

**MACHINE LEARNING BASED REDUCTION AND  
QUANTIFICATION OF ERROR IN IMAGE CLASSIFICATION  
PIPELINES**

**Aritra Chowdhury**

Submitted in Partial Fullfillment of the Requirements  
for the Degree of

*DOCTOR OF PHILOSOPHY*

Examining committee:  
Dr. Bulent Yener, Chair  
Dr. Malik Magdon-Ismail  
Dr. Charles V. Stewart  
Dr. Alberto Santamaria-Pang



*Department of Computer Science*  
Rensselaer Polytechnic Institute  
Troy, New York

[August 2018]  
Submitted June 2018

© Copyright 2018  
by  
Aritra Chowdhury  
All Rights Reserved

## CONTENTS

LIST OF TABLES . . . . .	vi
LIST OF FIGURES . . . . .	vii
ACKNOWLEDGMENT . . . . .	ix
ABSTRACT . . . . .	x
1. INTRODUCTION AND BACKGROUND . . . . .	1
1.1 Outline . . . . .	1
1.2 Contributions . . . . .	1
2. IMAGE DRIVEN MACHINE LEARNING METHODS FOR MICROSTRUCTURE RECOGNITION . . . . .	2
2.1 Introduction . . . . .	2
2.2 Alloy Fabrication and Sample Preparation . . . . .	3
2.3 Image Data Sets . . . . .	4
2.4 Approach . . . . .	5
2.5 Computer Vision and Machine Learning Methods . . . . .	6
2.5.1 Feature extraction . . . . .	8
2.5.2 Dimensionality reduction . . . . .	12
2.5.3 Classification . . . . .	13
2.5.4 Related Work in Materials Science . . . . .	15
2.6 Selection of Model Parameters . . . . .	16
2.7 Software and Hardware Specifications . . . . .	16
2.8 Results and Discussion . . . . .	17
2.8.1 Task 1 - Dendritic versus non-dendritic microstructures . . . . .	17
2.8.2 Task 2 - Longitudinal versus transverse cross-sectional views of dendrites	20
2.9 Limitations . . . . .	23
2.10 Conclusions . . . . .	25
2.11 Future Work . . . . .	26

3. BLOOD VESSEL CHARACTERIZATION USING VIRTUAL 3D MODELS AND CONVOLUTIONAL NEURAL NETWORKS IN FLUORESCENCE MICROSCOPY	27
3.1 Introduction	27
3.2 Data	28
3.2.1 Natural data	29
3.2.2 Virtual data	29
3.3 Methods	31
3.4 Experiments and results	32
3.5 Conclusions	34
4. A MACHINE LEARNING APPROACH TO QUANTIFYING NOISE IN MEDICAL IMAGES	35
4.1 Introduction	35
4.2 Data	35
4.2.1 Preliminary noise reduction	36
4.2.2 Combination of the three markers	37
4.2.3 Feature extraction	37
4.2.4 Data preprocessing	37
4.3 Methods and experiments	38
4.3.1 Random undersampling	38
4.3.2 Classification and analysis	38
4.3.3 Quality of image score	38
4.3.4 Defining the <i>ugly</i> class	39
4.4 Conclusions	40
5. RANDOM SEARCH BASED QUANTIFICATION OF CONTRIBUTION AND PROPAGATION OF ERROR IN IMAGE CLASSIFICATION PIPELINES	41
5.1 Introduction and Motivation	41
5.2 Foundations	43
5.2.1 Algorithm selection and hyper-parameter optimization	44
5.2.1.1 Hyper-parameter optimization (HPO)	44
5.2.1.2 Combined algorithm selection and hyper-parameter optimization (CASH)	44
5.2.2 Optimization methods	45
5.2.2.1 Grid search	45
5.2.2.2 Random search	45
5.2.2.3 Bayesian optimization	45

5.3	Proposed methods . . . . .	46
5.3.1	Error quantification . . . . .	46
5.3.1.1	Quantification of error from computational steps . . . . .	46
5.3.1.2	Quantification of error from algorithms . . . . .	47
5.3.2	Error propagation . . . . .	48
5.4	Experiments and results . . . . .	50
5.4.1	Optimization frameworks . . . . .	51
5.4.2	Datasets . . . . .	52
5.4.3	Error quantification experiments . . . . .	53
5.4.3.1	Error contribution from computational steps . . . . .	53
5.4.3.2	Error contribution from algorithms . . . . .	54
5.4.4	Error propagation experiments . . . . .	55
5.5	Conclusion . . . . .	57
6.	CONCLUSIONS AND FUTURE WORK . . . . .	59
6.1	Conclusions . . . . .	59
	REFERENCES . . . . .	60

## LIST OF TABLES

2.1	Feature extraction methods and corresponding classification accuracies (in %), that contributed to the maximum classification accuracy for each combination of feature selection and classification method tested for Task 1 . . . . .	18
2.2	Feature selection methods and corresponding classification accuracies (in %), that contributed to the maximum classification accuracy for each combination of feature extraction and classification method tested for Task 1 . . . . .	19
2.3	Classification methods that contributed to the maximum classification accuracy (in %) for each combination of feature extraction and feature selection method tested for Task 1. . . . .	20
2.4	Feature extraction methods and corresponding classification accuracies (in %), that contributed to the maximum classification accuracy for each combination of feature selection and classification method tested for Task 2. . . . .	21
2.5	Feature selection methods, and corresponding classification accuracies (in %), that contributed to the maximum classification accuracy for each combination of feature extraction and classification method tested for Task 2. . . . .	22
2.6	Classification models that yielded the maximum classification accuracy (in %) for each combination of feature extraction and feature selection method tested for Task 2. . . . .	23
3.1	Distribution of vascular morphologies . . . . .	29
3.2	Comparison of feature extraction methodologies . . . . .	32
3.3	Results of binary classification between RoundLumen and Twin . . . . .	33
3.4	Results of binary classification between <i>RoundLumen-</i> and <i>RoundLumen+</i> . . . . .	33
5.1	Notations for error definitions used in the error propagation model . . . . .	49
5.2	Notations for error definitions used in the error propagation model . . . . .	52

## LIST OF FIGURES

4.5	Discrete distributions of QoI scores for (a) E_cad (b) pck26 (c) CK15 . . . . .	40
5.1	Representation of a machine learning pipeline . . . . .	41
5.2	Naive methodology used in the error propagation model . . . . .	48
5.3	Representation of an image classification pipeline . . . . .	50
5.4	Representation of the image classification pipeline . . . . .	51
5.5	Combined algorithm selection and hyper-parameter optimization in a data analytic pipeline. The algorithms and corresponding hyper-parameters are optimized simultaneously. . . . .	52
5.6	Hyper-parameter optimization in a data analytic pipeline. Each path in the pipeline is individually optimized. . . . .	52
5.7	Plots of error contributions from computational steps in the pipeline. . . . .	53
5.8	Plots of error contributions from algorithms in the pipeline (method 1). . . . .	54
5.9	Pipeline including naive algorithms for error propagation . . . . .	55
5.10	Pipeline including naive algorithms for error propagation . . . . .	55
5.11	Plots of error propagation using the naive algorithm based methodology. . . . .	56
5.12	Plots of propagation factor $\gamma$ using the naive algorithm based methodology. . . .	56
5.13	Plots of error propagation using the naive algorithm based methodology. . . . .	57
5.14	Plots of propagation factor $\gamma$ using the naive algorithm based methodology. . . .	57
5.15	Plots of error contribution and error propagation on test data from <i>breast</i> data.	57

## **ACKNOWLEDGMENT**

## ABSTRACT

Image classification is an important problem. It is an approach of pattern recognition in computer vision. The goal of the approach is to differentiate between different classes of images based on the quantification of contextual information in the images. Image classification is performed using data analytic pipelines. These pipelines are organized as interdependent and individual components. These components include image acquisition, image preprocessing, feature extraction, feature preprocessing, dimensionality reduction, learning algorithms. More steps may be added or removed from the pipeline. The quality of the image classification pipeline is measured by the error in classification of new image instances. This error is an accumulation of the errors introduced from different parts of the pipeline starting from the acquisition of the image from the sample to the algorithm used for performing the classification. This error is propagated down the pipeline which finally accumulated in the form of the aforementioned classification error. In this thesis, we attempt to reduce, quantify and understand the error in image classification pipelines.

In the first part, we try to reduce the error in a holistic manner. The problem selected for demonstrating this is that of automatic microstructure recognition. The dataset for this problem is from the domain of material science. It consists of images of microstructures in the micrometer scale. We perform two classification tasks. The first task is of differentiating between dendritic and non-dendritic microstructures. The second task is for differentiating between longitudinal and transverse cross sections of microstructures. The objective for both the classification tasks is to find the best set of algorithms and hyper-parameters for minimizing the classification error. The approach used was an exhaustive search of the configuration space of algorithms and corresponding hyperparameters for three stages of the pipeline - feature extraction, dimensionality reduction and learning algorithms. This exhaustive grid search is able to perform classification of the two tasks with a sufficiently high degree of accuracy.

# **CHAPTER 1**

## **INTRODUCTION AND BACKGROUND**

### **1.1 Outline**

### **1.2 Contributions**

# CHAPTER 2

## IMAGE DRIVEN MACHINE LEARNING METHODS FOR MICROSTRUCTURE RECOGNITION

### 2.1 Introduction

Materials characterization is a critical aspect of the material design and discovery process. Recently, there has been much research in the field of materials informatics, a growing research area in which information technology and data science methods are used to interpret and analyze material data in order to accelerate the material discovery, design, and development process [?], [?], [1]–[3]. Currently, material design relies on chance discoveries and follows a classical synthesis-characterization-theory-computation approach [4]. Further, there is a heavy reliance on individual researcher background and experience which introduces significant bias and potentially error into the process of microstructure recognition, interpretation, and characterization. For example, quantification of microstructures traditionally is done using stereological measurements. Bias is introduced into stereological measurements through the requirement that an expert must first recognize and identify key microstructural features (inclusions, grains, or phases). This bias can be caused by a variety of factors, such as an individual’s background, education, and experiences [4].

Although there have been recent advances in the field of quantitative microstructural science, there is still a heavy dependence on expert knowledge to identify what microstructural features are of interest for quantification [5]. Therefore it is desirable to expand upon work previously presented by DeCost and Holm in Reference [5], and further explore methods of quantitative microstructure representations which do not require *a priori* knowledge of microstructural features of interest or significance [4]. This work aims to leverage existing computer vision and machine learning techniques specifically for the challenge of microstructure recognition. While the overlap between computer vision, machine learning, and materials science is currently small [4], this work is a step towards increasing cross-disciplinary studies that challenge the current paradigm for microstructure characterization.

---

This chapter previously appeared as: A. Chowdhury, E. Kautz, B. Yener, and D. Lewis. "Image driven machine learning methods for microstructure recognition." *Computational Materials Science*, vol. 123 pp. 176-187, 2016.

Dendritic morphologies were chosen for this small case-study since dendrites are a well-characterized microstructural feature that exists in a variety of material systems (from single to multi-component). Size, shape, and spacing of dendrites vary depending on solidification behavior and chemistry, thus micrographs of this single feature can vary widely. The sample preparation and imaging methods used also contribute to the variety of micrographs produced from dendritic microstructures. Despite this variability in image data, it is still possible for human experts to look at a micrograph that contains dendrites and identify that it contains this microstructural feature, even though different orientations of dendrites (transverse or longitudinal) look distinctly different.

Computer vision and machine learning methods were applied to the task of identifying a particular microstructural feature of interest (dendrites) from micrographs that do not contain this particular feature (just as a human expert would identify that a micrograph contains dendrites). This recognition task is referred to in this work as Task 1. Task 1 is a high-level microstructure recognition task in the sense that dendrites are a type of microstructural feature that are not specific to a material system. A second classification task (referred to here as Task 2) was also completed, and involved distinguishing between longitudinal and transverse cross-sectional views of dendritic microstructures. This task may be viewed as a logical next step following the identification of dendrites in Task 1. If the micrograph from Task 1 was identified as a dendrite, then a second binary classification task was performed, with the goal of distinguishing between two different cross-sectional views.

The contribution in this work is to investigate multiple computer vision and machine learning methods for microstructure recognition. We hypothesize that the approach and methods presented here can be generalized, and thus applied to a variety of microstructure recognition tasks that act as a necessary first step in characterization of a material system.

## 2.2 Alloy Fabrication and Sample Preparation

The alloy fabrication, processing, and metallographic sample preparation procedure followed to obtain images used in this work was based on the process detailed in References [6] and [7] and is summarized here.

A Materials Research Furnaces (MRF) three probe arc melter was used to fabricate alloys of varying Sn-Ag-Cu compositions. After melting alloys were allowed to solidify and were then prepared for directional solidification (DS). The solidified buttons were placed

in a beaker on a hot plate. The button was re-melted while constantly being purged with Ar gas to minimize sample oxidation. The alloy melt was then transferred into a 4 mm inner diameter quartz ampule with a mechanical pump. The rods were allowed to cool, then removed from the ampule, and placed in a larger quartz ampule (5 mm inner diameter) for DS. This ampule was then back-filled with Ar gas, sealed using a hydrogen torch, and inserted into the DS furnace.

DS was performed using a Bridgman-type apparatus in order to refine alloy microstructure. The tube furnace in this apparatus is a Thermolyne Type-21100 fitted with an Omega temperature controller. Following DS, alloys were removed from the quartz ampule, and small sections from the middle third of the rod were mounted in epoxy for metallographic sample preparation. Sections were mounted such that transverse and longitudinal orientations of  $\beta$ -Sn dendrites could be viewed. Samples were ground using silicon carbide (SiC) papers to 600 grit, then polished using 9, 3, and 1  $\mu\text{m}$  diamond slurries. Colloidal silica was used as the final polishing step and as a chemical etchant so that the dendritic microstructure would be readily visible using light optical microscopy.

### 2.3 Image Data Sets

Image data used in this study includes micrographs taken over a span of approximately three years by students in the Lewis Research Group in the Materials Science and Engineering Department at Rensselaer Polytechnic Institute (RPI). All images taken by Lewis Research Group members are of solder alloys with dendritic microstructures. These alloys were manufactured, processed, and imaged at RPI, and a representative process of sample preparation was presented in Section 2.2. In order to supplement micrographs obtained at RPI, micrographs from the publicly available Dissemination of Information Technology for the Promotion of Materials Science (DoITPoMS) library [8] were used. Example images used in classification are provided in Figure 2.1.

Images were grouped into two data sets: Data Set 1 and Data Set 2, corresponding to classification Tasks 1 and 2 respectively. Data Set 1 is comprised of micrographs with and without dendritic morphologies. This data set includes all images with dendrites (longitudinal and transverse cross-sections) from both the Lewis Group and the DoITPoMS micrograph library [8]. All micrographs that do not depict dendritic morphologies were obtained from the DoITPoMS library. Data Set 1 includes 528 images that are each 227 by

227 square pixels. 264 original images were used in making Data Set 1 (132 micrographs containing dendrites and 132 micrographs without dendrites). Each original image is 270 by 500 pixels but includes a scale bar that interferes with feature extraction, therefore each image was cropped to yield two 227 by 227 square pixel images. Thus, 528 images were used for classification.

Data Set 2 is a subset of Data Set 1, and is comprised of micrographs that only contain dendrites, where each micrograph is either a transverse or longitudinal cross-sectional view. Micrographs used in this data set include both images from the Lewis Group and the DoITPoMS micrograph library. Data Set 2 contains a total of 188 images that are each 227 by 227 square pixels. As was done for Data Set 1, original images were cropped to remove the scale bar: 47 micrographs of each longitudinal and transverse sections (total of 94 original images) were cropped to create a 188 images used for classification.

## 2.4 Approach

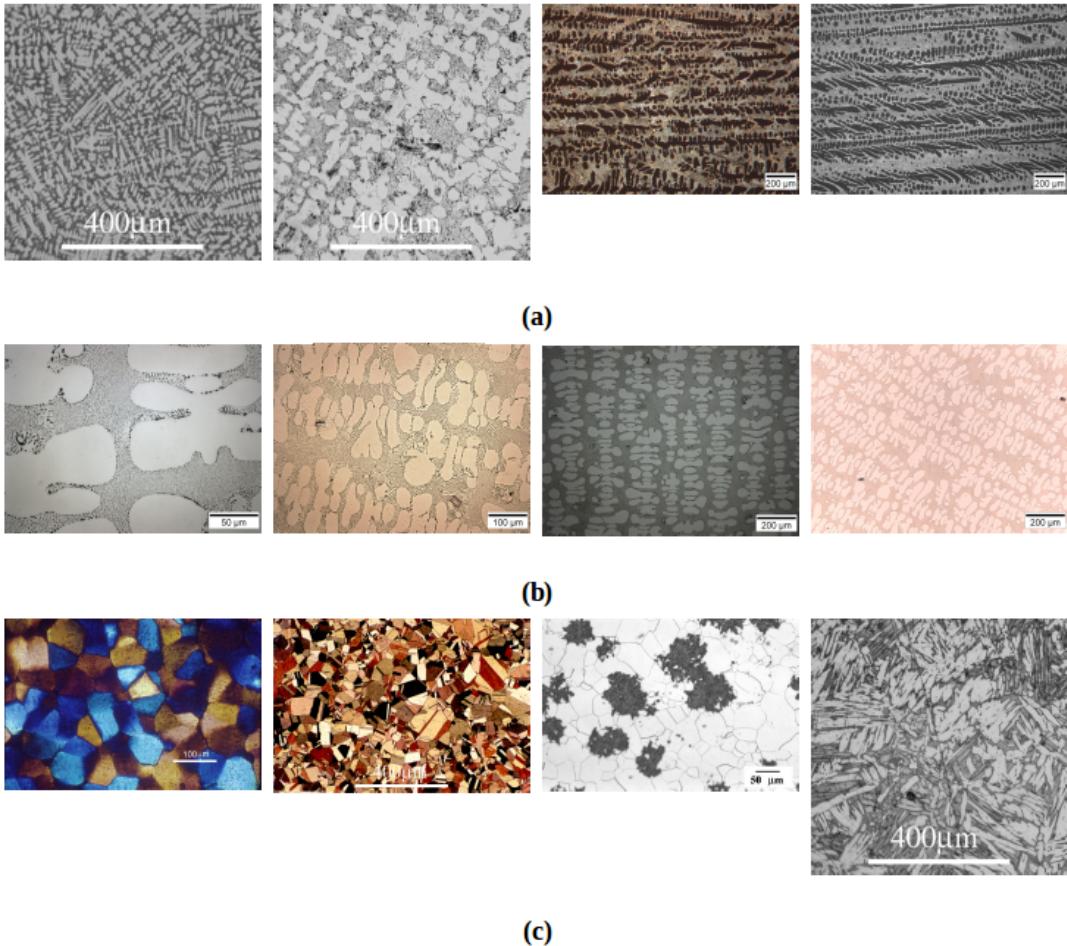
The general approach applied to the task of micrograph classification involved feature extraction and feature selection (dimensionality reduction) to compute feature vectors that were then used for training, validating, and testing various classification models. The same approach was applied to both Data Sets 1 and 2 and is shown schematically in Figure 2.2.

Feature extraction is the first step in the process of classifying micrographs. Feature extraction starts with feature detection, where features in an image are local regions of pixels that include an ‘interesting’ part of a microstructure, such as a corner, edge, or blob-like object. Features detected using computer vision algorithms are not necessarily semantically meaningful, however are pixel patterns that are mathematically repeatable and recognizable, thereby making the region a good feature. Detected features are then described (or represented) in the form of a vector, called a feature vector. A feature vector is comprised of a set of detected features in the image, and each image can thus be described or represented by a feature vector. Multiple feature extraction methods were tested in this work in order to compare classification accuracies of different image representations.

These feature vectors, derived in the process of feature extraction, have high dimensionality and in this work feature selection was performed after feature extraction for computational efficiency and to preclude the curse of dimensionality . Feature selection involves

---

The phrase ‘curse of dimensionality’ refers to issues associated with increasing data dimensionality in



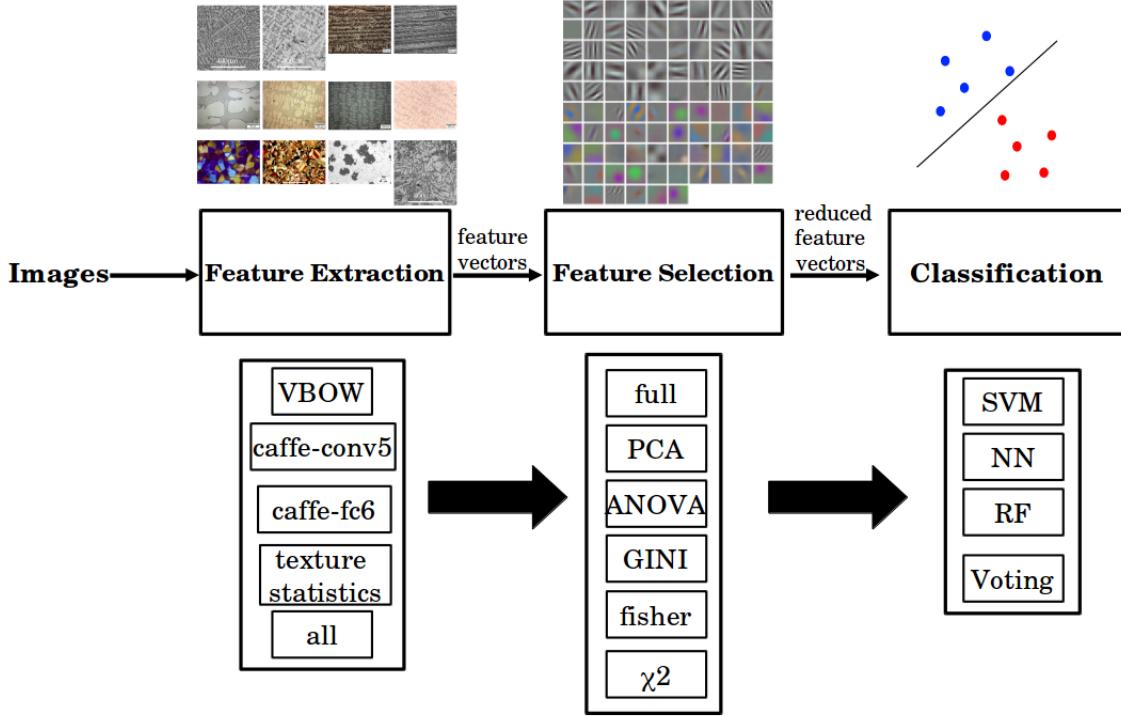
**Fig. 2.1. Examples of micrographs used in classification.** Micrographs shown in (a) and (b) are longitudinal and transverse cross-sectional views of dendrites, whereas micrographs in (c) do not contain dendrites. Micrographs in (a), (b) and (c) were used in Task 1, and micrographs in (a) and (b) were used in Task 2.

reducing the length of the feature vector while still retaining image information. For example, if a feature vector is sparse, dimensionality reduction would involve reducing the sparsity of this vector. Different dimensionality reduction methods were applied to test how each technique impacted classification accuracies.

After feature extraction and selection, various classification models were trained, validated, and tested. A 5 fold cross validation split was used for each data set. This split was selected in order to minimize variance in model parameters, and to avoid over-fitting. These validation accuracies were then averaged and the standard deviation was calculated in order

---

Euclidean space. Behavior of algorithms in low dimensional space, such as in three dimensions, may not generalize well to a higher dimensional space [9].

