# Biologically-Inspired Hierarchical Learning Architectures for Image Classification

by

Niki Martinel

Dissertation submitted to the
Polythecnic Department of Engineering of the
University of Udine in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
in Industrial and Information Engineering
XXIX Cycle

Supervisor:
Prof. Gian Luca Foresti

To my father, Mauro

# Abstract

This thesis focuses on the topics of biologically inspired hierarchical machine learning methods for object classification.

Mimicking the human brain to achieve human-level cognition performance has been a core challenge in artificial intelligence research for decades. Humans are very efficient in capturing the most important information while being exposed to a plethora of different stimuli, a capability that is used to represent and understand their surroundings in a concise fashion. Think about a kid that learns how to categorize objects through, either labeled or unlabeled, samples. It is a matter of fact that he/she is able to grasp the object concept by processing a few samples. This strong evidence highlights the existence of an extremely complex and engineered cortical mechanisms that allows such efficient learning.

This Thesis draw inspiration from the fascinating biological brain organisms and its ability to learn simple concepts as well as complex notions from a few samples to introduce novel hierarchical learning models. It also grounds on the belief that the study of the vision sensory domain can provide a uniquely concrete grasp on the relevant theoretical and practical dimensions of the problem of learning in hierarchies. Thus, this Thesis provides an in-depth investigation of biologically-inspired hierarchical learning architectures for image classification.

Pivoting on the belief that decomposability of the sensory data is a fundamental principle for learning good representations, the proposed hierarchical learning architectures adhere to such a property by aggregating simple features into more and more complex patterns as the structure becomes deeper and deeper. The underlying idea shared by these models is is to finally provide a different –artificial– hierarchy of computations that mimics the human brain by abstracting away from existing highly-"engineered" models that are quite in vogue (e.g., Deep Neural Networks). To support each approach, experimental results on public datasets are conducted. Results demonstrate that exploitation of subsequent filtering and pooling strategies are the main ingredients of a hierarchical architecture able to build meaningful data representation.

# Acknowledgments

First and foremost, I would like to express my deepest gratitude to my supervisor and mentor, Prof. Gian Luca Foresti for his constant support and guidance throughout these years. He introduced me to the interesting and challenging machine learning problems that absorbed most of my professional life over the past few years. I'm also grateful to Prof. Christian Micheloni and Prof. Claudio Piciarelli for their support and guidance. They were always a source of inspiration for many of the ideas that are in this Thesis.

I'm very grateful to many of my teachers at different institutions who instilled in me the quest for knowledge and the courage to probe the unknown. I would also like to thank many of my colleagues and friends for their help and advice, without which this Thesis would not have seen the light. I thank my fellow labmates for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last years. A special mention goes to Lubos Vaci for his endless complains, Matteo Chini that brought new fresh in our lab with his cross-pollination knowledge, Marco Vernier for his moral support and Mattia Zanier who kept a sense of humor when I had lost mine. An honorable mention also goes to the microwave hidden in the closet which warmed us in the cold winter days and the American coffee machine which has not been sufficiently strong to fight the Mother Russia. What a cracking place to work!

Words cannot express my gratitude and indebtedness to my parents, who have given up so much in life in order that I could reach this stage today. Mom and Giulia, you knew it would be a long and sometimes bumpy road, but encouraged and supported me along the way. Thank you. Dad, you were often in my thoughts on this journey – I miss you.

And finally, last but by no means least, I would like to express my deepest gratitude to my sweetheart Elisa. Her support, encouragement, quiet patience and unwavering love were undeniably the bedrock upon which the past years of my life have been built. She will always be a source of inspiration for me.

# Contents

# List of Figures

# List of Tables

# 1

# Introduction

*The first chapter briefly introduces the motivation behind the investigation of hierarchical learning architectures. Within this context, the contributions and the organization of this thesis are defined.*

## 1.1 Hierarchical Learning

Chomsky's Poverty of the Stimulus (POS) argument is one of the most famous and controversial arguments in the study of the human mind [23]. Specifically, as regards human learning, the POS argument captures the notion that biological organisms can learn complex concepts and tasks from an exceptionally small set of samples. Think about a kid that learns how to categorize objects through, either labeled or unlabeled, samples. It is a matter of fact that he/she is able to grasp the object concept by processing a few samples. This strong evidence highlights the existence of an extremely complex and engineered cortical mechanisms that allows such efficient learning.

The idea that complex systems have a hierarchical modular organization dates back in the early 1960s [157] and has recently attracted fresh support from quantitative studies of large scale, real-life mammalian brain networks [120]. These have demonstrated that human brain functional networks have a hierarchical organization which is significantly common between different individuals. Despite this similarities, little is known as to how exactly the neural connections adapt and perform efficient learning to achieve high degrees of invariance to complex transformations. It has been largely speculated that there exist a strong link between the human learning ability and the computational hierarchical organization of the human brain. Indeed, it has been hypothesized that circuits found in the human brain grounds on such a structure to promote modularity and reuse of redundant sub-circuits. This encounters the energy and space efficiency objectives that are inherited in the human nature. Though this is intuitively compelling, no solid theoretical foundation supports such an hypothesis.

Classical results from the brain-inspired learning theory [62, 102] have shown that, up to an arbitrary degree of accuracy, any continuous function can be approximated from an (infinite) set of samples by exploiting a single *layer* of computational learning elements. While such theories are proven and seems plausible, we argue that decomposability of the sensory data is a fundamental principle that supports the process of learning good representations. In particular, we will consider hierarchical learning architectures that exploit such a property by aggregating simple features into more and more complex patterns as the structure becomes deeper and deeper. Empirical results will demonstrate that exploitation of subsequent filtering and pooling strategies are the main ingredients of a hierarchical architecture able to build meaningful data representation.

The Thesis grounds on the belief that the study of the vision sensory domain can provide a uniquely concrete grasp on the relevant theoretical and practical dimensions of the problem of learning in hierarchies. Thus, this Thesis provides an in-depth investigation of biologically-inspired hierarchical learning architectures for image classification. Its basis draws inspiration from the fascinating human cognition system and aims to introduce a further step towards a deeper understanding of the mammalian brain. Following the current surge of effort that the vision and pattern recognition communities are giving to hierarchical learning solution, we seek to design novel learning models on the basis of anatomical and physiological data describing our visual cortex. The underlying idea is to finally provide a different –artificial– hierarchy of computations that mimics the human brain by abstracting away from existing highly-"engineered" models that are quite in vogue (e.g., [92, 158, 166, 54]).

Within this context, we start by conducting an empirical analysis that ties the discrimination and invariance properties of the biological vision with functional modules and computer vision theory of wavelets. In the course of such an exploration, the role of hierarchical structures that leverages on the combination of different visual clues visibly emerges. This analysis also takes a step towards demonstrating the importance of the groundbreaking work of Hubel and Wiesel [72] which gave a stimulus of paramount importance to the study of visual hierarchies analogous to the primate visual system.

Then, we follow the recent neuroscience discoveries regarding the evidence for parallel visual systems in the brain that are associated with different levels of visual clues [123, 61, 161]. The work experimentally evaluates such parallel assumptions by studying the effects of different visual representations within a framework consisting of shallow architectures. We adopt the notion of an ensemble committee-based model that, using a representation expressed by different visual appearance clues such as color, texture, shape and filter features, is able to identify important patterns to distinguish among image classes. In support of the approach, we conduct extensive experiments which show classification results that are on par or surpass the state-of-the-art.

A solution that leverages on the process of learning visual feature descriptors instead of using hand-crafted solutions is then introduced. The approach capitalizes

on the deep learning wave of enthusiasm by introducing a hierarchical learning architecture that is able to both identify the most discriminative features to describe an image and, at the same time, to correctly classify it. Exploitation of a recent residual learning scheme together with the introduction of a feature detector that is conceived to capture specific structured image information are considered. To validate the solution, we report on a large set of different experiments on multiple publicly available benchmark datasets. Results show that existing approaches are outperformed by a significant margin.

Lastly, we consider the problem that all the aforementioned approach suffer from a significant limitation that reside on the fact that such schemes require a substantial manual work to select and tune many algorithm hyperparameters (e.g., the number of hidden neurons/layers in an artificial neural network). Such a consideration find abundant support by the literature in cognitive studies [6, 76], which demonstrated that the human learning is a complex phenomenon requiring flexibility to create new/adapt existing brain functions to perform new neurophysiological activities to drive the desired behavior. This adaptability is embodied in the modular structure, which plays a critical role in evolution, development, and optimal network function. Thus, the brain network does not dwell on any external source/knowledge/decision to construct/modify its structure, but autonomously adapts through evolution. On the basis of such motivations, we introduce a hierarchical architecture that yields to more complex data abstractions as the information flows through more and more deeper processing layers. The hierarchical structure automatically adapts itself to the problem complexity by introducing new computational units. Preliminary results demonstrate the benefits of the solution which obtains similar performance to current state-of-the-art methods.

The vision-related work in this Thesis parallels the theoretical and cognitive brain findings by sharing the notion of a localized spots that are present in a restricted retinal region [28, 94]. This is combined with the layered analysis that is largely backed by mammalian brain analyses. Thus, the Thesis aims to introduce new computational architectures that adhere to the human models as discovered in the early stages of the visual cortices and in recent functional networks of the ventral visual stream. To support the underlying assumptions built into the abstract formalisms, all the introduced solutions are extensively evaluated in the context of a difficult, real-world learning task.

The remainder of this chapter is organized as follows. We start by listing the most relevant motivations that lay down the basis for the choice of hierarchical learning architectures over shallow alternatives. Then, a brief literature survey covering the most relevant works in the field is provided. To conclude, we draw the list of contributions and define the overall organization of the Thesis.

## 1.2 The Motivation

Why should hierarchical learning architectures be the best choice for image classification? Is it possible to emulate the human brain solution to the visual recognition problem with a learning scheme that does not hinge on the manual tuning of its hyperparameters? These and other related questions are the focus of this Thesis.

First and foremost we draw inspiration from the fascinating biological brain organisms and its ability to learn simple concepts as well as complex notions from a few samples. This establishes the ultimate goal of Artificial Intelligence (AI): is it possible to emulate/transfer this remarkable ability to machines? Addressing the problem in a whole would be cumbersome, thus we focus on the vision problem which has been recently shown to be one of the most promising windows into human intelligence. Interestingly, by following the hierarchical processing in the primate visual system that constitute a deep hierarchy of computations, we contrast the flat vision architectures that are still largely considered in mainstream computer vision and pattern recognition [94]. Within this context we believe that the deep hierarchies in the primate ventral system carry extremely valuable insights that should be considered to design AI algorithms, fostering increasingly interaction between biological and AI research.

It is a matter of fact that the human brain is organized in a hierarchical fashion with large circuit modularity and substantial reuse of general sub-circuits that yield to space and energy minimization. Such a feature is captured by a hierarchical model in which initial processing layers act as feature detectors producing information that is general and yet exploitable in the context of many specific classification problems. By going deeper in the hierarchical architecture, more task-specific information is captured. Such an aspect, connected with the fact that naturally occurring phenomena can be largely addressed by exploiting a "divide and conquer" approach, seems to indicate that hierarchical models are indeed a valuable learning scheme. It has been also empirically demonstrated that hierarchical learning solutions are ideally suited to domains and tasks which decompose into parts.

A final, yet relevant source of motivation for the work presented in this Thesis comes from the recent widespread success of recent deep hierarchical learning architectures in a plethora of different tasks (refer to the following literature survey for details). Despite their astonishing results, such models largely depend on a "trial-and-error" approach to determine the number and arrangement of the computational units which they are composed of. There have been efforts in addressing such a problem by starting with a large network, using more neurons than are expected to be needed, and then removing the neurons that have little effect on the final error. However, this solution is not only computationally inefficient (which is now still a problem, considering that one might need a week or more before obtaining any result) but, more importantly, it does not guarantee that an optimal architecture configuration is obtained. This, together with the evidence showing that organization of the human brain connection is not fixed but adapts dependently on several conditions, motivates the study of a hierarchical scheme that is not pivoted on the human manual labor.

## 1.3 Literature Survey

In this survey we discuss previous work related to the matter that will be presented in this Thesis. We begin with a review of the prior work involving highly-engineered hierarchies learning solutions, with particular focus on deep neural networks (DNN). Then, since our work grounds on neural trees solutions, we conclude by giving a review of the literature in such a field.

### 1.3.1 Deep Neural Networks

An artificial neural network (ANN) consists of many simple processing units, called *neurons*, each of which generates a real-valued *activation* output. Measurements obtained from environment sensors are assigned to the input neurons. Following neurons get activated through weighted connections from previously active neurons. The goal of an ANN is to find the optimal connection weights between *layers* of neurons such that an optimality criterion is met. This will make the ANN exhibit a desired behavior, such as recognizing the object in an image, describing its content or even driving a car in an autonomous fashion. Depending on the specific problem under consideration and on how the layers of neurons are connected together, long causal chains of computational stages that transform (often in a non-linear way) the aggregate activation of the network may be needed.

*Shallow* ANNs consist of such hierarchical architectures that have few such stages. These schemes have been around for many decades if not centuries. Probably, the earliest work in the field, which inspired a huge amount of researches, is the world famous work by McCulloch and Pitts [118]. It describes that the character of the human nervous activity, which is composed of a sequence of neural events and relations among them, can be treated as a network of logical expression units. Starting from such work, the activity in supervised ANNs saw a further boost with another highly relevant work. That is the Rosenblatt paper on the "perceptron" [141], the basic computational ANN unit.

It is worth noticing that, in some sense ANNs have been on the stage for even longer, since the initial models were essentially variants of linear regression methods which date back to the early 1800s [101, 39].

Models with more nonlinear layers of neurons were later proposed. This was due to the discoveries that Hubel and Wiesel [72] did on the cells found in the cat's visual cortex. The two Nobel awarded researches found that particular visual inputs such as specific orientation and edges stimuli make the animal cells fire. The fact that complex cells exhibit more spatial invariance than simple cells inspired the deeper nonlinear architectures that are somehow still driving the results of modern award-winning Deep Learning solutions.

A particular work that is probably deserving to be called the first "deep neural network" is the work *Neocognitron* system that was proposed by Fukushima in the

1980 [38]. It was the first work that, largely inspired by Hubel and Wiesel, tried to incorporate the neurophysiological insights in an artificial neural network system. More interestingly it introduced the Convolutional Neural Networks (today simply called CNNs or convnets) which exploits a particular connection in subsequent layers of neurons. Such a connection let the convolutional units look only at a small portion of the input (typically a rectangular region, also known as the receptive field). The weights of the convolutional unit define a convolutional kernel (or filter) which is shifted step by step across a (generally) 2-D array of input values, such as the pixels of an image. The output of such a process is a feature map of filter responses which defines the input to the higher-level units. Due to the fact that convolutional layers only require adaptation of the filter parameters, such an architecture has massive weight replication, hence it is more computational efficient than its *fully-connected* counterpart (in which the weights to be adapted are determined by the number of neurons in two subsequent layers).

Another feature that the Neocognitron system introduces to the field is given by the subsampling (also known as pooling or downsampling) layer. This consists of units that generally have fixed-weights connections with physical neighbors in the preceding convolutional layer. Such units aim to answer to the insensitiveness to certain small image shifts which the mammalian visual cortex showed. Nowadays, such subsampling units become active if at least one of their inputs is active (i.e., *max pooling*). In the Neocognitron architecture, convolutional layers are alternated to Spatial Averaging downsampling layers. Moreover, differently from modern solutions the weights are not adapted via a supervised scheme but through a local winner-takes-all unsupervised learning rules or by pre-wiring. Thus, despite the architecture was comparatively deep, the whole solution was not addressing the common deep learning problem in which all the weights are adapted end-to-end.

The answer to such an issue was not long in coming. Indeed, it was already in the 1970s that efficient error backpropagation (BP) was developed for arbitrary, discrete, possibly sparsely connected, ANN-like networks [105, 106], albeit without reference to NN. More ANN-specific works were published a few years later in [179] and in [13] with a method for multilayer threshold ANNs. Despite this, the paper that let BP emerge and be very significant for ANN training was proposed in [142]. Such a work experimentally showed that representations captured by the hidden layers were extremely useful and highly-driven by the task domain.

