Large-Margin Representation Learning for Small-Size Datasets

by

Jonathan DE MATOS

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, JULY 11, 2023

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Alessandro Lameiras Koerich, Thesis supervisor
Department of Software and Information Technology Engineering, École de technologie supérieure

Mr. Alceu de Souza Britto Junior, Thesis Co-Supervisor
Postgraduate Program in Informatics, Pontifical Catholic University of Paraná

Mr. Robert Sabourin, Chair, Board of Examiners
Department of Systems Engineering, École de technologie supérieure

Mr. Matthew Toews, Member of the Jury
Department of Systems Engineering, École de technologie supérieure

Mr. Vincent Andrearczyk, External Independent Examiner
HES-SO University of Applied Sciences and Arts Western Switzerland

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON JUNE 2, 2023

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGEMENTS

I want to thank my supervisor, Professor Alessandro Lameiras Koerich, and my co-supervisor, Professor Alceu de Souza Britto Junior, for supporting and guiding me during my studies and the development of this work. I also want to thank ETS and Livia for the structure that facilitated my work. The Natural Sciences and Engineering Research Council of Canada (NSERC) played an essential role in this work, providing me with financial support, likewise the State University of Ponta Grossa. The support from my colleagues from UEPG was also noteworthy. My friends from ETS also taught me a lot during this period. I special thank my wife and daughter, which were by my side during my studies. I want to dedicate my work to my mother and father (in memoriam), which made all of this possible.

Apprentissage de Représentation à Large-Marge pour Ensembles de Données de Petite Taille

Jonathan DE MATOS

RÉSUMÉ

Ce travail présente une nouvelle approche combinant des couches convolutives et un apprentissage métrique à large marge pour l'apprentissage de modèles supervisés sur des ensembles de données de petite taille. Cette approche comporte quatre composantes principales : i) un ensemble de couches convolutives avec une couche de mis-en-commun par moyenne globale; ii) une méthode de sélection d'instances pour choisir des ancres et les instances d'intérêt; iii) une fonction de perte pour induire la mise à jour du poids des couches convolutives; iv) un discriminant à large marge. Les images sont fournies aux couches convolutives pour générer une représentation latente, puis utilisées pour entraîner un discriminant à large marge. Les informations du discriminant (vecteurs de support et prédictions) aident à sélectionner des ancres et des instances d'intérêt pour construire une fonction de perte qui vise à minimiser la distance entre les échantillons choisis. La mise à jour du poids à l'aide de la fonction de perte cherche à modifier la représentation latente pour augmenter la marge entre les classes. L'avantage de la méthode proposée est qu'elle peut entraîner une banque de filtres avec une petite quantité de données en raison du nombre réduit de paramètres. Il amène aussi à des coûts d'entraînement réduit puisque seul un sous-ensemble d'instances est utilisé dans l'algorithme de rétropropagation. Les résultats expérimentaux avec un ensemble de données synthétiques, un ensemble de données d'images de texture et trois ensembles de données d'images histopathologiques (avec des caractéristiques de texture) montrent que la méthode proposée converge rapidement, en quelques époques. De plus, les autres avantages sont le faible coût de calcul, des précisions proches ou supérieures à ceux des méthodes équivalentes, un traitement satisfaisant des données déséquilibrées et l'adaptation des filtres à différents types de textures.

Mots-clés: Discriminant à large-marge, Couches convolutives, Points d'ancrage, Fonction de perte, Images de texture.

Large-Margin Representation Learning for Small-Size Datasets

Jonathan DE MATOS

ABSTRACT

This work presents a novel approach combining convolutional layers (CLs) and large-margin metric learning for training supervised models on small-size datasets. This approach has four main components: i) a set of CLs with a global average pooling; ii) an instance selection method to choose anchors and instances of interest; iii) a loss function to induce the weight update of the CLs; iv) a large-margin discriminant. Images are provided to the CLs to generate a latent representation and then used to train a large-margin discriminant. The information of the discriminant (support vectors and predictions) aids in selecting anchors and instances of interest to build a loss function that aims to minimize the distance between chosen samples. The weight update using the loss function seeks to change the latent representation to increase the margin between classes. The proposed method's advantage is that it can train a filter bank with a small amount of data due to the reduced number of parameters. It also has reduced training costs since only a subset of instances is used in the backpropagation algorithm. Experimental results with a synthetic dataset, a texture image dataset, and three histopathologic image datasets (with textural characteristics) showed that the proposed method converges within a few epochs. It also has a low computational cost, produces close or superior accuracy results than equivalent methods, deals well with imbalanced data, and can adapt the filters to different textures.

Keywords: Large-margin discriminant, Fully convolutional network, Anchor points, Loss function, Texture images.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

BCE	Binary Cross Entropy
CL	Convolutional Layer
CNN	Convolutional Neural Network
DML	Deep Metric Learning
FCN	Fully Convolutional Network
FC	Fully Connected
FV-CNN	Fischer Vector Convolutional Neural Network
GAP	Global Average Pooling
GLCM	Grey-level Cooccurrence Matrix
H&E	Hematoxylin and Eosin
HI	Histopathologic Image
LMFCN	Large-margin Fully Convolutional Network
ML	Metric Learning
OVA	One-vs-All
PCA	Principal Component Analysis
PFTAS	Parameter-free Threshold Adjacency Statistics
RBF	Radial-Basis Function
SV	Support Vectors
SVM	Support Vectors Machine

TCNN	Texture Convolutional Neural Network
UMAP	Uniform Manifold Approximation and Projection
WSI	Whole Slide Image

LIST OF SYMBOLS AND UNITS OF MEASUREMENTS

$\mathcal{X}$	Set of input instances $\{X^t, o^t\}$
$\mathbf{X}^t$	t-th image of $\mathcal{X}$
o^t	Label or class of t-th image of $\mathcal{X}$
n	Size of the set $\mathcal{X}$ or number of images
$\mathbf{T}$	Matrix of images' latent representations
ϕ	Size of the latent representation (number of attributes)
$\mathbf{P}$	Auxiliary matrix used to calculate $\mathbf{K}$ and $\mathbf{D}$
$\mathbf{K}$	Kernel matrix used in the large-margin classifier
$\mathbf{D}$	Distance matrix to help anchors selection
y	Predicted output of an image by the SVM classifier
$\mathcal{S}$	Set with the indices instances of interest related to anchors $\mathcal{A}_{sv}$ (Support Vectors)
$\mathcal{Q}$	Set with the indices instances of interest related to anchors $\mathcal{A}_{mc}$ (Misclassified instances)
$\mathcal{R}$	Set with the indices instances of interest related to anchors $\mathcal{A}_{cc}$ (Correct classified instances)
$\mathbf{A}$	Matrix with the indices of anchors $\mathcal{A}_{sv}$ for each instance of interest
$\mathbf{M}$	Matrix with the indices of anchors $\mathcal{A}_{mc}$ for each instance of interest
$\mathbf{G}$	Matrix with the indices of anchors $\mathcal{A}_{cc}$ for each instance of interest
$\mathcal{L}_{\text{sv}}$	Loss function related to SVs

$\mathcal{L}_{\text{mc}}$	Loss function related to misclassified instances
$\mathcal{L}_{\text{cc}}$	Loss function related to correct classified instances
sv_{close}	Number of closest anchors $\mathcal{A}_{sv}$ for each instance of interest
mc_{close}	Number of closest anchors $\mathcal{A}_{mc}$ for each instance of interest
sh_{close}	Number of closest anchors $\mathcal{A}_{cc}$ for each instance of interest
w	Width of an image
h	Height of an image
c	Number of channels of an image
n_{classes}	Number of classes
α_{fcn}	Number of weights of the FCN
α_{fc}	Number of weights of the fully-connected layers
n_{sv}	Number of support vectors
γ	Hyperparameter of RBF Kernel
C	Hyperparameter of SVM lenience with misclassified samples
$f_{\text{fcn}}(.)$	Fully Convolutional Network representation function
$f_{\text{sv}}(.)$	SVM classifier representation function
TR	Training set
VAL	Validation set
TS	Test set
nSVs	Number of Support Vectors

INTRODUCTION

Image recognition is a widespread problem in computing. It has been the focus of research for a long time and has faced a revolution with the emergence of convolutional neural networks (CNNs). CNNs have achieved state-of-the-art performance in many problems and defined a new level of recognition compared to traditional feature extraction methods and classifiers. The reason for this success is that the filters or feature extraction part is trained together with the classifier or FC layers. With more filters, there is also a high number of parameters (weights) that need more data for successful training. CNNs became more popular and achieved the best results initially in object recognition problems, e.g., cars, people, animals, and numbers. The reason is that this type of image or dataset is easier to acquire and label, not requiring specialists for the task, so huge amounts of data are available like the ImageNet project (Russakovsky *et al.*, 2015). From the object recognition problems emerged several CNN architectures, like ResNet (He, Zhang, Ren & Sun, 2016) and Inception (Szegedy, Vanhoucke, Ioffe, Shlens & Wojna, 2016), that are million-parameter networks, possible to train thanks to large data availability.

The first layers of CNNs are partially responsible for filtering patterns or textures on images, and more profound layers learn shape and spatial information. Deeper layers also store more domain-specific information. Based on these facts, researchers started using pre-trained CNNs to transfer knowledge between domains, known as transfer learning. Transfer learning can utilize the first and more generic pre-trained layers and fine-tune the deeper ones, adapting them to the new domain. However, there are some issues associated with this technique: i) images from a new domain have to be rescaled to the exact size of the original images used to train the network in some models, although new architectures already implemented a global average pooling (GAP) at their end, avoiding rescaling; ii) the outputs or classes of domains are usually different in size and context; iii) the more profound layers have to suffer more changes to learn the new domain (Andrearczyk & Whelan, 2016). When the new domain has sufficient data to fine-tune a CNN, these issues do not cause too much impact, and training is fast.

Transfer learning can be applied in contexts with few data or simple images without spatial and shape characteristics. Training deeper layers in this context is more straightforward than training a CNN from scratch but can still be difficult if too many parameters exist. Another approach similar to transfer learning is using pre-trained CNNs as feature extractors. In this technique, deeper layers are discarded, and images are fed to the CNN. Without the deeper layers, the output is a wide image representation vector. These representations are used to train a different classifier, like an SVM, a random forest, or a k-NN. However, this method also has shortcomings, such as the first layers may not generate good activations with the different textures, and the representation is not necessarily suitable for the chosen classifier. CNNs as feature extractors have similar behavior to handcrafted feature extractors. Both methods generate an immutable representation and cannot adapt to new textures nor adjust their output to target a specific need of classifiers.

There is an alternative to the CNNs as feature extractors that allow tuning its weights, so the representation is no more immutable. It is called Deep Metric Learning (DML), but it is usually applied in a different context. DML is a technique that uses deep learning methods to generate image representations and metrics to calculate similarity measures and obtain some categorization (Chopra, Hadsell & LeCun, 2005). The advantage of DML is that it allows learning a metric for the dissimilarity calculation and training the network that generates the representation. The context where DML is applied is on datasets with few samples per class, but a large number of categories, because there are not enough samples to train a classifier.

Small-size datasets with textural images are a complex problem for CNNs due to few training data, and the CNNs also have too many parameters in the face of image complexity. This type of problem has some characteristics related to DML, handcrafted feature extractors, CNNs, and monolithic classifiers. We proposed an approach that uses characteristics from some of these methods to deal with these datasets. Histopathological images (HIs) are one type of small-size

dataset that does not have macro object characteristics. These images originate, for example, from human tissue samples and work on the cellular level, allowing observe nuclei and support structures. At this level, depending on the resolution or magnifying factor, shape, and spatial information are absent, or there are only small shapes having more textural information. The number of images is usually small because acquiring them is expensive and laborious, and the labeling demands experienced pathologists. Despite a few labeled images, these datasets may have vast amounts of data due to the size of some whole slide images. Thus, we used this type of dataset to test our approach.

Problem Statement

Using large models is not the most suitable approach for problems with textural images and small-size datasets. Alternatively, more straightforward methods can obtain a better cost-benefit ratio relating to accuracy performance and computational cost. The model needs to adapt to different characteristics of texture and be able to train with small-size datasets. A limited amount of training data also poses a challenge when dealing with imbalanced datasets.

Research Question

Given the stated problem, two research questions are addressed in this thesis: i) May a strategy to learn a latent representation (similar to a feature vector) and a large margin-based discriminant from a small-size texture dataset provide competitive results compared to CNN-based approaches? ii) Could a large-margin discriminant replace the dense layers of a regular CNN and speed up the training convergence while maintaining accuracy?

Objectives

The main objective is to develop a new supervised learning method that generates a latent representation of the problem and uses it in a discriminant to classify small-size texture datasets. In the proposed end-to-end learning method, a large margin-based discriminant replaces the CNN dense layers, speeding up the training process while achieving high accuracy. We can also cite the following secondary objectives:

- Develop a CNN-like approach to create filter banks that automatically learn a discriminative latent representation of images.
- Adapt the deeper CNN layers to consider a large margin strategy in the discriminant part.
- Develop an algorithm to optimize the model weights (convolutional layers and discriminant) based on errors computed by a large margin-based discriminant.
- Evaluate the proposed end-to-end supervised learning method regarding computation time and accuracy compared to conventional CNNs.
- Evaluate the proposed method in the context of small-size datasets, imbalanced and with multiple classes.

Contributions

We developed an approach to train a set of convolutional layers (CLs) from scratch with little amount of data. It is a representation learning method that can adapt the CLs' filtering capacity to different texture datasets. The training is computationally efficient because it uses specially selected instances to guide the weight update. These instances are the support vectors (SVs) discovered during the optimization of the SVM classifier used in the method. The training acts on complex regions of the latent representation, where opposite class instances are close, so it tends to induce faster convergence than traditional CNNs training methods.

Although using fewer instances to train the CL weights than CNN training methods, they are more effective, positioned in the decision frontier, independent of the class. If a class has a higher number of instances, similar instances inside this class are not used in training, balancing the weights or the number of instances used per class. Thus, an additional contribution is a resilience to the class imbalance because clustered instances of a prevalent class are not used, only the ones in the margins. The effectiveness of the proposed approach was evaluated on several datasets, and it outperformed other methods, i.e., equivalent CNNs and shallow approaches employing handcrafted feature extractors and CNNs as feature extractors with monolithic classifiers.

Thesis Structure

This document is organized as follows: Chapter 1 presents a literature review of the significant concepts that involve the LMFCN development, like the CNNs and their relation to simple textural images, in Section 1.1. Section 1.2 presents a brief review of techniques related to Metric Learning and Deep Metric Learning. This chapter closes with a review, in Section 1.3, about HIs, from which emerge problems of image classification with small datasets of images with textural characteristics.

Chapter 2 describes the LMFCN with all its details and parts. Section 2.2 describes the concepts of anchors, the training algorithm, and the multiclass training procedure. We analyze the computational cost of our approach in Section 2.3, comparing it with the other equivalent approaches like CNNs and handcrafted feature extractors. Section 2.4 presents simulations of the anchors' effects in various latent representations that aim to clarify the importance of each type of anchor.

Chapter 3 describes the experiments and the results obtained at each comparison. Section 3.2 presents the datasets used in the comparisons with examples of images and how they were

prepared and split. We used a set of metrics described in Section 3.3 to evaluate the results and compare methods.

The last chapter presents our conclusions and future works (Chapter 4).

CHAPTER 1

LITERATURE REVIEW

This chapter will present some concepts we used in our approach. Section 1.1 describes some CNN concepts and their use on textural images and histopathologic images (HIs). We also present methods based on small networks designed to work with textural images that use alternative layers to work with this type of image. The concept of Deep Metric Learning (DML) is presented in Section 1.2, showing the difference between the usual context of DML and how we use it in our approach. Finally, Section 1.3 describes the HIs used to diagnose some types of tumor-related diseases. We show their importance, examples, and how the machine learning methods are applied to analyze them.

1.1 CNNs and Textures

Deep learning is used in various fields, e.g., speech, sound and visual object recognition, object detection, and drug discovery. Deep learning models are a set of linked layers trained by the backpropagation algorithm to abstract complex structures of large-scale data (LeCun, Bengio & Hinton, 2015). CNNs are a sequence of the connected convolutional layer (CLs), and the output of each one is also called a feature map. Pooling layers that may sit between CLs reduce the feature map size using some functions, e.g., maximum or average. Yet, activation functions like ReLU, Tanh, or Sigmoid exist between layers to bring non-linearity to the model. Without them, CNNs would be just a sequence of linear functions without depth. At the end of a CNN, some models use the fully connected (FC) layers that work as classifiers.

The idea behind the CNNs is that the weights on CLs work as filter banks that can extract motifs from images generating a feature map (LeCun *et al.*, 2015). The first CLs at the bottom of the network act as edge detectors, highlighting motifs spread around the image. The composition of detected motifs allows identifying complex objects.

The CLs weights need the training to extract the motifs that help to identify complex structures. The motifs are common in various images, so once trained, the first filters' weights are useful in other contexts to identify common patterns. The last layers, or the upper ones, are more specialized and task-specific, thus needing to be retrained or replaced to use on other domains. The upper layers are also capable of identifying the shape and position of objects or patterns.

In the context of texture images, using the first layers of a pre-trained CNN help with faster convergence, and the upper layers do not need to be very deep and complex. Hence, part of the network can be simplified and use fewer parameters. Pre-trained models and simplified upper layers contribute to achieving better results with few amount of data.

Humeau-Heurtier (2019) presented a review on texture feature extraction, proposing a taxonomy on the topic. The taxonomy classifies the CNN texture-related approaches as deep learning methods under the category of Learning-Based Approaches. As advantages of the CNN-based ones, they cite the capacity to learn high-level features more suitable to a specific task. Nonetheless, the training process requires more computational effort than LBP. Liu, Fieguth, Wang, Pietikäinen & Hu (2016) compared deep learning methods against LBP variants, concluding that the first ones present the best results but with more computational cost in processing and memory.

One advantage of the CLs is the parameter sharing, which allows the parameter count reduction, leading to less training cost and less data necessity. Due to their high connectivity, the upper FC layers have more parameters, and their spatial learning is not necessarily profitable in texture image recognition. In this context, Cimpoi, Maji & Vedaldi (2015) published the FV-CNN, which pools the last CL of a pre-trained network, using it as a latent representation. The pooling allows using input images without resizing them to fit the CNN with the FC (FC-CNN). They used the FV-CNN features as input to an SVM and compared the FC-CNN and SIFT features. The FV-CNN has shown to be a good texture descriptor, performing well on several benchmarks.

The texture CNN (TCNN) (Andrearczyk & Whelan, 2016) uses a similar approach to the FV-CNN, but with the classification accomplished by a sequence of FC layers. Its difference

from a traditional FC-CNN is the energy layer, a GAP at the last CL whose outputs go to the FC layers. In contrast to FV-CNN, the SVM replacement by the FC layers as a classifier permits training the CLs in conjunction with the classifier. This last approach does not require a pre-trained CNN as it can train it with the FC layers, but it can exploit transfer learning on low data scenarios, where the FC layers would require more data.

As cited by Humeau-Heurtier (2019), some medical images acquired by MRI, CT, radiography, or microscopy have characteristics of texture images. Microscopy usually produces HIs, which allow the observation of tissues, their cells, and their nuclei, with an appearance more texture oriented with repeated small patterns, with reduced shape and spatial information. Although these images have structures less complex than macro world images, they pose a challenge to classification, mainly due to the few data availability and variations in the staining process. Thus it is a defiant domain to explore with alternative CNNs methods.

Cheplygina, de Bruijne & Pluim (2019) dedicated a section of their review about medical images to the transfer learning technique, covering three situations: same domain and different tasks; different domain and the same task; and both different domains and tasks. They state that this technique is proper for medical images due to few labeled images for supervised learning methods.

Spanhol, Oliveira, Cavalin, Petitjean & Heutte (2017) used the features extracted from an AlexNet, pre-trained with ImageNet, in the context of CAD for breast cancer. They analyzed the features from the last three FC layers, 4096, 4096, and 1000 wide. The feature size creates difficulties for some classifiers due to computational complexity, so they used Logistic Regression as the base classifier, which allowed even the analysis of feature merging.

Compared with the AlexNet training from scratch proposed by Spanhol, Oliveira, Petitjean & Heutte (2016), the performance of the CNN feature extraction was similar and required less computational effort. de Matos, Britto, Oliveira & Koerich (2019) used an InceptionV3 and provided results comparable to the state-of-the-art by the time of the comparison, superior to the

previously mentioned works. The InceptionV3 also provides large feature vectors, with 2048 attributes, requiring a PCA before using it on the SVM classifier.

Deniz *et al.* (2018) used three CNNs to analyze which provides the best features for burned tissue images. They tested the features with four different classifiers, concluding that combining a ResNet-152 and SVM yields the best results.

The challenges with few amount of data and normalization problems on HIs require extensive data augmentation for CNNs. The reduced spatial and shape characteristics make it difficult to use well-known data augmentation methods based on macro object context. Therefore, methods that can work with fewer data and focus on these image characteristics could benefit this study area. The filters of CNNs, aside from detecting texture-like structures very well, produce an output vector with characteristics trained for the FC layer of the CNN. It is interesting to develop and train CL which output helps the classifier on its task, which are feasible by borrowing some concepts of Deep Metric Learning.

1.2 Metric Learning and Deep Metric Learning

Metric learning is an alternative to classification methods in situations where the number of classes is high, at the order of thousands, or where the number of samples of each category is low (Chopra *et al.*, 2005). Two examples of this situation are face recognition and signature verification. The idea of dissimilarity metric learning is to learn a representation to identify two samples' similarities, usually employing a distance metric, e.g., Euclidean Distance.

There are handcrafted techniques to map images to smaller dimensions, but they suffer from high sensibility to affine transformations, e.g., rotation, scaling, or shifting. Chopra *et al.* (2005) presented a method that uses siamese convolutional neural networks and the energy-based model. It consists of using raw energy values instead of probabilistic normalized ones. The concept of energy in their work is close to the one of Andrearczyk & Whelan (2016). The advantage of using CNNs is that they provide end-to-end training, learn low and high-level features, and provide shift-invariant detectors. Their method aggregates the energy output of

each siamese network into one neural network trained with contrastive loss. The loss allows training the system together (siamese networks and the single neural network), improving the sample representation on the energy layer.

The FaceNet (Schroff, Kalenichenko & Philbin, 2015) inspired our approach because it uses a single CNN and a monolithic classifier, a k-NN, at the end of the model. It focuses on the face recognition problem as in Chopra *et al.* (2005), especially the selection of positive and negative samples to compare the queries. Using all training instances may lead to slow convergence since some are similar and do not contribute meaningfully to the training. They proposed a batch process that improves the training instance selection. Moreover, the positive and negative comparing samples (triplet) use a filtering process, which only uses the farthest positive and the closest negative inside a minibatch.

The target of this approach is multiple class problems, each with a small number of samples. It includes a loss function and a batch construction method to maximize the comparisons with negative instances. The loss function minimizes the distance to a positive anchor at each batch while maximizing it to the negatives. It reduces the so-called hard mining that pools all negative instances to find a set of good ones that could maximize the separation. This method randomly selects many classes opposite the query class and extracts features from a few instances of each one. It randomly picks one of them and continuously adds more and more to the batch until N, selecting the ones that violate the loss constraints to the query class.

Wang *et al.* (2019) proposed a more elaborated training sample and anchors selection called Ranked List Loss. It relies on a threshold margin that gives more attention, using weighting, to the samples that maximize the margins between opposite classes. The margin determines the negative points too close to the query, breaking the margin constraint. There can be many negative points, so they proposed a weighting mechanism to penalize most ones that cause more constraint violations. The positive points do not need weighting since the face recognition problem usually presents more negative examples than positive ones. Their method was tested with three datasets, obtaining state-of-the-art results in all of them.

Although the deep metric learning approaches address few samples and a high number of class problems, the idea of learning a representation based on images resembles the one of feature extraction. When extracting features using CNNs, the embedding function usually maintains immutable, just as a handcrafted feature extractor. Deep metric learning can improve the feature extractor for similarity problems. Analogously, we based our method on this feature tuning but in the context of classification with large margin classifiers. Our idea still has the advantage of automatically selecting the positive and negative anchors based on the SVs.

1.3 Histopathologic Images

Cancer is a disease with a high mortality rate in developed and in developing countries. The impact of the disease is the loss of lives and the high costs for treatment, reflected in the population and governments. According to Torre *et al.* (2015), mortality rates among high-income countries are stabilizing or even decreasing due to programs to reduce the risk factors (e.g., smoking, overweighting, physical inactivity) and treatment improvements. In low and middle-income countries, mortality taxes increase due to increased risk factors. Improvements in treatment include early detection. In 140 out of 184 countries, breast cancer is the most prevalent type among women (Torre, Islami, Siegel, Ward & Jemal, 2017). Imaging exams like mammography, ultrasound, or CT can diagnose the presence of masses growing in breast tissue, but only the biopsy can confirm the tumor type.

Biopsies take more time to provide a result due to the acquiring procedure (e.g., fine-needle aspiration or surgical open biopsy), tissue processing (creation of the slide with the staining process), and pathologist analysis. Pathologist analysis is a highly specialized and time-consuming task prone to inter and intra-observer discordances (Bellocq *et al.*, 2011). The staining process can also cause variance in the analysis process. Hematoxylin and Eosin (H&E) is the most common and accessible stain, but they can produce different color intensities depending on the brand, the storage age, and temperature.

Formerly, pathologists used to do the biopsy process directly on the microscope, analyzing the slides with the prepared tissue. The advances in technologies allowed the coupling of the microscope with digital cameras. This provided a way to register the images of the slides and store them for post-analysis, e.g., by another pathologist. This process is not state-of-the-art on HIs. It still requires the direct intervention of the pathologist on the microscope to find regions of interest on slides and zoom in to obtain detailed images. Examples of these image types can be seen in datasets proposed by Spanhol *et al.* (2016), Kostopoulos *et al.* (2009), Ninos *et al.* (2016), and Glotsos *et al.* (2008). Another procedure to obtain the HI is the whole slide image (WSI). This process is almost automatic, requiring human intervention only in preparing and loading the slides onto a specialized machine. The idea behind the WSI is the high-resolution acquiring process. The slide is digitalized as a single image of, e.g., 1,600 megapixels, as a final product. This image allows the pathologist to analyze results remotely, requiring only a technician to prepare and place the slides into the machine. Despite the great advantages of WSI, it is much more expensive than microscope-mounted cameras and also requires many storage spaces for the final images. The data storage space also makes it difficult to create online available datasets of these images.

The time to return the patients' results from a biopsy is important because the treatment should start as soon as possible. As mentioned, biopsies are time-consuming and expensive. In the United States, 1.6 million women are submitted annually to biopsies, but only a quarter of the exams result positive for malignant breast tumors. This factor increases the queue of exams, possibly delaying the treatment's start for those requiring (Elmore *et al.*, 2015). Computer-aided diagnosis (CAD) systems can help reduce the time to analyze the images, but relying only on the computer to accomplish the analysis is risky. Even the results of pathologists, in some cases, require a second opinion or even a third one. Gurcan, Boucheron & Can (2009) raised questions related to inter-observer, intra-observer, and physical and psychological factors in the image analysis by pathologists. Inter-observer refers to the disagreement between malign and benign sample labeling by two pathologists. Intra-observer variations occur when a single pathologist observes the same sample within an interval of time, for example, one day. His conclusions can

differ between the two observations due to physiologic or psychological factors. CAD systems can provide more security to the pathologist to emit a result, working as a complementary opinion and a more deterministic actor.

The HI analysis can be seen as three pattern recognition problems: classification, regression, or segmentation. The segmentation on HI usually extracts image elements like morphological features related to nuclei size, density, and shape for posterior analysis. Sometimes, segmentation precedes the classification to remove irrelevant information like background or stroma. The regression problems are seen in prognosis studies, which estimate the survival time or rate based on an image. The classification task aims to label an image in a certain category, e. g. benign or malign tumors, types of tumor, a grade based on Gleason (Gleason, 1992) or Nottingham (Galea, Blamey, Elston & Ellis, 1992) systems.

Figure 1.1 depicts examples of HI from four datasets, BreaKHis (Spanhol *et al.*, 2016), HICL (Kostopoulos *et al.*, 2017), CRC (Kather *et al.*, 2016) and ICIAR 2018 Challenge (Aresta *et al.*, 2019). All these datasets are publicly available and based on the hematoxylin and eosin (H&E) staining process. We can cite P63 and immunohistochemistry (IHC) as other examples of staining. HI uses H&E stain in various tissues, e.g., brain, breast, colorectal, cervix, prostate, and lung, to create HI.

HIs contain cellular-level structures. Depending on the magnifying factor, it is possible to observe nuclei and stroma. The nuclei shape can identify, e.g., mitosis, indicating how the cells multiply. This is the case in works like the ones published by Awan, Aloraidi, Qidwai & Rajpoot (2016), Wan *et al.* (2017), Albarqouni *et al.* (2016), Zerhouni, Lányi, Viana & Gabrani (2017) and the ICPR contest of 2012, 2013, and 2014 (Veta *et al.*, 2015). Another task other than mitosis detection can be, e.g., necrosis detection as presented by Homeyer *et al.* (2013), stenosis detection on the liver presented by Vanderbeck, Bockhorst, Komorowski, Kleiner & Gawrieh (2014), and cervical cancer classification using textural analysis (Rahmadwati, Naghdy, Ros, Todd & Norahmawati, 2011). Although invasive ductal carcinoma, a type of breast cancer, can be detected by analyzing cell mitosis, it is important to observe other characteristics of HI from

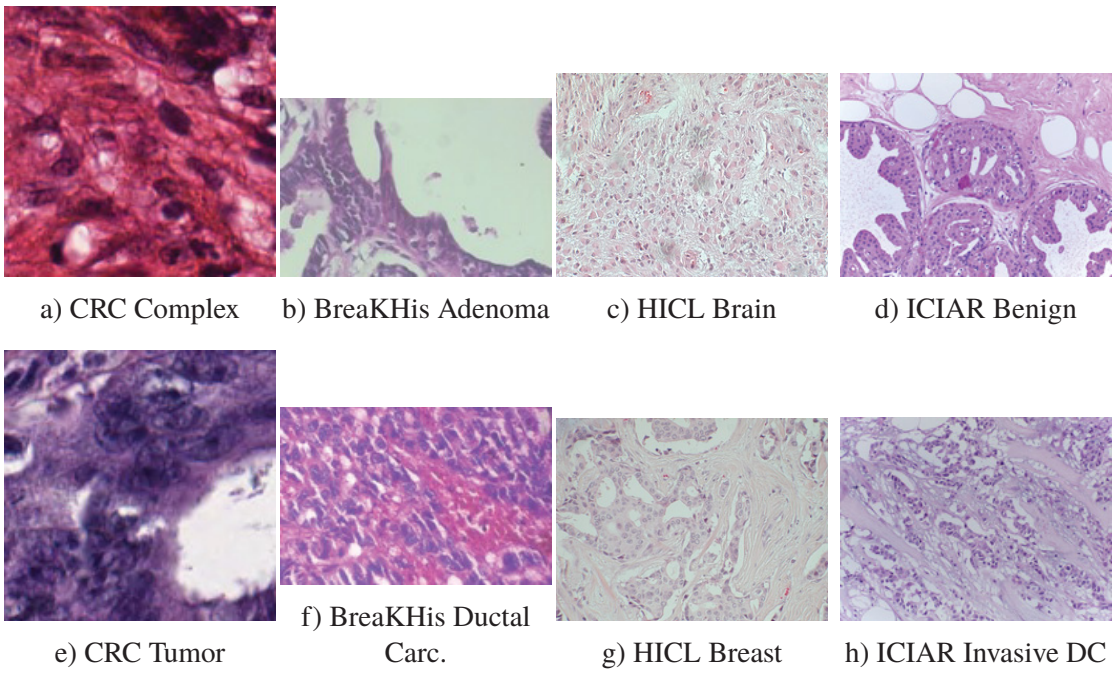


Figure 1.1 Examples of HIs

breast tissue to distinguish between other types of tumors. For example, lobular carcinoma, a malign breast tumor, affects the milk glands. Papillary carcinoma (also a malign tumor) attacks the ducts and epithelial cells but not the cells below it. The edge of the tumor also does not invade the stroma (Rubin, Strayer & Rubin, 2012). So it is also important to have an overall vision of the region to differentiate between tumors, having information of the rate or amount of some types of texture that appear together in a single image.

Textural descriptors are widely used to analyze HI stained with H&E (Caicedo, Gonzalez & Romero, 2008) (Kuse, Sharma & Gupta, 2010) (Atupelage *et al.*, 2013) (Fernández-Carrobles *et al.*, 2015) (Leo *et al.*, 2016) (Das *et al.*, 2017) (Peyret, Bouridane, Khelifi, Tahir & Al-Maadeed, 2018) (Michail, Dimitropoulos, Koletsa, Kostopoulos & Grammalidis, 2014) (Bruno *et al.*, 2016) (Chan & Tuszynski, 2016) (Phoulady, Zhou, Goldgof, Hall & Mouton, 2016) (Reis *et al.*, 2017). Morphometric features are also used in works like the ones presented by Ballarò *et al.* (2008), Loeffler *et al.* (2012), Song, Lee, Choi & Chun (2013), Michail *et al.* (2014), Fukuma, Prasath, Kawanaka, Aronow & Takase (2016) and Niazi, Parwani & Gurcan

(2016). Still, they require more steps for the analysis because usually, it is necessary to segment the image, find the elements, e.g., nuclei, and obtain measures.

According to de Matos, Ataky, de Souza Britto Jr, Soares de Oliveira & Lameiras Koerich (2021) and Zhou *et al.* (2020), there is an increase in the use of deep learning methods in the context of HIs. This trend follows the other areas where the use of these methods also increased. As mentioned before and stated by Banerji & Mitra (2022), between the challenges of HI datasets, the lack of labeled data contrasts with the needs of the training methods for deep learning. Thus, the pre-trained deep learning methods in the context of transfer learning are more frequently used (Zhou *et al.*, 2020) like in Wahab, Khan & Lee (2019), Mahbod, Ellinger, Ecker, Smedby & Wang (2018), and Vesal, Ravikumar, Davari, Ellmann & Maier (2018) that use pre-trained networks and fine-tuned the parameters in the context of breast cancer HI. Song, Zou, Chang & Cai (2017), Mehra *et al.* (2018), and Thuy & Hoang (2020) are examples of using deep learning methods as feature extractors to produce a representation to be classified by a traditional classifier. In Mehra *et al.* (2018), it is shown the ability of transfer learning in classifying HIs using three well know CNNs (Resnet50, VGG16, and VGG19) and a logistic regression classifier. Thuy & Hoang (2020) also used VGG16 and VGG19 as feature extractors and a neural network to classify HIs. They augmented the HI dataset using two generative adversarial networks based on StyleGAN and Pix2Pix methods, obtaining the best results compared to the non-augmented data. The work of Song *et al.* (2017) used a VGG19 allied to a Fisher Vector to better adapt the features to be classified by a linear SVM. A mitosis classification method is proposed by Wahab *et al.* (2019). They fine-tuned a VGG16 to segment nuclei in HIs and used a hybrid CNN based on pre-trained weights of an AlexNet to classify mitosis stages. The work of Mahbod *et al.* (2018) used the fine-tuning of two different depth ResNets and fused their results to obtain the final classification. Vesal *et al.* (2018) fine-tuned two CNNs, a ResNet50, and an Inception-V3, to classify patches of HIs and used majority voting over the results of the CNNs and patches to obtain the predicted class of a big image.

Other works like Jiang, Chen, Zhang & Xiao (2019), Chen, Dou, Wang, Qin & Heng (2016), and Alom, Yakopcic, Nasrin, Taha & Asari (2019) used more advanced deep learning methods.

In Jiang *et al.* (2019), a ResNet is coupled with a squeeze-and-excitation block to classify breast cancer HIs. A CasCNN is proposed by Chen *et al.* (2016) to detect mitosis. Their model includes a CNN that first selects mitosis candidate regions and a second pre-trained CNN to discriminate the mitosis. The work from Alom *et al.* (2019) used inception recurrent residual CNN that combines the three main CNNs architectures that compose its name, the inception and residual blocks, and the recurrent CNNs. The multi-instance learning (MIL) is also used in HI problems, like in Li *et al.* (2021b), Sudharshan *et al.* (2019), and Li *et al.* (2021a). This last work used MIL and an EfficientNet-B0 in multi-resolution problems of HIs.

The information that allows one to identify a tumor or classify its type may be small relative to the image size. This is one of the reasons the pathology analysis of these images is challenging. The problem worsens when the HIs are WSIs because the images have a high resolution of an entire slide, not patches, achieving gigapixel count (Hou *et al.*, 2016). The work of Huang *et al.* (2021) uses the attention mechanism (Guo *et al.*, 2022) to avoid the locality of the receptive field of the CNNs. Multiple attention blocks are used in various CNN levels to identify different information. Yang, Kim, Kim & Adhikari (2019) created a guided soft attention network including coarse-level annotation to guide the attention of a CNN. Their method surpasses the state-of-the-art in the BACH dataset and provides a visual explanation from activation maps that helps understand the classification results. In Dabass, Vashisth & Vig (2021), an attention-guided training method using a deep residual U-Net architecture is presented. It is used in the gland segmentation task and merges the concepts of the attention mechanism, residual learning, and multi-scale feature fusion. The contributions are the precise segmentation of complex scenarios of inconsistent appearance and complex touching margins glands. Another deep learning method that emerged in recent years is the transformer (Vaswani *et al.*, 2017). It originates in the attention mechanism, but unlike previous works, it relies solely on the attention mechanism and not on RNNs or CNNs. Chen *et al.* (2022) presented the GasHis-Transformer, which uses a transformer to detect Gastric cancer in HIs. It uses two modules, the global and local information modules. The first module extracts information from the whole HI, and then the second module uses the global information with an Inception-V3 architecture to process images in a multi-scale

fashion. Their method obtained excellent generalization ability with gastric HIs and other cancer HIs. The Deconv-Transformer (He *et al.*, 2022) is also a HI classification method based on transformers. It includes a color deconvolution module and surpasses the performance of compared traditional CNN methods. Stegmüller, Bozorgtabar, Spahr & Thiran (2023) presented the Scorenet, a transformer that leverages the score of image regions to improve classification results. They also include an augmentation module that, together with the transformer, obtained state-of-the-art results with only 50% of the data compared to traditional methods. Attention mechanisms and transformers are promising approaches that may be incorporated into many deep learning methods.

As the machine learning field evolved and deep learning approaches became common, they also started to be applied to HI problems. Although these methods demand a high amount of data to train and avoid overfitting, researchers could circumvent this problem, obtain reasonable results, and improve the results in the HI context.

CHAPTER 2

LARGE-MARGIN FULLY CONVOLUTIONAL NETWORKS

This section presents the LMFCN approach that includes the fully convolutional network (FCN) description, the large-margin discriminant training, the loss functions that are used to update the FCN weights and the multiclass approach. We also present a brief analysis of the LMFCN computational costs and the impacts of the loss functions in the weights update.

2.1 An overview of the LMFCN

The LMFCN is an approach that uses an FCN to generate a latent representation from images and trains a large-margin discriminant that can classify these images according to their classes. It uses information gathered during the training of the discriminant, like the support vectors (SVs) and classification results, to update the weights of the FCN and improve the latent representation to ease the training of the discriminant in the next iteration. Figure 2.1 shows an abstract representation of this approach. The FCN architecture is detailed in Figure 2.5.

The LMFCN has some distinct characteristics that can be summarized as follows:

- It learns representations directly from the raw data like CNNs.
- It uses a global average pooling layer to produce a low-dimensional latent representation.
- Instead of a fully connected layer, it uses an SVM as a discriminant.
- Besides learning a discriminant, the SVM also allows it to find out what training instances (SVs) most impact such a discriminant and the mis- and well-classified instances. Such instances are used as anchor points.
- The anchor points are used to calculate the specially developed loss functions to guide the update of the convolutional layers, similar to DML.
- The loss function aims to move the SVs in the latent representation space towards their closest well-classified anchors to increase the margins, move the misclassified instances in the direction of the closest SVs to correct them, and move the closest well-classified instances of the opposite class to the opposite direction, also increasing the margins.

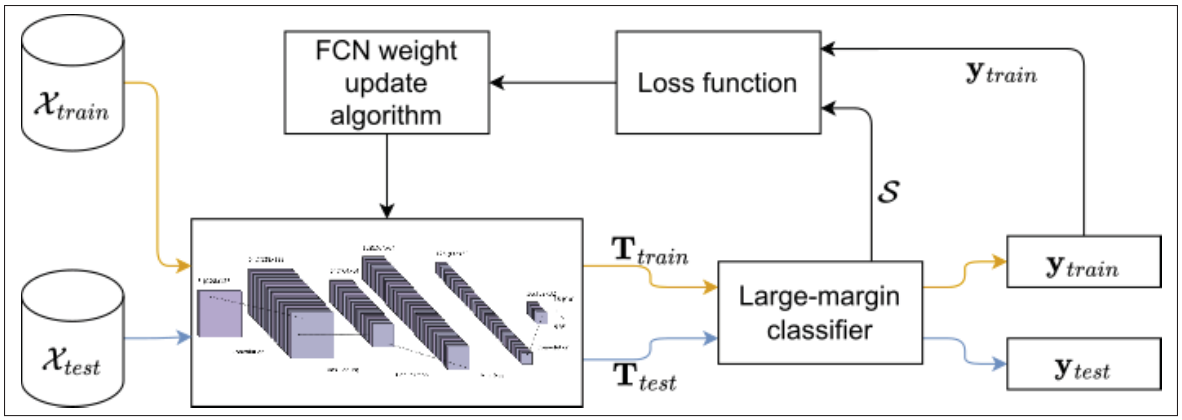


Figure 2.1 An abstract representation of the LMFCN. $\mathcal{X}_{train}$ and $\mathcal{X}_{test}$ are the train and test sets. $\mathbf{T}_{train}$ and $\mathbf{T}_{test}$ are the train and test latent representation matrices of all images from these sets. $\mathbf{T}_{train}$ is used to train the large-margin classifier and provide the classification results and the support vectors ($\mathcal{S}$) to the loss functions. Finally, it utilizes the loss error to update the FCN weights. This process is accomplished iteratively. In the end, $\mathcal{X}_{test}$ (blue line) can be fed to the FCN, generating the latent representation to the classifier to obtain the classification result.

- The training algorithm updates the parameters of the convolutional filters and, consequently, the latent representation in an iterative way, recalculating the SVM and anchor points at each iteration with the newly generated latent representation from the FCN trained in the previous iteration.
- At the end, the iterative training ends at a chosen stop criteria, producing a model comprising the trained FCN and SVM, forming an end-to-end raw data classifier.