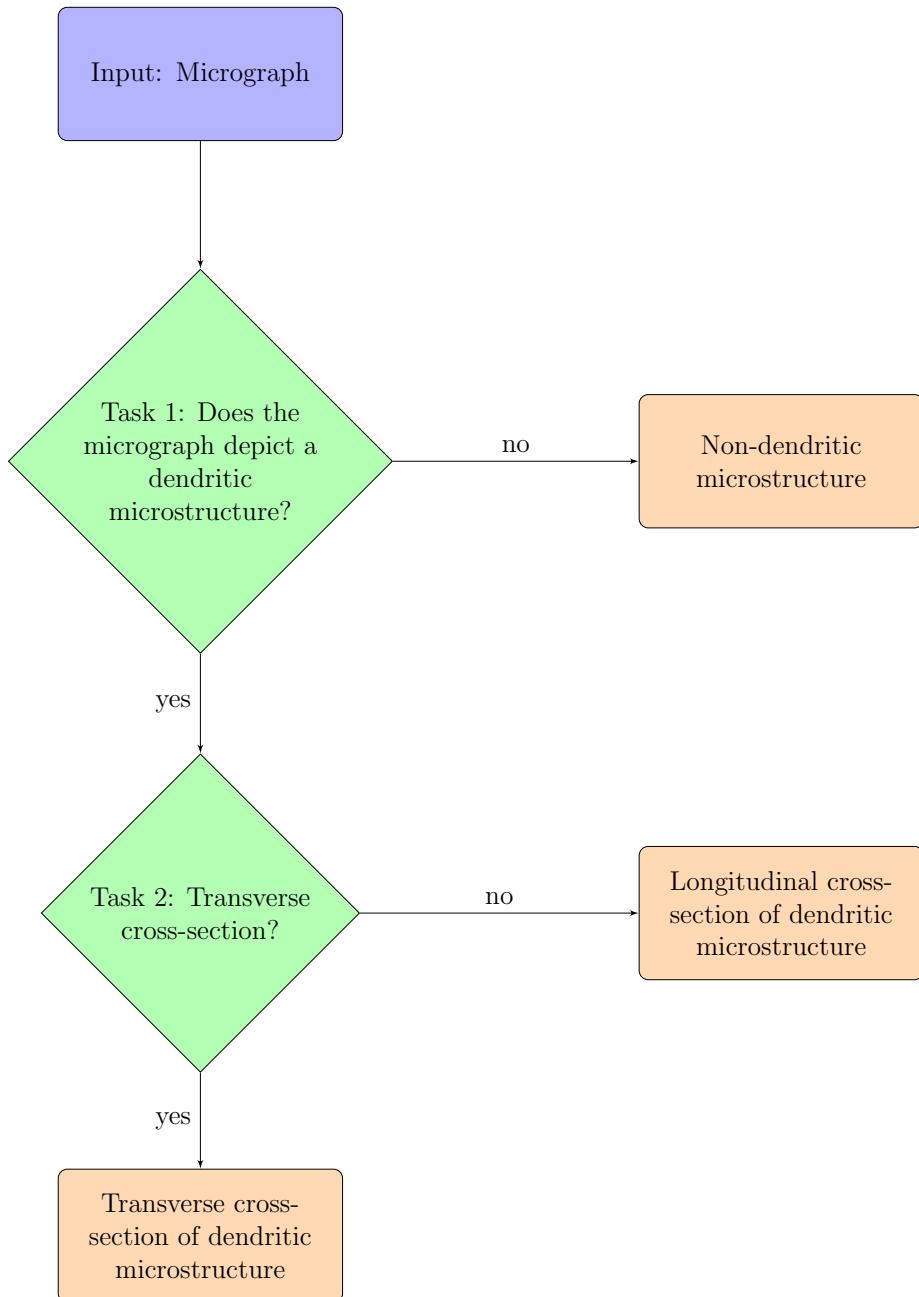


**Fig. 2.2.** Overview of approach used in classification of micrograph data. The approach summarized here shows 140 different combinations of feature extraction, feature selection and classification methods completed. The same approach presented here was completed first for Task 1 (Data Set 1), then for Task 2 (Data Set 2).

to evaluate model stability. We performed analysis on the two datasets (described in Section 2.3) in the form of two classification tasks: Task 1 and Task 2. The relationship between these tasks is depicted in Figure 2.3.

## 2.5 Computer Vision and Machine Learning Methods

As previously introduced in Section 2.4, multiple feature extraction and selection methods were tested with various classifiers in order to evaluate the best methods to apply to microstructure recognition tasks. The methods applied in this work are discussed in additional detail in Sections 2.5.1 - 2.5.3. Additionally, Section 2.5.4 provides a brief overview of computer vision and machine learning methods applied to other research in the field of materials science.



**Fig. 2.3.** Flowchart of microstructure recognition tasks, referred to here as **Task 1** and **Task 2**. Tasks 1 and 2 are both binary classification tasks. Task 1 involves distinguishing between micrographs that depict dendritic morphologies from those that do not contain this particular microstructural feature. From micrographs identified as containing dendrites (completed in Task 1), Task 2 involves distinguishing between different cross-sectional views (longitudinal or transverse) of dendritic microstructures.

### 2.5.1 Feature extraction

Feature extraction methods applied in this work are texture and shape statistics (which include Haralick texture features [10], histogram of oriented gradients (HoG) [11], local binary patterns (LBP) [12], Hu moments [13], [14], Zernike Moments [15], threshold adjacency statistics [16]), visual bag of words (Vbow) [17], [18], and pre-trained convolutional neural networks (CNN's). We used the publicly available Caffe framework [19] for all experiments involving pre-trained CNN's. The network was based on the architecture of Krizhevsky *et al.* for ImageNet [20]. The approach of using a pre-trained network was taken in this case-study because this is a good first step that does not require training to determine model parameters, which would be a difficult task with the small data sets used in this study [19].

Texture and shape statistics involve multiple feature extraction algorithms (previously listed) that operate on image texture and shape of objects within the image. These methods are introduced below:

Haralick features [10] are calculated using gray-level co-occurrence matrices ( $G_{ij}$ , shown below) which are square matrices of size  $N_g \times N_g$ , where  $N_g$  is the number of gray levels in an image:

$$G_{ij} = \begin{bmatrix} p(1, 1) & p(1, 2) & \cdots & p(1, N_g) \\ p(2, 1) & p(2, 2) & \cdots & p(2, N_g) \\ \vdots & \vdots & \ddots & \vdots \\ p(N_g, 1) & p(N_g, 2) & \cdots & p(N_g, N_g) \end{bmatrix} \quad (2.1)$$

Each matrix element,  $[i, j]$ , is calculated by counting the number of times a pixel with value  $i$  is adjacent to pixel with value  $j$ . When considering a square pixel image, there are four directions for which a gray-level co-occurrence matrix can be calculated: horizontal, vertical, left diagonal, and right diagonal. Haralick texture statistics can be calculated based on the four gray-level co-occurrence matrices from each of these four directions. Haralick features thus describe the texture of an image. Fundamentally, texture is the arrangement of a certain feature or property, relative to the environment. In images, texture is the spatial distribution (arrangement) of gray-level variations in an image. Haralick features, computed from image gray-level co-occurrence matrices, capture texture patterns in an image.

The histogram of oriented gradients (HoG) [11] method describes an image with a set

of histograms of gradient orientations from local regions of an image (referred to as cells). Each histogram reports the number of times a gradient orientation occurs in a given cell. In general, an image gradient is a two-vector that points in the direction of the greatest rate of change. The two components in the gradient vector include the gradient magnitude,  $M(x, y)$ , and the gradient orientation,  $\alpha(x, y)$ :

$$M(x, y) = \sqrt{g_x^2 + g_y^2} \quad (2.2)$$

$$\alpha(x, y) = \tan^{-1}\left(\frac{g_y}{g_x}\right) \quad (2.3)$$

where  $g_x$  and  $g_y$  are the x and y components of the gradient respectively. The normalized histograms of each cell are concatenated to form a descriptor vector for each image. The greater the number of bins, the more detailed the histogram, and thus feature vector.

Local Binary Patterns (LBP) [12] method also describes the texture of an image. Each image is split into cells and the neighborhood of each pixel (in each cell) is thresholded. The neighborhood surrounding each pixel is defined by the pixel's eight nearest neighbors. When the center pixel intensity value is greater than its neighbor, the pixel value is changed to 0, otherwise the pixel value is 1. For each pixel, an 8-digit binary number, which can be referred to here as a label, is created. A histogram is created for each cell that reports the frequency of each label. The histograms for each cell are normalized, then all histograms are concatenated to yield a feature vector for the entire image.

Hu moments [13], [14] are shape-based feature descriptors that describe, characterize, and quantify shapes of objects in an image. Hu moments are based on geometric moments, defined by  $m_{p,q}(x, y)$ :

$$m_{p,q}(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) dx dy \quad (2.4)$$

Since images are a collection of pixels, the moment of an image must be expressed in terms of summations instead of integrals, as follows:

$$m_{p,q}(x, y) = \sum_{-\infty}^{\infty} \sum_{-\infty}^{\infty} x^p y^q f(x, y) dx dy \quad (2.5)$$

From the above equation, the area of an image would simply be defined by the 0th moment. The centroid of an image can also be defined using this concept of geometric moments, as follows:

$$(\bar{x}, \bar{y}) = \left( \frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \quad (2.6)$$

When moments are taken with respect to an image centroid, the moments are translation invariant, therefore a geometric moment can be defined relative to the centroid by  $\mu_{p,q}(x, y)$ :

$$\mu_{p,q}(x, y) = \sum_{-\infty}^{\infty} \sum_{-\infty}^{\infty} (x - \bar{x})^p (y - \bar{y})^q f(x, y) dx dy \quad (2.7)$$

Hu derived seven moment invariants from the geometric moment equation,  $\mu_{p,q}(x, y)$ , that are invariant to scale, translation, and in-plane rotation and can be computed from a binary image.

In order to compute Hu moments, a binary image must be used. A binary image was generated from the gray scale input images, by thresholding each image using Otsu's adaptive threshold algorithm [14], which is a global thresholding method that selects the optimal threshold value based on the image histogram. Otsu's algorithm assumes the gray scale intensities of an image follow a bi-modal distribution with two classes, and the threshold value is selected to maximize the inter-class variance. Hu moments were computed based on connected components of pixels in the binary image.

Zernike moments [15] are similar to Hu moments in that they describe the shape of objects in an image. An important difference between Hu and Zernike moments is that Zernike polynomials are orthogonal basis functions for Zernike moments. The use of orthogonal polynomials makes Zernike moments rotation invariant, and more robust to noise and minor variations in shape than Hu moments. Zernike moments are also calculated from a thresholded (binary) image.

Threshold adjacency statistics [16] are calculated from thresholded binary images.

Three different binary images are generated from the original, using three different threshold values. For each white pixel in an image, the number of neighboring white pixels is counted [21]. The first statistic is the number of white pixels with no white neighbors. The second statistic is the number of white pixels with one white neighbor. The number of white pixels with white pixel neighbors up to eight is counted (where there are eight nearest neighbor for each pixel in an image). Such statistics can be combined to create a feature vector that is subsequently used to distinguish and classify images.

All texture and shape statistics used in feature extraction (Haralick texture features, HoG, LBP, Hu and Zernike moments, and threshold adjacency statistics) were used to create a 1,775 dimensional feature vector for each image that represents texture and shape of objects within the image.

In addition to texture and shape statistics, Visual Bag of Words (VBOW) [17], [18] was also used for feature extraction. The VBOW technique is inspired from the ‘bag of words’ method used in document classification, where documents are considered as unordered sets of words. The corresponding words in the domain of image processing are local image features. These word descriptors are constructed around key points located by the Speeded Up Robust Features (SURF) [18] key points detection algorithm. These key points are extracted at regular locations at different scales making them very robust. The key point descriptors are categorized using a vector quantization technique such as  $k$ -means. The output of this discretization is a dictionary. Each image can therefore be encoded in the form of a histogram. This is done by extracting feature descriptors from the image, followed by finding the corresponding words to which the key points correspond to in the dictionary. The VBOW representation produced a 100-dimensional feature vector which was the pre-selected vocabulary size of the model.

A pre-trained convolutional neural network (CNN) model (trained on the Imagenet database) was also used in feature extraction. The ImageNet database contains 1.2 million images with 1000 categories. Here, the fifth and sixth layers of the pre-trained neural network were extracted and then used as feature descriptors for micrographs. The sizes of the feature representation obtained from the pre-trained ImageNet model were 43,264 and 4,096 for the fifth layer (caffe-conv5) and sixth layer (caffe-fc6) respectively.

Thus, for each image a feature vector was computed using each method described above. Concatenation of all computed feature vectors yielded a single vector which is referred to as

“all” in Section 2.8. Feature selection (dimensionality reduction) and classification was then performed using feature vectors from each method and the combined (concatenated) feature vector.

### 2.5.2 Dimensionality reduction

Most of the image descriptors obtained from feature extraction methods lie in a high dimensional space (e.g. 1 x 43,264 for caffe-conv5). Therefore it becomes necessary to employ dimensionality reduction techniques to bypass the curse of dimensionality [22], and to improve computational efficiency. The process of reducing feature vector dimensionality is called dimensionality reduction, or feature selection. Six different dimensionality reduction techniques were tested in this work: principle component analysis (PCA) [23], ANOVA *F*-statistic (ANOVA) [24],  $\chi^2$  method [25], Fisher score [26], and Gini index [27]. A brief description of each method is provided in this section.

Principle component analysis (PCA) [23] is a feature transformation technique that uses orthogonal transformation to convert a set of observations of correlated variables to linearly uncorrelated ones. These orthogonal components are called principal components. In our work, we use only those principal components which account for 95 % of the variance in the data. PCA can be thought of as a transformation that reveals the structure of the data in a way that best explains the variance in the data. The drawback of this technique is that the features obtained after reduction are linear combinations of the original feature variables which occupy a different vector space than the original variables.

In addition to PCA, the Analysis of Variance (ANOVA) *F*-statistic [24] was used in feature selection. A one-way ANOVA F-test tests whether or not the different classes of the response variable have the same mean as the predictor variables. The *p*-value is calculated based on the *F*-statistic. These scores are then sorted in descending order and the features corresponding to the 30<sup>th</sup> percentile are selected for analysis.

The  $\chi^2$  test is used in statistics to test the independence of two events. In the domain of feature selection, it is used to test whether the occurrence of a particular variable and class are independent. The  $\chi^2$  scores are ranked and the scores corresponding highest scores within the 30<sup>th</sup> percentile are selected.

The idea behind the Fisher score is to find a subset of features such that the distance between data points in different classes is as large as possible and the distance between data

instances in the same class are as small as possible. We select the features that correspond to the 30<sup>th</sup> percentile of the highest Fisher scores.

The Gini index [27] is an impurity splitting method, where impurity is calculated based on the probability that any sample belongs to a particular class. The minimum value of the index is 0, indicating that all of the members of the set belong to the same class, allowing for maximum useful information to be obtained. The opposite is true of samples in the set that are equally distributed in each class. The 30<sup>th</sup> percentile threshold is used for selecting the features corresponding to Gini index scores.

### 2.5.3 Classification

Classification was performed on Data Sets 1 and 2 using four different methods: support vector machine (SVM) [28], random forest (RF) [29], nearest neighbor (NN) [30], and voting [31]. SVM, NN, and Naive Bayes [32] models were used as components of the voting classifier. Multiple classification algorithms were tested in order to determine which is the most accurate for micrograph classification. The classifiers applied to this case study were selected as representative models that have proven to yield high classification accuracies in other application areas, such as email filtering, text classification and more [33].

Support vector machines (SVMs) [28] are discriminative (or margin based) classifiers. A SVM constructs a hyperplane in high dimensional space that may be used for classification. The hyperplane is selected such that the distance from it to the nearest data point on each side is maximized, referred to as the maximum margin hyperplane. We perform both linear and non-linear classification. We perform linear SVM classification on the original feature space and the non-linearity is introduced by the radial basis function (RBF) kernel for transforming data points to an infinite dimensional space. Our approach is to experiment with different learning algorithms. Therefore, we choose a representative algorithm from particular families of learning algorithms. The RBF kernel has proven to be a good representation for non-linear kernels. Linear SVM is referred to here as SVM-L, and non-linear SVM using a RBF is referred to as SVM-RBF.

In addition to SVMs, nearest neighbor (NN) [30] classifier (an instance based classifier) was tested. The NN classifier assigns a test instance to a class based on the majority vote among the classes of the k-nearest neighbors of the training instances to the test instance. In our analysis, we selected k using five-fold cross validation.

Random Forest (RF) classifiers [29], also used in this work, are comprised of multiple decision trees and are therefore ensemble-based classifiers. A decision tree is a classification model that consists of a series of questions about the attributes of a data set, answers (such as yes or no), and follow-up questions. This series of questions is organized in a hierarchical manner and consists of edges that are connected to different types of nodes, including a root node, internal nodes, and leaf nodes. A root node has no incoming edges and zero or more outgoing edges. An internal node has one incoming edge and two or more outgoing edges. A leaf node (also referred to as a terminal node) has one incoming edge and no outgoing edges. Each leaf node is assigned a label, and all internal and root nodes are assigned test conditions. In order to perform classification using a decision tree model, a path is followed from root to terminal nodes. This process will involve applying different test conditions, obtaining an answer, and moving to another internal node, and repeating this general process until a leaf/terminal node is reached. The path followed in a decision tree model will yield a class label for a given data instance. Classification using RF is dependent on the prediction of class label from multiple decision trees. Each tree in the model gives a class label, and the RF chooses the class with the most votes among the decision trees. Similar to RF, the voting classifier [31] is also an ensemble-based classifier that is based on majority voting among various classification modules. With RF, the modules were decision trees, and in this work we perform voting among three independent algorithms: SVM-RBF, NN and Naive Bayes.

Naive Bayes is a classifier model that depends on the inherent distribution of the training data set, and is thus categorized as a generative classifier. The Naive Bayes classifier uses Bayes' probability model and a decision rule to predict class of a data instance, where in this work each data instance is an image.

Naive Bayes is a statistical classifier that uses the concept of posterior probability for classification of new data instances. Each new data instance is assigned the class label that has the highest posterior probability. For a given set of variables  $X = x_1, x_2 \dots x_d$ , the posterior probability for event  $C_j$  is calculated from the set of possible outcomes,  $C = c_1, c_2, \dots c_d$ , where the set of possible outcomes is all class labels. Bayes' rule defines posterior probability of a

class label (the probability  $X$  belongs to  $C_j$ ) as:

$$p(C_j|x_1, x_2, \dots, x_d) \propto p(x_1, x_2, \dots, x_d|C_j)p(C_j) \quad (2.8)$$

where  $p(C_j|x_1, x_2, \dots, x_d)$  is the posterior probability of class label  $C_j$ . Posterior probability can be re-written, given that conditional probabilities of independent variables are assumed to be statistically independent:

$$p(C_j|X) \propto \prod_{k=1}^d p(C_j)p(x_k|C_j) \quad (2.9)$$

Using the Naive Bayes classifier, the new data instance  $X$  is classified by the class label  $C_j$  that has the highest posterior probability. In addition to the probability model defined above, the maximum *a posteriori* decision rule is applied for classification, meaning the hypothesis that is most probable is selected as the class label.

Classification was completed using each of the methods detailed above, for the full data set in addition to the reduced data sets obtained by the different feature selection techniques detailed in Section 2.5.2. Model parameters were determined using five-fold cross-validation.

In order to determine model accuracy and stability, average classification accuracies and standard deviations were calculated for every possible combination of classification, feature extraction, and feature selection method used. Average classification accuracies were calculated by taking the average of 5 fold cross validation of the dataset. The parameters of each algorithm were selected using the training fold of each split. Therefore the parameters were selected using another 5 fold cross validation of the training set and the training set was re-trained using the best combination of the parameters and the test accuracy on the remaining validation fold was calculated. This was performed 5 times and the average and standard deviation of the 5-fold validation accuracies was calculated.

#### 2.5.4 Related Work in Materials Science

The methods discussed here are well established in the computer vision and machine learning disciplines, however only a select few have been previously applied to materials science and engineering challenges. Shape descriptor methods, such as Hu moment invariants, have previously been used to quantitatively describe microstructures for subsequent classification [34]–[38]. Specific applications of Hu moments in the field of materials science include

automated classification of precipitate shapes in a two-phase microstructure [37], classification of cast iron microstructures [34]–[36], and analysis of precipitate shapes in nickel-based superalloys [38].

Additionally, the Visual Bag of Words feature extraction method with a Support Vector Machine classifier were used for multi-class classification of microstructures by DeCost and Holm [5], as previously mentioned in Section 2.1.

## 2.6 Selection of Model Parameters

Each algorithm involved in our experimentation has their own set of parameters. Some of the parameters are set *a priori* and some are tuned using 5-fold cross-validation. Since we used cross validation to set the parameters for each combination of feature extraction, feature selection and classifier, there is no specific setting for each individual parameter for the algorithm. This approach of using 5-fold cross-validation to set parameters generates results that are more robust and thus more accurate with respect to each particular combination of feature extraction, feature selection, and classifier. We do a grid search over the parameter space using 5-fold cross-validation to estimate the best setting of the different parameters. Such parameters include number of neighbors in the nearest neighbor classifier, the SVM margin parameter ( $C$ ), the RBF kernel parameter ( $\gamma$ ), the number of trees or estimators in the random forests classifier. The parameters found by cross-validation are presented in detail in the supplementary article to this paper.

We also set a number of parameters before the experiments are run to limit the number of parameters to tune using cross-validation. Such set parameters include the those listed in Table ??.

Default values were used for other parameters. We set the parameters in the feature extraction algorithms because this would represent a consistent test bed for the thorough computational experimentation over the feature selection and classification algorithms.

## 2.7 Software and Hardware Specifications

All of our experimentation was carried out with Python version 2.7 with the help of various open-source libraries. The opencv, scipy, skimage, numpy, mahotas and caffe packages (compatible with the Python version used in this work) were used for feature

extraction. The skfeature library was used for experimentation with the various feature selection algorithms. The sklearn package was used for the various learning algorithms.

The hardware used for experimentation was a 3.3 GHz 4 core Intel i5 CPU.

## 2.8 Results and Discussion

This section reports the results from applying multiple feature extraction, feature selection, and classification methods to the task of micrograph recognition. Performance comparisons between feature extraction, selection, and classification methods are presented in Sections 2.8.1 and 2.8.2. The success of different methods are compared based on the mean classification accuracies and standard deviations. Results are presented in the form of tables, where each table compares all possible combinations of the three modalities (i.e. feature extraction, feature selection, and classification) that yield the maximum classification accuracy. The tables presented here summarize a large set of results that are provided in a supplement to this article.

### 2.8.1 Task 1 - Dendritic versus non-dendritic microstructures

Table 2.1 reports the feature extraction methods that yielded maximum classification accuracies for each combination of feature selection and classifier tested. It is clear from Table 2.1 that caffe-fc6 was the most effective feature extraction method because the majority of the maximum classification accuracies in Table 2.1 were computed using feature vectors obtained using caffe-fc6. These results suggest that microstructure image data is represented well using pre-trained neural networks (a type of deep learning algorithm), and this feature extraction method generalizes well.

It is also important to note that caffe-fc6 was not re-trained on Data Sets 1 and 2 used in this case study. The pre-trained weights determined via training on the millions of natural images in the ImageNet database were used, which were successfully applied to feature extraction here. This result is significant because in using a pre-trained neural network, no large database (i.e. millions) of micrographs was needed. The use of the sixth layer (fully connected) from the Caffe network [19] for feature extraction (caffe-fc6) resulted in superior classification performance over other feature extraction methods such as VBOW and texture and shape statistics for Task 1.

It is suggested that caffe-fc6 produced higher classification accuracies overall than

VBOW and texture and shape statistics for Task 1 based on the difference in image representations (in the form of feature vectors) between each method. Furthermore, caffe-fc6 provided a more accurate image representation than caffe-conv5 (the 5th convolution layer of the Caffe network). This result could be due to the fact that features extracted from the sixth layer was more representative of the image dataset.