It was however, only in the early 1990s that BP for ANN become an explicit research subject. Specifically, the diploma thesis by Hochreiter [59] lay down an extremely important milestone for the development of deep learning schemes trained via BP. Hochreiter's thesis formally identified the reason behind the difficulties in training traditional deep feed-forward or recurrent networks by BP: the now famous problem of *vanishing or exploding gradients*. This should be mainly attributed to the standard activation functions which yield to a rapid shrink or explosion of the backpropagated error signals which sooner or later approach to zero or grow out of bounds. Delineating such an insight in a formal way opened to the abundant research

that came years later to avoid suffering from the vanishing gradients problem.

Since then, a surge of effort has been devoted to understand single (hidden) layer ANNs in terms of statistical learning theory [42, 132, 67]. Despite this, there is little work that attempts to do the same for multilayer ANNs with nonlinear activation functions. In the context of regularization, the ANN community has largely resorted to a handful of effective but little understood heuristics such as weight decay and pruning techniques [12].

Although the discovery of such findings and the moderate success of ANNs on different tasks, the decade around 2000 saw the decline of such models in favor of non-neural machine learning methods such as Support Vector Machines (SVMs) [173, 150, 124] which had a more robust theoretical background and proved to be more effective. This happened despite the publication of the BP-trained CNN development obtained by LeCun [100], which is considered the most important milestone for the many present competition-winning pattern recognizers.

After such a decline, the study of deep belief networks rekindled the interest on ANNs within the machine learning community. Although the notion of a deep architecture was not new, Hinton and Salakhutdinov [57] came up with a novel training algorithm that leverages on a set of stacked layer-wise pretrained Restricted Boltzmann Machines (RBMs). After the pretraining, the RBMs are "unrolled" to create a deep autoencoder, which is then fine-tuned using BP. This work partially responded to the major issues on deep neural networks trained with BP which generally determined slow converge to poor local minima for most practical problems. Specifically, authors demonstrated that the combination of deep autoencoders unrolled from pretrained stacked RBMs was able to obtain excellent classification performance on practical applications such as handwritten digits and human face recognition.

Since then, plenty of relevant works involving deep architectures have been published. The majority of them can be categorized into two main groups: i) empirical applications and solutions to practical applications, and ii) theoretical attempts aiming to understand why deep networks work so well.

Due to its impact on real world, to date, the majority of the literature is devoted on efficient approaches for an effective training and inference in deep model as well as on empirical experiments covering very different application domains (see [149]). Since this Thesis focuses on a more practical approach, the following review will cover the most relevant works that share a similar viewpoint. We will nevertheless begin our discussion by presenting few works that have a more theoretical background.

An interesting paper on this matter was proposed by Bengio and LeCun [8]. In their work, authors started by considering boolean function learning in support to the use of deep networks, then listed the advantages of a multi-layer architecture over a Gaussian kernel SVM by discussing the fact that such a solution would ostensibly require more training patterns. Despite the significant set of intuitively appealing and informal arguments in favor of deep networks, the work was limited to specific examples that do not provide a rigorous justification. An additional step towards having a

more significant grasp on the expressive power of deep models was proposed in [99]. Authors first proved that adding hidden units in an RBM yields strictly improved modeling power, then showed that RBMs are universal approximators of discrete distributions. A first significant finding was that the performance of a two-layer network is limited by the representational ability of the first layer, thus suggesting that a first layer with a huge number of neurons is preferable. The second important result was to demonstrate that additional RBMs layers yield to better generalization, rather than introducing more representational power.

After the publication of such works, another milestone towards the deep learning era, was set up by [7]. The paper throughly discussed the motivations and principles behind recent learning algorithms for deep architectures. Specifically, it analyzed those learning schemes that exploits unsupervised learning of single-layer models as building blocks (e.g., RBMs) to construct deeper models, such as the Deep Belief Networks (DBNs).

As regards more practical approaches, it is safe to state that the deep learning era started with the world-famous paper by Krizhevsky *et al.* [92]. With their breakthrough paper, they trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet dataset. As a result, a top-1 and top-5 error rates of 37.5% and 17.0% were achieved, which at the time corresponded to a 10% improvement over the state-of-the-art.

After such work, a huge amount of research has been devoted to the design of more complex deep architectures. A lot of effort was spent to learn optimal feature representations [9], either by considering labeled [25] or unlabeled data [98, 31, 81]. Then, a more application-driven approach has been considered to tackle many visual perception tasks such as object [104, 29, 51, 162, 158, 166, 54, 186] and action recognition [84, 34, 26], image segmentation [43, 189, 50], visual question answering [195, 152], image captioning [56, 184, 79], natural language processing [64], etc..

Despite the multitude of interesting and innovative aspects of each of such solutions, we believe that it is worth describing more in detail only those more relevant works that opened to new research directions. In [158], authors introduced the well known *VGG*-network architecture. Authors demonstrated that by increasing depth in an architecture with very small (i.e., $3 \times 3$) convolution filters, a significant improvement on the prior-art configurations could be achieved by pushing the depth to 1619 weight layers. These findings let the work be on the top of the leaderboard for the 2014 ImageNet Challenge [143]. In addition, by exploiting the architecture pretrained on the ImageNet dataset to extract high-level representation on a different domain test set, authors demonstrated the robustness of the obtainable representations (see also [153, 29]). More or less at the same time, the Google research team, proposed the *GoogLeNet* network [166] which improved utilization of the computing resources inside the architecture. Following a a carefully crafted design, depth and width of the network were increased without incrementing the computational budget.

Even though the architecture were deeper, these still suffer from training difficulties. To try to overcome such a problem, in [54] authors reformulate the common

architecture layers such that residual residual functions with reference to the layer inputs are learned in place of unreferenced functions (i.e., authors introduced the *ResNets*). Such an approach facilitates the training of networks that are substantially deeper than those used previously (e.g., VGG [158] and GoogLeNet [166]) and have lower complexity. ResNets are able to scale up to thousands of layers and still have improving performance. However, each fraction of a percent of improved accuracy costs nearly doubling the number of layers, hence opening to the diminishing feature reuse problem [162] which makes these networks very slow to train. To tackle these problems, in [186], authors thoroughly analyzed the ResNet building blocks to introduce a novel architecture which is less deep but has more feature maps per layer. The resulting wide residual networks (*WRNs*) demonstrated to be far superior over thin and very deep ResNet counterparts.

The aforementioned works introduced novel generic architectures that were however evaluated on recognition tasks. A highly relevant work that addressed a slightly different problem was introduced [43]. Authors proposed a simple and scalable algorithm that combines CNNs with bottom-up region proposals in order to localize and segment objects. This work is better known as *R-CNN*: Regions with CNN features. Follow-up works exploited spatial pyramid pooling layers (*SPP*) [51], considered the problem as a regression problem to spatially separated bounding boxes and associated class probabilities (*YOLO*) [140], and simultaneously addressed object detection and segmentation (*Mask R-CNN*) [50].

Last but not least it is necessary to mention the deep generative models. Training such models has been extremely difficult because of the many intractable probabilistic computations that arise in maximum likelihood estimation and related strategies, and due to difficulty of leveraging the benefits of piecewise linear units in the generative context. To address such problems, in [44] authors introduced what is by LeCun "most interesting idea in the last 10 years in Machine Learning". That is, exploit a generative adversarial nets (*GAN*) framework, in which the generative model is pitted against an adversary: a discriminative model that learns to determine whether a sample is from the model distribution or the data distribution. This is reminiscent of a police-counterfeiters setting in which, through learning, the generative model aims to learn how to forge samples that are indistinguishable from the genuine articles while the discriminative model aims to detect the counterfeit currency. Since then, efforts have been spent to investigate a Laplacian pyramid framework to generate images in a coarse-to-fine fashion [27], to consider both the structure (the underlying 3D model) and the style (the texture mapped onto structure) of the samples [177], as well as to study unsupervised [135] and energy-based[190] models.

Little is known however, as to why deep networks work well, and why or when particular architectural choices (number of layers, number of units per layer, etc.) lead to optimal performance. This is the main motivation that opened to the research in other learning architectures that aim to overcome such limitations.

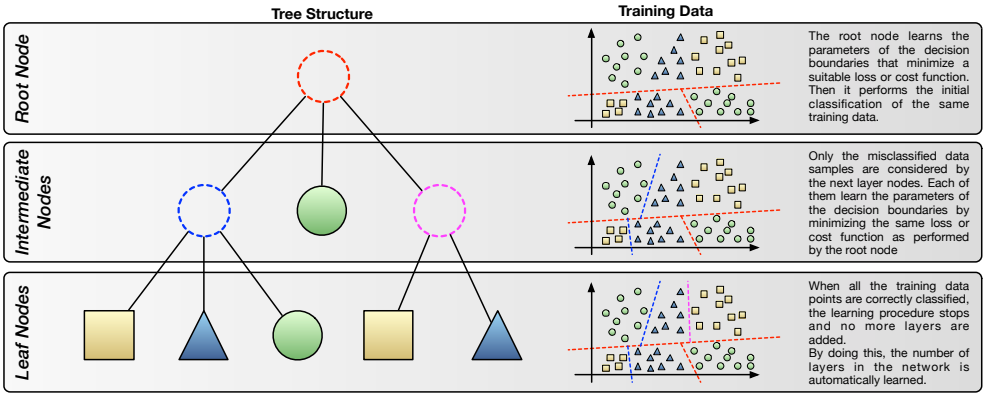



Figure 1.1: A general representation of a Neural Tree architecture trained on patterns belonging to $K = 3$ different classes.

### 1.3.2 Neural Trees

Despite the limited amount of research compared to the current deep learning literature, neural trees (NTs) are a quite old field of research. Indeed, NTs were born in the 1984 when Breiman *et al.* used linear threshold units as the nodes of a tree [18].

The creation of such an architecture was motivated by the shortcomings, that at the time, were encountered while working with ANNs. More specifically, the ANN-related literature presented in the preceding paragraphs have demonstrated that in the mid 1980s such architectures were already able to successfully tackle a variety of tasks by performing a significant set of computations in a highly-parallel fashion. To address the non-linearly separable problems, deeper version were designed. However, training such models should be considered an NP-complete problem since the number and arrangement of hidden units must be determined before training starts. Even if the number of hidden layers is fixed, it is still necessary to find the optimal number of units in each layer. Currently, there is no proven method to solve these limitations. A classical approach to address the problem is to start with a large network, using more neurons than are expected to be needed, and then removing the neurons that have little effect on the final error. It is a matter of fact that such a "trial-and-error" solution is highly inefficient and, more importantly, it does not guarantee that an optimal network configuration is obtained.

The NT architecture was conceived with the aim to overcome these limitations whilst exploiting the pros of ANNs. This is the key aspect that aims to eliminate the basic ANN problem, i.e., determining how many hidden layers/hidden neurons are needed to address a specific task. The first architecture resembling an NT can be dated back to 1984, when Breiman, Friedman, Olshen and Stone used linear threshold units as the nodes of a Decision Tree (DT) [18]. Even though such a hierarchical learning scheme has no ANN inside, its ideas largely influenced the upcoming research.

The procedure to train an NT is the same that was initially exploited to train the perceptron tree [171]. Everything starts with a single root node, then child nodes are added when needed. The root node is trained with whole training set. Once the training procedure for the model in the root node is completed, the learned model is exploited to classify the input set of data patterns. As a result this is partitioned into $K$ number of groups, where $K$ is the number of distinct classes in the input set. Patterns belonging to a specific group are then used to build a child node. Thus, $K$ child nodes are created. The process continues in a recursive fashion, with each of the subsequent nodes having a maximum of $K$ children. If a particular input set satisfies particular stop conditions, the group of data is considered to be homogeneous, hence the node is considered as a leaf node. Such a node has no model in it to train. The leaf node simply specifies the classification label/probability that is to be assigned to the patterns that reach it. A common representation of an NT model is depicted in Figure 1.1.

During the classification stage, the learned NT model is used to assign a specific label to previously unseen patterns. As for the training stage, any new pattern is first input to the root node. The in-node model computes the activation values for the new pattern according to the learned parameters, then passes the datum to the child node corresponding to the maximum activation value. This top-down process continues until a leaf node is reached which is in charge to assign the final classification label to the pattern.

The first work that exploited an ANN inside the nodes of a DT was proposed in the late 1980s by Utgoff [171]. In such a work the author introduced the perceptron Tree which alternates nodes with attribute tests and perceptrons. More in details, the perceptron tree learning solution starts by training a perceptron at the root node. Then, if the patterns input to a perceptron node, such as the root node, are not linearly separable, the note is replaced with a decision function that exploits an attribute test to split the input data into two groups. This results into two new child nodes that contain the patterns that lie above or below the learned threshold, respectively. The major shortcoming of this work was to trash the efforts spent by the perceptrons that are replaced by the attribute test functions.

As the research progresses in the field, the typical structure of the NT and its in-node models changed. In [159], authors proposed a binary NT, in which perceptrons are used at each node of the tree. Later, in [146], authors introduced a NT Network scheme which consists of of single-layer ANNs connected in a tree structure. The learning process considered the $\ell_1$-norm (measured as the distance between generated outputs and targets). To address the multi-class classification problems, a local encoding scheme that classifies an input pattern according to a winner-take-all rule was adopted. Thus, each in-node ANN model partitioned the input space into regions, which are then considered by child nodes.

The efforts in linking ANNs with NTs continued with the research being addressed towards the possible extraction of an NT structure such that it can be converted into an efficient ANN. The main idea is simple, yet very challenging: build a DT,

then convert it into an ANN. One of the first work in this direction was the Entropy Network proposed in [154] which introduced a procedure to shape a DT architecture into a multilayer ANN. The idea was analyzed more in details in [19], with a thorough complexity analysis and practical advancements. Later, the exploitation of linear DTs as the building blocks of an ANN was investigated in [129] Another interesting approach was proposed by modifying the ID3 algorithm [134] such that it can be exploited to convert decision trees into hidden layers [24]. During the learning process, the network is modified by adding new hidden until the task becomes linearly separable at the output layer. Thanks to this, self-generation of an ANN architecture is obtained.

Other works introduced hybrid NT solutions in which DTs and NNs are fused into one structure. In [37], the split node was introduced to divide the training set into two parts when the current in-node model performs the same classification of the parent node. The resulting learning strategy was demonstrated to converge in all the cases. Symbolic learning was explored to simulate the human reasoning process with an Hybrid Decision Tree (HDT) [193]. A Generalized Neural Tree (GNT) model was more recently proposed in [36]. The main contribution of such work was the definition of a novel training procedure that yields an overall optimization of the tree. However, it requires that each time the tree grows by including an additional child, the whole tree is parsed. It also trains the in-node models not using the patterns that reach a particular node only, (i.e., the LTS), but instead considers all the data and a weighted correction-rule that conveys information regarding the whole architecture, in-node models included. The procedure let a father node update its weights on the basis of the classification performed by its children. Interestingly, results showed that this often corresponds to a reduction in the tree dimension. An evolution of the GNT, was the Adaptive High-order Neural Tree (AHNT) [35], which considered high-order perceptrons (HOP) instead of simple perceptron. With such a solution first-order nodes can be exploited to divide the input space with hyperplane in case of linearly separable data. When such a characteristic is not present, HOPs can be considered to divide the input space with more flexibility, but at the expense of increased complexity. The proposed learning solution also considers the depth of the tree to favor HOPs at first levels. Moving toward leaf nodes, low-order perceptrons are favored as LTSs are easier and smaller. Such a scheme inherits from the results showed in [183], where authors observed that the complexity of the nodes affects the size of tree: a tree with complex in-node models may be quite small, while a tree with simple in-node models tends to become very large.

Pruning strategies were also investigated. In [109], a uniformity index was adopted to model the degree of correctness at each node. Results showed that consideration of such an index yields to a reduction in the depth of the tree with a good classification accuracy. Despite this outcome, the work still suffered from the hand-crafted selection of the in-node networks architectures (number of hidden layers and number of nodes for each layer) and the uniformity index. In [1], a new algorithm providing different methods to construct and use multivariate decision trees was proposed. Each node exploits multiple algorithms to split the input data, but only the best one is selected.

