Classification of yeast subcellular protein locations in microscopic images using random forest machine learning: comparison of different feature extraction methods

Zeng Fung Liew Yixing Lu

June 9, 2021

Abstract

High-throughput microscopic images of single cells provide a rich source of biological data. However, given its high dimension nature, proper feature extraction methods need to be used to extract informative features from these images for downstream analysis. A common task in cell biology is to classify cells based on the subcelluar locations of target proteins using microscopic images. In the current project, images of single yeast cells with fluorescently-tagged endogenous protein were used to evaluate different feature extraction methods in terms of their performance in subcellular protein location classification, using a random forest model. Among the five techniques considered (scale-invariant feature transform, stationary wavelet transform, local discriminant basis, Haralick texture features, and scattering transform), scattering transform features achieved the highest test set classification accuracy: 71%. Factors that restricted the performance of other feature extractors include low signal contrast in images, rotation and shift of cells, etc.

1 Introduction

High-throughput microscopic imaging of cells together with endogenous protein tagging enable large-scale systematic screening of target protein collections, investigating gene functions, looking for effect of gene mutation of protein abundance and location, etc. [11, 5] The microscopic imaging process can be automated, rendering large numbers of micrographs in a relatively short time, out-competing some time-consuming and error-prone biochemical assays. However, due to the large number of images, the image processing and subsequent feature extraction and statistical analyses had to be automated as well. The cell segmentation procedure is normally standard, but the feature extraction methods can be problem and image specific. In this project, different feature extraction methods were utilized to extract information from single-cell microscopic images of yeast cells, with the objective to classify subcellular protein locations.

Five feature extraction methods were evaluated in the current project: scale invariant feature extraction (SIFT), stationary wavelet transform + discrete cosine transform (SWT), local discriminant basis (LDB), Haralick texture features, and scattering transform. SIFT is developed by Davis Lowe [12] and has been widely used in pattern matching [4], object detection [1], image stitching [7] etc. It's algorithm was based on keypoint detection and description. One of the major advantage of this technique is that it is invariant to uniform scaling, orientation, and illumination changes [12]. SWT is a modification of discrete wavelet transform, where a low pass filter and high pass filter extracted low and high frequency information from a image respectively. Unlike discrete wavelet transform, stationary wavelet transform does not perform image subsampling, thus can be considered as a redundant process, giving it a much desired shift-invariant property with a time-complexity tradeoff. LDB is a feature extraction technique based off wavelet packet transforms that was developed by N. Saito and R. Coifman [19]. Wavelet packets have been used to represent and classify textures consisting of oscillatory patterns [18], and as a dimension reduction tool for hyperspectral images [10]. Haralick textures is a set of texture measures based on grayscale co-occurrence matrix of images, developed by Haralick [9]. Texture measures described the spatial distribution of intensity in an image in a single band. It is a commonly used method to extract texture information from cell microscopic images. The last feature extraction method we explored is scattering transform, a translation-invariant operator proposed by Bruna and Mallat [3]. It can be thought of as a cascade of convolution with wavelets, non-linear modulus, and local averaging operation, which makes it very similar to the deep convolutional neural network, only that the wavelet at each layer is fixed [13]. In order to compare across the feature extraction methods, a random forest model was fitted to each set of features extracted using different methods.

2 Methodology

The microscopic images used in the current project was obtained from a study by Parnamaa et al. [17].The images were high-throughput proteome-scale microscopic images of yeast cells. Each image has two channels: a red fluorescent protein marking the cell contour, and a green fluorescent protein (GFP) that tags different endogenous protein of interest, thus characterizes the abundance and sub-cellular localization of the protein. Based on different GFP locations, the images were annotated with 12 different labels (classes): cell periphery, cytoplasm, endosome, endoplasmic reticulum, golgi, mitochondrion, nuclear periphery, nucleolus, nucleus, peroxisome, spindle pole, and vacuole. The microscopic images used in the current project have already been segmented to identify individual cells and cropped to 64×64 patches centered on the cell midpoint. Example images from each class were demonstrated in Figure 1 The single cell RGB images were split into a training, a validation, and a test set of size 65,000, 12,500, and 12,500 respectively. Our image classification pipeline follows: **Image preprocessing $\longrightarrow$ Feature extraction $\longrightarrow$ Model fitting with random forest.**

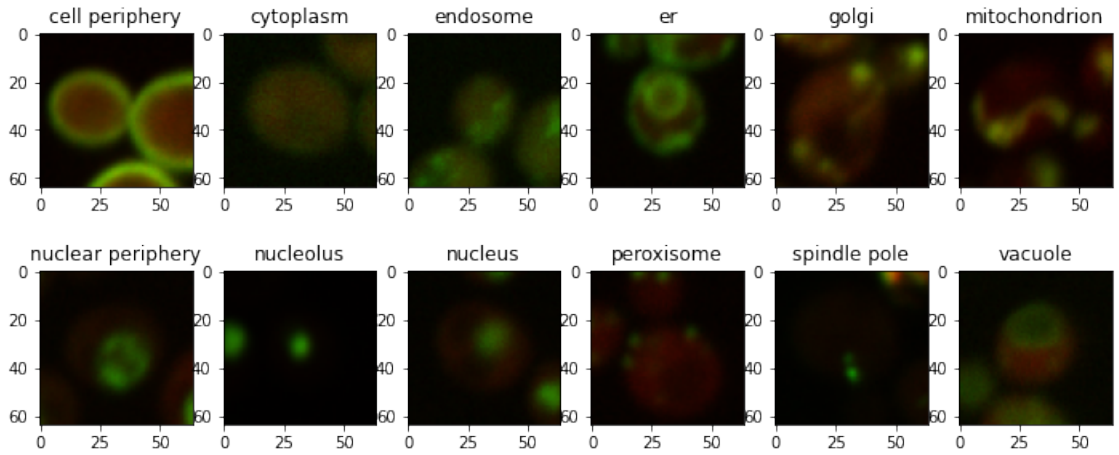


Figure 1: Example images from each class.

2.1 Image preprocessing

All our feature extraction methods require single channel images (2D array), but all the input images had three channels (RGB), thus it is necessary to pre-process the images to fulfill such requirement. The most common method is to convert the RGB images into gray scale ones, however since each channel of the microscopic images carried clearly-defined information: red channel marked the cell contour, green channel highlighted the location of the protein of interest, and blue channel was empty, we decided to split the channels, discard the empty blue one, use the red channel to create a mask that only highlighted the cell region of a image, and superimpose the mask on the green channel to define a region of interest (ROI), then use the masked green channel as the input for feature extraction. The steps in creating binary mask and defining ROI were summarized as following:

1. **Creating binary mask**: after splitting the channels, a Gaussian blur filter with specified variance and kernel size was applied to the red channel, followed by Otsu thresholding, then opening to remove noise and create the binary mask, where the foreground was the cell contour and background was set to zero.

   (a) **Otsu thresholding** is a method to automatically threshold a image by searching for a single threshold value that separate the pixels into two classes so that the intra-class variance is minimized [16]:

   $$\sigma_\omega^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

   where the class weights $\omega_0(t)$ and $\omega_1(t)$ are the probabilities of the two classes at threshold t, and $\sigma_0^2(t)$,$\sigma_1^2(t)$ are the variance of the two classes.

   (b) **Opening** is a process of erosion followed by dilation. A kernel with specified size scanned through a binary image where foreground had value 255, and background was 0. In erosion, a pixel would remain 255 only if all the pixels within the kernel were 255, otherwise would be 0. Such process eroded the boundary of the foreground.

In dilation, an opposite operation was performed so that a pixel value would be 255 as long as one of the pixels within the kernel was 255. After such erosion and dilation step, noises in the background were removed. Figure 2 gave an example of the process creating a binary mask.

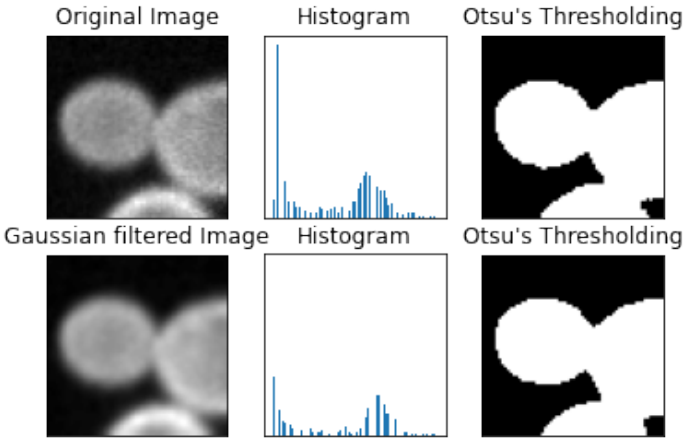


Figure 2: Gaussian filtering and Otsu thresholding applied to the red channel of an example image.

