constant of proportionality, given by the Gram matrix.

In Gatys et al. [2016], the ideas of style and content reconstruction were combined. Thus, one elects style $\mathbf{s}$ and content $\mathbf{c}$ reference images, and simultaneously reconstructs gradient-based reconstructions of a randomly initialised image $\mathbf{x}$,

$$\mathcal{L}(\mathbf{s}, \mathbf{c}, \mathbf{x}) = \alpha \cdot \mathcal{L}_{\text{content}}(\mathbf{c}, \mathbf{x}) + \beta \cdot \mathcal{L}_{\text{style}}(\mathbf{s}, \mathbf{x}) \tag{6.7}$$

for tunable weights α and β and where the content loss,

$$\mathcal{L}_{\text{content}}(\mathbf{c}, \mathbf{x}) = \frac{1}{2}||F^{(c)}(\mathbf{c}) - F^{(c)}(\mathbf{x})||_F^2 \tag{6.8}$$

for the vectorised cth layer (chosen by the programmer) of the pretrained network $F^{(c)} \in \mathbb{R}^{(N_l M_l \times K}$, and the style loss,

$$\mathcal{L}_{\text{style}}(\mathbf{s}, \mathbf{x}) = \sum_{s \in S} \frac{1}{4N_s^2 M_s^2} ||G^{(s)}(\mathbf{c}) - G^{(s)}(\mathbf{x})||_F^2 \tag{6.9}$$

where $G^{(s)} = F^{(s)}(F^{(s)})^T \in \mathbb{R}^{K \times K}$ is the Gram matrix of the matrix of vectorised feature maps, $F^{(s)}$. Note that the content reconstruction is usually performed on a single, deeper layer of the pretrained CNN, which is more invariant to style, while the style reconstruction can be performed at several

(a) Style

(b) Content

(c) Synthesis

Figure 6.6: A neural algorithm of artistic style combines uses a pretrained CNN to combine properties from a style source image (a) and a content source image (b) to synthesise a new image combining their characteristics (c). Example taken from https://github.com/jcboyd/vgg-fun

shallower layers at once, to capture style at different scales. The result is a fusion of content and style in a synthesised image. An example is given in Figure 6.6.

6.3.1 Conditional dilation for tessellating cell contours

We pursue an application of style transfer to synthesise images of cells in phase contrast images from our CAR-T dataset. We motivate this from the shortcomings of our automatic training set extraction presented in Chapter 5, which provides us with an incomplete annotation of our data, and fails, crucially, in scenarios where cells cluster profusely, such as after mitotic events. We moreover found that this translated into the degradation of performance in time of our both our fluorescence labeler and object detector. In principle, a perfectly annotated ground truth could be constructed using

a style transfer procedure:

1. specify our desired content image by "drawing" content
2. manually crop a region of a phase contrast image with similar content
3. run the style transfer algorithm on the above

Hypothetically, the outcome is a realistic-looking image of cells for which we have perfect knowledge of content, including cell position and size, hence the majority of information required for a fully-annotated object detection dataset (we set aside the issue of class information for now), and indeed relates to the concept data simulation. The idea of using style transfer for data augmentation has been shown to work in for example Zheng et al. [2019].

Given the simple geometric structure of cells, a good first approximation to a drawn cell is a circle. Therefore, in a space image plane X, the mask of an *isolated* cell k can be modeled as the set of points within distance r_k to a non-empty set or "site" P_k,

$$R_k = \{x \in X | d(x, P_k) \leq r_k\} \tag{6.10}$$

where the distance $d(x, P_k) = \inf\{d(x, p) | p \in P_k\}$, that is, the shortest distance to any point in the site. When P_k is a singleton set, the region becomes a circle. Otherwise, we may define the site P_k in any way we like (for example, a line or eroded mask) to model random, anisotropic variations in cell shape. The radius,

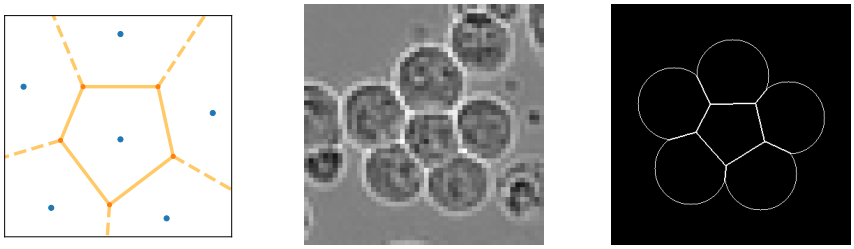
$$r_k \sim \mathcal{N}(\mu_i, \sigma_i) \tag{6.11}$$

can be sampled from an estimation of the radius distribution for class i. Once again, we are avoiding the complication of prescribing classes to cells for the present.

Though isolated cells are highly circular, closely neighbouring cells manifest a shared, flattened contour (see Figure 6.7b). Previously, Bock et al. [2010] have modeled the border forces of neighbouring cells as Voronoi-tessellating circles. A Voronoi tessellation can be defined as K regions of the form,

$$S_k = \Big\{x \in X | d(x, P_k) \leq d(x, P_j) \forall j \neq k\Big\} \tag{6.12}$$

The K regions constitute a partition of the space X, where points are assigned to the nearest site. An example is shown in Figure 6.7a where we



(a) (b) (c)

Figure 6.7: Example of Voronoi tessellation with six sites (blue points) and corresponding regions (yellow lines) (generated in `scipy` (Virtanen et al. [2020])) (a). Neighbouring cells exhibit a tessellating effect at the border (b). A conditional dilation algorithm can model the bordering when it comes to synthesising content images for style transfer.

invoke Euclidean distance for a set of six singleton sites. The borders between regions are lines perpendicular to the line joining the site centers. We propose to model our cells as the intersection of 6.10 and 6.12,

$$T_k = \left\{ x \in X | d(x, P_k) \leq r_k \wedge d(x, P_k) \leq d(x, P_j) \; \forall j \neq k \right\} \tag{6.13}$$

To compute the T_k, we design an algorithm for *conditional dilation*. The principle is to select the sites (cell centres) and to grow outwards until either a maximum distance from the source is reached, or an object border is encountered. The procedure is shown in Algorithm . This resembles an implementation of the watershed algorithm, albeit with all seeds flooded from the same height, implemented as a priority of 0. The dilation of sites is scheduled according to a priority queue, for which priority is assigned as the distance from the source. Thus, dilation propagates from the sites synchronously. The priority queue is implemented as a *heap*, as it can push and pop in $\mathcal{O}(\log N)$ time.

Applying this to drawing cell population content images, the space X becomes an image "canvas", that is, a two-dimensional array of numbers initialised to zero to indicate background. Sites are allocated as connected components (perhaps just points) of non-zero, foreground pixels. The canvas with its allocated sites and corresponding dilation radii are inputs to the ConditionalDilation algorithm. This works well in practice as long as the sites are carefully selected. For the time being, sites are chosen uniformly at random[4], however sites must be chosen outside all other regions. Provided the canvas is large enough relative to the cells, and the cell number, there is a large space of feasible allocations of sites. For efficiency, we

[4]One could imagine using a dynamic distribution to simulate cell clustering.

Algorithm 1 Dilates labeled image SiteImage until distance from source exceeds Radii.

```
1: procedure ConditionalDilation(SiteImage, Radii)
2:     PriorityQueue ← Heap()
3:     for marker in SiteImage do
4:         Elem ← HeapElement()
5:         Elem.value ← 0                                  ▷ Top priority
6:         Elem.source ← marker                            ▷ Record origin
7:         PriorityQueue.Push(Elem)
8:     end for
9:     while ¬ PriorityQueue.IsEmpty() do
10:        Elem ← PriorityQueue.Pop()
11:        for Neighbour in Elem.GetNeighbours() do
12:            if SiteImage[Neighbour] > 0 then
13:                continue                  ▷ Neighbour already visited
14:            end if
15:            NewElem ← HeapElement()
16:            NewElem.value ← Dist(Neighbour, Elem.source)
17:            if NewElem.value > Radii[Elem.source] then
18:                continue                        ▷ Cell border reached
19:            end if
20:            NewElem.source ← Elem.source
21:            PriorityQueue.Push(NewElem)
22:            SiteImage[Neighbour] = SiteImage[Elem.source]
23:        end for
24:    end while
25:    return SiteImage
26: end procedure
```

keep a record of the vacant coordinates in the drawing canvas, from which we subtract the coordinates of the anticipated region of each allocated site on-the-fly. If we naively allocate sites in arbitrary order, we risk the scenario by which a larger cell is allocated in the vicinity of a smaller one, placing the center of the smaller cell within the region of the larger. A neat solution is to simply allocate the sites in descending order of radius. This procedure is described in Algorithm 2.

Algorithm 2 Allocates sites for input to ConditionalDilation algorithm. VacantCoords are a list of (x, y) coordinates available for allocation (initially a full image), and Radii are the presampled radii for drawing cells.

```
procedure AllocateSites(VacantCoords, Radii)
    Sites ← []
    Radii ← Sort(Radii, descending=TRUE)
    for radius in radii do
        Site ← Sample(VacantCoords)
        Sites ← Sites ∪ Site
        Region ← {Coord ∈ VacantCoords | ||Coord - Site||_2 < radius}
        VacantCoords ← VacantCoords \ Region
    end for
    return Sites
end procedure
```

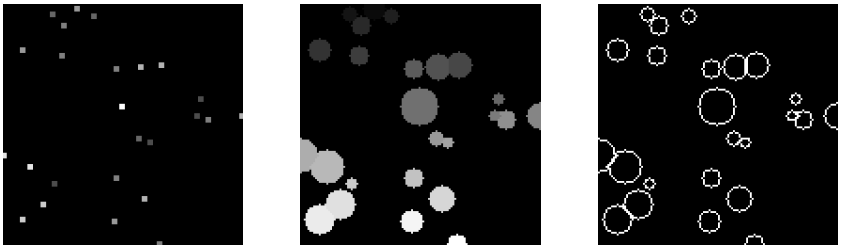
The outcome of the ConditionalDilation algorithm is a content image in the form of a labeled mask. While masks are a natural choice for cell content markers, an alternative that we find more useful is the cell contour defined as,

$$C_k = T_k - T_k \oplus B_D \tag{6.14}$$

where $\oplus$ represents a morphological dilation or maximum filter, M is the object mask, and B_D is a (for example, disk-shaped) structuring element. The full procedure for generating content images is illustrated in Figure 6.8.

Results

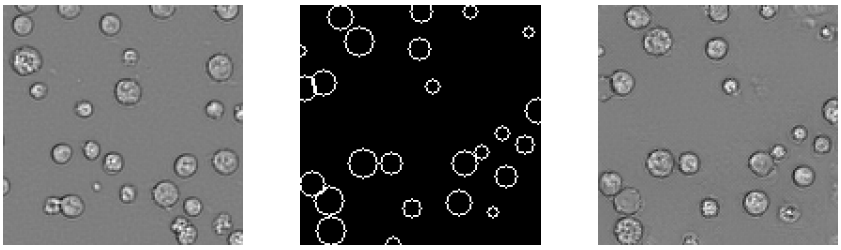
In Figure 6.9 we show a first result with style transfer. The contours of cells are generated according to Equation 6.11, which we can estimate from our library of pre-cropped cells in Chapter 5. The number of cells is chosen to match the chosen style image, and cell are placed uniformly randomly on the zeroed background plane. Here we see that the transfer succeeds– the background and foreground textures have been suitably apportioned



(a) Points (b) Regions (c) Contours

Figure 6.8: An example of generating a cell content image by first allocating sites with AllocateSites (a) (site intensity represents radius, dilated for visibility), with radii drawn from Equation 6.11; running the ConditionalDilation algorithm to obtain labeled regions as in Equation 6.13 (b); and extracting the contours with Equation 6.14 (c).

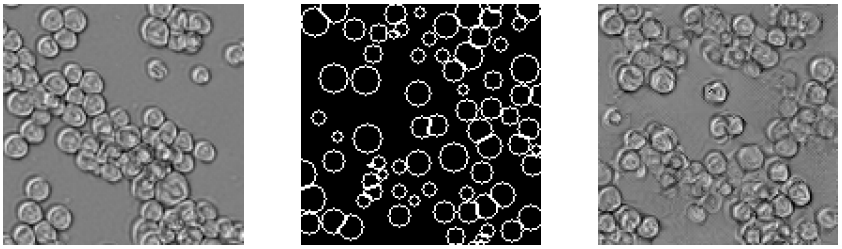
to the objects prescribed by the content image. Various artifacts exist, of course, but individual cells appear correctly rendered, and the main fault is the global arrangement of the cell population lacking sense (recall we have placed objects uniformly randomly).



(a) Style (b) Content (c) Synthesis

Figure 6.9: An application of a neural algorithm of artistic style to phase contrast images of CAR-T cells (a) and a style source image (b) to synthesise a new image combining their characteristics in a "simulated" image (c).

Note that the low-resolution and simple geometric structure of the cells puts us at a natural advantage. It is hard to imagine this approach working very well for anything but simple geometric shapes. There are, nevertheless, limitations to this approach. We see in Figure 6.10 a larger population of cells with a greater degree of failure. In general, we observe a tension between the content specification and the chosen style crop. Note that Equation 6.9 aims to ensure the same co-occurring neural activations invoked by the style target are invoked by the synthetic image also, albeit the global arrangement is arbitrary due to stationarity of the Gram matrix. The algorithm preferences applying these textures in the vicinity of recognisable contours, which is why



(a) Style (b) Content (c) Synthesis

Figure 6.10: More cells, more problems. The style transfer algorithm fails when the content image does not offer adequate "scaffolding" for the target texture patterns.

we can steer the synthesis reasonably well. Furthermore, we find the cell geometry is flexible: when the disk contours are replaced with diamonds, the algorithm still performs well. Yet, we find that it is crucial to approximately match the size and number of cells between the style and content images to obtain a good coverage of texture. Indeed, the style image appears to provide a certain "budget" that if underspent will be apportioned arbitrarily (such as to the background), and if overspent will mean some objects end up untextured.

Though as a proof of concept this approach is successful and yields various insights, the pre-trained network, with all its particularities, does not give us the necessary control for bootstrapping a robust ground truth. In addition, the approach entails the manual curation of a suitable style reference image. Finally, the generation is slow: one must backprop a large neural network (although Johnson et al. [2016] find ways of greatly improving the run time). To overcome these difficulties, in Section 6.4 we reformulate the problem as a learning problem of image translation.

