IP on AP: Image Processing on the Automata Processor

A Thesis

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

in partial fulfillment
of the requirements for the degree

Master of Science

by

Tiffany Thanh Ly

May 2017

APPROVAL SHEET

This Thesis
is submitted in partial fulfillment of the requirements
for the degree of
Master of Science

Author Signature: ______________________

This Thesis has been read and approved by the examining committee:

Advisor: Scott Acton

Committee Member: Kevin Skadron

Committee Member: Gustavo Rhode

Committee Member: ______________________

Committee Member: ______________________

Committee Member: ______________________

Accepted for the School of Engineering and Applied Science:

[illegible]

Craig H. Benson, School of Engineering and Applied Science

May 2017

Abstract

The Automata Processor is a novel hardware accelerator that can perform pattern matching in parallel. To date, this pattern matching has been limited to one-dimensional problems that can be implemented as flexible string-matching methods such as those found in genomics. In this thesis, we present a novel process of implementing image retrieval using a multinary representation for deployment on an automata framework. Images are encoded into discriminative and unique regular expression descriptors in such a way that can be used for classification purposes. The regular expression descriptors are streamed through sets of non-deterministic finite automata (NFA).

To improve performance of this multi-dimensional classification problem, we transform discriminative feature descriptors using a cumulative distribution transform. The transformed features are encoded into regular expressions which can be executed on the automata processor.

The thesis also highlights methods of evaluating the similarity between images using these regular expressions in the automata processor. Our image retrieval and classification method improves on classification accuracy and achieves a run-time of less than one one-hundredth of a second per image which represents a three-fold improvement over competing architectures.

Acknowledgements

This thesis could not have been done without the help of many people. First, I would like to thank my advisor, Dr. Scott Acton, for the tremendous support, wisdom and mentorship. I would also like to thank Dr. Kevin Skadron for allowing me to collaborate with the Center for Automata Processing, and for providing access to the resources to complete this project. I would also like to extend gratitude the rest of my committee, Dr. Gustavo Rohde. I would also like to thank the members of Virginia Image and Video Analysis lab for the abundance of support and guidance- especially Dr. Ritu Sarkar. Finally, I would like to thank my parents, Phong and Patricia, and my brother, Anhphan, for all the support.

TABLE OF CONTENTS

LIST OF TABLES

List of Figures

CHAPTER 1

INTRODUCTION

Current image retrieval problems involve massive datasets particularly in object detection and medical related applications. However, image retrieval is still a challenging problem in the field of image processing along two fundamental avenues – feature extraction and classification. One of the major challenges of large scale image retrieval is that it requires searching through large databases where images specific to an object category may have significant content variations. To accurately classify images, it is necessary to extract discriminant features that capture the intra-category variation while making the discrimination between categories more prominent. This specificity typically leads to generating high dimensional feature descriptors. Additionally, the problem necessitates the design of an efficacious classifier or similarity-based measure to accurately classify images. Both classifier design and similarity based image search for large scale retrieval are computationally expensive and consequently, online image retrieval applications require considerable parallelism.

The Automata Processor (AP) is a scalable hardware accelerator that can perform highly complex pattern matching applications [3]. It executes parallel processing of thousands of non-deterministic finite automata state machines that represent different regular expression patterns. This is ideal for pattern matching applications with large datasets, including Brill tagging, bioinformatics, and machine learning [4–6]. In [5], Roy *et al.* designed a method on the automata to solve the DNA motif search problem and found speed-ups compared to conventional methods. This automata framework has the potential to provide acceleration on many image retrieval applications, particularly those with massive datasets that may require parallelism.

1.1 Overall Methodology and Challenges

In order to take advantage of the automata framework, images need to be converted into strings. We developed a methodology to apply image classification on this processor that determines a unique pattern of character symbols for every image class, as shown in Figure 1.1. First, features are extracted from the image to attain a discriminative descriptor. The descriptor values from every image are then encoded into regular expression characters. Unique regular expression patterns of an image class can be represented as automata symbols. Thus, a regular expression descriptor from any image can be classified using this automata image classifier.

There are still a few challenges using this methodology. Applications are limited by the current AP architecture, including the number of counter elements allowed per automata and the lack of mathematical operator elements (i.e. addition, subtraction, multiplication, etc.). Therefore, popular similarity metrics and classifiers, such as Kullback-Leibler (KL) divergence [7], k-means, minimum distance and maximum likelihood, would be difficult to directly use on this framework. Thus, these limitations demand more discriminative feature descriptors for symbol matching classification.

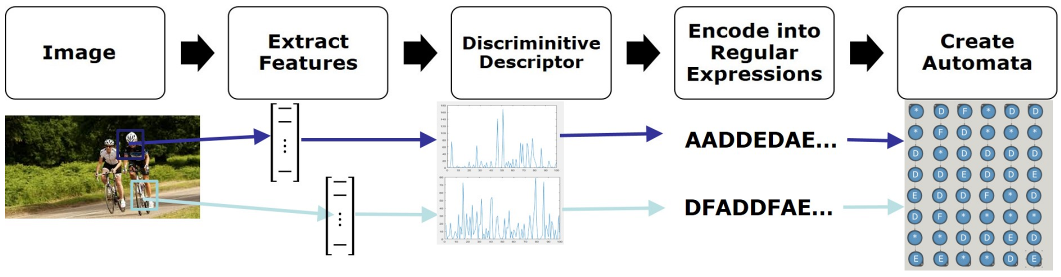


Figure 1.1: Overview of image classification on the automata.

1.2 Feature Extraction

One of the key contributions of this work is finding an appropriate feature representation for classification on an automata framework. A number of methods extract image characteristics, such as color intensity, shape, and texture etc. [8], and encode them into histograms. Other methods compute local features by exploiting neighborhood information, such as histogram of oriented gradients [9], local morphology [10, 11], or blob-based SIFT features [12]. Some works have employed encoding techniques, as with the bag of words [13], where the local features from each image in a dataset are accumulated as histograms. In [14], the authors exploited spatial relationships among local features to compute image representation. These mentioned descriptors are extensively used in object detection and classification. However, to increase precision and retain more information, they tend to become very high-dimensional descriptors.

Dimensionality reduction techniques aim to keep significant information such as applying principle component analysis (PCA) [15], Fisher vectors, Gaussian mixture model [16] etc. However, the descriptors are still significantly large and need appropriate similarity measures to compare features of images or learn a classifier using class labels associated with the images.

1.3 Feature transforms

Transforms can be beneficial for image classification to further separate feature descriptors that belong to different classes. These transformations, such as PCA [17], use transformations to convert data to linearly separated data. The cumulative distribution transform is a non-linear transformation that has been successfully used to improve linear separability of patterns and increase classification accuracy [18, 19]. Constructing feature representations that are distinct and more separable between classes is advantageous for image classification, particularly on an automata framework.

1.4 Thesis summary

The goal is to determine an efficient methodology of applying image retrieval problems on an automata framework. This project also seeks to determine whether this framework is appropriate for image processing applications. Chapter **2** provides an overview of the Automata Processor and the necessities required to apply image retrieval on this framework. The structure of the automata provides some limitations on image representation and classification. Chapter **3** proposes a method of encoding an image into regular expressions for classification on the automata. Chapter **4** uses the cumulative distribution transform to improve separation of features between classes to increase classification accuracy. It also suggests another novel method of image classification using the automata. Chapter **5** summarizes the findings and provide suggestions for future work.

CHAPTER 2

BACKGROUND OF THE AUTOMATA PROCESSOR AND APPLICATION TO IMAGE PROCESSING

2.1 Automata Processor

The AP is a scalable hardware that is specific to accelerating pattern matching application. The processor can compute thousands of non-deterministic finite automata in parallel by finding regular expression patterns in an input data stream. Users can program automata structures and load them onto the hardware.

Automata on the AP are made up of state transitions elements (STE) that can be programmed to match a set of symbols or arbitrary characters classes. A single STE can be programmed to implement combinations of 0 to 255. A start STE of an automaton is activated when symbols in the input data stream matches with a symbol in the STE. There are two types of start STEs. The start-of-data STE is only activated by the first input symbol. The all-input-start STE can be activated by any symbol in the input string. The AP board also contains Boolean elements and counter elements, which broaden functionality.

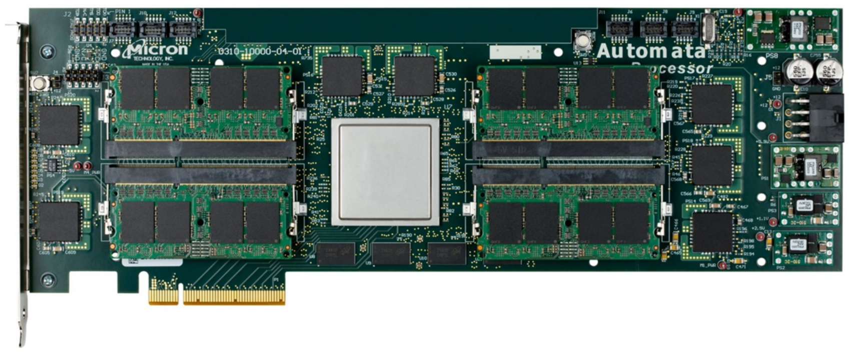


Figure 2.1: The Automata Processor PCIe board.

Boolean elements can be single input, such as an inverter, or multi-input, i.e. AND, OR, NAND, NOR, sum-of-product and product-of-sum. Counter elements can be programmed to a target value which, once reached, determines the set behavior. These elements are activated by a routing network of matching STEs. An automaton reports to post processors on the board when the final reporting state is activated.

STEs on the AP can be programmed using the AP Software Development Kit (SDK). It uses a graphical tool, the AP Workbench, that allows users to design and visualize the non-finite state machines (ANML-NFA). The ANML-NFAs are routed and optimized to be compiled into Finite State Machines (AP-FSM) which are then loaded onto the AP hardware [20, 21].

Figure 2.1 shows the Automata Processor PCIe board which is plugged into a server. The AP has two half cores with three output regions. These regions consist of local memory storage which each contain 1024 output event vectors, or report vectors (when an automata reports). So the maximum number of reporting events allowed per symbol cycle on an AP is 6144 events/cycle. This means that a design cannot have more than 6144 reporting elements. Once all symbol processing is complete, these output event vectors are transferred to the output buffer, which it can then be read by external hardware.

The first generation AP is a PCI-Express board mediated by a driver, but can also be programmed to raise interrupts when report bits are set. A chip holds 49K STEs, and the first generation AP contains 32 chips, so a single AP

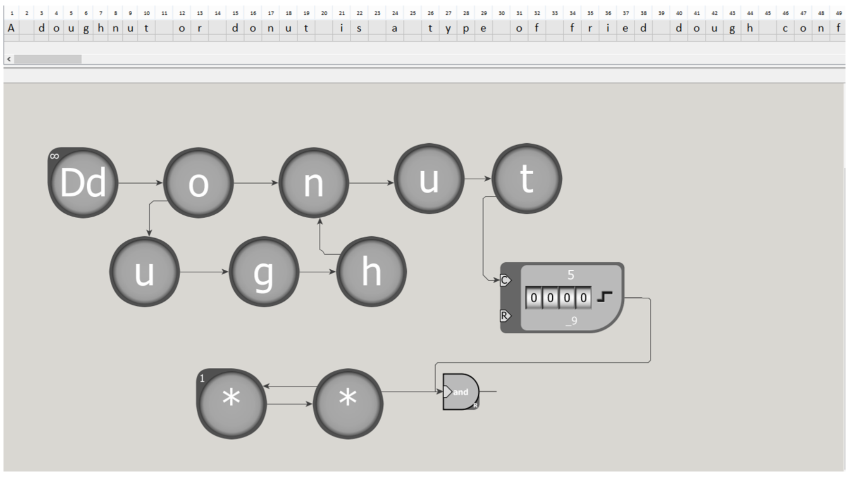


Figure 2.2: Example of pattern matching on the AP Workbench. This automata [Dd](o| |ough)nut is found five times and the input data stream is an even number. The input data stream is shown at the top of the figure.

board holds up to 1,536K STEs. All active STEs receive a new input symbol on every clock cycle. This means that thousands of state machines can be executed in parallel [3].