2. **Defining ROI and normalization**: since the green channels would be used as the input for feature extraction, the created binary masks were superimposed on the the green channel to define the region of interest (ROI). Moreover, since some classes of microscopic images had very low contrast between foreground and background in the green channel, there would be trouble of identifying key points in these images for feature extraction methods that relied on descriptors of key points. Therefore, normalization within the ROI region was applied to all the green channels after superimposing the masks to increase the contrast. Figure 3 showed examples of pre-processed images from each class.

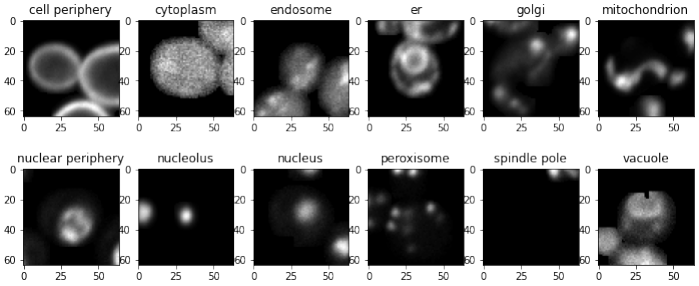


Figure 3: Example images from each class after ROI masking and normalization on the green channel.

2.2 Feature extraction

Feature extraction can also be thought of as a dimension reduction technique for image data, where only the most important set of features within an image is extracted and sent to the classification model. Throughout the years, the number of feature extraction techniques have increased drastically, each of them looking for different types of features within an image. In this project, we look into 5 types of feature extraction technique, namely the Scale Invariant Feature Transform (SIFT) with Bag of Features (BoF), the Stationary Wavelet Transform (SWT) with Discrete Cosine Transform (DCT), the Local Discriminant Basis (LDB), the Haralick texture features, and the Wavelet Scattering Transform.

2.2.1 Scale Invariant Feature Transform (SIFT) + Bag of Features (BoF)

The Scale Invariant Feature Transform (SIFT) is a feature extraction technique developed by David Lowe that focuses on the search of keypoints in images. This is extremely useful in the field of computer vision and pattern matching. In

this project, SIFT is paired with the Bag of Features (a.k.a. Bag of Words or Bag of Visual Words) procedure [15] before moving into the model fitting process.

A summary of the SIFT algorithm is as follows [12]:

1. **Scale-space extrema detection**: The aim of this step is to detect keypoints in an image that is scale invariant. This means that the same set of keypoints will be detected in a zoomed-in or zoomed-out version of the same image. The detection of keypoints follow the procedure below:

   (i) A Gaussian blur G with an arbitrarily selected noise level σ is applied onto an input image I. For each point (x, y) in the image, the blurred image is defined as

   $$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

   where $*$ is the convolution operation in x and y, and

   $$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

   (ii) Repeat step (i) with $G(x, y, k\sigma)$ for $k = \{2^{1/2}, 2^1, 2^{3/2}, 2^2, 2^{5/2}\}$. Notice that the values of k increases by a factor of $\sqrt{2}$.

   (iii) Compute the difference-of-Gaussian function (an approximation to the scale-normalized Laplacian of Gaussian function) of each adjacent blurred images. This can be written as

   $$D_i(x, y, \sigma) = L(x, y, k_{i+1}\sigma) - L(x, y, k_i\sigma)$$

   (iv) The difference-of-Gaussian for the first octave is now complete. To compute the DoG for the second octave, one has to subsample the third Gaussian blurred image from the first octave by keeping only the even-indexed pixels in each row and column. Then, steps (ii) and (iii) are repeated on this subsampled image, this time with $k = \{2^{3/2}, 2^2, 2^{5/2}, 2^3, 2^{7/2}\}$. Essentially, the values of k still increases by a factor of $\sqrt{2}$, but this time the starting value of k is the third k value from the previous octave. The same process can be repeated with more octaves. The number of octaves implemented in Lowe's paper and OpenCV is 3. One may refer to Figure 4 for a visual representation of steps (i) to (iv).

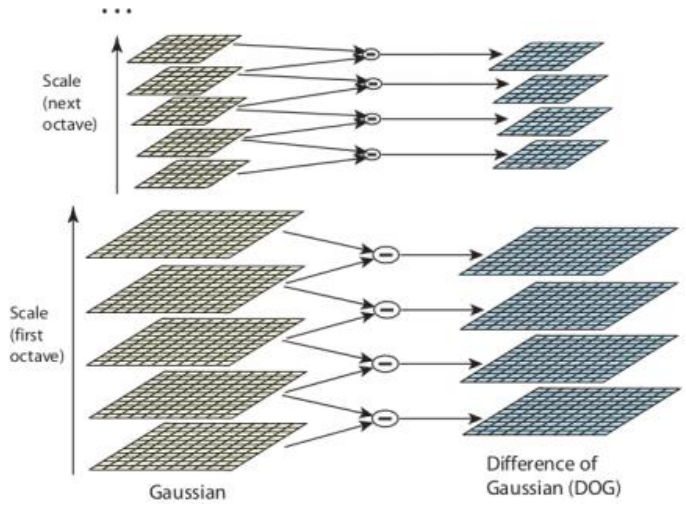


Figure 4: A visualization on how the difference of Gaussian (DoG) works. (Image obtained from original paper)

   (v) After computing the DoG, the next step is to select the local maxima of $D_i(x, y, \sigma)$. This is done by scanning through all the datapoints in the DoG matrices and comparing them with their respective neighbors. A point is selected as a local maxima (or a keypoint) if it is larger than all of its neighbors, as represented in Figure 5.

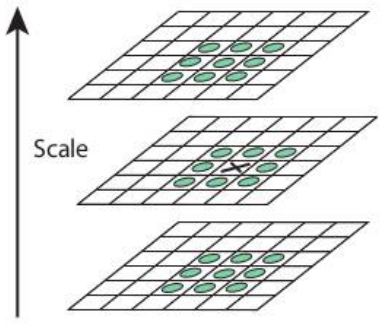


Figure 5: A visualization on how each point (x) is compared to its neighbors (marked in green). (Image obtained from original paper)

2. **Keypoint localization**: Once we obtain all the keypoints from step 1, we need to decide which ones to keep and which ones to reject. A good keypoint needs to have a good fit with nearby data for location, scale, and ratio of principal curvatures. In other words, keypoints with low contrasts or unclear edges are rejected.

   Let $D(\mathbf{x}) = D(x, y, \sigma)$ and $\hat{\mathbf{x}} = -\frac{\delta^2 D}{\delta \mathbf{x}^2}^{-1} \frac{\delta D}{\delta \mathbf{x}}$, then we denote

   $$D(\hat{\mathbf{x}}) = D(\mathbf{x}) + \frac{1}{2} \frac{\delta D}{\delta \mathbf{x}}^{\top} \hat{\mathbf{x}}$$

   The keypoints with values $|D(\hat{\mathbf{x}})|$ less than a certain threshold α are rejected for having low contrasts. In Lowe's paper, α is set to be 0.03.

   As for removing unclear edges, one has to first compute the Hessian matrix $\mathbf{H}$, where

   $$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix}$$

   Then, with a selected threshold value r ($r = 10$ in Lowe's paper), a keypoint is rejected if

   $$\frac{\text{tr}(\mathbf{H})^2}{\det(\mathbf{H})} > \frac{(r+1)^2}{r}$$

3. **Orientation assignment**: The completion of step 1 gives us keypoints that are scale invariant, while the completion of step 2 reduces the number of keypoints based on their surrounding contrasts and edge clarity. This step aims to provide each keypoint its individual orientation and direction. To do so, a Gaussian smoothed image segment L of size 16×16 is first selected based on the scale of a keypoint. Then, for each point (x, y) in L, the gradient magnitude $m(x, y)$ and the gradient orientation $\theta(x, y)$ is computed as

   $$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

   $$\theta(x, y) = \tan^{-1}\left(\frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}\right)$$

   Once the values of $m(x, y)$ and $\theta(x, y)$ are computed for all possible x and y, an orientation histogram is built with bin size of 10 degrees. In other words, a total of 36 bins are constructed corresponding to degree ranges of $0 - 9, 10 - 19, \ldots, 350 - 359$. The height of each bin then depends on the values of $m(x, y)$ and $\theta(x, y)$. Each point (x, y) in L will contribute to an aforemention bin depending on its $\theta(x, y)$ with a height proportional to its $m(x, y)$.

