12-2021

Local feature selection for multiple instance learning with applications.

Aliasghar Shahrjooihaghighi
University of Louisville

LOCAL FEATURE SELECTION FOR MULTIPLE INSTANCE LEARNING WITH APPLICATIONS

By

Aliasghar Shahrjooihaghighi
M.Sc., Computer Engineering, IAU, Iran, 2011
B.Sc., Computer Engineering, Shahid Chamran University of Ahvaz, Iran, 2006

A Dissertation
Submitted to the Faculty of the
J.B. Speed School of Engineering of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy
in Computer Science and Engineering

Department of Computer Science and Engineering
University of Louisville
Louisville, Kentucky

December 2021

LOCAL FEATURE SELECTION FOR MULTIPLE INSTANCE LEARNING WITH APPLICATIONS

By

Aliasghar Shahrjooihaghighi
M.Sc., Computer Engineering, IAU, Iran, 2011
B.Sc., Computer Engineering, Shahid Chamran University of Ahvaz, Iran, 2006

A Dissertation Approved On

November 19, 2021

by the following Dissertation Committee:

Hichem Frigui, Ph.D., Dissertation Director

Xiang Zhang, Ph.D.

Olfa Nasraoui, Ph.D.

Juw Won Park, Ph.D.

Hui Zhang, Ph.D.

DEDICATION

To my mother,

for her endless love, encouragement, sacrifices, and support.

ACKNOWLEDGEMENTS

I am incredibly thankful to my advisor, Dr. Hichem Frigui, for allowing me to pursue my doctoral studies under his guidance. He is a great mentor who worked with me side by side, and guided me through my research work. He inspired me with his broad knowledge and creativity. This thesis would not have been possible without his guidance and support.

I would like to thank Dr. Olfa Nasraoui, Dr. Juw Won Park, Dr. Hui Zhang, and Dr. Xiang Zhang for agreeing to serve on my dissertation committee and providing invaluable feedback. A very special thanks to Dr. Xiang Zhang for his support, guidance, and mentorship. I would also express my gratitude to Dr. Adel Elmaghraby for his support and kindness.

Many thanks to my friends at the University of Louisville, especially my friends and colleagues in the Multimedia Research Laboratory for their inspiration, motivation, and friendship.

Finally and most importantly, I am infinitely grateful to my family, my mother, for all her unconditional love, support, understanding, and sacrifices she made for the family, and my father, who never saw this adventure. Endless gratitude is for my sisters, for all their encouragement, love, and support. I could not make it without them. Thank you all.

ABSTRACT

LOCAL FEATURE SELECTION FOR MULTIPLE INSTANCE LEARNING WITH APPLICATIONS

Aliasghar Shahrjooihaghighi

November 19, 2021

Feature selection is a data processing approach that has been successfully and effectively used in developing machine learning algorithms for various applications. It has been proven to effectively reduce the dimensionality of the data and increase the accuracy and interpretability of machine learning algorithms. Conventional feature selection algorithms assume that there is an optimal global subset of features for the whole sample space. Thus, only one global subset of relevant features is learned. An alternative approach is based on the concept of Local Feature Selection (LFS), where each training sample can have its own subset of relevant features.

Multiple Instance Learning (MIL) is a variation of traditional supervised learning, also known as single instance learning. In MIL, each object is represented by a set of instances, or a bag. While bags are labeled, the labels of their instances are unknown. The ambiguity of the instance labels makes the feature selection for MIL challenging. Although feature selection in traditional supervised learning has been researched extensively, there are only a few methods for the MIL framework. Moreover, localized feature selection for MIL has not been researched.

This dissertation focuses on developing a local feature selection method for

the MIL framework. Our algorithm, called Multiple Instance Local Salient Feature Selection (MI-LSFS), searches the feature space to find the relevant features within each bag. We also propose a new multiple instance classification algorithm, called MILES-LFS, that integrates information learned by MI-LSFS during the feature selection process to identify a reduced subset of representative bags and instances. We show that using a more focused subset of prototypes can improve the performance while significantly reducing the computational complexity.

Other applications of the proposed MI-LSFS include a new method that uses our MI-LSFS algorithm to explore and investigate the features learned by a Convolutional Neural Network (CNN) model; a visualization method for CNN models, called Gradient-weighted Sample Activation Map (Grad-SAM), that uses the locally learned features of each sample to highlight their relevant and salient parts, and a novel explanation method, called Classifier Explanation by Local Feature Selection (CE-LFS), to explain the decisions of trained models.

The proposed MI-LSFS and its applications are validated using several synthetic and real data sets. We report and compare quantitative measures such as Rand Index, Area Under Curve (AUC), and accuracy. We also provide qualitative measures by visualizing and interpreting the selected features and their effects.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In the past decade, the increasing trend of high-dimensional data in various domains such as social media, health care, bioinformatics, and image processing raised new challenges in the effectiveness of data processing, data management, and the application of machine learning algorithms. One technique to reduce the drawbacks of the high dimensionality on machine learning algorithms is to integrate feature selection within the learning process. Feature selection seeks to find the most informative features in the data space by eliminating redundant and non-informative features. It can lead to a better understanding of the data, reducing the complexity of the learning algorithms and the data storage, and improving the generalization of the prediction models [4].

1.1 Feature Selection

Several feature selection algorithms have been proposed and applied successfully to various applications and domains within the machine learning community. Good reviews of feature selection algorithms can be found in [4–8]. Traditional feature selection algorithms strive to identify a global set of features that represents the behavior of the given data. They search the entire sample space to learn one opti-

mal subset of informative features that are common for the majority of the samples. The main limitation of this approach is that it ignores the potential variation of the relevant features across the different regions in the feature space.

Recently, in [9], the authors introduced a novel concept of localized feature selection by considering each sample of the training set as a representative of its neighboring region. In this approach, one subset of informative features is identified for each sample.

1.2 Multiple Instance Learning

Multiple Instance Learning (MIL) is a variation of supervised learning algorithms where each object is represented by a set of feature vectors, called a bag. Each feature vector is called an instance. In MIL, the bags are labeled, but the label of the instances are unknown. A bag is labeled negative if it contains only negative instances, and positive if it contains at least one positive instance. In Multiple Instance Classification (MIC) learning problems, we learn a model using the training set of bags to predict the label of unseen bags [10]. MIL has been actively studied since many applications can be better formulated using the multiple instance learning framework. Examples of such applications include drug discovery, classification of text documents, classification of images, speaker recognition, and bankruptcy prediction [10].

The characteristics of MIL problems, especially having ambiguous instance labels inside the bags, make the feature selection process in MIC applications more complex, challenging, and needed than the conventional classification problems. More-

over, while feature selection in conventional learning has been studied extensively, there are only few feature selection algorithms proposed for MIL [11–18]. Thus, feature selection in MIL is an area that can benefit from more research.

1.3 Contributions

1.3.1 Local Feature Selection for MIL

In this thesis, we address the feature selection problem in multiple instance learning using the concept of local feature selection. Unlike traditional feature selection methods where a global subset of features is assigned to the whole data set, the aim of local feature selection is to find a subset of features for each bag that represents its local behavior. To tackle this problem, we investigate and propose a feature selection method for MIL, called Multiple Instance learning Localized Salient Feature Selection (MI-LSFS). Our approach is an extension of the recently proposed logistic Localized Feature Selection (lLFS) [9] to multiple instance data. MI-LSFS learns a subset of features for each bag instead of learning one global set of selected features for the whole data set. To the best of our knowledge, this is the first approach that addresses localized feature selection for MIL.

To design the algorithm, we adapt a measure of distance for the non-vertical entities (bags) that allows comparing bags and is compatible with the assumptions of the distance function used in the lLFS (1-norm distance). In particular, we use the minimal Hausdorff distance [19]. Moreover, to evaluate the clustering performance during the learning process, we combine impurity with Matthews Correlation Coeffi-

cient (MCC) [20]. Adding the MCC term allows us to control the performance of the clustering by considering a balanced measure instead of using only the true positive values and false positive.

1.3.2 Applications of MI-LSFS

As MI-LSFS assigns a subset of features for each bag, common MIL classifiers cannot interpret and take advantage of this information. Therefore, we propose a new MIL classifier that uses local information provided by MI-LSFS. We call our proposed classifier MILES-LFS as it uses some techniques used in the MILES [21] algorithm. More specifically, we designed a classifier that uses the local information learned by MI-LSFS during the feature selection process to perform the classification task. This information includes the set of selected features for each bag, a filtering strategy for the irrelevant instances, and performing the instance selection task. The instance selection strategy decreases the dimensionality of the data set in the learning process and reduces the learning time. Therefore, our proposed method can be used as a feature selection and an instance selection. Using the locally selected informative features, we show that MI-LSFS leads to improving the prediction performance and explainability. Furthermore, we show that MI-LSFS can guide the learning algorithms in discovering the underlying information for the local regions. This underlying information can be helpful in interpreting the results of the learning algorithms.

Convolutional Neural Networks (CNN) have been successfully applied and adapted to many applications, especially in the computer vision domain. Despite the widespread usage of the CNNs, in classification or as feature extractors, they are

often being criticized for working as a black box. In general, black box models are insufficient in interpreting the learned features and explaining the results. To address this limitation, we propose an explanation method for the decisions of a CNN model. In particular, we use our MI-LSFS algorithm to explore and investigate the features learned by a CNN model.

Interpreting the features learned by CNN models are often done by using scientific visualization methods. Typically, the visualization methods highlight the salient segments of the input that are highly correlated to the predicted class. As another application of our proposed localized feature selection, we propose a visualization method, called Gradient-weighted Sample Activation Map (Grad-SAM), that integrates the locally learned features and highlights the relevant and salient parts of each sample.

Another application of our MI-LSFS that we propose is to explain the decisions made by trained machine learning models. We call our proposed approach Classifier Explanation by Local Feature Selection (CE-LFS). In particular, we use the set of locally salient features of each sample, to describe why the trained model classifies a given sample correctly/incorrectly. The ability to explain the decision of a classifier is critical in using these methods in real applications.

To summarize, the main contributions of this thesis include:

1. A local feature selection method for MIL, called Multiple Instance learning Localized Salient Feature Selection (MI-LSFS). To the best of our knowledge, this is the first approach that addresses localized feature selection for multiple instance data.

2. A new multiple instance classification algorithm, MILES-LFS, that uses the local information provided by MI-LSFS.

3. A new method that uses our MI-LSFS algorithm to explore and investigate the features learned by a CNN model.

4. A visualization method for CNN models, called Gradient-weighted Sample Activation Map (Grad-SAM), that uses the locally learned features of each sample to highlight their relevant and salient parts.

5. A novel explanation method, called Classifier Explanation by Local Feature Selection (CE-LFS), to describe the decisions of trained models.

1.4 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 provides a review of concepts that are highly relevant to our work. This includes feature selection, local feature selection, multiple instance learning, feature selection methods in MIL, and deep learning models (i.e., CNNs). Chapter 3 describes the proposed MIL feature selection method, MI-LSFS[1]. Chapter 4 presents our proposed applications of MI-LSFS, including a new MIL classification method called MILES-LFS, our CNN explainability method, our visualization method, called Grad-SAM, and our classifier explanation method called (CE-LFS). Chapter 5 evaluates the performance of our proposed methods and presents experimental results. Finally, chapter 6 concludes by summarizing and recommending some future research directions.

[1]This paper was published in [22].

CHAPTER 2

BACKGROUND

In this chapter, we review the literature and methods that are relevant to our proposed work. First, we review feature selection methods for the standard single instance learning approach. Then, we describe the Multiple Instance Learning framework. Next, we review literature related to feature selection in Multiple Instance Learning, and we provide a detailed description of local feature selection methods and describe the lLFS [9] algorithm. Finally, we review deep learning methods with a focus on Convolutional Neural Networks' (CNN) architectures, and we overview two common visualization techniques for CNNs.

2.1 Feature Selection Methods

Feature selection is an important step in machine learning, especially when dealing with high dimensional data (i.e., each sample is represented by a large number of features). Feature selection methods reduce the dimensionality of data by selecting a subset of the most discriminating features. This helps subsequent learning algorithms focus on the features that have maximum effect on the prediction models, improve the accuracy of predicting the samples' category, and reduce the time complexity of the algorithms [23].

2.2 Feature Selection Taxonomy

There are three main categories of feature selection methods [4, 24]. Filter methods [25–27] assign a score (or a rank) to the features by using a measure indicating their discriminating power. Features with low scores will be eliminated. Filter methods are fast and independent of the prediction models. Unlike the filter methods, wrapper methods depend on the algorithm used for the prediction [28–30]. They strive to find the optimal combination of features that maximizes the prediction's accuracy. The wrapper methods are typically computationally more expensive than the filter methods. The third category of feature selection methods are based on embedded methods. These methods are also dependent on the classifiers, but with a lower computational complexity. They interact with the classification method in the training phase and they use internal information from the classification method [5, 31, 32].

In [33], Chandrashekar et al. provide an introduction to feature selection techniques and compare multiple feature selection algorithms. Their objective was to provide a generic introduction to variable elimination. They compare the performance of various feature selection methods on 7 data sets using SVM and neural networks classifiers.

In [34], Miao et al. reviewed several feature selection methods. They performed experiments to check if feature selection can increase the performance of learning using 12 real life data sets, including three microarray data sets.

In [35], Cai et al. surveyed and reviewed feature selection methods and evaluation measures for machine learning applications. They reviewed the methods used

in several applications such as image recognition, image retrieval, bioinformatics data analysis. Their analysis concluded that the ensemble methods are especially useful when the data contain small number of samples.

2.2.1 Ensemble Methods for Feature Selection

Ensemble learning combines the output of multiple algorithms to take advantages of their strengths and overcome their weaknesses. There is an increasing trend of using ensemble methods in applications such as sequence analysis, microarray analysis and mass spectra analysis due to the small number of samples and large number of features. A review of ensemble methods that have been applied to bioinformatics can be found in [36].

Several studies have concluded that ensemble feature selection methods can improve the stability of algorithms, give a better approximation of optimal ranking of features and lead to more robust feature selection [31,36–42]. For instance, in [43], the authors provided a review of the stability of feature selection techniques for biomarker discovery. They concluded that ensemble learning can improve the stability of the discovered biomarkers.

2.3 Local Feature Selection

Conventional feature selection algorithms assume that there is a global subset of features that is optimal for the whole sample space. This optimal subset may be learned by considering the local behavior of the samples. However, only one global subset of relevant features [9] is learned. Examples of feature selection methods that

use the local information of samples during the learning process are based on the Relief feature selection algorithm [44]. These methods strive to capture the local structure of the data by setting the margins locally for each region of interest in the randomly selected samples. In [45], the authors proposed a feature selection algorithm for classification. The main idea is to first decompose an arbitrary complex nonlinear problem into a set of locally linear ones through local learning. Next, feature relevance weights are learned globally within the margin framework.

An alternative approach is based on the concept of Local Feature Selection (LFS) [46]. LFS overcomes the limitations of conventional feature selection methods by adapting to the statistical variations across the different regions of the sample space. It considers each training sample as a representative of the sample space in a local region. Therefore, different subsets of features will be selected for different regions instead of using one common subset for the whole sample space. The logistic Localized Feature Selection (lLFS) algorithm [9], which is a variation of LFS, is outlined in the following section.

2.4 Logistic Localized Feature Selection Method (lLFS)

The goal of the lLFS is to find an optimal feature subset for each sample by considering each sample as a representative of a local region in the sample space. Let $\{S_1, ..., S_N\}$ be the N training data samples where $S_i \in \mathbb{R}^M$ and M is the dimensionality of the original sample space. Let $f^{(i)} \in \{0,1\}^M$ indicate the selected features for the region centered at sample S_i. Let $f_m^{(i)}$ denote the selection status of the m^{th} feature of S_i. If the m^{th} feature of S_i is selected, then $f_m^{(i)}$ is set to 1, otherwise it is

set to 0. Let $S_p^{(i)}$ denote the projection of S_i onto the subspace of selected features represented by $f^{(i)}$. The goal is to find $f^{(i)}$ for S_i such that the projection of all the samples onto $f^{(i)}$ minimizes the intra-class distances and maximizes the inter-class distances simultaneously.

Let $D(S_i, S_j)$ denote the distance between two given samples S_i and S_j using all M features, and let $D(S_i, S_j, f^{(i)})$ denote the distance between the projection of samples S_i and S_j onto the subspace represented by $f^{(i)}$. To learn $f^{(i)}$, the authors in [9] define and optimize two objective functions. The first one is defined using the distance between sample i and all other samples that have the same label as sample i:

$$U_1(f^{(i)}) = \frac{1}{N_i - 1} \sum_{j;y_j=y_i;j\neq i} g\left(D(S_i, S_j, f^{(i)}), \sigma^{(i)}, \lambda\right). \tag{2.1}$$

The second objective function is defined using the distance between sample i and all other samples with different labels as i:

$$U_2(f^{(i)}) = \frac{1}{N - N_i} \sum_{j;y_j\neq y_i} g\left(D(S_i, S_j, f^{(i)}), \sigma^{(i)}, \lambda\right) \tag{2.2}$$

In (2.1) and (2.2), N_i is the number samples with the same label as S_i. The other parameters, λ , $\sigma^{(i)}$, and the logistic function g will be discussed later. The objective functions U_1 and U_2 measure the intra-class distance and inter-class distance, respectively. The goal is to find a subset of features $f^{(i)}$ where the intra-class distance is minimized and the inter-class distance is maximized. The resulting $f^{(i)}$ will represent the informative features for the local region centered at S_i that keep the intra-class samples as close as possible and the inter-class samples as far as possible.

Formally, using the two objective functions in (2.1) and (2.2), the optimization process can be formulated as:

$$\begin{cases} \min_{f^{(i)}} U_1(f^{(i)}) \\ \max_{f^{(i)}} U_2(f^{(i)}) \\ s.t. \begin{cases} f^{(i)} \in \{0,1\}^M \\ 1 \leq 1^T f^{(i)} \leq \alpha \end{cases} \end{cases} \tag{2.3}$$

By restricting $1^T f^{(i)}$ between 1 and α, the number of selected features is set to be at least 1 and at most α features.

In (2.1) and (2.2), the logistic function g is defined as:

$$g(z, \sigma, \lambda) = \frac{1}{1 + \exp(-\sigma z)} - 0.5 + \lambda z. \tag{2.4}$$

The role of this function is to transform the distance between the samples such that closer samples will have more influence on the objective function and dominate the feature selection process. Similarly, farther samples will have a minimal effect on the selection of $f^{(i)}$. In (2.4), the purpose of the linear term λz is to give a chance to the potentially relevant samples that are far from $S_p^{(i)}$ to get closer to $S_p^{(i)}$ in subsequent iterations of the optimization process.

2.4.1 Optimization

The optimization problem in (2.3) is a multi-objective optimization problem. One way to solve it is to use the Pareto optimality concept [47]. Using this strategy, the objective functions in (2.3) are combined to form the following single objective function:

$$\begin{cases} \min_{f_\beta^{(i)}} U_1(f_\beta^{(i)}) \\ s.t. \begin{cases} f_{m,\beta}^{(i)} \in [0,1] \\ 1 \leq 1^T f^{(i)} \leq \alpha \\ U_2(f_\beta^{(i)}) \geq \beta \epsilon_{max}^{(i)}. \end{cases} \end{cases} \tag{2.5}$$

In (2.5), β is a set of constraint values, in the range of [0,1], that control the number of Pareto points, $f_\beta^{(i)}$ represents the selected features of S_i for a given β, and $\epsilon_{max}^{(i)}$ is the maximum feasible value of U_2 for the given sample S_i that is computed using:

$$\begin{cases} \epsilon_{max}^{(i)} = \max_{f^{(i)}} U_2(f^{(i)}) \\ s.t. \begin{cases} 0 \leq f_m^{(i)} \leq 1, m = 1, ..., M \\ 1 \leq 1^T f^{(i)} \leq \alpha. \end{cases} \end{cases} \tag{2.6}$$

In other words, (2.5) finds a solution for the optimization problem for each value of β. In [9], the authors discussed the feasibility of this approach in details. A solution of (2.5), $f_\beta^{(i)}$, is a relaxed solution where its elements are within the continuous range [0,1]. Therefore, a randomized rounding process [47] can be applied to $f_\beta^{(i)}$ to convert it into a binary solution. Let $f_\beta^{*(i)}$ denote the binary solution where each element represents the selection status of its corresponding feature (i.e., 1 if the feature is selected and 0 if it is not).

2.4.2 Using Local Clustering to Find the Optimal β

The randomized rounding process generates multiple binary solutions for each sample S_i (i.e., $f_\beta^{*(i)}, \beta \in [0,1]$). The optimal solution, denoted $f^{*(i)}$, is selected based on the clustering performance in the local region using the training set.

To evaluate the clustering performance of each $f_\beta^{*(i)}$, the authors in [9] used the impurity measure. Impurity is defined as the ratio of the number of samples with different class labels to the number of samples with the same class label. For a given sample S_i, lLFS searches for a radius $r^{(i)}$ such that the impurity level inside the hypersphere, centered at S_i with radius $r^{(i)}$, is less than a fixed threshold (e.g., 0.2). Then, the feature set with the best clustering performance on the training set is selected as $f^{*(i)}$.

2.4.3 Parameters of the Logistic Function

The logistic function in (2.4) has two parameters that need to be assigned in the optimization process: λ and $\sigma^{(i)}$. The parameter $\sigma^{(i)}$ is assigned based on the initialization value of the selected features $f_\beta^{(i)}$. It is calculated based on the point from the S_i that sits on the knee point of the logistic function [9]. Formally, $\sigma^{(i)}$ is calculated using (2.7) where the value of 0.47 is set based on the knee point of the logistic function in (2.4).

$$\begin{cases} \frac{1}{1+\exp(-\sigma^{(i)}\varphi^{(i)})} - 0.5 = 0.47 \\ \varphi^{(i)} = \max_{j=1..N, j\neq i}\{D(S_i, S_j, f^{(i)})\}, \end{cases} \tag{2.7}$$

In (2.7), λ is set to the default value of $0.01/\alpha$ at the initialization step. The lLFS algorithm is outlined in Algorithm 1.

Algorithm 1: Logistic Localized Feature Selection

Input:

S : Samples

Y : labels

N : Number of Samples

M : Number of features

α : Controls the maximum number of features to be selected

Output:

$\{f^{*(i)}\}, i = 1, ..., N$: selected features for each sample

$f_\beta^{(i)} = \frac{1}{\alpha}(1, ..., 1), i = 1, ..., N$

$\lambda = \frac{0.01}{\alpha}$

$\beta = [0, 1]$

for $i = 1$ **to** N **do**

  Compute $\sigma^{(i)}$ by solving (2.7);

  Compute $\epsilon_{max}^{(i)}$ by solving (2.6);

  for $\beta = 0$ **to** 1 **do**

    Compute $f_\beta^{(i)}$ by solving (2.5);

    Randomized rounding process of $f_\beta^{(i)}$ to obtain binary feature vector $f_\beta^{*(i)}$;

  set $f^{*(i)}$ equal to the member of $f_\beta^{*(i)}, \beta \in [0, 1]$ which yields the best local clustering performance using a leave one out cross validation approach in the local region;

2.5 Multiple Instance Learning

In conventional supervised learning problems, each object is represented by a fixed length feature vector and each training feature vector is assigned a label. Multiple Instance Learning (MIL) is a variation of supervised learning algorithms where each object is represented by a bag. Each bag includes a set of feature vectors (instances) [48]. For the training data, labels are assigned only at the bag level and

instances within the bag are unlabeled.

Let $D = \{< B_i, y_i >\}_{i=1}^{N}$ denote the training data where B_i and y_i represent the i^{th} bag and its label, respectively, and N is the number of bags. Each bag $B_i = \{\vec{x}_{i,1}, \vec{x}_{i,2}, ..., \vec{x}_{i,k}\}_{k=1}^{|B_i|}$ has a set of $|B_i|$ instances where each instance, $\vec{x}_{i,k} \in \mathbb{R}^M$, is an M-dimensional feature vector.

Let y_i denote the label of B_i. In binary MIL problems, bags are labeled positive ($y_i = +1$) or negative ($y_i = -1$). The standard MIL algorithm assumption is that negative bags contain only negative instances and positive bags contain at least one positive instance. Formally, let $y_{i,k}$ denote the label of instances inside B_i and y_i denote the label of B_i.

$$y_i = \begin{cases} +1 \ if \ \exists y_{i,k} : y_{i,k} = +1 \\ \\ -1 \ if \ \forall y_{i,k} : y_{i,k} = -1. \end{cases} \tag{2.8}$$

We should emphasize here that $y_{i,k}$ are unknown even for the labeled training data. Despite this ambiguity in the instances' labels, MIL has proved to be an appropriate representation for various applications such as drug discovery [49], image annotation [50], text classification [51], buried object detection [52], video concept detection [53] and cancer nodules classification [54]. In fact, for most of the above applications MIL was shown to outperform standard supervised learning methods.

2.5.1 MIL Classification

Multiple Instance learning Classification (MIC), can be performed at two levels: bag and instance. In bag-level classification, the goal is to assign class label to the bags

and the individual instance labels are not needed. These methods cannot perform instance classification [55–57]. On the other hand, in the instance level classification, the objective is to assign labels to the individual instances [57], and instances' labels are used to the bag labels. Thus, the loss functions for the two classification tasks are different [58]. We should emphasize here that the performance of a MIL classification algorithm is typically evaluated at the bag level since instance labels are unknown.