**Table 2.1. Feature extraction methods and corresponding classification accuracies (in %), that contributed to the maximum classification accuracy for each combination of feature selection and classification method tested for Task 1.**

Feature Selection	Classifier				
	SVM-L	SVM-RBF	NN	RF	Voting
<b>Full</b>	<b>caffe-fc6</b> $90.52 \pm 4.9$	<b>caffe-fc6</b> $90.52 \pm 4.9$	<b>caffe-fc6</b> $87.69 \pm 4.69$	<b>caffe-fc6</b> $87.88 \pm 6.19$	<b>caffe-fc6</b> $89.02 \pm 5.54$
<b>PCA</b>	<b>caffe-fc6</b> $89.03 \pm 3.58$	<b>caffe-fc6</b> $89.03 \pm 3.58$	<b>caffe-fc6</b> $89.96 \pm 2.53$	VBOW $87.88 \pm 1.84$	<b>caffe-fc6</b> $88.64 \pm 3.09$
<b>ANOVA</b>	<b>caffe-fc6</b> $90.91 \pm 4.32$	<b>caffe-fc6</b> $90.91 \pm 4.32$	<b>caffe-fc6</b> $87.31 \pm 3.23$	<b>caffe-fc6</b> $87.13 \pm 5.35$	<b>caffe-fc6</b> $91.85 \pm 4.25$
$\chi^2$	<b>caffe-fc6</b> $90.54 \pm 3.99$	<b>caffe-fc6</b> $90.54 \pm 3.99$	<b>caffe-fc6</b> $87.3 \pm 3.73$	caffe-conv5 $87.49 \pm 4.5$	<b>caffe-fc6</b> $90.53 \pm 4.96$
<b>fisher</b>	<b>caffe-fc6</b> $89.01 \pm 5.82$	<b>caffe-fc6</b> $89.01 \pm 5.82$	<b>caffe-fc6</b> $89.58 \pm 5$	texture-statistics $87.5 \pm 6.1$	texture-statistics $89.58 \pm 5.76$
<b>GINI</b>	texture-statistics $89.01 \pm 4.05$	texture-statistics $89.01 \pm 4.05$	<b>caffe-fc6</b> $89.39 \pm 4.64$	texture-statistics $87.5 \pm 5.46$	caffe-conv5 $89.58 \pm 6.32$

Feature selection was also investigated in order to determine if there was an overall best method for dimensionality reduction, and results are provided in Table 2.2. From results presented here, there is no one feature selection method that provided maximum classification accuracies for the majority of feature extraction and classifier combinations tested. This result could be attributed to the variability in the micrographs that do not depict dendrites (in Data Set 1). The micrographs selected as depicting non-dendritic morphologies are not all from the same material system and there is no common microstructural feature amongst this class.

The results also show that the feature selection method is sensitive to the types of feature extraction techniques and classifier models applied. Therefore, choice of dimensionality reduction should depend on the methods used for classification and feature extraction.

**Table 2.2. Feature selection methods and corresponding classification accuracies (in %), that contributed to the maximum classification accuracy for each combination of feature extraction and classification method tested for Task 1.**

<b>Feature Extraction</b>	<b>Classifier</b>				
	<b>SVM-L</b>	<b>SVM-RBF</b>	<b>NN</b>	<b>RF</b>	<b>Voting</b>
<b>VBOW</b>	Full $86.93 \pm 2.09$	Full $86.93 \pm 2.09$	Full $84.45 \pm 3.16$	PCA $87.88 \pm 1.84$	Full $85.4 \pm 3.53$
	ANOVA $90.91 \pm 4.32$	ANOVA $90.91 \pm 4.32$	PCA $89.96 \pm 2.53$	Full $87.88 \pm 6.19$	ANOVA $91.85 \pm 4.25$
<b>caffe-fc6</b>	ANOVA $89.96 \pm 6.11$	ANOVA $89.96 \pm 6.11$	ANOVA $85.04 \pm 4.48$	$\chi^2$ $87.49 \pm 4.5$	$\chi^2$ $90.15 \pm 5.08$
	$\chi^2$ $89.96 \pm 5.37$	$\chi^2$ $89.96 \pm 5.37$	fisher $89 \pm 5.74$	fisher $87.5 \pm 6.1$	$\chi^2$ $89.78 \pm 4.82$
<b>texture-statistics</b>	$\chi^2$ $89.96 \pm 5.37$	$\chi^2$ $89.96 \pm 5.37$	fisher $89 \pm 5.74$	GINI $86.74 \pm 5.51$	$\chi^2$ $89.78 \pm 4.8$
<b>combined</b>	$\chi^2$ $89.96 \pm 5.37$	$\chi^2$ $89.96 \pm 5.37$	fisher $89 \pm 5.74$		

Performance comparisons of classification models were also done for Task 1, and results are provided in Table 2.3. Linear SVM (SVM-L) provides maximum classification accuracies for over 50% of the possible feature extraction and selection combinations, indicating that this model generalizes well. This result also demonstrates that the data is linearly separable and use of a Gaussian kernel is not necessary for Task 1.

**Table 2.3.** Classification methods that contributed to the maximum classification accuracy (in %) for each combination of feature extraction and feature selection method tested for Task 1.

		Feature Extraction			
Feature Selection	VBOW	caffe-fc6	caffe-conv5	texture-statistics	combined
Full	<b>SVM-L</b> $86.93 \pm 2.09$	<b>SVM-L</b> $90.52 \pm 4.9$	RF $87.12 \pm 4.37$	<b>SVM-L</b> $87.88 \pm 3.98$	<b>SVM-L</b> $87.88 \pm 3.98$
PCA	RF $87.88 \pm 1.84$	NN $89.96 \pm 2.53$	<b>SVM-L</b> $87.3 \pm 4.43$	<b>SVM-L</b> $88.64 \pm 3.32$	<b>SVM-L</b> $88.64 \pm 3.32$
ANOVA	RF $83.72 \pm 1.99$	Voting $91.85 \pm 4.25$	<b>SVM-L</b> $89.96 \pm 6.11$	<b>SVM-L</b> $89.78 \pm 5.76$	<b>SVM-L</b> $89.78 \pm 5.76$
$\chi^2$	<b>SVM-L</b> $80.29 \pm 3.58$	<b>SVM-L</b> $90.54 \pm 3.99$	Voting $90.15 \pm 5.08$	<b>SVM-L</b> $89.96 \pm 5.37$	<b>SVM-L</b> $89.96 \pm 5.37$
fisher	<b>SVM-L</b> $78.4 \pm 1.64$	NN $89.58 \pm 5$	Voting $89.57 \pm 6.5$	Voting $89.58 \pm 5.76$	Voting $89.58 \pm 5.76$
GINI	Voting $77.82 \pm 2.85$	NN $89.39 \pm 4.64$	Voting $89.58 \pm 6.32$	<b>SVM-L</b> $89.01 \pm 4.05$	<b>SVM-L</b> $89.01 \pm 4.05$

## 2.8.2 Task 2 - Longitudinal versus transverse cross-sectional views of dendrites

Similar performance comparisons presented for Task 1 in Section 2.8.1 were also completed for Task 2 (longitudinal vs. transverse cross-sectional views of dendrites). A similar result was observed when feature extraction methods were evaluated, where caffe-fc6 represented micrographs well.

Again, features extracted using pre-trained neural networks dominates Table 2.4. This result shows that the features extracted from the sixth, fully connected layer in the Caffe network (caffe-fc6) have meaningful and distinguishable characteristics in terms of microstructural images. Although the majority of the maximum classification accuracies used features extracted via caffe-fc6, even higher values for classification accuracy were obtained using texture statistics and caffe-conv5 with a linear SVM classifier. Classification accuracies obtained via SVM-L varied based on the feature selection method used, however the overall classification accuracies were highest for this classifier with features computed from either

texture statistics or caffe-conv5. One major disadvantage of using texture statistics for feature extraction is that this method is a combination of multiple different feature detectors, and required prior knowledge of various techniques, that all perform differently depending on shape and texture of input data. The success of SVM as a classifier is subsequently discussed in additional detail.

**Table 2.4. Feature extraction methods and corresponding classification accuracies (in %), that contributed to the maximum classification accuracy for each combination of feature selection and classification method tested for Task 2.**

Feature Selection	Classifier				
	SVM-L	SVM-RBF	NN	RF	Voting
<b>Full</b>	texture-statistics 95.79 ± 3.94			<b>caffe-fc6</b> 87.88 ± 1.64 VBOW 87.88 ± 2.95	
	texture-statistics 96.84 ± 3.07				
<b>PCA</b>	texture-statistics 97.37 ± 3.33			<b>caffe-fc6</b> 87.13 ± 1.64 caffe-conv5 87.49 ± 15.69	
	texture-statistics 96.78 ± 2.63				
<b>ANOVA</b>	texture-statistics 97.84 ± 2.65	<b>caffe-fc6:</b> 95.74 ± 3.73	<b>caffe-fc6:</b> 94.49 ± 1.46	texture-statistics 87.5 ± 8.25	
	texture-statistics 97.31 ± 2.42				
<b><math>\chi^2</math></b>					
<b>fisher</b>	texture-statistics 97.84 ± 2.65			texture-statistics 87.5 ± 8.25	
	texture-statistics 97.31 ± 2.42				
<b>GINI</b>					

Results from comparing feature extraction and classifier combinations indicate that in many cases feature selection did not improve calculated classification accuracies, as was seen in Task 1 results. This is shown in Table 2.5 by the majority of maximum classification accuracies computed using the full feature vector. When SVM-L was used for classification, feature selection methods improved results. The feature selection method used with SVM-L that yielded the maximum classification accuracy depended on the feature representation obtained in the feature extraction step. The same classification accuracies were obtained for each feature selection method, for a given feature extraction and classifier combination. In addition, the features extracted from the fifth layer (convolutional) in the pre-trained neural network perform poorly with respect to the other extraction methodologies. This can be attributed to the fact that caffe-conv5 produces the feature representation with the highest

dimensionality which is an order of magnitude greater than the second largest feature vector (caffe-fc6). Therefore, features extracted using caffe-conv5 are not robust (i.e. they have low accuracy with high standard deviation). Being lower in the network hierarchy of the CNN also means that the features may not be tuned properly to the task, and therefore features provide a less accurate image representation than features computed via caffe-fc6.

**Table 2.5. Feature selection methods, and corresponding classification accuracies (in %), that contributed to the maximum classification accuracy for each combination of feature extraction and classification method tested for Task 2.**

<b>Feature Extraction</b>	<b>Classifier</b>				
	<b>SVM-L</b>	<b>SVM-RBF</b>	<b>NN</b>	<b>RF</b>	<b>Voting</b>
<b>VBOW</b>	ANOVA $96.32 \pm 2.68$	<b>Full</b> $93.73 \pm 3.96$	<b>Full</b> $92.73 \pm 3.91$	PCA $87.88 \pm 2.95$	<b>Full</b> $93.98 \pm 3.09$
<b>caffe-fc6</b>	GINI $96.78 \pm 2.63$	<b>Full</b> $95.74 \pm 3.73$	<b>Full</b> $94.49 \pm 1.46$	<b>Full</b> $87.88 \pm 1.64$	<b>Full</b> $94.99 \pm 2.38$
<b>caffe-conv5</b>	fisher $97.84 \pm 2.65$	<b>Full</b> $84.46 \pm 13.6$	<b>Full</b> $77.19 \pm 14.62$	$\chi^2$ $87.49 \pm 15.69$	<b>Full</b> $78.45 \pm 16.66$
<b>texture-statistics</b>	ANOVA $97.37 \pm 3.33$	<b>Full</b> $93.23 \pm 11.58$	<b>Full</b> $94.24 \pm 5.03$	fisher $87.5 \pm 8.25$	<b>Full</b> $93.73 \pm 11.09$
<b>combined</b>	ANOVA $97.37 \pm 3.33$	<b>Full</b> $89.97 \pm 9.08$	<b>Full</b> $87.97 \pm 9.02$	GINI $86.74 \pm 2.98$	<b>Full</b> $84.46 \pm 14.08$

Lastly, classification models were evaluated for each combination of feature extraction and feature selection with results presented in Table 2.6. SVM-L provided maximum accuracy with feature vectors calculated from the majority of feature extraction and selection methods. Linear SVMs again yield the majority of maximum classification accuracies in this table (similar to Table 2.1, which shows that this classifier has the most distinguishable capacity among the different models tested).

**Table 2.6.** Classification models that yielded the maximum classification accuracy (in %) for each combination of feature extraction and feature selection method tested for Task 2.

		Feature Extraction				
Feature Selection	VBOW	caffe-fc6	caffe-conv5	texture-statistics	combined	
Full	<b>SVM-L</b> 95.26 ± 3.07	SVM-RBF 95.74 ± 3.73	<b>SVM-L</b> 88.57 ± 12.67	<b>SVM-L</b> 95.79 ± 3.94	<b>SVM-L</b> 95.79 ± 3.94	
PCA	<b>SVM-L</b> 94.21 ± 3.07	<b>SVM-L</b> 96.29 ± 3.56	<b>SVM-L</b> 96.29 ± 2.09	<b>SVM-L</b> 96.84 ± 3.07	<b>SVM-L</b> 96.84 ± 3.07	
ANOVA	<b>SVM-L</b> 96.32 ± 2.68	<b>SVM-L</b> 96.26 ± 3.19	<b>SVM-L</b> 91.9 ± 6.58	<b>SVM-L</b> 97.37 ± 3.33	<b>SVM-L</b> 97.37 ± 3.33	
$\chi^2$	<b>SVM-L</b> 94.74 ± 3.33	<b>SVM-L</b> 96.26 ± 3.19	<b>SVM-L</b> 96.78 ± 2.63	<b>SVM-L</b> 96.32 ± 3.16	<b>SVM-L</b> 96.32 ± 3.16	
fisher	<b>SVM-L</b> 94.15 ± 1.03	SVM-RBF 95.74 ± 3.73	<b>SVM-L</b> 97.84 ± 2.65	<b>SVM-L</b> 96.29 ± 2.67	<b>SVM-L</b> 96.29 ± 2.67	
GINI	Voting 93.98 ± 3.09	<b>SVM-L</b> 96.78 ± 2.63	<b>SVM-L</b> 97.31 ± 2.42	<b>SVM-L</b> 96.84 ± 3.07	<b>SVM-L</b> 96.84 ± 3.07	

Results presented here demonstrate that certain combinations of methods (feature extraction, selection and classification) yield higher classification accuracies than others. The sixth fully connected layer in the Caffe network proved to represent micrograph data well for both tasks. The feature selection method for Task 1 depended on which classification model was used. For Task 2, the full feature vector yielded the maximum classification accuracies, thus feature selection did not appear to improve classification results for most classifiers tested. For classification models tested, linear SVM yielded the best results (highest accuracy) for both Tasks 1 and 2.

## 2.9 Limitations

In this work, pre-trained CNN's were used [20]. Training a convolutional neural network on the small amount of image data provided in Data Sets 1 and 2 (528 and 188 images, respectively) could prove difficult and potentially provide poor results in comparison to the

results reported here. The reason for this is that deep neural networks require large image data sets for training because of their capacity to learn complex functions and are therefore prone to overfitting. The CNN used in this work was previously trained on millions of natural images, allowing for the model to generalize well to new data instances. Even though the original dataset that the neural network was trained on primarily consisted of natural images, we can see (from high classification accuracies) that the features extracted on micrographs provide meaningful representations.

There are an infinite number of potential material microstructures in the realm of materials science. Micrographs are two-dimensional images that represent a small sample of a three-dimensional microstructure, where ‘microstructure’ refers to material structure on a nanometer to centimeter length scale. Microstructures are formed from thermodynamic and kinetic processes, and depend on the crystal structure (atomic arrangement) of constituent elements. Although the elements on the periodic table are fixed, there are numerous combinations of these elements that can be processed and studied.

Let’s first consider a material with a specific chemistry. There are various processes (annealing, quenching, cold working, etc.) that could yield a variety of microstructures. Additionally, the processed sample could be sectioned in different ways (transverse versus longitudinal, for example), again impacting the microstructure visible for subsequent analysis. There are also multiple imaging techniques available over a broad range of length scales (nanometer to centimeter). Images generated from different techniques have different contrast, brightness, and ranges of grayscale intensities or RGB levels. Expand this thought process to consider all possible material systems that can be generated from known elements with numerous imaging techniques (light optical microscopy, scanning electron microscopy, transmission electron microscopy, etc.). The possible microstructures we can analyze in the form of image data seems endless.

Therefore, the variety in micrographs used for materials characterization is vast, which is why expert knowledge and experience is relied on for such tasks as recognition, interpretation, and characterization. Training an algorithm to perform such classification/characterization tasks on diverse micrograph data sets is a challenge not only in organizing a data set for training, validation, and testing, but in obtaining reasonably good classification results (greater than random). The limitation presented here, is in the inherent variability of micrograph data, which could make classification tasks challenging, particularly if only a

small data set is available for training and testing.

## 2.10 Conclusions

Computer vision and machine learning methods were successfully applied to the challenge of recognizing specific microstructural features of interest (e.g. dendrites) with varying magnifications, chemistries, and orientations. Two binary classification tasks were performed: classification between micrographs with and without dendrites (Task 1), and classification between longitudinal and transverse cross-sectional views of dendrites (Task 2). Data sets for Task 1 and Task 2 were 528 and 188 total images respectively. We are able to distinguish between microstructural images in terms of the presence of dendrites. A further refinement of the images can also be performed amongst the dendritic micrographs with respect to their cross-section, as shown in Figure 2.3.

Feature extraction and dimensionality reduction techniques were utilized in order to represent micrographs in the form of feature vectors. Classification was then performed using support vector machines (linear and non-linear), voting, nearest neighbor, and random forest models. For each model, classification accuracy was calculated for full and reduced feature vectors, for each feature extraction and selection method tested.

Results demonstrated that pre-trained neural networks represented micrographs well, and required no previous knowledge of the nature of shapes or object within the images. Furthermore, when pre-trained neural networks were used in feature extraction the highest classification accuracies for the majority of classifier and feature selection methods tested were achieved. Thus pre-trained neural networks generalize well. Maximum classification accuracies of  $91.85 \pm 4.25\%$  and  $97.37 \pm 3.33\%$  were obtained for Tasks 1 and 2 respectively. While computer vision and machine learning methods have previously been applied to microstructure image data [?], [5], [39]–[41], pre-trained deep neural networks have not yet been explored for this particular application, but have proven successful in other image recognition tasks [42].

Recognition of microstructural features traditionally requires expert knowledge of the particular material system under consideration. The image-driven machine learning approach to micrograph classification presented here challenges this paradigm.

## 2.11 Future Work

Neural networks are a promising feature extraction method for micrograph representation that could be further developed for application in more sophisticated characterization techniques, thereby eliminating the requirement of expert knowledge for recognition, interpretation, and characterization of microstructural image data. Future work related to the exploratory study presented here, could involve application of pre-trained CNN's to more diverse micrograph data sets, or higher-level characterization tasks, such as average grain size measurement or calculating area fraction of a second phase. Furthermore, we would also like to incorporate deep learning algorithms on larger and more diverse micrograph classification tasks.

## Acknowledgments

This work was supported by the NSF under Grant #1056704 through the Metals and Metallic Nanostructures Program of the Division of Materials Research, and Grant #1302231 through the Information Integration and Informatics Program of the Division of Information and Intelligent Systems. The authors wish to acknowledge Mr. Jesse Werden and Dr. Jie Mao for some of the image data used in this work. Additionally, the authors thank the reviewers for providing thoughtful and thorough comments that contributed to the quality and clarity of the manuscript.

# CHAPTER 3

## BLOOD VESSEL CHARACTERIZATION USING VIRTUAL 3D MODELS AND CONVOLUTIONAL NEURAL NETWORKS IN FLUORESCENCE MICROSCOPY

### 3.1 Introduction

Characterizing the morphology of vasculature in digital pathology is an important step in defining the microenvironment within brain tissue samples. In particular, understanding the geometry of vessel configuration and its changes during a disease may provide deeper insight into the progression of neuropathological degenerative diseases such as Alzheimer's disease. Images acquired using 20x objective from immunofluorescent (IF) stained 6 um tissue sections with collagen IV antibody are used in this work. We attempt to characterize three different types of blood vessel morphologies which are found in relative abundance in our image data set. They are singular blood vessels with no visible lumen, singular blood vessels with a distinct lumen and blood vessels appearing as a pair; which we have named RoundLumen-, RoundLumen+, and Twins correspondingly. In this work, we show that it is possible to characterize blood vessels using convolutional neural networks (CNN) as opposed to traditional image processing techniques which involve segmentation and hand-crafted feature extraction. Instead, here we use pre-trained CNN to extract features from the images. This technique of "deep transfer learning" is compared to the visual bag of words (VBW) method for feature extraction [43]. We conclude from the results that the features from pre-trained CNN are able to distinguish the morphologies of blood vessels better than the standard VBW method. Additionally, our work also shows that the construction of 3 dimensional (3D) virtual models of vasculature using parametric methods is a promising tool for dealing with infrequent scenarios in vascular analysis of brain tissue section. Acquisition of natural training samples is a time consuming and labor intensive process. Deep learning requires abundant training data for tuning the large number of parameters of the various

---

This chapter previously appeared as: A. Chowdhury, D. V. Dylov., Q. Li, M. MacDonald, D. E. Meyer, M. Marino and A. Santamaria-Pang, "Blood vessel characterization using virtual 3D models and convolutional neural networks in fluorescence microscopy" *IEEE International Symposium on Biomedical Imaging*, pp. 629-632, 2017

inherent models. If a certain class is imbalanced then the classification models could become prone to biased outcomes. The construction of 3D parametric models, presented here tackles these issues and creates a balanced high-fidelity classification model. In this study, we built a basic 3D vasculature model using our prior knowledge of blood vessel geometry, as guided by a pathologist. The 3D vasculature was repeatedly sliced at various angles and orientations to obtain 2D samples for training the machine learning model, thereby mimicking the physical sectioning of tissue during sample preparation for microscopy. In addition, a filtering technique was then used to fine-tune the virtual data to reflect the variability present in the naturally acquired samples. We train three models based on: virtual data, natural data and a mixture of both. The models are then tested on a reserved, independent portion of the naturally occurring data, with a hierarchical classification being performed to demonstrate a proof of concept. The first classification task involves distinguishing between singular blood vessels (RoundLumen) and pair of blood vessels (Twins). The second task of finer granularity is the classification between RoundLumen- and RoundLumen+. We report various metrics for both the classification tasks and observe that the artificial data improves upon the model trained from only the natural data. As far as we know, this is the first attempt to model vasculature using parametric 3D geometric methods exclusively. Statistical 2D and 3D shape models have been used extensively in medical image segmentation as detailed in [44]. However, these models use the training samples to statistically find a model of the object of interest. We do not generate virtual 2D models because we believe that we can obtain more variability in the training data by sectioning from the 3D models. Additionally, the 3D models may be extended to various other modalities in medical imaging.

### 3.2 Data

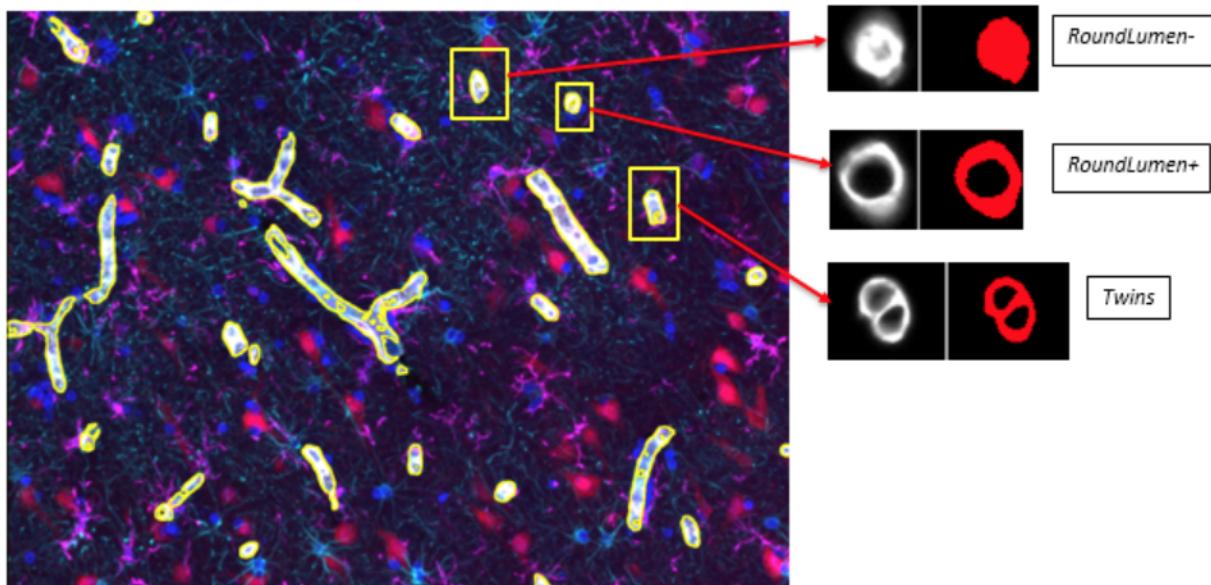
This section provides a detailed explanation of the different morphologies of blood vessels that are explored in this study. The first subsection is a description of the natural data curated from actual postmortem human tissue samples. The second subsection involves the description of the virtual 3D model of blood vessels which are used for generating samples for training the convolutional neural networks.

