FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Image-based chemical-genetic profiling using Deep Neural Networks

David Dao

FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Image-based chemical-genetic profiling using Deep Neural Networks

Bildbasierte chemisch-genetische Profilierung mittels Tiefer Neuronaler Netze

Author:	David Dao
Supervisor:	Prof. Dr. Björn Menze
Advisor:	Markus Rempfler
External Advisors :	Dr. Anne Carpenter, Dr. Shantanu Singh
Date:	Nov 2, 2016

Ich versichere, dass ich diese Masterarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

I assure the single handed composition of this master's thesis only supported by declared resources.

München, den 25. Mai 2017 David Dao

Acknowledgments

First, I would like to thank my advisors Prof. Dr. Björn Menze and Markus Rempfler for advising and supporting me during the process of this thesis and for giving me the opportunity to do my research abroad.
Furthermore, I would like to thank my external advisor, Anne Carpenter, for providing support, feedback and advice, and demonstrating an inspiring level of effectiveness. I am grateful for the chance of joining her lab at the Broad Institute of MIT and Harvard.
I'd like to thank Shantanu Singh, Mohammad Rohban and Jane Hung for our enjoyable collaborations and many fun and fruitful discussions. Our multilingual whiteboard will always be remembered.
Moreover, thanks to Alison Kozol, Lee Kamentsky, Allen Goodman, David Logan, Mark Bray, Holger Hennig and Kyle Karhohs for making the lab feel like a big family.
Finally, I want to thank my parents for reminding me that I have to finish my thesis, for supporting me, and for their love.
I gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X GPU used for this research.

Abstract

Image-based chemical-genetic profiling [1] is a powerful approach enabling the massive amounts of image data generated from high-throughput experiments to be used in functional genomics, systems biology, and drug discovery [2]. Thousands of measurements extracted from images of cell populations can be analyzed to predict previously unrecognized cell states induced by experimental perturbations. However, how to map these high-dimensional measurements to optimally useful perturbation profiles is not understood. Current approaches using population averaging [3] either fail to consider non-linear relationships between measurements [4] or to preserve single-cell information.

In this thesis, we investigate a promising alternative: By describing a cell population as a combination of subpopulations [5], therefore explicitly modeling heterogeneity, we show that we can extract interpretable profiles with lower dimensionality for chemical and genetic perturbations compared to current methods. However a fundamental question is whether subpopulation profiles are in general better than profiles averaging cell measurements across populations. In order to help us devise a test, we developed CellProfiler Analyst 2.0 [6][7], an interactive data exploration, analysis, and classification software for large biological image sets. Using CellProfiler Analyst, we were able to generate SUBPOP, a benchmark dataset of 2526 hand-labeled cells with 23 phenotypes extracted from the well-studied BBBC021 dataset of the Broad Bioimage Benchmark Collection [8]. We introduce a supervised deep neural network that is able to efficiently model subpopulations by learning visual features directly from images. Using a 32 layer deep residual neural network (ResNet-32) [9] on raw pixels, we were able to outperform handcrafted features and achieve 72% accuracy on a 5-fold cross validation on SUBPOP. Finally, we show that by using subpopulation profiles and careful noise reduction, we were able to correctly classify 83% of the biological mechanism-of-action for each treatment in BBBC021 using only the DNA channel and, hence, outperforming the best-performing classical method for profiling.

Contents

1. Introduction

The science of seeing the very small is increasingly an endeavour to extract knowledge from the very large. In recent years, biology has undergone a dramatic shift from a qualitative to a quantitative big-data-driven science. Technological advances and high-throughput experiments have led to an unprecedented accumulation of relevant biological raw data and hence a fundamental change in experimental strategy. While traditional, targeted screening experiments aim to quantify specific feature such as a single chemical process or cell function [10], it has now become possible to measure hundreds or even thousands of distinct properties from a biological sample after a chemical or genetic perturbation (i.e. chemical compound, genetic knockout) - a powerful method known as profiling [2]. The goal of profiling is to map all these raw biological measurements to optimally useful profiles, holding as much information about the given perturbation as possible. The resulting profiles can help to gain insights into the perturbation or derive clusters and specific patterns when compared to other profiles. Due to its low costs and robust results, profiling has numerous applications in biology: It has been used to map genetic interactions [11], identify small molecules with novel mechanism of action [12], classify the mechanism of action of chemical compounds [13], and create a performance-diverse compound library [14]. There is little doubt that profiling has the potential to transform many fields in biology.

Microscopy plays a key role to profiling, because it enables the inexpensive generation of large image datasets with single-cell resolution. Profiling aims to redefine microscopy images from a qualitative, subjective resource to a quantitative data source, where a signature of each sample is extracted based on single-cell measurements. In theory, these images are able to capture fine-grained visual changes caused by a given perturbation on every

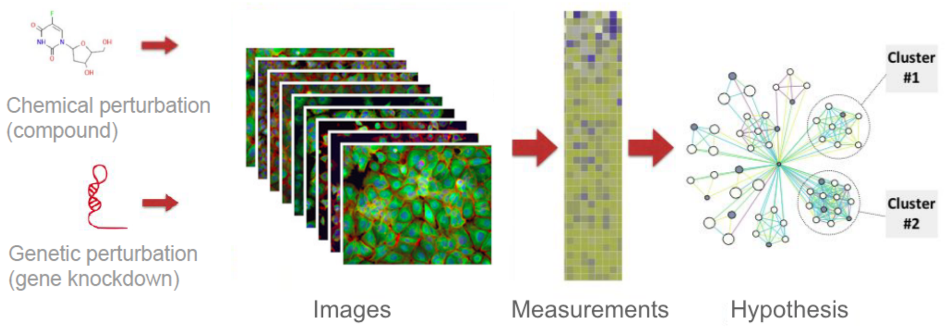


Figure 1.1.: **Image-based profiling** - Using measurements from microscopy images and the use of machine learning/data mining techniques, image-based profiling aims to find patterns caused by biological perturbations (chemical or genetic).

cell of the imaged population. In practice however, profiling faces many computational challenges due to the complexity and size of the data.
Hence, image-based profiling, also known as morphological or cytological profiling, is fundamentally an image analysis task. Figure 1.1 shows a schematic workflow for image-based profiling. First we treat cell populations with chemical-genetic perturbations and capture the morphology of the population using imaging. Further, we extract the best representation that captures the morphological impact of a known perturbation given only the raw image data. Creating these representations or profiles is an open research question. At present, there is no standard approach nor are there software packages that implement the variety of methods that have been proposed. A promising idea towards optimized profiles is to preserve single cell data and thus take into account the increasingly well-appreciated heterogeneity of cultured cell populations [5]. Almost all profiling methods rely on hand-crafted features (such as cell shape, intensity or texture) from automated image segmentation and object measurement software such as the open-source tool CellProfiler [15] [16] in order to model heterogeneity. However, in the only comparison between current profiling techniques [3] based on hand-crafted features, methods that attempt to leverage population heterogeneity were surprisingly outperformed by relatively simple population aggregation methods where a population's measurements have been averaged in order to compare populations against each other. This study allows two possible conclusions: either, for image-based profiling, there is no value in describing a cell population as a combination of subpopulations (that is, explicitly modeling heterogeneity) or existing methods for creating population profiles just do not capture heterogeneity well.

In this work, we focus on the latter hypothesis and investigate a promising alternative: Deep learning is improving various computer vision tasks by increasing the accuracy and efficiency of automated image analysis [17]. There are several recent studies confirming the potential of deep neural networks to microscopy imaging problems [18]. We look into deep learning techniques, especially deep residual neural networks [9] [19], which are able to learn fine-grained representations from image data and use it to train an efficient phenotype classifier, thus directly modeling cellular heterogeneity.

1.1. Scientific Contribution

Our main contribution is the implementation and analysis of a scalable and expert-guided neural approach to efficiently model heterogeneity in cultured cell populations for image-based profiling. Using a very deep ResNet-32 architecture trained on SUBPOP, a small and unbalanced data set, which we extracted from BBBC021 [8], we show that it outperforms traditional phenotype classification algorithms using hand-tuned visual features by a large margin. Moreover, subpopulations can be used as a compressed and more interpretable perturbation profile. Our results suggests that it is comparable to and (for our problem instance) even outperforms the best-performing classical method of profiling for detecting biological mechanism-of-actions (MOA) on BBBC021.

Our second contribution is the design and development of CellProfiler Analyst 2.0, an interactive data exploration, analysis, and classification software for large biological image sets, which we published in [7]. CellProfiler Analyst [6] was originally designed as a visualization tool for biologists. CellProfiler Analyst 2.0 introduces iterative machine learning which leverages researchers' knowledge to quickly analyze and curate training sets for machine learning. Included is a supervised machine learning system, integrating scikit-learn [20], which can be trained to recognize complicated and subtle phenotypes, for automatic scoring of millions of cells. Furthermore, CPA 2.0 provides additional tools for exploring and analyzing multidimensional data, particularly data from high-throughput, image-based experiments analyzed by its companion image analysis software, CellProfiler. The implementation of CellProfiler Analyst is open source and available for download at `https://github.com/CellProfiler/CellProfiler-Analyst`.

Our third contribution is the curation of our benchmark dataset SUBPOP, which consists of 2526 hand-labeled single cells with 23 distinct phenotypes. We were able to quickly label, train and evaluate baseline classifiers from scikit-learn using CellProfiler Analyst 2.0. The SUBPOP dataset is publicly available at `https://github.com/daviddao/SUBPOP`.

1.2. Structure of the thesis

The rest of this chapter is organized as follows: Initially, we discuss some preliminaries (Chapter 2) and present related work (Chapter 3). Subsequently, in Chapter 4, we present a short description of CellProfiler Analyst. Next, in Chapter 5, we provide an experimental evaluation of deep subpopulation profiling and describe datasets, classifiers and workflows used to model heterogeneity and extract profiles. Furthermore, we briefly explore the use of neural attention for automatic cell detection. Finally, we present a brief summary of our work and conclusion (Chapter 6).

Chapter 4 has been derived from the following journal paper: D. Dao *et al.* "CellProfiler Analyst: interactive data exploration, analysis, and classification of large biological image sets", that has been published in Bioinformatics [7].

2. Preliminaries

The goal of this chapter is to introduce the reader to the two-fold background of this thesis. In the following sections, we will give:

1. An introduction to image-based profiling, particularly, (1) its goal; (2) applications; (3) the current processing pipeline using CellProfiler and (4) its advantages and challenges.

2. A quick walkthrough of deep learning, a branch of machine learning based on learning representations of data using deep graphs with linear and nonlinear transformations.

2.1. Image-based Profiling

Image-based profiling aims to capture and encode as many properties of a biological sample as possible. Although both screening and profiling involve large-scale high throughput experiments, their goals differ [2]. Screening experiments aim to measure one or more particular features of a sample in response to a specific perturbation [10]. Hence, the identity of a feature is typically of particular importance and object to targeted analysis and further investigations. In contrast, profiling experiments capture a wide range of multiplexed readouts without prior knowledge and use machine learning and data mining techniques to identify similarities and differences among the measured patterns. Thus, the particular measured features themselves become relevant only when informative similarities or differences in patterns have been identified. Moreover, one can view profiling as an unbiased approach to group samples and perturbations. It therefore has a higher chance to capture unknown mechanisms compared to methods which depends on a biologist's expertise to interrogate a particular phenomenon (like in screening).

A standard profiling experiment consists of hundreds of plates, each of them can create more than 500 million single-cell measurements. Finding an optimal representation for the large amount of raw single-cell measurements is an open research question. A good profile not only has to be able to capture as much information about the biological perturbation as possible, but should also be robust against potential nuisance in the data (i.e experimental artifacts, biological noise).

2.1.1. Applications for Image-Based Profiling

Image-based profiling can be used for a wide-range of applications and the potential impact is immense. Studies in this field are varying from proof-of-principle to biological discovery.

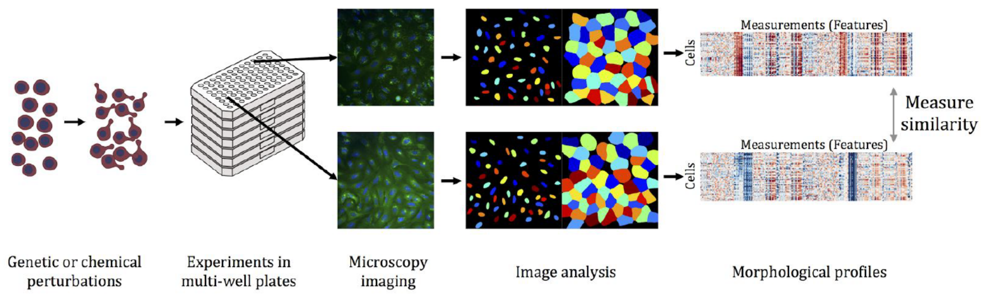


Figure 2.1.: **Current image-based profiling workflow** - The current workflow consists of following steps: data generation, feature extraction and profile calculation. Usually high-throughput experiments are used to generate microscopy images. Automated software such as CellProfiler then extracts hand-crafted features from the image set using image analysis. Finally profiles are extracted from the features and predictions are made by using similarity measurements.

Drug discovery: Identifying mechanism of actions

In this thesis, we will focus on one of the important applications of image-based profiling: Identifying biological mechanism of actions (MOA). A mechanism of action (e.g DNA damage, DNA replication, etc.) is a biological response which is induced by (or correlates with) a chemical perturbation. Being able to determine and group chemical compounds into similar biological mechanism of actions is an important step to discover new drugs [21] [22] and make predictions about novel and unknown compounds [12].

A substantial percentage of perturbations can produce morphological changes detectable by microscopy [1]. These changes can reveal similarities among compounds in terms of their phenotypic impact on cell populations. Many studies have demonstrated that morphological profiles can correctly predict mechanism of action for compounds, by grouping each unknown compound with already-annotated compounds, based on their phenotypic similarity.

Other applications

However, it should be noted that image-based profiling of chemical perturbations such as natural products, drugs, chemical compounds, has also been used to identify target molecules [13], lead hopping [2] and create small molecule enrichment libraries [1]. In functional genomics, image-based profiling of genetic perturbations (such as genetic overexpression, CRISPR, RNAi and deletion strains) has been used to identify genetic regulators [23], grouping disease-associated alleles [24], and disease-specific phenotypes [25].

2.1.2. Workflow using Expert-Engineered Features

The traditional workflow for image-based profiling is described in Figure 2.1. It can be categorized into four steps: data generation, feature extraction, profile calculation and prediction.

Data generation

Profiling experiments are usually large-scale high-throughput and involve imaging multi-well plates containing several stained samples of interest per well [26]. Biological samples are treated either with a compound concentrations of interest or DMSO, which serves as a negative control.

Each well is screened in regular time steps and generates nine (three by three) images per screening. It is important to keep track which images are sampled from the same well (or plate) because these images tend to correlate in feature space (inter-well and inter-plate variation). Another challenge biologists have to face are experimental artifacts. For example, wells located on the edge of a plate have a slightly lower temperature causing cells in these samples to behave differently (known as batch effect [4]).

After the data is generated, profiling becomes solely a computational task.

Feature extraction using CellProfiler

CellProfiler [15] [16] is a widely-used open-source image analysis software to quantitatively measure phenotypes from thousands of images automatically. CellProfiler is able to efficiently identify and segment single cells from image datasets. However, due to its use of traditional computer vision algorithms, such as Otsu's method [27], CellProfiler requires various user-defined inputs (e.g segmentation thresholds). After extracting each cell from the image, CellProfiler measures hundreds of features (i.e. cell texture, intensities, correlations, area, shape and size) and store them into a database. In the following chapters, we will refer to these features as CellProfiler (CP) features for clarity. These features are expert-engineered and several studies in the past have proven their usefulness for screening and profiling experiments. A complete list of all CP features for three channels (DNA, Actin, Tubulin) can be found in supplemental material of [3].

Profile calculation

Before applying any profiling methods, each feature is scaled such that the 1st percentile of DMSO-treated cells (negative control) is set to zero and and the 99th percentile is set to one for each plate separately in order to remove inter-plate variation.

There are several methods to compute per-sample profiles from single-cell measurements:

- **Means** - The average is taken over all scaled features for each sample. Variations of this method include taking medians, modes, or means combined with standard deviations.

- **KS statistic** - The KS statistic [28] is calculated by taking the maximum distance between the empirical cumulative distribution functions. The i-th element of the per-sample profile is the Kolmogorov-Smirnov (KS) statistic between the distribution of the i-th measurement of the cells in the sample with reference to the negative control on the same plate.

- **SVM** - Support Vector Machines [29] are trained to distinguish cells in each sample from negative control on the same plate. The normal vector of the seperating hyperplane is adopted as a profile of the sample.
- **Gaussian mixture (GM) modeling** - In order to better characterize heterogeneous cell populations, Slack *et al.* [30] proposed to model the data as a mixture of a small number of Gaussian distributions and profile each sample by the mean probabilities of its cells belonging to each of the Gaussians. The data was fit using the expectation-maximization (EM) algorithm and the best number of Gaussians was chosen empirically and by hand.
- **Factor analysis** - Although profiling captures many morphological features of each cell, it is the underlying biological effects that are of interest. Young *et al.* used factor analysis to discover such underlying effects. This method attempts to describe the covariance relationships between the image measurements x in terms of a few latent random variables y called factors.

Means and KS statistic can directly compute per-sample profiles while SVM and factor analysis requires a subset of the data for supervised training. Gaussian mixture models aim to incorporate subpopulation information into profiles but require manual model selection in order to determine the best number of Gaussians.

In the only comparison between current profiling methods [3], the means method outperforms the other presented methods. A detailed review of the results can be found in Section 3.1. This is surprising because the means method seems to throw away valuable single cell information by taking the population average instead of leveraging cellular heterogeneity (like Gaussian mixture modeling).

Prediction

After computing per-sample profiles, we can use a distance metric (cosine, euclidean, manhattan) between the profiles as a measure of similiarity. Each sample is predicted to have the class of the closest profile (nearest-neighbor classification). Clusters of profiles can be either detected via k-means or visualized via PCA or t-SNE.