An example of instance level classification is the mi-SVM algorithm [51] where the instance initial labels are inherited from the bag labels. Then, the SVM starts the training using the instance labels and continues training using the new labels assigned by the learned SVM model until the instances labels do not change. The trained SVM classifier is then used to predict the class of instances of new test bags. The authors in [51] also proposed MI-SVM algorithm which is an alternative approach of applying maximum margin concept to the MIL where the objective is to maximize the bag margin. In MI-SVM, the bag margin is defined by the most positive instance of each positive bag. In other words, only one pattern per positive bag matters, since it will determine the margin of the bag.

The Diverse Density (DD) algorithm [49] is another common MIL approach. The DD optimizes an objective function to find a soft region that maximizes instances from positive bags and minimizes instances from negative bags. The EM-DD [55], uses the Expectation-Maximization (EM) algorithm to maximize the DD objective function. SI-SVM [59], RSIS [60], and MIL-Boost [61] are other examples of instance level algorithms [4].

MIGraph and miGraph [62] are two bag-level algorithms proposed for MIL.

They treat the instances in a non independently and identically distributed (i.i.d.) way. The idea is that the instances of a bag are rarely independent, and a better performance can be expected if the instances are treated in an non-i.i.d. way where the relation between instances are considered. In MIGraph, every bag is mapped to an undirected graph. Then, a graph kernel is designed to distinguish between positive and negative bags. In miGraph, graphs are constructed by deriving affinity matrices and an efficient graph kernel that considers the clique information is proposed. Citation-kNN [19], MILES [21], and EMD-SVM [63] are some other bag-level algorithms. Comparative studies of several MIL algorithms can be found in [10,59,64,65].

2.5.1.1 Citation k-Nearest Neighbors

The Citation k-Nearest Neighbors (Citation-kNN) [19] is an adaptation of the k-Nearest Neighbors (kNN) classifier for MIL. In the standard kNN, to classify a query sample, a distance function is used to find the k nearest neighbors to the query sample. Then, the query sample label is computed using the label of the nearest neighbors. The Citation-kNN follows the kNN steps but using a two-level voting approach inspired from the notion of citations and references in library and research papers. Specifically, a new test bag, is labeled based on the labels of its neighboring bags (references) and the labels of the bags that consider the test bag as one of their kNN neighbors (citers).

Formally, Citation-kNN uses the Hausdorff distance to compute the distance between two given bags, B_1 and B_2, of instances $\{x_{1,j}\}_{j=1}^{m_1}$ and $\{x_{2,j}\}_{j=1}^{m_2}$, respectively.

The Hausdorff distance between B_1 and B_2 is defined as:

$$\mathbf{H}(B_1, B_2) = \min_{x_{1,j} \in B_1, x_{2,j} \in B_2} \big\{ dist(x_{1,j}, x_{2,j}) \big\}, \tag{2.9}$$

where *dist* is a distance measure between instances (e.g., Euclidean distance).

Let $Rank(B_1, B_2)$ equal n if B_2 is the nth nearest neighbor of B_1. For the sake of completeness, the authors in [19] set $Rank(B_1, B_1)$ to ∞. The *C-nearest citers* of B are the C bags that return the lowest neighbor ranking for B, i.e.,

$$Citers(B, C) = \{ B_i \mid Rank(B_i, B) \leq C,\ B_i \in \mathcal{B} \}, \tag{2.10}$$

where $\mathcal{B}$ is the set of all training bags.

The decision of the Citation-kNN relies not only on the neighbors of bag B (references), but also on the bags that count B as a neighbor (citers). In other words, given a bag B, its K-nearest references and C-nearest citers are identified using (2.9) and (2.10). Then, B is labeled based on the label of its references and citers. Thus, B is classified as positive if there more positive bags than negative bags in its combined K-references and C-citers. C is usually set to K+2.

2.5.1.2 MILES

The Multiple Instance Learning via Embedded Structures (MILES) [21] is a multiple instance learning classification algorithm. It converts the MIL problem into a standard supervised learning problem by mapping each bag to a feature space defined by the instances in the training bags using a similarity measure. The mapped bags are then represented as standard feature vectors in the instance feature space [66].

After mapping, 1-Norm SVM [21] is applied to construct the classifier. The main steps of MILES are discussed below.

Feature mapping: At this step, each bag B_i, is mapped to a feature space based on its similarity to a set of target concepts, $C = \{\vec{x}_1^t, \ldots, \vec{x}_j^t, \ldots, \vec{x}_{|C|}^t\}$, where $|C|$ is the number of instances in the target concept set. Let $m(B_i)$ denote the mapping of B_i defined as

$$m(B_i) = [S(\vec{x}_1^t, B_i), \ldots, S(\vec{x}_j^t, B_i), \ldots, S(\vec{x}_{|C|}^t, B_i)]. \tag{2.11}$$

In (2.11), $S(\vec{x}_j^t, B_i)$ is the similarity between bag $B_i = \{\vec{x}_{i,1}, \vec{x}_{i,2}, \ldots, \vec{x}_{i,k}\}_{k=1}^{|B_i|}$ and a concept $\vec{x}_j^t \in C$ which is determined by the closest instance in B_i to $\vec{x}_j^t$ and it is calculated using:

$$Pr(\vec{x}_j|B_i) \propto S(\vec{x}_j, B_i) = \max_k \Big\{ exp\Big(- \frac{||\vec{x}_{i,k} - \vec{x}_j^t||^2}{\sigma^2} \Big) \Big\}. \tag{2.12}$$

In 2.12, σ is a scaling factor.

MILES Training: First, MILES uses (2.12) to map the training bags and form the mapped training data. In [21], the authors assume that all of the instances within all training positive bags are in the target concept set. Thus, $C = \{\vec{x}_{i,j} | \vec{x}_{i,j} \in B_i, y_i = 1\}$. For a given training set that contains ℓ^+ positive bags and ℓ^- negative bags, the mapping representation of the data set is defined as:

$$[m(B_1^+),\ldots,m(B_{\ell^+}^+),m(B_1^-),\ldots,m(B_{\ell^-}^-)] = \begin{bmatrix} S(\vec{x}_1,B_1^+) & \ldots & S(\vec{x}_j,B_1^+) \\ \ldots & \ldots & \ldots \\ S(\vec{x}_1,B_{\ell^+}^+) & \ldots & S(\vec{x}_j,B_{\ell^+}^+) \\ S(\vec{x}_1,B_1^-) & \ldots & S(\vec{x}_j,B_1^-) \\ \ldots & \ldots & \ldots \\ S(\vec{x}_1,B_{\ell^-}^-) & \ldots & S(\vec{x}_j,B_{\ell^-}^-) \end{bmatrix} \quad (2.13)$$

In (2.13), each row represents a bag and each column represents the j^{th} feature in the mapped space.

Next, MILES apply the 1-Norm SVM [67], in the mapped feature space, to find a linear classifier that discriminates between positive and negative bags using:

$$y_i = sign\Big(\sum_{j=1}^{C} w_j * S(\vec{x}_j,B_i) + b\Big), \quad (2.14)$$

In (2.14), w_j is the weight associated with $S(\vec{x}_j,B_i)$ and b is a bias parameter.

MILES Testing: To test a new bag B_t, first MILES maps B_t by following the same approach used to map the training bags. Then, it uses the learned parameters of the 1-Norm SVM to compute the label.

2.5.2 Feature Selection for Multiple Instance Learning

Although feature selection in traditional supervised learning has been researched extensively, only few existing methods can be applied to Multiple Instance data. In the following subsections, we review these methods.

2.5.2.1 BP-MIP

In [11], the authors proposed a method to improve a MIL neural network, BP-MIP [68], by incorporating two different feature selection methods: feature scaling with Diverse Density [49] and feature reduction with principal component analysis [69]. They referred to these two variations as BP-MIP-DD and BP-MIP-PCA respectively. In particular, BP-MIP-DD uses the Diverse Density approach on the training data to learn the feature weights. Then, it scales the features using the learned weights and feeds them to the BP-MIP network. Alternatively, in BP-MIP-PCA, principle component analysis is used to project the features, using a linear transformation matrix, to a lower-dimensional feature space before feeding them to the network.

2.5.2.2 MI-Adaboost

In [12], the authors proposed an algorithm that maps each bag onto a feature vector using a set of instance prototypes. Then, an AdaBoost algorithm [70] was used to select the relevant features and learn the classifier simultaneously in the mapped feature space. More specifically, they followed [21] in considering all the instances in the positive bags as instance prototypes. Moreover, to identify few instance prototypes from the negative bags, they applied the k-means clustering [71] on instances from all negative bags. The c cluster centers are then used as instance prototypes that summarize all negative bags. Next, they followed [21] in mapping all the bags

to the new (m+c) dimensional feature vector. They used

$$d(p_t|B_i) = min\Big\{exp\Big(-\frac{||p_t - B_{i,j}||^2}{\sigma^2}\Big)\Big\}, \quad B_{i,j} \in B_i \tag{2.15}$$

to calculate the distance between a given instance p_t and a given bag B_i. In (2.15), $B_{i,j}$ represents the instances of B_i and σ^2 is a predefined constant.

Finally, a variation of Adaboost, with a weak linear classifier on the mapped feature space was used to build a classifier and learn the feature weights simultaneously. This algorithm focuses on selecting relevant features from the mapped feature vector and identifies the instances with higher representative power in the mapped feature space as the instance prototypes.

2.5.2.3 MIRVM

Rayker et al. [13], proposed a Bayesian multiple instance learning algorithm called MIRVM that identifies a subset of relevant features while learning (conceptually related) classifiers simultaneously. They use a noisy-OR model and a logistic sigmoid to calculate the probability of a target given the observed bags:

$$P(t|X) = 1 - \prod_{x \in X} \big(1 - \sigma(w^T x)\big) \tag{2.16}$$

In (2.16) the weight vector, w, is modeled using a Gamma prior. The optimization is performed using the Newton-Raphson method. Feature selection is optimized using type II maximum likelihood method.

2.5.2.4 ReliefF-MI

In [17], the authors proposed a filter feature selection approach for the MIL, called Relief-MI, which is an adaptation of the ReleifF [72] algorithm. Similar to ReliefF, ReliefF-MI randomly samples bags from the training data. Then, for each selected bag, it identifies the k nearest neighbors form the same class (nearest hits) and the other classes (nearest misses). Next, it updates the feature scores (weights) based on the distance difference between the nearest hits and nearest misses. Finally, using the feature scores, the most discriminating features are selected and other features are discarded.

In a similar fashion, HyDR-MI [18] has been adapted to MIL. HyDR-MI is a hybrid feature selection algorithm that uses ReleifF as the filter component and a genetic algorithm as the wrapper component in the learning process. The genetic algorithm is used to optimize the search for the optimal feature subset using the output of the filter component.

2.5.2.5 M3IFW

In [73], the authors adapted the maximum margin framework in the supervised feature weighting area [74,75] and proposed a maximum margin MIL feature weighting algorithm (M3IFW) to identify large classification margins in the weighted feature space. The idea was to combine the search of positive prototypes and the calculation of the weighting vector in a unified maximum margin framework.

The drawback of M3IFW is that it needs to optimize three unknown variables

(the positive prototypes, the weighting coefficients and the classification margin). To address this, the authors proposed a solution that optimizes one variable at a time while fixing the other variables. This iterative alternative optimization is repeated until the change in the objective function is less than a fixed threshold.

Similarly, in [15], the authors proposed a multiple instance feature-weighting algorithm at the bag level. They adapted the three bag-level distances (minimal Hausdorff, class-to-bag, and bag-to-bag) to design the bag-level feature weighting method.

2.5.2.6 Feature Selection based on Dimensionality Reduction

There are a few studies that use dimensionality reduction in MIL to select a subset of features [14, 76, 77]. For instance, in [14], the authors studied a dimensionality reduction algorithm using sparsity and orthogonality. The main idea was to formulate an optimization problem where the sparse term appears in the objective function and the orthogonality term is formed as a constraint. As the goal of these algorithms is to reduce the dimensionality by mapping the original features to another feature space, they are different from our focus which is feature selection in the original feature space.

2.6 Deep Learning

Representation Learning is a set of techniques that allow the machine to automatically discover the representations used for detection or classification from the raw data [78]. Deep learning methods are a type of representation learning based on

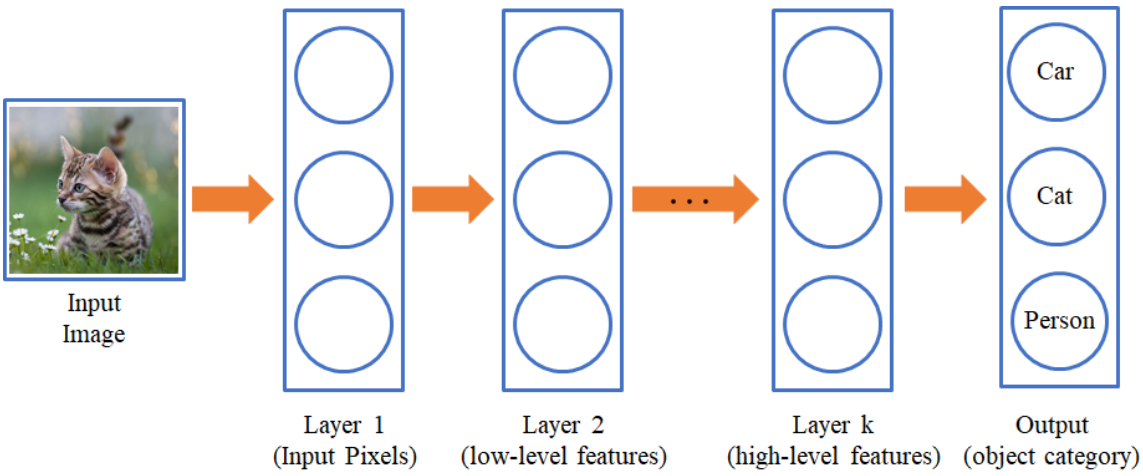


Figure 2.1: Illustrations of a deep learning model

artificial neural networks. Deep learning finds abstract representations by using simpler representations through successive layers of the network. In other words, deep learning builds complex concepts out of simpler concepts [79].

Figure 2.1 illustrates how a deep neural network can learn the features to represent the concept of an image. At the top layers, the model learns primitive low-level features. For instance, in the first layer the model starts with raw pixels of the image. In deeper layers of the network, higher level features are learned by combining the simpler features learned in the earlier layers. For instance, in the second layer, the model may be learning the edges of the input image and, in the third layer, more abstract features (e.g., eye or ear of the cat) can be extracted. The output layer uses a weighted combination of the high-level features to detect the category of the input image.

Deep learning methods have made advances in many domains specially the ones with high dimensional data such as computer vision [80–84], speech recognition

[85–88], drug discovery [89,90], bioinformatics [91,92], medical image analysis [93,94], and natural language processing [95–97].

2.6.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a type of neural network inspired by cells in visual neuroscience [98]. CNNs have their roots in the neocognitron network proposed by Fukushima and Miyake [99], which was inspired by the discovery of cells in the visual cortex that are responsible for detecting light in receptive fields [100].

CNNs are designed for processing data in form of multiple arrays such as a color image that contains three 2D arrays where each array represents a color channel (RGB) [78]. A CNN uses convolution, a specialized type of linear filtering, to extract visual features.

A typical architecture of a CNN is composed of a stack of convolutional layers, activation functions, pooling layers, and fully-connected layers and is illustrated in figure 2.2 and outlined in the following subsections.

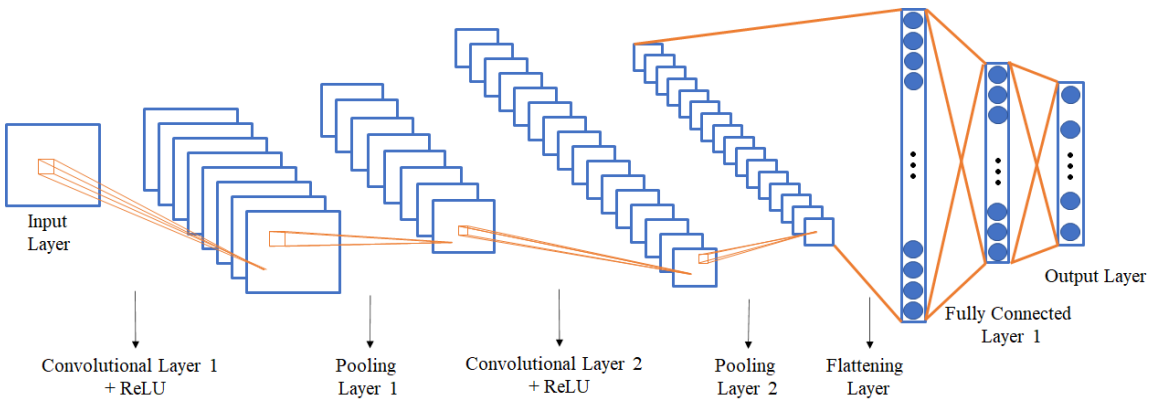


Figure 2.2: Architecture of a typical CNN

A CNN uses convolution operation in at least one of its layers. These convolutional layers are used to extract representative features from the inputs in order to detect local conjunctions of previous layer's features [78].

In CNNs, a convolutional layer usually followed by a pooling layer. Pooling layers replace the output of its previous convolutional layer with summary statistics of the local regions to merge semantically similar features into one. This allows detecting a motif when the relative positions of the features forming the motif vary [78]. Moreover, using pooling layers, reduces the dimensionality of feature maps, the computational cost, the memory usage, and the number of trainable parameters. Reducing the number of network parameters makes the learned features invariant to small transitions and controls the over-fitting.

The last few layers of a CNN are fully connected layers. They are used to learn the classifier. The last fully connected layer, the output layer, computes the class prediction. A typical output layer is a softmax layer that estimates the class probabilities where the category with the highest probability is considered as the predicted class. In a typical CNN, a differentiable loss function is used to perform the optimization with gradient descent algorithm.

2.6.2 CNN Architectures

In the previous section, the typical convolutional neural network architecture was discussed. Since the 1990s, various CNN architectures for numerous applications have been proposed and developed [80, 82–84, 101–112]. These architectures are built by making modifications in structural form, parameter optimizations, regularization,

and development of new blocks and processing units. In particular, the most novel ideas are developed by training deeper architectures [113]. This progress can be seen in the decreasing error rate of ImageNet Large-Scale Visual Recognition Challenge (ILSVRC), where the top-5 classification error rate are reported for each architecture. In this section, we review some of these popular architectures.

The LeNet architecture proposed by LeCun et al. [101] in 1998 is the most well-known CNN architecture that was developed to recognize the handwritten digits on the MNIST data set. Figure 2.3 depicts the architecture of LeNet. The AlexNet, proposed by Krizhevsky et al. [80], has a similar architecture to LeNet with a larger and deeper network. It was the first architecture that used a stack of convolutional layers on top of each other. Moreover, AlexNet utilized two regularization techniques, dropout and data augmentation, to reduce over-fitting.

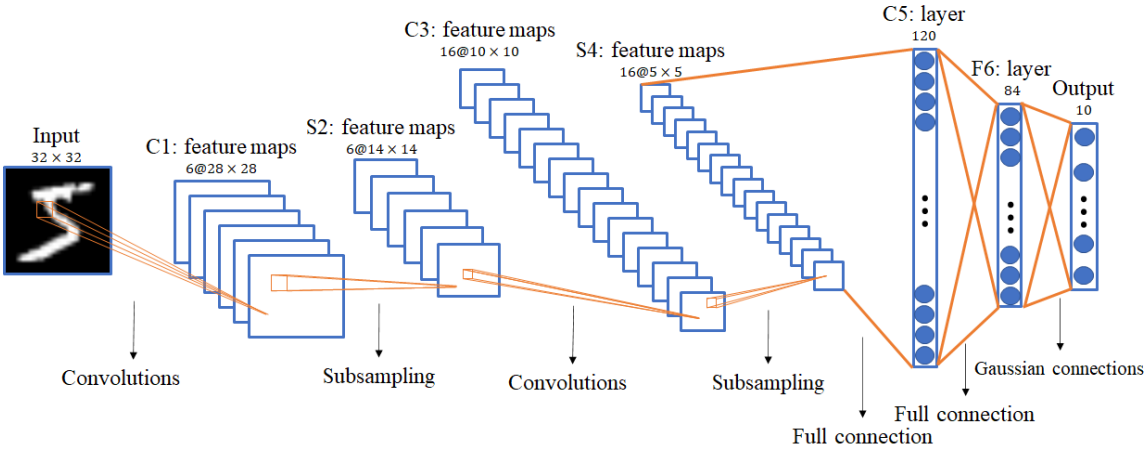


Figure 2.3: LeNet Architecture

The GoogLeNet (also called Inception-V1), developed by Szegedy el al. [83], uses a new level of layer organization in form of sub networks called inception mod-

ules which is inspired by the Network in Network paper proposed by Lin et al. [114] in conjunction with the theoretical work of Arora et al. [115]. Using inception modules reduces the number of training parameters and allows training deeper networks (GoogLeNet has 12 times fewer parameters than AlexNet).

Simonyan et al. [81], from the Visual Geometry Group (VGG) research lab, proposed easy and efficient CNN architectures with different numbers of trainable layers. They showed that powerful models with smaller number of parameters and lower computational complication can be built by using the stack of 3×3 convolutional filters. Moreover, they showed that adding more layers increases the performance of the network. Due to their deep, homogeneous, and simple architectures, VGG models have been widely used as the basis of new models in transfer learning [113].

Residual Network (ResNet) proposed by He et al. [82], is a very deep CNN architecture that is composed of 152 layers. In ResNet, the input of a layer is connected to the output of a layer located in a higher level. These connections, shortcut connections, makes the optimization process faster and easier. Figure 2.4 illustrated the shortcut connections.

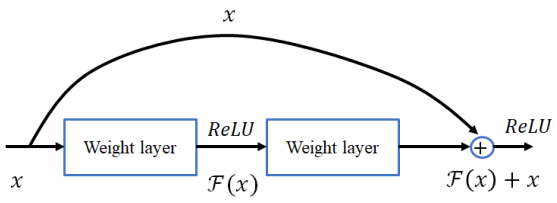


Figure 2.4: A building block of residual learning

High-Resolution Network (HRNet) [102, 103], unlike common architectures (e.g., VGG, ResNet), maintains the high-resolution representations through the whole

process on a multi-stage architecture. HRNet has been used in several applications in object detection, semantic segmentation and human pose prediction [113].

Inception-ResNet [104] and Inception-v4 [105] are the upgraded versions of Inception-V1/Inception-V2 architectures. Highway networks [106] is a novel architecture designed to ease the gradient-based training of very deep networks by allowing unimpeded information flow across several layers on information highways. DenseNet [107, 108] is designed to overcome the vanishing gradient problems by following the same direction as ResNet and Highway networks. ResNext, also called Aggregated Residual Transform Network [112], is an enhanced version of the Inception network. WideResNet [109] is designed to utilize the power of residual modules by making the ResNet wider instead of deeper. Pyramidal Net [110] overcomes the ResNet learning interference problem by slowly extending the residual units' width to cover the most visible places. Xception [84] utilized modified version of inception modules by making them wider by using 3×3 filters followed by a 1×1 filter. CapsuleNet [111] is using a type of structures called capsules to detect the presence of objects at a location considering size, orientation, and perspective of the input image. A comprehensive review of these architectures can be found in [113].

2.6.3 Transfer Learning and Fine Tuning

There are two main obstacles in training a deep neural network for a new application. First, it requires substantial amounts of data to assure the convergence of model's loss function without over-fitting. Second, training a deep neural network from scratch is a computationally expensive task that requires time and resources.

One solution, is to use a pre-trained network from a similar domain as an initial model to train and learn a new model for the target application [116]. This solution, called transfer learning, is based on the assumption that the information obtained in a domain with a sufficiently large training data can be used in another domain of interest, where the data may be in a different data distribution [117].