6.4 CycleGANs for cell population synthesis

CycleGANs (Isola et al. [2017]) extend the `pix2pix` framework (see Section 5.2.1) to the *unpaired* image-to-image translation setting, that is, translating between two (necessarily similar) image domains in a fully unsupervised way. Though the underlying neural network architectures remain virtually unchanged, the framework is quite different. CycleGANs introduce an additional pair of generator and discriminator. Now, one generator learns the mapping $G : X \to Y$ while the other learns $F : Y \to X$. The respective discriminators train these generators adversarially in the usual way. However, at the same time, there are cycle consistency losses, ensuring $F(G(X)) \approx X$

(forward cycle consistency) and $G(F(Y)) \approx Y$ (backward cycle consistency). We write,

$$\mathcal{L}_{CYC}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[||F(G(x)) - x||_1] + \mathbb{E}_{y \sim p_{data}(y)}[||G(F(y)) - y||_1], \tag{6.15}$$

that is, the expected l_1 error upon reconstructing real images via the stacked generators. The cycle consistency losses are combined with the usual GAN losses for each translation,

$$\max_{G,F} \min_{D_X, D_Y} \mathcal{L}_{CG} = \mathcal{L}_{GAN}(D_Y, G, X, Y) + \mathcal{L}_{GAN}(D_X, F, Y, X) + \lambda \mathcal{L}_{CYC}(G, F) \tag{6.16}$$

Thus, it must be that such a consistent translation can be discovered between the domains. If the domains are similar, this works surprisingly well in practice. Note that unlike `pix2pix` (paired image translation) CycleGANs are no longer conditional in that the discriminators no longer have access to the source image. The generators are still conditioned on their input image, however, instead of the noise vector in a vanilla GAN. A successfully trained CycleGAN can learn to perform a (bidirectional) style transfer between, for example, photographs and paintings, and maps to satellite imagery. In contrast with *A Neural Algorithm of Artistic Style*, however, it requires a full training corpus to learn from. At test time, the style transfer is performed in a single forward pass of the generator. Moreover, empirical results suggest the outcomes are superior (Isola et al. [2017]). This is unsurprising, as the CycleGAN is optimised to perform style transfer directly, rather than a clever utilisation of a pretrained network. the results are not as impressive as `pix2pix`, which they affirm to be an upper bound on performance. However, the CycleGAN is still trained on a surprisingly small number of images, despite being a very deep architecture. Even more recently, *RecycleGANs* (Bansal et al. [2018]) achieve the astonishing feat of unpaired video-to-video translation. These build on the CycleGAN framework, introducing "recurrent temporal predictor" networks, and replace the cycle consistency losses, with a similarly defined "recycle loss". Although we are indeed working with video data, Recycle-GANs seem excessive for the problem at hand; CycleGANs should be sufficient.

6.4.1 First results with CycleGANs

We therefore propose to use CycleGANs as an improvement for the style transfer algorithm in Section 6.3. We model the domain X as the content

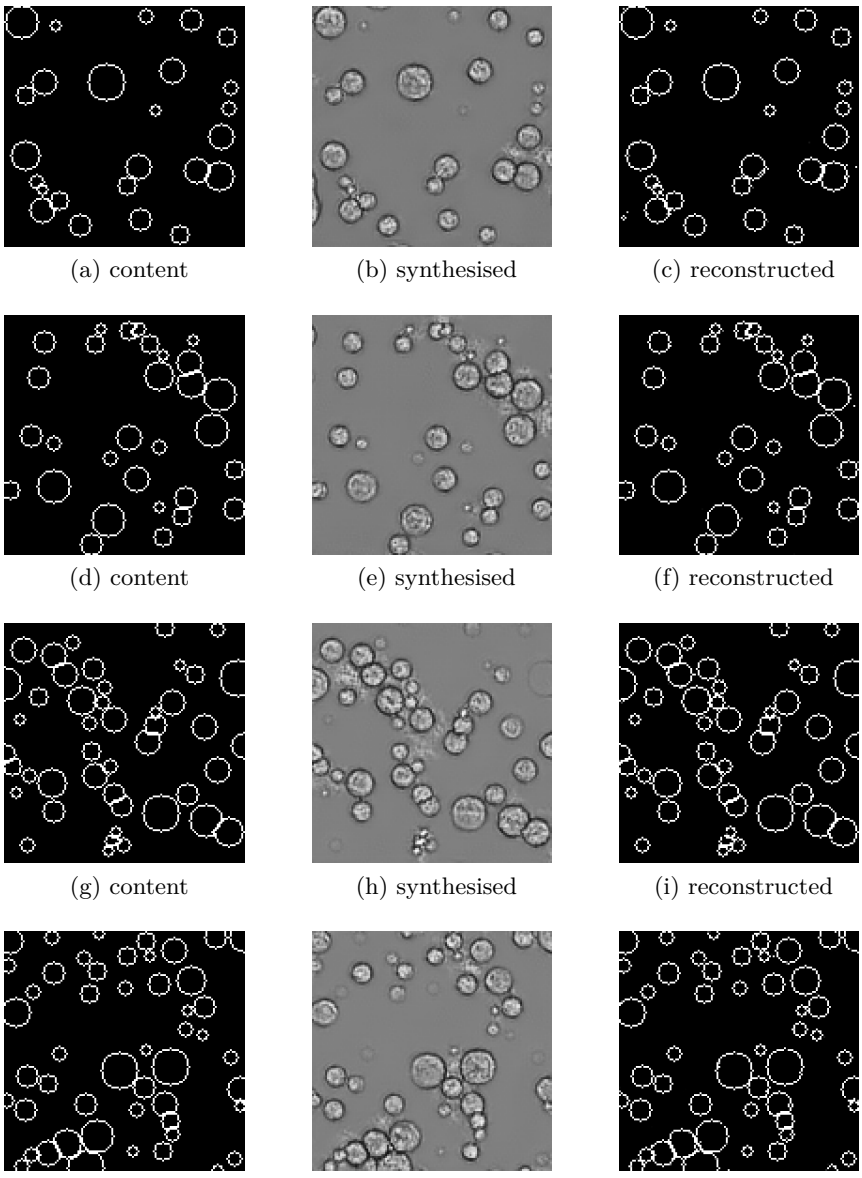
domain, where while training the CycleGAN, the content specifications are randomly generated according to the rule set we inject. The domain Y is again the style domain of phase contrast images, however now we can freely sample style image patches at random, that is, without the manual discretion involved in Section 6.3. The idea of synthesising a training set with image-to-image translation models has numerous precedents in recent times, including with `pix2pix` for object segmentation (Hollandi et al. [2019]) and CycleGANs for both segmentation and object detection (Fu et al. [2018], Ihle et al. [2019]).

We train CycleGANs to map between synthetic cell drawings and authentic phase contrast crops. We follow the alternating fashion of GAN training: the two discriminators are trained on a single batch, then the two generators. Our batch size is limited to a single image as in the related networks in Section 5.2.1, and, as in Section 6.2, we use the Adam optimiser with maximium learning rate $2e^4$) and $(\beta_0, \beta_1) = (0.5, 0.999)$. The generator and discriminator closely resemble the FNet and PatchGAN of Section 5.2.1, however, to decrease training time and avoid memory concerns, in practice we halve the number of filters in each generator and decreased the depth of the network. In addition, the batch normalisation layers are replaced by the instance normalisation layers advocated by Isola et al. [2017] (and first proposed by Krizhevsky et al. [2012]).

First results with this setup are given in Figure 6.11. The results come from the content-to-style generator G. Note that the other three networks of the CycleGAN have no immediate value for our application at present, and are effectively discarded. We deem these results to be highly satisfactory, with the system adhering closely to the specification given in our content drawings, while producing realistic textures for the cells.

6.5 Fine-tuning a state-of-the art object detection system

With the content-to-style generator from Section 6.4.1 acting as an "oracle" for perfectly-annotated training data, we can proceed to train a state-of-the-art object detection system, such as that which eluded us in Chapter 5. We choose Faster R-CNN (Girshick [2015]) for this task. A full discussion of the RCNN series of object detectors is available in Chapter 2. We take the Faster RCNN implementation available in the PyTorch framework (Paszke et al. [2017]). This has been pretrained on the COCO object detection dataset Lin et al. [2014]. A simple fine-tuning procedure is to replace the classification and bounding box regression "heads" of the network, that is, leaving the backbone and region proposal network untouched, and to continue gradient



(a) content (b) synthesised (c) reconstructed

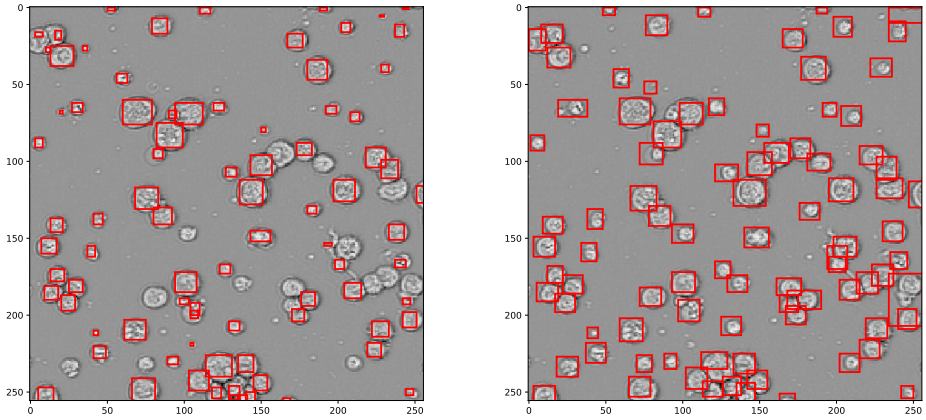
(d) content (e) synthesised (f) reconstructed

(g) content (h) synthesised (i) reconstructed

(j) content (k) synthesised (l) reconstructed

Figure 6.11: Examples of CycleGAN generated images. Columns organised with content specification input image X (left), $G(X)$ generated image (center) and reconstruction $F(G(X))$ (right).

descent on the new data with a lowered learning rate. This we do, with a learning rate of 10^{-4}, training with SGD with momentum. Training mini-batches are formed with 8 synthetic images. The (perfect) bounding box annotation (and, in principle, object class information) is determined by the drawing procedure of the synthetic image.



(a) Weakly supervised training set (b) Synthetic training set

Figure 6.12: Comparison of outcomes of Faster R-CNN on a test image when a) fine-tuned on a weakly supervised dataset and b) fine-tuned on synthetic images.

The outcome of training is a detection system capable of detecting cells (but not classification, for now). We compare Faster R-CNN trained on our synthetic images with the incomplete annotations arising from the automatic pipeline proposed in Section 6.6.1. Recall, our argument for pursuing a custom detector in Chapter 5 was that such a *weakly supervised* dataset would be insufficient to train a state-of-the-art system like Faster R-CNN. We see in Figure 6.12 evidence to support this intuition, where we compare the performance of the fine-tuned R-CNN on an authentic test image, when fine-tuned first on a weakly supervised dataset, then on our CycleGAN-generated images.

6.6 Perspectives on synthesising a full object detection dataset

We have shown in Section 6.5 the viability of training a state-of-the art object detection system on fully-annotated synthetic images. The missing component is to incorporate classification into the framework. In principle, the R-CNN detector can be trained to do this simultaneously, however, our

CycleGANs, being a fully unsupervised model have no means for injecting this information. Clearly, we could use the R-CNN as a region proposal algorithm and classify regions with a separately trained classifier (such as in Section 5.3.2). Our preliminary attempts with this approach demonstrate we could never be worse off. However, the R-CNN system should be altogether more powerful if trained on a full annotation of object class and location. We could, alternatively, attempt to incorporate the weakly annotated ground truth (as utilised in Section 6.5) into the R-CNN fine-tuning to attempt to teach it about Raji cell classes. This has its own drawbacks, however, as this weakly supervised ground truth has already been shown to be problematic, and would entail a careful configuration of a large neural network (R-CNN).

In the following, we instead conceptualise an extension to our CycleGAN system that would be theoretically capable of providing both a perfect annotation encompassing both object localisation and class information. This extension takes the form of an additional discriminator for the content-to-style generator, that, in contrast to the PatchGAN discriminator, discriminates on regions of interest. Such a region of interest discriminator, D_{roi} compares fake cells localised by bounding boxes inside generated images with a pre-cropped "library" of real cells. The library can include class information for the individual crops, which can be combined in the fashion of a conditional GAN. The content image would also have to encode class information. Thus, the D_{roi} discriminator could provide a mechanism to enforce class information at the resolution of an RoI. For an implementation of this idea, see Appendix E.2.

6.6.1 Region of interest discrimination

We define an auxiliary discriminator for image-to-image GANs,

$$d_{roi} : X, B \to \{0, 1\} \tag{6.17}$$

for image domain X and bounding box domain B, with $d_{roi}(\mathbf{x}, b) = d(\rho(f(\mathbf{x}), b))$, for input image $\mathbf{x} \sim X$, bounding box $b \sim B$, and where we conceive of a neural network comprising of feature extraction layers f, classification layers d, and separated by RoIPool layer ρ. A RoIPool layer[5] (Girshick [2015]) generalises the MaxPooling layer and quantises regions of interest (RoIs) into a fixed size,

$$\rho : X, B \to \mathbf{y} \tag{6.18}$$

[5] See also RoIAlign layer (He et al. [2017])

for input tensor $\mathbf{x} \in \mathbb{R}^{1\times W\times H\times C}$, set of K bounding box tuples B, and tensor output $\mathbf{y} \in \mathbb{R}^{K\times w\times h\times C}$ for quantised dimensions $w < W$ and $h < H$. To clarify, the incoming tensor $\mathbf{x}$ is transformed into a "pseudo-batch" of K tensors for the K bounding boxes of B. Akin to the PatchGAN (see Section 5.2.1), the full discriminator output averages over the individual bounding boxes,

$$D_{roi} = \frac{1}{|\mathcal{B}(\mathbf{x})|} \sum_{b\in\mathcal{B}(\mathbf{x})} d_{roi}(\mathbf{x}, b) \tag{6.19}$$

where the operator $\mathcal{B}$ returns the set of bounding boxes of image $\mathbf{x}$. Of course, in our case, the bounding boxes are decided in advance by virtue of drawing the content images. We train this in a loss function,

$$\max_{G,F} \min_{D_X,D_Y,D_{roi}} \mathcal{L}_{CG} + \mathcal{L}_{GAN}(D_{roi}, G, X, Y) \tag{6.20}$$

where $\mathcal{L}_{CG}$ is the CycleGAN loss function introduced in Equation 6.16. The setup is given in Figure 6.13. The problem of obtaining a crop library will vary from problem to problem. In our case, however, we have direct access to such a library from the automatic procedure described in Section . Some basic proofs of concept of using the RoIPool for classification and GAN discrimination are given in Appendix E.2.

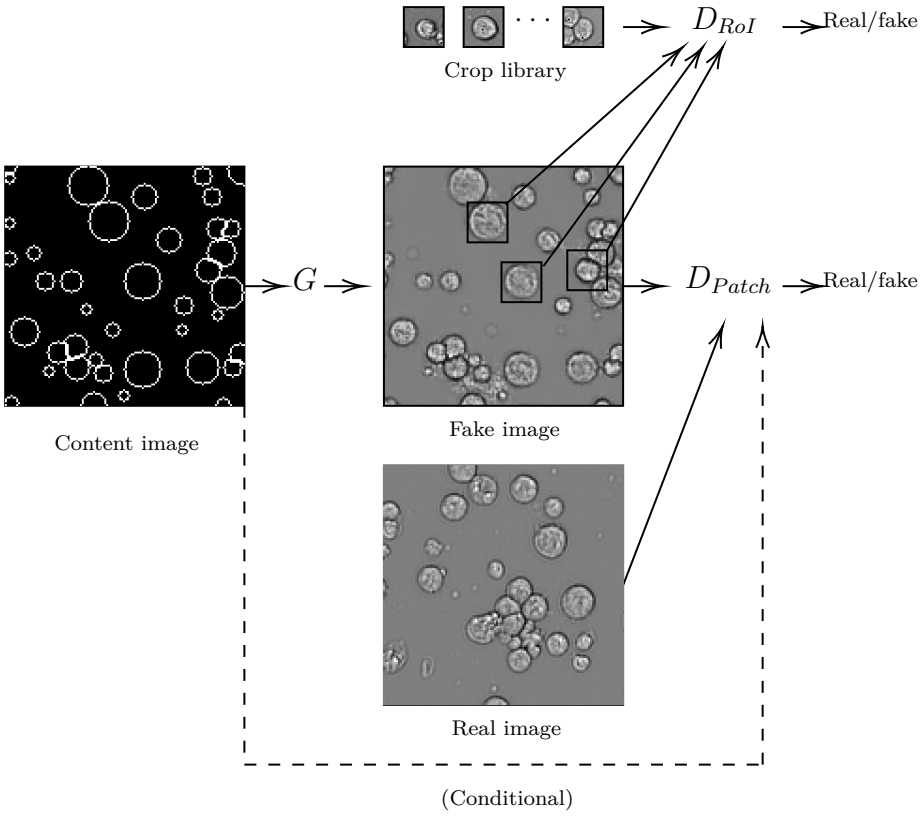


Figure 6.13: Conceptualisation of an extension to adversarial image-to-image networks with a region of interest discriminator D_{roi}. D_{roi} relies on a library of real crops to compare to cells synthesised by G in the prescribed regions of the content image.