2.1.1 Example of Pattern Matching on the automata

Figure 2.2 gives an example on the AP Workbench which uses Automata Network Markup Language (ANML). The example reports when the word "donut" or "doughnut" is found five times and the input symbol is an even number character in the data stream. The input data stream is also a set of regular expression or numerical characters. The counter is set to five and is activated once both of its inputs are triggered. The [$*$] is essentially a "don't care," which is a special AP character that can match with any symbol. The AND element is set as the reporting element which reports once both the counter reaches five and the second [*] is matched.

2.2 Applications on the Automata Processor

There are a number of applications that have been explored on this processor, including machine learning [6], natural language processing [4], bioinformatics [5, 22], high energy physics [20], and data analytics [23–25]. Many of these applications are string based searches. In [23], Bo *et al.* looked at Entity Resolution (ER) which searches through a social network database or multiple databases for common entities, or variations of a name. Compared to other text searching methods, they found that their AP-accelerated method found 400× speedup and better accuracy due to the AP's ability to handle flexible matching. Sequential pattern mining is another data analytics application tested on the AP that searches for frequencies of hierarchical patterns or itemsets in a database [24]. Searching for DNA motifs is another text-based application that found potential speed up in the AP [5, 22].

The AP accelerates string-based pattern matching but it does not necessarily limit usage to text-based pattern applications. [20] showed a proof of concept of applying pattern matching in high energy physics onto the automata framework. They detect patterns in the trajectory of particles by encoding the possible trajectories into four pixel addresses. Another application that was tested on the AP and more similar to image processing application is machine learning for classification. In [6], a random forest algorithm was implemented on the AP where binary decision trees was used for classification. They extracted features from a MNIST handwritten dataset and a Twitter dataset. These features were assembled into STEs on the AP to create decision trees.

The implementations of vast applications on the AP have found massive speedups due to high parallelism of pattern searches. This motivates us to design an automata classifier that could potentially speed up image classification particularly on large dataset. The automata structure does limit us from using typical similarity metrics in image retrieval. However, there are resemblances between image retrieval and these applications that allow the manipulation of an automata framework to design a novel image classifier.

CHAPTER 3
FEATURE EXTRACTION FOR AUTOMATA CLASSIFICATION

In this section, we provide a method of encoding feature descriptors that can be used on an automata framework. Two feature extraction methods are presented to generate feature descriptors that are then converted into regular expression patterns. These descriptors are represented on the automata for image classification.

3.1 Image Similarity using Multinary Representation

The AP has potential speed up on image retrieval applications with massive datasets by performing a massive number of image matching operations in parallel. To take advantage of this framework, images/image features need to be encoded into regular expressions or loaded onto the AP of its other programming modalities [26, 27]. These strings of characters which represent an image or a particular category of images are ultimately encoded as state machines on the automata. Test images are streamed as input and classified to an image category. Instead of using a similarity measure to classify feature descriptors, this image retrieval method uses exact matching of state machines. However, in Section 3.5, we propose a method to relax the need for exact matching, but in both scenarios the image descriptors must be highly distinctive. Figure 1.1 shows the overview of our method of implementing image retrieval on the automata structures. The two crucial steps for image retrieval on the automata are to extract discriminant features and mapping the features as regular expressions. Once different image categories have unique regular expression patterns, they can be represented as automata.

3.2 Feature Extraction

In the literature, both global and local image features have been employed for the task of image classification. Features extracted from local image regions are often aggregated in some manner to obtain a global feature [7], [8]. The features extracted form the images can be used for both supervised and unsupervised image classification. For unsupervised image classification, along with extracting discriminant feature descriptor, a robust similarity measure is also necessary. Based on the type of feature being used, or the type of feature encoding used, different similarity measures may be preferred. For example in [8], a histogram matching kernel was implemented on spatial pyramid features. Whereas in [14], sparse codes are compared using a compression-based similarity measure. For supervised classification, a robust classifier needs to be designed, which again demands modeling of proper classifier functions. Implementing the retrieval on AP can be an advantage in this context. Since the AP only allows for matching based on regular expressions, once the feature descriptors of the images are extracted, a robust mapping of the features to regular expressions is the only requisite, in contrast to designing case-specific similarity measures or classifiers.

3.2.1 Superpixel generation

Superpixels are clusters of pixels grouped together by distance and local similarity. Superpixels have been used to improve image applications including image segmentation and classification [28]. They can be attained using many different algorithms, including graph-based, which treats each pixel as a node and assigns each a weight [29], gradient-ascent-based [30], and simple linear iterative clustering (SLIC) [1, 31, 32].

We used the SLIC method by [1, 31] to generate superpixels by clustering based on the CIELAB color space and pixel position. This method was chosen because the number of superpixels can be chosen by the user, are smooth and approximately equal in size. The superpixel generation algorithm is fairly inexpensive. It takes into account the color similarity and spatial proximity of pixels within each superpixels which are regulated using a normalized Euclidean distance measurement.

The SLIC algorithm starts by initializing cluster centers based on the number of superpixels, K, and image size, N, to get the grid interval $S = \sqrt{N/K}$. Pixels are assigned to each cluster centers, $C_k = [l_k, a_k, b_k, x_k, y_k]^T$, based on a distance measurement, Equation 3.1, which takes color and spatial location in consideration. Once pixels are reassigned to cluster centers, the cluster centers are recomputed until it reaches convergence.

$$d_{lab} = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2}$$

$$d_{xy} = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2}$$

$$D_S = d_{lab} + \frac{m}{S} d_{xy} \tag{3.1}$$

In the distance equation, m controls how compact a superpixel is, and ranges from [1,20].

Compared to other superpixel generation methods, SLIC is much faster, even for larger images [1, 31].From Figure 3.1, which shows an example of an image that uses the SLIC algorithm with roughly 1200 superpixels, we can

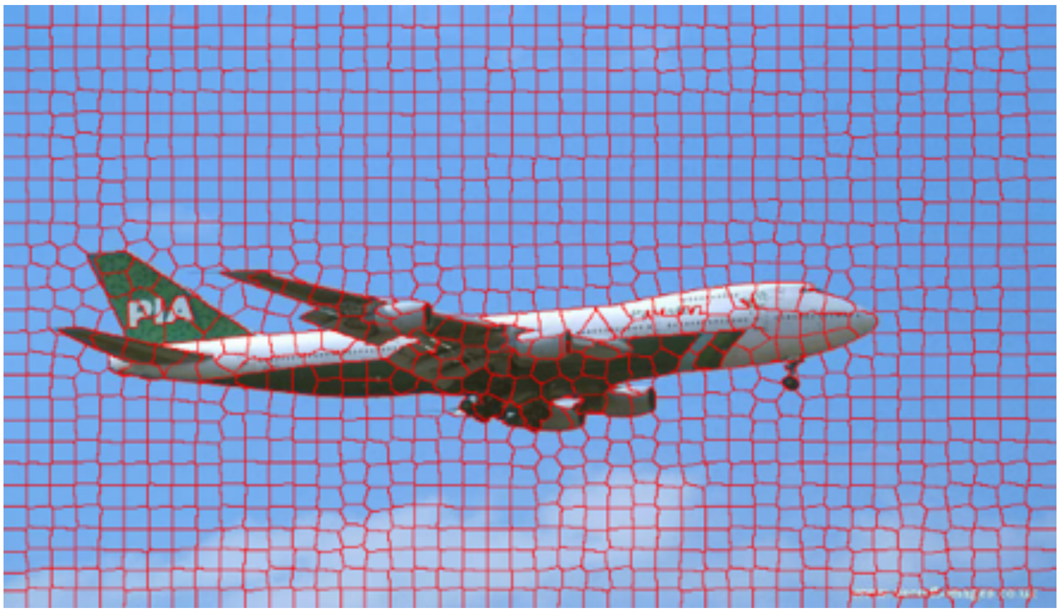


Figure 3.1: An example of superpixel generation of an image using SLIC [1].

see that the superpixels have a high boundary recall and are approximately equal in size.

3.2.2 Gabor features on Superpixels

Gabor filters are typically used for edge detection which is useful for gathering texture information from images [33]. To extract Gabor features, an image is convolved with a 2D Gabor filter.

$$g_{\lambda,\theta,\phi,}(x,y) = \exp \frac{x'^2+\gamma^2 y'^2}{2\sigma^2} \; cos(2\pi \frac{\pi'}{\lambda} + \phi) \tag{3.2}$$

$$x' = xcos\theta + ysin\theta \tag{3.3}$$

$$y' = -xcos\theta + ysin\theta \tag{3.4}$$

γ is the spatial aspect ratio which stays constant. The σ is the standard deviation which controls the size of the Gaussian. λ is the wavelength that

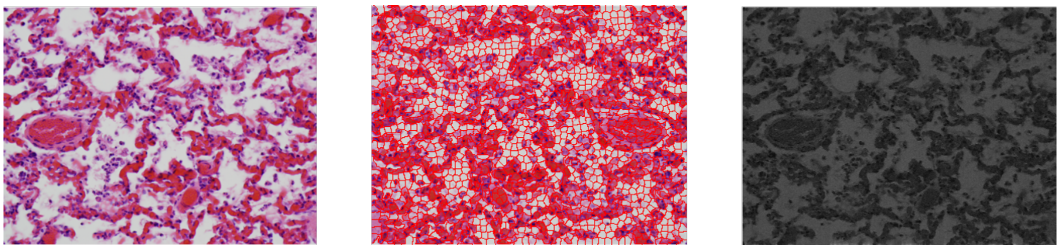


Figure 3.2: The original lung image, the superpixel generation of the original, and the magnitude response of Gabor filter of the original image with θ=0, $\lambda = 4$.

controls the frequency, so $\frac{\sigma}{\lambda}$ is the spatial frequency bandwidth of the filter. θ is the orientation of the Gabor filters.

In our experiments, we change the orientation and wavelength to extract Gabor features from an image, as shown in Figure 3.2. Prior to taking the Gabor features from an image, N number of superpixels are generated for each image, as shown in Figure 3.2. After an image is convolved with a X Gabor filters, we take the mean magnitude from each superpixel to get a matrix size $\mathbb{R}^{X \times N}$. This matrix is used as the feature descriptor for each image.

3.2.3 Histogram of Oriented Gradients on Superpixels

Histogram of Oriented Gradients (HOG) has provided reliable performance for human and object detection [9, 34]. HOG feature descriptors also have an ideal representation for our application because it can be easily represented as vectors of patterns. Therefore, we use the HOG applied on superpixels to attain local feature descriptors.

Classically, HOG computes edge orientations of uniformly spaced cells or "blocks". As in [34], HOG is found by computing the image gradient of blocks of an image, then creating a histogram of gradient magnitudes divided into bins of orientations $0° - 180°$ or $0° - 360°$ for each block.

Rather than using uniform blocks, our method builds HOG descriptors on superpixels of the image. The image gradient of an image is computed as usual but superpixels are generated as explained in Section 3.2.1. A HOG descriptor

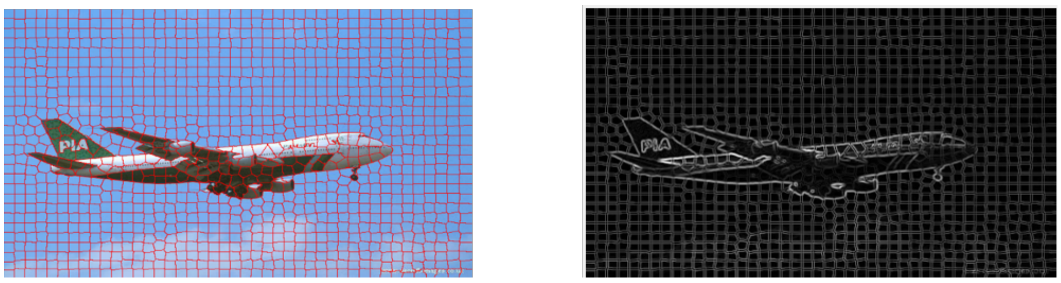


Figure 3.3: The histogram of oriented gradients is created for every superpixel in the image.

is created for every superpixel in the image, as depicted in Figure 3.3. To get more discriminative features, our histograms use 60 bins of orientations $0° - 180°$. The HOG descriptor for a single image would be a matrix of size $\mathbb{R}^{60 \times N}$, where N is the number of superpixels generated in that image.

3.3 Descriptor Dimension Reduction

After extracting features from images, we use dimension reduction techniques that attain more discriminant feature descriptors and allow patterns to be detected more efficiently.