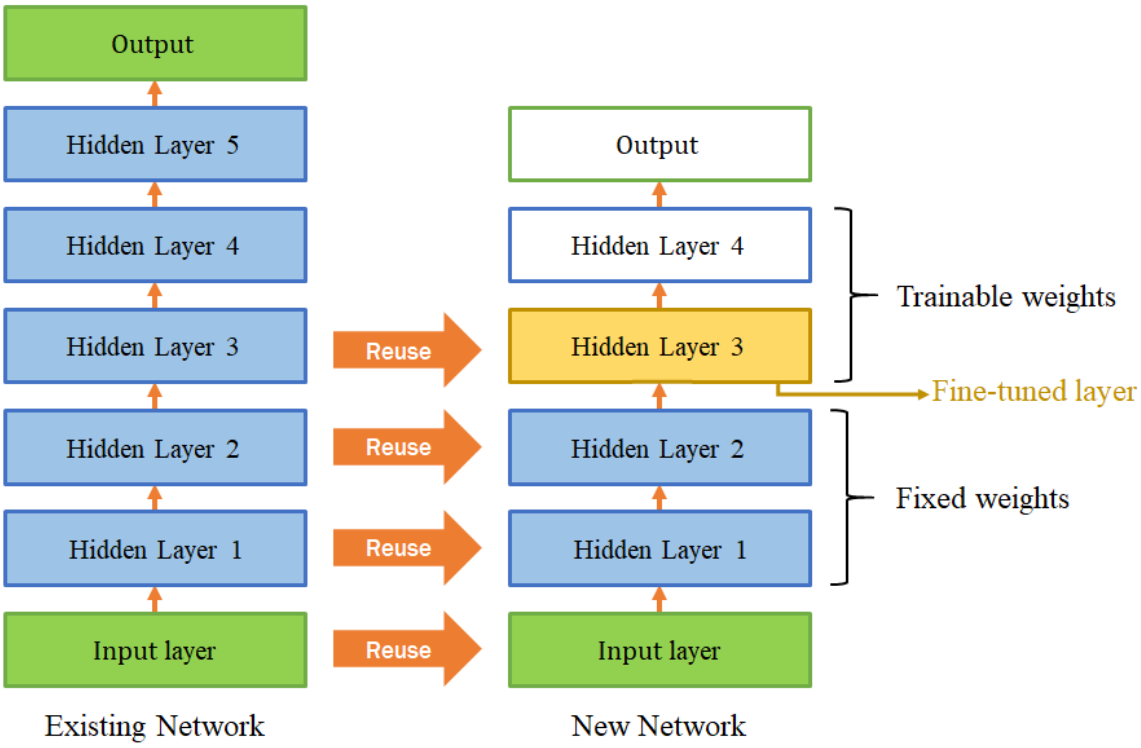


Figure 2.5: Reusing pre-trained layers [1]

In general, transfer learning is used to improve a model from one domain by transferring information learned from a similar domain [116]. Transfer learning leads to speeding up the training process of the new model considerably, and it requires much less data for the training [1]. A typical approach for transfer learning in building new deep neural networks is to reuse some layers from a large-scaled trained models as the base. Specifically, the transferred layers are used to extract features. By fixing

the base layer weights, and training the network to learn low-level weights of the trainable layers, the classifier can learn a new model that fits the new data while minimizing the risk of over-fitting. This approach is illustrated in figure 2.5.

An optional step in transfer learning is fine-tuning, where the features learned by the top base layers are adapted to the new data. This process is performed by allowing the model to tweak the weights of one (or more) of top base layers, i.e., hidden layer 3 in figure 2.5, during the learning process. Fine-tuning often improves the performance of the model by adapting the learned features.

2.6.4 Visualization Methods

Visualization methods are widely used to explain AI models [118] including deep learning models [119]. In deep neural networks, visualization methods are scientific approaches used to express an explanation for the network's behavior by highlighting the characteristics of an input that strongly influence the output. A good summary of the foundation methods is available in [120]. In the following subsections, we outline two widely used visualization approaches.

2.6.4.1 Class Activation Mapping

Class Activation Mapping (CAM) [2] is a visualization method that uses Global Average Pooling (GAP) in a CNN architecture. CAM identifies a class activation map for a particular category, that indicates the important regions of an image used by the CNN to identify that category [2]. The procedure for generating these maps is illustrated in figure 2.6.

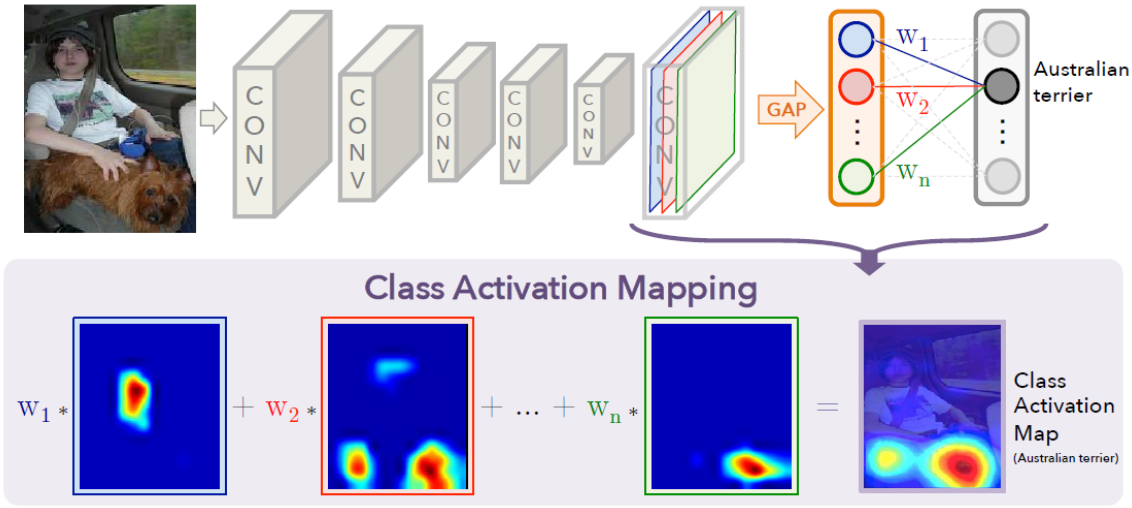


Figure 2.6: Overview of CAM [2]

Let K denote the number of feature maps in the last convolutional layer, and $A^k \in \mathbb{R}^{u \times v}$ indicate the kth feature map with width u and height v. These feature maps are aggregated and resulted into a score, y^c, for each category c:

$$y^c = \sum_k^K w_k^c \sum_i^u \sum_j^v A_{ij}^k \tag{2.17}$$

Let $L_{CAM}^c \in \mathbb{R}^{u \times v}$ indicate the class activation map for class c. CAM computes L_{CAM}^c by calculating the linear combination of the final feature maps using the learned weights of the final layer:

$$L_{CAM}^c = \sum_k^K w_k^c A^k \tag{2.18}$$

In (2.18), L_{CAM}^c is the weighted linear sum of the presence of the visual patterns at different spatial locations. By up-sampling L_{CAM}^c to the input image, the discriminative regions of class c can be identified.

CAM cannot be applied to networks with multiple fully-connected layers at the end of their architectures. This is the main limitation of CAM. In [3], the authors

proposed a solution to make CAM applicable to such networks by replacing the fully-connected layers with convolutional ones and re-training the network.

2.6.4.2 Gradient-weighted Class Activation Mapping

Gradient-weighted Class Activation Mapping (Grad-CAM) [3, 121] is a generalization of CAM that produces a localization map of the important regions of an input image by using the class-specific gradient information flowing into the final convolutional layer of a CNN [121]. Grad-CAM can be used to explain the CNNs output. Figure 2.7 outlines the main steps of Grad-CAM.

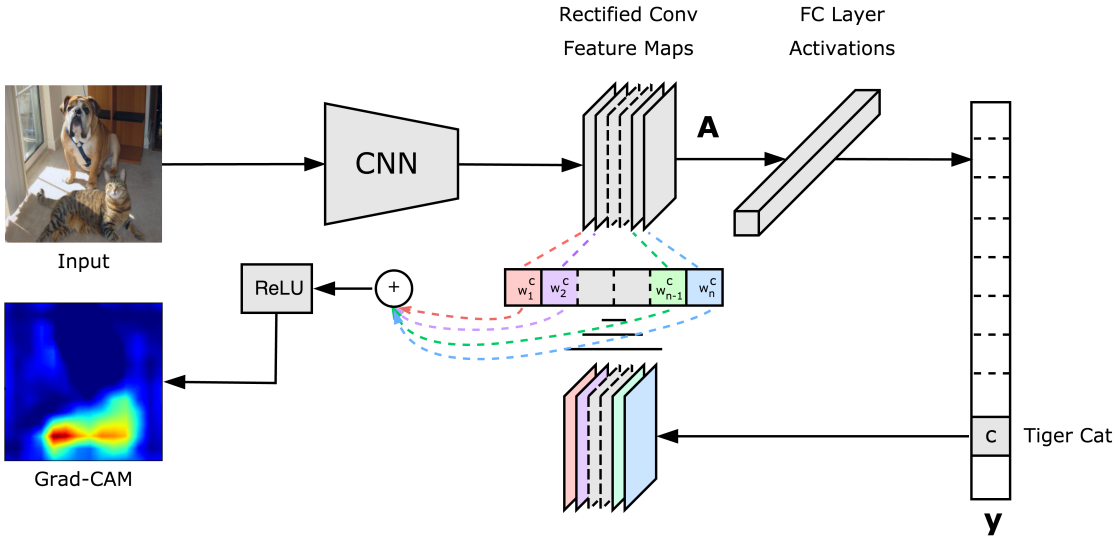


Figure 2.7: Overview of Grad-CAM [3]

Grad-CAM uses a generalized version of equation (2.18) to compute the class discriminative localization map denoted by $L^c_{Grad-CAM} \in \mathbb{R}^{u \times v}$. It computes $L^c_{Grad-CAM}$ using:

$$L^c_{Grad-CAM} \cong \sum_{k=1}^{K} \alpha^c_k A^k, \tag{2.19}$$

where α_k^c is the neuron importance weights that is estimated by computing the gradient of the score for class c, y^c, with respect to the feature maps A^k of a convolutional layer; i.e., $\frac{\partial y^c}{\partial A^k}$. Thus, α_k^c is calculated using:

$$\alpha_k^c = \frac{1}{uv} \sum_{i=1}^{u} \sum_{j=1}^{v} \frac{\partial y^c}{\partial A_{ij}^k} \tag{2.20}$$

where $\frac{\partial y^c}{\partial A_{ij}^k}$ is the linear effect of the pixel located at (i, j) in the kth feature map in the cth class. The weight α_k^c represents the discriminative power of feature map A^k for the target class c [121].

Then, a ReLU function is applied to a linear combination of the maps that considers only the features with positive influence on the target class. In particular, ReLU is applied to identify the pixels with positive values whose intensity should be increased to increase y^c and ignore the pixels with negative values that are likely to be the informative features of the other classes [121].

$$L_{Grad-CAM}^c = ReLU \left(\sum_{k=1}^{K} \alpha_k^c A^k \right) \in \mathbb{R}^{u \times v} \tag{2.21}$$

In (2.21), $L_{Grad-CAM}^c$ has the same dimension $(u \times v)$ as the last convolutional layer. To visualize $L_{Grad-CAM}^c$ of the original image, it needs to be up-sampled.

CHAPTER 3

LOCAL FEATURE SELECTION FOR MULTIPLE INSTANCE DATA

In this chapter, we propose a new feature selection algorithm for MIL inspired by the concept of local feature selection proposed in [9] for single instance learning. Unlike traditional feature selection algorithms, where a global set of relevant features is learned for the whole data set, our proposed method learns a potentially different relevant set of features for every sample (i.e., bag). We call our proposed algorithm Multiple Instance Local Salient Feature Selection (MI-LSFS).

3.1 Local Feature Selection for MIL

In this section, we outline the main steps of our proposed MIL feature selection method, which is an extension of lLFS [9] (outlined in section 2.4) to the MIL framework. To extend lLFS to MIL, we need to generalize it to use bags of instances instead of standard feature vectors. We consider bags as the representatives of local regions in the sample space. The details of our proposed feature selection method are discussed in the following sub-section.

3.1.1 Multiple Instance Local Salient Feature Selection

Let $f^{(i)} \in \{0,1\}^M$ denote the status of the features for the region centered at bag B_i, where M is the dimensionality of the instances in the original feature space. Let $f_m^{(i)}$ denote the selection status of the m^{th} feature of B_i. If the m^{th} feature of B_i is selected, then $f_m^{(i)}$ is set to 1, otherwise it is 0. We should note here that if $f_m^{(i)}$ is selected for B_i, then this feature is selected for every instance in B_i. Let $B^{(i)}$ be the original values of the B_i instances. We define $B_p^{(i)}$ as the projection of the B_i instances onto the subspace of features represented by $f^{(i)}$.

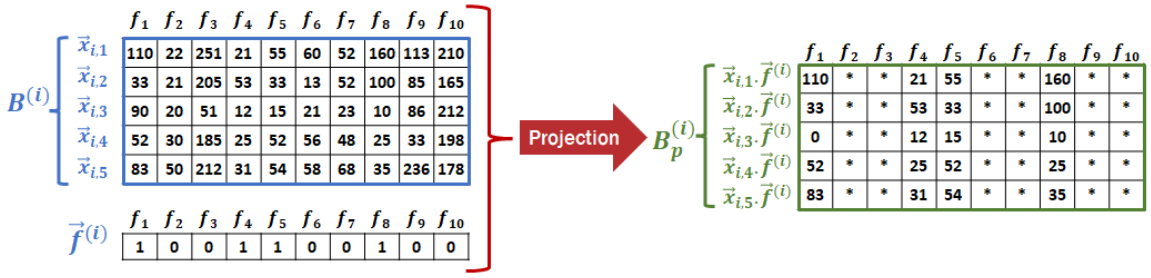


Figure 3.1: Projection of B_i into $\vec{f}^{(i)}$. In this example, B_i has five instances and 10 features. $B_p^{(i)}$ is the projection of the B_i instances onto the subspace of features represented by $\vec{f}^{(i)}$.

Figure 3.1 illustrates an example of computing $B_p^{(i)}$. In this example, bag B_i consists of five instances $(\vec{x}_{i,1}, \ldots, \vec{x}_{i,5})$, where each instance has 10 features. $\vec{f}^{(i)}$ indicates the selection status of B_i's features. Thus, $\vec{f}^{(i)} \in \{0,1\}^{10}$. For this example, we assume that features 1, 4, 5 and 8 of B_i are selected. The projection of B_i is calculated by only considering the selected features in $\vec{f}^{(i)}$.

Using the above definitions, we restate the goal of feature selection for MIL as

finding $f^{(i)}$ for each B_i such that by projecting all instances of all bags onto $f^{(i)}$, the clustering behavior in the neighborhood of $B_p^{(i)}$ is optimum with respect to the two following criteria:

1. Some bags with the same label as B_i should have some instances that are as close as possible to some instances of B_i in the projection space defined by $f^{(i)}$.

2. All instances of a bag that has a label different from that of B_i should be located as far as possible from all instances of B_i in the mapped space defined by $f^{(i)}$.

To formulate the above optimization, we need a function that computes the distance between two bags. Typically, this distance is computed by considering the distances between their instances. Treating each bag as a set of M dimensional vectors, any function that calculates the distance between two sets of points can be used to define the distance between bags. One commonly used measure is the Hausdorff distance [19]. In this thesis, we use a variation of the Hausdorff distance, called the minimal Hausdorff distance. This distance, $HD(B_i, B_j)$, calculates the minimal Hausdorff distance between two given bags B_i and B_j with instances $\{\vec{x}_{i,k}\}_{k=1}^{|B_i|}$ and $\{\vec{x}_{j,l}\}_{l=1}^{|B_j|}$ using:

$$HD(B_i, B_j) = \min_{\substack{\vec{x}_{i,k} \in B_i \\ \vec{x}_{j,l} \in B_j}} ||\vec{x}_{i,k} - \vec{x}_{j,l}||. \tag{3.1}$$

Furthermore, we define $HD(B_i, B_j, f^{(i)})$ as the distance between the projection of bags B_i and B_j onto the subspace represented by $f^{(i)}$; i.e.,

$$HD(B_i, B_j, \vec{f}^{(i)}) = \min_{\substack{\vec{x}_{i,k} \in B_i \\ \vec{x}_{j,l} \in B_j}} ||\vec{x}_{i,k} \cdot \vec{f}^{(i)} - \vec{x}_{j,l} \cdot \vec{f}^{(i)}||. \tag{3.2}$$

The proposed Multiple Instance Local Salient Feature Selection (MI-LSFS) algorithm optimizes the same objective functions defined in (2.1) and (2.2), but replacing $D(S_i, S_j)$ with $HD(B_i, B_j)$ and $D(S_i, S_j, f^{(i)})$ with $HD(B_i, B_j, f^{(i)})$. Thus, we restate the objective functions as:

$$U_1(f^{(i)}) = \frac{1}{N_i - 1} \sum_{j; y_j = y_i; j \neq i} g\left(HD(B_i, B_j, f^{(i)}), \sigma^{(i)}, \lambda\right) \tag{3.3}$$

and

$$U_2(f^{(i)}) = \frac{1}{N - N_i} \sum_{j; y_j \neq y_i} g\left(HD(B_i, B_j, f^{(i)}), \sigma^{(i)}, \lambda\right). \tag{3.4}$$

In (3.3) and (3.4), N is the number of training bags, N_i is the number bags with the same label as B_i, and g is the logistic function. The two other parameters, $\sigma^{(i)}$ and λ, are the logistic growth rate and the regularization parameter. Optimization of (3.3) and (3.4) can be formulated as optimizing:

$$\begin{cases} \min_{f^{(i)}} U_1(f^{(i)}) \\ \max_{f^{(i)}} U_2(f^{(i)}) \\ s.t. \begin{cases} f^{(i)} \in \{0,1\}^M \\ 1 \leq 1^T f^{(i)} \leq \alpha \end{cases} \end{cases} \tag{3.5}$$

By restricting $1^T f^{(i)}$ between 1 and α, the number of selected features is set to be at least 1 and at most α features.

In (3.3) and (3.4), the logistic function g,

$$g(z, \sigma, \lambda) = \frac{1}{1 + \exp(-\sigma z)} - 0.5 + \lambda z, \tag{3.6}$$

transforms the distance between bags such that closer bags will have more effect on the objective function and farther bags will have minimal effect. In (3.6), the purpose

of the linear term λz is to give a chance to the potentially relevant bags that are far from $B_p^{(i)}$ to get closer to $B_p^{(i)}$ in subsequent iterations of the optimization process.

3.1.2 Optimization

One way to solve the multi-objective optimization problem in (3.5) is to use the Pareto optimality concept [47] and combine the objective functions in (3.5) into a single one:

$$\begin{cases} \min_{f_\beta^{(i)}} U_1(f_\beta^{(i)}) \\ s.t. \begin{cases} f_{m,\beta}^{(i)} \in [0,1] \\ 1 \leq 1^T f^{(i)} \leq \alpha \\ U_2(f_\beta^{(i)}) \geq \beta \epsilon_{max}^{(i)}. \end{cases} \end{cases} \tag{3.7}$$

In (3.7), β is a set of constraint values, in the range of [0,1], that control the number of Pareto points, $f_\beta^{(i)}$ represents the selected features of B_i for a given β, and $\epsilon_{max}^{(i)}$ is the maximum feasible value of U_2 for the given sample B_i that is computed using:

$$\begin{cases} \epsilon_{max}^{(i)} = \max_{f^{(i)}} U_2(f^{(i)}) \\ s.t. \begin{cases} 0 \leq f_m^{(i)} \leq 1, m = 1, ..., M \\ 1 \leq 1^T f^{(i)} \leq \alpha. \end{cases} \end{cases} \tag{3.8}$$

Optimization of (3.7) leads to a solution for every value of β. The feasibility of this approach is discussed in details in [9]. Next, a randomized rounding process [47] is used to convert each relaxed solution $f_\beta^{(i)}$ into a binary solution $f_\beta^{*(i)}$. Each element of $f_\beta^{*(i)}$ represents the selection status of its corresponding feature (i.e., 1 if the feature is selected and 0 if it is not).

The steps of MI-LSFS algorithm are summarized in algorithm 2. MI-LSFS identifies different solutions, $f_\beta^{*(i)}$, for each given bag B_i. Then, one of these solutions needs to be identified as the optimal solution, $f^{*(i)}$, for bag B_i. In the next subsection, we describe the steps to find $f^{*(i)}$ for each given bag B_i.

Algorithm 2: MI-LSFS

Input:

B : Bags

Y : labels

N : Number of Bags

M : Number of features

α : Controls the maximum number of features to be selected

Output:

$\{f^{*(i)}\}, i = 1, ..., N$: selected features for each sample

Initialization:

$f_\beta^{(i)} = \frac{1}{\alpha}(1, ..., 1), i = 1, ..., N$

$\lambda = \frac{0.01}{\alpha}$

$\beta = [0, 1]$

for $i = 1$ **to** N **do**

  Compute $\sigma^{(i)}$ by solving (3.14);

  Compute $\epsilon_{max}^{(i)}$ by solving (3.8);

  for $\beta = 0$ **to** 1 **do**

    Compute $f_\beta^{(i)}$ by solving (3.7);

    Randomized rounding process of $f_\beta^{(i)}$ to obtain binary feature vector $f_\beta^{*(i)}$;

  Find $f^{*(i)}$ and $r^{(i)}$ using algorithm 3;

3.1.3 Identifying the Optimal Solution for each Bag

For each bag B_i in the data, algorithm 2 identifies a solution $f_\beta^{*(i)}$ for each value of β. Next, we propose using cluster validity measures to assess the quality of the different solutions and identify the optimal one. The selected features associated with the optimal solution, $f^{*(i)}$, will be considered the relevant features of bag B_i.

For cluster validity, we use the impurity and Matthews correlation coefficient (MCC) [20] measures to assess the goodness of each cluster of bags. Specifically, we generalize the standard impurity [122] to MIL and define it as the ratio of the number of bags with different class labels to the number of bags with the same class label. Formally, for the cluster identified around bag B_i and for a maximum distance threshold d, we define the impurity as:

$$IMP(i,d) = \frac{\sum_{j=1,..,N;i\neq j} (1 - I(y(i), y(j))) \times step(HS(B_i, B_j, f_\beta^{*(i)}))}{\sum_{j=1,..,N;i\neq j} I(y(i), y(j)) \times step(HS(B_i, B_j, f_\beta^{*(i)}))} \tag{3.9}$$

where

$$HS(B_i, B_j, f_\beta^{*(i)}) = d - HD(B_i, B_j, f_\beta^{*(i)}). \tag{3.10}$$

In (3.9),

$$step(z) = \begin{cases} 1, \ if \ z \geq 0 \\ 0, \ Otherwise, \end{cases} \tag{3.11}$$

and

$$I(x,y) = \begin{cases} 1, \ if \ x = y \\ 0, \ Otherwise. \end{cases} \tag{3.12}$$

The second performance measure, MCC, is a balanced measure of the binary classification quality and can be computed using:

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}. \tag{3.13}$$

In (3.13), TP, TN, FP and FN are the true positives, true negatives, false positives and false negatives, respectively. For the purpose of calculating MCC, these measures

are calculated based on the local cluster centered at B_i with a radius d, which can be considered as a weak classifier. Specifically, given the local region of a bag B_i with a radius d, and a testing bag B_t, we label B_t with the same label as B_i if it falls into the local region of B_i. Otherwise, it is predicted as others. Using this approach, we label every bag in the training data ($B_t \in \mathcal{B}$ where $\mathcal{B}$ is the set of all training bags). Next, using the predicted labels and the true label of the bags, we compute the confusion matrix. Finally, we use (3.13) to calculate the MCC value. Let $MCC(i,d)$ denote the MCC value for the local region centered at bag B_i with a radius d.

Algorithm 3: Finding the best Beta value for the MI-LSFS algorithm

Input:

$Bags$: Bags of instances

Y: Bags labels

N : Number of Bags

i : Bag index

$f_\beta^{*(i)}, \beta = [0,1]$: Learned feature weights for bag i using multiple β values

$impurity\_threshold$: maximum value of impurity

Output:

$f^{*(i)}$: selected features for B_i

$r^{(i)}$: radius of cluster centered at B_i

for $\beta = 0$ **to** 1 **do**

    D = unique distances of $D(B_i, B_j, f_\beta^{*(i)})$ for $i \neq j$;

    for $d \in D$ **do**

        $IMP_\beta(i,d) = IMP(i,d)$ using (3.9);

        $MCC_\beta(i,d) = MCC(i,d)$ using (3.13);

Find d and β s.t. $IMP_\beta(i,d) \leq impurity\_threshold$ with the highest $MCC_\beta(i,d)$ value;

$f^{*(i)} = f_\beta^{*(i)}$;

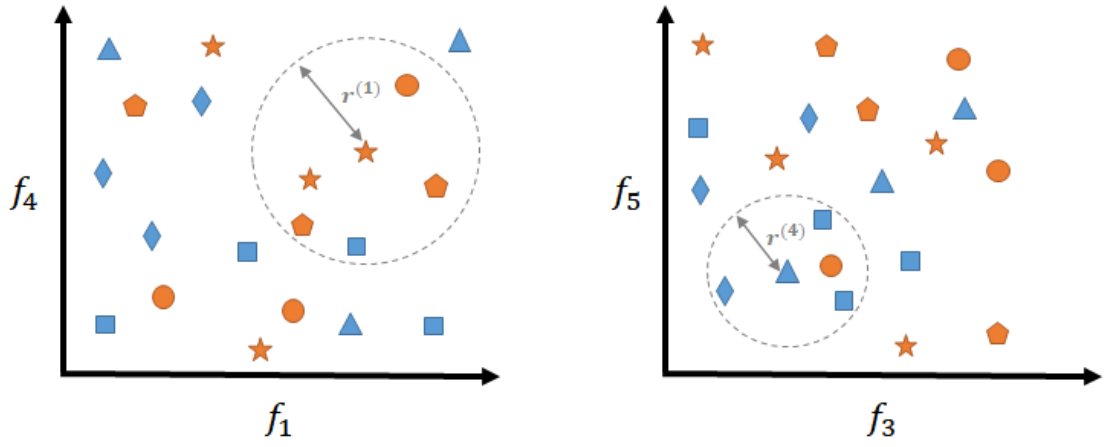
$r^{(i)} = d$;

In algorithm 3, we summarize the steps used in finding the optimal solution for a given bag B_i denoted by $f^{*(i)}$. First, for a given bag B_i, we project the instances

of all bags using $f_\beta^{*(i)}$ and calculate the distance of every bag in the training data to B_i in the projected space using (3.2). Let D denote the set of unique distances to B_i in the projection space of $f_\beta^{*(i)}$. Then, we search the projection space to find clusters of bags, that are within a distance $d \in D$ from B_i, that have the closest impurity that is smaller than a specified impurity threshold. Since the first part of the MI-LSFS (algorithm 2) generates $|\beta|$ sets of potential relevant features for each bag, the clustering part will generate $|\beta|$ different solutions. Finally, we select the solution with the highest MCC value as the optimal one, and we set $f^{*(i)}$ as the optimal set of selected features, and $r^{(i)}$ as the radius of the optimal cluster in the projected space $f^{*(i)}$.