   After the orientation histogram is constructed, the bin with the highest peak magnitude will be the orientation assigned to the keypoint. Additionally, any other bins with a peak magnitude of at least 80% of the highest peak magnitude will be used to create additional keypoints of the same location but with different orientation.

4. **Keypoint descriptor**: At this stage, all the necessary keypoints with the desired scale and orientation invariance properties are already obtained, what's left is to compute distinct "fingerprints" that describe each keypoint.

Much like the case of step 3, a 16×16 image segment is extracted around a keypoint. This time, the image segment is divided into 4×4 blocks. Within each block, the value $m(x, y)$ and $\theta(x, y)$ are computed as in step 3. This time, the orientation histogram constructed should only contain 8 bins, ie. bins with ranges $0 - 44, 45 - 89, \ldots, 315 - 359$. This is repeated for each block as shown in Figure 6. Then, the height of each bin becomes a feature value for the descriptor. With 16 blocks in total, each block producing histograms of 8 bins, this amounts to a total of $16 \cdot 8 = 128$ features in the descriptor.

Once this step is repeated with all the keypoints in the image, we will have successfully ran the SIFT algorithm.

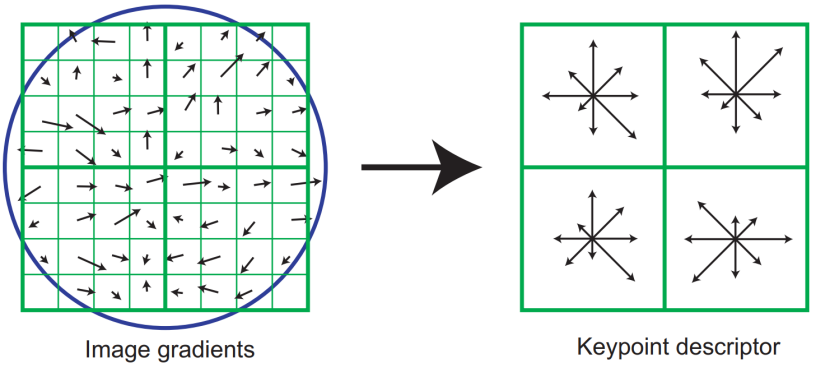


Figure 6: A quarter segment of the 16×16 block. Within each block, the gradient magnitudes $m(x, y)$ and orientations $\theta(x, y)$ are computed to construct orientation histograms that point to 8 directions as shown in the image on the right. (Image obtained from original paper)

The SIFT procedure is computed on all the images in the training set of size N. Once all the keypoints and descriptors are computed, we then stack all the descriptors into one large matrix of size $\sum_{i=1}^{N} k_i \times 128$, where k_i is the number of keypoints for image i, and move onto the Bag of Features (BoF) procedure as follows:

1. Run K-means clustering with C clusters on the large descriptor matrix.
2. For each image, assign its descriptors to their respective clusters. The number of descriptors in each cluster becomes the new set of feature values. In other words, there will now be a total of C features to be fed into a classification model.

At this point, the SIFT with BoF feature extraction method is complete. In this project, we configured our SIFT to compute 3 octave layers with $\sigma = 1.6$, without any form of contrast or edge thresholding due to lows contrasts and unclear edges in many of our images. For the BoF portion, we set $C = 40$ clusters for the K-means step.

2.2.2 Stationary Wavelet Transform + Discrete Cosine Transform

The stationary wavelet transform based feature extraction method is referenced from Qayyum et al's paper [2] of combining stationary wavelet transform features with discrete cosine transform for facial expression classification.

Before diving into the idea of stationary wavelet transform, one needs to understand the inner workings of the regular wavelet transform. In the regular discrete wavelet transform, a pair of quadrature mirror filters H and G are generated from a selected wavelet. These filters correspond to high pass and low pass filters respectively. To simply put, the high pass filter H detects the high frequency features in a signal, while the low pass filter G does the same to the low frequency features.

Assuming we have $H = \{h_0, h_1, \ldots, h_f\}$, $G = \{g_1, g_2, \ldots, g_f\}$, and a signal x, the convolution or the filtering step can be written as follows [23]:

$$(Hx)[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k]$$

$$(Gx)[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot g[n-k]$$

Upon the completion of the convolution step, the even-indexed coefficients of both Hx and Gx are kept, while the odd-indexed coefficients are removed. This process is known as *subsampling*. The subsampled Hx is known as the detail coefficients, while the subsampled Gx is known as the approximation coefficient.

The convolution and the subsampling step can be repeated for up to $\log_2 n$ times on the approximation coefficients, which is until there is only 1 approximation coefficient remaining. The full workflow of the discrete wavelet transform is shown in Figure 7.

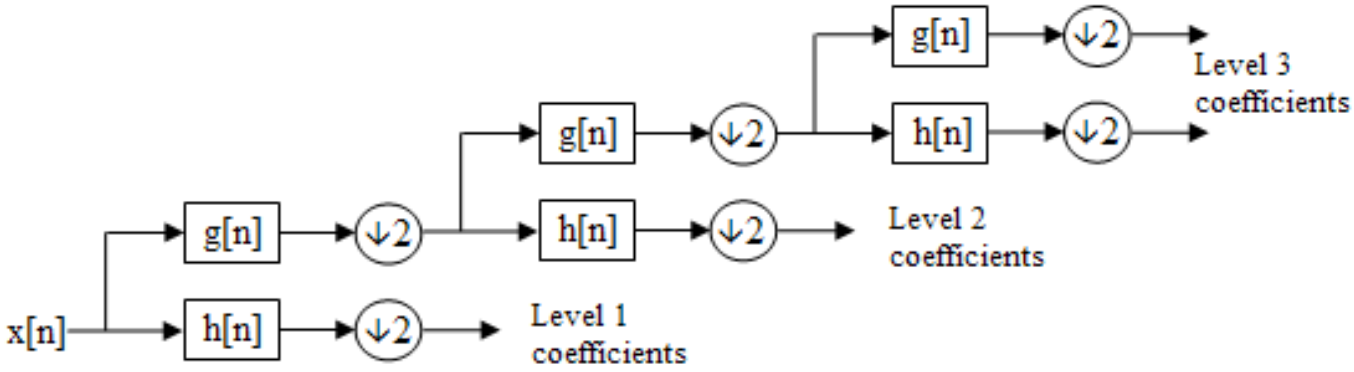


Figure 7: The standard discrete wavelet transform workflow. (Image obtained from Wikipedia [23])

Now, for the case of wavelet transform on images (which can be thought of as a 2D-signal), one will compute the discrete wavelet transform first on the rows before repeating the same thing on the columns, as shown in Figure 8. Note that in the case of images, the high-frequency features correspond to the edges present in an image.

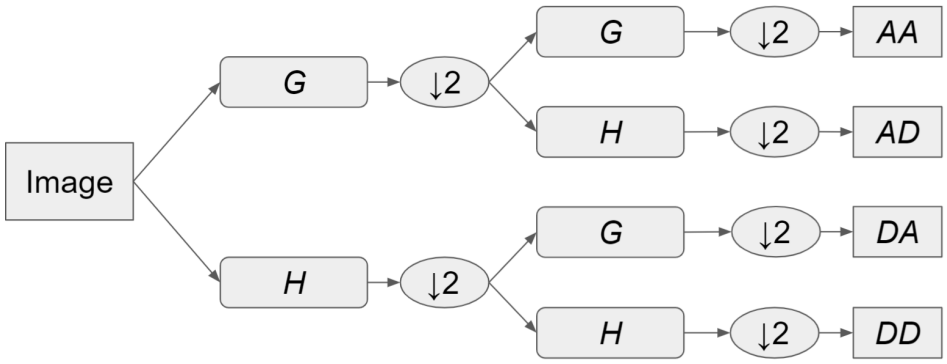


Figure 8: A single level of discrete wavelet transform on images. A denotes approximate coefficients, D denotes detail coefficients.

With all of these in mind, the stationary wavelet transform is simply a slight tweak of the discrete wavelet transform, where the subsampling procedure is not performed. For a 1D-signal of size n, this results in the total number of coefficients at level L to be $2n$ instead of $2^{\log_2 n - L}$. For an image of size $n \times n$, it would be $4n$ instead of $4^{\log_2 n - L}$. The stationary wavelet transform is considered to be a *redundant* transform, and this comes with the much desired shift-invariant property.