2.2 LMFCN

The key point of the proposed approach is that it does not use all training instances. Instead, it selects and uses only the most relevant instances during training, reducing the training complexity. Such instances, named anchors, are used with a novel loss function, which speeds up training.

The proposed method has four components: a fully convolutional network (FCN) with a global average pooling (GAP), a large-margin classifier, a set of anchor instances, and a novel loss

used to optimize the parameters of the convolutional filters. The LMFCN uses a sequence of CLs acting as a filter bank to generate a latent representation from raw data (images). This representation is fed to a large margin discriminant that can classify the data and provide information on how the data is distributed. In addition, the backpropagation algorithm is used with a newly created loss function to update the filters to generate a new latent representation with the same raw data but separated with larger margins. The loss function is based on anchor points defined by the information from the large margin discriminant and calculates the distance between instances of interest and their anchors. The loss function has three terms that can be used independently. Thus, only one may be used between the three, two combined, or three. The most important term uses only the SVs as instances of interest. According to our experiments, this term can solve most problems and use only a subset of the training set. The other two terms use mis- and well-classified instances as instances of interest. The three terms together end up using the whole training set, but the use of all terms together is not mandatory, as we show in the evaluations.

2.2.1 Anchor Instance Selection

Anchors are essential in the proposed method because the FCN weight update procedure relies only on such instances. Anchors are instances from the training set used as references by the loss function to calculate the distance to instances of interest. Instances of interest are the samples that we want to move in the latent representation to increase or decrease the distance (similarity) to the anchors. The backpropagation algorithm minimizes such distances during training.

The LMFCN addresses the problem of selecting anchors and instances of interest using the large margin principle. It creates a list of instances and anchors with the idea that increasing the margin also reduces within-class dissimilarity (or distance) and increases between-class dissimilarity. SVM is also based on margins, defining the largest possible margin to separate classes.

The SVM is a supervised learning method based on SVs. Its purpose, simplifying the definition, is to find instances in the training set that are close to other classes and also can draw the largest margin possible between classes. These instances are the SVs that, in the LMFCN, become instances of interest and anchors, depending on the situation our algorithm tries to solve. The SVs are the most complicated instances to classify, based on similarity measures, being close to the opposite category. The other training instances that are not SVs are near similar samples and away from dissimilar ones. This idea is akin to a DML problem, where there are instances we want to move away from the opposite and closer to the same class. The characteristics of SVs are similar to DML. Thus the closer to the same category we move them, the bigger the margin becomes. Typical classification problems with SVM do not change representation or features. The only option is to find the best possible SVs to draw margins, which differs from DML, where representation enhancement is allowed.

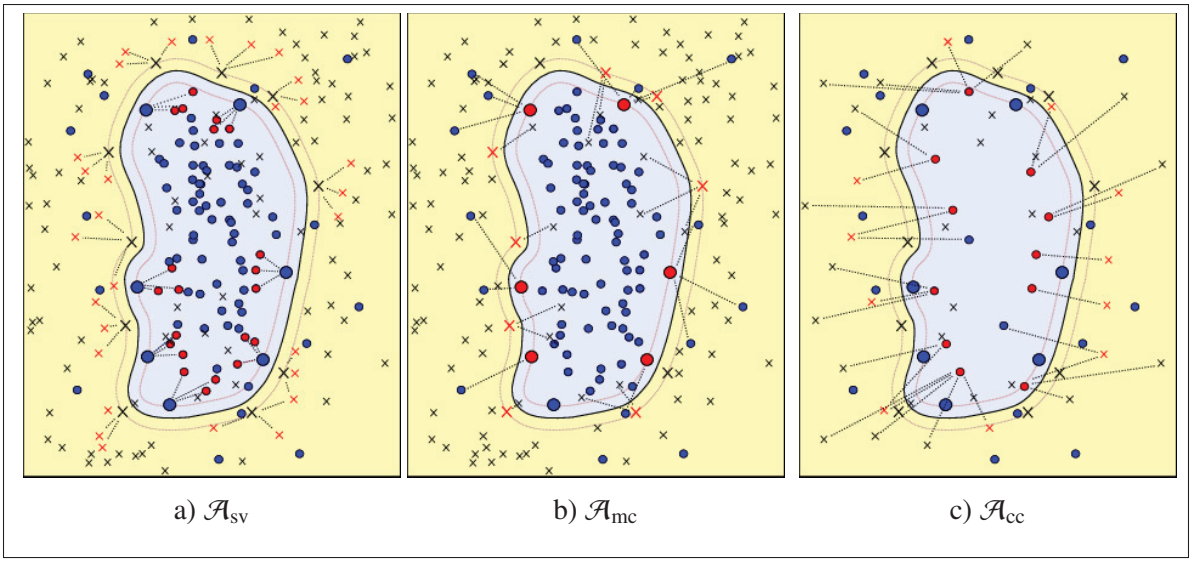
SVM also has two other interesting characteristics, kernel flexibility, and the soft margin definition. Easier problems, linearly separable, can rely on a simple linear kernel, reducing computational costs. Still, when the class separability is more complicated, we can use another kernel type. One of them is the Radial Basis Function (RBF), which has its similarity measure based on a Euclidean distance and allows more complex decision boundaries. The RBF kernel accepts a hyperparameter, called γ, that dictates the spread of the kernel function. Higher γ values create narrow margins, more islands, and sharpened contours, often leading to overfitting and many SVs to draw a complex margin.

Another characteristic of the SVM is its soft margin that controls the lenience to misclassified instances in the optimization process, defined by the hyperparameter C. Some problems cannot converge into an optimal solution due to complex cases, or they may require too many SVs and narrow margins producing an overfitted margin boundary. The soft margin allows controlling the tolerance to misclassified instances during the training. In this case, the SVs are difficult instances in the borders but not so overlapped with the opposite class.

We used the closest correct classified instances to SVs as our anchors and SVs as the instances of interest. The SVs and each instance's class prediction define the anchors and the instances of interest. Figure 2.2a presents a latent representation space split into two regions by an RBF SVM classifier. Our algorithm uses the distance calculation between SVs and their anchors as a loss function, which, when minimized, causes an approximation of SVs to anchors. As in DML, we can change the latent representation generated by the FCN using backpropagation with our loss function. Looking at all SVs in Figure 2.2a, the overall view shows that their movement towards their anchors produces margin enlargement. They are the instances in the most complicated situation (closer to the decision boundaries) in the latent representation space, apart from the misclassified ones. Figure 2.3a shows in detail a specific region and helps understand the movement direction of SVs from both classes. Of course, changing the FCN aiming to move SVs also impacts all instances around them with similar latent representations because they share some characteristics. Therefore, such instances will also experience movements in a similar direction. In our method, these anchors are denoted as $\mathcal{A}_{\text{sv}}$.

The impact of well-selected hyperparameters C and γ is visible in Figure 2.2a, with a smoother margin and misclassified instances. These parameters play an essential role in our method, reducing the number of SVs and generating more effective anchors with consistent movement directions. A high number of SVs due to a high C or γ can drastically reduce the number of correctly classified instances that are not SVs, reducing the number of anchors available. Fewer anchors lead to movements in directions that may not be adequate. Another challenge to this anchor selection method is a latent representation with highly overlapped classes, generating many SVs and reducing anchors' choice options.

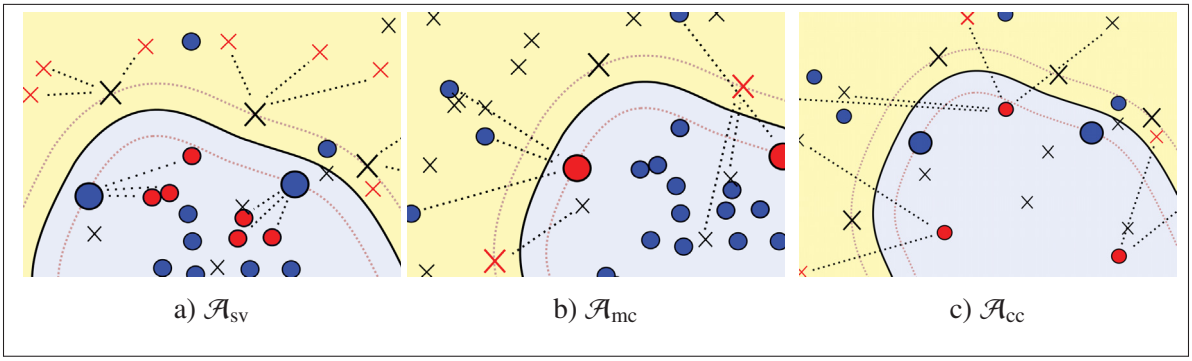
Unlike a conventional CNN, where the discriminant is improved at each epoch, the discriminant produced by the SVM in each epoch is discarded because the latent representation changes due to the updates in the FCN weights. Only the SVM model from the epoch determined by stop criteria, in our case, the SVM model which produced the biggest validation accuracy, is used as a discriminant for the latent representation from images generated by the FCN from the same epoch.



a) $\mathcal{A}_{sv}$ b) $\mathcal{A}_{mc}$ c) $\mathcal{A}_{cc}$

Figure 2.2 A 2D representation space for two-class instances and boundaries generated by an RBF SVM. Circles are samples of class 0, and crosses are samples of class 1. Bigger symbols mean the Support Vectors for each class. In blue is the region of class 0, and in yellow is the region of class 1. Dashed straight black lines link the instances to their anchors. In red are the anchors for three different situations. a) are anchors of reference to the SVs ($\mathcal{A}_{sv}$), b) are anchors to move the misclassified instances ($\mathcal{A}_{mc}$) and c) are anchors to increase the separation of instances from opposite classes ($\mathcal{A}_{cc}$)

.



a) $\mathcal{A}_{sv}$ b) $\mathcal{A}_{mc}$ c) $\mathcal{A}_{cc}$

Figure 2.3 Anchors type zoom

Two other anchor types are also defined in our approach, $\mathcal{A}_{mc}$ and $\mathcal{A}_{cc}$. $\mathcal{A}_{mc}$ are the SVs closest to a misclassified instance. In the latent representation space of Figure 2.2b, lines are linking the misclassified samples to their anchors (SVs), with a detailed representation in Figure 2.3b. This

anchor type aims to correct the misclassified instances. They use SVs as anchors because they are the closest correctly classified instances and do not induce substantial updates: the correctly classified instances that are not SVs are too distant from a misclassified instance. Thus, there is a small update that does not cause great disturbance in the other class.

$\mathcal{A}_{\text{cc}}$ are characterized by the closest correctly classified instances of the opposite class, as shown in Figure 2.2c. $\mathcal{A}_{\text{cc}}$ helps to maximize the distance between instances of different classes (between-class distance). This anchor spreads the instances in the representation space but may cause movements in unfruitful directions, which may cause overlapping in other parts of the space when dealing with complex data distributions. Such an anchor is also computationally costly; the more the model produces well-classified samples, the more costly it becomes with more instances of interest and anchors. Figure 2.3c allows a better observation and understanding of the $\mathcal{A}_{\text{cc}}$ definition.

It is essential to note that the number of anchors, regardless of their type, is a hyperparameter in LMFCN. For instance, in Figure 2.2 we have 3, 1, and 1 anchor for $\mathcal{A}_{\text{sv}}$, $\mathcal{A}_{\text{mc}}$, and $\mathcal{A}_{\text{cc}}$, respectively. The hyperparameters should be tuned according to the problem's difficulty and distribution. In Section 3.4.7, we analyze the impact of these hyperparameters values on the accuracy and computational cost of the LMFCN.

Section 2.2.2 describes the algorithm of LMFCN and how the training algorithm defines and uses the anchors.

2.2.2 LMFCN Training

The LMFCN training algorithm includes five basic steps illustrated by colored rectangles in Figure 2.4. They are:

1. Convolution of kernels with the image to produce feature maps and GAP (latent representation);
2. Calculation of distance matrices **P** and **D** and a Gram matrix (**K**);
3. Training a kernel SVM on **T** to learn a discriminant and information on support vectors (**S**);

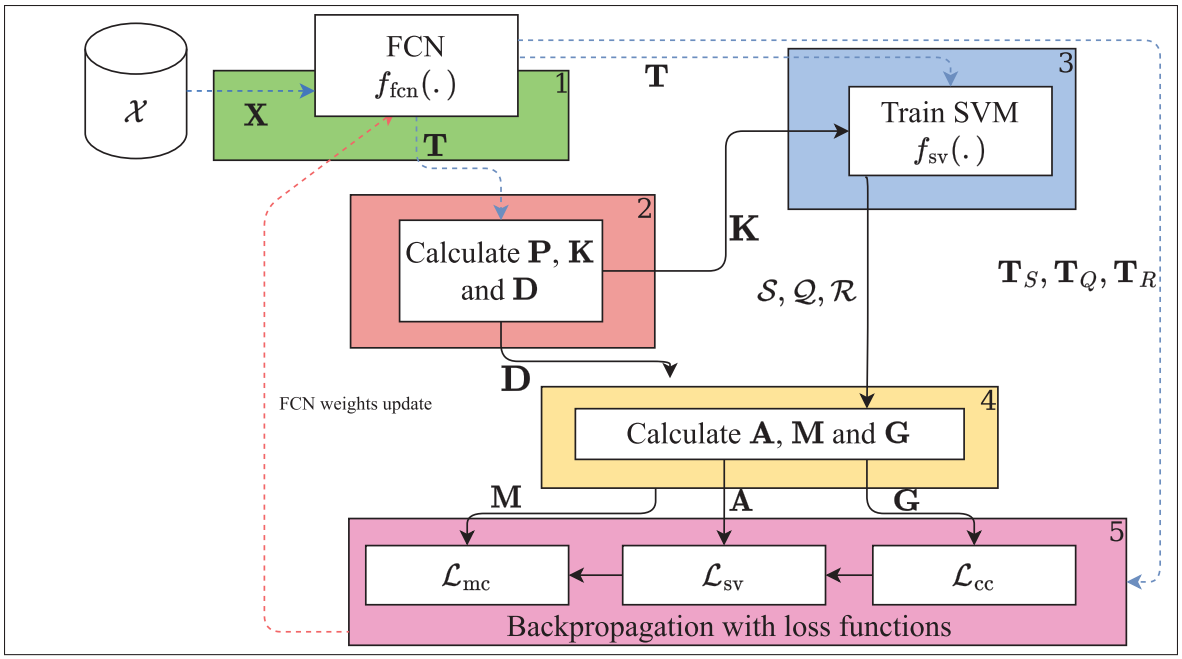


Figure 2.4 An overview of the LMFCN training scheme. Five main steps are highlighted: 1) An FCN generates a latent representation of the training set images. 2) The latent representation is used to compute the kernel of a large margin discriminant and the distance between instances. 3) Training of the large margin discriminant. 4) Calculation of auxiliary matrices to determine the anchor points of each instance of interest. 5) Loss terms computing and FCN weights update.

4. Creation of anchor matrices **A**, **M**, and **G**;
5. Calculation of a loss function that employs anchor points and backpropagation of the error to update the convolutional filters of the FCN and improve the latent representation;

The steps are also described in Algorithm 2.1, with steps 1 and 2 merged in the function *calculate_base_matrices*(.).

The training procedure starts with a dataset of size n denoted as $\mathcal{X} = \{\mathbf{X}^t, o^t\}_{t=1}^{n}$, where t indexes images $\mathbf{X}^t$ of width w, height h and c channels in $\mathcal{X}$, and $o^t \in [0, 1]$ is its label. All images from $\mathcal{X}$ are fed to a stack of multiple CLs denoted as $f_{\text{fcn}}(.)$ and produce a matrix $\mathbf{T}^{n\times\phi}$, with ϕ being the dimension of the latent representation (step 1 at Figure 2.4). The algorithm uses the matrix **T** to calculate the matrix **P** (Equation (2.1)), which in turn is used to calculate matrices $\mathbf{K}^{n\times n}$ and $\mathbf{D}^{n\times n}$ using Equations (2.2) and (2.3), respectively (step 2 at Figure 2.4) described in Algorithm

2.2. **D** is essential to the rest of the algorithm as it contains the pairwise distance of all instances and is used to create the anchor matrices.

Algorithm 2.1 LMFCN training algorithm

Input: Set of n training images $\mathcal{X} = \{\mathbf{X}^t, o^t\}_{t=1}^{n}$ and randomly initialized $f_{\text{fcn}}(.)$
Output: $f_{\text{fcn}}(.)$, $f_{\text{sv}}(.)$

while !*stop_criteria* **do**
  $\mathbf{T}, \mathbf{P}, \mathbf{K}, \mathbf{D} \leftarrow calculate\_base\_matrices(\mathcal{X})$; // Step 1 and 2
  $\mathcal{S}, \mathcal{Q}, \mathcal{R}, f_{\text{sv}}(.) \leftarrow calculate\_instances\_of\_interest(\mathbf{T}, \mathbf{K}, \mathcal{X})$; // Step 3
  $\mathbf{A}, \mathbf{M}, \mathbf{G} \leftarrow obtain\_anchor\_matrices(\mathcal{S}, \mathcal{Q}, \mathcal{R})$; // Step 4
  $f_{\text{fcn}}(.) \leftarrow f_{\text{backpropagation}}(loss\_sv(\mathbf{A}, \mathbf{T}_{\mathcal{S}}) +$
    $loss\_mc(\mathbf{M}, \mathbf{T}_{\mathcal{Q}}) - loss\_cc(\mathbf{G}, \mathbf{T}_{\mathcal{R}}))$; // Step 5
end while

Algorithm 2.2 LMFCN - Base matrices calculation - *calculate_base_matrices*()

Input: Set of n training images $\mathcal{X} = \{\mathbf{X}^t, o^t\}_{t=1}^{\mathbf{n}}$
Output: Matrices **T**, **P**, **D** e **K**

$\mathbf{T} \leftarrow f_{\text{fcn}}(\mathcal{X})$
$\mathbf{P} \leftarrow 0$
for $i \in [0, n[$ **do**
  for $j \in [0, n[$ **do**
    for $b \in [0, \phi[$ **do**
      // **P** is a auxiliary matrix to save effort on computing **K** and **D**
      $p_{ij} \leftarrow p_{ij} + (t_{ib} - t_{jb})^2$
    end for
    // Kernel matrix
    $k_{ij} \leftarrow exp(-\gamma p_{ij})$
    // Distance matrix
    $d_{ij} \leftarrow \sqrt{p_{ij}}$
  end for
end for

K is an RBF kernel matrix used to train a large margin discriminant $f_{\text{sv}}(.)$, by optimizing the Lagrangian multipliers as denoted in Algorithm 2.3. The large margin discriminant produces

the output y^t for each element t of $\mathcal{X}$, and also provides a set of support vectors (SVs) indexes $\mathcal{S} = \{s^u\}_{u=1}^{v}$, where $s^u \in [0, n[$ and v is the number of SVs (step 3 at Figure 2.4).

Algorithm 2.3 LMFCN - Creation of auxiliary sets ($\mathcal{S}$, $\mathcal{Q}$, $\mathcal{R}$) - $calculate\_instances\_of\_interest(\mathbf{T}, \mathbf{K}, \mathcal{X})$;

Input: Matrix of latent representations $\mathbf{T}$, kernel $\mathbf{K}$ and $\mathbf{o}$ from $\mathcal{X}$
Output: Trained large margin discriminant $f_{\text{sv}}(.)$ and the sets $\mathcal{S}$, $\mathcal{Q}$ and $\mathcal{R}$

1 $f_{\text{margin}}(.) \leftarrow \underset{w,b}{min} \frac{1}{2} \parallel w \parallel^2 + C \sum_{i=0}^{n} max(0, 1 - o_i \cdot y_i)$ `// Soft margin SVM`

2 $f_{\text{lagrange}}(w, b, \alpha) \leftarrow f_{\text{margin}}(.) - \sum_{i=0}^{n} \alpha_i [o_i(w \cdot x + b) - 1]$

3 $f_{\text{dual}}(\alpha, \mathbf{o}, \mathbf{K}) \leftarrow \sum_{i=0}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=0}^{n} \alpha_i \alpha_j o_i o_j k_{ij}$

4 $f_{\text{sv}}(.), \mathcal{S} \leftarrow \underset{\alpha}{max} f_{\text{dual}}(\alpha, \mathbf{o}, \mathbf{K})$

5 $j \leftarrow 0$
6 **for** $i \in [0, n[$ **do**
    `// Select misclassified vectors to form Q`
7 **if** $f_{sv}(\mathbf{t}_i) \neq o_i$ **then**
8 $q_j \leftarrow i$
9 $j \leftarrow j + 1$
10 **end if**
11 **end for**

12 $j \leftarrow 0$
13 **for** $i \in [0, n[$ **do**
    `// Select correct classified vectors, except S, to form R`
14 **if** $i \notin \mathcal{S}$ *and* $i \notin \mathcal{Q}$ **then**
15 $r_j \leftarrow i$
16 $j \leftarrow j + 1$
17 **end if**
18 **end for**

$$p_{ij} = \sum_{b=0}^{\phi} (t_{ib} - t_{jb})^2 \tag{2.1}$$

$$k_{ij} = \exp(-\gamma p_{ij}) \tag{2.2}$$

$$d_{ij} = \sqrt{p_{ij}} \tag{2.3}$$

The $\mathcal{A}_{sv}$ (step 4 at Figure 2.4) are obtained from matrix $\mathbf{D}$, the set $\mathcal{S}$, the expected output o^t, and the output from $f_{\text{sv}}(f_{\text{fcn}}(\mathbf{X}^t))$, as shown in Equation (2.4):

$$e_{ij} = \begin{cases} \tau & \text{if } i = j \textbf{ or } j \in \mathcal{S} \text{ or } o^i \neq o^j \text{ or } o^j \neq f_{\text{sv}}(f_{\text{fcn}}(\mathbf{X}^j)) \\ d_{ij} & \text{otherwise} \end{cases} \tag{2.4}$$

where e_{ij} is either the distance between SVs and their anchors (d_{ij}) or a large constant τ that indicates that there is no SV-anchor relation (uninteresting instances).

Furthermore, we also define a sorting function $f_{\text{argsort}}(.)$ that takes a vector as input and returns a vector of indices sorted in increasing order, in this case, the increasing order of distances between SVs and their anchors stored in the vector $\mathbf{e}$. So, the anchor matrix $\mathbf{A}^{s\times n}$ is calculated using Equation (2.5). Algorithm 2.4 describes how the $\mathbf{A}$ matrix is computed.

$$\mathbf{a}_u = f_{\text{argsort}}(\mathbf{e}_{s^u}) \text{ with } s^u \in \mathcal{S} \text{ and } u \in [0, |\mathcal{S}|[\tag{2.5}$$

Likewise $\mathbf{A}$, we calculate with Equations (2.7) and (2.9) in Algorithm 2.5 the matrices $\mathbf{M}^{m\times n}$ and $\mathbf{G}^{g\times n}$ for $\mathcal{A}_{\text{mc}}$ and $\mathcal{A}_{\text{cc}}$, respectively (step 4 at Figure 2.4). We define a set of numbers (indexes) to indicate the misclassified and correctly classified instances from $\mathcal{X}$ as $\mathcal{Q} = \{q \mid q \subset \mathbb{N} : [0, n[\cap f_{\text{sv}}(f_{\text{fcn}}(\mathbf{X}^q)) \neq o^q\}$, and $\mathcal{R} = \{r \mid r \subset \mathbb{N} : [0, n[\cap r \notin \mathcal{S} \cup \mathcal{Q}\}$, respectively.

$$z_{ij} = \begin{cases} d_{ij} & \text{if } i \in \mathcal{Q} \text{ and } j \in \mathcal{S} \text{ and } i \neq j \\ \tau & \text{otherwise} \end{cases} \tag{2.6}$$

$$\mathbf{m}_i = f_{\text{argsort}}(\mathbf{z}_{q^i}) \text{ with } q^i \in \mathcal{Q} \text{ and } i \in [0, |\mathcal{Q}|[\tag{2.7}$$

Algorithm 2.4 LMFCN - Loss auxiliary matrix **A** - *obtain_anchor_matrices*()

Input: Sets $\mathcal{S}$, $\mathcal{Q}$, $\mathcal{R}$ with the indexes to the instances of interest to each loss type
Output: Matrix **A** with the indexes to the closest anchors to each instance of interest

for $i \in [0, n[$ **do**
  for $j \in [0, n[$ **do**
    `// Use the auxiliary matrix` **E** `to determine the closest anchors to each SV`
    if $i = j$ ***or*** $j \in \mathcal{S}$ ***or*** $o^i \neq o^j$ ***or*** $o^j \neq f_{sv}(f_{fcn}(\mathbf{X}^j))$ **then**
      $e_{ij} \leftarrow \tau$
    else
      $e_{ij} \leftarrow d_{ij}$
    end if
  end for
end for
for $i \in [0, |\mathcal{S}|[$ **do**
  $\mathbf{a}_i \leftarrow f_{\text{argsort}}(\mathbf{e}_{s_i})$
end for

$$h_{ij} = \begin{cases} d_{ij} & \text{if } i \in \mathcal{R} \text{ and } j \in \mathcal{R} \text{ and } o^i \neq o^j \\ \tau & \text{otherwise} \end{cases} \tag{2.8}$$

$$\mathbf{g}_i = f_{\text{argsort}}(\mathbf{h}_{r^i}) \text{ with } r^i \in \mathcal{R} \text{ and } i \in [0, |\mathcal{R}|[\tag{2.9}$$

where z_{ij} and h_{ij} in Equations (2.6) and (2.8) are the distance between misclassified instances and the SVs (d_{ij}) and the distance between correctly classified instances from opposite classes (d_{ij}), respectively. Again, τ is a large constant that indicates uninteresting instances.

Step 5 is the backpropagation of the error computed at the output of the loss functions to calculate the gradients and update the CL weights. The loss functions use the latent representation of three subsets of images $\mathcal{X}^{\mathbf{S}}$, $\mathcal{X}^{\mathbf{Q}}$ and $\mathcal{X}^{\mathbf{R}}$, which are the images correspondent to the SV, the misclassified and correctly classified images, respectively. Steps 1 to 5 define one training epoch. After step 5, the process restarts from step 1, so the latent representation is recomputed, as well as the matrices **P**, **K** and **D** and the large-margin discriminant is retrained on such an updated

latent representation, producing another set of SVs allowing recalculation of matrices **A**, **M** and **G**. Therefore, the latent representation of training samples in matrix **T** differs slightly from the previous epoch because of the updated FCN weights. After each epoch, the accuracy of the validation set is verified, and when improved, the models (FCN and discriminant) are kept. The

Algorithm 2.5 LMFCN - Loss matrices **M** and **G** - *obtain_anchor_matrices*()

Input: Sets $\mathcal{S}$, $\mathcal{Q}$, $\mathcal{R}$ with the indexes to the instances of interest to each loss type
Output: Matrices **M** and **G** with the indexes to the closest anchors to each instance of interest

for $i \in [0, n[$ **do**
  for $j \in [0, n[$ **do**
    // Use the auxiliary matrix **Z** to determine the closest anchors to each misclassified instance
    if $i \in \mathcal{Q}$ *and* $j \in \mathcal{S}$ *and* $i \neq j$ **then**
      $z_{ij} \leftarrow d_{ij}$
    else
      $z_{ij} \leftarrow \tau$
    end if
  end for
end for
for $i \in [0, |\mathcal{Q}|[$ **do**
  $\mathbf{m}_i \leftarrow f_{\text{argsort}}(\mathbf{z}_{q_i})$
end for
for $i \in [0, n[$ **do**
  for $i \in [0, n[$ **do**
    // Use the auxiliary matrix **H** to determine the closest anchors to each correct classified instance
    if $i \in \mathcal{R}$ *and* $j \in \mathcal{R}$ *and* $o^i \neq o^j$ **then**
      $h_{ij} \leftarrow d_{ij}$
    else
      $h_{ij} \leftarrow \tau$
    end if
  end for
end for
for $i \in [0, |\mathcal{R}|[$ **do**
  $\mathbf{g}_i \leftarrow f_{\text{argsort}}(\mathbf{h}_{r_i})$
end for

stop criterion is usually defined as a number of epochs. After all the epochs, we have the best models (FCN and SV) saved at the best validation accuracy.

2.2.3 Loss Function

The proposed loss function relies on the similarity between examples, and it has three terms, as shown in Equation (2.10). It aims at finding a latent representation that maximizes the margin ($\mathcal{L}_{\text{sv}}$) while pushing misclassified examples towards the right side of the decision boundary ($\mathcal{L}_{\text{mc}}$) and moving well-classified examples farther away from the decision boundary ($\mathcal{L}_{\text{cc}}$).

$$\text{minimize } \mathcal{L}_{\text{sv}} + \mathcal{L}_{\text{mc}} - \mathcal{L}_{\text{cc}} \tag{2.10}$$

$\mathcal{L}_{\text{sv}}$ calculates the sum of distances between SVs and their anchors using matrix $\mathbf{A}$, as shown in Equation (2.11). This loss is affected by the gradients from the SVs with respect to the values of the $\mathcal{A}_{sv}$ stored in $\mathbf{T}$. The backpropagation procedure updates the weights in a way that the latent representation generated by $f_{\text{fcn}}(.)$ has the smallest possible distance between SVs and $\mathcal{A}_{sv}$ at the current epoch. Therefore, the weights are updated to move the SVs and not the anchors.

$$\mathcal{L}_{\text{sv}} = \frac{1}{|\mathcal{S}|} \sum_{i=0}^{|\mathcal{S}|} \sum_{j=1}^{sv_{\text{close}}} ||f_{\text{fcn}}(\mathbf{X}^{s^i}) - \mathbf{t}_{a_{ij}}||_2^2 \tag{2.11}$$

where sv_{close} is the number of anchors to use for each SV, $f_{\text{fcn}}(.)$ is the FCN to be updated by the backpropagation, $\mathbf{X}^{s^i}$ is the matrix that represents the s^i image from $\mathcal{X}$, a_{ij} is an index pointing to an anchor instance in the input set $\mathcal{X}$, $\mathbf{t}_{a_{ij}}$ is the latent representation of such an anchor instance (or image) $\mathbf{X}^{a_{ij}}$, and $|\mathcal{S}|$ is the number of SVs.

The training algorithm computes the loss over $\mathbf{T}^{\mathcal{S}} \in \mathbf{T}$ as a single batch. It is also possible to use mini-batches, but considering small-size datasets and an FCN with compact architecture that yields a low-dimensional latent representation, this is unnecessary. In the next epoch, the updated weights will affect the generation of the latent representation of the entire training set $\mathcal{X}$.

Therefore, the latent representation of anchors also changes, and the training algorithm must choose a new set of anchors.

$\mathcal{L}_{\text{mc}}$ calculates the summation of distances between misclassified instances and their anchors using matrix $\mathbf{M}$, as shown in Equation (2.12). We also use all misclassified instances as a single batch, although there is no limitation to performing it in mini-batches. $\mathcal{L}_{\text{mc}}$ influences the training algorithm by updating the weights to minimize the distance between the misclassified instances and their closest SVs. Consequently, the misclassified instances are pushed toward the right side of the decision boundary.

$$\mathcal{L}_{\text{mc}} = \frac{1}{|Q|}\sum_{i=0}^{|Q|}\sum_{j=0}^{wr_{\text{close}}} ||f_{\text{fcn}}(\mathbf{X}^{q^i}) - \mathbf{t}_{m_{ij}}||_2^2 \tag{2.12}$$

where wr_{close} is the number of anchors for each misclassified instance, $f_{\text{fcn}}(.)$ is the FCN updated by the backpropagation, $\mathbf{X}^{q^i}$ is the matrix that represents the q^i image from $\mathcal{X}$, m_{ij} is an index pointing to an anchor instance in the input set $\mathcal{X}$, $\mathbf{t}_{m_{ij}}$ is the latent representation of an image $\mathbf{X}^{m_{ij}}$, and $|Q|$ is the number of misclassified instances.

When looking at only a single misclassified instance, the weight updating may not be enough to move such an instance to the right side of the decision boundary because its anchors are SVs right at the margin of the decision boundary. If it is not moved to the right margin side, the misclassified example may become an SV in the following training epoch.

$\mathcal{L}_{\text{cc}}$ represents the distance between well-classified instances and their anchors, and it is computed by Equation (2.13). Since we want to maximize such a distance, it is incorporated as a negative term in the loss function, which is minimized during training.

$$\mathcal{L}_{\text{cc}} = \frac{1}{|\mathcal{R}|}\sum_{i=0}^{|\mathcal{R}|}\sum_{j=0}^{sh_{\text{close}}} ||f_{\text{fcn}}(\mathbf{X}^{r^i}) - \mathbf{t}_{g_{ij}}||_2^2 \tag{2.13}$$

where sh_{close} is the number of anchors for each correctly classified instance, $f_{\text{fcn}}(.)$ is the FCN to be updated by the backpropagation, $\mathbf{X}^{r^i}$ is the matrix that represents the r^i image from $\mathcal{X}$, g^{ij} is an index pointing to an anchor instance in the input set $\mathcal{X}$, $\mathbf{t}_{g_{ij}}$ is the latent representation of an image $\mathbf{X}^{g_{ij}}$, and $|\mathcal{R}|$ is the number of correctly classified instances.

The number of anchors used for each training instance is controlled by sv_{close}, wr_{close}, and sh_{close} in Equations (2.11) (2.12), and (2.13), respectively. Algorithms 2.6, 2.7 and 2.8 present the loss calculations until the backpropagation. Usually, complex classification problems require a higher number of anchors. Furthermore, using all three terms of the proposed loss function in the training process may not always be necessary. $\mathcal{L}_{\text{sv}}$ alone already leads to good representations as such a loss is directly related to SVs and margin maximization. The computational cost for computing this term of the loss function is not high and decreases as the number of instances used by the backpropagation algorithm reduces at each training epoch because the number of SVs decreases as the latent representation improves. Computing $\mathcal{L}_{\text{mc}}$ is also not expensive because the number of misclassified examples tends to decrease over the training epochs. On the other hand, computing $\mathcal{L}_{\text{cc}}$ can become very expensive because the number of well-classified instances tends to increase over the training epochs. Therefore, the term $\mathcal{L}_{\text{cc}}$ should be used wisely, preferably only on challenging problems where the other two terms of the loss function may not lead to a low training error. The use of this loss term is also recommended at a certain interval of epochs, not at all of them, for example, in intervals of three epochs. The behavior and impact of each loss function are further discussed in Section 2.4 and their cost in Section 2.3.

2.2.4 SVM Loss Function

The SVM base equation (Equation (2.14)) with hard margins seeks the optimization of the best w and b to draw the largest linear margin that can separate all data correctly.

$$\begin{gathered}\min_{w,b} \frac{1}{2} \parallel w \parallel^2 \\ \text{subject to } o_i(w \cdot \mathbf{t}_i + b) - 1 \geq 0, i = 0...n\end{gathered} \tag{2.14}$$

Algorithm 2.6 LMFCN - $\mathcal{L}_{\text{sv}}$ and weights update - $loss\_sv()$

Input: Set $\mathcal{S}$ and matrix $\mathbf{A}$
Output: $\mathcal{L}_{\text{sv}}$

$\mathcal{L}_{\text{sv}} \leftarrow 0$
for $i \in [0, |\mathcal{S}|[$ **do**
  for $j \in [0, sv_{close}[$ **do**
    `// δ is the distance of a SV to its anchor`
    $\delta \leftarrow 0$
    for $\varphi \in [0, \phi[$ **do**
      $\delta \leftarrow \delta + (f_{\text{fcn}}(\mathbf{X}_{s_i})_\varphi - \mathbf{t}_{a_{ij}\varphi})^2$
    end for
    $\mathcal{L}_{\text{sv}} \leftarrow \mathcal{L}_{\text{sv}} + \sqrt{\delta}$
  end for
end for
return $\frac{\mathcal{L}_{\text{sv}}}{|\mathcal{S}|}$

Algorithm 2.7 LMFCN - $\mathcal{L}_{\text{mc}}$ and weight update - $loss\_mc()$

Input: Set $\mathcal{Q}$ and matrix $\mathbf{M}$
Output: $\mathcal{L}_{\text{mc}}$

$\mathcal{L}_{\text{mc}} \leftarrow 0$
for $i \in [0, |\mathcal{Q}|[$ **do**
  for $j \in [0, mc_{close}[$ **do**
    `// δ is the distance of a misclassified instances to its anchor anchors`
    $\delta \leftarrow 0$
    for $\varphi \in [0, \phi[$ **do**
      $\delta \leftarrow \delta + (f_{\text{fcn}}(\mathbf{X}_{q_i})_\varphi - \mathbf{t}_{m_{ij}\varphi})^2$
    end for
    $\mathcal{L}_{\text{mc}} \leftarrow \mathcal{L}_{\text{mc}} + \sqrt{\delta}$
  end for
end for
return $\frac{\mathcal{L}_{\text{mc}}}{|\mathcal{Q}|}$

where:

w is the margin's weight

Algorithm 2.8 LMFCN - $\mathcal{L}_{\text{cc}}$ and weight update - $loss\_cc()$

Input: Set $\mathcal{R}$ and matrix $\mathbf{G}$
Output: $\mathcal{L}_{\text{cc}}$

$\mathcal{L}_{\text{cc}} \leftarrow 0$
for $i \in [0, |\mathcal{R}|[$ **do**
  for $j \in [0, sh_{close}[$ **do**
    `// δ is the distance of a correctly classified instance to a anchor`
    $\delta \leftarrow 0$
    for $\varphi \in [0, \phi[$ **do**
      $\delta \leftarrow \delta + (f_{\text{fcn}}(\mathbf{X}_{r_i})_\varphi - \mathbf{t}_{g_{ij}\varphi})^2$
    end for
    $\mathcal{L}_{\text{cc}} \leftarrow \mathcal{L}_{\text{cc}} + \sqrt{\delta}$
  end for
end for
return $\frac{\mathcal{L}_{\text{cc}}}{|\mathcal{R}|}$

b is the margin's bias

$\mathbf{t}_i$ is the latent representation of the i-th training instance

o_i is the expected class output of instance x_i

n is the number of training instances

Given the minimization problem and constraints, it can be solved using Lagrangian multipliers and transforming in a Wolfe dual problem as denoted in Equation (2.15).

$$\begin{aligned} &\max_{\alpha} \sum_{i=0}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=0}^{n} \alpha_i \alpha_j o_i o_j k_{ij} \\ &\text{subject to } \alpha_i \geq 0, i = 0 \ldots n, \sum_{i=0}^{n} \alpha_i y_i = 0 \end{aligned} \tag{2.15}$$

where:

α are the Lagrangian multipliers

k_{ij} is the kernel matrix value for a i, j instance pair

After solving the Wolfe dual problem from Equation (2.15) with a method like sequential minimal optimization (SMO), the values of w and b are obtained by Equations (2.16) and (2.17) respectively.

$$w = \sum_{i=0}^{n} \alpha_i o_i \mathbf{t}_i \tag{2.16}$$

$$b = \frac{1}{s} \sum_{i=0}^{s} (o_i - w \cdot \mathbf{t}_i) \tag{2.17}$$

The SVM has a deterministic output and converges to an optimal value, always producing the same results even when executed multiple times with the same data. Although it only converges if there is an optimal solution, thus in a set of instances with outliers, for example, it cannot converge because it does not accept misclassified instances due to the constraints $(o_i(w \cdot \mathbf{t}_i + b) \geq 1)$.

To overcome the hard margin problem of SVM, the soft margin method adds slack variables ζ_i to the constraints (Equation (2.18)).

$$o_i(w \cdot \mathbf{t}_i + b) \geq 1 - \zeta_i, i = 1...n \tag{2.18}$$

The problem with using ζ_i is that, in principle, any value is accepted and will allow all samples to satisfy the constraints. To limit the influence of ζ, the C regularization parameter is added so that the optimization problem changes from Equation (2.14) to Equation (2.19).

$$\begin{gathered} \min_{w,b,\zeta} \frac{1}{2} \parallel w \parallel^2 + C \sum_{i=0}^{n} \zeta_i \\ \text{subject to } o_i(w \cdot x_i + b) \geq 1 - \zeta_i, \zeta_i \geq 0, i = 0...n \end{gathered} \tag{2.19}$$

So, the C parameter can control the weight of the influence of ζ_i in contrast to the first term. The optimization equation for the soft margin SVM in its Wolfe dual form is represented by Equation (2.20), changing the constraint $\alpha_i \geq 0$ to $0 \leq \alpha_i \leq C$.

$$\begin{aligned} &\max_{\alpha} \sum_{i=0}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=0}^{n} \alpha_i \alpha_j o_i o_j k_{ij} \\ &\text{subject to } 0 \leq \alpha_i \leq C, i = 0...n, \sum_{i=0}^{n} \alpha_i y_i = 0 \end{aligned} \tag{2.20}$$

The hinge loss, defined by Equation (2.21), helps models pursue the largest possible margins for the classification problems. It works by applying the largest penalties on instances that violate the margins. It also tends to zero in the case of instances that are as far as possible from the margin on the correct side.

$$\mathcal{L}_{hinge} = max(0, 1 - o_i y_i) \tag{2.21}$$

The hinge loss is used in the soft margin SVM training, applied in the regularization term as the ζ_i value. It provides controlled feedback from the training instances relative to the margin. The problem of choosing a value for ζ_i is solved using this resource. The C parameter now controls the effect of the misclassified instances, or the margin distance, calculated by the hinge loss in the model training. The SVM problem with hinge loss is then represented by Equation (2.22).

$$\min_{w,b} \frac{1}{2} \parallel w \parallel^2 + C \sum_{i=0}^{n} \max(0, 1 - o_i \cdot y_i) \tag{2.22}$$

The use of hinge loss in SVM is different from using it in a CNN. The SVM is a shallow approach that does not change the representation but finds the best margins defined by the SVs. The soft margin term, obtained by the hinge loss, allows some lenience related to misclassified instances, while the large margin is defined by the first term of the SVM equation (Equation (2.22)). This loss applied in a CNN model induces updates in the representation generated by the network

with the objective of increasing the margins as a whole (all instances). The impact of the hinge loss in the LMFCN is different, as it is indirectly used (inside the SVM) to tolerate misclassified instances and avoid overemphasizing outliers that would produce large updates on the weights of the CLs. The soft margin's lenience greatly impacts the first steps of the LMFCN because, initially, the latent representation does not allow clear separation. Without the hinge loss, and consequently the soft margin SVM, the discriminant would not converge and would forbid the use of SVs and predicted output from the discriminant in the losses.