3.3.1 K-means clustering

K-means clustering is a popular unsupervised learning technique for pattern recognition that groups data by iteratively measuring its likelihood to the group [35]. For a set of points, the K-means algorithm needs to initialize the k centers of each cluster, usually by placing them far away from each other. Then, each point in the data is assigned to a center. The k centroids are iteratively set by minimizing the objective function, Equation 3.5, which is the sum of squared errors.

$$J_K = \sum_{i=1}^{K} \sum_{i \in C_k} \|x_i - m_k\|^2 \tag{3.5}$$

where K is the number of cluster centers, C_k are the data points in each cluster, and $m_k = \sum_{i \in C_k} x_i / n_k$, where n_k is the number of points in C_k [35].

The Gabor feature descriptors attained $\mathbb{R}^{XxN}$, as explained in Section 3.2.2, are reduced using k-means clustering. From the training images of the dataset, a dictionary is learned from k-centers which gives a matrix $D \in \mathbb{R}^{XxK}$. A histogram is attained for the training images by computing the Euclidean distance between the dictionary, D, and the feature descriptor. A histogram descriptor is attained for the test images by similarly computing the Euclidean distance between the same dictionary and the feature descriptor of the test images. The histogram descriptor, $\mathbb{R}^{1xK}$, is used as the final descriptor for classification.

3.3.2 K-SVD Dictionary learning

For the HOG descriptors attained, as explained above, we use dictionary learning to compute more discriminative patterns. Dictionary learning seeks to approximate a signal by finding a linear combination of atoms from a dictionary. K-SVD dictionary learning [36] is an algorithm that learns an overcomplete dictionary. Overcompleteness means that the number of basis vectors is greater than the dimension of the input signal. This typically gives a better approximation of the distribution than a complete unique solution or no solution. K-SVD method for dictionary learning has been successfully used for image recognition and image denoising [37–40].

Forming a dictionary for a large database of images can result in a high dimensional dictionary. Thus, dictionary learning via sparse coding seeks to approximate a signal using the fewest number of basis vectors, or atoms, as possible.. In Equation 3.6, an input signal, Y, is approximated by the linear combination DX, where X is sparse if most values are zero, and minimizes the number of non-zero elements.

$$\underset{x}{\arg\min} \|x\|_0 \text{ s.t. } Y = DX \tag{3.6}$$

A matching pursuit algorithm searches for the best sparse approximation of a signal. It is a greedy algorithm which searches for the locally optimal choice at each stage to get the global optimum. The orthogonal matching pursuit (OMP) iteratively finds an atom in the dictionary that best matches the input signal, updates all the coefficients by computing the orthogonal projection of the signal onto the atoms found, then finds the next atom that matches the remaining signal. This continues until the hard threshold condition is reached. This is shown by Equation 3.8 which is K-SVD dictionary learning using OMP.

$$< D,X >= \underset{D,x}{\arg\min} \|Y - DX\|_2^2 \text{ s.t.} \forall i, \lambda \|x\|_0 \leq T \tag{3.7}$$

$Y = [y_1...y_p] \in \mathbb{R}^{NxP}$ is the input signal, $D \in \mathbb{R}^{NxK}$ is the dictionary matrix with K atoms, and $X \in \mathbb{R}^{KxP}$ is the sparse representation. The algorithm finds the representation given the dictionary while updating the dictionary atoms after each iteration [41]. The dictionary update is similar to SVD as it solves the eigen values and updates the K atoms to get a new dictionary.

$$< D,X >= \underset{d,g}{\arg\min} \|E - dg^T\|_F^2 \text{ s.t.} \lambda \|d\|_2 = 1 \tag{3.8}$$

We use the K-SVD algorithm to learn a dictionary for the HoG descriptor found for all the images. Typically, image classification with dictionary learning uses the similarity measure between the sparse codes to classify an image. However, we use the patterns of the dictionary atoms to classify on the automata.

3.4 Encoding Feature Descriptors to Regular Expressions

Once a discriminative feature descriptor is attained for every image, the descriptors are encoded into regular expressions before compiling them into state machines. Since the automata does not take in floating point numbers, the descriptor values are binned to 8-bit characters using this equation. The range of the bins are divided equally and correspond to the minimum and maximum feature descriptor ranges. The regular expression descriptor is derived by

$$R_i = char(y_i(\beta/k)) \tag{3.9}$$

where i is the length of the descriptor, y is the image descriptor, k is the range of descriptor values for that method, and β is the number of characters used.

3.5 Image Retrieval on the AP

Automata are created for each category within a dataset from the regular expression descriptor of the training images, and these are loaded onto the AP board. The regular expression descriptors of the test images are then computed as a pre-processing step, and then sent as an input data stream to be matched in parallel with each candidate descriptor on the AP board. The test image is classified to the category with the most automata matches. Since the AP requires exact STE matching, a threshold can be applied on the regular expression patterns on the automata to allow for more lenient matching. Each STE can allow for up to 8-bit characters. In the AP, an STE with * symbol means that any character will match with the activated STE. Here, we set a threshold by comparing the characters at every index of the regular expression descriptor of the training dataset. If there is greater than a percentage of mismatches at that index then the symbol is hard-coded as * symbol, as shown in Figure 3.4. The threshold gives more flexibility in exact pattern matching without losing significant information. In the figure, each automata represents a pattern of an image in the training dataset that is to be matched by the input data stream.

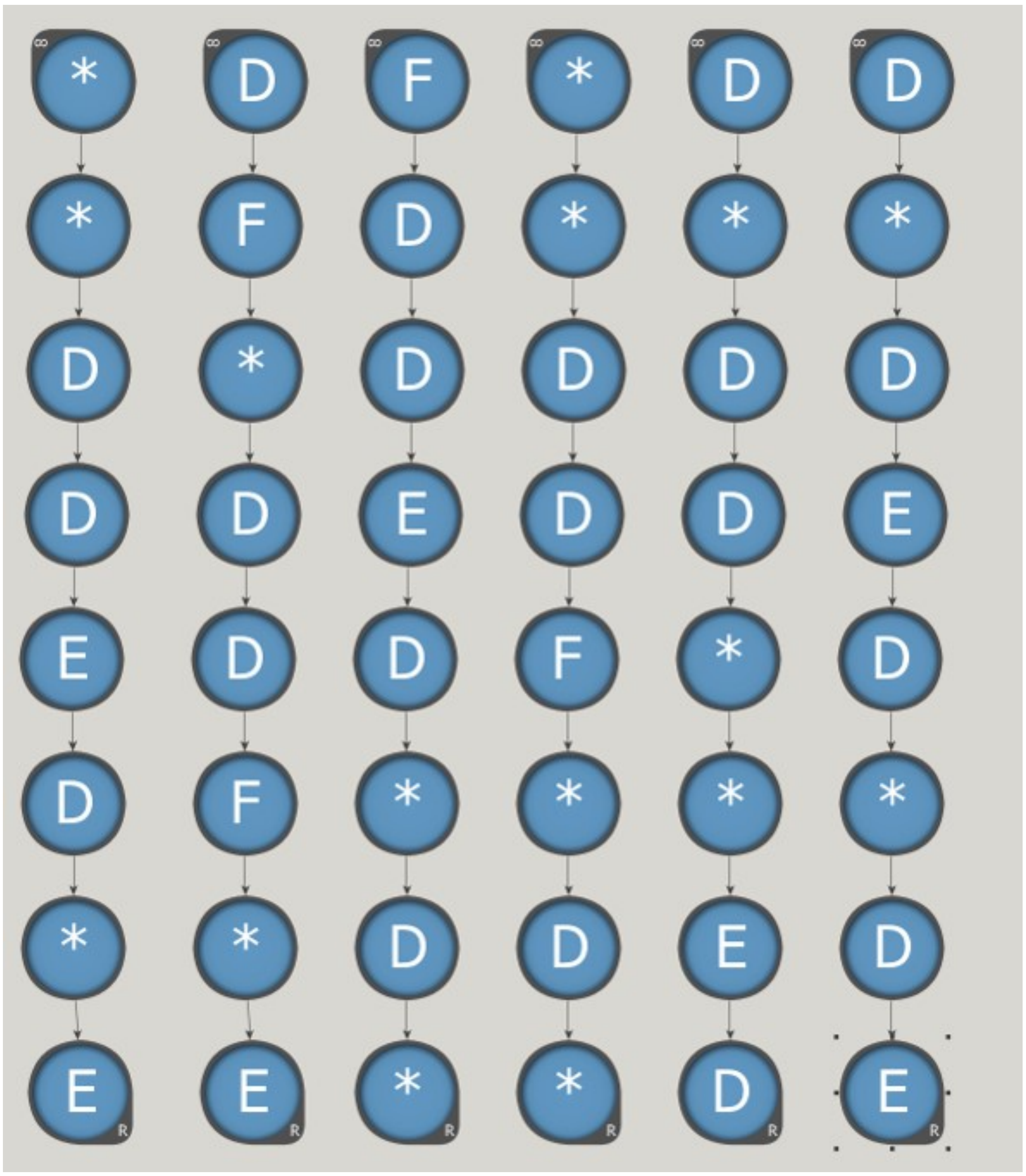


Figure 3.4: An example of automata with regular expression patterns. The [*] symbol matches with any input symbol.

3.6 Experiments

We evaluate our results with two datasets: the ADL dataset [42], and the vehicle dataset using two different feature extraction and representation methods. We compare our results with SHIRC [42].

3.6.1 Evaluation for ADL datasets

The ADL database contains kidney, lung and spleen tissue datasets with healthy and inflamed tissues. Each dataset contained about 330 images. Figure 3.5, 3.6, 3.7 exhibit the major challenge for each dataset is the slight dissimilarity between healthy and inflamed tissues. For each organ, 115 images per class are used for training and 40 images per class are used for testing.

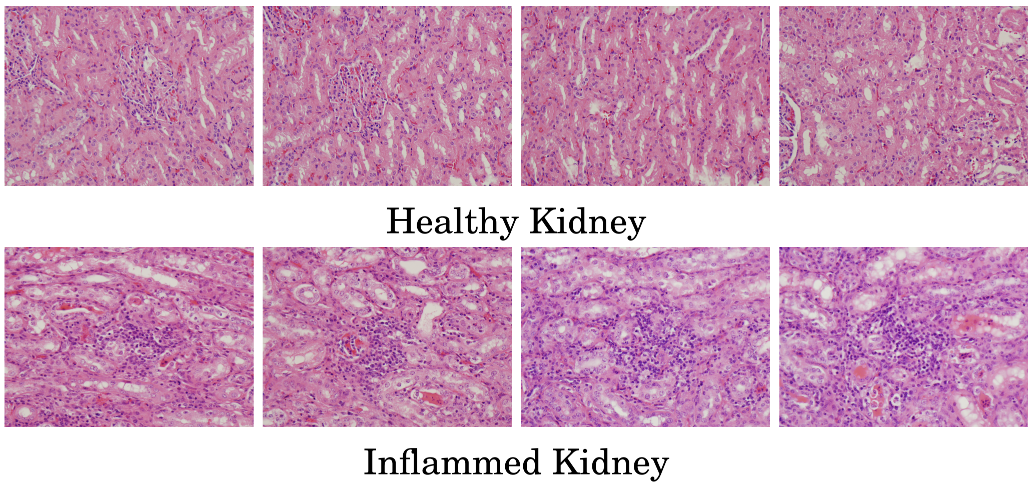


Figure 3.5: Samples of spleen ADL organ tissue dataset.

