Modeling the space of subcellular location patterns using images and other sources of information

Luis Pedro Coelho

September 2011

CMU-CB-11-104

Publisher

Lane Center for Computational Biology

School of Computer Science

Carnegie Mellon University

Pittsburgh, PA 15213

Thesis Committee

Robert F. Murphy (Carnegie Mellon University)

Takeo Kanade (Carnegie Mellon University)

Russell Schwartz (Carnegie Mellon University)

Chakra Chennubhotla (University of Pittsburgh)

Hagit Shatkay (University of Delaware)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computational Biology

Copyright 2011 Luis Pedro Coelho

This work has been funded by the Fulbright Program, NIH grant R01 GM075205, the Fundação Para a Ciência e a Tecnologia (grant SFRH/BD/37535/2007), and the Siebel Scholars Foundation. These organizations had no role in the design or execution of this work.

Keywords: subcellular proteomics, bioimage informatics, topic models, semi-supervised learning, local image features, cell nucleus segmentation, fluorescent microscopy.

To Anna Teresa Coelho

Acknowledgments

I wish to thank Prof. Robert F. Murphy, my advisor, for all his support and guidance in the work presented. I also thank Prof. Hagit Shatkay for all her help and support in framing and developing the vision of the project. I also thank Prof. Takeo Kanade, Prof. Russell Schwartz, and Prof. Chakra Chennubhotla for the several discussions and advice that they offered.

I also thank all of the members of the Murphy Lab for the many discussions. Dr. Tao Peng with whom I worked on pattern unmixing and Aabid Shariff with whom I shared an office and worked on nuclear segmentation were directly involved in the work in this dissertation. Josh Kangas, Armaghan Naik, and Jennifer Bakal were essential to the RandTag project.

Outside of the Murphy Lab, Prof. Jon Jarvik, Prof. Peter Berget, and Dr. Margaret Furhman were an invaluable part of the RandTag project and I thank them for their time, expertise, and patience in teaching me.

Finally, I thank the Fulbright Program and the Fundação Para a Ciência e a Tecnologia (grant SFRH/BD/37535/2007) for their financial support.

The study of proteins includes the study of protein location as one of its major areas of interest. This study can be approached one protein at a time, or systematically, in a high-throughput fashion, an approach that has been called *location proteomics*.

Subcellular location can either be *predicted*, based on the protein sequence, homology, or other circumstantial evidence such as interaction patterns; or *determined* by direct observation.

The prediction approach has the advantage that it requires less data (sometimes only the sequence). On the other hand, its results are not as conclusive as those obtained from direct data. Furthermore, prediction is, at least with the most widely used techniques, obtained from static data (sequence, functional annotations, binding patterns,…). Thus, most systems will predict the same location independently of cell type or cell state.

Direct data is normally in the form of images of fluorescently labeled proteins. The automatic analysis of such images has by now a decade long history. Most of the work has been done in the *supervised learning* mode: the researcher specifies a set of interesting location classes (corresponding to the organelles of interest), finds a few examples of each, and trains a classifier to recognise them in unlabeled data. Some work has shown usage of *unsupervised learning* methods for the problem. In this approach, the different proteins are clustered together into an hierarchy or a set of groups.

This work shows that direct and indirect data can be combined into a single model and inferences can be made which depend on all of it. In particular, the model can project multiple modalities into the same space and return a label which is based on all its input data.

I will also propose new image representations for use with subcellular location images. They are adapted from Speeded-Up Robust Features (SURF), but adapted to the setting where, in addition to the protein channel, a reference channel (in the case under study, a DNA marker) is present. I will use supervised classification as a validation problem and show that SURF outperforms traditional approaches and that adding DNA information outperforms traditional SURF.

Contents

List of Figures

§1 Introduction

1.1 Location Proteomics

Subcellular location is one of the most important characteristics of a protein as evidenced by the fact that it is carefully controlled and regulated by the cell. There is, therefore, a need to study subcellular location at a proteome-wide level, an enterprise that has been termed "location proteomics" (Murphy, 2005).

The best data source for location determination is fluorescent microscopy imaging of tagged proteins. Due to the large datasets, automatic image analysis is required. Additionally, automated systems are more reproducible than humans and achieve accuracies that are at least comparable if not better (Murphy et al., 2003; Nattkemper et al., 2003).

Typically, the approach taken has been to pre-select a subset of the cell organelles as the set of locations of interest and then attempt to visually or automatically classify each protein into one of these classes.

The disadvantage of this approach are twofold: (1) the set of classes is predefined by the researcher, and (2) many proteins are present in more than one location, as many as one third of proteins localise in multiple locations (S.-C. Chen, Zhao, et al., 2007; W.-K. Huh et al., 2003). The first problem can be mitigated by iteratively re-defining the set of classes, but one would benefit from being able to automate the class discovery step as it could then be performed in a more data-dependent fashion (i.e., classes would emerge from the images).

There have been some previous attempts at applying clustering or other forms of unsupervised learning to this problem (García Osuna et al., 2007; Hamilton, J. T. H. Wang, et al., 2009).

1.2 Multiple Data Sources

In addition to the images, other sources of information can be available. Many proteins have had a previous location characterisation by other researchers. We know the protein and gene sequence, we might have information about homologues in related species, or about its function in the cell. All of these pieces of information can aid in the determination of subcellular location. In fact, they have all been used in the *prediction* of subcellular location, an active research field in its own right, whose goal is a system which can give the best estimate of where a protein will localise in the absence of direct observations (see Section 2.3 and the references therein for more information).

Integration of multiple sources of data is a traditional problem in computational biology, particularly in areas closer to bioinformatics, and several approaches have been proposed and contrasted (Section 2.4 contains a very brief overview of this literature). These efforts have, however, not handled image data. Image data is still processed on its own.

In the computer vision literature, however, there have been many attempts at using image and other sources of information, most often associated text, structured or unstructured.

This dissertation bridges these two areas of research. I adapt and apply state-of-the-art computer vision techniques to the problem of determining subcellular location from images and other sources of information.

Eventually, this research program will result in reasoning systems that can integrate all known information about a specific protein into a single, best, conclusion. Towards that goal, I propose a novel method to integrate images with known annotations of location and function, as well as homology.

1.3 Thesis Statement

I make two claims that I will then defend.

1. There are numeric representations of biological images that are an improvement on traditional sets of image-level or cell-level features.
2. Image and non-image data can be integrated into a single representation of the protein location pattern. This can lead to semi-supervised methods that overcome disadvantages of both supervised and unsupervised approaches.

In particular, I show that graphical models derived from topic models can incorporate information from images and from other sources into a single model leading to better results that would be obtainable with images alone.

As part of the path towards the goal of integrating this image, I studied the problem of image representation and made contributions to that research problem. With Tao Peng, I worked in unsupervised subcellular pattern unmixing and we proposed a method which can be shown to give good results in unmixing punctate patterns. In a parallel effort, I used Speeded-Up Robust Features, SURF, introduced by Bay, Ess, et al. (2008), for representing images. The initial impetus for the use of such local features was that, with the use of clustering, the image can be seen as a bag of visual words, a good representation on which to apply topic models. For validation of the features, I used a classification problem where SURF obtains better results than the state of the art.

1.4 Randomly Tagged Proteins

The main source of image data I used was data from the RandTag project. It was previously described by García Osuna et al. (2007). Here I review the basic protocol behind the data generation.

CD Tagging

CD tagging is a technique for inserting a peptide into existing proteins by genomic modification, developed by Jarvik, Adler, et al. (1996). This allows for live-cell imaging (unlike with antibody staining methods).

An alternative to CD tagging for live-cell imaging is to insert a plasmid with the protein of interest tagged (typically at one of the terminals). CD tagging has the advantage that the genomic promoter is preserved and the regulation of gene transcription is less affected than with other methods that are used such as plasmid insertion (there is still the possibility the insertion disturbs a regulatory binding site in an intronic region of the gene). Further, plasmids are normally designed for over-expression of the protein. While this produces visually more appealing images, it can lead to artifacts. Consider the case where a protein binds to another membrane-bound protein which resides in a vesicular component. By over expressing it, the extra copies of the protein might not have binding partners and become mislocalized.

By looking at proteins that had previously been characterized with respect to their subcellular locations, Jarvik, Fisher, et al. (2002) showed that, in most cases, the modification to the protein does not change its location. In their work 42 out of 49 proteins (86%) were in their "expected" location. Telmer et al. (2002) equally reported that CD tagging did not affect localization of human nucleolin.

Imaging

Before imaging, the medium is replaced by Optimem with Hoechst (at a concentration of $1\mu g/mL$) for DNA labeling. Imaging was performed on an IC100 automated widefield microscope using $40\times$ magnification. The resulting pixel is $0.6\mu m$ wide.

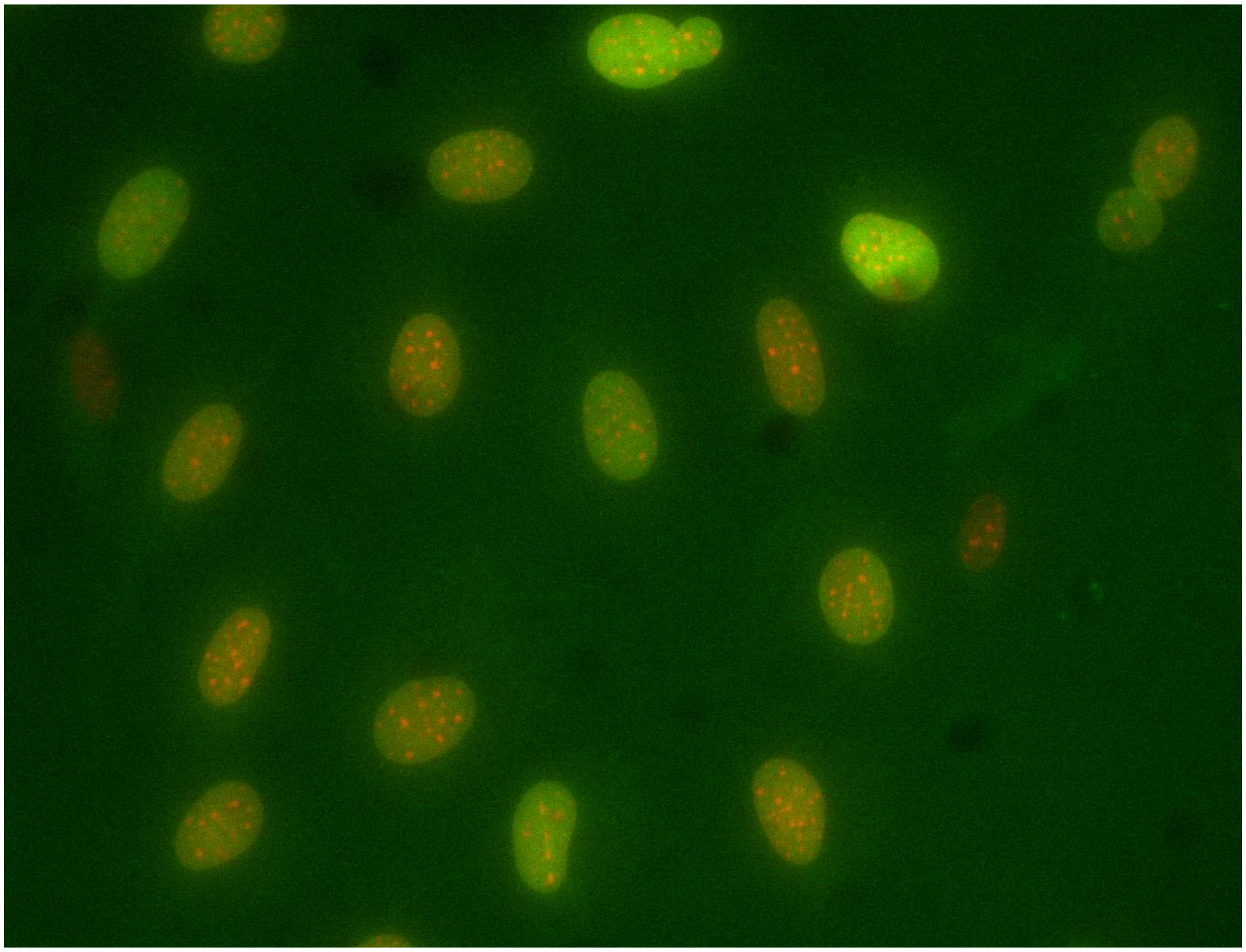


Figure 1.1: Example Widefield Microscope Image. Tagged gene is Lamin A (Lmna, Uniprot accession number B3RH23). Image is a false color image: red is a nuclear marker (Hoechst), while green is the GFP-tagged protein. The image has been contrast stretched for publication.

In addition the the widefield images described above, some clones were imaged on a confocal microscope as well. At the beginning of the project, only a few images were taken on that microscope due to time constraints. The set of clones to image was selected by visual inspection of the wide-field clones. At the end of the project, automation became available on the confocal microscope, allowing for imaging of full plates.

Figures 1.1 and 1.2 show examples of images obtained with these two technologies. Clearly, images are visually very different, with the confocal image having higher contrast between signal and background and appearing crisper. The images are both examples of tagging the same protein, Lamin A, but different clones (i.e., these were two independent insertions).

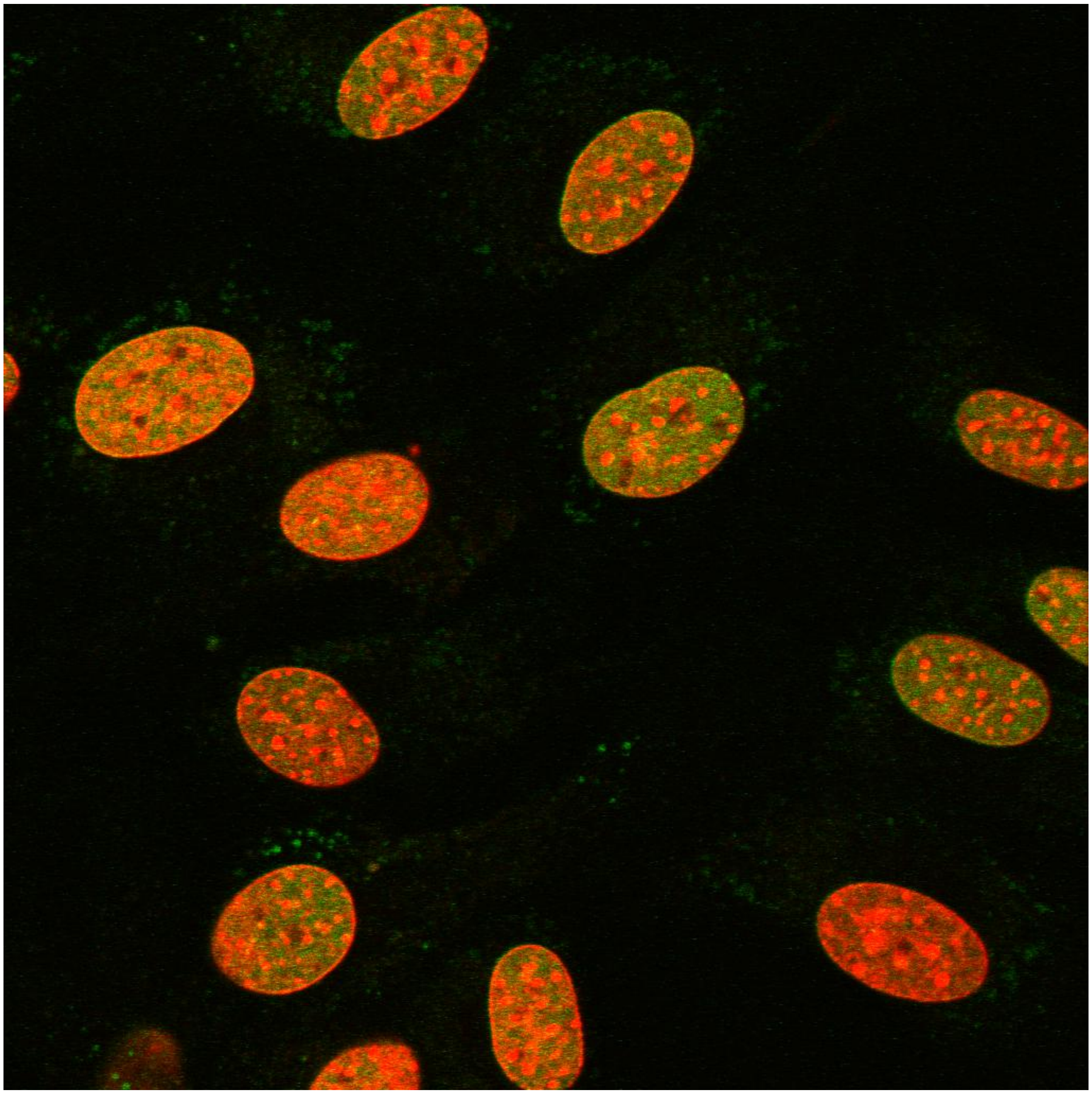


Figure 1.2: Example Confocal Microscope Image. Tagged gene is Lamin A (Lmna, Uniprot accession number B3RH23). Image is a false color image: red is a nuclear marker (Hoechst), while green is the GFP-tagged protein. The image has been contrast stretched for publication.

RandTag Collection

The RandTag collection thus contains images from 5475 different wells (more images were collected, these are numbers after filtering of out-of-focus and empty images, see Appendix B). Of these, there are 5064 wells with widefield images and 524 wells with confocal images (113 wells have both confocal and widefield images).

Unfortunately, for many of these, we do not know what the imaged protein is. Because the clones must be sequenced to learn where the insertion took place, and because this step is not infallible, it can happen that, after having imaged the clone, we are unable to determine which protein was tagged. Of the wells with images, the depicted protein is known for 208 of them. This corresponds to 144 distinct proteins.

The full dataset contains images that were collected by myself (the majority), some collected by Dr. Elvira Gárcia-Osuna, Dr. Margaret Fuhrman, Armaghan Naik, and Joshua Kangas.

§2 State of the Art

This chapter reviews the state of the art in the several fields that are combined in this work. In particular, it surveys past efforts at large-scale automated subcellular location determination with a special focus on the image representations that have been proposed. It also reviews the subcellular location prediction literature. This literature is very vast and only the most important systems are presented. As in other problems in bioinformatics, integration of information is a common approach. As I have also proposed a system which integrates information, I give an overview of the family of technologies that have been used to achieve this. Finally, I present an overview of the technology I chose for this work, namely topic models, and how they have been used for computer vision and image analysis.

2.1 Image Representations for Subcellular Location Analysis

Images can be processed simply as a two dimensional array of pixel values, of which I am aware of only one single report for subcellular location determination (Danckaert et al., 2002).[1]

Traditionally, however, images are summarised by a smaller set of features. The simplest model is to use, for each image, a fixed set of such pre-defined features that attempt to capture relevant visual properties of the image.

[1]Danckaert et al. (2002) used a shallow neural network, with a single hidden layer. I know of no reports of the application of more recently developed, multi-layer, "deep" belief networks (Hinton, 2007; H. Lee et al., 2009). This could be an interesting future research avenue.

Subcellular Location Features

Most prior work in subcellular location uses the feature representation of images, where a feature is a function that can be computed on the image or parts of it. Well designed features capture important variation in the patterns, while being insensitive to unimportant variations (e.g., rotation, illumination phenomena,...). Traditionally, fifty to a few thousand features are used making this a medium-dimension representation when compared to the higher dimension of the pixel space.

This approach for subcellular location has shown to be very effective in terms of both high accuracy and the applicability to different datasets (Boland, Markey, et al., 1998; Boland and Murphy, 2001; S.-C. Chen, Zhao, et al., 2007). The literature is quite vast, I will only review features that were used in this project.

Generally speaking, features are applicable either to an entire field (image) which can contain multiple cells (Huang and Murphy, 2004a) or to segmented single-cell regions. Naturally, field-level features are easier to compute as they do not depend on pre-segmentation. Most of the features presented below are of this type, except where otherwise noted.

Haralick Texture Features

Texture features are one of the most powerful field-level feature sets (Boland, Markey, et al., 1998; Huang and Murphy, 2004a; Nanni, Brahnam, et al., 2010). A popular family of texture features are the Haralick features (Haralick et al., 1973).

Haralick features are computed on the adjacency matrix, A, where $A(i,j)$ is the number of times that a pixel with the value i is observed next to a pixel of value j in a given direction. In the original definition, there are four directions (North, South, North-west, and North-east) and the adjacency matrix is symmetric.[2] The matrix can be normalised to be a joint probability distribution $P(i,j)$.

[2]A possible alternative is, if more than one channel is present in the image, to define $A(i,j)$ as the number of times that a pixel takes value i in the first channel and value j in the second channel. I know of no use of this construct.

Statistics are then computed on this matrix P. I first define a few standard aggregates:

$$P_i(k) = \sum_j P(k, j), \tag{2.1}$$

$$P_{x+y}(k) = \sum_{i,j} [\![\, i + j = k \,]\!] P(i, j), \tag{2.2}$$

$$P_{|x-y|}(k) = \sum_{i,j} [\![\, |i - j| = k \,]\!] P(i, j), \tag{2.3}$$

$$\mu = \sum_i P_i(i), \tag{2.4}$$

$$\sigma^2 = \sum_i (i - \mu)^2 P_i(i). \tag{2.5}$$

Based on these, I can define the first few features:

$$f_1 = \sum_{i,j} P(i, j)^2 \tag{2.6}$$

$$f_2 = \sum_k k^2 P_{|x-y|}(k) \tag{2.7}$$

$$f_3 = \frac{1}{\sigma^2} \left(\sum_{i,j} ijP(i, j) - \mu^2 \right) \tag{2.8}$$

$$f_4 = \sum_i (i - \mu)^2 P_i(i) \tag{2.9}$$

$$f_5 = \sum_{i,j} \frac{P(i, j)}{1 - (i - j)^2} \tag{2.10}$$

$$f_6 = \sum_k kP_{x+y}(k) \tag{2.11}$$

$$f_7 = \sum_j (k - f_6)^2 P_{x+y}(k) \tag{2.12}$$

$$f_8 = H \{P_{x+y}(i, j)\} \tag{2.13}$$

$$f_9 = H \{P(i, j)\} \tag{2.14}$$

$$f_{10} = \text{var} \{P(i, j)\} \tag{2.15}$$

$$f_{11} = H \left\{P_{|i-j|}(k)\right\} \tag{2.16}$$

$$f_{12} = \frac{f_9 - HXY1}{\max(HX, HY)} \tag{2.17}$$

$$f_{13} = (1 - \exp[-2.0(HXY2 - f_9)])^{1/2}, \tag{2.18}$$

where H is the entropy operator $H(P) = -\sum_i P(i) \log P(i)$ and

$$HX = H(p_x) \tag{2.19}$$

$$HY = H(p_y) \tag{2.20}$$

$$HXY1 = -\sum_{i,j} P(i,j) \log \{p_x(i)p_y(j)\} \tag{2.21}$$

$$HXY2 = -\sum_{i,j} p_x(i)p_y(j) \log \{p_x(i)p_y(j)\} \tag{2.22}$$

The 14th features is the "maximal correlation coefficient," defined as the square root of the second largest eigenvalue of Q, the matrix defined by

$$Q(i,j) = \sum_k \frac{P(i,k)P(j,k)}{\left[\sum_x P(x,i)\right]\left[\sum_y P(k,y)\right]}. \tag{2.23}$$

This feature is often not computed due to numerical issues or non-convergence in the computation (Boland, Markey, et al., 1998).[3] I have not used this feature either.

In addition to the features computed on the image, I computed the same feature set on down-sampled versions of the image. This approach to multi-resolution has been reported to give good results in classification of subcellular patterns (Chebira et al., 2007; J. Y. Newberg et al., 2009).

Local Binary Patterns

Local binary patterns are a recent proposal by Ojala et al. (2002), which have been reported to give good results when applied to bioimage classification (Nanni and Lumini, 2008).

Local binary patterns consider each pixel individually and assign it a code. The overall features are the normalised code histograms.

For each pixel, its neighbourhood is analysed by collecting p points, at a distance r of the central point (p and r are parameters of the algorithm). The points are at equal angles from each other. The raw signature of the point is then this p-long string of pixel values: $p_1, p_2, \cdots, p_P$. The sequence is binarised by considering the value of the central point p_c:

$$b_i = [\![\, p_i > p_c \,]\!]. \tag{2.24}$$

These binary signatures are further grouped in several ways, which depend on the desired invariance properties. For rotational invariance, all of the strings which result from a cyclic string rotation are considered equivalent (i.e.,

[3]It is not clear to me whether the main issue is that determining the second eigenvalue is a fundamentally ill-conditioned problem (for certain particular images of interest) or whether the reported issues are a limitation of the algorithms that are typically used for this problem.

0011001 is equivalent to its rotated form 1100100) and can be represented by the lowest valued element of the equivalence class.

In this work, 8 pixels, evenly distributed, at an 8 pixel distance from the central point were considered the neighbourhood.

Object Features

For object feature computation, objects (contiguous above-threshold regions) are first identified. The following features can be computed on a protein field without cell-level segmentation:

1. number of objects,
2. Euler number of the image (this is the number of objects minus the number of holes),
3. average object size (in pixel units),
4. variance of the object sizes,
5. ratio of the sizes of the largest and smallest object sizes.

With cell-level segmentation, I can additionally compute the following three features:

1. average distance of the object centre to the center of fluorescence (COF),
2. variance of object distance to COF,
3. ratio of largest to smallest distance to COF.

If a reference channel, such as a DNA channel, is present, I can additionally compute a few features, of which all but the last two require cell-level segmentation:

1. average distance of object to DNA COF,
2. variance of distances of objects to DNA COF,
3. ratio of largest to smallest distance of object to DNA COF,
4. distance of protein COF to DNA COF,
5. ratio of DNA area to protein area (after thresholding),
6. fraction of protein image that overlaps with DNA.

Additionally, the fraction of fluorescence that is *not present in objects* is also measured. These and the edge features presented below were designed by Boland and Murphy (2001) especially for subcellular location analysis.

Edge Features

These features are based on edge filtering. In the original work, this was implemented using Canny (1986) edge detection. In this work, I used Sobel edge detection (for ease of implementation).

The features include the fraction of pixels that are contained in edges as well as several measures of how homogeneous the edge directions and intensities are. See the work of Murphy et al. (2003) for more details.

Skeleton Features

Skeleton features are, again, based on individual objects. For each object, convex hull and skeleton are computed.[4] Based on these, the following measures are taken:

1. The skeleton length ℓ,
2. the hull area h,
3. the object area a,
4. the total fluorescence on the object and the skeleton, f and s,
5. the number of branch points b.

The features for this object are the vector $[\ell, \ell/h, \ell/a, f/s, b/\ell]$. Values are averaged across a whole image to obtain image-level features.

Subcellular Pattern Unmixing

Multiple copies of the same protein will not necessarily all locate in the same organelle type. In one large-scale study of the yeast proteome, a third of proteins were annotated with multiple locations, which demonstrates that this is not a problem confined to "special case" proteins (S.-C. Chen, Zhao, et al., 2007; W.-K. Huh et al., 2003).

The Murphy Lab has explored "pattern unmixing" approaches to handle this problem (T. Peng, Bonamy, et al., 2010). In that work, given examples of pure patterns (i.e., where a protein is known to localize to only a single type of organelle), a model is learned that can assign a mixed pattern to fractions of these pure patterns.

In Section 4 I will present details and extensions for working in the *unsupervised* setting. In this settings, pure patterns are not defined a priori, but must be discovered automatically. This formulation is more appropriate for large-scale imaging projects.

[4]These operations were implemented as in the *mahotas* Python computer vision package described in Appendix A.3.

Generative Models

Image-derived generative models are models that can generate images that are statistically similar to those on which the model was learned. Generative models for cellular images is a new, but active research field (Lehmussola et al., 2008; Zhao and Murphy, 2007). They are useful as an input to systems biology simulation studies (by providing a spatial map of protein locations), but can also be seen as a representation space. Particularly if a parametric modeling approach is used, then the values of the learned parameters can be seen as features. For some classes of models, these parameters even have a biological interpretation and fitting them to data yields useful information about the underlying system. Initial work in this area has focused on modeling the nucleus, the cell shape, and punctate organelles.

I will briefly describe the methods of Zhao and Murphy (2007) for punctate patterns. The method will model the shape and apperance of the nucleus, the shape of the cell, and the location and shape of objects.

For modeling the nuclear shape, a medial axis model is built (the system is built for HeLa cells whose nuclei are approximately spheroidal without major bifurcations, other cell types might not be amenable to this model).

The cell shape itself is dependent on the nucleus. To understand this model, picture rays emanating from the center of the cell. A ray, pointing at an angle θ will cross the boundary of the nucleus at distance r_1 and the boundary of the cell at distance $r_2 > r_1$. The model assumes that the distance to the center from the nucleus or the plasma membrane is a function of the angle, certain shapes cannot be modeled. By discretizing the angles to units of 1°, one obtains 360 measurements of the ratio r_2/r_1. The average cell is recorded and the residuals are modeled by a lower-dimensional principal component model.

The positions of objects are modeled relative to the position of the nucleus and the cell boundary.

Interestingly, the authors used classification as a validation problem. Using real data, the parameters of fitted generative models were used as features for classification. While the results were not directly competitive with the current state of the art for classification on the same dataset, they were still good and showed that this model could capture much of the important variation in cell images.

Recently, T. Peng and Murphy (2011) presented an extension of this model to generate three-dimensional images.

These methods are for punctate patterns only as they generate objects that are independent Gaussians. Shariff et al. (2010, 2011); Shariff, Rohde, et al. (2009) presented an approach for microtubular patterns, but there is no currently general approach which can capture all possible patterns.

Local features

Unfortunately, the word feature is used with two different meanings in the computer vision literature. It can be used as above, to mean a value derived from the image (this usage is most like the machine learning usage). Alternatively, it can be used to mean a particular point of interest in the image (for example, corners).

Local features are, in fact, values computed on small regions of the image, which often also correspond to points of interest in the second sense. In the computer vision literature, they have become the method of choice for many application.

There are several ways to select the locations to use for local features. The simplest method is to form a grid, a variation that is normally referred to as "dense" sampling (Dalal and Triggs, 2005; Fei-Fei and Perona, 2005).

Another way is to pick points randomly. These methods are sometimes known as "patch-based" methods (Nowak and Jurie, 2006; James Wang and Y. Chen, 2004), particularly if then traditional "whole image" features such as texture features are computed on the small region. Patch-based methods have already been used, with success, for bioimage analysis. Bhattacharya et al. (2005) defined "Vivos," a basic element of a biological image as being a representative patch. S. Huh et al. (2009) used patch-based methods for subcellular location (see also the work by Marée et al. (2007) for an earlier implementation of a similar idea).

Finally, local features can be computed at points determined automatically to be "interest points." The most well known method is the SIFT (scale invariant feature transform) from Lowe (1999). SIFT includes both a interest point detector and a local feature descriptor which can be used independently in dense or random-sample mode.

For interest point detection, there are several methods (Tuytelaars and Mikolajczyk (2007) provide an excellent review of the literature). Given that I am going to extend SURF (for Speeded-Up Robust Features), I will present that method here.[5] SURF was designed with the express purpose of being computationally fast so many approximations are made for speed. In particular, integral images (Crow, 1984) are used for most of the computations.

If an image is defined as a table of pixel values $V(i,j)$, the integral image is defined by

$$I(i,j) = \sum_{x \leq i, y \leq j} V(x,y). \tag{2.25}$$

This enables very fast computation of the sum of a rectangular area of the image

$$\sum_{A<x\leq B} \sum_{C<y\leq D} V(x,y) = I(A,C) + I(B,D) - I(A,D) - I(B,C), \tag{2.26}$$

which is *independent of the size of the rectangle.*

The SURF interest point detector is a blob detector. Given a point in the image x, y and a scale σ^2, the Hessian matrix H is defined as

$$H(x,y,\sigma^2) = \begin{pmatrix} L_{xx} & L_{xy} \\ L_{yx} & L_{yy} \end{pmatrix}, \tag{2.27}$$

[5]The presentation and the figures were naturally heavily inspired by the original presentation of SURF by Bay (2006); Bay, Ess, et al. (2008); Bay, Tuytelaars, et al. (2006).

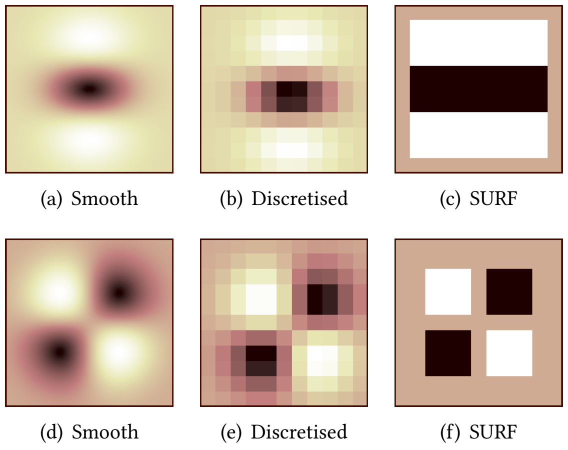


Figure 2.1: Approximate Gaussian Derivatives. On the left, are shown the smooth versions of $\frac{\partial^2 g}{\partial x^2}$ and $\frac{\partial^2 g}{\partial x \partial y}$, where g is the Gaussian function. In the middle, are shown the discretised version, in a small 11×11 patch. Finally, on the right, the fully approximated version, with constant boxes.

where L_{xx} is the convolution of the image with partial second derivative $\frac{\partial^2 g}{\partial x^2}$, where g is the Gaussian $g(x, y, s) = (2\pi\sigma^2)^{½} \exp\left\{-(x^2 + y^2)/\sigma^2\right\}$; L_{yy} and $L_{xy} = L_{yx}$ are defined analogously. The determinant of the Hessian, $|H|$, can be used to measure how well a Gaussian blob of size σ^2 matches at point x, y.

Other detectors had used this principle before (Lindeberg, 1998; Mikolajczyk and Schmid, 2002). What was innovative about SURF was the approximation that is used for computing the Hessian. As depicted in Figure 2.1, the true smooth derivative is never used for convolution with the image. Rather, a discretized version is used. SURF takes this process one step further and uses a version which is discretized to contain just constant values. The main advantage is that one can then take advantage of the integral image to *compute the convolutions in time independent of the scale* σ^2.

If D_{xx}, D_{xy} and D_{yy} are the responses of the box filters, then $|\tilde{H}|$

$$|\tilde{H}| = D_{xx}D_{yy} - w^2 D_{xy}^2 \tag{2.28}$$

is used in place of $|H|$. The correction factor w is presented as 0.9 in the original paper, with the justification that this is an approximation to

$$\frac{\|L_{xy}\|_F \|D_{xx}\|_F}{\|L_{xx}\|_F \|D_{xy}\|_F} = 0.9129\ldots \approx 0.9, \tag{2.29}$$

where $\|A\|_F = \sqrt{\sum_{ij} A_{ij}^2}$ (the Frobenius norm). The above is valid for $\sigma^2 = 1.2$, corresponding to an approximated box of side 9. For other values of σ^2 (and the corresponding approximations), the factor would be slightly different, but one can use this constant as an approximation.

Interestingly, however, the factor is elsewhere presented as 0.6 (e.g., in the review by Tuytelaars and Mikolajczyk (2007), whose first author is also a author in the SURF papers). The OpenSURF implementation also uses 0.6, which its author claims better matches the output of the closed-source SURF implementation.[6] Unfortunately, given that no source code was provided, it is impossible to know whether the good published results with SURF were computed using $w = 0.9$ or $w = 0.6$. In the current work, I used $w = 0.6$.

The above procedure, then gives a value for each point in x, y, σ^2 space. An interest point is then defined as those points which are simultaneously above a specified threshold and which are local maxima (in both spatial and scale dimensions). The sub-pixel location of the point is determined by fitting a quadratic curve locally and finding its maximum.

Interest points are finally assigned an orientation (this step can be skipped if rotational invariance is not necessary). In regions of size $4\sigma^2$ at a distance below $6\sigma^2$ away from the point, one computes the strength and angle of a Haar wavelet filter (which again can be done in constant time using the integral image). These response are further multiplied by a Gaussian of size $2\sigma^2$ centred on the interest point. The result is a collection of vectors. One sums all of the vectors in $\pi/3$ windows. The angle of the longest resulting vector is chosen as the angle of the interest point.

After interest points have been determined (or at any point otherwise chosen), a SURF descriptor is computed. It is a vector of 64 values, measuring the local gradients (as Haar wavelet responses).