Chapter 7

Conclusions

Computational phenotyping is an ascendant methodology for the extraction of information from biological matter. This thesis has explored a range of solutions for two problems in computational phenotyping. Despite the myriad overlaps, these problems ultimately sit on different axes of development. Part I explored ways of incorporating multiple biological models of different disease subtypes in a joint study. The molecular heterogeneity of these multiple cell lines led ultimately to a computational problem that we resolved with domain adaptation. Part II made attempts at fluorescence-free phenotyping by leveraging a dataset pairing phase contrast and fluorescence microscopy images. Part I can be considered as a stepping stone for computational phenotyping in the long road to precision medicine, and as such remains open-ended. On the other hand, Part II belongs to a more immediate trend for which it is easier to see an end, namely, that deep learning may soon revolutionise the microscopy tool set for conducting biological experiments.

In the following we first give a summary of the chapters and finally speculate about the future of high content screening.

7.1 Chapter summaries

Chapter 3 introduced the high content analysis pipeline and applied it to a drug and morphological screen datasets for multiple triple-negative breast cancer (TNBC) cell lines. We produced a series of use cases spanning each of univariate-, multivariate-, and machine learning-based analyses. While these were included partly as motivating examples, we were in particular able to highlight a systematic bias in a standard piece of analysis using causal inference formalisms. We were also able to observe a first instance of

the differential effects of drugs on cell lines in the form of viability, which we were additionally able to categorise by drug mechanism of action. We additionally saw how TNBC cell lines could be analysed in unison, and explored their phenotypic differences across two axes of measurement: firstly at the population level, as different molecular subtypes manifested distinct patterns of cell cycle and aberrant morphologies; secondly at the cellular level, as the fundamental wild type morphologies were easily distinguishable by a simple classifier. The sum of these insights were carried forward the following chapter.

In Chapter 4 we explored in great detail the construction of phenotypic profiles for charactersing drug effects. We reviewed and tested the most relevant existing approaches in a comparative study on mechanism of action (MOA) prediction. We then extended the methodology to a multi-cell-line drug screen, where we were able to show an autoencoder based on unsupervised domain adaptation could succeed in building domain-invariant features that outperformed baseline models in MOA prediction. We additionally found evidence to suggest that pooled multi-cell-line data provides more powerful representations for predicting the MOA of hold-out drugs than single-cell line datasets of the same size. In contrast with previous studies, we framed a use case for phenotypic profiling without tailoring the prediction problem to those MOAs with the most striking visual effects, yet confirmed MOA classification could be performed at well above a rate of random chance. This dataset has since been released in full to the scientific community, to our knowledge the first of its kind. In sum, this chapter validated a novel solution for handling the heterogeneity of multi-cell-line screens using deep and adversarial learning.

Chapter 5 compared two deep learning methodologies for quantifying CAR-T and Raji cell line cells in time lapse movies. With a setup providing a paired image dataset of transmitted light and fluorescent channels, the key barrier for training deep phenotypers is toppled. Each methodology explored a different use of fluorescence for automatically training a deep neural net. In doing so, the trained models progress towards replacing the need for fluorescence in the first place. The first approach, based on image-to-image translation networks, including a generative adversarial variant, showed excellent results in the fluorescent labeling of cell populations for mCherry and GFP markers. Though the path to full quantification of cells remains uncertain, this already works well as a visualisation tool. The second showed how a training set for object detection could be built, and that this is a viable alternative to fluorescence labeling, that moreover gets to the heart of the problem of phenotyping cells in time series. In a final result, we were able to reconcile the two methodologies and demonstrate the degree of agreement between them over a set of experimental replicates.

In the final Chapter 6, we sought a way to transcend the limitations of automatic ground truth assemblage encountered in Chapter 5. Here we found generative adversarial networks produce excellent results in the synthesis of individual cells, including when conditioned on a class indicator variable. From there, we were inspired to syntheisise full populations of cells, by formulating a style transfer problem, where style was grafted from a curated ground truth image to a content image of our choosing. The content image was constructed to emulate the properties of cells and cell populations. A feasibility study in classical style transfer demonstrated the validity of this concept. This was superseded by a learning approach to unpaired image-to-image translation, the CycleGAN, which conditioned one of its generators on the content image. Using this generator as an oracle for training data, we were able to train a state-of-the-art object detection system. In a final section, we sketched an extension to the CycleGAN so as to be able to specify class information in the content specification.

7.2 The future of high content screening

The future of high content screening lies plausibly in the data-driven domain of deep learning, to which so many areas of computer vision have yielded in recent years. Chapter 5 in particular showed how seamlessly deep convolutional networks can be assimilated into the screening process. These applications are arguably low-hanging fruit, however, and make an important yet singular contribution to screening process. A deeper question is how to improve the screening process altogether. Researchers such as Kraus et al. [2016], Kandaswamy et al. [2016], Godinez et al. [2017], and Sommer et al. [2017] have shown how the conventional pipeline can be partially or fully subsumed to a neural network optimised end-to-end. However, their impact has so far been rather superficial.

The main bottleneck, as always, is data. Most chapters in this thesis were about making intelligent use of a relatively limited data supply: first by domain adaptation in Chapter 4, then by leveraging fluorescence as annotation in 5, and finally augmenting the available data with generative models and style transfer in Chapter 6. In the long run, however, there is no replacement for a vastly increased data supply. After decades spent dormant, the power of deep learning was finally unleashed by large, general-purpose image datasets like ImageNet (Russakovsky et al. [2015]). A highly valuable discovery was that upon training neural networks a first time, they could redeployed to exceed the state-of-the-art of yesteryear on virtually any vision problem. This is the power of transfer learning and fine-tuning.

High content screens become different very quickly, however. For one, the

number and kind of fluorescent channels varies from screen to screen, as they are chosen for different sub-cellular regions of interest. As a result, intensity colocalisation across fluorescent channels generally have a very different meaning from screen to screen. There is therefore significant interest in models that perform across assays and microscopy modalities (for example, Hollandi et al. [2019] for cell segmentation).

While sizeable screening image sets exist for drug MOA prediction, their generality remains dubious. It remains that by using the conventional approach, (that is, classical image analysis features), the simplest imaginable classifier can bulldoze the benchmark MOA prediction task (Singh et al. [2014]). Our own drug screen (Part I) likely provides a more realistic picture, with drugs. Our results, though far better than random, are considerably more modest. There is no doubt, however, that if this pilot screen were followed up with a full primary screen, with more cell lines and a broader drug panel, the power of our approach would be greatly enhanced. A full primary screen would likely mean a 1-2 order of magnitude increase in the data volume available for learning (up to around one terrabyte).

But what if the data volume were increased by another few orders of magnitude again?

Chapter 4 confirmed the rationality of multi-cell-line high content screens.

Appendix A

Supplementary figures

A.1 Confusion matrix Chapter 2

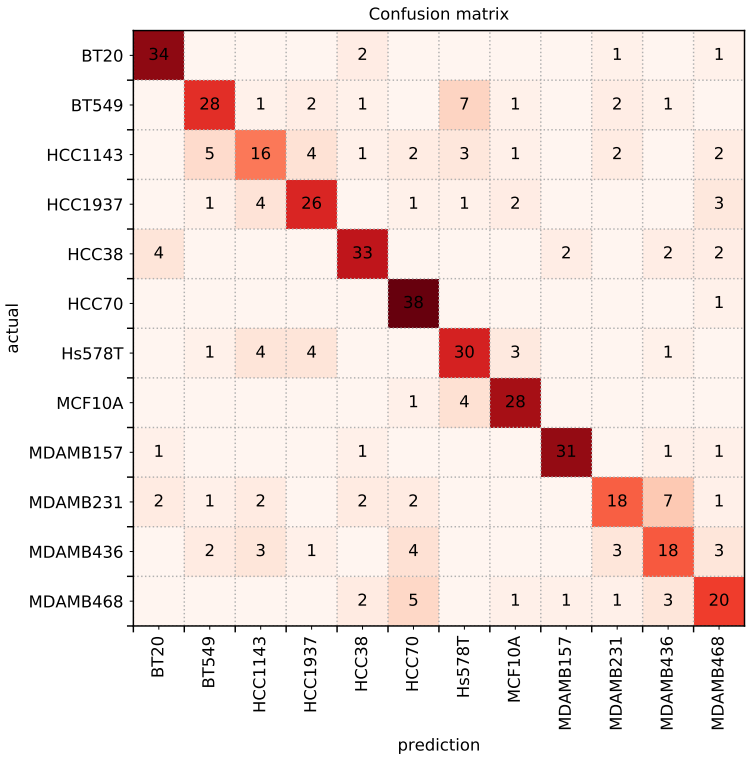


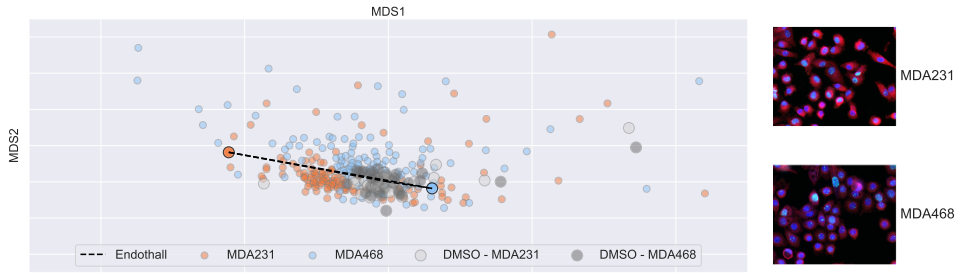
Figure A.1: Confusion matrix for a random forest classifier classifying cells from 12 TNBC cell lines.

A.2 Differential drug effects Chapter 3

A.3 Biological phenomena Chapter 4

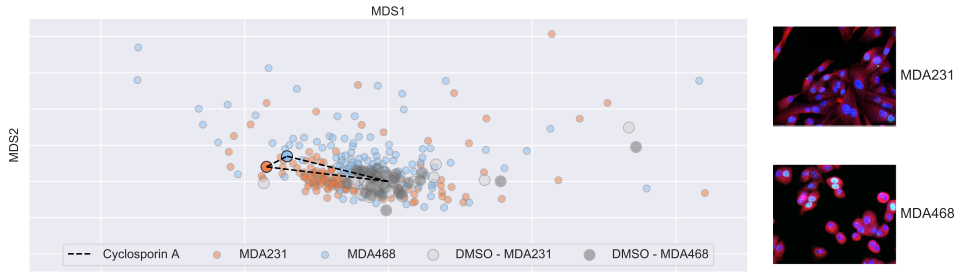
A.4 Blob detection Chapter 4

A.5 Image processing pipeline Chapter 4

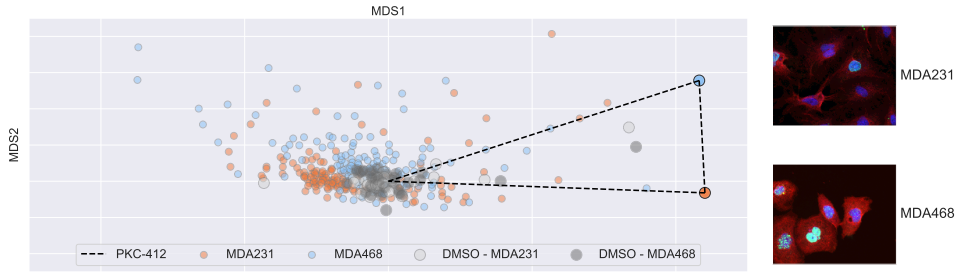


(a) Endothall takes effect in cell line MDA231 only.

(b) CL-82198 takes effect in cell line MDA468 only.



(c) Cyclosporin A takes a similar in both cell lines.

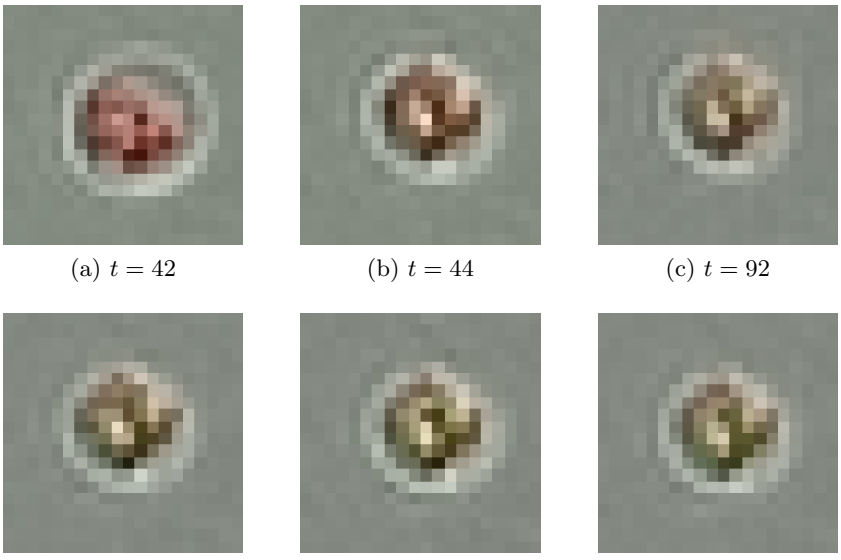


(d) PKC-412 takes differential effects in the two cell lines.

Figure A.2: MDS plots of each category of drug effect. The distances between the profiles are plotted as a line, as well as the respective distances to the centroid (origin).

(a) $t = 42$ (b) $t = 44$ (c) $t = 92$

Figure A.3: An application of a neural algorithm of artistic style to phase contrast images of CAR-T cells (a) and a style source image (b) to synthesise a new image combining their characteristics in a "simulated" image (c).



(a) $t = 42$ (b) $t = 44$ (c) $t = 92$

(d) $t = 42$ (e) $t = 44$ (f) $t = 92$

Figure A.4: An application of a neural algorithm of artistic style to phase contrast images of CAR-T cells (a) and a style source image (b) to synthesise a new image combining their characteristics in a "simulated" image (c).