**Table 3.1. Distribution of vascular morphologies**

<i>RoundLumen-</i>	689
<i>Roundlumen+</i>	3427
<i>Twins</i>	266
Total	4382

### 3.2.1 Natural data

FFPE Postmortem brain tissue samples from ten subjects with neurological disorders underwent sequential IF-multiplexing and fluorescent imaging. For each subject, approximately 25 images were acquired. This involves a cycling process of tissue section staining with 2-3 dye-labeled antibodies, imaging, dye inactivation and repeated staining with a new set of dye-labeled antibodies. Images underwent illumination correction, registration, stitching and auto-fluorescence subtraction. Collagen IV was used as a marker to detect all blood vessels. An image overlay is shown in Fig. ??.



**Fig. 3.1.** Depiction of the different morphologies in the natural data with respect to a multichannel image, overlaid with different protein markers. The three types of morphologies analyzed in this study is represented on the right.

The number of instances of the three different types of morphologies are depicted in Table ??.

### 3.2.2 Virtual data

The construction of the artificial model starts with defining a set of control points in three dimensional Cartesian coordinates. The control points reflect the basic structure that the blood vessel is supposed to represent. This is followed by interpolating between the points using a 3D cubic spline interpolator. This forms the skeleton or the center line that represents the center or the lumen of the blood vessel and is shown in Fig. 2(a). The 3D volume of the blood vessel is constructed after this step. We first define a number of parameters; the inner radius of the blood vessel ( $r$ ); the outer radius ( $R$ ). The number of sampling points along the spline ( $N$ ); the number of sampling points in the radial direction ( $N_r$ ). At each sampling point; we define a circular disk along the z-axis; by randomly perturbing the values of  $r$  and  $R$ . We also define an intensity model for the blood vessels depicted in Eq. 4.1. From the natural images, it seems that the intensity is high in the periphery of the blood vessel and decays towards the lumen and as we move away from the periphery. We model this using an exponential decay in the following form:

$$I(d) = I_{max} \exp(-\alpha|r' - d|) \quad (3.1)$$

where,  $I_{max}$  is the maximum intensity,  $\alpha$  is the calibration coefficient (in mm-1 units),  $r' = (R + r)/2$ .  $d$  is the distance from the center of the lumen. At each point on the disc, we define the voxel density as a normal distribution with mean  $I(d)$  and standard deviation 0.01. This is followed by formulating the rotation matrix by calculating the angle between the tangent to the spline at that sampling point and the z-axis. The points corresponding to each point on the disc are therefore mapped or rotated along the curve by multiplying the coordinates with the rotation matrix. An example of this rotation is depicted in Fig. 3.2(b). We then discretize the coordinates such that we obtain an actual 3D image in the form of an array. This is depicted in Fig. 3.2(c). The intensity values are normalized and assigned to the corresponding discretized points in the 3-dimensional array. The volume rendered version of the 3D image is depicted in Fig. 3.2(d). Therefore, by changing the parameters of the model we can build several different 3D images and slice it at various angles to mimic the natural tissue cross sections at various depths and angles. Some examples the various models are depicted in Fig. 3.3.

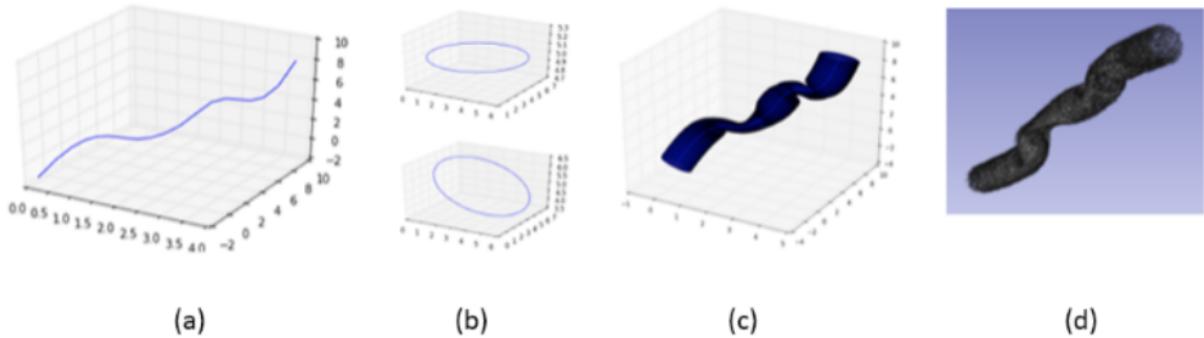


Fig. 3.2. Development of the 3D virtual model

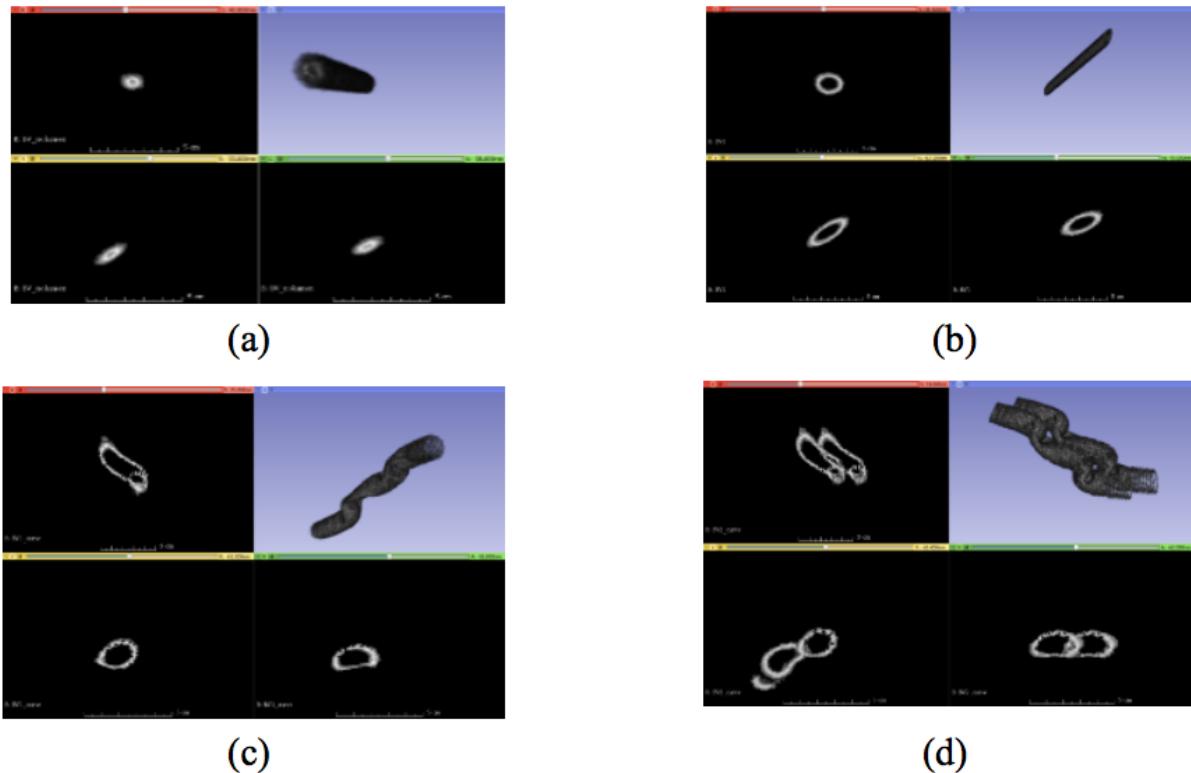


Fig. 3.3. 3D virtual models and their corresponding projections along different planes of view (a) Linear model of RoundLumen- (b) Linear model of RoundLumen+ (c) Non-linear model of RoundLumen+ (d) Non-linear model of Twins

The process of construction of the different types of morphologies in blood vessels is the same as that explained above. Fig. 3.3(a) is a blood vessel with a no lumen (RoundLumen-)

and a linear skeleton. The control points are chosen such that they lie on the major diagonal of the unit cube; i.e. on the line  $x=y=z$  in Cartesian coordinates. Figures 3.3(b) and 3.3(c) are blood vessel with a single lumen having linear and non-linear structures respectively. Fig. 3.3(d) is a model of a Twin. As we can see from the cross-sectional views, we obtain different types of morphologies that are similar to actual morphologies in the natural images. A very simple way of creating these multi-vessel structures is by perturbing or shifting the sampling points of the skeleton along a random direction. For example; to come up with a model of RoundLumen-, we simply set the inner radius  $r$  to 0. As shown in Fig. 3; the different morphologies that commonly occur naturally can be generated. As explained in the introduction; this serves as a viable alternative to using natural data for training convolutional networks.

### 3.3 Methods

A convolutional neural network is a type of artificial neural network in which are inspired by the organization of the animal visual cortex. CNNs consist of multiple neuron collections which process portions of the input image called receptive fields. The outputs are then tiled so that the input regions overlap and this in turn produces a better representation of the original image. This is what makes CNNs translation invariant. The CNN is made up of four types of layers: the input layer, the convolutional layer, the non-linear layer, the pooling layer; and the fully connected layer. The input layer is where the networks accept the images. The images consist of raw pixel values depicted by width, height and the number of channels. The convolutional layer will compute the output of the neurons that are connected to local regions in the input, each computing a dot product between their weights and a receptive field. The non-linear layer is the activation function responsible for introducing the non-linearity in the model. The various types of non-linear functions include the sigmoid, the tanh, and the rectified linear unit. The pooling layer performs a down sampling operation. The high-level reasoning in the neural network is done by fully connected layers. Their activations can be performed by simple matrix multiplication. In this work, we use the pre-trained convolutional neural networks as a feature extractor. This network consists of weights trained on the ImageNet dataset. We extract the 6th layer in the network which is a 4096-dimensional vector as a representation of the image. This may be considered as a transfer learning model because we transfer the weights learnt from another

domain to blood vessel recognition. We use a pre-trained neural network called AlexNet [45] to extract features from the data.

We perform an experiment to show that pre-trained CNNs are efficient in representing the vascular morphology. The experiment is performed on the natural data where 33 % of the data is held out as test data and the rest is used for training. Two models are developed. One of them uses the visual bag of words (VBW) [43] feature extraction method to extract the features; the other uses the AlexNet architecture to extract the features. A three-class classification (one vs rest) is performed using the logistic regression classifier. The accuracy, f1-score, precision and recall calculated on the same test data are reported for comparison.

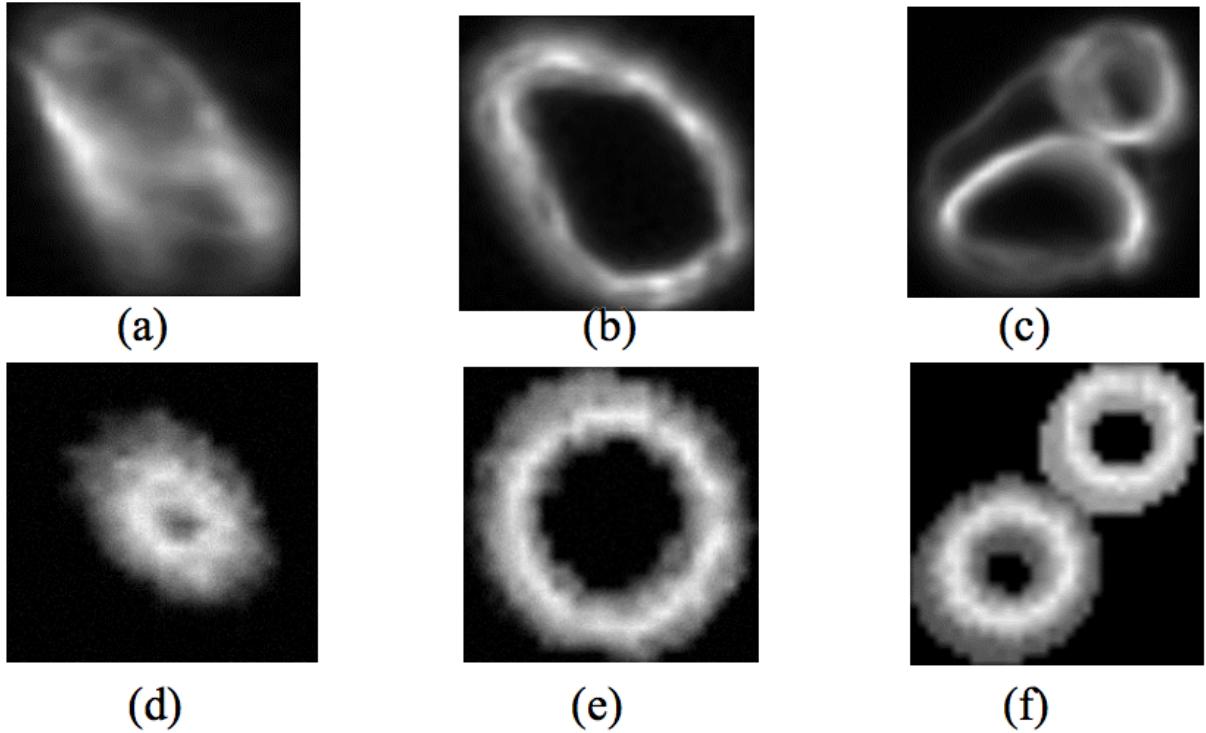
**Table 3.2. Comparison of feature extraction methodologies**

Feature extractor	Accuracy	f1-score	Precision	Recall
<i>AlexNet</i>	91.92	91.93	91.98	91.92
<i>VBW</i>	78.38	77.38	76.71	78.38

The results in Table 3.2 show that pre-trained convolutional neural networks are a good choice for representation of vascular morphology.

### 3.4 Experiments and results

Features are extracted using the AlexNet architecture which is trained on the ImageNet [46] database. The weight parameters are used to extract the features in a feedforward manner. This is called transfer learning. 33 % of the natural data is held out as test data. All the experiments are performed on this dataset for maintaining consistency in the results. A filtering technique is introduced to appropriately extract slices from the 3D volumes. This is done by obtaining the probabilities of the artificial data using a model trained on the natural training data. The probabilities of the corresponding images are then sorted and the images with the highest probabilities are selected. This is a way to boost the artificial model. The filtered virtual data is then used to retrain the classifier. Examples of both the natural and artificial data are represented in the Fig. 3.4.

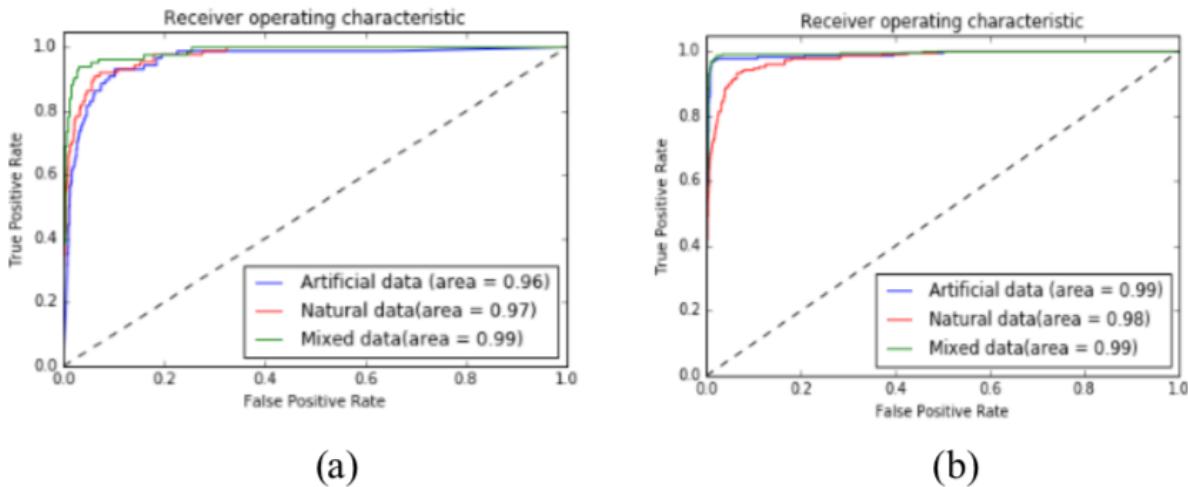


**Fig. 3.4.** Examples of vessel classes RoundLumen- (a/d), RoundLumen+ (b/e) and Twins (c/f) for natural (a/b/c) and virtual data (d/e/f)

A hierarchical classification is performed to first classify the single blood vessels from blood vessels that bundled in pairs i.e RoundLumen vs Twins. The second classification task involves distinguishing between RoundLumen- and RoundLumen+. We perform three different types of training to demonstrate our proposed methods. The first type of training is done with only the naturally occurring data. The second type of training data consists only of the artificial data that has been filtered by the natural model as explained before. Finally, the third type consists of both the artificial and natural training samples. We refer to this as Mixed. All the results are reported on the held out 33 % of the natural data. The accuracy, f1-score, precision, recall and receiver operating characteristic (ROC), and precision-recall (PR) curves are reported in the following tables and figures for each of the two classification tasks.

**Table 3.3.** Results of binary classification between RoundLumen and Twin

Data	Accuracy	f1-score	Precision	Recall
<i>Artificial</i>	92.81	59.36	45.24	<b>86.36</b>
<i>Natural</i>	96.34	71.03	68.42	73.86
<i>Mixed</i>	<b>97.71</b>	<b>81.76</b>	<b>79.57</b>	84.01

**Fig. 3.5.** ROC curves for classification: (a) *RoundLumen* and *Twins* and (b) *RoundLumen-* and *RoundLumen+* .**Table 3.4.** Results of binary classification between *RoundLumen-* and *RoundLumen+*

Data	Accuracy	f1-score	Precision	Recall
<i>Artificial</i>	98.38	99.02	99.38	98.67
<i>Natural</i>	96.34	71.03	68.42	73.86
<i>Mixed</i>	<b>98.60</b>	<b>99.16</b>	<b>99.29</b>	<b>99.03</b>

The ROC curves are calculated using the minority class in both the classification tasks, i.e. Twins for the first classification task and RoundLumen- for the second task. From Table 2.8, we can see that the artificial data captures the differences between the two classes. It is also able to identify *Twins*, which is the minority class in Task 1, from the high recall.

Therefore, the results are boosted when we combine both the artificial and natural data. In addition, the ROC curves in Figure 6 confirms our hypothesis that virtual data may be used for building the models. The naturally trained model performs better than the virtual trained model. However, as we can see from the ROC curves, the model built from the mixed data improves the performance. Table ?? presents the corresponding results for the second classification task. In this case, we see that the virtual data performs better than the natural data and boosts the performance when trained on its own or in union with the natural data.

### 3.5 Conclusions

We have shown that the use of deep learning algorithms ? trained with a mixture of virtual and natural data ? results in a more accurate prediction of vasculature morphologies than the use of standard feature extraction methods (such as visual bag of words). The methodology of complementing natural data with synthetic samples holds potential for becoming a standard approach in deep learning with unbalanced datasets. In neuroscience, it could be applied to help elucidate the underlying mechanisms of common neurological degenerative diseases.

# CHAPTER 4

## A MACHINE LEARNING APPROACH TO QUANTIFYING NOISE IN MEDICAL IMAGES

### 4.1 Introduction

As advances in medical imaging technology are resulting in significant growth of biomedical image data, new techniques are needed to automate the process of identifying images of low quality. Automation is needed because it is very time consuming for a domain expert such as a medical practitioner or a biologist to manually separate good images from bad ones. While there are plenty of de-noising algorithms in the literature, their focus is on designing filters which are necessary but not sufficient for determining how useful an image is to a domain expert. Thus a computational tool is needed to assign a score to each image based on its perceived quality. In this paper, we introduce a machine learning-based score and call it the Quality of Image (QoI) score. The QoI score is computed by combining the confidence values of two popular classification techniques support vector machines (SVMs) and Nave Bayes classifiers. We test our technique on clinical image data obtained from cancerous tissue samples. We used 747 tissue samples that are stained by four different markers (abbreviated as CK15, pck26, E\_cad and Vimentin) leading to a total of 2,988 images. The results show that images can be classified as good (high QoI), bad (low QoI) or ugly (intermediate QoI) based on their QoI scores. Our automated labeling is in agreement with the domain experts with a bi-modal classification accuracy of 94 %, on average. Furthermore, ugly images can be recovered and forwarded for further post-processing.

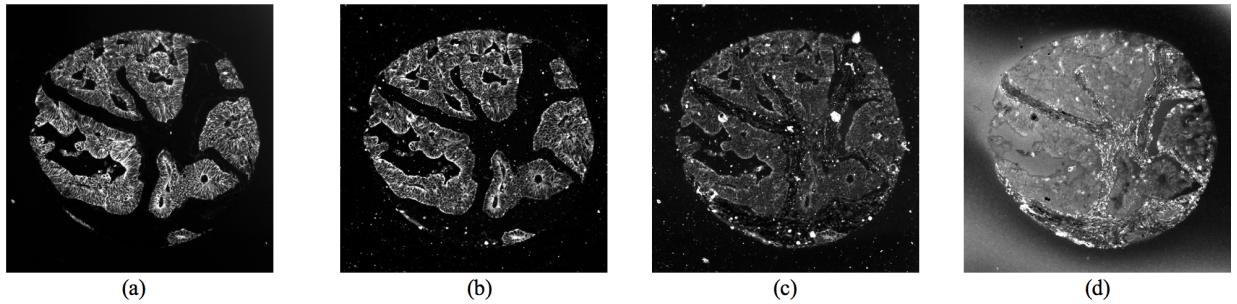
### 4.2 Data

The data we use is in the form of microscopic images. The colon cohort in this analysis was collected from the Clearview Cancer Institute of Huntsville Alabama from 1993 until 2002, with 747 patient tumor samples collected as formalin-fixed paraffin-embedded specimens. The median follow-up time of patients in this cohort is 4.1 years, with a maximum

---

This chapter previously appeared as: A. Chowdhury, K. S. Aggour, S. M. Gustafson, B. Yener, “A Machine Learning Approach to Quantifying Noise in Medical Images.” *Medical Imaging 2016: Digital Pathology*, vol. 9791, pp. 979110U, 2016.

of over ten years. Stage 2 patients comprise 38 % of this cohort, stage 1 and 2 combined are 65 % of the total patients. We have stained and processed 747 CRC subjects described above on tissue microarrays for 63 target proteins of consequence to cancer biology and ancillary image processing and analysis. A full description of materials and methods was described recently in [47]. The images are stained with four different markers. The markers are Ecadherin (E\_cad), pan-Keratin (pck26), Keratin15 (CK15) and Vimentin. The first three markers stain epithelial cells in the tissue, while Vimentin stains mesenchymal cells - a complementary set of cells in the image obtained from the same tissue sample, as shown in Fig. 1. The raw dataset consists of 747 tissue samples and each tissue sample has four images from each marker; resulting in a total of 2,988 images. Since images with the Vimentin marker provides complementary information to the other three markers; we do not use this marker for our analysis and prediction. We do, however, perform preliminary noise reduction on all four markers. The images marked with E\_cad have the least amount of noise associated with them. Images stained with pck26 and CK15 have more noise and undesirable artifacts in them.

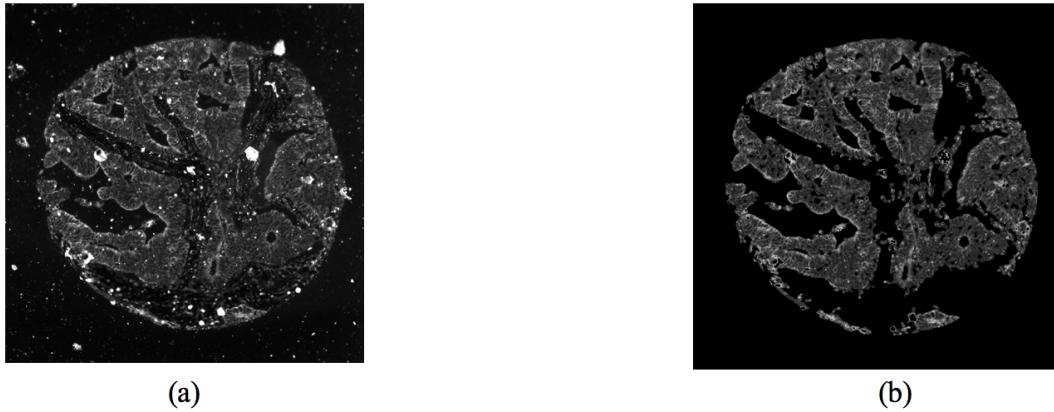


**Fig. 4.1. Examples of a single colon tumor tissue sample stained with (a) E\_cad (b) pck26 (c) CK15 (d) Vimentin**

#### 4.2.1 Preliminary noise reduction

We perform preliminary noise reduction on all four types of images using two techniques. We remove salt and pepper noise using median filtering [48] followed by removal of white blob-shaped artifacts using intensity thresholding. Fig. 4.2 represents an example of the noise reduction on an image from CK15. The QoI score (defined in the next section) increases by 67.85 % on average after noise removal. This shows that the QoI score provides a more accurate estimate after noise removal has been performed. Hence, this step is essential

in the process.