Models Based on Local Features

Most models were presented as a *bag of visual words* model, in which images are seen as an unordered collection of salient features (the visual words). Extracted features are often obtained by k-means clustering on the space of local features (Moosmann et al., 2008). These methods only take into account local features and can only represent the presence or absence of certain features. Concepts that depend on the location of the object (such as the distinction between multiple punctate patterns) cannot be easily represented or captured by these models.[7]

However, some work has taken into account the geometrical relationships between sub-objects that co-occur in an image (Lazebnik et al., 2006; Philbin et al., 2008). Most of the approaches presented so far, learn from labeled data (i.e., the images are grouped into predefined categories), but there is some tentative work in learning from unlabeled data (Ahuja and Todorovic, 2007). To my knowledge, these methods have not yet been applied to cell image data.

Other Methods

Grammar based methods for representation of images were first proposed in the 1970's (Kanade, 1977), but have had a resurgence in recent years (S.-C. Zhu and Mumford, 2006). This framework is inspired by natural language

[6]Personal communication.

[7]A trick around this is to "sneak in" location information into the feature set. Zhao and Murphy (2007) used *distance to the DNA center of fluorescence* as an object feature in their object-based generative models. A similar technique could be applied here.

techniques. Images are represented as a collection of visual words, which appear in defined relationships (e.g., "next to" or "above"). The grammatical approach is not a continuous approach, but it can be a sparse approach.

Eigenvector based representations attempt to capture the principal *modes* of variation in a collection of images. This leads to methods that perform principal component analysis. The representations are, in general, not sparse, but sparsity can be imposed (Chennubhotla and Jepson, 2001). These methods operate on modeling the pixel value directly and are therefore not directly applicable to images of cells. However, it is possible to apply the model to transformed spaces to obtain a more applicable version. For example, Zhao and Murphy's (2007) generative model of cell shape is an eigenvector model in a space of shape features (see also Pincus and Theriot (2007) for a review of several methods many of which employ component analysis in the space of shape descriptors—a form of eigenshape analysis even if the authors do not use that terminology).

Diffeomorphic models have a long history in the biomedical research field (Miller and Qiu, 2009), but have only recently started to be applied to modeling cell images (Rohde et al., 2008). In this framework, shapes are deformed to one another. It can naturally model shapes such as nuclear or cell shape and characterize and catalog them.

2.2 Large Scale Studies of Subcellular Location

Large scale imaging studies for subcellular location determination have been performed in several organisms. W.-K. Huh et al. (2003) worked with fusion proteins in yeast (S. cerevisiae). S.-C. Chen, Zhao, et al. (2007) worked with this dataset and processed automatically using supervised learning methods. They obtained high accuracy (81%) and, most importantly, led to a reconsideration of some of the human labels.

The Human Protein Atlas (HPA) project is another high-throughput imaging project (Barbe et al., 2008). Again, the Murphy Lab is collaborating with this group to automatically process and analyse the images (J. Newberg and Murphy, 2008; J. Y. Newberg et al., 2009).

The Berkeley Drosophila Transcriptional Network Project has collected images of fruit fly (Drosophila) embryos showing spatial gene expression patterns (Luengo Hendriks et al., 2006). The public availability of this resource has led to many researchers attempting to attempt to extract information from the data computationally (Keränen et al., 2006; Pan et al., 2006; H. Peng, Long, et al., 2006; H. Peng and Myers, 2004; Zhou and H. Peng, 2007). Although the driving biological problem is different, some of the techniques employed for large scale mining of unsupervised image data can be applicable to subcellular location studies.

Several authors have presented results of applying clustering algorithms to large collection of images exhibiting different subcellular pattern (X. Chen and Murphy, 2005; X. Chen, Velliste, et al., 2003; Hamilton, J. T. H. Wang, et al., 2009).

2.3 Prediction of Subcellular Location

There is a vast literature in predicting subcellular location from sequence information or associated data. Different representations of the amino-acid sequence have been explored and often used together.

In the field of subcellular location prediction from multiple data sources there are many examples of data integration. An early work by Drawid and Gerstein (2000) combined both sequence motifs and expression data for an analysis of the yeast proteome (Kumar et al., 2002, see also). SherLoc is one more recent system whose current version (SherLoc2) integrates sequence properties, text properties (from published literature mentioning the protein of interest), and gene ontology terms (predicted from the sequence) (Briesemeister et al., 2009; Shatkay et al., 2007). Other notable systems include Proteome Analyst (Lu et al., 2004). Its recent version also parses abstracts with the aid of background knowledge given by the Gene Ontology (Fyshe et al., 2008). Protein-protein interaction data has also been shown to improve results under the assumption that binding partners will tend to co-localize (Mintz-Oron et al., 2009).

2.4 Integration of Multiple Sources of Information for Biological Inference

For subcellular prediction, data integration is common. Other fields of computational biology have also explored the problem of aggregating information.

Integration of different types of biological data has been explored in different problems, with many authors proposing different solutions such as integrating the prior knowledge as a Bayesian prior for gene expression analysis (Bernard and Hartemink, 2005), hierarchical clustering across multiple data-types (Segal and Koller, 2002), generative models for predicting protein function (Segal, Taskar, et al., 2001; Troyanskaya et al., 2003), amongst many others (Troyanskaya, 2005).

There is less work that uses image data. The efforts by K. Lee et al. (K. 2008) are the most advanced. Their work analyzed the results from the UCSF yeast dataset as well as protein–protein interaction network. Still, in their work, the images are not analyzed computationally; only the results of visual image tagging are used.

In the biomedical image field, G. Lee et al. (G. 2009) recently present a system for simultaneously representing image (pathology images) and non-image (mass spectrometry output) data for a medical application.

2.5 Topic Models

Topic modeling was first proposed for latent semantic discovery in text by Blei, Ng, et al. (2003) as a Bayesian version of the earlier probabilistic latent semantic indexing technique (Hofmann, 1999), itself derived from latent semantic

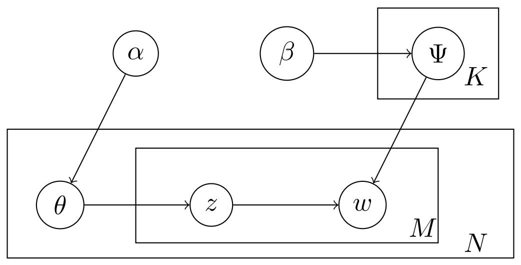


Figure 2.2: Traditional Latent Dirichlet Allocation. This is a Bayesian network representation of the model described by Blei, Ng, et al. (2003) and reviewed in the text.

indexing (Deerwester et al., 1990).[8]

Latent Dirichlet Allocation

Latent Dirichlet allocation is the basic text-based topic model. It is a generative model, using a bag-of-words representation (i.e., it can generate word lists, but not their ordering). It is best explained as a generative process.

The number of topics K is fixed and given in advance as are the hyperparameters $\alpha > 0$, $\beta > 0$, and the total number of possible words W. Topics are described by a categorical distribution over words Ψ_k (which is sampled from a symmetric Dirichlet distribution with parameter β). Each document is sampled in a multi-stage process. First, the document's topic distribution θ_i is sampled from the Dirichlet distribution parameterized by α. Secondly, for each word, first its topic and then its value are sampled (the number of words per document is assumed to be known—in practice it is always fixed, given in the data). The topic of a word is sampled from the categorical distribution parameterized by θ_i. Given a topic z_{ij}, I sample the word from the categorical distribution parameterized by $\Psi_{z_{ij}}$. In summary,

$$\Psi_k \sim \mathcal{D}_{\mathcal{W}}(\beta), \tag{2.30}$$

$$\theta_i \sim \mathcal{D}_{\mathcal{K}}(\alpha), \tag{2.31}$$

$$z_{ij} \sim \mathcal{C}(\theta_i), \tag{2.32}$$

$$w_{ij} \sim \mathcal{C}(\Psi_{z_{ij}}), \tag{2.33}$$

where $\mathcal{D}_N$ is the Dirichlet distribution with N dimensions and $\mathcal{C}$ is the categorical distribution.

The model is, of course, rarely used in its generative mode. Rather, given data, parameters are fit to it. Several methods have been proposed for this. In the original work of Blei, Ng, et al. (2003), a variation expectation-maximisation (EM) algorithm is derived. An alternative is Gibbs sampling, normally using a collapsed variation,

[8]An earlier work of Pritchard et al. (2000) had used the same probabilistic model for another problem but it did not use the interpretation of topic.

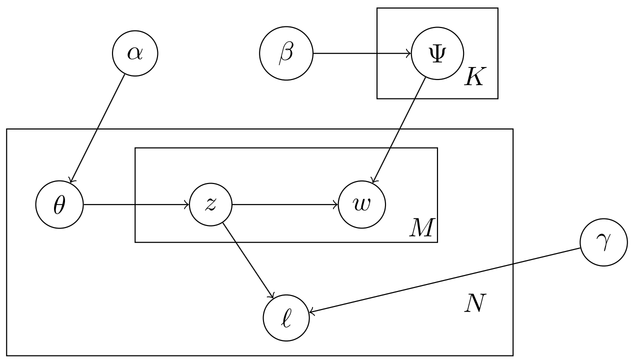


Figure 2.3: Supervised Latent Dirichlet Allocation. Extends the pure LDA model with a label node ℓ, which is dependent on $\bar{z}$.

where θ and Ψ are integrated out (Griffiths and Steyvers, 2004). Several variations have been presented for both of these techniques (Asuncion et al., 2008).

Supervised Latent Dirichlet Allocation

The representation of LDA (the θ vector for each document) can certainly be used for classification or regression. However, it has not been optimised for this usage. The supervised LDA (SLDA), proposed by Blei and Mcauliffe (2007), extends the LDA model to directly incorporate an output label (see the work of Shao et al. ("Semi-Supervised Topic Modeling for Image Annotation") for an alternative extension). The model is generic and can be used for either regression or classification. Given my usage, I describe a classification model.

The output label ℓ is assumed to be given in the training examples and is dependent on all of the z values through their normalised histogram:

$$\bar{z}_k^{(i)} = \frac{\sum_j [\![z_j^{(i)} = k]\!]}{\text{Nr of words in document } i}. \tag{2.34}$$

ℓ can be defined in several ways, the simplest of which is that, if it is a binary value, $\log P(\ell|\bar{z}, \gamma) = \bar{z}^T \gamma$. By depending on the observed z values, when the model is fit to training data, a relationship is learned between the model words and the label.[9] In Section 6.1, I will describe a slightly more complex model for generating the label.

For learning γ, one can maximise the likelihood, set a prior, or even use a max-margin approach (J. Zhu et al., 2009).

2.6 Topic Models with Images

Although proposed initially for use in text documents, topic models were soon being applied to image data (Blei and Jordan, 2003). In recent years, the computer vision community has actively and profusely explored the use of

[9]This would not be the case if the dependency was directly on θ, even though $E[\bar{z}] = \theta$. This point was made to me by Sean Gerrish in a personal communication.

topic modeling with images (or models which should perhaps be referred to as being “derived from topic models,” as the modeling of topics is often no longer the major concern) (Du et al., 2009; Li et al., 2009; Moosmann et al., 2008; Philbin et al., 2008; Porway et al., 2008; L. Zhu et al., 2009; S.-C. Zhu and Mumford, 2006).

Some of these models also make use of other sources of data. Of particular inspiration to the model I developed is the model of Li et al. (2009) which incorporated images and tags, the short text-based annotations, popular in many online services, such as flickr, (http://www.flickr.com).

§3 Nuclear Segmentation

Most of the material in this chapter first appeared in Nuclear Segmentation in Microscope Cell Images: A Hand-Segmented Dataset and Comparison of Algorithms *by Luis Pedro Coelho, Aabid Shariff, and Robert F. Murphy; in Proceedings of the IEEE International Symposium on Biomedical Imaging, pp. 518–521, 2009, [DOI].*

3.1 Introduction

Some of the features used below are cell-level features and require that the field be segmented into single cell regions. We perform this segmentation as a two-step process: first we segment in the nuclear region using only the nuclear channel and, subsequently, the individual nuclei serve as seeds for seeded methods, such as watershed (Beucher and Lantuéjoul, 1979). This is a standard procedure (Glory and Murphy, 2007; Jones et al., 2005; Velliste and Murphy, 2002).

For nuclear segmentation, we tried several methods and evaluated them quantitatively. For that, we built a dataset of hand-segmented images. The dataset is composed of two different collections (Table 3.1). The first collection is of U2OS cells, originally created for a study of pattern unmixing algorithms (T. Peng, Bonamy, et al., 2010). Figure 3.1 shows two images from this collection. An initial set of 50 images from this collection was chosen, but 2 images were rejected as containing no in-focus cells.

	U2OS	NIH3T3
Size in Pixels	1349×1030	1344×1024
Nr. Cells	1831	2178
Avg. Cover	23%	18%
Min Nr. Cells	24	29
Max Nr. Cells	63	70

Table 3.1: Main Properties of the Two Collections. Avg. cover denotes the percentage of pixels covered by cells. The minimum and maximum are over all the images in each collection.

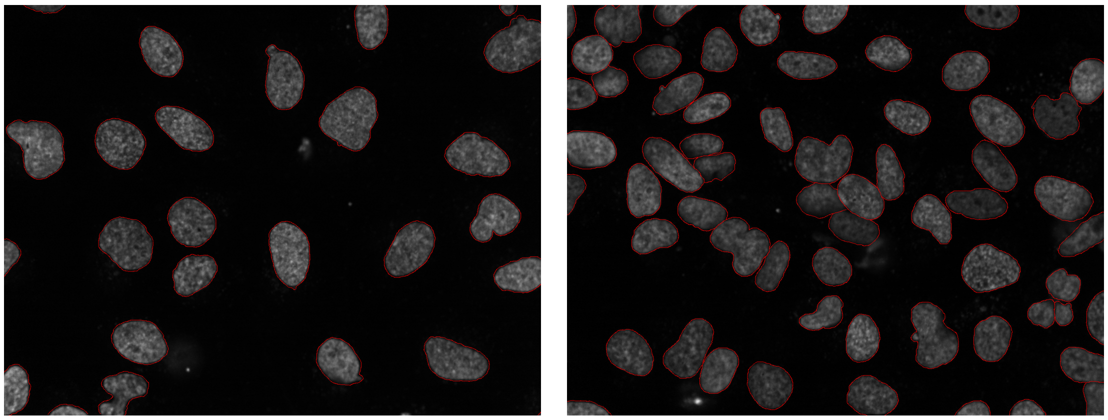

(a) "Easy" image (b) "Difficult" image

Figure 3.1: Two example images from the U2OS collection. (a) shows nuclei that are well separated. Automatic segmentation is expected to do well. (b) has many clustered nuclei and is expected to challenge segmentation algorithms. Most images in the collection lie in between these two examples. Reproduced from (Coelho, Shariff, et al., 2009).

The second collection is of NIH3T3 cells, from the RandTag project (see Section 1.4). Nuclei in this group are further apart and there is less clustering. They are also more homogeneous in shape and size (data not shown). On the other hand, nuclei in single images vary greatly in brightness and images often contain visible debris. Therefore, we consider this a more challenging dataset for automated methods. Fifty images were initially chosen, but one was rejected as containing no in-focus cells.

Manual segmentation was performed by outlining nuclei with a computer mouse. Only the nuclear marker image was used for this process. All images were segmented by me and a subset of 10 images (5 from each collection) were independently segmented by Aabid Shariff.

3.2 Segmentation Methods

Thresholding

We considered 3 thresholding methods: Ridler-Calvard (1978), Otsu (1979), and mean pixel value. All above-threshold contiguous regions are considered objects. To remove some noise, we filter the thresholded image with a median filter (window of size 5).

To remove small non-nuclear objects, we filter out objects smaller than 2500 pixels, circa 64 square microns. This post-filtering was applied to all segmentation results in this section.

Seeded watershed

We implemented two versions of seeded watershed, both run on a thresholded version of the image (using the mean as threshold, which, as we show below, is the better thresholding method for these images). One operates directly on a blurred version of the image,[1], while the second one operates on the gradient of the image. In both cases, seeds are regional maxima of the blurred image.

Active masks

Active masks are a recent proposal by Srinivasa, Fickus, Gonzalez-Rivero, et al. (2008); Srinivasa, Fickus, Guo, et al. (2009). The algorithm assumes that there are two classes of objects, foreground and background. Its only parameters are the mean value and standard deviation of the background region.[2]

Manual tuning led to the following semi-automatic procedure for parameter setting: the value of the background mean is assumed to be the histogram peak plus 3, while the background standard deviation is set to 0.5.

Merging Algorithm

G. Lin et al. (2003) described an algorithm that is based on merging multiple regions obtained from watershed segmentation, using shape information learned from a labeled dataset. We have implemented a slight variation of their algorithm, but retained the structure. In particular, we use the mean thresholding method for segmentation, and as shape features: fraction of area that is contained in the convex hull, roundness, eccentricity, area, perimeter, semi-major, and semi-minor axes (all, except the first, computed on the convex hull). Apart from these minor changes, the algorithm is unchanged.

For the studies below, the set segmented by Aabid Shariff was used for training and the set segmented by me (except the images that are common to both segmentations) were used for testing.

3.3 Evaluation Methods

Several metrics have been proposed for evaluation of segmentation results against a hand-labeled standard. Some approaches stem from viewing segmentation as a form of clustering of pixels. This allows the use of metrics devel-

[1]We used a Gaussian blur with a width of 12 pixels.

[2]The active mask framework is more general than this, but we restrict ourselves to the original proposal.

oped for the evaluation of clustering results. From this family of approaches, we used the Rand and Jaccard indices (Rand, 1971; Saporta and Youness, 2002).

The disadvantage of such metrics is that they do not take into account the spatial characteristics of segmentation. In fact, the exact location of the border between foreground and background is often fuzzy. An algorithm that returns a nucleus which almost matches the gold-standard except for a one-pixel-wide sliver around the border should be judged very highly even if that sliver contains a large number of pixels. Previous work on evaluation of bright-field microscopy images by Bamford (2003) used the Hausdorff metric, which is a worst-case metric. Here, we propose a new metric, *normalised sum of distances*, whose value depends on the mean distance rather then the worst-case.

The Rand and Jaccard Indices

Let S be a (binary) segmented image and R be a (binary) reference image. Let i and j range over all pairs of pixels where $i \neq j$, then each pair falls into one of four categories: (a) $R_i = R_j$ and $S_i = S_j$, (b) $R_i \neq R_j$ and $S_i = S_j$, (c) $R_i = R_j$ and $S_i \neq S_j$, (d) $R_i \neq R_j$ and $S_i \neq S_j$. If we let a, b, c, d refer to the number of pairs in its corresponding category, then the Rand index is defined as:

$$\mathrm{RI}(R, S) = \frac{a + d}{a + b + c + d}. \tag{3.1}$$

That is, the Rand index measures the fraction of the pairs where the two clusterings agree. The Rand index ranges from 0 to 1, with 1 corresponding to perfect agreement. Note that it not necessary that the number of clusters be the same for the metric to be meaningful.

Based on the same definitions for a, b, c, d, the Jaccard index is defined as:

$$\mathrm{JI}(R, S) = \frac{a + d}{b + c + d}. \tag{3.2}$$

The Jaccard index is not upper-bounded, but higher values correspond to better agreement.

Error Counting

Each object in the segmented image is assigned to the object in the reference image with which it shares the most pixels. Based on these assignments, we can define the following classes of errors: **split**: two segmented nuclei are assigned to a single reference nucleus; **merged**: two reference nuclei are assigned to a single segmented nucleus; **added**: a segmented nucleus is assigned to the reference background; and **missing**: a reference nucleus is assigned to the segmented background.

Spatially-Aware Evaluation Methods

We implemented two spatially-aware evaluation metrics. Both are based on assigning segmented nuclei to reference nuclei as above, as they are computed between pairs of matched objects.

Algorithm	RI	JI	Hausdorff	NSD ($\times 10$)
AS Manual	95%/93%	2.4/3.4	9.7/12.0	0.5/0.7
RC Threshold	92%/77%	2.2/2.1	34.8/26.4	1.2/2.6
Otsu Threshold	92%/74%	2.2/2.1	34.9/36.7	1.2/3.5
Mean Threshold	96%/82%	2.2/1.9	26.5/24.4	1.0/2.3
Watershed (direct)	91%/78%	1.9/1.6	34.9/19.3	3.6/3.7
Watershed (gradient)	90%/78%	1.8/1.6	34.6/21.7	3.0/3.8
Active Masks	87%/72%	2.1/2.0	148.3/98.0	5.5/5.0
Merging Algorithm	96%/83%	2.2/1.9	12.9/15.9	0.7/2.5

Algorithm	Split	Merged	Spurious	Missing
AS Manual	1.6/1.0	1.0/1.2	0.8/0.0	2.2/3.2
RC Threshold	1.1/1.0	2.4/2.4	0.3/1.9	5.5/22.1
Otsu Threshold	1.1/0.8	2.4/2.1	0.3/1.7	5.6/26.6
Mean Threshold	1.3/1.4	3.4/5.1	0.9/3.1	3.6/4.8
Watershed (direct)	13.8/2.9	1.2/2.4	2.0/11.6	3.0/5.5
Watershed (gradient)	7.7/2.6	2.0/3.0	2.0/11.4	2.9/5.4
Active Masks	10.5/1.9	2.1/1.5	0.4/3.9	10.8/31.1
Merging Algorithm	1.8/1.6	2.1/3.0	1.0/6.8	3.3/5.9

Table 3.2: Comparison of Segmentation Algorithms. Result of various segmentation approaches are compared against the hand-segmented standard. Each entry contains two values corresponding to the statistic for two datasets used, U2OS and NIH3T3, respectively. The top half shows the numeric quality measures, while the bottom half shows the average number (per image) of each possible type of error.

For each pixel, we compute its distance to the reference border. The normalised sum of distances is then defined as:

$$\text{NSD}(R, S) = \frac{\sum_i [\![R_i \neq S_i]\!] * D(i)}{\sum_i D(i)}, \tag{3.3}$$

where the sum index i ranges over pixels in the union of both objects and $D(i)$ is the distance of pixel i to the border of the reference object. From the equation, it is obvious that NSD$(R, S) \in [0, 1]$, with 0 corresponding to perfect agreement and 1 to no-overlap. We note that the sum of distances is *not* a metric as it is neither symmetric nor does it satisfy the triangle inequality.

The Hausdorff metric is computed as described by Bamford (2003). In the notation above it can be defined as:

$$H(R, S) = \max \{D(i) : S_i \neq R_i\}. \tag{3.4}$$

3.4 Results

Table 3.2 summarises the results obtained.

Both manual segmentations are in general agreement. Disagreements can be traced down to an image where the authors differed on whether some small bright objects should be marked as nuclei or debris.

Both Otsu and Ridler-Calvard thresholding score poorly, missing many cells, particularly in the NIH3T3 collection. In this collection, the presence of very bright cells leads the algorithm to set a threshold between the very bright cells and the rest of the cells, instead of setting it between the foreground and background. The mean thresholding is better suited for these images, which consist mainly of background with objects of very different intensities.

Watershed results in fewer merges than mean-based segmentation, but more split nuclei and spurious objects. Active masks score poorly mainly due to nuclei over-segmentation and missing objects. Lin et al.'s merging algorithm obtains very good results, dominating other algorithms in almost all metrics.

We also notice the Rand and Jaccard indices while distinguishing the alternative manual segmentation from the automatic ones are not good measures for this data as they fail to distinguish between the better and the worse algorithms. Both the Hausdorff and the NSD measures capture the relationships between the algorithms well.

Since this work was presented, C. Chen et al. (2011) presented another algorithm, based on supervised learning, whose results are competitive with the merging algorithm on this same dataset.

§4 Subcellular Pattern Unmixing

Most of the material in chapter section is joint work with Tao Peng and first appeared in Quantifying the distribution of probes between subcellular locations using unsupervised pattern unmixing *by Luis Pedro Coelho, Tao Peng, and Robert F. Murphy, in Bioinformatics, vol. 26 (12), pp. i7–i12, 2010 [DOI].*

A preliminary version had been presented at the ICML-UAI-COLT *2009 Workshop on Automated Interpretation and Modeling of Cell Images (Cell Image Learning) as* Unsupervised Unmixing of Subcellular Location Patterns*, by Luis Pedro Coelho and Robert F. Murphy.*

4.1 Introduction

In Section 2.1, I reviewed the principles behind subcellular pattern unmixing, in its supervised mode: given images of pure patterns, the goal is to represent images of mixed patterns as being composed of different fractions of the pure patterns.

However, the supervised approach still requires the researcher to specify the fundamental patterns of which other patterns are composed. For example, for the quantitative analysis of translocation experiments as a function of time or drug concentration, the extreme points could be easily identified as the patterns of interest. However, they are still inapplicable to proteome-wide studies where it would be a difficult (and perhaps impossible) task to identify all fundamental patterns that are present. We note that the set of fundamental patterns that can be identified depends both on the specific cell type and the technology used for imaging, high resolution confocal microscopes being able to distinguish patterns that lower resolution systems cannot.

Therefore, it is necessary to tackle the unsupervised pattern unmixing problem: Given a large collection of images, where none has been tagged as being a representative of a fundamental pattern, map all images into a set of mixture coefficients automatically derived from the data.

4.2 Object Typing

Overview

All the methods developed for this problem so far are based on a bag of objects model, where an image is interpreted as a collection of regions of above-background fluorescence (Zhao, Velliste, et al., 2005). Each object is then characterized by a small set of object features, and objects are clustered into groups (object types). Patterns are then defined as distributions over these groups. This is illustrated in Figure 4.1.

The intuition is to capture patterns such as the fact that lysosomes are small mostly circular objects, while mitochondria consist of stringy objects. The methods need to be robust to stochastic variation, however, as mitochondrial patterns are also observed to contain circular objects and agglomerations of lysosomes may appear as a single stringy object. In fact, the algorithms need to capture not only the fact that mitochondrial patterns are composed of stringy objects, but that the proportions of different types of objects are present in statistically different proportions.

Image Preprocessing and Segmentation

Images are first preprocessed to remove uneven illumination. The illumination bias is estimated by fitting a plane to the average pixel intensity at each location across the whole collection of images. Every image pixel is then divided by this illumination estimate to regularize across the whole image.

Images are segmented by using the model-based method of G. Lin et al. (2003) on the nuclear channel (see Section 3). The segmentation is extended to the whole field by using the watershed method with the segmented nuclei as seeds.

Object Detection

In our previous supervised unmixing work, objects were simply defined as contiguous pixel regions above a global threshold. In the work described here, we use both a global threshold, using the Ridler-Calvard method (1978), and a local threshold, the mean pixel value of a 15×15 window centered at the pixel. We have found that the global threshold achieves a good separation of the general cell areas from the background, while, inside those regions, local thresholding is better at capturing detail.

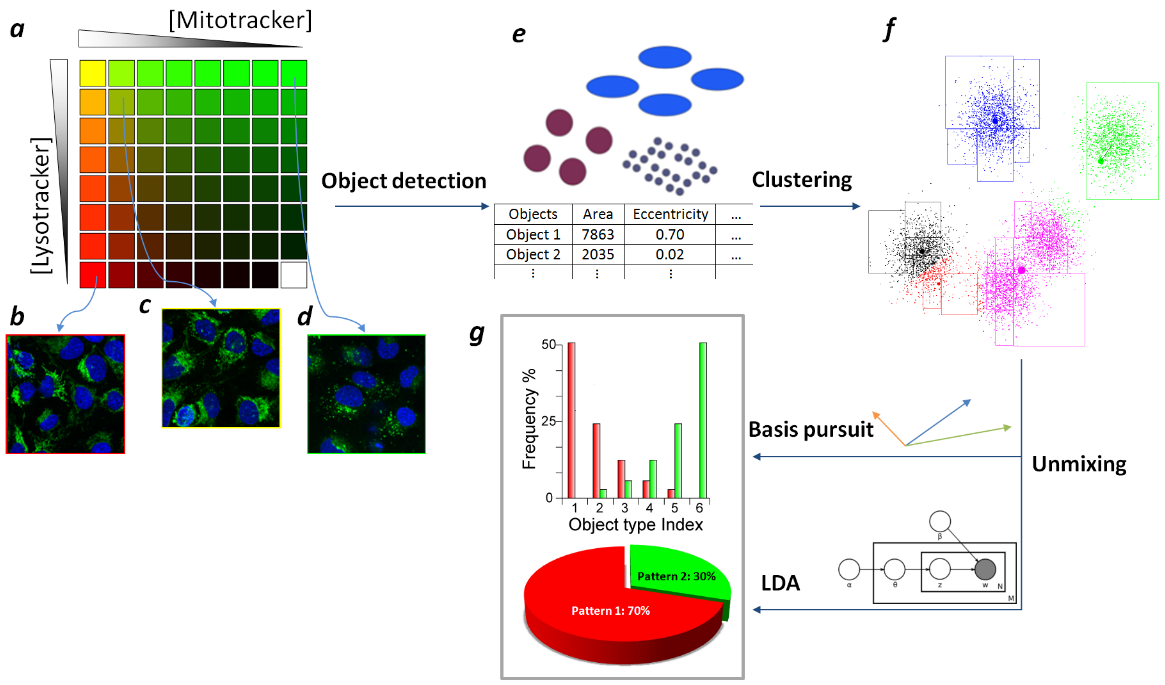


Figure 4.1: Overview of unmixing methods. (a) The algorithms use a collection of images as input in which various concentrations of two probes are present (the concentrations of the Mitotracker and Lysotracker probes are shown by increasing intensity of red and green, respectively). Example images are shown from wells containing only Mitotracker (b), only Lysotracker (c) and a mixture of the two probes (d). (e) Objects with different size and shapes are extracted and object features are calculated. (f) Objects are clustered into groups in feature space, shown with different colors. (g) Fundamental patterns are identified and the fractions they contribute to each image are estimated. Adapted from (Coelho, T. Peng, et al., 2010).

Objects that are smaller than 5 pixels are filtered out as they are likely to represent imaging noise.

Object Features

Each object is characterized by a set of features, previously defined as SOF1 (subcellular object features 1). This is a combination of morphological features for describing the shape and size of the object and features which capture the relationship to the nuclear marker (Zhao, Velliste, et al., 2005):

1. Size (in pixel units) of the object.
2. Distance of object center of fluorescence to DNA center of fluorescence.
3. Fraction of object that overlaps with DNA.
4. Eccentricity of object hull.

5. Euler number of object.
6. Shape factor of convex hull.
7. Size of object skeleton.
8. Fraction of overlap between object convex hull and object.
9. Fraction of binary object that is skeleton.
10. Fraction of fluorescence contained in skeleton.
11. Fraction of binary object that constitutes branch points in the skeleton.

Object Clustering

In order to be able to reason about object types, objects are clustered into groups using k-means on the z-scored feature space. Multiple values of k are tried and the one with the resulting lowest BIC (Bayesian information criterion) score is selected.

Based on this clustering, each object can be assigned a numerical identifier, its cluster index, which serves as its type.

After this step, the algorithms diverge in how they handle the cluster indices.

4.3 Basis Pursuit

In this model, each image is represented by a vector $\boldsymbol{x}^{(i)}$ such that entry $\boldsymbol{x}_\ell^{(i)}$ represents the fraction of objects in condition i that have type ℓ (if there are multiple images for the same condition, a common situation, they are counted together). We have one vector per input condition (i.e., $i = 1 \ldots C$, where C is the number of conditions), and the dimension of this vector is the number of clusters that was automatically identified in the clustering step (i.e., $\ell = 1 \ldots k$).

Using fractions instead of the direct object counts normalizes for the different number of cells in each image and different cell sizes.

In this model, bases (fundamental patterns) are represented as a set of vectors in the same space and a mixture is defined by a set of coefficients α_j for each $\boldsymbol{b}^{(j)}$ ($j = 1 \ldots B$, where B is the number of basis vectors, and each $\boldsymbol{b}^{(j)}$ is of the same dimension as the $\boldsymbol{x}^{(i)}$s):

$$\boldsymbol{x}^{(i)} = \sum_j \boldsymbol{b}^{(j)} \alpha_j^{(i)} + \boldsymbol{\varepsilon}^{(i)}, \tag{4.1}$$

where $\boldsymbol{\varepsilon}^{(i)}$ encapsulates both the stochastic nature of the mixing process and the measurement noise.

Given a set of observations, the task is to identify the bases $\boldsymbol{b}^{(j)}$ and coefficients $\boldsymbol{\alpha}^{(i)}$, which minimize the squared norm of the error terms $\sum_i \|\varepsilon^{(i)}\|^2$.

Without additional constraints, principal component analysis is the simplest solution to this problem. However, this is unsatisfactory as it could result in negative mixtures, which are not meaningful. Independent component analysis suffers from the same problem. Therefore, we add a non-negativity constraint on the vector $\boldsymbol{\alpha}$ and use non-negative matrix factorization (NNMF) possibly with sparsity constraints so solve the problem (Hoyer, 2004; D. D. Lee and H. S. Seung, 1999; D. D. Lee and H. Seung, 2001).

As the result below show, an additional constraint can help obtain more meaningful results: require the basis vectors to be members of the input dataset (i.e., for all j, there is some i, such that $\boldsymbol{b}^{(j)} = \boldsymbol{x}^{(i)}$). This condition, which encapsulates the expectation that the input dataset is large enough to contain both fundamental and mixed patterns, requires a search method.

Some preliminary results showed that this model was still too sensitive to the trend, i.e., to the average value of $x_{i,j}$ across the dataset (data not shown). If one basis vector was allocated to handle this trend, good fits were obtained but poor interpretability. We found that removing the mean from the data led to more meaningful results. In this de-trended dataset, $\hat{x}_j^{(i)}$ may take negative values, but the mixing coefficients $\alpha_{i,j}$ are still constrained to be non-negative.

Thus the final optimization problem is the:

$$\min_{\boldsymbol{b}^{(j)},\boldsymbol{\alpha}} \|\boldsymbol{\varepsilon}^{(i)}\|^2 \tag{4.2}$$

$$\hat{\boldsymbol{x}}^{(i)} = \boldsymbol{x}^{(i)} - \bar{\boldsymbol{x}} \tag{4.3}$$

$$\boldsymbol{\varepsilon}^{(i)} = \hat{\boldsymbol{x}}^{(i)} - \sum_j \boldsymbol{b}^{(j)} \alpha_j^{(i)} \tag{4.4}$$

Subject to the constraint, that for all j, there exists an i, such that $\boldsymbol{b}^{(j)} = \boldsymbol{x}^{(i)}$. In order to find the best basis, we resort to simulated annealing as an optimization method. In this class of methods, the number of fundamental patterns B must be pre-specified by the user.

Principal and independent component analysis were also performed on detrended data, but NNMF could not be (as the detrended data contains negative numbers, it cannot be the product of two positive matrices). Before applying NNMF, we therefore removed very frequent objects (those that appeared in more than 90% of the images). The intuition is that very frequent objects also correspond to the background.

Latent Dirichlet Allocation

As described in more detail in Section 2.5, topic modeling of text corpura using latent Dirichlet allocation (LDA) is a technique to solve an analogous class of problems (Blei, Ng, et al., 2003). In this framework, documents are seen as simple “bags of words” and topics are distributions over words. Observed bags of words can be generated by choosing mixture coefficients for topics followed by a generation of words according to: pick a topic from which to generate, then pick a word from that topic.

In our setting, we view object classes as visual words over which to run LDA. This is similar to work by other researchers in computer vision which use keypoints to define visual words (Csurka et al., 2004; Philbin et al., 2008; L. Zhu et al., 2009).

The process of generating objects in images to represent mixtures of multiple fundamental patterns follows the Bayesian network in Figure 2.2 (page 19). The generative process is as follows: for each of M images, a mixture $\boldsymbol{\theta}_i$ is first sampled (conditioned on the hyper-parameter α). $\boldsymbol{\theta}_i$ is a vector of fractions of the fundamental pattern distributions $\boldsymbol{b}$. N_i objects are sampled for each image in two steps: select a basis pattern according to $\boldsymbol{\theta}_i$ and then an object is sampled from the corresponding object type distribution.

To invert this generative process, we used the *variational EM* algorithm of Blei, Ng, et al. (2003) to estimate the model parameters of fundamental patterns $\boldsymbol{\beta}$ and mixture fractions $\boldsymbol{\theta}$. It should be noted that this is an approximation approach liable to getting trapped in local maxima and returning non-optimal results. Therefore, we ran the algorithm multiple times with different random initializations and chose the one with the highest log-likelihood.

We choose the number of fundamental patterns B to maximize the log-likelihood on a held-out dataset (using cross-validation to obtain more accurate estimate).