2.2.5 FCN Architecture

The FCN is a sequence of CLs similar to those used in CNNs, used as filter banks to learn representation related to textures, as presented in Table 2.1. Pooling layers follow these layers to reduce the input size progressively. The deeper the layers, the narrower the latent representation, but with an increasing number of channels, which allows more filters combination, increasing the richness of the representation.

After the last CL, a GAP layer builds a latent representation where each element represents the activation of a sequence of filters to the input characteristics. The GAP layer also makes the output dimension independent of the input size, and the latent representation will always have the same number of channels at the end of the FCN. Such a latent representation feeds a large-margin discriminant to learn classification tasks. We compared our approach with two CNNs with identical architecture (Table 2.1 and Figure 2.5) but with a sequence of fully connected (FC) layers as discriminant after the GAP layer. We employed binary cross-entropy (BCE) loss and hinge loss for binary problems and cross-entropy loss for multiclass problems.

The basic layers of the FCN architecture are CLs, MaxPooling, and ReLU. The current deep learning scenario introduced new layers and blocks, e.g., inception, residual, squeeze-and-excitation blocks, and attention mechanisms. Although we have not employed such new layers and blocks, our approach does not prohibit their use. The purpose of choosing simple layers was to show that our approach can obtain high performance in the chosen context even with basic

layers, avoiding extra complexity and computational costs. In our case, we are not exploring new blocks or layers' capabilities but focusing on the training procedure. Complex mechanisms would cause an extra variable to the analysis of our approach.

In Table 2.1, we use w, h, and c to represent the input image's width, height, and number of channels. We used this generic representation to highlight that our FCN is not restricted to an input size like other CNNs without GAP, where the activation maps depend on the dataset image size. Our kernel sizes, the number of layers, and strides were defined empirically after a set of experiments. We started using an FCN similar to AlexNet, and TCNN (Andrearczyk & Whelan, 2016) and tried modifying these parameters until we obtained the average best result in all datasets. The parameter ϕ is the size of the latent representation. In our experiments, we have started using ϕ equals two to allow the representation in a low dimensional space (dispersion graphs). The value of ϕ=16 used in CNNs and LMFCN comparison was also defined based on the best accuracy with the smallest latent representation. Therefore increasing it does not produce a significant accuracy improvement. This parameter also depends on the dataset's complexity. If it is necessary to represent more image details and different textures, a latent representation of higher dimensionality (large ϕ) is required as it tends to improve the representation capability. ϕ must also be chosen carefully in multiclass problems due to the latent representation concatenation in our OVA method (Section 2.2.7).

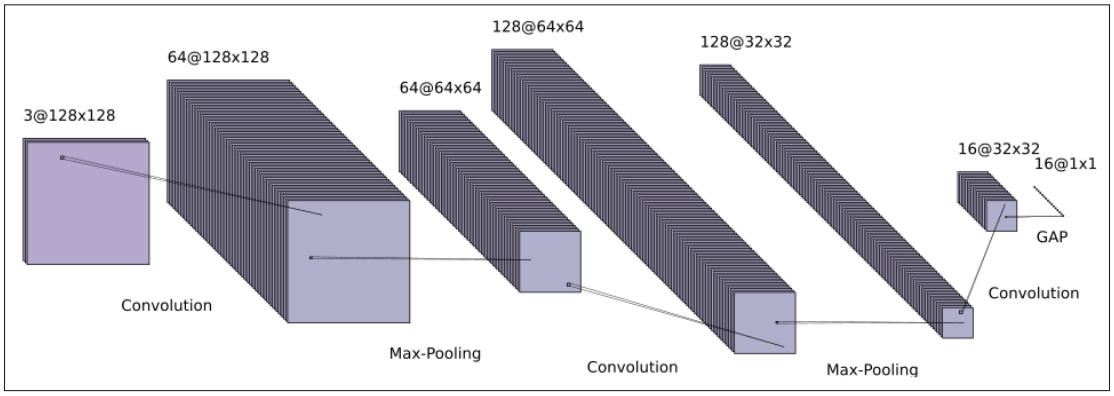


Figure 2.5 FCN Architecture with c=3, w=128, h=128, and ϕ=16

Table 2.1 FCN used on the LMFCN and in the CNN comparison. (w: image width, h: image height, c: number of channels, ϕ: size of the latent representation

Layer	**Input**	**Output**
Convolutional Layer	$w \times h \times c$	$w \times h \times 64$
Batch Normalization		
ReLU		
Max Pooling	$w \times h \times 64$	$w/2 \times h/2 \times 64$
Convolutional Layer	$w/2 \times h/2 \times 64$	$w/2 \times h/2 \times 128$
Batch Normalization		
ReLU		
Max Pooling	$w/2 \times h/2 \times 128$	$w/4 \times h/4 \times 128$
Convolutional Layer	$w/4 \times h/4 \times 128$	$w/4 \times h/4 \times \phi$
Batch Normalization		
ReLU		
Global Average Pooling	$w/4 \times h/4 \times \phi$	$1 \times 1 \times \phi$

2.2.6 Large-Margin Discriminant

The large-margin discriminant of the LMFCN is a support vector machine (SVM) with an RBF kernel, which Gram matrix $\mathbf{K}$ is obtained by Equation (2.2) calculated jointly with the distance matrix $\mathbf{D}$. The SVM training can be split into two phases, building the kernel and the optimization of w and b. Thus, part of the SVM training is performed in GPU, based on the pre-built matrix $\mathbf{P}$, also used to compute $\mathbf{D}$, accelerating this stage. We chose the pre-computed RBF kernel due to its ease of computation using the GPU resources and space separation capacity. Although a linear kernel is less computationally costly than an RBF kernel, learning a discriminant representation would require more training epochs of the FCN weights. In the preliminary studies, we compared the two kernel approaches and verified the advantage of the RBF.

2.2.7 Multiclass LMFCN

A large-margin discriminant is inherently binary, and to deal with multiclass problems, we use the one-vs-all (OVA) approach, which reduces multiclass problems into n_{classes} multiple binary classification problems corresponding to the number of classes. In the training stage, all instances are provided to n_{classes} LMFCN pairs (FCN+SVM). Our method trains the FCNs to learn a representation to discriminate a single class against all others, so we build multiple binary feature extractors and discriminants.

The multiclass LMFCN only uses the multiple binary discriminants to provide information about the representation space generated by the FCNs, i.e., the SVs, well, and misclassified instances. After training all the n_{classes} FCNs, we discard all SVMs, and a new discriminant is trained using a latent representation with $\phi \times n_{\text{classes}}$ attributes as shown in Figure 2.6. At the end of the training process, the full model comprises n_{classes} FCNs with an output of ϕ attributes and a multiclass SVM with an input of $\phi \times n_{\text{classes}}$ attributes. The multiclass SVM internally holds multiple binary SVMs on OVA configuration using RBF kernel. Although this approach seems similar to using the multiple SVMs trained with the FCNs, all new classifiers have access to the latent representations of all binary classifiers.

2.3 Computational Cost

The computational training cost analysis has to be split into three parts to compare the LMFCN with equivalent CNNs. Both approaches have similar backpropagation complexity until the GAP if we assume similar FCN architectures. It is proportional to the number of input images n, their size $w \times h \times c$ and the number of weights α_{fcn}. The LMFCN does not have FC layers as in CNNs, which have a high number of parameters due to the high connectivity of units, so there is a reduction in the computational cost of α_{fc} against n in the LMFCN. Despite lacking FC layers, the LMFCN replaces them with an SVM classifier, which implies in computational cost for kernel calculation (n^2) and solving the quadratic programming problem by an SMO algorithm, which requires (n^3 or $n \times n^2$) in the worst case. Considering a small-size dataset

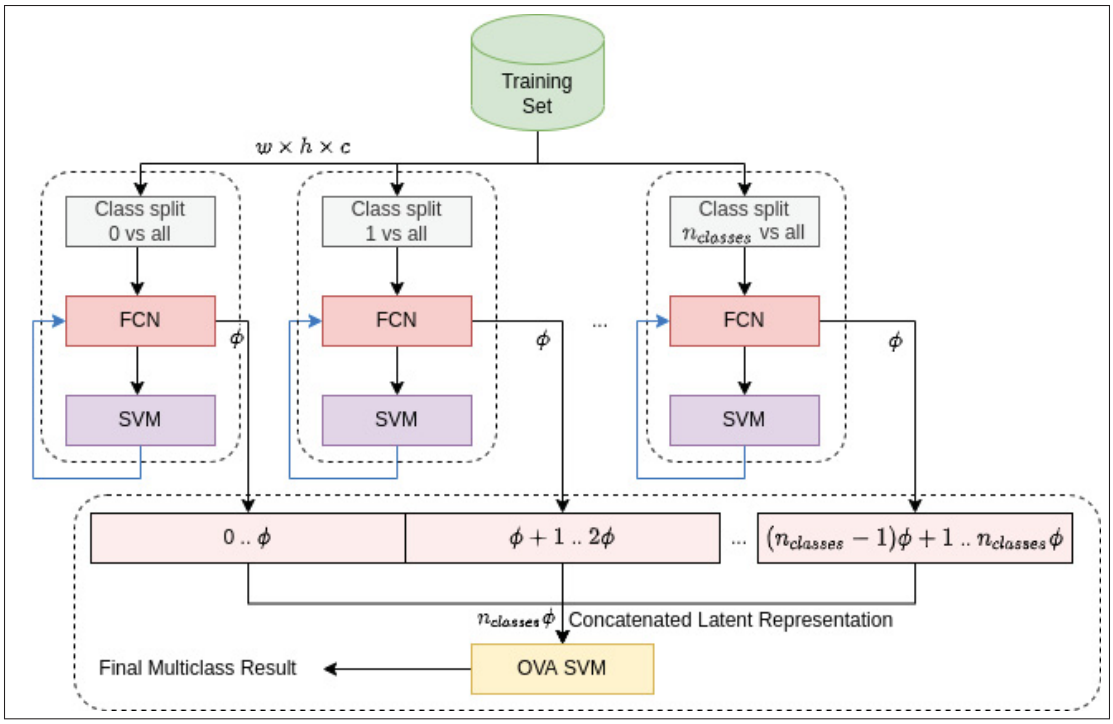


Figure 2.6 Multiclass LMFCN. Blue arrows mean part of the OVA filter training, including backpropagation with loss functions. Black arrows represent the data flow of the multiclass SVM classifier training with concatenated latent representations of all OVA subproblems

(small n), even the SMO cubic cost is lower than $n \times \alpha_{\text{fc}}$. Therefore, the $n^2 = \alpha_{\text{fc}}$ is the upper bound of the LMFCN. For higher values of n, the advantages of using the LMFCN fade away, and conventional CNNs should perform equally or better due to the training data availability.

One of the advantages of the LMFCN is the possibility of computing the loss function only for SVs instead using all images from the training set (n) when using only $\mathcal{L}_{\text{sv}}$, which reduces this phase cost from n to $|\mathcal{S}|$. The FCN, used on both LMFCN and CNNs, has more parameters than the FC layers, so using fewer training samples has a significantly positive performance impact on the weights update of the first layers. Furthermore, as we observed in our experiments, the LMFCN converges faster than equivalent CNNs, requiring fewer epochs to achieve a low training error. With this characteristic, the LMFCN training is even more efficient, and the considered upper bound is then $n_{\text{sv}}^2 > \alpha_{\text{fc}}$ instead of $n^2 > \alpha_{\text{fc}}$.

In multiclass problems, it is necessary to use multiple FCNs and SVs, according to the number of classes, due to the OVA approach. At the end of the LMFCN OVA training, there is a multiclass SVM with a more comprehensive latent representation ($\phi \times n_{\text{classes}}$). We compared the LMFCN with a CNN of equivalent size in terms of the number of parameters, number, and the dimension of layers, increasing both the convolutional and discriminant parts to expand this network representation capacity. With more parameters and the high connectivity of the FC layers, the CNN backpropagation costs also increased. Additionally, the higher the number of parameters, the more training data is needed, the higher the cost of each sample through the network, and the convergence for more parameters takes more epochs. However, in the LMFCN, we used multiple small FCNs that can converge faster with high-accuracy results.

Shallow approaches have a reduced total cost because algorithms for handcrafted feature extraction process images into a single pass, and the classifiers are trained once with the extracted features. Despite their computational cost advantage, they produce generic characteristics not adapted for a particular classifier, resulting in worse accuracy values than the LMFCN. Multiclass shallow approaches are also much less expensive than the LMFCN multiclass approach. Still, they always produce the same representation (features) independent of the number of classes, not exploiting the intra-class and between-class properties.

2.4 Analysis of Loss Terms

The three terms of our loss function play a specific role in the training process. To demonstrate the influence of each loss term, we have generated synthetic distributions to simulate the output of an FCN ($f_{\text{fcn}}(.)$). Therefore, we simulate the representation learning mechanism using our loss function with the backpropagation algorithm, which aims to reduce the distances between instances of interest and their anchors (loss terms $\mathcal{L}_{\text{sv}}$ and $\mathcal{L}_{\text{mc}}$ using Equation (2.23)), and also maximizing the distances between instances of interest and their anchors (loss term $\mathcal{A}_{\text{cc}}$ using Equation (2.25)). The update of the latent representation for $\mathcal{L}_{\text{sv}}$ and $\mathcal{L}_{\text{mc}}$ is based on a convex combination between instances of interest and anchors, prioritizing (heavier weights) the closest anchors. Our simulation also updates generic instances close to the instances of interest with

movements inversely proportional to their distance as in Equation (2.24). Both Equations (2.23) and (2.24) can be used for simulating the behavior of loss terms ($\mathcal{L}_{\text{sv}}$ and $\mathcal{L}_{\text{mc}}$), replacing sv_{close} by wr_{close} and $\mathcal{S}$ by $\mathcal{Q}$.

The synthetic distributions are defined as $\mathbf{T} = \{\mathbf{t}^n \| \mathbf{t} \in \mathbb{R}^2\}$, where n is the number of instances. As a parallel to our approach, the ϕ of the synthetic data for the simulation is $\mathbb{R}^2$ to create correspondence to the 2-D Euclidean space, making it easier to visually interpret the algorithm's effect as it works by reducing (or augmenting) the Euclidean distances of instances to their anchors.

$$\mathbf{\Delta t}_i = \sum_{j=0}^{sv_{\text{close}}+1} \psi_{j+1} \mathbf{t}_{a_{ij}} \qquad \text{, with } \{i | i \in \mathbb{N} \cap i \in \mathcal{S}\} \tag{2.23}$$

where:

$\mathbf{t}_i$ are the coordinates of an i-th SV or a misclassified instance

ψ is the weight of the convex combination with $\sum_{i=0}^{sv_{\text{close}}+1} \psi_i = 1$ and $\sum_{i=1}^{sv_{\text{close}}+1} \psi_i < \psi_0$

$\mathbf{t}_{a_{ij}}$ are the $j-th$ anchor coordinates of the $i-th$ SV or misclassified instance

$$\mathbf{\Delta t}_i = \sum_{j=0}^{|\mathcal{S}|} \left(1 - \frac{1}{d_{i\mathcal{S}^j}}\right) \Delta \mathbf{t}_j \qquad \text{, with } \{i | i \in \mathbb{N} \cap i \notin \mathcal{S}\} \tag{2.24}$$

where:

$\mathbf{t}_i$ are the coordinates of a synthetic instance to update

d_{is^j} is the distance between instance i and $\mathcal{S}^j$

$\Delta \mathbf{t}_j$ is the Δ applied to each SV.

The update of correctly classified instances position, like the update of the filter parameters of the FCN ($f_{\text{fcn}}(.)$) during the backpropagation, is accomplished by adding an update vector to their coordinates. Such a vector is the weighted average vector between a correctly classified

instance and its anchors. The farthest the anchor, the smaller the weight on average. Equation (2.25) expresses the calculation of the update of a correct classified instance.

$$\mathbf{\Delta t}_i = \sum_{j=0}^{sh_{\text{close}}} \Gamma(\mathbf{t}_i - \mathbf{t}_{g_{ij}}) \left(\frac{1}{j+1} \right) \qquad \text{, with } \{i | i \in \mathbb{N} \cap i \in \mathcal{R}\} \tag{2.25}$$

where:

$\mathbf{t}_i$ are the coordinates of the i-th instance in $\mathcal{R}$

g_{ij} is index to the j-th closest instance from the i-th instance

$\mathbf{t}_{g_{ij}}$ are the coordinates of the j-th anchor of the i-th instance

Γ is the learning rate

$$\mathbf{\Delta t}_i = \sum_{j=0}^{|\mathcal{R}|} \left(1 - \frac{1}{d_{ir^j}} \right) \mathbf{\Delta t}_{r^j} \qquad \text{, with } \{i | i \in \mathbb{N} \cap i \notin \mathcal{R}\} \tag{2.26}$$

where:

$\mathbf{t}_i$ are the coordinates of a i-th instance

d_{ir^j} is the distance between instance i and r^j

$\mathbf{\Delta t}_{r^j}$ is the Δ applied to a correct classified instance in $\mathcal{R}$

The update of other than the misclassified instances is calculated by Equation (2.26). It adds the weighted average of the update vectors of well-classified neighbors to other instances. The weight is based on the distance between them.

The first synthetic distribution we used is depicted in Figure 2.7a. It was generated by two Gaussian distributions, one at the center and the other at the sides of the representation space. There are also two outliers on the right side.

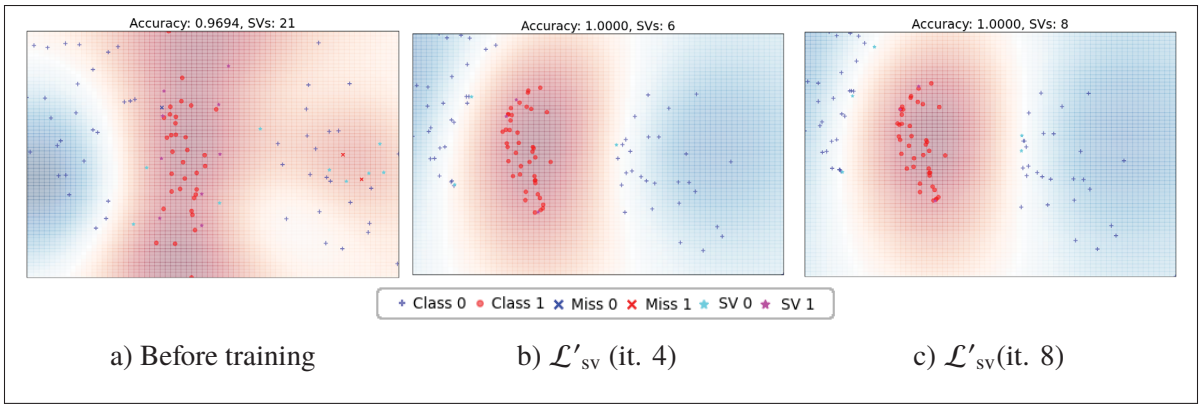


Figure 2.7 Displacement of (a) instances of a two-class synthetic distribution shifted with $\mathcal{L}'_{sv}$ after b) four and c) eight iterations.

Figure 2.7 represents the $\mathcal{L}_{sv}$ simulation, denoted by $\mathcal{L}'_{sv}$, shifting the SVs towards their anchors. After eight iterations, there is a separation in the instances similar to iteration four. It creates a considerable inter-class margin and corrects the misclassified samples preserving the opposite side margin. The large margin in the training set also impacts the test dataset (unseen instances) because it improves the class separation. The outliers do not directly affect the optimization process because the SVs are the object of interest. The relaxation or soft margin of the SVM classifier allows some instances not to become SVs and limits the overfitting, limiting the effect of the outliers.

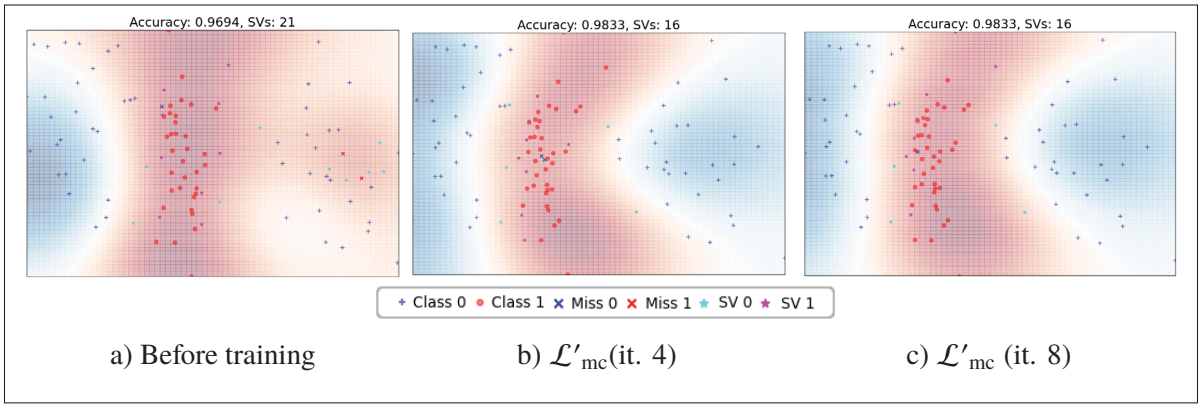
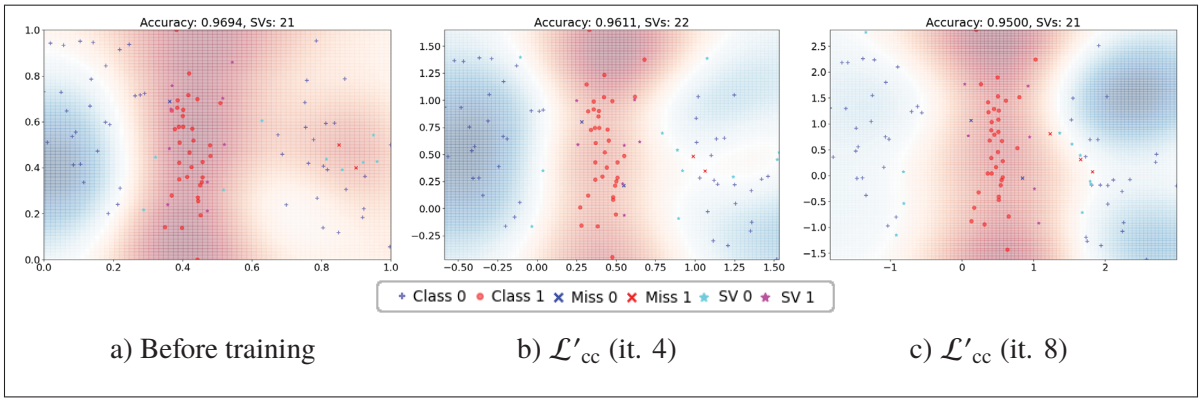


Figure 2.8 Displacement of (a) instances of a two-class synthetic distribution shifted with $\mathcal{L}_{mc}$ after b) four and c) eight iterations.

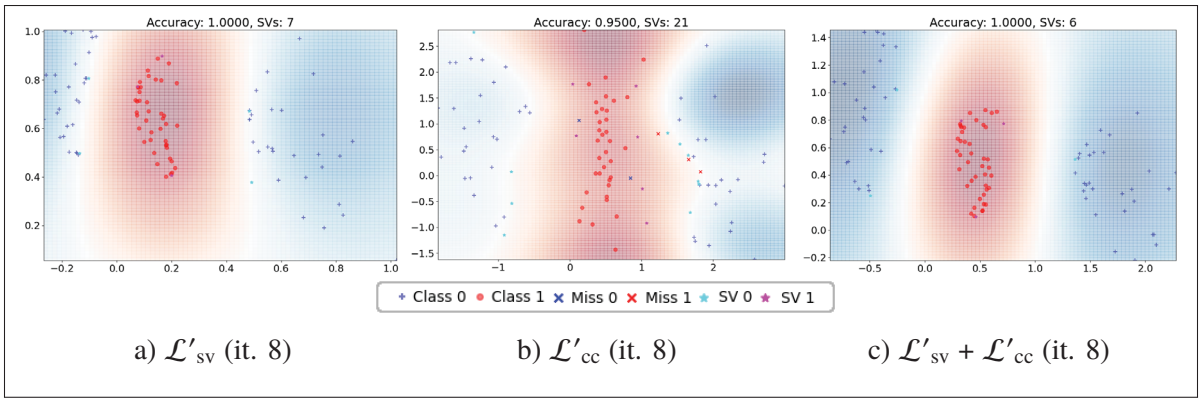
The $\mathcal{L}'_{\text{mc}}$ is computationally lighter than the $\mathcal{L}'_{\text{sv}}$ because the first has fewer instances of interest. The margins produced by $\mathcal{L}'_{\text{sv}}$ and $\mathcal{L}'_{\text{mc}}$ losses are similar, so the last one is more interesting due to its lower cost and good results. The update to correct the (red class) outliers on the right side of the space representation also shifts the (red) instances in the center to the left, seen in Figure 2.8. The $\mathcal{L}'_{\text{mc}}$ also stops the optimization process after the training accuracy achieves 100% because there will be no instance to calculate an update. On the other hand, $\mathcal{L}'_{\text{sv}}$ sustains the optimization process, even without misclassified instances, because there will always be SVs to guide the process and increase the margin. In Figure 2.8c, the left side margins are narrower than in Figure 2.7c. The space of blue instances at the right side also infiltrates the space of red instances more than $\mathcal{L}'_{\text{sv}}$, so in this simple scenario, we see the effects of outliers comparing the two first loss terms.



a) Before training b) $\mathcal{L}'_{\text{cc}}$ (it. 4) c) $\mathcal{L}'_{\text{cc}}$ (it. 8)

Figure 2.9 Displacement of (a) instances of a two-class synthetic distribution shifted with $\mathcal{L}_{\text{cc}}$ after b) four and c) eight iterations.

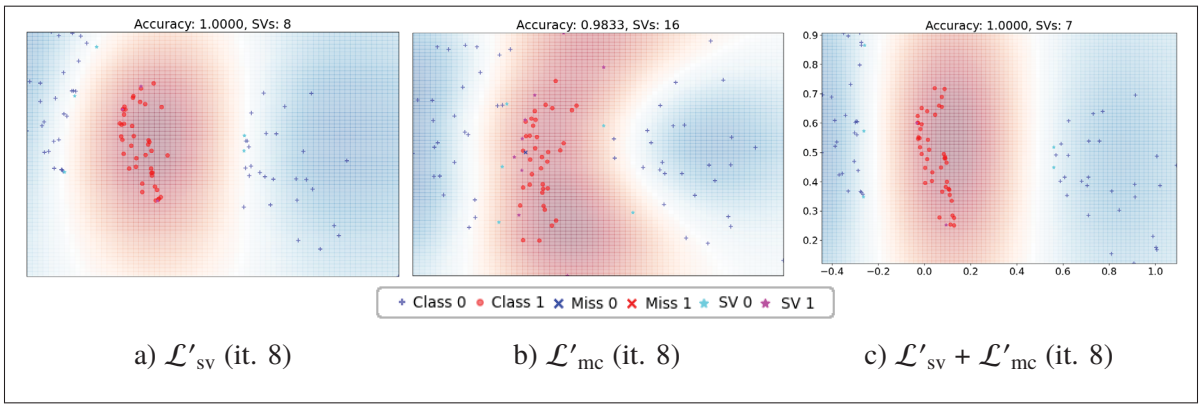
Figure 2.9 shows the simulation of the third term of the loss function. In this figure, the axis x and y have the ticks numbering the axes, contrasting Figures 2.7 and 2.8. $\mathcal{L}'_{\text{cc}}$ can shift instances, but using correctly classified instances, it does not focus on correcting the outliers or other misclassified instances. The focus is on inter-class distance so we can see the red instances in the middle more squeezed by the two clouds of blue instances at each side. Blue samples are also pushed away from the red ones, so the loss induces the spreading of samples across the representation space, as can be seen comparing the scale of axes in Figures 2.9a and 2.9c.



a) $\mathcal{L}'_{sv}$ (it. 8) b) $\mathcal{L}'_{cc}$ (it. 8) c) $\mathcal{L}'_{sv} + \mathcal{L}'_{cc}$ (it. 8)

Figure 2.10 Displacement of instances of a two-class synthetic distribution shifted with a) $\mathcal{L}'_{sv}$ b) $\mathcal{L}'_{cc}$ and c) $\mathcal{L}'_{sv}$ plus $\mathcal{L}'_{cc}$ after eight iterations.

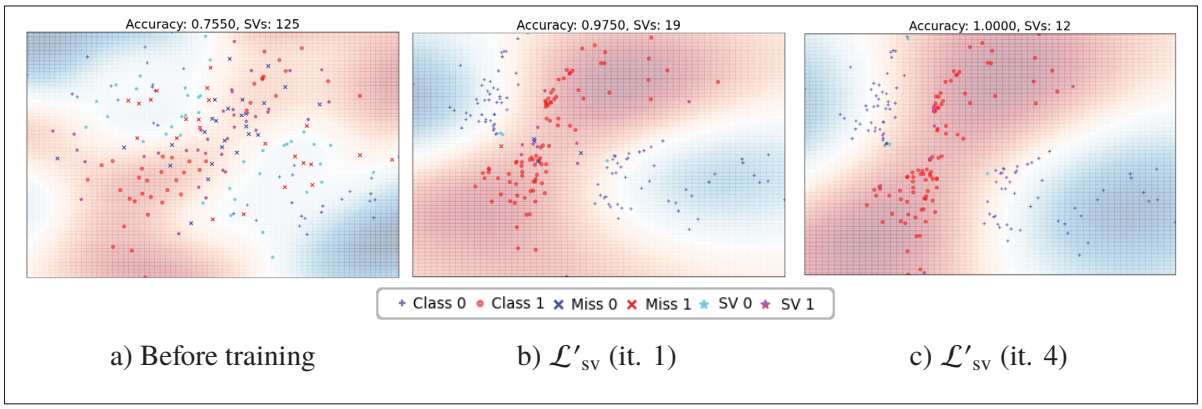
Each loss term has a specific way of guiding the training of the $f_{fcn}(.)$, and they can be used together to combine their behavior. Figure 2.10 presents a comparison of a simulation using $\mathcal{L}'_{sv}$ and $\mathcal{L}'_{cc}$, and the combination of both. The difference in scale at the eighth iteration is perceptible. $\mathcal{L}'_{sv}$ keeps samples compacted with a well-defined margin, while $\mathcal{L}'_{cc}$ spreads the instances but does not separate them nicely. The representation in Figure 2.10c has better-defined margins than the ones using only $\mathcal{L}'_{sv}$ due to the inter-class distance maximization of $\mathcal{L}'_{cc}$. $\mathcal{L}'_{sv}$ does not allow $\mathcal{L}'_{cc}$ to increase the intra-class distance like in Figure 2.9c. It is also important to mention that $\mathcal{L}'_{cc}$ is only applied at a three-epoch interval; otherwise, it would conflict with $\mathcal{L}'_{sv}$ and generate high computational cost in an actual situation.



a) $\mathcal{L}'_{sv}$ (it. 8) b) $\mathcal{L}'_{mc}$ (it. 8) c) $\mathcal{L}'_{sv} + \mathcal{L}'_{mc}$ (it. 8)

Figure 2.11 Displacement of instances of a two-class synthetic distribution shifted with a) $\mathcal{L}'_{sv}$ b) $\mathcal{L}'_{mc}$ and c) $\mathcal{L}'_{sv} + \mathcal{L}'_{mc}$ after eight iterations.

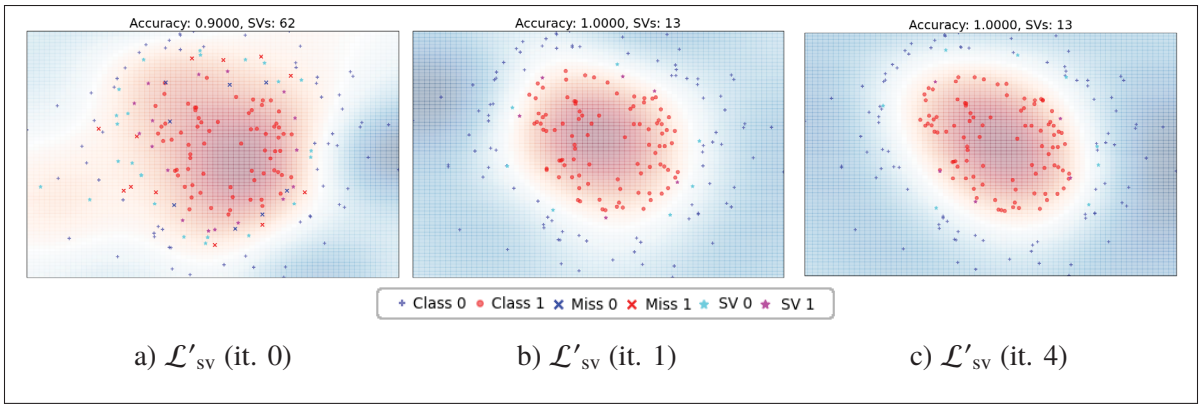
The combination of $\mathcal{L}'_{\text{sv}}$ and $\mathcal{L}'_{\text{mc}}$ is presented in Figure 2.11, where it obtained a better margin compared to the margins achieved by the two loss terms separately. The red samples are compacted, and the blue instances are not infiltrated on the right side of the space. Both loss terms are applied inside each epoch because their objectives do not differ, and the computational cost is not high.



a) Before training b) $\mathcal{L}'_{\text{sv}}$ (it. 1) c) $\mathcal{L}'_{\text{sv}}$ (it. 4)

Figure 2.12 Displacement of (a) instances of a two-class synthetic four blobs distribution shifted with $\mathcal{L}'_{\text{sv}}$ after b) one and c) four iterations

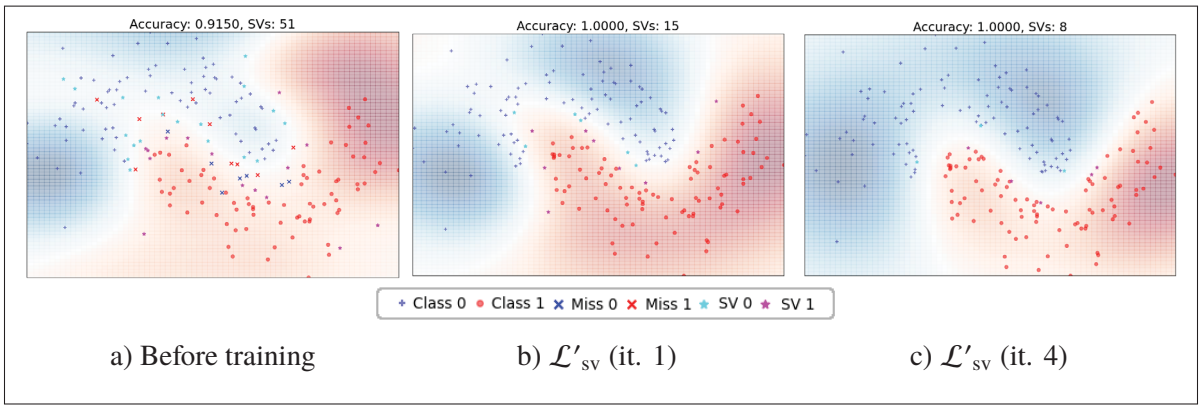
The synthetic samples generated from three Gaussian distributions shown in Figures 2.7 to 2.11 are relatively easy to optimize. We have also generated samples from other more challenging synthetic distributions to evaluate the proposed loss function in more complex representation spaces. Figure 2.12 shows the simulation on samples distributed in four alternate blobs on the corners of the space representation. In epoch 0 (no shift), there are class interleaving in the middle of the latent representation graph with misclassified instances. In epoch 1 (Figure 2.12b), there is enough motion to shift blue instances to the corners and allow the separation of classes. The correct movement was possible because the SVs in the class's limit were pushed to the corners, not the middle. The instances in the middle of the representation space are either misclassified or SVs and cannot serve as anchors of the instances of interest. Thus, $\mathcal{L}'_{\text{sv}}$ is enough to induce class separation.



a) $\mathcal{L}'_{\text{sv}}$ (it. 0) b) $\mathcal{L}'_{\text{sv}}$ (it. 1) c) $\mathcal{L}'_{\text{sv}}$ (it. 4)

Figure 2.13 Displacement of (a) instances of a two-class synthetic concentric rings distribution shifted with $\mathcal{L}'_{\text{sv}}$ after b) one and c) four iterations

The distribution shown in Figure 2.13 is shaped as two concentric rings, one of each class. Again, the $\mathcal{L}'_{\text{sv}}$ was enough to expand the margin, compacting the red samples and moving the outer ring away from the center.



a) Before training b) $\mathcal{L}'_{\text{sv}}$ (it. 1) c) $\mathcal{L}'_{\text{sv}}$ (it. 4)

Figure 2.14 Displacement of (a) instances of a two-class synthetic interleaved moons distribution shifted with $\mathcal{L}'_{\text{sv}}$ after b) one and c) four iterations

The interleaved moons distribution shown in Figure 2.14 evaluates the behavior of the $\mathcal{L}'_{\text{sv}}$ with a scenario that requires shifting in diverse directions. There are also misclassified samples across the class boundaries. In the fourth epoch, the simulation with $\mathcal{L}'_{\text{sv}}$ was able to enlarge the margin and drastically reduce the number of SVs, as shown in Figure 2.14c.

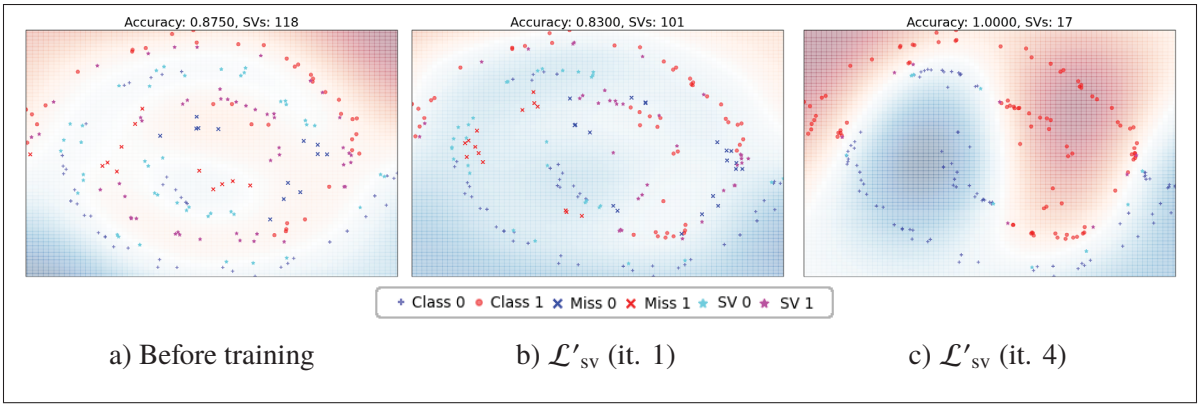


Figure 2.15 Displacement of (a) instances of a two-class synthetic spiral distribution shifted with $\mathcal{L}'_{sv}$ after b) one and c) four iterations

Finally, we used a challenging distribution with two classes interleaved in a spiral shape, as shown in Figure 2.15. The complexity is reflected by the number of SVs and low balanced accuracy in epoch zero compared to the other simulations. The higher the number of SVs, the more updates are performed on the data representation, so after the first update, there is a drastic shift in the instances. The movement increases the grouping of samples, but it is not enough to improve the class separation (Figure 2.15b). In the fourth epoch (Figure 2.15c), margins are already defined, the number of SVs is reduced, and the classifier achieves 100% accuracy.

The complexity of the spiral distribution led us to use it to simulate the combination of the loss terms. Figure 2.16 shows the comparison of the combination after nine epochs. There is a subtle difference between Figures 2.16a and 2.16b because the two loss terms ($\mathcal{L}'_{sv}$ and $\mathcal{L}'_{sv}$+$\mathcal{L}'_{mc}$) are based on distance reduction and misclassified instances are close to the SVs, but still, there is an improvement in the margins and $\mathcal{L}'_{sv}$ and $\mathcal{L}'_{mc}$ combination has one less SV than $\mathcal{L}'_{sv}$ alone. On the other hand, combining loss terms $\mathcal{L}'_{sv}$ and $\mathcal{L}'_{cc}$ (Figure 2.16c) produces a different representation, where it is possible to note the upper left corner blue instances pushing the red ones around them. The blue samples in the middle are also far from the red ones compared to $\mathcal{L}'_{sv}$+$\mathcal{L}'_{mc}$.

Although the simulations presented in this section do not implement the real forward and backward passes, gradient computation and weight updating performed by the backpropagation

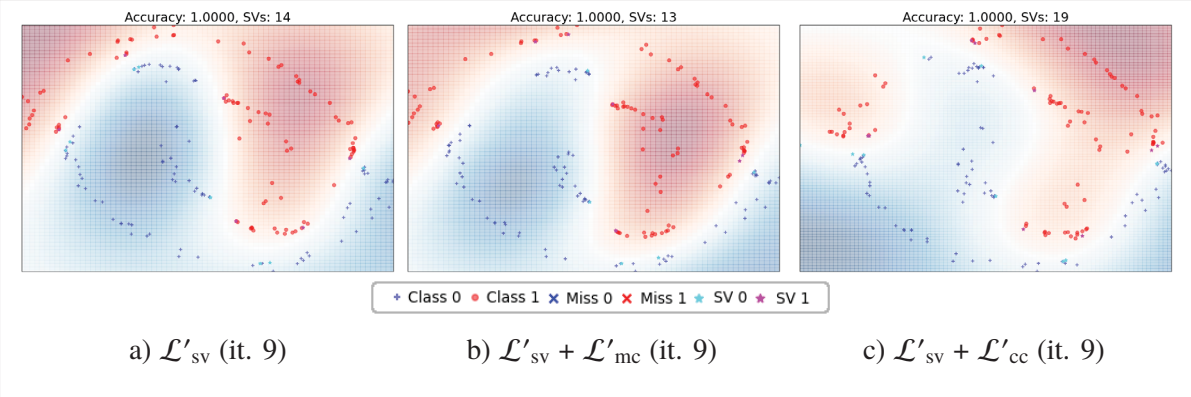


Figure 2.16 Displacement of instances of a two-class synthetic spiral distribution shifted with a) $\mathcal{L}_{sv}$ b) $\mathcal{L}'_{sv}$ + $\mathcal{L}'_{mc}$ terms and c) $\mathcal{L}'_{sv}$ + $\mathcal{L}'_{cc}$ after nine iterations.

algorithm used to train the FCN ($f_{fcn}(.)$), they are a good approximation of its behavior. It is expected that the weights update during the backpropagation would affect the instances that are more similar or close in the latent representation space, no matter the class they belong to. Thus, we moved the instances of interest with respect to their anchors and applied a proportional shift in a similar direction to the closest instances of interest, no matter their classes. We intended to visualize the behavior of the loss terms in typical data distributions.