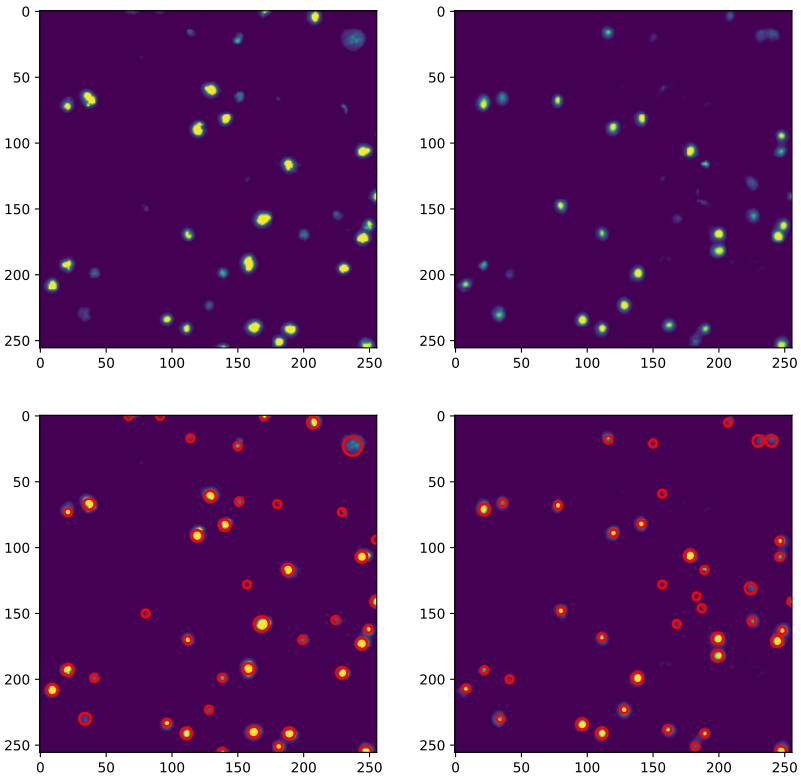
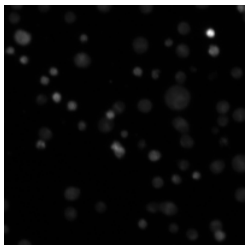
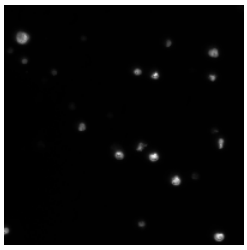


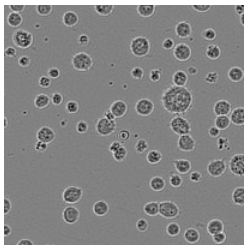
Figure A.5: True positives: 31, False positives: 9, False negatives: 11



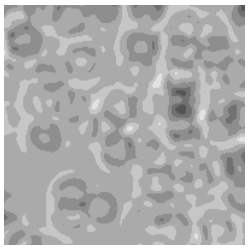
(a) mCherry and GFP fluorescence (left and center) and phase contrast signals (right) at $t = 0$



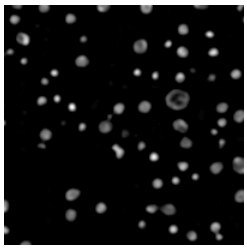
(b) mCherry and GFP fluorescence (left and center) and phase contrast signals (right) at $t = 70$



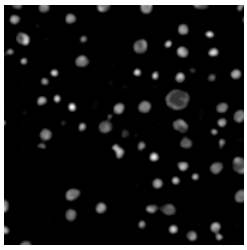
(c) mCherry and GFP fluorescence (left and center) and phase contrast signals (right) at $t = 70$



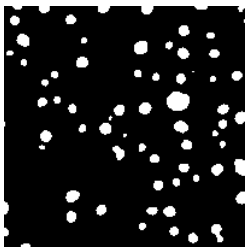
(d) mCherry and GFP fluorescence (left and center) and phase contrast signals (right) at $t = 0$



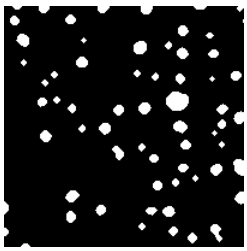
(e) mCherry and GFP fluorescence (left and center) and phase contrast signals (right) at $t = 70$



(f) mCherry and GFP fluorescence (left and center) and phase contrast signals (right) at $t = 70$



(g) mCherry and GFP fluorescence (left and center) and phase contrast signals (right) at $t = 0$



(h) mCherry and GFP fluorescence (left and center) and phase contrast signals (right) at $t = 70$

(i) mCherry and GFP fluorescence (left and center) and phase contrast signals (right) at $t = 70$

Figure A.6: Comparison of crops of the same cell population at times (a) $t = 0$ and (b) $t = 70$. Mitosis has brought about agglomerations of cells that are difficult to separate. The problem is exacerbated by a loss of fluorescence "signal quenching" over time.

Appendix B

Drug screen plate map

A - 01: DMSO, A - 02: DMSO, A - 03: DMSO, A - 04: DMSO, A - 05: DMSO, A - 06: DMSO, A - 07: DMSO, A - 08: DMSO, A - 09: DMSO, A - 10: DMSO, A - 11: DMSO, A - 12: DMSO, A - 13: DMSO, A - 14: DMSO, A - 15: DMSO, A - 16: DMSO, A - 17: DMSO, A - 18: DMSO, A - 19: DMSO, A - 20: DMSO, A - 21: DMSO, A - 22: DMSO, A - 23: DMSO, A - 24: DMSO, B - 01: Calpain Inhibitor II, B - 03: Calpeptin, B - 05: E-64, B - 07: E-64-c, B - 09: E-64-d, B - 11: MDL-28170(Z-VF-CHO), B - 13: Z-Leu-Leu-CHO, B - 15: CA-074, B - 17: CA-074-Me, B - 19: Napsul-Ile-Trp-CHO, B - 21: Pepstatin A, B - 23: 3,4-Dichloroisocoumarin, C - 01: PD-98059, C - 02: Cantharidic acid, C - 03: U-0126, C - 04: Cantharidin, C - 05: SB-203580, C - 06: Endothall, C - 07: H-7, C - 08: Benzylphosphonic acid, C - 09: H-9, C - 10: L-p-Bromotetramisole oxalate, C - 11: Staurosporine, C - 12: RK-682, C - 13: AG-494, C - 14: RWJ-60475, C - 15: AG-825, C - 16: RWJ-60475 (AM)3, C - 17: Lavendustin A, C - 18: Levamisole HCl, C - 19: RG-14620, C - 20: Tetramisole HCl, C - 21: Tyrphostin 23, C - 22: Cypermethrin, C - 23: Tyrphostin 25, C - 24: Deltamethrin, D - 01: Actinonin, D - 03: CL-82198, D - 05: Epigallocatechin gallate, D - 07: GM6001, D - 09: NNGH (BML-205), D - 11: Gliotoxin, D - 13: MG-132, D - 15: AAF-CMK, D - 17: Arphamenine A, D - 19: Bestatin (Ubenimex), D - 21: Boc-GVV-CHO, D - 23: Captopril, E - 01: Tyrphostin 46, E - 02: Fenvalerate, E - 03: Tyrphostin 47, E - 04: Tyrfphostin 8, E - 05: Tyrphostin 51, E - 06: CinnGel, E - 07: Tyrphostin 1, E - 08: CinnGel 2 Me, E - 09: Tyrphostin AG 1288, E - 10: BN-82002, E - 11: Tyrphostin AG 1478, E - 12: Shikonin, E - 13: Tyrphostin AG 1295, E - 14: NSC-663284, E - 15: Tyrphostin 9, E - 16: Cyclosporin A, E - 17: HNMPA (Hydroxy-2-naphthalenylmethylphosphonic acid), E - 18: Pentamidine, E - 19: PKC-412, E - 20: BVT-948, E - 21: Piceatannol, E - 22: B4-Rhodanine, E - 23: PP1, E - 24: BML-268, F - 01: Elastatinal, F - 03: Phosphoramidon, F - 05: PPACK, F - 07: Z-Prolyl-Prolinal, F - 09:

Thiorphan (DL), F - 11: TLCK, F - 13: TPCK, F - 15: BML-244, F - 17: Fumagillin, F - 19: 5-nitroisatin, F - 21: P32/98, F - 23: Nafamostat mesylate, G - 01: AG-490, G - 02: 9,10-Phenanthrenequinone, G - 03: AG-126, G - 04: BML-260, G - 05: AG-370, G - 06: PD-144795, G - 07: AG-879, G - 08: BML-267, G - 09: LY 294002, G - 10: BML-267 Ester, G - 11: Wortmannin, G - 12: OBA, G - 13: GF 109203X, G - 14: OBA Ester, G - 15: Hypericin, G - 16: Gossypol, G - 17: Ro 31-8220, G - 18: Alendronate, G - 19: Sphingosine, G - 21: H-89, G - 23: H-8 , H - 01: Gabexate mesylate, H - 03: Dec-RVKR-CMK, H - 05: Z-PLG-NHOH, H - 07: Z-FA-FMK, H - 09: Z-VAD-FMK, H - 11: Ebelactone B, H - 13: Apstatin , H - 15: Argatroban , H - 17: URB-597 , H - 19: 4-Amino-D, L-benzylsuccinic acid , H - 21: N-Ethylmaleimide , H - 23: Acivicin, I - 01: HA-1004, I - 03: HA-1077, I - 05: HDBA (2-Hydroxy-5-(2,5-dihydroxybenzylamino)benzoic acid), I - 07: KN-62, I - 09: KN-93, I - 11: ML-7, I - 13: ML-9, I - 15: 2-Aminopurine, I - 17: N9-Isopropyl-olomoucine, I - 19: Olomoucine, I - 21: iso-Olomoucine, I - 23: Roscovitine, J - 01: Tazobactam , J - 03: Lisinopril, J - 05: Enalapril, J - 07: Cilastatin, J - 09: PKSI-527, K - 01: 5-Iodotubercidin, K - 03: LFM-A13, K - 05: SB-202190, K - 07: PP2, K - 09: ZM 336372, K - 11: SU 4312, K - 13: AG-1296, K - 15: GW 5074, K - 17: Palmitoyl-DL-carnitine Cl, K - 19: Rottlerin, K - 21: Genistein, K - 23: Daidzein, M - 01: Erbstatin analog, M - 03: Quercetin dihydrate, M - 05: SU1498, M - 07: ZM 449829, M - 09: BAY 11-7082, M - 11: DRB (5,6-Dichloro-1-b-D-ribofuranosylbenzimidazole), M - 13: HBDDE (2,2',3,3',4,4'-Hexahydroxy-1,1'-biphenyl-6,6'-dimethanol dimethyl ether), M - 15: SP 600125, M - 17: Indirubin , M - 19: Indirubin-3'-monoxime, M - 21: Y-27632, M - 23: Kenpaullone, O - 01: Terreic acid, O - 03: Triciribine, O - 05: BML-257, O - 07: SC-514, O - 09: BML-259, O - 11: Apigenin, O - 13: BML-265 (Erlotinib analog), O - 15: Rapamycin, P - 01: DMSO, P - 03: DMSO, P - 05: DMSO, P - 07: DMSO, P - 09: DMSO, P - 11: DMSO, P - 13: DMSO, P - 15: DMSO, P - 17: DMSO, P - 19: DMSO, P - 21: DMSO, P - 22: Olaparib, P - 23: DMSO, P - 24: Cisplatine

Appendix C

Glossary of neural network architectures

C.1 Fully-connected GAN

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_42 (Dense)             (None, 256)               25856
_________________________________________________________________
batch_normalization_17 (Batc (None, 256)               1024
_________________________________________________________________
activation_17 (Activation)   (None, 256)               0
_________________________________________________________________
dense_43 (Dense)             (None, 784)               201488
_________________________________________________________________
batch_normalization_18 (Batc (None, 784)               3136
_________________________________________________________________
activation_18 (Activation)   (None, 784)               0
_________________________________________________________________
dense_44 (Dense)             (None, 576)               452160
_________________________________________________________________
reshape_9 (Reshape)          (None, 24, 24, 1)         0
=================================================================
Total params: 683,664
Trainable params: 681,584
Non-trainable params: 2,080
_________________________________________________________________

_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_45 (Dense)             (None, 512)               295424
_________________________________________________________________
leaky_re_lu_26 (LeakyReLU)   (None, 512)               0
_________________________________________________________________
dense_46 (Dense)             (None, 256)               131328
_________________________________________________________________
leaky_re_lu_27 (LeakyReLU)   (None, 256)               0
_________________________________________________________________
dense_47 (Dense)             (None, 1)                 257
=================================================================
Total params: 427,009
Trainable params: 427,009
Non-trainable params: 0
_________________________________________________________________
```

C.2 Deep convolutional GAN

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_16 (Dense)             (None, 4608)              465408
_________________________________________________________________
batch_normalization_7 (Batch (None, 4608)              18432
_________________________________________________________________
activation_7 (Activation)    (None, 4608)              0
_________________________________________________________________
reshape_4 (Reshape)          (None, 6, 6, 128)         0
_________________________________________________________________
up_sampling2d_7 (UpSampling2 (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 12, 12, 64)        204864
_________________________________________________________________
batch_normalization_8 (Batch (None, 12, 12, 64)        256
_________________________________________________________________
activation_8 (Activation)    (None, 12, 12, 64)        0
_________________________________________________________________
up_sampling2d_8 (UpSampling2 (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 24, 24, 1)         1601
=================================================================
```

```
Total params: 690,561
Trainable params: 681,217
Non-trainable params: 9,344
_________________________________________________________________

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_18 (Dense)             (None, 256)               25856
_________________________________________________________________
batch_normalization_9 (Batch (None, 256)               1024
_________________________________________________________________
activation_9 (Activation)    (None, 256)               0
_________________________________________________________________
dense_19 (Dense)             (None, 784)               201488
_________________________________________________________________
batch_normalization_10 (Batc (None, 784)               3136
_________________________________________________________________
activation_10 (Activation)   (None, 784)               0
_________________________________________________________________
dense_20 (Dense)             (None, 576)               452160
_________________________________________________________________
reshape_5 (Reshape)          (None, 24, 24, 1)         0
=================================================================
Total params: 683,664
Trainable params: 681,584
Non-trainable params: 2,080
_________________________________________________________________
```