Dataset

In order to validate the algorithms, we used a test set that was built to evaluate pattern unmixing algorithms (T. Peng, Bonamy, et al., 2010).

In this dataset, U2OS cells were exposed to different concentrations of two fluorescent probes with differing localization profiles (mitochondrial and lysosomal) but similar fluorescence. The probes were imaged using the same fluorescence filter and therefore could not be distinguished. This simulates the situation in which a fluorophore is present in two different locations. For each probe, eight concentrations were used, for a total of 64 combinations.

In parallel to the marker image, a nuclear marker was imaged to serve as a reference point. A total of 12569 images were collected. Figure 4.2 shows images from this collection.

Computation Time

Most of the computation time is dominated by segmenting the images (ca. 30s per image in our implementation) and computing features (ca. 10s per image). However, this is an embarrassingly parallel problem and can be computed on multiple machines simultaneously. The clustering takes increasing time for different numbers of clusters, but we limited each clustering run to circa one hour (while relying on multiple initialization as a guard against local

	Mitochondrial	Lysosomal
Pattern 0	**99%**	18%
Pattern 1	1%	**82%**

Table 4.1: Unmixed coefficients for images of fundamental patterns and of mixed samples using basis pursuit with $B = 2$. For the two fundamental patterns, we display the average coefficient for the inferred fundamental patterns.

minima). Again, we note that the runs for multiple k can be easily be run in parallel, a task for which the jug framework, described in Appendix A.2, is particularly well suited. Both basis pursuit and LDA then take only on the order of minutes to run.

Basis Pursuit

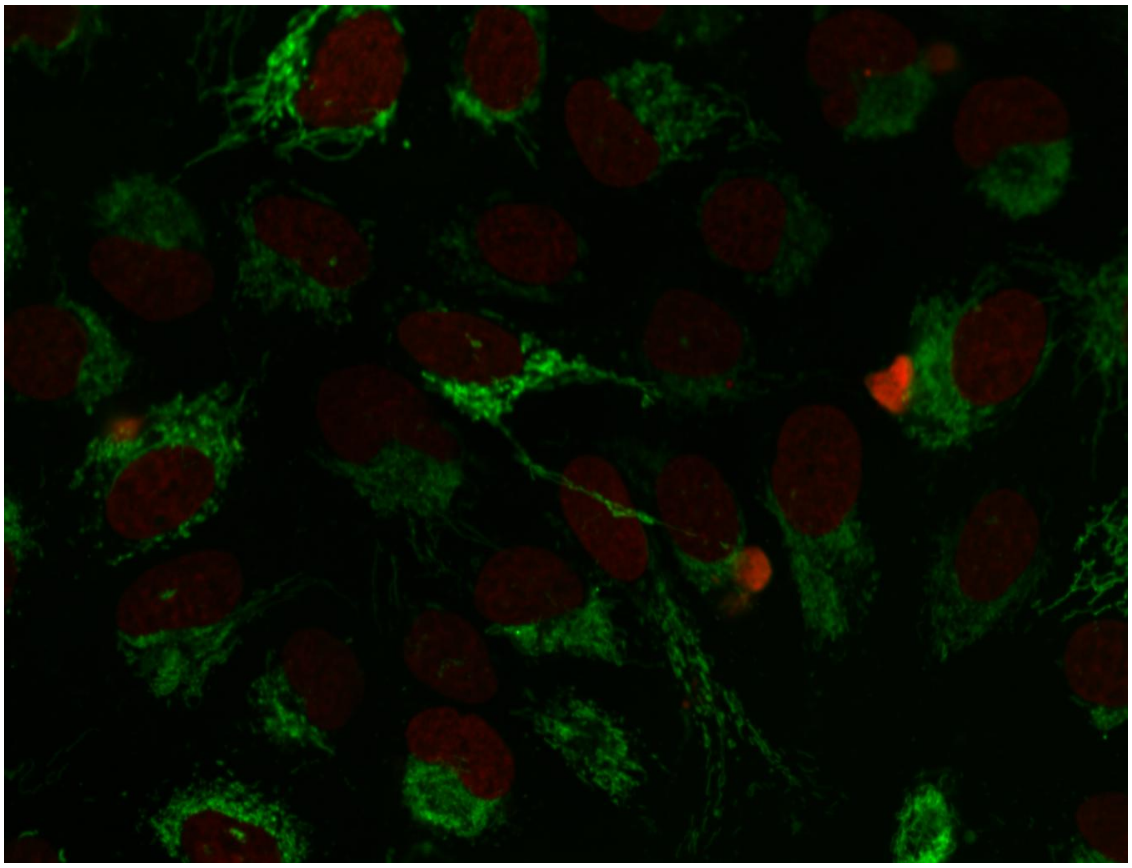
We measured how well the identified coefficients $\alpha_j^{(i)}$ correlated with the underlying fractions, which were estimated as linearly proportional to the ratio of the relative concentration of the mitochondrial probe to the sum of the relative concentration of the mitochondrial and lysosomal probes (relative concentration is defined as fraction of the maximum subsaturating concentration).

Using PCA, the correlation coefficient between predicted fractions and the underlying relative concentrations was 0.20. Non-negative matrix factorization performed better on this metric, achieving a correlation coefficient of 0.65. Independent component analysis performed very poorly, returning correlations on the order of less than 0.10. This is not unexpected as the independence assumptions that underly ICA fail to hold even as an approximation.

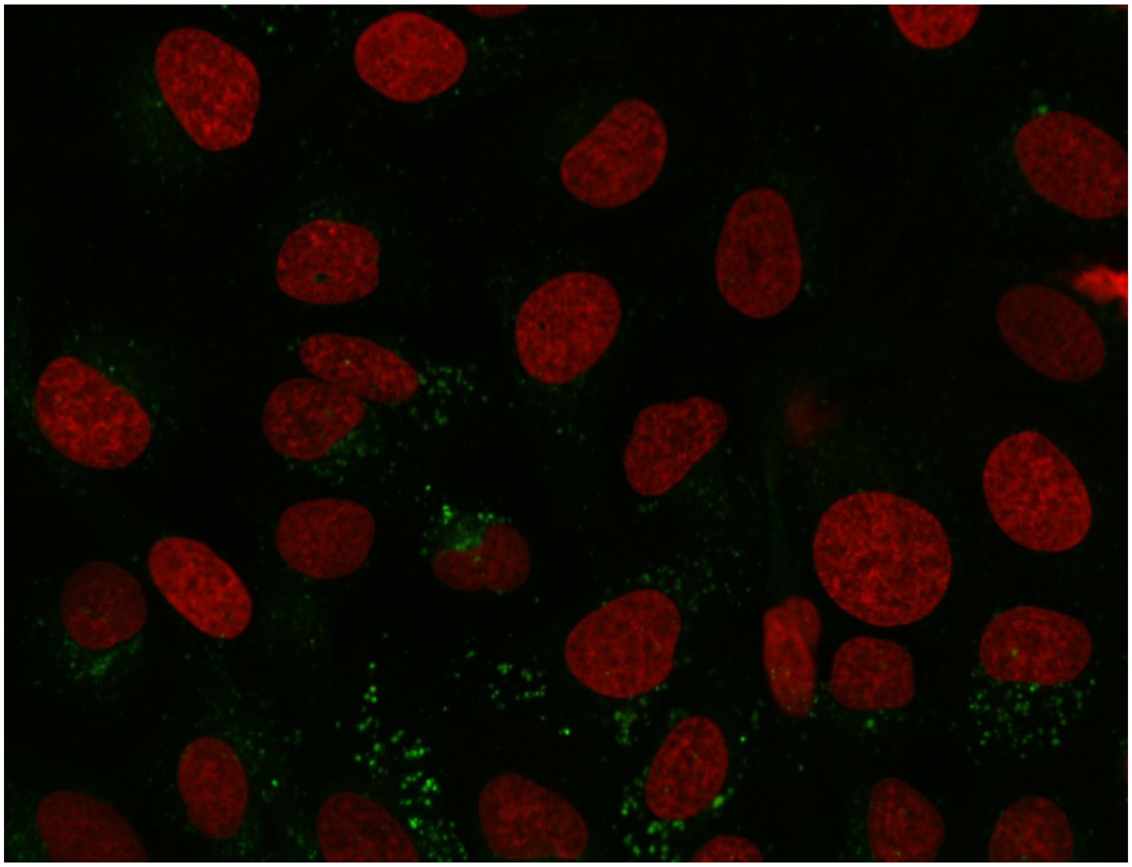
However, we are also interested in having the basis vectors line up with the underlying fundamental patterns and, in this regard, non-negative matrix factorization performs poorly. One of the patterns corresponded roughly to the total concentration and they did not align well with the fundamental patterns in the data (data not shown).

The fully constrained basis pursuit algorithm performed better. It achieved a 0.90 correlation with the underlying relative concentration. It identified as a basis a vector that has the maximal concentration of the mitochondrial probe (and some lysosomal probe, at a relative concentration of 19%) and another that consists of the maximal concentration of the lysosomal probe and 20% mitochondrial probe. Table 4.1 shows that the identified pattern 0 matches the mitochondrial probe, while pattern 1 matches the lysosomal probe. Figure 4.2 shows two images from these patterns.

The results above were obtained by specifying $B = 2$ as an input to the algorithm. For different values of B, we obtain decreasing reconstruction error as plotted in Figure 4.3. As it is clear in this figure, most of the contribution to the reconstruction comes from the first two or three vectors. Therefore, we can expect that a researcher would be able to estimate $B = 2$ or $B = 3$.



(a)



(b)

Figure 4.2: Example of Basis Found With Basis Pursuit. These are the first images of the basis set, using basis pursuit

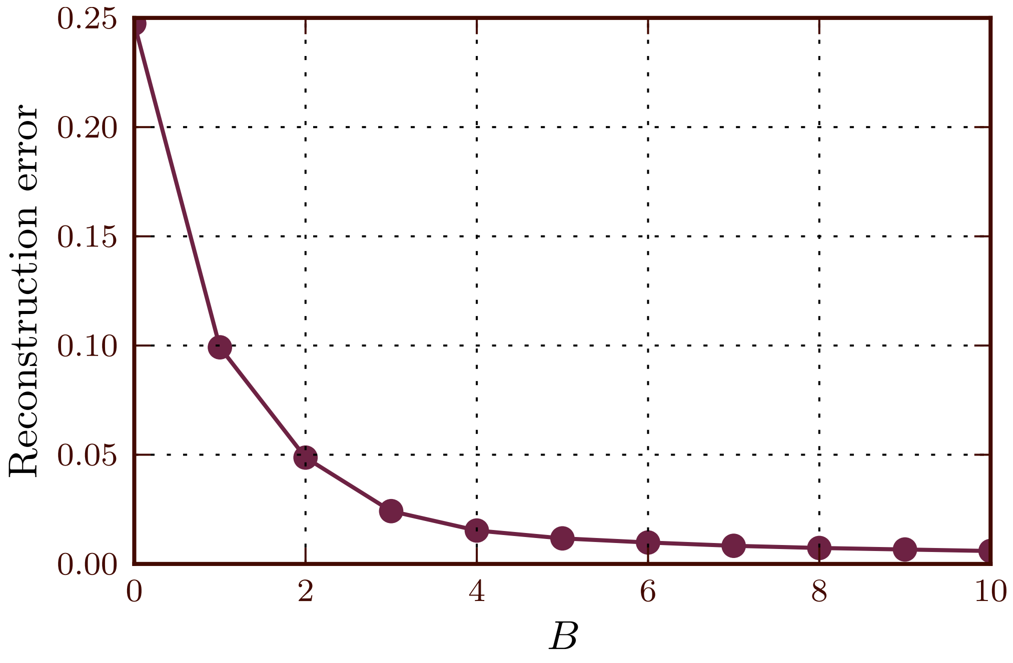


Figure 4.3: Average squared reconstruction error as a function of the number of patterns B for basis pursuit. This is the value of $\sum_i \|\varepsilon\|^2$ in (4.2). For $B = 0$, we show the total variance, i.e. $\sum_i \|\hat{\boldsymbol{x}}^{(i)}\|^2$. Adapted from (Coelho, T. Peng, et al., 2010)

	Mitochondrial	Lysosomal
Pattern 0	0.0%	0.0%
Pattern 1	8.8%	**99.9%**
Pattern 2	**91.2%**	0.1%

Table 4.2: Unmixed coefficients for fundamental patterns and mixed samples for the discovered patterns (using LDA method). For the two fundamental patterns, we display the average coefficient for the 3 discovered fundamental patterns.

Latent Dirichlet Allocation

To estimate the number of fundamental patterns using the LDA approach, we measured the log likelihood of the dataset for different numbers of bases using cross validation. The results are shown in Figure 4.4. We can see that the best result is obtained for $B = 3$, although the underlying dataset only has two fundamental patterns.

Table 4.2 shows the average coefficients inferred for pure pattern inputs after the algorithm had been applied on the whole dataset. Pattern 1 obviously corresponds to the lysosomal component, while pattern 2 corresponds to the mitochondrial component. Pattern 0 appears to be a "non-significant" pattern capturing the new object types arising in the mixture patterns. The overall correlation coefficient is 0.95 with pattern 0 removed.

Using the LDA approach with $B = 2$, which is the ground truth, the overall correlation coefficient between estimated and actual pattern fractions was found to be 0.91.

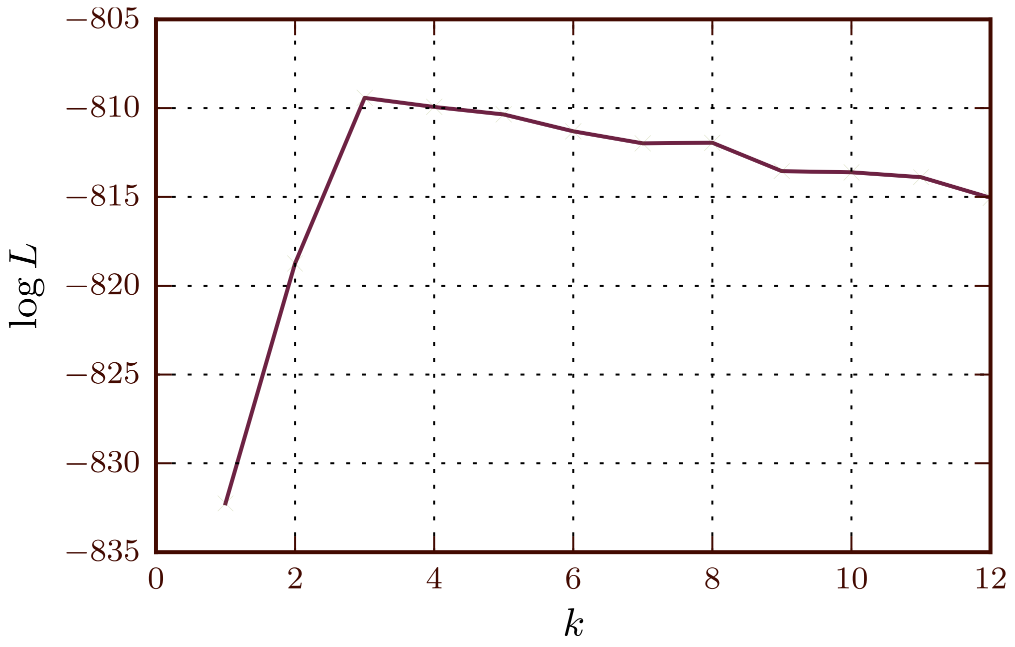
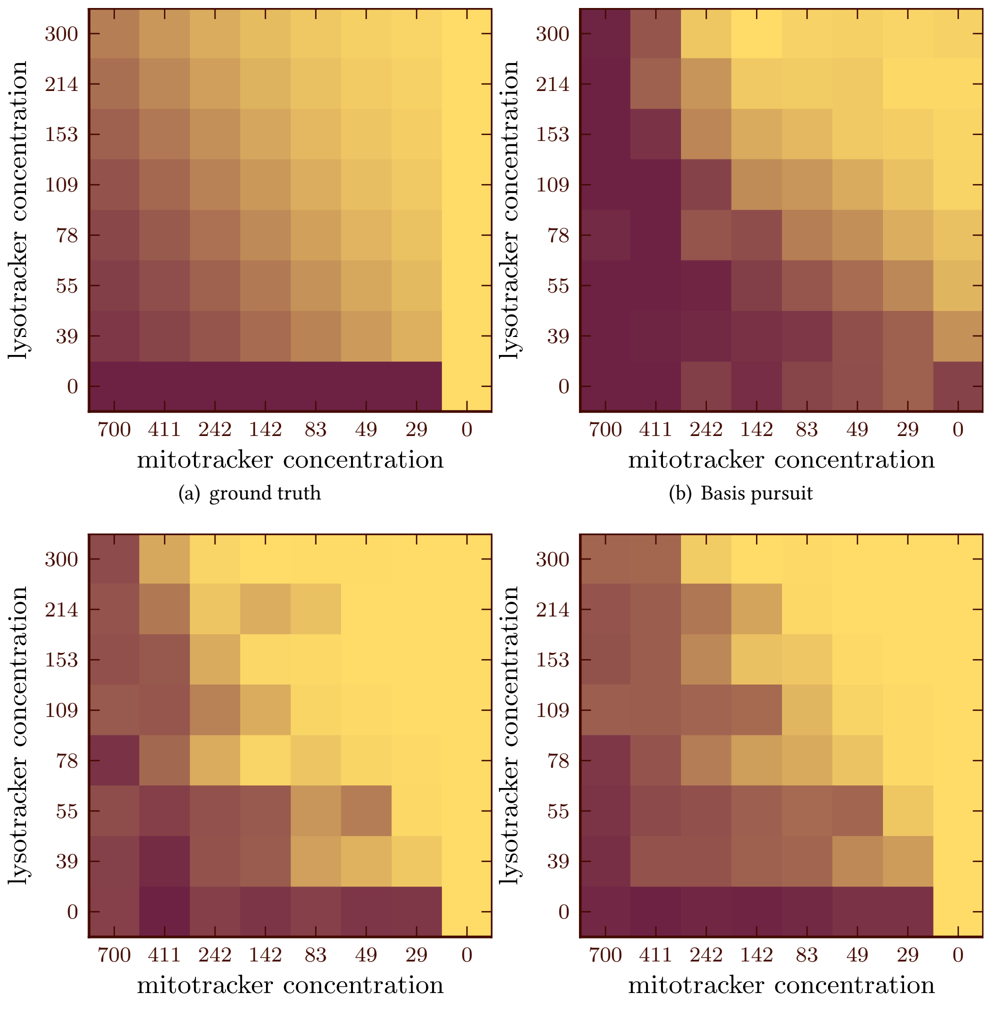


Figure 4.4: Log-likelihood as a function of the number of fundamental patterns. Adapted from (Coelho, T. Peng, et al., 2010).

Comparisons

Figure 4.5 shows the results of one inferred fraction as a function of the underlying concentrations (the plots for the other fraction, not shown, are, of course, symmetric as they sum to 1). Figure 4.6 plots all the estimates in a single plot as a function of the underlying concentration fractions.



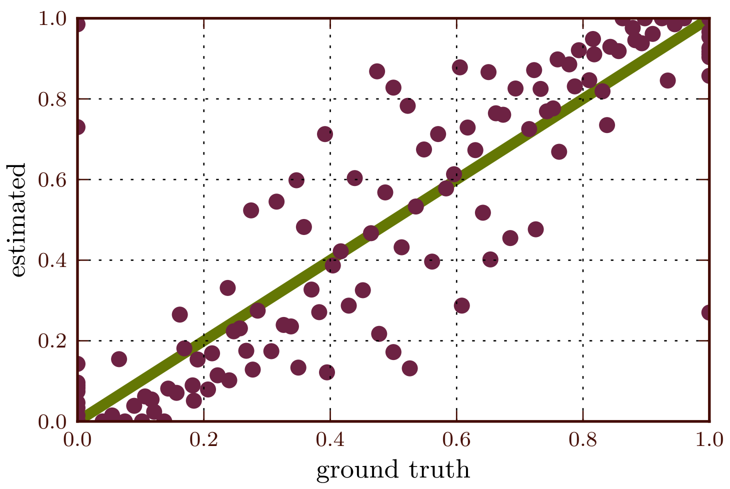
(a) ground truth

(b) Basis pursuit

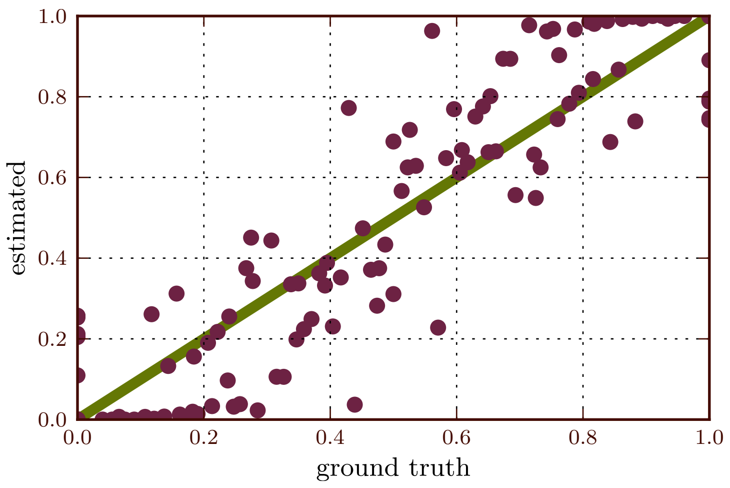
(c) LDA, 2 topics

(d) LDA, 3 topics

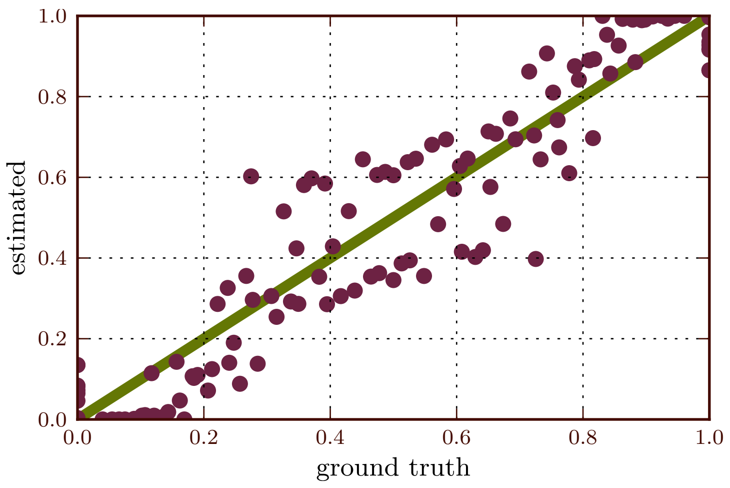
Figure 4.5: Comparison of results for different unmixing methods. The inferred fraction of pattern 1 is displayed as different colors (dark purple corresponding to pure pattern 1). The design matrix, which was kept hidden from the algorithms is shown on the top left, for comparison; the other three panels are results of computation for different algorithms. Adapted from Coelho, T. Peng, et al. (2010).



(a) Basis Pursuit



(b) LDA, 2 topics



(c) LDA, 3 topics

Figure 4.6: Estimated concentration as a function of the underlying relative probe concentration. Perfect result would be along the dashed diagonal. Adapted from Coelho, T. Peng, et al. (2010).

§5 Local Features

5.1 Introduction

As described in Section 2.1, the use of local features has led to many good results in various computer vision tasks. However, most of bioimage analysis is still performed using global features, such as texture features. Therefore, I wanted to investigate the performance of local features on this task.

Therefore, I adapted Speeded-Up Robust Features, SURF, developed by Bay, Ess, et al. (2008). They are designed to be fast to run (in my implementation, it takes only a few seconds per image) while having state of the art performance. Given the large collection of images, the low computational cost was a significant advantage.

5.2 SURF and Extensions

As detailed in Section 2.1, SURF works as a two pass algorithm. In the first pass, interest points are detected by using an approximate Gaussian blob detector. These interest points are localised in both space (i.e., at a specific pixel location), but also in scale (i.e., they have an automatically determined size).

SURF works on a single channel (a grey-scale image), while our images are multi-channel: in addition to the protein channel, we have a nuclear reference. Therefore, the traditional SURF can only be applied to the protein channel. I created SURF variants which incorporate the reference channel information.

The first variant I tried were based on the following idea: run the *point detection on the protein channel* and compute feature descriptors *on both channels*. The feature descriptor for each point is then the concatenation of both descriptors.

Another variation I implemented consisted of first thresholding the DNA channel, then computing its distance map, and, finally, computing descriptors on the distance map. As above, the final descriptor for a point was the concatenation of the descriptor computed on the protein image and the descriptor computed on the distance map.

For using all of these descriptors, we clustered the descriptor values (using k-means clustering, choosing the number of clusters automatically by AIC). This process assigns each descriptor to a cluster index. We represent an image as a normalized histogram of membership in the various clusters Willamowski et al., 2004. This is known as the "bag of visual words" model.

An alternative way to add information about the relationship of the protein to the reference channel is to simply add the field-level features. After clustering and summarisation of the image by a histogram, we concatenate the field-level features to this histogram to form the final feature vector. This is a simple way to use the information in the reference channel.

5.3 Empirical Evaluation

Although the overall goals of this work go beyond classification (namely unmixing and topic modeling as described in Chapters 4 and 6), it is a task on which evaluation of the features is straight-forward. I will present results in four different datasets. The first dataset will be explored in more detail while only single values will be reported for the other datasets.

For the measurement of statistical significance, I employed a Bayesian approach. If I assume that each algorithm has an underlying accuracy r, the question is then to ask what is the probability that one algorithm has a higher accuracy than another, given the observed data. Therefore, given two observed accuracies c_0 and c_1 on a dataset of size n, I compute

$$P(r_0 > r_1 | c_0, c_1, n) = \frac{\int_0^1 \int_0^1 [\![r_0 > r_1]\!] p(c_0, n|r_0) p(c_1, n|r_1) dr_0 dr_1}{\int_0^1 \int_0^1 p(c_0, n|r_0) p(c_1, n|r_1) dr_0 dr_1}. \tag{5.1}$$

In this method, I assume that the performance of the algorithms is independent,

$$p(c_0, c_1, n | r_0, r_1) = p(c_0, n|r_0) p(c_1, n|r_1). \tag{5.2}$$

And I will further assume that the individual probabilities can be accurately described by a binomial distribution:

$$p(c, n|r) = r^c (1-r)^{n-c}. \tag{5.3}$$

In this framework, higher values are better, which is the opposite of the traditional statistical practice. Therefore, I will report $1 - P(r_0 > r_1 | c_0, c_1, n)$ as a significance value. If the assumptions (5.2) and (5.3) are accepted, this

ER	G	UL	NO	N	Mito	Cyto	PM	Lyso	Cytosk
3	3	12	5	14	9	10	3	4	9
50	63	254	113	255	200	155	51	69	197

Table 5.1: Properties of Labeled Dataset. Ten classes. Shown are number of wells (first line) and images (second line) per class. Legend: ER: endoplasmic reticulum, G: Golgi, UL: unlabeled, NO: nucleoli, N: nuclear, Mito: mitochondria, Cyto: cytoplasmic, PM: plasma membrane, Lyso: lysosome, Cytosk: cytoskeleton.

significance value is then directly interpretable as the probability of making a Type I error (i.e., falsily rejecting the null hypothesis that $r_0 \leq r_1$).

RandTag Widefield Images

The first dataset consists of wide-field images obtained as described in Section 1.4. The images were labeled by three experts[1] using a protocol where the experts first labeled the images independently and were then given an opportunity to change their minds given the other labelings. Only images where all experts agreed after this second step were retained. There were 51 proteins in the final dataset (4 proteins were rejected *without any agreement*, and for 15 proteins only two of the experts agreed).

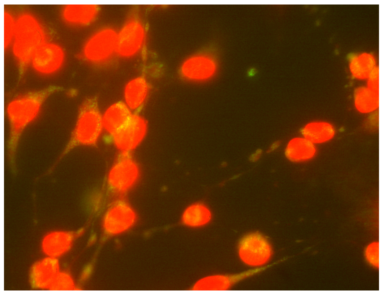
The basic metrics of the dataset are listed in table 5.1. Using global, field-level, features on this dataset achieves **61% accuracy** (all reported accuracies are estimated using cross-validation, which is performed over clones—images from the same clone never appear in both the training and testing sets of any fold). I extended this dataset by labeled 3 additional classes: ER, Golgi, and *unlabeled*.

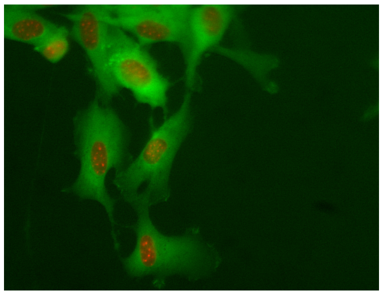
The clustering of the local descriptors is done automatically using k-means, which takes two parameters: k, the number of clusters; and an initial set of centroids (implemented simply by setting the random number generator seed to different values and randomly selecting elements the input). In order to choose the overall best result, minimising Akaike Information Criterion is often used.

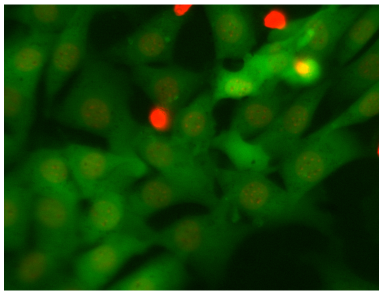
Interestingly, the accuracy obtained in classification was observed to be highly dependent on the clustering that was used in a way that is not very consistent. K-means clustering depends on both k and the random initialisation. As Figure 5.2 shows, there is a large variation in accuracy for different choices of the random seed even for the same value of k. Furthermore, as Figure 5.3 shows, there is no relationship between the value of the Akaike information criterion, which is often minimised to choose a set of clusters, and the resulting classification accuracy.

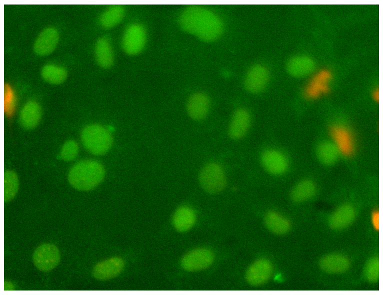
Jinjun Wang et al. (2010) proposed *locally constrained linear coding* as an alternative representation for a feature given a code-book (set of centroids). Instead of assigning a point to its nearest centroid, it is fractionally assigned to multiple close-by points. I implemented the simple, slow, version of the method. Unfortunately, the results (shown in Figure 5.4) were worse than those obtained from hard assignment. A weakness with that method is that there is a

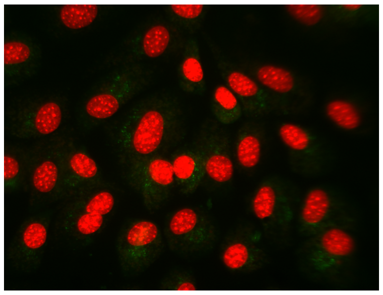
[1]The experts were myself, Dr. Elvira Osuna-Highley, and Dr. Estelle Glory-Afshar.

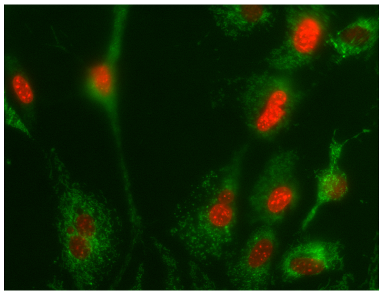

(a) ER

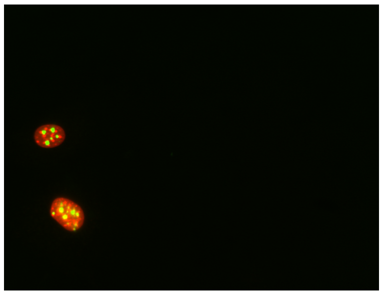

(b) Cytoplasmic

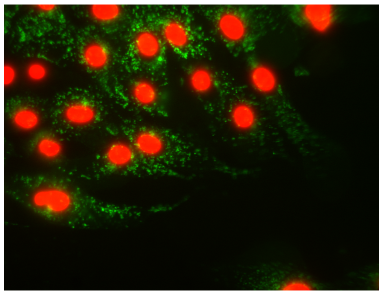

(c) P. Membrane

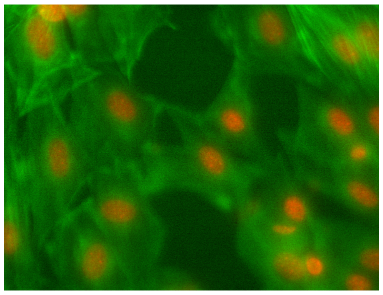

(d) Nuclear


(e) Golgi


(f) Mitochondria


(g) Nucleoli


(h) Lysosome


(i) Cytoskeleton

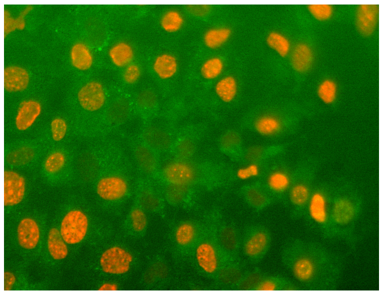

(j) Unlabeled

Figure 5.1: Examples of RandTag Widefield Dataset. This shows an image from each of the 10 classes in the extended widefield dataset. Images have been contrast stretched for publication.

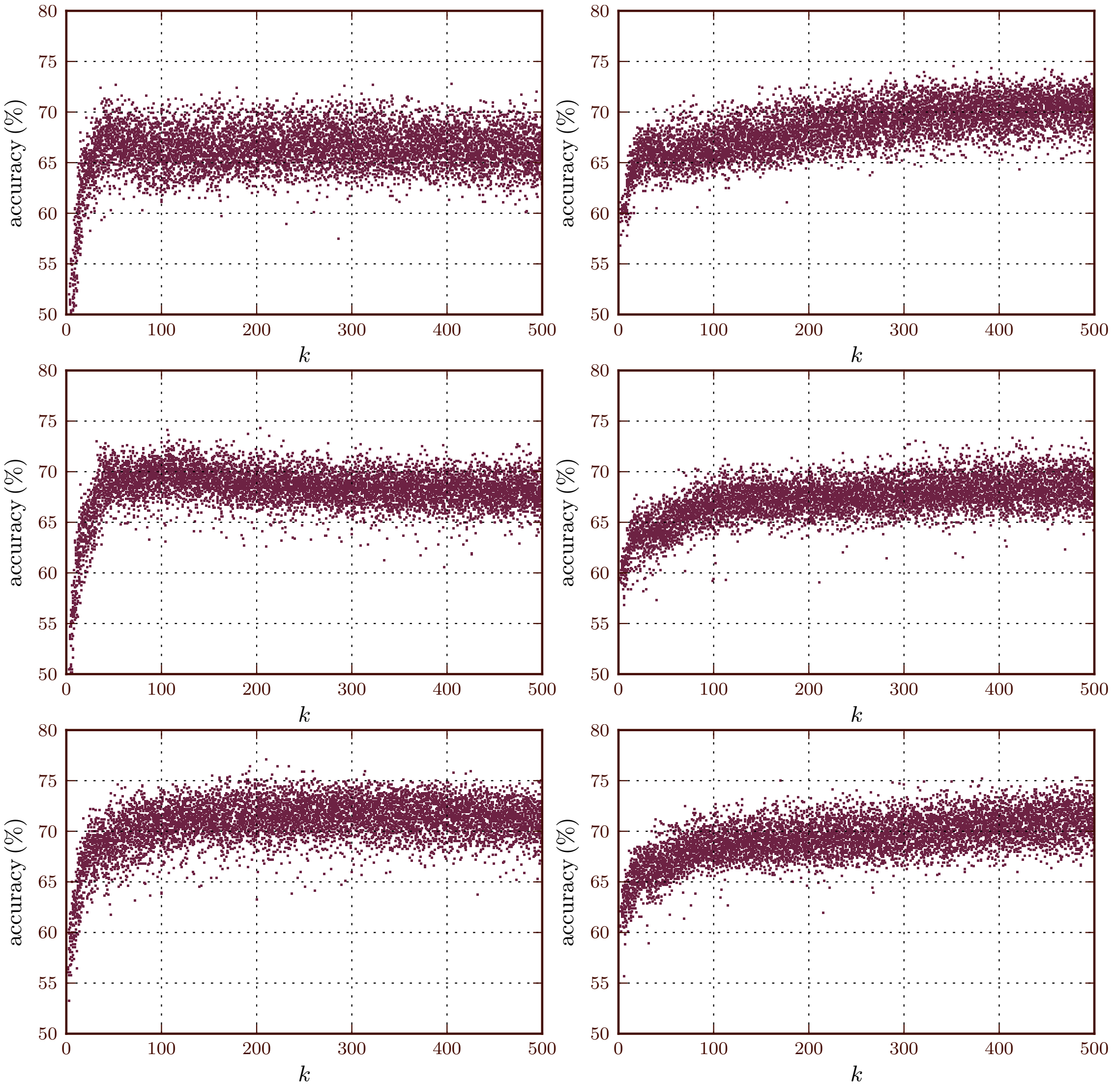


Figure 5.2: Results of Classification As a Function of the Number of clusters k. Each dot is the result of one clustering of the data (differing by a different number of clusters and a different initial set of clusters). Left panels show the results of using only the local features, right panels the results of using local features concatenated with global features. Different rows show different local feature sets. From top to bottom: SURF, SURF-ref, and SURF-dist.