Since the MCC, computed using (3.13), considers the four confusion matrix measures, the selected region can be more informative than the one identified using only the impurity measure. Also, by using the impurity to cut off the regions before calculating the MCC, we reduce the risk of over-fitting since the regions include false positive values.

In figure 3.2, we illustrate the identification of the local regions of two sample bags. In these illustrations, there are three positive bags and three negative bags. Each bag has a few instances that are displayed using different shapes. Instances from positive bags are displayed in orange and instances from negative bags are shown in blue. Figure 3.2(a) shows the local region of a sample positive bag B_1 where its instances are depicted with star. All the instances are shown in the projected feature space using f_1 and f_4, which are the relevant features of B_1. These are the features that are activated in $f^{*(1)}$. The local region of B_1 centered at one of its instances



(a) (b)

Figure 3.2: Sample examples for the local clusters: (a) clustering of sample bag B_1 where f_1 and f_4 are its relevant features. (b) clustering of sample bag B_4 where f_3 and f_5 are its relevant features. Instances of positive and negative bags are color-coded with orange and blue, respectively, and instances of each bag are shown in a different shape.

with a radius $r^{(1)}$ is shown with a gray dashed line. By optimizing the MI-LSFS objective function, we expect that some positive instances get as close as possible to the center of this cluster and negative instances get as far as possible from this cluster. However, we allow some impurity (i.e., the blue square in figure 3.2(a)) in the cluster. We should mention here that a positive bag contains both positive and negative instances. Therefore, we expect that the negative instances of positive bags to be distributed anywhere outside the cluster. Figure 3.2(b) shows the local region of a sample negative bag B_4 centered at one of its instances with a radius $r^{(4)}$ where only features f_3 and f_5 are activated in $f^{*(4)}$. The instances of B_4 are shown with

triangles. The local region of B_4 is shown with a gray dashed line.

3.1.4 Parameters of the Logistic Function

The logistic function in (3.6) has two parameters, λ and $\sigma^{(i)}$, that need to be assigned. The parameter $\sigma^{(i)}$ is calculated based on the point from B_i that sits on the knee point of the logistic function [9]. Formally, $\sigma^{(i)}$ is calculated using (3.14) and the initialization value of the selected features, $f_\beta^{(i)}$, where the value of 0.47 is set based on the knee point of the logistic function in (3.6).

$$\begin{cases} \frac{1}{1+\exp(-\sigma^{(i)}\varphi^{(i)})} - 0.5 = 0.47 \\ \varphi^{(i)} = \max_{j=1..N, j\neq i}\{DH(B_i, B_j, f^{(i)})\}, \end{cases} \tag{3.14}$$

In (3.14), λ is set to the default value of $0.01/\alpha$ at the initialization step.

CHAPTER 4

APPLICATIONS OF MI-LSFS

In this chapter, we propose various applications that take advantage of the locally learned features. MI-LSFS treats each bag as a local region in the feature space and learns a subset of relevant features for each bag. Since common MIL classifiers cannot handle data samples with different sets of features, they cannot take advantage of the valuable information learned by MI-LSFS. To address this limitation, we propose an efficient MIL classifier that incorporates the information learned by MI-LSFS. Our MIL classifier, called MILES-LFS, is inspired by the well-known MILES [21] algorithm. MILES-LFS is a robust MIL classifier that improves the performance and efficiency of the MIL classification task.

A second application of our local feature selection that we are proposing is the investigation and exploration of the features learned by a Convolutional Neural Network (CNN). In particular, a CNN learns a high dimensional set of features and performs classification using these learned features. This works as a black box and explaining and illustrating what the model has learned is an active research area [121, 123–125]. We propose using our MI-LSFS algorithm to develop a method that can explain the decisions of a CNN model. In particular, we use the local feature selection approach to identify and visualize the relevant features of each target sample.

We also propose a visualization method, called Gradient-weighted Sample Activation Map (Grad-SAM), that utilizes the locally selected features of each target sample to highlight its salient parts.

The third application of our MI-LSFS is to explain the decisions made by a trained machine learning model. Using the fact that local feature selection identifies the relevant features for every object, we propose a new approach that can be trained to identify the features that lead to the correct and incorrect classification of each sample.

4.1 Classification of Multiple Instance Data

MI-LSFS learns a set of relevant features for each bag. Since these sets can vary from one bag to another, this information cannot be used within standard MIL classification algorithms. In this section, we propose a new MIL classification algorithm, inspired by MILES [21], that can explore the information learned by MI-LSFS. The proposed classifier, called MILES-LFS, adds two main steps to MILES. First, it only uses the identified relevant features of the bag to project it onto the prototype space. Second, instead of using all training instances as prototypes, it selects only a subset of bags and a subset of relevant instances from each bag.

As outlined in Algorithm 2, for each bag B_i, MI-LSFS learns an optimal set of relevant features, $f^{*(i)}$, and a local region with a radius $r^{(i)}$ centered at B_i, in the mapped space. The cluster of bags is optimized with respect to impurity and MCC.

Let $Clust_i$ denote this cluster; i.e.,

$$Clust_i = \{B_j \underset{j \neq i}{\in} \mathcal{B} \mid HD(B_i, B_j, f^{*(i)}) \leq r^{(i)}\}. \tag{4.1}$$

In (4.1), $\mathcal{B}$ is the set of all training bags. Using this information, we identify the most representative bags and their representative instances. First, we assign a score to each bag B_i based on the impurity measure of its cluster $Clust_i$. Next, we assign a score to each instance $\{\vec{x}_{i,k}\}_{k=1}^{|B_i|}$ in B_i based on its proximity to all bags in $Clust_i$. Specifically, we compute

$$Score(\vec{x}_{i,k}) = \sum_{B_j \in Clust_i} I\big(HD(\vec{x}_{i,k}, B_j, f^{*(i)}), HD(B_i, B_j, f^{*(i)})\big) \tag{4.2}$$

where I is defined in (3.12).

Using the bags' scores, we identify a subset of relevant prototypes by selecting the N_B bags with the lowest impurity. For each selected bag we identify its N_I instances with the highest score computed using (4.2). Let $\mathcal{P}$ denote this set of selected prototypes, where $|\mathcal{P}| = N_B \times N_I$. Each prototype $x_j \in \mathcal{P}$ inherits the optimal set of relevant features, denoted $f^{*(j)}$, and cluster radius, denoted $r^{(j)}$, of the bag it originated from.

Next, we compute the similarity between each bag, B_i, and each instance prototype, $x_j \in \mathcal{P}$, using

$$S(\vec{x}_j, B_i) = \max_{\vec{x}_{i,k} \in B_i} \Big\{ exp\Big(- \frac{||\vec{x}_{i,k} * f^{*(j)} - \vec{x}_j * f^{*(j)}||^2}{\alpha . r^{(j)}} \Big) \Big\}, \tag{4.3}$$

where α is a constant scaling factor.

We should note here that in (4.3), we consider only the features selected by MI-LSFS for each target concept and we normalize the distance by the radius of

the region centered at the target concept ($r^{(j)}$). Thus, this measure represents the normalized similarity of B_i to the local region centered at target concept $\vec{x_j}$.

Next, we map the training data to the new space defined by the selected instance prototypes, $\mathcal{P}$, and the locally selected features using:

$$m(B_i) = [S(\vec{x}_1, B_i), \ldots, S(\vec{x}_{|\mathcal{P}|}, B_i)]. \tag{4.4}$$

The training data set, consisting of multiple instance data, are then mapped to standard single instance features in a $|\mathcal{P}|$ dimensional space and represented as:

$$[m(B_1^+), \ldots, m(B_{\ell^+}^+), m(B_1^-), \ldots, m(B_{\ell^-}^-)] = \begin{bmatrix} S(\vec{x}_1, B_1^+) & \ldots & S(\vec{x}_{|\mathcal{P}|}, B_1^+) \\ \ldots & \ldots & \ldots \\ S(\vec{x}_1, B_{\ell^+}^+) & \ldots & S(\vec{x}_{|\mathcal{P}|}, B_{\ell^+}^+) \\ S(\vec{x}_1, B_1^-) & \ldots & S(\vec{x}_{|\mathcal{P}|}, B_1^-) \\ \ldots & \ldots & \ldots \\ S(\vec{x}_1, B_{\ell^-}^-) & \ldots & S(\vec{x}_{|\mathcal{P}|}, B_{\ell^-}^-) \end{bmatrix}. \tag{4.5}$$

In (4.5), ℓ^+ and ℓ^- represent the number of positive and negative bags in the training data set. Each row represents a bag and and each column represents a feature in the mapping space (a concept). Finally, to learn the classifier, we apply the 1-Norm SVM [67] to the mapped data.

To test a new bag, we first map it to a standard feature vector using (4.4). Then, we test it using the learned SVM classifier. Figure 4.1 illustrates our proposed MILES-LFS.

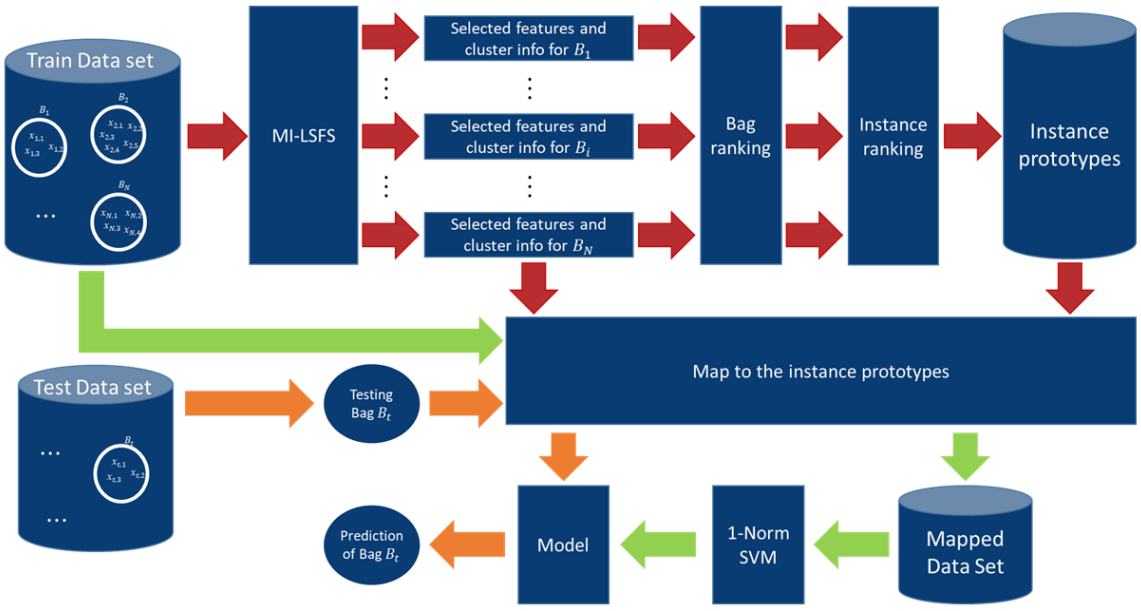


Figure 4.1: Architecture of the proposed MILES-LFS classifier

4.2 Using MI-LSFS to Explain CNN Decisions

Convolutional Neural Networks have been successfully applied to many computer vision applications. However, CNNs, like other deep learning models, work as a black box, and they are often criticized for the lack of transparency in interpreting the learned features and explaining the outcome of the model [126]. Moreover, due to the good performance of deep CNN models, instead of using handcrafted features, they are commonly used as feature extractors, especially in computer vision applications. However, interpreting the models that utilize the extracted features can be challenging. Recently, there has been a significant increase in developing explainable deep learning techniques to interpret the models.

Scientific visualization techniques [2, 3, 121, 127, 128] are widely used in interpreting the features learned by deep models. They highlight the characteristics of

an input that are strongly correlated to the decision of a network (i.e., predicted class) [120]. However, interpreting a subset of features through a visualization approach is not trivial. In this section, we propose a gradient-based visualization method that utilizes the locally selected features for a target sample and visualizes its salient parts. In particular, using our proposed visualization technique, we can validate the performance of our local feature selection method in identifying the salient features that are extracted from a deep CNN through visualization.

We should mention here that few Multiple Instance Convolutional Neural Networks have started to emerge lately [129–133]. However, no standard approach have been established yet. Moreover, although our MI-LSFS has been developed for MIL, we can apply it to single instance data by treating the samples as single instance bags. Thus, we perform MI-LSFS on the extracted features from the fully connected layers by treating them as single-instance bags.

We perform the following steps to investigate a CNN and visualize the local discriminative segments of an input image. First, we extract the features from the deep model to form the training data set. Next, we use our MI-LSFS algorithm to identify a subset of most relevant features for each sample. Finally, we apply our visualization method using the learned features to display the locally significant parts of the input image. Our proposed visualization method is described below.

Grad-CAM, as outlined in section 2.6.4.2, obtains the class-discriminative localization map for any class c by computing the gradient score for class c with respect to the given feature map A^k of a convolutional layer k [121]. In this work, we propose a generalization of Grad-CAM that identifies the discriminative map of a given sample

by computing the gradient score of only the sample's locally selected features. We will show that visualization of this map highlights the significance of its locally selected salient features. Our proposed visualization approach, called Gradient-weighted Sample Activation Map (Grad-SAM), computes the activation map for each sample. Figure 4.2, summarizes the steps of Grad-SAM.

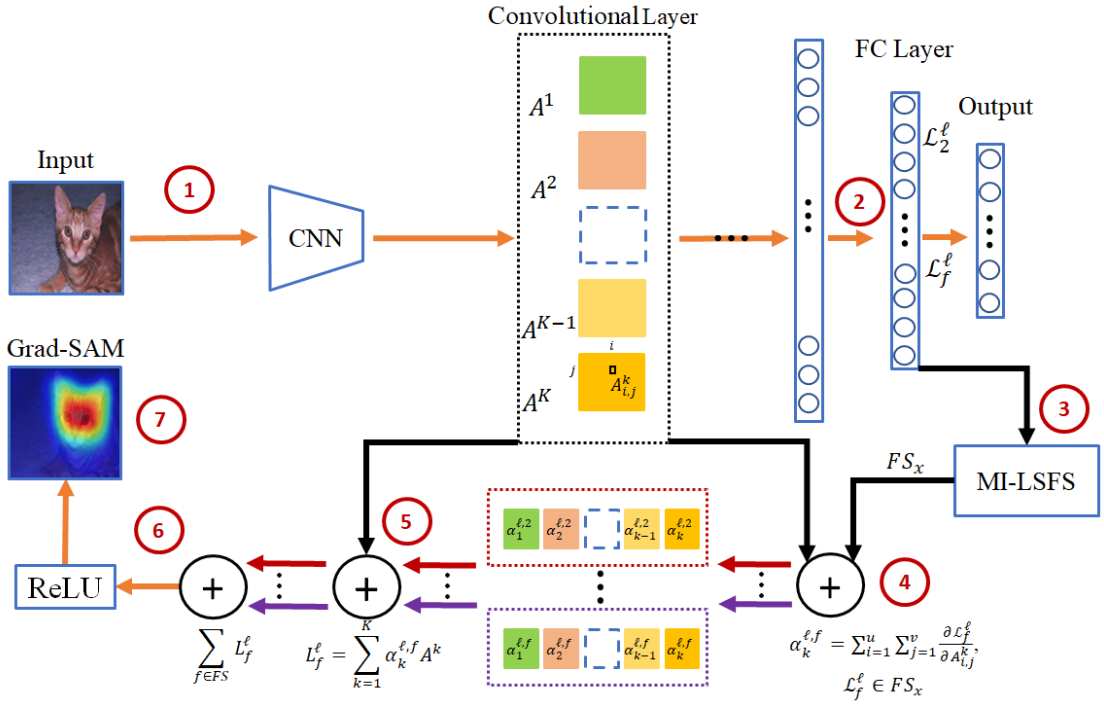


Figure 4.2: Overview of Grad-SAM. The sequence of steps are illustrated by numbers in the circles. 1- Feeding the input image into the trained CNN. 2- Extracting the features from the fully connected layer. 3- Identifying the salient features by applying MI-LSFS. 4- Computing the weight of each feature map in the target convolutional layer for each selected feature. 5- Calculating the feature maps that indicate the importance of each selected feature. 6- Computing the linear combination of feature maps. 7- Up-sampling and visualizing the results.

Let $\mathcal{L}_f^\ell$ denote the fth feature from layer ℓ and let A^k represent the kth feature map of the given convolutional layer. Similar to Grad-CAM, we first calculate the gradient of $\mathcal{L}_f^\ell$ with respect to the elements of feature map $A^k \in \mathbb{R}^{u \times v}$ at the given convolutional layer. Then, we average pool these gradients to obtain $\alpha_k^{\ell,f}$, the importance of feature map A^k for $\mathcal{L}_f^\ell$; i.e.,

$$\alpha_k^{\ell,f} = \frac{1}{Z} \sum_i^u \sum_j^v \frac{\partial \mathcal{L}_f^\ell}{\partial A_{ij}^k}, \tag{4.6}$$

where Z is a normalization factor that is set to the total number of pixels in feature map A^k (i.e., $u \times v$).

After computing the weights of feature map, $\alpha_k^{\ell,f}$, we calculate the importance of feature $\mathcal{L}_f^\ell$, denoted by L_f^ℓ, by computing the weighted combination of the forward activation maps; i.e.,

$$L_f^\ell = \sum_{k=1}^K \alpha_k^{\ell,f} A^k. \tag{4.7}$$

In (4.7), $L_f^\ell \in \mathbb{R}^{u \times v}$. Finally, we calculate the linear combination of feature maps $L_f^\ell, f \in FS_x$ followed by a ReLU function, where FS_x is the set of selected features for sample x. In other words, we calculate the localization map of a given sample x by considering only its selected features in the computation. Thus, we have:

$$L_{Grad-SAM}^{\ell,FS_x} = ReLU\left(\sum_{f \in FS_x} w_f^\ell L_f^\ell\right), \tag{4.8}$$

where $L_{Grad-SAM}^{\ell,FS_x} \in \mathbb{R}^{u \times v}$ is the localization map. To visualize $L_{Grad-SAM}^{\ell,FS_x}$ of the original image, it needs to be up-sampled.

We should mention here that Grad-SAM, similar to Grad-CAM, can be extended for any convolutional layer. However, the patterns learned in the early layers

of a CNN model are more general (low level features), while the later layers learn more abstract patterns (high level features). Therefore, visualization techniques are typically applied on the higher layers of a deep CNN.

Figure 4.3 illustrated an example of Grad-Sam visualization, where MI-LSFS selected 16 features out of the total 128 features for the input image. The 16 selected features were used to highlight the important regions of the input image. In this example, the eyes of the cat are the most important segments of the image.

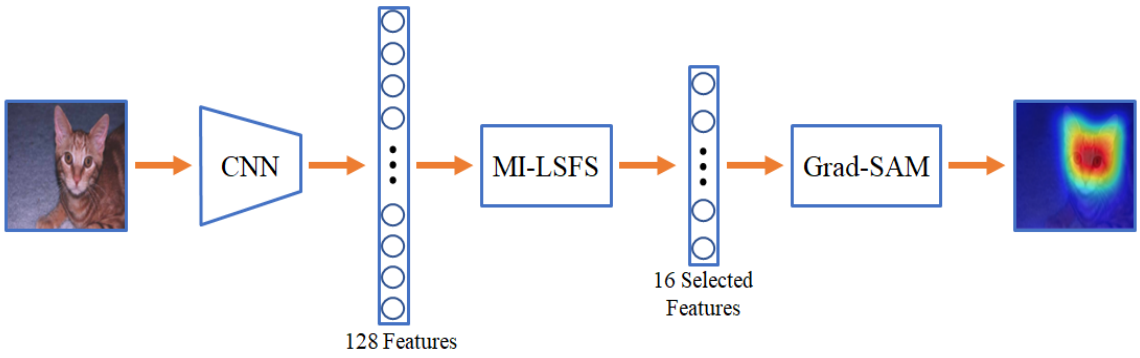


Figure 4.3: An example of applying Grad-SAM. First, using the trained CNN, we extract the features for the input image. Next, we apply MI-LSFS to identify the most discriminative features for the given input. For this example, 16 features out of the total 128 features are selected. Finally, we use Grad-SAM to visualize the selected features where it highlights the eyes and forehead of the cat.

4.3 Using MI-LSFS to Explain the Decision of a Classifier

Understanding the behavior of a classifier and the reasons for its correct or incorrect prediction is crucial in designing safe and explainable machine learning

systems. However, most of the efforts in designing machine learning algorithms have focused on improving the performance of models while understanding the behavior of trained models received less attention [134, 135].

The standard approach in accessing the trustworthiness of the prediction made by a model is to use the confidence score of the classifier. Researchers have proposed alternative measures to enhance this process. For instance, the trust score, proposed by Jiang et al. [134], assesses whether the prediction of a classifier for a test example can be trusted or not by calculating a ratio of distances between the high-density set of samples from the target class and a different class. While these methods are helpful to measure the trustworthiness of the classifier, they do not provide an explanation to interpret the result other than the score.

In [134–138], researchers have proposed and developed tools to explain the classifiers' predictions. For instance, LIME [135], explores the sampled instances of the local region of a sample and tries to form a linear boundary to separate the classes to find a simpler explanation of the decision made by a complex classifier. However, they consider all the features to form the neighboring samples. In the following, we propose an approach to interpret and explain the decision made by a given classifier. In particular, we use our MI-LSFS to identify relevant features of each sample that lead to the correct or incorrect classification of each sample.

4.3.1 Classifier Explanation by Local Feature Selection

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ denote the training data where x_i and y_i are the observation and its label, respectively. Let $h(x_i) = \hat{y}_i$ be the classifier that has been trained

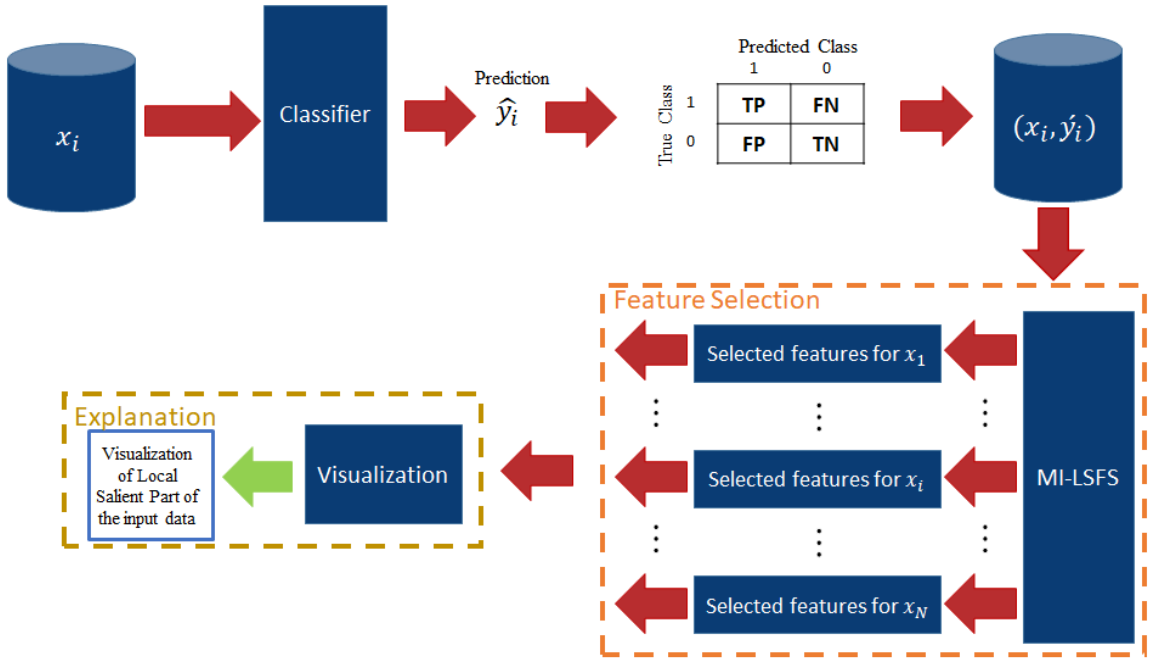


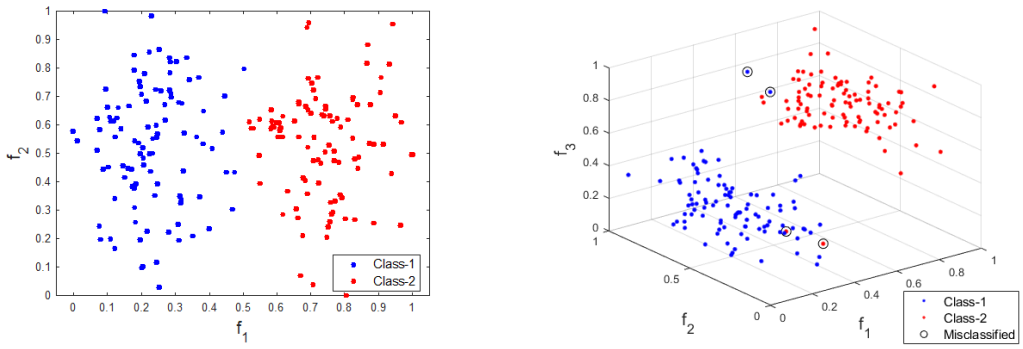
Figure 4.4: Overview of our local classifier explanation. First, we use the trained classifier to predict the labels of the input samples. Next, we form the confusion matrix and assign a label to each sample (i.e., TP, FP, TN, or FN). Using the obtained labels and samples, we form a data set and apply MI-LSFS on that. Finally, we use the selected features of each sample to explain the decision made by the classifier.

on $\mathcal{D}$ where $\hat{y}_i$ denotes the predicted label for sample x_i. Next, by comparing the predicted labels with the ground truth labels, we form the confusion matrix. Without loss of generality, in the following we assume binary classification. In this case, we can split the data into four categories using the confusion matrix. In particular, we assign a label that indicates whether the sample is a True Positive (TP), False Positive (FP), True Negative (TN), or False Negative (FN). We denote the new label assigned to sample i by $y_i{}'$ and we form a new data set $\mathcal{D}' = \{(x_i, y_i{}')\}_{i=1}^{N}$.