C.3 F-Net

```
_____________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
=============================================================================================
input_2 (InputLayer)             (None, 256, 256, 1)   0
_____________________________________________________________________________________________
conv2d_15 (Conv2D)               (None, 128, 128, 64)  1088        input_2[0][0]
_____________________________________________________________________________________________
leaky_re_lu_8 (LeakyReLU)        (None, 128, 128, 64)  0           conv2d_15[0][0]
_____________________________________________________________________________________________
conv2d_16 (Conv2D)               (None, 64, 64, 128)   131200      leaky_re_lu_8[0][0]
_____________________________________________________________________________________________
leaky_re_lu_9 (LeakyReLU)        (None, 64, 64, 128)   0           conv2d_16[0][0]
```

```
_________________________________________________________________
batch_normalization_13 (BatchNo (None, 64, 64, 128)  512         leaky_re_lu_9[
_________________________________________________________________
conv2d_17 (Conv2D)              (None, 32, 32, 256)  524544      batch_normaliz
_________________________________________________________________
leaky_re_lu_10 (LeakyReLU)      (None, 32, 32, 256)  0           conv2d_17[0][0
_________________________________________________________________
batch_normalization_14 (BatchNo (None, 32, 32, 256)  1024        leaky_re_lu_10
_________________________________________________________________
conv2d_18 (Conv2D)              (None, 16, 16, 512)  2097664     batch_normaliz
_________________________________________________________________
leaky_re_lu_11 (LeakyReLU)      (None, 16, 16, 512)  0           conv2d_18[0][0
_________________________________________________________________
batch_normalization_15 (BatchNo (None, 16, 16, 512)  2048        leaky_re_lu_11
_________________________________________________________________
conv2d_19 (Conv2D)              (None, 8, 8, 512)    4194816     batch_normaliz
_________________________________________________________________
leaky_re_lu_12 (LeakyReLU)      (None, 8, 8, 512)    0           conv2d_19[0][0
_________________________________________________________________
batch_normalization_16 (BatchNo (None, 8, 8, 512)    2048        leaky_re_lu_12
_________________________________________________________________
conv2d_20 (Conv2D)              (None, 4, 4, 512)    4194816     batch_normaliz
_________________________________________________________________
leaky_re_lu_13 (LeakyReLU)      (None, 4, 4, 512)    0           conv2d_20[0][0
_________________________________________________________________
batch_normalization_17 (BatchNo (None, 4, 4, 512)    2048        leaky_re_lu_13
_________________________________________________________________
conv2d_21 (Conv2D)              (None, 2, 2, 512)    4194816     batch_normaliz
_________________________________________________________________
leaky_re_lu_14 (LeakyReLU)      (None, 2, 2, 512)    0           conv2d_21[0][0
_________________________________________________________________
batch_normalization_18 (BatchNo (None, 2, 2, 512)    2048        leaky_re_lu_14
_________________________________________________________________
up_sampling2d_8 (UpSampling2D)  (None, 4, 4, 512)    0           batch_normaliz
_________________________________________________________________
conv2d_22 (Conv2D)              (None, 4, 4, 512)    4194816     up_sampling2d_
_________________________________________________________________
batch_normalization_19 (BatchNo (None, 4, 4, 512)    2048        conv2d_22[0][0
_________________________________________________________________
concatenate_7 (Concatenate)     (None, 4, 4, 1024)   0           batch_normaliz
                                                                 batch_normaliz
_________________________________________________________________
up_sampling2d_9 (UpSampling2D)  (None, 8, 8, 1024)   0           concatenate_7[
_________________________________________________________________
```

```
conv2d_23 (Conv2D)              (None, 8, 8, 512)    8389120     up_sampling2d_9[0][0]
__________________________________________________________________________________________
batch_normalization_20 (BatchNo (None, 8, 8, 512)    2048        conv2d_23[0][0]
__________________________________________________________________________________________
concatenate_8 (Concatenate)     (None, 8, 8, 1024)   0           batch_normalization_20|
                                                                 batch_normalization_16|
__________________________________________________________________________________________
up_sampling2d_10 (UpSampling2D) (None, 16, 16, 1024) 0           concatenate_8[0][0]
__________________________________________________________________________________________
conv2d_24 (Conv2D)              (None, 16, 16, 512)  8389120     up_sampling2d_10[0][0]
__________________________________________________________________________________________
batch_normalization_21 (BatchNo (None, 16, 16, 512)  2048        conv2d_24[0][0]
__________________________________________________________________________________________
concatenate_9 (Concatenate)     (None, 16, 16, 1024) 0           batch_normalization_21|
                                                                 batch_normalization_15|
__________________________________________________________________________________________
up_sampling2d_11 (UpSampling2D) (None, 32, 32, 1024) 0           concatenate_9[0][0]
__________________________________________________________________________________________
conv2d_25 (Conv2D)              (None, 32, 32, 256)  4194560     up_sampling2d_11[0][0]
__________________________________________________________________________________________
batch_normalization_22 (BatchNo (None, 32, 32, 256)  1024        conv2d_25[0][0]
__________________________________________________________________________________________
concatenate_10 (Concatenate)    (None, 32, 32, 512)  0           batch_normalization_22|
                                                                 batch_normalization_14|
__________________________________________________________________________________________
up_sampling2d_12 (UpSampling2D) (None, 64, 64, 512)  0           concatenate_10[0][0]
__________________________________________________________________________________________
conv2d_26 (Conv2D)              (None, 64, 64, 128)  1048704     up_sampling2d_12[0][0]
__________________________________________________________________________________________
batch_normalization_23 (BatchNo (None, 64, 64, 128)  512         conv2d_26[0][0]
__________________________________________________________________________________________
concatenate_11 (Concatenate)    (None, 64, 64, 256)  0           batch_normalization_23|
                                                                 batch_normalization_13|
__________________________________________________________________________________________
up_sampling2d_13 (UpSampling2D) (None, 128, 128, 256 0           concatenate_11[0][0]
__________________________________________________________________________________________
conv2d_27 (Conv2D)              (None, 128, 128, 64) 262208      up_sampling2d_13[0][0]
__________________________________________________________________________________________
batch_normalization_24 (BatchNo (None, 128, 128, 64) 256         conv2d_27[0][0]
__________________________________________________________________________________________
concatenate_12 (Concatenate)    (None, 128, 128, 128 0           batch_normalization_24|
                                                                 leaky_re_lu_8[0][0]
__________________________________________________________________________________________
up_sampling2d_14 (UpSampling2D) (None, 256, 256, 128 0           concatenate_12[0][0]
```

```
_________________________________________________________________________________
conv2d_28 (Conv2D)              (None, 256, 256, 1)  2049        up_sampling2d_
=================================================================================
Total params: 41,837,185
Trainable params: 41,828,353
Non-trainable params: 8,832
_________________________________________________________________________________
```

C.4 PatchGAN

```
_________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
=================================================================================
input_7 (InputLayer)            (None, 256, 256, 1)  0
_________________________________________________________________________________
input_8 (InputLayer)            (None, 256, 256, 1)  0
_________________________________________________________________________________
concatenate_15 (Concatenate)    (None, 256, 256, 2)  0           input_7[0][0]
                                                                 input_8[0][0]
_________________________________________________________________________________
conv2d_39 (Conv2D)              (None, 128, 128, 64) 2112        concatenate_15
_________________________________________________________________________________
leaky_re_lu_23 (LeakyReLU)      (None, 128, 128, 64) 0           conv2d_39[0][0]
_________________________________________________________________________________
conv2d_40 (Conv2D)              (None, 64, 64, 128)  131200      leaky_re_lu_23
_________________________________________________________________________________
leaky_re_lu_24 (LeakyReLU)      (None, 64, 64, 128)  0           conv2d_40[0][0]
_________________________________________________________________________________
batch_normalization_31 (BatchNo (None, 64, 64, 128)  512         leaky_re_lu_24
_________________________________________________________________________________
conv2d_41 (Conv2D)              (None, 32, 32, 256)  524544      batch_normaliz
_________________________________________________________________________________
leaky_re_lu_25 (LeakyReLU)      (None, 32, 32, 256)  0           conv2d_41[0][0]
_________________________________________________________________________________
batch_normalization_32 (BatchNo (None, 32, 32, 256)  1024        leaky_re_lu_25
_________________________________________________________________________________
conv2d_42 (Conv2D)              (None, 16, 16, 512)  2097664     batch_normaliz
_________________________________________________________________________________
leaky_re_lu_26 (LeakyReLU)      (None, 16, 16, 512)  0           conv2d_42[0][0]
_________________________________________________________________________________
batch_normalization_33 (BatchNo (None, 16, 16, 512)  2048        leaky_re_lu_26
_________________________________________________________________________________
```

```
conv2d_43 (Conv2D)              (None, 16, 16, 1)    8193        batch_normalization_33|
==========================================================================================
Total params: 2,767,297
Trainable params: 2,765,505
Non-trainable params: 1,792
------------------------------------------------------------------------------------------
```

Appendix D

Analysing double-strand breaks in cultured cells for drug screening applications by causual inference

The contents of this appendix were published in the Proceedings of the 14th IEEE International Symposium on Biomedical Imaging (ISBI). Washington D.C., United State of America. April, 2018.

Double strand breaks (DSB) are a hallmark of DNA damage and genetic instability, which are important features of cancer cells. In addition, repair of DSBs provide interesting therapeutic targets. Fluorescence microscopy allows us to visualise DSBs in cells using a dedicated fluorescent marker, which is therefore an informative readout for drug screening applications. We therefore need robust methods in image analysis and statistical analysis to quantify DSBs in single cells and thereby to assess the drug effect with respect to the related pathways. The contribution of this paper is twofold: first, we compare different DSB quantification schemes; and second we provide a sound statistical framework based on causal inference in order to detect drugs acting directly on DSBs. In particular this second aspect is so far notoriously neglected in the field, even though it is essential for the specific assignment of the drug effect.

D.1 Introduction

Dysfunction of the DNA repair mechanisms is a major hallmark of cancer. Monitoring DNA damage by fluorescently labeling double strand breaks (DSB) in cells is therefore an important readout in drug screening of cancer cell lines. DSBs occur when both strands of the DNA double helix are broken. DSBs have various causes, for example, cytotoxic radiation, or DNA replication over an existing single-strand break. Despite the natural DNA repair mechanisms of the cell, DSBs can be irreparable, leading ultimately to apoptosis, or hazardous DNA rearrangements. As such, DSBs are an interesting property to assess when analysing the effects of small compounds upon cancer cells.

The context of this research is a drug screen in triple-negative breast cancer (TNBC), a type of cancer that is characterised by the absence of genetic markers that are common targets for cancer therapies. The difficulty of treatment of TNBC make the search for effective new treatments particularly important. It must be noted however that DSB quantification is not limited to this application; it is a general marker often used in cancer research, and in particular in screening applications.

In Section D.2 we describe the experimental setup of the drug screen. In Section D.3 we compare three approaches to quantifying DSBs over the imaged cell populations. Then, we show that analysis of DSB numbers can be misleading for two reasons: first, the number of DSBs may depend on the cell cycle and drugs assigned to affect DSBs might actually only affect the cell cycle. Second, in drug screening, severely perturbed cells–such as mitotic or dead cells–are typically washed away prior to image acquisition. Such a *loss to followup* introduces a potential selection bias when performing significance tests. Both aspects have been neglected so far in the drug screening literature (Avondoglio et al. [2009], Garcia-Canton et al. [2013]). We use the tools of causal inference to probe for such systematic bias in our analysis in Section D.4. A comparison of the DSB quantifiers used and the analytic corrections are presented in Section D.5. Closing remarks are given in Section D.6.

D.2 Experimental setup

In a pilot study on multiple TNBC cell lines, we assembled 168 drugs, including two positive controls, for a drug screen at high concentration ($10\mu M$), alongside replicated negative controls of neutral agent *dimethyl sulfoxide* (DMSO) and untreated cells. The wells were seeded with a controlled 1000 and 1250 cells for MDA231 and MDA468 cell lines respectively, on sepa-

rate 384-well microplates. The drugs were administered identically on both plates. Following hibernation, the cells were fixed, washed, and stained with four fluorescent markers. Fluorescence microscopy (20× widefield with deconvolution image restoration) taken at four fields in each well produced two experimental datasets of 1536 multiplexed images apiece.

Four fluorescent stains were used for our microscopy: DAPI, Cyanine 3 (Cy3), Cyanine 5 (Cy5), and FITC. Of interest to our analysis are DAPI and Cy3, fluorescent markers for DNA and DSBs, respectively.

The final, washing step evacuates all unfastened cells from the well, such as mitotic or dead cells, as a consequence of perturbation or otherwise. The array of perturbations show a range of effects on cell mortality, from no apparent effect to a near or complete elimination of cells.

D.3 Approaches to measuring double-strand breaks

While biochemical techniques exist for DSB quantification (Chailleux et al. [2014]), obtaining faithful counts of DSBs directly from bioimages is problematic as the correspondence between DSBs and the recorded signal is unclear. Our strategy is therefore to extract a feature that has a high chance of being proportional to the amount of DSBs. The Cy3 signal typically manifests as a nebulous cloud, with intermittent peaks or spots of higher intensity. Samples are given in Figure D.1. DSBs may be underrepresented by a simple spot count, as high intensity spots may correspond with multiple, localised breaks. On the other hand, DSBs may be overrepresented by a noisy Cy3 channel. We here compare three approaches to quantifying double-strand breaks. The approaches give correlated readouts, in particular between spot count and granulometry (Figure D.2) with $\rho = 0.84$ over the entire plate, and it seems that any will serve as a reasonable proxy to the true number of DSBs. Granulometry and spot density correlate with average intensity at $\rho = 0.44$ and $\rho = 0.42$ respectively.

In general, the features differ most in the rare cases when the marker is saturated and nothing stands out to be counted as a spot, yet the average intensity is artificially high (staining artifacts). We therefore give preference to spot-related features. The computation was performed by Cell Cognition (Held et al. [2010]), an open-source tool for the visualisation and analysis of HCS assays. As part of a standard computational pipeline, individual cell nuclei were first segmented by a combination of filtering and local thresholding (Held et al. [2010]). Touching nuclei are split as described in Naylor et al. [2017], using morphological local contrast dynamics (morphological dynamics) giving superior results to the traditional approach of filtering the

Euclidean distance map (data not shown). DSB analysis was then performed for each individual nucleus.

D.3.1 Counting spots with diameter openings

One approach to measuring DSBs is to count individual spots. For this, we used *diameter openings* (Walter et al. [2007]), a technique based on a similar flooding technique to the watershed algorithm, with the notable difference that no separation of regions is built and that the flooding stops as soon as the maximal extension of the region exceeds a user-defined diameter λ. Mathematically, the operator can be written as:

$$\begin{aligned} \left[\gamma^{\circ}_{\lambda}(f)\right](x) &= \sup\left\{s \leq f(x) \mid \alpha\left(C_x\left[X^{+}_{s}(f)\right]\right) \geq \lambda\right\} \\ &= \bigcup\{\gamma_B(f) \mid \alpha(B) \geq \lambda\} \end{aligned} \tag{D.1}$$

where $X^{+}_{s}(f)$ is the set of all pixels with with $f(x) \geq s$, $C_x[A]$ is the connected component of set A containing point x and $\alpha(C_x)$ its maximal extension. Equation D.1 shows that this can also be written as the supremum of *all* morphological openings $\gamma_{B_i}(f)$ with structuring elements B_i whose diameter greater or equal to λ.

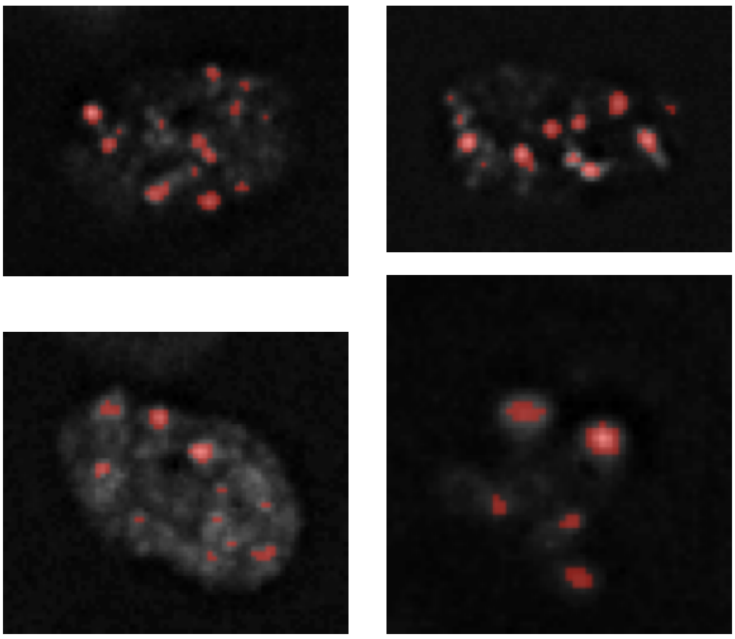


Figure D.1: Examples of spots (red) on cell nuclei detected with diameter openings on the Cy3 channel (grey).