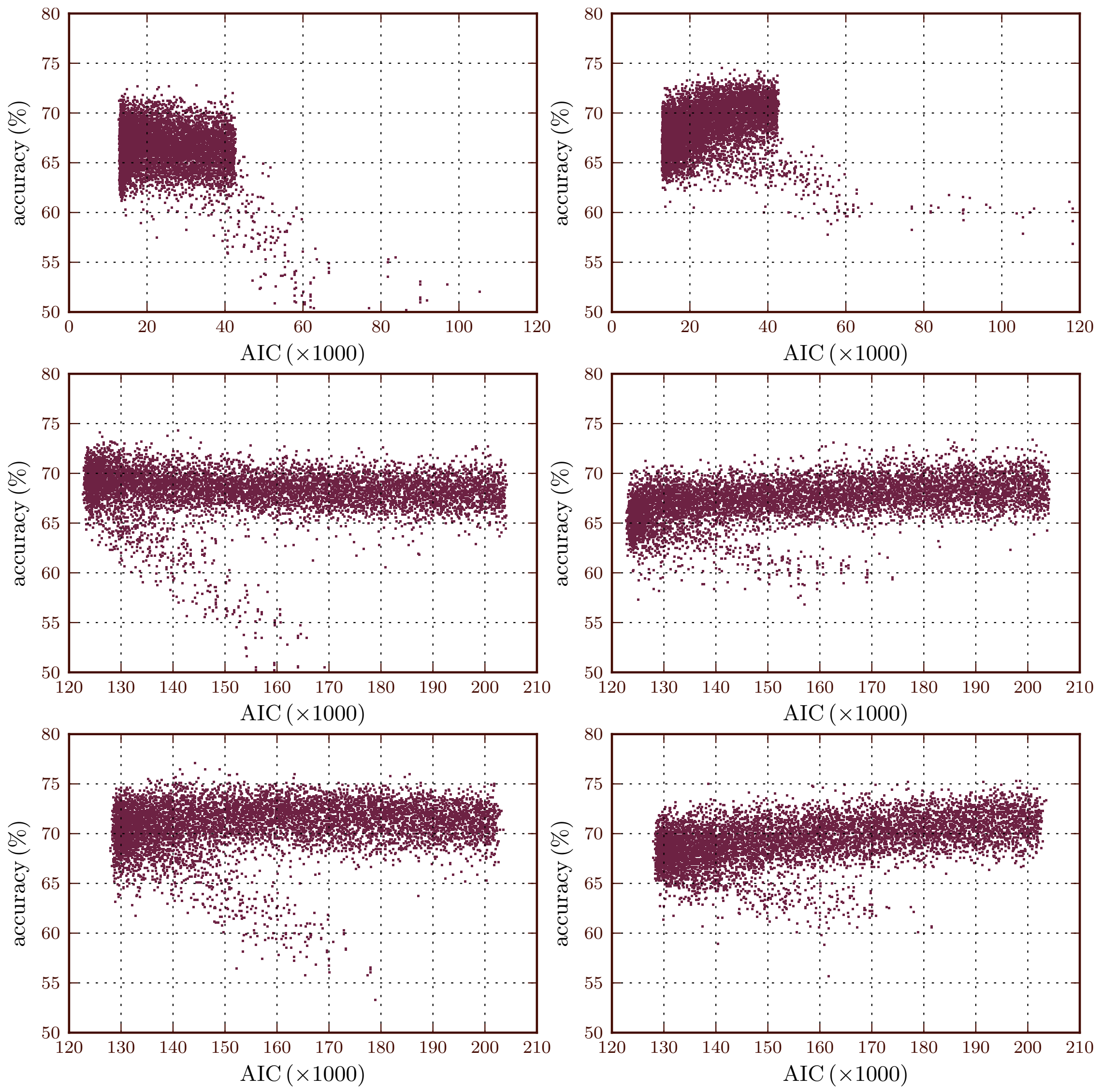


Figure 5.3: Results of Classification As a Function of the Value of the Akaike Information Criterion (AIC). Each dot is the result of one clustering of the data (differing by a different number of clusters and a different initial set of clusters). Left panels show the results of using only the local features, right panels the results of using local features concatenated with global features. Different rows show different local feature sets. From top to bottom: SURF, SURF-ref, and SURF-dist.

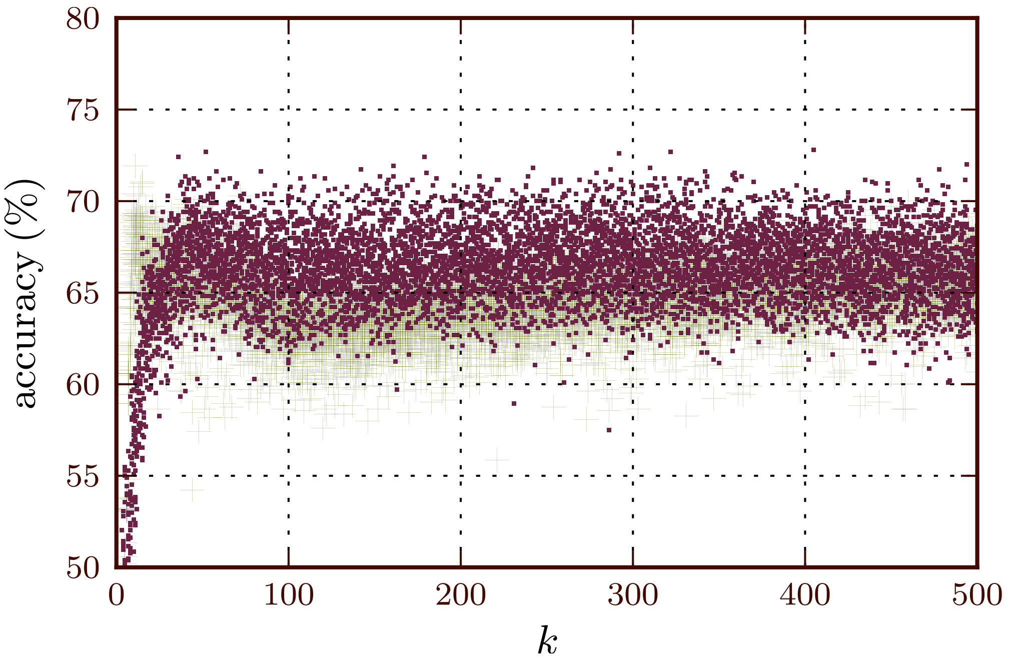


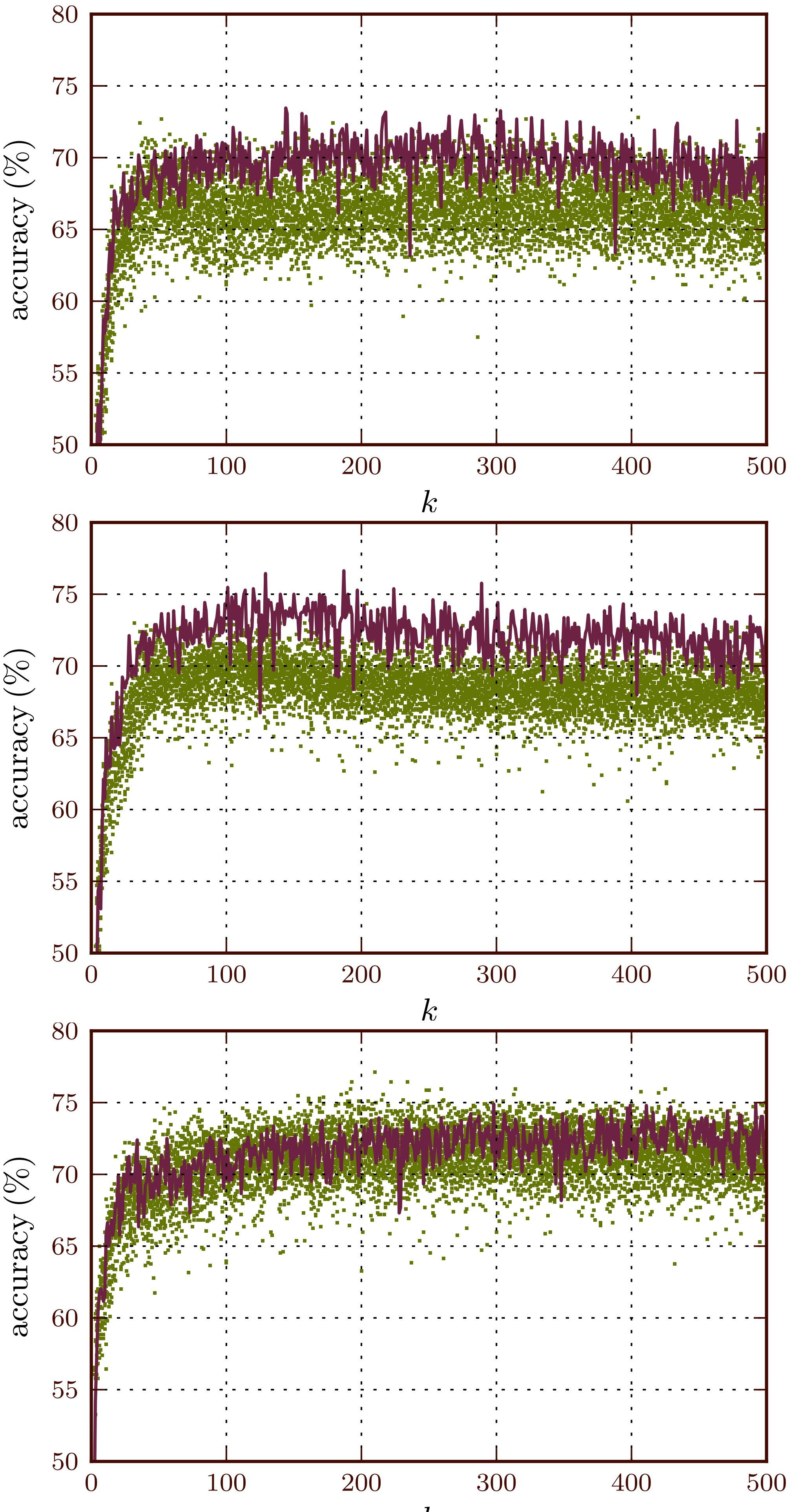
Figure 5.4: Comparison With Locally Constrained Linear Coding. Results of classification with SURF features after clustering, for different values of k. The green crosses represent the results with LLC, while the purple dots, the results with hard assignment (the dots are the same as presented in Figure 5.2).

free parameter, λ, which, for lack of guidance in the original paper, I set to 0.5. Given that my implementation was very slow and the initial results were not encouraging, I did not experiment with other values.

An alternative approach I implemented was to *learn the codebook*. Under the assumption that there are better and worse codebooks (the alternative hypothesis is that the variation observed in Figure 5.2 is simply noise), it should be possible to distinguish a good codebook through internal cross-validation. Figure 5.5 shows the result of testing several codebooks for each value of k (16 different codebooks, generated from different random seeds; codebooks are independent for different values of k).

A few qualitative conclusions are possible: local features clearly outperform the global features. This is the major result of this chapter (further testing below will further strengthen this statement). Secondly, incorporating other information improves the result. It seems that SURF-dist performs slightly better than the other methods of doing so. However, when attempting to learn a codebook for these features is slightly less successful than for SURF-ref. Therefore, these will be used for the rest of the current work.

The final method is thus the following: for several values of k and random initialisation, learn a codebook; perform cross-validation to choose the best one. This is a computationally heavy procedure (several rounds of cross-validation are used), but several of its steps can be run in parallel. Milk, the package described in Appendix A.4, supports all of these operations with a simple interface.


80
75
70
65
60
55
50
accuracy (%)
0
100
200
300
400
500
k

	SURF	SURF-dist	SURF-ref
Without field-level features	70%	**71%**	70%
With field-level features	63%	69%	70%

Table 5.2: Summary of Results for Widefield Dataset. Accuracies were estimated through 3 fold cross-validation. Using field-level features alone achieves 61% accuracy (with the same classification system).

	Cyto	Cytosk	Lyso	PM	Mito	N	NO
Cyto	115	10	3	15	8	4	0
Cytosk	14	147	3	2	30	1	0
Lyso	3	1	14	0	50	0	1
PM	31	6	2	9	2	1	0
Mito	22	30	15	0	126	6	1
N	25	1	0	1	0	219	9
NO	1	0	0	0	1	16	95

Table 5.3: Confusion Matrix for Widefield Seven Class Dataset. This was estimated using SURF-ref and 3 fold cross-validation. Legend: NO: nucleoli, N: nuclear, Mito: mitochondria, Cyto: cytoplasmic, PM: plasma membrane, Lyso: lysosome, Cytosk: cytoskeleton.

	SURF	SURF-ref
Without field-level features	61%	68%
With field-level features	**69%**	**69%**

Table 5.4: Summary of Results for Widefield Dataset. Accuracies were estimated through 3 fold cross-validation. Using field-level features alone achieves 61% accuracy (with the same classification system).

The results of applying this method for the smaller dataset are summarised in Table 5.2. The full confusion matrix is presented as Table 5.3. Because the smallest class (*plasma membrane*) only has 3 clones (but a total of 51 images), estimates were obtained through 3 fold cross-validation.

Recall that the same classifier using only field level features achieves only 61% accuracy. All of the SURF variations tried outperformed that method, by a wide-margin (except for SURF with field-level features, which has only a small advantage).

For the full, ten class, dataset, using field-level features achieves 60%. The results are summarised in Table 5.4 and the full confusion matrix is displayed as Table 5.5. The difference between SURF and the field-level features is not significant ($P(r_0 > r_1|\text{data}) = 0.79$), but the difference between SURF and SURF-ref is significant at the 1.7×10^{-5} level. The boost gained from adding the field-level features to SURF-ref is not statistically significant ($P(r_0 > r_1|\text{data}) = 0.83$).

As promised, SURF-ref is also very efficient in terms of computational time. On average, on my implementation, computation requires 7 sec. per image for both interest point detection and feature descriptor computation.

As part of interest point detection, each point is ranked according to a metric of how strongly it matches the

	ER	G	Cyto	Cytosk	Lyso	PM	Mito	N	NO	UL
ER	27	0	1	0	12	0	6	4	0	0
Golgi	1	14	7	0	19	1	14	2	0	4
cytoplasmic	0	1	98	5	3	12	2	12	0	22
cytoskeleton	0	0	11	157	1	1	25	2	0	0
lysosome	0	7	6	0	26	0	28	0	0	2
membrane	0	0	22	5	3	15	4	1	0	1
mitochondria	7	2	17	27	13	0	125	5	0	4
nuclear	9	0	5	0	0	1	1	218	1	20
nucleoli	0	1	1	0	0	0	0	10	99	2
unlabeled	0	0	12	1	0	2	2	56	0	159

Table 5.5: Confusion Matrix for Widefield Seven Class Dataset. This was estimated using SURF-ref and 3 fold cross-validation. Legend: ER: endoplamic reticulum, G: GOlgi, Lyso: lysosomes, NO: nucleoli, N: nuclear, Mito: mitochondria, Cyto: cytoplasmic, PM: plasma membrane, Lyso: lysosome, Cytosk: cytoskeleton.

approximate filter used—see the SURF paper for details (Bay, Ess, et al., 2008). Because of the large size of the datasets that were being generated from the RandTag images, I limited the number of interest points per image to 1024 (which are the 1024 highest matches according to the metric alluded to above). The traditional SURF consists of 64 descriptor values. With the addition of some interest point related values (location, size, match strength), one needs to store 70 floating point values per interest point. Using 4 Byte floats, 1024 descriptors can be saved in 280 KiB (560 KiB for SURF-ref). This is smaller than the average image file size. All of the results above are obtained with the number of points limited to 1024 points per image.[2]

The classifier learned from this labeled dataset can be applied to the whole collection. In order to do so, I added to the dataset a "unlabeled" class consisting of unlabeled control clones. The results of this operation are presented as Table C.1, page 95.

Uniprot is a database of protein information. Among other facts, it annotates proteins with Gene Ontology (GO) terms. Additionally, each annotation is meta-annotated with an *evidence code*, a term from a vocabulary also defined by GO. Of these, the code which is expected to be the most reliable is IDA: inferred from direct assay.

Table 5.6 shows the automatic classification and the IDA-marked GO terms for all proteins in our collection, for which the system return a high-confidence location label (arbitrarily defined as greater than 65%), that was not "unlabeled," and that are tagged with an IDA term.

The unlabeled label is likely to correspond to proteins that are not present in high enough concentrations to be distinct from auto-fluorescence in this cell type under these conditions.[3] Therefore, no subcellular assignment is possible.

[2]The highest value interest points are the most well matched ones. One can expect that they will be the least likely to be the result of the noise in the image. It is thus possible that limiting the processing to high value interest point has a positive effect in the image characterisation.

[3]It is also possible that the chimeric protein, while present, is not fluorescent as the native portion quenches the

Gene	Recommended name	Location (confidence)	Uniprot IDA GO terms
LMNA	Prelamin-A/C	nuclear (96%)	nucleus
LMNA	Prelamin-A/C	nuclear (92%)	nucleus
LMNA	Prelamin-A/C	nuclear (92%)	nucleus
LMNA	Prelamin-A/C	nuclear (92%)	nucleus
HMGA2	High mobility group protein HMGI-C	nuclear (91%)	nucleus
PRRX1	Paired mesoderm homeobox protein 1	nuclear (81%)	nucleus
LMNA	Prelamin-A/C	nuclear (81%)	nucleus
DHE3	Glutamate dehydrogenase 1, mitochondrial	mitochondria (77%)	mitochondrion
TERA	Transitional endoplasmic reticulum ATPase	cytoplasmic (72%)	other membranes
ARPC2	Actin-related protein 2/3 complex subunit 2	mitochondria (72%)	other cellular component, plasma membrane
PRRX1	Paired mesoderm homeobox protein 1	nuclear (66%)	nucleus

Table 5.6: Uniprot Terms and Automated Classification Comparison. This table shows the results of our classifier and the Uniprot annotations for proteins for which this annotation is available. Shown are the *gene name* and *recommended name* fields from Uniprot. The original GO have been replaced by the MGI GO-slim mapping (if multiple terms mapped to the same term, only a single instance was retained). Lmna appears several times as different clones in which this gene was tagged in our experiments. This table is a subset of Table C.1 where all the GO terms and the full well reference are shown.

Table 5.6 shows that for the high confidence labels (both Uniprot and the classifier having high confidence), there is very good agreement with the Uniprot labels. The exception is TERA. From visual inspection (see Figure 5.6), this seems to be a classification mistake.

RandTag Confocal Images

As mentioned in Section 1.4, in addition to the wide-field images, confocal microscopy images were also collected as part of the RandTag project.

We created a labeled dataset containing 217 images from 44 different clones divided into 6 major classes: Nucleoli, Mitochondria, Golgi, Cytoskeleton, Nuclei, Endoplasmic Reticulum (ER). The annotations were obtained by two separate experts.[4] Certain clones are annotated into subclasses (e.g., "Nuclei (uniform)" versus "Nuclei (without nucleoli)"), but these were ignored in this work.

As above, I built an extended dataset by unlabeled, cytoplasmic, plasma membrane, and lysosomal patterns to match the ten classes defined above. Table 5.7 shows the main statistics of this dataset.

When I blindly reannotated the same dataset (I had access to the set of labels that had been used and to the clone images, but not to the assignments), I agreed with the existing annotations for 43 out of 44 clones (98% agreement).

fluorofore.

[4]The experts were Dr. Estelle Glory-Afshar and Armaghan Naik.

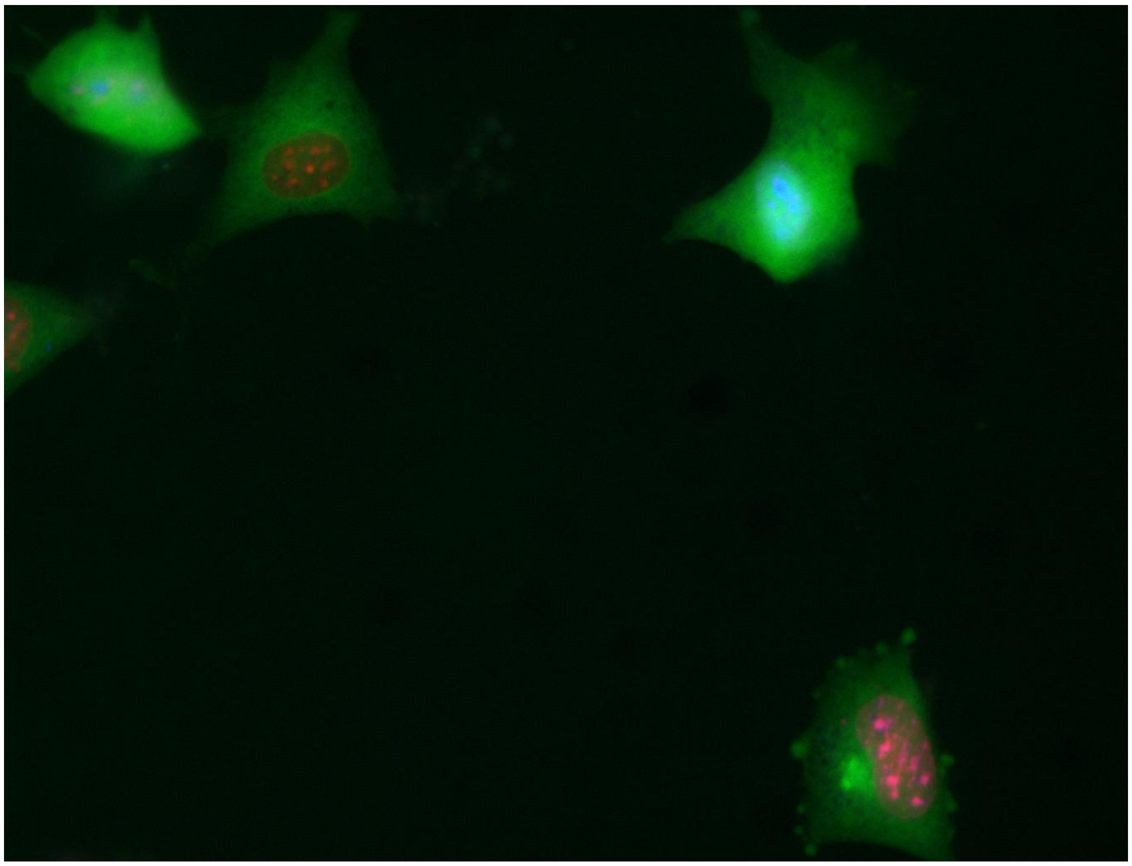
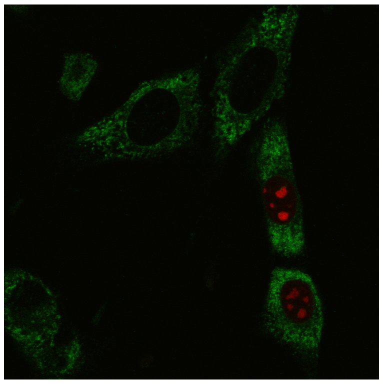
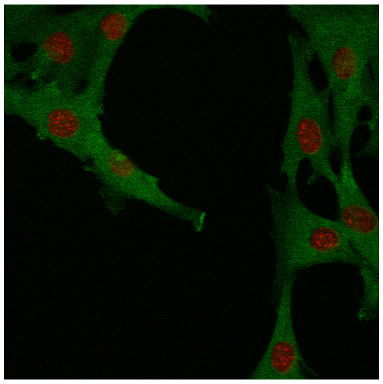


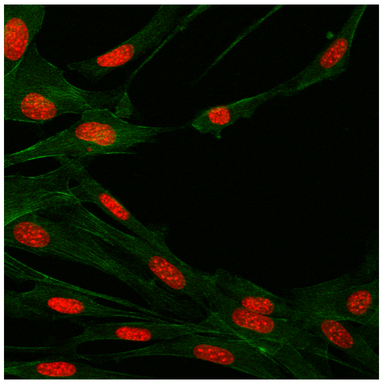
Figure 5.6: Illustrative Image of TERA. This appears to be an endoplamic reticulum pattern, which was misclassified.

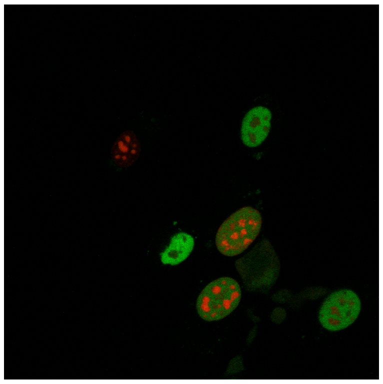
	N	NO	G	ER	PM	L	UL	M	Cyto	Cytosk
Nr. proteins	9	4	4	3	3	3	5	12	8	20
Nr. images	34	20	13	10	11	12	19	49	30	68

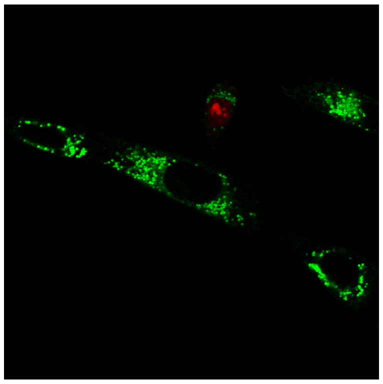
Table 5.7: Statistics of Ten Class Confocal Dataset. Shown are the number of different proteins and images for each class. Legend: ER: endoplamic reticulum, G: GOlgi, Lyso: lysosomes, NO: nucleoli, N: nuclear, Mito: mitochondria, Cyto: cytoplasmic, PM: plasma membrane, Lyso: lysosome, Cytosk: cytoskeleton.

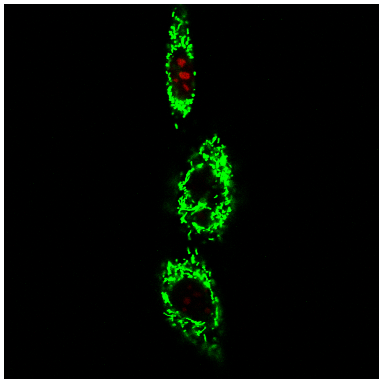

(a) ER

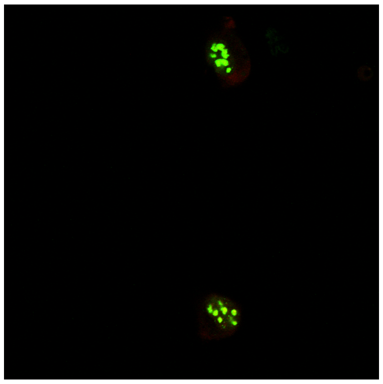

(b) Cytoplasmic

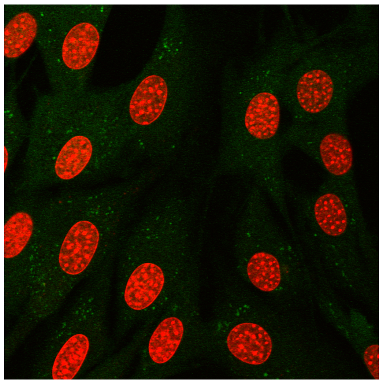

(c) P. Membrane

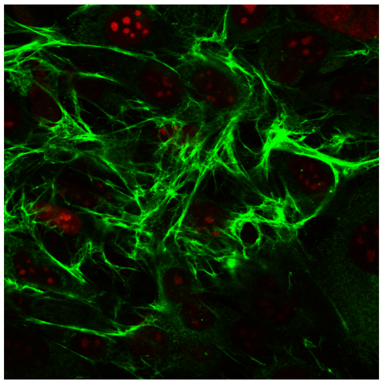

(d) Nuclear


(e) Golgi


(f) Mitochondria


(g) Nucleoli


(h) Lysosome


(i) Cytoskeleton

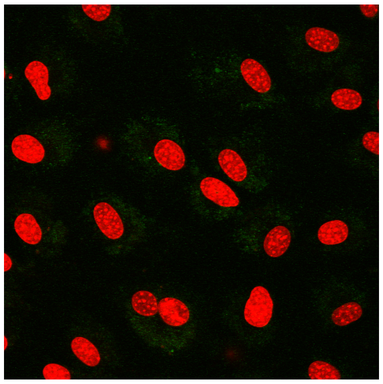

(j) Unlabeled

Figure 5.7: Examples of Confocal Widefield Dataset. This shows an image from each of the 10 classes in the extended widefield dataset. Images have been contrast stretched for publication.

	SURF	SURF-REF
Seven classes: Without field-level features	76%	70%
Seven classes: With field-level features	**82%**	76%
Ten classes: Without field-level features	**72%**	67%
Ten classes: With field-level features	70%	62%

Table 5.8: Summary of Results for Confocal Dataset With. Accuracies were estimated through 10 fold cross-validation. The classifier used, as in other experiments, was based on libSVM with internal cross-validation for parameter selection. Using field-level alone, achieves 72% and 53% accuracy (for seven and ten classes).

	ER	G	Cyto	Cytosk	Lyso	PM	Mito	N	NO	UL
ER	1	1	5	1	0	0	2	0	0	0
Golgi	0	7	0	3	0	0	3	0	0	0
cytoplasmic	1	0	25	0	0	2	0	0	0	2
cytoskeleton	0	1	1	64	0	0	1	0	0	1
lysosome	0	0	3	0	6	0	0	0	0	3
membrane	0	0	0	7	0	0	0	0	0	4
mitochondria	1	4	0	2	0	0	42	0	0	0
nuclear	1	0	3	0	0	0	0	24	6	0
nucleoli	0	0	2	0	0	0	1	6	11	0
unlabeled	1	0	3	0	2	2	0	0	0	11

Table 5.9: Confusion matrix of Ten Class Confocal Dataset. Legend: ER: endoplamic reticulum, G: GOlgi, Lyso: lysosomes, NO: nucleoli, N: nuclear, Mito: mitochondria, Cyto: cytoplasmic, PM: plasma membrane, Lyso: lysosome, Cytosk: cytoskeleton.

Note that I evaluated one clone at a time, observing all of the images from that clone before assigning a label.

The results of classification for this dataset are summarized in Tables 5.8. Table 5.9 shows the confusion matrix for the 10 class problem using SURF. Using field-level features alone achieves 72% and 53% accuracy in the seven and ten class problem, respectively.

SURF-ref achieves only 70% in the seven class problem, but this is not a statistically significant difference when compared to the 72% accuracy obtained with field level features ($P(r_0 > r_1 \mid \text{data}) = 0.76$). The gain of a few percentage points to 76% is also not significant (using either SURF or SURF-ref and field-level features), but the jump to 82% is, at the 0.01 level (compared with the 72% baseline).

In the ten class problem, the difference between 53% and 62% (the lowest local feature result compared with the field-level features) is significant at the 0.02 level, while the difference to the best result (72%) is significant at the 2×10^{-6} level. Adding field-level features results in a drop in accuracy, but the differences are not significant ($P(r_0 > r_1 \mid \text{data}) = 0.12$, for the SURF-ref case).

Locate Images

This subsection describes work with Ramanuja Simha (University of Delaware).

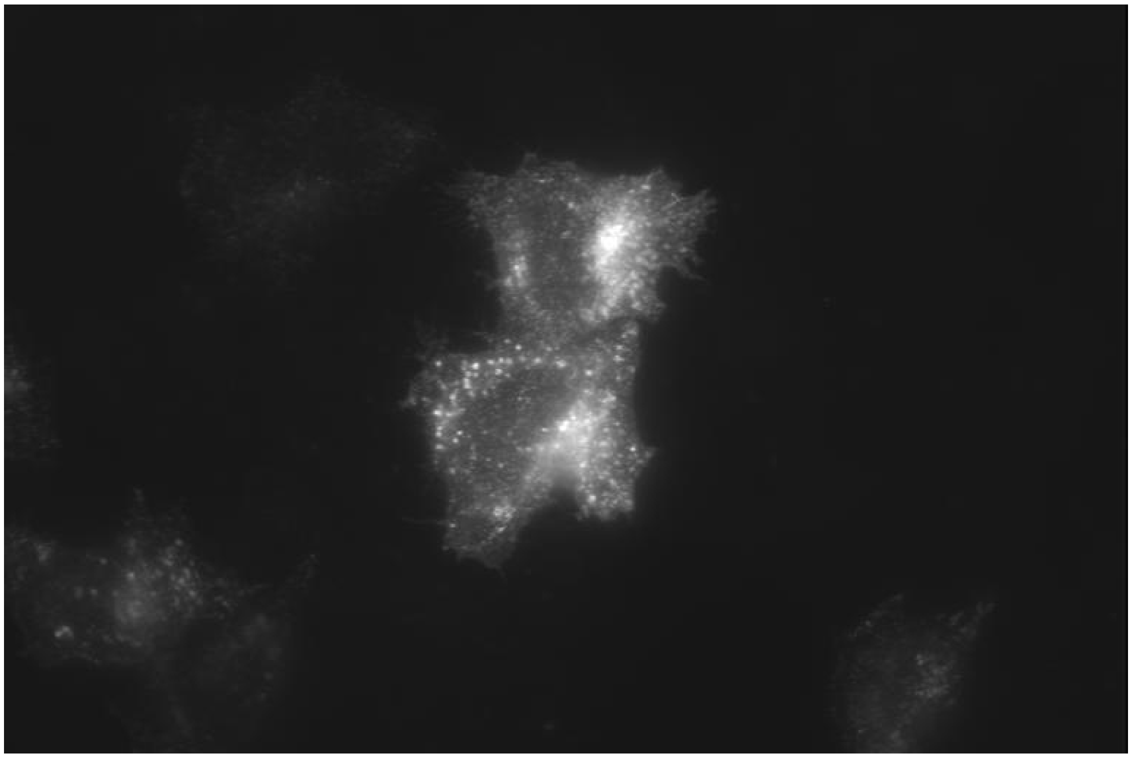


Figure 5.8: Example Image From Locate Dataset. This is an image of Calveolin I, which is annotated with the labels "Cytoplasmic punctate" and "Plasma membrane-like." Originally, the image file was named `01D7_01_60X_HeLa`. The image has been rescaled to fit the page.

Aturaliya et al. (2006) presented a collection of mouse membrane bound proteins imaged with confocal microscopy. The images are available online in the LOCATE database.[5] It consists of 6985 images of 2047 different mouse proteins expressed in HeLa cells. Figure 5.8 shows an example image from this collection.

The images were visually annotated and most proteins are annotated with more than one location. In order to deal with multiple labels, The system I built learns a binary classifier for each label and, at evaluation time, outputs all the labels whose corresponding binary classifier returned a positive label. Each binary classifier was learned independently.

Instead of accuracy, it is more meaningful to measure the F_1 score, defined as:

$$p = \frac{T_p}{T_p + F_p}, \tag{5.4}$$

$$r = \frac{T_p}{T_p + F_n}, \tag{5.5}$$

$$F_1 = 2\frac{pr}{p + r}. \tag{5.6}$$

[5]Available at `http://locate.imb.uq.edu.au/`.

Feature set	Precision	Recall	F_1
SURF	79%	50%	62%
SURF-means	**81%**	53%	64%
field-level	79%	57%	66%
field-level + SURF	80%	**60%**	**69%**

Table 5.10: Results for Locate Dataset. Shown are precision, recall, and the F_1 measure.

	SURF	SURF-REF
Without SLF7DNA	90%	92%
With SLF7DNA	**94%**	**94%**

Table 5.11: Summary of Results for Hela-2D Dataset. Estimates obtained through 10 fold cross-validation for different feature sets.

Where T_p stands for *true positives*, F_p for *false positives*, and F_n for *false negatives* (p and r are normally called *precision* and *recall*).