2.1.3. Advantages and Challenges

Until recently, only a single modality - mRNA profiling - has been feasible in high-throughput. Gene-expression profiling has seen many successes, but is limited to detecting mRNA levels only. Biological images contain far more information than is typically harvested. Furthermore, image-based profiling has several promising advantages:

- **Reproducibility** - It is rapidly becoming clear that microscopy images hold sufficiently rich and reproducible quantitative information [31].
- **Single-cell resolution** - Unlike mRNA profiling, imaging offers single-cell resolution, capturing population heterogeneity and perturbations that affect small subsets of cells.

- **Usefulness** - Centuries of experience testify to the usefulness of visual phenotypes for interrogating biological processes, pathways, and complex disease processes.
- **Low-cost** - Morphological profiling is currently a fraction (25%) the cost of high-throughput gene expression profiling (L1000 or RNA-Seq).
- **Not redundant** - Morphological profiling is not redundant with gene expression profiling. Initial data from both small molecules and gene overexpression indicates the two readouts capture different information about cell state.

However the field of image-based profiling faces multiple computational challenges:

- **Optimal representation** - How to map raw, single-cell measurements to optimally useful perturbation profiles is an open research question. There is a need for further research on methods for capturing heterogeneity in profiles.
- **Feature redundancy** - CellProfiler features themselves are typically redundant. Furthermore, due to its non-linear dependencies, it is difficult to identify these associations, thereby making feature selection non-trivial. Identifying appropriate similarity measures and dimensionality reduction methods for morphological profiles is an open problem.
- **Manual labeling** - Although there is a wealth of raw data, cell-level labels are scarce and tedious to generate, making research on heterogeneity profiles difficult.

2.2. Deep Learning

Computer vision is currently experiencing a paradigm shift, rapidly moving from traditional feature engineering to modern feature learning. Since Krizhevsky *et al.*'s deep neural network won the ImageNet competition in 2012 [32], representation learning using deep learning has become the predominant strategy in computer vision tasks such as object detection [9] or segmentation [33]. Intriguingly, different flavors of deep neural networks also proved to be superior in a variety of tasks such as natural language processing and speech recognition [34], machine translation [35], automatic image captioning [36], novel art generation [37], human-level video game play [38] and even mastering the game of Go [39]. In this section, we will go into details of a feed-forward neural network and talk about tricks of the trade such as batch normalization and initialization. Furthermore, we will introduce two extensions of the basic feed-forward neural network architecture: convolutional neural networks for computer vision [40] and residual learning for training very deep neural networks [9].

2.2.1. Feed-Forward Neural Networks

A feed-forward neural network consist of one input layer, H hidden layers and one output layer. Each layer consist of units (or neurons) where each unit of a given layer is fully-connected to every unit of the previous layer. See Figure 2.2 for a schematic view.

The graphical representation can be described mathematically as a combination of matrix multiplication and activation functions. The activation function models when an artificial neuron fires and is usually a non-linearity. In theory, it is this non-linear activation function, which allows the network to learn any function approximation. Each unit in a hidden layer can be mathematically described by:

$$y_j = f(z_j) \tag{2.1}$$

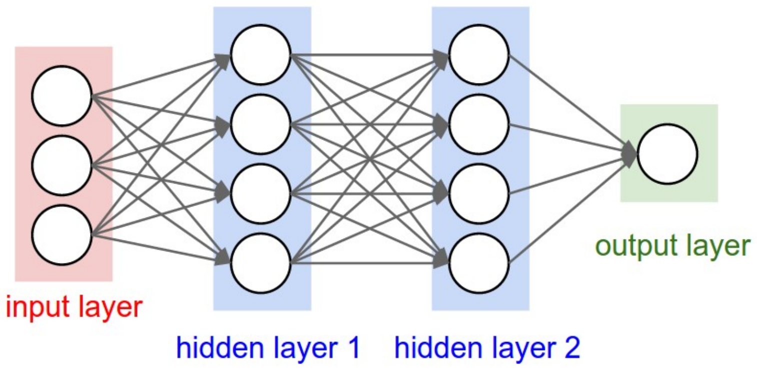


Figure 2.2.: **Schematic view of a FCN**. Schematic of a simple two layer fully connected neural network (FCN). The red marked units are the inputs. The blue marked units represent the hidden layer. The green units represent the output layer. Each arrow corresponds to a weight connecting the units. Image reprinted from [41]

$$z_j = \sum_i w_{ij} x_i + b_j \tag{2.2}$$

where y_j is the output of the unit from the current layer, $f(\cdot)$ is a (non-linear) activation function, z is called the preactivation, w denotes the edge weights and x_i is the output of a unit from the previous layer. At each layer, we first compute the total input z to each unit, which is a weighted sum of the outputs of the units in the layer below. Then a non-linear function $f(\cdot)$ is applied to z to get the output of the unit.

Any differential function can serve as activation function of a neural network. Some common activation functions are the sigmoid, $f(z) = 1/(1 + exp(-z))$, the hyperbolic tangent, $f(z) = (\exp(z) - \exp(-z))/(\exp(z) + \exp(-z))$, or the rectifier linear unit (ReLU). Unless stated otherwise, we will use ReLU (see Equation 2.3) as activation function throughout this work.

$$f(z) = \max(0, z) \tag{2.3}$$

Training a neural network consists of iterating between two steps: A forward pass (see Figure 2.2 a), followed by error backpropagation (see Figure 2.2 b). Given the input, a forward pass calculates the output of each unit. A predefined cost function E then compares the resulting outputs y_{out} with the correct answer and determines the error derivatives. Common cost functions are mean squared error or crossentropy loss.

Initialization

In order to be able to train a deep neural network, it is important to correctly initialize the weights and introduce a source of asymmetry between neurons to enable learning and prevent vanishing gradients (as discussed in detail in Subsection 2.2.3). In the following we will describe some common initialization strategies:

- **Zero-initialization** - With proper data processing, approximately half of the weights will be positive and half of them will be negative. A reasonable-sounding idea then might be to set all the initial weights to zero. Unfortunately this would cause every neuron in the network to compute the same output, thus undergo the exact same parameter updates and learning would fail.

- **Random initialization** - Weights are all randomly sampled (e.g from a Gaussian), close to zero and unique in the beginning, thus computing distinct updates.

- **Xavier initialization** - One problem with random initialization is that the distribution of the outputs from a randomly initialized neuron has a variance that grows with the number of inputs [42] . Glorot *et al.* hence recommend to initialize from a gaussian distribution with zero mean and Var$(w) = 2/(n_{in} + n_{out})$ where n_{in}, n_{out} are the number of units in the previous layer and the next layer.

- **He initialization** - Based on the Xavier initialization, a recent paper by He *et al.* [42] derives an initialization specifically for ReLU neurons and suggest to initialize the weights from a gaussian distribution with zero mean and Var$(w) = 2/n_{in}$. We will use He intialization throughout the thesis.

Batch normalization

Normalization (zero-mean and unit variance) is one of the most important data preprocessing steps, making the data comparable across features. However, as the data propagates through a deep network, the weights and parameters adjust those values, sometimes making the data too big or too small again - a problem known as "internal covariate shift". Batch normalization [43] avoids this problem by normalizing the data in each mini-batch:

$$x \Rightarrow \hat{x} = \frac{x - \mu}{\sigma} \Rightarrow x_{norm} = \gamma\hat{x} + \beta \tag{2.4}$$

In Equation 2.4, μ and σ are mean and standard deviation of a mini-batch (during training) or the whole training set (during test time) , γ and β are scale and shift parameters, which are learnable (analogous to weights). Batch normalization greatly accelerates training time and makes a deep network less sensitive to initialization issues.

Backpropagation

Deep neural networks learn via iterative backpropagation. Backpropagation is a gradient based method that uses the chain rule of derivatives

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x} \tag{2.5}$$

to obtain updates for the weights based on the loss. In order to teach our neural network the desired output y_{out}, we optimize the cost function E using gradient descent. The gradient of the loss with respect to the weights W_l of a layer l and a neural network with depth n can then be found by using the chain rule:

$$\Delta_W E = \frac{\partial E}{\partial W_l} = \frac{\partial E}{\partial y_{out}}\frac{\partial y_{out}}{\partial z_{out}}\frac{\partial z_{out}}{\partial y_{n-1}}\frac{\partial y_{n-1}}{\partial z_{n-1}}\cdots\frac{\partial z_l}{\partial W_l} \tag{2.6}$$

Each term can be evaluated independently:

$$\frac{\partial y_l}{\partial z_l} = \frac{\partial f(z_l)}{\partial z_l} \tag{2.7}$$

$$\frac{\partial z_l}{\partial y_{l-1}} = W_l \tag{2.8}$$

$$\frac{\partial z_l}{\partial W_l} = y_{l-1} \tag{2.9}$$

Using those results one can easily multiply the derivatives to obtain the gradient $\Delta_W E(y_{out})$. This gradient can then be used to perform a gradient descent update:

$$W_l^{t+1} = W_l^t - \eta\Delta_{W_l^t}E(y_{out}) \tag{2.10}$$

where W^t, W^{t+1} are the current weight matrix and weight matrix after the update, respectively and η is the learning rate.

Weight update

Normally, it is inefficient to calculate the gradient over the whole dataset to obtain the true gradient. Because of that, neural networks often use mini-batches to approximate the true gradient with random sampled data points. Once the analytic gradient is computed with backpropagation, the gradients are used to perform a parameter update. There are several approaches for performing the update:

- **Stochastic gradient descent (SGD)** - is the simplest possible update and changes the parameters along the negative gradient direction.
- **Momentum** - is another approach that almost always enjoys better converge rates on deep networks than SGD. Instead of directly integrating the position (like in SGD), the gradient only influences the "velocity" term, which in turn has an effect on the position.
- **Nesterov Momentum** - is a slightly different version of the momentum update [44]. The main idea is instead of evaluating the gradient at the current position, we first perform a momentum step and then evaluate the gradient at a "looked-ahead" position. Due to its stronger theoretical converge guarantees for convex functions, we will use Nesterov momentum as our default parameter update in this thesis (unless stated otherwise).

We note that optimization for deep networks is currently a very active area of research and there are a variety of different strategies besides SGD, such as second-order methods (L-BFGS) and per-parameter adaptive learning rate methods (RMSprop [45], Adam [46])

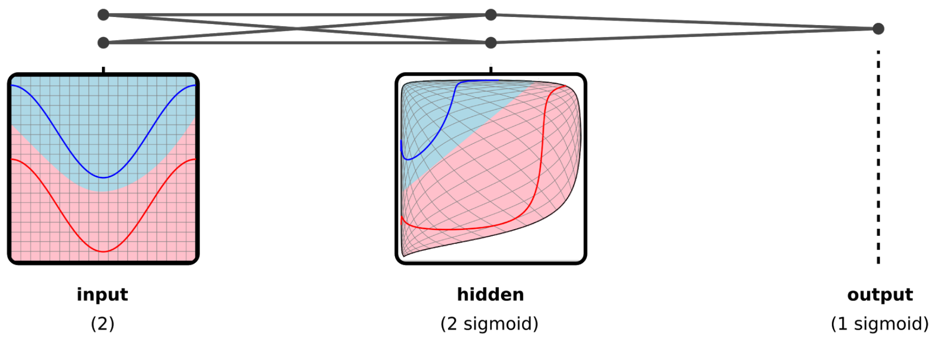


Figure 2.3.: **Representation learning** - In this illustrative example we use two input units, two hidden units and one output unit as our feed-forward network (visualized by the connected dots). The task is to classify the values of a two-dimensional function according to the blue or red class. One can see how the network draws a non-linear decision boundary in input space by transforming the input space into a hidden space in which the data is linearly separable. This can be seen beneath the hidden layer of the neural network. Image reprinted with permission from C. Olah (`http://colah.github.io/`)

Classification

One of the most common problems for neural networks is the classification of data into given classes. To solve this task, neural networks use its hidden layers as multiple abstractions to transform the data into a space in which the data points are linearly separable. One can think of the hidden layers of a classification network as a feature extraction algorithm. Figure 2.3 shows a simple two layer network with sigmoid output layer used for classification. The network can distort the input space to make the classes of data linearly separable.

2.2.2. Convolutional Neural Networks

Inspired by the biological receptive field in our eyes, convolutional neural networks (CNN) are the standard neural networks for computer vision. CNNs incorporate spatial invariance in its architecture (see Figure 2.4) by applying multiple convolutional layers to the image, while filtering the information using pooling layers. Invariance to spatial information is important for tasks like object recognition, where we only care about the presence of an object independent of its location. Convolutions are a sequence of inner products of a given filter (or kernel) with pieces of a larger image. Mathematically a one-dimensional discrete convolution of a kernel vector g of size over a vector f at position c can be expressed as:

$$(f * g)(c) = \sum_a f(a)g(c-a) \tag{2.11}$$

Equation 2.11 is highly parallelizable (thus optimal for GPUs), since the kernel is the same throughout the image. We can think of a convolution as sliding one function on top of another, multiplying and then adding.

Convolutional layer

In CNNs, convolutions are stacked to extract multiple layers of spatial related features. Each convolutional layer of a neural network extracts multiple so-called feature maps. Each feature map relates to a separate convolutional kernel which is applied to the input

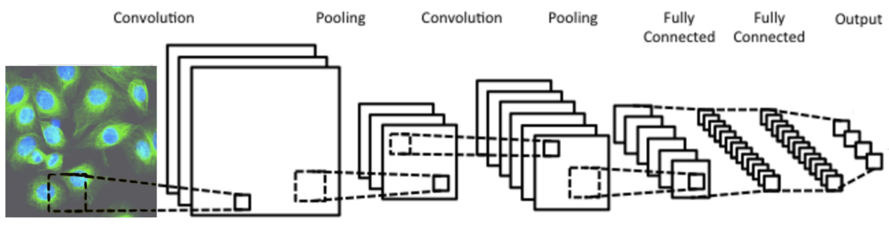


Figure 2.4.: **Architecture of a convolutional neural network** - Convolutional neural networks (CNN) consist of convolutional and pooling layers for feature learning, followed by a fully connected network for classification.

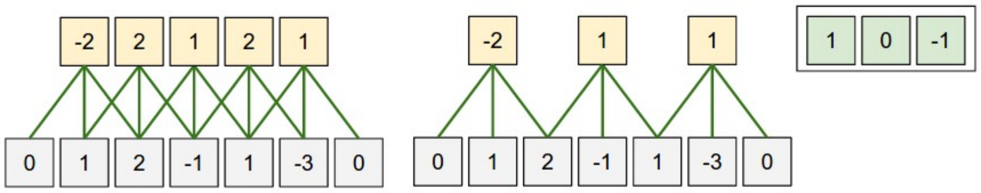


Figure 2.5.: **1-d convolution with stride** - 1-d convolution of a kernel (green) on input (gray) using stride set to one (left) and stride set to two (right). Image reprinted from [41].

of the layer. A stride parameter defines how far to move the filter between each step of the convolution (see Figure 2.5). Similar to feed-forward neural networks, the weights are initialized and learned via backpropagation (see Section 2.2.1).

As already demonstrated in Figure 2.3 for feed-forward networks, a CNN uses its hidden layers as feature extraction. The output (not the filters) of each layer is a feature map corresponding to the output for one of the learned features, detected at each of the image positions. Zeiler *et al.* [47] showed that earlier convolutional layers act as lower-level feature extractor (i.e. oriented edge detection), while later layers are able to detect high-level features (i.e. detecting a wheel from a car).

Pooling layer

Another important building block for CNNs is the pooling layer. A pooling layer reduces the size of the feature maps, thus reducing the model size and keeping computations tractable. One simple version is the max-pooling layer, which uses a sliding window over the input and selects the maximum of each window (see Figure 2.6).

Data augmentation

Convolutional neural networks tend to overfit on small datasets. To overcome overfitting, it is common to increase the size of the dataset using data augmentation. One augmentation method is to add cropped versions of each image and their horizontal flipped images

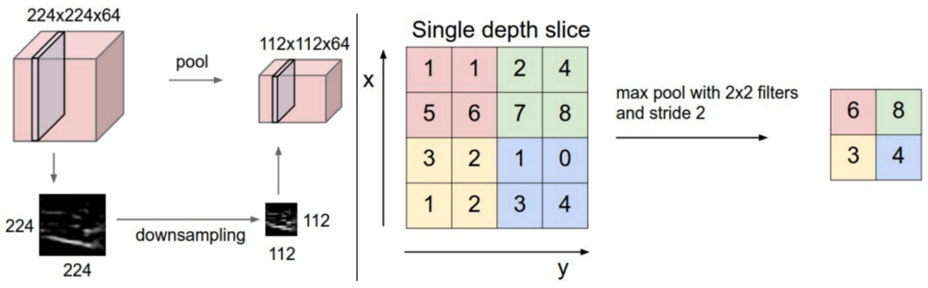


Figure 2.6.: **Pooling layer** - Stacked convolutions can cause very large feature maps, making the next convolution computationally intractable. Max pooling is a simple and effective way to downsample the feature maps. Image reprinted from [41]

to the dataset. To save memory, this method is usually performed online while training the CNN. Before passing the input to the hidden layers, the neural network will instead pass all images to a data transformation unit. The unit then flips the image horizontally with probability 0.5. Afterwards, it randomly crops fixed tiles of the image and rescales it back to its original size. The new augmented input is then given to the neural network as training data.

Model selection & VGG architecture

CNNs introduce a number of additional hyperparameters such as kernel size, depth of the feature maps and numbers of convolutional and pooling layer. How to optimize these hyperparameters in a systematic way is not well understood. Strategies vary from random search through parameter space to bayesian optimization, where probabilistic methods are used to predict the best next hyperparameter to test given previously tested hyperparameters [48]. Thus, Simonyan *et al.* [40] introduced a simple deep convolutional architecture, VGG, which was able to win the second place in the 2014 ImageNet competition. A VGG network consists of blocks of three convolutional layers (with kernel size 3×3) followed by a max pooling layer. The corresponding feature map depth of each block increases linearly by power of two. For example for ImageNet (224×224 px input), VGG uses following feature map depths: $64, 128, 256$ and 512.