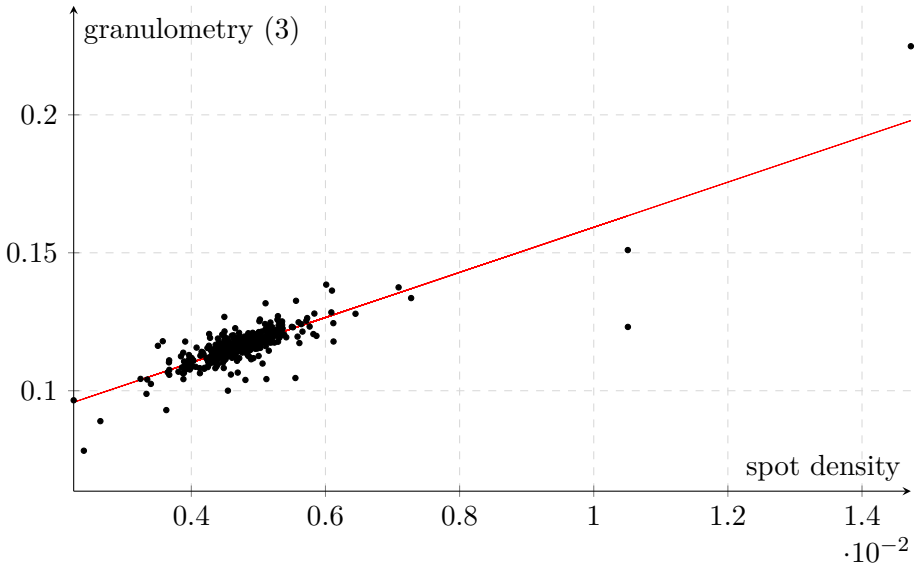


Figure D.2: Granulometry and spot density readouts show high correlation on average over all wells.

Building the difference to the original image $f - \gamma_\lambda^\circ(f)$ and a simple threshold operation thus allows one to extract all bright details with a maximal extension smaller than a certain value. The tunable parameters of the algorithm are the diameter of the structuring element and the value of the intensity threshold[1]. Nuclei happen to have different sizes and we thus normalise the spot count into *spot density* by dividing by the nuclear area.

D.3.2 Granulometry-based features

An alternative strategy to get a proxy of DSB numbers without relying on hard segmentation, is to use *morphological granulometries*, i.e. the sum of the difference between opened versions of the initial image.

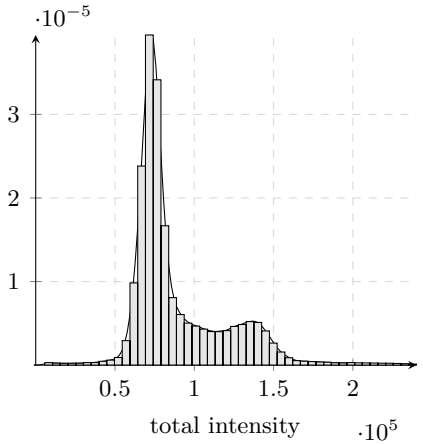
$$\sum_{x \in S_k} \gamma_{B_i} f(x) - \gamma_{B_{i+1}} f(x) \tag{D.2}$$

where the structuring elements B_i fulfill the condition $B_i \subset B_{i+1}$ and S_k is the k-th cell.

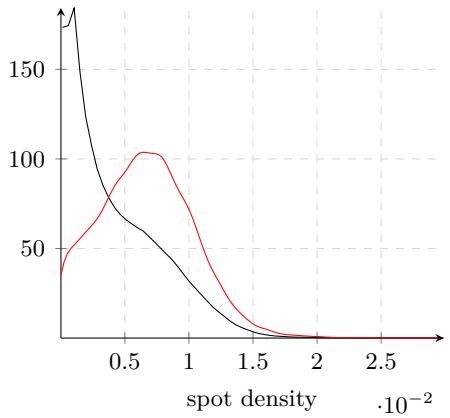
D.3.3 Average intensity

We also hypothesised that an accumulation of γ-H2AX can be seen as an indicator of DNA breaks without a detectable organisation into spots, in

[1] These were manually tuned to a diameter of 5 pixels, and threshold of 8.



(a) mCherry and GFP fluorescence (left and center) and phase contrast signals (right) at $t = 0$



(b) mCherry and GFP fluorescence (left and center) and phase contrast signals (right) at $t = 70$

Figure D.3: Distribution of cell total intensities on the DAPI channel for MDA231 cell line (a). The bimodal distribution is a consequence of the growth sustained between the G_1 (black) and G_2 (red) phases of the cell cycle. A suitable threshold of DAPI intensity significant divergence in the DSB distributions of the respective groups on the Cy3 channel (b).

particular in the case of very high densities. We therefore added the average intensity over the region of interest as a feature.

$$\frac{1}{\#S_k} \sum_{x \in S_k} f(x) \tag{D.3}$$

D.4 Analysis

The aim of our analysis is to compare DSB levels from drug perturbations to levels in the negative control, DMSO. We used the Kolmogorov-Smirnov (K-S) test as the basis of comparison with significance declared at the 0.01 level.

As can be seen in Figure D.3a, there is a bimodal distribution in DAPI intensity considered over all cells, corresponding to cell cycle phases (pre- and post-DNA synthesis). The higher frequency of low intensities corresponds to the longer G_1 phase and more cells will be fixed in this phase. Secondly, the DNA content in this phase is constant, whereas the DNA content in S-phase is somewhere between G_1 and G_2. Third, there are many aberrant morphologies, for which the DNA content is larger than the one of normal G_1 cells, but the exact value is not clearly determined. All of these elements

lead to a wide distribution of DAPI intensities for all nuclei that are not in G_1. We can roughly estimate a suitable threshold between the two classes to create high and low intensity groups. Visualising the respective spot densities for these groups (Figure D.3b) reveals a shift in DSB distribution between the groups.

D.4.1 Causal considerations

Causal inference (Pearl et al. [2009]) is a framework for conducting statistical analysis that enables reasoning about causal factors in an experimental setting, so as to determine a relationship between a potential cause or *treatment* and a potential effect or *outcome*. Such an analysis is necessarily first endowed with a directed, acyclic causal graph, such as Figure D.4, by a domain expert. The causal graph postulates a feasible causal relation, respecting the otherwise impossible task of establishing causations from statistical data alone (Pearl et al. [2009]). In particular, causal models help to identify biases and necessary adjustments in analysis.

One aspect we looked into was the simultaneous effect of perturbations on cell death and DSBs, themselves potentially a cause of cell death. Due to the washing step in the experimental protocol, our observations are confined to those cells surviving the cytotoxic effect, that is, not undergoing apoptosis A. This is an example of *selection bias*, whereby the phenomenon of apoptosis determines those cells available for measurement. The bias is illustrated in Figure D.4, which traces a path from perturbation P to apoptosis A, both via DSBs levels D, as well as directly, in a path representing other, unmeasured, cytotoxic causes.

In certain scenarios, a selection bias may be corrected for by standardisation or IP-weighting, giving an unbiased estimate of the causal effect. However, the particular causal structure we observe here defies correction, as the levels of A (cell death or not) are not *exchangeable* with respect to D, a necessary condition for corrective techniques. This clearly indicates the need for experimental techniques where apoptotic cells remain observable. Despite

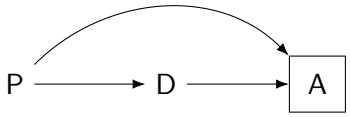


Figure D.4: Causal diagram including selection bias: P is the perturbation; D is the distribution of double-strand breaks and; A is the frequency of apoptosis. Conditioning (represented as a square) on the common effect of treatment P and outcome D creates a selection bias.

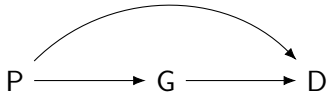


Figure D.5: Causal diagram showing effect modification of cell cycle phase: G is cell cycle phase; P is the perturbation, and D is the DNA damage.

this bias, independent analyses showed little difference in the distributions of DSBs within sparse (therefore highly apoptotic) and dense cell populations.

Another use case is in the effect of the cell cycle phase: given that some perturbations P affect the cell cycle phase G, and that DSBs levels D are influenced by both P and G, cell cycle variability acts as an *effect modifier* on D. These relations are expressed in the causal graph in Figure D.5. Though they are not a source of systematic bias, identifying effect modifiers can lead to an insightful stratified analysis. Through stratification, we were able to determine whether increased DSBs levels were a direct effect of perturbation, or an indirect effect of a modified cell cycle. In a stratified analysis, we compared the pooled cells of DMSO wells, as untreated subjects, with the cells of each perturbed well in turn. As we assessed 168 small compound perturbations, a multiple testing correction is due in our analysis. We compensate with the Benjamini-Hochberg correction for controlling the false discovery rate. We compare the effect of stratification in Section D.5.

D.5 Results

Despite the strong correlation between the three DSB measuring techniques, the number of perturbations exhibiting significance (hits) under each varies greatly. We reject the coarse approach of average intensity as being too sensitive to the ambient Cy3 signal and to staining artifacts. This shows in Table D.1 with unrealistically high hit rates for mean intensity. The most conservative are the granulometric features. The table presents the number of significant deviations from the negative control with and without stratification for the MDA231 cell line. The effects were seen for MDA468 also.

When we stratify, we report the number of hits without adjustment (all), for the G_1 and G_2 sub-populations of the wells separately, and the number of times these agree (joint). Taken apart, the subpopulations see fewer hits. This is a sign the perturbation influence on cell cycle is crucial.

	Double strand break quantifiers		
Modifier	Spot density	Granulometry	Mean intensity
All	45	34	87
G_1	40	10	62
G_2	15	14	34
Joint	9	4	18

Table D.1: Number of hits (out of 168) for each DSB quantifier on the MDA231 cell line. Significance at the 0.01 level with the Benjamini-Hochberg correction.

D.6 Conclusions

In this study, we have examined three viable approaches to measuring double strand breaks in fluorescence microscopy images. The more nuanced approaches of diameter openings and granulometry were found to give favourable outcomes. We have also introduced consideration of causal inference into a standard piece of analysis. We believe this framework can impact the statistical precision of other analyses in high content screening. An awareness of causal relationships and potential systematic biases has the potential to greatly improve experimental design and the precision of hit predictions in drug screening.

Appendix E

Supplementary analysis

E.1 Discovering phenotypic classes with unsupervised learning

Whereas in the morphological screen most cells fall neatly into recognisable classes, our drug screen is more obscure. By inspection, the cell populations are dramatically less rich phenotypically than the morphological screen. There are many aberrant morphologies (see for example Figure E.1) among the perturbed cell populations, but mitotic cells are scarce, and apoptotic cells are non-existent due to the washing of fluorescent dyes (and along with them, unfastened cells) prior to imaging. Machine learning is therefore an appropriate tool for wheedling out more subtle phenotypes.

We use clustering algorithms on the respective cell line datasets in an attempt to discover cell classes. To reduce dimensionality and to select between features with high mutual correlation between some of the extracted features[1], we apply principal components analysis (PCA). The large dataset remains restrictive in size to applying certain clustering algorithms. We therefore subsample the data. However, rather than uniform random sampling, we do the following: first, we apply k-means clustering to the dataset, choosing a potentially excessive number of clusters ($K = 30$). We expect this to identify small isolated clusters that may be lost by random sampling alone, as it is anticipated that interesting phentypes will be scarce. Then, we create a subsample of the dataset by sampling 50 cells (where available) from each cluster. With this new, smaller dataset, we apply clustering algorithms robust to anisotropic data, including: DBSCAN (Ester et al. [1996]), spectral clustering (Von Luxburg [2007]), and hierarchical clustering (Ward Jr

[1] For example, many shape features extracted by Cell Cognition, such as area, perimeter, convex hull, will increase together.

[1963]). We visualise the outcomes of the clustering, choosing $K = 6$, as t-SNE embeddings in two dimensions in Figure **??**.

We explore the clusters visually, by cropping examples of cells corresponding to each cluster. In Figure E.1 we see some subtle patterns emerge. The first cluster, the largest, appears to contain normal interphase cells. A second shows a mixture of normal and large interphase nuclei. Note that given our approach, it is anticipated that the predominant class, normal interphase, leaks into other clusters. Elsewhere, we see a smaller cluster comprising of many micro-nucleated cells, and another containing cells with high levels of DNA damage.

The ultimate specification of morphological classes should be determined by domain experts, with unsupervised learning merely providing a starting point. After this, one could pursue an analysis such as in Section **??**. While we feel this is an interesting approach, we turn rather to fully unsupervised methods in Chapter 3 to address the motivating questions of the assay.

E.2 Training a RoI classifier

We design a training set based on randomly placing 16 randomly chosen 28×28px digits from the MNIST digits dataset on a 256×256 background "canvas" image as shown in Figure E.2 (left).

We implement an region of interest (RoI) classifer as a convolutional neural network $\texttt{Conv}_{1,8} \rightarrow \texttt{ReLU} \rightarrow \texttt{MaxPool} \rightarrow \texttt{Conv}_{8,16} \rightarrow \texttt{ReLU} \rightarrow \texttt{RoIAlign} \rightarrow \texttt{FC}_{784,10}$, that is, a series of convolutional layers with ReLU activations followed by a RoIPool layer, and finally a fully-connected classification layer. The classifier is trained by a forward pass of such a canvas image, with the bounding box coordinates (x_0, y_0, x_1, y_1) for each of the digits. According to the bounding boxes, the RoIAlign layer separates the image into a pseudo-batch of RoIs, which are subsequently classified into one of the 10 digit classes by a softmax activation. Note that in this case, we choose the RoIAlign layer to quantise the incoming 14×14 tensors to 7×7 tensors. The stages of the forward pass are shown in Figure E.2. Trained in this manner, the classifier achieves 97% accuracy as a classifier of MNIST digits, thus a small percentage shy of the accuracy of a typical MNIST classifier. In a single forward pass, the classifier can make predictions for the whole canvas, as illustrated in Figure E.4.

We can also incorporate our RoI classifier as a discriminator (that is, as a binary real/fake classifier) in conditional image-to-image GAN(`pix2pix`) framework. Now, in addition to an input canvas of MNIST digits, we create conditioning images for each class (here '0' or '1' digits). An example is

given in Figure E.4. We succeed in training a `pix2pix` system with a region of interest discriminator. We see in Figure **??**. The `pix2pix` generator has thus learned to synthesis object detection images according to programmed localisation and classification information.

In Figure E.5 we compare (using the same specification of object localisation and class) a ground truth image with that synthesised by a `pix2pix` system with our RoI discriminator and that with a plain `pix2pix` system. We see a far greater degree of mode collapse on the baseline compared with our proposed model.