Table 5.10 summarises the results obtained. One variation that is simple and achieved very decent results is to simply average the SURF descriptors for each image (denoted as SURF-means). SURF does not do as well as the field-level features on this dataset. However, again, the combination with the field-level features leads to the best overall results.

Murphy Lab HeLa 2D Dataset

The Murphy Lab HeLa dataset is by now a benchmark in this area, used by many researchers (Boland, Markey, et al., 1998; Boland and Murphy, 2001; Chebira et al., 2007; Huang and Murphy, 2004b; Marée et al., 2007; Nanni, Brahnam, et al., 2010; Rajapakse, 2008). This is a ten class problem. Nanni, Brahnam, et al. (2010) obtained the best reported results results on this dataset, 96% accuracy, using a combination of texture and other features.

Images contain both a protein and a DNA channel. Therefore, SURF-ref can be applied. For these experiments, the optimal codebook was learned through cross-validation. The same classification system was used.

Results are summarised in Table 5.11. The overall accuracy is 94%, comparable to the best reported results (the different is significant at the 2.7% level). The full confusion matrix for this case is shown in Table 5.12. With SLF7DNA alone, and the same classifier set up, the overall accuracy is 87% (the difference to 94% is significant at the 10^{-7} level).

5.4 Conclusions

Local features (in particular, SURF and SURF-ref) outperform field-level features on four very different datasets. Additionally, there is value-added by including information related to the reference channel or combining these features with field-level features.

	DNA	ER	GI	GII	L	M	N	A	E	T
DNA	86	0	1	0	0	0	0	0	0	0
ER	0	84	0	0	0	1	0	0	0	1
GI	0	0	84	2	0	1	0	0	0	0
GII	0	0	4	79	0	1	0	0	1	0
L	0	0	1	0	72	0	1	0	10	0
M	0	3	1	0	1	64	0	0	3	1
N	0	0	1	1	0	0	78	0	0	0
A	0	0	0	0	0	0	0	98	0	0
E	0	2	3	0	5	1	0	0	79	1
T	0	1	0	0	0	1	0	0	1	88

Table 5.12: Confusion Matrix for Hela-2D Dataset with SURF-ref and SLF7DNA Features. Classifier is SVM-based and estimates were obtained through 10 fold cross-validation. Legend: DNA: nuclear, ER: endoplasmic reticulum, GI: Golgi I (giantin), GII: Golgi II (gpp130), L: lysosomes, M: mitochondria, N: nucleoli, A: actin, E: endosomes, T: tubulin.

Interestingly, we observed a strong dependency on which clustering was used for forming the bag of words model. However, I also showed that you can learn the best codebook by cross-validation. Therefore, this is the recommended method for supervised classification of these images.

In all of the datasets, the results of using field-level features could be improved by addition of the SURF variants.

Local features are also very fast, computationally (a few seconds per image); but do require pre-clustering before use. This step can be slow, but one can work with a subset of the data for efficiency. Therefore, the process can scale to very large image collections. The space requirements for saving all of the detected keypoints can become very large. Again, we used only part of the data (in this case, the most strongly detected keypoints) to keep the space requirements below those of saving the input image data.

§6 Topic Modeling With Multiple Sources of Data

As discussed in Section 1.4, out of the RandTag project, there were over 5000 wells with images, of which 208 are of known proteins (144 different proteins).

For these proteins, there is information available from online databases, which can inform subcellular assignments. A small minority of proteins have previous assignments derived from direct assays, but the majority has only predicted assignments or no assignments.

For modeling multiple sources of data, I used topic models derived from the basic LDA model reviewed in Section 2.5. In particular, there were two goals: (1) to add labels to proteins and (2) to incorporate numeric features such as the field-level features.

I will first present the model, validate it on a small test dataset for the supervised classification task, then apply it to the larger dataset. Finally, I will show how the model can be used successfully in semi-supervised mode.

6.1 Extensions to LDA

Topic models, and in particular Latent Dirichlet Allocation, have been extended to incorporate multiple sources of information, including numeric image features (Blei and Jordan, 2003). For example, Ahmed et al. (2009) developed a model for biological images and their textual captions where the images were represented as a set of numeric features. We use a similar model here.

Images are represented by local features (see Chapter 5) and field-level features. The local features are clustered into visual words. The field-level features are treated as numeric values, combined into an array F.

To sample each of numeric features, we first sample a topic indicator z_f (which depends on θ, like the topic indicator for words). For each topic, there is an associated pair (μ_k, σ_k^2), which parameterizes a normal distribution. The set of these parameters is $\mathcal{F}$. Thus, given z_f, we sample F_f from the corresponding normal:

$$P(\vec{F} \mid \mathcal{F}, \theta) = \prod_f \mathcal{N}(F_f \mid \mathcal{F}_{f,z_f})\mathcal{C}(z_f \mid \theta). \tag{6.1}$$

When compared to the corr-LDA, we note that our model is being applied in a different setting. Blei and Jordan (2003) were attempting to derive annotations from numerical features. Therefore, the best results were obtained when the annotation was dependent on the topics of the numerical features. In our case, there is no need to make the word outputs be a stochastic function of the numeric outputs as they all constitute features of the image.

Multiple modalities of words are modeled as additional LDA-like blocks, with independent multinomial parameters Ψ^i. In Figure 6.1, two modalities are shown, but more are possible (my current implementation uses three: widefield images, confocal images, and gene ontology terms).

In order to model the possible labels that can be associated with an image, we implement a variation on the supervised LDA (sLDA) model described in Section 2.5. For each possible label, there is an associated γ_ℓ weight vector that gives: weight to each topic.

Unlike the model described in Section 2.5, the generation of the label is a two step process:

$$v_\ell \sim \mathcal{N}(\bar{z}^T\gamma_\ell, 1)\mathcal{N}(0, \sigma_0^2), \tag{6.2}$$

$$u_\ell = [\![\, v > 0 \,]\!]. \tag{6.3}$$

Thus, v_ℓ is $\bar{z}^T\gamma_\ell$ plus some Gaussian noise with a zero-centered Gaussian prior, while u_ℓ is a deterministic function of v. u_ℓ is observed, while v_ℓ is kept latent. This is a similar structure to the one that was used by Bae and Mallick (2004) for gene selection.

The values of gamma are given a Laplacian prior, parameterised by λ. This is equivalent to setting an ℓ_1 penalty on γ (Park and Casella, 2008).

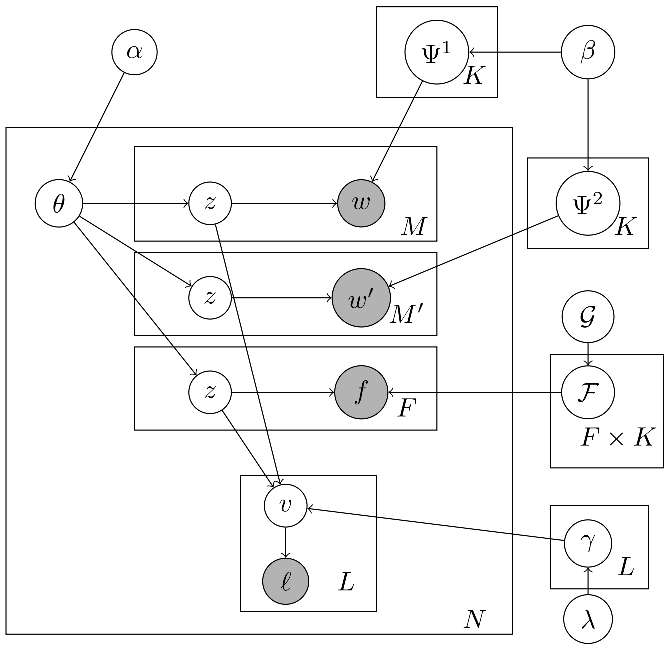


Figure 6.1: Supervised Latent Dirichlet Allocation with Gaussian Mixture. Shown in grey are the nodes that are observed. w represents the first type of word-like features, w' the second (additional types are, naturally, possible), f represents the numeric features, and ℓ the document labels.

The full model is thus:

$$\Psi^a_k \sim \mathcal{D}_{\mathcal{W}}(\beta), \tag{6.4}$$

$$\mathcal{F}_{k,f} \sim \mathcal{N}/\mathcal{G}(\mathcal{G}), \tag{6.5}$$

$$\theta_i \sim \mathcal{D}_{\mathcal{K}}(\alpha), \tag{6.6}$$

$$z_{ij} \sim \mathcal{C}(\theta_i), \tag{6.7}$$

$$w_{ij} \sim \mathcal{C}(\Psi_{z_i j}), \tag{6.8}$$

$$f_{ij} \sim \mathcal{N}(\mathcal{F}_{z_i j}), \tag{6.9}$$

$$\gamma_\ell \sim \mathcal{L}(0, \lambda), \tag{6.10}$$

$$v_{i,\ell} \sim \mathcal{N}(\bar{z}^T \gamma_\ell, 1), \tag{6.11}$$

where $\mathcal{N} - \mathcal{G}$ is a normal-gamma distribution, $\mathcal{L}$ is a Laplacian distribution (as before, $\mathcal{N}$ represents the normal distribution, $\mathcal{D}_{\mathcal{N}}$ a Dirichlet of size N, and $\mathcal{C}$ a categorical distribution).

A preliminary version of the model extended $\bar{z}$ with a constant element (i.e., $\bar{z}_0 = 1$). This was a mistake, as can be seen by the fact that for every γ, there is a γ' defined by $\gamma'_{k \neq 0} = \gamma_k + \lambda$ and $\gamma'_0 = \gamma_0 - \lambda$ such that $\bar{z}^T \gamma = \bar{z}^T \gamma'$ (recall that $\sum \bar{z}_k = 1$).[1]

[1]In fact, numerically, $\bar{z}^T \gamma \neq \bar{z}^T \gamma'$ because of rounding errors. Therefore, the optimisation will search for minor

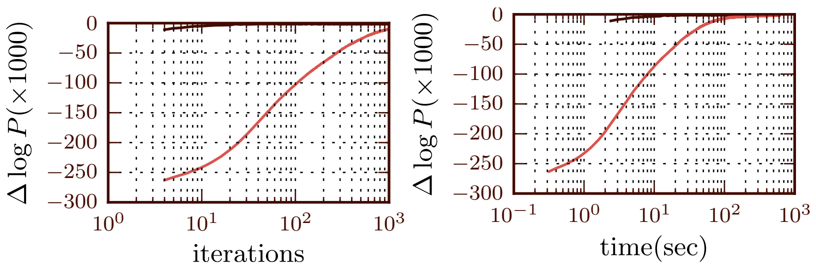


Figure 6.2: Convergence of Collapsed and Uncollapsed Gibbs Sampling. Shown is $\Delta \log P$ plotted as a function of the number of iterations (left panel) and the wall clock time (right panel) on the Associated Press dataset from Blei, Ng, et al. (2003). $\Delta \log P$ is $\log P - \max \log P$, where the maximum is for that sampling model—likelihoods are not directly comparable as the collapsed model as fewer variables. The uncollapsed sampler uses 8 processors. The number of iterations of collapsed sampling is the same in both panels.

6.2 Sampling

For sampling, I initially used uncollapsed Gibbs sampling. Collapsed Gibbs sampling is normally preferred (Griffiths and Steyvers, 2004). However, the uncollapsed version has two advantages that apply to our situation: (1) although the standard collapsed LDA is easier to implement than the uncollapsed variant, more complex variations need to be derived individually, slowing down progress when multiple variants need to be implemented; and (2) uncollapsed Gibbs sampling is embarrassingly parallel (all of the documents are independent of each other, given the topic parameters), allowing us to take advantage of multi-core machines.

In Fig. 6.2 I show the log-likelihood of collapsed and uncollapsed Gibbs sampling as a function of the number of iterations and of wall clock time, using 8 processors. Shown is the difference, at each point, to the best observed model, so that the best observed model is defined to have $\Delta \log P = 0$. While collapsed and uncollapsed sampling can represent the same model, the log likelihoods are not directly comparable as the models are in different parameter spaces.

The convergence of collapsed sampling is faster in both terms of iterations and wall clock time, but the difference is smaller in time. There have been proposals to use distributed variation of collapsed sampling, notably by Newman et al. (2009), but they rely on approximations and are harder to implement.

For simple LDA, I do not need to save the value of all the z variables. The implementation samples each z, but save only aggregate statistics in θ_i for each document. This makes the memory usage much better per document as well as the process slightly faster than the alternative where you save all of the zs. Unfortunately, for SLDA variations, we need to save the zs as, within the same document, they are not independent given θ, Ψ, $\mathcal{F}$ and v_ℓ.

Sampling is similar to in standard LDA for θ, z, w. Sampling f given z and $\mathcal{F}$ is trivial. The parameters $\mathcal{F}$ are sampled from the Gaussians and Gamma distributions (for the mean and standard deviation, respectively).

gains obtained through rounding and result in very large absolute values for γ.

$p(v_\ell \mid \bar{Z}, \gamma)$ is a truncated Normal distribution, which is sampled using the algorithm from Robert (1995).

In the current version, γ_ℓ is not sampled. It is rather "solved for," or equivalently, set to its highest likelihood value. The following equation is solved

$$\gamma_\ell = \arg\min_{\gamma} \left(\bar{Z}\gamma - \boldsymbol{v_\ell}\right)^2 . \tag{6.12}$$

Collapsed Gibbs Sampling

As shown above, collapsed Gibbs sampling is much faster than the uncollapsed variation. Therefore, after the model specification was finalised, I derived the collapsed sampler for this model.

In general, I wish to sample a particular z value ($z_{i,j}$: the jth entry for document i) given the hyperparameters and the values of the other z values, denoted by $z_{-(i,j)}$.

I start by showing a general equality, which will be used later. Assume the goal is to integrate away parameter ε, which has prior $P(\varepsilon \mid \xi)$ and leads to emission $P(e \mid \varepsilon)$. Now, further assume that $P(\varepsilon \mid \xi)$ has the form $f(\varepsilon;\xi)/D(\xi)$ and is the conjugate prior to $P(e \mid \varepsilon)$.

$$\int_\varepsilon P(\varepsilon \mid \xi) P(e \mid \varepsilon) d\varepsilon = \int_\varepsilon \frac{1}{D(\xi)} f(\varepsilon;\xi) P(e \mid \varepsilon) d\varepsilon, \tag{6.13}$$

$$= \frac{1}{D(\xi)} \int_\varepsilon f(\varepsilon;\xi) P(e \mid \varepsilon) d\varepsilon, \tag{6.14}$$

$$= \frac{1}{D(\xi)} \int_\varepsilon k(e) f(\varepsilon;\xi') d\varepsilon. \tag{6.15}$$

Here we used the property: $P(\varepsilon \mid \xi)P(e \mid \varepsilon) = k(e)f(\varepsilon;\xi')$, which is a consequence of conjugacy. Now, we can multiply and divide by $D(\xi')$ inside the integral:

$$\int_\varepsilon P(\varepsilon \mid \xi) P(e \mid \varepsilon) d\varepsilon = \frac{1}{D(\xi)} k(e) \int_\varepsilon \frac{D(\xi')}{D(\xi')} f(\varepsilon;\xi') d\varepsilon, \tag{6.16}$$

$$= \frac{D(\xi')}{D(\xi)} k(e) \int_\varepsilon \frac{1}{D(\xi')} f(\varepsilon;\xi') d\varepsilon, \tag{6.17}$$

$$= \frac{D(\xi')}{D(\xi)} k(e) \int_\varepsilon P(\varepsilon \mid \xi') d\varepsilon, \tag{6.18}$$

$$= \frac{D(\xi')}{D(\xi)} k(e). \tag{6.19}$$

Therefore, integrating away the parameter ε results in the ratio of the posterior and the prior normalisation. We can further simplify by remarking that neither $D(\xi)$ nor $k(e)$ depend on the value of $z_{i,j}$. The final result is $P \propto D(\xi')$. Given the particular form of this expression, there may be other terms that are independent of $z_{i,j}$ and the expression may be simplified even further.

I am going to present a solution where γ_ℓ is not integrated out (although v_ℓ is). Unfortunately, it was not clear to me whether it is possible to also integrate this parameter away or whether it is impossible to do so in closed form (or using standard numeric functions for which fast implementations are available).

There are actually two variations, P_w for when $z_{i,j}$ corresponds to a word emission, and P_f for when it corresponds to a numeric one. I will start by presenting P_w. I will also assume a single label (the general case simply multiplies all of the labels together):

$$\begin{aligned} P_w(z_{i,j} \mid \alpha, \beta, z_{-(i,j),w}, \gamma) &\propto \int_\theta \int_\psi \int_v P_w(z, \theta, w, v, \ell \mid \alpha, \beta) d\theta d\psi dv \\ &\propto \int_\theta P(z \mid \theta) P(\theta \mid \alpha) d\theta \times \\ &\quad \times \int_\psi P(w \mid \psi) P(\psi \mid \beta, z) d\psi \times \\ &\quad \times \int_v P(\ell \mid v) P(v \mid z, \gamma) dv. \end{aligned}$$

The first two integrals fit the framework above and are standard (Griffiths and Steyvers, 2004; Heinrich, 2009), but the third is not. Assume that ℓ is true:

$$\int_v P(\ell \mid v) P(v \mid z, \gamma) dv = \int_v [\![v > 0]\!] \mathcal{N}(v \mid \bar{z}^T \gamma, 1) dv = \Phi(\bar{z}^T \gamma), \tag{6.20}$$

where Φ is the cummulative distribution function for the Gaussian distribution. If ℓ is negative, then the above becomes $\Phi(-\bar{z}^T\gamma)$. If we defined $s = +1$ if ℓ, and $s = -1$ otherwise, we obtain the expression $\Phi(s\bar{z}^T\gamma)$.

The final expression for P_w is thus

$$P_w(z_{i,j} = k) \propto (n_{i,j}^k + \alpha) \frac{w_k + \beta}{N_k + W\beta} \Phi(s\bar{z}^T\gamma), \tag{6.21}$$

where $n_{i,j}^k$ is the number of words in document assigned to topic k, w_k is the number of times that the corresponding word was assigned to topic k, N_k is the number of times that topic k was assigned to anyword, and W is the number of words. All of the these counts are *excluding* the current value $z_{i,j}$ and must be updated when $z_{i,j}$ is updated. To sample $z_{i,j}$, (6.21) is computed for all values of k and the resulting vector is normalised.

The case P_f is similar, but I am not aware of a published derivation, so I present it here.

$$P_f\left(z_{i,j} \mid \alpha, \beta, z_{-(i,j)}, \gamma, \mathcal{G}\right) \propto \int_\theta \int_\mathcal{F} \int_v P(z, \theta, f, v, \ell \mid \alpha, \beta, \gamma, \mathcal{G}) \tag{6.22}$$

$$\propto \int_\theta P(z \mid \theta) P(\theta \mid \alpha) d\theta \times \tag{6.23}$$

$$\times \int_\mathcal{F} P(f \mid \mathcal{F}) P(\mathcal{F} \mid z, \mathcal{G}) d\mathcal{F} \times \tag{6.24}$$

$$\times \int_v P(\ell \mid v) P(v \mid z, \gamma) dv. \tag{6.25}$$

The term in f is the single one that differs from the case for P_w. It fits the schema above:

$$\int_\mathcal{F} P(f \mid \mathcal{F}) P(\mathcal{F} \mid \mathcal{G}) d\mathcal{F} \propto \frac{D(\mathcal{G}')}{D(\mathcal{G})}, \tag{6.26}$$

where $D(\mathcal{G})$ is the normalisation factor for the Gaussian-Gamma distribution, i.e.,

$$D(a, b, n_0, \mu_0) = \frac{b^a \sqrt{n_0}}{\Gamma(a)\sqrt{2\pi}}. \tag{6.27}$$

The posterior parameters are given by:

$$a' = a + n/2, \tag{6.28}$$

$$b' = b + \frac{1}{2}\sum_i (f_i - \bar{f})^2 + \frac{1}{2}\frac{nn_0(\bar{f} - \mu_0)^2}{n_0 + n}, \tag{6.29}$$

$$n' = n_0 + n, \tag{6.30}$$

$$\mu' = \frac{n_0\mu_0 + n\bar{f}}{n_0 + n}, \tag{6.31}$$

where n is the number of observed elements and $\bar{f}$ is the observed mean value. In order to implement this equation efficiently, the algorithm needs to keep track of three variables for each topic and numeric output:

$$n_{j,k} = \sum_i [\![z_{i,j} = k]\!], \tag{6.32}$$

$$F_{j,k} = \sum_i [\![z_{i,j} = k]\!] f_{i,j}, \tag{6.33}$$

$$F^2_{j,k} = \sum_i [\![z_{i,j} = k]\!] f^2_{i,j}. \tag{6.34}$$

All of these can be updated very fast after each assignment to $z_{i,j}$. Unfortunately, the expression does not lead itself to further simplification:

$$D(a', b', n') = \frac{\Gamma(a')}{b'^{a'}}\sqrt{n'}. \tag{6.35}$$

Therefore, the full value of P_f is

$$P_f(z_{i,j} \mid z_{-(i,j)}, a, b, \mu_0, n_0, \alpha, \beta) \propto (n^k_{i,j} + \alpha)\frac{\Gamma(a')}{a'^{b'}}\sqrt{n'}\Phi(s\bar{z}^T\gamma). \tag{6.36}$$

The implementation uses logarithmic space for the computation, but expensive to compute functions are unavoidable.

As mentioned above, γ must still be solved for explicitly:

$$\gamma^* = \arg\max_\gamma \sum_i \log P(\ell|\gamma, \bar{z}_i). \tag{6.37}$$

In terms of the cummulative Gaussian function, Φ, the likelihood can be expressed as (assuming that ℓ is true):

$$\begin{aligned} P(\ell|\gamma, \bar{b}_i) &= \int_u P(\ell|u)P(u|\gamma, \bar{z}_i)du && (6.38)\\ &= \int_u [\![u > 0]\!] \mathcal{N}(u|\gamma^T\bar{z}_i, 1)\mathcal{N}(u|0, \sigma_0^2)du && (6.39)\\ &\propto \Phi(\frac{\sigma_0^2}{1 + \sigma_0^2}\gamma^T\bar{z}_i). && (6.40)\end{aligned}$$

If ℓ is false, the sign of argument to Φ is reversed. Therefore, the full expression to maximise, including the prior, is:

$$\gamma^* = \arg\max_\gamma \sum_i \log\Phi\left(s_i\frac{\sigma_0^2}{1 + \sigma_0^2}\gamma^T\bar{z}_i\right) - \lambda\|\gamma\|_1. \tag{6.41}$$

One possible way to do perform this minimisation is to use the Newton-Raphson method: starting with an initial guess γ^0, newer approximations are obtained by

$$\gamma^{t+1} = \gamma^t - H^{-1}(\nabla\gamma^t), \tag{6.42}$$

where H is the Hessian matrix and $\nabla\gamma^t$ is the gradient vector at γ^t. This maximisation can be quite costly (because of the need to evaluate many calls to Φ, which is slow even if we approximate it[2]). Therefore, I used a simple approximation: at each iteration of the Gibbs sampler, only a single Newton-Raphson iteration is performed.

The Dirichlet hyper-parameters α and β can be fit using similar techniques (Minka, 2000). I have not implemented this, but set them to fixed values (0.01).

Currently, the GO terms are output as independent words. Therefore, the model can not learn good parameters for rarer topics and I used a small subset of GO, a GO slim. An alternative would be to model the structure of the GO so that information could flow from nearby terms for rare terms, but more precision would still be possible.

Possible Extensions

There are a few possible extensions to the model. I present them here, but did not implement them.

Currently, the word emissions are parameterised by Ψ_k^a, which has a flat Dirichlet prior $\boldsymbol{\beta} = (\beta, \beta, \cdots, \beta)$. There is no reason why there could not be separate values for different regions of the space (even for each individual word). For example, all GO terms could share β_{GO}, whilst the visual words would share a different value β_{WF}. These could either (1) be fit to the data or (2) be specified in a way as to encode prior information about which classes of data should be valued the highest (smaller values of β correspond to higher confidence as they generate more peaked distributions—normally this is evaluated over the word index of $\Psi_{i,k}$, but it is similarly true over the topic index k, i.e., smaller values of β will generate values of $\Psi_{i,k}$, where for fixed i, one has more distinctions between different topics).

Supervised Classification Results

Like for the case of validating SURF-ref, a supervised classification task allows easy validation of the model. We used the same RandTag widefield dataset as described in Section 5.3.

We used SURF-ref features, with a k-means derived codebook to transform the features into visual words. Additionally, we used field-level features with the Gaussian emission model described above.

Similarly to what was done in Section 5.3, the codebook was learned using cross-validation. The field-level features, when used, were first normalised to z-scores (i.e., first the training set mean was subtracted and the result

[2]My implementation uses a 4 term polynomial approximation. An initial implementation used the GNU scientific library and its implementation of Φ. With this more exact implementation Φ was the most costly step in the sampling.

	ER	G	Cyto	Cytosk	Lyso	PM	Mito	N	NO	UL
ER	30	0	1	0	12	0	4	0	3	0
Golgi	1	18	6	1	10	1	16	3	3	3
cytoplasmic	1	0	94	15	4	12	2	11	2	14
cytoskeleton	0	0	7	161	3	1	23	1	0	1
lysosome	1	9	11	0	17	0	31	0	0	0
membrane	0	0	26	9	0	8	2	1	0	5
mitochondria	8	2	14	28	21	0	123	2	0	2
nuclear	7	2	2	2	4	2	2	218	9	7
nucleoli	0	0	0	0	0	0	0	21	88	4
unlabeled	1	5	12	7	7	0	10	21	6	163

Table 6.1: Supervised LDA Classification Results. This is the ten class widefield dataset described in Section 5.3. Legend: ER: endoplasmic reticulum, G: Golgi, UL: unlabeled, NO: nucleoli, N: nuclear, Mito: mitochondria, Cyto: cytoplasmic, PM: plasma membrane, Lyso: lysosome, Cytosk: cytoskeleton.

was divided by the training set empirical standard deviation). Thus, the prior on each topic exactly matches the overall distribution.

When applying a learned model to a new example, the document is first projected into topic space. For this, a small number of Gibbs sampling iterations (30) are run and the $\bar{z}$ vector is observed. Finally, a single label is obtained according to

$$\ell^* = \arg\max_\ell v_\ell = \arg\max_\ell \theta^T \gamma_\ell. \tag{6.43}$$

This was done even if $\forall \ell, v_\ell < 0$ (i.e., no label was expected to be set) or if $v_\ell > 0$ for more than one ℓ value (i.e., multiple labels were expected to be set). Both cases were observed. Implicitly, this gives us the most likely label.

Interestingly, the results obtained by including the numeric features are not as good as those using visual words (SURF-ref) alone. I posit that this is due to the influence of non-informative features which could potentially be removed in a feature-selection step.[3]

Full results are presented in Table 6.1, using 32 topics on the ten class dataset. The results were obtained using only visual words and no field-level features. The overall accuracy is 66% which is lower than the 72% accuracy obtained with support vector machines. The difference is significant at the 1% level.

Using field-level features results in slightly lower accuracy (64%). The difference is not significant, but I still decided to use only visual words going forward. There is a computational cost associated with using both types of features.

As before, the main purpose of the supervised study is to validate the methods. The true value of this model is in the ability to handle multiple data modalities and work in a semi-unsupervised mode. I discuss these aspects next.

[3]I have looked carefully for a calculus or coding error, which was my first hypothesis, but found none. The topics for the numeric features also seem to have a similar distribution to the ones for the visual words.

6.3 Multiple Data Modalities

The results presented in the previous section validate the general topic modeling, but are simply an alternative technique to perform a supervised classification using images. The real gain from topic modeling comes from using other sources of data and from going beyond simple supervised classification settings.

Sources of Data

I considered several sources of data:

1. Wide-field images from the RandTag project,
2. confocal images from the same source,
3. information from online databases on protein location,
4. information from online databases on protein function.

The two types of image are treated as separate types as they are visually different enough that we should not expect the features to be directly comparable (see Figures 1.1 and 1.2).

The information from online databases was obtained in the form of gene ontology (GO) terms (Ashburner et al., 2000). With R. F. Murphy, H. Shatkay, S. Quinn, and J. Liddie, I developed a system specifically for aggregating this information. Nicknamed "waldo," it is described in Appendix A.1.

I used data from Uniprot. Uniprot provides, for each protein, a list of GO terms in all three domains (cellular component, molecular function, and biological process). In order to improve the ability of the model to learn, I used the GO-slim provided by Mouse Genome Informatics (MGI) (Blake et al., 2011). A GO-slim is a subset of GO and a mapping from the full GO to the GO-slim. Unlike GO, there is no standard GO-slim. Given that these are mouse proteins, I decided to simply adopt the one developed by MGI. Compared to using the full GO, there are fewer terms for which a parameter value must be learned.[4]

For orthology information, I relied on the eggNOG database (Muller et al., 2010). This database groups proteins across several species into ortholog groups (OG) defined by sequence similarity. To obtain an ortholog, I query the database for the OG containing the gene of interest and return the found genes in the target species.[5] If a matching protein is found, then its GO terms are brought in as words. These are different words than the corresponding mouse term (i.e., there is both a "nuclear-mouse" and a "nuclear-human" term), but they share the same multinomal Ψ^{GO}.

From the modeling and implementation point of view, there are only three types of information:

[4] An arguably better solution would be to reflect the graph structure of GO as a prior on the parameters. Therefore, one could fit a set of parameters for each term, but still benefit from information from terms adjacent in the graph. This is outside the scope of the current work, but would be an interesting extension.

[5] This operation is a single function call to the *waldo* API.

words These are both visual words and the gene ontology terms. These are modeled as w above.

real Real numbers, such as the field-level features, are modeled as f above.

label These are modeled as ℓ above. The only labels available are those from human annotation.

Finally, I will note that there is no issues in modeling missing data. If a protein is not mapped to an ortholog, then there will not be any orthologous GO term words associated with that protein. If there is only widefield data, then only widefield features will be present and no value need be imputed to the confocal image features. This would not be the case if I had just proposed to concatenate feature vectors.

Results

Supervised Results

A single topic model was learned using both labeled and unlabeled data. The unlabeled data can then be labeled from the rest of the model (this is a form of transductive learning).

All the features for a single protein from different images (even different clones) were grouped together as a single bag of words. This results in 137 different document for topic modeling, each document consisting of all information about a protein.

As above, the label assigned is the one which obtains to the highest response. An estimate of confidence is obtained by subtracting the second largest response. This value does not have a direct correspondence to a fraction, but higher values correspond to higher confidence.

Table 6.2 shows the results of this label assignment for proteins with Uniprot cellular component annotations marked as "inferred from direct assay" (it is analogous to Table 5.6). Shown are the 10 highest confidence proteins for which the label was not "unlabeled." Appendix Table C.2 shows results for all RandTag proteins.

Figure 6.3 shows images from each of the proteins where the algorithm and Uniprot disagree. In the cases of RL7 and PTRF, the pattern is outside of the vocabulary that the algorithm uses (RL7 is a rybosomal protein, which exhibits a mixture of nucleoli and cytoplasmic; while PTRF locates to the caveola, which was not part of the vocabulary).[6] In the cases of ACTN4, CAZA1, and SND1, Uniprot contains a cytoplasmic annotation (agreeing with our labeling). This label is, however, of lower quality than IDA. Our results are thus validation for these.

Semi-supervised Results

The true result of the topic model is a topic-distribution for each protein. One visualization of this structure is defining a proximity relationship between the proteins. One could simply use the Euclidean distance between the

[6]These and similar statements in this paragraph can be sourced to Uniprot, version 104.

Gene	Location	Uniprot IDA GO terms
PRRX1	nuclear	nucleus
LMNA	nuclear	nucleus
PTRF	cytoskeleton	plasma membrane
SUH	nuclear	nucleus
ACTN4	cytoplasmic	cytoskeleton, other cellular component
CAZA1	cytoplasmic	cytoskeleton, other membranes
DREB	cytoskeleton	cytoskeleton
SND1	cytoplasmic	mitochondrion
RL7	mitochondria	cytosol
DHE3	mitochondria	mitochondrion
TERA	membrane	other membranes

Table 6.2: Comparison of LDA Results and Uniprot. Comparison of results reported obtain from the topic model with IDA annotation in Uniprot. Only proteins with a confidence value higher than 1 are shown.

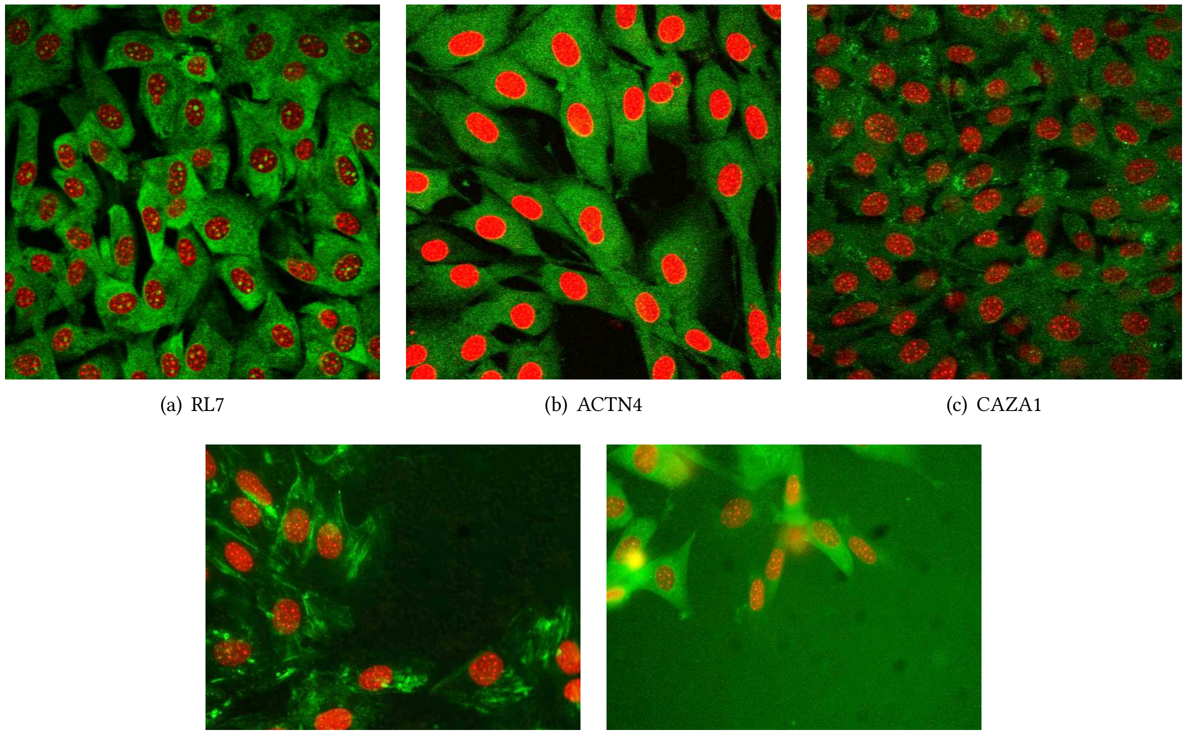

(a) RL7 (b) ACTN4 (c) CAZA1 (d) PTRF (e) SND1

Figure 6.3: Images of Proteins Where There is Disagreement Between the Topic Model and Uniprot. See Table 6.2. Images have been constrast stretched for publication.

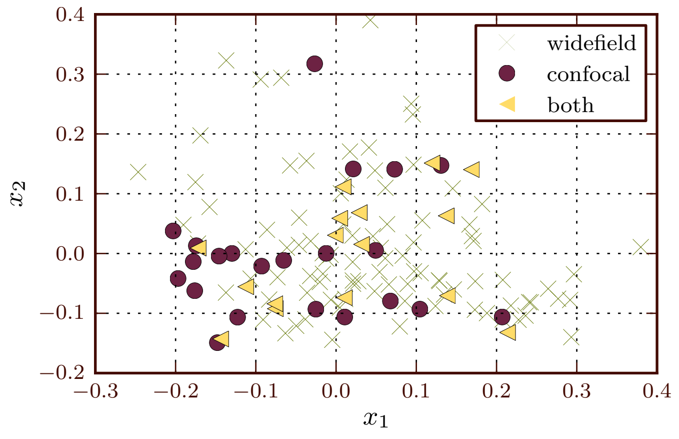


Figure 6.4: Proteins in Two Dimensional Projection. Topics were projected into two dimensions using multidimensional scaling. The green crosses represent the proteins with only widefield images, the purple circles only those with confocal images, and the beije triangles those with both types of images.

topic vectors, but a more appropriate measure is the symmetric Kullback-Leibler divergence (D_s) as the distance measure for two topic distribution:

$$D_s(\theta_1, \theta_2) = D_{\text{KL}}(\theta_1 \| \theta_2) + D_{\text{KL}}(\theta_2 \| \theta_1), \tag{6.44}$$

where D_{KL} is the more widely used non-symmetric variant, defined by:

$$D_{\text{KL}}(P \| Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}. \tag{6.45}$$

This construction allows one to navigate the space of subcellular location by looking at proteins that are close to any protein of interest (it is easy to imagine a graphical interface which would allow the user to perform this operation).