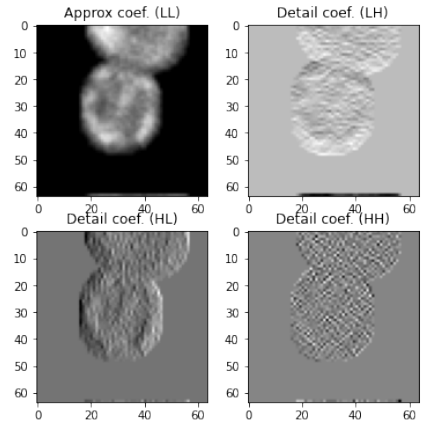
After computing a single level of decomposition, only the detail coefficient images (AD, DA, DD) are kept and used for the next step, which is the discrete cosine transform step. This is essentially a step used for reducing the length of feature vectors. In particular, 8×8 blocks of DCT are applied to each of the AD, DA, DD subbands as follows:

$$X(u,v) = \frac{C(u)C(v)}{4} \sum_{m=0}^{7} \sum_{n=0}^{7} x[m,n] \cos(\frac{(2m+1)u\pi}{16}) \cos(\frac{(2n+1)v\pi}{16})$$

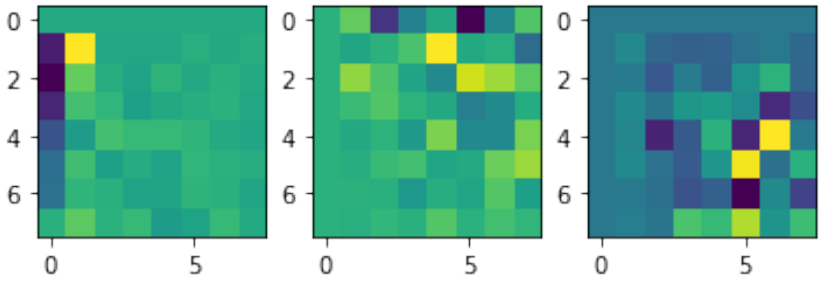
where

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u = 0 \\ 1 & \text{for } 1 \leq u \leq 7 \end{cases}, C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } v = 0 \\ 1 & \text{for } 1 \leq v \leq 7 \end{cases}$$

With our 64×64 sized images, the stationary wavelet transform would result in 3 64×64 coefficient matrices (Figure 9(a)), and the discrete cosine transform step reduces the size of those matrices to 3 8×8 matrices (Figure 9(b)), totalling to 192 features for our model fitting step.



((a)) Example of an image after undergoing SWT



((b)) Resulting matrices after the 3 detail coefficient matrices in 9(a) undergo 8×8 block DCT

Figure 9: An example of the feature extraction process using SWT and DCT

2.2.3 Local Discriminant Basis (LDB)

The Local Discriminant Basis (LDB) [19] is a feature extraction technique based off wavelet packet transforms that was developed by N. Saito and R. Coifman. The main idea of this method is that each wavelet coefficient of the wavelet packet decomposition is compared between labels, and a subset of those coefficients that best discriminate the different classes are selected based on a distance metric such as the Kullback-Leibler divergence. Generally, the LDB algorithm goes as follows:

1. Each signal of size n in the training data is decomposed up to $\log n$ levels. Due to the downsampling properties of the wavelet transform, one can visualize that each signal can be decomposed into an $n \times L$ matrix, where $L \leq \log n$ is the number of decomposition levels. Since this is repeated for the training data of size N, one will end up with an array of size $n \times L \times N$.

2. An energy map is computed for each class label. There are a few measures that can be used to compute the energy map. The simplest way to build energy maps is based on the normalized energies of each class of signals, a.k.a. the *normalized time-frequency energy map*, ie.

$$\Gamma^{(c)}(j,k,l) = \frac{\sum_{i=1}^{N_c} (w_{j,k,l}^{\top} x_i^{(c)})^2}{\sum_{j=1}^{N_c} ||x_j^{(c)}||^2}$$

   where $j = 0, \ldots, L$, $k = 0, \ldots, 2^j - 1$, and $l = 0, \ldots, 2^{n-j} - 1$.

   However, the normalized time-frequency energy map comes with a limitation, which is some loss of information since we are shrinking N_c matrices of size $n \times L$ to a single matrix of size $n \times L$ for each class c.

   An improved version of constructing energy maps is based on the differences among the *probability density functions* (PDFs) [20] of each wavelet coefficient between classes. Essentially, for each class c, a total of $n \times L$ PDFs were estimated using average shifted histograms [21]. This means that there are a total of C arrays of size $n \times L \times M$, where M is the length of the PDF estimate. Compared to the normalized time-frequency energy map, the probability density energy map retains more information, but takes a significantly longer time to compute.

3. The distance between energy maps are computed based on some discriminant measures. A straightforward metric for such purpose is the *relative entropy* (a.k.a. *Kullback-Leibler divergence*). Basically, given two classes c_p and c_q, the "distance" between the wavelet coefficient $\alpha_{j,k,l}$ of the two classes is denoted as

$$I(\alpha_{j,k,l}^{(p)}, \alpha_{j,k,l}^{(q)}) = \sum_{m=1}^{M} \alpha_{j,k,l;m}^{(p)} \log \frac{\alpha_{j,k,l;m}^{(p)}}{\alpha_{j,k,l;m}^{(q)}}$$

Note that the Kullback-Leibler divergence is not symmetric, ie. $I(\alpha_{j,k,l}^{(p)}, \alpha_{j,k,l}^{(q)}) \neq I(\alpha_{j,k,l}^{(q)}, \alpha_{j,k,l}^{(p)})$. To overcome this, the *symmetric relative entropy*, denoted as

$$J(\alpha_{j,k,l}^{(p)}, \alpha_{j,k,l}^{(q)}) = I(\alpha_{j,k,l}^{(p)}, \alpha_{j,k,l}^{(q)}) + I(\alpha_{j,k,l}^{(q)}, \alpha_{j,k,l}^{(p)})$$

may be computed.

This is repeated for each coefficient in the energy map, ie. nL times in total. Now, denote the discriminant measure between two classes c_p and c_q at $\alpha_{j,k,l}$ as $D(\alpha_{j,k,l}^{(p)}, \alpha_{j,k,l}^{(q)})$, where D can be either of the asymmetric or the symmetric relative entropy. To measure the discrimination among all C classes, the following is computed:

$$D(\{\alpha_{j,k,l}^{(c)}\}_{c=1}^{C}) = \sum_{p=1}^{C-1} \sum_{q=p+1}^{C} D(\alpha_{j,k,l}^{(p)}, \alpha_{j,k,l}^{(q)})$$

Once this step is complete, what's remained should be a single $n \times L$ matrix.

4. A basis tree that best separates the C classes is computed. This is highly similar to the entropy-based best basis selection algorithm developed by Coifman and Wickerhauser [6]. The only difference here is that Coifman and Wickerhauser's approach focused on minimizing entropies, while this best basis selection focuses on *maximizing* the discriminant measures.

5. Once the best basis tree, is obtained, the corresponding wavelet coefficients of each signal are selected and ordered by their power of discrimination. Here, the measure for discriminant power is measured using the *Robust Fisher's class separability*:

$$\frac{\sum_{c=1}^{C} \pi_c |\text{med}_i(\alpha_{j,k,l;i}^{(c)}) - \text{med}_c(\text{med}_i(\alpha_{j,k,l;i}^{(c)}))|}{\sum_{c=1}^{C} \pi_c \text{mad}_i(\alpha_{j,k,l;i}^{(c)})}$$

where med_i and mad_i are the sample median and median absolute deviation across the samples indexed i at class c, while med_c is the sample median from all the C classes. At this point, one can then decide on the number of coefficients to be used for model fitting.

In this project, we had to first reshape each 64×64 image into signals of length 4096 since there is currently no open-source libraries that support 2D-signals. Here, we set $L = 8$ and used the probability density energy map with symmetric relative entropy as its discriminant measure.

2.2.4 Gray Level Co-occurrence Matrix based Texture Features

Texture describes the spatial distribution of intensity in an image, indicating the consistency of patterns and/or colors. One way of describing the texture of an image is through gray level co-occurrence matrix (GLCM). In a gray-scale image, co-occurrence matrix is a matrix containing information about how often pairs of pixel values co-occur in a pre-defined adjacency, including direction and distance. For an image with N different pixel values, the resulting GLCM would have size $N \times N$. From each GLCM, 13 features were extracted as described by Haralick [9]. The steps in constructing GLCM and brief description of the 13 Haralick features were presented here:

1. **Define adjacency offset**: There are four different directions of adjacency of two pixel values and the unit apart in that direction can be specified by a scale parameter (Δx, Δy): top-to-bottom, left-to-right, top left-to-bottom right, and top right-to-left bottom. In the current project, 6 different offset distance: 1 to 6 pixels, were used. Therefore, 24 GLCM were constructed for all the images.