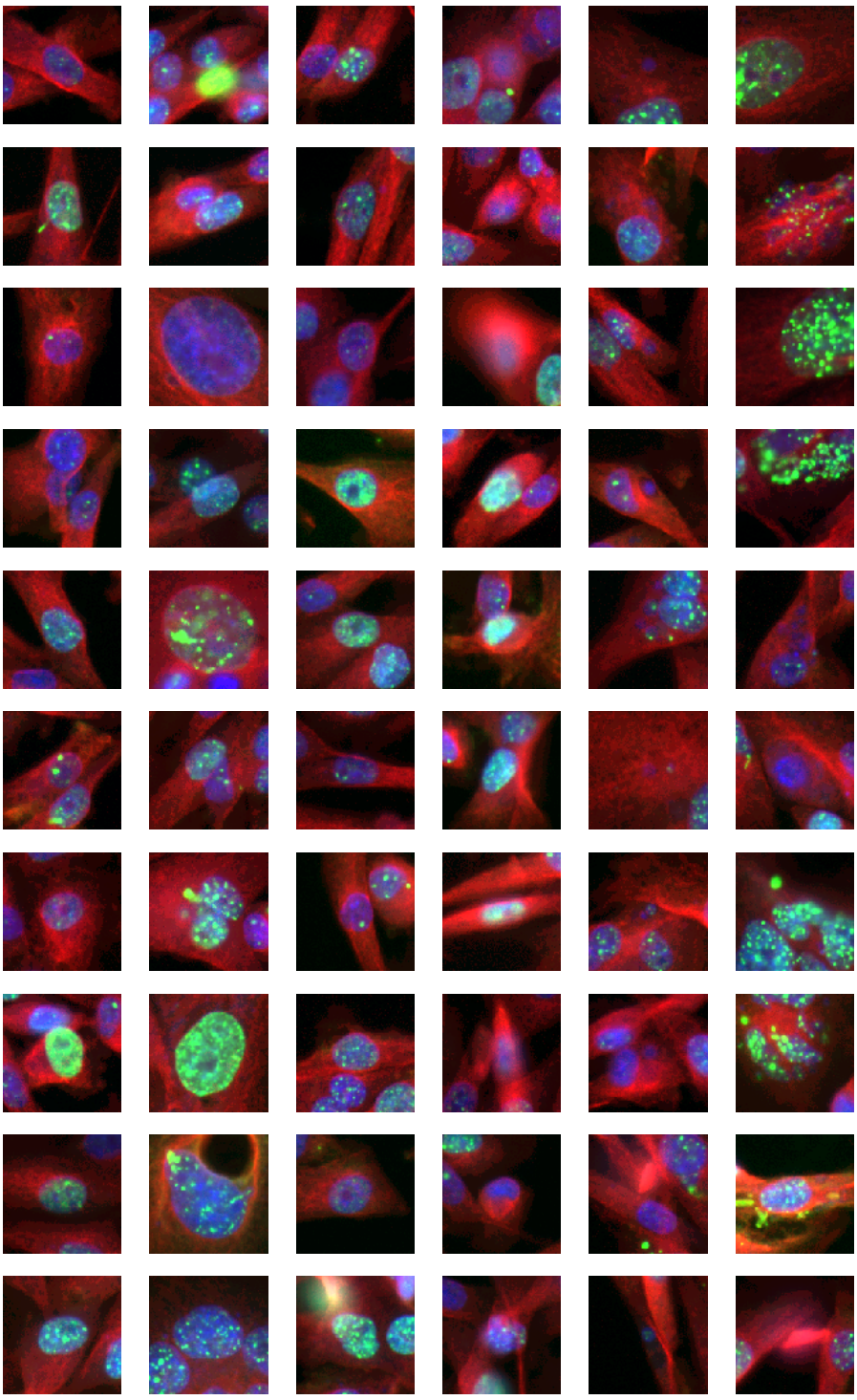


Figure E.1: Examples of spots (red) on cell nuclei detected with diameter openings on the Cy3 channel (grey).

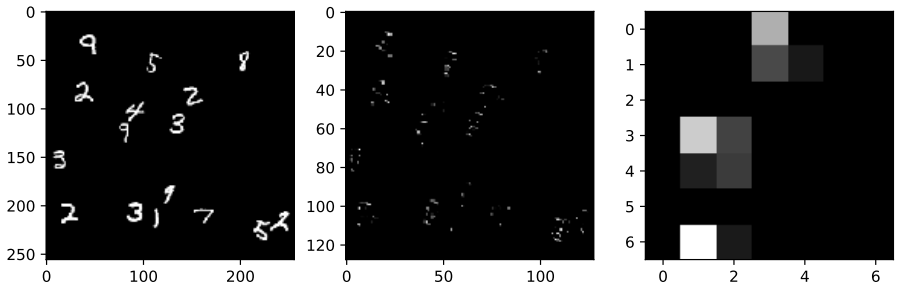


Figure E.2: From left to right: canvas input, activation map after MaxPooling layer, activation map after RoIAlign layer (reduced to single bounding box).

Figure E.3: Prediction of 16 digits over canvas, performed in a single forward pass.

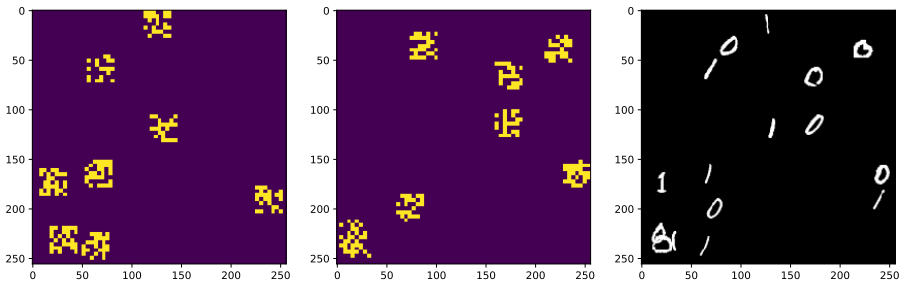
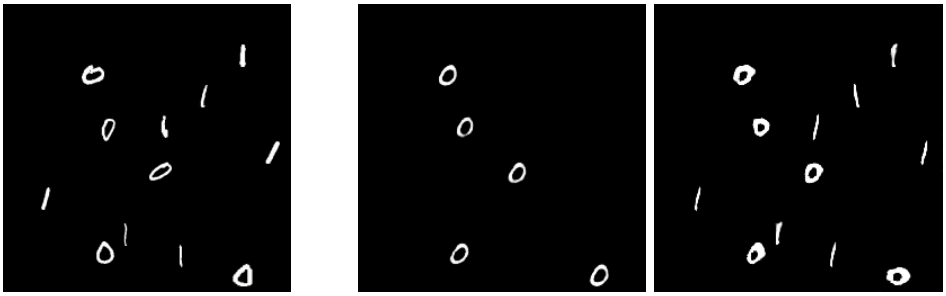


Figure E.4: Encodings of class and localisation specificiation (left and center) and corresponding train image for training conditional GAN with RoI discriminator.



(a) Ground truth (b) `pix2pix` (c) `pix2pix` + RoI discrimination

Figure E.5: With the same specification of object localisation and class, a ground truth image (a), a synthetic image from a plain `pix2pix` system (b) and synthetic image from our `pix2pix` with RoI discrimination.

Bibliography

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: https://doi.org/10.1038/s41592-019-0686-2.

Andrej Karpathy. What i learned from competing against a convnet on imagenet. *Andrej Karpathy Blog*, 5:1–15, 2014.

Kathryn J Chavez, Sireesha V Garimella, and Stanley Lipkowitz. Triple negative breast cancer cell lines: one tool in the search for better treatment of triple negative breast cancer. *Breast disease*, 32(1-2):35, 2010.

Eric S Lander, Lauren M Linton, Bruce Birren, Chad Nusbaum, Michael C Zody, Jennifer Baldwin, Keri Devon, Ken Dewar, Michael Doyle, William FitzHugh, et al. Initial sequencing and analysis of the human genome. 2001.

David Houle, Diddahally R Govindaraju, and Stig Omholt. Phenomics: the next challenge. *Nature reviews genetics*, 11(12):855–866, 2010.

Martin Mahner and Michael Kary. What exactly are genomes, genotypes and phenotypes? and what about phenomes? *Journal of theoretical biology*, 186(1):55–63, 1997.

Gene Myers. Why bioimage informatics matters. *Nature methods*, 9(7):659, 2012.

Steven A Haney. *High content screening: science, techniques and applications*. John Wiley & Sons, 2008.

Beate Neumann, Thomas Walter, Jean-Karim Hériché, Jutta Bulkescher, Holger Erfle, Christian Conrad, Phill Rogers, Ina Poser, Michael Held, Urban Liebel, et al. Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes. *Nature*, 464(7289):721, 2010.

Jeremy C Simpson, Brigitte Joggerst, Vibor Laketa, Fatima Verissimo, Cihan Cetin, Holger Erfle, Mariana G Bexiga, Vasanth R Singan, Jean-Karim Hériché, Beate Neumann, et al. Genome-wide rnai screening identifies human proteins with a regulatory function in the early secretory pathway. *Nature cell biology*, 14(7):764–774, 2012.

Claudio Collinet, Martin Stöter, Charles R Bradshaw, Nikolay Samusik, Jochen C Rink, Denise Kenski, Bianca Habermann, Frank Buchholz, Robert Henschel, Matthias S Mueller, et al. Systems survey of endocytosis by multiparametric image analysis. *Nature*, 464(7286):243–249, 2010.

Michael V Boland, Mia K Markey, and Robert F Murphy. Automated recognition of patterns characteristic of subcellular structures in fluorescence microscopy images. *Cytometry: The Journal of the International Society for Analytical Cytology*, 33(3):366–375, 1998.

Steven A Haney, Peter LaPan, Jing Pan, and Jing Zhang. High-content screening moves to the front of the line. *Drug discovery today*, 11(19): 889–894, 2006.

Rainer Pepperkok and Jan Ellenberg. High-throughput fluorescence microscopy for systems biology. *Nature reviews Molecular cell biology*, 7(9): 690–696, 2006a.

David C Swinney and Jason Anthony. How were new medicines discovered? *Nature reviews Drug discovery*, 10(7):507–519, 2011.

William F Scherer, Jerome T Syverton, and George O Gey. Studies on the propagation in vitro of poliomyelitis viruses: Iv. viral multiplication in a stable strain of human malignant epithelial cells (strain hela) derived from an epidermoid carcinoma of the cervix. *The Journal of experimental medicine*, 97(5):695–710, 1953.

Vivien Marx. Cell-line authentication demystified. *Nature methods*, 11(5): 483, 2014.

Stefan Terjung, Thomas Walter, Arne Seitz, Beate Neumann, Rainer Pepperkok, and Jan Ellenberg. High-throughput microscopy using live mammalian cells. *Cold Spring Harbor Protocols*, 2010(8):pdb–top84, 2010.

Zachary E Perlman, Michael D Slack, Yan Feng, Timothy J Mitchison, Lani F Wu, and Steven J Altschuler. Multidimensional drug profiling by automated microscopy. *Science*, 306(5699):1194–1198, 2004.

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. In *Methods in enzymology*, volume 414, pages 440–468. Elsevier, 2006.

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.

Nikita Orlov, Lior Shamir, Tomasz Macura, Josiah Johnston, D Mark Eckley, and Ilya G Goldberg. Wnd-charm: Multi-purpose image classification using compound image transforms. *Pattern recognition letters*, 29(11): 1684–1693, 2008.

Virginie Uhlmann, Shantanu Singh, and Anne E Carpenter. Cp-charm: segmentation-free image classification made accessible. *BMC bioinformatics*, 17(1):51, 2016.

Oren Z Kraus, Jimmy Lei Ba, and Brendan J Frey. Classifying and segmenting microscopy images with deep multiple instance learning. *Bioinformatics*, 32(12):i52–i59, 2016.

Michael D Slack, Elisabeth D Martinez, Lani F Wu, and Steven J Altschuler. Characterizing heterogeneous cellular responses to perturbations. *Proceedings of the National Academy of Sciences*, pages pnas–0807038105, 2008.

William J Godinez, Imtiaz Hossain, Stanley E Lazic, John W Davies, and Xian Zhang. A multi-scale convolutional neural network for phenotyping high-content cellular images. *Bioinformatics*, 33(13):2010–2019, 2017.

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, 2017.

Euan A Ashley. Towards precision medicine. *Nature Reviews Genetics*, 17 (9):507, 2016.

Chayakrit Krittanawong, HongJu Zhang, Zhen Wang, Mehmet Aydar, and Takeshi Kitai. Artificial intelligence in precision cardiovascular medicine. *Journal of the American College of Cardiology*, 69(21):2657–2664, 2017.

Tim Hulsen, Saumya Shekhar Jamuar, Alan Moody, Jason Hansen Karnes, Varga Orsolya, Stine Hedensted, Roberto Spreafico, David A Hafler, and Eoin McKinney. From big data to precision medicine. *Frontiers in medicine*, 6:34, 2019.

James C Costello, Laura M Heiser, Elisabeth Georgii, Mehmet Gönen, Michael P Menden, Nicholas J Wang, Mukesh Bansal, Petteri Hintsanen, Suleiman A Khan, John-Patrick Mpindi, et al. A community effort to assess and improve drug sensitivity prediction algorithms. *Nature biotechnology*, 32(12):1202, 2014.

Steven J Altschuler and Lani F Wu. Cellular heterogeneity: do differences make a difference? *Cell*, 141(4):559–563, 2010.

France Rose, Sreetama Basu, Elton Rexhepaj, Anne Chauchereau, Elaine Del Nery, and Auguste Genovesio. Compound functional prediction using multiple unrelated morphological profiling assays. *SLAS TECHNOLOGY: Translating Life Sciences Innovation*, 23(3):243–251, 2018.

Scott J Warchal, John C Dawson, and Neil O Carragher. Development of the theta comparative cell scoring method to quantify diverse phenotypic responses between distinct cell types. *Assay and drug development technologies*, 14(7):395–406, 2016.

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.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

Vebjorn Ljosa, Katherine L Sokolnicki, and Anne E Carpenter. Annotated high-throughput microscopy image sets for validation. *Nature methods*, 9 (7):637–637, 2012.

Peter D Caie, Rebecca E Walls, Alexandra Ingleston-Orme, Sandeep Daya, Tom Houslay, Rob Eagle, Mark E Roberts, and Neil O Carragher. High-content phenotypic profiling of drug response signatures across distinct cancer cells. *Molecular cancer therapeutics*, 9(6):1913–1926, 2010.

Chetak Kandaswamy, Luís M Silva, Luís A Alexandre, and Jorge M Santos. High-content analysis of breast cancer using single-cell deep transfer learning. *Journal of biomolecular screening*, 21(3):252–259, 2016.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 181–196, 2018.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

Eric M Christiansen, Samuel J Yang, D Michael Ando, Ashkan Javaherian, Gaia Skibinski, Scott Lipnick, Elliot Mount, Alison OâĂŹNeil, Kevan Shah, Alicia K Lee, et al. In silico labeling: Predicting fluorescent labels in unlabeled images. *Cell*, 173(3):792–803, 2018.

Chawin Ounkomol, Sharmishtaa Seshamani, Mary M Maleckar, Forrest Collman, and Gregory R Johnson. Label-free prediction of three-dimensional fluorescence images from transmitted-light microscopy. *Nature methods*, 15(11):917, 2018.

Sajith Kecheril Sadanandan, Petter Ranefall, Sylvie Le Guyader, and Carolina Wählby. Automated training of deep convolutional neural networks for cell segmentation. *Scientific reports*, 7(1):7860, 2017.

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

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014a.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

Anton Osokin, Anatole Chessel, Rafael E Carazo Salas, and Federico Vaggi. Gans for biological image synthesis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2233–2242, 2017.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks.

In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

Stephan J Ihle, Andreas M Reichmuth, Sophie Girardin, Hana Han, Flurin Stauffer, Anne Bonnin, Marco Stampanoni, Karthik Pattisapu, János Vörös, and Csaba Forró. Unsupervised data to content transformation with histogram-matching cycle-consistent generative adversarial networks. *Nature Machine Intelligence*, 1(10):461–470, 2019.

Joseph Boyd, Alice Pinhiero, Elaine Del Nery, Fabien Reyal, and Thomas Walter. Analysing double-strand breaks in cultured cells for drug screening applications by causal inference. In *Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on*, pages 445–448. IEEE, 2018.

Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*, 2014.

Joseph C Boyd, Alice Pinheiro, Elaine Del Nery, Fabien Reyal, and Thomas Walter. Domain-invariant features for mechanism of action prediction in a multi-cell-line drug screen. *Bioinformatics*, 36(5):1607–1613, 2020.

Joseph Boyd, Fabien Reyal, Alice Pinheiro, Elaine Del Nery, and Thomas Walter. Domain-invariant features for mechanism of action prediction in a multi-cell-line drug screen, May 2019a. URL https://zenodo.org/record/2677923.

Joseph Boyd, Zelia Gouveia, Franck Perez, and Thomas Walter. Lymphocytes dataset, Oct 2019b. URL https://zenodo.org/record/3515446.