In order to evaluate whether the different modalities had been projected into the same space, I used multidimensional scaling to project to a two-dimensional space. Figure 6.4 shows where the proteins are projected using different symbols for the modalities that were available for chat protein. As can been seen, there is mixing of the modalities.

I also attempted to train a classifier to distinguigh among the modalities in this label space. The accuracy of the model was statistically indistinguishable from that of a model which classifies all its inputs as coming from the majority class (158 out of 255, when the majority class, widefield, contains 155 proteins). This shows that even in the higher dimensions not depictable on paper, the modalities are not separated.

Figure 6.5 is in the same space as Figure 6.4, but instead of markers, there are single images for each protein. The positions were slightly shifted to separate closely lying points[7] so that individual images can be seen better.

[7]The final position is computed to minimize the weighted sum of the distance to the initial position and the sterical interactions with other thumbnails. The parameters of this were chosen by visual tuning.

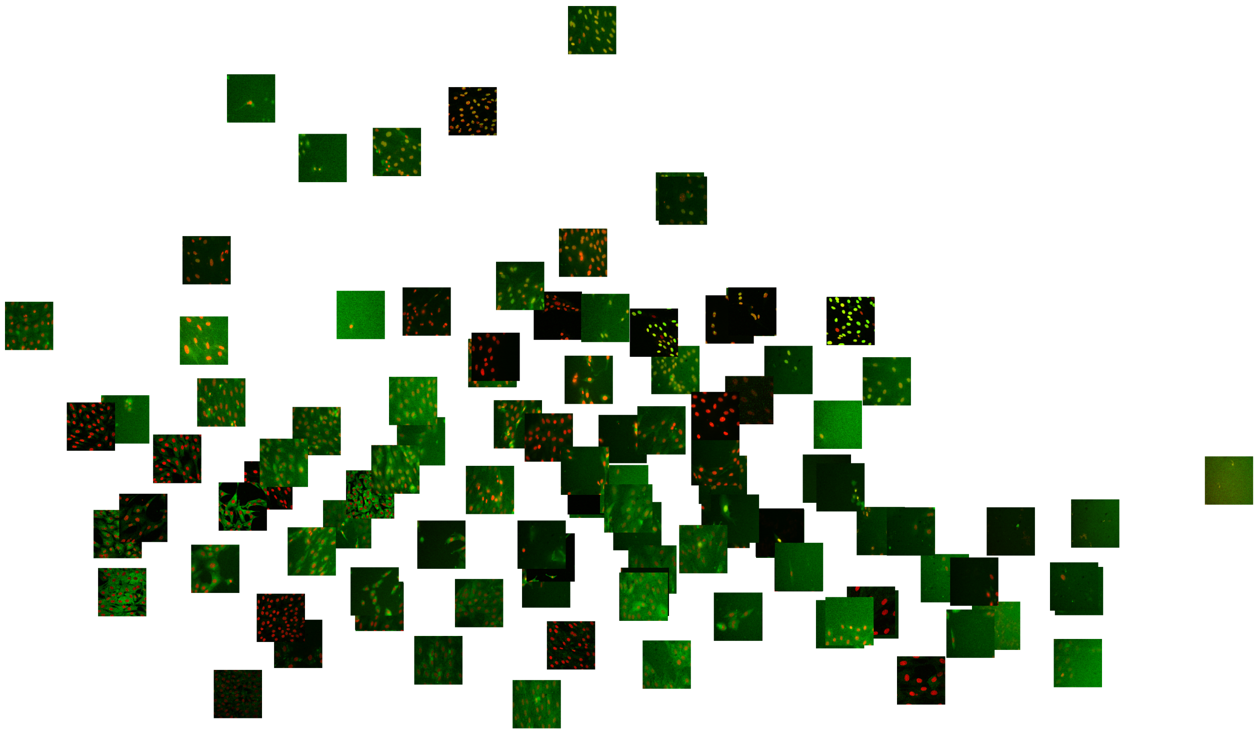


Figure 6.5: Proteins in Two Dimensional Projection. Topics were projected into two dimensions using multidimensional scaling and close by lying proteins were separated for easier visualisation.

6.4 Conclusions

The current chapter presented a model which can use several sources of information in a semi-supervised way: while some proteins may be annotated with one of a predefined set of labels, others need not be and all are modeled together. The labels constrain the learned representation.

Although our goal was not supervised classification, it is an easy task on which to validate the system. In this task, I obtained *better results* than the support vector machine classification. Interestingly, the addition of numeric features was a pessimisation as the results were not as good.

The semi-unsupervised results are more difficult to evaluate. Unfortunately, even though all of the sources are brought into the same space, the topic results for each protein still dependend on the data modality that was used. However, once the transformation to label space is performed, that connection is lost. This would have been impossible in a purely unsupervised topic modeling, which shows the necessity and power of adding the labeled data.

§7 Conclusions & Future Work

7.1 Conclusions

In this work, I made two interconnected claims: (1) that there were better numeric image representations than a feature vector and (2) that inference would benefit from other sources of information (besides images).

The underlying goal was to analyse images from the RandTag collection. Technically, this meant that methods needed to scale to many images and proteins, with a small computational cost per image.

When applied to the RandTag images, my methods were able to reproduce known location assignments, in the few cases where these were available, with high accuracy. Most importantly, this project was able to assign locations to many proteins for which no quality assignment existed before.

In addition to the assignments, the topic model has an unsupervised component in that it maps the proteins into a low dimensional space, the space of topic distributions.

Image Representations

I explored two different alternative representations for bioimages: (1) subcellular pattern mixtures and (2) local features.

Subcellular pattern mixtures were an existing method, but there were no methods to derive the mixture for an unannotated collection of images. For the context of this work, it was necessary to extend them to unsupervised methods. The methods I proposed with Tao Peng work well for this task, with results comparable to the supervised mode. This is a very good result and makes the methods applicable to large collections.

Local features are very good representation for biological images over a variety of datasets achieving performance comparable to or better than feature sets specifically designed for this problem. They can be combined with existing features and the best results are combinations. In fact, on the four datasets that I used, which have very different characteristics, the combination of field-level and SURF-ref (where applicable) always outperformed our previous methods. The methods were also fast.

Therefore, I recommend that local features become part of the standard toolkit for bioimage analysis.

Topic Modeling

Topic modeling was used twice in this work. First, in Section 4.3, it was used as part of an unmixing solution. Then, in Chapter 6, they were the basic technology enabling the use of multiple sources of data in different modalities.

In that chapter, I showed that topic models could work very well even for the task of supervised classification. This slightly outperformed the support vector machine based system using only SURF-ref. Together, this shows that topic models can be very useful for bioimage analysis. The computer vision field has explored this area and it is natural that some of the advances there will be helpful to bioimage analysis after suitable adaptation.

Modeling With Multiple Sources of Data

The topic model discussed in Chapter 6 was therefore developed to be modular so that multiple sources of data could be used and more cold be plugged-in in the future. The goal, which was achieved by the topic model was to be able to bring several data sources into a single topic space (which is a sparse, low-dimensional, representation of the proteins).

One source of data that I made use of was online databases. The online databases that i have relied on, and that some systems have even used as training data, are not so helpful at answering the question “where will this protein localise?” In fact, that is not the question they answer. Rather, they answer the question “where are all of the places where this protein has been observed?” They do not register the cell type, the conditions (were this “normal conditions” or were the cells subject to an external stimulus?) in which the observation took place. It is typically impossible, without further study to know whether which of the several annotations present will be applicable in any given cell type.

However, to disregard the information that is provided would also be a mistake as there is valid information (for many cases, the protein will localize in the same compartment in different cell types and conditions). The model I presented can reason probabilistic and will therefore assign weights to each possible input datum.

The current system can also make use of functional annotations (even from homologous proteins). This model can thus bridge the gap between the field of subcellular location determination and that of subcellular location prediction as it incorporates both direct and indirect data. In the topic space, data from different modalities is brought together in a way that it becomes impossible to distinguish what data was available for each protein.

Models similar to the one presented had been used before for images and text in other contexts, but not for the problem of biological image understanding.

7.2 Future Work

The system here could be improved in many small ways, several of which were alluded to in the text. This is natural, given that it was the first system proposed with as much flexibility.

The system demonstrated here collected information from several sources of information, but it was not exhaustive and it is desirable to extend it. For example, the inclusion of the free text from scientific publication mentioning a protein has shown good results in enhancing prediction systems (Briesemeister et al., 2009; Shatkay et al., 2007).

Another system in which I worked during my PhD was the Structure Information Literature Finder, SLIF, (Coelho, Ahmed, et al., 2010). This system aggregates the images *inside* scientific publications. Thus, it can go one step further than just integrating the free-text of a document, but also integrate its images. Potentially, one could "peek over the shoulder of the researchers" and even provide a reinterpretation of the data.

The system as designed assumes that all cells in an image display the same pattern. It cannot properly handle heterogeneity in the population. An additional level of modeling, which took into account the spatial distribution of protein would likely be necessary to capture this variation. If cell-level segmentation of sufficient quality is available, then the modeling could be performed at the cell level, taking into account the spatial relationships. The work of S.-C. Chen, Gordon, et al. (2008) could serve as starting point.

I will finish, as I started, by describing my long term vision of a system which can integrate all of the information that is publicly available, automatically weigh its strength (by a measure of internal consistency), and output the very best structured conclusions about the location of proteins. My work was but a step in that direction.

§A Software

This chapter describes the software that was produced as a result of the work for this dissertation. All of this software is available as open source software.

A.1 Waldo: Aggregating Subcellular Location Information

This section describes joint work with S. Quinn, J. Liddie, H. Shatkay, and R. F. Murphy.

Introduction

The first step towards the integration of information is its aggregation. We were not aware of any active subcellular information database that was up to date and comprehensive (many projects have not been updated in many years).

Generic protein information resources such as Uniprot and Mouse Genome Informatics (MGI) also list subcellular location annotation by annotating proteins with terms from the cellular component branch of the gene ontology (GO terms).

We implemented a system, nicknamed "waldo," for aggregating and normalising this information. The basic question that waldo answers is of the form: "Where is protein P?"

Waldo supports three interaction modes:

1. It can be used as a Python library.

2. It can be accessed with a web browser; this HTML-based interface is geared towards human users.

3. It can be accessed as a web-based service, returning JSON (Javascript Object Notation) formatted answers; this interface is geared towards programmatic access. Currently, only a few functions are exposed through this interface.

Architecture

The current architecture has a *loading* step which must be run before the data can be accessed. This step involves downloading information dumps from the online services and then parsing them into a local database. Henceforth, all information is served from this local copy of the information.

We envisioned the system being used for large scale studies where many queries are performed. Therefore, we avoided online querying of the original databases for both performance reasons and to avoid overstepping any usage limits that these databases impose.

One major issue with dealing with many databases is the handling of identifiers, as each database chooses its own internal format. For a well-studied organism, such as mouse, there are, though, mappings between these identifiers. In Waldo, we chose to use Ensembl gene and peptide ids as the major internal identifiers of a protein.

Interface

The library is structured using object-oriented principles. Each information source implements the following functions:

`load` Load from the corresponding database dump.

`retrieve.from_ensembl_peptide_id` Translate to the source's internal identifier.

`retrieve.retrieve_go_annotation` Given an identifier, retrieve a list of GO terms.

For example, here is how to retrieve and print all of the information from both MGI and uniprot relevant to the protein ENSMUSP00000116259 (a.k.a. Serine/threonine-protein kinase SRPK1, a splicing regulator):

```
import waldo.uniprot.retrieve
import waldo.mgi.retrieve
from waldo.go.go import id_to_term

for source in [waldo.uniprot.retrieve, waldo.mgi.retrieve]:
    internal = source.from_ensembl_peptide_id('ENSMUSP00000116259')
    ids = source.retrieve_go_annotations(internal)
    terms = map(id_to_term, ids)
    for t in terms:
        print t
```

You can see that we used the fact that both source implement the same interface by looping over the sources. We also used one of the other support mechanisms in waldo, namely the `waldo.go` module which provides functionality for generic GO processing, in this case, mapping from an ID (which is a numeric string, such as GO:0005634 to its corresponding term, *nucleus*).

Other functionality is in generic identifier translation and implementation of the MGI GO slim (a GO slim is a subset of GO and a mapping from the full GO to this subset). Additionally, it has information on homogology relationships and retrieval of amino-acid sequences (Muller et al., 2010)

Discussion

A widely reported problem in the provision of online databases is that they often become unavailable after publication (Veretnik et al., 2008; Wren, 2004, 2008). Our system implementation is designed to overcome this problem. The source code is available and will run immediately on a modern Linux server. It requires only packages that are part of the debian distribution and its derivatives (the same is likely to be true of other distributions, but the system was developed on Ubuntu, a debian derivative).

Furthermore, for heavy usage, it is recommended that the user use a local installation to avoid saturating network links.

A.2 jug

Jug is a task-based framework for Python, which supports saving and sharing of intermediate results and parallelisation on computer clusters (or multi-core machines).

Task Based Architecture

Jug is designed around tasks. A task is defined as a Python function and a set of arguments, which may be Python values or the output of other tasks. A task should be a pure function, but there is no way of mandating this in the language.

```
from jug import Task

def count(imname):
    ...
```

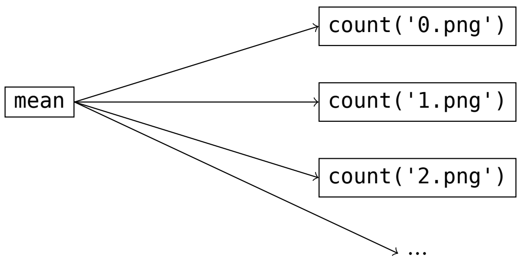


Figure A.1: Simple Dependency Structure for Example in the Text. This assumes that the directory had a collection of images names 0.png, 1.png,...

```
    return value

def mean(args):
    return sum(args)/float(len(args))

images = glob('*.png')
counts = [Task(count, im) for im in images]
final = Task(mean, counts)
```

This defines the task dependency structure represented in Figure A.1. As we can see, all of the `count` operations can be run in parallel, while the `mean` operation must wait the result of all of the other computation. The dependency structure is always a DAG (directed acyclic graph). Only by bypassing the normal use mechanisms is it even possible to construct a cycle.

The code above has the construct `Task(f, args)` repeated several times. Using the decorator `TaskGenerator` decorator this can be simplified to a more natural syntax.

```
from jug import TaskGenerator

@TaskGenerator
def count(imname):
    ...
    return value

@TaskGenerator
def mean(args):
    return sum(args)/float(len(args))
```

```
$jug status images.py
Task name          Waiting      Ready   Finished    Running
-----------------------------------------------------------
images.mean              1          0          0          0
images.count             0         20          0          0
...........................................................
Total:                   1         20          0          0
```

Figure A.2: Output of `jug status`. The $ sign shown is the command line prompt, and the status subcommand was run. At this point, nothing has been run. The output has been edited for space reasons (spacing columns were removed).

```
images = glob('*.png')
counts = map(count, images)
final = mean(counts)
```

As the reader can appreciate, this is identical to a traditional Python script, except for the `@TaskGenerator` decorators.

By default, jug looks for a file called `jugfile.py`, but any filename can be used. Generically, we refer to the script being run as the jugfile.

Jug subcommands

Jug is structured as a series of subcommands, the most important of which are *execute*, *status*, and *shell*.

Execution is the command used for running the tasks. In broad terms it performs the loop below.

```
tasks = alltasks in topological sort order
while tasks:
    next = task.pop()
    if not next.has_run() and not next.is_running():
        while not next.can_run():
            wait a while
        with locked(next):
            next.run()
```

If run on a single processor, this will just run all of the tasks in order. It is most interesting when it is run on multiple processors. There it executes all of the tasks that can be executed in parallel.

Of course, the actual code is more complex than what is shown above, particularly to make sure that the locking is performed correctly and that the waiting step eventually times out (in order to handle the situation where another process is hung).

The status command prints out a summary of the status of all of the tasks. Figure A.2 shows the output of this command using the example jugfile above. We assume that the jugfile was called `images.py` on disk and that there were 20 images in the directory. We can see that there are 20 tasks ready to run, while the `mean` task is still waiting for the results of the other tasks.

Backends

A basic feature of jug is its ability to save and load results. Each task `Task(f, args)` is represented by a hash of `f` and `args` in a way that uniquely identifies it.

A jug backend must then support four basic operations:

save Saving a Python object by its hash name.

load Loading a Python object by its hash name.

lock Creating a lock by hash name. Naturally, this lock must be created atomically.

release Releasing the lock.

A few other operations, such as deletion and listing of names are also supported.

The filesystem can support all of the above operations if coded correctly to avoid race conditions. This is the default backend, identified simply by a directory name. Inside this directory, files named by a hexadecimal representation of their hashes. Objects are saved using Python's `pickle` module with zlib compression. As a special case, `numpy` arrays are saved to disk directly. For functionality, there was no need for this inelegant special case, but `numpy` arrays are a very common data type in scientific programming and saving them directly allows for very fast saving and loading (they are represented on disk as a header followed by the binary information they contain).

Another backend currently included with jug is a redis backend. Redis a name-key database system.[1] Redis is particularly recommended for the case where there are many small objects being saved. In this case, keeping each as a separate file on disk would incur a large space penalty, while redis keeps them all in the same file.

Finally, there is an in-memory backend. This was initially developed for testing, but can be useful on its own.

Results can be obtained from jug in two ways: (1) One can simply write a task that outputs the desired format. (2) They can be inspected interactively using the *shell* subcommand.

The shell subcommand, invokes an IPython instance with all the objects in the jugfile loaded. IPython is an enhanced interactive shell for Python (Perez and Granger, 2007). A few functions are added to the namespace, in particular, `value` can load the results of a task object if necessary.

[1]See the redis webpage, at redis.io, for detailed information about redis.

```
$jug shell images.py

=========
Jug Shell
=========


Available jug functions:
    - value() : loads a specific object
    - load_all() : loads all objects

Enjoy...


In [1]: value(final)
Out[1]: 7.5

In [2]: value(counts)
Out[2]: [7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8]
```

Figure A.3: Interaction with `jug shell`.

Figure A.3 shows a possible interaction session with jug shell. While having to explicitly load all the results is a bit bothersome, it has some advantages versus the alternative of pre-loading all. It is much faster at startup. Consider too that the user might not load more than a few objects. In some cases, loading all of the objects simultaneously might even be impossible due to memory constraints. Further, this allows exploration of the task structure for debugging.

Jug includes a full test suite. There are no known bugs.

Jug has been available in the Python Package Index[2] since May 2009 and has been downloaded over 6000 times (some of these downloads may, of course, represent upgrades, and users which downloaded it from some other source will not be counted).

A.3 mahotas

Mahotas is a computer vision library for Python. It operates on numpy arrays, but its inner loops are implemented in C++ for speed and ease of implementation.

Functionality and Interface

The interface is a procedural interface, with most functions working independently of each other (there is code sharing at the implementation level).

[2]The Python Package Index, PyPI, is accessible at `http://pypi.python.org`.

features Some feature descriptors. In particular, Haralick texture features, Zernike moments, local binary patterns, threshold adjacency statistics.

morphological functions Erosion and dilation, as well as some more complex operations.

watershed seeded watershed.

thresholding some thresholding methods.

polygon operations convex hull, polygon drawing.

SURF Speeded-up Robust Features.

There are a few interface conventions which apply to many functions.

When meaningful, a structuring element is used to define neighbourhoods or adjacency relationships (some functions which use this concept are: watershed, erosion, and median filtering). Generally, the default is to use a 3×3 cross as the default if no structuring filter is given (the exception to this rule is the median filter, where the default is a 3×3 square).

Often, functions take an argument named `output` where the output will be stored. This argument is often much more restricted in type. In particular, it must often be a contiguous array.[3] Since this is a performance feature, it is natural that the interface is less flexible (accessing a contiguous array is much more efficient than a non-contiguous one).

Implementation

Mahotas is written in C++, but almost always, the user calls a Python function which checks types and then calls the internal function. This is slightly slower, but it is easier to develop this way.

The main reason that mahotas is in C++ (and not in pure C) is to use templates. Almost C++ functionality is split across 2 functions:

1. A `py_function` which uses the Python C API to get arguments and check them. This is almost always pure C.

2. A template `function<dtype>` which works for the type `dtype` performing the actual operation.

So, for example, this is how *erode* is implemented. `py_erode` consists mostly of boiler-plate code:

```
PyObject* py_erode(PyObject* self, PyObject* args) {
    PyArrayObject* array;
```

[3]Numpy supports non-contiguous arrays, which are most often slices into other, larger, contiguous arrays (e.g., given a 128×128 contiguous array, one can build a 64×128 non-contiguous array by taking every other row).

```
    PyArrayObject* Bc;
    if (!PyArg_ParseTuple(args,"OO", &array, &Bc)) {
        return NULL;
    }
    PyArrayObject* res_a = (PyArrayObject*)PyArray_SimpleNew(
                                array->nd,
                                array->dimensions,
                                PyArray_TYPE(array));
    if (!res_a) return NULL;
    PyArray_FILLWBYTE(res_a, 0);
    switch(PyArray_TYPE(array)) {
#define HANDLE(type) \
    erode<type>(numpy::aligned_array<type>(res_a), \
                numpy::aligned_array<type>(array), \
                numpy::aligned_array<type>(Bc));

        HANDLE_INTEGER_TYPES();
#undef HANDLE
    ...
```

These functions normally contain a lot of boiler-plate code: read the arguments, perform some sanity checks, perhaps a bit of initialisation, and then, the switch on the input type with the help of the `HANDLE_INTEGER_TYPES()`, `HANDLE_FLOAT_TYPE()`, and `HANDLE_TYPES()` macros, which call the right specialisation of the template that does the actual work. In this example `erode` implements (binary) erosion:

```
template<typename T>
void erode(numpy::aligned_array<T> res,
            numpy::aligned_array<T> array,
            numpy::aligned_array<T> Bc) {
    gil_release nogil;
    const unsigned N = res.size();
    typename numpy::aligned_array<T>::iterator iter = array.begin();
    filter_iterator<T> filter(res.raw_array(), Bc.raw_array());
    const unsigned N2 = filter.size();
    T* rpos = res.data();
    for (int i = 0;
                i != N;
                ++i, ++rpos, filter.iterate_both(iter)) {
        for (int j = 0; j != N2; ++j) {
            T arr_val = false;
            filter.retrieve(iter, j, arr_val);
            if (filter[j] && !arr_val) {
                goto skip_this_one;
            }
        }
        *rpos = true;
        skip_this_one: continue;
    }
}
```

The template machinery is not that complicated and the functions using it are very simple and easy to read. The only downside is that there is some expansion of code size when the compiler instanciates the function for the several integer and floating point types. Given the small size of these functions, this is not a big issue.

In the snippet above, you can see some other C++ machinery:

`gil_release` This is a "resource-acquisition is object initialisation" (RAII)[4] object that release the Python global interpreter lock (GIL)[5] in its constructor and gets it back in its destructor. Normally, the template function will release the GIL after the Python-specific code is done.

`array` This is a thin wrapper around `PyArrayObject` that knows its data type and has iterators which resemble the C++ standard library.

`filter_iterator` This is taken from `scipy.ndimage` and it is useful to iterate over an image and use a centered filter around each pixel (it keeps track of all of the boundary conditions).

The inner loop is as direct an implementation of erosion as one would wish for: for each pixel in the image, look at its neighbours. If all are true, then set the corresponding output pixel to **true** (else, skip it as it has been initialised to zero).

[4] RAII is a design pattern in C++, or other languages with scope linked deterministic object destruction, such as D, where a resource is represented by an object, whose constructor acquires it and whose destructor releases it. This guarantees that the object is correctly released even if the scope is left through an exception (Stroustrup, 1994).

[5] In the CPython interpreter, the most commonly used implementation of Python, there is a global lock for many Python related functionality, which limits parallelism.

Most of the functions follow this architecture.

Mahotas includes a full test suite. Every function has at least one corresponding test case. There are no known bugs.

It has been available in the Python Package Index since April 2010 and has been downloaded over 5500 times. Mahotas is available as a FreeBSD package and is under evaluation for inclusion in the Debian distribution (which will cause it to be available as a native package for its derivatives, most famously Ubuntu).

Acknowledgements

Mahotas includes code ported and incorporated from other projects. In particular, the SURF implementation is a port from the code from *dlib*,[6] a very good C++ library by Davis King. I also gleaned some insight into the implementation of these features from Christopher Evan's OpenSURF library and its documentation (**evans2009**).[7] The image loading code, which interfaces with the FreeImage library, was written by Zachary Pincus and some of the support code was written by Peter J. Verveer for the `scipy.ndimage` project. All of these contributions were integrated while respecting the software licenses under which the original code had been released. Robert Webb, a summer student, worked with me on the local binary patterns implementation (see Section 2.1 for a definition of LBP). Finally, I thank the several users who have reported bugs and submitted small fixes and participated on the project mailing list.

A.4 milk

Milk, short for *machine learning toolkit* is a collection of Python machine learning utilities. Like mahotas, it is written in a mix of Python and C++. There are two main submodules: the *supervised* and the *unsupervised* modules.

In the unsupervised module, the interface is functional with the following major features:

kmeans k-means. This is a version of k-means optimised for medium-sized datasets (millions of data points, but fitting in memory).

Self organising maps Simple implementation of self-organising maps.

Non-negative matrix factorisation Implementation of the algorithm by D. D. Lee and H. Seung (2001) for non-negative matrix factorisation and the extension of Hoyer (2004), which includes sparsity constraints.

Affinity propagation This is an implementation of the message passing algorithm of Frey and Dueck (2007). The implementation in milk was taken from `scikit.learn`, where it had initially been written by Alexandre Gramfort

[6]Dlib's webpage is at `http://dlib.net`.

[7]OpenSURF is available at `http://www.chrisevansdev.com/computer-vision-opensurf.html`, where several documents describe details of the implementation.

and Gaël Varoquaux. My changes were all space optimisations. Python using numeric libraries leads to a very expressive and flexible language. Still, one must be careful to take into account the memory usage and intermediate object allocations.

The supervised modules have a compositional object-oriented interface. There is a basic division between *learners* and *models*. A learner is an object that, given labeled data, can learn a model. A model can then be applied to new data to return a label. The word “classifier” is avoided as it is often used to refer to either a learner of a model.

The basic learner interface is the following function:

`train` Takes features and labels and returns the appropriate model.

Models have two functions:

`apply` Given a single example, return its label.

`apply_many` Given a sequence of examples, return a list of their labels. The default implementation of `model.apply_many(exs)` is simply `map(model.apply, exs)`. In some cases, however, it can be slightly more efficient to directly implement several applications at once.

Most other machine learning packages combine this functionality into a single object. I find that interface to be inferior (I used it in the first versions of *milk* partly following the examples I had observed). Normally the implementation looks something like the following:

```
class classifier(object):
    def __init__(self):
        self.trained = False
    def train(self, features, labels):
        # perform training
        self.trained = True

    def apply(self, f):
        if not self.trained:
            raise Exception(...)
        else:
            # perform actual computation
```

The need to always verify the `trained` attribute is error prone and is inelegant. The type system is a better place to check this functionality (this would be even more valuable in a statically typed language, where the mistake of attempting to call `apply` before `train` would be impossible; in Python, the check is still at runtime). Furthermore, while writing code in the above model (e.g., the multi-class adaptors described below), there is a need to use factory objects. While in Python they can simply be implemented with a callable such as a lambda function, it is still a more awkward interface and less composable than the current architecture.

Simple learners are combined to form more complex ones. For example, there are several strategies which take a binary classifier and transform it into a multi-class classifier. They are implemented as different adaptors that take a binary learner as input and return a multi-class learner. All adaptors have a `base` attribute which points to a base learner which they adapt.

The supervised module supports the following features:

Support vector machines Support vector machines, based on libsvm (Chang and C.-J. Lin, 2001).

Random forests Simple binary random forests (Breiman, 2001).

Logistic regression Gradient descent logistic regression. If the `scipy.optimize` module is present, it is used; otherwise, milk falls back on a hand-written Python gradient descent routine, which is correct, but slow.

Adaptors The previously mentioned binary to multi-class adaptors, but also adaptors for multiple views on the data (where the classification is obtained by combining the models in several ways).

Feature selection using stepwise discriminant analysis (Jennrich, 1977a,b).

For ease of use, the `defaultlearner` function returns a flexible multi-class learner that performs support vector machine learning after feature normalization and selection with with stepwise discriminant analysis. The kernel used is a radial basis function. Both the width of the kernel and the penalty value (often called C in the svm literature) are learned using internal cross-validation (a simple algorithm will avoid computing more than is strictly necessary to determine the best combination of parameters so that, it is likely that not all folds will be computed for all combinations). This was the classifier used throughout this work.

The cross-validation support in milk is extensive. In particular, it support assigning each item an origin, so that two elements which share an origin will never be separated (i.e., you will never test on an object if objects with the same origin were part of the training set). I make use of this feature in the current work, when using cross-validation over clones, while keeping the evaluation over individual images. The results are not exactly the same as simply choosing different clones as the algorithm takes into account how many images are available for each clone and attempts to balance the folds.

The learner which uses cross-validation to learn the best parameters, is itself a learner object, which returns a model. Therefore, it is trivial to perform two level cross validation, in the correct way (one outer level for accuracy estimation and an inner level to estimate the parameters for each fold). In fact, calling the `nfoldcrossvalidation` function without an explicit classifier defaults to this method.

Milk has support for parallel and distributed processing. It interfaces with *jug* to break up cross-validation into tasks, which can be run on different machines (there are no dependencies between different folds). Additionally, the inner cross validation for parameter estimation can be run on multiple cores on the same machine. Using these two features in combination, one can obtain a very high degree of parallelism.

Milk includes a full test suite. There are no known bugs.

Milk has been publicly available since April 2010. At the moment, it has been downloaded more than 9000 times.

Acknowledgements

Milk includes code from libsvm (Chang and C.-J. Lin, 2001) and *scikits.learn*,[8] both projects available under the BSD license. Several users have reported bugs and contributed fixes either on the public mailing list or by private email.

A.5 elgreco

The name El Greco originally stood for "Graphical Model Compiler" as the original implementation was as a Python module which internally compiled the graphical model to C++ and loaded it. The particular strategy used did not scale very well, however (for each node in the graph, a few lines of code were output: this means millions of lines of code for even medium sized models). Therefore, I abandoned this implementation strategy,[9] but the name remained.

It is now a direct C++ implementation of Gibbs sampling for the graphical model depicted in Figure 6.1 (page 61), in either collapsed or uncollapsed modes (using an objected oriented interface, so that it is easy to use either one or the other). There is a Python interface for convenience.

Unlike the modules presented above, this is very specific to the current project and, in its current form, likely to not be directly applicable in other problem settings (although one can certainly use it as a good, direct, SLDA implementation). Even though it is publicly available[10] (and has been available since the start), I have no report of it being used by others.

[8] SCIKITS.LEARN's webpage is at `http://scikit-learn.sourceforge.net/stable/`

[9] A less naïve strategy for the compiler could certainly work well and result in a very good tool for Bayesian machine learning, but developing and implementing one was outside of the goals of this work.

[10] The code can be obtained at `http://github.com/luispedro/elgreco`

§B Image Processing and Filtering

B.1 Image Filtering

The first step in image processing is to remove out-of-focus or otherwise irrelevant images.

The image filtering pipeline is a standard image classification pipeline, with the following features used:

1. Haralick texture features (Haralick et al., 1973),
2. local binary patterns (Ojala et al., 2002),
3. some special purpose features.

The special purpose features are the following two:

1. The ratio of the average value of the morphological gradient to the average value of the input image.[1]
2. The ratio of the variance of adjacent pixel differences to the variance of the pixel values.

The two features attempt to capture the fact that empty or out-of-focus images are smooth white noise without sharp constrasts, while the border between the nucleus and the background, as well as the texture inside a in focus nucleus provides edges.

[1] I thank Ṭaráz Buck for suggesting this feature to me.

	Haralick	LBPs	Haralick+LBPs	Haralick+Other	All
Accuracy (%)	88	73	85	**92**	89

Table B.1: Out of Focus Detection. Shown are the estimated generalisation accuracies based on different sets of features: (a) only the Haralick features, (b) only the LBP features, (c) Haralick and LBP, and (d) Haralick and LBP and our features. In all cases, a support vector machine classifier was used with a radial basis function kernel and accuracy is estimated through cross-validation.

Classification is performed using the default method of the milk library described in Section A.4. All accuracies reported in this work are estimated using cross-validation. In this case, 10 folds were used.

Results are shown in Table B.1. There were 1244 blurred images and 1319 in-focus images in this dataset, so the estimates are tight.

Using LBP brings down the accuracy and the best set is obtained by using Haralick texture features and our specially designed features.

B.2 Subcellular Location Features

As described in Section 2.1, the most widely used representation for bioimages is as a set of features. I made some contributions to this field.

One texture set that we used were Threshold Adjacency Statistics (TAS) proposed by Hamilton, Pantelic, et al. (2007). In the original work, there are two hard-coded values, both set to 30, but logically distinct. The first is the image threshold, used to distinguish foreground from background, which we will call T. The second we call the margin m. The general procedure is illustrated as Algorithm 1.

Algorithm 1: Threshold Adjacency Statistics.

```
Input: An image img of size $N \times M$
Input: A threshold $t$
Input: A margin $m$
Output: TAS values in the $\lambda_i$ array
$\mu$ := mean(img $> T$);
bin := $\mu - m <$ img $< \mu + m$ ;
the comparisons are to be interpreted as pixelwise operations;
for $i \in \{0 \dots 8\}$ do
    $\lambda_i := 0$;
for $i, j \in \{1 \dots N-1, 1 \dots M-1\}$ do
    $c := 0$;
    for $i', j' \in \{0 \dots N, 0 \dots M \mid |i - i'| \leq 1, |j - j'| \leq 1\}$ do
        $c := c + \text{bin}_{ij}$
    $\lambda_i := \lambda_i + c$;
for $i \in \{0 \dots 8\}$ do
    $\lambda_i := \lambda_i / (N * M)$;
```

There are two variations on the algorithm, obtained by replacing line 2 by either

$$\mathbf{bin} := \mathbf{img} < \mu + m$$

or

$$\mathbf{bin} := \mu - m < \mathbf{img}.$$

In order to use TAS on our images, where the illumination is not perfectly constant (and a single threshold is likely not possible), I wished to avoid the hard coded values. In a previous work, I had introduced parameter-free versions (PFTAS), defined by setting T by the automatic method of Ridler and Calvard (1978), and setting m to be the standard deviation of the above threshold pixels (Coelho, Ahmed, et al., 2010). It is this version that I used and the implementation in mahotas (Appendix A.3) defaults to it (the user must explicitly set a parameter in order to obtain the older, version).

Overlap Features

A reference channel, imaged in addition to the protein one, can carry a large amount of information on the localization of the protein. In our case, we imaged a nuclear marker (Hoechst), which is typical, although some research has imaged more than one reference channel (Barbe et al., 2008). Therefore, several features which use the reference channel have been proposed (J. Newberg and Murphy, 2008). In this work, I used some of them and introduce others.

For the definition of these features, I use both the raw values of the protein (which we will refer to as p_i) and reference (r_i), as well as binary versions, defined by:

$$B(p_i) = [\![\, p_i > T_p \,]\!]. \tag{B.1}$$

Where T_p is the automatically found threshold for the protein image. T_r and $B(r_i)$ are defined analogously. Additionally we make use of the distance map, where d_i is the euclidean distance to the nearest above threshold reference pixel:

$$d_i = \min\{d(j,i) \mid B(r_j)\}. \tag{B.2}$$

We define corr to be the correlation between two variables:

$$\mathrm{corr}(p,r) = \frac{\sum_i (p_i - \bar{p})(r_i - \bar{r})}{\hat{\sigma}_p \hat{\sigma}_r}, \tag{B.3}$$

where $\hat{\sigma}_p$ and $\hat{\sigma}_r$ refer to the estimated standard deviation of p and r, respectively.