2.2.3. Residual Learning

Deeper neural networks are more difficult to train. Beyond a certain depth (usually around 10+ layers), traditional deeper networks start to show severe underfitting caused by two reasons:

- **Vanishing / exploding gradients** - gradients in deep neural networks tend to be unstable and either vanish [49] or explode in earlier layers, causing it to learn at wildly different speeds. This problem can be addressed with initialization techniques (see Section 2.2.1) that try to start the optimization process with an active set of neurons.
- **Harder optimization** - according to He *et al.* [9], when a model introduces more parameters, it becomes more difficult to train the network due to (yet unknown) optimization issues. This is not simply an overfitting problem, since adding more layers leads to even more training errors.

Residual learning is a recently introduced framework which eases the training of networks that are substantially deeper than those used in previous settings. It opens up the possibility to train very deep residual neural networks (ResNets) with compelling accuracy while keeping the complexity low. On the ImageNet dataset, He *et al.* evaluate residual nets with a depth of up to 152 layers (also called ResNet-152) and show that, although $8\times$ deeper than VGG nets [40], it has fewer parameters.

Residual layer

The main idea behind residual learning is simple: Instead of learning a new representation $H(x)$ at each layer, deep residual networks use identity mappings for every two layers to

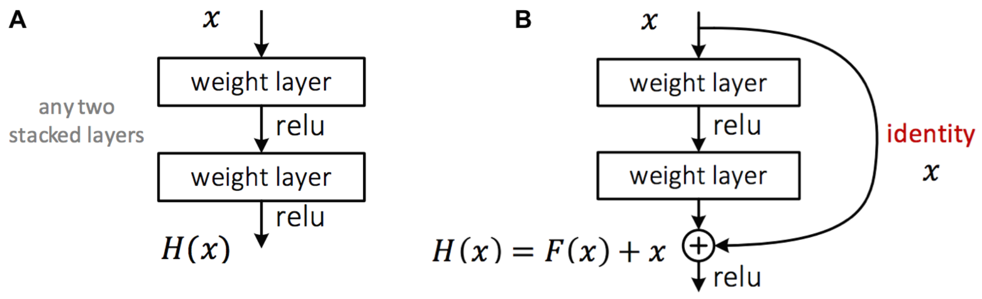


Figure 2.7.: **Residual layer architecture** - Given any two stacked layers, a residual layer introduces a identity connection (a simple addition) between the input and the output. Surprisingly, this simple trick will allow us to train very deep neural networks. Image reprinted from [9]

learn an "easier" residual function $F(x)$ where $H(x) = F(x) + x$ (see Figure 2.7). The hope is that the two weight layers fit rather residual $F(x)$ than $H(x)$. Furthermore, assuming that our data might have a strong linear component, residual connections allow us to incorporate this assumption directly into our network's architecture. Thus, if the identity mapping was the optimal transformation (or close to it), all weights are set to 0 (or to a small fluctuation around 0).

Smooth propagation

Another advantage of residual neural networks is their smooth forward and backward propagation. While plain neural networks have a multiplicative outcome: $x_L = \prod_{i=l}^{L-1} W_i x_L$, the forward propagation of a ResNet is additive:

$$x_L = x_l + \sum_{i=l}^{L-1} f(x_i) \tag{2.12}$$

Furthermore, using the insight that any x_L can be fully described by any x_l plus residual, one can easily derive an additive term for backward propagation:

$$\frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L}\frac{\partial x_L}{\partial x_l} = \frac{\partial E}{\partial x_L}(1 + \frac{\partial}{\partial x_l}\sum_{i=l}^{L-1} f(x_i)) \tag{2.13}$$

In contrast to the multiplicative term in a plain feed-forward network: $\frac{\partial E}{\partial x_l} = \prod_{i=l}^{L-1} W_i \frac{\partial E}{\partial x_L}$, any $\frac{\partial E}{\partial x_L}$ is directly back-prop to any derivative $\frac{\partial E}{\partial x_l}$ plus residual. Moreover, the additive property also makes the gradient unlikely to vanish, thus preserving it even through backward propagation of hundreds of layers.

Bottleneck architecture trick

In order to increase depth, while keeping the training time affordable, ResNets use a clever trick called *Bottleneck architecture* to replace two convolutional layers with three layers. For each residual function $F(x)$, He *et al.* use a stack of three layers instead of two (see Figure

2.7). The three layers have 1×1, 3×3, and 1×1 convolution kernels, where the 1×1 layers are responsible for reducing and then increasing (restoring) dimensions, leaving the 3×3 layer a bottleneck with smaller input/output dimensions. The advantage is that it is deeper than the two layer design while having similar complexity (number of parameters).

3. Related Work

This chapter presents related work in machine learning for image-based profiling. The first section focuses on traditional machine learning methods as presented in Ljosa *et al.* [3]. The author reviews different classical profiling methods which have been tested on the BBBC021 dataset. As seen in later sections, the results serve as useful baseline for comparisons. Second, we will explore how deep learning has been used to directly improve image-based profiling. However, limited research has been done in this field.

3.1. Comparison of Profiling Methods on BBBC021

Ljosa *et al.* [3] use the BBBC021 dataset and extracted single-cell features as described in Section 2.1. Proposed methods to generate treatment profiles for MOA classification are then compared using those features. Predictions about the MOA of each treatment is made using a nearest neighbor classifier based on the cosine distance. The authors evaluated the classifier by using the leave-one-compound-out cross-validation method, where all profiles of one compound are held out and the model was trained only on the remaining compounds. This is important because in the absence of a appropriate holdout strategy, a model is able to train on images of the same compound which introduces a significant bias in testing. The pipeline is shown in Figure 2.1.

The methods being compared are mean profiling, factor analysis, KS statistics, Gaussian Mixture Models and SVM, which we introduced in the previous section. Factor analysis was conducted on a subset of the features from the negative control set, which is not used for classification. The found factors were then used to transform the rest of the data. It turns out several methods from literature are not better than the simplest method, mean profiles. Mean profile classification achieved an MOA classification accuracy of 83%, whereas actor analysis followed by mean profiling achieved 94% (see Table 3.1). As mentioned in the previous chapter, these high accuracies are rather surprising due to the fact that population averaging seems to throw away a lot of valuable data about cellular het-

Method	Accuracy in %
Means	0.83
KS statistic	0.83
Normal vector to support vector machine hyperplane	0.81
Distribution over Gaussian mixture models	0.83
Means + FA analysis	0.94

Table 3.1.: Accuracies for classifying compound treatments into mechanism of actions as reported in Ljosa *et al.* [11]

erogeneity.

Even though population averaging already achieves high accuracies, there is reason to believe that methods, which can efficiently capture cellular heterogeneity, could increase the classification accuracy and make profiling more interpretable (see Chapter 5). This is because cellular subpopulations have been observed to be a fundamental property of cellular systems.

3.2. Autoencoders for MOA classification

First attempts on using deep learning for image-based profiling on BBBC021 were presented in a paper by Kandaswamy *et al.* [50]. The authors used stacked autoencoders for layer-wise pretraining of a neural network on single-cell CellProfiler features. The CellProfiler features were generated using the same workflow as described in Ljosa *et al.* [3].

The pretrained network is then used directly as a classifier to classify single cells into one of the given MOAs. Moreover, the authors experimented with transfer learning: The dataset was divided into two sets of six MOAs each. Using this setup, they tested how transferable features learned on one set were for the classification of the other set.

Using leave-one-compound-out cross-validation, the neural network was able to achieve a classification accuracy of 87.9%. This outperforms most classic profiling method but is more complex and worse than the presented factor analysis approach. Furthermore, it is argued that the division of the data to form a transfer learning task shows that pretrained models can be used to save time of training the model. However, it is still necessary to fine-tune the model on the new dataset.

3.3. Convolutional Neural Networks for MOA classification

A recent paper by Kraus *et al.* [18] presents a novel CNN pooling layer for image-based profiling. The main idea is to use a convolutional neural network as an initial feature extractor, which outputs multiple feature maps that correspond to the total number of classes present in the dataset (for BBBC021 that is the number of MOAs). A custom global pooling function $g(\cdot)$ was used to account for multiple instances of different classes and aggregate all instance probabilities (see Figure 3.1). The author argued that a noisy AND pooling function achieved the best results.

The noisy AND pooling was given by

$$P_i = g_i(p_{ij}) = \frac{\sigma(a(\bar{p}_{ij} - b_i)) - \sigma(-ab_i)}{\sigma(a(1 - b_i)) - \sigma(-ab_i)} \tag{3.1}$$

$$\bar{p}_{ij} = \frac{1}{|j|} \sum_j p_{ij} \tag{3.2}$$

and p_{ij} refers to the probability of an instance being present in feature map i at position j, a is a hyperparameter defining the sharpness of this transformation and b are the weights learned by the network.

Using this method, it is reported that a MOA classification accuracy of 98.8% was achieved. However, those results are not comparable with the results obtained by Ljosa *et al.* [3],

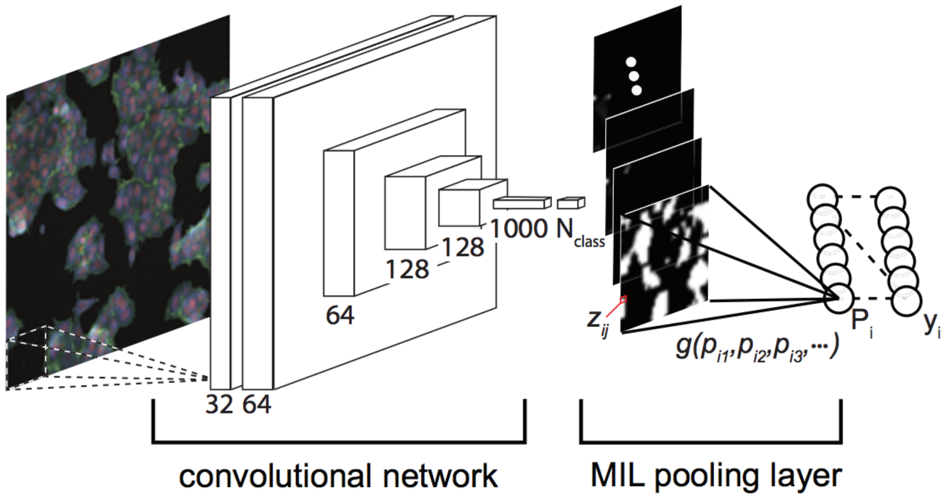


Figure 3.1.: **Schematic multiple instance learning model** - Used by Kraus *et al.*, a global pooling function aggregates instance probabilities from feature maps, extracted by the initial CNN. Image reprinted from [18]

because the method was tested without using leave-one-compound-out cross-validation. Furthermore, only a subset of the whole dataset was evaluated.

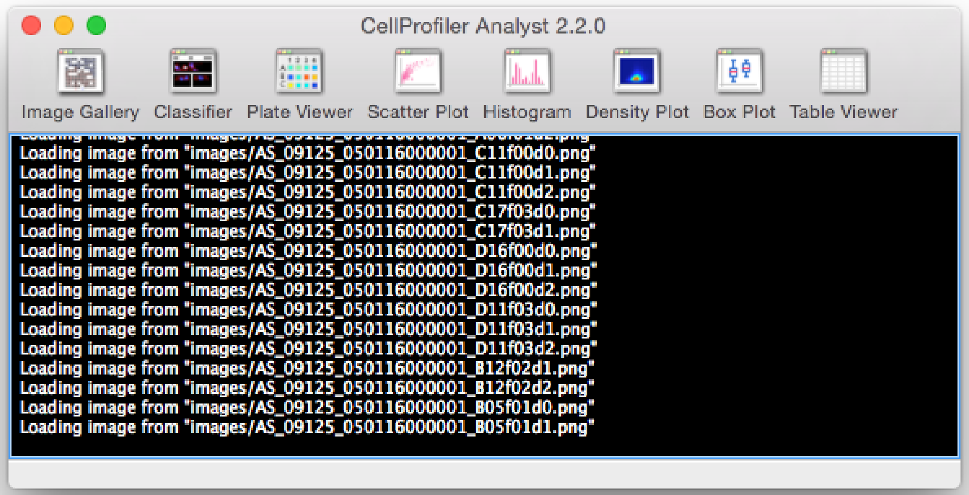
4. CellProfiler Analyst

CellProfiler Analyst allows the exploration and visualization of image-based data, together with the classification of complex biological phenotypes, via an interactive user interface designed for biologists and data scientists. In the later chapters, we will extensively use CellProfiler Analyst in our workflow to extract subpopulation phenotypes and train baseline classifiers.

Thus, the goal of this chapter is to introduce the reader to CellProfiler Analyst and present (1) an overview of its functionality and (2) explore its key features, Image Gallery and Classifier, in more detail.

4.1. System overview

CellProfiler Analyst is an open-source software for biological image-based classification, data exploration and visualization with an interactive graphical user interface. Using data from feature extraction software such as CellProfiler, CellProfiler Analyst offers easy-to-use tools for exploration and mining of image data, which is being generated in ever in-



```
Loading image from "images/AS_09125_050116000001_C11f00d0.png"
Loading image from "images/AS_09125_050116000001_C11f00d1.png"
Loading image from "images/AS_09125_050116000001_C11f00d2.png"
Loading image from "images/AS_09125_050116000001_C17f03d0.png"
Loading image from "images/AS_09125_050116000001_C17f03d1.png"
Loading image from "images/AS_09125_050116000001_D16f00d0.png"
Loading image from "images/AS_09125_050116000001_D16f00d1.png"
Loading image from "images/AS_09125_050116000001_D16f00d2.png"
Loading image from "images/AS_09125_050116000001_D11f03d0.png"
Loading image from "images/AS_09125_050116000001_D11f03d1.png"
Loading image from "images/AS_09125_050116000001_D11f03d2.png"
Loading image from "images/AS_09125_050116000001_B12f02d1.png"
Loading image from "images/AS_09125_050116000001_B12f02d2.png"
Loading image from "images/AS_09125_050116000001_B05f01d0.png"
Loading image from "images/AS_09125_050116000001_B05f01d1.png"
```

Figure 4.1.: **Main view** - CPA offers a wide variety of tools such as Image Gallery, Classifier, Plate Viewer and standard plotting tools. They can be navigated via the main view.

creasing amounts, particularly in image-based profiling. Its tools can help identify complex and subtle phenotypes, improve quality control and provide single-cell and population-level information from experiments. CellProfiler Analyst 2.0, completely rewritten in Python, builds on these features and adds enhanced supervised machine learning capabilities (Classifier), as well as visualization tools to overview an experiment (Plate Viewer and Image Gallery).

Compared to other commonly-cited open-source biological image classification software like Ilastik [51], CellCognition [52] and WND-CHARM [53], CellProfiler Analyst has the advantage of containing companion visualization tools, being suitable for high-throughput datasets, having multiple classifier options, and allowing both cell and field-of-view classification. Advanced Cell Classifier [54] shares many of the classification features of CellProfiler Analyst, but it lacks HCS data exploration and visualization tools. Compared to command-line-based data exploration software like cellHTS [55] and imageHTS [56] and the web tool web CellHTS2 [57], CellProfiler Analyst provides interactive object classification and image viewing. Several other software tools (e.g. the HCDC set of modules for KNIME [58]) are no longer available/maintained.

The software is free and open source, available at `http://www.cellprofiler.org` and from GitHub under the BSD-3 license. It is available as a packaged application for Mac OS X and Microsoft Windows and can be compiled for Linux. We implemented an automatic build process that supports nightly updates and regular release cycles for the software.

Furthermore, we provide a complete and updated manual for CellProfiler Analyst 2.0 at `http://cellprofiler.org/cpa/`.

4.2. Visualization tools

Many large-scale imaging experiments take place in multi-well plate format. Researchers are often interested in seeing their data overlaid on this format, to check for systematic sample quality issues, or to see results from controls placed in particular locations, at a glance. CellProfiler Analyst's Plate Viewer tool displays aggregated and/or filtered measurements (according to customizable color maps) or a thumbnail image for each well. Automatically imported annotations can be viewed, and individual annotations can be manually added or deleted for each well.

In addition to Plate Viewer and standard visualization/exploration tools such as histograms, scatter and box plots, CellProfiler Analyst 2.0 also offers a convenient Image Gallery tool (see Figure 4.2). Image Gallery provides a grid view allowing an overview of images from imaging experiments. A variety of options are provided to filter images based on experiment-specific metadata, e.g. gene name, compound treatments, compound concentration. Multiple filters can be combined to refine the search. Images can be displayed and exported as a custom-sized thumbnail or in full resolution, and the color assigned to each channel in the image can be customized to highlight structures of interest. Individual segmented cells can be extracted and viewed for each image, and can be dragged and dropped into the Classifier window. This is especially useful in order to quickly generate deep learning training sets.