**Fig. 4.2. Preliminary noise reduction on an image from CK15 (a) Original image (b) Noise-reduced image**

#### 4.2.2 Combination of the three markers

In addition to the three markers (CK15, pck26, and E\_cad) that we are working with; we use image analysis techniques to form a superimposition of the three markers in such a way that the different types of cells that the markers stain are all present in this one super image. We call this the combined marker. Diffusion like interference exists in the noisy images. The combined image is a way to reduce this kind of noise. The QoI score (defined in the next section) of the combined image is on an average higher (10.76 %) than the QoI scores of images stained with the rest of the markers for a particular tissue sample. The superposition is performed after preliminary noise elimination. It is done by taking the minimum pixel intensity value among the three images in low intensity regions. We take the maximum intensity value in other regions. An example of this is shown in Fig. 4.3. This combination increases the size of the data set; which is inherently imbalanced. It also aids in the recovery of images from the *ugly* class based on the QoI results, as we shall see in the following section.



**Fig. 4.3. Images from (a) E-cad, (b) pck26 and (c) CK15 are used to form the (d) Combined image**

#### 4.2.3 Feature extraction

We quantify the information in the four markers (E-cad, pck26, CK15 and Combined) by creating 24 domain-specific features. Features are based on gray-level intensity values, and texture information [49], [50] in the images.

In addition to the features, labels are assigned to each image by domain experts based on the amount of noise in them. The label +1 is assigned if the image has some signal in it. This class is colloquially referred to as the *good* class. Images with a lot of noise in them where there is almost no signal are labelled as -1 or the *bad* class.

#### 4.2.4 Data preprocessing

We use 276x4 (1,104) images as the training set. The four markers (CK15, pck26, E-cad, and combined) are used corresponding to each tissue sample. The rest of the images (1,884) are used for testing. The test and training sets were selected randomly, with 10-fold cross validation. The data is normalized such that the mean of every feature is 0 and standard deviation is 1. Singular value decomposition (SVD) is performed to reduce the number of features from 24 such that it captures 95 % of the information. This reduces the number of dimensions to 12.

### 4.3 Methods and experiments

In this section, we describe the methods that we used in our analysis of the images and also show the results that we achieved from the application of these methods on our data.

### 4.3.1 Random undersampling

The data in the two classes is imbalanced because the number of images in the *good* class (images with high amount of signal) far outnumbers the number of instances in the *bad* or noisy class. The ratio of imbalance is approximately 1:30. We perform random undersampling to redress the imbalance and ensure that the sizes of the two classes (good and bad) are the same.

### 4.3.2 Classification and analysis

We use two popular classification techniques - Support Vector Machines (SVM) and Nave Bayes classifiers. The accuracy of the Nave Bayes classifier for bi-modal classification on the test set is 96.38 % and that of SVM is 91.27 %.

### 4.3.3 Quality of image score

The models returned by the two classifiers are used to classify the test data as well as assign a score to each test image. The SVM classifier returns a margin value corresponding to each test point. This value is a representation of the distance of the instance from the classifier margin. The margin values are calibrated to fit a probability distribution using [51]. The Nave Bayes classifier also returns a posterior probability that represents the membership of that point to that class. We combine these two values to form a score that is an indication of the quality of the image. We call this score the *QoI score*. A high QoI score represents a high value of the signal and a small score value indicates that noise predominates the image. For image  $i$ , let  $p_{i1}$  and  $p_{i2}$  be the probabilities returned by SVM and Nave Bayes classifiers respectively. Let  $l_{i1}$  and  $l_{i2}$  be corresponding labels returned the classifier models. The QoI score  $S$  is calculated as;

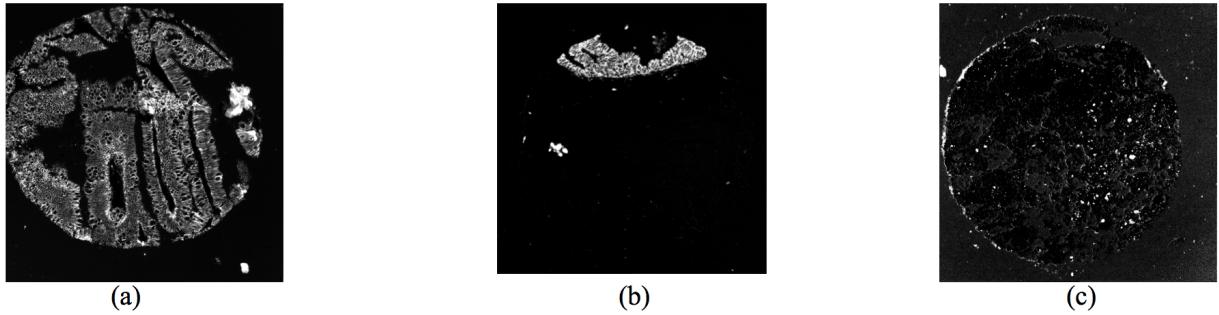
$$\begin{aligned} S &= \sqrt{p_{i1}p_{i2}} \text{ if } l_{i1} = l_{i2} \\ &= 0.5 \text{ if } l_{i1} \neq l_{i2} \end{aligned} \tag{4.1}$$

We take the geometric mean of the two scores that we get from each of the two classifiers, if the labelings are in agreement. A score of 0.5 is assigned if they are not. We tested other linear and non-linear functions as well. However, results were mostly invariant for these functions and we observed that the geometric mean was a better representation of the image quality. It is also intuitive that an image should be given the score 0.5 if the two

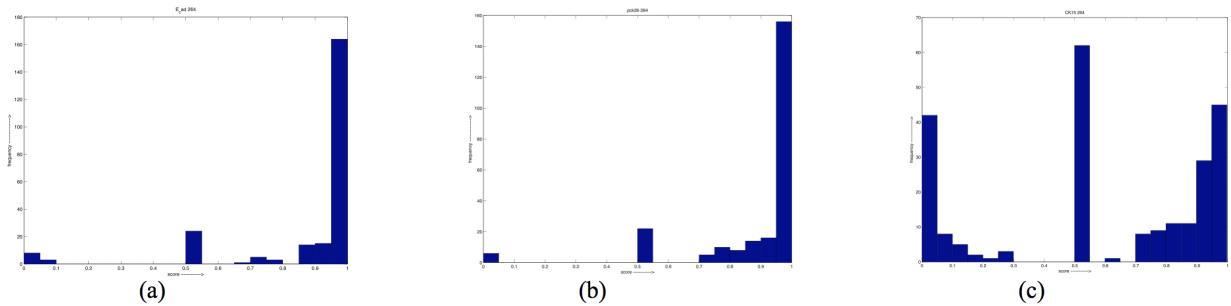
classifiers do not predict the same thing. The scores of the test images are normalized such that they vary from 0 to 1; and they form a discrete distribution. The three sample images in Fig.5, (a), (b) and (c) received QoI scores of 0.9995, 0.5001, and 1.7595e-11 respectively using Eq. 4.1.

#### 4.3.4 Defining the *ugly* class

In addition to the good and bad class, we observe that there exist some images, which have both quantities of signal and noise in relatively equal proportion. Classifying this kind of image as good or bad becomes very subjective. Therefore, it seems intuitive to define a third class, referred to as the ugly class. We define and extract this class based on the QoI scores in the following manner. We plot histograms of these discrete distributions as shown in Fig. 4.5. We can clearly see from the distributions that the scores form three distinct clusters. Therefore, we performed K-means clustering on the normalized score values with K=3. The cluster of images corresponding to the scores in the middle range is termed the ugly class. Images in the ugly class require further (possibly manual) processing. Thus, we are most interested in the images that are near the middle range, the images that we can recover. We threshold out the ugly images with the help of the middle cluster in K-means and using a threshold for scores. The thresholds are taken to be between 0.3 to 0.6. We have found that 7.22 %, 9.77 % and 29.30 % of images in E\_cad, pck26 and CK15 respectively, lie in this middle cluster. The results are intuitive because the amount of noise increases progressively from E\_cad to CK15 to pck26. A number of things may be done with the images that belong to the ugly class. They may be forwarded to medical practitioners for inspection. Further noise elimination may also be performed, or as we have done, the ?combined? image could be used as a proxy for the images belonging to this ugly class. This is because the ?combined? image best captures the important information in the images stained with the three different markers.



**Fig. 4.4.** Three sample images from E\_cad stained tissues



**Fig. 4.5.** Discrete distributions of QoI scores for (a) E\_cad (b) pck26 (c) CK15

#### 4.4 Conclusions

We introduce in this paper a novel way to combine SVM and Nave Bayes classifiers to form an image quality score. The score is based on the confidence of a data point with respect to the classifier margin; and the distribution of the data captured by the Nave Bayes classifier. Images that have high or low information content are easy to label as good or bad respectively. A labeling of the ugly class becomes dependent on the observer. Therefore computing a score is a more natural way of extracting this third class of images. The score helps us retrieve images that may have been discarded by a simple binary classification; or may have gone unnoticed by a human eye. Also, we use image analysis techniques to form a combined image that tries to capture the signal from all three markers.

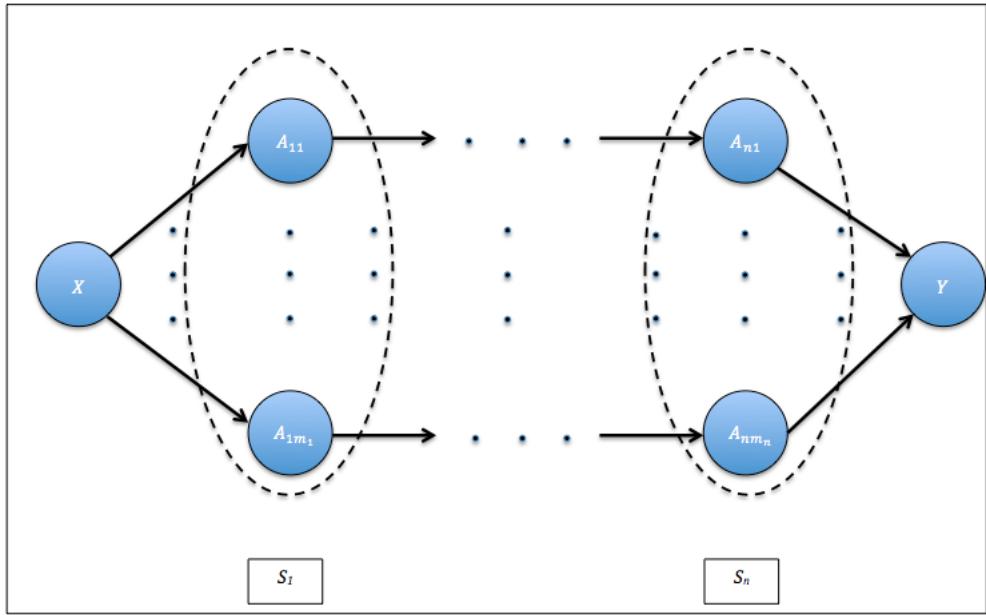
# **CHAPTER 5**

## **RANDOM SEARCH BASED QUANTIFICATION OF CONTRIBUTION AND PROPAGATION OF ERROR IN IMAGE CLASSIFICATION PIPELINES**

### **5.1 Introduction and Motivation**

Machine learning and data science has entered many domains of human effort in modern times. The number of self-reported data scientists has doubled in recent years [52]. They have entered various domains including academia, industry and business among others. There has therefore been a demand for machine learning tools that are flexible, powerful and most importantly interpretable. The effective application of machine learning tools unfortunately requires an expert understanding of the frameworks and algorithms that are present in a machine learning pipeline. It also requires knowledge of the problem domain and understanding of the assumptions used in the analysis. In order for these tools to be used adequately by non-experts; additional tools must be developed for understanding and interpreting the results of the application of a data analytic pipeline on a domain specific problem.

---



**Fig. 5.1. Representation of a machine learning pipeline**

Pipelines in machine learning and data science are commonly organized in the form of interdependent components. Such components that make up a data analytic pipeline include data preprocessing, feature extraction, feature transformation like dimensionality reduction, model building and model evaluation among others. Such pipelines provide a natural way to organize such tasks, and they play a key part in the design and implementation of large scale data science projects. Machine learning toolboxes like scikit-learn [53], RapidMiner [54] and Apache Spark [55] independently provide frameworks for implementing pipelines. Fig. 5.1 shows a generic representation of data analytic pipelines. Each computational step of the pipeline  $S_i$  consists of several algorithms ( $A_{ij}$ ) to choose from. Each algorithm in the pipeline consists of its own hyperparameters  $\theta_{ij}$  that must be optimized for using the algorithm. Therefore, there are an exponential number of combinations of algorithms and hyperparameters in a given analytic pipeline skeleton. This then is an extremely computationally intensive task of optimizing the pipeline. Tuning this pipeline can be viewed as an optimization of a blackbox objective function that is noisy and expensive to evaluate. The input to the blackbox are the input dataset ( $X$ ) the algorithms ( $A_{ij}$ ) and corresponding hyper-parameters ( $\theta_{ij}$ ) and the output performance ( $Y$ ) such as validation error, accuracy, F1-score , which are examples of the objective function. The goal of a data scientist is to find the best set of algorithms and hyper-parameters in this pipeline that optimizes the objective function. This corresponds

to finding an optimal path through the pipeline in 5.1. Simple methods such as grid and random search [56] have been used to tackle this problem. More complicated approaches such as Bayesian optimization [57] have been used successfully for approaching more difficult problems. We use the aforementioned methods for optimization of the pipeline.

Interpretation of machine learning pipelines is extremely important for their adoption in various domains. Domain experts do not prefer the use of black-box pipelines. They would prefer to understand how predictive decisions are made by the pipeline. Recently there has been an advent of models and techniques for improving the interpretability of machine learning. [58] introduces a model-agnostic method for interpreting machine learning blackboxes. [59] attempts at a definition of interpretability in this context and how it should be measured. In this work, we attempt to provide an interpretation of machine learning pipelines as a whole as opposed to the approaches which are geared toward interpretation of individual machine learning algorithms in general. To our knowledge, this type of approach to interpretation has not been done before. To this end, we propose the understanding of the contribution and flow of error in data analytic pipelines. We use the validation error as the performance metric of the optimization algorithms and basis of error quantification in the image classification pipelines. Understanding the source of the error in the predictive model is important for experts to design the data analytic pipelines. In addition, it also provides non-experts in machine learning insight into the predictions of the model. We also introduce a methodology to quantify the contribution of error from different components of the data analytic pipeline, namely the computational steps and algorithms in the pipeline. In addition, a model of error propagation is proposed in this work. This model represents the propagation of error from the computational steps in the pipeline starting from the first step that could be data preprocessing in Fig. 1 to the last step that is usually learning algorithms in a machine learning problem.

Pipeline optimization algorithms like grid search, random search [56] and Bayesian optimization [57] are used to optimize the pipeline for performing experiments with the error quantification and propagation methodology. We take two different approaches to optimization. The first is hyper-parameter optimization (HPO) where each and every computational path in 5.1 is optimized. This is equivalent to a combined optimization of the hyperparameters of the algorithms on that path. The second type of optimization is denoted as combined algorithm selection and hyperparameter optimization (CASH). This is a more difficult prob-

lem, because pipeline is optimized globally, in that the result of the optimization is a single optimized path that produces the best performance over all the paths in the machine learning workflow.

We use four datasets to demonstrate the error quantification and propagation methodology. The machine learning problem is that of image classification. We show the performance of both the optimization frameworks (HPO and CASH) for the experiments. We show experimentally that CASH using random search can be efficiently used for quantification of errors from the different stages of the pipeline. In addition, HPO frameworks of both Bayesian optimization and random search provides accurate estimates of error contributions from the steps and algorithms in the pipeline. We use the computationally efficient CASH framework of random search to perform experiments on the error propagation model. We demonstrate from the results that the error quantification methodology and the error propagation models maybe used by both data science and domain experts to interpret the results of machine learning pipelines. The paper is organized as follows. Section 5.2 describes the methods that are used in this work and the proposed methodology of error quantification and model for error propagation. The experimental frameworks, datasets, results and discussion makes up section 5.3. This is followed by the conclusion in section 5.4.

## 5.2 Foundations

In this section we describe the optimization problem and the black-box optimization methods that are used in this work.

### 5.2.1 Algorithm selection and hyper-parameter optimization

We approach the problem of optimization of the pipeline from two frameworks. In one framework, each path in the pipeline in 5.1 is individually optimized. This essentially boils down to problem of hyper-parameter optimization (HPO) because the hyperparameters of each algorithm are optimized for each individual path. In the second framework, the entire pipeline is optimized. This means that the algorithms and hyperparameters are optimized together. This is denoted as combined algorithm selection and hyper-parameter optimization (CASH).

### 5.2.1.1 Hyper-parameter optimization (HPO)

Let the  $n$  hyperparameters in a path be denoted as  $\theta_1, \theta_2, \dots, \theta_n$ , and let  $\Theta_1, \Theta_2, \dots, \Theta_n$  be their respective domains. The hyperparameter space of the path is  $\Theta = \Theta_1 \times \Theta_2 \times \dots \times \Theta_n$ .

When trained with  $\theta \in \Theta$  on data  $D_{train}$ , the validation error is denoted as

$\mathcal{L}(\theta, D_{train}, D_{valid})$ . Using k-fold cross-validation, the hyperparameter optimization problem for a dataset  $D$  is to minimize:

$$f^D(\theta) = \frac{1}{k} \sum_{i=1}^k \mathcal{L}(\theta, D_{train}^{(i)}, D_{valid}^{(i)}) \quad (5.1)$$

Hyperparameters  $\theta_i$  may be numerical, categorical or conditional with a finite domain. The minimization of this objective function provides the optimal configuration of hyperparameters on a particular path in the pipeline in 5.1.

### 5.2.1.2 Combined algorithm selection and hyper-parameter optimization (CASH)

We can define the CASH formulation using Fig. 1. Let there be  $n$  computational steps in the pipeline. Each step  $i$  in the pipeline consists of algorithms  $A_i(\Theta_i)$ , where  $A_i(\Theta_i) = \{A_{i1}(\theta_{i1}), \dots, A_{im_i}(\theta_{im_i})\}$ ,  $m_i$  is the number of algorithms in step  $i$ ,  $A_{ij}$  represents the  $j$ -th algorithm in step  $i$ , and  $\theta_{ij}$  represents the set of hyperparameters corresponding to  $A_{ij}$ . The entire space of algorithms and hyperparameters is therefore given by

$\mathcal{A} = A_1(\Theta_1) \times A_2(\Theta_2) \times \dots \times A_n(\Theta_n)$ . The objective function to be minimized for CASH is given by

$$f^D(A) = \frac{1}{k} \sum_{i=1}^k \mathcal{L}(A, D_{train}^{(i)}, D_{valid}^{(i)}) \quad (5.2)$$

where,  $A \in \mathcal{A}$  and other notations are the same as those introduced in the previous section.

## 5.2.2 Optimization methods

The critical step in HPO or CASH is to choose the set of trials in the search space, which is  $\Theta$  for HPO and  $\mathcal{A}$  for CASH. In this section, methods that are used in this paper for optimization of Eq. 5.1 and Eq. 5.2 are described. Grid search, random search and Bayesian optimization are used in this work.

### 5.2.2.1 Grid search

Grid search is the simplest of all methods for coming up with trials in the search space. The set of trials in grid search is formed by assembling every possible set of values in  $\Theta$  and  $\mathcal{A}$  and computing the validation loss. The configuration  $\theta \in \Theta$  or  $A \in \mathcal{A}$  that minimizes the validation loss  $\mathcal{L}$  is chosen as the optimum configuration. Unfortunately grid search is computationally very expensive. For HPO, the number of trials corresponds to  $\prod_{i=1}^n |\Theta_i|$ , and for CASH this is  $\prod_{i=1}^n |A_i(\Theta_i)|$ . This product makes grid search suffer from the *curse of dimensionality*. This is because the number of trials grows exponentially with the number of hyperparameters. However, grid search has certain advantages. Firstly, parallelization and implementation is trivial. In addition, grid search is robust in the sense that results maybe replicated easily. Therefore, grid search is the preferred optimization method in low-dimensional hyperparameter spaces.

### 5.2.2.2 Random search

Random search is the optimization method where trial configurations are randomly sampled from the search space of HPO ( $\Theta$ ) or CASH ( $\mathcal{A}$ ). [56] shows empirically and theoretically that randomly selecting trials is more efficient than performing optimization using grid search.

### 5.2.2.3 Bayesian optimization

Sequential model based Bayesian optimization (SMBO) [60] is the method of choice when it comes to optimization of complicated black-box functions. In a nutshell, it consists of two components. The first is a probabilistic model and the second is an acquisition function. The probabilistic model can be modelled using Gaussian processes (Spearmint) [57], random forests (SMAC) [60] and using density estimation with Tree-structured Parzen estimators (TPE) [61]. The acquisition function determines the future candidates or trials for evaluation. The acquisition function is relatively cheap to evaluate compared to the actual objective function  $f^D$ . One of the most prominent acquisition functions is *expected improvement* (EI) [62]. We use the sequential model-based algorithm configuration (SMAC) that uses random forests as the Bayesian optimization framework.

## 5.3 Proposed methods

In this section the proposed methodology for error quantification and propagation is presented. Both of these methods are independent of the black-box optimization methods that maybe used for both the HPO and CASH formulations.

### 5.3.1 Error quantification

Machine learning pipelines maybe understood and interpreted by quantifying the contribution of error from different parts of the pipeline. For example, it is useful for machine learning experts and domain experts to understand and identify where the source of the error is in a pipeline. Referring back to Fig. 1, if it was known that most of the error in the final result originated from feature extraction, then machine learning practitioners would devote more time and energy to coming up with better algorithms for feature extraction or fine-tuning the algorithms in that step to reduce the error. In addition, if it were possible to quantify the contribution of errors from certain algorithms in the pipeline, then the data scientists would try to fine tune the algorithms in the pipeline or even try to replace the algorithms with better alternatives.

We propose an *agnostic* methodology for quantifying error contributions from different parts of the pipeline. It is quantified as the minimum error obtained by being agnostic to a particular component of the pipeline (computational step or algorithms). We shall define what *agnostic* refers to for both computational steps and algorithms individually.

#### 5.3.1.1 Quantification of error from computational steps

The *agnostic* methodology maybe used for quantification of error contributions from computational steps like feature extraction, data pre-processing and and learning algorithms. Being *agnostic* to a computational step means that the algorithms in that step are selected randomly for that step while the remaining pipeline is optimized. The average of the minimum errors obtained with each algorithm in the step used as the only algorithm in that particular step, provides an estimate of the agnostic error from a particular pipeline. More formally, the agnostic methodology is implemented for computational steps in the following manner. Using 5.1 as a reference, let  $n$  be the number of steps in the pipeline. Each step in the pipeline is denoted as  $S_i$ .  $|S_i|$  is the number of algorithms in step  $i$ .  $A_{ij}$  denotes the  $j$ -th algorithm in the  $i$ -th step.  $E^*$  represents the minimum validation error found after

optimization of the entire pipeline (using CASH or HPO framework).  $E_{A_{ij}}^*$  is the minimum validation error found with  $A_{ij}$  as the only algorithm in step  $i$ . The error contribution from step  $i$ ,  $EC_{S_i}^*$  is given by Eq. 5.3.

$$EC_{S_i}^* = \frac{1}{|S_i|} \sum_{j=1}^{|S_i|} E_{A_{ij}}^* - E^*, \quad (5.3)$$

where,  $i = 1, \dots, n, j = 1, \dots, |S_i|$  Taking the difference with respect to the global minimum in Eq. 5.3 provides an estimate of the error contribution from step  $i$  of the pipeline.