To analyze $\mathcal{D}'$, we perform our MI-LSFS on $\mathcal{D}'$ and obtain locally informative

features for each sample. In particular, we are interested in investigating the features responsible for assigning a sample to a particular class (i.e., TP, FP, TN, and FN). Finally, we use the learned features of each sample to explain the classifier prediction by presenting a visual representation to understand the relationship between the input sample and the model's prediction [135]. The intuition is that by having the labels assigned based on the classifier prediction, the selected features will demonstrate locally important features for the assigned class, explaining why a sample is classified correctly or incorrectly by classifier h.

For instance, if a sample is predicted as a False Positive, its locally selected features can explain why it is similar to the FP class. These steps are illustrated in figure 4.4. We call our proposed approach Classifier Explanation by Local Feature Selection (CE-LFS).



(a) (b)

Figure 4.5: Toy examples of CE-LFS. (a) scatter plot of two features f_1 and f_2. The two classes are separable using these two features. (b) scatter plot of features f_1, f_2, and f_3 where four samples are misclassified because of f_3.

4.3.2 Toy Example of CE-LFS

A toy example of applying CE-LFS is illustrated in figure 4.5. In this example, as it is illustrated in figure 4.5(a), the two classes are separable in the two dimensional feature space (i.e., f_1 and f_2). We added another feature, f_3, to the data set with the same distribution, but we adjusted it for some of the samples such that the classifier assigned them to the wrong class (figure 4.5(b)). We performed kNN on the 3-dimensional data set, and four samples were classified incorrectly. Next, we performed CE-LFS using the classification results.

TABLE 4.1

Summary of learned relevant features for some samples from the toy data set

Sample	**Class**	**Predicted Class**	**Learned relevant features**		
			f_1	f_2	f_3
Sample-1	1	1	*		
Sample-2	1	1	*	*	
Sample-3	2	2		*	*
Sample-4	2	2		*	
Sample-5	1	2			*
Sample-6	1	2			*
Sample-7	2	1			*
Sample-8	2	1			*

Table 4.1 illustrates the results of CE-LFS for a few samples of the toy data set. For each sample, we showed its class, predicted class, and its learned relevant features. For instance, sample-1 is predicted correctly as class-1, and f_1 is its learned relevant

feature. In other words, f_1 explains the reason that sample-1 is classified correctly as class 1. Similarly, sample-3 is classified correctly as class-2 due to features f_2 and f_3. In other words, in the feature space formed by f_2 and f_3, sample-3 is closer to class-2 samples than class-1 samples. Sample-5, one of the samples that we adjusted its f_3, is classified incorrectly as class-2 due to feature f_3. The same explanation is valid for samples 6, 7, and 8.

We should note that although our MI-LSFS has been developed for MIL, we can apply it to single instance data by treating the samples as single instance bags. Thus, as in section 4.2, we validate CE-LFS for single instance data. The extension of that for MI data is straight forward.

CHAPTER 5

EXPERIMENTAL RESULTS

In this chapter, we illustrate the performance of our proposed methods on synthetic and real benchmark data sets. First, we use synthetically generated data, with known truth about relevant/irrelevant features for each sample, to demonstrate the performance of MI-LSFS in selecting the correct set of locally discriminative features. Next, to visualize the selected features and explore our method's explainability power, we generate testing data sets using the well-known MNIST data [139]. Finally, to validate the effectiveness of the information learned by MI-LSFS to select representative bags/instances as potential target concepts and the performance of the proposed MILES-LFS classification algorithm, we compare its accuracy and efficiency to MILES using benchmark MIL data sets.

Moreover, we investigate the performance of our local feature selection in identifying the significant features learned by deep convolutional networks (e.g., VGG19). In particular, we use the proposed Grad-SAM to visualize the selected features and highlight the most important part of a given sample learned by the deep model. Then, we investigate the classification power of the selected features.

Finally, we use synthetic and MNIST data set to explain the decisions made by trained models. We use our proposed CE-LFS method to explore and explain the

decisions made by a trained classifier.

5.1 Synthetic Data set

In this section, we evaluate the ability of MI-LSFS in selecting the locally important features using a synthetic data set, where we have the labels of features. We start by explaining our data generation process. Then, we report the performance of MI-LSFS using the generated data.

5.1.1 Synthetic Data set Generation

We generate a synthetic data that includes negative and positive bags. The positive bags are generated to include two target concepts that have different relevant features and many irrelevant ones. First, we generate a large number of negative instances. Each negative instance has 106 features sampled randomly using a uniform distribution in $[-10, 10]$ ($\mathcal{U}[-10, 10]$). Then, we generate two sets of positive instances that contain different relevant features and many irrelevant ones. The first 3 features of the first set are generated using a normal distribution with zero mean and unit standard deviation ($\mathcal{N}(0, 1)$). The remaining 103 features are generated using $\mathcal{U}[-10, 10]$ and are meant to be irrelevant. We will refer to this set of features as target concept 1. The second set of features is generated in a similar way, except that features 4, 5 and 6 are the relevant ones and are generated using $\mathcal{N}(0, 1)$. The remaining features ($[1, 2, 3, 7, 8, \ldots, 106]$) are generated using $\mathcal{U}[-10, 10]$.

Next, we generate 100 negative bags and 100 positive bags. Negative bags includes only negative instances. Each bag contains a random number ($\in [2, 9]$) of

instances, selected randomly from the set of negative instances. Positive bags contain a random number ($\in [1, 3]$) of positive instances and a random number ($\in [1, 6]$) of negative instances. The positive instances are sampled randomly from target concept 1 for 50 positive bags and from target concept 2 for the remaining 50 positive bags. The negative instances of all positive bags are sampled from the set of negative instances.

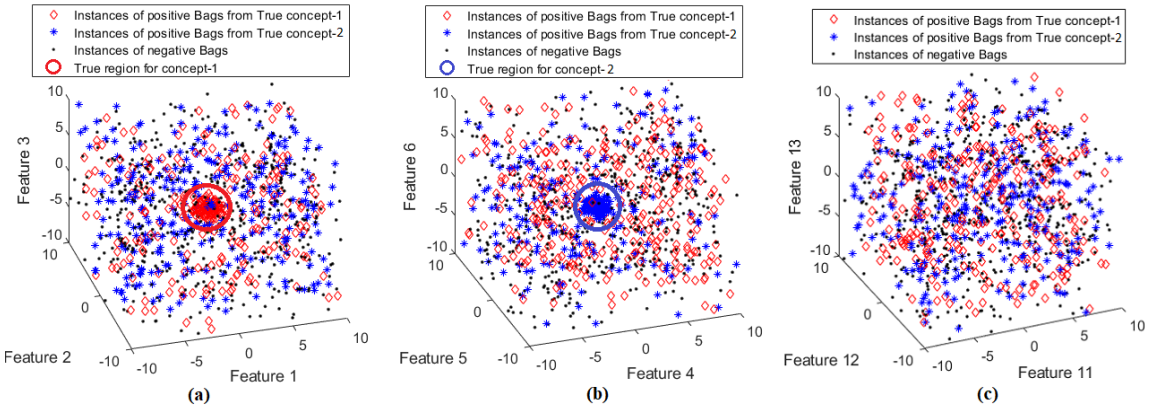


Figure 5.1: Representation of the synthetic data set using 3 sets of 3 different features. (a) using features F_1, F_2 and F_3 that are the discriminative features of the first concept group. (b) using features F_4, F_5 and F_6 that are the discriminative features of the second concept group. (c) using features F_{11}, F_{12} and F_{13} that are 3 random non informative features.

Figure 5.1, illustrates the generated data using 3 sets of 3 selected features. All the instances of all the positive bags that contain positive instances from target concept 1 and target concept 2 are illustrated with red diamond and blue asterisks, respectively. The negative instances are depicted with black dots. Figure 5.1(a) depicts all the instances in the feature space formed by the three relevant features

(F_1, F_2, and F_3) of target concept 1. The positive instances drawn from target concept 1 form a dense cluster, as shown by the red circle, while the other instances are distributed uniformly in the feature space regardless of their bag labels. A similar pattern can be observed in figure 5.1(b) for instances of target concept 2, where data are projected using the 3 relevant features of target concept 2 (F_4, F_5, and F_6). Figure 5.1(c), depicts the data projected using 3 non discriminative features (F_{11}, F_{12}, and F_{13}), where no dense clusters can be observed.

5.1.2 Synthetic Data set Results

Knowing the relevant features of each bag in the synthetically generated data allows the quantitative evaluation of the proposed MI-LSFS. For this data, we set $\alpha = 20$ (used in (3.7) and (3.8)). The actual number of informative features for the positive samples in this data set is 3. We let $\alpha = 20$ to have a higher value than the expected number of correct features in the data set.

MI-LSFS selects a set of relevant features for each bag. To quantify the method's performance in selecting the correct subset of features, we need a validity measure that represents the agreement of the selected features with the actual labels of the features. We use the rand index [140], which is a measure of agreement between two partitions or clusters. The adjusted rand index [141] is a corrected version of the rand index for the chance.

First, we apply MI-LSFS and identify the relevant features for each bag in the data. Then, for each positive bag, B_i^+, we use its selected feature vector, $f^{*(i)}$, and the vector that represents the ground truth labels of the features of B_i^+ to calculate its

rand index and adjusted rand index score. Table 5.1 summarizes the results where we show the average scores for positive bags that contain positive instances from target concept 1 and target concept 2, and the average scores for all the positive bags. For positive bags that contain instances from concept 1, the average rand index value is 0.9806, and the adjusted rand index value is 0.8451. These high values confirm the high level of agreement between the selected set of features for each bag and the actual labels of features (relevant or irrelevant).

TABLE 5.1

Agreement between the true relevant features of each positive bag and the relevant features learned by MI-LSFS

	Rand Index	**Adjusted Rand Index**
Target concept 1	0.9806	0.8451
Target concept 2	0.9804	0.8413
Average	0.9805	0.8432

5.2 MNIST Data set

In this section, we evaluate the performance of MI-LSFS in identifying discriminative features and explore its explainability power by visualizing the selected features using data sets generated from the benchmark MNIST data [139]. The MNIST data includes a large set of 28×28 pixel images of handwritten digits. It contains 60,000 training images and 10,000 testing images where each image represents a digit.

5.2.1 MIL MNIST Data sets

To generate an MIL version of the MNIST data, we treat each image (digit) as an instances. Thus, each bag is a set of digits. We generate binary classification data sets that include positive and negative bags. Positive bags are generated to include at least one instance from the positive concept and negative bags contain only negative instances. Considering each digit (i.e., $0, \ldots, 9$) as a concept, we can generate 10 different data sets by selecting one digit as the positive concept and the rest of digits as negative concepts at a time. We denote these data sets by DS_{digit} where $digit = 0, \ldots, 9$ is the positive concept used to generate the data set.

Each data set, DS_{digit}, contains 100 bags that where split equally into positive and negative bags. Each bag contains a random number ($\in [2, 9]$) of instances. Negative bags includes only negative instances that are sampled from the negative instances. Each positive bag contains a random number ($\in [1, 3]$) of positive instances, sampled from the positive concept instances, and a random number ($\in [1, 6]$) of negative instances sampled from the set of negative instances. For instance, in DS_0 the positive bags include at least one "0" digit and negative bags contain $1, 2, \ldots, 9$ digits.

5.2.2 Visualization of Selected Features

Since each image in the MNIST data has 28×28 pixels, we represent each image by a $28 \times 28 = 784$ dimensional vector where each component of the vector represents a pixel in the original image. Therefore, the dimensionality of instances

(i.e., number of features) in the generated data sets is 784. We apply MI-LSFS on all generated data sets ($DS_{digit}, digit = 0, \ldots, 9$).

In figure 5.2, we visualize some positive instances and their selected relevant features for all the data sets. Each row represents instances from one of the training data sets (DS_0 to DS_9). In each row, we show ten images, where each image represents one positive instance from one positive bag. On each image, we also show the selected relevant features (i.e., pixels) in red.

The first column of figure 5.2 depicts the visualization of a sample from each data set, $DS_{digit}, digit = 0, \ldots, 9$, and its selected features. Each image represents one positive instance from one positive bag. As it is illustrated, MI-LSFS identified a different set of features for each data set based on the positive concept representing the data set. In the first three rows of figure 5.2, we visualize ten positive instances from data sets DS_0, DS_1, and DS_2 where the digits have different writing styles. As it is illustrated, MI-LSFS selects a different set of features for different writing styles.

The first row depicted instances from DS_0, where digit 0 is the positive concept. For each instance, a different set of features are selected, representing the locally discriminant features. However, regardless of the different bags (that include different shapes of zeros), the features inside the 0 digit are selected as the most relevant one in characterizing this class. The consistency of having the selected features inside the zero digits confirms the agreement of locally selected features regardless of their shape. We should mention here that most of these features depict the parts that are not activated by positive instances (0s), but they were activated by the other instances (1-9). In other words, these features are selected based on the two criteria described

Figure 5.2: Instances drawn from data set ($DS_{digit}, digit = 0, \ldots, 9$) and their locally selected features. Each row contains ten images representing positive instances from ten sample bags and their selected features. The selected features are shown in red.

in section 3.1.1 (minimizing the distance between positive bags and maximizing the distance between positive bags and negative bags).

The second row of figure 5.2 represents examples from DS_1, where the positive

concept is digit 1. For these bags, MI-LSFS tends to select the features at the two sides of 1. Similar to DS_0, we observe that regardless of the shape and orientation of 1s, the selected features are consistently located at both sides of the digit. The remaining rows in figure 5.2 show similar observations for the remaining data sets. We should mention here that MI-LSFS selects an average of 20 features per bag for all the generated data sets.

5.2.3 Classification Results

In the next experiment, we check the discriminating power of the locally selected features when integrated into the classification. We generate 10 testing data sets $DST_{digit}, digit = 0, \ldots, 9$ following the same steps used to generate the training data sets, $DS_{digit}, digit = 0, \ldots, 9$. Then, we train two classifiers on each training data set using the benchmark classifier MILES [21], and the proposed MILES-LFS. Next, we use the trained models to label the bags in the testing data sets. Finally, we compute the area under the ROC curve (AUC) for each testing data set using both classifiers. For MILES-LFS, we construct a set of reduced prototypes by setting the number of selected instances from each bag, t, to 3 and using all the bags. As a result, only 57% of the data is used to train MILES-LFS. For the MILES algorithm, we set λ and σ^2 for each data set using 2-fold cross-validation on the training set.

In table 5.2, we report the AUC of MILES and MILES-LFS algorithms on the MNIST testing data sets. As it can be seen, MILES-LFS has similar or better results than MILES for all data sets except DST_7 and DST_8. We should note here that MILES-LFS was trained with only 57% of the training data while MILES used

TABLE 5.2

AUC of classification results on 10 testing MNIST data sets

Data set	**MILES**	**MILES-LFS**
DST_0	0.9560	0.9512
DST_1	0.9956	0.9980
DST_2	0.8980	0.8832
DST_3	0.9160	0.9184
DST_4	0.8928	0.9192
DST_5	0.8140	0.8452
DST_6	0.9708	0.9900
DST_7	0.9540	0.9360
DST_8	0.8048	0.7568
DST_9	0.8804	0.8836

all the training data. Furthermore, MILES-LFS used only a subset of features that MI-LSFS selected for each bag in the learning process. These results confirm that MILES-LFS has comparable accuracy to the MILES algorithm. Moreover, the locally selected features and the selected prototype instances can reduce the complexity of the training and provide a more explainable classification model.

5.3 Benchmark Data sets

In this section, we use other real benchmark MIL data sets to evaluate the performance of our methods in identifying sample dependent salient features. These data sets are selected from two common MIL applications: the drug activity prediction and image classification. Table 5.3 summarizes the statistics of the used benchmark data sets. We should note here that these data sets are the common real benchmark data sets used in the MIL literature to validate the new algorithms [14,17,18,73,131].

The first data set, MUSK1, is for the drug activity prediction adapted by [48] for the MIL framework. It represents a set of unique molecules (bags), and each molecule has different conformations (instances). Positive bags are the molecules with at least one shape of a molecule that binds strongly to a target protein. There is no shape of a molecule in negative bags that can bind tightly to the target [18].

TABLE 5.3

Summary statistics of benchmark data sets

Data set	**#Pos Bags**	**#Neg Bags**	**Avg #instances**	**# Features**
MUSK1	47	45	5.17	166
Elephant	100	100	6.60	230
Fox	100	100	6.96	230
Tiger	100	100	6.10	230

The Elephant, Fox, and Tiger data sets are adapted for the MIL framework in [51]. They are subsets of the COREL data set where each bag is a representation of an image. Images (bags) are broken into segments (instances) represented by color, texture, and shape descriptors. Each data set contains 100 positive bags from the target class animal and 100 negative bags drawn randomly from the other animal classes. The positive bags consist of instances with at least one instance from the target animal (positive instances), and the negative bags contain other animals (negative instances). In the next sub-sections, we investigate the performance of MILES-LFS using benchmark data sets.

5.3.1 Feature Selection by MI-LSFS

MI-LSFS identifies a subset of relevant features for each bag where the number of selected features for each bag can vary. We investigate the performance of MI-LSFS in reducing the dimensionality of features per bag. In particular, we inspect the number of features selected by MI-LSFS for each bag. To investigate this, we perform MI-LSFS on all the benchmark data sets using 4-fold cross-validation. We repeat each experiment 7 times using different random initialization. For each run, we calculate the number of selected features for each bag. Table 5.4 summarizes the results where we show the total number of features in each data set and the minimum, maximum, and average number of selected features using MI-LSFS.

TABLE 5.4

Summary statistics of number of selected features per bag on benchmark data sets

Data set	# of Features	Minimum # of selected features	Maximum # of selected features	Average # of selected features
MUSK1	166	2	28	8
Elephant	230	4	64	23
Fox	230	2	80	20
Tiger	230	2	60	14

For instance, as it is shown in table 5.4, the Tiger data set contains 230 features and MI-LSFS selects an average of only 14 relevant features per bag out of these 230 features. The minimum and the maximum number of selected features on the Tiger data set are 2 and 60, respectively. The statistics show similar results for the other

benchmark data sets. These results indicate the capability of MI-LSFS in reducing the dimensionality of instances within bags by identifying the most relevant features.

5.3.1.1 Maximum Number of Selected Features

In this experiment, we investigate the average number of selected features as we vary the maximum number of features that can be selected, (i.e., as we vary the parameter α), for the Tiger data set. We set α to the values in the range of 10 to 230 (maximum number of features in Tiger data set) by an increment of 10. Then, we calculate the average number of features selected by MI-LSFS for each sample. The results are shown in figure 5.3. Using this plot, we can obtain the saturation value for a given data set. For the Tiger data set, it can be seen that the 14 is the saturation value.

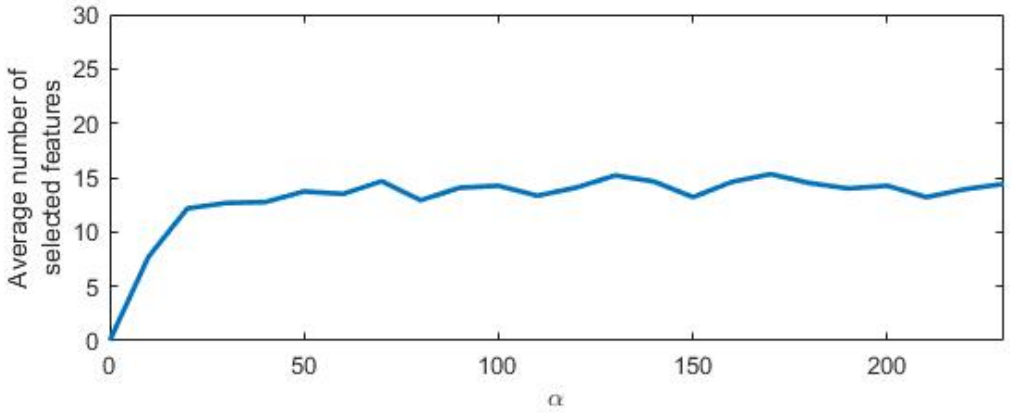


Figure 5.3: Averaged number of selected features on Tiger data set when we set α to the values in the range of 10 to 230 by an increment of 10.

5.3.2 Prototype Instance Selection using MILES-LFS

In this sub-section, we investigate the performance of MILES-LFS when we reduce the number of instances that can be selected as prototypes. To explore this, we set a parameter, t, to limit the maximum number of instance prototypes that can be selected from each bag. In particular, for a given bag B_i, we use equation (4.2) to score all of its instances and select the top t instances as instance prototypes. Next, we train and test the MILES-LFS classifier on the Tiger data set as we let $t \in \{1, 2, 3, 4, 5, 6\}$. For comparison purposes, we also trained and tested MILES-LFS using all the instances of all the bags. For each value of t, we train and test MILES-LFS using 4-fold cross-validation. For each fold, we repeat the experiment 7 times with different random initialization. For each run, we compute the area under the ROC Curve (AUC).

The results are summarized in figure 5.4. The horizontal axis represents the parameter t, and "All" is where all the instances of all the bags are used to train the classifier. The vertical axis depicts the summary statistics for AUC where it summarizes the average and standard deviation calculated using all the 7 runs for the given t value. As it is shown, using only the top one instance per bag, $t = 1$, leads to an average AUC of 0.79. By increasing t, the average AUC value increases until t reaches 3. Increasing t further does not improve the performance. Thus, by using only about 65% of the data set instances to train the classifier ($t = 4$) we achieved the same performance as when all of the training data instances were used. In the Tiger data set, the average number of instances is 6.10. Moreover, the small value

of standard deviation in all the experiments (for different values of t) indicates the consistency of results for different random initialization. This experiment confirms the MILES-LFS strength in identifying the most informative instance prototypes of the bags and the robustness of the results.

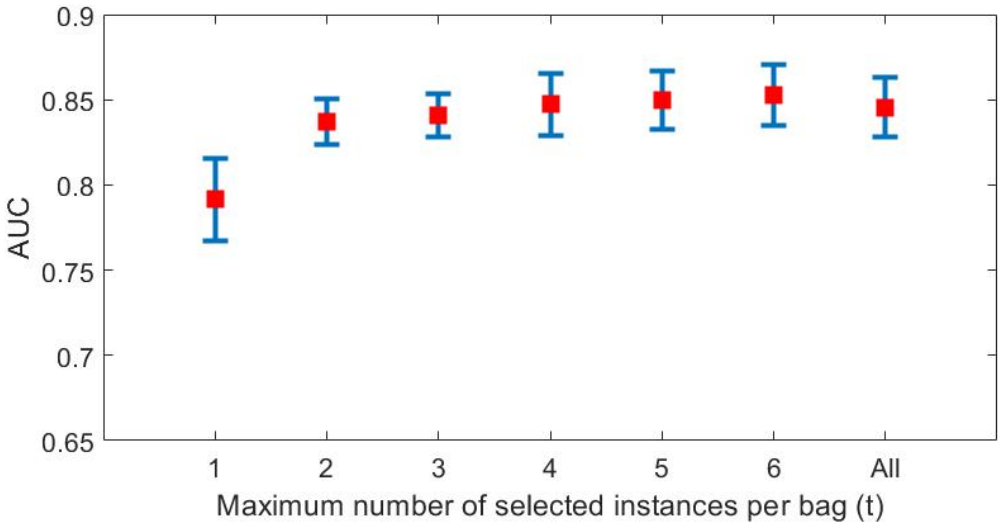


Figure 5.4: Performance of MILES-LFS using the top t instances per bag on the Tiger data set. This data has an average of 6.10 instances per bag.