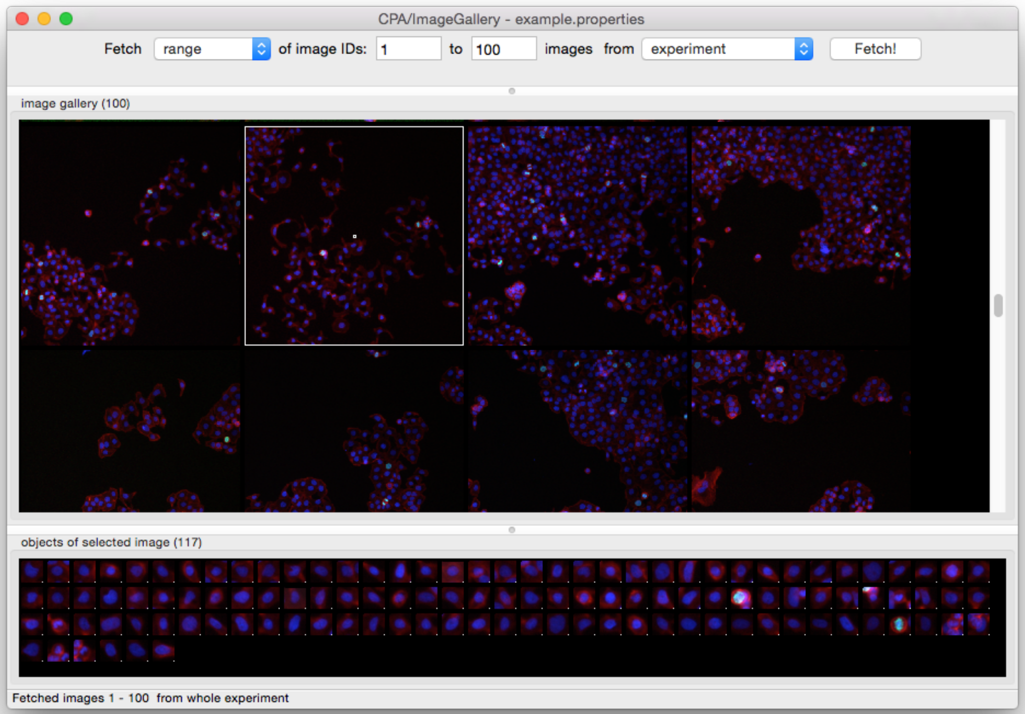
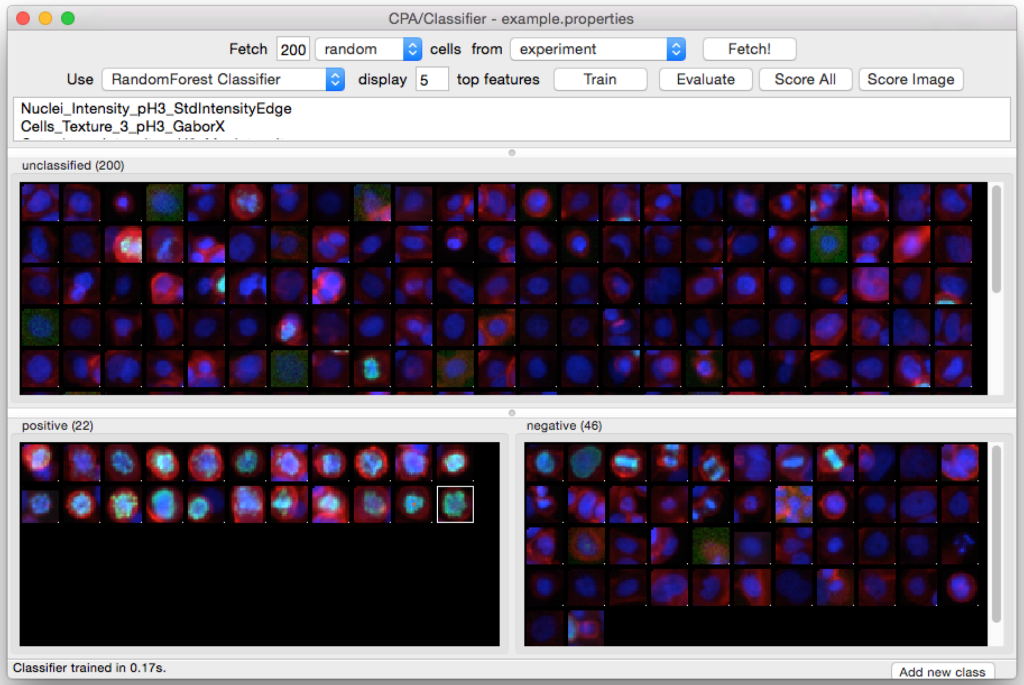

CPA/ImageGallery - example.properties
Fetch
range
of image IDs:
1
to
100
images
from
experiment
Fetch!
image gallery (100)
objects of selected image (117)
Fetched images 1 - 100 from whole experiment

Figure 4.2.: **Image Gallery** - CellProfiler Analyst's gallery view allows to explore whole images in an experiment, and cells in individual images. The upper area displays full images from the imaging experiment. Using right-click on these images, one can display all its single cells as image tiles in the lower area. Furthermore, it is possible to conveniently export single cell tiles in order to train machine learning models.

4.3. Classifier

CellProfiler Analyst 2.0's Classifier (see Figure 4.3) can perform cell and image-level classification of multiple phenotypes (multi-class) using popular models like Random Forest and LDA from the high performance machine learning library scikit-learn [20]. First, cell or whole image samples from the experiment are fetched and sorted by drag and drop into researcher-defined classes, making up the annotated training set. Fetching can be random, based on filters, based on per class predictions of an already-trained classifier, or based on active learning. The active learning option speeds annotation by presenting uncertain cases. In addition, researchers can view full images of each sample and drag and drop cells from the image for annotation. Next, a classifier is trained on this set. After training on the annotated set, a model's performance can be evaluated by a k-fold cross validation (where k is adjustable by the user) in the form of a confusion matrix and precision, recall and F1 score per class. The model can then be used to quantify cell phenotypes or whole-image phenotypes.


CPA/Classifier - example.properties
Fetch 200 random cells from experiment Fetch!
Use RandomForest Classifier display 5 top features Train Evaluate Score All Score Image
unclassified (200)
positive (22)
negative (46)
Add new class

Nuclei_Intensity_pH3_StdIntensityEdge
Cells_Texture_3_pH3_GaborX

Classifier trained in 0.17s.

Figure 4.3.: **Classifier** - CellProfiler Analyst's Classifier provides multiple machine learning algorithms that can be trained to identify multiple phenotypes in single cells or whole images, by using simple drag and drop of cell tiles. The biologist can fetch single cell data from the experiment using different strategies (random, predictive, active). The images will then be displayed in the upper area and can then be sorted into classes (lower area). This is useful for quickly hand-labeling a training set.

5. Subpopulation Profiling

A central challenge of biology is to understand how individual cells respond to perturbations. It is known that cell-to-cell differences are always present to some degree in any populations of ”seemingly identical” cells. Thus, heterogeneity has been long speculated to be a fundamental property of cellular systems and it has been observed for essentially all dimensions of single-cell measurements at high resolution.

Despite that, most prior image-based profiling ignore population heterogeneity (See Chapter 3). In fact, the common profiling approach is to aggregate and summarize the single-cell measurements feature-wise across the cells in a sample. However, mean profiles (averaging population distributions) can mask the presence of rare or small subpopulations of cells and ultimately diminish signals.

The idea of subpopulation profiling is to extract different modes of phenotypes in cells and represent a sample by proportions of cells it has in each category. Thus, subpopulation profiling faces two problems:

- How can we identify meaningful subpopulations of cells?
- How can we use subpopulation information to create profiles?

In this chapter, we introduce a supervised subpopulation workflow for image-based profiling using deep residual learning. First, we describe the previously published BBBC021 dataset. All experiments within this dissertation have been conducted on this dataset. Second, we describe how we use CellProfiler in combination with CellProfiler Analyst to identify and extract a subpopulation dataset of 2526 cells with 23 different nuclear phenotypes

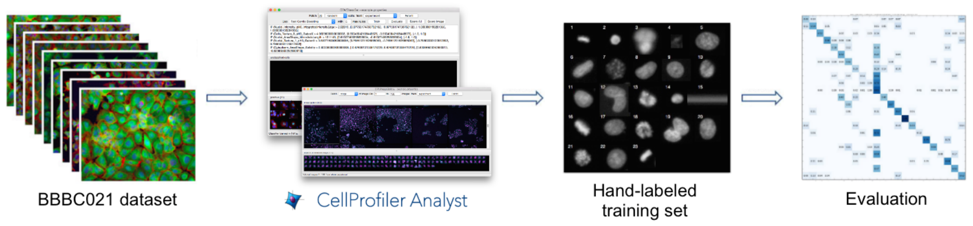


Figure 5.1.: **Workflow for supervised subpopulation training** - The BBBC021 dataset contains images of breast cancer cells which have been treated with 38 different compounds at various concentrations. We used CellProfiler Analyst (CPA) to identify 23 different nuclear phenotypes and create a training set containing 2526 hand-labeled cells. Furthermore, we evaluated the training set using CPA's classification algorithms.

from BBBC021. Afterwards, we train a deep residual neural network on the subpopulation dataset and classify all cells from BBBC021. The accuracy of the residual network is compare with several traditional machine learning methods from CellProfiler Analyst. Finally we leverage the extracted subpopulation information to create compressed and interpretable morphological profiles.

5.1. BBBC021 dataset

The BBBC021 dataset is a well studied benchmark imaging dataset, freely available from the Broad Bioimage Benchmark Collection (BBBC). It was originally created through a compound-profiling experiment to study the biological mechanism of action (MOA) induced by a single treatment (see Section 2.1.1). A treatment is specified by the compound and its concentration. The dataset contains 13200 images of human MCF7 cancer cells which have been treated with 38 different compounds at various concentrations (108 treatments in total). Additionally, each image contains three stains (DNA, Actin and Tubulin). Cells treated with DMSO serve as negative control.

An extensive comparison of traditional image-based profiling on this dataset are from

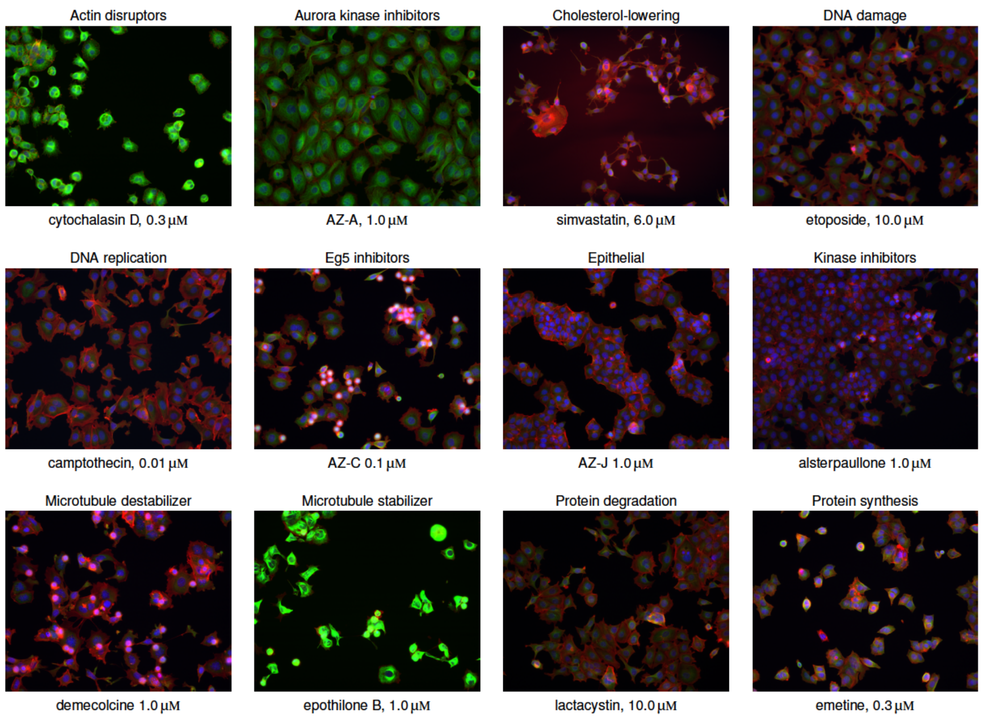


Figure 5.2.: **MOA classes in BBBC021** - BBBC021 image set contains 38 unique compounds that fall into 12 MOA classes. In this figure, we listed 12 compounds, each representing a distinct MOA. As one can see, MOAs are visually distinctive (Eg5 inhibitor and actin disrupter), while others are visually similar to each other (DNA damage and DNA replication). Red color shows actin stain, green shows tubulin stain, and blue shows DNA stain.

Ljosa *et al.* [3] as presented in Chapter 3.1. The corresponding deep learning results were published by Kraus *et al.* [18] as explained in Chapter 3.3.

A subset of the compound-concentrations have been identified as clearly having one of 12 (including DMSO/negative control) different primary mechanism of action (see Figure 5.2). The classification of different MOAs depends on the presence of visual differences in the microscopy images. This turns out to be a non-trivial task due to the fact that BBBC021 contains a hierarchical similarity structure: Although images with the same underlying MOA look similar, images from the same compound usually look more similar than images from another compound but with the same MOA. Furthermore, differences between phenotypes were in some cases very subtle: Ljosa *et al.*, who first published this dataset, were only able to identify 6 of the 12 mechanisms visually; the remainder were defined based on the literature.

In order to answer the question, if subpopulation profiles can improve image-based profiling, we devised the following test using BBBC021 as benchmark dataset.

Existing profiling methods (as described in Section 3.1), had achieved up to 94% accuracy in predicting compound's mechanism of action. In the remaining DNA channel, the visual differences were much more subtle. Using CellProfiler, due to only one information channel (instead of three) we were only able to extract 94 features (in comparison to 453 features for three channels). We reproduced Ljosa *et al.*'s experiment on this more difficult data set and observed that the accuracy of the two best methods, mean profiles and factor analysis, fell to 67% and 68% (see Table 5.1). Both methods did not model cell heterogeneity, as they were simply averages of image features across the cells (mean profile) or transformations of population averages (factor analysis). Hence, they serve as useful baseline for comparison.

5.2. Hand-Labeled Subpopulations

The following experiments do not use the raw images of the BBBC021 dataset because of their large size of $1280\text{px} \times 1024\text{px}$, which strains memory constraints and computational cost. Instead, CellProfiler is used to segment single cells within the images using Otsu's method. Using CellProfiler Analyst's Classifier (see Section 4.2), those segmentations are then used to crop single cells and center them into images of $96\text{px} \times 96\text{px}$. In total, we extracted up to 500000 single cells from the original dataset.

Using CellProfiler Analyst's visualization tools, we were able to sort 2526 nuclei into 21 morphological phenotypes (see Figure 5.4 and Appendix) and two artifactual classes (blurry and lines). A more detailed description of each class can be found in the appendix.

Method	Three channels (453 features)	DNA only (94 features)
Means	0.83	0.66
Means + FA analysis	0.94	0.68

Table 5.1.: We compares the original MOA accuracies of the two best methods as reported in Ljosa *et al* on BBBC021 for three channels (DNA, Actin, Tubulin) and our results for MOA classification (cosine distance) and only DNA information.

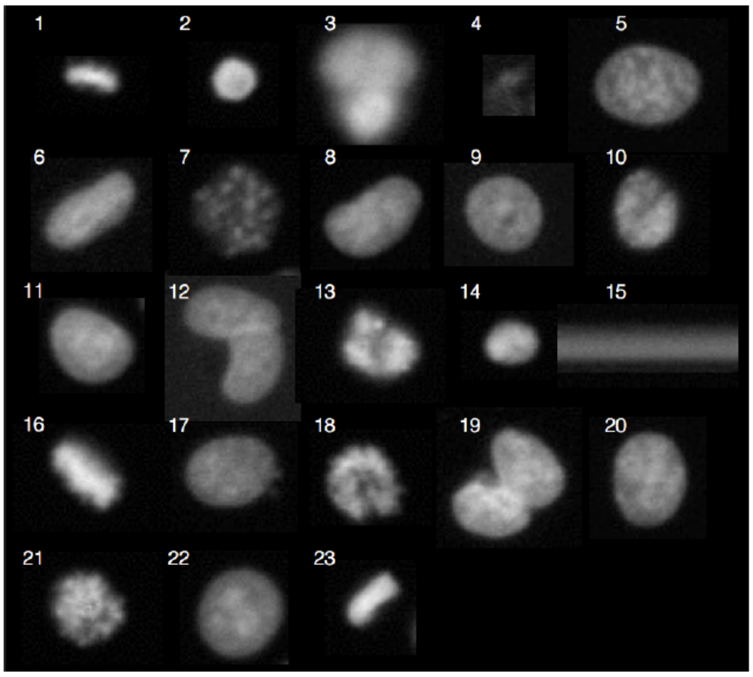


Figure 5.3.: **SUBPOP dataset** - 23 hand-labeled subpopulations extracted from BBBC021 image set: (1) anaphase, (2) apoptotic, (3) blurry, (4) debris, (5) early prophase, (6) elongated, (7) fragmented, (8) halfcircle, (9) holey, (10) indented, (11) interphase, (12) kidney, (13) late prophase/early anaphase, (14) late telophase, (15) lines, (16) metaphase, (17) micronucleus, (18) monopole, (19) multinucleate, (20) nucleolirim, (21) prophase, (22) round and (23) telophase

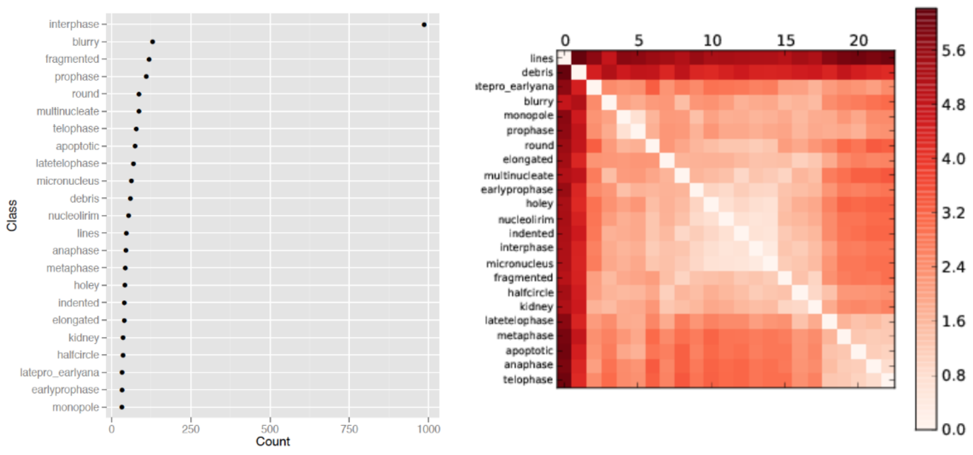


Figure 5.4.: **SUBPOP statistics** - Left: SUBPOP dataset is heavily unbalanced. There are about 20 times more interphase cells (991 cells) than monopole cels (52 cells). Right: Heatmap of euclidean distances between class means (CP features); Clusters between classes suggest high visual similarity.

Figure 5.3 displays a collection of image tile representing each of the 23 subpopulation classes. Hand-labeling cells, though tedious and non-exhaustive compared to fully automated methods, leverages researchers' knowledge. Furthermore, CellProfiler Analyst can help to semi-automate labeling (and thus speed up the process) using expert-guided iterative machine learning (see Section 4.3).