Peter Naylor, Joseph Boyd, Marick Laé, Fabien Reyal, and Thomas Walter. Predicting residual cancer burden in a triple negative breast cancer cohort. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 933–937. IEEE, 2019.

Beyrem Khalfaoui, Joseph Boyd, and Jean-Philippe Vert. Adaptive structured noise injection for shallow and deep neural networks. 2019.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

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

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5): 359–366, 1989.

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.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *Icdar*, volume 3, 2003.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014b.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Aayush Bansal, Shugao Ma, Deva Ramanan, and Yaser Sheikh. Recycle-gan: Unsupervised video retargeting. In *Proceedings of the European conference on computer vision (ECCV)*, pages 119–135, 2018.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Clifford A Hudis and Luca Gianni. Triple-negative breast cancer: an unmet medical need. *Oncologist*, 16, 2011.

Homero Gonçalves Jr, Maximiliano Ribeiro Guerra, Jane Rocha Duarte Cintra, Vívian Assis Fayer, Igor Vilela Brum, and Maria Teresa Bustamante Teixeira. Survival study of triple-negative and non–triple-negative breast cancer in a brazilian cohort. *Clinical Medicine Insights: Oncology*, 12:1179554918790563, 2018.

Rana Hatem, Rania El Botty, Sophie Chateau-Joubert, Jean-Luc Servely, Dalila Labiod, Ludmilla de Plater, Franck Assayag, Florence Coussy, Céline Callens, Sophie Vacher, et al. Targeting mtor pathway inhibits tumor growth in different molecular subtypes of triple-negative breast cancers. *Oncotarget*, 7(30):48206, 2016.

Jorge Filmus, Michael N Pollak, Relda Cailleau, and Ronald N Buick. Mda-468, a human breast cancer cell line with a high number of epidermal growth factor (egf) receptors, has an amplified egf receptor gene and is growth inhibited by egf. *Biochemical and biophysical research communications*, 128(2):898–905, 1985.

Shantanu Singh, M-A BRAY, TR Jones, and AE Carpenter. Pipeline for illumination correction of images for high-throughput microscopy. *Journal of microscopy*, 256(3):231–236, 2014.

Xiaofeng Dai, Hongye Cheng, Zhonghu Bai, and Jia Li. Breast cancer cell line classification and its relevance with breast tumor subtyping. *Journal of Cancer*, 8(16):3131, 2017.

Etienne Y Lasfargues and Luciano Ozzello. Cultivation of human breast carcinomas. *Journal of the National Cancer Institute*, 21(6):1131–1147, 1958.

Adeline J Hackett, Helene S Smith, E Louise Springer, Robert B Owens, Walter A Nelson-Rees, John L Riggs, and Murray B Gardner. Two syngeneic cell lines from human breast tissue: the aneuploid mammary epithelial (hs578t) and the diploid myoepithelial (hs578bst) cell lines. *Journal of the National Cancer Institute*, 58(6):1795–1806, 1977.

Adi F Gazdar, Venkatesh Kurvari, Arvind Virmani, Lauren Gollahon, Masahiro Sakaguchi, Max Westerfield, Duli Kodagoda, Victor Stasny, H Thomas Cunningham, Ignacio I Wistuba, et al. Characterization of paired tumor and non-tumor cell lines established from patients with breast cancer. *International journal of cancer*, 78(6):766–774, 1998.

BR Brinkley, PT Beall, LJ Wible, ML Mace, Donna S Turner, and RM Cailleau. Variations in cell form and cytoskeleton in human breast carcinoma cells in vitro. *Cancer research*, 40(9):3118–3129, 1980.

Herbert D Soule, Terry M Maloney, Sandra R Wolman, Ward D Peterson, Richard Brenz, Charles M McGrath, Jose Russo, Robert J Pauley, Richard F Jones, and SC Brooks. Isolation and characterization of a spontaneously immortalized human breast epithelial cell line, mcf-10. *Cancer research*, 50(18):6075–6086, 1990.

T Huang, GJTGY Yang, and Greory Tang. A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(1):13–18, 1979.

Serge Beucher and Christian Lantuéjoul. Use of watersheds in contour detection. 1979.

Michel Grimaud. New measure of contrast: the dynamics. volume 1769, pages 292–305, 1992. doi: 10.1117/12.60650. URL http://dx.doi.org/10.1117/12.60650.

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.

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

Michael Held, Michael HA Schmitz, Bernd Fischer, Thomas Walter, Beate Neumann, Michael H Olma, Matthias Peter, Jan Ellenberg, and Daniel W Gerlich. Cellcognition: time-resolved phenotype annotation in high-throughput live cell imaging. *Nature methods*, 7(9):747–754, 2010.

Thomas Walter, Michael Held, Beate Neumann, Jean-Karim Hériché, Christian Conrad, Rainer Pepperkok, and Jan Ellenberg. Automatic identification and clustering of chromosome phenotypes in a genome wide rnai screen by time-lapse imaging. *Journal of structural biology*, 170(1):1–9, 2010a.

Iurie Caraus, Abdulaziz A. Alsuwailem, Robert Nadon, and Vladimir Makarenkov. Detecting and overcoming systematic bias in high-throughput screening technologies: A comprehensive review of practical issues and methodological solutions. 16(6):974–986. ISSN 1467-5463. doi: 10.1093/bib/bbv004. URL https://doi.org/10.1093/bib/bbv004.

Vebjorn Ljosa and Anne E. Carpenter. Introduction to the Quantitative Analysis of Two-Dimensional Fluorescence Microscopy Images for Cell-Based Screening. 5(12):e1000603. doi: 10.1371/journal.pcbi.1000603. URL https://doi.org/10.1371/journal.pcbi.1000603.

DE Pegg. Viability assays for preserved cells, tissues, and organs. *Cryobiology*, 26(3):212–231, 1989.

Leland McInnes and John Healy. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Rainer Pepperkok and Jan Ellenberg. High-throughput fluorescence microscopy for systems biology. *Nature reviews Molecular cell biology*, 7(9): 690, 2006b.

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.

Serge PJM Horbach and Willem Halffman. The ghosts of hela: How cell line misidentification contaminates the scientific literature. *PloS one*, 12 (10):e0186281, 2017.

Yansheng Liu, Yang Mi, Torsten Mueller, Saskia Kreibich, Evan G Williams, Audrey Van Drogen, Christelle Borel, Max Frank, Pierre-Luc Germain, Isabell Bludau, et al. Multi-omic measurements of heterogeneity in hela cells across laboratories. *Nature biotechnology*, 37(3):314, 2019.

Thomas Walter, Michael Held, Beate Neumann, Jean-Karim Hériché, Christian Conrad, Rainer Pepperkok, and Jan Ellenberg. Automatic identification and clustering of chromosome phenotypes in a genome wide RNAi screen by time-lapse imaging. *Journal of structural biology*, 170(1):1–9, apr 2010b. ISSN 1095-8657. doi: 10.1016/j.jsb.2009.10.004. URL http://www.ncbi.nlm.nih.gov/pubmed/19854275.

Daniel Müllner et al. fastcluster: Fast hierarchical, agglomerative clustering routines for r and python. *Journal of Statistical Software*, 53(9):1–18, 2013.

Lit-Hsin Loo, Lani F Wu, and Steven J Altschuler. Image-based multivariate profiling of drug responses from single cells. *Nature methods*, 4(5):445, 2007.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-

learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

François Chollet et al. Keras. https://keras.io, 2015.

Stéfan van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in Python. *PeerJ*, 2:e453, 2014. ISSN 2167-8359. doi: 10.7717/peerj.453. URL https://doi.org/10.7717/peerj.453.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.

Mohamed-Reda Benmebarek, Clara Helke Karches, Bruno Loureiro Cadilha, Stefanie Lesch, Stefan Endres, and Sebastian Kobold. Killing mechanisms of chimeric antigen receptor (car) t cells. *International journal of molecular sciences*, 20(6):1283, 2019.

Shannon L Maude, David T Teachey, David L Porter, and Stephan A Grupp. Cd19-targeted chimeric antigen receptor t-cell therapy for acute lymphoblastic leukemia. *Blood*, 125(26):4017–4023, 2015.

M Sadelain, R Brentjens, and I Riviere. The basic principles of chimeric antigen receptor (car) design 3 (4):, 2013.

Stephen P Hunger and Charles G Mullighan. Acute lymphoblastic leukemia in children. *New England Journal of Medicine*, 373(16):1541–1552, 2015.

Shannon L Maude, Noelle Frey, Pamela A Shaw, Richard Aplenc, David M Barrett, Nancy J Bunin, Anne Chew, Vanessa E Gonzalez, Zhaohui Zheng, Simon F Lacey, et al. Chimeric antigen receptor t cells for sustained remissions in leukemia. *New England Journal of Medicine*, 371 (16):1507–1517, 2014.

Challice L Bonifant, Hollie J Jackson, Renier J Brentjens, and Kevin J Curran. Toxicity and management in car t-cell therapy. *Molecular Therapy-Oncolytics*, 3:16011, 2016.

Zhenguang Wang, Yelei Guo, and Weidong Han. Current status and perspectives of chimeric antigen receptor modified t cells for cancer treatment. *Protein & cell*, 8(12):896–925, 2017.

MA Epstein, BG Achong, YM Barr, B Zajac, G Henle, and W Henle. Morphological and virological investigations on cultured burkitt tumor lym-

phoblasts (strain raji). *Journal of the National Cancer Institute*, 37(4): 547–559, 1966.

Gyuhyun Lee, Jeong-Woo Oh, Mi-Sun Kang, Nam-Gu Her, Myoung-Hee Kim, and Won-Ki Jeong. Deephcs: Bright-field to fluorescence microscopy image conversion using deep learning for label-free high-content screening. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 335–343. Springer, 2018.

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.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

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.

Maithra Raghu, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. Transfusion: Understanding transfer learning with applications to medical imaging. *arXiv preprint arXiv:1902.07208*, 2019.

Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, pages 2528–2535. IEEE, 2010.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Advances in neural information processing systems*, pages 262–270, 2015.

Xu Zheng, Tejo Chalasani, Koustav Ghosal, Sebastian Lutz, and Aljosa Smolic. Stada: Style transfer as data augmentation. *arXiv preprint arXiv:1909.01056*, 2019.

Martin Bock, Amit Kumar Tyagi, Jan-Ulrich Kreft, and Wolfgang Alt. Gen-

eralized voronoi tessellation as a model of two-dimensional cell tissue dynamics. *Bulletin of mathematical biology*, 72(7):1696–1731, 2010.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

Reka Hollandi, Abel Szkalisity, Timea Toth, Ervin Tasnadi, Csaba Molnar, Botond Mathe, Istvan Grexa, Jozsef Molnar, Arpad Balind, Mate Gorbe, et al. A deep learning framework for nucleus segmentation using image style transfer. *bioRxiv*, page 580605, 2019.

Chichen Fu, Soonam Lee, David Joon Ho, Shuo Han, Paul Salama, Kenneth W Dunn, and Edward J Delp. Three dimensional fluorescence microscopy image synthesis and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2221–2229, 2018.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

Dane Avondoglio, Tamalee Scott, Whoon Jong Kil, Mary Sproull, Philip J Tofilon, and Kevin Camphausen. High throughput evaluation of gamma-h2ax. *Radiation Oncology*, 4(1):31, 2009.

Carolina Garcia-Canton, Arturo Anadon, and Clive Meredith. Assessment of the in vitro γh2ax assay by high content screening as a novel genotoxicity test. *Mutation Research/Genetic Toxicology and Environmental Mutagenesis*, 757(2):158–166, 2013.

Catherine Chailleux, François Aymard, Pierre Caron, Virginie Daburon, Céline Courilleau, Yvan Canitrot, Gaëlle Legube, and Didier Trouche. Quantifying dna double-strand breaks induced by site-specific endonucleases in living cells by ligation-mediated purification. *Nature protocols*, 9(3):517–528, 2014.

Peter Naylor, Marick Lae, Fabien Reyal, and Thomas Walter. Nuclei segmentation in histopathology images using deep neural networks. In *Proceedings of the 12th IEEE International Symposium on Biomedical Imaging (ISBI): From nano to macro*, pages 933–936, 04 2017.

Thomas Walter, Pascale Massin, Ali Erginay, Richard Ordonez, Clothilde Jeulin, and Jean-Claude Klein. Automatic detection of microaneurysms in color fundus images. *Medical Image Analysis*, 11(6):555–66, 2007.

Judea Pearl et al. Causal inference in statistics: An overview. *Statistics Surveys*, 3:96–146, 2009.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

RÉSUMÉ

Cuius acerbitati uxor grave accesserat incentivum, germanitate Augusti turgida supra modum, quam Hannibaliano regi fratris filio antehac Constantinus iunxerat pater, Megaera quaedam mortalis, inflammatrix saevientis adsidua, humani cruoris avida nihil mitius quam maritus; qui paulatim eruditiores facti processu temporis ad nocendum per clandestinos versutosque rumigerulos conpertis leviter addere quaedam male suetos falsa et placentia sibi discentes, adfectati regni vel artium nefandarum calumnias insontibus adfligebant.

Saraceni tamen nec amici nobis umquam nec hostes optandi, ultro citroque discursantes quicquid inveniri poterat momento temporis parvi vastabant milvorum rapacium similes, qui si praedam dispexerint celsius, volatu rapiunt celeri, aut nisi impetraverint, non inmorantur.

Vita est illis semper in fuga uxoresque mercenariae conductae ad tempus ex pacto atque, ut sit species matrimonii, dotis nomine futura coniunx hastam et tabernaculum offert marito, post statum diem si id elegerit discessura, et incredibile est quo ardore apud eos in venerem uterque solvitur sexus.

Sed tamen haec cum ita tutius observentur, quidam vigore artuum inminuto rogati ad nuptias ubi aurum dextris manibus cavatis offertur, inpigre vel usque Spoletium pergunt. haec nobilium sunt instituta.

MOTS CLÉS

Caesar licentia post honoratis haec adhibens urbium honoratis nullum Caesar.

ABSTRACT

Verum ad istam omnem orationem brevis est defensio. Nam quoad aetas M. Caeli dare potuit isti suspicioni locum, fuit primum ipsius pudore, deinde etiam patris diligentia disciplinaque munita. Qui ut huic virilem togam deditÅąnihil dicam hoc loco de me; tantum sit, quantum vos existimatis; hoc dicam, hunc a patre continuo ad me esse deductum; nemo hunc M. Caelium in illo aetatis flore vidit nisi aut cum patre aut mecum aut in M. Crassi castissima domo, cum artibus honestissimis erudiretur.

Et eodem impetu Domitianum praecipitem per scalas itidem funibus constrinxerunt, eosque coniunctos per ampla spatia civitatis acri raptavere discursu. iamque artuum et membrorum divulsa conpage superscandentes corpora mortuorum ad ultimam truncata deformitatem velut exsaturati mox abiecerunt in flumen.

Erat autem diritatis eius hoc quoque indicium nec obscurum nec latens, quod ludicris cruentis delectabatur et in circo sex vel septem aliquotiens vetitis certaminibus pugilum vicissim se concidentium perfusorumque sanguine specie ut lucratus ingentia laetabatur.

Ego vero sic intellego, Patres conscripti, nos hoc tempore in provinciis decernendis perpetuae pacis habere oportere rationem. Nam quis hoc non sentit omnia alia esse nobis vacua ab omni periculo atque etiam suspicione belli?

KEYWORDS

Delatus delatus nominatus onere aut trahebatur quod tenus et bonorum.