2. **Constructing GLCM**: In a co-occurrence matrix with size $N \times N$, $C_{i,j}$ is the probability that the i^{th} pixel value is found adjacent to the j^{th} pixel value ($p(i, j)$), as defined by the adjacency offset.

3. **Haralick features**: There were 14 features from the original paper of Haralick [9], but the last feature was not computationally stable, thus it is common to just use the other 13 features. These features described the co-occurrence matrix in terms of homogeneity, contrast, randomness, complexiity, etc. The description below was adapted from documentation of CellProfiler:

    - Angular Second Moment: Measure of image homogeneity. A higher value indicates that the intensity varies less in an image.
    - Contrast: Measure of local variation in an image.
    - Correlation: Measure of linear dependency of intensity values in an image. For an image with large areas of similar intensities, correlation is much higher than for an image with noisier, uncorrelated intensities.
    - Variance: Measure of the variation of image intensity values.
    - Inverse Difference Moment: Another feature to represent image contrast. Has a relatively higher value for homogeneous images.
    - Sum Average: The average of the normalized gray scale image in the spatial domain.
    - Sum Variance: The variance of the normalized gray scale image in the spatial domain.
    - Sum Entropy: A measure of randomness within an image.
    - Entropy: An indication of the complexity within an image.
    - Difference Variance: The image variation in a normalized co-occurrence matrix.
    - Difference Entropy: Another indication of the amount of randomness in an image.
    - InfoMeas1: A measure of the total amount of information contained within a region of pixels derived from the recurring spatial relationship between specific intensity values.
    - InfoMeas2: An additional measure of the total amount of information contained within a region of pixels derived from the recurring spatial relationship between specific intensity values. It is a complementary value to InfoMeas1 and is on a different scale.

4. **Intensity measurement**: Since the Haralick featured did not account for the pixel intensity values from the original images, five intensity measurements were made on the green channel of each image before any image pre-processing:

    - MeanIntensity: the average pixel intensity within ROI
    - StdIntensity: the standard deviation of pixel intensities within ROI
    - MinIntensity: the minimum of intensity with ROI
    - MaxIntensity: the maximum of intensity within ROI
    - MassDisplacement: difference between the centroids of ROI defined objects in grayscale representation and that in binary representation.

    Concatenating the intensity measurements and Haralick features, it resulted in $5 + 24 \times 13 = 317$ features in total.

2.2.5 Scattering transform

Scattering transform computes locally translation-invariant image descriptors proposed by Bruna and Mallat [3]. It carries the same locally translation-invariant characterization of SIFT but contains more high frequency information. It's algorithm included a cascade of three operations: Wavelet decomposition, complex modulus, local averaging. The parameters that need to be set for scattering transform include: number of layers (k), number of orientations of wavelet transformation (L), and log2 of the highest scattering scale (J). The algorithm is summarized here:

1. The scattering coefficient from the 1st layer of scattering operator can be computed as:

$$|f * \psi_{j_1,\lambda_1}| * \phi_J$$

    where f is the signal,ψ_{j_1,λ_1} is the wavelet at scale j_1 and orientation λ_1, and ϕ_J is the averaging filter.

2. The local averaging will reduce some high frequency information, which can be recovered by convolving $|f * \psi_{j_1,\lambda_1}|$ with another set of wavelets at different scale j_2. Thus, the scattering coefficients after kth layer are obtained by performing the following operations iteratively k times:

$$S_{K,J}(f(x)) = ||f * \psi_{j_1,\lambda_1}| * ... * \psi_{j_k,\lambda_k}|| * \phi_J,\ j_k < ... < j_1 < J, (\lambda, ..., \lambda_k) \in \Gamma^k$$

The kth layer output has a size of $L^k \times \binom{J}{k}$.

3. In the current project, the parameters were set to $k = 2, J = 3, L = 4$. That is, a combination of 4 orientations and 3 scales of wavelets transformation will be applied to the original 2D signal. After 2 layers of operations, it resulted in $2 * (1 + JL + L^2 * J(J-1)/2) = 61$ transformed images with pixel size 8×8.

4. For each resulting scattering transformed image, the mean and variance of the signals were calculated, resulting in 122 features.

2.3 Random Forest Model

For the sake of fairness, we used random forest as the classifier for the training on the discussed feature extraction outputs. To understand what a random forest is and what it does, one has to first understand the mechanics of decision trees.

For the purpose of classification, the decision tree goes through the following procedure:

1. To build a decision tree, one has to start off at the root node. At this point of time, the entire training data is used as input to the root node.

2. The node is split based on a splitting criterion such as the cross-entropy (a similar metric to the *Kullback-Leibler divergence* discussed previously) and the Gini Impurity criterion. The mathematical formulation of both criteria is discussed below.

   Consider the classes $c \in \{1, \ldots, C\}$ and feature sets $m \in \{1, \ldots, M\}$. The empirical probability of the class c based on the feature m is denoted as

$$\hat{p}_{mc} = \frac{1}{|\{x_i : x_i \in \mathbb{R}_m\}|} \sum_{x_i \in \mathbb{R}_m} \mathbf{1}_{\{y_i \in c\}}$$

   The cross-entropy criterion is defined as follows:

$$E(X) = -\sum_{c=1}^{C} \hat{p}_{mc} \log \hat{p}_{mc}$$

   The Gini Impurity criterion is defined as follows:

$$G(X) = \sum_{c=1}^{C} \hat{p}_{mc}(1 - \hat{p}_{mc})$$

3. Once a splitting criterion is selected, the nodes are recursively splitted until it reaches a pre-specified maximum depth or when all the data points in a node belong to the same class. Some ways to avoid overfitting is to reduce the maximum depth, or increase the minimum required number of samples in a node.

4. In the prediction step, the $\hat{y}$ is predicted as

$$\hat{y} = \text{argmax}_c \hat{p}_{mc}$$

However, while the pruning methods of decision trees may be fruitful in reducing the phenomenon of overfitting, it is generally still a major overhead when it comes to decision trees. To overcome such issue, the random forest classifier is a good alternative as it utilizes ideologies of *bagging* and *bootstrapping* of samples to reduce the variance of the classifier. The algorithm of random forest is as follows:

1. Randomly sample n rows of training data with replacement (bootstrapping).

2. Randomly sample d predictors from the training data.

3. Construct a decision tree based on the sampled data of n rows and d features. Save the decision tree.

4. Repeat steps 1-3 for a total of B times. A good selection of B is typically between 100 to 500, but this value varies by dataset and is dependent on the features and data size.

5. In the prediction step, the $\hat{y}$ is predicted as

$$\hat{y} = \text{argmax}_c \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x)$$

where $\hat{f}_b(x) = \{\hat{p}_{m1}, \ldots, \hat{p}_{mC}\}$ is the probabilities that x is classified into class c. One can think of this step as the average voting of B decision trees based on the probabilities of x being classified in to class c.

The ways to reduce overfitting is the same as those mentioned for decision trees. Additionally, one can also tune the number decision tree estimators to improve classification results. In this project, we ran a grid-search cross validation over the minimum number of samples at each split (2, 20, 50, 200, 500, 2000), while setting the number of estimators to be 500, the splitting criterion to be the Gini Impurity criterion, and the maximum number of features considered in each split to be $\sqrt{d}$, where d is the number of features. We also set the number of bootstrap sampled rows drawn to train each base estimator to be 30% of the total training samples.