Our simulations showed that the proposed approach has the potential to change the latent representation to expand the margins and ease the work of the large-margin discriminant. The direction of the shifts produced by the algorithm guided by the loss function terms is consistent in different synthetic distributions. We used simple distributions, like three and four blobs, and more complex ones, like the spirals and interleaved moons. In all the synthetic distributions, we chose average and standard deviation to generate some misclassification and complex margins. The simulations showed that our anchor and instance selection and loss function works well to increase the margins together with the loss function. In the next chapter, we evaluate the proposed approach on real data to confirm the behavior found in the synthetic data distributions with the algorithm simulation.

CHAPTER 3

EXPERIMENTS

This chapter describes the datasets used in our experiments, the metrics, and the results. Our experiments aim to evaluate the LMFCN's capability of efficiently classifying texture images of small-size datasets. We compared it with other methods with similar size and complexity to show that the LMFCN can obtain equivalent or superior accuracy results with the capability of adaptation and low computational cost. The experiments used five datasets: three HI, one textural, and another synthetic dataset. Although only one is texture specific, they all have texture characteristics without object definition, like ImageNet. Image sizes from these datasets ranged from 128×128 to 512×384, from two to eight classes. The smallest one has 400, and the largest one has 2081 images.

We chose as the primary metric the balanced accuracy to compare the methods directly. In addition, we used complexity measures to compare the latent representation or features generated by each method and understand the impact of such features on the accuracy of the discriminants. We also used UMAPs to render a graphic view of each method's latent representations and compare them. Additionally, we analyzed the number of convergence epochs, the number of SVs over the training (in the case of LMFCN), and the time to execute some steps of the compared methods.

The experiments with the real datasets used five stratified hold-outs with a split of 55/30/15% for training, test, and validation sets, respectively. We chose this split as an in-between all the splits from the works that used such datasets. In the experiments with the synthetic dataset, we used 200 images for each set (training, test, and validation), with 100 images of each class, and repeated the experiment five times. We used the same hold-outs for all methods and the same random seed for each equivalent comparison experiment with the same hold-out in all experiments.

We have compared LMFCN with other equivalent methods to analyze whether our method effectively classifies small-size datasets with texture characteristics. The comparison includes CNNs similar to TCNNs presented by Andrearczyk & Whelan (2016), handcrafted feature extractors, pre-trained CNNs as feature extractors, and pre-trained CNNs with fine-tuning. We have chosen a CNN with a small latent representation to create a direct visual comparison with the LMFCN.

3.1 Experimental Protocol

We have carried out four types of experiments to assess the performance of the LMFCN and to compare such a performance with other state-of-the-art approaches. The first one (Section 3.4.1) evaluates the behavior of our method using a low-dimensional latent representation capable of being directly depicted in scatter plots. Subsection 3.4.2 compares the LMFCN and CNNs trained with two different loss functions: binary cross-entropy and hinge loss. Due to the 16-wide latent representation used in the comparisons, the analysis employs complexity measures and accuracy metrics. Shallow methods employing feature extraction such as LBP (Ojala, Pietikainen & Maenpaa, 2002), GLCM (Haralick, Shanmugam & Dinstein, 1973) and PFTAS (Coelho *et al.*, 2010) are compared to the LMFCN in Subsection 3.4.3. We also used complexity measures and dispersion graphs with the dimensionality reduction performed by UMAPs in these last comparisons. As a method between CNNs and handcrafted feature extractors, we evaluated the performance of two pre-trained CNNs, ResNet and Inception, in the context of transfer learning. Finally, the last comparison employs pre-trained networks, but instead of using them only to extract features, we added extra FC layers to adapt their output to the binary problems, training them (fine-tuning) for some epochs.

Our approach relies on a large-margin binary classifier, so there is no direct implementation of multiclass classification. Therefore, to perform this type of classification, we used the OVA approach where the latent representation generated by each FCN is concatenated to make up a single latent vector as described in Section 2.2.7. The CNN used the cross-entropy loss that allows direct multiclass training. Handcrafted feature extractors are not sensitive to the number

of classes. Still, the SVM classifier that uses their features needs a multiclass approach, so we also opted to use the OVA approach. The principal evaluation metric of the multiclass comparisons in Section 3.4.5 is the accuracy and balanced accuracy. We also presented the accuracy for each subproblem in the OVA approach of the LMFCN to highlight its performance in imbalanced scenarios.

3.2 Datasets

The dataset selection for our experiments considers the LMFCN target problems, which means small-size datasets with a focus/particular interest on texture datasets. In the first analysis of our approach, we used a synthetic dataset with images generated with two Gaussian distributions with striped patterns. Another dataset we used was the Salzburg Texture Image Database (STex), which has texture images of 32 classes, from which we used only the two most prevalent classes. Additionally, we used three HI datasets, two based on mammary tissue and one on colorectal tissue. The synthetic datasets aimed to provide a controlled way of testing the LMFCN capabilities, avoiding extra variables intrinsic to real datasets. In this dataset, we could configure the image size, the difference between classes, levels of noise, and the number of images, allowing us to test the limits of the LMFCN. The STex dataset allowed an evaluation of an actual texture dataset. The HI datasets are a complex problem, with several publications (de Matos *et al.*, 2021) that explored diverse methods, including the ones designed for texture characteristics. As a challenging problem, they highlighted the advantages of the LMFCN against the commonly used methods.

3.2.1 Gaussian Images

Our synthetic dataset contains 600 RGB images of 200×200 pixels generated from two Gaussian distributions with μ=128 and 125 and σ^2=35 and 30 for classes 0 and 1, respectively. Each pixel from the three channels of the images has its value drawn from its respective Gaussian distribution. Besides the Gaussian distribution pixel value, we also applied a striped pattern for each class tilted with 22°± 5 and -22°± 5 for classes 0 and 1, respectively. We chose and tuned

the distributions' mean and variance values to allow us to explore the LMFCN to its limits, where it could no longer produce acceptable prediction values. We intended to create a controlled scenario to explore the recognition capacities of the LMFCN, avoiding having other sources of variability, such as those related to image acquisition, quality, standardization, and labeling. The train/test/validation split has 200/200/200 images for each partition. As this dataset is used as a controlled evaluation, the same size splits are intended to avoid extra complexity from the size of the splits. Unlike the actual datasets, the subset images have similar characteristics around the two distributions. Figure 3.1 presents examples of these images.

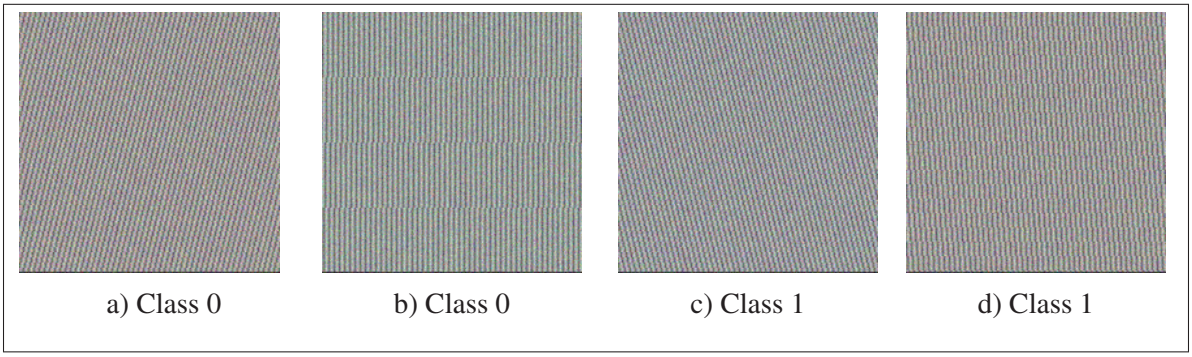
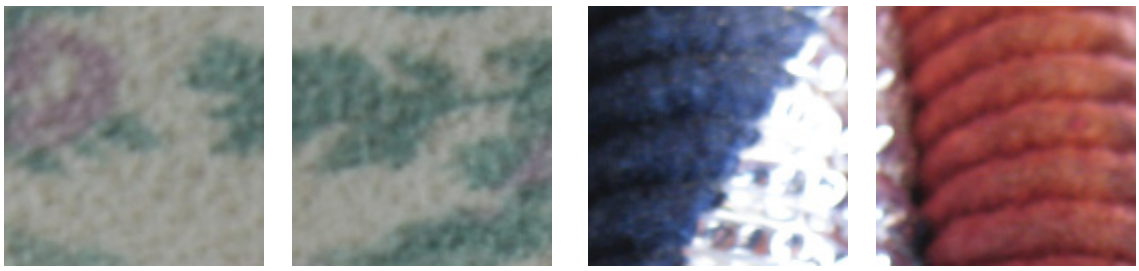


Figure 3.1 Examples of synthetic images generated based on Gaussian distributions and striped pattern

3.2.2 Salzburg Dataset

The Salzburg dataset (Kwitt & Meerwald, 2020) has 32 classes of texture images. Owing to the binary nature of the LMFCN and the fact that its multiclass version was not designed for such a high number of classes, we have selected samples from the Miscellaneous (704 images) and Fabric (1232 images) classes. The criteria for choosing these two classes is to get the most prevalent ones, avoiding the bias of getting the ones that could benefit our approach. This dataset has RGB images of 128×128 pixels resolution. We chose the train/test/validation split of 55/30/15%, respectively, and all comparisons between methods used the same image distribution on these partitions.



a) Fabric b) Fabric c) Miscellaneous d) Miscellaneous

Figure 3.2 Examples of Salzburg images from the two classes selected

3.2.3 BreaKHis Dataset

The BreaKHis dataset contains HIs obtained by open surgical biopsy from tumors in mammary tissue (Spanhol *et al.*, 2016). The samples were sliced and stained with Hematoxylin and Eosin (H&E), highlighting different types of structures at the cellular level. Hematoxylin exhibits a violet color on cell nuclei, reacting to the DNA molecules. Eosin renders purple and red colors to the cytoplasm, extracellular macromolecules, minerals, and blood cells. This dataset can be used in binary classification tasks, distinguishing tumors from benign or malignant, or in multiclass classification tasks, with eight classes, identifying between four types of benign and malignant tumors.

The dataset images are initially divided patient-wise and in four magnification factors (40×, 100×, 200×, and 400×). We used only one of the magnifications (100×), ignoring the patient split. Patient-wise split is important when testing the capability of a method relative to a specific problem. Our purpose in this experiment was to evaluate the ability of the proposed method to deal with general textures. Patient-wise splits usually require problem-specific image preprocessing, which could introduce extra variables to our experiments. This dataset presents variations in image quantity per class and coloration due to the staining process's inherent features. Creating our splits allows us to analyze and compare our approach objectively against other methods avoiding specificities of the dataset that need domain-specific image preprocessing and techniques. We used RGB images resized to 350×230 pixels. The experiments do not

employ normalization or data augmentation techniques to avoid extra complexities and biases to the experiments. The train/test/validation split was 55/30/15%, and images were allocated at each split randomly, not patient-wise.

The state-of-the-art results are not comparable to our approach results because we did not follow the same split as the original work presenting the dataset. However, our separation enables comparison between equivalent methods in this work. As in the Salzburg dataset, all comparisons between methods used the same split and image allocation. Figure 3.3 shows images identifying both the binary and multiclass image distribution.

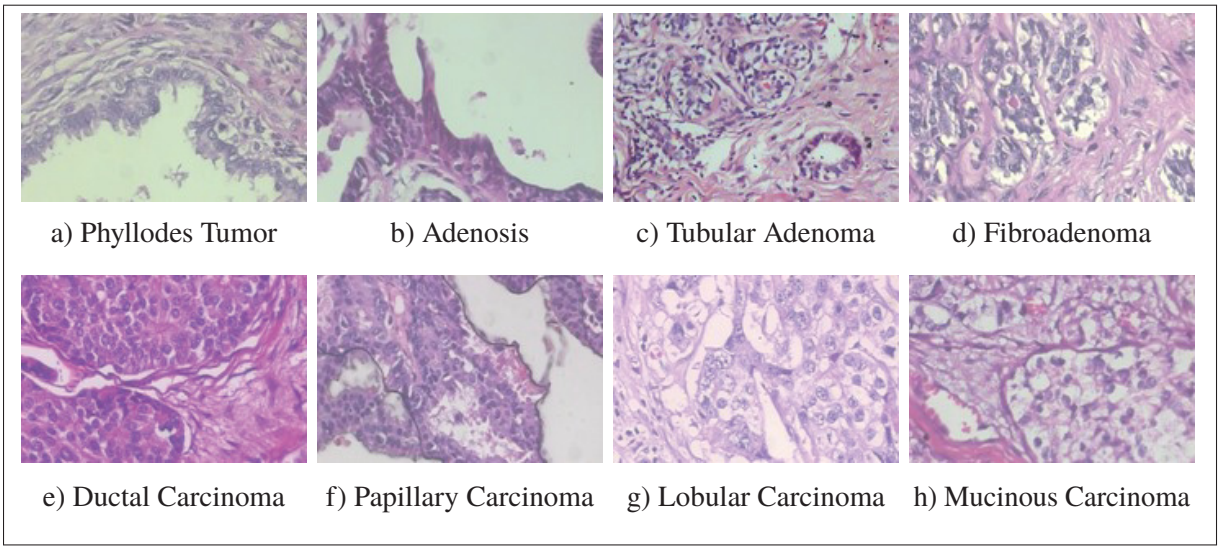


Figure 3.3 Example of BreaKHis dataset images, (a) Phyllodes Tumor (b) Adenosis, (c) Tubular Adenoma, (d) Fibroadenoma, (e) Ductal Carcinoma, (f) Papillary Carcinoma, (g) Lobular Carcinoma, (h) Mucinous Carcinoma. (a,b,c,d) Benign Tumors, (e,f,g,h) Malignant Tumors

3.2.4 CRC Dataset

The CRC dataset (Kather *et al.*, 2016) contains images of eight types of colorectal tissue stained with H&E. There are two ways of processing this dataset, one is using its 5000×5000-pixel images for segmentation, identifying each tissue, and another is to use tissue labeled 150×150-pixel patches from the bigger images as a classification problem.

We used only stroma and tumor images to analyze and compare CNNs, handcrafted feature extractors, and the LMFCN, as suggested by Kather *et al.* (2016) to perform binary classification experiments. We also used it as a multiclass problem with the eight classes. All eight classes have 625 images each, which is a balanced dataset. In our experiments with this dataset, we split it into 55%, 15%, and 30% for train, validation, and test, respectively. This dataset does not provide patient-wise information. As in the BreaKHis dataset, we avoided preprocessing the images because we focused on comparing methods and not on state-of-the-art results on the datasets. Figure 3.4 shows images of the CRC dataset identifying each class.

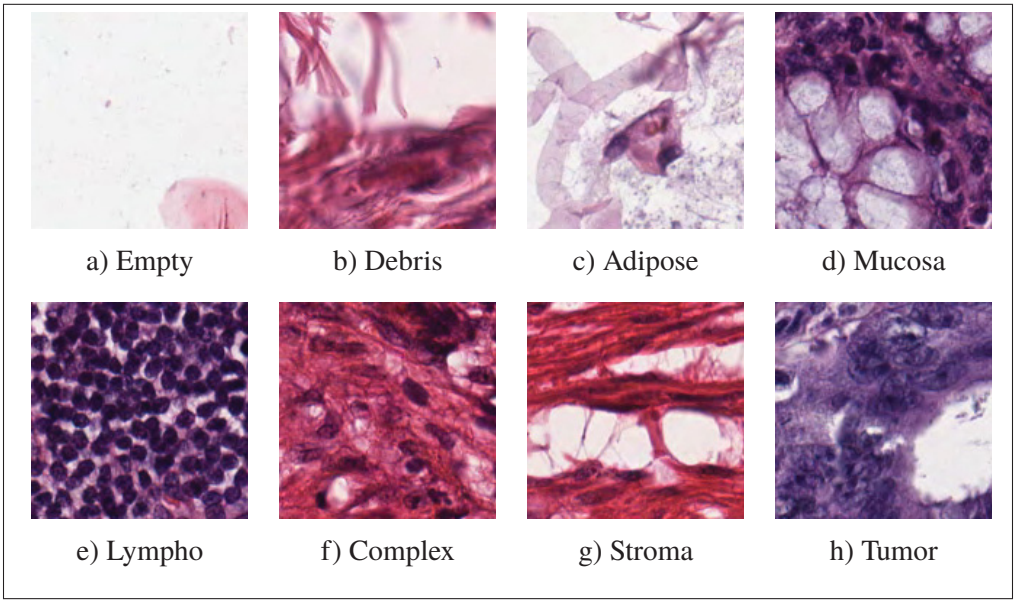


Figure 3.4 Examples of the CRC dataset images. (a) Empty, (b) Debris,(c), Adipose, (d) Mucosa, (e) Lympho, (f) Complex, (g) Stroma, (h) Tumor

3.2.5 BACH

BACH is an HI dataset of mammary tissue samples stained with H&E (Aresta *et al.*, 2019). It was the ICIAR 2018 competition dataset and did not have a publicly labeled test set. We used only the original training set and did not use the public system to evaluate the performance of our method with the test set. Using the test evaluation system would cause an overhead in our experimental procedure. Researchers should submit their test predictions to a system that returns

the accuracy for any state-of-the-art comparison. We intend to use this dataset to compare the equivalent approaches instead of improving the final classification performance. That would require several pre-processing and domain-specific optimizations, as in the BreaKHis dataset. BACH has four classes: normal, benign, *in situ*, and invasive. They represent tumor absent tissue, tissue with a benign tumor, and the last two, malignant tumors. We grouped the normal and benign in one class and *in situ* and invasive into another for a binary classification task.

All images of the BACH dataset have a resolution of 2048×1536 pixels, and each class has 100 images, so it is a balanced dataset. We have resized the images four times to 512×384 pixels to make the training more efficient and fit samples into memory. The dataset split is 55%, 15%, and 30% for training, validation, and testing, respectively. Figure 3.5 shows samples of the four classes.

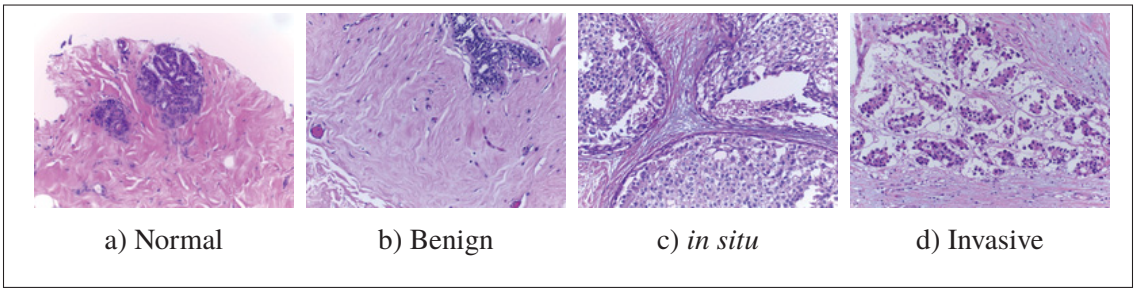


Figure 3.5 Example of BACH images. (a) Normal, (b) Benign, (c) *in situ*, (d) Invasive

3.3 Performance Metrics

This section presents the metrics used in the analysis, which includes balanced accuracy and metrics of complexity based on the latent representation generated by the $f_{\text{fcn}}(.)$. We also used Uniform Manifold Approximation and Projection (UMAP) to plot the latent representation in low dimensionality.

3.3.1 Accuracy and Balanced Accuracy

We have used accuracy, defined by Equation (3.1), and balanced accuracy, defined by the imbalanced class weighting from Equation (3.3) and the metric defined by Equation (3.2). They are the main evaluation metrics in the LMFCN experiments. Although they are simple metrics, they aid in comprehending if the method is working and allow us to compare our approach to other methods. However, these two metrics cannot inform how and why the technique works; in the case of increasing values, they do not clarify the reason for this behavior. Thus, we used complementary evaluation metrics, like the value of the losses and the number of SVs in the case of the LMFCN. Instead of using more complex final evaluation metrics such as F1-score, Area Under the Curve, Jaccard score, or Cohen-Kappa score, we used balanced accuracy. We showed its variations concerning the latent representation change.

$$\text{accuracy}(\mathbf{y}, \mathbf{o}) = \frac{1}{n} \sum_{i=0}^{n-1} 1(y_i = o_i) \tag{3.1}$$

where $\mathbf{y}$ is the vector of predicted outputs, $\mathbf{o}$ is the vector of expected outputs, n is the number of instances, and $1(.)$ is the indicator function, which returns 1 if y_i is equals o_i, otherwise it returns 0.

$$\text{balanced\_accuracy}(\mathbf{y}, \mathbf{o}, \mathbf{w}') = \frac{1}{n_{classes}} \sum_{i=0}^{n-1} 1(y_i = o_i) {w'}_{o_i} \tag{3.2}$$

where ${w'}_{o_i}$ is the weight of instance i in the class o_i.

$${w'}_{o_i} = \frac{1}{\sum_{j=0}^{n-1} 1(o_j = o_i)} \tag{3.3}$$

3.3.2 Complexity Measures

The complexity measures presented by Ho & Basu (2002) help us explain the relationship among the classifiers' accuracy performance, the latent representation, and images from a dataset. They have the advantage of being based on geometric characteristics and class distribution. The complexity measures can generate a meaningful value from a broad latent representation created by a feature extraction mechanism and based on distances between instances. They provide a better analysis than dimensionality reduction methods (e.g., PCA, random projections) or a low-dimensional latent representation ($\mathbb{R}^2$). Compared with the first, no bias is introduced by the chosen method, like hyperparameters and configurations that could benefit one or another approach. High-dimensional latent representations can be used with complexity measures, increasing the representation capabilities. Thus, the accuracy values in experiments are more realistic than the ones resulting from a low-dimensional representation with reduced representation power.

There are six categories of complexity measures based on six aspects: single features, neighborhood, linearity, dimensionality, class balance, and network. In our work, we have used five complexity measures related to our classifier and the distance metrics: N1, N2, and T1, from the neighborhood categoy and F1 and F2 from the single feature category (Lorena, Garcia, Lehmann, Souto & Ho, 2019).

The neighborhood measures inform the class overlapping and the shape of the decision boundaries. They are directly based on pairwise instance distances. The N1 measure, defined in Equation (3.4), expresses the fraction of borderline points by building firstly a minimum spanning tree (MST) from the instances where the edges are weighted based on the inter-vertice distance. It then calculates the percentage of connected vertices from opposite classes. These are samples located on the border or overlapping regions. This measure estimates the margins' complexity and the boundaries definition's complexity. The higher the value, the more complex the problem. Although, N1 may also have a high value if the distance between inter-class samples in the boundaries is smaller than intra-class distances (Basu & Ho, 2006).

$$\text{N1} = \frac{1}{n}\sum_{i=1}^{n} I(d(x_i, x_j) \in MST \wedge o_i \neq o_j) \tag{3.4}$$

where MST is the minimum spanning tree from all instances, $d(x_i, x_j)$ is the distance between instances i and j, o_i and o_j is the required output of instances i and j, and n is the number of instances.

The ratio of intra-class/inter-class nearest neighbor distance (N2) is calculated by dividing two summations, the intra- and the inter-class distances, according to Equation (3.5). The intra-class distance is calculated by summing the distance of all instances to their closest same-class instance. The inter-class distance is obtained by summing the distance of all opposite-class closest neighbors. A low N2 indicates a simple problem where the overall inter-class distance is greater than the intra-class distance. This measure is not sensitive to boundary characteristics but to the within-class data distribution. Similar to the N1, the N2 measure also has some drawbacks, like linearly separable data along two thin parallel lines can be simple to separate but will produce a high N2.

$$\text{N2} = \frac{\text{intra\_extra}}{1 + \text{intra\_extra}} \tag{3.5}$$

$$\text{intra\_extra} = \frac{\sum_{i=1}^{n} d(x_i, NN(x_i) \in o_i)}{\sum_{i=1}^{n} d(x_i, NN(x_i) \in o_j \neq o_i)} \tag{3.6}$$

where $d(.)$ is pairwise distance between instances i and j, $NN(.)$ is the nearest neighbour of instance x_i, and o_i and o_j are the expected output of instances i and j, respectively.

The T1 measure is the ratio between the number of hyperspheres encompassing groups of samples and the total number of samples, as defined in Equation (3.7). Each sample has a hypersphere centered in it that grows until touching an instance from another class. After growing all hyperspheres, the smaller ones contained completely inside a bigger one are eliminated, reducing their total number so that $n_{\text{hyperspheres}} < n$. A set of instances with latent representation

that generates few spheres suggests a simple distribution. In this case, with same-class instances packed together. Similar to the N2 measure, it expresses more within-class data distribution, not boundaries.

$$\mathrm{T1} = \frac{n_{\text{hyperspheres}}}{n} \tag{3.7}$$

where $n_{\text{hyperspheres}}$ is the total number of hyperspheres to cover all instances of the dataset, excluding small hyperspheres entirely contained in larger ones, and n is the total number of instances.

The single feature F1 measure (maximum Fisher's discriminant ratio) represents the overlapping of features from different classes. It is calculated by Equation (3.8) (Lorena *et al.*, 2019).

$$\mathrm{F1} = \frac{1}{1 + \max\limits_{i=[1,\phi]} r_{f_i}} \tag{3.8}$$

where ϕ is the number of features, and r_{f_i} is the discriminant ratio for feature f_i calculated by Equation (3.9).

$$r_{f_i} = \frac{\sum\limits_{j=1}^{n_c} n_{c_j}(\mu_{c_j}^{f_i} - \mu^{f_i})^2}{\sum\limits_{j=1}^{n_c}\sum\limits_{l=1}^{n_{c_j}}(x_{l_i}^{j} - \mu_{c_j}^{f_i})^2} \tag{3.9}$$

where n_c is the number of samples of a class, n_{c_j} is the number of samples in class c_j, $\mu_{c_j}^{f_i}$ is the mean of attribute f_i of class c_j, μ^{f_i} is the mean of attribute f_i of all classes, and $x_{l_i}^{j}$ is the value of attribute f_i of one sample of class c_j.

Considering Equation (3.8), when the latent representation of a set has at least one feature with a high discriminant ratio, F1 will be lower than one. Thus a low F1 means simple problems with little overlapping of at least one attribute.

The volume of the overlapping region (F2) is calculated by Equation (3.10) according to Ho & Basu (2002).

$$\text{F2} = \prod_{i=1}^{\phi} \frac{\min[\max(\mathbf{x}_i, c_0), \max(\mathbf{x}_i, c_1)] - \max[\min[\mathbf{x}_i, c_0), \min(\mathbf{x}_i, c_1)]}{\max[\max(\mathbf{x}_i, c_0), \max(\mathbf{x}_i, c_1)] - \min[\min(\mathbf{x}_i, c_0), \min(\mathbf{x}_i, c_1)]} \tag{3.10}$$

where ϕ is the number of features, $\mathbf{x}_i$ is a vector with values of the feature i of all samples, c_0 is the class 0, and c_1 is the class 1.

The numerator becomes zero when there is no overlapping in at least one feature, so low values mean a simple problem. Likewise other metrics, it also has drawbacks, like the representation shown in Figure 2.7a. Despite small overlapping, the blue class's mean will match the red class's mean. Outliers may also cause negative impacts on min and max calculations, incorrectly pointing to a problem as being hard. This metric, jointly with the others, is not an absolute answer to evaluate how simple a problem is but may help to understand how the latent representation is being updated by the FCN.

3.3.3 Uniform Manifold Approximation and Projection (UMAP)

We have generated graphical representations based on the instances' latent representation using UMAPs (McInnes, Healy, Saul & Großberger, 2018). UMAPs are a dimensionality reduction method that, as aforementioned, may insert biases in the resulting reduced representation considering the method's hyperparameters or configurations. However, UMAPs allow choosing Euclidean distance as the distance metric used to build the visual representation. Our approach relies on the Euclidean distance (to calculate the anchors and the loss function). Thus, UMAPs tend to produce results related to the latent representations generated by our approach. They target a reduced dimensionality mapping that keeps the characteristics of distance and neighborhood.

We used two parameters different from the defaults in our plots of latent representation using UMAPs, the minimum distance and the number of neighbors, changed from 0.1 to 0.05 and from 15% to 40% of n, respectively. The minimum distance impacts the graphs' packing; small values produce too close instances and make it difficult to distinguish between instances. The minimum distance is the minimum value of distance allowed for instances to be apart in the low-dimensional representation. Small values produce clumpier embedding, and larger values focus on preserving the overall topological structure. We experimented with a sequence of parameters and found the best graphical representation with a value of 0.05. Another parameter we tuned was the number of neighbors. A high value produces a more global representation. We set this value proportionally to the number of instances as a fixed value could be too high in small datasets, and small values produced too local results. In this case, we used 40% of the instances.

3.4 Experimental Results

This section presents the results of our comparisons of the LMFCN with deep and shallow methods. We also show results from the analysis of the LMFCN with a low-dimensional latent representation, which intends to reveal the evolution of the latent representation over the training epochs, given the weight updates carried out by the backpropagation algorithm with the error calculated by the proposed loss function presented in Section 2.2.3. Such a low dimensionality allowed us to visualize the latent representation. We also show the comparison results with a hybrid method using CNNs as feature extractors. The results comprise the balanced accuracy comparison, graphical representation of the latent representation (using low dimensionality and UMAPs), and complexity measures.

3.4.1 Low-Dimensional Latent Representation

This experiment aims to analyze the updates in the latent representation by the backpropagation algorithm based on the developed loss function. We used ϕ=2, which facilitates the representation plotting and observing the class separation. This ϕ value is not advantageous because it can

not carry enough information to help the classification. This is why we used only the synthetic dataset in this experiment. A real dataset would present a challenging problem with such a low dimensionality of the latent representation.

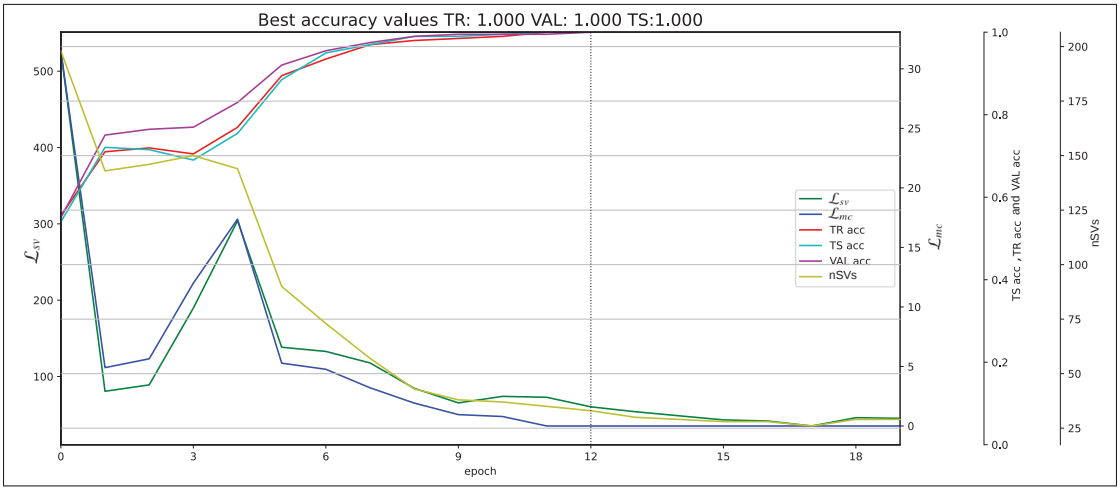
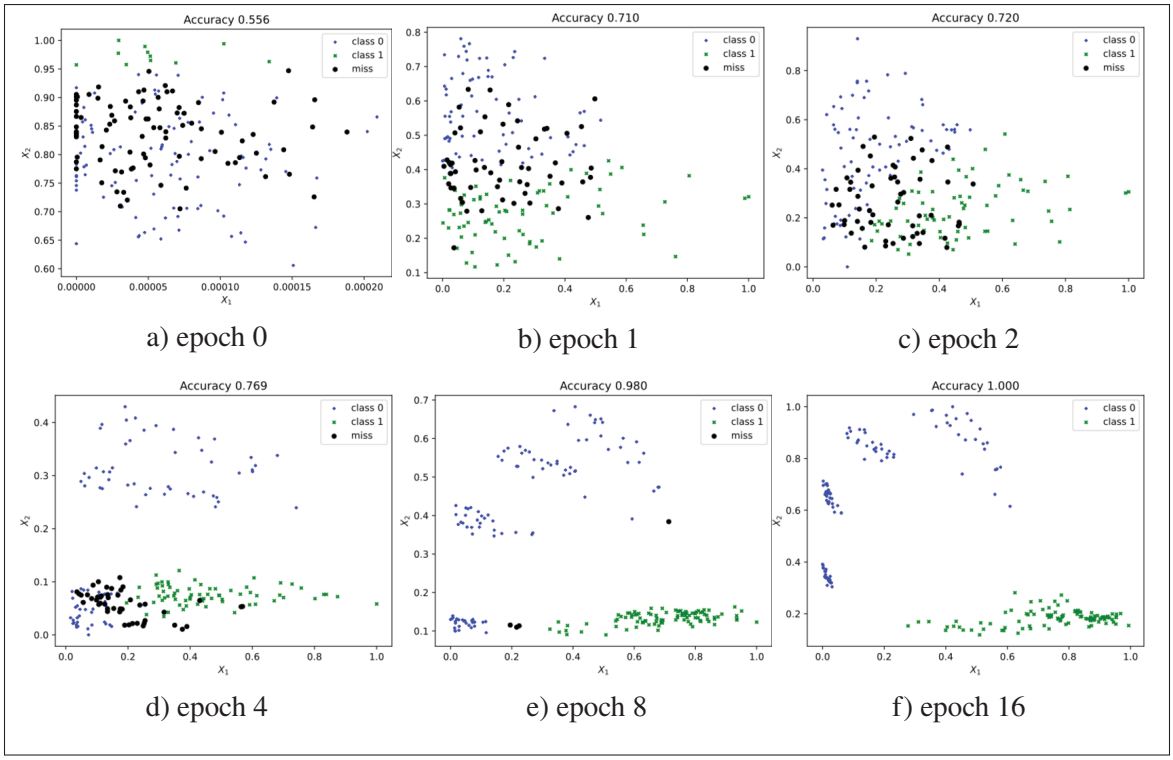


Figure 3.6 Evolution of accuracy and number of SVs according to the training epochs for the LMFCN with a 2-dimensional latent representation using the synthetic Gaussian images dataset.



a) epoch 0 b) epoch 1 c) epoch 2

d) epoch 4 e) epoch 8 f) epoch 16

Figure 3.7 Scatter plot of the latent representation of Gaussian images with two attributes $\mathbf{x} = \{x_1, x_2\}$ of each training instance along the training epochs.

Figure 3.6 shows a training graph on the Gaussian image dataset over 20 epochs with the LMFCN. The FCN in this experiment used w=200, h=200, c=3, and ϕ=2. This graph displays the following information: loss value from SVs ($\mathcal{L}_{\text{sv}}$), loss value from misclassified instances ($\mathcal{L}_{\text{mc}}$), accuracy on training (TR), validation (VAL), and test (TS) sets, the number of support vectors (nSVs), and the epoch of the best validation accuracy (vertical dotted line). The best-balanced accuracy values for training, validation, and test sets correspond to the peak validation accuracy. The value of $\mathcal{L}_{\text{cc}}$ (opposite class instances loss) is not shown in this graph since it was not used in this experiment. It is possible to observe that the values of the loss terms $\mathcal{L}_{\text{sv}}$ and $\mathcal{L}_{\text{mc}}$, as well as the number of SVs decreases over the epochs, while the accuracy rises. This information tells us that the SVs are getting close to their anchors, as this distance is computed by $\mathcal{L}_{\text{sv}}$. The number of SVs also reduces, indicating that fewer SVs are needed at each epoch to separate the examples of different classes. This implies a latent representation that is easier to classify. This execution used the following parameters: sv_{close}=5, wr_{close}=1, sh_{close}=0, C=100 and the RBF kernel.

Figure 3.7 presents a sequence of scatter plots showing the changing in the latent representation space of the training instances from epochs 0 to 16 of Figure 3.6. On epoch 0, with randomly initialized weights, the instances of two classes (blue circles and green circles) cannot be easily separated by the LMFCN, generating a balanced accuracy of about 55.6%. Misclassified instances are shown as black circles. After the first epoch, the classes are less overlapped, and the LMFCN achieves an accuracy of 71%. Such an accuracy increases to 98% and 100% on epochs 8 and 16, respectively, where the separation of the examples of the two classes is clear. This shows the effects of weight updating based on the $\mathcal{L}_{\text{sv}}$ and $\mathcal{L}_{\text{mc}}$ loss terms. Between epochs 2 and 4, the accuracy and the loss value increase due to the high number of SVs in the decision boundaries. The SVs are also distant from the instances from their classes. In epoch 8, the disentanglement is clearer, and the model achieves stability.

These experimental results illustrate the evolution of the quality of the latent representation over the epochs in the scatter plots, noticeable by the increasing class separation of the instances, with a consequent increase in accuracies. It also shows the reduction in the number of SVs, which leads to less complex discriminants that can generalize better on unseen data.

3.4.2 Comparison Between CNNs and the LMFCN

In this experiment, we compare the LMFCN with deep learning methods. The goal is to show that our approach can obtain equivalent or better performance than CNNs (texture CNNs in this case) of similar size. Furthermore, considering that both CNNs and the LMFCN use weight updating based on the backpropagation algorithm, we want to analyze the impact of the proposed loss function and the hinge and BCE loss functions on the update of the parameters of the convolutional filters.

The results for each dataset aggregates a set of graphs with the five complexity measures computed on the latent representation generated by the FCN at each epoch for images of the training sets. In these experiments, we compare the LMFCN with CNNs with hinge loss (CNN-H) and CNNs with BCE loss (CNN-BCE). Both experiments use the same CNN architectures but different loss functions. The CLs of the CNNs are similar to the LMFCN (Figure 2.5 and Table 2.1), which implies the same dimension, number of filters, and values for additional parameters. The additional deeper layers (FC) have two 32-wide layers and a last layer with an output of two (binary classification). All the CNNs training used a learning rate of 1.0, defined empirically. Both methods (LMFCN and CNNs) used Ada Delta as the optimizer. For LMFCN, we used a learning rate of 0.1, sv_{close}=5, wr_{close}=1, sh_{close}=0, C=100 and $\gamma = 1/(\phi \times \sigma^2)$. σ^2 is the variance of the latent representation of all training instances. All experiments were repeated five times to verify the consistency, but we presented the graphs for the first run for conciseness. Figures 3.8 to 3.12 show all datasets' five complexity measures and corresponding accuracy. The accuracy for training, validation, and test sets is reported for the best-balanced accuracy epoch. To help interpret the complexity measure graphics, we provide Table 3.1, which summarizes the meaning of each measure described in Section 3.3.2.

When observing F1 and F2 measures in Figure 3.8, both CNN-H and CNN-BCE showed more variation than the LMFCN, so the weight update was more significant to obtain better results in both CNNs. In contrast, to achieve more than 90% accuracy in the initial epochs, the CLs of the LMFCN did not suffer substantial updates, producing minor changes in the latent representation

Table 3.1 Complexity measures summary

Metric	Value	Meaning
F1	F1 < 1	One high discriminative single feature, simple problem
	F1~1	No discriminative single feature
F2	F2~0	No overlapping in at least one feature, simple problem
	High	Hard problem
N1	Low	More same class instances connected than opposite class, simple problem
	High	More complex margin
N2	Low	Inter-class distance higher than intra-class, simple problem
	High	Interleaved instances from different classes, hard problem
T1	Low	Few hyperspheres, more clustered data, simple problem
	High	More hyperspheres, but may be caused by few clusters and high number of SVs

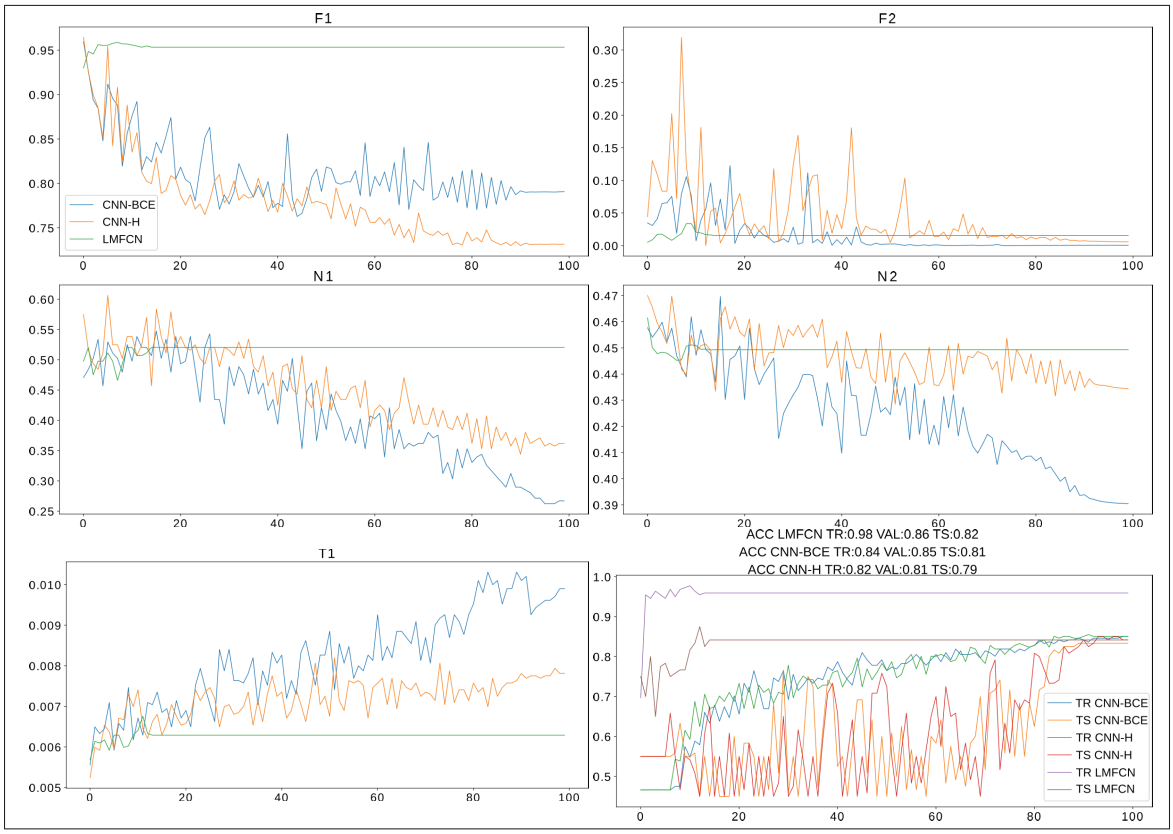


Figure 3.8 Complexity measures and accuracies for the BACH dataset with a 16-dimensional latent representation. The models were trained up to 100 epochs.

of training samples. High F1 and F2 values indicate that the latent representation does not have a single feature to ease the classification (single non-overlapping attribute). The LMFCN used

the initial latent representation (randomly initialized) more efficiently than CNNs. The classifier did not need a single feature with high discriminative power because even with high F1 and F2, it obtained higher initial accuracy than CNNs.