Thus, the architecture considers the best fitting algorithm at each node.

Despite the differences with the initial NTs, all the NT-related works have a common feature that characterized the tree-based hierarchical architectures: the structure of a NT is determined on-the-fly during the training process. The basic idea is that dividing the training set into smaller local sets (according to the subregions into which the feature space is divided) is easier than learning the classification surface parameters from the whole set. Such a solution is shared with DTs, but often these methods consider classification hyperplanes that are perpendicular to the axes of the feature space. NTs have no restrictions, hence should be considered more flexible in capturing the different characteristics of the pattern distribution.

## 1.4 Contributions and Organization of the Thesis

The Thesis is organized along the lines described in the previous section. In the following we summarize the main contributions of the Thesis in order by chapter.

**Chapter 2 - A Forest of Random Trees for the Optimal Selection of Feature Encodings**

This chapter provides an empirical analysis of a famous and widely used hierarchical learning scheme. The analysis is conducted on hand-crafted visual description solutions that aim to mimic the process that the human visual cortex perform to grasp the relevant and discriminative image traits. Specifically, we have considered five of the most important texture filter banks and measured their performances when used in image classification tasks. Since the responses of such filters have high dimensionality, we also analyzed the effects of adopting different feature encoding schemes to obtain a compact feature representation. This yields to the basis results and limitations upon which the following chapters build on.

**Chapter 3 - Learning to Rank with a Committee of Shallow Neural Networks**

Regardless of the specific application, image classification is a problem with many challenges which cannot be addressed only by considering simple filter-response features and a unique classification solution. As performed in chapter 2, a large portion of the literature addressed the problem challenges by designing ad-hoc image representations based on some *a priori* knowledge of the problem. Despite their often successful applications, this might not be sufficient to correctly handle all the challenges. In this chapter, a possible solution to sidestep the aforementioned problems is introduced. Specifically, we propose a system that uses as many different features as possible but exploits only a subset of those to perform the classification task. Following this idea, a classification system based on a committee of shallow neural networks is introdu-

ced. In addition, we introduce a Structural Support Vector Machine [170] as the committee supervisor to fuse the discordant members' classifications. The proposed solution showed that improved classification performances as well as optimal rankings are achieved through individual ranking fusion.

**Chapter 4 - Going Deeper and Wider with Wide-Slice Residual Networks**

So far, works exploiting hand-crafted visual feature representations have been explored. However, the recent discoveries on representational learning have demonstrated that learning specific image representations yields to significantly better performances. Specifically, we introduce a deep neural network architecture which introduces the following contributions: i) it leverages on the idea that many object have a specific vertical structure to propose a vertical layer feature detector; ii) combines the so extracted representation with a wide residual learning architecture to obtain a more generic representation. Both such representations as well as the classifier are learned end-to-end in a supervised fashion. Results on benchmark datasets showed that the proposed solution outperforms existing hierarchical architectures. The analysis of the source of performance demonstrated that robust feature representations are learned and, more interestingly, that such solution is able to automatically focus on the image portion that contains the object of interest to perform its classification.

**Chapter 5 - Extreme Deep Learning Trees: The Evolution of Neural Learning Systems**

All the methods previously discussed hinge on the manual specification of multiple hyperparameters (e.g., the number of features to consider, the number of hidden layers/neurons in an ANN, etc.). The process of selecting such hyperparameters can be either done manually or obtained through cross-validation schemes. However, in both cases there is no guarantee that the obtained values are optimal for the task. To overcome such limitations, the last chapter of this Thesis introduces a novel NT-based hierarchical architecture that combines three key ingredients: i) does not require the specification of any of the common ANN hyperparameters; ii) at each node of the tree, performs very fast learning and inference by analytically solving the parameter learning problem in a shallow neural network; iii) extracts relevant image features by means of a convolutional neural network architecture. An analysis of the obtained results on different visual classification tasks has shown the benefits of the solution, both in terms of accuracy as well as with regard to the computational performance.

**Chapter 6 - Concluding Remarks and Future Works**

The Thesis concludes with a discussion on the obtained results and emphasizes the advantages and limitations of hierarchical architectures. In particular, the key aspects of the hierarchical model presented in chapter 5 and its possible contributions on other

tasks are presented. Finally, we provide a speculation on future works in the field.

# 2

# A Forest of Random Trees for the Optimal Selection of Feature Encodings

*In this chapter an empirical in-depth analysis of a famous and widely used hierarchical learning scheme jointly exploited with existing hand-crafted visual description solutions is conducted. Specifically, we have considered five of the most important texture filter banks and measured their performances when used in image classification tasks.*

## 2.1 Introduction

Mainstream computer vision is largely focused to solve specific methods related to specific tasks using hand-crafted visual feature representations of the datum. Such feature-based descriptors are taken as inputs and then processed by the task-dependent learning solutions. This resulted in a divorce with the initial works jointly conducted with biologically related approaches. The only references to such mechanisms were commonly limited to individual functional modules or feature choices such as the filter banks that are related to the V1 area of the human brain [94].

Leveraging on this connection with the human brain, there have recently been a growing interest in such biologically-inspired feature representations. Within this class of approaches, the majority of the existing works in literature agree on the fact that features extracted by texture filters are among the most relevant ones for image recognition tasks. However, there is still a lack of a study showing which features are likely to be more useful among the huge plethora of available ones. The aim of this work is thus to evaluate and compare different approaches based on texture filters, in order to gain a better insight on their performances. In this paper, we have considered five of the most important texture filter banks, namely Laws filters [96], Gabor

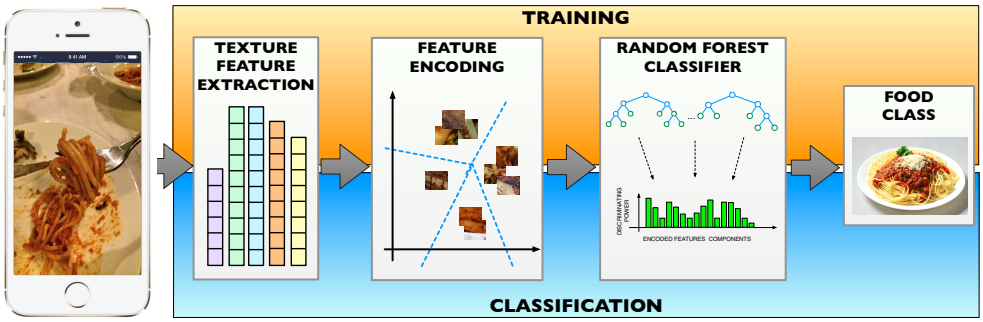



Figure 2.1: Proposed system architecture based on three main stages: (i) texture feature extraction; (ii) feature encoding and (iii) training/classification. *(Best viewed in color)*

filters [178], Schmid filters [148], Leung-Malik filters [103] and Maximum Response filters [174], and measured their performances when used in image classification tasks. Since the responses of such filters have high dimensionality, we also analyzed the effects of adopting different feature encoding schemes to obtain a compact feature representation. The task is performed by computing either the Bag-of-Words, the Fisher Vector or the Vector of Locally Aggregated Descriptors (VLAD) representation for each filter bank. We finally compared the discriminative power of each texture filter alone, and show that better results can be achieved using a proper combination of different filters. This is obtained by using a Random Forest of Decision Trees (RF) [17] classifier, which automatically detects and uses only the relevant features to produce a reliable classification. The main novelty of this work thus lies in giving a better understanding of texture filter approaches through comparative results, despite the adopted techniques are not novel *per se*. This work extends the preliminary results we discussed in [114].

## 2.2 Testing framework

The pipeline of the proposed system is depicted in Figure 2.1. It consists of three main main phases: (i) texture feature extraction, (ii) feature encoding, and (iii) random forest classification.

Since the proposed approach introduces a classification scheme based on a learning mechanism, the system requires a training stage before its deployment. In the training stage, each image is given to the feature extraction module that extracts the texture features by means of filter banks. To reduce the high dimensionality of the obtained features, these are processed by the feature encoding module. The encoding features are finally given to the RF classifier that learns the parameters of the decision boundaries that best separate the image classes.

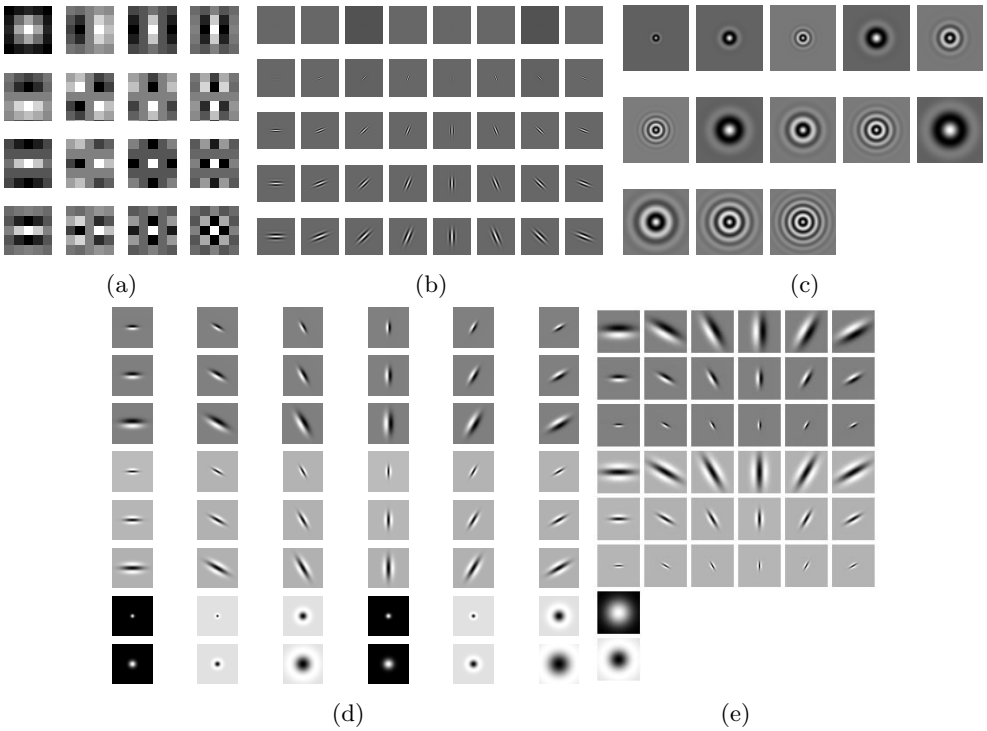



Figure 2.2: (a) Laws filter bank consisting of 16 5x5 masks; (b) Gabor filter bank with 8 orientations and 5 sizes; (c) Standard Schmid filter bank; (d) Leung-Malik filter bank. The set consists of first and second derivatives of Gaussians at 6 orientations and 3 scales making a total of 36 filters; 8 Laplacian of Gaussian filters; and 4 Gaussians. (e) MR8 filter bank. The set consists of 2 oriented filters with 6 orientations and 3 scales plus 2 isotropic filters.

During the classification phase, given a test image to classify, the same types of features are extracted and nonlinear encoding procedure is applied using the learned codebook. Then, the obtained encoded features are given to the trained RF classifier that produces the final classification decision.

### 2.2.1 Texture Feature Extraction

Texture features are of fundamental importance for image recognition. However, at a current state, there is no work that has deeply analyzed the performance of existing filter banks for the given task. This motivates a study of the importance of such filters. In particular, given a grayscale image $\mathbf{I} \in \mathbb{R}^{M \times N}$ of a particular class, we have considered the most widely used bank of filters that produce features robust to object scale and rotations:

**Laws filters** [96] combine multiple texture energy features computed in a local neighborhood to obtain rotation invariance properties. Texture energy features are obtained by combining the 4 Laws mask into 16 2D convolutional kernels (see Figure 2.2(a)). The outputs of the 16 convolutions are then combined into 9 feature vectors which are separately vectorized, then collected in the matrix $\boldsymbol{\Upsilon}(\mathbf{I}) \in \mathbb{R}^{(MN)\times 9}$.

**Gabor filters** [178] with 5 different sizes and 8 orientations have been used (see Figure 2.2(b)). The results of the convolution process with each single filter is then vectorized. The resulting vectors computed for every Gabor filter are finally stacked to obtain $\boldsymbol{\Gamma}(\mathbf{I}) \in \mathbb{R}^{(MN)\times G}$, where $G = 40$ indicates the number of exploited Gabor filters.

**Schmid filters** [148] are obtained by convolution with isotropic "Gabor-like" filters and have rotation invariant properties. A zero DC component of the filter is also computed to achieve invariance to intensity translations. The convolution of the image with the $S = 13$ Schmid filters (see Figure 2.2(c)) results in the stacked filter responses $\boldsymbol{\Psi}(\mathbf{I}) \in \mathbb{R}^{(MN)\times 13}$.

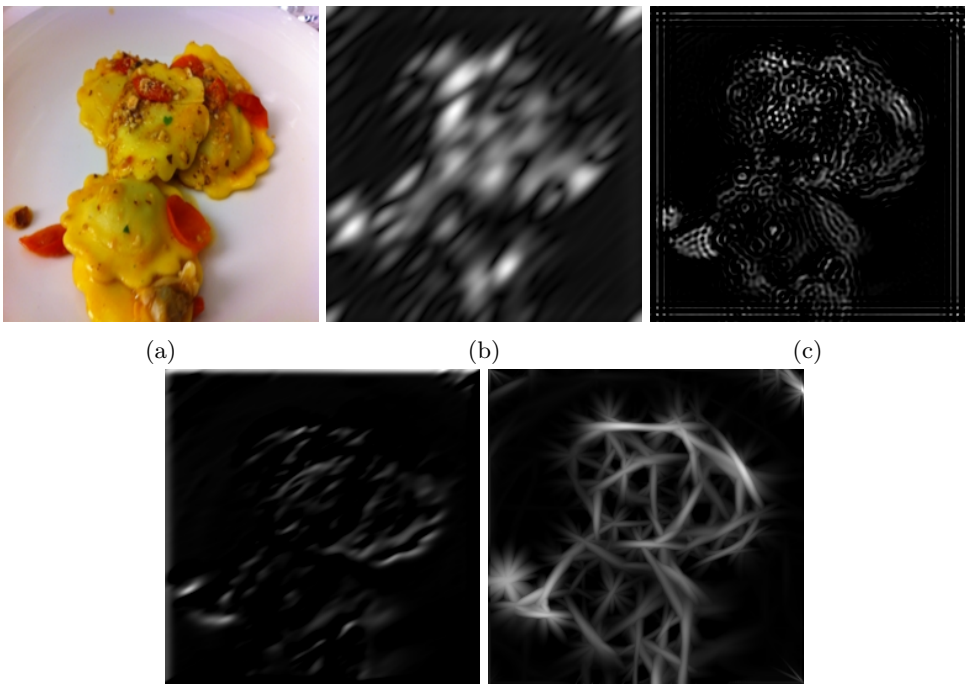
The **Leung-Malik (LM)** [103] filter bank is also used to get texture features. The LM filter bank consists of first and second derivatives of Gaussians at 6 orientations and 3 scales, 8 Laplacian of Gaussian (LoG) filters, and 4 Gaussians (see Figure 2.2(d)). After convolving the image with such filters the responses are collected in the feature vector $\boldsymbol{\Lambda}(\mathbf{I}) \in \mathbb{R}^{(MN)\times 48}$.

The **Maximum Response 8 (MR8) filter bank** [174] has been also considered. The MR8 bank inherits from the Root Filter Set (RFS) consisting of 38 filters similar to the LM ones (see Figure 2.2(e)). However, RFS filters are not rotation invariant and the MR8 ones were conceived to sidestep such a problem. This is achieved by taking only the maximum RFS filter responses across all orientations for the two anisotropic filters. In particular, measuring only the maximum response across orientations allows to reduce the number of responses from 38 (6 orientations at 3 scales for 2 oriented filters, plus 2 isotropic) to 8 (3 scales for 2 filters, plus 2 isotropic). Thus, the MR8 filter bank consists of 38 filters but only 8 filter responses which are finally collected in $\boldsymbol{\Delta}(\mathbf{I}) \in \mathbb{R}^{(MN)\times 8}$. An example of the responses of the different filter banks is shown in Figure 2.3.