For simplicity, we will refer to the hand-labeled subpopulation dataset as SUBPOP dataset. The dataset is heavily unbalanced because some phenotypes are much rarer than others. The left part of Figure 5.4 visualizes the counts for each class. As one can see, cells labeled with interphase have about 20 times more samples than cells labeled with monopole. Visually, especially for a non-expert, it is very difficult to distinguish certain classes. For example, Figure 5.3 suggest visual similarities between interphase and micronucleus. Using the mean of all CP features, extracted from the image tiles using CellProfiler, we can plot a euclidean distance matrix (see right part of Figure 5.4). This helps us evaluate the quality of our dataset.

As we can see from Figure 5.4, artifactual classes (lines and blurry) are visually very distinct from other classes. Unfortunately, the plot also confirms our belief that interphase, the largest class in our dataset, has a low euclidean distance (thus is visually similar) towards many other classes (i.e indented and micronucleus).

5.3. Classification using CellProfiler Analyst

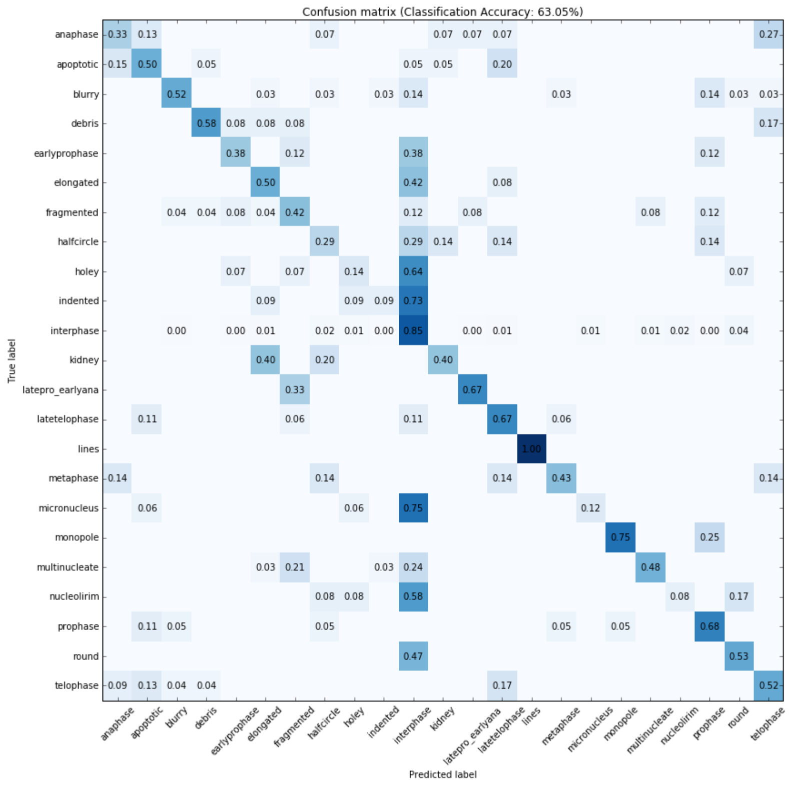
Our aim is to train a classifier to distinguish the 23 morphological phenotypes. The trained classifier is then used to sort all cells in BBBC021 into one of the 23 classes. Thus it is important to train a good classifier. In order to evaluate the performance of the classifier, we train two models, linear discriminant analysis and random forests, as baseline experiment. For both methods, we use a 5 fold cross-validation on SUBPOP training set. CellProfiler Analyst provides us with a unified framework to train and evaluate classification algorithms from scikit-learn without writing a single line of code (see Section 4.3).

5.3.1. Linear Discriminant Analysis

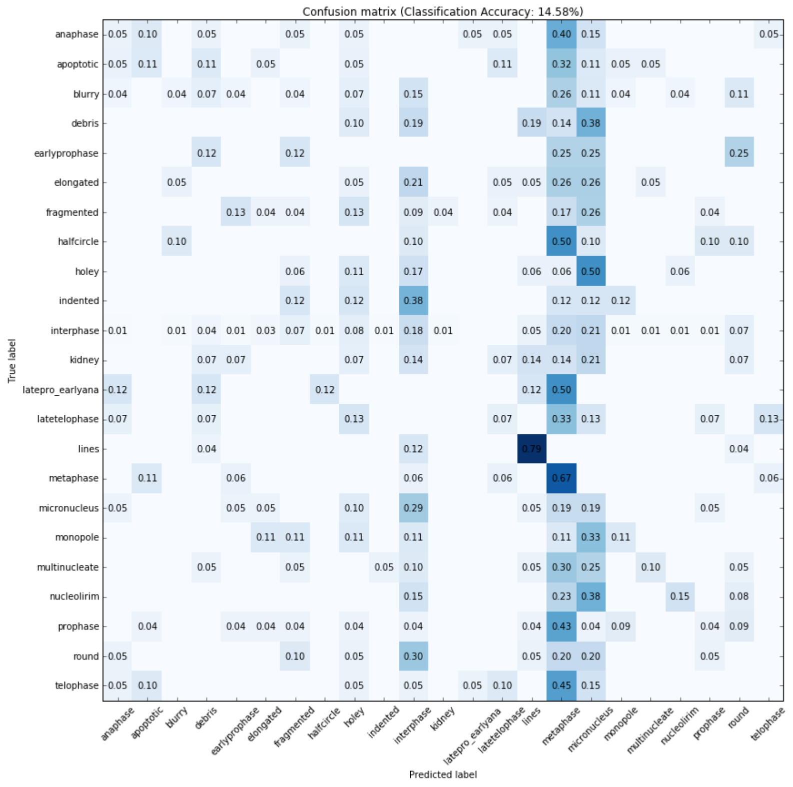
Linear Discriminant Analysis (LDA) is often used as baseline algorithm. LDA uses a linear decision boundary, generated by fitting class conditional densities to the data $P(X|y=k)$ for each class k and using Bayes rule:

$$P(y=k|X) = \frac{P(X|y=k)P(y=k)}{P(X)} = \frac{P(X|y=k)P(y=k)}{\sum_l P(X|y=l) \cdot P(y=l)} \tag{5.1}$$

The model fits a Gaussian density to each class, assuming that all classes share the same covariance matrix. LDA has proven to work well in practice and is inherently multiclass. Furthermore, it is fast to evaluate due to its closed-form solution and no need for hyperparameters. Figure 5.5 shows the confusion matrices for LDA trained on SUBPOP using a 5-fold crossvalidation. Using CellProfiler features (94 expert-engineered features), LDA was able to correctly predict 63% of all cell phenotypes. However, as expected, LDA confuses interphase with micronucleus and intended (see Figure 5.5a). This is due to the heavily unbalanced dataset. There is much less class data for micronucleus available than for interphase. Thus, over 75% of all micronucleus cells get misclassified as interphase cells. Next, we used raw pixel data ($96 * 96$ pixels) as input. LDA is not able to seperate morphological classes in this feature space and only achieves 14.5% accuracy.



(a) LDA using CP features



(b) LDA using pixel data

Figure 5.5.: Confusion matrix for phenotype classification using LDA

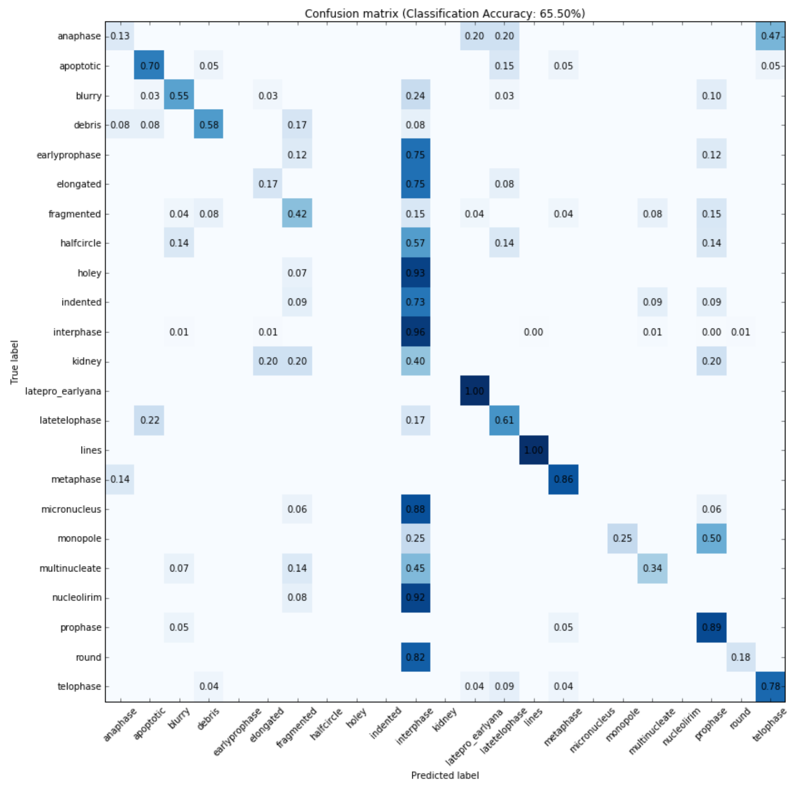
Number of trees	Accuracy (CP features)	Accuracy (pixel only)
10	0.616	0.472
100	0.653	0.545
1000	0.655	0.560

Table 5.2.: This table shows the performance for random forest classification of 23 morphological classes using CP features and raw pixel input.

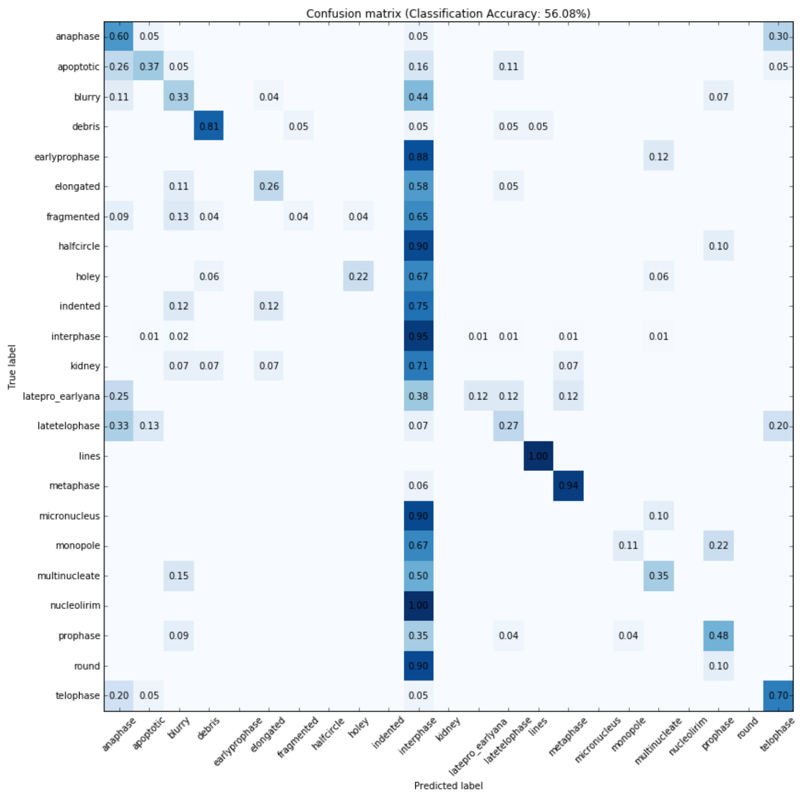
5.3.2. Random Forest

Random forests are an ensemble learning method for classification, that construct a multitude of decision trees at training time. The output of a random forest is the majority vote of the individual trees (the mode of the classes). For our experiments, we used random forests with 10, 100 and 1000 decision trees.

Table 5.2 shows that, using CP features, random forests perform around the same as LDA (between 61% and 65%). Furthermore, random forests require the selection of several hypterparameters, such as number of trees and maximal depth. In our experiments, nodes are expanded until all leaves are pure. We used weight balancing in scikit-learn which automatically adjust weights inversely proportional to class frequencies in the input data. Using raw pixel data, random forests perform much better than LDA, achieving 47% to 56% accuracy. However, as Figure 5.6 shows, the random forest is overall not suited for phenotype classification due to the fact, that it is not able to predict rare classes (such as micronucleus). As we will see later, especially phenotypes with few training examples, thus making them biological rare subpopulations, seem to have the largest influence on the mechanism of action of a biological sample.



(a) Random Forest using CP features



(b) Random Forest using pixel data

Figure 5.6.: Confusion matrix for phenotype classification using a random forest (n =1000)

Model	Accuracy (CP features)
Feed-forward Net (2 layers)	0.582
Feed-forward Net (4 layers)	0.621
Feed-forward Net (8 layers)	0.605

Table 5.3.: Deep learning models, trained over 100 epochs using momentum and a learning rate of 0.1. Batch normalization is used in all models.

5.4. Deep Learning for Subpopulation classification

In this section, we explore deep learning models for subpopulation classification. All experiments are run using neon 1.6.0 and TensorFlow 0.9 each model is trained using momentum, batch normalization and weight decay. We start each model with a learning rate of 0.1 and decay it to 0.01 and 0.001 using a staircase method. Our method is evaluated using a 5-fold crossvalidation.

Table 5.3 shows the accuracy of deep neural networks using CellProfiler features. In general, they perform worse than the baselines set by CellProfiler Analyst's random forest and LDA. A feed-forward network with four layers achieves the best results (62.1% accuracy). The performance is not surprising due to the fact, that deep neural networks need enough training data to learn a sufficient representation (as described in Chapter 2). CellProfiler features are represented as 94-dimensional vector and thus provides not enough information for a neural network.

The question arises whether we can learn better features than CellProfiler's expert-engineered feature set. Instead of using CellProfiler features, we directly use single cell image tiles (96×96px) as input. This allows us to apply data augmentation (see Section 2.2.2) and provides us with a larger feature space (9216 instead of 94 dimensions).

Table 5.4 shows the results of four different models: VGG-16, ResNet-32, ResNet-56 and ResNet-110. As one can see, a residual network with 32 layers, trained on pixels, achieves the highest accuracy 72.4%. It achieves a relative improvement of 10% compared to the best performing traditional classifiers using CP features and a 22% improvement compared to traditional classifiers using only raw pixels as input. This is a surprising result due to the fact, that SUBPOP is relatively small (2526 cells and 23 classes) compared

Model	Data Augmentation	Accuracy (pixel only)
VGG-16	No	0.612
VGG-16	Yes	0.670
ResNet-32	No	0.648
ResNet-32	**Yes**	**0.724**
ResNet-56	Yes	0.705
ResNet-110	Yes	0.689

Table 5.4.: Deep learning models, trained over 100 epochs using momentum and a learning rate of 0.1. Batch normalization is used in all models.

to the usual training sets for deep neural networks (i.e. ImageNet with 1 million images and 1000 classes). We discovered that data augmentation helps the network to cope with the small dataset by artificially generating more training examples, yielding a significant accuracy improvement of 6% for VGG-16 and 8% for ResNet-32.

As we can see in Figure 5.7, a residual networks not only perform well in terms of classification accuracy, but also are able to detect classes with only few training examples. For example, the morphological class holey only has around 100 training examples and traditional methods only achieve low accuracy using CP features (LDA correctly classifies 14%; random forest misclassified all training examples). A ResNet, however, was able to classify the class correctly in 61% of all cases. This suggest that traditional CellProfiler features are not able to sufficiently capture this class, while a neural network was able to learn better features from pixel data.

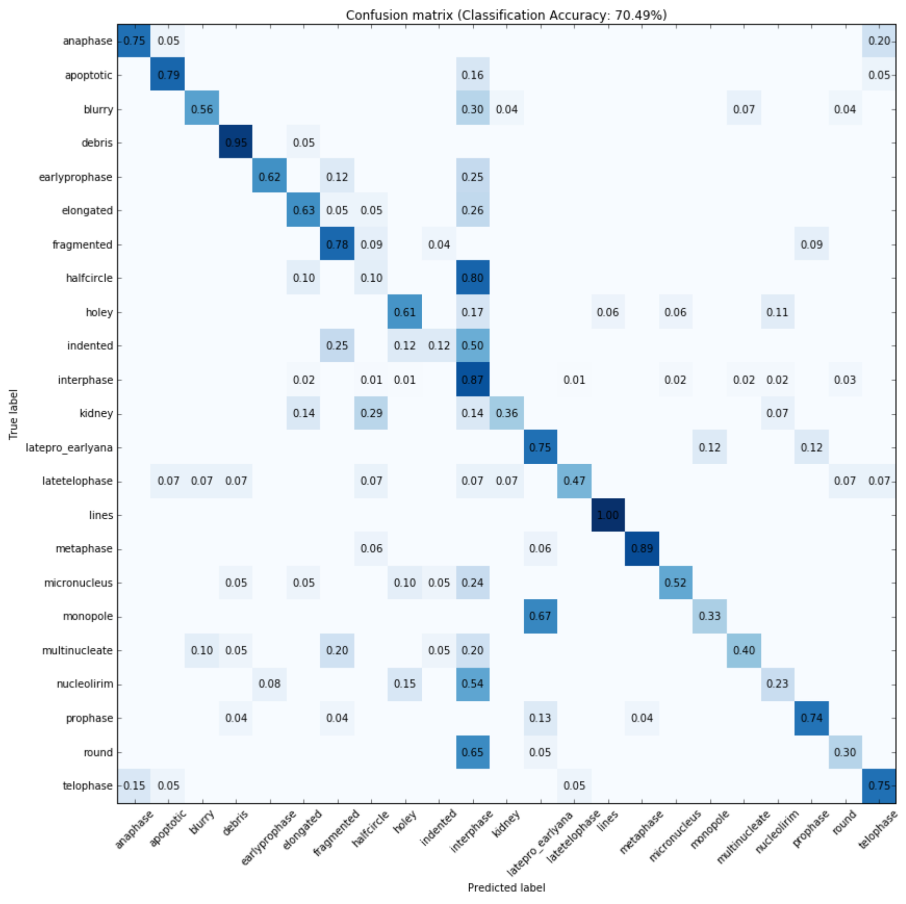


Figure 5.7.: **Confusion matrix for phenotype classification using a ResNet-56** - In contrast to LDA and random forest, a ResNet-56 is able to classify rare phenotypes such as holey

5.5. Neural Attention for Automatic Cell detection

One disadvantage of our supervised profiling workflow is the need of cropped cell image tiles in order to train our deep neural network. Therefore, we need to preprocess and extract the location of single cells in an image. Thus, the question arises, if we can integrate this preprocessing step into our neural network (and detect single cells from whole image data alone) without requiring a third-party software (e.g CellProfiler). In this section, we explore the use of spatial attention models [59] to automatically detect cells with a specific phenotype. Using a spatial transformer network, we were able to detect and crop single cells within a 96×96 px frame. Furthermore, we open-sourced our implementation as part of TensorFlow [60] in order to bootstrap further research in this direction. Our model can be downloaded at `https://github.com/tensorflow/models/tree/master/transformer`.