N1 and N2 metrics can reflect the separation of the samples based on all attributes, not isolated attributes like F1 and F2 metrics. Thus, in Figure 3.8, the CNNs needed more inter-class distance, with small clusters, as the T1 value also increased during training. Looking at N1 and N2 curves, the LMFCN achieved a much higher accuracy with a slight separation. Analyzing the behavior of the T1 curve for the LMFCN, the number of spheres grows over the training epochs. This behavior is explained by the nature of the classification method based on SVs that define the margins. Depending on the complexity of the margins, spheres can not grow too much around the SVs without touching other class instances. Thus, the RBF kernel creates a more complex margin over the training epochs, increasing T1 values. It is worth observing that the balanced accuracy achieved by the LMFCN for all subsets (training, validation, and test) outperformed the CNNs.

The initial absolute values of N1 and N2 indicate that the latent representation generated by the FCN for the BreakHis dataset is more powerful than in the BACH dataset in Figure 3.9. Hence, the first weight update produced even more separation. The T1 values indicate a similar substantial update and the definition of relevant SVs that generate better margin outlining. The CNN approaches started improving training accuracy only around epoch 18, where we can see that by the values of N1, T1, and N2, the latent representation begins to enhance and undergoes more substantial modifications. Thus, training the FC layers of the CNNs delays the optimization of the latent representation, even if the FC layers have fewer parameters than the CLs. Furthermore, the CLs training behavior in the CNNs shows that the needs of the FC layer as classifiers are different from the SVM, so in a feature extraction approach with pre-trained CNNs, the features are not the most suitable for a large margin classifier.

The three compared methods (LMFCN, CNN-H, and CNN-BCE) achieved high accuracy on training, validation, and test sets of the CRC dataset, as shown in Figure 3.10. The two-class

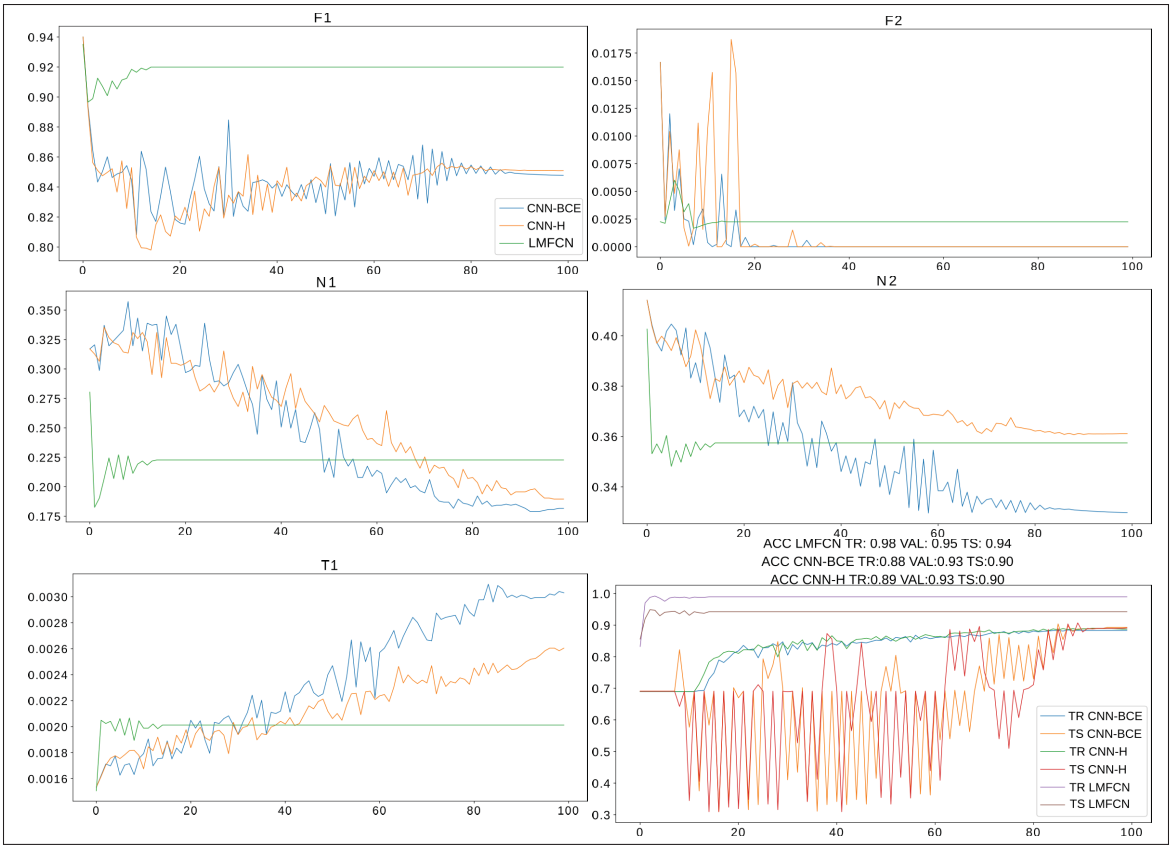


Figure 3.9 Complexity measures and accuracy for the BreaKHis dataset with a 16-dimensional latent representation. The models were trained up to 100 epochs.

version of the CRC dataset uses only stroma and tumor images. Such images have very distinct textures (Figures 3.4g and 3.4h), which reflects in the high accuracy values achieved on the first epoch, with randomly initialized (untrained) filters in the LMFCN. As in the BreaKHis experiments, the LMFCN exploited the class separation and reduced N1 and N2 over the epochs. Comparing the results achieved on BreaKHis and CRC datasets, the network complexity measures (N1 and N2) have low values in the latter. The CRC dataset has some class separation at the beginning (N1 and N2), although it has a high F1 indicating that the high initial accuracy value does not depend on a single component. The CNN approaches presented higher accuracy at epoch 20 than the experiments with other datasets. Despite the CNNs' early success, the F1, N1, and N2 measures show that the training procedure of the CNNs modified the CLs to

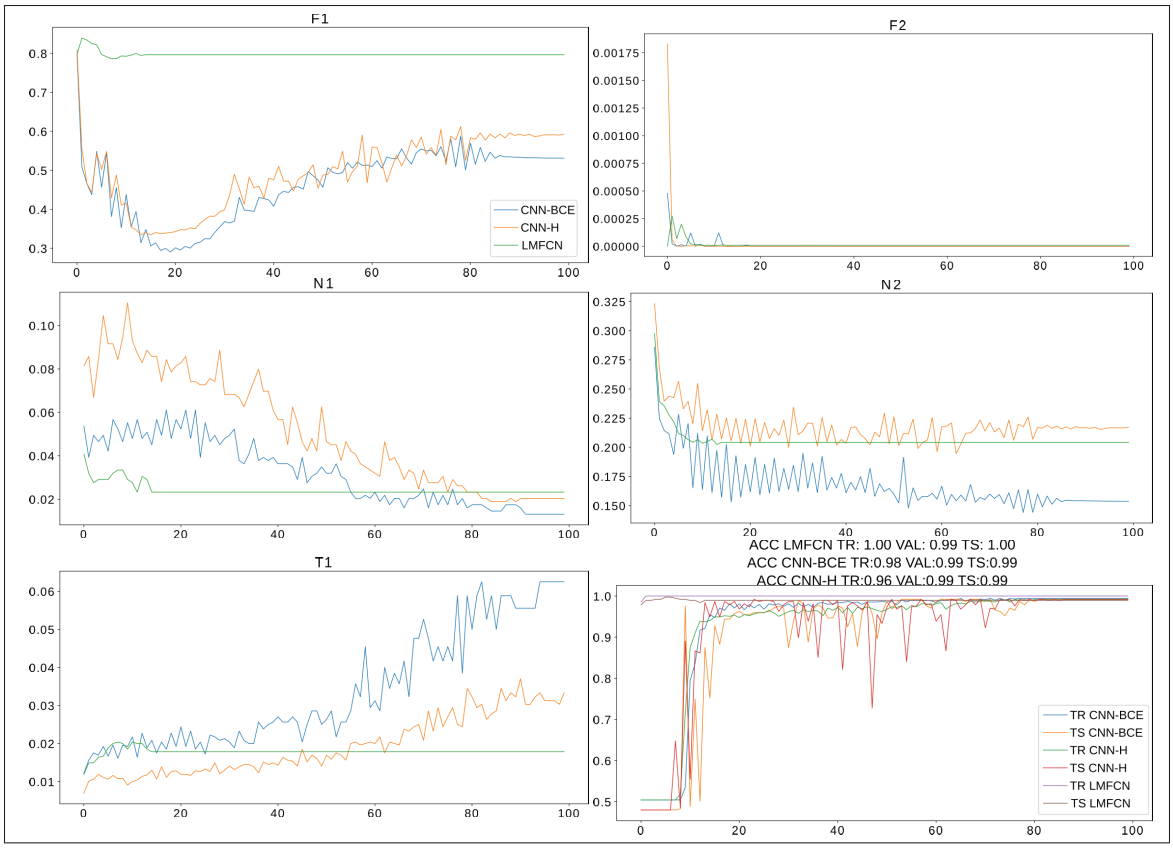


Figure 3.10 Complexity measures and accuracies for the CRC dataset with a 16-dimensional latent representation. The models were trained up to 100 epochs.

generate a latent representation while also updating the FC layers. In contrast, the LMFCN only updates the CLs.

The LMFCN reached high balanced accuracy on the first training epoch in the Gaussian image dataset, as seen in Figure 3.11. The drastic difference between Figures 3.11 and 3.6 is due to the difference in the latent representation dimension. Due to the early convergence of the LMFCN, updating the parameters of the convolutional filters of the FCN did not cause significant changes in the latent representation after the first epoch. This dataset highlights the power of the large-margin classifier with the RBF kernel, which can obtain good accuracy due to the capabilities of drawing more complex margins and higher-dimensionality mapping. The CNNs needed more epochs to learn suitable parameters for the convolutional filters to generate a latent representation that allows the FC layers to achieve higher classification accuracy. Compared to

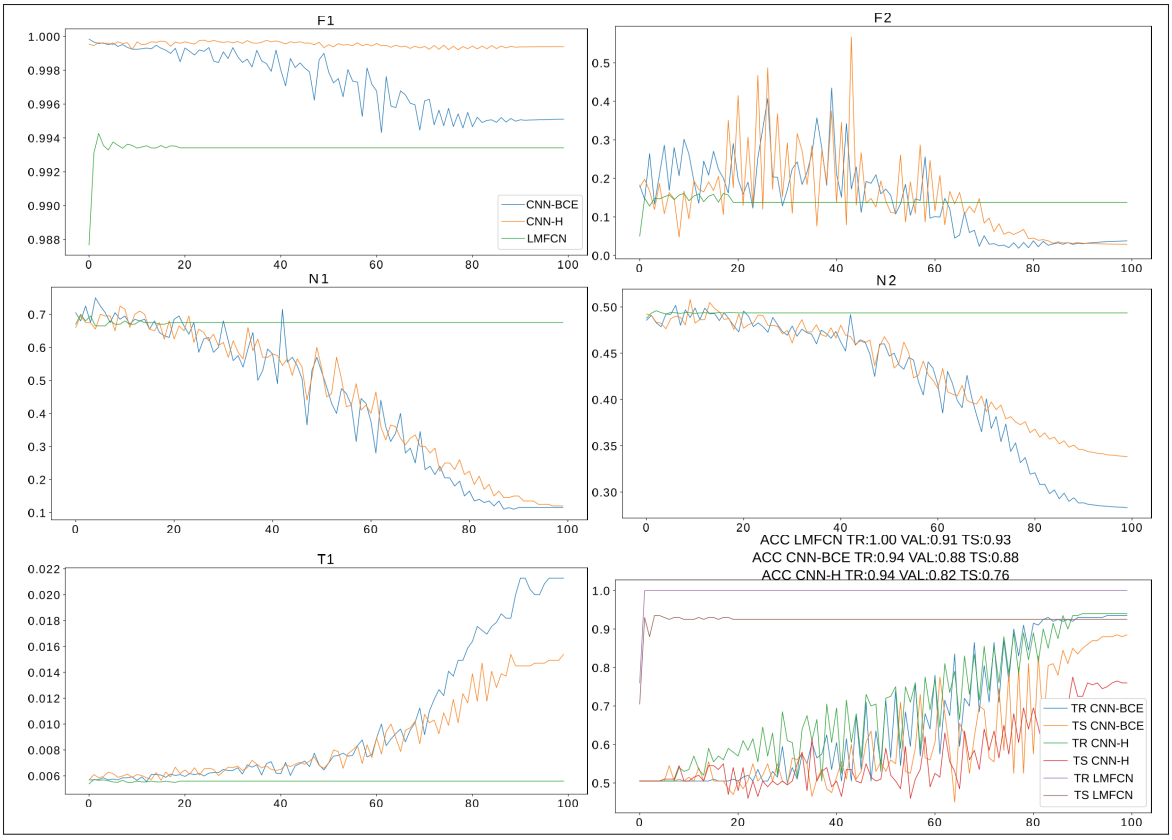


Figure 3.11 Complexity measures and accuracy for the Gaussian dataset with a 16-dimensional latent representation. The models were trained up to 100 epochs.

the other datasets evaluated, the most symbolic measure representing such changes is the N1 measure, varying from 0.7 to 0.1 approximately. The increase of the T1 measure over the epochs indicates that the classes are not well separated, but the absolute value difference is not high, indicating the formation of small clusters.

Figure 3.12 presents the changes in the complexity measures and accuracy for the Salzburg dataset over 100 training epochs. The behavior is comparable to the one seen in the BreaKHis dataset. We found these results interesting as Salzburg is a pure texture dataset. As in other experiments, the LMFCN achieved high accuracy on the first epochs and then stabilized, but the CNNs needed almost 90 epochs to reach a similar accuracy. As the CNNs training needs to update both the CL and FC layers, they did not benefit from the initial class separation, indicated

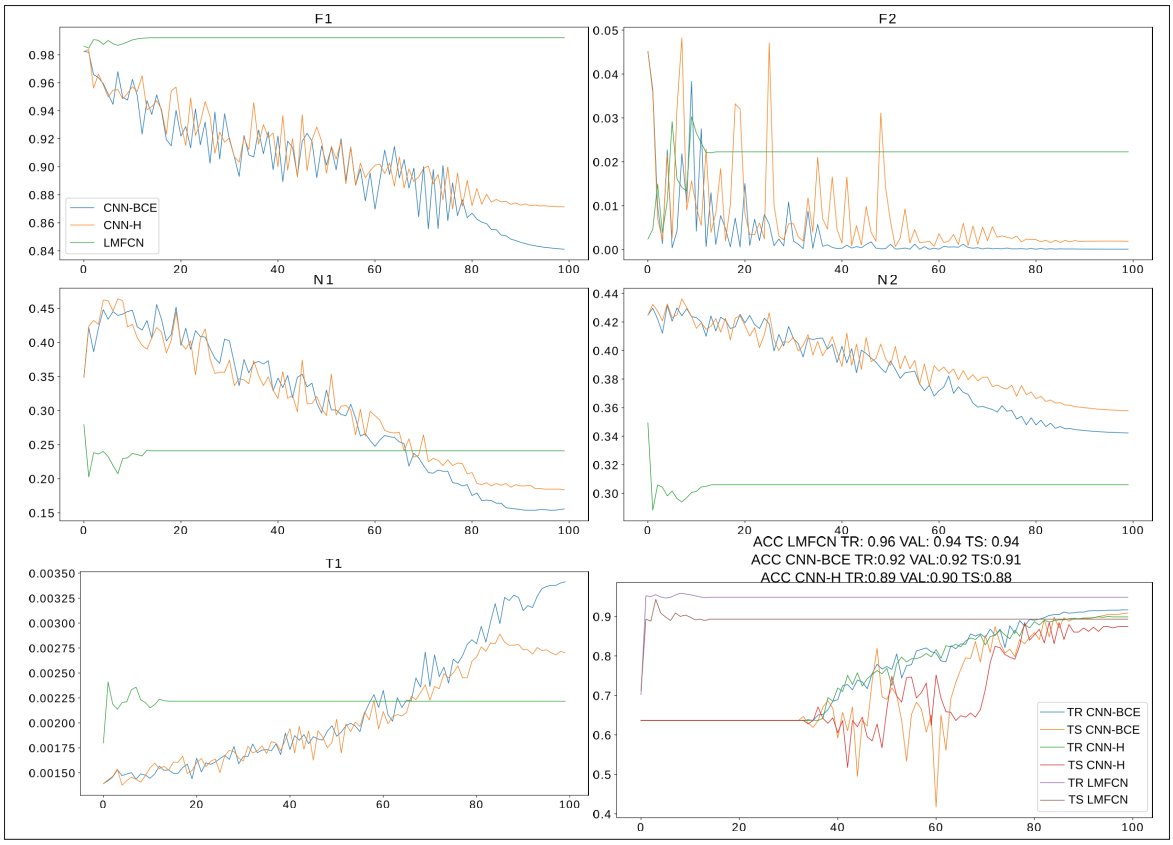


Figure 3.12 Complexity measures and accuracy for the Salzburg dataset with a 16-dimensional latent representation. The models were trained up to 100 epochs.

by the initial values of N1 and N2 measures. On the other hand, the LMFCN took advantage of such an initial class separation (low N1 and N2 values), even with high F1, which indicates that there is no single discriminative component in the representation.

Figure 3.13 aggregates the results achieved by the LMFCN for all five complexity measures, training and test balanced accuracies, and the number of SVs over 15 epochs for all the datasets. This figure aims to plot the LMFCN analysis information without the fluctuation and scale differences caused by the CNNs in the previous graphs. Both training and test accuracy show that the LMFCN improved the values over the iterations. The number of SVs shows this value reduction over training epochs, meaning that the classes are more separated, needing fewer SVs to define the large-margin discriminant. The value of the N2 measure decreases for all datasets

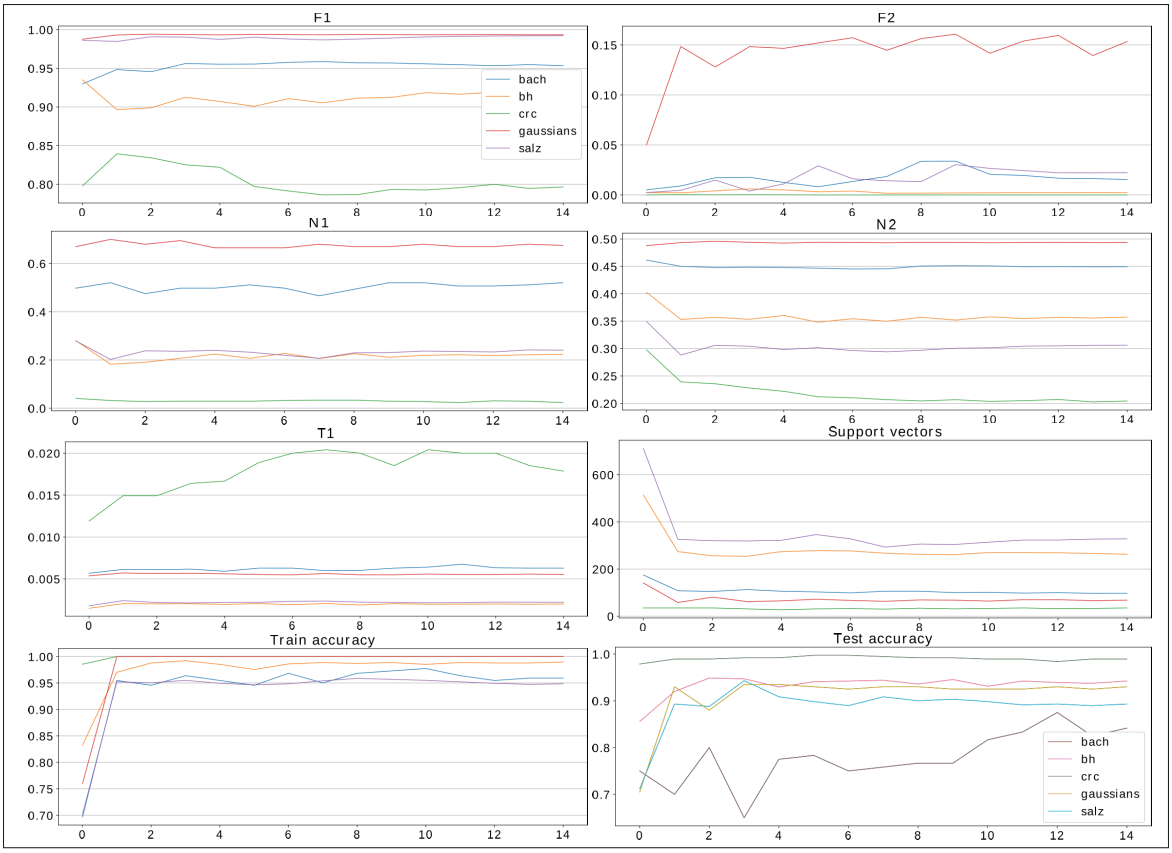


Figure 3.13 Complexity measures and balanced accuracy for all tested datasets with the LMFCN using a 16-dimensional latent representation.

but Gaussian images, which is compensated by the T1 measure. An increase in the T1 measure for all datasets reflects the definition of the boundaries by the SVs. T1 would hypothetically decrease if the latent representation gets updated by searching for grouping around some chosen instances. In this supposed situation few spheres around the chosen instances would contain many instances from the same class before touching other class instances. The LMFCN strategy does not focus on grouping or clustering instances. Instead, it focuses on margins, so it tends to draw more spheres around the SVs that end up touching opposite class instances earlier because they are in the limit of the decision boundary. Another fact that causes the T1 increase in LMFCN is the complexity of the margins drawn with the RBF kernel. The difference in the F2 measure between Gaussian datasets and the others and the good performance achieved by the

LMFCN in all datasets point towards the adaptation capabilities of our approach to different datasets' characteristics.

With a latent representation generator trained with information from a nearest neighbor-based algorithm, the T1 measure would decrease as the data tends to group around some points. Here, it is clear the difference in the algorithms' strategies. The difference in the F2 measure between Gaussian datasets and the others and the good performance achieved by the LMFCN in both situations point towards the adaptation capabilities of our approach to different situations.

Finally, Tables 3.2 and 3.3 compare the results achieved by all approaches on all datasets for a 16-dimensional latent representation. Table 3.2 shows the average balanced accuracy of five executions with the same training, validation, and test split for each approach. Tables 3.2 and 3.3 also present the results for two pre-trained CNNs (ResNet18 and InceptionV3) used as feature extractors named ResNet18 FE and InceptionV3 FE. Since the representation produced by these two CNNs has more than 16 dimensions 512 and 2048, respectively), the PCA was used to reduce the dimensionality of such representations to 16 dimensions. Both tables also compare with three pre-trained CNNs (ResNet18 FT, SqueezeNet FT, and DenseNet FT). These three CNNs received two FC layers to adapt their output to the binary problem and were fine-tuned with the first layers frozen. The ResNet18 presents high accuracy on the training set but does not generalize well. The LMFCN is superior to the other methods in most cases. In the situations where it loses, the values are close, as in the CRC dataset, where the training dataset presents lower accuracy than the ResNet, which overfits. The most complicated situation for the LMFCN was in the BACH dataset, but it achieved its best accuracy in an average of six epochs.

The binary classification experiments also included the LMFCN with alternative blocks replacing each FCN convolutional layer: the Residual blocks (LMFCN-res) and Inception blocks (LMFCN-inc). They did not show improvement in most experiments compared to the LMFCN with convolutional layers. Their best results were in the Gaussian images dataset, where the LMFCN-res outperformed the other methods but took more epochs to obtain the best validation accuracy than LMFCN.

Therefore, the comparison shows that the LMFCN achieved better or equivalent performance than the other methods but in fewer epochs than the CNNs. ResNet18 outperformed LMFCN in the training set but did not generalize well. The complexity measures helped to verify the difference in the latent representations of CNNs (CNN-BCE and CNN-H) and the LMFCN over the training epochs. They also showed that the latent representation produced by CNNs, used as a feature extractor, is less suitable to be used with a large-margin discriminant than the one obtained with the LMFCN.

Table 3.2 Balanced accuracy achieved by the LMFCN, equivalent CNN architectures, and CNNs as feature extractors for the BACH, BreaKHis, and CRC datasets. Epoch refers to the training epoch where the best validation accuracy was achieved. NA: Not applicable for models not trained iteratively.

Dataset	Architecture	Training	Validation	Test	Epoch
BACH	LMFCN	0.8811 ± 0.0426	0.7857 ± 0.0221	0.8056 ± 0.0203	6
	LMFCN-res	0.8845 ± 0.0264	0.7323 ± 0.0388	0.7570 ± 0.0417	18
	LMFCN-inc	0.8750 ± 0.0451	0.7759 ± 0.0531	0.7773 ± 0.0392	3
	CNN-BCE	0.8926 ± 0.0135	0.8200 ± 0.0291	0.7864 ± 0.0212	88
	CNN-H	0.8534 ± 0.0410	0.8038 ± 0.0261	0.7838 ± 0.0242	68
	ResNet18 FE	0.9981 ± 0.0026	0.7596 ± 0.0122	0.7584 ± 0.0270	NA
	InceptionV3 FE	0.6295 ± 0.0746	0.5924 ± 0.0625	0.6049 ± 0.0630	NA
	ResNet18 FT	0.9184 ± 0.0158	0.8198 ± 0.0281	0.7812 ± 0.0251	10
	DenseNet FT	0.9740 ± 0.0243	0.7782 ± 0.0334	0.7906 ± 0.0302	17
	SqueezeNet FT	0.6690 ± 0.0677	0.7058 ± 0.0538	0.6512 ± 0.0125	37
BreaKHis	LMFCN	0.9882 ± 0.0064	0.9442 ± 0.0137	0.8942 ± 0.0372	5
	LMFCN-res	0.9175 ± 0.0089	0.9040 ± 0.0185	0.8814 ± 0.0071	9
	LMFCN-inc	0.9252 ± 0.0091	0.8953 ± 0.0349	0.8876 ± 0.0272	2
	CNN-BCE	0.8926 ± 0.0092	0.9122 ± 0.0226	0.8926 ± 0.0140	91
	CNN-H	0.8560 ± 0.0041	0.8856 ± 0.0166	0.8638 ± 0.0042	91
	ResNet18 FE	0.9777 ± 0.0062	0.8034 ± 0.0120	0.8262 ± 0.0122	NA
	InceptionV3 FE	0.6058 ± 0.0207	0.6064 ± 0.0209	0.5949 ± 0.0207	NA
	Resnet18 FT	0.9154 ± 0.0342	0.8890 ± 0.0190	0.8640 ± 0.0143	37
	DenseNet FT	0.8776 ± 0.0738	0.8900 ± 0.0171	0.8640 ± 0.0150	25
	SqueezeNet FT	0.4996 ± 0.0089	0.5174 ± 0.0149	0.5108 ± 0.0175	23
CRC	LMFCN	0.9924 ± 0.0046	0.9914 ± 0.0024	0.9914 ± 0.0060	6
	LMFCN-res	0.9912 ± 0.0029	0.9862 ± 0.0088	0.9873 ± 0.0086	19
	LMFCN-inc	0.9906 ± 0.0041	0.9903 ± 0.0045	0.9900 ± 0.0068	17
	CNN-BCE	0.9828 ± 0.0070	0.9934 ± 0.0025	0.9880 ± 0.0111	31
	CNN-H	0.9928 ± 0.0038	0.9924 ± 0.0027	0.9928 ± 0.0070	30
	ResNet18 FE	0.9991 ± 0.0013	0.9683 ± 0.0081	0.9680 ± 0.0150	NA
	InceptionV3 FE	0.7718 ± 0.0352	0.7670 ± 0.0375	0.7842 ± 0.0355	NA
	ResNet18 FT	0.9936 ± 0.0094	0.9788 ± 0.0165	0.9814 ± 0.0112	42
	DenseNet FT	0.9906 ± 0.0094	0.9884 ± 0.0086	0.9862 ± 0.0081	34
	SqueezeNet FT	0.9104 ± 0.0123	0.9258 ± 0.0162	0.9306 ± 0.0098	41

Table 3.3 Balanced accuracy achieved by the LMFCN, equivalent CNN architectures, and CNNs as feature extractors for the Salzburg and Gaussian images datasets. Epoch refers to the training epoch where the best validation accuracy was achieved. NA: Not applicable for models not trained iteratively.

Dataset	Architecture	Training	Validation	Test	Epoch
Salzburg	LMFCN	0.9842 ± 0.0049	0.9446 ± 0.0083	0.9230 ± 0.0081	8
	LMFCN-res	0.8729 ± 0.0187	0.8411 ± 0.0184	0.8362 ± 0.0123	9
	LMFCN-inc	0.8877 ± 0.0174	0.8666 ± 0.0301	0.8450 ± 0.0214	3
	CNN-BCE	0.8966 ± 0.0103	0.8762 ± 0.0100	0.8602 ± 0.0115	91
	CNN-H	0.8258 ± 0.0257	0.8292 ± 0.0277	0.8046 ± 0.0314	84
	ResNet18 FE	0.9946 ± 0.0033	0.9046 ± 0.0221	0.8932 ± 0.0088	NA
	InceptionV3 FE	0.6680 ± 0.0332	0.6607 ± 0.0323	0.6644 ± 0.0473	NA
	ResNet18 FT	0.9744 ± 0.0104	0.9494 ± 0.0188	0.9470 ± 0.0130	45
	DenseNet FT	0.9792 ± 0.0170	0.9474 ± 0.0173	0.9460 ± 0.0132	38
	SqueezeNet FT	0.5712 ± 0.0808	0.7332 ± 0.0169	0.7242 ± 0.0266	28
Gaussian	LMFCN	1.0000 ± 0.0000	0.9131 ± 0.0115	0.9315 ± 0.0082	2
	LMFCN-res	1.0000 ± 0.0000	0.9970 ± 0.0067	0.9708 ± 0.0304	15
	LMFCN-inc	1.0000 ± 0.0000	1.0000 ± 0.0000	0.9586 ± 0,0472	9
	CNN-BCE	0.9421 ± 0.0123	0.8834 ± 0.0186	0.8812 ± 0.0213	72
	CNN-H	0.9418 ± 0.0223	0.8221 ± 0.0127	0.7611 ± 0.0253	56
	ResNet18 FE	0.9740 ± 0.0109	0.6126 ± 0.0198	0.6176 ± 0.0257	NA
	InceptionV3 FE	0.5102 ± 0.0235	0.5311 ± 0.0185	0.5021 ± 0.0258	NA
	ResNet18 FT	0.9700 ± 0.0050	0.9310 ± 0.0129	0.9210 ± 0.0156	19
	DenseNet FT	0.9960 ± 0.0089	0.8958 ± 0.0175	0.8802 ± 0.0167	33
	SqueezeNet FT	0.5046 ± 0.0071	0.5236 ± 0.0370	0.5188 ± 0.0310	38

3.4.3 Comparison Between the LMFCN and Handcrafted Feature Extractors

This section compares our approach with three handcrafted feature extractors, i.e., LBP, GLCM, and PFTAS. We evaluate balanced accuracy values, complexity measures, and UMAPs. The comparison aims to highlight the adaptability of the LMFCN facing the other approaches commonly used in texture datasets. We also show that our approach performs well compared to native texture feature extractors, pointing out that it is a competitive solution to texture classification problems.

For a fair comparison, the LMFCN approach uses a 59-dimensional latent representation, which has the same dimensionality as the representation produced by the LBP with uniform patterns. The other techniques, PFTAS and GLCM, use 162-dimensional and 169-dimensional representations, respectively. With a 59-dimensional latent representation, the LMFCN already

achieves high accuracy, which is higher than the others. Thus latent representations with higher dimensions are unnecessary. We replicated the values for the handcrafted feature extractors for 15 epochs to ease the comparisons in the graphs. A vertical dotted line in the graphs identifies the epoch where the LMFCN achieved the best-balanced accuracy on the validation set. We used an RBF SVM with the same hyperparameters in all these experiments, even with the LMFCN. The γ is given by $1/(\phi \times \sigma^2)$, where σ^2 is the variance of the latent representation of all training instances, and C=1000. The γ value is calculated according to the characteristics of each latent representation adapting to each feature extractor. We performed a grid search cross-validation varying C and found that C=1000 was the best value for all methods. With PFTAS, C=1000 and C=100 led to very close values of balanced accuracy. The grid search cross-validation found that 100 was the best value, but using all the hold-outs in the experiments, 1000 was slightly better, so we fixed 1000 for all methods. One fact that allowed the same C is the adaptable γ. Please, refer to Table 3.1, which summarizes the meaning of each measure that was described in Section 3.3.2 to interpret the complexity measure graphics.

The classification task with the BACH dataset is challenging, given the difference in the training and test accuracies. The SVM trained with PFTAS achieved around 100% accuracy in the training set, whereas only 77% on the test set, as shown in Figure 3.14. This performance gap between training and test sets also happens with the SVM trained with LBP, meaning a poor generalization. Part of the difficulty in this dataset is due to its size. It has a few large images. Although smaller, the performance gap between training and test is also present in the LMFCN, but with a better generalization.

Interestingly, the PFTAS and the LMFCN produced more similar representations for all complexity measures than the other two methods. The LBP generated features with less separation (high N1 and N2 values). Still, a low T1 value indicates more clustered samples, making generalization more complicated than the other methods with higher test accuracy.

For the BreaKHis dataset, we used 59-dimensional and 162-dimensional latent representations in the LMFCN. We chose to use more dimensions due to the slight advantage of our approach,

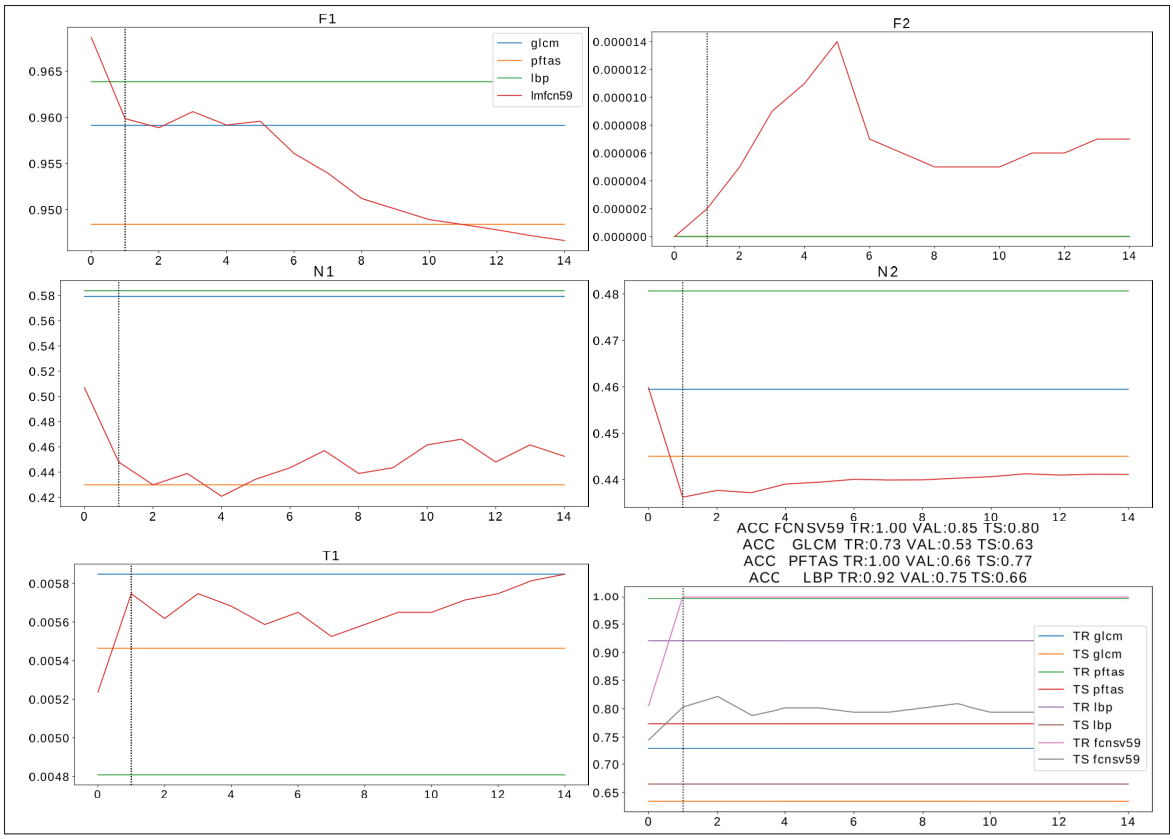


Figure 3.14 Comparison between the LMFCN and handcrafted feature extractors on the BACH dataset.

with 59 dimensions, over the PFTAS. One can notice the difference in the test set accuracy values between the two dimensions of the latent representation in the LMFCN in Figure 3.15. Regarding complexity measures, both latent representations obtained similar values, mostly on N1 and N2 measures. The LMFCN reached smaller values in these two measures than in the other approaches. Although the BACH dataset was more challenging than the BreaKHis dataset, complexities for the LBP behaved akin in both datasets, including the difficulty in generalization. The GLCM presented stable training and test accuracy values, but the measures' values were varied, e.g., high N1 for BACH and medium for BreakHis and low for the Gaussian images dataset, when compared to other approaches.

The two-class version of the CRC dataset posed no difficulty to the SVMs trained with handcrafted features that reached high balanced accuracy, as shown in Figure 3.16. The LBP was the only

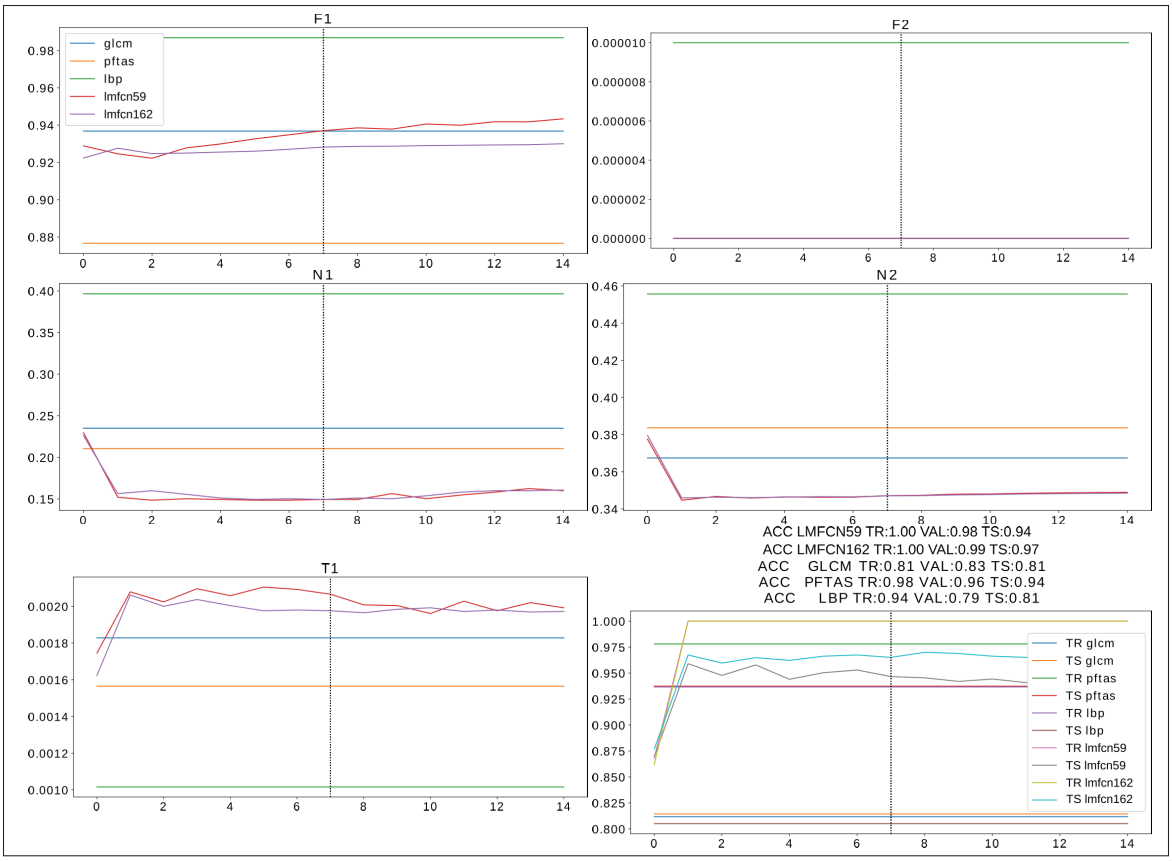


Figure 3.15 Comparison between the LMFCN and shallow approaches that use handcrafted feature extractors on the BreaKHis dataset.

approach that showed a reduced generalization power, and its complexity measures revealed behavior analogous to the ones in the BreaKHis dataset. The LBP was the method with the low T1 measure, indicating clustering. Given the ease in the classification task on this dataset and the wide latent representation, the F2 measure achieved zero with all methods, so there is at least one feature with no overlapping bounding boxes between classes. Here, the LMFCN used a 59-dimensional latent representation, posing less burden to the SVM classifier than the PFTAS, which achieved a similar accuracy but with a 162-dimensional feature vector.