### 2.2.2 Feature Encoding

Feature encoding techniques are commonly exploited to reduce the dimensionality of the feature vector used to represent an image. Such methods rely on the idea that an image can be described as a composition of "visual words". Such visual words form a codebook which is commonly learned in an unsupervised fashion during a training phase. More specifically, common encoding schemes define a "visual word" as the centroid of a set of clusters into which the feature space is split.

Let $\mathcal{X}(\mathbf{I}) = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be the set of $n$ $d$-dimensional column vectors representing the feature descriptors extracted from a single image $\mathbf{I}$. In our case, each feature



(a) (b) (c)

(d) (e)

Figure 2.3: Response images after convolutions with a single filter randomly selected from the four different filter banks shown in Figure 2.2. All filter responses are scaled for visualization. (a) Input image. (b) Response after convolution with a Gabor filter. (c) Response after convolution with a Schmid filter. (d) Response after convolution with a Leaung-Malik filter. (e) Response after convolution with a MR8 filter.

descriptor is given by the concatenation of filter responses at a single location, therefore $n = MN$ and $d \in \{9, G, 13, 48, 8\}$ depending on which filter bank is used. Let $\mathcal{X}^{\text{Tr}} = \{\mathcal{X}(\mathbf{I}_1), \cdots, \mathcal{X}(\mathbf{I}_{\text{Tr}})\}$ be the set of feature vectors extracted from all the Tr training images. Finally, let $d(\mathbf{x}_i, \mathbf{x}_j)$ be the dissimilarity between any two feature vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ belonging to $\mathcal{X}^{\text{Tr}}$. Such distance is used to cluster the feature space defined by $\mathcal{X}^{\text{Tr}}$.

### Bag-of-Words Encoding

The Bag-of-Words (BoW) encoding model idea inherits from document processing techniques where a document is seen as a set of words each of which has a difference frequency of appearance. In the BoW encoding for images, each image is represented by counting the frequency of a particular feature present in the learned codebook.

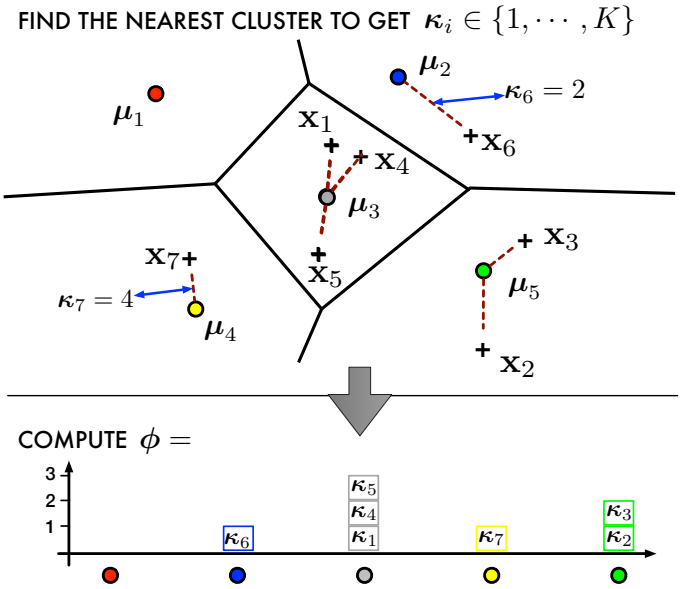



Figure 2.4: Bag-of-Words encoding of feature vectors $\mathbf{x}$. First row shows the nearest cluster out of the $K$ possible ones obtained by k-means. Second row describes the final encoded vector obtained by computing the histogram counting the frequency of each nearest cluster.

See Figure 2.4 for the BoW feature encoding scheme.

To compute the codebook of "visual words", k-means is applied on the set $\mathcal{X}^{\mathrm{Tr}}$. Let $\boldsymbol{\mu}_k^{bow}$, for $k = 1, \ldots, K$ be the so computed $K$ cluster means. Once the codebook is computed, the features extracted from a new image are quantized as follows. Each feature vector $\mathbf{x}_i \in \mathcal{X}$ is mapped to the corresponding "visual word", i.e., to the nearest cluster as

$$\boldsymbol{\kappa}_i = \underset{k^*=1,\cdots,K}{\arg\min}\; d\left(\mathbf{x}_i, \boldsymbol{\mu}_{k^*}^{bow}\right). \tag{2.1}$$

Hence, the $i$-th entry in $\boldsymbol{\kappa} \in \mathbb{N}^n$ denotes the index of the nearest cluster to the $i$-th feature vector.

Then, the BoW encoding is obtained by computing the $K$ dimensional histogram of such "visual words" as

$$\boldsymbol{\phi}^{bow}(\mathbf{I}) = \begin{bmatrix} \sum_{i=1}^{n} \mathbb{1}_{(\boldsymbol{\kappa}_i = 1)} \\ \vdots \\ \sum_{i=1}^{n} \mathbb{1}_{(\boldsymbol{\kappa}_i = K)} \end{bmatrix} \tag{2.2}$$

where $\mathbb{1}_{(\cdot)}$ is the indicator function.

### Fisher Vector Encoding

The Fisher Vector (FV) encoding [130] was introduced to provide a softer way to compute the cluster prototypes and a more robust feature encoding. This technique, first introduced by Jaakkola and Haussler [73], defines an aggregation mechanism based on the Fisher Kernel (FK) principle by combining the benefits of generative and discriminative approaches to pattern classification.

In the FV encoding scheme, clustering is performed through a Gaussian Mixture Model (GMM). Let $\boldsymbol{\mu}_k^{fv}, \boldsymbol{\Sigma}_k, \boldsymbol{\pi}_k^{fv}$, be the mean, covariance and the prior of each of the $k = 1, \cdots, K$ Gaussian Models, respectively. The clustering process is performed on the training data $\mathcal{X}^{\text{Tr}}$. The task is accomplished by model fitting based on the Expectation Maximization (EM) algorithm.

Once the GMM model is obtained, the posterior probability of each Gaussian Model is computed for every feature vector $\mathbf{x}_i \in \mathcal{X}$ as

$$\boldsymbol{\omega}_k(i) = \frac{\exp\left[-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_k^{fv})^T \ \boldsymbol{\Sigma}^{-1} \ (\mathbf{x}_i - \boldsymbol{\mu}_k^{fv})\right]}{\sum_{t=1}^{K} \exp\left[-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_t^{fv})^T \ \boldsymbol{\Sigma}^{-1} \ (\mathbf{x}_i - \boldsymbol{\mu}_t^{fv})\right]}. \tag{2.3}$$

Then, for each dimension $j = 1, \cdots, d$ and Model $k$, the $d$-dimensional mean and covariance deviation vectors are obtained as

$$\widetilde{\boldsymbol{\mu}}_k^{fv}(j) = \frac{1}{n\sqrt{\boldsymbol{\pi}_k^{fv}}} \sum_{i=1}^{n} \boldsymbol{\omega}_k(i) \frac{\mathbf{x}_i(j) - \boldsymbol{\mu}_k^{fv}(j)}{\boldsymbol{\sigma}_k^{fv}(j)} \tag{2.4}$$

$$\widetilde{\boldsymbol{\Sigma}}_k^{fv}(j) = \frac{1}{n\sqrt{2\boldsymbol{\pi}_k^{fv}}} \sum_{i=1}^{n} \boldsymbol{\omega}_k(i) \left[ \left( \frac{\mathbf{x}_i(j) - \boldsymbol{\mu}_k^{fv}(j)}{\boldsymbol{\sigma}_k^{fv}(j)} \right)^2 - 1 \right] \tag{2.5}$$

The FV encoded feature representation of a given image, denoted as $\boldsymbol{\phi}^{fv}(\mathbf{I}) \in \mathbb{R}^{2Kd}$, is obtained by stacking the column vectors $\widetilde{\boldsymbol{\mu}}^{fv}$ and $\widetilde{\boldsymbol{\Sigma}}^{fv}$ computed for each of the $k$ Gaussian mixtures

$$\boldsymbol{\phi}^{fv}(\mathbf{I}) = \begin{bmatrix} \widetilde{\boldsymbol{\mu}}_1^{fv} \\ \vdots \\ \widetilde{\boldsymbol{\mu}}_K^{fv} \\ \widetilde{\boldsymbol{\Sigma}}_1^{fv} \\ \vdots \\ \widetilde{\boldsymbol{\Sigma}}_K^{fv} \end{bmatrix} \tag{2.6}$$

Such an encoding is not limited to the number of occurrences of each visual word but it also includes additional information about the distribution of the descriptors. Recent works [130, 145] have shown that such an approach leads to better classification performances with respect to the standard BoW encoding methods.

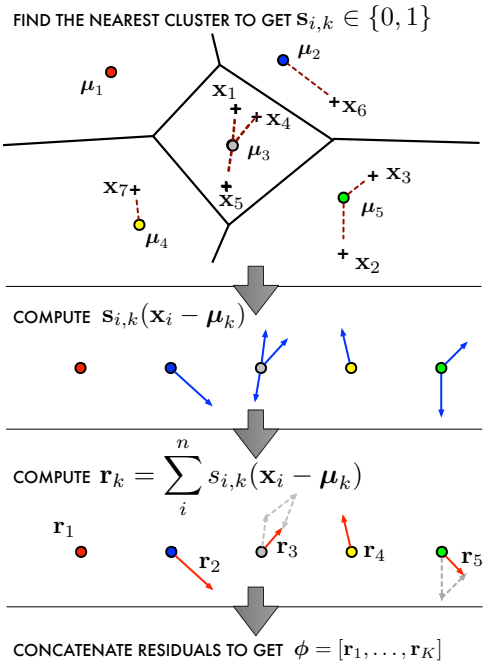



Figure 2.5: Vector of Locally Aggregated Descriptors (VLAD) encoding of feature vectors $\mathbf{x}$. Top row shows the hard assignments of feature vectors to the k-means quantized space. Then, second and third rows show the computation of the residuals. The last row describes the final encoded vector obtained by stacking the residuals.

### VLAD Encoding

The Vector of Locally Aggregated Descriptors (VLAD) image encoding was initially proposed by Jegou *et al.* [75] with the objective to provide an excellent search accuracy with a reasonable vector dimensionality. The VLAD feature encoding is shown in Figure 2.5 and briefly described in the following.

VLAD is a feature encoding and pooling method, similar to Fisher vectors. It encodes the feature vector set by first constructing a dictionary of such features. This is usually done by applying a clustering method such as a Gaussian Mixture Model (GMM) or k-means on a set of features extracted from training images. Let $\boldsymbol{\mu}_k^{vlad}$, for $k = 1, \ldots, K$ be the so computed $K$ cluster means. Let also $s_{i,k}$ be the strength of the association of a feature $\mathbf{x}_i \in \mathcal{X}$ to the $k$-th cluster. Such a strength can be computed either in a soft (e.g., obtained as the posterior probabilities of the GMM clusters) or hard (e.g., obtained by vector quantization with k-means) way, but it should satisfy

$$\sum_{k=1}^{K} s_{i,k} = 1 \quad \wedge \quad s_{i,k} \geq 0 \quad \forall k. \tag{2.7}$$

The cluster means $\boldsymbol{\mu}_k^{vlad}$ are of the same dimension as the features in $\mathcal{X}$. The VLAD encoding scheme exploits such a property to encode the $n$ features by considering their residuals with respect to the $K$ cluster means as

$$\mathbf{r}_k = \sum_{i=1}^{n} s_{i,k} \left( \mathbf{x}_i - \boldsymbol{\mu}_k^{vlad} \right). \tag{2.8}$$

Thus, for each cluster a vector of the same dimensionality of the original feature is obtained.

The encoded representation of the whole image is finally computed by stacking all the vectors of residuals as

$$\boldsymbol{\phi}^{vlad}(\mathbf{I}) = \begin{bmatrix} \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_K \end{bmatrix} \tag{2.9}$$

where $\boldsymbol{\phi}^{vlad}(\mathbf{I}) \in \mathbb{R}^{(Kd)}$.

### 2.2.3 Image Classification

Let $\mathbf{I}$ be a given image of a particular class. From such an image, all the aforementioned features are separately extracted and encoded. Once the encoding process is completed, the resulting vectors are concatenated to obtain the joint encoded representation $\boldsymbol{\Phi}(\mathbf{I}) = [\boldsymbol{\phi}(\boldsymbol{\Upsilon}(\mathbf{I}))\boldsymbol{\phi}(\boldsymbol{\Gamma}(\mathbf{I}))\boldsymbol{\phi}(\boldsymbol{\Psi}(\mathbf{I}))\boldsymbol{\phi}(\boldsymbol{\Lambda}(\mathbf{I}))\boldsymbol{\phi}(\boldsymbol{\Delta}(\mathbf{I}))] \in \mathcal{F}$. Then, the goal of classification is to learn a mapping from the joint encoded feature space $\mathcal{F}$, to the label space, $\mathcal{Y}$. Where each element $y \in \mathcal{Y}$ defines a particular class.

However, it might be that not every component in the joint encoded representation has the same discriminative power [111]. Motivated by this, and due to the multi-class classification nature of the image classification problem, we exploit a Random Forest classifier [17]. The RF classifier is able to automatically detect, and hence exploit, only the relevant features for the given task. It builds a large collection of de-correlated trees with the objective of reducing the variance of an estimated prediction function by pooling many noisy but approximately unbiased models. Trees are ideal candidates for bagging as they capture complex interaction structures in the data and have low bias. Also, trees are very noisy, hence they benefit greatly from the pooling procedure. As shown in [17], an average of $N$ i.i.d. random variables, each with variance $\sigma^2$, has variance $1/N\sigma^2$. If the variables are simply i.d. (identically distributed, but not necessarily independent) with positive pairwise correlation $\rho$, the variance of the average is $\rho\sigma^2 + \frac{1-\rho}{N}\sigma^2$. As $N$ increases, the second term disappears, but the first remains, and hence the size of the correlation of pairs of bagged trees limits the benefits of pooling. The idea in random forests is to improve the variance reduction of bagging by reducing the correlation between the trees, without increasing the variance too much. This is achieved in the tree-growing process through random selection of the input variables.

**Algorithm 1:** Random Forest for Classification of DFVs

**Input :** Training Encoded Features
**Output:** Trained ensemble of decision trees
**for** $n \leftarrow$ **to** $N$ **do**
  Draw a bootstrap sample $\mathbf{Z}^*$ of size $S$ from the training data;
  Grow a random-forest tree $T_n$ to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $s_{min}$ is reached:
  i. Select $m$ variables at random from the $p$ variables.
  ii. Pick the best variable/split-point among the $m$.
  iii. Split the node into two child nodes.
**end**
Output the ensemble of trees $\{T_n\}_{n=1}^{N}$;

To learn the parameters of the decision surfaces that separate the classes in $\mathcal{F}$ we trained an RF classifier as follows. Let $\mathcal{M} = \{\mathcal{T}_t\}_{t=1}^{T}$ be a forest of $T$ trees each one denoted as $\mathcal{T}_t$. Each tree is trained with a set of Tr data points $\mathcal{S} = \{(\mathbf{\Phi}(\mathbf{I}_i), y_i)\}_{i=1}^{\text{Tr}}$. Since bagging can also be applied, each tree can be trained with a different set of randomly selected data points $\mathcal{S}^* \subseteq \mathcal{S}$. The dimensionality of $\mathcal{S}^*$ is controlled through the bagging hyper-parameter $\eta$ as $\mathcal{S}^* = \eta|\mathcal{S}|$. Let also $\mathcal{S}_j \subseteq \mathcal{S}^*$ be the set of data points reaching node $j$. Each $j$-th node is associated with a binary split function $h(\mathbf{\Phi}(\mathbf{I}), \mathbf{\Theta}_j) \in \{0, 1\}$, i.e., the weak learner, which is characterized by its parameters $\mathbf{\Theta}_j = \{\boldsymbol{\kappa}, \boldsymbol{\psi}, \boldsymbol{\tau}\}$ where $\boldsymbol{\psi}$ defines the geometric primitive used to separate the data (e.g. an axis-aligned hyperplane). The parameter vector $\boldsymbol{\tau}$ captures thresholds for the inequalities used in the binary test. The filter function $\boldsymbol{\kappa}$ randomly selects some features of choice out of the entire vector $\mathbf{\Phi}(\mathbf{I}_i)$. In the current framework, we select

$$h\left(\mathbf{\Phi}(\mathbf{I})_i, \mathbf{\Theta}_j\right) = \mathbb{1}_{(\langle \boldsymbol{\kappa}(\mathbf{\Phi}(\mathbf{I})_i), \boldsymbol{\psi} \rangle > \tau)}, \tag{2.10}$$

where $\tau = 0$ and $\boldsymbol{\psi}$ denotes a hyperplane.