The first features are simply estimates of different overlap properties:

$$\text{prot/ref overlap} = \frac{\sum_i B(p_i)}{\sum_i B(r_i)} \tag{B.4}$$

$$\begin{aligned}\text{fraction bin prot in bin ref} &= \frac{\sum_i B(r_i)B(p_i)}{\sum_i B(r_i)} \\ &\approx P(B(p_i)|B(r_i)),\end{aligned} \tag{B.5}$$

$$\begin{aligned}\text{fraction prot in bin ref} &= \frac{\sum_i B(r_i) * p_i}{\sum_u B(p_i)} \\ &\approx E[p_i|B(r_i)],\end{aligned} \tag{B.6}$$

$$\begin{aligned}\text{fraction of proc protein in bin ref} &= \frac{\sum_i B(p_i)B(r_i)p_i}{\sum_i B(p_i)p_i} \\ &\approx E[p_i|B(r_i) \wedge B(p_i)],\end{aligned} \tag{B.7}$$

$$\begin{aligned}\text{fraction of binary prot in binary ref} &= \frac{\sum_i B(r_i)B(p_i)}{\sum_i B(p_i)} \\ &\approx P(B(r_i)|B(p_i)).\end{aligned} \tag{B.8}$$

I also measure correlations

$$\text{correlation of bin protein and bin ref} = \text{corr}(B(p_i), B(r_i)), \tag{B.9}$$

$$\text{correlation of proc protein and binary ref} = \text{corr}(p_i B(p_i), B(r_i)), \tag{B.10}$$

$$\text{correlation of protein and ref} = \text{corr}(p_i, r_i). \tag{B.11}$$

The distance map is used in the last features:

$$\text{median protein} \times \text{distance to ref} = \text{med}\left\{d_i B(p_i) p_i \mid B(p_i)\right\}, \tag{B.12}$$

$$\text{mean protein} \times \text{distance to ref} = \text{mean}\left\{d_i B(p_i) p_i \mid B(p_i)\right\}. \tag{B.13}$$

§C Results

This appendix provides the output of running the various learning processes on the whole colllection.

Table C.1: Results of 10 Class Classification based on the widefield dataset and local features as described in Section 5.3. Shown is the plurality class (there were several images per clone) and the confidence according to the rule $c = (s+1)/(n+2)$, where s is the number of images in the plurality class and n is the total number of images.

Well	Uniprot name	Location (confidence)	Uniprot C. C. Terms
CW1G5	D3Z6I7_MOUSE	cytoskeleton (96%)	
CZ1C10	LMNA_MOUSE	nuclear (96%)	lamin filament, nuclear envelope, perinuclear region of cytoplasm
CW1D10	AHSA1_MOUSE	nuclear (96%)	cytosol, endoplasmic reticulum
CW1G4	E9Q7Z2_MOUSE	nuclear (96%)	
CZ1C6	unknown	nucleoli (96%)	

continued on next page...

Results of 7 Class classification (continued from previous page).

Well	Uniprot name	Location (confidence)	Uniprot C. C. Terms
EZ3C3	Q8BMC5_MOUSE	nuclear (96%)	
CW1F8	CF064_MOUSE	nuclear (96%)	integral to membrane
EO2B7	PACN2_MOUSE	unlabeled (96%)	cytoplasmic membrane-bounded vesicle, cytosol
BV1G6	ENOX1_MOUSE	mitochondria (95%)	extracellular space, plasma membrane
EO2F10	NDUA7_MOUSE	unlabeled (95%)	
EO3B5	PDLI1_MOUSE	unlabeled (95%)	cytoplasm, cytoskeleton, transcription factor complex
BV2G6	A8Y5L4_MOUSE	cytoplasmic (94%)	
BT1C5	FND3B_MOUSE	nucleoli (93%)	integral to membrane
BM1B9	RS28_MOUSE	unlabeled (93%)	
CZ1F9	D6RE43_MOUSE	nucleoli (92%)	
EZ3E3	B3RH23_MOUSE	nuclear (92%)	
CZ1G5	LMNA_MOUSE	nuclear (92%)	lamin filament, nuclear envelope, perinuclear region of cytoplasm
CZ1C2	LMNA_MOUSE	nuclear (92%)	lamin filament, nuclear envelope, perinuclear region of cytoplasm
CW1E8	LMNA_MOUSE	nuclear (92%)	lamin filament, nuclear envelope, perinuclear region of cytoplasm
EZ3C8	Q9ESU7_MOUSE	unlabeled (92%)	
FB1D3	CO5A2_MOUSE	unlabeled (92%)	
FB2B3	COSA1_MOUSE	unlabeled (92%)	basement membrane, collagen
CY1B11	TKT_MOUSE	cytoplasmic (92%)	
FX1D5	Q8BMC5_MOUSE	nuclear (92%)	
EZ3C10	Q7TMN5_MOUSE	cytoskeleton (92%)	

continued on next page...

Results of 7 Class classification (continued from previous page).

Well	Uniprot name	Location (confidence)	Uniprot C. C. Terms
DA1G11	SET_MOUSE	nuclear (92%)	cytosol, endoplasmic reticulum, nucleoplasm, perinuclear region of cytoplasm
CW1B2	D3Z6I7_MOUSE	cytoskeleton (92%)	
EZ3F5	B3RH23_MOUSE	nuclear (92%)	
BM1G3	HMGA2_MOUSE	nuclear (91%)	chromatin, nuclear chromosome
EO2E10	CO5A1_MOUSE	unlabeled (90%)	basement membrane
CY1B5	Q3U741_MOUSE	nucleoli (88%)	
CW1C11	D3Z6I7_MOUSE	unlabeled (88%)	
CU1E5	E9Q5N9_MOUSE	unlabeled (87%)	
BU1E7	B9EIV2_MOUSE	nuclear (86%)	
CU1D10	Q8C6C1_MOUSE	unlabeled (85%)	
CZ1G8	MDGA2_MOUSE	cytoskeleton (85%)	anchored to membrane, plasma membrane
CZ1E9	D3Z6I7_MOUSE	cytoskeleton (85%)	
FB1C3	DHX57_MOUSE	unlabeled (85%)	
FX2F9	Q7TMN5_MOUSE	cytoskeleton (85%)	
FX2D2	Q7TMN5_MOUSE	cytoskeleton (84%)	
BV2B3	BTBD9_MOUSE	cytoplasmic (83%)	
BU1D8	SPAG7_MOUSE	lysosome (82%)	nucleus
FX2F11	Q8BMC5_MOUSE	nuclear (82%)	
BV1D6	FGL1_MOUSE	nuclear (81%)	extracellular space
EZ3E8	Q7TMN5_MOUSE	cytoskeleton (81%)	
FB1B8	PRRX1_MOUSE	nuclear (81%)	nucleus
CZ1E4	MDGA2_MOUSE	cytoskeleton (81%)	anchored to membrane, plasma membrane
CW1C5	LMNA_MOUSE	nuclear (81%)	lamin filament, nuclear envelope, perinuclear region of cytoplasm
FX2B11	Q8BMC5_MOUSE	unlabeled (80%)	

continued on next page...

Results of 7 Class classification (continued from previous page).

Well	Uniprot name	Location (confidence)	Uniprot C. C. Terms
BW2D10	B2RUB9_MOUSE	nuclear (80%)	
EM1D10	B3RH23_MOUSE	nuclear (79%)	
EO2D7	Q3TKX9_MOUSE	unlabeled (78%)	
EZ3B3	F10A1_MOUSE	unlabeled (77%)	cytoplasm
FB1D10	DHE3_MOUSE	mitochondria (77%)	mitochondrial inner membrane, mitochondrial matrix
CZ1B10	E9QKD6_MOUSE	cytoplasmic (76%)	
EZ3F11	SMD1_MOUSE	unlabeled (76%)	cytoplasm
BV2C8	NAA20_MOUSE	nuclear (76%)	cytoplasm, nucleus
EO3F6	WDR43_MOUSE	unlabeled (76%)	nucleolus
BM1D7	ARP2_MOUSE	cytoplasmic (75%)	cell projection, cytoplasm, cytoskeleton
CY1D8	Q8BG11_MOUSE	unlabeled (75%)	
EZ3E2	TPM4_MOUSE	unlabeled (75%)	cortical cytoskeleton, podosome
EZ3F7	PSA5_MOUSE	unlabeled (75%)	cytoplasm, nucleus, proteasome core complex, alpha-subunit complex
EM1E7	B3RH23_MOUSE	nuclear (75%)	
EZ3D6	PDLI1_MOUSE	unlabeled (74%)	cytoplasm, cytoskeleton, transcription factor complex
EZ3G10	KDM6B_MOUSE	membrane (74%)	nucleus
FB1E9	Q3UAM6_MOUSE	unlabeled (73%)	
EO2B6	RS26_MOUSE	nuclear (73%)	ribosome
EM1E9	DAP1_MOUSE	unlabeled (73%)	
FX1F3	Q8BMC5_MOUSE	nuclear (73%)	
BU2G3	E9QLD6_MOUSE	cytoplasmic (72%)	
FX2C8	TERA_MOUSE	cytoplasmic (72%)	cytosol, endoplasmic reticulum, microsome, nucleus

continued on next page...

Results of 7 Class classification (continued from previous page).

Well	Uniprot name	Location (confidence)	Uniprot C. C. Terms
FX1E9	ARPC2_MOUSE	mitochondria (72%)	Arp2/3 protein complex, cell leading edge, cell projection, focal adhesion
BU2C3	TPD53_MOUSE	cytoplasmic (71%)	perinuclear region of cytoplasm
EX1C5	PDLI1_MOUSE	unlabeled (71%)	cytoplasm, cytoskeleton, transcription factor complex
EO2G7	Q3TVJ3_MOUSE	nucleoli (70%)	
EO3B7	ANXA5_MOUSE	cytoplasmic (69%)	
FB1C6	PRRX1_MOUSE	nuclear (66%)	nucleus
BT1B3	E9PVJ2_MOUSE	cytoskeleton (66%)	
BT1B10	E9PUR9_MOUSE	cytoplasmic (66%)	
BV1E8	GRM8_MOUSE	nuclear (66%)	integral to plasma membrane
FX2C10	Q8BMC5_MOUSE	unlabeled (66%)	
EM1G9	RAC1_MOUSE	unlabeled (66%)	cytosol, extrinsic to plasma membrane, lamellipodium, melanosome, membrane fraction
EO2G4	Q9CQ38_MOUSE	unlabeled (66%)	
EO2F7	DDX3X_MOUSE	nuclear (66%)	cytoplasm, nuclear speck
EZ3E6	ALR_MOUSE	cytoplasmic (66%)	mitochondrial intermembrane space
FX1E2	Q7TMN5_MOUSE	cytoskeleton (66%)	
EO3F11	RL18_MOUSE	unlabeled (66%)	ribosome
CY1D9	CTP5C_MOUSE	nuclear (65%)	integral to membrane
EO2G9	LRC59_MOUSE	unlabeled (64%)	endoplasmic reticulum membrane, integral to membrane, microsome
CY1G8	E9Q941_MOUSE	cytoplasmic (62%)	
FX2E7	Q7TMN5_MOUSE	cytoskeleton (62%)	
EZ3D10	Q9CRA2_MOUSE	unlabeled (62%)	

continued on next page...

Results of 7 Class classification (continued from previous page).

Well	Uniprot name	Location (confidence)	Uniprot C. C. Terms
EX1C4	PTRF_MOUSE	lysosome (61%)	caveola, cytosol, endoplasmic reticulum, microsome, mitochondrion, nucleus
CU1G4	Q8C6C1_MOUSE	unlabeled (61%)	
BU2G8	E2F3_MOUSE	cytoplasmic (60%)	transcription factor complex
FX1F4	SUH_MOUSE	unlabeled (60%)	cytoplasm, transcription factor complex
DA2C4	IF4B_MOUSE	golgi (60%)	
EO2F9	Q9CQL3_MOUSE	cytoplasmic (60%)	
CY1F6	B1ARS0_MOUSE	cytoplasmic (59%)	
BV2E5	Q8CG29_MOUSE	cytoplasmic (58%)	
BU2B10	HN1L_MOUSE	cytoplasmic (56%)	cytoplasm, nucleus
CZ1F10	E9QKV3_MOUSE	cytoplasmic (55%)	
CW1G10	PCKGM_MOUSE	nuclear (55%)	mitochondrion
FX2B3	ARPC2_MOUSE	unlabeled (55%)	Arp2/3 protein complex, cell leading edge, cell projection, focal adhesion
CZ1F5	SRS10_MOUSE	unlabeled (55%)	nuclear speck
FX2B10	Q8R1B7_MOUSE	unlabeled (55%)	
BW2G3	PACN2_MOUSE	unlabeled (54%)	cytoplasmic membrane-bounded vesicle, cytosol
BV1D7	A2A8Q4_MOUSE	nuclear (53%)	
CX1B6	D3YTP8_MOUSE	unlabeled (52%)	
EM3E7	Q7TMN5_MOUSE	cytoskeleton (52%)	
EO2E3	VIME_MOUSE	mitochondria (52%)	cell leading edge, cytoplasm, type III intermediate filament
CU1E2	Q8C6C1_MOUSE	nuclear (50%)	
DA1C10	E9QNV9_MOUSE	unlabeled (50%)	

continued on next page...

Results of 7 Class classification (continued from previous page).

Well	Uniprot name	Location (confidence)	Uniprot C. C. Terms
BM1F6	ARPC4_MOUSE	cytoplasmic (50%)	Arp2/3 protein complex, cell projection, cytoplasm
BU1G9	PCKGM_MOUSE	cytoplasmic (50%)	mitochondrion
BV2B10	PTHB1_MOUSE	cytoplasmic (50%)	cilium membrane, cytoplasm
BU1E10	D3Z6I7_MOUSE	cytoplasmic (50%)	
BW1D9	NUP93_MOUSE	unlabeled (50%)	nuclear pore
BM1C5	A2API5_MOUSE	cytoplasmic (50%)	
FY4C9	ACTN4_MOUSE	membrane (50%)	cortical cytoskeleton, pseudopodium, ribonucleoprotein complex, stress fiber
BU2G5	ADK_MOUSE	nuclear (50%)	cytosol, nucleus
DA2G11	TIF1A_MOUSE	unlabeled (50%)	cytoplasm, nuclear euchromatin, perichromatin fibrils
EZ3D9	Q05CR3_MOUSE	mitochondria (48%)	
CW1D7	GSHR_MOUSE	unlabeled (48%)	external side of plasma membrane, mitochondrion
BV1D3	CK5P2_MOUSE	unlabeled (48%)	Golgi apparatus, microtubule, pericentriolar material, perinuclear region of cytoplasm, spindle pole
EM1D2	Q8BG13_MOUSE	nucleoli (47%)	
BV2B5	GRIP1_MOUSE	unlabeled (46%)	cell junction, cytoplasmic membrane-bounded vesicle, endoplasmic reticulum, membrane raft, postsynaptic membrane
CU1B2	Q8C6C1_MOUSE	nuclear (45%)	
FY4F10	HDGF_MOUSE	membrane (44%)	cytoplasm, extracellular space, nucleus
CZ1D11	LMNA_MOUSE	cytoplasmic (44%)	lamin filament, nuclear envelope, perinuclear region of cytoplasm

continued on next page...

Results of 7 Class classification (continued from previous page).

Well	Uniprot name	Location (confidence)	Uniprot C. C. Terms
EO3G4	LPP_MOUSE	cytoplasmic (44%)	cytoplasm, nucleus
CZ1D7	D3Z6I7_MOUSE	cytoplasmic (44%)	
EM3G6	SC61B_MOUSE	cytoplasmic (44%)	endoplasmic reticulum membrane, integral to membrane
FB1F10	SEPT9_MOUSE	unlabeled (44%)	
FX2E6	SND1_MOUSE	cytoplasmic (44%)	melanosome, mitochondrion, nucleus, RNA-induced silencing complex
BV1D10	PIGU_MOUSE	nuclear (44%)	GPI-anchor transamidase complex
FX1C6	Q7TMN5_MOUSE	cytoplasmic (43%)	
DA1D10	E9QNV9_MOUSE	unlabeled (42%)	
EO3B4	Q3UCI5_MOUSE	nucleoli (41%)	
EX1B8	MCM2_MOUSE	cytoskeleton (40%)	MCM complex, nuclear origin of replication recognition complex
CW1E4	AHSA1_MOUSE	nuclear (40%)	cytosol, endoplasmic reticulum
DA1C4	IF4B_MOUSE	cytoplasmic (38%)	
CZ1B3	E9QPA6_MOUSE	unlabeled (38%)	
CW1F10	LMNA_MOUSE	unlabeled (37%)	lamin filament, nuclear envelope, perinuclear region of cytoplasm
EZ3F4	ENSMUSP00000020078	cytoskeleton (34%)	
DA1D11	IF4B_MOUSE	membrane (33%)	
EO2G5	ACOD1_MOUSE	golgi (33%)	endoplasmic reticulum membrane, integral to membrane
EZ3C9	Q3UUT0_MOUSE	cytoskeleton (33%)	
FB1C7	TIMP3_MOUSE	cytoplasmic (33%)	basement membrane
FX1E10	MACD2_MOUSE	mitochondria (33%)	
EO3D5	RL18_MOUSE	golgi (31%)	ribosome

Table C.2: Results of 10 Class Classification based on the topic model and both the widefield and confocal labeled datasets as described in Section 6.2.

Well	Uniprot name	Location (confidence)	Uniprot C. C. Terms
CW1G5	D3Z6I7_MOUSE	cytoskeleton (96%)	
CZ1C10	LMNA_MOUSE	nuclear (96%)	lamin filament, nuclear envelope, perinuclear region of cytoplasm
CW1D10	AHSA1_MOUSE	nuclear (96%)	cytosol, endoplasmic reticulum
CW1G4	E9Q7Z2_MOUSE	nuclear (96%)	
CZ1C6	unknown	nucleoli (96%)	
EZ3C3	Q8BMC5_MOUSE	nuclear (96%)	
CW1F8	CF064_MOUSE	nuclear (96%)	integral to membrane
EO2B7	PACN2_MOUSE	unlabeled (96%)	cytoplasmic membrane-bounded vesicle, cytosol
BV1G6	ENOX1_MOUSE	mitochondria (95%)	extracellular space, plasma membrane
EO2F10	NDUA7_MOUSE	unlabeled (95%)	
EO3B5	PDLI1_MOUSE	unlabeled (95%)	cytoplasm, cytoskeleton, transcription factor complex
BV2G6	A8Y5L4_MOUSE	cytoplasmic (94%)	
BT1C5	FND3B_MOUSE	nucleoli (93%)	integral to membrane
BM1B9	RS28_MOUSE	unlabeled (93%)	
CZ1F9	D6RE43_MOUSE	nucleoli (92%)	
EZ3E3	B3RH23_MOUSE	nuclear (92%)	
CZ1G5	LMNA_MOUSE	nuclear (92%)	lamin filament, nuclear envelope, perinuclear region of cytoplasm
CZ1C2	LMNA_MOUSE	nuclear (92%)	lamin filament, nuclear envelope, perinuclear region of cytoplasm

continued on next page...

Results of 7 Class classification (continued from previous page).

Well	Uniprot name	Location (confidence)	Uniprot C. C. Terms
CW1E8	LMNA_MOUSE	nuclear (92%)	lamin filament, nuclear envelope, perinuclear region of cytoplasm
EZ3C8	Q9ESU7_MOUSE	unlabeled (92%)	
FB1D3	CO5A2_MOUSE	unlabeled (92%)	
FB2B3	COSA1_MOUSE	unlabeled (92%)	basement membrane, collagen
CY1B11	TKT_MOUSE	cytoplasmic (92%)	
FX1D5	Q8BMC5_MOUSE	nuclear (92%)	
EZ3C10	Q7TMN5_MOUSE	cytoskeleton (92%)	
DA1G11	SET_MOUSE	nuclear (92%)	cytosol, endoplasmic reticulum, nucleoplasm, perinuclear region of cytoplasm
CW1B2	D3Z6I7_MOUSE	cytoskeleton (92%)	
EZ3F5	B3RH23_MOUSE	nuclear (92%)	
BM1G3	HMGA2_MOUSE	nuclear (91%)	chromatin, nuclear chromosome
EO2E10	CO5A1_MOUSE	unlabeled (90%)	basement membrane
CY1B5	Q3U741_MOUSE	nucleoli (88%)	
CW1C11	D3Z6I7_MOUSE	unlabeled (88%)	
CU1E5	E9Q5N9_MOUSE	unlabeled (87%)	
BU1E7	B9EIV2_MOUSE	nuclear (86%)	
CU1D10	Q8C6C1_MOUSE	unlabeled (85%)	
CZ1G8	MDGA2_MOUSE	cytoskeleton (85%)	anchored to membrane, plasma membrane
CZ1E9	D3Z6I7_MOUSE	cytoskeleton (85%)	
FB1C3	DHX57_MOUSE	unlabeled (85%)	
FX2F9	Q7TMN5_MOUSE	cytoskeleton (85%)	
FX2D2	Q7TMN5_MOUSE	cytoskeleton (84%)	
BV2B3	BTBD9_MOUSE	cytoplasmic (83%)	
BU1D8	SPAG7_MOUSE	lysosome (82%)	nucleus

continued on next page...

Results of 7 Class classification (continued from previous page).

Well	Uniprot name	Location (confidence)	Uniprot C. C. Terms
FX2F11	Q8BMC5_MOUSE	nuclear (82%)	
BV1D6	FGL1_MOUSE	nuclear (81%)	extracellular space
EZ3E8	Q7TMN5_MOUSE	cytoskeleton (81%)	
FB1B8	PRRX1_MOUSE	nuclear (81%)	nucleus
CZ1E4	MDGA2_MOUSE	cytoskeleton (81%)	anchored to membrane, plasma membrane
CW1C5	LMNA_MOUSE	nuclear (81%)	lamin filament, nuclear envelope, perinuclear region of cytoplasm
FX2B11	Q8BMC5_MOUSE	unlabeled (80%)	
BW2D10	B2RUB9_MOUSE	nuclear (80%)	
EM1D10	B3RH23_MOUSE	nuclear (79%)	
EO2D7	Q3TKX9_MOUSE	unlabeled (78%)	
EZ3B3	F10A1_MOUSE	unlabeled (77%)	cytoplasm
FB1D10	DHE3_MOUSE	mitochondria (77%)	mitochondrial inner membrane, mitochondrial matrix
CZ1B10	E9QKD6_MOUSE	cytoplasmic (76%)	
EZ3F11	SMD1_MOUSE	unlabeled (76%)	cytoplasm
BV2C8	NAA20_MOUSE	nuclear (76%)	cytoplasm, nucleus
EO3F6	WDR43_MOUSE	unlabeled (76%)	nucleolus
BM1D7	ARP2_MOUSE	cytoplasmic (75%)	cell projection, cytoplasm, cytoskeleton
CY1D8	Q8BG11_MOUSE	unlabeled (75%)	
EZ3E2	TPM4_MOUSE	unlabeled (75%)	cortical cytoskeleton, podosome
EZ3F7	PSA5_MOUSE	unlabeled (75%)	cytoplasm, nucleus, proteasome core complex, alpha-subunit complex
EM1E7	B3RH23_MOUSE	nuclear (75%)	
EZ3D6	PDLI1_MOUSE	unlabeled (74%)	cytoplasm, cytoskeleton, transcription factor complex

continued on next page...

Results of 7 Class classification (continued from previous page).

Well	Uniprot name	Location (confidence)	Uniprot C. C. Terms
EZ3G10	KDM6B_MOUSE	membrane (74%)	nucleus
FB1E9	Q3UAM6_MOUSE	unlabeled (73%)	
EO2B6	RS26_MOUSE	nuclear (73%)	ribosome
EM1E9	DAP1_MOUSE	unlabeled (73%)	
FX1F3	Q8BMC5_MOUSE	nuclear (73%)	
BU2G3	E9QLD6_MOUSE	cytoplasmic (72%)	
FX2C8	TERA_MOUSE	cytoplasmic (72%)	cytosol, endoplasmic reticulum, microsome, nucleus
FX1E9	ARPC2_MOUSE	mitochondria (72%)	Arp2/3 protein complex, cell leading edge, cell projection, focal adhesion
BU2C3	TPD53_MOUSE	cytoplasmic (71%)	perinuclear region of cytoplasm
EX1C5	PDLI1_MOUSE	unlabeled (71%)	cytoplasm, cytoskeleton, transcription factor complex
EO2G7	Q3TVJ3_MOUSE	nucleoli (70%)	
EO3B7	ANXA5_MOUSE	cytoplasmic (69%)	
FB1C6	PRRX1_MOUSE	nuclear (66%)	nucleus
BT1B3	E9PVJ2_MOUSE	cytoskeleton (66%)	
BT1B10	E9PUR9_MOUSE	cytoplasmic (66%)	
BV1E8	GRM8_MOUSE	nuclear (66%)	integral to plasma membrane
FX2C10	Q8BMC5_MOUSE	unlabeled (66%)	
EM1G9	RAC1_MOUSE	unlabeled (66%)	cytosol, extrinsic to plasma membrane, lamellipodium, melanosome, membrane fraction
EO2G4	Q9CQ38_MOUSE	unlabeled (66%)	
EO2F7	DDX3X_MOUSE	nuclear (66%)	cytoplasm, nuclear speck
EZ3E6	ALR_MOUSE	cytoplasmic (66%)	mitochondrial intermembrane space
FX1E2	Q7TMN5_MOUSE	cytoskeleton (66%)	

continued on next page...

Results of 7 Class classification (continued from previous page).

Well	Uniprot name	Location (confidence)	Uniprot C. C. Terms
EO3F11	RL18_MOUSE	unlabeled (66%)	ribosome
CY1D9	CTP5C_MOUSE	nuclear (65%)	integral to membrane
EO2G9	LRC59_MOUSE	unlabeled (64%)	endoplasmic reticulum membrane, integral to membrane, microsome
CY1G8	E9Q941_MOUSE	cytoplasmic (62%)	
FX2E7	Q7TMN5_MOUSE	cytoskeleton (62%)	
EZ3D10	Q9CRA2_MOUSE	unlabeled (62%)	
EX1C4	PTRF_MOUSE	lysosome (61%)	caveola, cytosol, endoplasmic reticulum, microsome, mitochondrion, nucleus
CU1G4	Q8C6C1_MOUSE	unlabeled (61%)	
BU2G8	E2F3_MOUSE	cytoplasmic (60%)	transcription factor complex
FX1F4	SUH_MOUSE	unlabeled (60%)	cytoplasm, transcription factor complex
DA2C4	IF4B_MOUSE	golgi (60%)	
EO2F9	Q9CQL3_MOUSE	cytoplasmic (60%)	
CY1F6	B1ARS0_MOUSE	cytoplasmic (59%)	
BV2E5	Q8CG29_MOUSE	cytoplasmic (58%)	
BU2B10	HN1L_MOUSE	cytoplasmic (56%)	cytoplasm, nucleus
CZ1F10	E9QKV3_MOUSE	cytoplasmic (55%)	
CW1G10	PCKGM_MOUSE	nuclear (55%)	mitochondrion
FX2B3	ARPC2_MOUSE	unlabeled (55%)	Arp2/3 protein complex, cell leading edge, cell projection, focal adhesion
CZ1F5	SRS10_MOUSE	unlabeled (55%)	nuclear speck
FX2B10	Q8R1B7_MOUSE	unlabeled (55%)	
BW2G3	PACN2_MOUSE	unlabeled (54%)	cytoplasmic membrane-bounded vesicle, cytosol
BV1D7	A2A8Q4_MOUSE	nuclear (53%)	

continued on next page...

Results of 7 Class classification (continued from previous page).

Well	Uniprot name	Location (confidence)	Uniprot C. C. Terms
CX1B6	D3YTP8_MOUSE	unlabeled (52%)	
EM3E7	Q7TMN5_MOUSE	cytoskeleton (52%)	
EO2E3	VIME_MOUSE	mitochondria (52%)	cell leading edge, cytoplasm, type III intermediate filament
CU1E2	Q8C6C1_MOUSE	nuclear (50%)	
DA1C10	E9QNV9_MOUSE	unlabeled (50%)	
BM1F6	ARPC4_MOUSE	cytoplasmic (50%)	Arp2/3 protein complex, cell projection, cytoplasm
BU1G9	PCKGM_MOUSE	cytoplasmic (50%)	mitochondrion
BV2B10	PTHB1_MOUSE	cytoplasmic (50%)	cilium membrane, cytoplasm
BU1E10	D3Z6I7_MOUSE	cytoplasmic (50%)	
BW1D9	NUP93_MOUSE	unlabeled (50%)	nuclear pore
BM1C5	A2API5_MOUSE	cytoplasmic (50%)	
FY4C9	ACTN4_MOUSE	membrane (50%)	cortical cytoskeleton, pseudopodium, ribonucleoprotein complex, stress fiber
BU2G5	ADK_MOUSE	nuclear (50%)	cytosol, nucleus
DA2G11	TIF1A_MOUSE	unlabeled (50%)	cytoplasm, nuclear euchromatin, perichromatin fibrils
EZ3D9	Q05CR3_MOUSE	mitochondria (48%)	
CW1D7	GSHR_MOUSE	unlabeled (48%)	external side of plasma membrane, mitochondrion
BV1D3	CK5P2_MOUSE	unlabeled (48%)	Golgi apparatus, microtubule, pericentriolar material, perinuclear region of cytoplasm, spindle pole
EM1D2	Q8BG13_MOUSE	nucleoli (47%)	

continued on next page...

Results of 7 Class classification (continued from previous page).

Well	Uniprot name	Location (confidence)	Uniprot C. C. Terms
BV2B5	GRIP1_MOUSE	unlabeled (46%)	cell junction, cytoplasmic membrane-bounded vesicle, endoplasmic reticulum, membrane raft, postsynaptic membrane
CU1B2	Q8C6C1_MOUSE	nuclear (45%)	
FY4F10	HDGF_MOUSE	membrane (44%)	cytoplasm, extracellular space, nucleus
CZ1D11	LMNA_MOUSE	cytoplasmic (44%)	lamin filament, nuclear envelope, perinuclear region of cytoplasm
EO3G4	LPP_MOUSE	cytoplasmic (44%)	cytoplasm, nucleus
CZ1D7	D3Z6I7_MOUSE	cytoplasmic (44%)	
EM3G6	SC61B_MOUSE	cytoplasmic (44%)	endoplasmic reticulum membrane, integral to membrane
FB1F10	SEPT9_MOUSE	unlabeled (44%)	
FX2E6	SND1_MOUSE	cytoplasmic (44%)	melanosome, mitochondrion, nucleus, RNA-induced silencing complex
BV1D10	PIGU_MOUSE	nuclear (44%)	GPI-anchor transamidase complex
FX1C6	Q7TMN5_MOUSE	cytoplasmic (43%)	
DA1D10	E9QNV9_MOUSE	unlabeled (42%)	
EO3B4	Q3UCI5_MOUSE	nucleoli (41%)	
EX1B8	MCM2_MOUSE	cytoskeleton (40%)	MCM complex, nuclear origin of replication recognition complex
CW1E4	AHSA1_MOUSE	nuclear (40%)	cytosol, endoplasmic reticulum
DA1C4	IF4B_MOUSE	cytoplasmic (38%)	
CZ1B3	E9QPA6_MOUSE	unlabeled (38%)	
CW1F10	LMNA_MOUSE	unlabeled (37%)	lamin filament, nuclear envelope, perinuclear region of cytoplasm
EZ3F4	ENSMUSP00000020078	cytoskeleton (34%)	

continued on next page...

Results of 7 Class classification (continued from previous page).

Well	Uniprot name	Location (confidence)	Uniprot C. C. Terms
DA1D11	IF4B_MOUSE	membrane (33%)	
EO2G5	ACOD1_MOUSE	golgi (33%)	endoplasmic reticulum membrane, integral to membrane
EZ3C9	Q3UUT0_MOUSE	cytoskeleton (33%)	
FB1C7	TIMP3_MOUSE	cytoplasmic (33%)	basement membrane
FX1E10	MACD2_MOUSE	mitochondria (33%)	
EO3D5	RL18_MOUSE	golgi (31%)	ribosome

Bibliography

Ahmed, Amr, Eric P. Xing, William W. Cohen, and Robert F. Murphy (2009). "Structured correspondence topic models for mining captioned figures in biological literature". In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*, p. 39. [DOI] (cit. on p. 59).

Ahuja, Narendra and Sinisa Todorovic (Oct. 2007). "Learning the Taxonomy and Models of Categories Present in Arbitrary Images". In: *2007 IEEE 11th International Conference on Computer Vision* October, pp. 1–8. ISSN: 1550-5499. [DOI] (cit. on p. 16).

Ashburner, M, C A Ball, J A Blake, D Botstein, H Butler, J M Cherry, A P Davis, K Dolinski, S S Dwight, J T Eppig, M A Harris, D P Hill, L Issel-Tarver, A Kasarskis, S Lewis, J C Matese, J E Richardson, M Ringwald, G M Rubin, and G Sherlock (May 2000). "Gene ontology: tool for the unification of biology. The Gene Ontology Consortium." In: *Nature genetics* 25.1, pp. 25–9. ISSN: 1061-4036. [DOI] (cit. on p. 68).

Asuncion, Arthur, Max Welling, and Padhraic Smyth (2008). "On Smoothing and Inference for Topic Models". In: 24.Ml, pp. 1–8 (cit. on p. 20).

Aturaliya, Rajith N, J Lynn Fink, Melissa J Davis, Melvena S Teasdale, Kelly a Hanson, Kevin C Miranda, Alistair R R Forrest, Sean M Grimmond, Harukazu Suzuki, Mutsumi Kanamori, Chikatoshi Kai, Jun Kawai, Piero Carninci, Yoshihide Hayashizaki, and Rohan D Teasdale (May 2006). "Subcellular localization of mammalian type II membrane proteins." In: *Traffic (Copenhagen, Denmark)* 7.5, pp. 613–25. ISSN: 1398-9219. [DOI] (cit. on p. 55).

Bae, Kyounghwa and Bani K Mallick (Dec. 2004). “Gene selection using a two-level hierarchical Bayesian model.” In: *Bioinformatics (Oxford, England)* 20.18, pp. 3423–30. ISSN: 1367-4803. [DOI] (cit. on p. 60).

Bamford, P (2003). “Empirical comparison of cell segmentation algorithms using an annotated dataset”. In: *Image Processing, 2003. ICIP 2003. Proc. 2003 International Conference on*. Vol. 2, [Online Version] (cit. on pp. 26, 27).

Barbe, Laurent, Emma Lundberg, Per Oksvold, Anna Stenius, Erland Lewin, Erik Björling, Anna Asplund, Fredrik Pontén, Hjalmar Brismar, Mathias Uhlén, and Helene Andersson-Svahn (2008). “Toward a confocal subcellular atlas of the human proteome.” In: *Molecular & cellular proteomics : MCP* 7.3, pp. 499–508. ISSN: 1535-9484. [DOI] (cit. on pp. 17, 93).

Bay, Herbert (2006). “From Wide-baseline Point and Line Correspondences to 3D”. PhD. Swissh Federal Institute of Technology, ETH Zurich (cit. on p. 14).

Bay, Herbert, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool (June 2008). “Speeded-Up Robust Features (SURF)”. In: *Computer Vision and Image Understanding* 110.3, pp. 346–359. ISSN: 10773142. [DOI] (cit. on pp. 3, 14, 41, 50).

Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool (2006). “SURF: Speeded Up Robust Features”. In: *9th European Conference on Computer Vision*. Graz: Springer, pp. 404–417. [DOI] (cit. on p. 14).

Bernard, Allister and Alexander J Hartemink (Jan. 2005). “Informative structure priors: joint learning of dynamic regulatory networks from multiple types of data.” In: *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pp. 459–70. [Online Version] (cit. on p. 18).

Beucher, Serge and Christian Lantuéjoul (Sept. 1979). “Use of watersheds in contour detection”. In: *International Workshop on Image Processing*. Rennes: CCETT, pp. 2.1–2.12. [Online Version] (cit. on p. 23).

Bhattacharya, A., V. Ljosa, M.R. Verardo, C. Faloutsos, and A.K. Singh (2005). “ViVo: Visual Vocabulary Construction for Mining Biomedical Images”. In: *Fifth IEEE International Conference on Data Mining (ICDM’05)*. IEEE, pp. 50–57. ISBN: 0-7695-2278-5. [DOI] (cit. on p. 14).