For this dataset, we use a Gabor bag-of-words method to represent the feature descriptor. 32 Gabor features were extracted from 1910 generated superpixels of every tissue image. A codebook was trained via k-means clustering with k=500 centers to then attain a histogram descriptor for every image, size $\mathbb{R}^{1\times 500}$. The regular expression descriptors of all the training images are concatenated. If there are X images in the training dataset, then there would

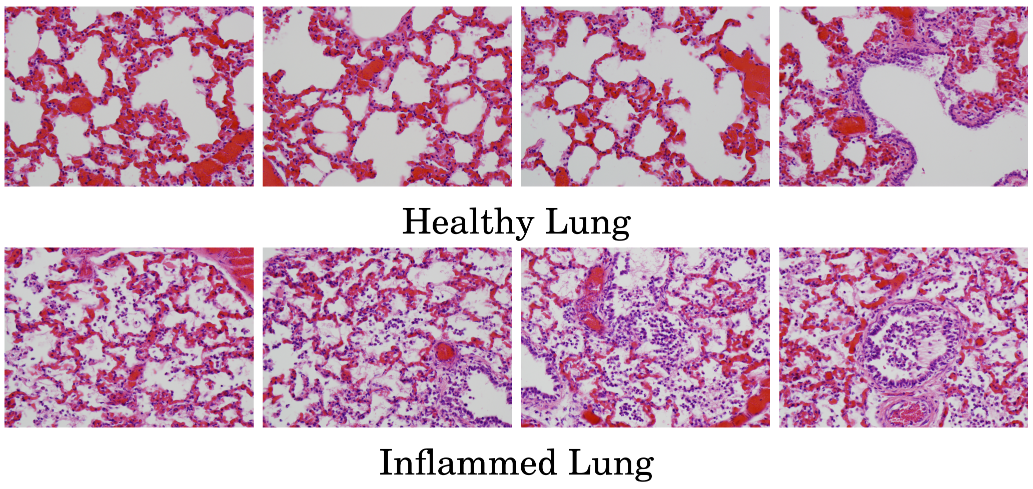


Figure 3.6: Samples of lung ADL organ tissue dataset.

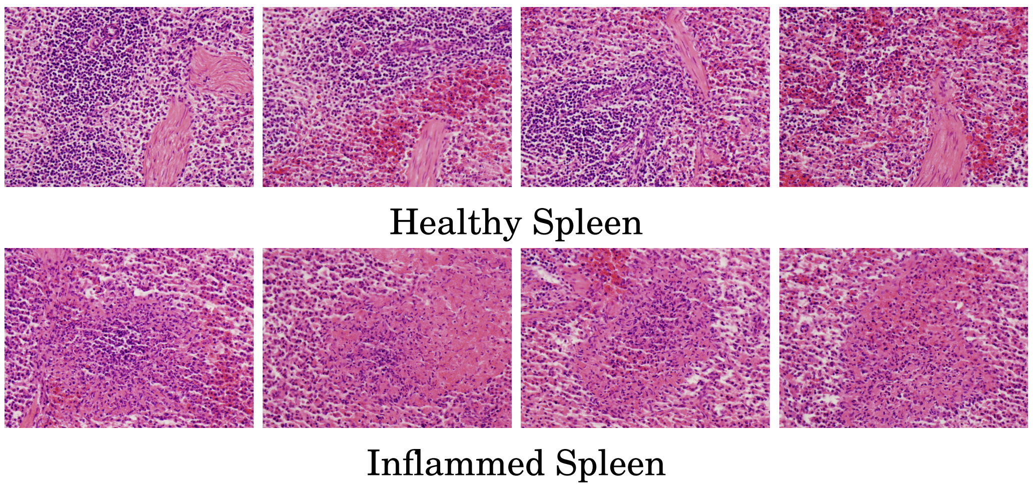


Figure 3.7: Samples of spleen ADL organ tissue dataset.

be X automata of length 500.The descriptor values for the first dataset range from [0,1]. These descriptors have an unknown nonlinear distribution where most of the values lie close to 0 and a few values have large arbitrary values. This is a challenge when matching with the automata because it requires pattern matching, rather than using a similarity measure between the descriptors. To aid this, a threshold is set for the allowable mismatches at each index of the descriptor using 250 randomly chosen training images. Figure 3.8 shows the effect of the threshold for allowable mismatches on classification accuracy with T=10%, 20%, and 30%. For every dataset, a threshold of T=10% reported the best accuracy.

Table 3.1 shows a confusion matrix using our method at a 10% threshold in comparison with the simultaneous sparsity model for histopathological image representation and classification (SHIRC), which is a sparsity model that learns a dictionary for RGB color channels [42]. There are three confusion matrices for each of the organ datasets. The bold numbers are the true positives attained using our retrieval method. The method deployed on the automata framework does not do better in most cases. However, the lower performance was largely due to unclassified images since our method uses exact pattern matching.

Table 3.1: Confusion Matrix for ADL Dataset

		Kidney		Lung		Spleen	
	Class	Healthy	Inflamed	Healthy	Inflamed	Healthy	Inflamed
Gabor w / Superpixels	Healthy	**61.3**	38.7	**63.6**	36.4	**64.6**	35.4
	Inflamed	32.5	**67.5**	47.3	**52.6**	24.4	**75.6**
SHIRC [42]	Healthy	**92.0**	8.0	**91.0**	9.0	**90.8**	9.2
	Inflamed	16.3	**83.7**	28.6	**71.4**	30.6	**69.4**

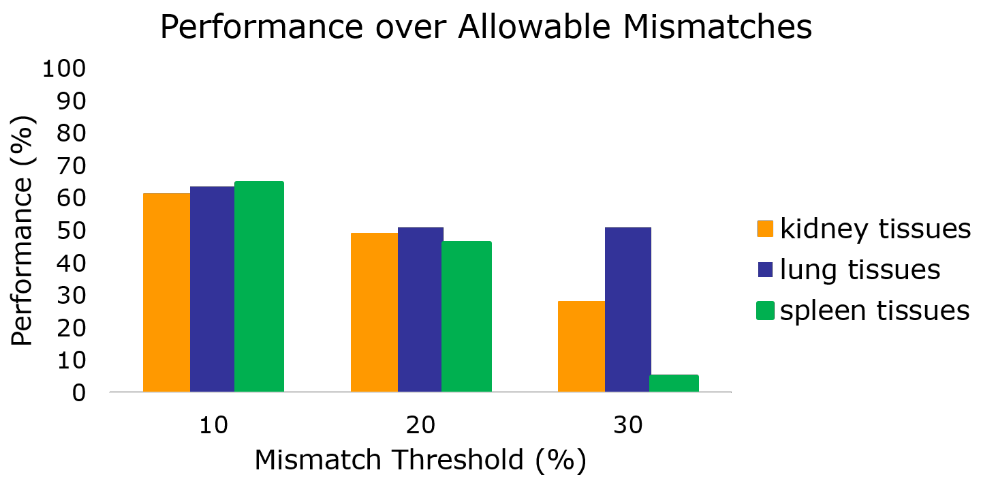


Figure 3.8: Performance for the kidney, spleen, and lung datasets when the threshold for allowable mismatches is set to T= 10, 20 and 30 %.

3.6.2 Evaluation for Vehicle dataset

The vehicle dataset contain four categories-airplane, car, motorbike, and ships with 70, 50, 70, and 36 images, respectively. The images were obtained partly from Google, Caltech-101 dataset [43] and the Inria GRAZ02 dataset [44]. The sample images are shown in Figure 3.9 and depicts a few challenges with this dataset. Within each category, the objects are variable in rotation, color, size and type. The car images have varying frequency of objects in a single image. Some images also have low contrast between foreground and background which makes it difficult to extract objects features. Superpixels were generated on every image to mitigate this challenge by grouping the image into segments based on spatial and color information. For each image, a dictionary was learned from the HOG descriptors using the minimization in Section 3.3.2.

The HOG descriptor represents the input signal, $Y \in \mathbb{R}^{60\times1200}$, which is represented as linear combination of dictionary, $D \in \mathbb{R}^{60\times500}$, while minimizing the reconstruction error. X are the sparse codes for the signal Y, and e is the sparsity constraint [40]. The values of the atoms range from [-1,1] and are encoded into regular expression patterns without a threshold. Each dictionary atom in the training dataset are represented as automata on the AP. The dictionary atoms of the test images are sent through the input data stream to be matched. Thus the learned dictionary atoms are used as image features for this dataset. A test image is classified using an automata framework as explained in Section 3.5.

The retrieval accuracy results using an automata framework for both datasets are given in Table 3.2 and are compared with state-of-the-art methods implemented on similar datasets. Image retrieval on an automata framework performed significantly better when using dictionary atoms as descriptors than when using Gabor bag of words descriptors. Retrieval using regular expressions still did slightly worse than the SLIDE method for the vehicle dataset.

Figure 3.9: Vehicle dataset with four categories: (a) airplane (b) motorbike (c) car and (d) ship.

Table 3.2: Overall retrieval performance

Method	Acc. (%)	Runtime (s)
Gabor w/ Superpixels on ADL dataset using AP	63.2	**2.967e-5**
SHIRC [42] on ADL dataset	80.5	.55
HOG w/ Superpixels on Vehicle dataset using AP	79.5	**.06**

3.6.3 Run-time Comparison

The computational cost of image classification on an automata framework was computed using a run-time estimation as explained by the authors in [21]. The AP processes a new, 8-bit input symbol every clock cycle and is stalled by 40 nanoseconds in an output buffer every time it reports. The run-time estimation for the AP is calculated to take $[(16+40p+l)]*7.5$ nanoseconds to run, where 16 is the inital setup latency, p is the number of output vectors per cycle, l is the number of STEs reported in that cycle, and 7.5 ns is the clock cycle. The computational cost for both datasets deployed on the AP are compared to the runtime of comparison methods, as shown in Table 3.2. The comparison only reports the time it takes to classify the descriptors. The SHIRC method takes .55s to classify one image running in Matlab on a 64-bit Windows 7 system equipped with Intel Core $i7-2600$ 3.4-GHz processor and 8 GB RAM [42]. We compute the classification run-time in each method for one image and see a significant speedup.

3.7 Discussion

There were several notable discoveries and challenges in our work that can be improved on in future work. Overall, our experiments show that multiple feature extraction methods can be encoded into regular expressions and used to implement image retrieval on an automata framework. However, because the AP methods do not yet achieve state-of-the-art accuracy, our results suggest a trade-off between speed and accuracy. The reduced accuracy may be acceptable in applications requiring significant speedup. However, this work is just the first step in exploring potential feature descriptors and mappings for automata processing. While the accuracy of image classification with regular expressions has not reached the current state-of-the-art accuracy, there is clear room for future work to achieve improvement.

Feature Extraction. The second experiment shows that more discriminative descriptors are simpler to implement on exact matching automata and attain better retrieval accuracy. Future work includes combining feature extraction methods and reducing to a discriminative representation.

Mapping distributions. As in the first experiment, the feature descriptor may not have a known distribution which is a challenge when encoding the descriptor to regular expressions. The experiment demonstrated manipulation of automata STE symbols to aid this challenge. There are other potential methods to encode these descriptors such that information is not lost and thus improve performance. Non-linear feature descriptors may be more efficiently encoded to regular expressions by using kernel mapping methods.

Image Matching on the AP. We showed a method of matching on an automata framework where images are classified based on how many automata they match. The automata structure limits our image classsification methodology. For example, classification using dictionary learning typically measures the similarity between sparse codes. However, our method of classification on the automata uses the dictionary atoms to represent an image in order to take advantage of automata pattern matching. Further manipulation of AP elements and re-configuring automata structures that do not require exact descriptor matching can be applied. This could improve accuracy and reduce run-time of image retrieval applications [21].

CHAPTER 4

IMPROVING THE AUTOMATA CLASSIFIER

In the previous chapter, we found that some feature descriptors have patterns that are non-linear or have an unknown distribution. This can be a problem when deciding how to encode an image descriptor into regular expression patterns. Further, improving the image classification method on the automata could increase classification accuracy.

4.1 Image representation

In this section, two feature representations are explored on multiple datasets. For both, Gabor features are extracted from the images to attain texture information.

4.1.1 Bag of Words

The bag-of-words model is widely used for feature extraction for image retrieval applications including scene detection, object recognition and content-based classification [45–47]. The bag of features method is a locally orderless representation that gathers the frequencies of visual words, or codebook [48]. In our case, Gabor features are extracted from training images to cluster and generate into a frequency of codes that make up the codebook. Each feature of the test image is assigned to the nearest code in the codebook to attain a histogram. Computing the nearest code can be done by a nearest neighbor calculation as explained in 3. The representation is refered to as a "bag" because the local features are orderless and do not keep spatial information.

4.1.2 Spatial Pyramid

Spatial pyramid is a feature encoding technique that exploits the spatial relationship between local features to compute an image representation. It attains a descriptor by getting the bag of features at different pyramid levels. At each pyramid level, an image is divided into an increasing number of blocks. The bag of features are again represented as histograms and are concatenated into a vector [14].