5.3.3 Prototype Bags and Instances Selection using MILES-LFS

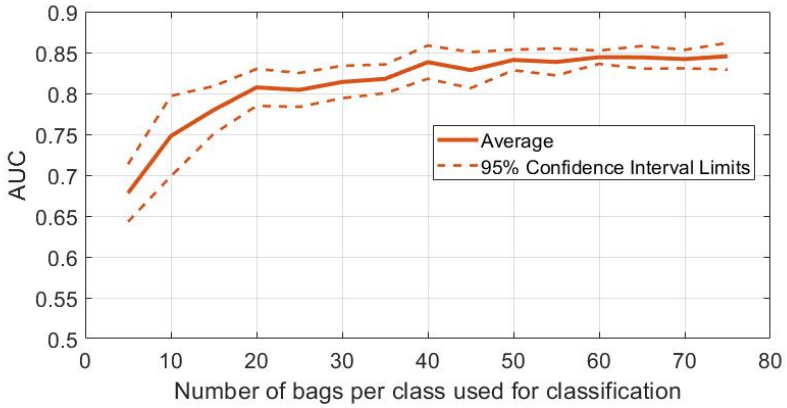
In this experiment, we investigate the performance of MILES-LFS when first, a reduced set of prototype bags is selected. Then, for each selected bag, we identify its subset of representative instances. To explore this, we follow the approach discussed in section 4.1 to assign a score to every bag B_i in the training data. Using this score, we select the top b informative bags. Next, using the instance scores, we select the top t informative instances of each bag.

Figure 5.5 shows the AUC of MILES-LFS on the Tiger data set. The results,

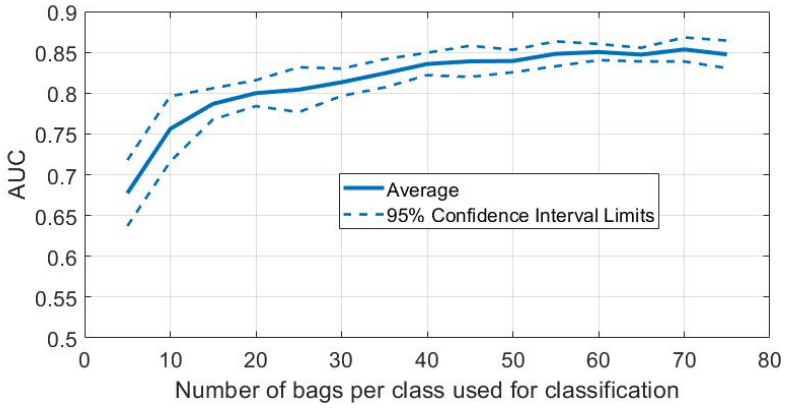
those reported in the previous section, are calculated using a 4-fold cross-validation and repeating the experiment for 7 different random initialization. The vertical axis represents the AUC value, and the horizontal axis shows the number of bags per class used for the classification (from 5 to 75 with steps of 5). In figure 5.5(a), we show the AUC of MILES-LFS algorithm when no instance selection was applied. That is, we use all of the instances for each selected bag. The average and the 95% confidence intervals are reported for the seven different runs.

In 5.5(b), we used the top 4 instances per bag (i.e., $t = 4$) and varied the number of bags as in figure 5.5(a). By increasing the number of bags in the learning process, the classification performance increases until it reaches 55 bags, where it remains constant. Therefore, by using 55 bags per class, and their top 4 instances, we obtain a slightly better classification accuracy than the one obtained using all data. In other words, by using only 48% of the instances ($\textit{selected instances} \ / \ \textit{total instances} = (110 \times 4)/(6.1 \times 150)$), we obtain slightly better results than using 100% of the instances.

We have investigated alternative approaches to identify bag prototypes and instance prototypes, such as using different variations of distance metrics in the learning process and to score the instances. Furthermore, we investigated alternative measures such as MCC, recall, and F1 score to form the clusters of bags and rank them to identify the bag prototypes. Based on the results, our current approach had the best performance.



(a)



(b)

Figure 5.5: AUC of classification results on the Tiger data set: (a) all instances of the bags are used. (b) maximum number of instances per bag used is set to 4 ($t = 4$).

5.3.4 Effect of Instance Selection on the Computational Efficiency of the Classifier

As we discussed in section 4.1, one of the main drawbacks of MILES is that it does not scale to very large data sets. Moreover, when the number of instances per bag is very large, MILES projects the data to a very high dimensional space. In this case, the 1-Norm SVM may not be able to identify the few target concepts.

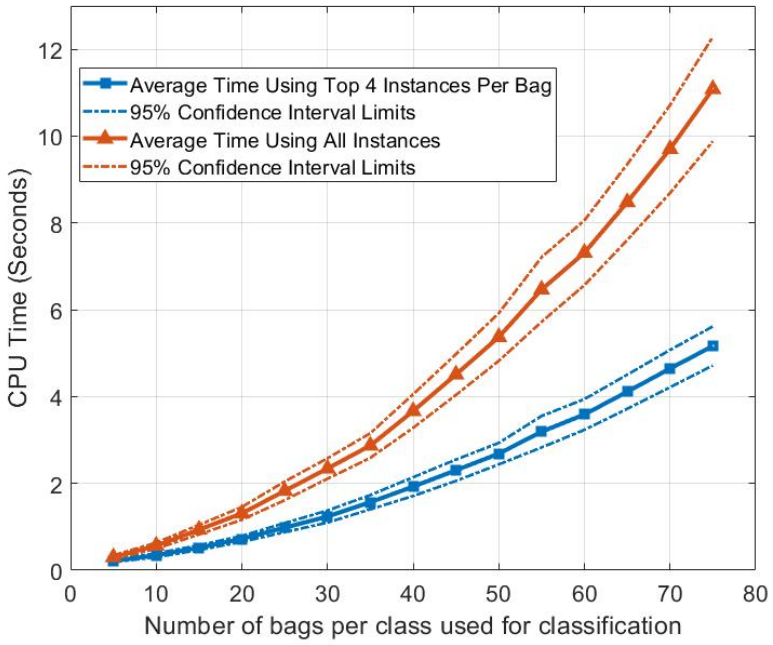


Figure 5.6: Comparison of the learning time when using all the instances and when using top 4 instances per bag

In figure 5.6, we report the learning CPU time of MILES-LFS for the Tiger data set. The orange line shows the average learning time in seconds where we used all the instances, and the blue line depicts the average learning time where we used the top 4 selected instances per bag. We vary the number of bags used to train the classifier from 5 to 75 per class with steps of 5. Increasing the number of training bags leads to an increase in the learning process for the 1-Norm SVM. However, our instance selection approach leads to a significant decrease in the learning process while it reaches the same classification results.

As reported in the previous section (figure 5.5), MILES-LFS can reach its best accuracy using 40 bags and 4 instances per bag. Using this setting, the learning time

is 2 seconds, compared to 11.08 sec ±1.21 for MILES when all bags and all instances are used (last point in orange curve in figure 5.6).

5.3.5 Classification Results

In this experiment, we compare the classification accuracy of MILES-LFS and the standard MILES algorithm using the benchmark data sets in table 5.5. For the COREL data sets, we select a set of reduced prototypes by setting the number of bags, b, to 60 and the number of instances from each bag, t, to 4. For the MUSK1 data set, we used $t = 3$, and we used all the bags as the size of the training data set is small. We set the parameters according to 2-fold cross-validation on the training set.

TABLE 5.5

Comparison of the AUC of MILES and MILES-LFS using 4 benchmark data sets

Dataset	**MILES**	**MILES-LFS**
Elephant	0.8908 ± 0.011	0.8882 ± 0.010
Fox	0.6588 ± 0.029	0.6639 ± 0.019
Tiger	0.8828 ± 0.023	0.8503 ± 0.011
MUSK1	0.8831 ± 0.010	0.8782 ± 0.017

In table 5.5, we report the average AUC of MILES and MILES-LFS algorithms on the benchmark data sets for seven runs. For each run we use 4-fold cross-validation. We report the average and standard deviation across all runs. As it can be seen, MILES-LFS has similar results to MILES for all data sets except Tiger. It is also

worth noting that MILES-LFS uses only 52% of the training data to train the model. These results confirm that our proposed algorithms are computationally more efficient and results in comparable accuracy. Moreover, the selected relevant bags and their prototype instances can provide more explainable classification models.

5.4 Using MI-LSFS to Explain CNN Decisions

In this section, we evaluate the performance of MI-LSFS in identifying the informative features among the features extracted from a CNN network. We investigate the explainability and classification power of the learned features. In the following experiments, we use two data sets to validate our results: the public Vehicle Make and Model Recognition (VMMR) data set [142], and the Dogs vs. Cats data set [143]. First, we train a CNN model on each data set using transfer learning and the VGG19 model as the base. Then, we extract features from the trained model and form a training data set using the extracted features. Next, we apply our MI-LSFS algorithm on the training data set to identify the informative features for each input sample. Finally, we investigate the classification power of the selected features by looking at the neighbors of a target sample and using its selected features. Moreover, we use our visualization method (Grad-SAM) to visually explain the learned features.

5.4.1 Data sets

5.4.1.1 Dogs vs. Cats Data set

The first data set, Dogs vs. Cats, contains a large set of dogs and cats images that was created by the Microsoft Research team to compare the performance of humans and machines in discriminating between cats and dogs. The images have a wide diversity in backgrounds, angles, poses, and lighting, which makes the classification task challenging. The ASIRRA test [143] is created using that. Since then, this data set has been studied in the machine learning attacks domain [144]. Moreover, it is used in a Kaggle competition.

The Dogs vs. Cats data set contains 25,000 training images and 10,000 testing images. Each image has a size of 150×150 pixels. We use a subset of 2,000 training images and 1,000 testing images.

5.4.1.2 VMMR Data set

The Vehicle Make, and Model Recognition (VMMR) data set [142] is a large-scale and diverse data set that includes car images from different makes and models. It was used to develop and validate different classification and object detection models. Moreover, it was used to train feature extractor models for different applications. The data includes car models manufactured between 1950 to 2016. It contains 291,752 images from 9,170 different classes. For our experiments, we use a subset of the VMMR data that includes three classes of vehicles (Pickup, SUV, and Sedan). For this data, we also fix the image sizes to 150×150 pixels. This subset contains 15,000

training images and 6,000 testing images.

5.4.2 Transfer Learning and Feature Extraction

To build our models, we use transfer learning to fine-tune the parameters of a pre-trained VGG19 [81] network on each data set. To adapt the VGG19 architecture, we exclude the top layers, the fully connected layers. Then, we add two fully connected layers with 256 and 64 nodes, followed by a dense output layer with one node for each class. Moreover, we fine-tune the weights of the last convolutional block. Figure 5.7 depicts the architecture of our adapted network. Using this architecture, we build a model for each data set by using the training data sets to learn and adapt the weights. We used 15,000 of training images from VMMR data set to train the network. For the Dogs vs. Cats data set, we used the subset of 2,000 of training images. The accuracy of the trained models on the testing data sets are 90.22% and 93.51%, for the VMMR and Dogs vs. Cats data sets, respectively.

After building the models, we use them as feature extractors to extract features from the data sets. We feed each image into the learned model and extract the computed values at the last fully connected layer as features. Thus, each image is represented by a 64-dimensional feature vector. Finally, we make new training and testing data sets by mapping the respective images to their extracted features.

We use the new training data sets as input to the MI-LSFS to identify the relevant features for each training sample.

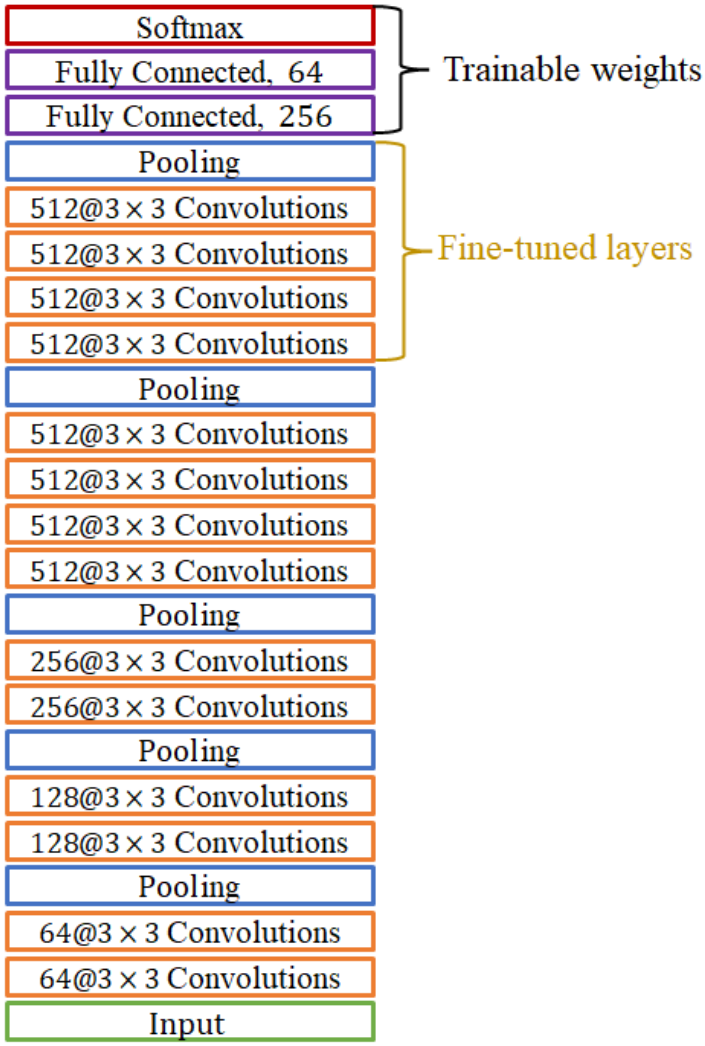


Figure 5.7: The network architecture for the transfer learning based on VGG19 architecture. Fully connected layers and the classification layer are changed to adapt the architecture on the data sets.

5.4.3 Applying MI-LSFS

Using the mapped features of the VMMR and Dogs vs. Cats images, we apply MI-LSFS to identify the local features for each sample. We perform MI-LSFS on a subset of the new training data sets (1050 images from VMMR and 1000 images from Dogs vs. Cats Data sets). For these experiments, we set α based on the saturation

value of the training data set. MI-LSFS selects an average of 28(or 43.7%) and 39(or 60.9%) features for samples of Dogs vs. Cats and VMMR data sets, respectively.

5.4.4 Illustrating the Relevancy of the Learned Features

To investigate the identified relevant features for some samples, we use the kNN classifier as it is easy to interpret and has a high performance on the testing data sets. For this experiment, we are interested in identifying the nearest neighbors of samples to study how informative the selected features are. For a given testing sample, which we call the query-image, we identify its k nearest neighbors from the training data set. We apply the kNN two times: The first time we use all the features to identify the nearest neighbors of the query-image; and the second time we use only the learned relevant features of each training sample obtained from the MI-LSFS.

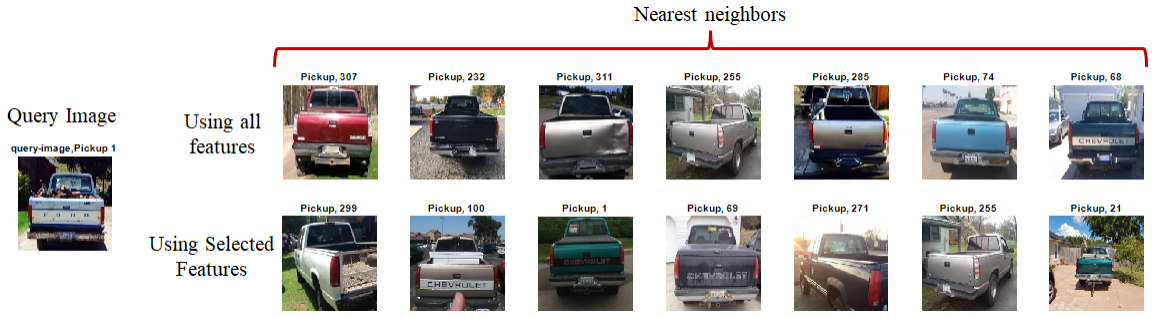


Figure 5.8: The nearest neighbors of a query-image-1. The first row shows the nearest neighbors using all the features. The second row shows the nearest neighbors considering only the learned features of each training samples in distance calculation. The class label of each image is displayed above the image.

In figure 5.8, we show the seven nearest neighbors of a query-image from the VMMR data set. We refer to this image as query-image-1. Query-image-1 is an

easy to predict Pickup car. Thus, regardless of using all the features or the selected features, all of its 7 closest neighbors are from the same class as the query image (i.e., Pickup).

The second example, query-image-2, is a hard to detect Sedan car. In figure 5.9, we display the query-image and its closest neighbors. As shown, when all features are used, many of the closest neighbors show the car's rear, and there are images from incorrect class in the closest neighbors (i.e., images showing the rear of a Pickup). This makes predicting the correct class of query-image-2 hard. However, when using only the selected features in the distance calculation, all the closest neighbors were from the same class as the query-image-2 (i.e., Sedan) and with different angles, depth, rotation, and models. Thus, the model can identify the correct class with perfect confidence.

Figure 5.9: The nearest neighbors of a query-image-2. The first row shows the nearest neighbors using all the features. The second row shows the nearest neighbors considering only the learned features of each training samples in distance calculation. The class label of each image is displayed above the image. Nearest images from the incorrect class are indicated with red labels.

Figure 5.10 shows the result for query-image-3, which is a Pickup. As it is illustrated, when we used all the features, the closest neighbors were very similar to the query image but from the wrong class (i.e., SUV). Thus, the kNN classifier failed to predict its correct class when it used all the features in the distance calculation. However, when we use only the selected features, all of the closest samples were from the same class as the query-image-3 (i.e., Pickup). In this case, the kNN predicted the correct label of query-image-3.

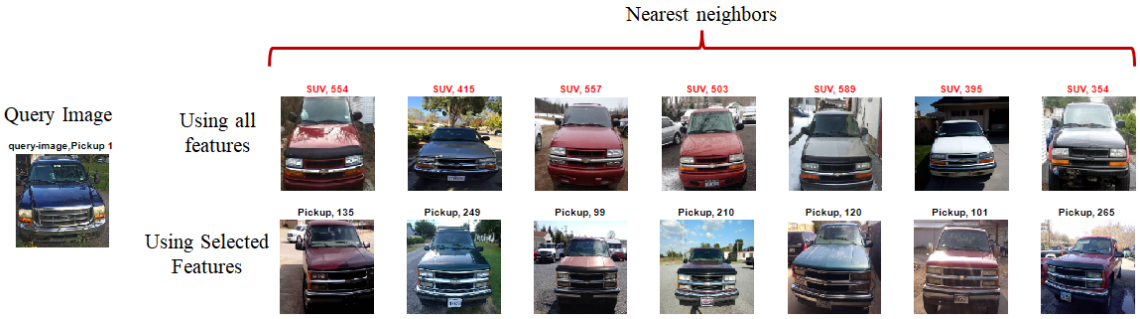


Figure 5.10: The nearest neighbors of a query-image-3. The first row shows the nearest neighbors using all the features. The second row shows the nearest neighbors considering only the learned features of each training samples in distance calculation. The class label of each image is displayed above the image. Nearest images from the incorrect class are indicated with red labels.

In conclusion, integrating the selected features in building the model makes the closest neighbors more similar to the query image. Moreover, using a smaller number of features can make the similarity more semantic and does not require the objects in the images to have similar viewing angle, depth, rotation, etc.

To illustrate how using subsets of relevant features helps the kNN classifier,

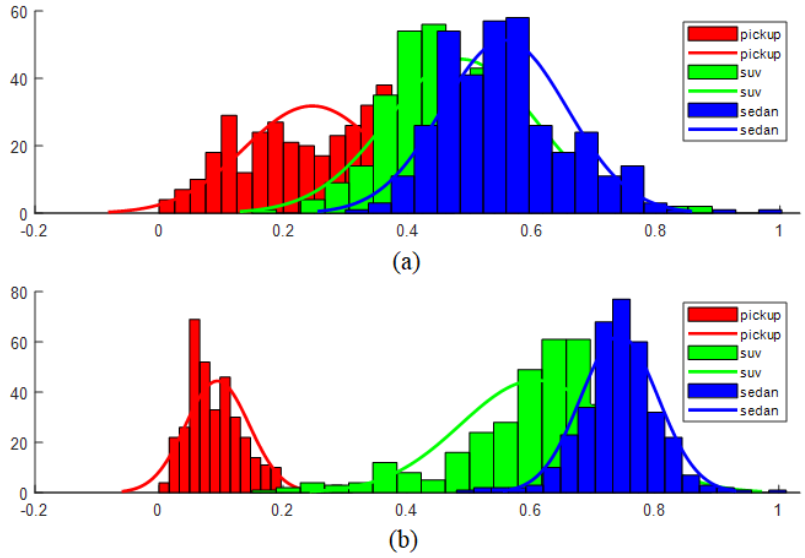


Figure 5.11: Histogram of distances of query-image-1 from the training samples color coded for the three classes of VMMR data set. (a) using all the features. (b) using the selected features of training samples.

we compare the distances obtained for the query images when we use all the features to those obtained when we use only the relevant ones. Figures 5.11, 5.12 and 5.13 depict the histograms of the distances of the query images from the training samples, when we use all/subsets of features. For instance, figure 5.11 illustrates the distances to query-image-1. Each color shows the histogram of distances of query-image-1 to the training samples of one class. For instance, the red histogram shows the distances between query-image-1 and the training samples of class Pickup. Figure 5.11(a) and 5.11(b) depict the histograms when using all the features and when using the selected features in the distance calculation. As it can be seen, integrating the selected features helps separate the distribution of distances of different classes. Moreover, it makes the samples from the Pickup class (correct label of query-image-1) closer to the query-

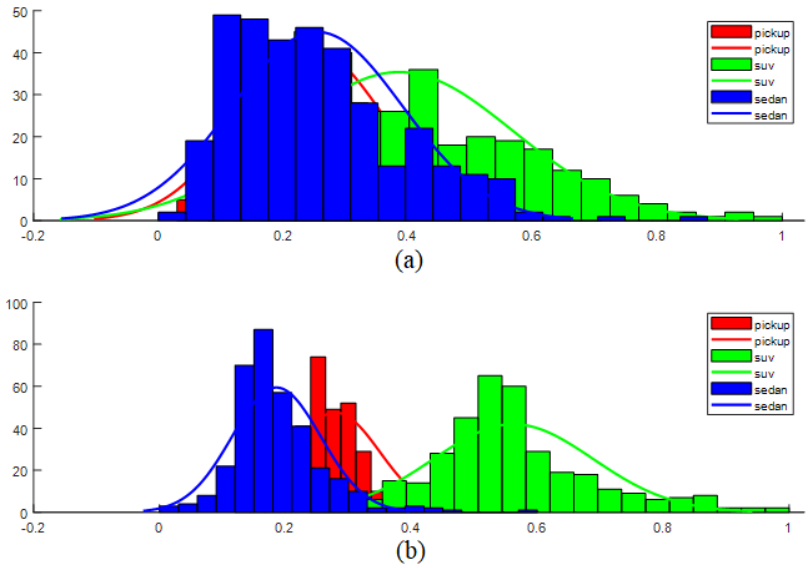


Figure 5.12: Histogram of distances of query-image-2 from the training samples color coded for the three classes of VMMR data set. (a) using all the features. (b) using the selected features of training samples.

image-1 while making the distances of the samples from the incorrect classes, SUV and Sedan, farther.

Figure 5.12 illustrates the distances of query-image-2. As it can be seen in figure 5.12(a), the distributions of the three classes are overlapping, and it is hard to identify the correct class. However, using the selected features in figure 5.12(b), the distributions are separated and the samples from the correct class (i.e., Sedan) are closer to the query image. This makes the classifier predict the label more confidently.

Figure 5.13 illustrates the distances of query-image-3. Here again in figure 5.13(a), the distribution of the three classes, especially Pickup and SUV, are overlapping. In this example, the query image is very similar to both SUV and Pickup classes. This makes the separation of the two similar classes very hard. However,

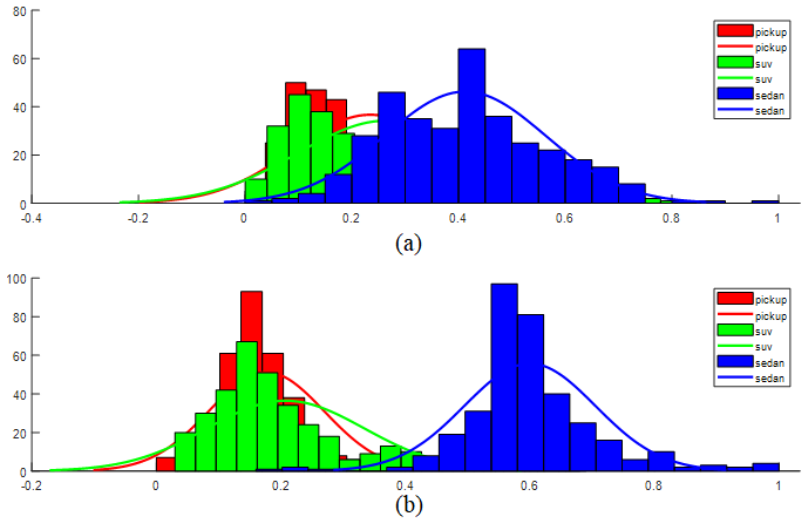


Figure 5.13: Histogram of distances of query-image-3 from the training samples color coded for the three classes of VMMR data set. (a) using all the features. (b) using the selected features of training samples.