### 5.3.1.2 Quantification of error from algorithms

The *agnostic* methodology for algorithms is implemented as follows. Similar to the *agnostic* methodology for steps defined above, we define the *agnostic* methodology for algorithms. In this case, we focus on a single path in the pipeline in Fig. 1. Let's assume we are trying to quantify the error contribution of a particular algorithm  $A_{ij}$  that lies on path  $p$ . Being *agnostic* to  $A_{ij}$  means we optimize everything else on the path except the algorithm. This means that we pick the hyperparameters  $\theta_{ij}$  of algorithm  $A_{ij}$  randomly while optimizing the rest of the algorithms on the path. This is formally calculated by taking the average of the optimum errors on the path for each configuration of  $\theta_{ij}$ . This minimum validation error on the path is then subtracted from this error to give us the error contribution from algorithm  $A_{ij}$  on path  $p$ .

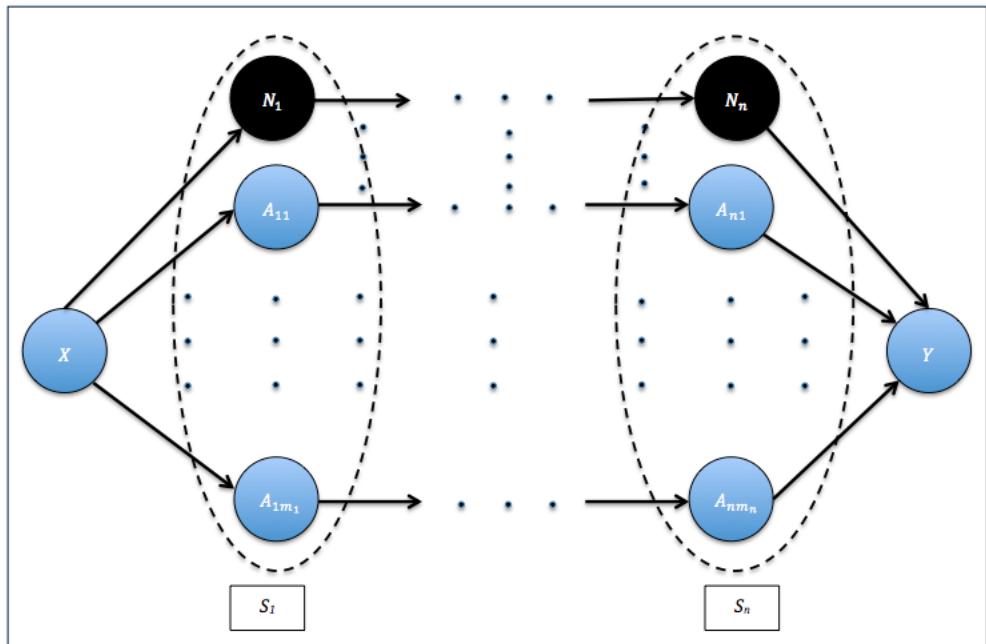
$$EC_{A_{ij}}^* = \frac{1}{|\theta_{ij}|} \sum_{k=1}^{|\theta_{ij}|} E_{A_{ij}}^k - E_{A_{ij}^p}^*, \quad (5.4)$$

where,  $i = 1, \dots, n, j = 1, \dots, |\theta_{ij}|$ ,  $|\theta_{ij}|$  represents the number of hyperparametric configurations of  $A_{ij}$ ,  $E_{A_{ij}}^k$  is the minimum error obtained with the  $k$ -th configuration of  $\theta_{ij}$  and  $E_{A_{ij}^p}^*$  is the minimum error found over the path  $p$  that consists of algorithm  $A_{ij}$ .

### 5.3.2 Error propagation

Understanding how error propagates along a machine learning pipeline is important for its analysis. The final error that is obtained in a pipeline consists of two sources. The first is the actual error that originates from the various components of the pipeline. The second is the error that is accumulated along the pipeline because of the propagation or

the propagated error. For example an algorithm like principal components analysis that is used for transforming the features may provide a better performance with one feature extraction algorithm over another. Data scientists and domain experts alike may use this tool to understand and analyze a pipeline in detail to make sense of how the error propagates. This difference in performance maybe attributed to propagation. We propose a method to model the error propagation for both algorithms and steps using the error contributions computed using the formulation in the previous section. This is done by introducing *naive* algorithms at each step of the pipeline. This model is represented in Fig. 5.2.



**Fig. 5.2. Naive methodology used in the error propagation model**

As shown in the figure above, *naive* algorithms are introduced at each step. They are denoted as  $N_i$ . This naive formulation is an integral component for the error propagation model. The assumption we make is that all of the error that is made by the naive algorithms is because of the algorithm itself and no error is propagated because of it. We can therefore define 6 different types of errors using this approach. These errors are represented in Table 1.

**Table 5.1. Notations for error definitions used in the error propagation model**

Notation	Definition
$E_{opt \rightarrow opt}$	Minimum error found by using the optimum algorithm or hyperparameter the current step or algorithm respectively and all the steps or algorithms that proceed it.
$E_{agnostic \rightarrow opt}$	Minimum error found by using the <i>agnostic</i> methodology for the current step or algorithm and optimizing all the steps or algorithms that proceed it.
$E_{naive \rightarrow opt}$	Minimum error found by using the naive algorithm for the current step or algorithm and optimizing all the steps or algorithms that proceed it.
$E_{opt \rightarrow naive}$	Minimum error found by using the optimum algorithm or hyperparameters for the current step or algorithms respectively and naive algorithms for all the steps or algorithms that proceed it.
$E_{naive \rightarrow naive}$	Minimum error found by using the naive algorithm for the current step or algorithm and naive algorithms all the steps that proceed it.
$E_{agnostic \rightarrow naive}$	Minimum error found by using the <i>agnostic</i> methodology for the current step or algorithm and naive algorithms all the steps that proceed it.

Let us now define 3 types of error contributions using the notations in Table 1.

$$\begin{aligned}\Delta E_1 &= E_{agnostic \rightarrow opt} - E_{opt \rightarrow opt} \\ \Delta E_2 &= E_{agnostic \rightarrow naive} - E_{opt \rightarrow naive} \\ \Delta E_3 &= E_{naive \rightarrow naive} - E_{naive \rightarrow opt}\end{aligned}\tag{5.5}$$

Eq. 5.5 represents the error contributions from each step of the pipeline using the naive and agnostic methodology.  $\Delta E_1$  is the same as the error contribution  $EC_{S_i}$  for steps or  $EC_{A_{ij}}$  for algorithms. described in Section 3.1. It is the error contribution from a particular step or algorithm with respect to the pipeline or path respectively.  $\Delta E_2$  is the error contribution

from a step or algorithm with naive algorithms in the steps proceeding it. This value represents the error contribution without error propagation.  $\Delta E_3$  is the quantification of the actual error that is added by the naive algorithms. Let us define the actual error contributed by a step or algorithm as  $\alpha$  and the propagated error as a multiple of the actual error. Let this be denoted as  $\gamma\alpha$ . We denote  $\gamma$  as the *propagation factor*. Then we can write  $\Delta E_1$  as,

$$\Delta E_1 = \alpha + \gamma\alpha \quad (5.6)$$

Similarly,  $\Delta E_2$  consists of the actual error from the step or algorithm and the propagated error. The propagated error is a multiple of the actual error and the error added by the naive algorithms. Therefore, we can formalize this as:

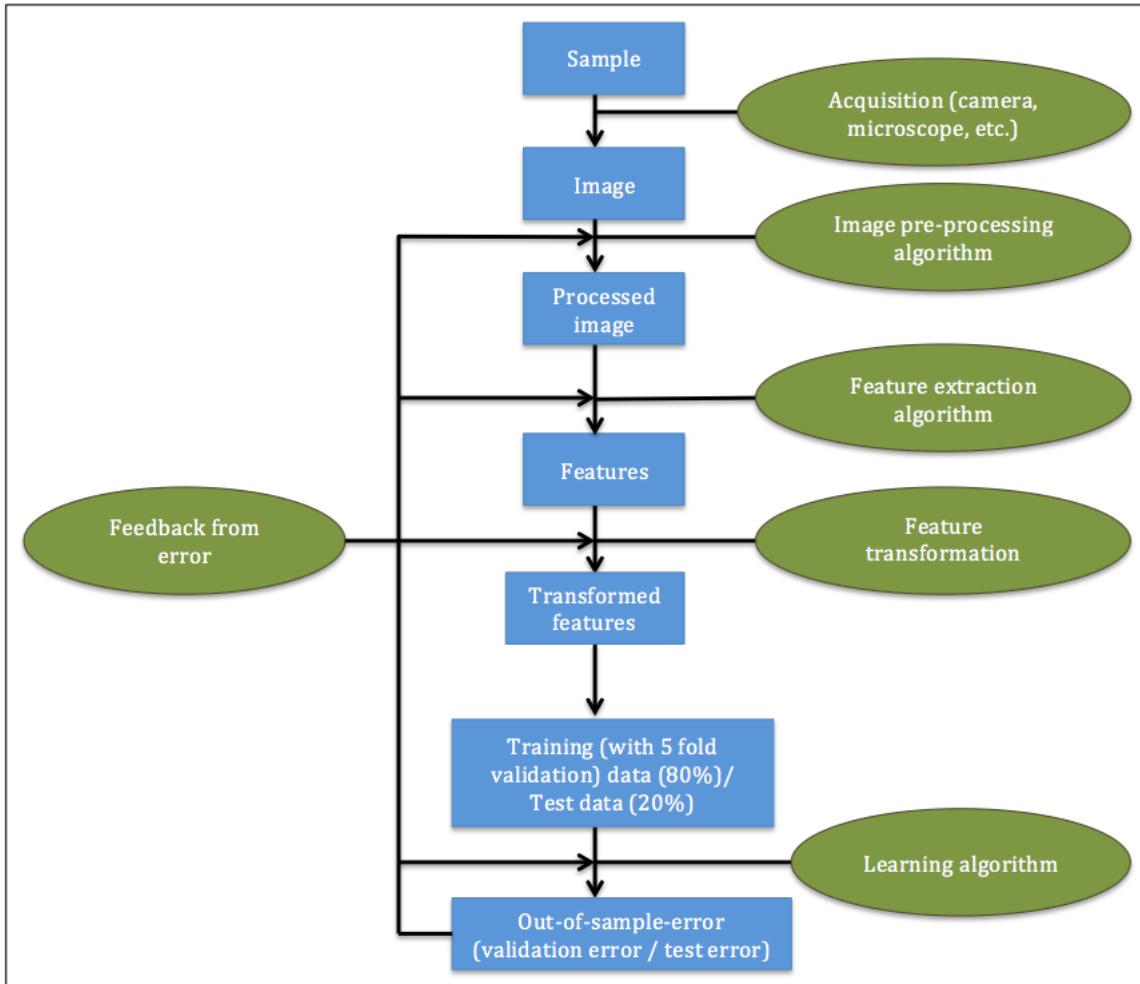
$$\Delta E_2 = \alpha + \gamma(\alpha + \Delta E_3) \quad (5.7)$$

Let us denote the actual error as  $E_{direct}$  which is the same as  $\alpha$ , and the propagated error as  $E_{propagation}$  which is  $\gamma\alpha$ . Solving Eq. 7 and Eq. 8 simultaneously, we get the values for  $E_{direct}$ ,  $E_{propagation}$  and the *propagation factor*  $\gamma$

$$\begin{aligned} E_{direct} &= \frac{\Delta E_1 \Delta E_3}{\Delta E_2 + \Delta E_3 - \Delta E_1} \\ E_{propagation} &= \frac{\Delta E_1 (\Delta E_2 - \Delta E_1)}{\Delta E_2 + \Delta E_3 - \Delta E_1} \\ \gamma &= \frac{\Delta E_2 - \Delta E_1}{\Delta E_3} \end{aligned} \quad (5.8)$$

## 5.4 Experiments and results

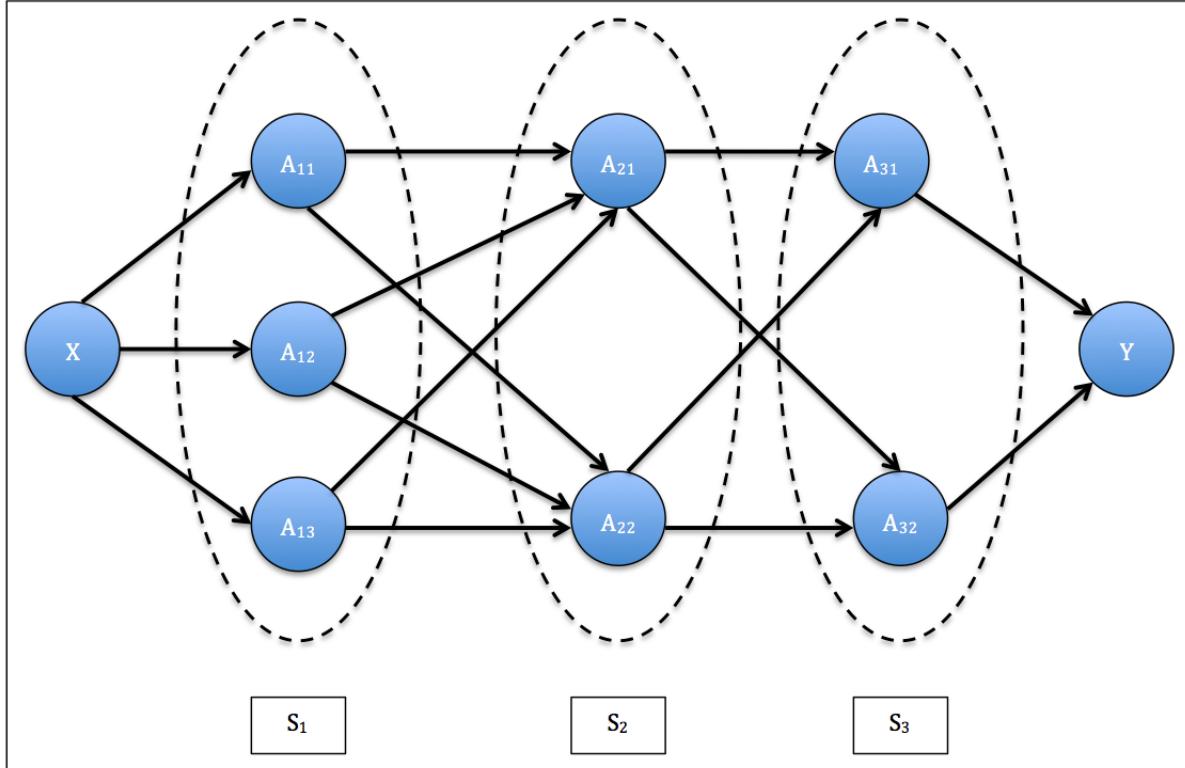
In this section, we describe the experiments performed on the data analytic pipeline to quantify the error contributions and propagation from different components of the pipeline. Image classification is the data analytic problem chosen for demonstrating the error quantification experiments. A representation of an image classification pipeline is shown in Fig. 3 in the form of a flowchart.



**Fig. 5.3. Representation of an image classification pipeline**

In this work, we focus on real world datasets from the domains of medical pathology and material science. Therefore, the flowchart starts with a sample that is imaged with acquisition technology like a camera or a microscope. The image is then processed using image pre-processing algorithms like normalization and standardization. This may also include image segmentation algorithms. This is followed by feature extraction algorithms that extract useful information from the images. Sometimes, the features extracted are transformed to a different vector space using feature transformation (feature selection or dimensionality reduction) algorithms. The dataset is then divided into training and test datasets in an 80-20 split. Finally, learning algorithms like random forests and logistic regression are used as learning algorithms to build a predictive model for the image classification problem. The performance of the pipeline is evaluated using classification metrics like F1-score, accuracy,

precision and recall. We use the validation and test errors as the estimate of the out-of-sample error. This error is then used as a feedback to quantify the contribution of the errors from different components of the pipeline. The specific pipeline used in this work is shown in the following figure.



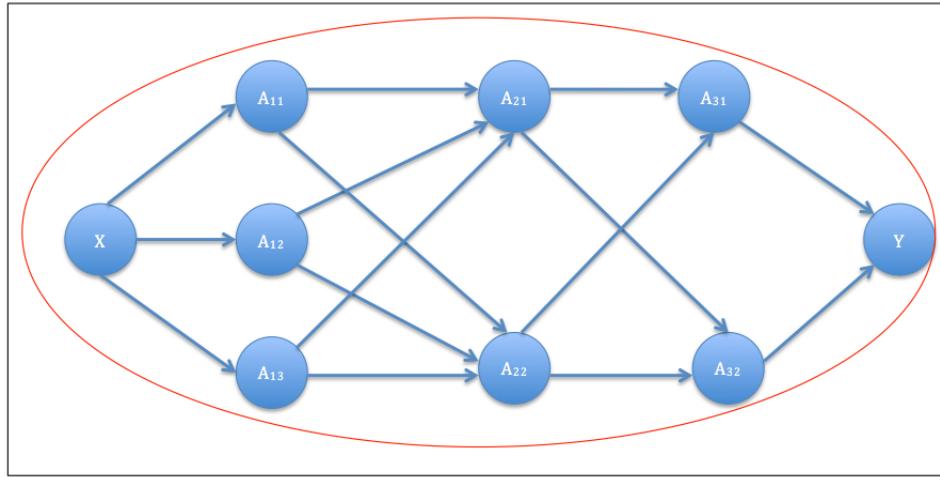
**Fig. 5.4. Representation of the image classification pipeline**

The above figure shows the pipeline used in this work. There are 3 computational steps in this pipeline, namely feature extraction ( $S_1$ ), feature transformation ( $S_2$ ) and learning algorithms ( $S_3$ ). The steps, algorithms and corresponding hyperparameters  $A_{ij}(\theta_{ij})$  is described in Table 2.

The algorithms defined above in the table are selected for making up the components of the pipeline in Fig. 4. This is meant to serve as an example for demonstrating the experiments using the error quantification framework described in section 3. It can easily be generalized to any data analytic problem that involve pipelines.

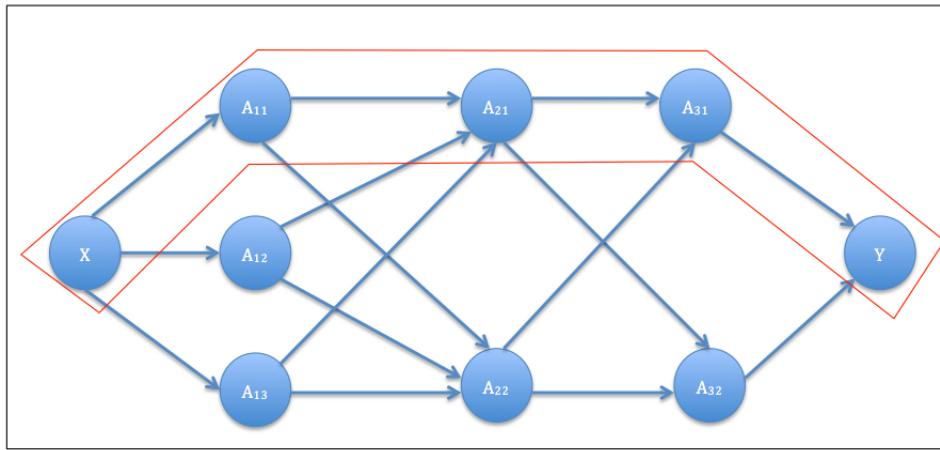
### 5.4.1 Optimization frameworks

Experiments are performed using two optimization frameworks. These frameworks have been described in detail in Section 2.1. The first global optimization framework is the CASH framework described in Section 2.1.2. Here, the pipeline is optimized as a whole including the algorithms, which are themselves considered as hyper-parameters in this framework. The following figure is a representation of this. This is used for quantification of the contribution and propagation of error with respect to steps in the pipeline.



**Fig. 5.5.** Combined algorithm selection and hyper-parameter optimization in a data analytic pipeline. The algorithms and corresponding hyper-parameters are optimized simultaneously.

The second is the hyperparameter optimization (HPO) framework where each path in the pipeline is optimized as shown in the following figure. This framework is used for quantifying the contribution and propagation of error with respect to algorithms in the each path of the pipeline.



**Fig. 5.6.** Hyper-parameter optimization in a data analytic pipeline. Each path in the pipeline is individually optimized.

### 5.4.2 Datasets

Four datasets from the domains of medicine and material science are used in this work. They are image datasets of breast cancer [63], brain cancer [64], and two datasets of microstructures in material science [65]. They are described in the following table.

**Table 5.2. Notations for error definitions used in the error propagation model**

Dataset (notation)	Distribution of classes
Breast cancer ( <i>breast</i> ) [63]	<i>benign</i> : 151, <i>in-situ</i> : 93, <i>invasive</i> : 202
Brain cancer ( <i>brain</i> ) [64]	<i>glioma</i> : 16, <i>healthy</i> : 210, <i>inflammation</i> : 107
Material science 1 ( <i>matsc1</i> ) [65]	<i>dendrites</i> : 441, <i>non-dendrites</i> : 132
Material science 2 ( <i>matsc2</i> ) [65]	<i>transverse</i> : 393, <i>longitudinal</i> : 48

The above table represents datasets from the scientific domain. These datasets have been chosen because they represent examples of real world datasets. They are noisy in the sense that they have artefacts in the images, are heavily imbalanced and are small in terms of number of samples. They are different from the very large datasets like ImageNet [46], where deep learning techniques like convolutional neural networks have been shown to be superior. Even though deep neural networks represent an end-to-end workflow where the input image is fed into the network and the output classification is obtained at the other end, they may also be represented as pipelines, if the hyper-parameters of the network are considered. [66] has shown that machine learning problems involving datasets from medical imaging may be solved using pre-trained and fine-tuned neural networks rather than training them from scratch. We have therefore used pre-trained models such as VGGnet [67] and InceptionNet [68] as pre-trained feature extraction models that fit naturally in the pipeline framework described here for the purpose of illustrating the error quantification methodology.