Blake, Judith A, Carol J Bult, James A Kadin, Joel E Richardson, and Janan T Eppig (Jan. 2011). “The Mouse Genome Database (MGD): premier model organism resource for mammalian genomics and genetics.” In: *Nucleic acids research* 39.Database issue, pp. D842–8. ISSN: 1362-4962. [DOI] (cit. on p. 68).

Blei, David M. and Michael I. Jordan (2003). “Modeling annotated data”. In: *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval - SIGIR ’03*, p. 127. [DOI] (cit. on pp. 20, 59, 60).

Blei, David M. and Jon D. Mcauliffe (2007). “Supervised topic models”. In: *Neural Information Processing Systems*, pp. 1–8 (cit. on p. 20).

Blei, David M., Andrew Y. Ng, and Michael I. Jordan (2003). “Latent dirichlet allocation”. In: *The Journal of Machine Learning Research* 3, pp. 993–1022. [Online Version] (cit. on pp. 18, 19, 33, 34, 62).

Boland, Michael V., Mia K. Markey, and Robert F. Murphy (Nov. 1998). “Automated recognition of patterns characteristic of subcellular structures in fluorescence microscopy images.” In: *Cytometry* 33.3, pp. 366–75. ISSN: 0196-4763. [Online Version] (cit. on pp. 8, 10, 56).

Boland, Michael V. and Robert F. Murphy (Dec. 2001). “A neural network classifier capable of recognizing the patterns of all major subcellular structures in fluorescence microscope images of HeLa cells.” In: *Bioinformatics (Oxford, England)* 17.12, pp. 1213–23. ISSN: 1367-4803. [Online Version] (cit. on pp. 8, 11, 56).

Breiman, Leo (2001). “Random Forests”. In: *Machine Learning* 45.1, pp. 5–32. [DOI] (cit. on p. 89).

Briesemeister, Sebastian, Torsten Blum, Scott Brady, Yin Lam, Oliver Kohlbacher, and Hagit Shatkay (Nov. 2009). “SherLoc2: a high-accuracy hybrid method for predicting subcellular localization of proteins.” In: *Journal of proteome research* 8.11, pp. 5363–6. ISSN: 1535-3907. [DOI] (cit. on pp. 18, 75).

Canny, John (Nov. 1986). “A Computational Approach to Edge Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6, pp. 679–698. ISSN: 0162-8828. [DOI] (cit. on p. 12).

Chang, Chih-Chung and Chih-Jen Lin (2001). “LIBSVM: a library for support vector machines”. In: *Computer*, pp. 1–30. [DOI] (cit. on pp. 89, 90).

Chebira, Amina, Yann Barbotin, Charles Jackson, Thomas Merryman, Gowri Srinivasa, Robert F. Murphy, and Jelena Kovacević (Jan. 2007). “A multiresolution approach to automated classification of protein subcellular location images.” In: *BMC bioinformatics* 8.1, p. 210. ISSN: 1471-2105. [DOI] (cit. on pp. 10, 56).

Chen, Cheng, John a. Ozolek, Wei Wang, and Gustavo K. Rohde (2011). “A General System for Automatic Biomedical Image Segmentation Using Intensity Neighborhoods”. In: *International Journal of Biomedical Imaging* 2011, pp. 1–12. ISSN: 1687-4188. [DOI] (cit. on p. 28).

Chennubhotla, Chakra and Allan Jepson (2001). “Sparse PCA. Extracting multi-scale structure from data”. In: *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, pp. 641–647. [DOI] (cit. on p. 17).

Chen, Shann-Ching, Geoffrey J. Gordon, and Robert F. Murphy (2008). “Graphical Models for Structured Classification, with an Application to Interpreting Images of Protein Subcellular Location Patterns”. In: *Journal of Machine Learning Research* 9, pp. 651–682. [DOI] (cit. on p. 75).

Chen, Shann-Ching, Ting Zhao, Geoffrey J Gordon, and Robert F. Murphy (2007). “Automated image analysis of protein localization in budding yeast.” In: *Bioinformatics (Oxford, England)* 23.13, pp. i66–71. ISSN: 1367-4811. [DOI] (cit. on pp. 1, 8, 12, 17).

Chen, Xiang and Robert F. Murphy (2005). "Objective clustering of proteins based on subcellular location patterns." In: *Journal of biomedicine & biotechnology* 2005.2, pp. 87–95. ISSN: 1110-7243. [DOI] (cit. on p. 17).

Chen, Xiang, Meel Velliste, Shmuel Weinstein, Jon W. Jarvik, and Robert F. Murphy (2003). "Location proteomics-Building subcellular location trees from high resolution 3D fluorescence microscope images of randomly-tagged proteins". In: *SPIE*. Vol. 4962, pp. 298–306. [Online Version] (cit. on p. 17).

Coelho, Luis Pedro, Amr Ahmed, Andrew Arnold, Joshua Kangas, Abdul-Saboor Sheikh, Eric P. Xing, William W. Cohen, and Robert F. Murphy (Jan. 2010). "Structured Literature Image Finder: Extracting Information from Text and Images in Biomedical Literature." In: *Lecture notes in computer science* 6004, pp. 23–32. ISSN: 0302-9743. [DOI] (cit. on pp. 75, 93).

Coelho, Luis Pedro and Robert F. Murphy (2009). "Unsupervised Unmixing of Subcellular Location Patterns". In: *ICML-UAI-COLT 2009 Workshop on Automated Interpretation and Modeling of Cell Images (Cell Image Learning)*. Montreal, Canada. [Online Version].

Coelho, Luis Pedro, Tao Peng, and Robert F. Murphy (2010). "Quantifying the distribution of probes between subcellular locations using unsupervised pattern unmixing". In: *Bioinformatics* 26.12, pp. i7–i12. [DOI] (cit. on pp. 31, 37–40).

Coelho, Luis Pedro, Aabid Shariff, and Robert F. Murphy (2009). "Nuclear segmentation in microscope cell images: A hand-segmented dataset and comparison of algorithms". In: *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*. IEEE, pp. 518–521. ISBN: 978-1-4244-3931-7. [DOI] (cit. on p. 24).

Crow, Franklin C. (1984). "Summed area tables for texture mapping". In: *Computer Graphics SIGGRAPH proceedings* 11.3, pp. 200–220 (cit. on p. 14).

Csurka, Gabriella, C Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray (2004). "Visual categorization with bags of keypoints". In: *Workshop on Statistical Learning in Computer Vision, ECCV*. Vol. 1. In Workshop on Statistical Learning in Computer Vision, ECCV May. Citeseer, pp. 1–22 (cit. on p. 34).

Dalal, Navneet and Bill Triggs (2005). "Histograms of Oriented Gradients for Human Detection". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. IEEE, pp. 886–893. ISBN: 0-7695-2372-2. [DOI] (cit. on p. 14).

Danckaert, A., E. Gonzalez-Couto, L. Bollondi, N. Thompson, and B. Hayes (2002). "Automated recognition of intracellular organelles in confocal microscope images." In: *Traffic* 3.1, pp. 66–73. [DOI] (cit. on p. 7).

Deerwester, Scott, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman (1990). "Indexing by Latent Semantic Analysis". In: *Journal of the American Society for Information Science* 41.6, pp. 391–407. ISSN: 00028231. [DOI] (cit. on p. 19).

Drawid, Amar and Mark Gerstein (Aug. 2000). “A Bayesian system integrating expression data with sequence patterns for localizing proteins: comprehensive application to the yeast genome.” In: *Journal of molecular biology* 301.4, pp. 1059–75. ISSN: 0022-2836. [DOI] (cit. on p. 18).

Du, Lan, Lu Ren, David B Dunson, and Lawrence Carin (2009). “A Bayesian Model for Simultaneous Image Clustering, Annotation and Object Segmentation”. In: *Advances in Neural Information Processing Systems*, pp. 486–494 (cit. on p. 21).

Fei-Fei, Li and Pietro Perona (2005). “A Bayesian Hierarchical Model for Learning Natural Scene Categories”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. IEEE, pp. 524–531. ISBN: 0-7695-2372-2. [DOI] (cit. on p. 14).

Frey, Brendan J. and Delbert Dueck (2007). “Clustering by passing messages between data points.” In: *Science* 315.5814, pp. 972–976. [Online Version] (cit. on p. 87).

Fyshe, Alona, Yifeng Liu, Duane Szafron, Russ Greiner, and Paul Lu (2008). “Improving subcellular localization prediction using text classification and the gene ontology.” In: *Bioinformatics (Oxford, England)* 24.21, pp. 2512–7. ISSN: 1367-4811. [DOI] (cit. on p. 18).

García Osuna, Elvira, Juchang Hua, Nicholas W Bateman, Ting Zhao, Peter B. Berget, and Robert F. Murphy (June 2007). “Large-scale automated analysis of location patterns in randomly tagged 3T3 cells.” In: *Annals of biomedical engineering* 35.6, pp. 1081–7. ISSN: 0090-6964. [DOI] (cit. on pp. 2, 3).

Glory, Estelle and Robert F. Murphy (2007). “Automated subcellular location determination and high-throughput microscopy.” In: *Developmental cell* 12.1, pp. 7–16. ISSN: 1534-5807. [DOI] (cit. on p. 23).

Griffiths, Thomas L and Mark Steyvers (Apr. 2004). “Finding scientific topics.” In: *Proceedings of the National Academy of Sciences of the United States of America* 101 Suppl, pp. 5228–35. ISSN: 0027-8424. [DOI] (cit. on pp. 20, 62, 64).

Hamilton, Nicholas A., Radosav S. Pantelic, Kelly Hanson, and Rohan D. Teasdale (2007). “Fast automated cell phenotype image classification.” In: *BMC bioinformatics* 8, p. 110. ISSN: 1471-2105. [DOI] (cit. on p. 92).

Hamilton, Nicholas A., Jack T. H. Wang, Markus C. Kerr, and Rohan D. Teasdale (2009). “Statistical and visual differentiation of subcellular imaging.” In: *BMC bioinformatics* 10, p. 94. ISSN: 1471-2105. [DOI] (cit. on pp. 2, 17).

Haralick, Robert M., Its’hak Dinstein, and K. Shanmugam (1973). “Textural features for image classification”. In: *Ieee Transactions On Systems Man And Cybernetics* 3.6, pp. 610–621. [Online Version] (cit. on pp. 8, 91).

Heinrich, Gregory (2009). *Parameter estimation for text analysis.* Tech. rep. Fraunhofer IGD. [Online Version] (cit. on p. 64).

Hinton, Geoffrey E. (2007). “To recognize shapes, first learn to generate images”. In: *Computational Neuroscience: Theoretical Insights into Brain Function*. Ed. by Trevor Drew Paul Cisek and John F. Kalaska. Vol. 165. Progress in Brain Research. Elsevier, pp. 535–547. [DOI] (cit. on p. 7).

Hofmann, Thomas (1999). “Probabilistic latent semantic indexing”. In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR ’99*, pp. 50–57. [DOI] (cit. on p. 18).

Hoyer, Patrik O. (2004). “Non-negative matrix factorization with sparseness constraints”. In: *Journal of Machine Learning Research* 5, pp. 1457–1469. [Online Version] (cit. on pp. 33, 87).

Huang, Kai and Robert F. Murphy (2004a). *Automated classification of subcellular patterns in multicell images without segmentation into single cells*. IEEE, pp. 1139–1142. ISBN: 0-7803-8388-5. [DOI] (cit. on p. 8).

— (2004b). “Boosting accuracy of automated classification of fluorescence microscope images for location proteomics”. In: *BMC Bioinformatics* 5.1, p. 78. ISSN: 1471-2105. [DOI] (cit. on p. 56).

Huh, Seungil, Donghun Lee, and Robert F. Murphy (2009). “Efficient framework for automated classification of subcellular patterns in budding yeast.” In: *Cytometry. Part A : the journal of the International Society for Analytical Cytology* 75.11, pp. 934–40. ISSN: 1552-4930. [DOI] (cit. on p. 14).

Huh, Won-Ki, James V Falvo, Luke C Gerke, Adam S Carroll, Russell W Howson, Jonathan S Weissman, and Erin K O’Shea (Oct. 2003). “Global analysis of protein localization in budding yeast.” In: *Nature* 425.6959, pp. 686–91. ISSN: 1476-4687. [DOI] (cit. on pp. 1, 12, 17).

Jarvik, Jon W., S A Adler, C A Telmer, V Subramaniam, and A J Lopez (1996). “CD-tagging: a new approach to gene and protein discovery and analysis.” In: *BioTechniques* 20.5, pp. 896–904. ISSN: 0736-6205. [Online Version] (cit. on p. 3).

Jarvik, Jon W., G W Fisher, C Shi, L Hennen, C Hauser, S Adler, and Peter B. Berget (2002). “In vivo functional proteomics: mammalian genome annotation using CD-tagging.” In: *BioTechniques* 33.4, 852–4, 856, 858–60 passim. ISSN: 0736-6205. [Online Version] (cit. on p. 3).

Jennrich, R. I. (1977a). “Stepwise Discriminant Analysis”. In: *Statistical Methods for Digital Computers*. Ed. by K. Enslein, A. Ralston, and H. Wilf. John Wiley & Sons (cit. on p. 89).

— (1977b). “Stepwise Regression”. In: *Statistical Methods for Digital Computerss*. Ed. by K. Enslein, A. Ralston, and H. Wilf. John Wiley & Sons (cit. on p. 89).

Jones, Thouis R., Anne Carpenter, and Polina Golland (2005). "Voronoi-Based Segmentation of Cells on Image Manifolds". In: *Computer Vision for Biomedical Image Applications*. Ed. by Yanxi Liu, Tianzi Jiang, and Changshui Zhang. Vol. 3765. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 535–543. ISBN: 978-3-540-29411-5 (cit. on p. 23).

Kanade, Takeo (1977). "Model Representations and Control Structures in Image Understanding". In: *5th International Joint Conference on Artificial Intelligence*, pp. 1074–1082 (cit. on p. 16).

Keränen, Soile V. E., Charless C. Fowlkes, Cris L. Luengo Hendriks, Damir Sudar, David W. Knowles, Jitendra Malik, and Mark D. Biggin (2006). "Three-dimensional morphology and gene expression in the Drosophila blastoderm at cellular resolution II: dynamics." In: *Genome biology* 7.12, R124. ISSN: 1465-6914. [DOI] (cit. on p. 17).

Kumar, Anuj, Seema Agarwal, John A Heyman, Sandra Matson, Matthew Heidtman, Stacy Piccirillo, Lara Umansky, Amar Drawid, Ronald Jansen, Yang Liu, Kei-Hoi Cheung, Perry Miller, Mark Gerstein, G Shirleen Roeder, and Michael Snyder (2002). "Subcellular localization of the yeast proteome." In: *Genes & development* 16.6, pp. 707–19. ISSN: 0890-9369. [DOI] (cit. on p. 18).

Lazebnik, Svetlana, Cordelia Schmid, and Jean Ponce (2006). "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories". In: *CVPR* (cit. on p. 16).

Lee, Daniel D. and H. Sebastian Seung (Oct. 1999). "Learning the parts of objects by non-negative matrix factorization." In: *Nature* 401.6755, pp. 788–91. ISSN: 0028-0836. [DOI] (cit. on p. 33).

Lee, Daniel D. and H.Sebastian Seung (2001). "Algorithms for non-negative matrix factorization". In: *Advances in neural information processing systems*. Vol. 13. 1. Citeseer, pp. 556–562. [Online Version] (cit. on pp. 33, 87).

Lee, George, Scott Doyle, James Monaco, Anant Madabhushi, Michael D. Feldman, Stephen R. Master, and John E. Tomaszewski (2009). "A knowledge representation framework for integration, classification of multi-scale imaging and non-imaging data: Preliminary results in predicting prostate cancer recurrence by fusing mass spectrometry and histology". In: *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*. IEEE, pp. 77–80. ISBN: 978-1-4244-3931-7. [DOI] (cit. on p. 18).

Lee, Honglak, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng (2009). "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations". In: *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*. New York, New York, USA: ACM Press, pp. 1–8. ISBN: 9781605585161. [DOI] (cit. on p. 7).

Lee, Kiyoung, Han-Yu Chuang, Andreas Beyer, Min-Kyung Sung, Won-Ki Huh, Bonghee Lee, and Trey Ideker (2008). "Protein networks markedly improve prediction of subcellular localization in multiple eukaryotic species." In: *Nucleic acids research* 36.20, e136. ISSN: 1362-4962. [DOI] (cit. on p. 18).

Lehmussola, Antti, Pekka Ruusuvuori, Jyrki Selinummi, Tiina Rajala, and Olli Yli-Harja (Aug. 2008). “Synthetic Images of High-Throughput Microscopy for Validation of Image Analysis Methods”. In: *Proceedings of the IEEE* 96.8, pp. 1348–1360. ISSN: 0018-9219. [DOI] (cit. on p. 13).

Li, Li-Jia, Richard Socher, and Li Fei-Fei (June 2009). “Towards total scene understanding: Classification, annotation and segmentation in an automatic framework”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 2036–2043. ISBN: 978-1-4244-3992-8. [DOI] (cit. on p. 21).

Lindeberg, Tony (1998). “Feature Detection with Automatic Scale Selection”. In: *International Journal of Computer Vision* 30.2, pp. 77–116 (cit. on p. 15).

Lin, Gang, Umesh Adiga, Kathy Olson, John F Guzowski, Carol a Barnes, and Badrinath Roysam (2003). “A hybrid 3D watershed algorithm incorporating gradient cues and object models for automatic segmentation of nuclei in confocal image stacks.” In: *Cytometry. Part A : the journal of the International Society for Analytical Cytology* 56.1, pp. 23–36. ISSN: 1552-4922. [DOI] (cit. on pp. 25, 30).

Lowe, David G. (1999). “Object recognition from local scale-invariant features”. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision* 2.[8, 1150–1157 vol.2. [DOI] (cit. on p. 14).

Luengo Hendriks, Cris L, Soile V E Keränen, Charless C Fowlkes, Lisa Simirenko, Gunther H Weber, Angela H DePace, Clara Henriquez, David W Kaszuba, Bernd Hamann, Michael B Eisen, Jitendra Malik, Damir Sudar, Mark D Biggin, and David W Knowles (2006). “Three-dimensional morphology and gene expression in the Drosophila blastoderm at cellular resolution I: data acquisition pipeline.” In: *Genome biology* 7.12, R123. ISSN: 1465-6914. [DOI] (cit. on p. 17).

Lu, Z., D. Szafron, R. Greiner, P. Lu, D.S. Wishart, B. Poulin, J. Anvik, C. Macdonell, and R. Eisner (2004). “Predicting subcellular localization of proteins using machine-learned classifiers”. In: *Bioinformatics* 20.4, pp. 547–556. ISSN: 1460-2059. [DOI] (cit. on p. 18).

Marée, Raphaël, Pierre Geurts, and Louis Wehenkel (2007). “Random subwindows and extremely randomized trees for image classification in cell biology.” In: *BMC cell biology* 8 Suppl 1, S2. ISSN: 1471-2121. [DOI] (cit. on pp. 14, 56).

Mikolajczyk, Krystian and Cordelia Schmid (May 2002). “An Affine Invariant Interest Point Detector”. In: pp. 128–142. [Online Version] (cit. on p. 15).

Miller, Michael I. and Anqi Qiu (Mar. 2009). “The emerging discipline of Computational Functional Anatomy.” In: *NeuroImage* 45.1 Suppl, S16–39. ISSN: 1095-9572. [DOI] (cit. on p. 17).

Minka, Tom (2000). *Estimating a Dirichlet distribution*. Tech. rep. Microsoft Research. [Online Version] (cit. on p. 66).

Mintz-Oron, Shira, Asaph Aharoni, Eytan Ruppin, and Tomer Shlomi (June 2009). “Network-based prediction of metabolic enzymes’ subcellular localization.” In: *Bioinformatics (Oxford, England)* 25.12, pp. i247–52. ISSN: 1460-2059. [DOI] (cit. on p. 18).

Moosmann, Frank, Eric Nowak, and Frederic Jurie (Sept. 2008). “Randomized clustering forests for image classification.” In: *IEEE transactions on pattern analysis and machine intelligence* 30.9, pp. 1632–46. ISSN: 0162-8828. [DOI] (cit. on pp. 16, 21).

Muller, J, D Szklarczyk, P Julien, I Letunic, a Roth, M Kuhn, S Powell, C von Mering, T Doerks, L J Jensen, and P Bork (Jan. 2010). “eggNOG v2.0: extending the evolutionary genealogy of genes with enhanced non-supervised orthologous groups, species and functional annotations.” In: *Nucleic acids research* 38.Database issue, pp. D190–5. ISSN: 1362-4962. [DOI] (cit. on pp. 68, 79).

Murphy, Robert F. (June 2005). “Location proteomics: a systems approach to subcellular location.” In: *Biochemical Society transactions* 33.Pt 3, pp. 535–8. ISSN: 0300-5127. [DOI] (cit. on p. 1).

Murphy, Robert F., Meel Velliste, and Gregory Porreca (Nov. 2003). “Robust Numerical Features for Description and Classification of Subcellular Location Patterns in Fluorescence Microscope Images”. In: *The Journal of VLSI Signal Processing-Systems for Signal, Image, and Video Technology* 35.3, pp. 311–321. ISSN: 0922-5773. [DOI] (cit. on pp. 1, 12).

Nanni, Loris, Sheryl Brahnam, and Alessandra Lumini (Jan. 2010). “Novel features for automated cell phenotype image classification.” In: *Advances in experimental medicine and biology* 680, pp. 207–13. ISSN: 0065-2598. [DOI] (cit. on pp. 8, 56).

Nanni, Loris and Alessandra Lumini (June 2008). “A reliable method for cell phenotype image classification.” In: *Artificial intelligence in medicine* 43.2, pp. 87–97. ISSN: 0933-3657. [DOI] (cit. on p. 10).

Nattkemper, Tim W., Thorsten Twellmann, Helge Ritter, and Walter Schubert (Jan. 2003). “Human vs machine: evaluation of fluorescence micrographs.” In: *Computers in biology and medicine* 33.1, pp. 31–43. ISSN: 0010-4825. [Online Version] (cit. on p. 1).

Newberg, Justin and Robert F. Murphy (June 2008). “A framework for the automated analysis of subcellular patterns in human protein atlas images.” In: *Journal of proteome research* 7.6, pp. 2300–8. ISSN: 1535-3893. [DOI] (cit. on pp. 17, 93).

Newberg, Justin Y., Arvind Rao, Emma Lundberg, Fredrik Ponten, Mathias Uhlen, and Robert F. Murphy (2009). “Automated analysis of Human Protein Atlas immunofluorescence images”. In: *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*. IEEE, pp. 1023–1026. ISBN: 978-1-4244-3931-7. [DOI] (cit. on pp. 10, 17).

Newman, David, Arthur Asuncion, Padhraic Smyth, and Max Welling (2009). “Distributed Algorithms for Topic Models”. In: *Journal of Machine Learning Research* 10, pp. 1801–1828 (cit. on p. 62).

Nowak, Eric and F Jurie (2006). “Sampling strategies for bag-of-features image classification”. In: *Computer Vision–ECCV 2006* 3954, pp. 490–503. [DOI] (cit. on p. 14).

Ojala, Timo, Matti Pietikäinen, and Topi Mäenpää (July 2002). “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.7, pp. 971–987. ISSN: 0162-8828. [DOI] (cit. on pp. 10, 91).

Otsu, Nobuyuki (1979). “A Threshold Selection Method from Gray-Level Histograms”. In: *Ieee Transactions On Systems Man And Cybernetics* 9.1, pp. 62–66. ISSN: 00189472. [DOI] (cit. on p. 24).

Pan, Jia-Yu, André Guilherme, Ribeiro Balan, Eric P. Xing, Christos Faloutsos, and Agma Juci Machado Traina (2006). “Automatic mining of fruit fly embryo images”. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’06*. c. New York, New York, USA: ACM Press, p. 693. ISBN: 1595933395. [DOI] (cit. on p. 17).

Park, Trevor and George Casella (June 2008). “The Bayesian Lasso”. In: *Journal of the American Statistical Association* 103.482, pp. 681–686. ISSN: 0162-1459. [DOI] (cit. on p. 60).

Peng, Hanchuan, Fuhui Long, Michael B. Eisen, and Eugene W. Myers (2006). “Clustering gene expression patterns of fly embryos”. In: *Proceedings of the IEEE International Symposium on Biomedical Imaging (ISBI): Nano to Macro*, pp. 1144–1147. [DOI] (cit. on p. 17).

Peng, Hanchuan and Eugene W. Myers (2004). “Comparing in situ m RNA expression patterns of drosophila embryos”. In: *Proceedings of the eighth annual international conference on Computational molecular biology - RECOMB ’04*. New York, New York, USA: ACM Press, pp. 157–166. ISBN: 1581137559. [DOI] (cit. on p. 17).

Peng, Tao, Ghislain M. C. Bonamy, Estelle Glory-Afshar, Daniel R. Rines, Sumit K. Chanda, and Robert F. Murphy (2010). “Determining the distribution of probes between different subcellular locations through automated unmixing of subcellular patterns”. In: *Proceedings of the National Academy of Sciences of the United States of America* 107.7, pp. 2944–2949. [Online Version] (cit. on pp. 12, 23, 34).

Peng, Tao and Robert F. Murphy (May 2011). “Image-derived, three-dimensional generative models of cellular organization.” In: *Cytometry. Part A : the journal of the International Society for Analytical Cytology* 79.5, pp. 383–91. ISSN: 1552-4930. [DOI] (cit. on p. 13).

Perez, Fernando and Brian E. Granger (2007). “IPython: A System for Interactive Scientific Computing”. In: *Computing in Science & Engineering* 9.3, pp. 21–29. ISSN: 1521-9615. [DOI] (cit. on p. 82).

Philbin, James, Josef Sivic, and Andrew Zisserman (2008). “Geometric LDA: A Generative Model for Particular Object Discovery”. In: *Proceedings of the British Machine Vision Conference*. Leeds (cit. on pp. 16, 21, 34).

Pincus, Zachary and Julie A. Theriot (Aug. 2007). “Comparison of quantitative methods for cell-shape analysis.” In: *Journal of microscopy* 227.Pt 2, pp. 140–56. ISSN: 0022-2720. [DOI] (cit. on p. 17).

Porway, Jake, Benjamin Yao, and Song Chun Zhu (2008). “Learning Compositional Models for Object Categories From Small Sample Sets”. In: *Object Categorization: Computer and Human Vision Perspectives*. Ed. by Sven Dickinson, Bernt Schiele Aleš Leonardis, and Michael J. Tarr. Cambridge University Press, pp. 1–17 (cit. on p. 21).

Pritchard, J K, M Stephens, and P Donnelly (June 2000). “Inference of population structure using multilocus genotype data.” In: *Genetics* 155.2, pp. 945–59. ISSN: 0016-6731. [Online Version] (cit. on p. 19).

Rajapakse, Jagath C. (June 2008). “Protein localization on cellular images with Markov random fields”. In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 2127–2132. [DOI] (cit. on p. 56).

Rand, William M. (1971). “Objective Criteria for the Evaluation of Clustering Methods”. In: *Journal of the American Statistical Association* 66.336, pp. 846–850. ISSN: 01621459. [DOI] (cit. on p. 26).

Ridler, T. W. and S. Calvard (1978). “Picture Thresholding Using an Iterative Selection Method”. In: *IEEE Transactions On Systems Man And Cybernetics* 8.8, pp. 630–632. ISSN: 00189472. [DOI] (cit. on pp. 24, 30, 93).

Robert, Christian P. (June 1995). “Simulation of truncated normal variables”. In: *Statistics and Computing* 5.2, pp. 121–125. ISSN: 0960-3174. [DOI] (cit. on p. 63).

Rohde, Gustavo K., Alexandre J. S. Ribeiro, Kris N. Dahl, and Robert F. Murphy (2008). “Deformation-based nuclear morphometry: capturing nuclear shape variation in HeLa cells.” In: *Cytometry Part A* 73A.4, pp. 341–50. ISSN: 1552-4930. [DOI] (cit. on p. 17).

Saporta, Gilbert and Genane Youness (2002). “Comparing two partitions : Some Proposals and Experiments”. In: *Compstat Proceedings in Computational Statistics 15th Symposium Held in Berlin Germany 2002*. Physica Verlag, pp. 243–248. ISBN: 3790815179. [Online Version] (cit. on p. 26).

Segal, Eran and Daphne Koller (2002). “Probabilistic hierarchical clustering for biological data”. In: *Proceedings of the sixth annual international conference on Computational biology - RECOMB ’02*, pp. 273–280. [DOI] (cit. on p. 18).

Segal, Eran, Ben Taskar, Audrey Gasch, Nir Friedman, and Daphne Koller (Jan. 2001). “Rich probabilistic models for gene expression.” In: *Bioinformatics* 17 Suppl 1.1, S243–52. ISSN: 1367-4803. [DOI] (cit. on p. 18).

Shao, Yuanlong, Yuan Zhou, Xiaofei He, Deng Cai, and Hujun Bao. “Semi-Supervised Topic Modeling for Image Annotation”. In: *Proceedings of the 17th ACM international conference on Multimedia*. [DOI] (cit. on p. 20).

Shariff, Aabid, Robert F. Murphy, and Gustavo K. Rohde (May 2010). "A generative model of microtubule distributions, and indirect estimation of its parameters from fluorescence microscopy images." In: *Cytometry. Part A : the journal of the International Society for Analytical Cytology* 77.5, pp. 457–66. ISSN: 1552-4930. [DOI] (cit. on p. 13).

— (2011). "Automated Estimation of Microtubule Model Parameters from 3-D Live Cell Microscopy Images". In: *Proceedings of the 2011 IEEE International Symposium on Biomedical Imaging (ISBI 2011)*, pp. 1330–1333 (cit. on p. 13).

Shariff, Aabid, Gustavo K. Rohde, and Robert F. Murphy (2009). "Indirect learning of generative models for microtubule distribution from fluorescence microscope images". In: *ICML-UAI-COLT 2009 Workshop on Automated Interpretation and Modeling of Cell Images (Cell Image Learning)*. Montreal, Canada. [Online Version] (cit. on p. 13).

Shatkay, Hagit, Annette Höglund, Scott Brady, Torsten Blum, Pierre Dönnes, and Oliver Kohlbacher (2007). "SherLoc: high-accuracy prediction of protein subcellular localization by integrating text and protein sequence data." In: *Bioinformatics (Oxford, England)* 23.11, pp. 1410–7. ISSN: 1367-4811. [DOI] (cit. on pp. 18, 75).

Srinivasa, Gowri, Matthew C. Fickus, Manuel N. Gonzalez-Rivero, Sarah Yichia Hsieh, Adam D Linstedt, and Jelena Kovacevic (May 2008). "Active mask segmentation for the cell-volume computation and Golgi-body segmentation of hela cell images". In: *2008 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*. Vol. C. IEEE, pp. 348–351. ISBN: 978-1-4244-2002-5. [DOI] (cit. on p. 25).

Srinivasa, Gowri, Matthew C. Fickus, Yusong Guo, Adam D. Linstedt, and Jelena Kovacević (2009). "Active mask segmentation of fluorescence microscope images." In: *IEEE Transactions on Image Processing* 18.8, pp. 1817–1829. [Online Version] (cit. on p. 25).

Stroustrup, Bjarne (Jan. 1994). *The design and evolution of C++*. Addison-Wesley Professional. ISBN: 0-201-54330-3. [Online Version] (cit. on p. 86).

Telmer, C. A., Peter B. Berget, B. Ballou, Robert F. Murphy, and Jon W. Jarvik (Mar. 2002). "Epitope tagging genomic DNA using a CD-tagging Tn10 minitransposon." In: *BioTechniques* 32.2, pp. 422–4, 426, 428–30. ISSN: 0736-6205. [Online Version] (cit. on p. 3).

Troyanskaya, Olga G. (Mar. 2005). "Putting microarrays in a context: integrated analysis of diverse biological data." In: *Briefings in bioinformatics* 6.1, pp. 34–43. ISSN: 1467-5463. [DOI] (cit. on p. 18).

Troyanskaya, Olga G., Kara Dolinski, Art B. Owen, Russ B. Altman, and David Botstein (July 2003). "A Bayesian framework for combining heterogeneous data sources for gene function prediction (in Saccharomyces cerevisiae)." In: *Proceedings of the National Academy of Sciences of the United States of America* 100.14, pp. 8348–53. ISSN: 0027-8424. [DOI] (cit. on p. 18).

Tuytelaars, Tinne and Krystian Mikolajczyk (2007). “Local Invariant Feature Detectors: A Survey”. In: *Foundations and Trends in Computer Graphics and Vision* 3.3, pp. 177–280. ISSN: 1572-2740. [DOI] (cit. on pp. 14, 15).

Velliste, Meel and Robert F. Murphy (2002). “Automated determination of protein subcellular locations from 3D fluorescence microscope images”. In: *Proceedings IEEE International Symposium on Biomedical Imaging*. IEEE, pp. 867–870. ISBN: 0-7803-7584-X. [DOI] (cit. on p. 23).

Veretnik, Stella, J Lynn Fink, and Philip E Bourne (Jan. 2008). “Computational biology resources lack persistence and usability.” In: *PLoS computational biology* 4.7. Ed. by Barbara Bryant, e1000136. ISSN: 1553-7358. [DOI] (cit. on p. 79).

Wang, James and Yixin Chen (2004). “Image Categorization by Learning and Reasoning with Regions”. In: *Journal of Machine Learning Research* 5, pp. 913–939 (cit. on p. 14).

Wang, Jinjun, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong (June 2010). “Locality-constrained Linear Coding for image classification”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 3360–3367. ISBN: 978-1-4244-6984-0. [DOI] (cit. on p. 43).

Willamowski, Jutta, Damian Arregui, Gabriela Csurka, Chris Dance, and Lixin Fan (2004). “Categorizing Nine Visual Classes using Local Appearance Descriptors”. In: *ICPR 2004 Workshop Learning for Adaptable Visual Systems*. Cambridge, United Kingdom (cit. on p. 42).

Wren, Jonathan D. (Mar. 2004). “404 not found: the stability and persistence of URLs published in MEDLINE.” In: *Bioinformatics (Oxford, England)* 20.5, pp. 668–72. ISSN: 1367-4803. [DOI] (cit. on p. 79).

— (June 2008). “URL decay in MEDLINE–a 4-year follow-up study.” In: *Bioinformatics (Oxford, England)* 24.11, pp. 1381–5. ISSN: 1367-4811. [DOI] (cit. on p. 79).

Zhao, Ting and Robert F. Murphy (2007). “Automated learning of generative models for subcellular location: building blocks for systems biology.” In: *Cytometry. Part A : the journal of the International Society for Analytical Cytology* 71.12, pp. 978–90. ISSN: 1552-4930. [DOI] (cit. on pp. 13, 16, 17).

Zhao, Ting, Meel Velliste, Michael V. Boland, and Robert F. Murphy (Sept. 2005). “Object type recognition for automated analysis of protein subcellular location.” In: *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society* 14.9, pp. 1351–9. ISSN: 1057-7149. [Online Version] (cit. on pp. 30, 31).

Zhou, Jie and Hanchuan Peng (2007). “Automatic recognition and annotation of gene expression patterns of fly embryos.” In: *Bioinformatics (Oxford, England)* 23.5, pp. 589–96. ISSN: 1367-4811. [DOI] (cit. on p. 17).

Zhu, Jun, Amr Ahmed, and Eric P. Xing (2009). “MedLDA: Maximum margin supervised topic models for regression and classification”. In: *Proceedings of the 26th Annual International Conference on Machine Learning - ICML ’09*. New York, New York, USA: ACM Press, pp. 1–8. ISBN: 9781605585161. [DOI] (cit. on p. 20).

Zhu, Long, Yuanhao Chen, and Alan Yuille (Jan. 2009). “Unsupervised learning of Probabilistic Grammar-Markov Models for object categories.” In: *IEEE transactions on pattern analysis and machine intelligence* 31.1, pp. 114–28. ISSN: 0162-8828. [DOI] (cit. on pp. 21, 34).

Zhu, Song-Chun and David Mumford (2006). “A Stochastic Grammar of Images”. In: *Foundations and Trends in Computer Graphics and Vision* 2.4, pp. 259–362. ISSN: 1572-2740. [DOI] (cit. on pp. 16, 21).

Index

Colophon

This dissertation was typeset with TEX, using the `memoir` class. The main text is typeset using Linux Libertine. Chapter headers are typeset in Anaktoria. Section, subsection, and subsection headers are typeset in Lucida Bright, Lucida Sans, and Lucida Sans Oblique; respectively.