3 Results and Discussion

For each feature extraction method, features extracted from the training image sets were used to fit a random forest model. The following hyperparameters were pre-determined: `n_estimators` (number of trees in the forest) was 500, `|max_samples|`(maximum number of samples drawn to build each base estimator) was 0.3 of total training samples. Two other hyperparameters were tuned using the validation set: `max_features` (maximum number of features considered when looking for the best split): square root or all of the total input features, and `min_samples_split` (minimum number of samples at a split): 2,20, 50. Metrics used for parameter tuning was the mean classification accuracy. After finding the best estimator, features from the test image set was input into the trained model to predict classification. The performance of each model was assessed by the class-wise recall and precision value, ie. for each class c

$$\text{Precision}^{(c)} = \frac{\text{True Positive}^{(c)}}{\text{True Positive}^{(c)} + \text{False Positive}^{(c)}}$$

$$\text{Recall}^{(c)} = \frac{\text{True Positive}^{(c)}}{\text{True Positive}^{(c)} + \text{False Negative}^{(c)}}$$

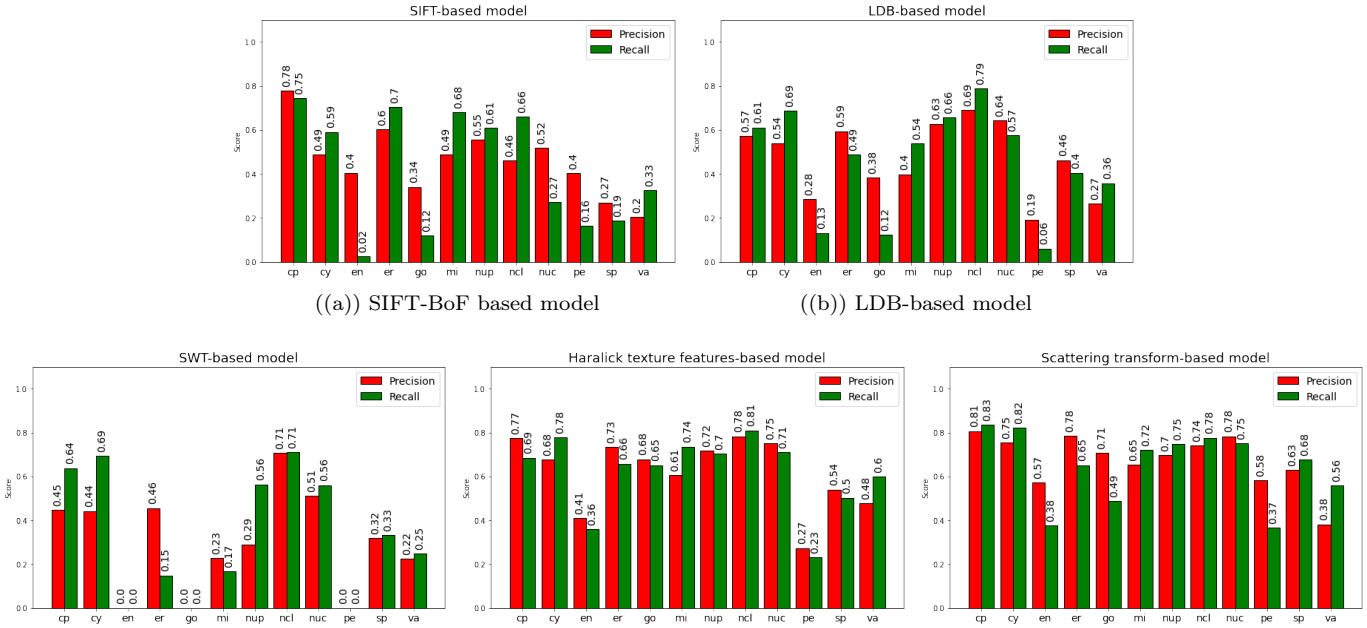
The results for each model are shown in Figure 10 while the accuracies are described in Table 1. Detailed results will be discussed in the upcoming sections.

Table 1: Overall accuracy of test set classification results for random forest models using different feature extraction methods.

	Overall accuracy		
Feature extraction method	Training	Validation	Test
SIFT-BoF	0.56	0.48	0.51
LDB	0.91	0.55	0.54
SWT-DCT	0.74	0.45	0.42
Haralick texture-Intensity	0.94	0.68	0.67
Scattering transform	0.95	0.71	0.71

3.1 Scale Invariant Feature Transform (SIFT) + Bag of Features (BoF)

The SIFT-based model, with the best hyper-parameters tuned to `max_features="sqrt"` and `min_sample_split=20`, produced a worse than expected result. Its strength of being scale and orientation invariant did not help with the cause.



((a)) SIFT-BoF based model

((b)) LDB-based model

((c)) SWT-DCT based model

((d)) Haralick texture-Intensity based model

((e)) Scattering transform based model

Figure 10: Precision and recall scores of test data of each group based on different models

As the algorithm is based mainly on finding keypoints in images via observing the orientations and directions of each pixel, it became less effective in this case since there is a significant number of images where keypoints are not found. The low contrast images, paired with the noise input for the Guassian blur, makes it near impossible to find specific patterns within images.

With the large number of images having close to no keypoints, it is inevitable that these images will be classified into the same category, which in this case is the nucleolus class. The classes of images with such issues are the endosome and golgi classes, resulting in the classes' extremely low recall scores. This also explains the low precision value of nucleolus as shown in Figure 10(a).

In addition to the low classification rate of this algorithm, it is also computationally expensive. As we have to loop through (almost) all pixels in each image and its Gaussian blurred versions to determine the keypoints, the SIFT algorithm took a significant amount of computation time, yet still led to bad results. This method would be significantly more useful under two conditions: good contrasts on all images and training time not being the most important consideration factor (ie. not used for real time classification).

3.2 Stationary Wavelet Transform (SWT) + Discrete Cosine Transform (DCT)

As another shift-invariant feature extractor, the SWT is useful for detecting edges in the images. Here, the hyper-parameters were tuned to `max_features="sqrt"` and `min_samples_split=20`, plus an additional setting of `max_depth=50`. However, this model did extremely poor. While the lack of grid search parameters may have contributed to this, the main issue stems from the type of feature extractor used, ie. using DCT as our dimension reduction procedure may not be the best idea. Compared to Qayyum et al.'s work, cell images differ from facial expressions in the sense that cells can be shifted and rotated whereas human faces have a fixed orientation.

This does not mean that the SWT is not a good feature extractor, especially when there are various papers proving otherwise. What may be done instead is to use different feature selection methods. For instance, Nayak et al. [14] paired SWT with energy and entropy based features for pathological brain detection, while Wang and Cui [22] paired SWT with Principal Component Analysis (PCA) for hyperspectral image feature classification purposes.

3.3 Local Discriminant Basis (LDB)

In its own ways, the LDB-based model with parameters `max_features="sqrt"` and `min_samples_split=2` was a successful model achieving a higher accuracy than both the SIFT and SWT based models. A strength of the LDB is that our feature extraction results may be significantly different depending on the type of wavelets and discriminant measure selected, making the algorithm robust. As the resulting number of coefficients selected is quite small compared to the other feature extraction techniques, this also results in a quick model fitting process.

However, there are still some caveats associated with this method resulting in the lower scores than the Haralick texture-Intensity and Scattering transform based models. One of them is the interpretation of images as 1D-signals due to the unavailability of 2D-LDB packages. In this sense, the vertical and diagonal features were not picked up by the algorithm. This led to some information loss resulting in bad generalization.

Additionally, the standard wavelet decomposition algorithm being non-redundant also means that this algorithm is not shift invariant, meaning a shifted or rotated version of an image may lead to different classification results. In Saito's paper [19], he used the "spin-cycle" technique to introduce augmented versions of images to improve generalization results. For the sake of a fair comparison with the original paper, this method was not implemented in our experiments.

3.4 Gray Level Co-occurrence Matrix based Texture Features

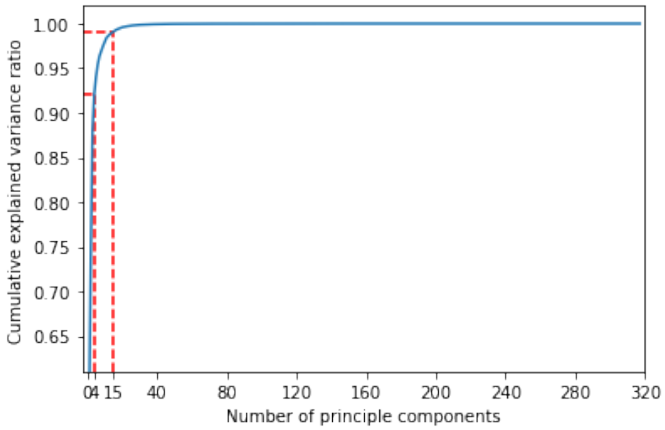


Figure 11: Cumulative explained variance ratio as a function of principle components in PCA of intensity and Haralick texture features.

Concatenating intensity measurements and Haralick texure features resulted in a feature vector of size 317. Dimension reduction was performed using PCA after standardizing the features. Cumulative explained variance ratio as a function of the number of principle components was shown in Figure 11. It could be observed that first 16 PC's accounted for over 99% of total variance. It was expected that only a few principle components accounted for the majority of the total variance within the Haralick texture features extracted. For the same image, features calculated from co-occurrence matrices with adjacency offsets defined by different direction and distance were not expected to vary substantially. In order to reduce computation time for fitting the random forest model, 16 principle components with largest explained variance were selected as the input for the model. A model was also fitted with all the features without dimension reduction, no further improvement in model performance was observed, thus was not shown in the report.