With a slightly better generalization, the LMFCN achieved training accuracy equivalent to the SVM trained with LBP, as seen in Figure 3.17 for the Salzburg dataset. As in other datasets, the SVM trained with LBP reached a smaller T1 measure than the other methods. On the other

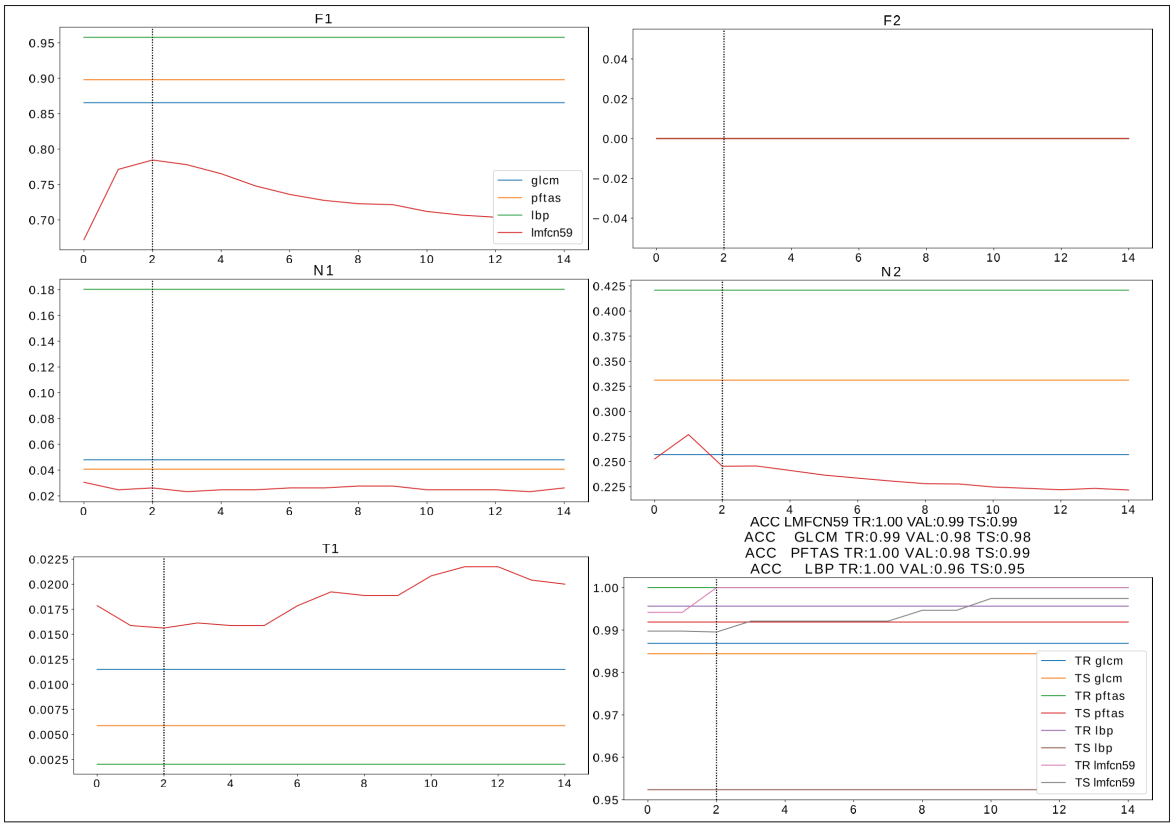


Figure 3.16 Comparison between the LMFCN and shallow approaches with handcrafted feature extractors on the CRC dataset.

hand, its N1 value is smaller than other methods and smaller than the other datasets. This last value means more class separation, which helped the generalization and achieved better accuracy performance than the SVM trained with PFTAS. The behavior of the LMFCN over training epochs was consistent on the Salzburg dataset, analogous to the other datasets, presenting low N1 and N2 and high T1 measures. The LMFCN achieved an accuracy performance superior to the other methods.

Finally, Figure 3.18 shows that amidst handcrafted feature extractors, the SVM trained with GLCM reached higher accuracy and good generalization on synthetic images, contrasting to what took place in other experiments. The LMFCN performed well, with low N1 and N2 measures, although higher than the GLCM, which explains the high accuracy of both methods.

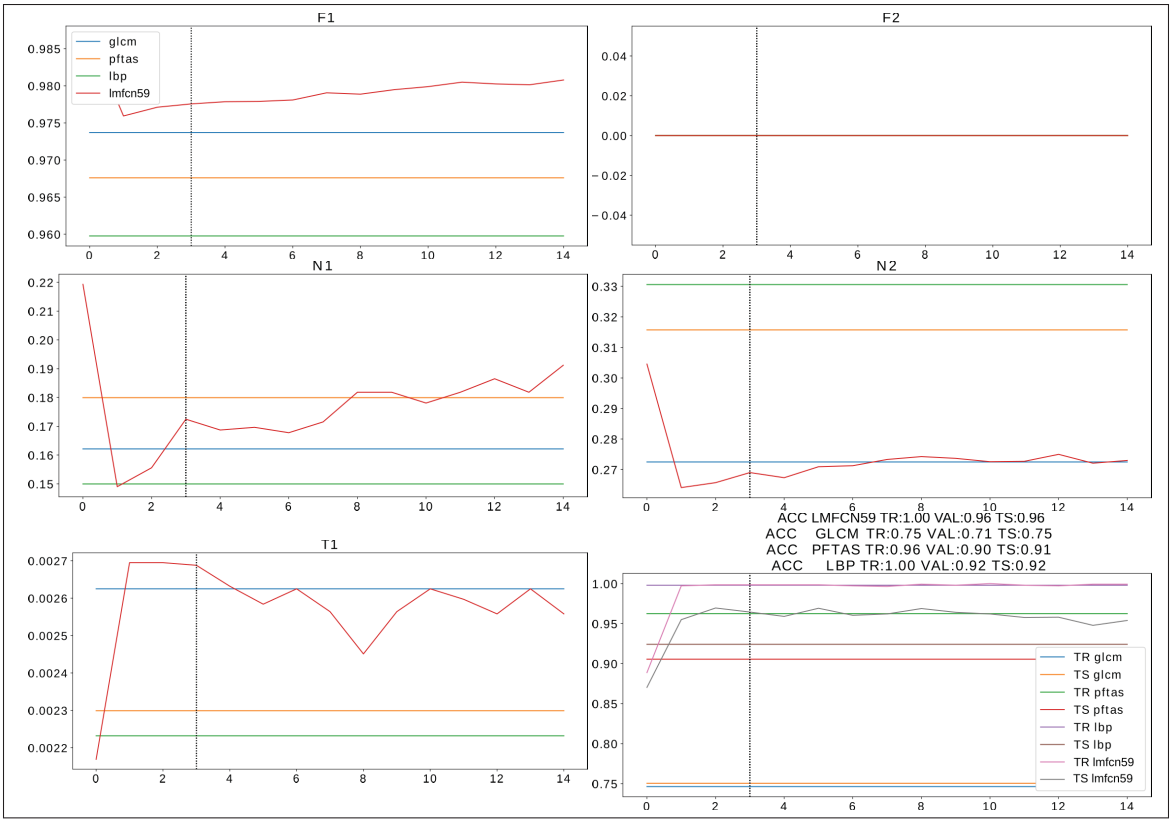


Figure 3.17 Comparison between the LMFCN and shallow approaches that use handcrafted feature extractors on the Salzburg dataset.

The PFTAS had its worst N1 and N2 measures in this dataset, and even with a low F2 measure, it could not overcome the class separation scarcity pointed out by network measures.

The complexity measures showed that each handcrafted feature extractor generated representations with different characteristics across the datasets, so despite the textural context of the datasets, the features do not share similar measures for all datasets with the same method. These feature extractors need to be chosen accordingly to each problem. The variability in the complexity measures forces the discriminant to adapt its capabilities to classify the data. Thus, trying combinations of each feature extractor with a set of discriminants is interesting to discover the best pair (feature extractor and discriminant) to classify the data. It is also important to point out that comparing both N1 and N2 of the LMFCN and the other feature extractors, these measures are reduced during the training, indicating that the latent representations adapt to

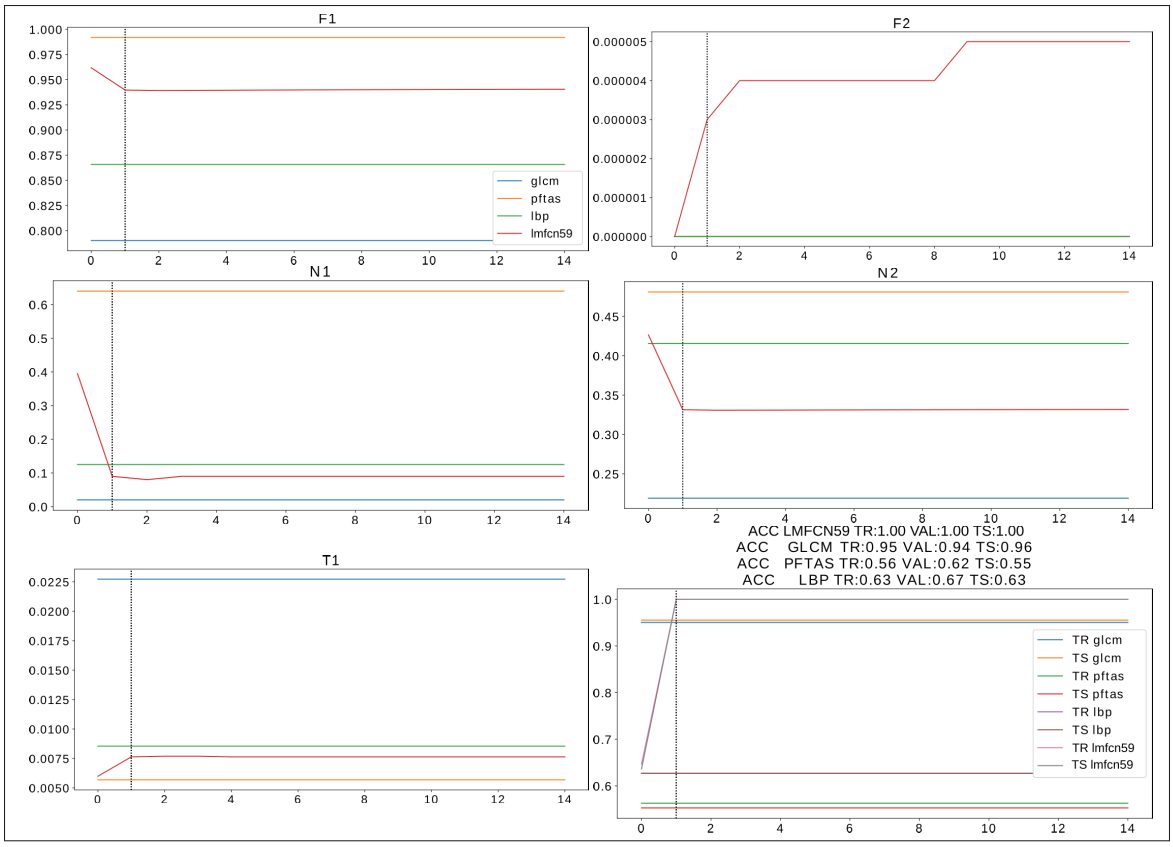


Figure 3.18 Comparison between the LMFCN and shallow approaches that use handcrafted feature extractors for synthetic images generated from Gaussian distributions.

better suit the discriminant. It is also noticeable that these two measures for the handcrafted feature extractors are higher in most cases than those of the LMFCN.

Table 3.4 shows the average balanced accuracy of five repetitions comparing shallow methods trained on handcrafted features and our approach, underlining the best values. The SVM trained with PFTAS was the best shallow method in three datasets (BACH, BreaKHis, and CRC). The SVM trained with LBP was the best in the Salzburg dataset, and the SVM trained with GLCM in the synthetic image dataset. The LMFCN outperformed all the traditional methods, even with the lowest dimensional latent representation of 59 dimensions, indicating that our approach is more adaptable to the problems' particularities.

Table 3.4 Balanced accuracy for the LMFCN and SVMs trained with GLCM, LBP, and PFTAS on five datasets.

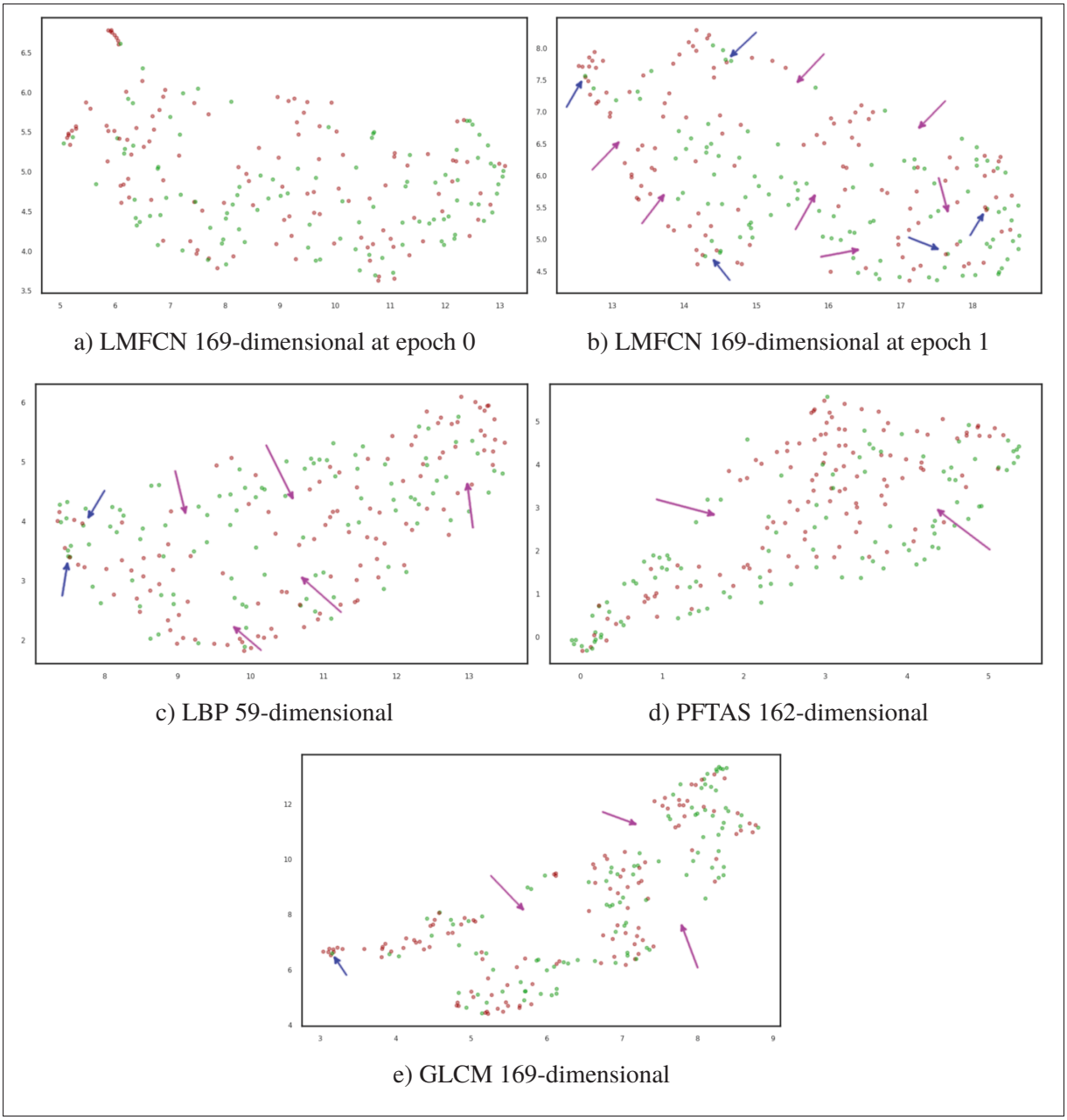
Dataset	Method	Train	Validation	Test
BACH	GLCM	0.7346 ± 0.0199	0.6309 ± 0.0346	0.6618 ± 0.0711
	LBP	0.9146 ± 0.0154	0.7455 ± 0.0442	0.7005 ± 0.0228
	PFTAS	0.9899 ± 0.0037	0.7372 ± 0.0634	0.7608 ± 0.0361
	LMFCN59	0.9664 ± 0.0751	0.7888 ± 0.0577	0.7684 ± 0.0435
	LMFCN162	1.0000 ± 0.000	0.8068 ± 0.0600	0.7934 ± 0.0536
BreaKHis	GLCM	0.8128 ± 0.0086	0.8117 ± 0.0375	0.8076 ± 0.0071
	LBP	0.9292 ± 0.0059	0.7998 ± 0.0099	0.7925 ± 0.0098
	PFTAS	0.9801 ± 0.0034	0.9237 ± 0.0236	0.9145 ± 0.0168
	LMFCN59	1.0000 ± 0.0000	0.9639 ± 0.0247	0.9476 ± 0.0075
	LMFCN162	1.0000 ± 0.0000	0.9666 ± 0.0183	0.9595 ± 0.0050
CRC	GLCM	0.9864 ± 0.0036	0.9828 ± 0.0055	0.9853 ± 0.0101
	LBP	0.9974 ± 0.0012	0.9431 ± 0.0099	0.9479 ± 0.0061
	PFTAS	1.0000 ± 0.0000	0.9852 ± 0.0025	0.9872 ± 0.0097
	LMFCN59	0.9977 ± 0.0022	0.9937 ± 0.0023	0.9883 ± 0.0084
Salzburg	GLCM	0.7706 ± 0.0163	0.7257 ± 0.0186	0.7338 ± 0.0142
	LBP	0.9987 ± 0.0012	0.9282 ± 0.0209	0.9116 ± 0.0208
	PFTAS	0.9650 ± 0.0051	0.9096 ± 0.0124	0.8973 ± 0.0137
	LMFCN59	0.9994 ± 0.0006	0.9672 ± 0.0112	0.9500 ± 0.0180
Gaussian	GLCM	0.9583 ± 0.0028	0.9327 ± 0.0110	0.9527 ± 0.0191
	LBP	0.6303 ± 0.0091	0.6710 ± 0.0131	0.6109 ± 0.0075
	PFTAS	0.5578 ± 0.0080	0.5953 ± 0.0077	0.5608 ± 0.0076
	LMFCN59	1.0000 ± 0.0000	0.9354 ± 0.1254	0.9880 ± 0.0121

3.4.4 UMAPs for the LMFCN and Shallow Methods

The UMAPs were generated with the latent representation of the LMFCN and with the feature vectors generated by the three handcrafted feature extraction methods (GLCM, LBP, PFTAS) for the training set of the five datasets. Each handcrafted feature extraction method generated a UMAP scatter plot that shows the dispersion of the data in a low-dimensional representation space, helping us to figure out the class separation and margin between classes. The values (coordinates) of x and y axes result from reducing the dimensionality of the original representation space. They do not have an absolute meaning. For the LMFCN, we used two graphs, one created with the initial latent representation (FCN weights initialized randomly) and the other with the latent representation of the epoch where the LMFCN obtained the best training accuracy. We

used the training information (accuracy and latent representation) with UMAPS to illustrate the effect of the training procedure in the latent representation, increasing the margins and comparing the difference between features from handcrafted feature extractors and their effect on the classifier results. Using the test dataset could not represent the filter changes pursuing the large-margin discriminant needs. All UMAPs used the following parameters, as discussed in Section 3.3.3: Euclidean distance, a minimum distance equal to 0.05, the number of components equal to 2, and the number of neighbors equal to 40% of the number of instances. Figures 3.19 to 3.23 show the scatter plots for all methods and datasets. The scatter plots show the dispersion of the representations of two classes (red and green). Since the UMAPS building relies on keeping topological characteristics of the original representation, mainly regarding the inter-instance distances and neighborhood, the scatter plots hint at the relationship between instances. Moreover, the geometric relation between instances is linked to the proposed loss function since such a function induces an increase in the margins and a discriminant that separates well instances of different classes.

Features generated by LBP and PFTAS for the BACH dataset (Figures 3.19c and 3.19d) produced similar UMAP representations when looking at the big picture, both representations are more dispersed. Training accuracy is 75.2% and 99.5% for LBP and PFTAS, respectively. Looking in more detail, LBP presents more red and green overlapping, and the instances are more mixed in the representation space. For PFTAS, there is less overlapping, and there are regions concentrating instances of the same class compared to LBP. Overlapping instances from red and green classes mean the classes are more difficult to discriminate. As UMAPs try to preserve the distances (Euclidean distance) and neighborhood between instances and the SVM kernel is also based on this distance, when there is overlapping in the reduced-dimension representation, the discriminant has more difficulty in classifying the instances. GLCM features (with training accuracy of 72.9%) in Figure 3.19e show the grouping of instances from different classes, but it also has mixed red and green instances, with a small margin. Regarding the LMFCN in Figure 3.19a and 3.19b (with training accuracy going from 80.3% to 100% from epoch zero to one), there are few overlaps, but in epoch zero, red and green classes' instances are more



a) LMFCN 169-dimensional at epoch 0

b) LMFCN 169-dimensional at epoch 1

c) LBP 59-dimensional

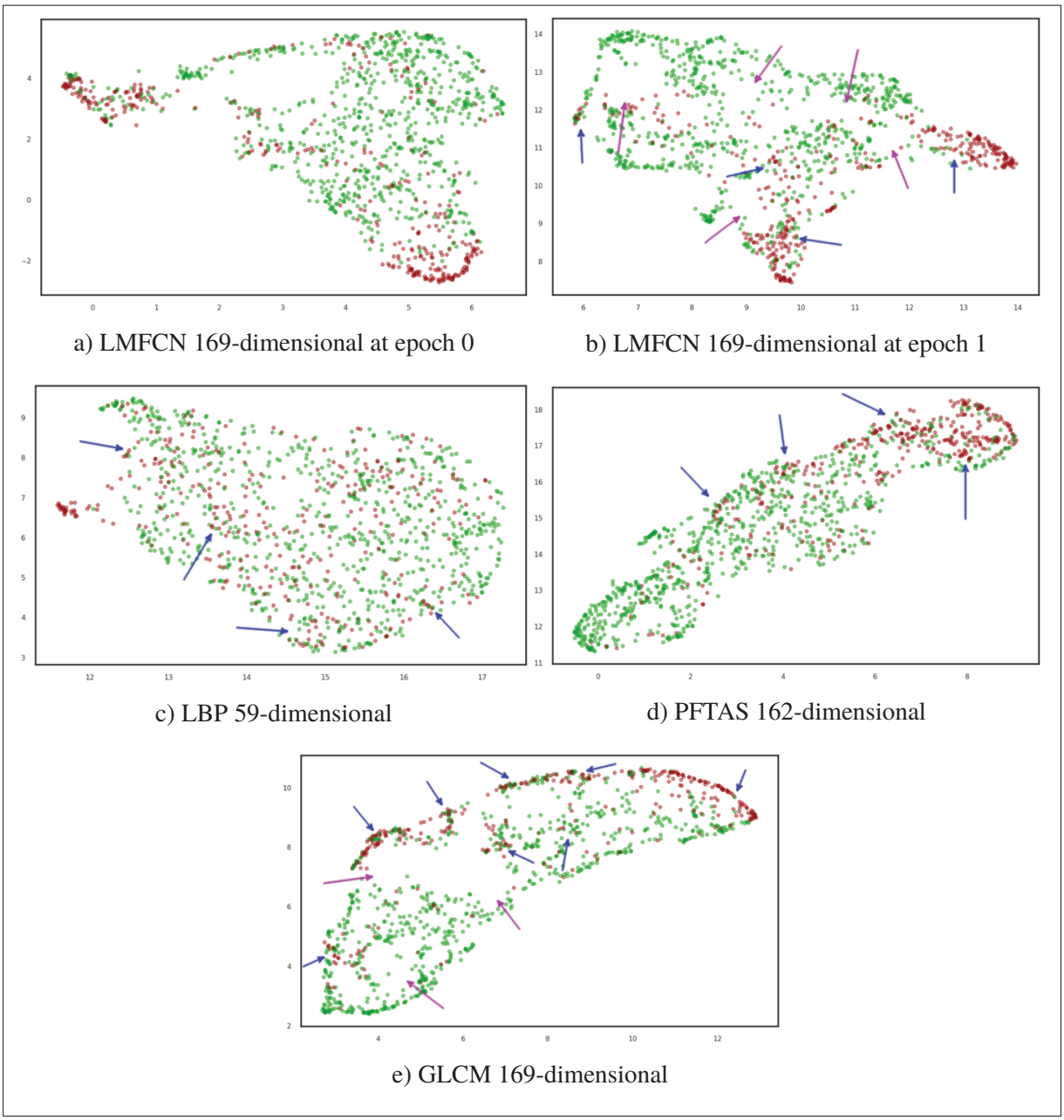
d) PFTAS 162-dimensional

e) GLCM 169-dimensional

Figure 3.19 UMAPs generated with features from three handcrafted methods and LMFCN for the BACH dataset. Axes x and y represent the two dimensions of the latent representation reduced by the UMAP method. Blue arrows point to regions with instance overlapping. Magenta arrows point to regions with more separation.

dispersed and mixed. In epoch one, small clusters are formed, with more distance between

classes in some parts of the representation space. This behavior is compatible with our loss function.



a) LMFCN 169-dimensional at epoch 0

b) LMFCN 169-dimensional at epoch 1

c) LBP 59-dimensional

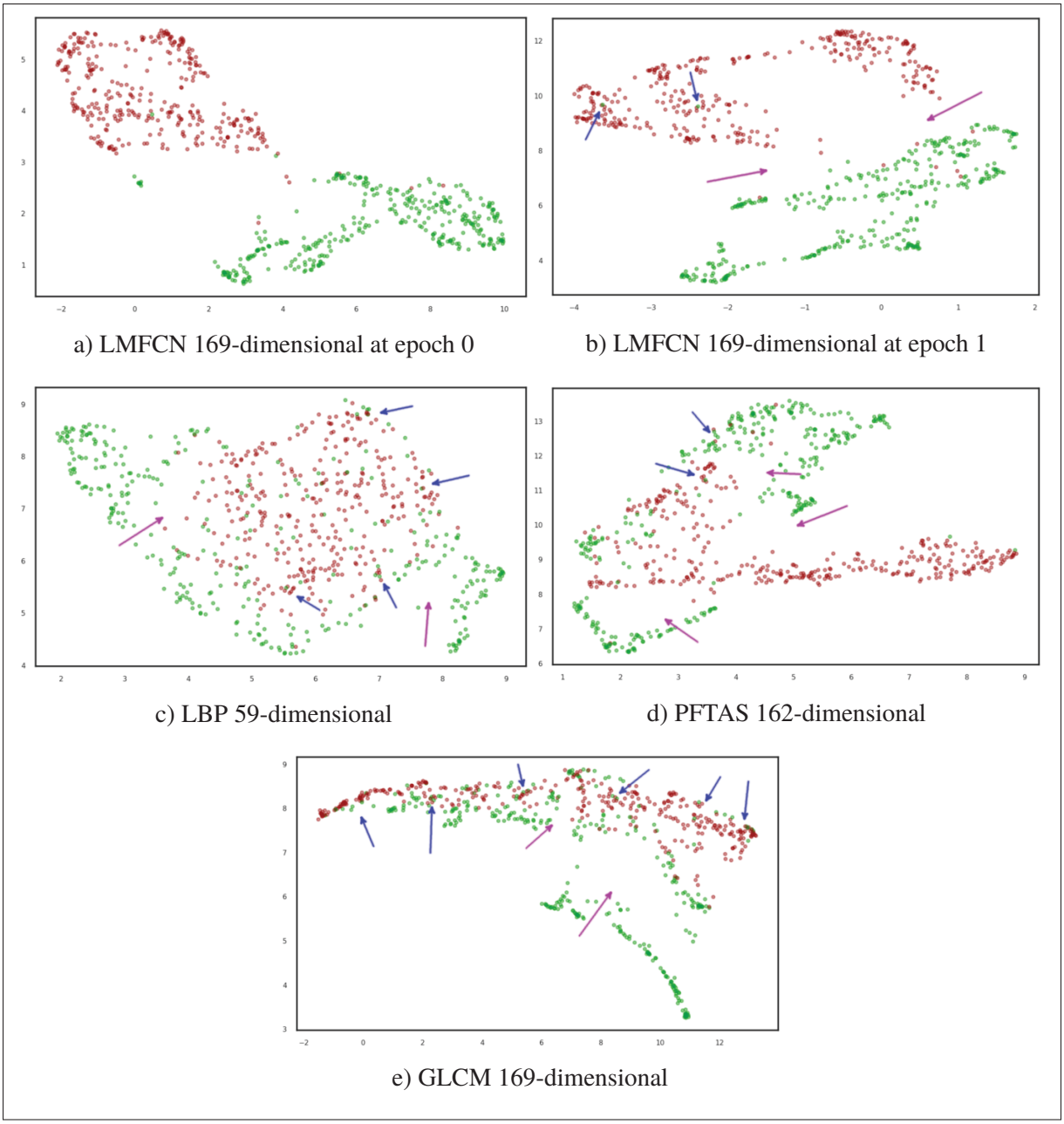
d) PFTAS 162-dimensional

e) GLCM 169-dimensional

Figure 3.20 UMAPs generated with features from three handcrafted methods and LMFCN for the BreaKHis dataset. Horizontal and vertical axes represent the two dimensions of the latent representation reduced by the UMAP method. Blue arrows point to regions with instance overlapping. Magenta arrows point to regions with more separation.

Figure 3.20 shows the UMAPs generated for the BreaKHis dataset. The LBP features, which obtained a training accuracy of 93.6%, demonstrated in Figure 3.20c some overlapping instances of red and green classes. It also shows samples distributed uniformly, mixed, and without clusters in the representation space. Mixed instances, distributed uniformly, are inadequate for the SVM classifier because they will require more SVs to cover each small region in the representation space. It will also have no large margins. The GLCM features (Figure 3.20e) obtained a training accuracy of 81.1%. They are more densely populated on the boundaries of the distribution. The density and distribution, with some clustering, are similar to PFTAS (Figure 3.20d), but the last one obtained a training accuracy of 97.8%. The most significant difference between the representation of both methods is the strong overlapping in the upper border of the space. Overlapping makes it more difficult for the SVM to classify instances. The difference in LMFCN representations from epoch zero, in Figure 3.20a (accuracy of 88.8%), to epoch one, in Figure 3.20b (accuracy of 100%), shows more clustering of green instances after training. There is a concentration of these instances in some points and more blank spaces (margins) in the middle. The representation is similar to the one from PFTAS in terms of overlapping and clustering, which is compatible with the high training accuracy of both methods, but with more blank spaces for LMFCN. The margins in the center for LMFCN resulted in better test accuracy (93.7% of PFTAS against 97.6% of LMFCN). The UMAPs of LMFCN at epoch zero and one are mirrored, with the red instances in the left corner (epoch zero) plotted on the right at epoch one. The UMAPs generation of epoch zero and one are not linked or share information, so there is no commitment that each representation will place instances in the space in a similar absolute position (considering the representation components as vertical and horizontal coordinates in the scatter plot). The instances position at each UMAP is absolute and not linked to the previous epoch. However, the instance's distances and dispersion are correlated between epochs because the UMAP algorithms focus on this information.

The two-class version of the CRC dataset is one of the most straightforward scenarios in our experiments. The representation generated by the LMFCN depicted this fact on the UMAP plots in Figures 3.21a and 3.21b, with red and green classes' instances well separated even



a) LMFCN 169-dimensional at epoch 0

b) LMFCN 169-dimensional at epoch 1

c) LBP 59-dimensional

d) PFTAS 162-dimensional

e) GLCM 169-dimensional

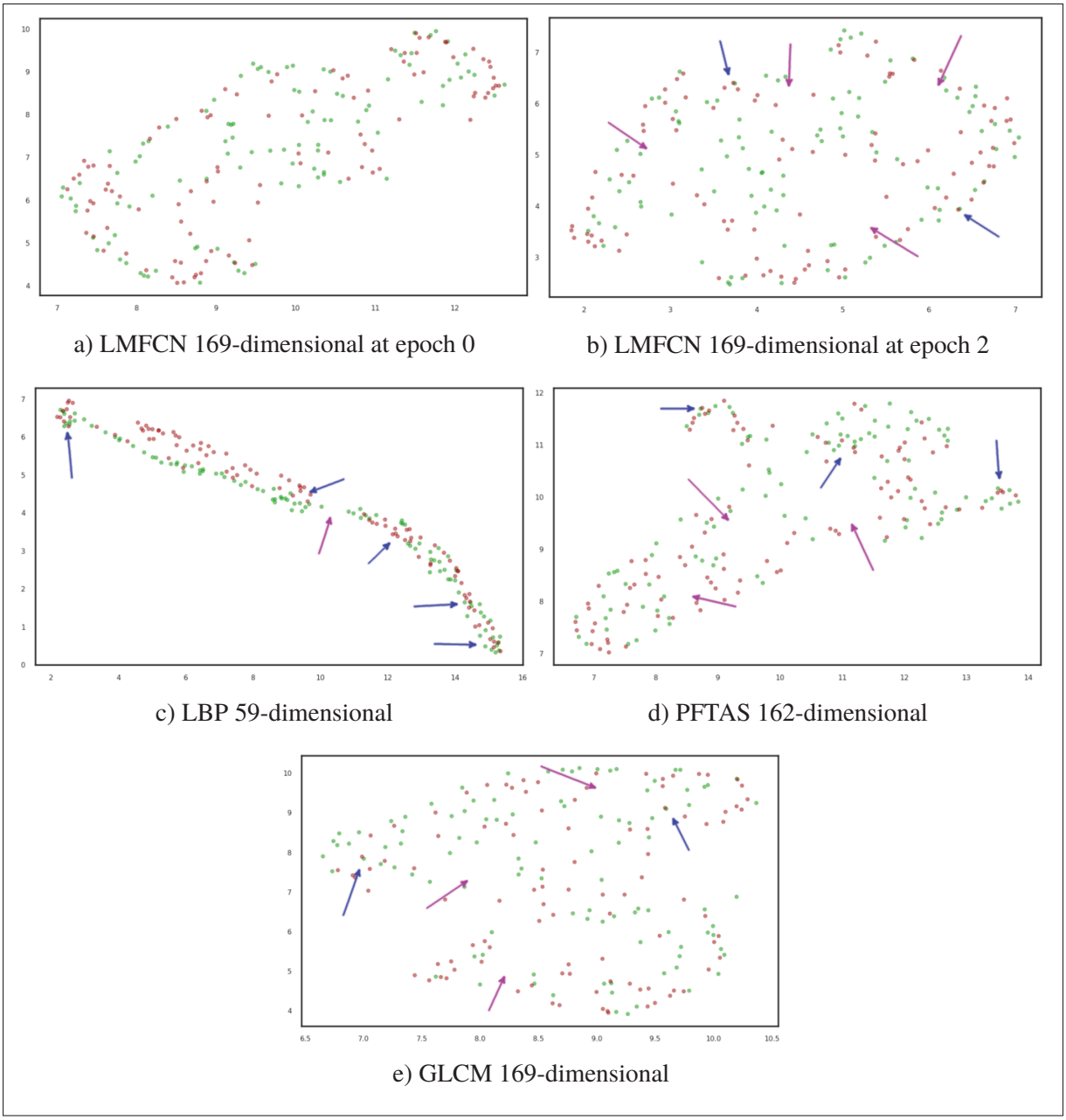
Figure 3.21 UMAP generated with features from three handcrafted methods and the LMFCN for the CRC dataset. Horizontal and vertical axes represent the two dimensions of the latent representation reduced by the UMAP method. Blue arrows point to regions with instance overlapping. Magenta arrows point to regions with more separation.

in the epoch zero. The training accuracy went from 99.5% to 100% from epoch zero to one.

The PFTAS features (Figure 3.21d) do not present the same level of instance separation in the UMAPs as the LMFCN. Still, it achieved highly training balanced accuracy (100%) due to the RBF SVM capabilities of drawing complex margins. We also need to emphasize that UMAPs represent a reduced space, so the overlapping in PFTAS representation may not pose a considerable challenge to SVM in the higher dimension (162-dimensional), and the kernel increased dimensionality. Finally, there is a slight advantage for the LBP features (Figure 3.21c) compared to the GLCM features (Figure 3.21e) regarding balanced accuracy (99.5% and 98.6%, respectively). The clustered or compacted instances in GLCM make it more challenging to separate the classes. The instances in the UMAP generated from the LBP features are more spread, like in the other datasets, but they are not too mixed. The test accuracy of LBP was 95.2% and 98.4% for GLCM, showing that the mixed instances in the middle of the representation cause more problems in generalization due to reduced margins.

In the synthetic image dataset, LBP and PFTAS (with training accuracy of 89.9% and 58.3% respectively) produced red and green overlapped instances with more clustering in the LBP features, as seen in Figure 3.22c. The UMAP produced by the GLCM features, which obtained 61.3% of training accuracy, showed homogeneous spacing between instances with few overlapping. Still, the red and green instances are mixed, hindering the establishment of decision boundaries. The UMAP produced by the LMFCN in Figures 3.22a and 3.22b shows the difference between epochs zero and two with improving in accuracy from 74.9% to 98.9%. Instances of opposite classes (red and green) have few overlaps, and the spacing between instances is improved, with the formation of bigger white spaces in the middle when compared to epoch zero, which allowed improvement in test accuracy from 71.3% to 91.9% due to the possibility of establishment of larger margins.

The scatter plots of UMAPs in Figure 3.23 show instances very close to each other, even from opposite classes (red and green) for the GLCM (Figure 3.23e) that hinder the decision boundary definition of the SVM, producing low balanced training accuracy on the Salzburg dataset of 77.8%. The LMFCN increased the clustering of instances and created spaces between clusters of instances from epoch zero to one, improving training accuracy from 96.7% to 100%. The



a) LMFCN 169-dimensional at epoch 0

b) LMFCN 169-dimensional at epoch 2

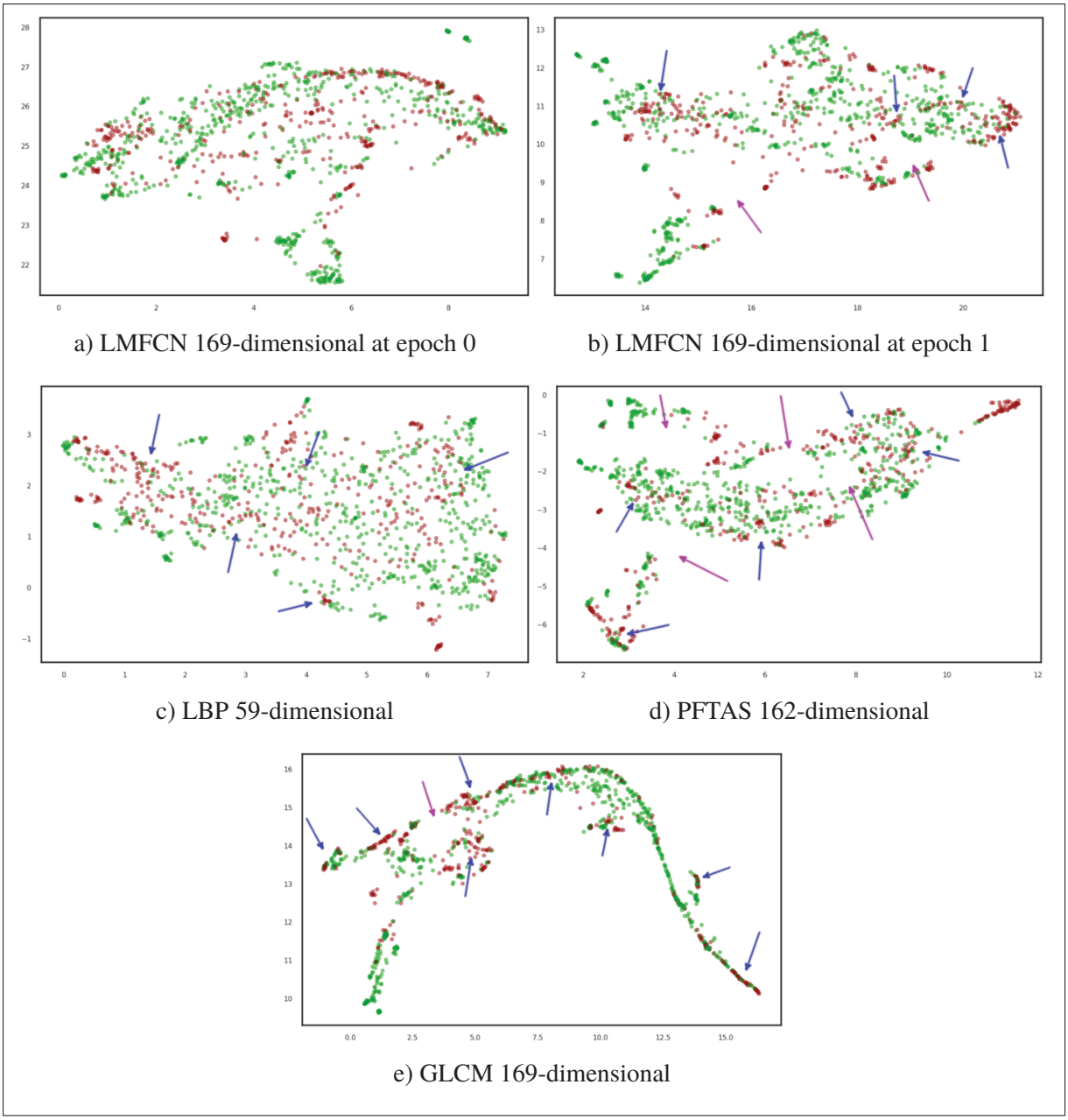
c) LBP 59-dimensional

d) PFTAS 162-dimensional

e) GLCM 169-dimensional

Figure 3.22 UMAP generated with features from three handcrafted methods and LMFCN for the synthetic Gaussian image dataset. Horizontal and vertical axes represent the two dimensions of the latent representation reduced by the UMAP method. Blue arrows point to regions with instance overlapping. Magenta arrows point to regions with more separation.

grouping and empty areas between them enlarge the margins and promote more generalization capacity. The PFTAS (with a training accuracy of 99.1%) generated a feature distribution akin



a) LMFCN 169-dimensional at epoch 0

b) LMFCN 169-dimensional at epoch 1

c) LBP 59-dimensional

d) PFTAS 162-dimensional

e) GLCM 169-dimensional

Figure 3.23 UMAP generated with features from three handcrafted methods and the LMFCN for the Salzburg dataset. Horizontal and vertical axes represent the two dimensions of the latent representation reduced by the UMAP method. Blue arrows point to regions with instance overlapping. Magenta arrows point to regions with more separation.

to the LMFCN, grouping same-class instances but with more aggregation. The LBP, which achieved the best-balanced accuracy performance among the handcrafted feature extractors

in this dataset, 99.7%, produced an instance distribution more dispersed, with less clustering than LMFCN or PFTAS. Although dispersed instances in the UMAP in the other datasets had reduced generalization capacity, in this case, the red and green instances are not too equally spaced, not so mixed, and with more clustering, despite less than in the LMFCN and PFTAS.