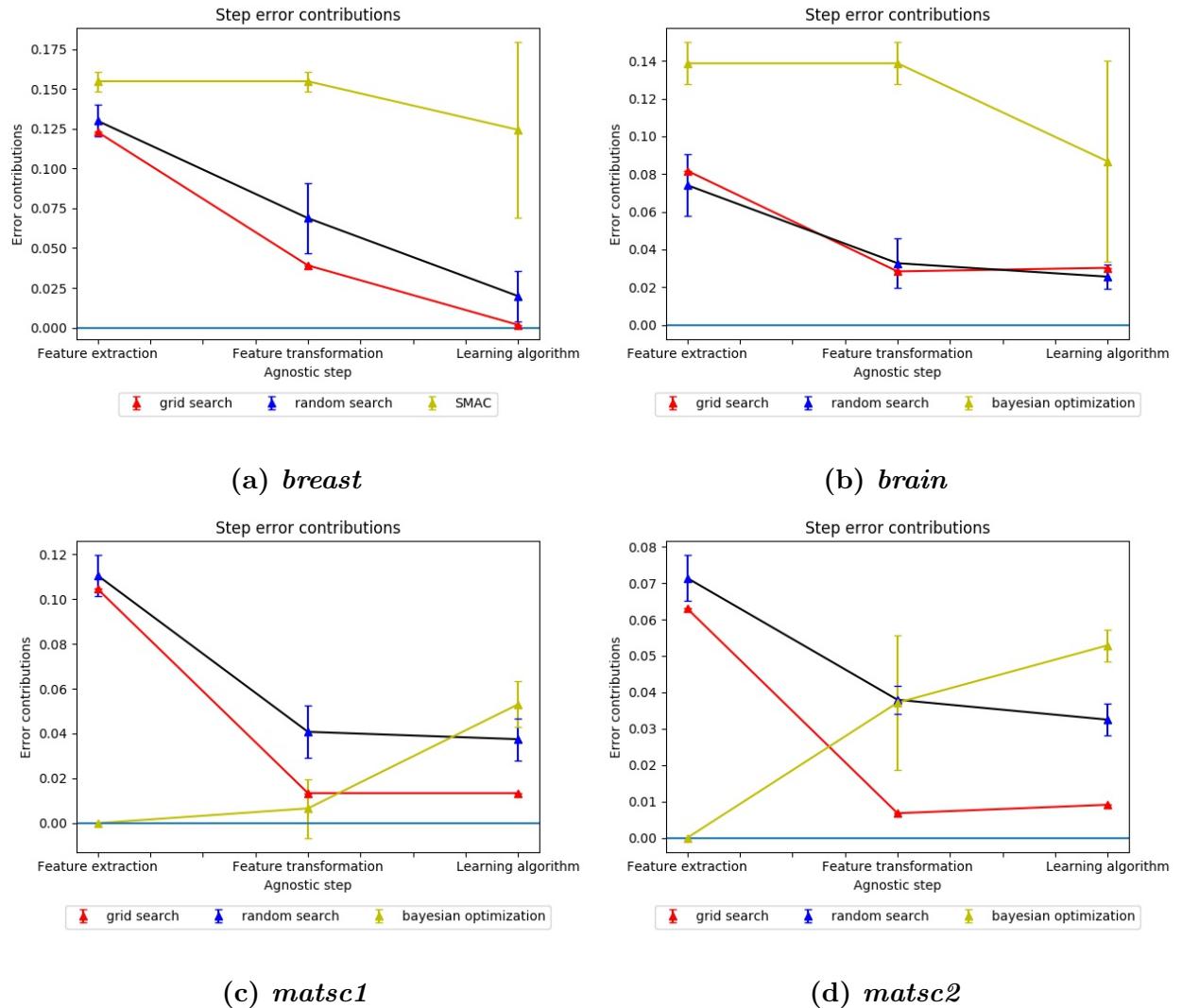
### 5.4.3 Error quantification experiments

Experiments based on the quantification of error contributions framework described in section 3.1 are presented here. The plots are of the error contribution values calculated using Eqs. 3, 4 and 5 on the 4 datasets described in Table 3. The error contribution values are obtained from the trials in the results of optimization methods in Section 2. Grid search is only run once while the other algorithms are averaged over 5 runs with the mean and

standard deviation shown in the following plots. These results are on the validation error obtained at the end of the pipeline. Random search and Bayesian optimization (using the SMAC algorithm) are also implemented on both the frameworks described in Section 4.1.

#### 5.4.3.1 Error contribution from computational steps

The error contributions from computational steps in the pipeline is shown in the following figure. We can observe that most of the error comes from feature extraction algorithms. The error is calculated using the formulation of  $EC_{S_i}$  in section 3.1.1. This confirms our belief that feature extraction algorithms are the most important components of a machine learning pipeline. The grid search results maybe used as the gold standard to compare the performance of other optimization algorithms.



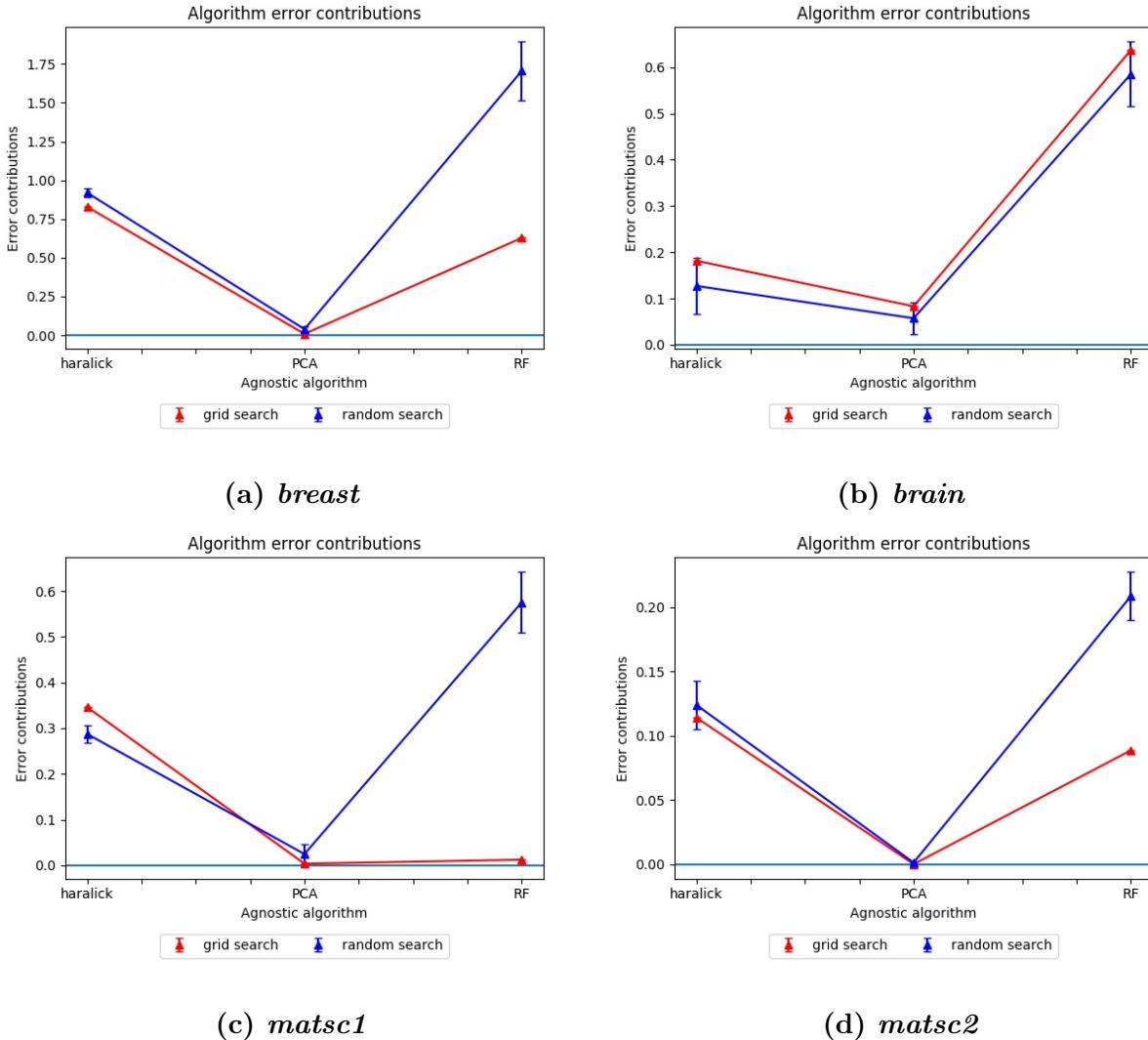
**Fig. 5.7. Plots of error contributions from computational steps in the pipeline.**

The mean and standard deviation values of  $EC_{S_i}$  calculated using Eq. 5.3 is represented in the figure above for the 4 datasets in Table 3. The standard deviation of grid search is 0 because it was only run once for each dataset due to the time required for computation and also because of the robustness of the grid search method (the results don't change because we try out every single configuration). We observe from the plots results for both random search and Bayesian optimization (SMAC) follow the behavior of grid search. The random search results using the CASH framework mirrors the behavior of grid search, but the results are not robust because of the relatively high standard deviation. This is because the search trials do not include all the configurations of algorithms in the pipeline, which are necessary

for the computation  $EC_{S_i}$  using Eq. 5.3. The results of SMAC in the CASH framework are even worse. This is because the trials for SMAC are even more sparse with respect to the algorithms it selects for optimization of the error in Eq. 2. SMAC only samples a few configurations based on the updated probabilistic model as it narrows in on the optimum error. Therefore, it sometimes gives erroneous results as can be seen for *matsc1* and *matsc2* in Figs. 7(c) and 7(d) respectively.

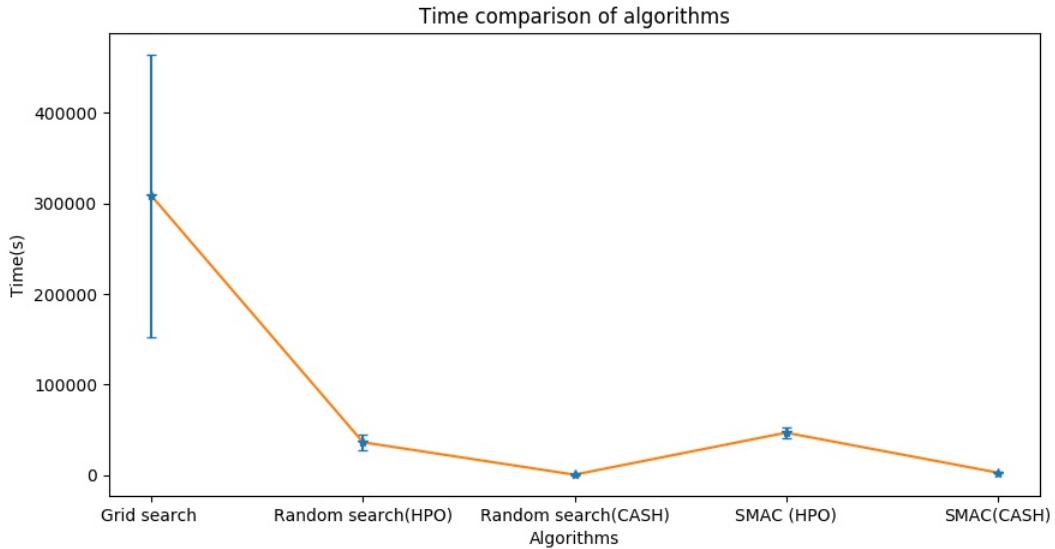
#### 5.4.3.2 Error contribution from algorithms

In the following figure, the error contributions from algorithms are quantified using the formulation in Section 3.1.2. We select *haralick texture features*, *PCA* and *random forests* as the path to demonstrate the contribution of error from algorithms. We observe a trend here, in that, the error contributed from *haralick texture features* and *random forests* is more than *PCA*. This maybe attributed to the fact that both *haralick texture features* and *random forests* have 2 hyperparameters each while *PCA* has only one hyperparameter. Again, we see the trend that random search performs better in terms of following the behavior of grid search than Bayesian optimization.



**Fig. 5.8. Plots of error contributions from algorithms in the pipeline (method 1).**

The following figure shows the average computation times of each of the algorithms used in the above error quantification experiments. We can clearly observe that random search is the most efficient in terms of computational time among the 3 algorithms used in this work.

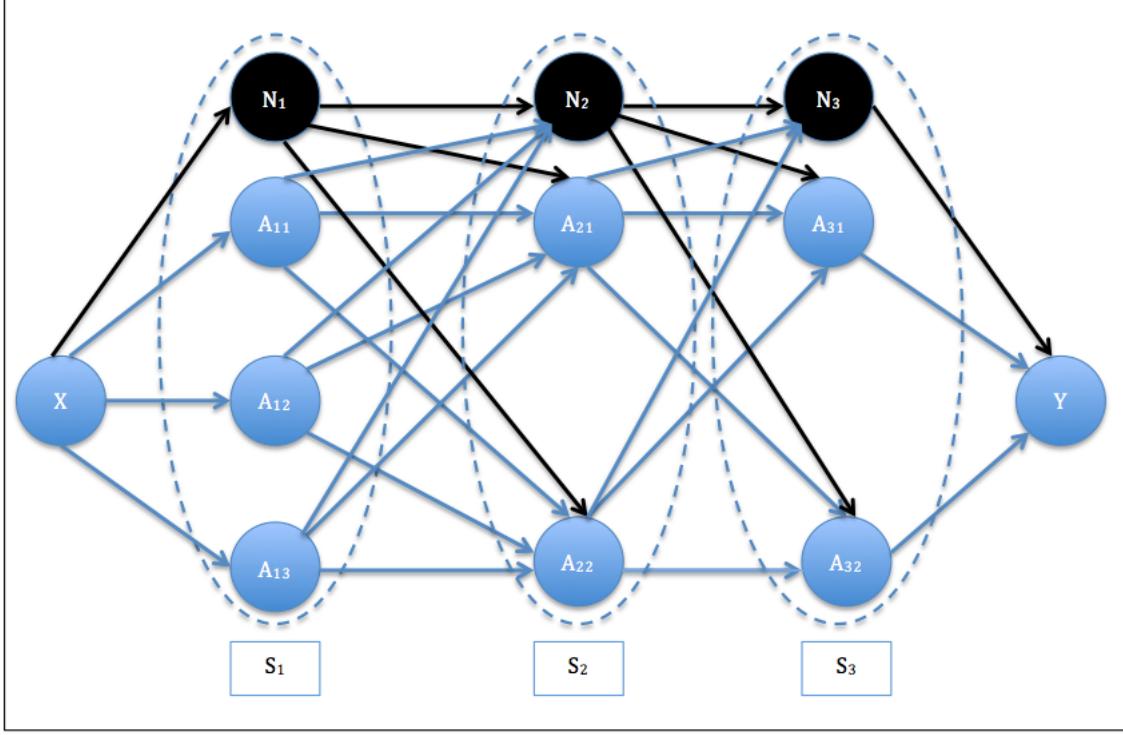


**Fig. 5.9. Pipeline including naive algorithms for error propagation**

From Figs. 7, 8 and 9, we observe that the random search based optimization on the CASH framework follows the behavior of grid search and can be used for quantification of error contributions from computational steps. In addition, we see from Fig. 10, that the average computation time for random search on the CASH framework is the least among all the optimization frameworks. Therefore, we use the random search framework for quantification of error contributions in the pipeline in the following section

#### 5.4.4 Error propagation experiments

The set of experiments in this section correspond to the error propagation model formulated in Section 3.2. The corresponding pipeline including the naive algorithms are represented in the following figure.

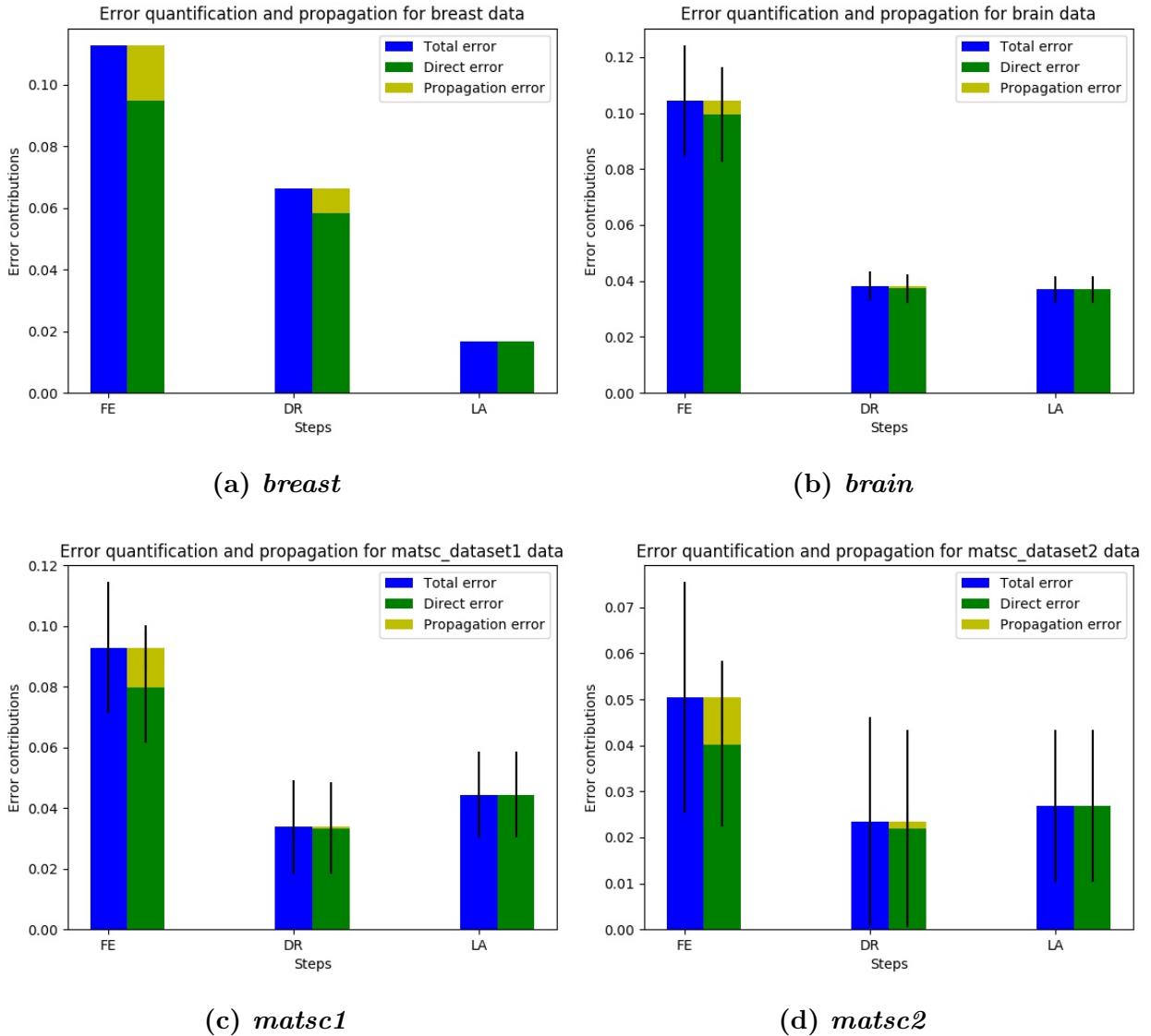


**Fig. 5.10. Pipeline including naive algorithms for error propagation**

The naive algorithms used in this work are *vectorization* (flattening the image into a vector) for feature extraction, *no transformation* for feature transformation, and *1-nearest neaighbors* algorithm as the classification or learning algorithm. The naive algorithms are selected such that the only error that result from them is the direct error, as stated in Section 3.2.

The following figures shows the results of the error propagation framework. The results are obtained based on the CASH based random search for steps and HPO based random search on paths for algorithms. Fig. 11 shows the plots for error propagation along the steps in the image classification pipeline. The plots show the total error  $EC_{S_i}$ , the direct error  $E_{direct}$  and the propagation error  $E_{propagation}$ . These values are computed using error propagation model defined in section 3.2.1. The total error is the error contribution calculated using Eq. 3. We can see from the results that the total error is dominated by the direct error from the algorithms in the steps. The propagation error is the difference between the total error and the direct error from the algorithms. We also note that the propagation error progressively reduces in magnitude from feature extraction algorithms (that are at the beginning of the pipeline) to learning algorithms (that are at the end of the pipeline). Therefore,

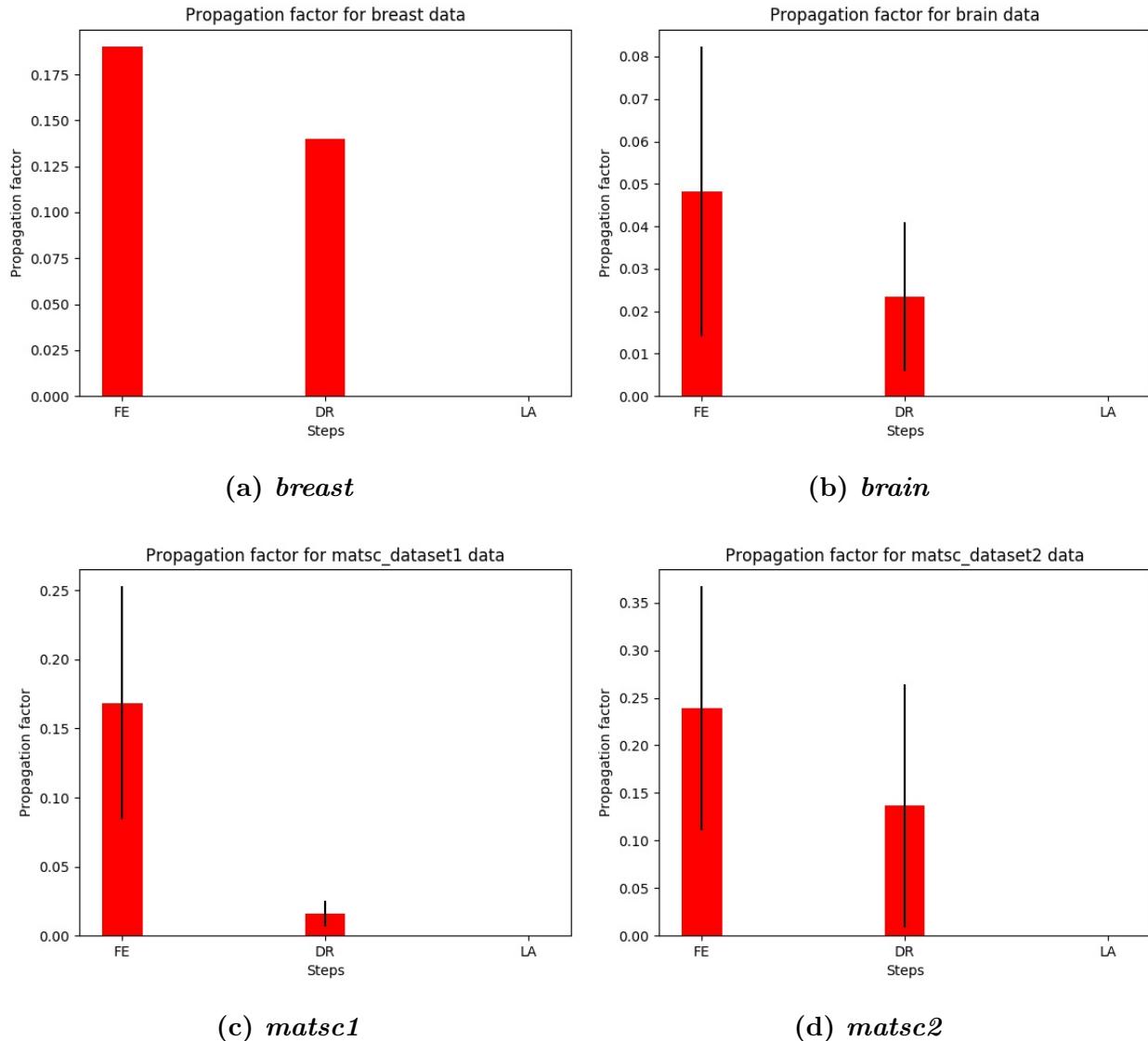
the model captures our intuition that usually more error is propagated from steps that exist closer to the beginning of the pipeline.