Integrating the selected features, as it can be seen in 5.13(b), helps the classifier to predict the correct class. The above examples confirm our claim that the learned relevant features help in identifying the semantically most similar images. A similar trend was observed for the Dogs vs. Cats data set.

5.4.5 Visualization by using Grad-SAM

In this section, we use our proposed visualization method, Grad-SAM, to highlight the discriminative part of the images. Figure 5.14 illustrates some images from the VMMR data set and their Grad-SAM visualization. For each image, we apply Grad-SAM using its learned features to visualize and highlight its salient parts. As it can be seen, for each image, the highlighted parts show different segments and with different concentrations.

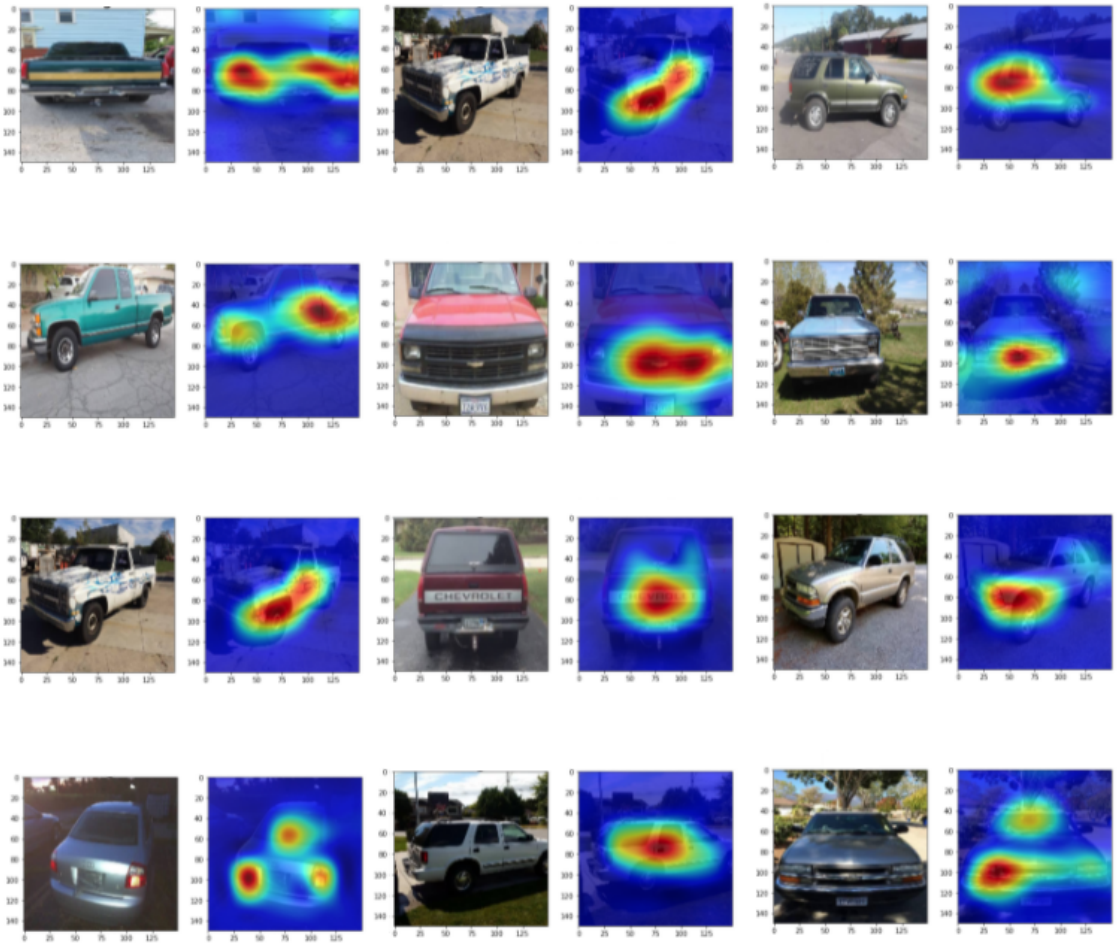


Figure 5.14: Visualization of some images from VMMR data set using Grad-SAM

Figure 5.15 depicted some images from the Dogs vs. Cats data set paired with their Grad-SAM visualization. The visualizations confirm that MI-LSFS is selecting salient local features for each sample. Moreover, it confirms the performance of Grad-SAM in visualizing the subset of selected features which helps to validate the locally learned features visually.

In Figure 5.16, we depicted two sample images that were incorrectly classified when using all the features. However, using the selected features, the samples are correctly classified. For each image, we visualize the sample, and its visualization

Figure 5.15: Visualization of some images from Dogs vs. Cats data set using Grad-SAM

using all the features and the selected features. As it can be seen in 5.16(a), when we used all the features, the most significant part that is highlighted are irrelevant to the target concept (dog). However, using the subset of selected features, the head of the dog is highlighted. In 5.16(b), by using all the features, the front of the vehicle (the grill and light) are highlighted more than the other parts (back of the vehicle). However, by using the learned features, the concentration of the highlighted parts changed. As a result, the vehicle's back, the discriminative part between Pickup and SUV, is highlighted more than the front of the vehicle, the similar parts between Pickup and SUV. Thus, the set of learned features helps the classifier

to discriminate between Pickup and SUV more efficiently. These experiments confirm the efficiency of Grad-SAM in visualizing the locally learned features. Moreover, it confirms that MI-LSFS learns a salient subset of features with higher explainability and classification power.

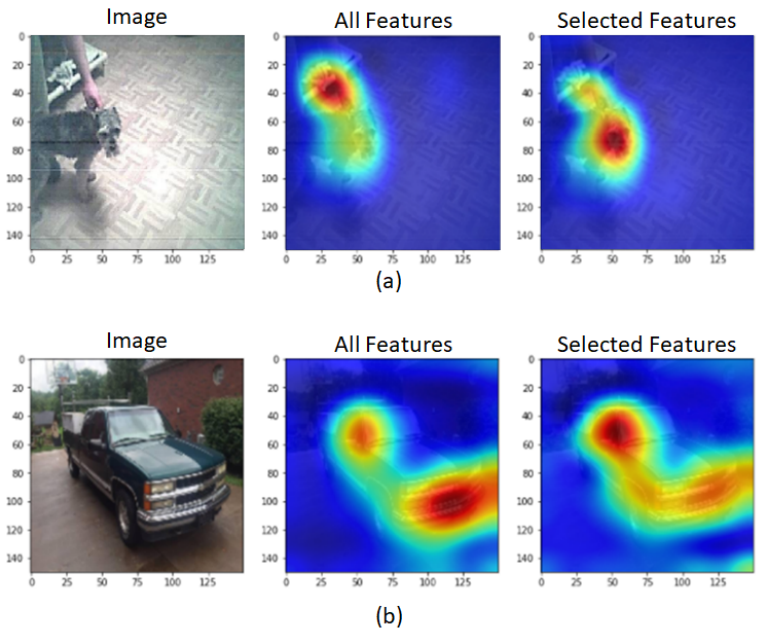


Figure 5.16: Visualization of images using all the features and selected features: (a) a sample from Dogs vs. Cats data set. (b) a sample from VMMR data set.

5.4.6 Classification

In this experiment, we check the discriminating power of the locally selected features when integrated into the classification. We compare the classification accuracy of the trained CNN model, kNN using all the features, kNN using the selected features, and our MILES-LFS algorithm. For the kNN models, we set k to the default value of 10. We learned the parameters of MILES-LFS algorithm using 2-fold

cross-validation on the training data sets.

In table 5.6, we summarized the classification results. The results indicate that the kNN classifier, that uses all the features, has similar accuracy to the CNN model. As it can be seen, kNN + FS also has comparable accuracy to the CNN classifier for both data sets, and MILES-LFS has the highest performance for both data sets. We should note here that kNN + FS and MILES-LFS were trained using the reduced set of features. The high classification performance of the models where they integrate the selected features confirms the performance of our approach in learning the informative set of features for each object. Moreover, the results confirm that we obtained higher accuracy and higher explainability power with lower computational complexity by using the learned features.

TABLE 5.6

Comparison of the accuracy of models using VMMR and Dogs vs. Cats data sets

Method	**VMMR**	**Dogs vs. Cats**
CNN (VGG19)	90.22	93.51
kNN (All features)	90.28	94.10
kNN + FS	90.31	93.32
MILES-LFS	90.63	94.18

5.5 Classifier Explanation using Local Feature Selection

In this section, we investigate our proposed CE-LFS method in explaining the decisions made by trained models. In particular, we use CE-LFS to explain the decisions made by classifiers.

We generate a synthetic data set, similar to the toy example discussed in 4.3.2, to investigate our proposed CE-LFS method. Our synthetic data set contains two classes of samples, and the classes are separable in a 4-dimensional feature space (i.e., f_1, f_2, f_3, and f_4). We generate the first class samples using a normal distribution with mean $\mu_1 = 5$ and standard deviation $\sigma_1 = 1.8$, $\mathcal{N}(5, 1.8)$, and second class using a normal distribution with mean $\mu_2 = 1.8$ and standard deviation $\sigma_2 = 1.8$, $\mathcal{N}(8, 1.8)$.

We select four samples from each class and adjust a combination of two features (e.g., f_2 and f_3), such that the kNN classifier cannot assign them to their correct class in the 4-dimensional feature space. However, the classifier can correctly predict these samples using the other features, e.g., f_1 and f_4. We set the number of samples in the synthetic data set to 400 (200 per class). Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ denote this data set where N is the number of samples.

5.5.1 Form the Data set and Applying ML-LSFS

To form the new data set, $\mathcal{D}' = \{(x_i, y_i')\}_{i=1}^{N}$, we apply kNN classifier to the 4-dimensional feature space. Next, we form the confusion matrix using the correct labels and predicted labels of the model. Using the confusion matrix, we assign a new label, y_i', to each sample. Next, we perform MI-LSFS on $\mathcal{D}'$. Table 5.7 illustrates the results for a few samples.

5.5.2 Explaining the Results

In table 5.7, we report the results of some samples. For each sample, we showed its class, predicted class, and learned relevant features. For instance, sample-

TABLE 5.7

Summary of learned relevant features for some samples from the synthetic data set

Sample	**Class**	**Predicted Class**	**Learned relevant features**			
			f_1	f_2	f_3	f_4
Sample-1	1	1	*	*		
Sample-2	1	1	*			
Sample-3	1	1		*		*
Sample-4	1	1	*		*	
Sample-5	2	2		*		
Sample-6	2	2			*	*
Sample-7	2	2		*	*	
Sample-8	2	2	*		*	
Sample-9	1	2	*	*		
Sample-10	1	2			*	*
Sample-11	1	2	*	*		
Sample-12	1	2			*	*
Sample-13	2	1	*			*
Sample-14	2	1			*	*
Sample-15	2	1			*	*
Sample-16	2	1		*	*	

1 is predicted correctly as class-1, and features f_1 and f_2 are its learned relevant features. In other words, features f_1 and f_2 explain the reason that sample-1 is classified correctly as class-1. Similarly, sample-6 is classified correctly as class-2 due to features f_3 and f_4. The interpretation is that sample-6 is closer to samples of class-2 (its correct class) than the samples of class-1 (its incorrect class) in the feature space formed by using f_3 and f_4. The same explanation is valid for the other correctly classified samples.

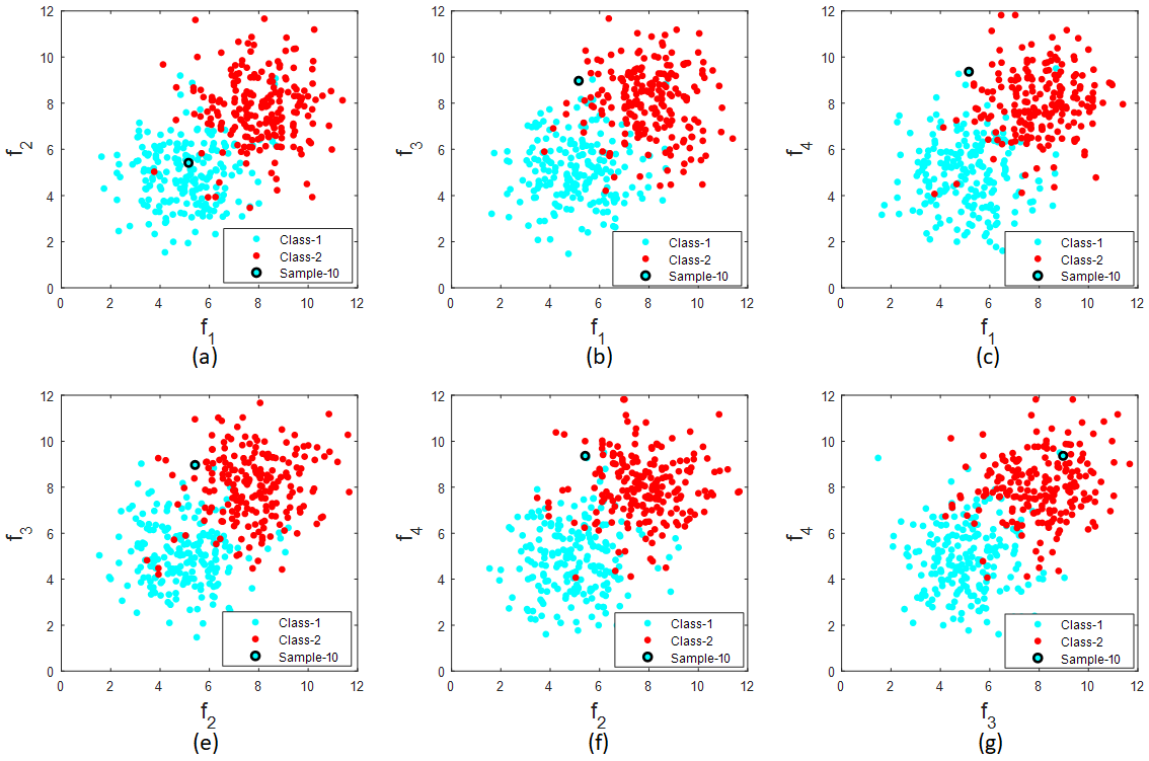


Figure 5.17: Visualization of sample-10. Each sub figure, illustrates the data set in a 2-dimensional feature space.

Sample-10, one of the samples that we adjusted two of its features (i.e., f_3 and f_4), is classified incorrectly as class-2. As it is illustrated in table 4.1, features f_3 and f_4 are selected for sample-10. The interpretation is that these features are the reason that this sample is classified incorrectly. Figure 5.17, depicts six figures, where each sub figure displays a scatter plot of the data set using one of the 2-combinations of the four features. The samples are color-coded based on their class labels, and sample-10 is highlighted with a black circle. As can be seen in figure 5.17(a), sample-10 which is classified incorrectly by the classifier can be correctly classified as class-1 by using features f_1 and f_2. In the feature space formed by f_1 and f_2, sample-10 has the same distribution as the other samples of class-1 (its correct class). Moreover, as depicted

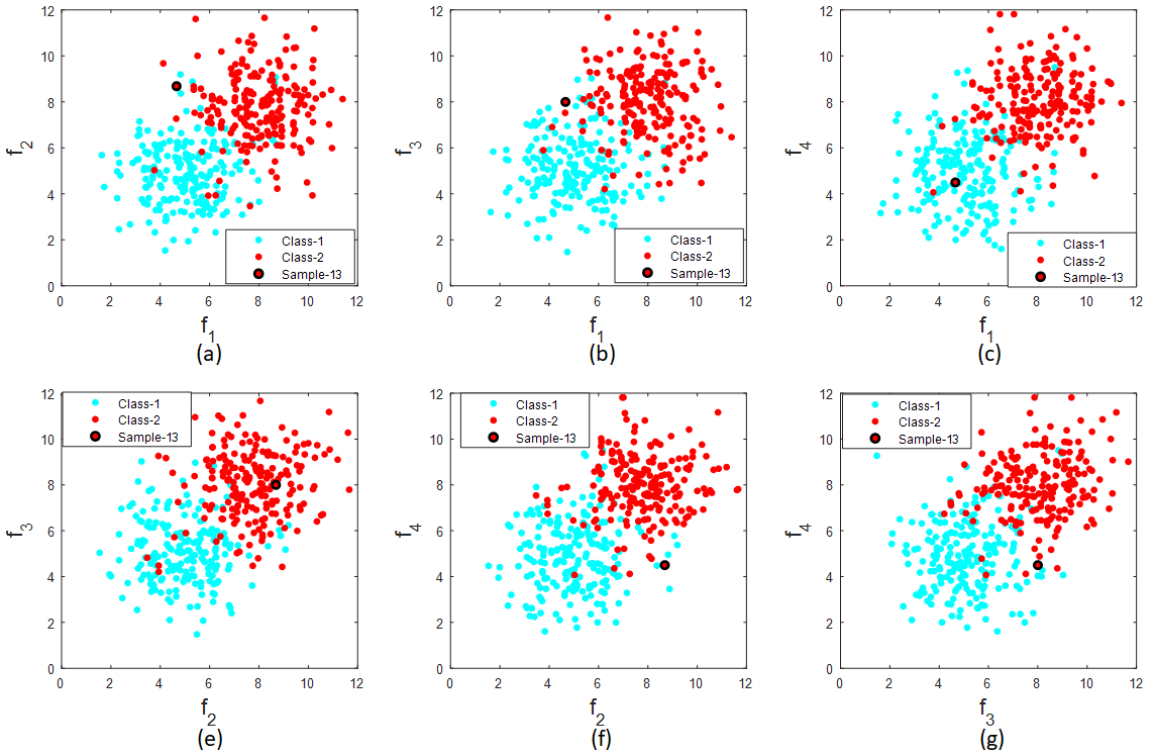


Figure 5.18: Visualization of sample-13. Each sub figure, illustrates the data set in a 2-dimensional feature space.

in 5.17(g) features f_3 and f_4 are the reason why this sample is classified incorrectly. In the feature space formed by f_3 and f_4, sample-10 has the same distribution as class-2 samples (its incorrect class).

In figure 5.18, we highlighted sample-13 that is classified incorrectly as class-1. As it can be seen in figure 5.18(e), sample-13 can be correctly classified as class-2 by using features f_2 and f_3. Moreover, as it is depicted in 5.18(c) features f_1 and f_4 are the reason that sample-13 is classified incorrectly. The same explanation is valid for the other samples.

CHAPTER 6

CONCLUSIONS AND POTENTIAL FUTURE WORK

6.1 Conclusions

We proposed a local feature selection method for multiple instance learning. The proposed algorithm, MI-LSFS, searches for the relevant features within each bag in the feature space. We investigated and illustrated the performance of MI-LSFS in selecting the relevant features on synthetic as well as real benchmark data sets. The rand index score of selected features and visualization of selected features using synthetic and real data sets confirmed that MI-LSFS learns the relevant set of features for each bag.

As an application of MI-LSFS, we have proposed a new classification method for multiple instances learning, called MILES-LFS, which explores the information learned by MI-LSFS during the feature selection process. We investigated the performance of MILES-LFS on several real benchmark data sets. Our results indicate that by using only 52% of benchmark data sets to train MILES-LFS, the classification accuracy is comparable to that of the MILES algorithm, which uses all the training data. The results also confirm that using the information learned by MI-LSFS, we can select a small subset of representative bags and instances. Furthermore, the reduced set of prototypes significantly reduces computational time without affecting the clas-

sification accuracy. For instance, for one benchmark data, our results indicated that MILES-LFS is 5.5 times faster than MILES while maintaining comparable accuracy.

Convolutional Neural Networks (CNNs) are a category of multi-layer neural networks that have been used in image recognition, image classification, object detection, image captioning and other applications. During the last few years, due to the increasing computational power, the increasing amount of data available, and advances in training the neural network models, these models become the standard in finding complex patterns. However, explaining the decisions made by these black boxes remain a challenging task. In an effort to gain an understanding of a CNN model, we investigated the features extracted by trained CNN models. Another application of the proposed MI-LSFS includes a new method that uses our MI-LSFS algorithm to explore and investigate the features learned by a CNN model. We performed comprehensive experiments on CNN models trained on real data sets. The comparison of the quantitative classification accuracy results confirmed the better performance of our method. To investigate the qualitative measures and the explainability of our method, we also proposed a visualization method for CNN models, called Gradient-weighted Sample Activation Map (Grad-SAM), that uses the locally learned features of each sample to highlight their relevant and salient parts. The visualization experiments confirm both the explainability power of the selected features and the performance of Grad-SAM in visualizing the locally learned features of a CNN model.

The third proposed application is a novel explanation method, called Classifier Explanation by Local Feature Selection (CE-LFS), that can justify the decisions of a

trained model. The experimental results confirm the ability of CE-LFS in explaining correctly and incorrectly classified samples. The explanation generated by CE-LFS can be used as a feedback to improve the classification performance.

6.2 Potential Future Work

Although the proposed algorithms are fully developed and have shown promising results, there is still room for improvement. Future work can include an extension of our proposed applications, exploring the CNN models and CE-LFS algorithm, to the MIL data. Another potential work is to speed up the learning of MI-LSFS by adding additional assumptions to its optimization process and extending it to larger data sets.

Future research may also include formulating MI-LSFS as a deep metric learning problem by treating each sample as one anchor and generating the pair of anchor-positive and anchor-negative samples using the same class and other class labels. This can accelerate the learning process by using GPUs.

Moreover, we recommend the evaluation of our MI-LSFS algorithm on other domains and applications such as biomarker discovery for omics data [41,42]. In omics data, features are extracted from 3D spectrums where there is a chance of information loss in the feature extraction process. However, different extraction parameters can extract a set of different instances for each sample, and the data can be mapped in multiple instance form. Applying MI-LSFS on this mapped data to identify the biomarkers could be a potential future research topic.

CE-LFS introduces a new research topic for explaining the decisions of trained

classifiers. Based on the promising results of CE-LFS, we suggest applying it to other applications by considering alternative assumptions in identifying the salient local features.

REFERENCES

[1] Aurélien Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*, O'Reilly Media, 2019. [Cited on pages xii and 32.]

[2] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929. [Cited on pages xii, 33, 34, and 52.]

[3] Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra, "Grad-cam: Why did you say that?," *arXiv preprint arXiv:1611.07450*, 2016. [Cited on pages xii, 34, 35, and 52.]

[4] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu, "Feature selection: A data perspective," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, pp. 1–45, 2017. [Cited on pages 1, 8, and 17.]

[5] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga, "A review of feature selection techniques in bioinformatics," *bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007. [Cited on pages 1 and 8.]

[6] Vipin Kumar and Sonajharia Minz, "Feature selection: a literature review," *SmartCR*, vol. 4, no. 3, pp. 211–229, 2014. [Cited on page 1.]

[7] Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos, "A review of feature selection methods on synthetic data," *Knowledge and information systems*, vol. 34, no. 3, pp. 483–519, 2013. [Cited on page 1.]

[8] Yun Li, Tao Li, and Huan Liu, "Recent advances in feature selection and its applications," *Knowledge and Information Systems*, vol. 53, no. 3, pp. 551–577, 2017. [Cited on page 1.]

[9] Narges Armanfard, James P Reilly, and Majid Komeili, "Logistic localized modeling of the sample space for feature selection and classification," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 5, pp. 1396–1413, 2018. [Cited on pages 2, 3, 7, 9, 10, 11, 13, 14, 37, 41, and 47.]

[10] Jaume Amores, "Multiple instance classification: Review, taxonomy and comparative study," *Artificial intelligence*, vol. 201, pp. 81–105, 2013. [Cited on pages 2 and 18.]

[11] Min-Ling Zhang and Zhi-Hua Zhou, "Improve multi-instance neural networks through feature selection," *Neural processing letters*, vol. 19, no. 1, pp. 1–10, 2004. [Cited on pages 3 and 22.]

[12] Xun Yuan, Xian-Sheng Hua, Meng Wang, Guo-Jun Qi, and Xiu-Qing Wu, "A novel multiple instance learning approach for image retrieval based on adaboost feature selection," in *2007 IEEE International Conference on Multimedia and Expo*. IEEE, 2007, pp. 1491–1494. [Cited on pages 3 and 22.]

[13] Vikas C Raykar, Balaji Krishnapuram, Jinbo Bi, Murat Dundar, and R Bharat Rao, "Bayesian multiple instance learning: automatic feature selection and inductive transfer," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 808–815. [Cited on pages 3 and 23.]

[14] Hong Zhu, Li-Zhi Liao, and Michael K Ng, "Multi-instance dimensionality reduction via sparsity and orthogonality," *Neural computation*, vol. 30, no. 12, pp. 3281–3308, 2018. [Cited on pages 3, 25, and 71.]

[15] Jing Chai, Zehua Chen, Hongtao Chen, and Xinghao Ding, "Designing bag-level multiple-instance feature-weighting algorithms based on the large margin principle," *Information Sciences*, vol. 367, pp. 783–808, 2016. [Cited on pages 3 and 25.]

[16] Amelia Zafra, Mykola Pechenizkiy, and Sebastián Ventura, "Feature selection is the relieff for multiple instance learning," in *2010 10th International Conference on Intelligent Systems Design and Applications*. IEEE, 2010, pp. 525–532. [Cited on page 3.]

[17] Amelia Zafra, Mykola Pechenizkiy, and Sebastin Ventura, "Relieff-mi: An extension of relieff to multiple instance learning," *Neurocomputing*, vol. 75, no. 1, pp. 210–218, 2012. [Cited on pages 3, 24, and 71.]