After gridsearch using the validation set, the random forest model hyperparameter `min_samples_split` was chosen to be 2 and `max_features` was chosen to be 'sqrt'. The model performance was evaluated using the test set based on class-wise precision and recall (Figure 10(d)), as well as the overall accuracy (Table 1). The overall accuracy of class prediction on test set was 68%. Among all the protein subcellular location classes, two of them had especially low prediction accuracy: endosome (en) and peroxisome (pe). Possible reasons could be: (1) these two classes were low in sample size in training, validation, and test sets (Figure S1); (2) many images within these two classes had low signal intensity in the green channel, thus the features extracted from these images carried very little information.

The Gini importance of the input features to the random forest model was also calculated. It was defined as the normalized total reduction in Gini impurity brought by a certain feature. Since the input features into the random forest model were the principle components from PCA, the three input features with the highest importance were PC 1,6, and 3,

in the order of descending importance. Looking into the loading of original features on these three PC's, it was found that the features with largest absolute values of loadings were entropy, contrast and correlation calculated from co-occurrence matrices with different direction and distance adjacency offset. It indicated that same Haralick texture features from different co-occurrence matrices were highly correlated to each other, which went along with the PCA results that the first few PC's explained for majority of the variance. Among the three important texture measures identified, entropy and contrast were referred to as edge textures, while correlation fell into the category of interior features [8]. When a edge appear in an image, there would be abrupt changes in the pixel values locally, resulting in large differences in pixel pairs within this neighborhood. Contrast, as its names suggested, would take high values when such big differences occurred. Entropy measured the complexity of the image patch. High entropy values often resulted from irregular or incoherent edges, while low entropy values from straighter edges. Correlation, on the other hand, belonged to a group of texture features that yielded high value in the region where there was no coherent edges, but more subtle and irregular variations, thus referred to as interior texture features. One advantage of using Haralick texture features to fit the random forest model is that the features are highly interpretable and are informative of what aspects of the image texture are captured. The coherence and complexity of edges might be the most important features in the current classification task.

3.5 Scattering transform

Result of PCA on scattering transform-based features was shown in Figure 12. First 32 principle components explained for over 99% of total variance, thus were selected as input to the random forest model.Hyperparameters were tuned to `min_samples_split`=2 and `max_features`='sqrt'. The class-wise precision and recall of test set prediction and overall prediction accuracy of the model were shown in Figure 10(e) and Table 1 respectively. The test set mean prediction accuracy of the scattering transform-based model was 71%, the highest among all five feature extraction methods. It was observed that the prediction performance for the endosome and peroxisome classes was improved compared with the Haralick texture features-based model.

One reason that the scattering transform-based model outperformed others was that it not only had shift-invariant characteristics, but also recovered more high frequency information that was lost in other wavelet transformation methods. Also, since it did not use key points detection, it did not suffer from low contrast problem as SIFT did. However, computation time for calculating scattering transform coefficients for more than 2 layers, and with more direction and scales would increase exponentially.

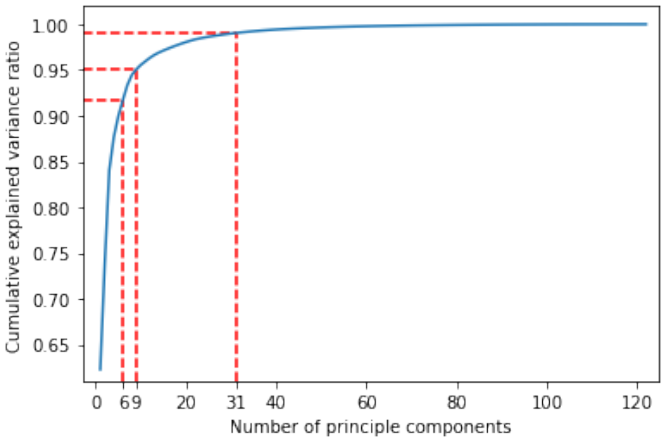


Figure 12: Cumulative explained variance ratio as a function of principle components in PCA of scattering transform-based features.

4 Conclusion

Compared to the random forest model from the original paper, it may seem that all the proposed methods here did not do well, but that is not necessarily the case. Pärnamaa and Parts built their random forest model using 435 different features which are a concatenation of multiple feature extraction techniques such as the Haralick texture filters and the Gabor and

Zernike filters, whilst we built our random forest models using one feature extraction techniques each. Concatenating some of our feature sets such as the Haralick and intensity features with the scatter transform features may possibly yield better results.

Some important takeaways from this experiment are the strength of wavelet transform based feature extraction and the weakness of the SIFT algorithm. While the performances of LDB and SWT-based models on the test set were subpar, both showed great potential for improvements, where 2D-LDB is required for better local feature extractions and a different feature selector should be used on the SWT-based model. The scattering transform can also be improved with more convolution layers, wavelet transformation orientations, and scattering scales. On the other hand, SIFT produced some disappointing results despite it being considered a very strong feature extraction technique due to low contrasts of our dataset.

Other things to consider when trying to build a better model than the ones described in this paper are data augmentation and K-Fold cross-validation. The latter would help reduce our model variances while the former is important in balancing out the sample size of each class. Some data augmentation techniques include rotation of images and adding white noise to a random set of images. For the purpose of direct comparison with the original paper, these methods were not experimented.

References

[1] Craig Belcher and Yingzi Du. "Region-based SIFT approach to iris recognition". In: *Optics and lasers in engineering* 47.1 (Jan. 2009), pp. 139–147. ISSN: 01438166. DOI: 10.1016/j.optlaseng.2008.07.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S0143816608001644 (visited on 06/09/2021).

[2] Simone Bianco et al. *Facial Expression Recognition Using Stationary Wavelet Transform Features*. 2017. DOI: 10.1155/2017/9854050. URL: https://doi.org/10.1155/2017/9854050.

[3] Joan Bruna and Stephane Mallat. "Classification with scattering operators". In: IEEE, June 2011, pp. 1561–1566. ISBN: 978-1-4577-0394-2. DOI: 10.1109/{CVPR}.2011.5995635. URL: http://ieeexplore.ieee.org/document/5995635/ (visited on 06/08/2021).

[4] Warren Cheung and Ghassan Hamarneh. "N-sift: n-dimensional scale invariant feature transform for matching medical images". In: *2007 4th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*. IEEE, Apr. 2007, pp. 720–723. ISBN: 1-4244-0671-4. DOI: 10.1109/{ISBI}.2007.356953. URL: http://ieeexplore.ieee.org/document/4193387/ (visited on 06/09/2021).

[5] Yolanda T Chong et al. "Yeast Proteome Dynamics from Single Cell Imaging and Automated Analysis." In: *Cell* 161.6 (June 2015), pp. 1413–1424. DOI: 10.1016/j.cell.2015.04.051. URL: http://dx.doi.org/10.1016/j.cell.2015.04.051 (visited on 10/31/2018).

[6] Ronald R. Coifman and Mladen Victor Wickerhauser. *Entropy-based Algorithms for Best Basis Selection*. URL: https://www.math.wustl.edu/~victor/papers/ebafbbs.pdf.

[7] Xiang Fan and Shun-ren Xia. "Feature based automatic stitching of microscopic images". In: *Advanced intelligent computing theories and applications. with aspects of contemporary intelligent computing techniques*. Ed. by De-Shuang Huang, Laurent Heutte, and Marco Loog. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 791–800. ISBN: 978-3-540-74281-4. DOI: 10.1007/978-3-540-74282-1_88. URL: http://link.springer.com/10.1007/978-3-540-74282-1%5C_88 (visited on 06/09/2021).

[8] Mryka Hall-Beyer. "Practical guidelines for choosing GLCM textures to use in landscape classification tasks over a range of moderate spatial scales". In: *International journal of remote sensing* 38.5 (Mar. 2017), pp. 1312–1338. ISSN: 0143-1161. DOI: 10.1080/01431161.2016.1278314. URL: https://www.tandfonline.com/doi/full/10.1080/01431161.2016.1278314 (visited on 06/09/2021).

[9] Robert M. Haralick, K. Shanmugam, and Its'Hak Dinstein. "Textural Features for Image Classification". In: *IEEE transactions on systems, man, and cybernetics* 3.6 (Nov. 1973), pp. 610–621. ISSN: 0018-9472. DOI: 10.1109/{TSMC}.1973.4309314. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4309314 (visited on 06/08/2021).

[10] Hong Huang et al. "Spatial-spectral local discriminant projection for dimensionality reduction of hyperspectral image". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 156 (Oct. 2019), pp. 77–93. ISSN: 09242716. DOI: 10.1016/j.isprsjprs.2019.06.018. URL: https://linkinghub.elsevier.com/retrieve/pii/S0924271619301595 (visited on 06/09/2021).