All the aforementioned parameters are optimized at each split node as

$$\mathbf{\Theta}_j^* = \underset{\mathbf{\Theta}_j^* \in \mathcal{P}_j}{\arg\max}\, J(\mathcal{S}_j, \mathcal{S}_L, \mathcal{S}_R, \mathbf{\Theta}_j), \tag{2.11}$$

where

$$J(\mathcal{S}, \mathbf{\Theta}) = H(\mathcal{S}) - \sum_{i=\{L,R\}} \frac{|\mathcal{S}_i|}{|\mathcal{S}|} H(\mathcal{S}_i), \tag{2.12}$$

with Shannon entropy $H(\cdot)$ and $\mathcal{S}_L$ and $\mathcal{S}_R$ denoting the subsets of training points going to the left and to the right children, respectively. $\mathcal{P}_j \subset \mathcal{P}$ is the random subset of available parameters $\mathbf{\Theta}$. The dimensionality of such a random subset is controlled by $\rho = |\mathcal{P}_j|$. The training continues to split the samples until the maximum depth $D$ is reached, a node contains a single sample or samples of a single class are left.

Once the training of the whole forest is completed, a set of leaves $\mathcal{L}_t$ is associated to each tree $\mathcal{T}_t$. To each leaf $\ell_t \in \mathcal{L}_t$ is associated a probabilistic model $p_t(y|\mathbf{\Phi}(\mathbf{I}_i))$ with $y \in \mathcal{Y}$ indexing the class. Therefore, to classify a new image $\hat{\mathbf{I}}$, first its feature-encoded representation is computed. Then, for each tree, $\mathbf{\Phi}(\hat{\mathbf{I}})$ follows the path from the root node down to a leaf one. Once a leaf is reached for each tree, the RF classifier assigns $\mathbf{\Phi}(\hat{\mathbf{I}})$ the class probability

$$p(y|\mathbf{\Phi}(\hat{\mathbf{I}})) = \frac{1}{T}\sum_{t=1}^{T} p_t(y|\mathbf{\Phi}(\hat{\mathbf{I}})). \tag{2.13}$$

The final class label is computed as $\hat{y} = \arg\max_y p(y|\mathbf{\Phi}(\hat{\mathbf{I}}))$.

## 2.3 Experimental Results

To validate the proposed approach, results on two benchmark food recognition datasets have been computed. As commonly performed in the evaluation of food recognition approaches [22, 32, 33], the achieved performances are provided in terms of recognition accuracy. The performance achieved by the existing methods have been taken from the corresponding works or have been provided by the authors.

### 2.3.1 Experimental Settings

In the current framework, we have used a Gabor filter bank with 8 orientations and 5 sizes, therefore $G = 40$. When not explicitly specified, the feature encoding has been obtained by means of the FV approach with $K = 300$ clusters. Similarly, by default, the RF classifier has been trained with $T = 2000$ trees having maximum depth $D = 100$. Each tree has been trained by setting $\eta = 0.6$. At each node the function $\kappa$ randomly selects $\sqrt{d}$ features, where $d$ is the dimensionality of the input feature vector.

Finally, images have been rescaled to $128 \times 128$ to remove the effect of different image sizes on the classifier performances.

### 2.3.2 Datasets

#### UNICT-FD889 Dataset

The UNICT-FD889 Dataset[1] has been recently introduced by Farinella *et al.* [32]. The UNICT-FD889 dataset has the largest number of different classes to recognize. It comes with 3583 images related to 889 distinct dishes of food of different nationalities (e.g., Italian, English, Thai, Indian, Japanese, etc.) which have been collected in a real and uncontrolled scenario (e.g., different backgrounds and illumination conditions)

[1] Available at http://iplab.dmi.unict.it/UNICT-FD889

Figure 2.6: 15 randomly selected samples from the UNICT-FD889 dataset. Columns correspond to a different type of food (i.e., to a different class). Rows show the appearance variations between samples belonging to the same class.

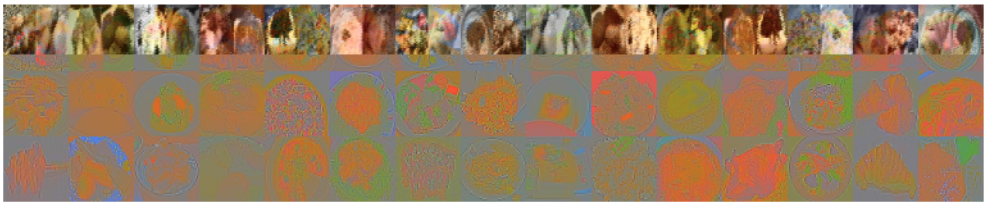


Figure 2.7: 15 randomly selected samples from the UECFood100 dataset. Columns correspond to a different type of food (i.e., to a different class). Rows show the appearance variations between samples belonging to the same class.

by means of smartphones. Hence, the UNICT-FD889 dataset is a collection of food images acquired by users in real cases of meals. Each food belonging to a particular class has been acquired multiple times (four on average) to ensure geometric and photometric variabilities (see Figure 2.6 for a few examples).

To provide a fair comparison with existing methods, the following results have been computed by averaging the performance on the same three splits as advised by Farinella *et al.* [32].

### UECFood100 Dataset

The UECFood100 Dataset [2] is one of the largest food recognition datasets [117]. This dataset contains approximately 100 images for each of the 100 different food categories. Thus it contains approximately 14000 real-world food images. The UECFood100 dataset was built to implement a practical food recognition system [3] which was intended to be used in Japan. Because of this, it was collected in such a way that multiple food items were present in a single image, thus with the objective to perform both the detection and the recognition tasks.

Since the proposed system is designed to focus only on the recognition task, the given ground truth bounding boxes have been used to obtain a dataset of images containing single food items only (see Figure 2.7). Despite this, the same protocol

[2] Available at http://foodcam.mobi/dataset100.html
[3] http://foodcam.mobi/

Figure 2.8: 15 randomly selected samples from the Food-101 dataset. Columns correspond to a different type of food (i.e., to a different class). Rows show the appearance variations between samples belonging to the same class.

proposed by Matsuda *et al.* [117] has been followed to fairly compare the obtained performance with existing methods.

#### Food-101 Dataset

The Food-101 Dataset[4] is the largest food recognition dataset [16]. It has been collected by downloading images from foodspotting.com. The top 101 most popular and consistently named dishes were selected. Then, for each category 750 training and 250 test images were collected and manually cleaned. On purpose, the intense colors and sometimes wrong labels included in the training images were not cleaned. As a result the dataset contains 101'000 real-world food images (see Figure 2.8).

### 2.3.3 Performance Analysis

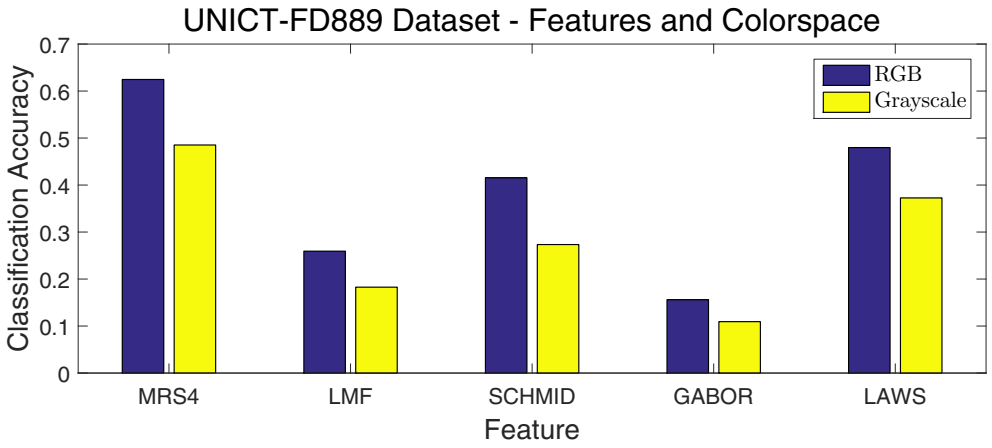
To evaluate the proposed approach, we have analyzed the following aspects: (i) performance of single filter banks using grayscale or RGB color images. In the latter case, each image plane is separately processed; (ii) performance of different encodings, also as a function of the codebook size; (iii) influence of the RF hyper-parameters on the classification accuracy.

#### Filter Banks

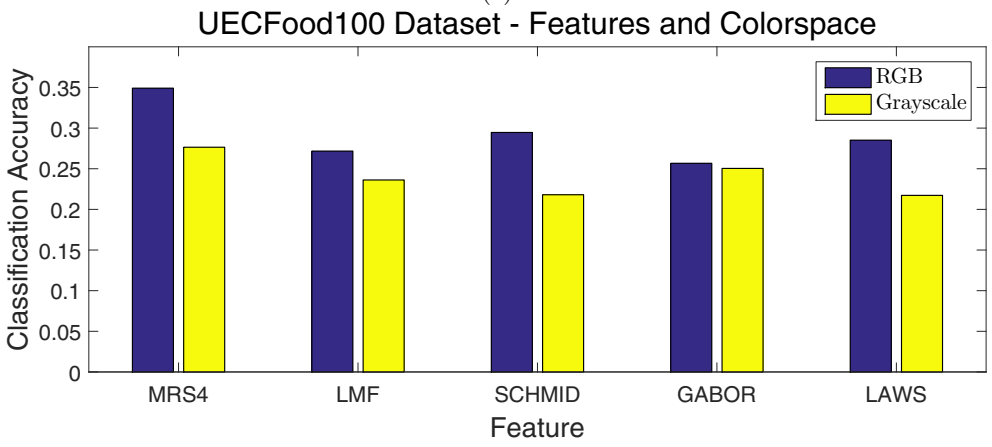
To analyze the accuracy performance of each filter bank we have computed the results shown in Figure 2.9.

**UNICT-FD889:** Results in Figure 2.9(a) show that, for every considered filter bank, the best performance are achieved when the full color information provided by three channels is exploited. In particular, when Schmid filters are used, the performance drops from 41.54% to 27.34% if only grayscale information is used. The best performance are obtained by filtering the RGB color images with the MR8 filter bank.

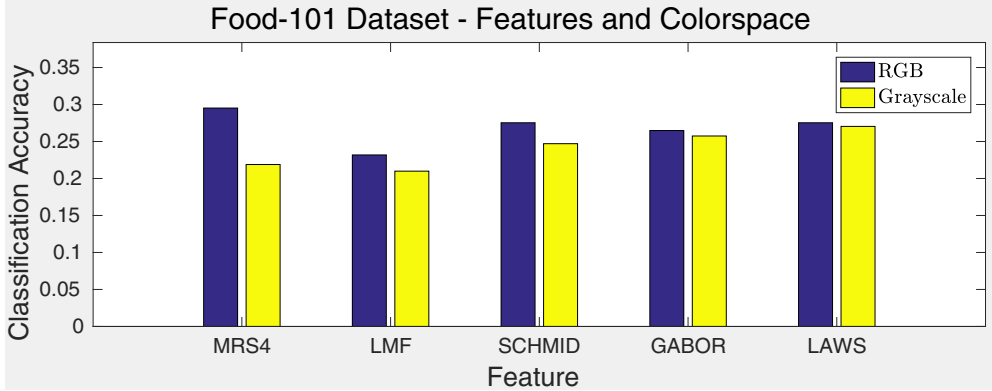
[4]Available at http://www.vision.ee.ethz.ch/datasets/food-101




(a)




(b)




(c)

Figure 2.9: Accuracy performance on the (a) UNICT-FD889, (b) UECFood100, and (c) Food101 datasets achieved by filtering RGB and grayscale images with each single filter bank.

In such a case the accuracy is 62.46%. More interestingly, results show that while being simpler compared to other filters, Laws filters yield the second best accuracy (i.e., 47.97%).

**UECFood100:** Results in Figure 2.9(b) reflect those obtained considering the UNICT-FD889 dataset. Indeed, overall, the best performance are achieved when color information is considered. However, for the considered dataset the accuracy difference between grayscale and RGB filtered images is never more than 9% (Schmid filters). The best performance are obtained by using the MR8 filter bank on RGB images. In such a case the accuracy is 34.71%. More interestingly, results show that there is less than 1% accuracy difference when Gabor filters are applied on RGB images instead of grayscale ones.

**Food101:** Food101 is the most complex of the three datasets, as it has a large intra-class variance. This affects the results shown in Figure 2.9(b), where the best performing filter bank (MR8) achieved just a 29% of classification accuracy. The other banks, except LMF, reached similar performances. It is worth noting that, with the exception of MR8, in the other cases working on RGB or grayscale data does not drastically affect the performance. This is probably due to the intra-class shape variance, leading to spatial features be more relevant than color information.

### Feature Encoding

In the preceding section we have reported on the performance of each single filter bank obtained by using the FV encoding scheme. To evaluate the performance of different encodings and the influence of the codebook dimensionality, we have conducted the following experiments. Each filter bank response is encoded using the same method and the same number of "visual words" $K$. Then, the 5 obtained encoded vectors obtained for each image are stacked and input to the RF classifier.

In addition, to see if the RF classifier is able to select the optimal features as well as the optimal encoding, we have run one additional experiment. Specifically, the three encoding methods are separately applied to each filter bank response. The value of $K$ has been kept same for each of them. The so obtained 15 encoded vectors are finally stacked and given to the RF classifier. In the following, results obtained with such a procedure are referred to as *All*.

Results in Figure 2.10 show that for VLAD, FV and *All* optimal accuracy performance are achieved when the number of clusters $K$ is of either 20 or 50. In particular, 20 clusters are considered in the FV encoding scheme the classification accuracy is about 63.29% which is very close to the performance of the single MR8 filter bank encoded with the same scheme but with 300 clusters (see Figure 2.9(a)). For all the encodings, the performance decrease when the number of clusters increase. In particular, it is worth noticing that when the *All* solution is considered, performance are even lower than a single adopted encoding scheme. Such a behavior is mainly due to the exploding dimensionality of the encoded feature vectors which yields to a very sparse space, hence introduces the curse of dimensionality issue. Such a problem

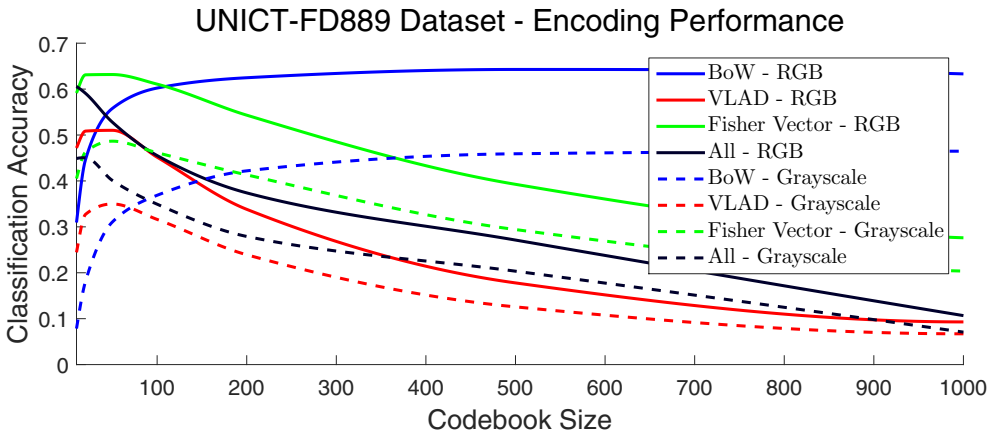



Figure 2.10: Accuracy performance on the UNICT-FD889 dataset achieved by using different encoding schemes and color spaces.

could be addressed by growing the number of trees in the RF model.