4.2 Cumulative Distribution Transform

Classification on the automata relies heavily on distinguishing patterns, which means the feature descriptors between classes should be unique and discriminative. However, separating and mapping descriptors with non-linear distributions can be difficult to encode and classify directly on the AP. The automata architecture is inherently a processor of 1-D strings that can be manipulated and analyzed via regular expressions. Hence, we need a multi-dimensional to 1-D transformation from which we can generate regular expressions. Transforming our feature descriptors with the cumulative distribution transformation can accomplish the M-D to 1-D mapping and ensure linear separability between our classes even if the class distributions are non-linear.

Further, we want our automata classifier to be able to classify similar images even with variations within the image such as rotation, translation and scaling. The cumulative distribution transform achieves this invariance by taking in account the location of signal intensities with respect to a chosen reference [18, 19].

The cumulative distribution transform (CDT) requires two probability density functions, one as the input pdf to be transformed and projected with respect to the second pdf. The pdf for each image, represented as I_1, is transformed in reference to a chosen template pdf, I_0, which, for ease of implementation, should be uniform density. X and Y are connected sets in $\mathbb{R}$. Then the cumulative distribution function of I_1 is mapped into $\hat{I}_1$, $X \to Y$ by a set of unique continuous functions, f_1 [18, 19], given by the following equation.

$$\hat{I}_1 = (f_1(x) - x)\sqrt{I_0(x)}, x \in X \tag{4.1}$$

Where $f_1(x)$ must satisfy the following equation.

$$\int_{inf(X)}^{x} I_0(\tau)d\tau = \int_{inf(Y)}^{f_1(x)} I_1(\tau)d\tau \tag{4.2}$$

For continuous functions of I_0 and I_1, f_1 is also continuous. If f_1 is also differentiable then the inverse CDT can be found.

We use the cumulative distribution function [18, 19] to make our feature descriptors more linearly separable. Once feature descriptors are extracted from every image, they are represented as normalized histograms which represent I_1 and can be transformed using the CDT. The transformed histograms, $\hat{I}_1$, are quantized and used as feature descriptors for classification.

4.2.1 Multinary representation of Images

After the transformation, explained in Section 4.2, is performed on the feature descriptors of all images in the dataset, we encode them into string descriptors. The STEs are designed to hold up to 8-bit set of symbols or an arbitrary set of character classes–single character, regular expression characters, or numerical characters. The values within feature descriptors vary depending on the method of extracting data from your image. Mapping these values into string descriptors can become a problem when the descriptors are non-linearly separable. Encoding data into string representation without losing information can be difficult. Thus, when the descriptors are mapped to strings, we also want to maintain the pattern information that distinguishes one class from another. As explained in the previous section, the CDT eased this problem by linearly separating the image descriptors.

Each value of the image descriptor is encoded to characters using the following equations.

$$y_i = l + \frac{\phi}{\beta} x \tag{4.3}$$

where y is the value of the image descriptor, i is length of each vector in the image descriptor, l is the lower bound of the global image descriptor and β is the maximum bits the user chooses to encode to. x corresponds to the distance of the character value from the lower bound on the string space. The character values are defined as follows:

$$R_i = L + x \tag{4.4}$$

where R_i is the character value and L is the lower bound of the character space. Finally, we obtain the encoding equation by using the following:

$$R_i = L + (y_i - l)\frac{\beta}{\phi} \tag{4.5}$$

Eq. 4.5 encodes every value of the feature descriptor to a set of character symbols, which is used for classification on the automata.

4.2.2 Automata Classifier Design

Typically, images are classified by computing the similarity or dissimilarity of an image to a class. In our method of classification, we employ pattern matching on the automata using the string descriptors. Once all images have been encoded into strings, the dataset is split into testing and training images. The training images are used to represent the patterns on the automata.

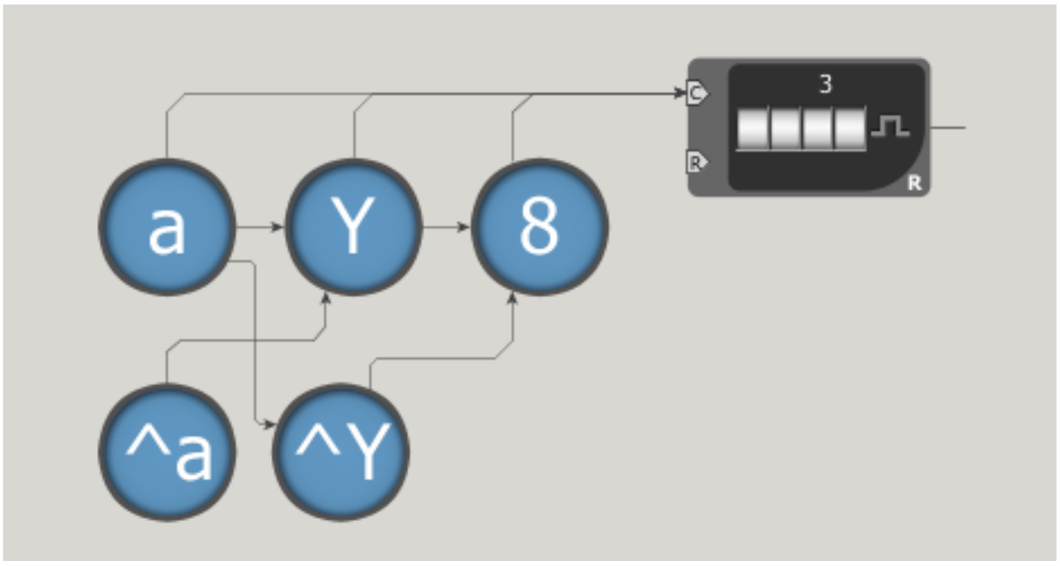


Figure 4.1: Example of the automata classifier design with mismatches and counter.

Every value, i, from each training image's string descriptor represent an STE creating a pattern for each class. The string descriptor of every image in the testing set is loaded into the input data stream to be matched on the automata. Essentially, every test image is matched against every training image from all classes.

A test image is classified in the class for which the most matches occur. This classifier on the AP would have a state machine of length one STE connected to a counter element. Fig. 4.1 depicts a STE of one image in a class. There is one automaton for every training image for all classes in the dataset. The length of the automaton corresponds to the length of the image descriptor. This design allows for mismatches. A counter is connected to each STE and increments by one every time an STE is activated. The string descriptor of the test image is matched with the automata and is classified to the class with most STE matches.

4.3 Experimental results and Analysis

We evaluate our retrieval results for a two-class and multi-class problem on a colorectal cancer tissue dataset [2]. Our experiments simulate the automata design mentioned in Section 4.2.2.

4.3.1 Colorectal Cancer Dataset

Both experiments used histological images of human colorectal cancer [2] which includes seven different types if tissues, shown in Figure 4.4. Up until Kather's *et al.* work [2], in terms of colorectal histological images, there have only been published studies on the classification of two types of tissues- tumor and stroma.

In histopathological imaging, tumor tissues are recognized as having abnormal growth of tissues from cells not dying when they should or divide excessively. The stroma tissue is a vital supportive tissue of the organ [49**?**]. The stroma monitors and regulates through cellular pathways growth in the bottom layer of the skin. Since any misregulation of growth factors in the stromal tissue layer can lead to cancer, it would be useful to identify the mass of simple stroma and complex stroma(containing single tumor and/or immune cells) tissues, not solely the binary case. In these terms, it would also be beneficial to identify the other layers of the skin cross section, immune, mucosal gland, adipose (fat layer) and debris (dead skin), since the stroma also regulates skin growth and death. The morphological and density changes in each tissue may be beneficial in diagnosis of colorectal cancer.

Figure 4.4 shows the challenges there may be in differentiating between the classes considering some types of tissues contain cells from other images. The complex stroma is stroma tissue that contains single tumor cells or single immune cells. The immune cells are made up of immune cell conglomerate and sub-mucosal lymphoid follicles. The debris image includes necrosis, hemorrhage and mucus. The overlap in some of the classes may create difficulty in separating features.

4.3.2 Feature Extraction and Representation

Gabor features: In our experiments we employ local Gabor features [50, 51] to represent the superpixels. In the literature, it has been shown that Gabor filters can approximate the characteristics of certain cells in the mammalian visual cortex and can be exploited in getting the texture information of an image. 2D Gabor filters are obtained by combining Gaussian kernel with sinusoidal functions of different frequency and orientation. The Gabor filters are regulated by the standard deviation of the Gaussian filters and the orientation of the sinusoidal functions. An image is convolved with different combinations of the standard deviation and the frequency to get the Gabor filter response. The mean response of each filter bank can be used as a global texture feature of the image. For a local region, we first convolve the image with 32 Gabor kernels, for 8 orientations and 4 different Gaussian scales. We then compute the mean response within a superpixel for each of the 32 filters and thus obtain a 32 dimensional Gabor feature for each region. To account for the variation in intensity, we also include local color features in our descriptor. For each super-pixel, we compute the mean color of the region in CIE L^*a^*b color space along each channel. We concatenate the color and Gabor feature of a region to get the final local descriptor.

We compare our results with SLIDE [52] and spatial pyramid matching (SPM) [14]. SLIDE [52] is a saliency guided dictionary learning method which uses compressibility of sparse codes to design a similarity measure between images.

Spatial pyramid matching (SPM) is described in [14]. SPM combines local image features while retaining the spatial correspondence. In this method an image is partitioned into subregions, and for each subregion a histogram of local features is computed. Finally the histograms from each subregion are concatenated to obtaining the final feature representation. For our experiments we use the same local image features as described above and two pyramid levels. We compare the spatial pyramid histograms using KL divergence, such that if p and q are the spatial pyramid histograms of two images, then the

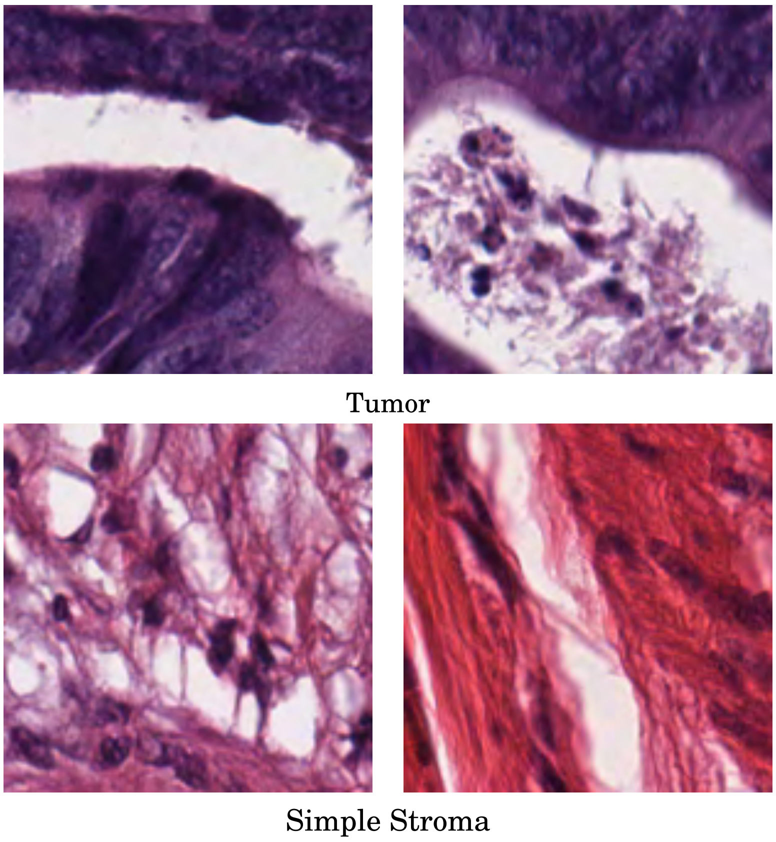


Figure 4.2: Two-class dataset: tumor versus stroma tissue. Shows sample images from the two classes

symmetric KL divergence is given as:

$$KL(\mathbf{p}||\mathbf{q}) = \sum_i \mathbf{p}(i)\log\frac{\mathbf{p}(i)}{\mathbf{q}(i)} + \sum_i \mathbf{q}(i)\log\frac{\mathbf{q}(i)}{\mathbf{p}(i)} \tag{4.6}$$

where p is the histogram of a training image and q is the histogram descriptor of a query image. The query image is classified to the class with the lowest KL divergence value. This classification method was used to compare against our methods on an automata framework.