[11] Won-Ki Huh et al. "Global analysis of protein localization in budding yeast." In: *Nature* 425.6959 (Oct. 2003), pp. 686–691. ISSN: 1476-4687. DOI: 10.1038/nature02026. URL: http://dx.doi.org/10.1038/nature02026 (visited on 06/09/2021).

[12] David G. Lowe. "Distinctive Image Features from Scale Invariant Keypoints". In: (2004). DOI: 10.1023/B:VISI.0000029664.99615.94. URL: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[13] Shervin Minaee, AmirAli Abdolrashidi, and Yao Wang. "Iris recognition using scattering transform and textural features". In: IEEE, Aug. 2015, pp. 37–42. ISBN: 978-1-4673-9169-6. DOI: 10.1109/{DSP}-{SPE}.2015.7369524. URL: http://ieeexplore.ieee.org/document/7369524/ (visited on 06/09/2021).

[14] Deepak Ranjan Nayak, Ratnakar Dash, and Banshidhar Majhi. "Stationary Wavelet Transform and AdaBoost with SVM Based Pathological Brain Detection in MRI Scanning". In: 16 (2 2017). DOI: 10.2174/1871527315666161024142036. URL: https://doi.org/10.2174/1871527315666161024142036.

[15] Stephen O'Hara and Bruce A. Draper. "Introduction to the Bag of Features Paradigm for Image Classification and Retrieval". In: *CoRR* abs/1101.3354 (2011). arXiv: 1101.3354. URL: http://arxiv.org/abs/1101.3354.

[16] Nobuyuki Otsu. "A Threshold Selection Method from Gray-Level Histograms". In: *IEEE transactions on systems, man, and cybernetics* 9.1 (Jan. 1979), pp. 62–66. ISSN: 0018-9472. DOI: 10.1109/{TSMC}.1979.4310076. URL: http://ieeexplore.ieee.org/document/4310076/ (visited on 06/08/2021).

[17] Tanel Pärnamaa and Leopold Parts. "Accurate Classification of Protein Subcellular Localization from High-Throughput Microscopy Images Using Deep Learning." In: *G3 (Bethesda, Md.)* 7.5 (May 2017), pp. 1385–1392. DOI: 10.1534/g3.116.033654. URL: http://dx.doi.org/10.1534/g3.116.033654 (visited on 09/30/2019).

[18] Nasir Rajpoot. "Local discriminant wavelet packet basis for texture classification". In: *Wavelets: Applications in Signal and Image Processing X*. Ed. by Michael A. Unser, Akram Aldroubi, and Andrew F. Laine. Vol. 5207. SPIE Proceedings. SPIE, Nov. 2003, p. 774. DOI: 10.1117/12.507681. URL: http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.507681 (visited on 06/09/2021).

[19] Naoki Saito and Ronald R Coifman. "Local Discriminant Basis and their applications". In: *Journal of Mathematical Imaging and Vision* 5 (1995). DOI: 10.1007/BF01250288. URL: https://doi.org/10.1007/BF01250288.

[20] Naoki Saito et al. *Discriminant feature extraction using empirical probability density estimation and a local basis library*. Nov. 2001.

[21] David W. Scott. *Average Shifted Histograms: Effective Nonparametric Density Estimators in Several Dimensions*. Sept. 1985. URL: http://www.jstor.org/stable/2241123?origin=JSTOR-pdf.

[22] Yonghui Wang and Suxia Cui. "Hyperspectral image feature classification using stationary wavelet transform". In: *2014 International Conference on Wavelet Analysis and Pattern Recognition*. 2014, pp. 104–108. DOI: 10.1109/ICWAPR.2014.6961299.

[23] Wikipedia. *Discrete Wavelet Transform*. URL: https://en.wikipedia.org/wiki/Discrete_wavelet_transform.

A Resources

The detailed codes for reproducing the results can be obtained on Zeng Fung's Github repository here.

B Supplement figures

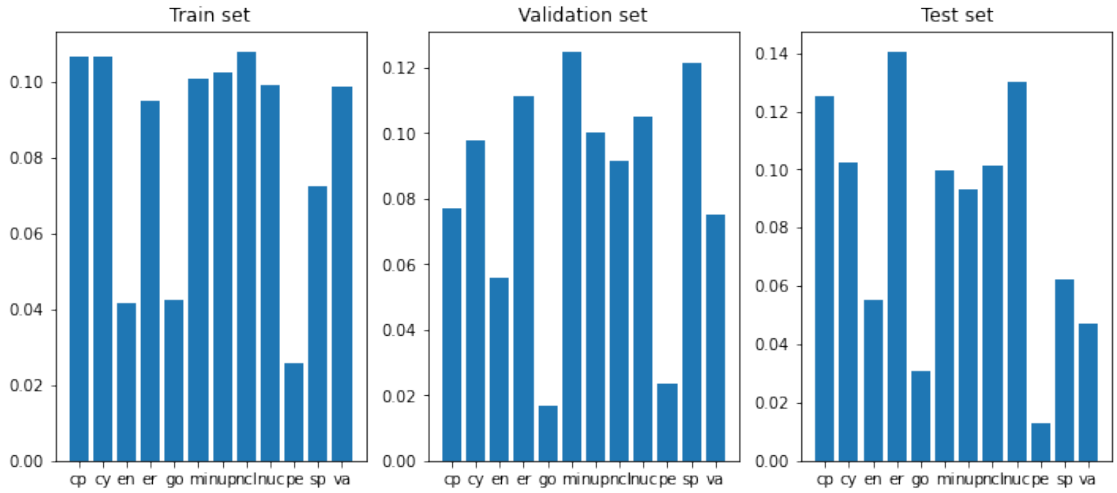
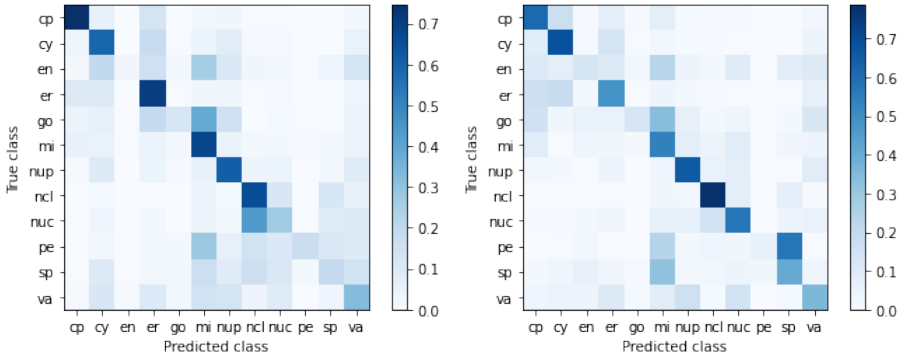
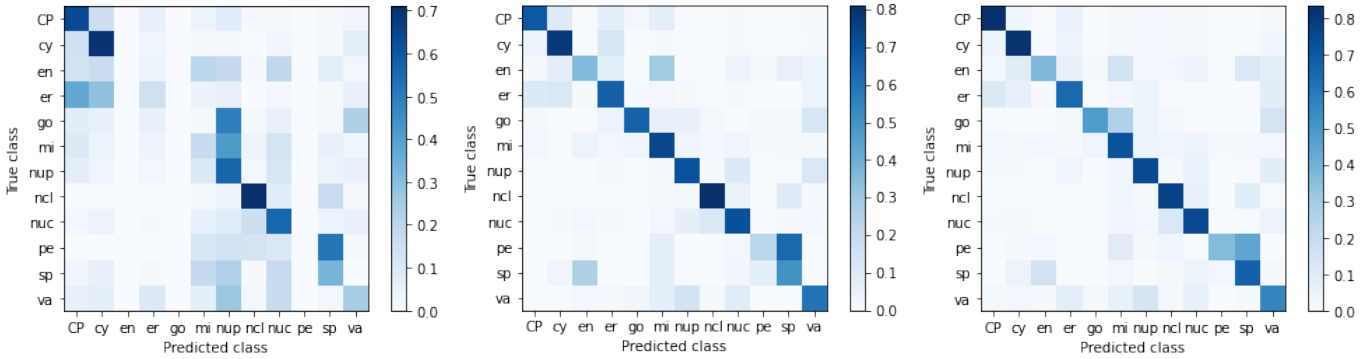


Figure S1: Proportion of each class in training, validation, and test sets.



((a)) SIFT-BoF based model ((b)) LDB-based model



((c)) SWT-DCT based model ((d)) Haralick texture-Intensity based model ((e)) Scattering transform based model

Figure S2: Test set prediction confusion matrix from different models.