On the other hand, performance obtained using the BoW encoding increases if a large number of visual words is used. The best accuracy under such scheme is obtained when the number of clusters is of 500. When more clusters are considered performance decrease. Differently from VLAD and FV, BoW encoding does not suffer from the curse of dimensionality problem since the number of dimensions in the encoded feature space is determined by the codebook size (see Section 2.2.2).

Finally, as previously shown in the filter banks analysis, better performance are achieved when RGB color information is considered.

### Random Forest Classifier

The RF classifier model is controlled by different hyper-parameters: the number of trees $T$, the maximum depth $D$, the bagging size defined by $\eta$ and the number of features that are selected through the filter function $\kappa$. To evaluate the influence of such hyper-parameters on the performance, we have proceeded by fixing the values of the hyper-parameters as defined in Section 2.3.1, then we have varied the value of a single hyper-parameter. The results of such analysis, shown in Figure 2.11 and Figure 2.12, have been computed considering all the filter banks encoded using 300 clusters and the FV method. Color information has been used.

**UNICT-FD889:** Results in Figure 2.11(a) have been computed by varying the number of trees such that $T \in [1, 5000]$. Under such scenario, the classification accuracy drastically increases when $T$ grows from 1 to 1000. For larger values the performance improves but with less gain. The best performance is obtained when $T = 5000$ trees have been used. Such results demonstrate the benefits of the RF

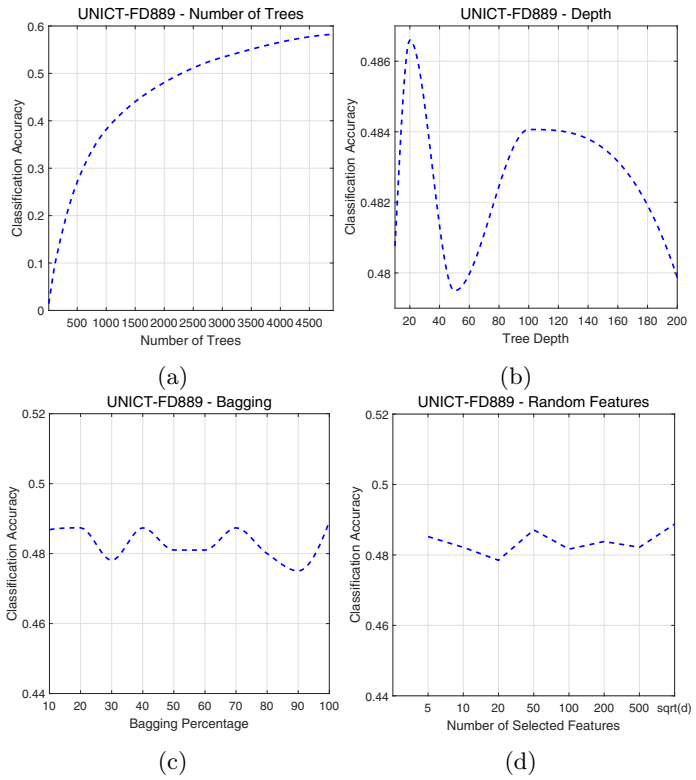



Figure 2.11: Performances of the random forest classifier on the UNICT-FD889 dataset varying the different hyper-parameters. Results have been computed by first fixing the hyper-parameters as described in Section 2.3.1, then we varied the: (a) Number of trees in the forest; (b) Maximum depth of each single tree; (c) Percentage of the features selected at each node; (d) Number of randomly selected features at each node.

classifier, and in general of aggregation methods, which are able to strongly improve the accuracy produced by a single weak classifier.

The results on the analysis of the depth hyper-parameter are shown in Figure 2.11(b). Results show that by varying $D$ in the range $[10, 200]$ two main classification accuracy peaks can be found: the first is at $D = 20$, the second at $D = 100$. For smaller and larger values performance decreases due to underfitting and overfitting problems respectively. Indeed, with a very shallow tree it is not possible to have enough decision boundaries to discriminate all the samples well. Similarly, if the tree is too deep, the number of decision boundaries is too high, thus causing the RF model to not generalize well on new samples. The performance decrease for values in between the two peaks is caused by the two random components of the RF, i.e., bagging and

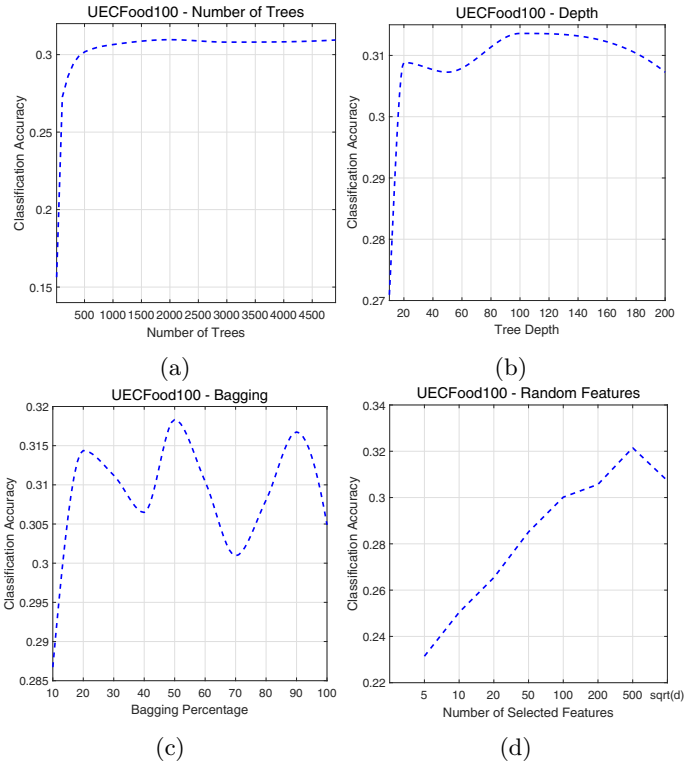



Figure 2.12: Performances of the random forest classifier on the UECFood100 dataset varying the different hyper-parameters. Results have been computed by first fixing the hyper-parameters as described in Section 2.3.1, then we varied the: (a) Number of trees in the forest; (b) Maximum depth of each single tree; (c) Percentage of the features selected at each node; (d) Number of randomly selected features at each node.

the random feature selection at each node. Despite this, it should be noticed that for all the considered cases the gap between the best (48.67%) and the worst performance (47.94%) is less than 0.8%, thus showing the robustness of the model to such hyper-parameter.

Finally, in Figure 2.11(c) and Figure 2.11(d) the RF model performance are computed by varying the bagging percentage and the number of selected random features at each node, respectively.

The results in Figure 2.11(c) show that there is consistency in the results when the bagging hyper-parameter $\eta$ is changed from 10% to 100% (notice that in such a last case no bagging is used since all the available data is used by each tree). Indeed, no matter what bagging percentage is considered, the classification accuracy is never

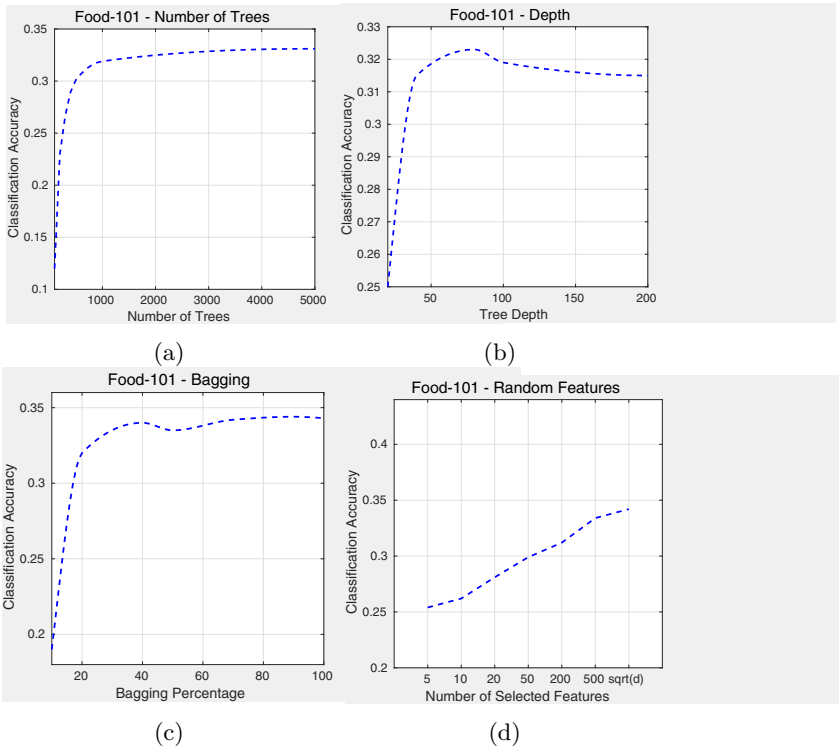



Figure 2.13: Performances of the random forest classifier on the Food-101 dataset varying the different hyper-parameters. Results have been computed by first fixing the hyper-parameters as described in Section 2.3.1, then we varied the: (a) Number of trees in the forest; (b) Maximum depth of each single tree; (c) Percentage of the features selected at each node; (d) Number of randomly selected features at each node.

less than 47.71% ($\eta = 0.9$) nor higher than 48.86% ($\eta = 1$).

Similarly, results in Figure 2.11(d) show that the RF model is insensitive to the number of randomly selected features at each node. Varying such a hyper-parameter the classification accuracy gap between the best and the worst performance is less than 0.9%. The optimal performance is obtained when the number of selected features is computed as $\sqrt{d}$.

**UECFood100:** Results in Figure 2.12(a) have been computed by varying the number of trees such that $T \in [1, 5000]$. Under such scenario, the classification accuracy drastically increases when $T$ grows from 1 to 500. For larger values the performance remains stable with a peak at $T = 2000$.

In Figure 2.12(b), results of the proposed approach are given as a function of the depth RF hyper-parameter. Differently from the results shown in Figure 2.11(b), the

accuracy performance obtained using a very shallow tree (i.e., $D = 10$) is much worse than the one achieved using a deeper forest. This is due to the complexity of the dataset, which requires more separating hyperplanes to well discriminate between the 100 classes. The optimal results are achieved when $D = 100$.

Results in Figure 2.12(c) show the performance of the RF model varying the bagging percentage. The depicted results demonstrate that by considering an extreme bagging in which each tree sees only 10% of the total training samples the worst performance are achieved. However, it should be noticed that in such case the accuracy performance is only about 3% less than the optimal one (obtained by setting $\eta = 0.5$).

Finally, results in Figure 2.12(d) show that the RF model trained on the UECFood100 dataset is more sensitive to the selected number of randomly selected features at each node. Indeed, compared to the case when only 5 features are selected at each node, almost a 10% improvement is obtained by varying such a hyper-parameter to select 500 features.

To summarize, the RF hyper-parameter analysis has shown that the number of trees and the number of randomly selected features at each node in the forest should be carefully selected to obtain the best possible performance. On the other hand, the model performances do not heavily depend on the tree depth and the bagging percentage.

**Food101:** The performances on Food101 are similar to the ones achieved on UECFood100 and confirm the conclusions described above. Figure 2.13(a) confirms the initial performance boost in the range $T \in [1, 500]$, even though there is a small but constant improvement when augmenting the number of trees up to 5000.

Figure 2.13(b) again shows very poor results when the tree depth is too low ($D < 30$), due to the complexity of the dataset, and a slight performance decrease due to overfitting when $D >= 100$. The best results have been obtained with $D = 80$.

Regarding the bagging percentage, in Figure 2.13(c) we see again that this hyper-parameter drastically influence the results only when $\eta < 0.1$, while giving substantially stable results for any value higher than that.

The last result analyzes the system performances with respect to the number of randomly selected features at each node (Figure 2.13(d)). As already seen in UECFood100 dataset, there is an approximately linear increase in the classification accuracy while increasing the number of selected features from 5 to $\sqrt{d}$, where the system reaches its best result.

To summarize, the RF hyper-parameter analysis has shown that the number of trees and the number of randomly selected features at each node in the forest should be carefully selected to obtain the best possible performance. On the other hand, the model performances do not heavily depend on the tree depth and the bagging percentage.

### Discussion

As a result of the conducted performance analysis, we can draw the following conclusions:

1. While more computationally demanding color information should be retained to obtain better recognition performance.
2. The number of clusters determining the codebook size for encoding should be carefully selected on the basis of the chosen encoding scheme.
3. Exploiting more encoding schemes simultaneously may yield to performance degradation due to the very high-dimensionality of the obtained encoded feature vector.
4. For the specific tasks, the selected RF model has demonstrated to be sensitive to two hyper-parameters, namely the number of trees in the forest and the number of randomly selected features at each node. This can be exploited to simplify the cross-validation procedure usually required to select the optimal hyper-parameters values.

To qualitatively evaluate the performance of the proposed approach we have computed the results in Figure 2.14 and Figure 2.15. The performance achieved by the proposed method are shown for 18 query images from the UNICT-FD889 and the UECFood100 datasets, respectively (see caption for additional details). The depicted results demonstrate that, even only texture information is exploited, the proposed approach is able to well capture the global appearance of the images and it also has the capacity to reliably find the true match under challenging conditions (see the 3 cases in the first row). When the query image is not correctly classified, or the considered cases are very challenging, the resulting scores are very close to each other, thus meaning there is uncertainty in the given answer.

In the following section, the given results have been computed following the aforementioned conclusions. Thus, RGB color information, FV encoding with $K = 20$ clusters and $T = 5000$ trees have been considered. The other RF model hyper-parameters have been kept as defined in Section 2.3.1.

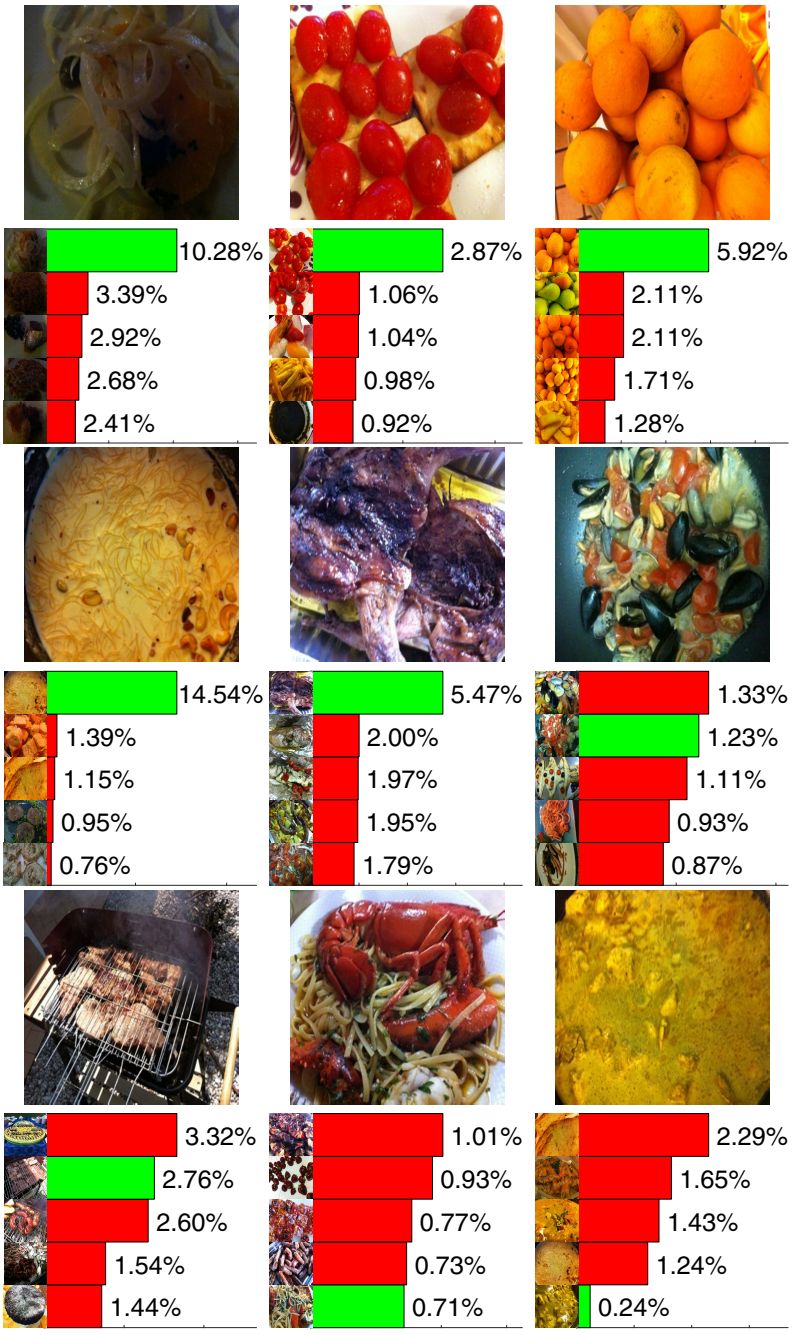



Figure 2.14: Performance achieved by the proposed method on the UNICT-FD889 dataset are shown for 9 query images (organized in two rows). At the bottom of each of those, the bar histograms show the score (in percentage) of the proposed approach for the true match (in green) and for the remaining top 4 matches (in red). On the y-axis of each bar histogram a randomly selected training image corresponding to the food class is depicted. *(Best viewed in color)*