5.5.1. Spatial Transformer Network

A spatial transformer network or STN was first introduced by Jaderberg *et al.* [59] and allows the spatial manipulation of data within a deep network. STNs are self-contained modules which can be dropped into a CNN architecture at any point.

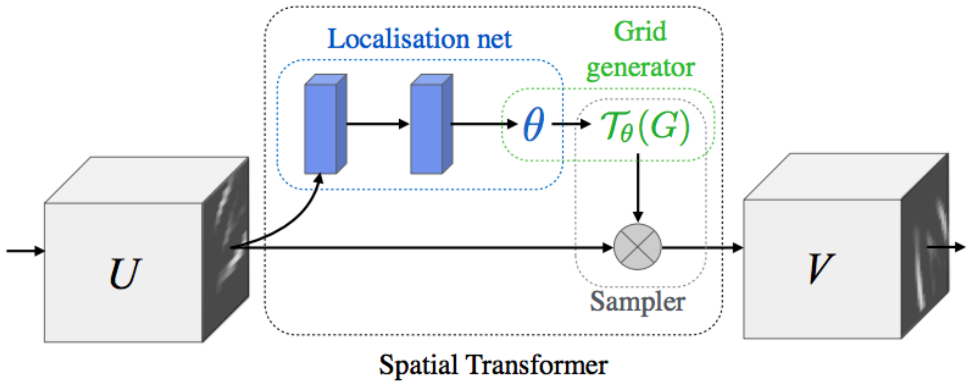
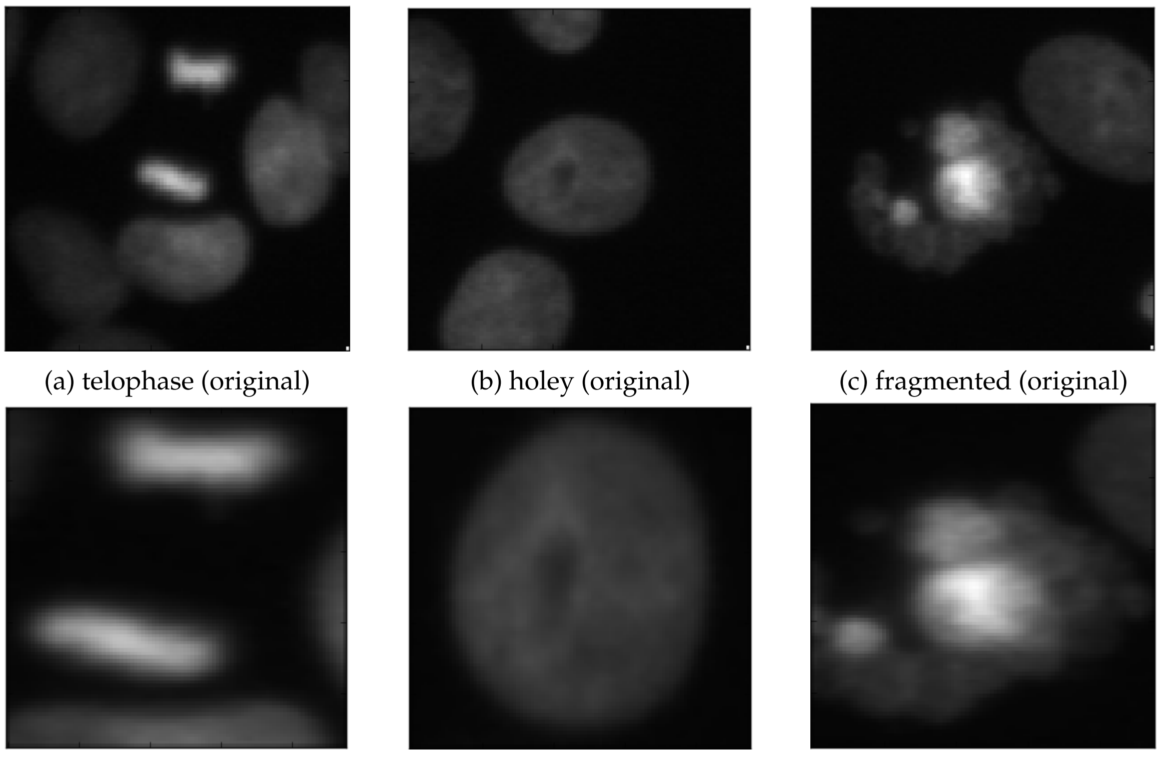


Figure 5.8.: **Spatial Transformer Network** - A spatial transformer network consists of three components: localisation network, grid generator and sampler. U and V are input and output feature maps within a CNN. Reprinted from [59].

It has three main components (see Figure 5.8): A localisation network, grid generator and sampler. The localisation network takes as input the feature map of a convolutional layer and learns parameters θ of a transformation we want to apply. The grid generator $\mathcal{G}$ then generates a grid of coordinates in the input image corresponding to each pixel from the output image. Equation 5.2 describes a pointwise affine transformation

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(\mathcal{G}_i) = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} \tag{5.2}$$


(a) telophase (original) (b) holey (original) (c) fragmented (original)

(d) telophase (attention) (e) holey (attention) (f) fragmented (attention)

Figure 5.9.: **Neural attention for phenotype detection** - we cropped 96 × 96 px tiles from the image around a phenotype (upper row). A spatial transformer network was then able to 'attend' on the corresponding phenotype in the image (lower row).

where $(x_i^t, y_i^t)^T$ are the target coordinates of the grid and $(x_i^s, y_i^s)^T$ are the source coordinates in the input feature map and θ are the parameters of the affine transformation matrix. Finally, in order to perform a differential spatial transformation, a sampler generates the output image using the grid given by the grid generator. For our experiments, we will use a bilinear sampling kernel.

5.5.2. Experiments and Results

The goal of these experiments is to show that it is possible to detect single cells (or regions of interest) on-the-fly while training a neural network for phenotype classification. First, we plugged a STN on top of the first CNN layer of our ResNet-32 (see Section 5.4) for visualization purpose and trained the network using 96 × 96 px tiles (with multiple cells) with one class label. Figure 5.9 shows the original input image (with class label) and the spatial transformed output. As one can see, the STN is capable of identifying the correct cell for the corresponding classes. For example, it correctly differentiates a holey cell (see Subfigures 5.9b and 5.9e) from three other cells and is able to 'focus' on it. This indicates that the neural network considers the morphology of specific cells within a population as a strong signal for the class label of an image. One could imagine to extend this experiment and 'focus' on regions of interest instead of single cells for the whole image. However, during our experiments, we observed that STNs are not able to learn to detect single cells on larger tile sizes (larger than 150 $\times$ 150 px). Furthermore each spatial transformer network can only attend on one single cell, thus making the simultanous detection of hundreds of cells intractable. Thus, we conclude that, for small tile sizes, one can train a cell detector using weakly supervised labels (class label of cell of interest). In practice, however, in order to accurately detect single cells for a large field of view with multiple cells, we still require third-party software or a model (e.g Faster R-CNN [61]), specifically trained for this purpose.

5.6. Subpopulation Profiles for MOA classification

This section discusses how we can generate profiles using subpopulation information. After training a ResNet-32 on SUBPOP, we use our deep model to predict phenotypes for each of the 500000 unlabeled cells in BBBC021. The subpopulation information allows us to represent each treatment as a 23-dimensional profile:

$$s = (a_1 s_1, a_2 s_2, \dots, a_{23} s_{23})^T \tag{5.3}$$

each element s_i of which contains the number of nuclei classified into the corresponding phenotype i and a_i models the "biological influence" of this phenotype, which we choose either manually (no additional data required) or infer it using a factor model (additional labeled data required). In this section, we will choose the parameters based on expert

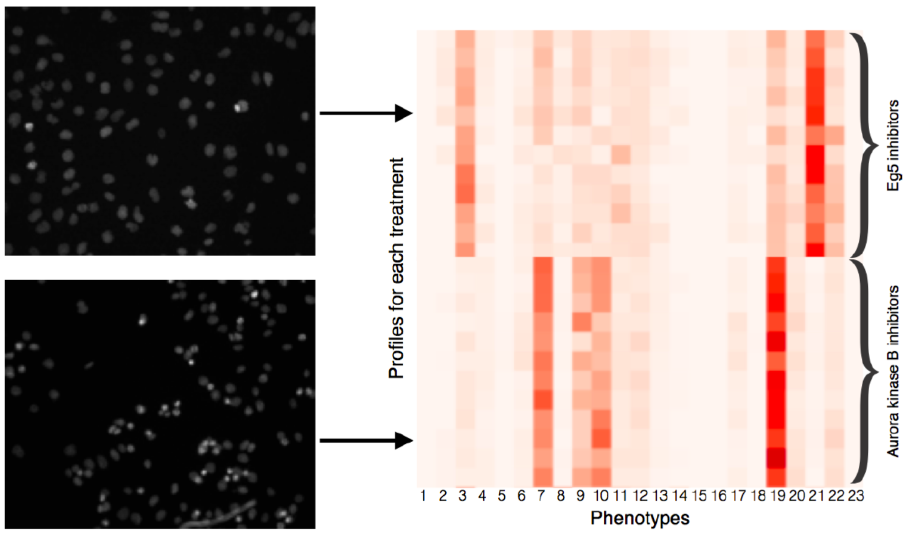
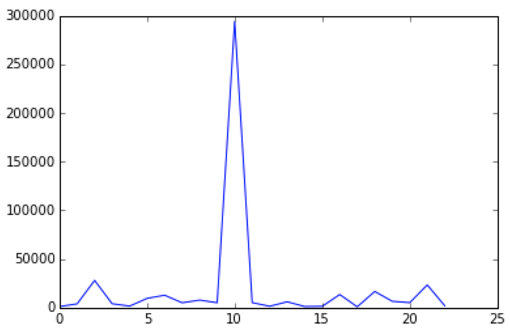
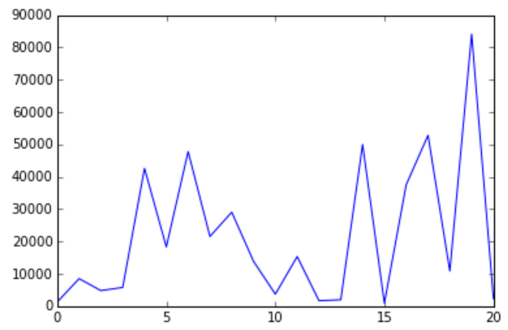


Figure 5.10.: **Clustering using subpopulation profiles** - For each treatment, we extract the subpopulation profile s by extracting the cell counts of each morphological class via a trained ResNet-32. Using a distance metric d, we can then cluster treatment profiles into MOAs. Here, aach row corresponds to an instance of a treatment experiment. As one can see, subpopulation profiles are low dimensional (only 23 dim) and interpretable.



(a) using all classes



(b) without interphase, blurry and lines

Figure 5.11.: **Cell count distribution** - Cell count distribution for all morphological classes, as predicted by ResNet-32 on BBBC021. Due to its large cell count, interphase diminishes the relative signal of rarer classes (Note: because we start at zero, interphase is labeled 10 in this example)

knowledge: For example, we hypothesize that our two artifactual phenotypes: blurry (3) and lines (15) shouldn't have any influence on which mechanism of action a treatment belongs to. After computing the subpopulation treatment profile, we can use a distance metric between the profiles as a measure of similarity to cluster similar profiles together and predict MOA using 1-nearest-neighbor (as described in Section 3.1). In Figure 5.10, one can see, that these profiles are highly interpretable. For example, we can visually observe that treatments which are known to be Aurora Kinase B inhibitors have a high cell count of monopole (18) phenotypes in their population among all treatments.

Our naive assumption $a_1, \ldots, a_{23} = 1$, giving each phenotype the same weight on the subpopulation profile results in lower accuracy than mean profiles. Our best performing model achieves 60% accuracy on MOA classification using cityblock distance (see Table 5.5). As seen in Figure 5.11a, more than half of all cells in BBBC021 are classified as in-

Model	Metric	MOA accuracy
Means	Cosine	0.66
Means + FA analysis	Cosine	0.68
ResNet-32	Cosine	0.58
ResNet-32 trained without interphase, blurry and lines	Cosine	0.75
ResNet-32 with weighting	**Cosine**	**0.83**
ResNet-32	Euclidean	0.57
ResNet-32 trained without interphase, blurry and lines	Euclidean	0.69
ResNet-32 with weighting	Euclidean	0.78
ResNet-32	Cityblock	0.60
ResNet-32 trained without interphase, blurry and lines	Cityblock	0.70
ResNet-32 with weighting	Cityblock	0.77

Table 5.5.: Comparison of MOA classification results for ResNet-32 and mean profiles

terphase (11). Although this might resemble the true distribution, it diminishes the signal of rare phenotypes (such as holey or micronucleus). Thus, we speculate that it might be these rare phenotypes which have the most biological influence (hence determining which mechanism of action a compound belongs to).

To test this hypothesis, we set the weights of interphase and the artefactual classes (lines and blurry) to zero ($a_3 = 0, a_{11} = 0, a_{15} = 0$). Subpopulation profiles with these new weights (which we call weightings) turn out to outperform both, means and means + factor analysis, by a large margin. Our best model, a ResNet-32 model with weightings using cosine distance outperforms population averaging and factor analysis by 15% (relative improvement of 18%). Using weight adjustment, other metrics yield similar improvements, all outperforming population averaging: A deep model using euclidean distance achieves 78% while a deep network using cityblock distance results in 77% accuracy. We assume that cosine distance is best suited for subpopulation profiling, because it is invariant to scaling, thus making it more robust towards fluctuations in cell counts. The question arises if, instead of setting weights a_3, a_{11} and a_{15} to zero, we could just train our deep neural network without interphase, blurry and lines. Thus, we removed all cells belonging to interphase, blurry and lines from our training set. One can see in Figure 5.5 that ResNets-32 trained without interphase, blurry and lines perform better than ResNet-32 subpopulation profiles, trained on the full data set without weightings. However, it performs worse than with weightings. We believe that this is due to missing training data, resulting in worse classification accuracy on rare phenotypes. Without interphase training cells, ResNet-32's phenotype classification accuracy dropped from 72.4 % (see Figure 5.4) to 69.0 %.

6. Conclusion

In this chapter, we will summarize our findings, discuss its potential impact and explore future research directions.

6.1. Summary

Subpopulations are a fundamental property of cellular systems and thus, recent computational development for image-based profiling aim to preserve single-cell data and take into account the heterogeneity of cultured cell populations. The goal of this thesis was to explore how the use of supervised deep learning models enables us to capture cellular heterogeneity in profiles.

To accomplish this goal, we had to overcome several challenges:

1. Can we design a system to leverage experts' knowledge and curate a small subpopulation training set?
2. What are good features to train a model to classify phenotypes?
3. Does subpopulation profiles yield additional information to outperform state-of-the-art population averaging?

To resolve the first question, we developed CellProfiler Analyst 2.0, a user-friendly tool for expert-guided machine learning in bioimaging, that allowed us to analyze and sort more than 2500 cells into 23 morphological classes, creating classification baselines on the fly. CellProfiler Analyst is designed for biologists with no coding experience and allows to extract cells into image tiles. Additional features for biologists further allow them to quickly analyze post-process their image collections. CellProfiler Analyst 2.0 has been published in [7] and is routinely used by numerous researchers around the world.

After extracting the subpopulation training set, we then showed that CellProfiler (CP) features were not able to capture enough information about rare phenotypes. Traditional machine learning models, such as LDA and random forests, were not able to classify rare morphological classes when trained with CP features. Instead of using expert-engineered features, we thus propose to directly learn features from a deep neural network trained on raw image pixels. Surprisingly, although we were working with a relatively small and custom dataset, a very deep residual network with 32 layers (ResNet-32) was able to achieve the highest classification accuracy among all tested models. We believe that there are two reasons for this: First, because we are working with raw pixels instead of expert-engineered features, data augmentation methods allowed us to artificially increase our dataset by flipping and translating the image without changing the class (which is not possible in CP feature space). Second, it has been shown in a recent study [62] that residual networks act as an ensemble of smaller networks, allowing it to be trained with less data than a single large network. Our results showed that current deep learning techniques can come up with better features for single cell image tiles without requiring any segmentation and user-input from a third-party software. Further, it has been found that deep neural networks are applicable on relatively small single-cell datasets. These promising findings have a wide-ranging impact across drug discovery and basic science as they pave the way towards better and fully automated bioimage analysis software for high-throughput experiments.

Finally, the fundamental question we wanted to answer is whether subpopulation profiles, thus explicitly modeling cellular heterogeneity, are in general better than averaging profiles. Or simply put: do subpopulations matter for image-based profiling? We were able to show that subpopulation-based analysis can improve the accuracy of signature identification in a previously published dataset. However, it seems that the examined ground-truth dataset is too small and thus the proposed classification accuracy might not be a reliable metric for comparing profiling methods. Therefore, further research is needed and profiling methods in general have to be validated across a wider range of profiling experiments.

The success of subpopulation profiles for image-based profiling will depend on how useful it is for biologists and how robust the method is in the presence of artefactual and biological noise. Hence, finding a useful similarity measurement among subpopulation profiles is an open research question. However, our proposed method has several promising advantages:

- Though tedious and non-exhaustive compared to fully automated methods, our method leverages biologists' knowledge and thus is able to extract and identify subtle phenotypes.
- In contrast to unsupervised or weakly-supervised learning methods, our models, given similar experimental settings, don't need to be retrained for new datasets. This makes its use computationally efficient and scalable.
- Subpopulation profiles are interpretable and, in contrast to population averages of CP features, have low dimensionality. Both properties are valuable for basic research and help to not only discover but also explain biological findings.