[18] Amelia Zafra, Mykola Pechenizkiy, and Sebastián Ventura, "Hydr-mi: A hybrid algorithm to reduce dimensionality in multiple instance learning," *Information sciences*, vol. 222, pp. 282–301, 2013. [Cited on pages 3, 24, 71, and 72.]

[19] Jun Wang and Jean-Daniel Zucker, "Solving multiple-instance problem: A lazy learning approach," 2000. [Cited on pages 3, 18, 19, and 39.]

[20] Brian W Matthews, "Comparison of the predicted and observed secondary structure of t4 phage lysozyme," *Biochimica et Biophysica Acta (BBA)-Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975. [Cited on pages 4 and 43.]

[21] Yixin Chen, Jinbo Bi, and James Ze Wang, "Miles: Multiple-instance learning via embedded instance selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 1931–1947, 2006. [Cited on pages 4, 18, 19, 20, 22, 48, 49, and 70.]

[22] Aliasghar Shahrjooihaghighi and Hichem Frigui, "Local feature selection for multiple instance learning," *Journal of Intelligent Information Systems*, pp. 1–25, 2021. [Cited on page 6.]

[23] Soha Ahmed, Mengjie Zhang, and Lifeng Peng, "Improving feature ranking for biomarker discovery in proteomics mass spectrometry data using genetic programming," *Connection Science*, vol. 26, no. 3, pp. 215–243, 2014. [Cited on page 7.]

[24] Cosmin Lazar, Jonatan Taminau, Stijn Meganck, David Steenhoff, Alain Coletta, Colin Molter, Virginie de Schaetzen, Robin Duque, Hugues Bersini, and Ann Nowe, "A survey on filter techniques for feature selection in gene expression microarray analysis," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 9, no. 4, pp. 1106–1119, 2012. [Cited on page 8.]

[25] Yi Yang, Heng Tao Shen, Zhigang Ma, Zi Huang, and Xiaofang Zhou, "L 2, 1-norm regularized discriminative feature selection for unsupervised learning," in *IJCAI international joint conference on artificial intelligence*, 2011. [Cited on page 8.]

[26] Kenji Kira and Larry A Rendell, "A practical approach to feature selection," in *Machine Learning Proceedings 1992*, pp. 249–256. Elsevier, 1992. [Cited on page 8.]

[27] Alexander Shishkin, Anastasia Bezzubtseva, Alexey Drutsa, Ilia Shishkov, Ekaterina Gladkikh, Gleb Gusev, and Pavel Serdyukov, "Efficient high-order interaction-aware feature selection based on conditional mutual information," in *Advances in neural information processing systems*, 2016, pp. 4637–4645. [Cited on page 8.]

[28] Ron Kohavi, George H John, et al., "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997. [Cited on page 8.]

[29] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti A Zadeh, *Feature extraction: foundations and applications*, vol. 207, Springer, 2008. [Cited on page 8.]

[30] Hiromasa Arai, Crystal Maung, Ke Xu, and Haim Schweitzer, "Unsupervised feature selection by heuristic search with provable bounds on suboptimality," in *Proceedings of the Thirtieth AAAI conference on Artificial Intelligence*, 2016, pp. 666–672. [Cited on page 8.]

[31] Yvan Saeys, Thomas Abeel, and Yves Van de Peer, "Robust feature selection using ensemble feature selection techniques," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2008, pp. 313–325. [Cited on pages 8 and 9.]

[32] Jun Chin Ang, Andri Mirzal, Habibollah Haron, and Haza Nuzly Abdull Hamed, "Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 13, no. 5, pp. 971–989, 2015. [Cited on page 8.]

[33] Girish Chandrashekar and Ferat Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014. [Cited on page 8.]

[34] Jianyu Miao and Lingfeng Niu, "A survey on feature selection," *Procedia Computer Science*, vol. 91, pp. 919–926, 2016. [Cited on page 8.]

[35] Jie Cai, Jiawei Luo, Shulin Wang, and Sheng Yang, "Feature selection in machine learning: A new perspective," *Neurocomputing*, vol. 300, pp. 70–79, 2018. [Cited on page 8.]

[36] Pengyi Yang, Yee Hwa Yang, Bing B Zhou, and Albert Y Zomaya, "A review of ensemble methods in bioinformatics," *Current Bioinformatics*, vol. 5, no. 4, pp. 296–308, 2010. [Cited on page 9.]

[37] Myungjin Moon and Kenta Nakai, "Stable feature selection based on the ensemble l 1-norm support vector machine for biomarker discovery," *BMC genomics*, vol. 17, no. 13, pp. 1026, 2016. [Cited on page 9.]

[38] Qiang Shen, Ren Diao, and Pan Su, "Feature selection ensemble.," *Turing-100*, vol. 10, pp. 289–306, 2012. [Cited on page 9.]

[39] Zhi-Hua Zhou, *Ensemble methods: foundations and algorithms*, CRC press, 2012. [Cited on page 9.]

[40] Kang Leng Chiew, Choon Lin Tan, KokSheik Wong, Kelvin SC Yong, and Wei King Tiong, "A new hybrid ensemble feature selection framework for machine learning-based phishing detection system," *Information Sciences*, vol. 484, pp. 153–166, 2019. [Cited on page 9.]

[41] Aliasghar Shahrjooihaghighi, Hichem Frigui, Xiang Zhang, Xiaoli Wei, Biyun Shi, and Ameni Trabelsi, "An ensemble feature selection method for biomarker discovery," in *2017 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. IEEE, 2017, pp. 416–421. [Cited on pages 9 and 101.]

[42] AliAsghar ShahrjooiHaghighi, Hichem Frigui, Xiang Zhang, Xiaoli Wei, Biyun Shi, and Craig J McClain, "Ensemble feature selection for biomarker discovery in mass spectrometry-based metabolomics," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019, pp. 19–24. [Cited on pages 9 and 101.]

[43] Wael Awada, Taghi M Khoshgoftaar, David Dittman, Randall Wald, and Amri Napolitano, "A review of the stability of feature selection techniques for bioinformatics data," in *2012 IEEE 13th International Conference on Information Reuse & Integration (IRI)*. IEEE, 2012, pp. 356–363. [Cited on page 9.]

[44] Ryan J Urbanowicz, Melissa Meeker, William La Cava, Randal S Olson, and Jason H Moore, "Relief-based feature selection: Introduction and review," *Journal of biomedical informatics*, vol. 85, pp. 189–203, 2018. [Cited on page 10.]

[45] Yijun Sun, Sinisa Todorovic, and Steve Goodison, "Local-learning-based feature selection for high-dimensional data analysis," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1610–1626, 2009. [Cited on page 10.]

[46] Narges Armanfard, James P Reilly, and Majid Komeili, "Local feature selection for data classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 6, pp. 1217–1227, 2015. [Cited on page 10.]

[47] Stephen Boyd and Lieven Vandenberghe, *Convex optimization*, Cambridge university press, 2004. [Cited on pages 12, 13, and 41.]

[48] Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles," *Artificial intelligence*, vol. 89, no. 1-2, pp. 31–71, 1997. [Cited on pages 15 and 72.]

[49] Oded Maron and Tomás Lozano-Pérez, "A framework for multiple-instance learning," in *Advances in neural information processing systems*, 1998, pp. 570–576. [Cited on pages 16, 17, and 22.]

[50] Xiaojun Qi and Yutao Han, "Incorporating multiple svms for automatic image annotation," *Pattern Recognition*, vol. 40, no. 2, pp. 728–741, 2007. [Cited on page 16.]

[51] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann, "Support vector machines for multiple-instance learning," in *Advances in neural information processing systems*, 2003, pp. 577–584. [Cited on pages 16, 17, and 72.]

[52] Andrew Karem, Mohamed Trabelsi, Mahdi Moalla, and Hichem Frigui, "Comparison of several single and multiple instance learning methods for detecting buried explosive objects using gpr data," in *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXIII*. International Society for Optics and Photonics, 2018, vol. 10628, p. 106280G. [Cited on page 16.]

[53] Faezeh Tafazzoli and Hichem Frigui, "Vehicle make and model recognition using local features and logo detection," in *2016 International Symposium on Signal, Image, Video and Communications (ISIVC)*. IEEE, 2016, pp. 353–358. [Cited on page 16.]

[54] W. Safta, M. M. Farhangi, B. Veasey, A. Amini, and H. Frigui, "Multiple instance learning for malignant vs. benign classification of lung nodules in thoracic screening ct data," in *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, 2019, pp. 1220–1224. [Cited on page 16.]

[55] Qi Zhang and Sally A Goldman, "Em-dd: An improved multiple-instance learning technique," in *Advances in neural information processing systems*, 2002, pp. 1073–1080. [Cited on page 17.]

[56] Zhi-Hua Zhou and Min-Ling Zhang, "Solving multi-instance problems with classifier ensemble based on constructive clustering," *Knowledge and information systems*, vol. 11, no. 2, pp. 155–170, 2007. [Cited on page 17.]

[57] Marc-André Carbonneau, Veronika Cheplygina, Eric Granger, and Ghyslain Gagnon, "Multiple instance learning: A survey of problem characteristics and applications," *Pattern Recognition*, vol. 77, pp. 329–353, 2018. [Cited on page 17.]

[58] MA Carbonneau, E Granger, and G Gagnon, "Decision threshold adjustment strategies for increased accuracy in multiple instance learning," in *Proceedings of the international conference on image processing theory, tools and application*, 2016. [Cited on page 17.]

[59] Soumya Ray and Mark Craven, "Supervised versus multiple instance learning: An empirical comparison," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 697–704. [Cited on pages 17 and 18.]

[60] Marc-André Carbonneau, Eric Granger, Alexandre J Raymond, and Ghyslain Gagnon, "Robust multiple-instance learning ensembles using random subspace instance selection," *Pattern recognition*, vol. 58, pp. 83–99, 2016. [Cited on page 17.]

[61] Boris Babenko, Piotr Dollár, Zhuowen Tu, and Serge Belongie, "Simultaneous learning and alignment: Multi-instance and multi-pose learning," in *Workshop on Faces in'Real-Life'Images: Detection, Alignment, and Recognition*, 2008. [Cited on page 17.]

[62] Zhi-Hua Zhou, Yu-Yin Sun, and Yu-Feng Li, "Multi-instance learning by treating instances as non-iid samples," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 1249–1256. [Cited on page 17.]

[63] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas, "The earth mover's distance as a metric for image retrieval," *International journal of computer vision*, vol. 40, no. 2, pp. 99–121, 2000. [Cited on page 18.]

[64] Boris Babenko, "Multiple instance learning: algorithms and applications," *View Article PubMed/NCBI Google Scholar*, pp. 1–19, 2008. [Cited on page 18.]

[65] Gitte Vanwinckelen, Daan Fierens, Hendrik Blockeel, et al., "Instance-level accuracy versus bag-level accuracy in multi-instance learning," *Data mining and knowledge discovery*, vol. 30, no. 2, pp. 313–341, 2016. [Cited on page 18.]

[66] Faezeh Tafazzoli, "Vehicle make and model recognition for intelligent transportation monitoring and surveillance.," 2017. [Cited on page 19.]

[67] Ji Zhu, Saharon Rosset, Robert Tibshirani, and Trevor J Hastie, "1-norm support vector machines," in *Advances in neural information processing systems*, 2004, pp. 49–56. [Cited on pages 21 and 51.]

[68] Zhi-Hua Zhou and Min-Ling Zhang, "Neural networks for multi-instance learning," in *Proceedings of the International Conference on Intelligent Information Technology, Beijing, China*, 2002, pp. 455–459. [Cited on page 22.]

[69] Ian T Jolliffe, "Principal components in regression analysis," in *Principal component analysis*, pp. 129–155. Springer, 1986. [Cited on page 22.]

[70] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al., "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000. [Cited on page 22.]

[71] Stuart Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982. [Cited on page 22.]

[72] Igor Kononenko, "Estimating attributes: analysis and extensions of relief," in *European conference on machine learning*. Springer, 1994, pp. 171–182. [Cited on page 24.]

[73] Jing Chai, Hongtao Chen, Lixia Huang, and Fanhua Shang, "Maximum margin multiple-instance feature weighting," *Pattern recognition*, vol. 47, no. 6, pp. 2091–2103, 2014. [Cited on pages 24 and 71.]

[74] Kilian Q Weinberger and Lawrence K Saul, "Distance metric learning for large margin nearest neighbor classification.," *Journal of Machine Learning Research*, vol. 10, no. 2, 2009. [Cited on page 24.]

[75] Bo Chen, Hongwei Liu, Jing Chai, and Zheng Bao, "Large margin feature weighting method via linear programming," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 10, pp. 1475–1488, 2008. [Cited on page 24.]

[76] Yu-Yin Sun, Michael K Ng, and Zhi-Hua Zhou, "Multi-instance dimensionality reduction.," in *AAAI*. Citeseer, 2010. [Cited on page 25.]

[77] Saehoon Kim and Seungjin Choi, "Local dimensionality reduction for multiple instance learning," in *2010 IEEE International Workshop on Machine Learning for Signal Processing*. IEEE, 2010, pp. 13–18. [Cited on page 25.]

[78] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015. [Cited on pages 25, 27, and 28.]

[79] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, MIT press, 2016. [Cited on page 26.]

[80] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012. [Cited on pages 26, 28, and 29.]

[81] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. [Cited on pages 26, 30, and 83.]

[82] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. [Cited on pages 26, 28, and 30.]

[83] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9. [Cited on pages 26, 28, and 29.]

[84] François Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258. [Cited on pages 26, 28, and 31.]

[85] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012. [Cited on page 27.]

[86] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. Ieee, 2013, pp. 6645–6649. [Cited on page 27.]

[87] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al., "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International conference on machine learning*. PMLR, 2016, pp. 173–182. [Cited on page 27.]

[88] Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al., "State-of-the-art speech recognition with sequence-to-sequence models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4774–4778. [Cited on page 27.]

[89] Erik Gawehn, Jan A Hiss, and Gisbert Schneider, "Deep learning in drug discovery," *Molecular informatics*, vol. 35, no. 1, pp. 3–14, 2016. [Cited on page 27.]

[90] Antonio Lavecchia, "Deep learning in drug discovery: opportunities, challenges and future prospects," *Drug discovery today*, vol. 24, no. 10, pp. 2017–2032, 2019. [Cited on page 27.]

[91] Yu Li, Chao Huang, Lizhong Ding, Zhongxiao Li, Yijie Pan, and Xin Gao, "Deep learning in bioinformatics: Introduction, application, and perspective in the big data era," *Methods*, vol. 166, pp. 4–21, 2019. [Cited on page 27.]

[92] Yongqing Zhang, Jianrong Yan, Siyu Chen, Meiqin Gong, Dongrui Gao, Min Zhu, and Wei Gan, "Review of the applications of deep learning in bioinformatics," *Current Bioinformatics*, vol. 15, no. 8, pp. 898–911, 2020. [Not cited.]

[93] Dinggang Shen, Guorong Wu, and Heung-Il Suk, "Deep learning in medical image analysis," *Annual review of biomedical engineering*, vol. 19, pp. 221–248, 2017. [Cited on page 27.]

[94] Amitojdeep Singh, Sourya Sengupta, and Vasudevan Lakshminarayanan, "Explainable deep learning models in medical image analysis," *Journal of Imaging*, vol. 6, no. 6, pp. 52, 2020. [Cited on page 27.]

[95] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio, "On using very large target vocabulary for neural machine translation," *arXiv preprint arXiv:1412.2007*, 2014. [Cited on page 27.]

[96] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al., "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, pp. 9, 2019. [Cited on page 27.]

[97] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al., "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020. [Cited on page 27.]

[98] David H Hubel and Torsten N Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, pp. 106–154, 1962. [Cited on page 27.]

[99] Kunihiko Fukushima and Sei Miyake, "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern recognition*, vol. 15, no. 6, pp. 455–469, 1982. [Cited on page 27.]

[100] David H Hubel and Torsten N Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of physiology*, vol. 148, no. 3, pp. 574–591, 1959. [Cited on page 27.]

[101] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. [Cited on pages 28 and 29.]

[102] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al., "Deep high-resolution representation learning for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, 2020. [Cited on pages 28 and 30.]

[103] Bowen Cheng, Bin Xiao, Jingdong Wang, Honghui Shi, Thomas S Huang, and Lei Zhang, "Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5386–5395. [Cited on pages 28 and 30.]

[104] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826. [Cited on pages 28 and 31.]

[105] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-first AAAI conference on artificial intelligence*, 2017. [Cited on pages 28 and 31.]

[106] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015. [Cited on pages 28 and 31.]

[107] Songtao Wu, Shenghua Zhong, and Yan Liu, "Deep residual learning for image steganalysis," *Multimedia tools and applications*, vol. 77, no. 9, pp. 10437–10453, 2018. [Cited on pages 28 and 31.]

[108] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708. [Cited on pages 28 and 31.]

[109] Sergey Zagoruyko and Nikos Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016. [Cited on pages 28 and 31.]

[110] Dongyoon Han, Jiwhan Kim, and Junmo Kim, "Deep pyramidal residual networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5927–5935. [Cited on pages 28 and 31.]

[111] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton, "Dynamic routing between capsules," *arXiv preprint arXiv:1710.09829*, 2017. [Cited on pages 28 and 31.]

[112] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500. [Cited on pages 28 and 31.]

[113] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, no. 1, pp. 1–74, 2021. [Cited on pages 29, 30, and 31.]

[114] Min Lin, Qiang Chen, and Shuicheng Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013. [Cited on page 30.]

[115] Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma, "Provable bounds for learning some deep representations," in *International conference on machine learning*. PMLR, 2014, pp. 584–592. [Cited on page 30.]

[116] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016. [Cited on page 32.]

[117] Sinno Jialin Pan and Qiang Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009. [Cited on page 32.]

[118] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al., "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information Fusion*, vol. 58, pp. 82–115, 2020. [Cited on page 33.]

[119] Gabriëlle Ras, Marcel van Gerven, and Pim Haselager, "Explanation methods in deep learning: Users, values, concerns and challenges," in *Explainable and interpretable models in computer vision and machine learning*, pp. 19–36. Springer, 2018. [Cited on page 33.]

[120] Ning Xie, Gabrielle Ras, Marcel van Gerven, and Derek Doran, "Explainable deep learning: A field guide for the uninitiated," *arXiv preprint arXiv:2004.14545*, 2020. [Cited on pages 33 and 53.]

[121] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626. [Cited on pages 35, 36, 48, 52, and 53.]

[122] Jerome Friedman, Trevor Hastie, and Robert Tibshirani, *The elements of statistical learning*, vol. 1, Springer series in statistics New York, 2001. [Cited on page 43.]

[123] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi, "A survey of methods for explaining black box

models," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018. [Cited on page 48.]

[124] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al., "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018. [Cited on page 48.]

[125] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba, "Understanding the role of individual units in a deep neural network," *Proceedings of the National Academy of Sciences*, 2020. [Cited on page 48.]

[126] Vanessa Buhrmester, David Münch, and Michael Arens, "Analysis of explainers of black box deep neural networks for computer vision: A survey," *arXiv preprint arXiv:1911.12116*, 2019. [Cited on page 52.]

[127] Matthew D Zeiler, Graham W Taylor, and Rob Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 2018–2025. [Cited on page 52.]

[128] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje, "Learning important features through propagating activation differences," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3145–3153. [Cited on page 52.]

[129] Le Hou, Dimitris Samaras, Tahsin M Kurc, Yi Gao, James E Davis, and Joel H Saltz, "Efficient multiple instance convolutional neural networks for gigapixel resolution image classification," *arXiv preprint arXiv:1504.07947*, p. 7, 2015. [Cited on page 53.]

[130] Miao Sun, Tony X Han, Ming-Chang Liu, and Ahmad Khodayari-Rostamabad, "Multiple instance learning convolutional neural networks for object recognition," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 3270–3275. [Cited on page 53.]

[131] Maximilian Ilse, Jakub Tomczak, and Max Welling, "Attention-based deep multiple instance learning," in *International conference on machine learning*. PMLR, 2018, pp. 2127–2136. [Cited on pages 53 and 71.]

[132] Heather D Couture, James Stephen Marron, Charles M Perou, Melissa A Troester, and Marc Niethammer, "Multiple instance learning for heterogeneous images: Training a cnn for histopathology," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2018, pp. 254–262. [Cited on page 53.]

[133] Kausik Das, Sailesh Conjeti, Abhijit Guha Roy, Jyotirmoy Chatterjee, and Debdoot Sheet, "Multiple instance learning of deep convolutional neural networks for breast histopathology whole slide classification," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. IEEE, 2018, pp. 578–581. [Cited on page 53.]

[134] Heinrich Jiang, Been Kim, Melody Y Guan, and Maya Gupta, "To trust or not to trust a classifier," *arXiv preprint arXiv:1805.11783*, 2018. [Cited on page 57.]

[135] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144. [Cited on pages 57 and 59.]

[136] Scott M Lundberg and Su-In Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st international conference on neural information processing systems*, 2017, pp. 4768–4777. [Cited on page 57.]

[137] Pang Wei Koh and Percy Liang, "Understanding black-box predictions via influence functions," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1885–1894. [Cited on page 57.]

[138] Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee, "From local explanations to global understanding with explainable ai for trees," *Nature machine intelligence*, vol. 2, no. 1, pp. 56–67, 2020. [Cited on page 57.]

[139] Yann LeCun, Corinna Cortes, and Christopher JC Burges, "The mnist database of handwritten digits, 1998," *URL http://yann. lecun. com/exdb/mnist*, vol. 10, no. 34, pp. 14, 1998. [Cited on pages 62 and 66.]

[140] William M Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971. [Cited on page 65.]

[141] Lawrence Hubert and Phipps Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985. [Cited on page 65.]

[142] Faezeh Tafazzoli, Hichem Frigui, and Keishin Nishiyama, "A large and diverse dataset for improved vehicle make and model recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 1–8. [Cited on pages 81 and 82.]

[143] Jeremy Elson, John R Douceur, Jon Howell, and Jared Saul, "Asirra: a captcha that exploits interest-aligned manual image categorization.," *CCS*, vol. 7, pp. 366–374, 2007. [Cited on pages 81 and 82.]

[144] Philippe Golle, "Machine learning attacks against the asirra captcha," in *Proceedings of the 15th ACM conference on Computer and communications security*, 2008, pp. 535–542. [Cited on page 82.]

CURRICULUM VITAE

NAME: Aliasghar Shahrjooihaghighi

ADDRESS: Department of Computer Science and Engineering

Speed School of Engineering

University of Louisville

Louisville, KY 40292

EDUCATION:

Ph.D., Computer Science & Engineering , December 2021

University of Louisville, *Louisville, Kentucky*

M.Sc., Computer Engineering, September 2011

Islamic Azad University, *Ahvaz, Iran*

B.Sc., Computer Engineering, July 2006

Shahid Chamran University of Ahvaz, *Ahvaz, Iran*

JOURNAL PUBLICATIONS:

1. **A. Shahrjooihaghighi** and H. Frigui, *"Local Feature Selection for Multiple Instance Learning"*, Journal of Intelligent Information Systems, 2021.

2. **A. ShahrjooiHaghighi**, L. He, X. Ma, H. Frigui, X. Zhang, X. Wei, B. Shi, C. McClain, *"Limited Separation and Matrix Effect Induce a High Rate of False Biomarker Discovery in LC-MS based Metabolomics and Proteomics"*, 2022 (under rev).

SELECTED CONFERENCE PUBLICATIONS:

1. **A. ShahrjooiHaghighi**, H. Frigui, X. Zhang, B. Shi, A, Bouzid, C. McClain, *"Ensemble Feature Selection for Biomarker Discovery in Mass Spectrometry-based Metabolomics"*, , SAC, Limassol, Cyprus, 2019.

2. **A. ShahrjooiHaghighi**, H. Frigui, X. Zhang, X. Wei, B. Shi, A. Trabelsi, *"An Ensemble Feature Selection Method for Biomarker Discovery"*, ISSPIT, Bilbao, Spain, 2017.

3. A. Trabelsi, B. Shi, X. Wei, H. Frigui, X., C. McClain, **A. Shahrjooihaghighi**. *"Molecule specific normalization for protein and metabolite biomarker discovery"*, ISSPIT, Bilbao, Spain, 2017.

4. **A. ShahrjooiHaghighi**, M. Abbasi Dezfuli, S.M. Fakhrahmad, *"Applying Mining Schemes to Software Fault Prediction: A Proposed Approach Aimed at Test Cost Reduction"*, World Congress on Engineering 2012 (WCE), London, UK, 2012.

HONORS AND AWARDS:

1. Scholarship to CMD-IT/ACM Richard Tapia Celebration of Diversity in Computing Conference, 2021.
2. Graduate Travel Fund, University of Louisville, 2018, 2021.
3. Arthur M. Riehl Award, University of Louisville, 2020.
4. Nominated as a Faculty Favorite, University of Louisville, 2019-2020.
5. Certificate of Appreciation, ICML, 2020.
6. IBM Tapia Scholarship, Tapia Conference, 2020.
7. Graduate Research Fund, University of Louisville, 2018.
8. Speed Up Entrepreneurial Program Award, University of Louisville, 2017.
9. Student Scholarship Award, Midwest SAS Users Group, 2017.
10. Graduate Scholarship Award, University of Louisville, 2015-2021.