The visual representation provided by the UMAPs contributes to discovering the difference between the latent representation (from the LMFCN) and feature representations (from PFTAS, GLCM, and LBP). In the UMAPs, we can see that there is high between-class overlapping in some experiments, and in these cases, the balanced accuracy was also negatively affected. The LMFCN showed low overlapping or dispersed instances but produced good results in both situations, unlike other feature representations that did not perform satisfactorily in all cases. We notice the adaptability of the LMFCN in various conditions, for example, its UMAPs and results for the synthetic image and BreaKHis datasets. The LMFCN performed well on both datasets generating UMAPs with distinct characteristics, contrasting with the feature representations produced by GLCM and PFTAS. The GLCM performed well on synthetic images and BreaKHis datasets, while the PFTAS performed well only on the BreaKHis dataset. The LMFCN and GLCM produced spread instances on the synthetic image dataset. In contrast, in the BreaKHis dataset, the PFTAS and LMFCN performed best, had a compact latent representation with some clustering and less between-class overlapping than the other methods.

3.4.5 Results for Multiclass Experiments

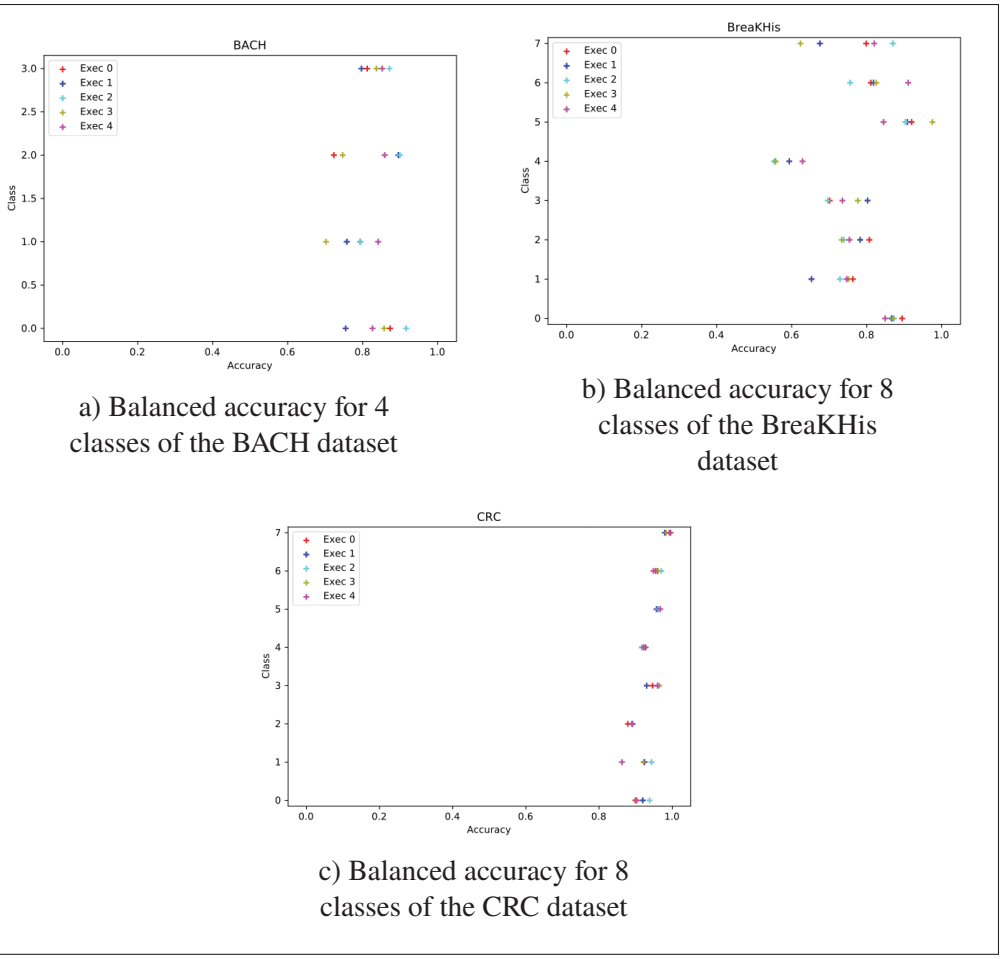
We only used the three HI datasets (BACH, BreaKHis, and CRC) in the multiclass experiments. The synthetic image dataset was used to illustrate the training process of the proposed method, and we designed it as a two-class dataset. The Salzburg dataset initially has 32 classes, placing it outside our method's scope due to the many categories. Our multiclass approach uses the OVA strategy, which requires n_{classes} FCNs and an extended latent representation made up of the concatenation of n_{classes} latent representations. The computational cost becomes prohibitively high when the number of classes grows too much. Small datasets with many classes are a

problem suitable to DML or other dissimilarity-based methods. Thus the LMFCN is not focused in this scenario.

The multiclass LMFCN is compared to a TCNN with similar architecture, with the same number of layers, but with more channels. We increased the number of channels to provide more weights to learn the multiclass representation, given that the problem with more classes is more challenging. The number of layers was preserved to prevent the TCNN from learning more spatial and shape characteristics. In the LMFCN, each FCN learns a latent representation and discriminant to distinguish one class from the rest. Thus, using a TCNN with a number of parameters equivalent to one of the FCNs from LMFCN would lead to an unfair comparison. The TCNN would have less representation capacity with fewer parameters because LMFCN would have the number of parameters of the FCN multiplied by $n_{classes}$. Therefore, the number of TCNN parameters per layer was increased proportionally to the number of classes. Our experiments also compared the LMFCN with the handcrafted feature extractors GLCM, LBP, and PFTAS.

We used two pre-trained CNNs (ResNet18 and InceptionV3) as feature extractors with an SVM classifier to analyze their performance compared to LMFCN in the multiclass scenario. The SVM's C and γ parameters were 1000 and 'scale', respectively. As a further analysis, we also used ResNet18, SqueezeNet, and DenseNet101 pre-trained CNNs with two additional FC layers to adapt their output to the number of classes of each dataset. We trained them for 400 epochs with their first layers frozen, fine-tuning the last additional layers.

The multiclass LMFCN encompasses multiple sub-models, each comprised of an FCN and an SVM so that their performance can be analyzed individually. The accuracy of these sub-models can provide valuable information regarding the potential of our approach in imbalanced scenarios. The OVA strategy creates multiple two-class imbalanced problems since one class has instances of a single class, and the other concentrates the instances of all other classes. We plot the balanced accuracy graphs for each sub-problem for five repetitions of the HI datasets. These plots are presented in Figures 3.24a, 3.24b, and 3.24c.



a) Balanced accuracy for 4 classes of the BACH dataset

b) Balanced accuracy for 8 classes of the BreaKHis dataset

c) Balanced accuracy for 8 classes of the CRC dataset

Figure 3.24 Balanced accuracy for each class of the three HI datasets.

The two-class sub-problems of the BreaKHis dataset presented a smaller balanced accuracy amidst our experiments. Both CRC and BACH datasets are balanced, minimizing the impact of data imbalance. The BreaKHis dataset is imbalanced in its two ways of analysis, binary and multiclass. The worst case was in class four (Phyllodes Tumor) against all other categories. This class has few images (71), creating a heavily imbalanced scenario. Class zero (Ductal Carcinoma) is the most prevalent, with 504 samples on the training set. Even with the imbalance, the LMFCN approach produced good models.

Finally, the balanced accuracy results were compiled by comparing all methods with the LMFCN in Table 3.5, where the LMFCN's superiority over all others is noticeable. In the case of the

CNN experiments, 400 training epochs were used. For the LMFCN, we let each sub-model train for ten epochs. The ResNet18 showed overfitting in the feature extraction experiment with high training accuracy. The DenseNet showed higher training accuracy for the CRC dataset than other methods and was also the best test accuracy in this dataset among the other methods, excluding the LMFCN.

Table 3.5 Balanced accuracy and standard deviation for five runs in multiclass experiments on three HI datasets. The best results for each dataset and data partition are underlined. The three CNNs ResNet18, SqueezeNet, and DenseNet, followed by the 'FT' text, identify their use as CNN with fine-tuning.

Dataset	Method	Train	Validation	Test
BACH	LMFCN	0.9690 ± 0.0348	0.7170 ± 0.0210	0.6854 ± 0.0278
	CNN	0.8390 ± 0.0184	0.6788 ± 0.0406	0.6444 ± 0.0187
	GLCM	0.4081 ± 0.0183	0.4003 ± 0.0753	0.3637 ± 0.0303
	LBP	0.4643 ± 0.0334	0.4871 ± 0.0729	0.3951 ± 0.0896
	PFTAS	0.6229 ± 0.0146	0.5690 ± 0.0464	0.6043 ± 0.0235
	ResNet18	1.0000 ± 0.0000	0.6056 ± 0.0578	0.5671 ± 0.0300
	InceptionV3	0.4628 ± 0.0807	0.4269 ± 0.0904	0.4057 ± 0.0261
	ResNet18 FT	0.6721 ± 0.0855	0.5513 ± 0.0078	0.4668 ± 0.0575
	DenseNet FT	0.9524 ± 0.0843	0.6220 ± 0.0453	0.4842 ± 0.0502
	SqueezeNet FT	0.5692 ± 0.1180	0.6238 ± 0.0676	0.5462 ± 0.0681
BreaKHis	LMFCN	0.9688 ± 0.0285	0.8125 ± 0.0321	0.7895 ± 0.0162
	CNN	0.8032 ± 0.0319	0.7347 ± 0.0238	0.7040 ± 0.0307
	GLCM	0.4697 ± 0.0111	0.4218 ± 0.0212	0.4091 ± 0.0050
	LBP	0.9235 ± 0.0084	0.5373 ± 0.0174	0.5218 ± 0.0147
	PFTAS	0.9537 ± 0.0086	0.6643 ± 0.0109	0.6768 ± 0.0147
	ResNet18	1.0000 ± 0.0000	0.5933 ± 0.0341	0.5834 ± 0.0213
	InceptionV3	0.1848 ± 0.0182	0.1745 ± 0.0194	0.1767 ± 0.0123
	ResNet18 FT	0.3496 ± 0.0633	0.4055 ± 0.0343	0.3546 ± 0.0182
	DenseNet FT	0.8582 ± 0.1184	0.5561 ± 0.0475	0.5194 ± 0.0655
	SqueezeNet FT	0.3038 ± 0.0652	0.3689 ± 0.0108	0.3589 ± 0.0186
CRC	LMFCN	0.9834 ± 0.0127	0.9379 ± 0.0065	0.9338 ± 0.0073
	CNN	0.7209 ± 0.2614	0.3946 ± 0.0453	0.3901 ± 0.0402
	GLCM	0.6062 ± 0.0057	0.5960 ± 0.0069	0.6049 ± 0.0084
	LBP	0.6148 ± 0.0045	0.6086 ± 0.0270	0.6144 ± 0.0123
	PFTAS	0.8359 ± 0.0058	0.8271 ± 0.0152	0.8297 ± 0.0097
	ResNet18	0.9506 ± 0.0033	0.5070 ± 0.0141	0.5066 ± 0.0138
	InceptionV3	0.1509 ± 0.0036	0.1546 ± 0.0075	0.1477 ± 0.0082
	ResNet18 FT	0.7285 ± 0.0189	0.7530 ± 0.0155	0.7460 ± 0.0115
	DenseNet FT	0.9955 ± 0.0063	0.9032 ± 0.0054	0.8933 ± 0.0037
	SqueezeNet FT	0.8233 ± 0.0163	0.8126 ± 0.0190	0.8041 ± 0.0112

3.4.6 Computational Complexity Analysis

This section presents the computational costs using execution times of the LMFCN training with the BreaKHis dataset. Although execution time is not a formal way to analyze a method's complexity, it can give us an idea of how it performs. The execution time depends on several variables, such as the hardware characteristics, operating system, computer language, and frameworks. For these experiments, we used a machine with an Intel Xeon E5-2620 processor, 64GB of RAM, and a Tesla P100 GPU with 12 GB of VRAM. The operational system was an Ubuntu 20.04.4 LTS, and the source code was written in Python 3.6.13 and Pytorch 1.2.0+cu92.

The LMFCN execution time was compared to a CNN with an equivalent architecture, with the number and dimension of FCN and FC layers similar to the experiments on Section 3.4.2. We chose the two-class BreaKHis dataset because it represents a complex problem, the second most difficult in the previous experiments. A simple dataset with a high accuracy rate would not pose a challenge and would hinder the time evaluation.

The LMFCN, despite its extra calculations related to the large-margin discriminant, is still very competitive against CNN approaches. We executed the LMFCN and a CNN-BCE with all five folds from the dataset to allow the comparison. The performance was analyzed based on the performance of a single fold and the average of all folds. The LMFCN requires a hyperparameter setting. Thus, for these experiments, we used sv_{close}=1, wr_{close}=4, and C=10. These values differ from the previous experiments because the focus here was not to compare the best accuracy but the parameter influence. We compare the hyperparameters combination concerning execution time and accuracy in Tables 3.6, 3.9, and 3.7. To simplify and summarize the presentation of the results, we had to choose one of the combinations to present these experiments. The two first hyperparameters do not significantly impact the execution time. They do not affect the FCN (latent representation creation), matrices calculation, and SMO times. They influence the backpropagation time, where the distance from the instances of interest to the multiple anchors is calculated. The impact is lighter because the anchors' coordinates (latent representation) are pre-calculated at each epoch, stored at matrix $\mathbf{T}$. The backpropagation time depends more on

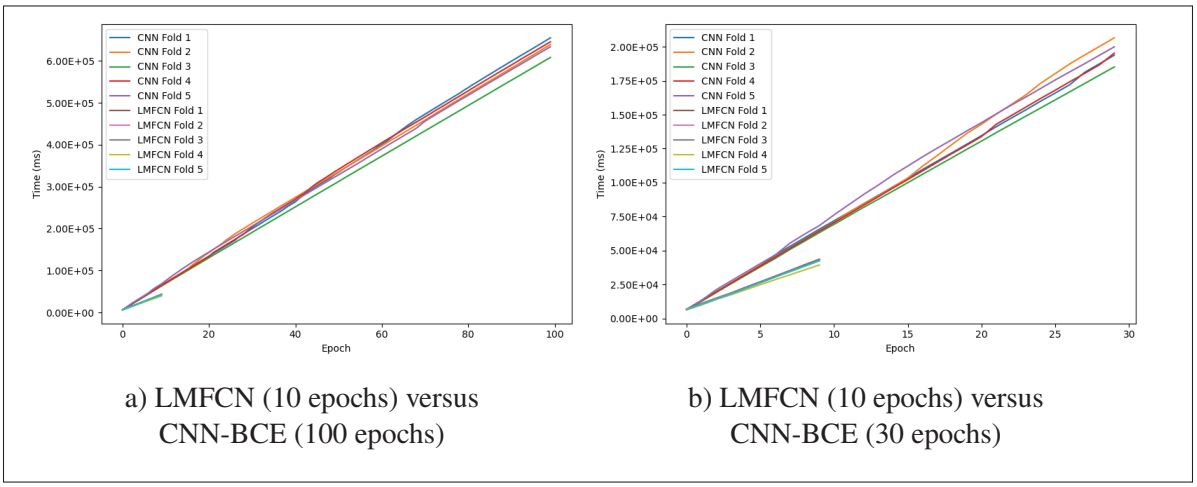
the number of instances of interest directly affecting the gradients' calculation. As an example of the hyperparameters' impact on the execution time, we present Table 3.6, with the total backpropagation for ten epochs of the BreaKHis dataset. It shows the time when varying both hyperparameters, wr_{close} and sv_{close}. There is a subtle difference from $wr_{\text{close}} = 0$ to $wr_{\text{close}} = 1$, because for $wr_{\text{close}} = 0$ the $\mathcal{L}_{mc}$ is not calculated. From $wr_{\text{close}} = 1$ to $wr_{\text{close}} = 2$ the difference is slight, and the difference is even small for the rest of the wr_{close} values. The difference is small for sv_{close}. Thus, to produce a more concise analysis, we fixed the hyperparameters. The hyperparameter C has more impact on the execution time than sv_{close} and wr_{close} because it affects the number of SVs. Its effect is not so deterministic and easy to formulate since it depends on the dataset's complexity. A low C value tends to generate a model more lenient to misclassified training instances, thus with fewer SVs defining a softer decision border. A high C produces more SVs leading to a more contoured decision border and making each LMFCN epoch computationally expensive, even more than the CNN epochs. Although the high cost, the LMFCN tends to converge earlier, so the total time also tends to be smaller than in the CNN. Usually, a C too high may produce as many SVs as training instances, which is not a good characteristic of the discriminant, indicating a high possibility of overfitting. We also set this hyperparameter to an intermediary value defined empirically to avoid overfitting. We limited the CNN execution to 100 epochs and the LMFCN to 10.

Table 3.6 Total backpropagation time (in miliseconds) for 10 epochs impacted by varying wr_{close} and sv_{close} for the dataset BreaKHis

		wr_{close}				
		0	**1**	**2**	**4**	**8**
sv_{close}	**1**	23783.8	25841.2	27494.6	27570.1	27718.6
	2	23901.3	26748.0	27433.5	27489.2	27609.1
	4	24174.1	26130.8	27246.5	27508.8	27085.6
	8	24123.2	26366.9	27278.6	27890.9	27584.5

Figure 3.25a shows the total time for training up to the last epoch, limiting the number of training epochs to 10 and 100 for the LMFCN and the CNN-BCE, respectively. Albeit the LMFCN has extra calculations as the generation of the latent representation, calculation of several matrices,

and SMO, the training time is lower than the CNN. The CNN's backpropagation time is higher than the LMFCN's because fewer instances are used for training by the last one. Reducing the number of training instances compensates for the extra costs of the LMFCN in the face of CNN training time. To ease the visual comparison, we highlight the training time between the first and the 30^{th} epoch in Figure 3.25b.

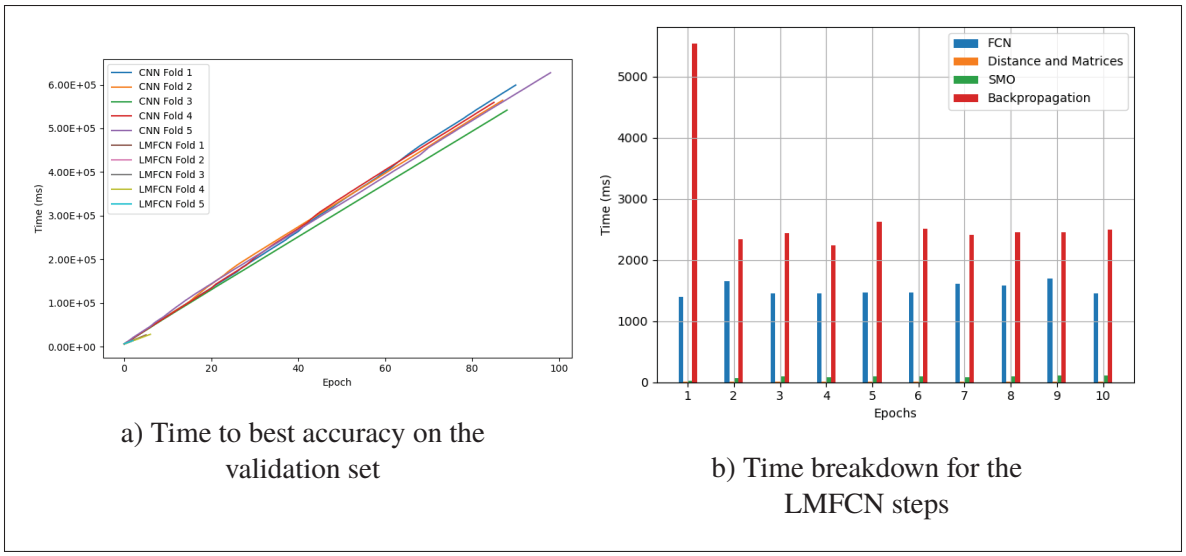


a) LMFCN (10 epochs) versus CNN-BCE (100 epochs)

b) LMFCN (10 epochs) versus CNN-BCE (30 epochs)

Figure 3.25 a) Total execution to 100 epochs for the CNN-BCE and ten epochs for the LMFCN and b) Total execution for only 30 epochs for the CNN-BCE and ten epochs for the LMFCN

Figures 3.25a and 3.25b consider the training time limited by the number of epochs regardless of the accuracy of the validation set. For convergence efficiency comparison, Figure 3.26a shows LMFCN and CNN-BCE runtime to achieve the best validation accuracy. The LMFCN took less than ten epochs to obtain a validation accuracy superior to the CNN-BCE, which took more than 80 epochs. To better understand the time advantage of the LMFCN at each epoch, we present in Figure 3.26b the time each step our approach takes inside one epoch. We used the epoch time for fold 1 of the BreaKHis dataset. The most costly steps are generating the latent representation by the FCN, the backpropagation algorithm, the matrices calculation (kernel and distance), and the execution of the SMO algorithm. The execution time of the SMO increases as the number of SV decreases, and the execution time of the backpropagation algorithm decreases with fewer SVs, so one compensates the other. However, the absolute execution time of the SMO algorithm is much smaller than the execution time of the backpropagation algorithm.



a) Time to best accuracy on the validation set

b) Time breakdown for the LMFCN steps

Figure 3.26 a) Total execution time to achieve the best validation accuracy for both methods (CNN-BCE and LMFCN) and (b) Execution time to each step of the LMFCN (latent representation generation, matrices, distance calculation, SMO, and backpropagation over ten epochs).

Figures 3.27a and 3.27b compare the percentage of the total time taken by each step of the LMFCN in epochs 1 and 2. These complementary representations of Figure 3.26b make it easier to see each step's percentage and the relation between the execution time of the SMO and the backpropagation algorithms.

The performance analysis for the BACH dataset, the most challenging dataset of our experimental setup, showed a different time profile from the BreaKHis dataset. The BACH dataset has fewer samples than the BreaKHis dataset, but they have a higher dimensionality than the samples of the BreaKHis dataset. This fact has two impacts: (i) with fewer instances, matrix calculations and the execution of the SMO algorithm take less time; (ii) the generation of the latent representation by the FCN takes longer due to the convolution operation on bigger images. In this direction, the LMFCN and the CNN would have similar runtime, with two drawbacks for the LMFCN because it requires a forward pass of the FCN and the calculation of the large-margin discriminant. Despite this, in Figure 3.28, the LMFCN presented an execution time shorter than the CNN and used only ten epochs instead of 100 of CNN to achieve similar accuracy. The reason for that is

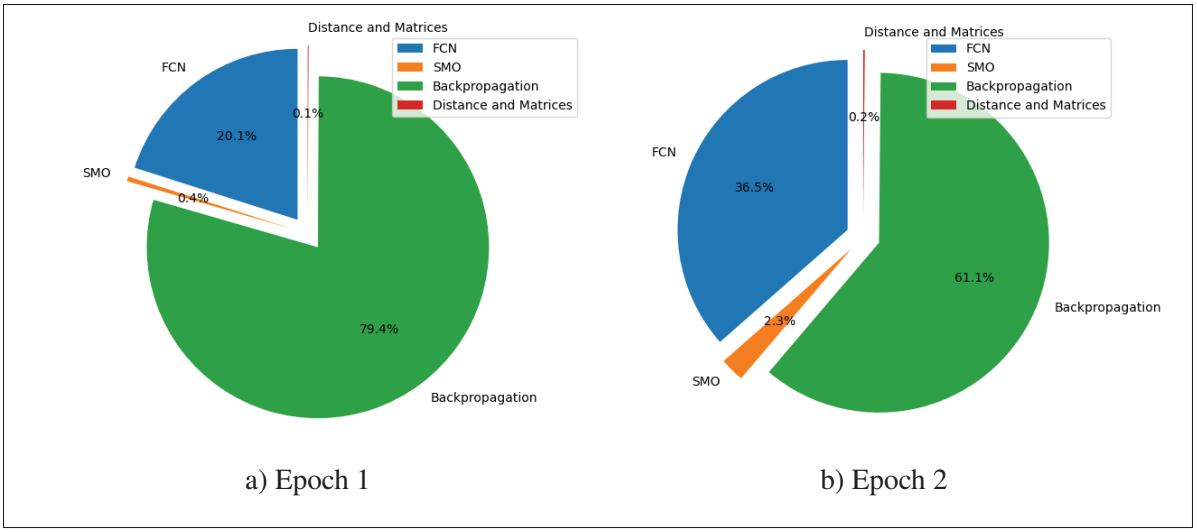


Figure 3.27 Percentage of the total execution time taken by each step of the LMFCN in the first and the second training epochs.

the reduction in the number of SVs in the backpropagation, compensating for the extra forward pass and the other calculations. With a reduced n, the computation of matrices and kernel and the execution of the SMO takes much less time than the execution of the backpropagation algorithm compared to the experiments with the BreaKHis dataset. Such a difference is shown in Figures 3.29a and 3.29b.

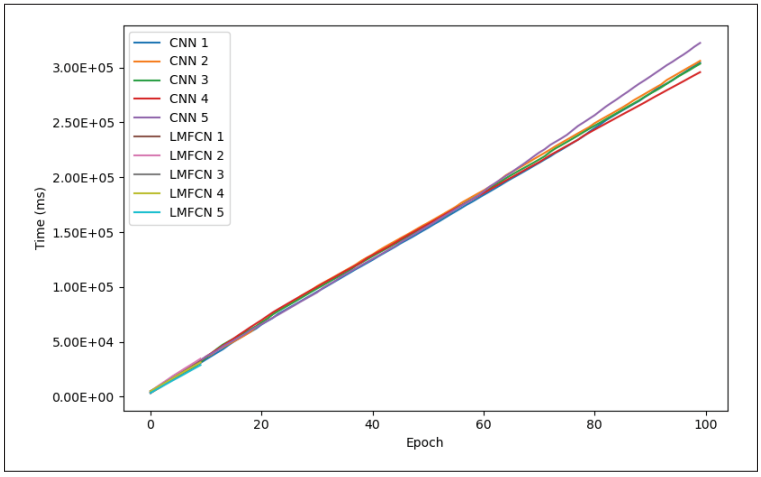
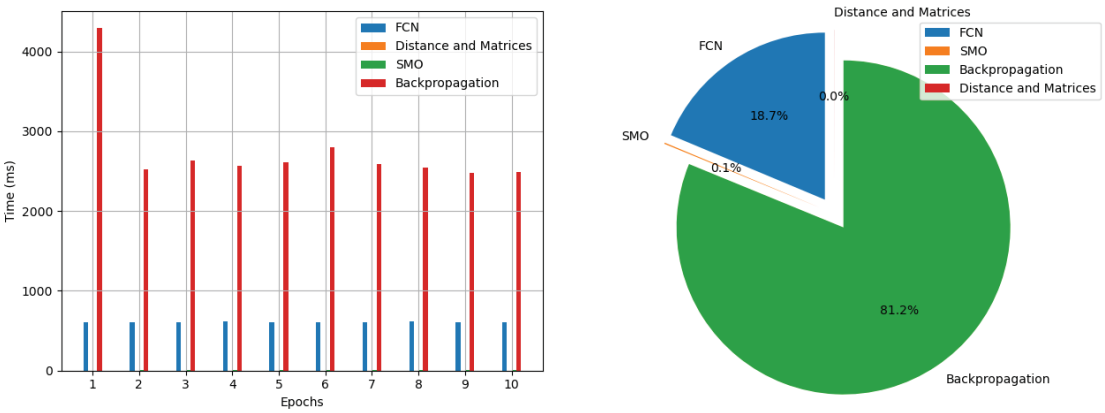


Figure 3.28 Total execution time of 100 epochs for the CNN-BCE and the LMFCN for the BACH dataset)



a) Time breakdown for the LMFCN steps for 10 epochs

b) Time breakdown for the LMFCN steps for epoch one

Figure 3.29 a) Step time inside each epoch of the LMFCN for fold one BACH of the dataset over 10 epochs (first epoch takes longer due to more SVs) and (b) Execution time to each step of the LMFCN for fold one of BACH dataset (latent representation generation, matrices, distance calculation, SMO, and backpropagation over ten epochs)

This section compared the execution times between the LMFCN and the CNN-BCE for BreaKHis and BACH datasets. These datasets have different characteristics. The BreaKHis dataset has more, but smaller, images. The BACH dataset has fewer but larger images. Their aspects impact distinct parts of the LMFCN. Larger images increase the time to generate a latent representation and the execution of the backpropagation algorithm, while a high number of samples increases the execution time to compute matrices and large-margin discriminant. However, thanks to the reduction in the number of SVs and the consequent impact on the number of instances that undergo backpropagation, the LMFCN is still competitive at each epoch. Another aspect is that due to the high discriminative power of the RBF SVM and the latent representation adaptation (FCN weight training), the LMFCN converges to a high accuracy earlier than a CNN.

3.4.7 Impact of Hyperparameters on the LMFCN Performance

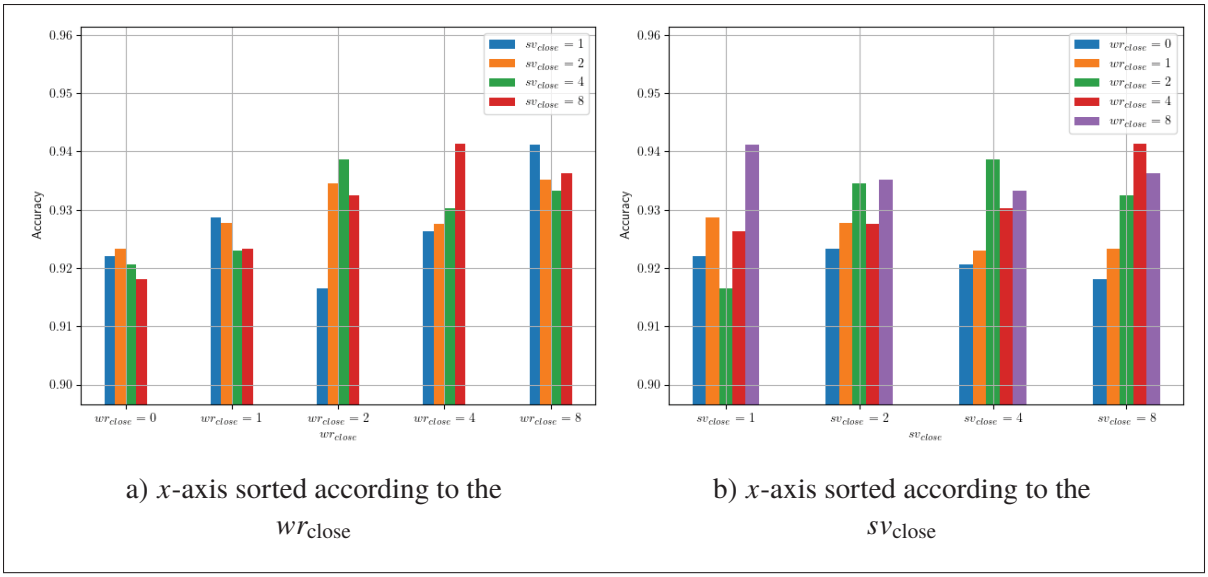
This section analyzes how hyperparameters sv_{close} and wr_{close} impact the balanced accuracy of the LMFCN. Like in the empirical computational analysis carried out in Section 3.4.6, we used the two most challenging datasets, BreaKHis and BACH. The hyperparameter' values were set to

1, 2, 4, and 8 for sv_{close} and 0, 1, 2, 4, and 8 for wr_{close}. The experiments combined these values in a grid search for all five folds of both datasets. wr_{close} started with zero to give us an idea of the impact of the LMFCN without the misclassified instances loss term ($\mathcal{L}_{\text{mc}}$). Disabling $\mathcal{L}_{\text{sv}}$ in our approach ($sv_{\text{close}} = 0$) turns off the main advantage of it, the building of a large-margin representation. It also creates a problem when the training accuracy achieves values close to 100% because the training will halt without instances (almost no misclassified instances) to calculate the loss function if using only $\mathcal{L}_{\text{mc}}$. Thus, using $\mathcal{L}_{\text{sv}}$ higher than one is essential for the LMFCN.

Figure 3.30 presents the values of balanced accuracy for the fold one test set. In Figure 3.30a, bars are ordered in the x-axis firstly by the value of the wr_{close} parameter and secondly by the sv_{close}. Thereby independent of the value of sv_{close}, when not using wr_{close}, we have the smallest accuracy values in this comparison. The best accuracy was obtained with wr_{close} equal to four and eight, also observable in Table 3.7. Figure 3.30b shows that varying sv_{close} does not make significant differences in the accuracy, showing that our method is not so sensitive to this parameter.

Figure 3.31 shows the average balanced accuracy for the test set of all folds of the BreaKHis dataset. In this case, $wr_{\text{close}} = 4$ performs better than the other values with a low influence of sv_{close}. Therefore, our approach is not highly sensitive to its most crucial hyperparameter (sv_{close}), making it easier to set. Tables 3.7 and 3.8 bring the values used to create the bar graphs (Figures 3.30 and 3.31) underlining the best-balanced accuracy values, with averages of the groups (rows and columns). Observing Table 3.8 makes it easy to surmise that $wr_{\text{close}} = 4$ and $sv_{\text{close}} = 8$ produce the best accuracy and that varying sv_{close} produces small changes.

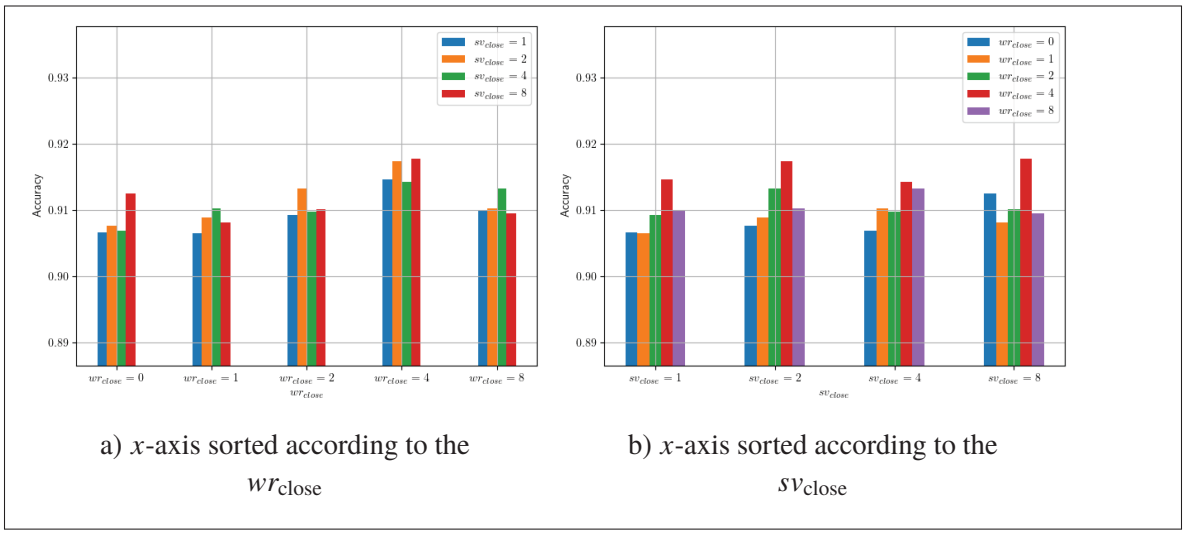
The sensitivity analysis for the BACH dataset is presented in Figures 3.32 and 3.33. For this dataset, the wr_{close} parameter set to zero for fold one resulted in lower balanced accuracy than other values (1, 2, 4, and 8) in Figure 3.32a. Thus, this parameter is essential for training the LMFCN with this dataset, but when it is not zero, its value does not imply in a significant difference. A similar behavior occurs in the average of all folds of the BACH dataset, with



a) x-axis sorted according to the wr_{close}

b) x-axis sorted according to the sv_{close}

Figure 3.30 Accuracy for fold one of the BreaKHis dataset varying wr_{close} and sv_{close}



a) x-axis sorted according to the wr_{close}

b) x-axis sorted according to the sv_{close}

Figure 3.31 Average accuracy for five folds of BreaKHis dataset varying wr_{close} and sv_{close}

lower accuracy values for $wr_{\text{close}} = 0$ (Figure 3.33a), but it is less discrepant than for fold one alone. Unlike the analysis with the BreaKHis dataset, the peak balanced accuracy was obtained for $wr_{\text{close}} = 1$. The BACH dataset is smaller than the BreaKHis dataset (≈2000

Table 3.7 Balanced accuracy on the test set for fold one of the BreaKHis dataset varying wr_{close} and sv_{close}

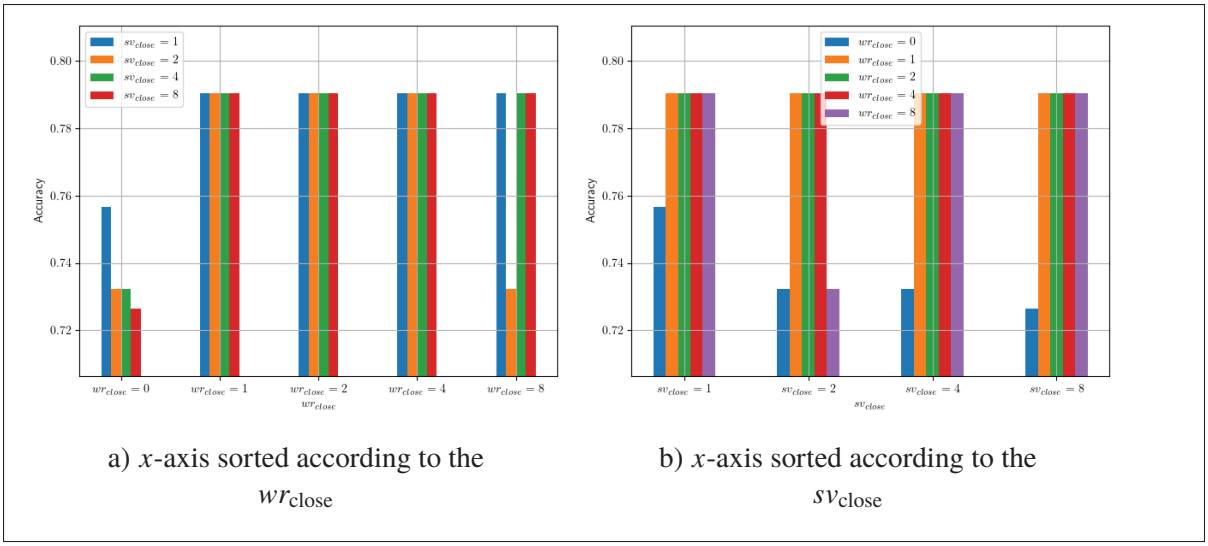
		sv_{close}				**Avg.**	**Std. Dev.**
		1	**2**	**4**	**8**		
wr_{close}	0	0.9221	0.9233	0.9207	0.9181	0.9211	0.0022
	1	0.9287	0.9277	0.9230	0.9233	0.9257	0.0029
	2	0.9166	0.9345	0.9386	0.9325	0.9306	0.0096
	4	0.9264	0.9276	0.9302	0.9414	0.9314	0.0069
	8	0.9411	0.9351	0.9333	0.9362	0.9364	0.0033
Avg.		0.9270	0.9296	0.9292	0.9303		
Std. Dev.		0.0091	0.0050	0.0074	0.0095		

Table 3.8 Test set balanced accuracy for all folds of the BreaKHis dataset varying wr_{close} and sv_{close}

		sv_{close}				**Avg.**	**Std. Dev.**
		1	**2**	**4**	**8**		
wr_{close}	0	0.9067	0.9076	0.9069	0.9125	0.9084	0.0027
	1	0.9065	0.9089	0.9103	0.9081	0.9084	0.0016
	2	0.9093	0.9133	0.9098	0.9101	0.9106	0.0018
	4	0.9147	0.9174	0.9143	0.9178	0.9160	0.0018
	8	0.9100	0.9103	0.9133	0.9095	0.9108	0.0017
Avg.		0.9094	0.9115	0.9109	0.9116		
Std. Dev.		0.0033	0.0039	0.0030	0.0038		

against 400 images). Thus high wr_{close} uses anchors that may be too spread, causing shifts in the misclassified instances to wrong directions. The influence of sv_{close} is more homogeneous, and the best accuracy value was achieved with $sv_{\text{close}} = 4$ (Figure 3.33b). Tables 3.9 and 3.10 summarize the data from the graphs, highlighting the best-balanced accuracy for the experiments and the averages of the groups of experiments. The average of the wr_{close} grouping shows little variability when changing sv_{close}, again pointing it as a stable and easy-to-use hyperparameter. The hyperparameter wr_{close} has its use recommended, as seen in this experiment, and the one with the BreaKHis dataset, but its value is more difficult to choose.