4.3.3 Evaluation for two-class classification

The first experiment for image retrieval using the cumulative distribution transform and an automata classifier is tested on a two-class classification of

No. of Gabor fea.		Tumor	Stroma	Tumor	Stroma	Tumor	Stroma	Tumor	Stroma
16	Tumor	**81.5**	18.5	**83.4**	16.6	**82.7**	17.3	**87.3**	12.7
	Stroma	12.1	**87.9**	9.6	**90.4**	10.2	**89.8**	24.2	**75.8**
24	Tumor	**79.6**	20.4	**78.3**	21.7	**75.8**	24.2	**68.2**	31.8
	Stroma	26.1	**73.9**	17.8	**82.2**	17.8	**82.2**	26.8	**73.2**
32	Tumor	**80.3**	19.7	**77.1**	22.9	**70.7**	29.3	**76.4**	23.6
	Stroma	15.3	**84.7**	16.6	**83.4**	22.9	**77.1**	26.1	**73.9**
40	Tumor	**80.0**	20.0	**71.3**	28.7	**76.4**	23.6	**77.7**	22.3
	Stroma	14	**86.0**	18.5	**81.5**	24.2	**75.8**	24.8	**75.2**
		50		100		200		500	
		No. of Clusters in BoW							

Table 4.1: Confusion Matrix for Two-Class CRC Dataset. Confusion matrix for the two class CRC dataset with varying parameters. We use 16, 24, 32, and 40 Gabor features and 50, 100, 200, and 500 BoW clusters for each. The bold values are the true positives.

malignant and benign cells from a colorectal cancer colon tissue dataset [2] as shown in Fig. 4.2. We see that the tumor and stroma tissue images can be particularly difficult to distinguish because they are similar in texture and intensity.

The first dataset is split into 468 training images and 157 testing images per class. Prior to classification, Gabor features were extracted from all of the images using 16 features (two wavelengths and eight orientations). Two feature representation methods are used to test the transformation and classification. We used the bag-of-words model to represent the Gabor features as histogram descriptors. We transform the normalized descriptors with the CDT used in [18, 19] for every image with respect to a template signal. We use a template $I_0 = 1$, a uniform probability density with zero mean and unit variance, for all of our experiments. The descriptor length for the images using BoW representation is $\mathbb{R}^{1x100}$ and using spatial pyramid is $\mathbb{R}^{1x500}$. The size of state machine for every class is dependent on this vector and the number of training images. Since there are two classes in this dataset, there are 936 automata utilized for this implementation.

As stated in the previous chapter, feature extraction has significant impact on classification accuracy, particularly using pattern matching on an automata framework. Therefore, it is advantageous to analyze the parameters of our method, which may also effect the run-time evaluation and classifier design implementation. We use 16, 24, 32, and 40 Gabor features (by changing the wavelength) and use 50, 100, 200, and 500 clusters in the bag-of-words model.

Table 4.1 shows the confusion matrix with the varied parameters. For nearly all the experiments, using 50 clusters in the Bag-of-Words model result with better accuracy. The accuracy slightly decreased as we increased the number of clusters, as more bags will lead to sparsity and less variation in features within the histograms. We further test the sensitivity of Gabor features is keeping a constant number of clusters ($k = 50$) for the Bag of Words model. Figure 4.3 shows that accuracy slightly decreases as the number of Gabor features increase with the maximum accuracy using 16 Gabor features (tumor: 81.5%, stroma: 87.9%).

Additionally, the stability of this automata classifier is tested by varying the number of training images. Table 4.2 shows that our classifier is stable in terms of changing the number of training images.

Table 4.2: Stability test

Ratio of Training Images	Acc. (%)
3:1	82.5
4:1	82.4
5:1	81.3
9:1	81.7

4.3.4 Evaluation for the multi-class classification

The second experiment for image retrieval using cumulative distribution transform and an automata classifier was tested on colon cancer tissue dataset [2]. The dataset contains seven classes of colon tissues, as shown in Fig. 4.4. Each class contains 625 images, which was split into 468 images for

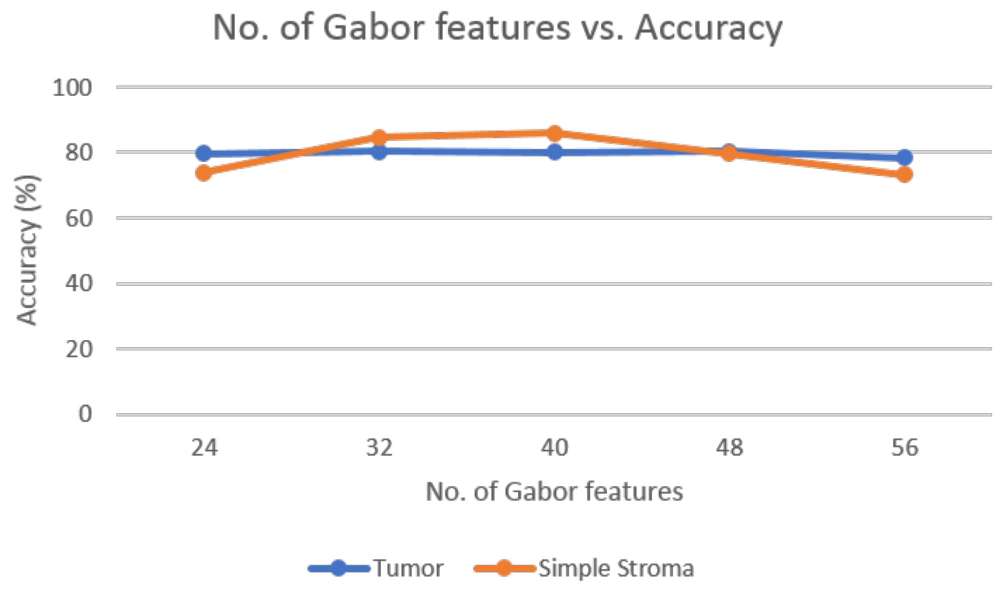
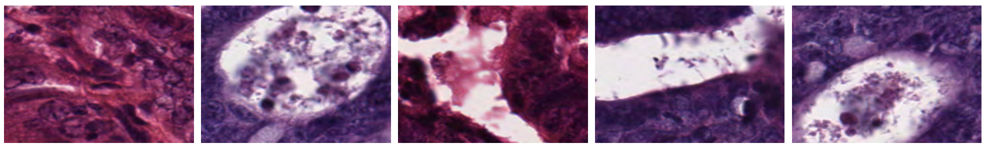


Figure 4.3: Accuracy with varying number of Gabor features.

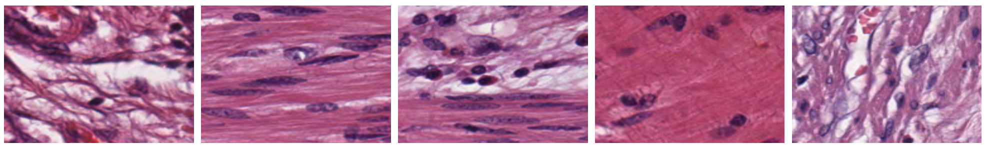
training and 157 images for testing. From Fig. 4.4, we see that the multi-class classification problem can also be quite difficult since these classes have a combination of intensity and texture similarities.

The feature descriptors are attained using the same methods as the two-class problem with 16 Gabor features and 50 clusters in the BoW model. Since there are seven classes in this dataset there are 3276 automata represented on the AP, which is one-fifth of the STEs available on the current hardware.

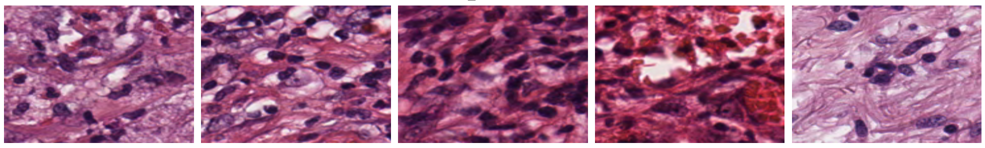
The retrieval results for this dataset are compared to those of in [2], which is the first published result on a multi-class texture separation, shown in Table 4.3. From the overall accuracy, we can see that this multi-class classification is particularly difficult to classify. It is important to note that the classes with similar features and overlapping characteristics perform worse for all classifiers. As mentioned in Section 4.3.1, the makeup of the tumor, simple stroma and complex stroma tissues and debris and mucosa tissues contain corresponding cells which make differentiating their features and classification more difficult. Our retrieval method using the AP classifier surpasses standard



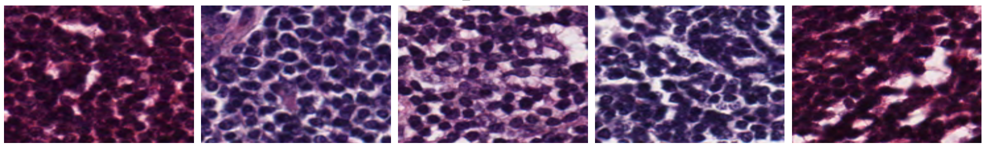
(a) Tumor



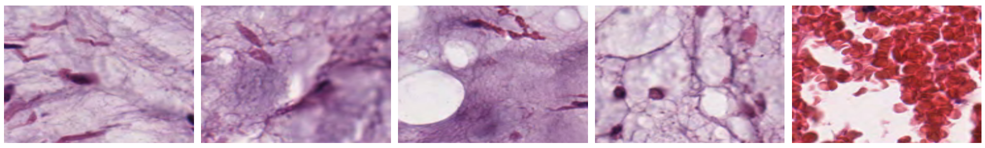
(b) Simple Stroma



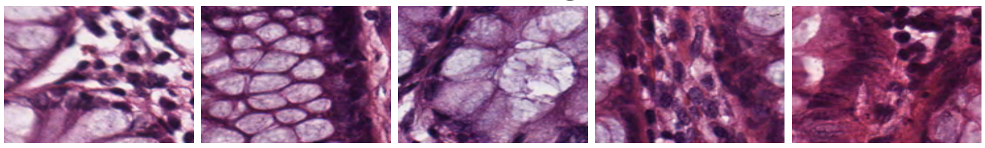
(c) Complex Stroma



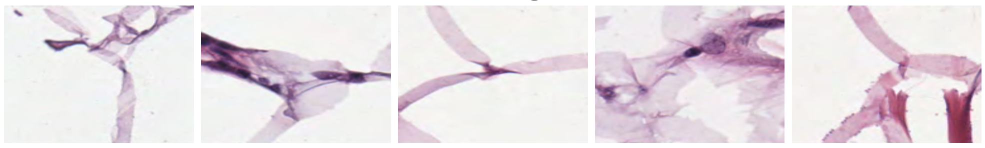
(d) Immune cells



(e) Debris (including mucus)



(f) Mucosal glands



(g) Adipose tissue

Figure 4.4: Seven classes of colon tissue dataset [2]. (a) tumor epithelium. (b) simple stroma (c) complex stroma (stroma that contains single tumor cells and/or single immune cells). (d) immune cell. (e) debris and mucus. (f) mucosal glands. (g) adipose tissue.

Table 4.3: Overall retrieval performance for the multi-class dataset

Method	Accuracy (%)						
BoW w / CDT using AP	**58.6**	8.92	**58.6**	20.4	38.9	**45.9**	**100**
Sppyr w / CDT using AP	37.6	38.2	23.6	48.4	32.4	24.9	91.7
SLIDE [52]	38.9	31.8	2.5	37.6	20.4	6.4	60.1
SPM [14]	42.6	**58.6**	35.6	**68.7**	**42.6**	40.7	**100**
Class	Tumor	Stroma	Complex	Lympho	Debris	Mucosa	Adipose

similarity measures for some classes. The classification results are comparable to SPM, as they exceed results for two classes. As explained in [18, 19], the cumulative distribution function can create more separable data depending on the given input signal. Determining a feature representation method that is suitable for a given image dataset may improve the accuracy.