**Fig. 5.11. Plots of error propagation using the naive algorithm based methodology.**

The following figure shows the propagation factor  $\gamma$  computed in Eq. 6. The value of this factor quantifies the propagation from a particular step in the pipeline. A relatively high value means that the factor by which the error is propagated down the pipeline from a particular step is large. As expected, we observe that there the propagation factors are large for feature extraction algorithms and they progressively reduce in magnitude as we move

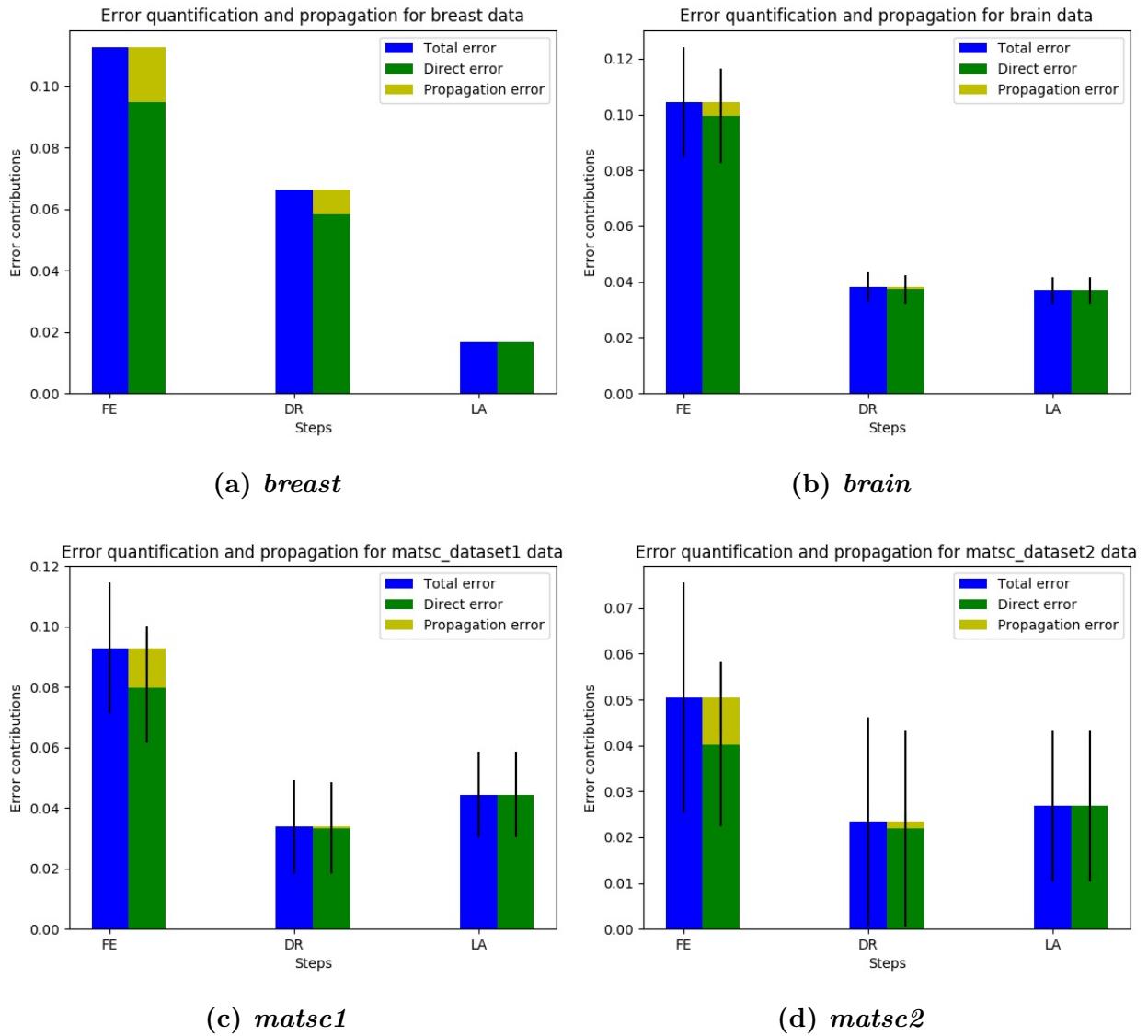
closer towards the end of the pipeline. The propagation factor for learning algorithms is 0 because this step is at the end of the pipeline, and all the error from the step is directly due to the algorithms themselves.



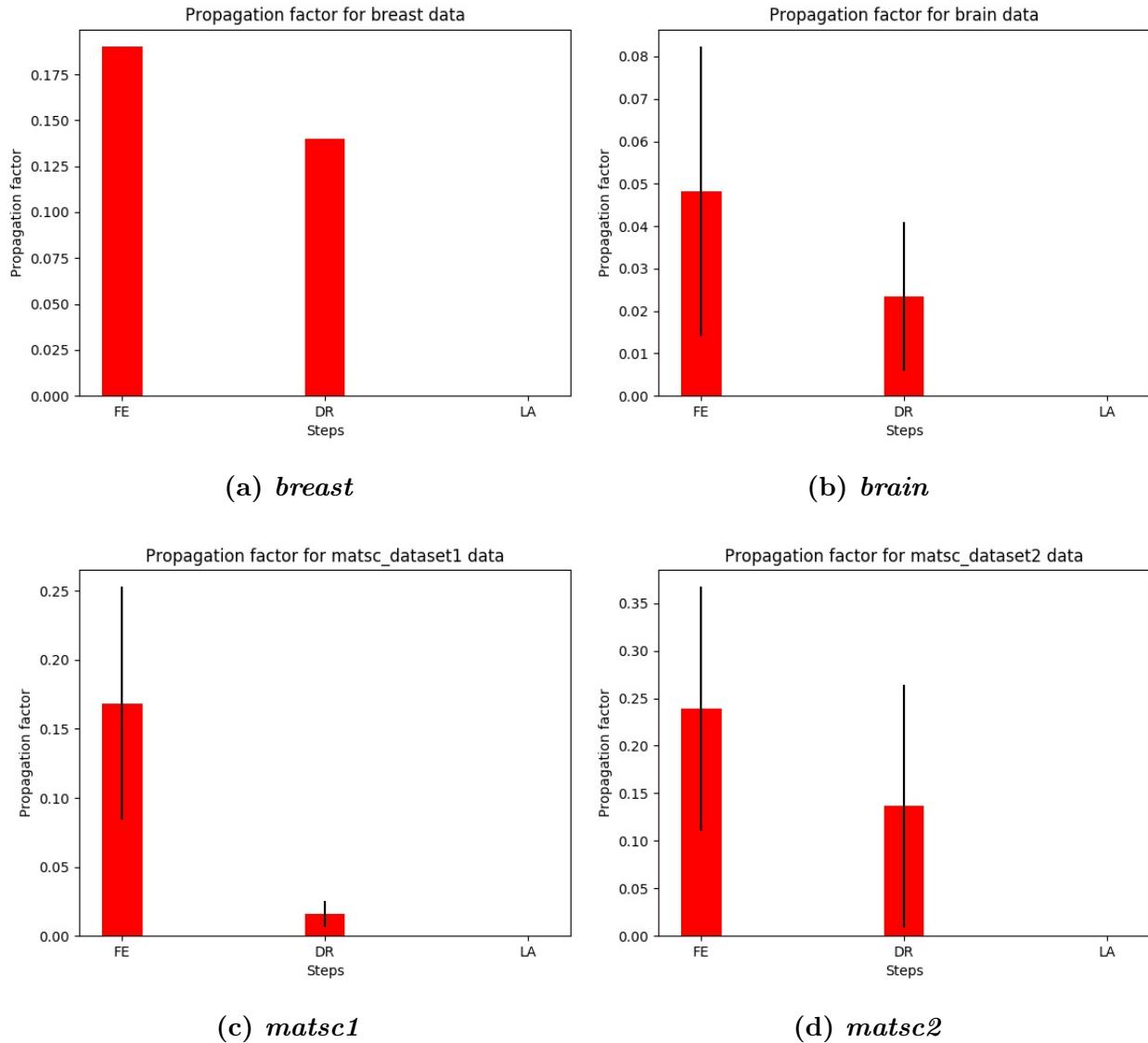
**Fig. 5.12.** Plots of propagation factor  $\gamma$  using the naive algorithm based methodology.

The same computations for error propagation are carried out with respect to algorithms. Again in this formulation, we optimize a path in the pipeline. The path used here is the same as that used in the error contribution experiments for algorithms in section 4.3.2 - *haralick texture features, PCA* and *random forests*. The following figures represent the

corresponding plots representing the error propagation and propagation factors with respect to algorithms.



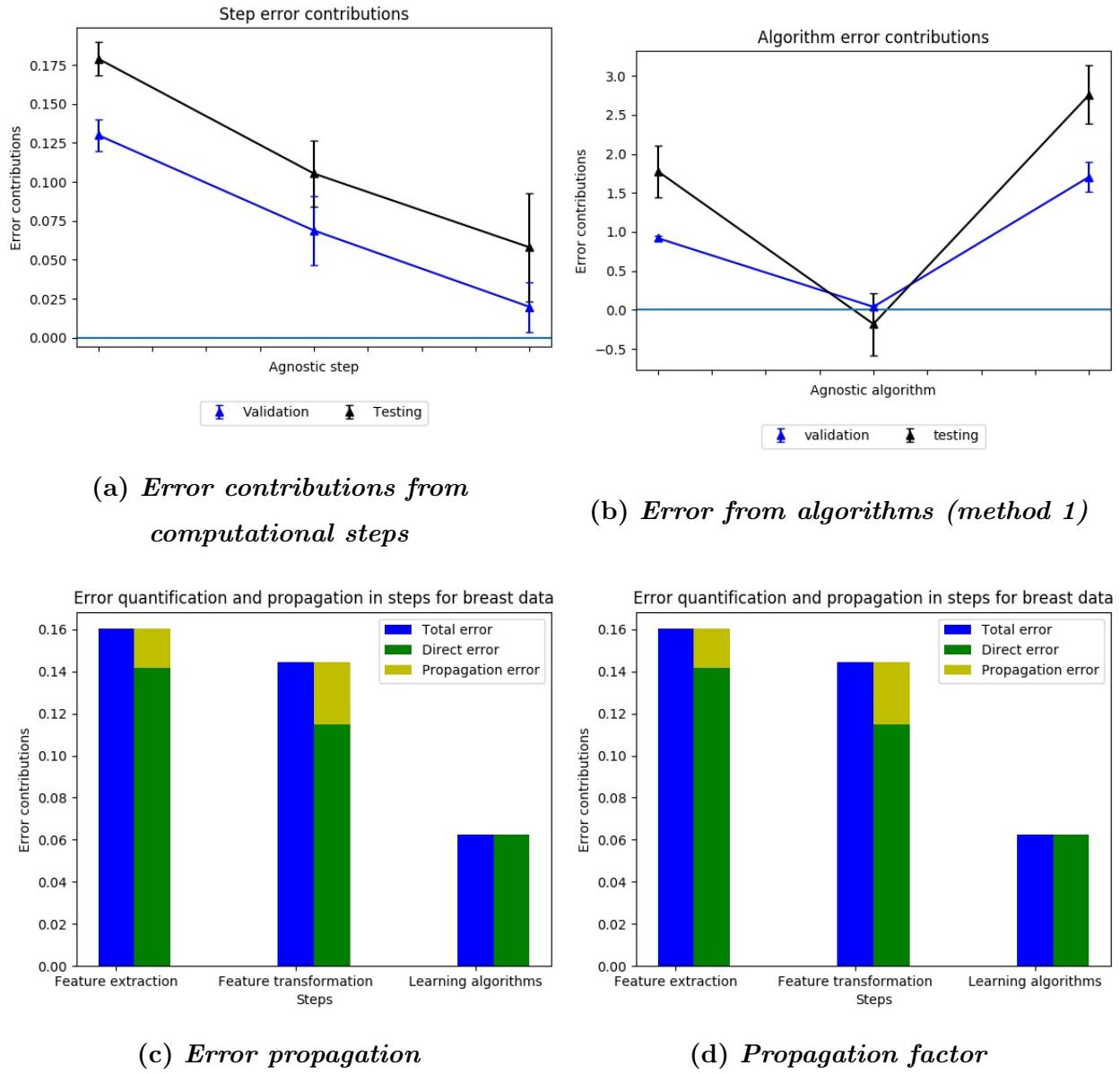
**Fig. 5.13.** Plots of error propagation using the naive algorithm based methodology.



**Fig. 5.14.** Plots of propagation factor  $\gamma$  using the naive algorithm based methodology.

All of the figures shown above are computed with respect to the validation dataset. We also perform the same experiments on the held out test data. The following figure shows the results of the error contribution and propagation on the test data for *breast* dataset. We compare the results of the held out test data with the results from the validation data. As we can see, the magnitude of error contributions from the test data are slightly larger than that from validation data. This can be explained due to slight overfitting on the validation dataset. The error propagation results on the test dataset is also different from the validation

dataset. This can also be due to error from generalization.



**Fig. 5.15. Plots of error contribution and error propagation on test data from *breast* data.**

## 5.5 Conclusion

This work proposes a methodology for making black box machine learning pipelines interpretable. The suggested approach involves understanding the sources and flow of error in data analytic pipelines. Specifically, we propose a methodology to quantify the error

contributions from different parts of an image classification pipeline, namely computational steps and algorithms. This methodology is described as the *agnostic* method described in Section 3.1. The results in Section 4.3 show that the black-box optimization methods like random search and Bayesian optimization is able to quantify the error contributions as well as grid search although the CASH framework of Bayesian optimization is not as accurate due to reasons specified above. The error quantification methodology maybe used by machine learning practitioners to understand and interpret results of a specific machine learning problem on a dataset. Understanding the source of error in terms of steps and algorithms will help data scientists quickly iterate and find the best set of algorithms along with their hyper-parameter configurations for solving a particular task. We also propose a model for error propagation to analyse the error from the pipeline even further. This methodology is denoted as the *naive* framework, where we use *naive* algorithms in each step of the pipeline. *Naive* algorithms represent algorithms where the assumption is that all of the error is a result of the direct error from the algorithm itself and not due to accumulation and propagation of error. This is described in detail in section 3.4. The results from section 4.4 show that most of the error propagates from the feature extraction algorithms followed by feature transformation algorithms. This support our intuition that error accumulates and flows along the pipeline, with the most amount of error coming from steps at the beginning of the machine learning pipeline and it gradually reduces to zero at the end of the pipeline (learning algorithms), where all of the error is directly due to algorithms themselves. In terms of future work, this formulation could be expanded to cover more data analytic problems that involve complicated algorithms. For example, this framework maybe used to interpret problems in other applications like speech recognition, text classification and even unsupervised machine learning problems. Essentially, the error quantification framework maybe used by any practitioner that works with pipelines for solving a machine learning problem. The pipeline may even be extended to include processes like data cleaning. This framework may also be included to understand and interpret deep neural networks, which are end-to-end in nature. This maybe used for comparing the performance of candidate networks for solving the problem.

## **CHAPTER 6**

### **CONCLUSIONS AND FUTURE WORK**

#### **6.1 Conclusions**

## REFERENCES

- [1] J. R. Rodgers and D. Cebon, "Materials informatics," *MRS Bulletin*, vol. 31, pp. 975–980, 12 2006, [Online]. Available: [http://journals.cambridge.org/article\\_S0883769400011702](http://journals.cambridge.org/article_S0883769400011702)
- [2] S. Broderick, C. Suh, J. Nowers, B. Vogel, S. Mallapragada, B. Narasimhan *et al.*, "Informatics for combinatorial materials science," *JOM*, vol. 60, no. 3, pp. 56–59, 2008, [Online]. Available: <http://dx.doi.org/10.1007/s11837-008-0035-x>
- [3] S. R. Kalidindi, S. R. Niezgoda, and A. A. Salem, "Microstructure informatics using higher-order statistics and efficient data-mining protocols," *JOM*, vol. 63, no. 4, pp. 34–41, 2011, [Online]. Available: <http://dx.doi.org/10.1007/s11837-011-0057-7>
- [4] B. G. S. Sergei V. Kalinin and R. K. Archibald, "Big-deep-smart data in imaging for guiding materials design," *Nature Materials*, vol. 14, pp. 973–980, September 2015.
- [5] B. L. DeCost and E. A. Holm, "A computer vision approach for automated analysis and classification of microstructural image data," *Computational Materials Science*, vol. 110, pp. 126 – 133, 2015, [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0927025615005066>
- [6] R. J. Schaefer and D. J. Lewis, "Directional solidification in a agcuso eutectic alloy," *Metallurgical and Materials Transactions A: Physical Metallurgy and Materials Science*, vol. 36A, pp. 2775–2783, October 2005.
- [7] J. Mao and D. J. Lewis, *Nucleation promotion of Sn-Ag-Cu lead-free solder alloys via micro alloying.*, 2014, [Online]. Available: <http://libproxy.rpi.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=cat02306a&AN=rlc.3566213&site=eds-live&scope=site>
- [8] T. C. Z.H. Barber, J.A. Leake, "The doitpoms project - a web-based initiative for teaching and learning materials science," *J. Mater. Educ.*, vol. 29, no. 1-2, pp. 7–16, 2003.
- [9] E. Keogh and A. Mueen, "Encyclopedia of machine learning." Boston, MA: Springer US, 2010, ch. Curse of Dimensionality, pp. 257–258, [Online]. Available: [http://dx.doi.org/10.1007/978-0-387-30164-8\\_192](http://dx.doi.org/10.1007/978-0-387-30164-8_192)
- [10] R. M. Haralick, K. Shanmugam, and I. H. Dinstein, "Textural features for image classification," *Systems, Man and Cybernetics, IEEE Transactions on*, no. 6, pp. 610–621, 1973.
- [11] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 2005, vol. 1, pp. 886–893.

- [12] T. Ojala, M. Pietikäinen, and T. Mäenpää, “Multi-resolution gray-scale and rotation invariant texture classification with local binary patterns,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.
- [13] M.-K. Hu, “Visual pattern recognition by moment invariants,” *IRE Transactions on Information Theory*, vol. 8, no. 2, pp. 179–187, 1962.
- [14] N. Otsu, “A threshold selection method from gray-level histograms,” *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.
- [15] A. Khotanzad and Y. H. Hong, “Invariant image recognition by zernike moments,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 5, pp. 489–497, 1990.
- [16] W. Lievers and A. Pilkey, “An evaluation of global thresholding techniques for the automatic image segmentation of automotive aluminum sheet alloys,” *Materials Science and Engineering: A*, vol. 381, no. 1?2, pp. 134 – 142, 2004, [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0921509304003703>
- [17] J. Yang, Y.-G. Jiang, A. G. Hauptmann, and C.-W. Ngo, “Evaluating bag-of-visual-words representations in scene classification,” in *Proceedings of the international workshop on multimedia information retrieval*, 2007, pp. 197–206.
- [18] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in *Computer vision-ECCV 2006*. Springer, 2006, pp. 404–417.
- [19] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick *et al.*, “Caffe: Convolutional architecture for fast feature embedding,” *arXiv preprint arXiv:1408.5093*, 2014.
- [20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105, [Online]. Available: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>
- [21] K. Klinder, Ed., *Machine Learning: Concepts, Methodologies, Tools and Applications*. IGI Global, 2012, vol. 1.
- [22] P. Indyk and R. Motwani, “Approximate nearest neighbors: towards removing the curse of dimensionality,” in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998, pp. 604–613.
- [23] G. H. Duntzman, *Principal Components Analysis*. Sage Publications, Inc., 1989, vol. 69.
- [24] R. G. Lomax and D. L. Hahs-Vaughn, *Statistical concepts: A second course*. Routledge, 2013.

- [25] H. Liu and R. Setiono, “Chi2: Feature selection and discretization of numeric attributes,” in *In Proceedings of the Seventh International Conference on Tools with Artificial Intelligence*, 1995, pp. 388–391.
- [26] R. Kohavi and G. H. John, “Wrappers for feature subset selection,” *Artificial intelligence*, vol. 97, no. 1, pp. 273–324, 1997.
- [27] M. A. Hall, “Correlation-based feature selection for machine learning,” Ph.D. dissertation, The University of Waikato, 1999.
- [28] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf, “Support vector machines,” *Intelligent Systems and their Applications, IEEE*, vol. 13, no. 4, pp. 18–28, 1998.
- [29] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [30] T. Denoeux, “A k-nearest neighbor classification rule based on dempster-shafer theory,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 25, no. 5, pp. 804–813, 1995.
- [31] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, “Boosting the margin: A new explanation for the effectiveness of voting methods,” *Annals of statistics*, pp. 1651–1686, 1998.
- [32] I. Rish, “An empirical study of the naive bayes classifier,” in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, 2001, vol. 3, no. 22, pp. 41–46.
- [33] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, “Supervised machine learning: A review of classification techniques,” 2007.
- [34] P. C. Pattan, V. D. Mytri, and P. S. Hiremath, “Classification of cast iron based on graphite grain morphology using neural network approach,” pp. 75 462S–75 462S–6, 2010, [Online]. Available: <http://dx.doi.org/10.1117/12.853286>
- [35] P. Prakash, V. Mytri, and P. Hiremath, “Comparative analysis of spectral and spatial features for classification of graphite grains in cast iron,” *Sci. Technol.*, vol. 29, pp. 31–40, 2011.
- [36] A. Kumara, V. Sundararaghavan, M. DeGraefb, and L. Nguyenb, “A markov random field approach for microstructure synthesis,” *Modelling Simul. Mater. Sci. Eng.*, vol. 24, no. 035015, pp. 1–13, 2016.
- [37] J. MacSleyn, J. Simmons, and M. D. Graef, “On the use of 2-d moment invariants for the automated classification of particle shapes,” *Acta Materialia*, vol. 56, no. 3, pp. 427 – 437, 2008, [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1359645407006702>
- [38] J. V. Sluytman and T. Pollock, “Optimal precipitate shapes in nickel-base ???? alloys,” *Acta Materialia*, vol. 60, no. 4, pp. 1771 – 1783, 2012, [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1359645411008706>

- [39] G. Impoco and L. Tuminello, “Incremental learning to segment micrographs,” *Computer Vision and Image Understanding*, vol. 140, pp. 144 – 152, 2015, [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1077314215000533>
- [40] R. Bostanabad, A. T. Bui, W. Xie, D. W. Apley, and W. Chen, “Stochastic microstructure characterization and reconstruction via supervised learning,” *Acta Materialia*, vol. 103, pp. 89 – 102, 2016, [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1359645415007259>
- [41] W. Z. Taffese, E. Sistonen, and J. Puttonen, “Caprm: Carbonation prediction model for reinforced concrete using machine learning methods,” *Construction and Building Materials*, vol. 100, pp. 70 – 82, 2015, [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0950061815304165>
- [42] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, “Deep learning for visual understanding: A review,” *Neurocomputing*, vol. 187, pp. 27 – 48, 2016, [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0925231215017634>
- [43] J. Yang, Y.-G. Jiang, A. G. Hauptmann, and C.-W. Ngo, “Evaluating bag-of-visual-words representations in scene classification,” in *Proceedings of the international workshop on Workshop on multimedia information retrieval*, 2007, pp. 197–206.
- [44] T. Heimann and H.-P. Meinzer, “Statistical shape models for 3d medical image segmentation: a review,” *Medical image analysis*, vol. 13, no. 4, pp. 543–563, 2009.
- [45] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [46] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 2009, pp. 248–255.
- [47] M. J. Gerdes, C. J. Sevinsky, A. Sood, S. Adak, M. O. Bello, A. Bordwell *et al.*, “Highly multiplexed single-cell analysis of formalin-fixed, paraffin-embedded cancer tissue,” *Proceedings of the National Academy of Sciences*, vol. 110, no. 29, pp. 11 982–11 987, 2013.
- [48] T. Huang, G. Yang, and G. Tang, “A fast two-dimensional median filtering algorithm,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, no. 1, pp. 13–18, 1979.
- [49] R. M. Haralick, “Statistical and structural approaches to texture,” *Proceedings of the IEEE*, vol. 67, no. 5, pp. 786–804, 1979.
- [50] R. M. Haralick, K. Shanmugam *et al.*, “Textural features for image classification,” *IEEE Transactions on systems, man, and cybernetics*, no. 6, pp. 610–621, 1973.

- [51] J. Platt *et al.*, “Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods,” *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.
- [52] L. D. Harrison, “The validity of self-reported data on drug use,” *Journal of Drug Issues*, vol. 25, no. 1, pp. 91–111, 1995.
- [53] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel *et al.*, “Scikit-learn: Machine learning in python,” *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.
- [54] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler, “Yale: Rapid prototyping for complex data mining tasks,” in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 935–940.
- [55] A. Spark, “Apache spark: Lightning-fast cluster computing,” *URL <http://spark.apache.org>*, 2016.
- [56] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.
- [57] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” in *Advances in neural information processing systems*, 2012, pp. 2951–2959.
- [58] M. T. Ribeiro, S. Singh, and C. Guestrin, “Model-agnostic interpretability of machine learning,” *arXiv preprint arXiv:1606.05386*, 2016.
- [59] F. Doshi-Velez and B. Kim, “Towards a rigorous science of interpretable machine learning,” 2017.
- [60] F. Hutter, H. H. Hoos, and K. Leyton-Brown, “Sequential model-based optimization for general algorithm configuration,” in *International Conference on Learning and Intelligent Optimization*, 2011, pp. 507–523.
- [61] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyper-parameter optimization,” in *Advances in neural information processing systems*, 2011, pp. 2546–2554.
- [62] J. Močkus, “On bayesian methods for seeking the extremum,” in *Optimization Techniques IFIP Technical Conference*, 1975, pp. 400–404.
- [63] C. Bilgin, C. Demir, C. Nagi, and B. Yener, “Cell-graph mining for breast tissue modeling and classification,” in *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, 2007, pp. 5311–5314.
- [64] C. Gunduz, B. Yener, and S. H. Gultekin, “The cell graphs of cancer,” *Bioinformatics*, vol. 20, no. suppl\_1, pp. i145–i151, 2004.

- [65] A. Chowdhury, E. Kautz, B. Yener, and D. Lewis, “Image driven machine learning methods for microstructure recognition,” *Computational Materials Science*, vol. 123, pp. 176–187, 2016.
- [66] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues *et al.*, “Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning,” *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.
- [67] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [68] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.