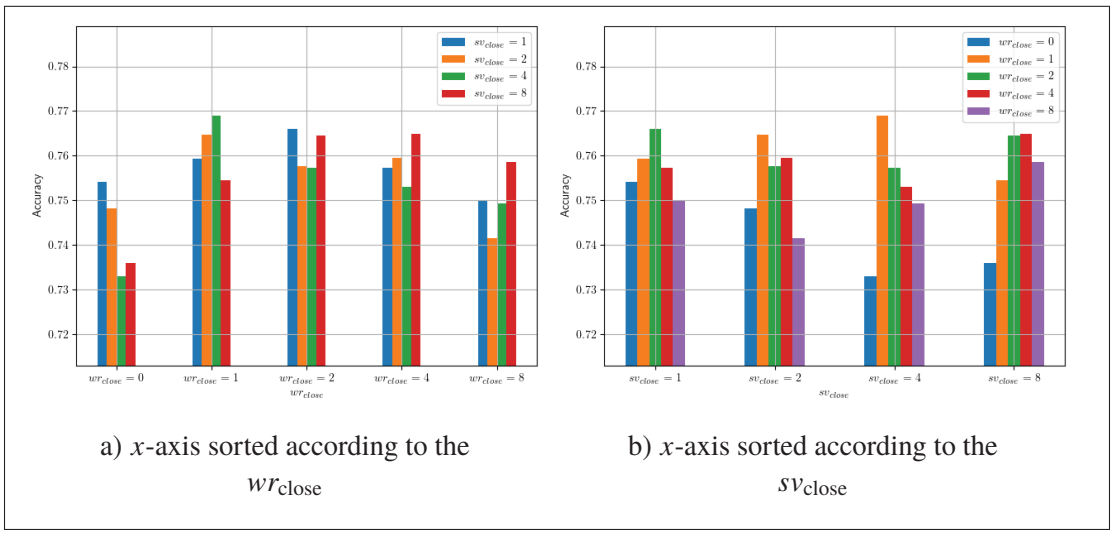
This section analyzed the influence of the two main hyperparameters of the LMFCN, sv_{close} and wr_{close}. The third hyperparameter, related to $\mathcal{L}_{\text{cc}}$, was not analyzed because it is recommended



a) x-axis sorted according to the wr_{close}

b) x-axis sorted according to the sv_{close}

Figure 3.32 Test accuracy for fold one of the BACH dataset varying wr_{close} and sv_{close}



a) x-axis sorted according to the wr_{close}

b) x-axis sorted according to the sv_{close}

Figure 3.33 Average test accuracy for five folds of the BACH dataset varying wr_{close} and sv_{close}

only for specific cases where the other two hyperparameters are insufficient and the data is too packed. It is not the case for the datasets we used in this section's experiments. The hyperparameter sv_{close} showed a more consistent behavior, being more stable in both datasets, so changing its value does not produce catastrophic results. On the other hand, the hyperparameter wr_{close} showed more impact in the different analyses. Its impact is related to the dataset's size

Table 3.9 Balanced accuracy for the test set of fold one of the BACH dataset varying wr_{close} and sv_{close}

		$sv_{\textbf{close}}$				**Avg.**	**Std. Dev.**
		1	**2**	**4**	**8**		
$wr_{\textbf{close}}$	**0**	0.7567	0.7323	0.7323	0.7264	0.7369	0.0135
	1	0.7904	0.7904	0.7904	0.7904	0.7904	0.0000
	2	0.7904	0.7904	0.7904	0.7904	0.7904	0.0000
	4	0.7904	0.7904	0.7904	0.7904	0.7904	0.0000
	8	0.7904	0.7323	0.7904	0.7904	0.7759	0.0291
Avg.		0.7837	0.7672	0.7788	0.7776		
Std. Dev.		0.0151	0.0318	0.0260	0.0286		

Table 3.10 Average accuracy on the test set for five folds of the BACH dataset varying wr_{close} and sv_{close}

		$sv_{\textbf{close}}$				**Avg.**	**Std. Dev.**
		1	**2**	**4**	**8**		
$wr_{\textbf{close}}$	0	0.7541	0.7482	0.7329	0.7359	0.7428	0.0100
	1	0.7593	0.7646	0.7690	0.7544	0.7618	0.0064
	2	0.7659	0.7576	0.7572	0.7645	0.7613	0.0046
	4	0.7572	0.7595	0.7530	0.7649	0.7586	0.0050
	8	0.7501	0.7415	0.7493	0.7585	0.7499	0.0069
Avg.		0.7573	0.7543	0.7523	0.7556		
Std. Dev.		0.0059	0.0093	0.0131	0.0118		

and complexity. When too small, the number of instances in the dataset may produce updates in the FCN guided by $\mathcal{L}_{\text{mc}}$ that could lead to the shift of misclassified instances to the direction of faraway and not recommended anchors. The burden the dataset presents to the LMFCN also affects this hyperparameter setting because challenging problems tend to start the training with many misclassified instances, leading to several updates in the FCN weights. In this case, the ratio between correctly and misclassified samples is low. As a result, it may produce updates that move the correctly classified instances towards undesirable directions, in contrast to the $\mathcal{L}_{\text{sv}}$ that tries to wrap them within regions delimited by SVs.

The BACH dataset is an example of a small and challenging dataset in contrast to the BreaKHis dataset inside the reach of the LMFCN. Therefore, the experiments pointed out that the LMFCN

is not strongly sensitive to the sv_{close} hyperparameter while showing that wr_{close} improves the accuracy. The last one needs some tuning to find its best value for each problem, suggesting that small-size datasets should benefit from lower values of wr_{close} and large-size ones from higher values of wr_{close}. This section's analysis is limited to the use of only two datasets. Still, it intends to show the stability of the LMFCN in the face of two of its most inner hyperparameters.

3.4.8 Discussion

The LMFCN achieved equivalent performance or outperformed the compared methods. It is a hybrid approach inherited from deep learning (FCN) and large-margin classifiers (SVM). It is common sense that deep learning methods are suitable for large datasets. With small-size datasets, the high number of parameters of deep learning methods are challenging to train and obtain good generalization. To circumvent this problem, methods like data augmentation increase the amount of data and also help to increase the variability of data. Although using FCNs, the LMFCN works with small-size datasets, and instead of using data augmentation, it requires even less training data than the amount available in the original dataset. Our approach selects some instances of interest to participate in the loss calculation and the backpropagation procedure. Our approach requires reduced training data because it only relies on instances close to the margins or at the wrong side of the decision boundary. The other instances do not need to participate in the training process because they will not effectively contribute to the margin definition.

On the other hand, deep learning training algorithms must use all instances available because the well-classified ones reduce aggressive updates, acting like a weighting mechanism. Therefore, their training also follows the majority. Updates based only on misclassified instances could shift well-classified instances to the wrong side of the decision boundary because it mainly focuses on correcting complicated instances. In the LMFCN, the SVs that define different margins balance themselves to avoid pushing well-classified instances to the wrong side of decision boundaries. That is better explained in Section 2.4. CNNs do not work with a large-margin discriminant,

so their training algorithm does not need to bother with margin definition, unlike the LMFCN approach.

Another relevant aspect tying CNNs to large datasets is their number of parameters and the multiclass capabilities, reaching up to thousand classes in architectures designed to deal with the ImageNet dataset. The number of parameters must be high to create an internal representation appropriate to complex problems by the weights. However, as the LMFCN is designed to work with small-size datasets and few classes, it only needs a few parameters to model the problem.

Using datasets with few instances and an update method like the one of CNNs, employing the entire training set, can generate problems with dispersed and misclassified samples. These instances cause a consistent updating, which makes margin enlargement difficult. On the other hand, the LMFCN, thanks to the soft margin SVM, ignores such harmful instances in the loss calculation. The LMFCN focuses on margin definition rather than correcting misclassified instances. Our experiments helped to understand this behavior by improving balanced accuracy over the epochs. The feature complexity measures also allowed us to compare the features generated by different methods and find the reasons for our approach's good classification performance and adaptative capabilities.

CONCLUSION AND RECOMMENDATIONS

This thesis aimed to create an end-to-end strategy to learn the representation and a large-margin discriminant from a small-size texture dataset with competitive results compared to approaches based on CNNs. Our results and analysis showed that the FCN and the discriminant formed an end-to-end method of supervised learning, named LMFCN, with the representation being generated and modified to improve the accuracy of the discriminant. The balanced accuracy results were equivalent or superior to approaches based on CNNs and shallow approaches based on handcrafted feature extractors in a scenario of small-size texture datasets. It also outperformed the compared methods in multiclass scenarios.

A question raised in this thesis was whether a large-margin discriminant replacing FC layers of conventional CNNs could speed up the training convergence while maintaining or improving classification accuracy. The large-margin discriminant contributed in two ways to speed up the training convergence: (i) by reducing the number of parameters to update during the backpropagation procedure; (ii) by producing more complex decision boundaries. Another point that reduced the computational complexity of the proposed approach is that the loss function conceived to train the FCN used fewer instances during the loss calculation and backpropagation. This last characteristic did not speed up the convergence directly but contributed to reducing the computational effort of the training. The synergy of the latent representation generation by the FCN and the requirements of the large-margin discriminant also helped speed up the convergence.

Training complex machine learning models on small-size datasets is challenging because they tend to overfit, producing unsatisfying generalization results. On the other hand, simple models may not be able to represent the data accordingly, generating underfitting. Large deep-learning models currently present the most relevant results in object classification. Part of this success is the availability of large datasets of this type. Compared to images with objects, medical

HIs are harder to acquire and label, limiting the data for training. Many shallow methods, handcrafted feature extractors combined with monolithic classifiers, produced relevant accuracy results for HI classification according to de Matos *et al.* (2021). Handcrafted feature extractors' disadvantages are that they cannot adapt to different textural characteristics, and their generated features are not tailored to a classifier. On the other hand, deep learning methods, like CNNs, can adjust the weights of the convolutional filters to better deal with images while improving the discriminant in deep layers. Given these two categories of machine learning methods, we created the LMFCN, a hybrid approach combining the adaptability of CLs as filter banks and a large-margin discriminant that works well with small-size datasets.

The LMFCN uses a novel loss function to train CLs from scratch to filter relevant characteristics of diverse textural images. The weight updating of the CLs generates a latent representation better suited to train a large-margin discriminant. The LMFCN targets small-size datasets with textural characteristics and a limited number of classes. Using a shallow FCN requires training of a few parameters, making it suitable for training with few data. Compared to traditional CNNs, our approach replaces deep FC layers that act as a discriminant by a large-margin classifier, specifically an SVM. The proposed loss function aims to enlarge the inter-class margin by shifting SVs toward their anchors, the closest correctly classified instances of the same class, in the latent representation space. It also moves misclassified instances toward the SVs and pushes opposite-class instances apart.

The LMFCN converges faster than equivalent CNNs due to the reduced amount of parameters and the discriminant tailored loss function. The large-margin discriminant also helps achieve comparable or better generalization than other methods. In the experiments comparing LMFCN and equivalent CNNs, using 16-dimensional latent representations, our approach obtained training stability and high accuracy within 20 epochs, outperforming CNNs that took 100 epochs to achieve comparable performance. The complexity measures revealed that the LMFCN

produces substantial weight updates on the first epochs correlating to the balanced accuracy metric. It also kept a consistent training behavior across different datasets. They also indicated that the CLs of the CNNs (with hinge and BCE loss) needed more updates to work well with FC layers and reach an accuracy equivalent to the LMFCN. The complexity measures provided a hint of why the LMFCN performed better. It does not need huge updates on latent representation because it better fits the classifier requirements. In CNNs, even if the GAP layer already produces discriminative representations, the loss function depends on the combination of FCN+FC. Hence, the backpropagation algorithm updates the entire network over the entire training set.

Regarding handcrafted feature extractors, our approach adapts to different textures due to the FCN weight updating. However, the LMFCN takes more time to execute. Still, the accuracy was equivalent or superior to the best of all other methods. Thus, while different handcrafted feature extractors obtained the best accuracies in distinct datasets, the LMFCN accuracy performance was close to the best on each dataset. The results from handcrafted feature extractors revealed that the PFTAS was the best feature extractor for three datasets (BACH, BreaKHis, and CRC), the LBP was the best for the Salzburg dataset, and the GLCM was the best for the Gaussian image dataset. Nonetheless, the LMFCN outperformed all of them on all datasets showing that the representation learning adapts to the characteristics of each type of image and simultaneously adapts the latent representations to best suit the large-margin discriminant. For a fair comparison of the LMFCN with handcrafted feature extractors, we used latent representations of the same size as the ones generated by each feature extractor: 59-dimensional for the LBP, 162-dimensional for the PFTAS, and 169-dimensional for the GLCM. The LMFCN achieved higher balanced accuracy in these experiments than the compared methods. The LMFCN already obtained good accuracy with a 16-dimensional latent representation. However, augmenting the latent representation's dimensionality improved the results' quality. It was noticed that the performance improvement from 16 to 59 dimensions is meaningful, with a slight gain when increasing latent representations to 162 dimensions and an almost negligible gain from 162 to 169 dimensions.

Although the LMFCN did not target multiclass scenarios, its performance was superior to the other compared methods. The multiclass LMFCN uses multiple models with the OVA technique allowing it better distinguish each class because it can learn subtle characteristics of each one against the others. Even if the OVA is unsuitable for imbalanced data, selecting instances of interest embedded in the LMFCN training somewhat circumvents such a limitation because it selects instances of the underrepresented classes and just a few from the most prevalent ones. For example, suppose there is a class with numerous clustered instances. In that case, the LMFCN approach only uses instances close to the decision boundaries. Hence, the number of instances of interest used in calculating the loss function is not too imbalanced. The LMFCN showed equivalent or superior accuracy performance compared to other methods. Nevertheless, it has the drawback of the high computational cost, which grows with the number of classes. However, the computational cost is not extremely high, given that we used only ten epochs to train each model's subproblem. Besides, we used small FCNs than CNNs. CNNs took 400 training epochs to achieve an equivalent but smaller accuracy than the LMFCN. It is also important to emphasize that training CNNs requires all training instances in all training epochs. In contrast, each binary FCN of the LMFCN needed only the SVs discovered in each subproblem at the end of each training epoch. For instance, when the number of classes increases, it is more advantageous to use DML (a very high number of classes and few training instances) or CNNs (a medium number of classes and a high number of instances).

The analysis of the latent representation produced by the LMFCN with complexity measures and UMAPs showed that the representation produced by CNNs is different. For instance, a CNN as a feature extractor generated a representation that does not necessarily match the needs of an SVM. Handcrafted feature extractors also create features not necessarily tailored to an SVM. However, the LMFCN takes advantage of updating the weights of the CLs responsible for capturing the input images' characteristics producing a latent representation that is more suitable for training a large-margin classification model. Information gathered during the training of

the LMFCN indicates a progressive reduction in the number of SVs over the epochs. It occurs concomitantly with the improvement in classification results, depicted by the balanced accuracy and the separation of instances in the latent representation space, expressed by decreasing N1 and N2 measures. The decrease in the number of SVs means that the samples, under the perspective of their latent representation generated by the FCN, are well-positioned in the representation space, facilitating the definition of the separation margins. Having fewer SVs also implies that the training procedure of the FCN requires less computational effort, as the main term of the proposed loss function depends on the number of SVs. Each epoch becomes more and more computationally cheap as the discriminative power of the latent representation generated by the FCN improves, best fitting the SVM needs. The LMFCN, compared to equivalent CNNs, has an extra cost related to the creation matrices (kernel and distance) and the SMO algorithm that discover the SVs. Matrices calculations have quadratic complexity proportional to the number of instances. The SMO algorithm has cubic complexity proportional to the number of samples. On the other hand, the computational cost related to CNNs is proportional to the number of images and network parameters. In summary, CNNs have more parameters, use more instances with the backpropagation algorithm, and take more epochs to converge than the LMFCN. Therefore, CNNs are more computationally costly than the LMFCN in scenarios with few training data.

The usual procedure when training CNNs on small-size datasets is to use data augmentation to raise the variability and the amount of training data. The LMFCN approach works oppositely; although focused on small-size datasets, it uses fewer instances (only SVs) for training. This fact can lead to questions about the resilience to outliers or mislabeling. As shown in our simulations in Section 2.4, selecting SVs and using the lenience of the SVM to misclassified instances reduce the adverse effects of outliers or mislabeled instances. The accuracy of CNNs usually improves with data augmentation, although augmenting data causes two side effects: (i) increases the number of training instances, increasing the cost of backpropagation at each epoch; (ii) extra computing needed by the data augmentation algorithm.

The main contribution of this thesis is a novel instance selection method for the loss function used in the backpropagation procedure. The instance selection method focuses on the SVs and the misclassified instances, ignoring instances that do not contribute to the training procedure. After the large-margin discriminant optimization, these instances are defined, so the weight update aims to improve the discriminant's latent representation. As the loss function uses only certain selected instances, it reduces the burden of the backpropagation procedure.

Overall, contributions of this thesis can be summarized as follows:

- Creation of a machine learning approach that can train an FCN and a large-margin discriminant from scratch with small-size datasets.
- A method that can adapt the filters to different textures.
- An efficient training method that is lighter than traditional CNNs using only some selected instances as training data
- Compared to CNNs as feature extractors, the representation is targeted to the classifier needs.
- Developing a multiclass classification method that performs well even on imbalanced data.

Limitations

The proposed loss function has hyperparameters allowing the training algorithm to select the number of anchors for each instance of interest. Our experiments used a small range of values for these hyperparameters to find the ones which render the best results. We found that small values of sv_{close} are more suitable for smaller datasets. In more challenging datasets, many SVs necessary to draw complex decision borders also require small values of sv_{close} because the method finds more SVs than anchors, considering the entire problem. Thus, the sensitivity and the selection of best values for the hyperparameters are a limitation of our method. However, this is also a characteristic of many machine-learning algorithms.

Another limitation is the OVA approach and the latent representation concatenation in multiclass situations. With many classes, the multiclass latent representation tends to grow, like the number of filter banks, also bringing more complexity to the model and increasing its computational costs. Therefore, this fact limits the use of the LMFCN to scenarios with fewer classes.

Future Work

The LMFCN permits the use of different FCN architectures. One analysis worth exploring is the behavior of our training method with other FCN architectures having different widths, depths, and blocks. The combination of different sizes of training sets, different complexities of FCNs, and different dimensions of the latent representation is an extensive and valuable analysis that can assess the limits of the LMFCN. However, it requires an experimental design that covers the possibilities to achieve consistent conclusions.

The development of the LMFCN focused on natural texture images and HIs in small-size datasets. The context of object recognition works better with methods that can extract a complete set of features, like spatial, shape, and position information. Although, before the advent of CNNs, some works used handcrafted feature extractors and their combinations. It will be interesting to assess the capacities of FCNs, or even small CNNs, in datasets more complex than those used in our experiments to characterize the limits of the LMFCN. For instance, replacing the GAP layer with a less aggressive aggregation method allows learning more characteristics from images to the detriment of training simplicity. Exploration of this relationship can discover new applications of our method.

We also consider the analysis of adversarial attacks resilience an essential issue regarding future studies of the LMFCN. Large-margin representations tend to increase this resilience, but we do not have mechanisms in this method to address the large gradient problem. We intend to evaluate

our approach with various adversarial attack methods and apply some defenses described in the literature related to CNNs.

Publications

- de Matos, J., de Souza Britto, A., Oliveira, L. E. & Koerich, A. L. (2019). Double transfer learning for breast cancer histopathologic image classification. *International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8.
- de Matos, J., de Souza Britto, A., de Oliveira, L. E. & Koerich, A. L. (2019). Texture CNN for histopathological image classification. *IEEE 32nd international symposium on computer-based medical systems (CBMS)*, pp. 580–583.
- Ataky, S. T. M., de Matos, J., Britto, A. d. S., Oliveira, L. E. & Koerich, A. L. (2020). Data augmentation for histopathological images based on gaussian-laplacian pyramid blending. *International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8.
- de Matos, J., Ataky, S. T. M., de Souza Britto, A., Oliveira, L. E. & Koerich, A. L. (2021). Machine learning methods for histopathological image analysis: A review. *Electronics*, 10(5), 562.
- Ataky, S.T.M.; Saqui, D.; de Matos, J.; de Souza Britto Junior, A.; Lameiras Koerich, A. (2023). Multiscale Analysis for Improving Texture Classification. *Applied Sciences*, 13(3), 1291.
- de Matos, J., de Oliveira, L. E. S., Junior, A. D. S. B., & Koerich, A. L. (2023). Large-margin representation learning for texture classification. Pattern Recognition Letters, 170, 39-47.

BIBLIOGRAPHY

Albarqouni, S., Baur, C., Achilles, F., Belagiannis, V., Demirci, S. & Navab, N. (2016). AggNet: Deep Learning From Crowds for Mitosis Detection in Breast Cancer Histology Images. *IEEE Transactions on Medical Imaging*, 35(5, SI), 1313–1321. doi: 10.1109/TMI.2016.2528120.

Alom, M. Z., Yakopcic, C., Nasrin, M. S., Taha, T. M. & Asari, V. K. (2019). Breast cancer classification from histopathological images with inception recurrent residual convolutional neural network. *Journal of digital imaging*, 32, 605–617.

Andrearczyk, V. & Whelan, P. F. (2016). Using filter banks in Convolutional Neural Networks for texture classification. *Pattern Recognition Letters*, 84, 63 - 69. doi: https://doi.org/10.1016/j.patrec.2016.08.016.

Aresta, G., Araújo, T., Kwok, S., Chennamsetty, S. S., Safwan, M., Alex, V., Marami, B., Prastawa, M., Chan, M., Donovan, M., Fernandez, G., Zeineh, J., Kohl, M., Walz, C., Ludwig, F., Braunewell, S., Baust, M., Vu, Q. D., To, M. N. N., Kim, E., Kwak, J. T., Galal, S., Sanchez-Freire, V., Brancati, N., Frucci, M., Riccio, D., Wang, Y., Sun, L., Ma, K., Fang, J., Kone, I., Boulmane, L., Campilho, A., Eloy, C., Polónia, A. & Aguiar, P. (2019). BACH: Grand challenge on breast cancer histology images. *Medical Image Analysis*, 56, 122 - 139. doi: https://doi.org/10.1016/j.media.2019.05.010.

Atupelage, C., Nagahashi, H., Yamaguchi, M., Abe, T., Hashiguchi, A. & Sakamoto, M. (2013). Computational grading of hepatocellular carcinoma using multifractal feature description. *Computerized Medical Imaging and Graphics*, 37(1), 61-71. doi: https://doi.org/10.1016/j.compmedimag.2012.10.001.

Awan, R., Aloraidi, N., Qidwai, U. & Rajpoot, N. (2016, feb). How divided is a cell? Eigenphase nuclei for classification of mitotic phase in cancer histology images. *2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, pp. 70–73. doi: 10.1109/BHI.2016.7455837.

Ballarò, B., Florena, A. M., Franco, V., Tegolo, D., Tripodo, C. & Valenti, C. (2008). An automated image analysis methodology for classifying megakaryocytes in chronic myeloproliferative disorders. *Medical Image Analysis*, 12(6), 703–712. doi: https://doi.org/10.1016/j.media.2008.04.001.

Banerji, S. & Mitra, S. (2022). Deep learning in histopathology: A review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 12(1), e1439.

Bardou, D., Zhang, K. & Ahmad, S. M. (2018). Classification of breast cancer based on histology images using convolutional neural networks. *Ieee Access*, 6, 24680–24693.

Basu, M. & Ho, T. K. (2006). *Data complexity in pattern recognition*. Springer Science & Business Media.

Bellocq, J.-P., Anger, E., Camparo, P., Capron, F., Chenard, M.-P., Chetritt, J., Chigot, J.-P., Cochand-Priollet, B., Coindre, J.-M., Copin, M.-C., Fléjou, J.-F., Galateau, F., Gaulard, P., Guiu, M., Michiels, J.-F., Saint-André, J.-P., Scoazec, J.-Y. & Vacher-Lavenu, M.-C. (2011). Sécuriser le diagnostic en anatomie et cytologie pathologiques en 2011. L'erreur diagnostique: entre discours et réalité. *Annales de Pathologie*, 31(5, Supplement), S92 - S94. doi: https://doi.org/10.1016/j.annpat.2011.08.006. Carrefour pathologie - 21 au 25 novembre 2011.

Bruno, D. O. T., do Nascimento, M. Z., Ramos, R. P., Batista, V. R., Neves, L. A. & Martins, A. S. (2016). {LBP} operators on curvelet coefficients as an algorithm to describe texture in breast cancer tissues. *Expert Systems with Applications*, 55, 329–340. doi: https://doi.org/10.1016/j.eswa.2016.02.019.

Caicedo, J. C., Gonzalez, F. A. & Romero, E. (2008). A semantic content-based retrieval method for histopathology images. *Information Retrieval Technology*, 4993(Lecture Notes in Computer Science), 51+.

Chan, A. & Tuszynski, J. A. (2016). Automatic prediction of tumour malignancy in breast cancer with fractal dimension. *Royal Society open science*, 3(12), 160558.

Chen, H., Dou, Q., Wang, X., Qin, J. & Heng, P. (2016). Mitosis detection in breast cancer histology images via deep cascaded networks. *Proceedings of the AAAI conference on artificial intelligence*, 30(1), 1.

Chen, H., Li, C., Wang, G., Li, X., Rahaman, M. M., Sun, H., Hu, W., Li, Y., Liu, W., Sun, C. et al. (2022). GasHis-Transformer: A multi-scale visual transformer approach for gastric histopathological image detection. *Pattern Recognition*, 130, 108827.

Cheplygina, V., de Bruijne, M. & Pluim, J. P. (2019). Not-so-supervised: A survey of semi-supervised, multi-instance, and transfer learning in medical image analysis. *Medical Image Analysis*, 54, 280 - 296. doi: https://doi.org/10.1016/j.media.2019.03.009.

Chopra, S., Hadsell, R. & LeCun, Y. (2005). Learning a SIMILARITY metric discriminatively, with application to face verification. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 1, 539-546 vol. 1. doi: 10.1109/CVPR.2005.202.

Cimpoi, M., Maji, S. & Vedaldi, A. (2015). Deep filter banks for texture recognition and segmentation. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3828-3836.

Coelho, L. P., Ahmed, A., Arnold, A., Kangas, J., Sheikh, A.-S., Xing, E. P., Cohen, W. W. & Murphy, R. F. (2010). Structured literature image finder: extracting information from text and images in biomedical literature. In *Linking Literature, Information, and Knowledge for Biology* (pp. 23–32). Springer.

Dabass, M., Vashisth, S. & Vig, R. (2021). Attention-Guided deep atrous-residual U-Net architecture for automated gland segmentation in colon histopathology images. *Informatics in Medicine Unlocked*, 27, 100784.

Das, D. K., Mitra, P., Chakraborty, C., Chatterjee, S., Maiti, A. K. & Bose, S. (2017). Computational approach for mitotic cell detection and its application in oral squamous cell carcinoma. *Multidimensional Systems and Signal Procesing*, 28(3, SI), 1031–1050. doi: 10.1007/s11045-017-0488-6.

de Matos, J., Britto, A. d. S., Oliveira, L. E. & Koerich, A. L. (2019). Double transfer learning for breast cancer histopathologic image classification. *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8.

de Matos, J., Ataky, S. T. M., de Souza Britto Jr, A., Soares de Oliveira, L. E. & Lameiras Koerich, A. (2021). Machine learning methods for histopathological image analysis: A review. *Electronics*, 10(5), 562.

Deniz, E., Şengür, A., Kadiroğlu, Z., Guo, Y., Bajaj, V. & Budak, Ü. (2018). Transfer learning based histopathologic image classification for breast cancer detection. *Health Information Science and Systems*, 6(1), 18. doi: 10.1007/s13755-018-0057-x.

Elmore, J. G., Longton, G. M., Carney, P. A., Geller, B. M., Onega, T., Tosteson, A. N. A., Nelson, H. D., Pepe, M. S., Allison, K. H., Schnitt, S. J., O'Malley, F. P. & Weaver, D. L. (2015). Diagnostic Concordance Among Pathologists Interpreting Breast Biopsy Specimens. *Journal of American Medical Association*, 313(11), 1122-1132.

Fernández-Carrobles, M. M., Bueno, G., Déniz, O., Salido, J., García-Rojo, M. & Gonzández-López, L. (2015). Frequential versus spatial colour textons for breast {TMA} classification. *Computerized Medical Imaging and Graphics*, 42, 25-37. doi: https://doi.org/10.1016/j.compmedimag.2014.11.009.

Fukuma, K., Prasath, V. B. S., Kawanaka, H., Aronow, B. J. & Takase, H. (2016). A Study on Nuclei Segmentation, Feature Extraction and Disease Stage Classification for Human Brain Histopathological Images. *Procedia Computer Science*, 96, 1202–1210. doi: https://doi.org/10.1016/j.procs.2016.08.164.

Galea, M. H., Blamey, R. W., Elston, C. E. & Ellis, I. O. (1992). The Nottingham prognostic index in primary breast cancer. *Breast Cancer Research and Treatment*, 22(3), 207–219. doi: 10.1007/BF01840834.

Gleason, D. F. (1992). Histologic grading of prostate cancer: A perspective. *Human Pathology*, 23(3), 273–279. doi: 10.1016/0046-8177(92)90108-F.

Glotsos, D., Kalatzis, I., Spyridonos, P., Kostopoulos, S., Daskalakis, A., Athanasiadis, E., Ravazoula, P. & Nikiforidis,G.and Cavouras, D. (2008). Improving accuracy in astrocytomas grading by integrating a robust least squares mapping driven support vector machine classifier into a two level grade classification scheme. *Computer Methods and Programs in Biomedicine*, 90, 251-261.

Guo, M.-H., Xu, T.-X., Liu, J.-J., Liu, Z.-N., Jiang, P.-T., Mu, T.-J., Zhang, S.-H., Martin, R. R., Cheng, M.-M. & Hu, S.-M. (2022). Attention mechanisms in computer vision: A survey. *Computational visual media*, 8(3), 331–368.

Gurcan, M. N., Boucheron, L. E. & Can, A. (2009). Histopathological Image Analysis: A Review. *IEEE Reviews in Biomedical Engineering*, 2(1), 147-171.

Haralick, R. M., Shanmugam, K. & Dinstein, I. H. (1973). Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, (6), 610–621.

He, K., Zhang, X., Ren, S. & Sun, J. (2016). Deep residual learning for image recognition. *IEEE/CVF Conf Comp Vis Patt Recog*, pp. 770–778.

He, Z., Lin, M., Xu, Z., Yao, Z., Chen, H., Alhudhaif, A. & Alenezi, F. (2022). Deconv-transformer (DecT): A histopathological image classification model for breast cancer based on color deconvolution and transformer architecture. *Information Sciences*, 608, 1093–1112.

Ho, T. K. & Basu, M. (2002). Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3), 289-300. doi: 10.1109/34.990132.

Homeyer, A., Schenk, A., Arlt, J., Dahmen, U., Dirsch, O. & Hahn, H. K. (2013). Practical quantification of necrosis in histological whole-slide images. *Computerized Medical Imaging and Graphics*, 37(4), 313–322. doi: https://doi.org/10.1016/j.compmedimag.2013.05.002.

Hou, L., Samaras, D., Kurc, T. M., Gao, Y., Davis, J. E. & Saltz, J. H. (2016, June). Patch-Based Convolutional Neural Network for Whole Slide Tissue Image Classification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Huang, P., Tan, X., Zhou, X., Liu, S., Mercaldo, F. & Santone, A. (2021). FABNet: fusion attention block and transfer learning for laryngeal cancer tumor grading in P63 IHC histopathology images. *IEEE Journal of Biomedical and Health Informatics*, 26(4), 1696–1707.

Humeau-Heurtier, A. (2019). Texture Feature Extraction Methods: A Survey. *IEEE Access*, 7, 8975-9000.

Jiang, Y., Chen, L., Zhang, H. & Xiao, X. (2019). Breast cancer histopathological image classification using convolutional neural networks with small SE-ResNet module. *PloS one*, 14(3), e0214587.

Kather, J. N., Weis, C.-A., Bianconi, F., Melchers, S. M., Schad, L. R., Gaiser, T., Marx, A. & Zöllner, F. G. (2016). Multi-class texture analysis in colorectal cancer histology. *Scientific Reports*, 6(1), 27988. doi: 10.1038/srep27988.

Khan, S., Islam, N., Jan, Z., Din, I. U. & Rodrigues, J. J. C. (2019). A novel deep learning based framework for the detection and classification of breast cancer using transfer learning. *Pattern Recognition Letters*, 125, 1–6.

Komura, D. & Ishikawa, S. (2018). Machine learning methods for histopathological image analysis. *Computational and structural biotechnology journal*, 16, 34–42.

Kostopoulos, S., Glotsos, D., Cavouras, D., Daskalakis, A., Kalatzis, I., Georgiadis, P., Bougioukos, P., Ravazoula, P. & Nikiforidis, G. (2009). Computer-based association of the texture of expressed estrogen receptor nuclei with histologic grade using immunohistochemically-stained breast carcinomas. *Analytical and Quantitative Cytology and Histology*, 31, 187-196.

Kostopoulos, S., Ravazoula, P., Asvestas, P., Kalatzis, I., Xenogiannopoulos, G., Cavouras, D. & Glotsos, D. (2017). Development of a Reference Image Collection Library for Histopathology Image Processing, Analysis and Decision Support Systems Research. *Journal of Digital Imaging*, 30(3), 287–295. doi: 10.1007/s10278-017-9947-8.

Kuse, M., Sharma, T. & Gupta, S. (2010). A classification scheme for lymphocyte segmentation in H&E stained histology images. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6388 LNCS, 235–243. doi: 10.1007/978-3-642-17711-8_24.

Kwitt, R. & Meerwald, P. [Accessed: 2021-10-04]. (2020). STex, Salzburg texture image database (STex). Retrieved from: https://wavelab.at/sources/STex/.

LeCun, Y., Bengio, Y. & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436-444. doi: 10.1038/nature14539.

Leo, P., Lee, G., Shih, N. N., Elliott, R., Feldman, M. D. & Madabhushi, A. (2016). Evaluating stability of histomorphometric features across scanner and staining variations: prostate cancer diagnosis from whole slide images. *Journal of medical imaging*, 3(4), 047502.

Li, H., Yang, F., Zhao, Y., Xing, X., Zhang, J., Gao, M., Huang, J., Wang, L. & Yao, J. (2021a). DT-MIL: deformable transformer for multi-instance learning on histopathological image. *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part VIII 24*, pp. 206–216.

Li, J., Li, W., Sisk, A., Ye, H., Wallace, W. D., Speier, W. & Arnold, C. W. (2021b). A multi-resolution model for histopathology image classification and localization with multiple instance learning. *Computers in biology and medicine*, 131, 104253.

Liu, L., Fieguth, P., Wang, X., Pietikäinen, M. & Hu, D. (2016). Evaluation of LBP and Deep Texture Descriptors with a New Robustness Benchmark. *Computer Vision – ECCV 2016*, pp. 69–86.

Loeffler, M., Greulich, L., Scheibe, P., Kahl, P., Shaikhibrahim, Z., Braumann, U.-D., Kuska, J.-P. & Wernert, N. (2012). Classifying Prostate Cancer Malignancy by Quantitative Histomorphometry. *The Journal of Urology*, 187(5), 1867–1875. doi: https://doi.org/10.1016/j.juro.2011.12.054.

Lorena, A. C., Garcia, L. P., Lehmann, J., Souto, M. C. & Ho, T. K. (2019). How Complex is your classification problem? A survey on measuring classification complexity. *ACM Computing Surveys (CSUR)*, 52(5), 1–34.

Mahbod, A., Ellinger, I., Ecker, R., Smedby, Ö. & Wang, C. (2018). Breast cancer histological image classification using fine-tuned deep network fusion. *Image Analysis and Recognition: 15th International Conference, ICIAR 2018, Póvoa de Varzim, Portugal, June 27–29, 2018, Proceedings 15*, pp. 754–762.

McInnes, L., Healy, J., Saul, N. & Großberger, L. (2018). UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software*, 3(29), 861.

Mehra, R. et al. (2018). Breast cancer histology images classification: Training from scratch or transfer learning? *ICT Express*, 4(4), 247–254.

Michail, E., Dimitropoulos, K., Koletsa, T., Kostopoulos, I. & Grammalidis, N. (2014). Morphological and textural analysis of centroblasts in low-thickness sliced tissue biopsies of follicular lymphoma. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference*, 2014, 3374–3377. doi: 10.1109/EMBC.2014.6944346.

Niazi, M. K. K., Parwani, A. V. & Gurcan, M. N. (2016). Computer-Assisted bladder cancer grading: α-shapes for color space decomposition. *Progress in Biomedical Optics and Imaging - Proceedings of SPIE*, 9791, 40–47. doi: 10.1117/12.2216967.

Ninos, K., Kostopoulos, S., Kalatzis, I., Sidiropoulos, K., Ravazoula, P., Sakellaropoulos, G., Panayiotakis, G., Economou, G. & Cavouras, D. (2016). Microscopy image analysis of p63 immunohistochemically stained laryngeal cancer lesions for predicting patient 5-year survival. *European Archives of Oto-Rhino-Laryngology*, 273, 159-168.

Ojala, T., Pietikainen, M. & Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7), 971–987.

Peyret, R., Bouridane, A., Khelifi, F., Tahir, M. A. & Al-Maadeed, S. (2018). Automatic classification of colorectal and prostatic histologic tumor images using multiscale multispectral local binary pattern texture features and stacked generalization. *Neurocomputing*, 275, 83–93. doi: https://doi.org/10.1016/j.neucom.2017.05.010.

Phoulady, H. A., Zhou, M., Goldgof, D. B., Hall, L. O. & Mouton, P. R. (2016, sep). Automatic quantification and classification of cervical cancer via Adaptive Nucleus Shape Modeling. *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 2658–2662. doi: 10.1109/ICIP.2016.7532841.

Qian, Z., Li, K., Lai, M., Chang, E. I.-C., Wei, B., Fan, Y. & Xu, Y. (2022). Transformer based multiple instance learning for weakly supervised histopathology image segmentation. *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 160–170.

Rahmadwati, Naghdy, G., Ros, M., Todd, C. & Norahmawati, E. (2011, jul). Cervical Cancer Classification Using Gabor Filters. *2011 IEEE First International Conference on Healthcare Informatics, Imaging and Systems Biology*, pp. 48–52. doi: 10.1109/HISB.2011.15.

Reis, S., Gazinska, P., Hipwell, J. H., Mertzanidou, T., Naidoo, K., Williams, N., Pinder, S. & Hawkes, D. J. (2017). Automated Classification of Breast Cancer Stroma Maturity from Histological Images. *IEEE Transactions on Biomedical Engineering*, 64(10), 2344–2352. doi: 10.1109/TBME.2017.2665602.

Rubin, R., Strayer, D. S. & Rubin, E. (2012). *Rubin's pathology : clinicopathologic foundations of medicine* (ed. sixth). Lippincott Williams & Wilkins.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *Intl Journal of Computer Vision*, 115(3), 211-252. doi: 10.1007/s11263-015-0816-y.

Schroff, F., Kalenichenko, D. & Philbin, J. (2015, June). FaceNet: A Unified Embedding for Face Recognition and Clustering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823.

Song, J.-W., Lee, J.-H., Choi, J.-H. & Chun, S.-J. (2013). Automatic differential diagnosis of pancreatic serous and mucinous cystadenomas based on morphological features. *Computers in Biology and Medicine*, 43(1), 1–15. doi: https://doi.org/10.1016/j.compbiomed.2012.10.009.

Song, Y., Zou, J. J., Chang, H. & Cai, W. (2017). Adapting fisher vectors for histopathology image classification. *2017 IEEE 14th international symposium on biomedical imaging (ISBI 2017)*, pp. 600–603.

Spanhol, F. A., Oliveira, L. S., Petitjean, C. & Heutte, L. (2016). Breast cancer histopathological image classification using Convolutional Neural Networks. *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 2560-2567.

Spanhol, F. A., Oliveira, L. S., Cavalin, P. R., Petitjean, C. & Heutte, L. (2017). Deep features for breast cancer histopathological image classification. *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1868-1873.

Stegmüller, T., Bozorgtabar, B., Spahr, A. & Thiran, J.-P. (2023). Scorenet: Learning non-uniform attention and augmentation for transformer-based histopathological image classification. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 6170–6179.

Sudharshan, P., Petitjean, C., Spanhol, F., Oliveira, L. E., Heutte, L. & Honeine, P. (2019). Multiple instance learning for histopathological breast cancer image classification. *Expert Systems with Applications*, 117, 103–111.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *IEEE/CVF Conf Comp Vis Patt Recog*, pp. 2818–2826.

Thuy, M. B. H. & Hoang, V. T. (2020). Fusing of deep learning, transfer learning and gan for breast cancer histopathological image classification. *Advanced Computational Methods for Knowledge Engineering: Proceedings of the 6th International Conference on Computer Science, Applied Mathematics and Applications, ICCSAMA 2019 6*, pp. 255–266.

Torre, L., Bray, F., Siegel, R., Ferlay, J., Lortet-Tieulent, J. & Jemal, A. (2015). Global cancer statistics, 2012. *CA Cancer Journal for Clinicians*, 65(2), 87-108. doi: 10.3322/caac.21262. cited By 9258.

Torre, L. A., Islami, F., Siegel, R. L., Ward, E. M. & Jemal, A. (2017). Global Cancer in Women: Burden and Trends. *CEBP Focus: Global Cancer in Women*, 26(4), 444-457. doi: 10.1158/1055-9965.EPI-16-0858.

Vanderbeck, S., Bockhorst, J., Komorowski, R., Kleiner, D. E. & Gawrieh, S. (2014). Automatic classification of white regions in liver biopsies by supervised machine learning. *Human Pathology*, 45(4), 785–792. doi: https://doi.org/10.1016/j.humpath.2013.11.011.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30, 1-11.

Vesal, S., Ravikumar, N., Davari, A., Ellmann, S. & Maier, A. (2018). Classification of breast cancer histology images using transfer learning. *Image Analysis and Recognition: 15th International Conference, ICIAR 2018, Póvoa de Varzim, Portugal, June 27–29, 2018, Proceedings 15*, pp. 812–819.

Veta, M., van Diest, P. J., Willems, S. M., Wang, H., Madabhushi, A., Cruz-Roa, A., Gonzalez, F., Larsen, A. B. L., Vestergaard, J. S., Dahl, A. B., Cireşan, D. C., Schmidhuber, J., Giusti, A., Gambardella, L. M., Tek, F. B., Walter, T., Wang, C.-W., Kondo, S., Matuszewski, B. J., Precioso, F., Snell, V., Kittler, J., de Campos, T. E., Khan, A. M., Rajpoot, N. M., Arkoumani, E., Lacle, M. M., Viergever, M. A. & Pluim, J. P. W. (2015). Assessment of algorithms for mitosis detection in breast cancer histopathology images. *Medical Image Analysis*, 20(1), 237–248. doi: https://doi.org/10.1016/j.media.2014.11.010.

Wahab, N., Khan, A. & Lee, Y. S. (2019). Transfer learning based deep CNN for segmentation and detection of mitoses in breast cancer histopathological images. *Microscopy*, 68(3), 216–233.

Wan, T., Zhang, W., Zhu, M., Chen, J., Achim, A. & Qin, Z. (2017). Automated mitosis detection in histopathology based on non-gaussian modeling of complex wavelet coefficients. *Neurocomputing*, 237, 291–303. doi: https://doi.org/10.1016/j.neucom.2017.01.008.

Wang, X., Hua, Y., Kodirov, E., Hu, G., Garnier, R. & Robertson, N. M. (2019, June). Ranked List Loss for Deep Metric Learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5207–5216.

Wang, X., Yang, S., Zhang, J., Wang, M., Zhang, J., Huang, J., Yang, W. & Han, X. (2021). Transpath: Transformer-based self-supervised learning for histopathological image classification. *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part VIII 24*, pp. 186–195.

Yang, H., Kim, J.-Y., Kim, H. & Adhikari, S. P. (2019). Guided soft attention network for classification of breast cancer histopathology images. *IEEE transactions on medical imaging*, 39(5), 1306–1315.

Zerhouni, E., Lányi, D., Viana, M. & Gabrani, M. (2017, apr). Wide residual networks for mitosis detection. *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, pp. 924–928. doi: 10.1109/ISBI.2017.7950667.

Zhi, W., Yueng, H. W. F., Chen, Z., Zandavi, S. M., Lu, Z. & Chung, Y. Y. (2017). Using transfer learning with convolutional neural networks to diagnose breast cancer from histopathological images. *Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14–18, 2017, Proceedings, Part IV 24*, pp. 669–676.

Zhou, X., Li, C., Rahaman, M. M., Yao, Y., Ai, S., Sun, C., Wang, Q., Zhang, Y., Li, M., Li, X. et al. (2020). A comprehensive review for breast histopathology image analysis using classical and deep neural networks. *IEEE Access*, 8, 90931–90956.