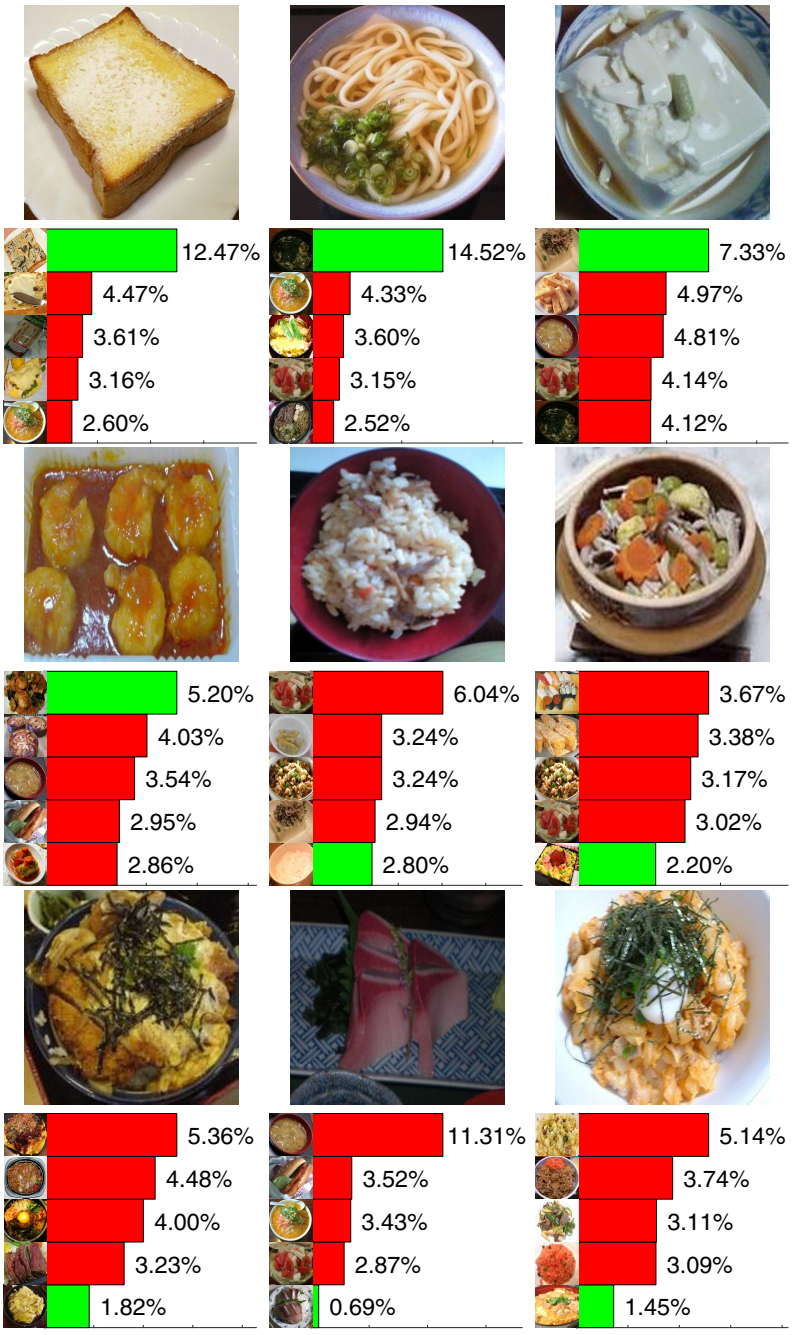



Figure 2.15: Performance achieved by the proposed method on the UECFood100 dataset are shown for 9 query images (organized in two rows). At the bottom of each of those, the bar histograms show the score (in percentage) of the proposed approach for the true match (in green) and for the remaining top 4 matches (in red). On the y-axis of each bar histogram a randomly selected training image corresponding to the food class is depicted. *(Best viewed in color)*

### 2.3.4 State-of-the-art Comparisons

Comparisons with state-of-the-art methods are given in Figure 2.16. Performances achieved by our method are labeled as TFE-RF.

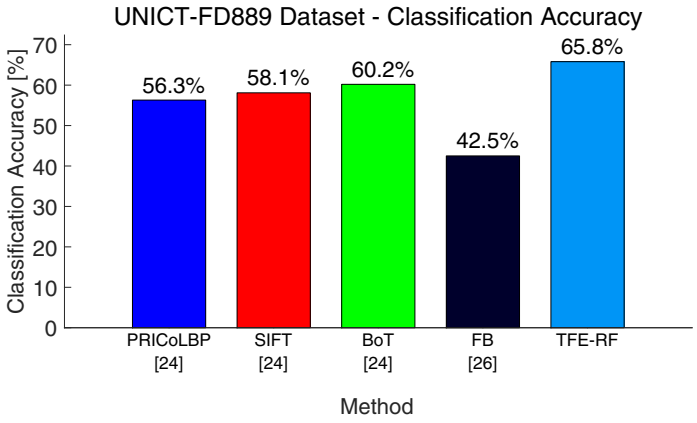
**UNICT-FD889:** In Figure 2.16(a), the obtained performance are compared to 4 state-of-the-art methods, namely PRICoLBP [32], SIFT [32], BoT [32] and FB [114]. Results demonstrate that the proposed approach outperforms existing ones by improving the previous best accuracy by more than 5%. This is an interesting result since other approaches (i) require complex procedure for feature extractions (e.g., PRICoLBP [133]); (ii) use the original input images of size $320 \times 240$, which carry more information than the resized ones.

**UECFood100:** A comparison with state-of-the-art approaches is given in Figure 2.16(b). The obtained performance are compared to the ones achieved by 6 state-of-the-art methods [117]. Methods like Circle, JSEG, DCR, DPM, and Whole use a detector to identify the location of the food, while GTBB [117] uses the same ground truth as TFE-RF. Results demonstrate that TFE-RF outperforms detector-based approaches but has lower performance than GTBB. This is mainly due to (i) the discriminative power of the additional color and shape features which these methods exploits [117] and (ii) the relevant information which full-size images has with respect to the resized ones which our approach considers.

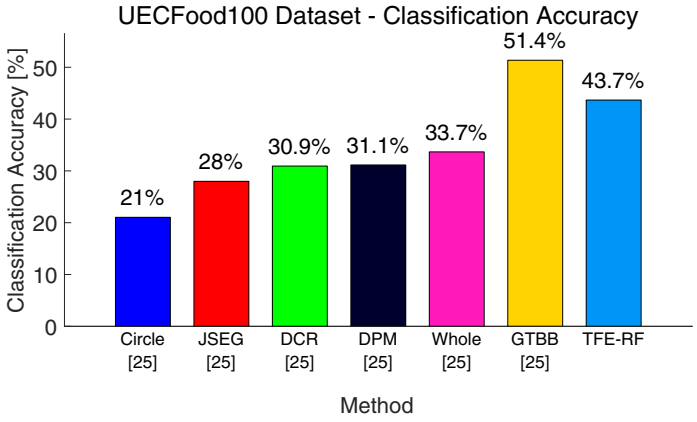
**Food101:** Figure 2.16(c) shows a comparison of the proposed method with state-of-the-art approaches on the Food101 dataset. The competitors performances have been taken from [16], and include among others approaches based on HoG, BoW, SURF, AlexNet, Random Forests, etc. (see [16] for exact references for each method). As it can be seen, in this challenging dataset the proposed method reaches an accuracy of 38.1%, comparable to the other approaches based on random forests, but outperformed by the results obtained by deep learning strategies, as in the case of AlexNet, based on a Convolutional Neural Network approach.

## 2.4 Conclusion

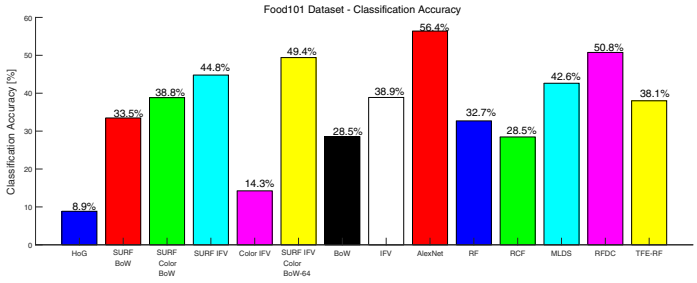
In this chapter an analysis of existing filter banks and feature encoding schemes for image classification has been provided. The work has been motivated by the lack of studies showing which biologically-inspired filters for texture features are likely to be the most useful for the task. In particular, we have considered five of the most widely used filter banks in computer vision, namely Laws, Gabor, Schmid, Leung-Malik and MR8. To reduce the high dimensional representations produced by filter banks we have used (and analyzed the performance of) the three main encoding schemes in literature: BoW, FV and VLAD. Then, the image classification task is accomplished by exploiting such encoded representations within an RF classifier. Results on two food classification benchmark datasets have been provided. Specifically, an in-depth analysis of the performance of each filter bank and encoding scheme has been provided.




(a)




(b)




(c)

Figure 2.16: Performance of the proposed TFE-RF approach are compared to state-of-the-art ones. Results are shown for the (a) UNICT-FD889, (b) UECFood100, and (c) Food101 datasets.

# 3

# Learning to Rank with a Committee of Shallow Neural Networks

*In this chapter, we introduce an image recognition system based on a committee of classifiers. Each committee member, i.e., an Extreme Learning Machine, is trained to specialize on a single feature type. Then, a supervisor, i.e., a Structural Support Vector Machine, is exploited to produce the final ranking of possible matches by filtering out the irrelevant features and thus merging only the relevant ones. Thus, the supervisor automatically selects the optimal features for image recognition out of the existing plethora of available ones (e.g., color, texture, etc.). Experimental results show that the proposed system outperforms state-of-the-art works on four publicly available benchmark datasets.*

## 3.1 Introduction

In the previous chapter we have provided an in-depth analysis of the image classification performance achieved by exploiting hand-crafted visual features that aim to mimic the feature extraction process conducted by the human brain. Such representations were considered as the input to a random forest of decision trees classifier that performs an efficient feature selection through the hierarchy. This solution followed the process conducted by most of the existing methods which address the image classification challenges by designing ad-hoc image representations based on some *a priori* knowledge of the problem (e.g., [33, 116, 182]). Despite their successful applications, the *a priori* knowledge might not be sufficient enough to correctly handle all the problem challenges. We hypothesize that a possible solution to sidestep the aforementioned problems might be a system that uses as many different features as

possible but exploits only a subset of those to perform the image classification task. Following this idea, an image classification system based on a supervised learning committee is introduced.

As demonstrated in [168, 167, 151], learning with a committee has two main benefits: (i) a committee might exhibit performance unobtainable by an individual committee member on its own. This is due to the fact that individual errors made by the committee members cancel out to some degree when their predictions are combined; (ii) a committee of learning machines has modularity properties. Since different members can focus on a particular region in the input space, the mapping from input to target is not approximated by one estimator but by several estimators. Despite these benefits, since many different possible visual features can be used to address the task, they cannot be just integrated in a single feature vector of very high dimensionality. Indeed, this might yield to intractable computational loads as well as to the curse of dimensionality problem. To address the aforementioned issues, the Supervised Extreme Learning Committee (SELC) approach is introduced.

SELC relies on a committee of Extreme Learning Machines (ELM) [71], where each ELM is trained with a specific feature type only. In this way, each member specializes on classifying an image only by using a certain feature type. This has the advantage of both reducing computational loads as well as to keep the committee learning benefits. Among all the possible neural-based learning systems [112], Extreme Learning Machines have been chosen for their excellent performances in terms of computational burden while maintaining a classification accuracy comparable to similar learning tools.

Committee-based approaches require the selection of a supervisor to fuse the discordant members' classifications. The typical output of the supervisor is a class. However, when classification results must be presented to users, a rank could be more appropriate. While ranking information can be obtained from single committee members, none of the existing works have adopted a supervisor considering it. Motivated by this, we introduce a Structural Support Vector Machine [170] as supervisor. It automatically selects the ranking produced by the members and combines them to obtain optimal classification performance as well as an optimal ranking.

The rest of the chapter is organized as follows: in Section 3.2 we describe the proposed approach, explaining how ELMs can be applied to image classification and how the committee outputs can be merged into a final rank by means of a Structured Support Vector Machine. In Section 3.3 we give some comparative experimental results, showing how our system performs with respect to state-of-the-art methods. Finally, conclusions are drawn in Section 3.4.

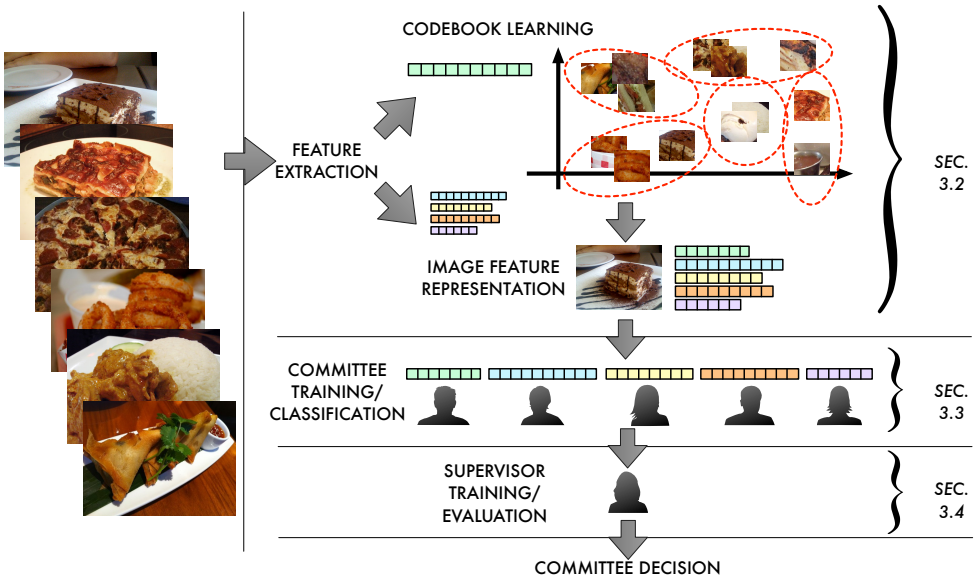



Figure 3.1: Proposed system architecture based on three main stages: (i) image feature representation; (ii) committee training and (iii) supervisor training. *(Best viewed in color)*

## 3.2 The Approach

### 3.2.1 System Overview

As shown in Figure 3.1, the proposed image recognition system consists of three main phases: (i) image feature representation, (ii) committee training/classification, and (iii) supervisor training/decision.

Since the proposed approach introduces a committee-based learning mechanism, a training phase is required. During such a phase, each training image is given to the feature extraction module that computes discriminative visual features capturing color, shape and texture information. To reduce the high dimensionality of a subset of such features, a codebook learning submodule that provides nonlinear feature encoding is exploited. The obtained encodings and the remaining features are finally considered as the image feature representation (details in Section 3.2.2). Then, the whole set of such representations is given to the committee training module (Section 3.2.3) where each committee member specializes in classifying the image by exploiting a single type of feature only (i.e., either color, shape or texture). Once the committee members are trained, their answers are evaluated by a committee supervisor whose aim is to learn how to properly combine them such that the optimal ranking is obtained (Section 3.2.4).