4.3.5 Classification Run-time Computation

The computational cost of classification on the automata is computed using the run-time estimation that is explained in [21]. The AP processes a new 8-bit input symbol every clock cycle and is by 40 nanoseconds on every report. The initial setup latency is 16 nanoseconds, p is the number of output vectors per cycle, l is the number of STEs that is reported in that cycle, and 7.5 ns is the clock cycle. The run-time estimation for the AP is calculated from

$$[(16+40p+l)]\times 7.5ns \tag{4.7}$$

The computational cost for the multi-class problem is .024 s for BoW representation and .124 s for the spatial pyramid representation. The computational cost for the two-class problem is .0073 s for BoW representation and .0366 s for the spatial pyramid representation. The classification for the multi-class dataset using SPM and KL-divergence took 11.09 s to classify each image. The multi-class classification using BoW method on the automata found a 462 × speedup, while the classification using the spatial pyramid method on the automata found a 303 × speedup. The classification run-time is limited by the input stream which, in this application, corresponds to the number of test images and descriptor length.

CHAPTER 5

Conclusion and Future Work

The main goal of the project was to apply image processing on the Automata Processor and determine if image classification performance found improvements compared to state-of-the-art methods. The Automata Processor provides significant acceleration for pattern matching on non-finite automata applications. Therefore, we are implementing a multi-dimensional classification problem onto a string-matching processor.

In this thesis, we proposed a process for encoding an image to regular expressions for implementation of image retrieval on an automata framework. The process requires acquiring discriminate feature descriptors and encoding them into regular expressions. Experiments showed that our method successfully classified images on our automata design with some accuracies comparable to state-of-the-art methods.

However, classification on the automata needed further discrimination of features. To increase accuracy, feature descriptors were linearly separated by applying the cumulative distribution transformation. Neither classifier designs on an automata framework surpassed state-of-the-art methods for image classification. However, to increase classification accuracy we can improve the linear separability of the data in the CDT space. This can be done in a few ways including choosing a reference signal that is more similar to the data, using larger datasets, and using a representation method with more order.

While the accuracy for image classification deployed on the AP has not reached that of state-of-the-art methods, we find a large run-time speed-up, up to three-fold. This may motivate further work to identify feature extraction and encoding methods that provide more separable feature descriptors to increase classification accuracy. Since different feature extraction methods measure different aspects of image information, merging different methods have been proven to improve accuracy [2]. Additionally, manipulations of AP elements and structure can be added to the automata image classifier design to improve performance. This work could also be extended to multimodal classification and online learning.

5.1 Concluding Analysis of Image Classification on the AP

The scope of this thesis was to determine the capability and practicality of implementing image processing on the AP. The current experimental accuracy of image classification on an automata framework is comparable to state-of-the-art methods with room for improvement. The next evident step for improving feature extraction would be to implement deep learning for training.

Deep learning architectures have has consistently seen high accuracy in image retrieval applications. These networks have achieved high classification accuracy due to the hidden multi-layers. Current state-of-the-art CNN architectures have roughly up to 152 layers before diminishing performance [53–56]. However, having a large number of training images is a sufficient condition in order to prevent overfitting, which is alleviated by data augmentation, such as cropping, rotating, flipping, and rescaling the training images.

In general, CNN frameworks find improved accuracy with the larger amount of training data. Furthermore, the deep convolutional layers (for training and testing) and backpropagation steps to learn the features have a substantial computation complexity, thus requiring high performance hardware. Recent CNN frameworks run on multiple GPUs or a powerful GPU which can take over one week to train [57].

From this analysis, we should consider situations in which image classification on the AP be advantageous over state-of-the-art processors, such as GPUs. The current AP hardware is not capable of directly performing convolution, which is one of the advantages of GPUs over other processors. It would be beneficial to further explore applications in which the AP would be advantageous over such processors. In real world applications, it is difficult to perform deep learning on smart phones or other devices with low computational power. Estava *et al.* [58], proposed a skin cancer detection smartphone application using deep neural networks. The classifier is pretrained with labels provided by medical practitioners and images taken through the smartphone are sent to a cloud-based data storage. From this perspective, the AP could be advantageous in image applications where a cloud based data storage is unnecessary or cannot be used, such as cases where data security is a concern.

Another stance to explore in which the AP may be advantageous are cases where there is a limited amount of image data. For example, in histopathological datasets, there is not typically benchmark datasets because phenotypical cancer detection varies (between cancers, people and the images taken). Thus, training a deep learning framework would be done on different datasets and would result in high computation complexity. Furthermore, there may be an insufficient amount of data in histopathological images to achieve high accuracy with a deep learning method. It would be valuable to investigate the effect of varying number of training data on classification accuracy with the AP and deep learning frameworks.

CHAPTER 6

APPENDIX A

6.1 Preliminary Experiments

A few preliminary experiments were done to apply image classification on an automata framework. These experiments provided an understanding of how to design our classifier while showing the limitations and complications that come from encoding regular expression patterns and classifying on the automata.

6.1.1 Finding patterns for different objects in an Image

The first experiment done was to understand how to encode an image or objects within an into regular expressions. The first experiment was to prove that individual objects within an image can have its own unique pattern. We used simple images that had one object with a clear foreground and background then took the object boundary, as shown in Figure 6.1. One way to encode these images into strings is to extract feature descriptor value. The contour of each of these objects were parametrized with the distance values from the

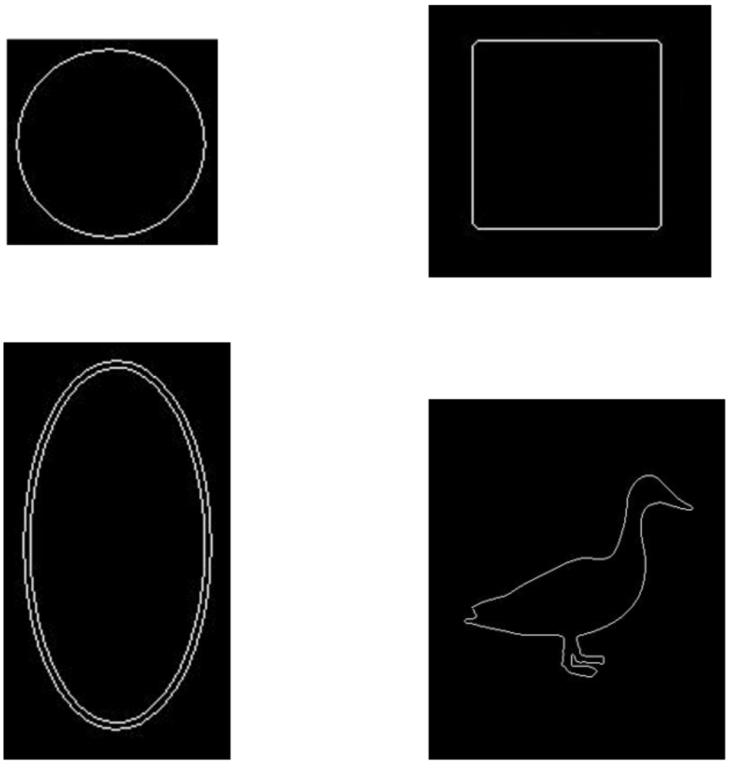
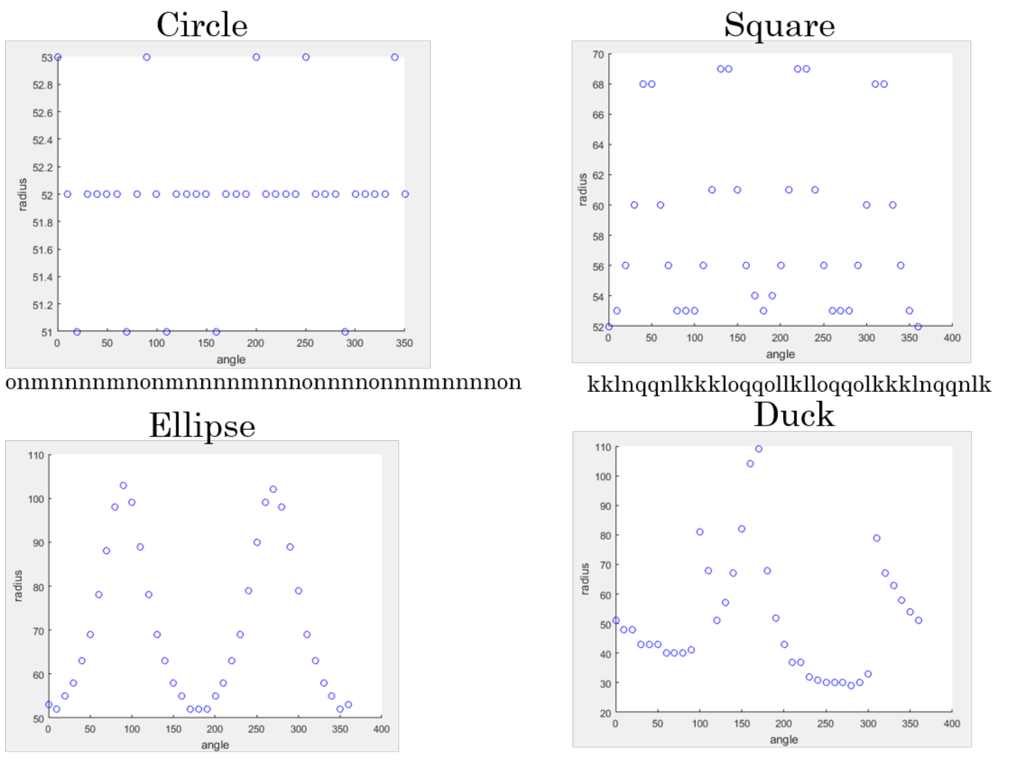


Figure 6.1: Contour of four images.

center point starting at angle $0\,\text{deg} - 360\,\text{deg}$, as shown in Figure **??**. These values were encoded into 26 lower case letters of the alphabet using "n" as the midpoint, as seen on below each graph of the corresponding object. We can see that each object contour has a very unique and distinguishable pattern. However, there are some limitations and design questions that we observed.

The string pattern of the circle in Fig. **??** depicts quantization errors when encoding from values to strings. The quantization errors can be accounted for using regular expression depiction. For the circle, "[m-o]37" finds a pattern of either "m", "n", or "o" repeated 37 times. The circle and ellipse also have repeating patterns but more complex shapes, such as the duck, evidently will not have such patterns. This feature extraction and encoding method could



onmnnnnnmnonmnnnnnmnnnonnnnonnnmnnnnon

kklnqqnlkkkloqqollklloqqolkkklnqqnlk

hhijlnquxzxuqnljihhhijlnruxyxurnljih

mmmkkkkkkvrmprvrnkjjihhhhhhiurqpo

Figure 6.2: The graphs show parametrization of the contour for each object. Below the graph shows the string pattern encoded from these values. The bolded strings are regular expression representations.

account for rotation invariance by repeating the entire string twice. However, this method will not account for uniform or non-uniform scaling.

This preliminary experiment depicted the complications faced when extracting features and encoding them into regular expressions. While regular expressions could account for some quantization errors, a better method of encoding values could provide a more robust descriptor. Further, we want to explore the best fitting length of a regular expression descriptor to attain a descriptor that would distinguish between more objects or images. Finally, as the dataset become more detailed with more objects, color and background variations a more relevant and robust feature detection method could be used.

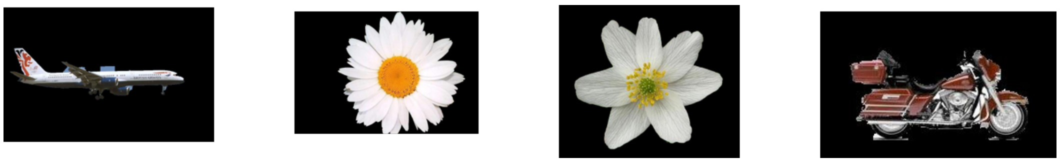


Figure 6.3: Salient objects of the four classes: airplane, daisy, windflower and motorbike.

6.1.2 Exploring more methods for object classification on the AP

In a following experiment we used multiple feature extraction methods to test object classification on an automata framework.

In one experiments we used a dataset that contained four classes of objects-airplane, daisy, windflower, and motorcycle. Each class had 100-200 images. As a preprocessing step, superpixels were computed using SLIC [31] in order to attain salient images, as shown in Figure 6.3. HOG descriptors were taken for each 1200 superpixels of each salient image, as explained in Chapter 3. These histograms were max pooled to get one vector of size $mathbbR^{1\times 60}$, where 60 is the number of bin orientations. A histogram is attained for every image in the training and testing class.