6.2. Future work

The field of deep learning for biological subpopulation profiling offers multiple possible directions for future work and can be broadly categorized into two themes: Integration of deep learning models into existing image processing software and the development of fully unsupervised subpopulation methods.

CellProfiler Analyst currently supports only traditional machine learning algorithms. In order to democratize deep neural networks for image-based profiling, we plan to integrate popular deep learning libraries such as TensorFlow and Torch into CellProfiler Analyst.

A bottleneck in supervised workflows to the need to train on labeled datasets. Furthermore, as mentioned above, the small available ground-truth data complicates the validation of novel profiling methods. Although we introduced software to ease the labeling of these datasets, another approach would be to develop automated clustering technique to extract subpopulations with subtle phenotypes. Furthermore, it was attempted to explore the use of attention models, such as spatial transformer networks [59], for unsupervised single-cell phenotype detection during this thesis. Detection of multiple cells on a larger frame can be accomplished using recurrent spatial transformer architectures [63] but time constraints precluded further research.

Moreover, an interesting approach would be to train a neural network end-to-end directly from image to mechanism-of-action, without requiring the use of any subpopulations. However due to relatively small ground-truth datasets available for MOA classification, training a model would possibly fail. One could try to circumvent this by training neural models on image labels, such as treatment information, instead of MOA labels. However, we think a disadvantage, by taking the human out of the loop, is that we miss out a chance on interpretable profiles and leveraging expert-guided knowledge.

Nevertheless, deep learning presents a promising approach to image-based chemical-genetic profiling. Being able to model cellular heterogeneity and preserving single-cell information will not only have a wide-ranging impact on the performance of image-based profiling but will also expand our understanding of biological phenomena.

Appendix

A. Detailed results

A.1. Subpopulation class descriptions

Phenotype	Description
anaphase	chromatin in relatively straight line, edges of chromatin are irregular, amount of DNA is clearly 2N
apoptotic	small bright cells that look unhealthy but amount of DNA is roughly 2N
blurry	nuclei of any shape or phenotype that are quite blurry
debris	small bits of DNA where the amount of DNA looks less than 2N
earlyprophase	still has an interphase shape overall, but the chromatin is crinklier
elongated	relatively normal-looking interphase nuclei that are elongated
fragmented	multiple small or medium-sized DNA regions that are typically blebby
halfcircle	relatively normal-looking interphase nuclei that are shaped like a half circle
holey	relatively normal-looking interphase nuclei where dark holes are clearly visible for the nucleoli
indented	relatively normal-looking interphase nuclei but with one or two indentations but without much DNA staining around the holes
interphase	normal-looking interphase nuclei of any size (except tiny; see late telophase) that do not have another phenotype listed
kidney	relatively normal-looking interphase nuclei that are kidney-shaped
latepro-earlyana	either just before or just after metaphase; like prophase but segmentation is off-center
latetelophase	a bit less of a straight line versus telophase; essentially tiny interphase cells
lines	artifactual horizontal lines
metaphase	chromatin in straight line, amount of DNA is clearly 4N
micronucleus	relatively normal-looking interphase nuclei with one or two tiny micronuclei just outside the nucleus

monopole	like prophase but with a clear divot in the middle, segmentation is centered
multinucleate	two or three otherwise relatively normal-looking interphase nuclei
nucleolirim	relatively normal-looking interphase nuclei where dark holes are relatively visible for the nucleoli and there is noticeable DNA staining around the holes
prophase	roundish with more tube-like chromatin, segmentation is centered
round	relatively normal-looking interphase nuclei that are round
telophase	chromatin in relatively straight line, edges of chromatin are smooth, amount of DNA is clearly 2N

Table A.1.: Phenotype descriptions of all 23 SUBPOP classes

A.2. ResNet-32 detailed results for MOA classification and TensorFlow implementation graphs

```
True mechanistic class     Predicted class                      Acc.
------------------------   ----------------------------------   -----
Actin disruptors            -  -  -  -  -  -  -  -  1  -  -  4    0 %
Aurora kinase inhibitors    - 10  -  -  2  -  -  -  -  -  -  -   83 %
Cholesterol-lowering        -  -  4  -  -  -  -  -  -  -  2  -   67 %
DNA damage                  -  -  -  9  -  -  -  -  -  -  -  -  100 %
DNA replication             -  -  -  -  6  -  -  -  -  -  2  -   75 %
Eg5 inhibitors              -  -  -  -  -  6  -  -  5  -  1  -   50 %
Epithelial                  -  -  1  -  -  -  4  -  -  -  2  1   50 %
Kinase inhibitors           -  -  -  -  -  -  -  2  3  -  -  -   40 %
Microtubule destabilizers   -  -  -  -  -  -  -  1 11  2  -  -   79 %
Microtubule stabilizers     -  -  -  -  -  -  -  -  3  6  -  -   67 %
Protein degradation         -  -  1  -  3  -  1  -  -  -  1  1   14 %
Protein synthesis           1  -  -  -  -  -  3  -  -  -  1  3   38 %
------------------------   ----------------------------------   -----
                                    Overall accuracy: 62 / 103 = 60 %
```

(a) Cityblock distance

```
True mechanistic class     Predicted class                      Acc.
------------------------   ----------------------------------   -----
Actin disruptors            3  -  -  -  -  1  -  -  -  -  -  1   60 %
Aurora kinase inhibitors    -  9  -  -  2  -  -  -  -  -  -  1   75 %
Cholesterol-lowering        -  -  6  -  -  -  -  -  -  -  -  -  100 %
DNA damage                  -  -  -  6  3  -  -  -  -  -  -  -   67 %
DNA replication             -  -  -  3  3  -  -  2  -  -  -  -   38 %
Eg5 inhibitors              -  -  -  -  -  6  -  -  5  -  1  -   50 %
Epithelial                  -  -  1  -  -  -  4  1  -  -  2  -   50 %
Kinase inhibitors           -  -  -  -  -  -  -  3  2  -  -  -   60 %
Microtubule destabilizers   -  -  -  -  -  2  -  1  8  3  -  -   57 %
Microtubule stabilizers     -  1  -  -  -  -  -  -  2  6  -  -   67 %
Protein degradation         -  -  -  -  1  -  1  1  1  -  3  -   43 %
Protein synthesis           3  -  -  -  -  -  2  -  -  -  -  3   38 %
------------------------   ----------------------------------   -----
                                    Overall accuracy: 60 / 103 = 58 %
```

(b) Cosine distance

```
True mechanistic class     Predicted class                      Acc.
------------------------   ----------------------------------   -----
Actin disruptors            1  1  -  -  -  1  -  -  -  -  -  2   20 %
Aurora kinase inhibitors    -  9  -  -  1  -  1  -  -  -  -  1   75 %
Cholesterol-lowering        -  -  4  1  -  -  -  -  -  -  1  -   67 %
DNA damage                  -  -  -  9  -  -  -  -  -  -  -  -  100 %
DNA replication             -  -  1  -  6  -  -  -  -  -  1  -   75 %
Eg5 inhibitors              -  -  -  -  -  6  -  -  6  -  -  -   50 %
Epithelial                  -  -  1  1  -  -  3  -  -  -  2  1   38 %
Kinase inhibitors           -  -  -  -  -  -  -  3  2  -  -  -   60 %
Microtubule destabilizers   -  -  -  -  -  2  -  1  9  2  -  -   64 %
Microtubule stabilizers     -  -  -  -  -  -  -  -  3  6  -  -   67 %
Protein degradation         -  -  3  -  2  -  2  -  -  -  -  -    0 %
Protein synthesis           1  -  1  -  -  -  3  -  -  -  -  3   38 %
------------------------   ----------------------------------   -----
                                    Overall accuracy: 59 / 103 = 57 %
```

(c) Euclidean distance

Figure A.1.: Confusion matrix for MOA classification using ResNet-32

```
True mechanistic class     Predicted class                        Acc.
------------------------   ----------------------------------    -----
Actin disruptors             1  1  -  -  -  1  -  -  -  -  -  2    20 %
Aurora kinase inhibitors     - 11  -  -  -  -  -  -  -  1  -  -    92 %
Cholesterol-lowering         -  -  5  -  -  -  1  -  -  -  -  -    83 %
DNA damage                   -  -  -  8  -  -  -  1  -  -  -  -    89 %
DNA replication              -  -  -  -  4  -  -  -  -  -  4  -    50 %
Eg5 inhibitors               1  -  -  -  - 10  -  -  1  -  -  -    83 %
Epithelial                   2  -  1  -  -  -  4  -  -  -  -  1    50 %
Kinase inhibitors            -  -  -  2  -  -  -  3  -  -  -  -    60 %
Microtubule destabilizers    -  -  -  -  -  -  -  1 10  3  -  -    71 %
Microtubule stabilizers      -  -  -  -  -  -  -  -  3  6  -  -    67 %
Protein degradation          -  -  1  -  4  -  -  -  -  -  2  -    29 %
Protein synthesis            -  -  -  -  -  -  -  -  -  -  -  8   100 %
------------------------   ----------------------------------    -----
                                 Overall accuracy: 72 / 103 = 70 %
```

(a) Cityblock distance

```
True mechanistic class     Predicted class                        Acc.
------------------------   ----------------------------------    -----
Actin disruptors             1  1  -  -  -  -  -  -  1  -  -  2    20 %
Aurora kinase inhibitors     - 12  -  -  -  -  -  -  -  -  -  -   100 %
Cholesterol-lowering         -  -  4  -  -  -  2  -  -  -  -  -    67 %
DNA damage                   -  -  -  9  -  -  -  -  -  -  -  -   100 %
DNA replication              -  -  -  -  6  -  -  -  -  -  2  -    75 %
Eg5 inhibitors               -  -  -  -  -  8  -  -  4  -  -  -    67 %
Epithelial                   -  -  1  -  -  -  5  -  -  -  -  2    62 %
Kinase inhibitors            -  -  -  1  -  -  -  4  -  -  -  -    80 %
Microtubule destabilizers    -  -  -  -  -  -  -  1 11  2  -  -    79 %
Microtubule stabilizers      -  -  -  -  -  -  -  -  3  6  -  -    67 %
Protein degradation          -  -  1  -  3  -  -  -  -  -  3  -    43 %
Protein synthesis            -  -  -  -  -  -  -  -  -  -  -  8   100 %
------------------------   ----------------------------------    -----
                                 Overall accuracy: 77 / 103 = 75 %
```

(b) Cosine distance

```
True mechanistic class     Predicted class                        Acc.
------------------------   ----------------------------------    -----
Actin disruptors             2  -  -  -  -  1  -  -  -  -  -  2    40 %
Aurora kinase inhibitors     - 12  -  -  -  -  -  -  -  -  -  -   100 %
Cholesterol-lowering         -  -  5  -  -  -  1  -  -  -  -  -    83 %
DNA damage                   -  -  -  4  4  -  -  1  -  -  -  -    44 %
DNA replication              -  -  -  7  -  -  -  1  -  -  -  -     0 %
Eg5 inhibitors               -  -  -  -  - 12  -  -  -  -  -  -   100 %
Epithelial                   -  -  1  -  -  -  7  -  -  -  -  -    88 %
Kinase inhibitors            -  -  -  -  2  -  -  3  -  -  -  -    60 %
Microtubule destabilizers    -  -  -  -  1  -  -  - 10  3  -  -    71 %
Microtubule stabilizers      -  -  -  -  -  -  -  -  4  5  -  -    56 %
Protein degradation          -  -  1  -  1  -  -  1  -  -  3  1    43 %
Protein synthesis            -  -  -  -  -  -  -  -  -  -  -  8   100 %
------------------------   ----------------------------------    -----
                                 Overall accuracy: 71 / 103 = 69 %
```

(c) Euclidean distance

Figure A.2.: Confusion matrix for MOA classification using ResNet-32 trained without interphase, blurry and lines cells

```
True mechanistic class     Predicted class                      Acc.
------------------------   ---------------------------------   -----
Actin disruptors             1  -  -  -  -  -  -  -  1  -  -  3   20 %
Aurora kinase inhibitors     - 11  -  -  1  -  -  -  -  -  -  -   92 %
Cholesterol-lowering         -  -  6  -  -  -  -  -  -  -  -  -  100 %
DNA damage                   -  -  -  9  -  -  -  -  -  -  -  -  100 %
DNA replication              -  -  -  -  6  -  -  -  -  -  2  -   75 %
Eg5 inhibitors               -  -  -  -  -  7  -  -  5  -  -  -   58 %
Epithelial                   -  -  1  -  -  -  5  -  -  -  -  2   62 %
Kinase inhibitors            -  -  -  -  -  -  -  5  -  -  -  -  100 %
Microtubule destabilizers    -  -  -  1  -  -  -  - 11  2  -  -   79 %
Microtubule stabilizers      -  -  -  -  -  -  -  -  3  6  -  -   67 %
Protein degradation          -  -  1  -  2  -  -  -  -  -  4  -   57 %
Protein synthesis            -  -  -  -  -  -  -  -  -  -  -  8  100 %
------------------------   ---------------------------------   -----
                                    Overall accuracy: 79 / 103 = 77 %
```

(a) Cityblock distance

```
True mechanistic class     Predicted class                      Acc.
------------------------   ---------------------------------   -----
Actin disruptors             4  -  -  -  -  -  -  -  -  -  1  -   80 %
Aurora kinase inhibitors     - 12  -  -  -  -  -  -  -  -  -  -  100 %
Cholesterol-lowering         -  -  6  -  -  -  -  -  -  -  -  -  100 %
DNA damage                   -  -  -  7  2  -  -  -  -  -  -  -   78 %
DNA replication              -  -  -  4  3  -  -  -  -  -  1  -   38 %
Eg5 inhibitors               -  -  -  -  - 11  -  -  1  -  -  -   92 %
Epithelial                   -  -  -  -  -  -  7  1  -  -  -  -   88 %
Kinase inhibitors            -  -  -  1  -  -  -  4  -  -  -  -   80 %
Microtubule destabilizers    -  -  -  -  -  -  -  1 10  3  -  -   71 %
Microtubule stabilizers      -  1  -  -  -  -  -  -  -  8  -  -   89 %
Protein degradation          -  -  -  -  -  -  -  1  -  -  5  1   71 %
Protein synthesis            -  -  -  -  -  -  -  -  -  -  -  8  100 %
------------------------   ---------------------------------   -----
                                    Overall accuracy: 85 / 103 = 83 %
```

(b) Cosine distance

```
True mechanistic class     Predicted class                      Acc.
------------------------   ---------------------------------   -----
Actin disruptors             3  -  -  -  -  -  -  -  1  -  -  1   60 %
Aurora kinase inhibitors     - 11  -  -  1  -  -  -  -  -  -  -   92 %
Cholesterol-lowering         -  -  6  -  -  -  -  -  -  -  -  -  100 %
DNA damage                   -  -  -  9  -  -  -  -  -  -  -  -  100 %
DNA replication              -  -  -  -  6  -  -  -  -  -  2  -   75 %
Eg5 inhibitors               -  -  -  -  -  9  -  -  3  -  -  -   75 %
Epithelial                   2  -  1  -  -  -  3  -  -  -  -  2   38 %
Kinase inhibitors            -  -  -  -  -  -  -  5  -  -  -  -  100 %
Microtubule destabilizers    -  -  -  1  -  1  -  -  9  3  -  -   64 %
Microtubule stabilizers      -  -  -  -  -  -  -  -  1  8  -  -   89 %
Protein degradation          -  -  2  -  2  -  -  -  -  -  3  -   43 %
Protein synthesis            -  -  -  -  -  -  -  -  -  -  -  8  100 %
------------------------   ---------------------------------   -----
                                    Overall accuracy: 80 / 103 = 78 %
```

(c) Euclidean distance

Figure A.3.: Confusion matrix for MOA classification using ResNet-32 and a class weight adjusted metric (weighting)