The mean of all the histograms of the training class was calculated to attain one histogram of length $\mathbb{R}^{1\times 60}$. Thus, one histogram vector represents each class. This vector is normalized from [0,1] and encoded into regular expression strings. Each regular expression pattern for each class can then be represented as a state machine on the automata. The histogram vector of each test image is also similarly encoded into regular expression patterns. The test image is classified to the class with matching patterns. The classification results are shown in Table 6.1.2.

Table 6.1.2 does not show a confusion matrix because this classifier can classify an image to more than one class. This is helpful to note for future classification designs on an automata framework. The table does show that the patterns extracted using HOG descriptors are discriminative enough to differentiate some classes. However, similar looking objects, such as the daisy

	Airplane	Daisy	Windflower	Spleen
Airplane	**.72**	.006	.009	.28
Daisy	0	**.77**	.75	.04
Windflower	0	.61	**.74**	.04
Motorbike	.17	.17	.17	**.83**

Table 6.1: The table shows the percent of images classified to each class. The values in bold are the true positives. This is not a confusion matrix because an image can be classified to more than one class.

and windflower, are more difficult to distinguish. Table 6.2 shows the hamming distance between these four classes. The hamming distance calculates the number of positions between two strings at which the symbols are different. Therefore, these values represent, on average, the dissimilarities in the string patterns intra-class and between classes. The hamming distances tell us that the string patterns between the classes need to be more distinct than the string patterns within a class.

Table 6.2: Hamming Distance

	Airplane	Daisy	Windflower	Spleen
Airplane	**1.5**	40.2	41.8	12.3
Daisy	47.7	**4.5**	4.5	21.9
Windflower	48.2	7.7	**6.1**	26.2
Motorbike	22.2	11.5	16.3	**3.1**

Table 6.3: The hamming distance between the four classes with the maximum value possible as 60. The bolded values represent the hamming distance between strings of the same class.

This experiment illustrates a need for extracting discriminant features, particularly when images of different classes are similar. Further, a few parameters, including the number of superpixels, length of histogram vector, and number of character symbols, were tested in this experiment to optimize classification accuracy. These parameters were implemented in the experiments mentioned in this thesis.

BIBLIOGRAPHY

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[2] Jakob Nikolas Kather, Cleo-Aron Weis, Francesco Bianconi, Susanne M Melchers, Lothar R Schad, Timo Gaiser, Alexander Marx, and Frank Gerrit Zöllner, "Multi-class texture analysis in colorectal cancer histology," *Scientific Reports*, vol. 6, 2016.

[3] Paul Dlugosch, Dave Brown, Paul Glendenning, Michael Leventhal, and Harold Noyes, "An efficient and scalable semiconductor architecture for parallel automata processing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 12, pp. 3088–3098, 2014.

[4] Keira Zhou, Jeffrey J Fox, Ke Wang, Donald E Brown, and Kevin Skadron, "Brill tagging on the micron automata processor," in *Semantic Computing (ICSC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 236–239.

[5] Indranil Roy and Srinivas Aluru, "Finding motifs in biological sequences using the micron automata processor," in *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*. IEEE, 2014, pp. 415–424.

[6] Tommy Tracy II, Yao Fu, Indranil Roy, Eric Jonas, and Paul Glendenning, "Towards machine learning on the automata processor," in *International Conference on High Performance Computing*. Springer, 2016, pp. 200–218.

[7] Pedro J Moreno Purdy P Ho and Nuno Vasconcelos, “A kullback-leibler divergence based kernel for svm classification in multimedia applications,” *Proc Adv Neural Inf Process Syst*, vol. 16, pp. 1385–1392, 2004.

[8] Arnold WM Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain, “Content-based image retrieval at the end of the early years,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 12, pp. 1349–1380, 2000.

[9] Navneet Dalal and Bill Triggs, “Histograms of oriented gradients for human detection,” in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. IEEE, 2005, vol. 1, pp. 886–893.

[10] Scott T Acton and Nilanjan Ray, “Biomedical image analysis: Segmentation,” *Synthesis Lectures on Image, Video, and Multimedia Processing*, vol. 4, no. 1, pp. 1–108, 2009.

[11] Dipti Prasad Mukherjee and Scott T Acton, “Cloud tracking by scale space classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 2, pp. 405–415, 2002.

[12] Alaa E Abdel-Hakim and Aly A Farag, “Csift: A sift descriptor with color invariant characteristics,” in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. IEEE, 2006, vol. 2, pp. 1978–1983.

[13] David G Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[14] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” in *Computer vision and pattern recognition, 2006 IEEE computer society conference on*. IEEE, 2006, vol. 2, pp. 2169–2178.

[15] Yan Ke and Rahul Sukthankar, "Pca-sift: A more distinctive representation for local image descriptors," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. IEEE, 2004, vol. 2, pp. II–II.

[16] Florent Perronnin, Yan Liu, Jorge Sánchez, and Hervé Poirier, "Large-scale image retrieval with compressed fisher vectors," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 3384–3391.

[17] Imran S Bajwa and S Irfan Hyder, "Pca based image classification of single-layered cloud types," in *Emerging Technologies, 2005. Proceedings of the IEEE Symposium on*. IEEE, 2005, pp. 365–369.

[18] Se Rim Park, Soheil Kolouri, Shinjini Kundu, and Gustavo Rohde, "The cumulative distribution transform and linear pattern classification," *arXiv preprint arXiv:1507.05936*, 2015.

[19] Soheil Kolouri, Se Rim Park, and Gustavo K Rohde, "The radon cumulative distribution transform and its application to image classification," *IEEE transactions on image processing*, vol. 25, no. 2, pp. 920–934, 2016.

[20] Michael HLS Wang, Gustavo Cancelo, Christopher Green, Deyuan Guo, Ke Wang, and Ted Zmuda, "Using the automata processor for fast pattern recognition in high energy physics experimentsâĂŤa proof of concept," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 832, pp. 219–230, 2016.

[21] Indranil Roy, Ankit Srivastava, Marziyeh Nourian, Michela Becchi, and Srinivas Aluru, "High performance pattern matching using the automata processor," in *Parallel and Distributed Processing Symposium, 2016 IEEE International*. IEEE, 2016, pp. 1123–1132.

[22] Indranil Roy and Srinivas Aluru, "Discovering motifs in biological sequences using the micron automata processor," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 13, no. 1, pp. 99–111, 2016.

[23] Chunkun Bo, Ke Wang, Jeffrey J Fox, and Kevin Skadron, "Entity resolution acceleration using micronâĂŹs automata processor," *Architectures and Systems for Big Data (ASBD), in conjunction with ISCA*, 2015.

[24] Ke Wang, Elaheh Sadredini, and Kevin Skadron, "Sequential pattern mining with the micron automata processor," in *Proceedings of the ACM International Conference on Computing Frontiers*. ACM, 2016, pp. 135–144.

[25] Ke Wang, Yanjun Qi, Jeffrey J Fox, Mircea R Stan, and Kevin Skadron, "Association rule mining with the micron automata processor," in *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*. IEEE, 2015, pp. 689–699.

[26] Kevin Angstadt, Westley Weimer, and Kevin Skadron, "Rapid programming of pattern-recognition processors," in *ACM SIGPLAN Notices*. ACM, 2016, vol. 51, pp. 593–605.

[27] Jack Wadden, Vinh Dang, Nathan Brunelle, Tommy Tracy II, Deyuan Guo, Elaheh Sadredini, Ke Wang, Chunkun Bo, Gabriel Robins, Mircea Stan, et al., "Anmlzoo: a benchmark suite for exploring bottlenecks in automata processing engines and architectures," in *Workload Characterization (IISWC), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 1–12.

[28] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto, "Class segmentation and object localization with superpixel neighborhoods," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 670–677.

[29] Chuan Yang, Lihe Zhang, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang, “Saliency detection via graph-based manifold ranking,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 3166–3173.

[30] Alex Levinshtein, Adrian Stere, Kiriakos N Kutulakos, David J Fleet, Sven J Dickinson, and Kaleem Siddiqi, “Turbopixels: Fast superpixels using geometric flows,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 12, pp. 2290–2297, 2009.

[31] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk, “Slic superpixels,” Tech. Rep., 2010.

[32] Kwang-Shik Kim, Dongni Zhang, Mun-Cheon Kang, and Sung-Jea Ko, “Improved simple linear iterative clustering superpixels,” in *Consumer Electronics (ISCE), 2013 IEEE 17th International Symposium on*. IEEE, 2013, pp. 259–260.

[33] Simona E Grigorescu, Nicolai Petkov, and Peter Kruizinga, “Comparison of texture features based on gabor filters,” *IEEE Transactions on Image processing*, vol. 11, no. 10, pp. 1160–1167, 2002.

[34] Qiang Zhu, Mei-Chen Yeh, Kwang-Ting Cheng, and Shai Avidan, “Fast human detection using a cascade of histograms of oriented gradients,” in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. IEEE, 2006, vol. 2, pp. 1491–1498.

[35] Paul S Bradley and Usama M Fayyad, “Refining initial points for k-means clustering.,” in *ICML*. Citeseer, 1998, vol. 98, pp. 91–99.

[36] Michal Aharon, Michael Elad, and Alfred Bruckstein, “rmk-svd: An algorithm for designing overcomplete dictionaries for sparse representation,” *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[37] Qiang Zhang and Baoxin Li, “Discriminative k-svd for dictionary learning in face recognition,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2691–2698.

[38] Zhuolin Jiang, Zhe Lin, and Larry S Davis, “Learning a discriminative dictionary for sparse coding via label consistent k-svd,” in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 1697–1704.

[39] Zhuolin Jiang, Zhe Lin, and Larry S Davis, “Label consistent k-svd: Learning a discriminative dictionary for recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2651–2664, 2013.

[40] Michael Elad and Michal Aharon, “Image denoising via sparse and redundant representations over learned dictionaries,” *IEEE Transactions on Image processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[41] Ron Rubinstein, Michael Zibulevsky, and Michael Elad, “Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit,” *Cs Technion*, vol. 40, no. 8, pp. 1–15, 2008.

[42] Umamahesh Srinivas, Hojjat Mousavi, Charles Jeon, Vishal Monga, Arthur Hattel, and Bhushan Jayarao, “Shirc: A simultaneous sparsity model for histopathological image representation and classification,” in *Biomedical Imaging (ISBI), 2013 IEEE 10th International Symposium on*. IEEE, 2013, pp. 1118–1121.

[43] Li Fei-Fei, Rob Fergus, and Pietro Perona, “Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories,” *Computer vision and Image understanding*, vol. 106, no. 1, pp. 59–70, 2007.

[44] Marcin Marszalek and Cordelia Schmid, "Accurate object localization with shape masks," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007, pp. 1–8.

[45] Eric Nowak, Frédéric Jurie, and Bill Triggs, "Sampling strategies for bag-of-features image classification," in *European conference on computer vision*. Springer, 2006, pp. 490–503.

[46] Jun Yang, Yu-Gang Jiang, Alexander G Hauptmann, and Chong-Wah Ngo, "Evaluating bag-of-visual-words representations in scene classification," in *Proceedings of the international workshop on Workshop on multimedia information retrieval*. ACM, 2007, pp. 197–206.

[47] Thomas Deselaers, Lexi Pimenidis, and Hermann Ney, "Bag-of-visual-words models for adult image classification and filtering," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*. IEEE, 2008, pp. 1–4.

[48] Stephen O'Hara and Bruce A Draper, "Introduction to the bag of features paradigm for image classification and retrieval," *arXiv preprint arXiv:1101.3354*, 2011.

[49] Mikala Egeblad, Elizabeth S Nakasone, and Zena Werb, "Tumors as organs: complex tissues that interface with the entire organism," *Developmental cell*, vol. 18, no. 6, pp. 884–901, 2010.

[50] Bangalore S Manjunath and Wei-Ying Ma, "Texture features for browsing and retrieval of image data," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 18, no. 8, pp. 837–842, 1996.

[51] Alan C. Bovik, Marianna Clark, and Wilson S. Geisler, "Multichannel texture analysis using localized spatial filters," *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, no. 1, pp. 55–73, 1990.

[52] Rituparna Sarkar and Scott T Acton, "Slide: Saliency guided image dictionary and image similarity evaluation," in *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 216–220.

[53] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[54] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[55] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[56] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[57] Kaiming He and Jian Sun, "Convolutional neural networks at constrained time cost," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5353–5360.

[58] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, pp. 115–118, 2017.