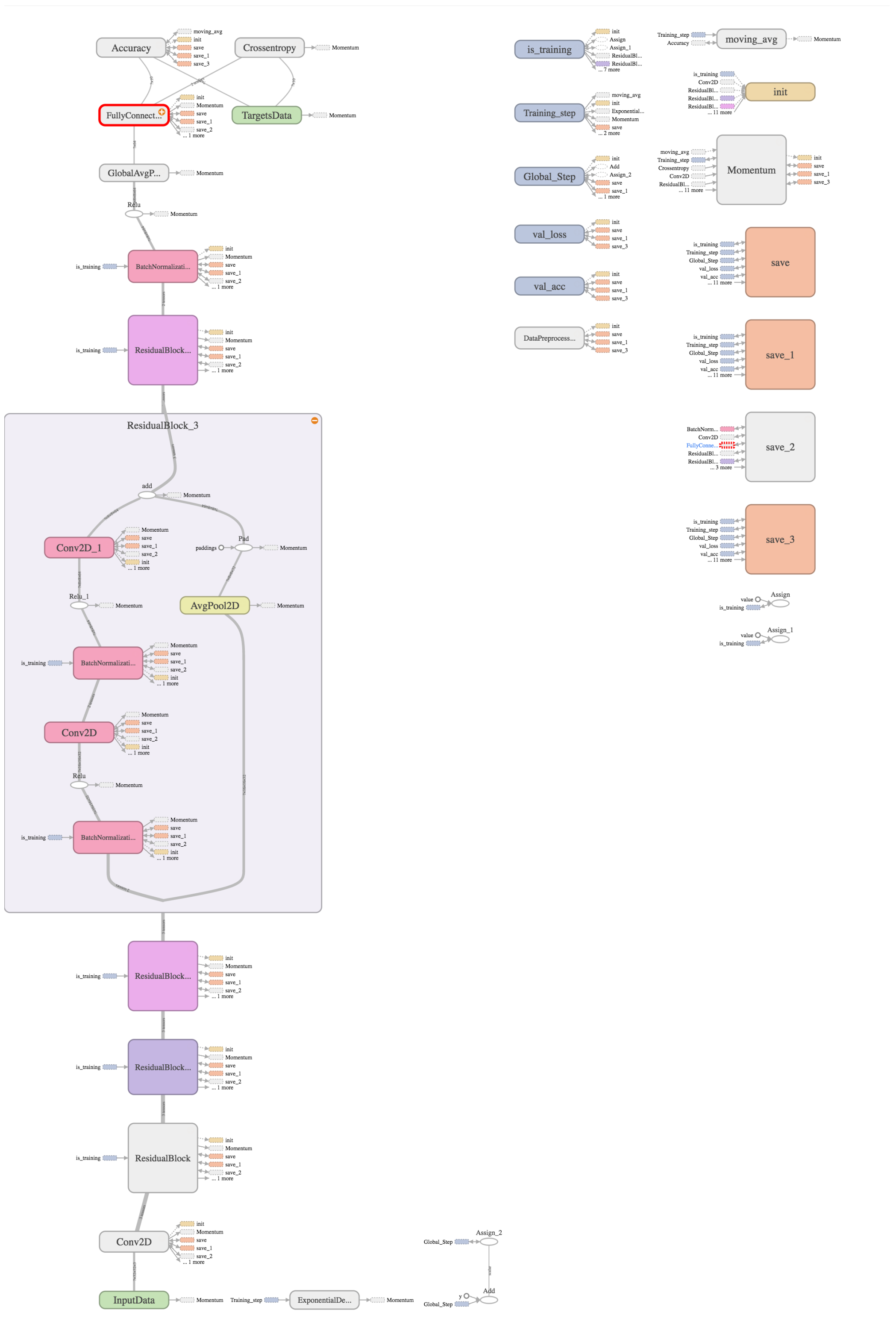


Figure A.4.: ResNet-32 implementation graph in TensorBoard

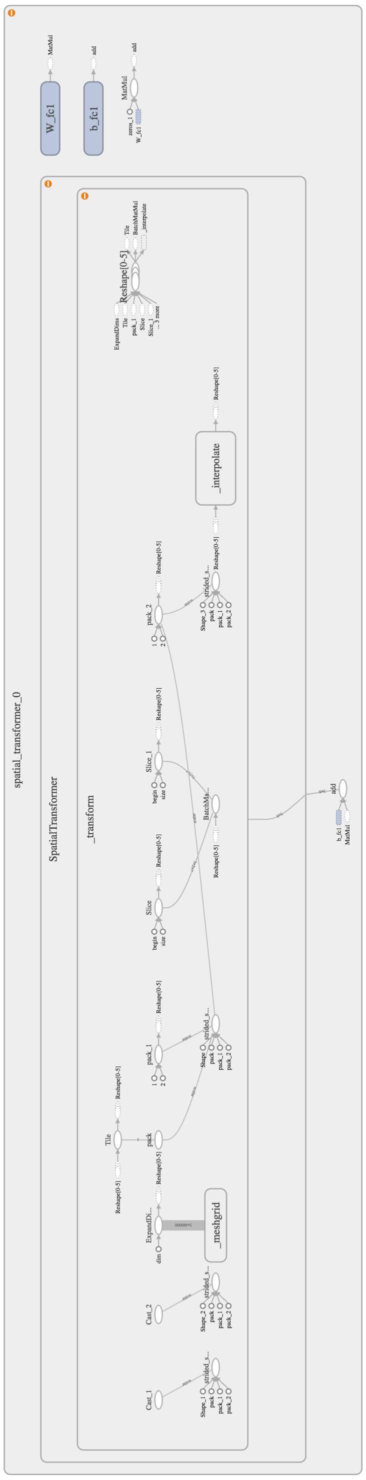


Figure A.5.: Spatial Transformer Network implementation graph in TensorBoard

A.3. Subpopulation cells and profiles

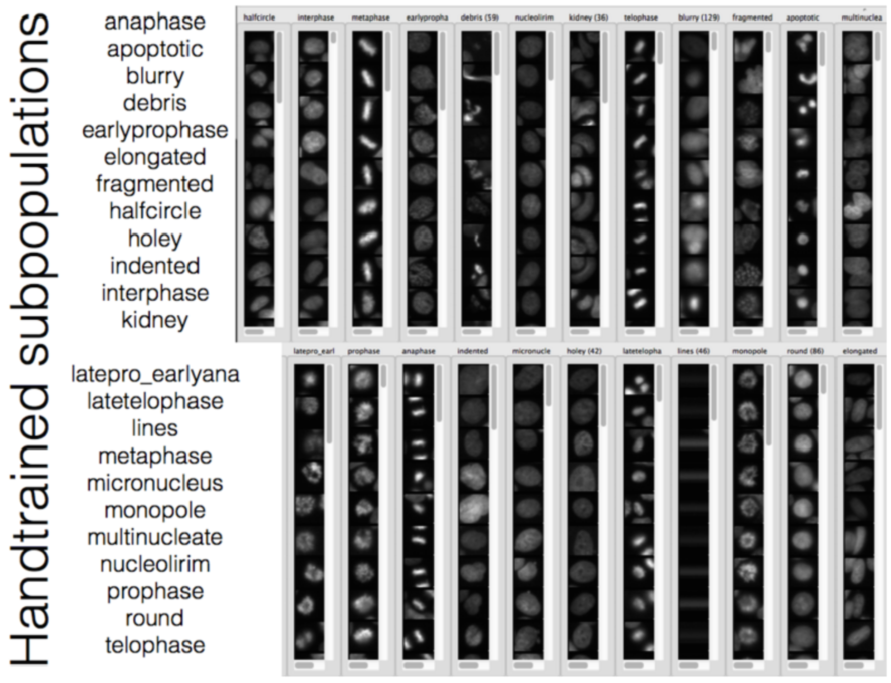


Figure A.6.: CPA Classifier window while curating SUBPOP

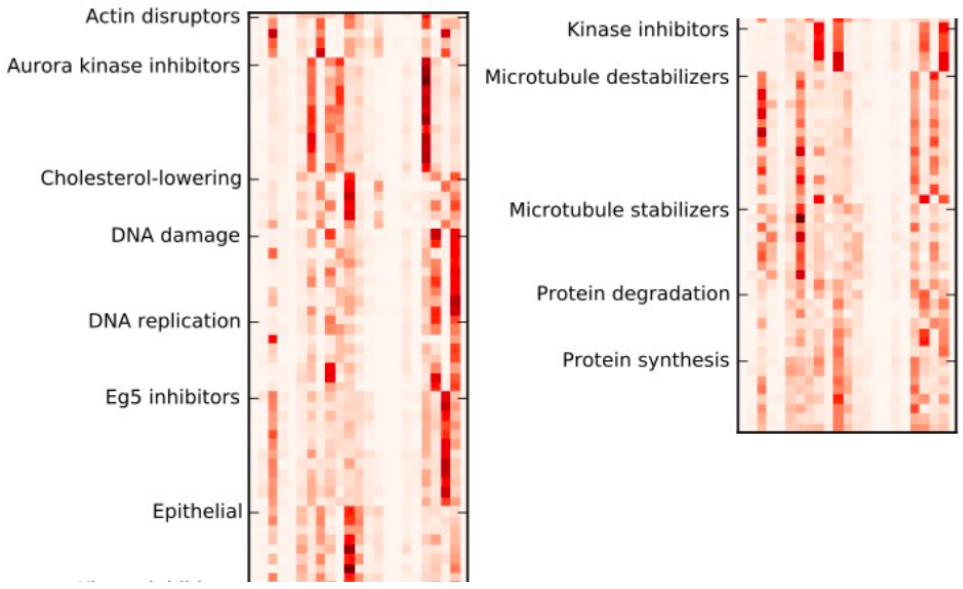


Figure A.7.: Visualization of all subpopulation treatment profiles

Bibliography

1. Gustafsdottir, S. M. *et al.* Multiplex cytological profiling assay to measure diverse cellular states. en. *PLoS One* **8,** e80999 (Feb. 2013).
2. Caicedo, J. C., Singh, S. & Carpenter, A. E. Applications in image-based profiling of perturbations. en. *Curr. Opin. Biotechnol.* **39,** 134–142 (June 2016).
3. Ljosa, V. *et al.* Comparison of methods for image-based profiling of cellular morphological responses to small-molecule treatment. *J. Biomol. Screen.* **18,** 1321–1329 (Dec. 2013).
4. Leek, J. T. *et al.* Tackling the widespread and critical impact of batch effects in high-throughput data. *Nat. Rev. Genet.* **11,** 733–739 (Oct. 2010).
5. Altschuler, S. J. & Wu, L. F. Cellular heterogeneity: do differences make a difference? en. *Cell* **141,** 559–563 (14 05 2010).
6. Jones, T. R. *et al.* CellProfiler Analyst: data exploration and analysis software for complex image-based screens. *BMC Bioinformatics* **9,** 482 (15 11 2008).
7. Dao, D. *et al.* CellProfiler Analyst: interactive data exploration, analysis and classification of large biological image sets. en. *Bioinformatics* (26 06 2016).
8. Ljosa, V., Sokolnicki, K. L. & Carpenter, A. E. Annotated high-throughput microscopy image sets for validation. en. *Nat. Methods* **9,** 637 (July 2012).
9. He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. arXiv: `1512.03385 [cs.CV]` (Oct. 2015).
10. Carpenter, A. E. Image-based chemical screening. *Nat. Chem. Biol.* **3,** 461–465 (Aug. 2007).
11. Laufer, C., Fischer, B., Billmann, M., Huber, W. & Boutros, M. Mapping genetic interactions in human cancer cells with RNAi and multiparametric phenotyping. *Nat. Methods* **10,** 427–431 (May 2013).
12. Feng, Y., Mitchison, T. J., Bender, A., Young, D. W. & Tallarico, J. A. Multi-parameter phenotypic profiling: using cellular effects to characterize small-molecule compounds. *Nat. Rev. Drug Discov.* **8,** 567–578 (July 2009).
13. Young, D. W. *et al.* Integrating high-content screening and ligand-target prediction to identify mechanism of action. *Nat. Chem. Biol.* **4,** 59–68 (Jan. 2008).
14. Li, K., Gustafsdottir, S. M., Ljosa, V., *et al.* Toward performance-diverse small-molecule libraries for cell-based phenotypic screening using multiplexed high-dimensional profiling. *Proceedings of the National Academy Sciences* (2014).
15. Carpenter, A. *et al.* CellProfiler: image analysis software for identifying and quantifying cell phenotypes. *Genome Biology* **7.** `<http://genomebiology.com/2006/7/10/R100>` (Oct. 2006).

16. Kamentsky, L. *et al.* Improved structure, function and compatibility for CellProfiler: modular high-throughput image analysis software. *Bioinformatics* **27,** 1179–1180 (15 04 2011).
17. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. en. *Nature* **521,** 436–444 (28 05 2015).
18. Kraus, O. Z., Ba, L. J. & Frey, B. Classifying and Segmenting Microscopy Images Using Convolutional Multiple Instance Learning. arXiv: `1511.05286 [cs.CV]` (17 11 2015).
19. He, K., Zhang, X., Ren, S. & Sun, J. Identity Mappings in Deep Residual Networks. arXiv: `1603.05027 [cs.CV]` (16 03 2016).
20. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **12,** 2825–2830 (2011).
21. Loo, L.-H., Wu, L. F. & Altschuler, S. J. Image-based multivariate profiling of drug responses from single cells. *Nat. Methods* **4,** 445–453 (Jan. 2007).
22. Swinney, D. C. & Anthony, J. How were new medicines discovered? en. *Nat. Rev. Drug Discov.* **10,** 507–519 (July 2011).
23. Peck, D. *et al.* A method for high-throughput gene expression signature analysis. en. *Genome Biol.* **7,** R61 (2006).
24. Fuchs, F. *et al.* Clustering phenotype populations by genome-wide RNAi and multiparametric imaging. *Mol. Syst. Biol.* **6,** 370 (Aug. 2010).
25. Ohnuki, S. *et al.* Diversity of Ca2+-induced morphology revealed by morphological phenotyping of Ca2+-sensitive mutants of Saccharomyces cerevisiae. en. *Eukaryot. Cell* **6,** 817–830 (May 2007).
26. Bray, M.-A. *et al.* Cell Painting, a high-content image-based assay for morphological profiling using multiplexed fluorescent dyes. en. *Nat. Protoc.* **11,** 1757–1774 (Sept. 2016).
27. Zhang, J. & Hu, J. *Image Segmentation Based on 2D Otsu Method with Histogram Analysis* in *Computer Science and Software Engineering, 2008 International Conference on* **6** (Dec. 2008), 105–108.
28. Lilliefors, H. W. On the Kolmogorov-Smirnov Test for Normality with Mean and Variance Unknown. *J. Am. Stat. Assoc.* **62,** 399–402 (1967).
29. Scholkopf, B. & Smola, A. J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond* (MIT Press, Cambridge, MA, USA, 2001).
30. Slack, M. D., Martinez, E. D., Wu, L. F. & Altschuler, S. J. Characterizing heterogeneous cellular responses to perturbations. en. *Proc. Natl. Acad. Sci. U. S. A.* **105,** 19306–19311 (Sept. 2008).
31. Singh, S. *et al.* Morphological Profiles of RNAi-Induced Gene Knockdown Are Highly Reproducible but Dominated by Seed Effects. *PLoS One* **10,** e0131370 (21 07 2015).
32. Krizhevsky, A., Sutskever, I. & Hinton, G. E. in *Advances in Neural Information Processing Systems 25* (eds Pereira, F., Burges, C. J. C., Bottou, L. & Weinberger, K. Q.) 1097–1105 (Curran Associates, Inc., 2012).

33. Ren, S., He, K., Girshick, R. & Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. en. *IEEE Trans. Pattern Anal. Mach. Intell.* (June 2016).

34. Amodei, D. *et al.* Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. arXiv: `1512.02595 [cs.CL]` (Aug. 2015).

35. Bahdanau, D., Cho, K. & Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. arXiv: `1409.0473 [cs.CL]` (Jan. 2014).

36. Xu, K. *et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention* in *Proceedings of The 32nd International Conference on Machine Learning* (2015), 2048–2057.

37. Gatys, L. A., Ecker, A. S. & Bethge, M. A Neural Algorithm of Artistic Style. arXiv: `1508.06576 [cs.CV]` (26 08 2015).

38. Mnih, V. *et al.* Playing Atari with Deep Reinforcement Learning. arXiv: `1312.5602 [cs.LG]` (19 12 2013).

39. Silver, D. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529,** 484–489 (27 01 2016).

40. Simonyan, K. & Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv: `1409.1556 [cs.CV]` (Apr. 2014).

41. Karpathy, A. Stanford University CS231n: Convolutional Neural Networks for Visual Recognition. <`http://cs231n.stanford.edu/syllabus.html`>.

42. He, K., Zhang, X., Ren, S. & Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proc. IEEE* (2015).

43. Ioffe, S. & Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv: `1502.03167 [cs.LG]` (Nov. 2015).

44. Sutskever, I., Martens, J., Dahl, G. & Hinton, G. On the importance of momentum and initialization in deep learning. *Conference on Machine Learning* (2013).

45. Tieleman, T. & Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning* (2012).

46. Kingma, D. & Ba, J. Adam: A Method for Stochastic Optimization. arXiv: `1412.6980 [cs.LG]` (22 12 2014).

47. Zeiler, M. D. & Fergus, R. *Visualizing and Understanding Convolutional Networks* en. in *Computer Vision – ECCV 2014* (eds Fleet, D., Pajdla, T., Schiele, B. & Tuytelaars, T.) (Springer International Publishing, June 2014), 818–833.

48. Snoek, J., Larochelle, H. & Adams, R. P. in *Advances in Neural Information Processing Systems 25* (eds Pereira, F., Burges, C. J. C., Bottou, L. & Weinberger, K. Q.) 2951–2959 (Curran Associates, Inc., 2012).

49. Hochreiter, S. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *Internat. J. Uncertain. Fuzziness Knowledge-Based Systems* **06,** 107–116 (1998).

50. Kandaswamy, C. *et al. Improving transfer learning accuracy by reusing Stacked Denoising Autoencoders* in *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (Oct. 2014), 1380–1387.

51. Sommer, C., Straehle, C., Kothe, U. & Hamprecht, F. A. *Ilastik: Interactive learning and segmentation toolkit* in *Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on* (Mar. 2011), 230–233.

52. Held, M. *et al.* CellCognition: time-resolved phenotype annotation in high-throughput live cell imaging. *Nat. Methods* **7,** 747–754 (Sept. 2010).

53. Orlov, N. *et al.* WND-CHARM: Multi-purpose image classification using compound image transforms. *Pattern Recognit. Lett.* **29,** 1684–1693 (Jan. 2008).

54. Horvath, P., Wild, T., Kutay, U. & Csucs, G. Machine learning improves the precision and robustness of high-content screens: using nonlinear multiparametric methods to analyze screening results. en. *J. Biomol. Screen.* **16,** 1059–1067 (Oct. 2011).

55. Boutros, M., Brás, L. P. & Huber, W. Analysis of cell-based RNAi screens. *Genome Biol.* (2006).

56. Pau, G., Zhang, X., Boutros, M. & Huber, W. *imageHTS: Analysis of high-throughput microscopy-based screens* 2013.

57. Pelz, O., Gilsdorf, M. & Boutros, M. web cellHTS2: A web-application for the analysis of high-throughput screening data. *BMC Bioinformatics* **11,** 1–6 (2010).

58. Berthold, M. R., Cebron, N., Dill, F., Gabriel, T. R., Kötter, T., *et al.* KNIME: The Konstanz information miner (2008).

59. Jaderberg, M., Simonyan, K., Zisserman, A. & Kavukcuoglu, K. in *Advances in Neural Information Processing Systems 28* (eds Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M. & Garnett, R.) 2008–2016 (Curran Associates, Inc., 2015).

60. Abadi, M. *et al.* TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. arXiv: `1603.04467 [cs.DC]` (14 03 2016).

61. Ren, S., He, K., Girshick, R. & Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv: `1506.01497 [cs.CV]` (Apr. 2015).

62. Veit, A., Wilber, M. & Belongie, S. Residual Networks are Exponential Ensembles of Relatively Shallow Networks. arXiv: `1605.06431 [cs.CV]` (20 05 2016).

63. Sønderby, S. K., Sønderby, C. K., Maaløe, L. & Winther, O. Recurrent Spatial Transformer Networks. arXiv: `1509.05329 [cs.CV]` (17 09 2015).