During the recognition phase, given a test image to classify, the same features are extracted. The nonlinear encoding procedure is applied to a subset of those by using the learned codebook. Then, the obtained representation is provided to the committee members for classification. Each member produces a classification considering only the type of feature it has been trained with. Finally, the members answers are given to the committee supervisor which combines them and produces the final decision (ranking).

### 3.2.2 Image Feature Representation

Literature in image recognition [33, 116, 182, 82] usually represents an object as a combination of different features (e.g., color, shape, spatial relationships, etc.). Recent works [32, 33] have also shown the benefits of feature encodings [83, 175, 155] for the same task.

Despite this, as also demonstrated in the previous chapter, it is difficult to identify which are the best features for image recognition. The SELC approach has been designed to tackle such an interesting problem. It aims to autonomously select only the relevant features out of a large pool of given ones. Thus, to obtain the image feature representation, a large set of features has been considered. This includes (i) color, (ii) shape, (iii) texture, (iv) local and (v) data-driven features.

**Color:** Due to their rotation and location invariant properties, color histogram features are the most widely used features to capture the global appearance of an image. However, as shown in [41, 172], histograms are discriminative feature representations of a datum only if the input image is projected in an appropriate color space. By following the advices in [41, 172], in the current framework the HSV, CIELab, RGB, normalized RGB and Opponent color spaces have been considered. Thus, for a given image $\mathbf{I}$, a histogram is extracted from each of such color space components. Histograms belonging to the same color space are finally concatenated.

**Shape:** To capture the shape of a given image the Pyramid Histogram of Oriented Gradients (PHOG) [15] and the GIST features [127] are used. The PHOG feature captures the local shape and the spatial layout of the shape in a given image [15] by exploiting the pyramidal framework proposed in [97]. The GIST feature models the shape of an image by computing the dominant spatial structure of a very low dimensional representation of the image itself (i.e., the Spatial Envelope).

**Texture:** As previously discussed (see chapter 2), texture features are of fundamental importance for image recognition. Indeed, looking at general images, its reasonable to claim that the image recognition problem is closely related to that of texture discrimination.

To capture the texture information, Local Binary Pattern (LBP) [126], Local Phase Quantization (LPQ) [136], Local Configuration Pattern (LCP) [45], Pairwise Rotation Invariant LBP (PRICoLBP) [133], Binary Gabor Patterns [188] and textons features [83, 175, 155] are used. These have been selected on the basis of the fact that

each of them captures different texture aspects which can be worth to inspect.

In particular, to extract textons features, the MRS4 filter bank [175] has been considered for textons encoding. To achieve invariance to affine illumination transformations, the z-scores of the image intensity are computed before convolution with the $\ell_1$ normalized filters. The obtained responses at each pixel location are then contrast normalized as in [33].

Differently from previous works in the field using standard textons encodings schemes like [32, 33], in this work a more robust feature encoding based on the Improved Fisher Vector (IFV) technique [73, 130, 145] is used. The IFV method suppresses the lossy process [14] of common Bag-of-Words (BoW) hard quantization methods which yield to performance degradation. In a nutshell, given a set of training images, after convolution with the MRS4 filters, the obtained filter responses are clustered by means of a Gaussian Mixture Model. Then, for each image, every feature descriptor (i.e., the filter response at a location) is assigned to a particular mode in the mixture with a strength given by the posterior probability. In addition, for each mode the mean and covariance deviation vectors are considered. The result of such a process, computed for every feature response, yields to the encoded feature representation.

**Local features:** Local feature are particular image regions which are different from their surroundings. Such features lead to a powerful and discriminative image representation that has been widely applied in a large range of applications. In this work, we have exploited three different types of local features, which also consider color information, namely: (i) DSP-SIFT [30], (ii) OpponentSIFT [172], and (iii) C-SIFT [172].

To obtain a fixed-length feature representation for each image, we followed [11] and used a standard Bag-of-Words approach. To avoid very high dimensional feature representations, which may yield to curse of dimensionality issues, the IFV approach has not been used in such a case.

**Data-Driven:** All the aforementioned features are the result of an hand-craft designing process that is conducted by humans on the basis of the a priori knowledge of the problem. However, in the recent past such a task has been widely obscured by the now well known feature learning procedure. With such a task, a machine learning algorithm is trained to learn the most suitable image representation for the given classification/regression task. Following such a motivation, we have considered feature representations that were learned from natural images. Specifically, we have followed [139] to compute the data-driven feature representation: the given image is fed to the OverFeat Convolutional Neural Network (CNN) [153]. Then, the CNN features are taken from the output of the last convolutional layer.

### 3.2.3 Extreme Learning Machines

An Extreme Learning Machine (ELM) [70] is a particular Single Layer Feed-forward Network (SLFN) which was inspired by biological learning and proposed to overcome

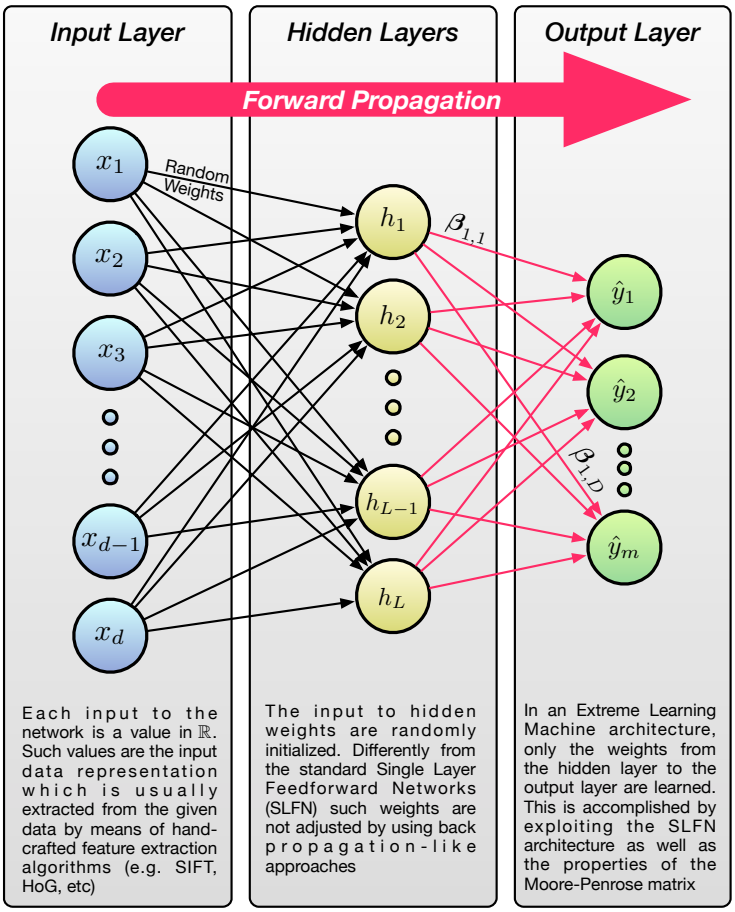



Figure 3.2: General architecture of a standard Extreme Learning Machine. The input-to-hidden weights are randomly generated whereas the hidden-to-output weights are learned analytically, without the need of an iterative process.

the issues faced by back propagation-based learning algorithms. In particular, the hidden nodes of an ELM compute random combinations of the input values. Thus, the input-to-hidden weights do not need to be learned with computationally expensive learning algorithms and can be randomly generated, independently from the input data (see Figure 3.2). In such a network, it is proven that the universal approximation theorem still holds under mild assumptions, provided that the hidden layer has enough nodes [70]. Computing the final output is then just a matter of finding the optimal hidden-to-output weights, which can be conveniently done in an analytical way, without the need of computationally expensive iterative processes, such as backpropagation.

### Preliminaries

More formally, let $\mathbf{x}$ be a $d$-dimensional row feature vector belonging to one out of $m$ possible classes. To such a feature vector corresponds a class label $\mathbf{y}$ represented by a $m$-dimensional unit row vector. Its single positive $c$-th component, denoted as $y_c$, indicates that $\mathbf{x}$ belongs to class $c \in \mathcal{C} = \{1, \ldots, m\}$. The vector $\mathbf{x}$ is the input for the ELM. Thus, the value of the $j$-th input neuron corresponds to the $j$-th component in a data sample $\mathbf{x}$.

Let the hidden layer be composed of $L$ hidden neurons and let its connection with the input neurons be denoted as the random matrix $\mathbf{R} \in \mathbb{R}^{d \times L}$. The output of the $j$-th neuron in the hidden layer is computed as $h_j(\mathbf{x}) = \sigma(\mathbf{x}, \mathbf{R}_j, b_j)$, where $\mathbf{R}_j$ is the $j$-th column of $\mathbf{R}$, $b_j$ is a random bias parameter associated with the $j$-th hidden neuron and $\sigma(\cdot)$ is a differentiable activation function, such as the:

(i) **Sigmoid function:**

$$\sigma(\mathbf{x}, \mathbf{R}_j, b_j) = \frac{1}{1 + \exp(-(\mathbf{x}^T \mathbf{R}_j + b_j))} \tag{3.1}$$

(ii) **Hyperbolic tangent function:**

$$\sigma(\mathbf{x}, \mathbf{R}_j, b_j) = \tanh(\mathbf{x}^T \mathbf{R}_j + b_j) \tag{3.2}$$

(iii) **Gaussian function:**

$$\sigma(\mathbf{x}, \mathbf{R}_j, b_j) = \exp(-b_j \|\mathbf{x} - \mathbf{R}_j\|^2) \tag{3.3}$$

Let $\boldsymbol{\beta}_j \in \mathbb{R}^m$ be a row vector of connection weights between the $j$-th hidden neuron and the output layer such that the output of the network is given by

$$\hat{\mathbf{y}} = \sum_{j=1}^{L} \boldsymbol{\beta}_j h_j(\mathbf{x}) \tag{3.4}$$

which, expressed in a more compact form, results in

$$\hat{\mathbf{y}} = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} \tag{3.5}$$

where $\boldsymbol{\beta} \in \mathbb{R}^{L \times m}$ is the matrix of weights connecting the hidden layer composed of $L$ nodes with the $m$ output nodes and $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \ldots, h_L(\mathbf{x})]$ the output row vector of the hidden layer with respect to the input $\mathbf{x}$.

In particular, $\mathbf{h}(\mathbf{x})$ maps the data from the d-dimensional input space $\mathcal{X}$ to the L-dimensional hidden layer random feature space $\mathcal{H}$ where the input-to-hidden node weights $\mathbf{R}$ are randomly generated according to any continuous sampling distribution probability. This has the advantage of projecting the data onto a lower dimensional space (i.e., compress the data dimensionality) or to a higher dimensional space (i.e., to increase sparsity) [85]. Ideally, to not lose valuable discriminative property of the input data, the random weights (i.e., the projection matrix) $\mathbf{R}$ should provide a stable embedding that approximately preserves the distance between all pairs of original features. As proved in [70], if the original points are projected onto a randomly selected subspace with suitably high dimensions, then the Johnson-Lindenstrauss lemma [80]

is satisfied with high probability and thus the distances between the points in the random space are preserved. As shown [69], a matrix satisfying such restricted isometry property is the random Gaussian matrix where each element of the random projection matrix $r_{i,j} \sim \mathcal{N}(0,1)$. Finally, the hidden layer output mapping $\mathbf{h}(\mathbf{x})$ has universal approximation capability property, i.e.

$$\lim_{L \to \inf} \| \sum_{j=1}^{L} \boldsymbol{\beta}_j h_j(\mathbf{x}) - \mathbf{y} \| = 0 \tag{3.6}$$

holds with probability one with appropriate hidden-to-output connection weights $\boldsymbol{\beta}$.

### ELM Learning

Let $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{n}$ be the set of $n$ training samples pairs and

$$\mathbf{H}(\mathbf{R}, b) = \begin{bmatrix} \mathbf{h}(\mathbf{x}^{(1)}) \\ \mathbf{h}(\mathbf{x}^{(2)}) \\ \vdots \\ \mathbf{h}(\mathbf{x}^{(n)}) \end{bmatrix} \begin{bmatrix} \mathbf{h}(\mathbf{x}_1), \cdots, \mathbf{h}(\mathbf{x}_n) \end{bmatrix} \tag{3.7}$$

$$= \begin{bmatrix} h_1(\mathbf{x}_1) & \cdots & h_L(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ h_1(\mathbf{x}_n) & \cdots & h_L(\mathbf{x}_n) \end{bmatrix} \tag{3.8}$$

$$= \begin{bmatrix} \sigma(\mathbf{x}^{(1)}, \mathbf{R}_1, b_1) & \cdots & \sigma(\mathbf{x}^{(1)}, \mathbf{R}_L, b_L) \\ \vdots & \ddots & \vdots \\ \sigma(\mathbf{x}^{(n)}, \mathbf{R}_1, b_1) & \cdots & g(\mathbf{x}^{(n)}, \mathbf{R}_L, b_L) \end{bmatrix} \tag{3.9}$$

be the hidden layer output matrix (i.e., the projection of the input feature vectors onto the ELM random feature The $j$-th column of $\mathbf{H}$ is the output of the $j$-th hidden node with respect to inputs $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(n)}$, parametrized by $\mathbf{R}_j$ and $b_j$.

Let also

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}^{(1)} \\ \vdots \\ \mathbf{y}^{(n)} \end{bmatrix} = \begin{bmatrix} y_1^{(1)} & \cdots & y_m^{(1)} \\ \vdots & \ddots & \vdots \\ y_1^{(n)} & \cdots & y_m^{(n)} \end{bmatrix} \tag{3.10}$$

be the training data target matrix. The ELM aims to learn the set of hidden-output weights $\boldsymbol{\beta}$ by solving

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{Y}. \tag{3.11}$$

As shown in [65], if the number of hidden neurons is equal to the number of distinct training samples (i.e., $n = L$), then the hidden layer output matrix $\mathbf{H}$ is square and invertible. Hence, a standard SLFN can approximate the given training samples with zero error. However, in the majority of the cases the number of hidden

neurons is different from the number of distinct training samples (i.e., $n \neq L$), hence, $\mathbf{H}$ is non-square matrix and there may not exist suitable parameters such that there exist a solution for $\mathbf{H}\boldsymbol{\beta} = \mathbf{Y}$.

A common approach to address such a problem is to estimate specific parameters $\hat{\mathbf{R}}$, $\hat{b}$, $\hat{\boldsymbol{\beta}}$ such that

$$\arg\min_{\mathbf{R},b,\boldsymbol{\beta}} \|\mathbf{H}(\mathbf{R},b)\boldsymbol{\beta} - \mathbf{Y}\| \tag{3.12}$$

which is equivalent to minimizing the cost function

$$J(\mathbf{R},b,\boldsymbol{\beta}) = \sum_{i=1}^{n}\left(\sum_{j=1}^{L}\boldsymbol{\beta}_j\sigma\left(\mathbf{x}^{(i)},\mathbf{R}_j,b_j\right) - \mathbf{y}^{(i)}\right)^2 \tag{3.13}$$

$$= \sum_{i=1}^{n}\left(\sum_{j=1}^{L}\boldsymbol{\beta}_j h_j\left(\mathbf{x}^{(i)}\right) - \mathbf{y}^{(i)}\right)^2. \tag{3.14}$$

When $\mathbf{H}(\mathbf{R},b)$ is unknown, gradient-based learning algorithms, like back propagation, are generally used to search the minimum of $\|\mathbf{H}(\mathbf{R},b)\boldsymbol{\beta} - \mathbf{Y}\|$. However, as proved in [70] the input weights and the hidden layer thresholds of an SLFN need not to be adjusted and can be arbitrarily given. For fixed input-to-hidden weights $\mathbf{R}$ and hidden layer thresholds $b$, training an SLFN is equivalent to finding a least-squares solution $\hat{\boldsymbol{\beta}}$ of the linear system

$$\arg\min_{\boldsymbol{\beta}} \|\mathbf{H}\boldsymbol{\beta} - \mathbf{Y}\|. \tag{3.15}$$

According to [70], ELMs exploit the aforementioned property and find the least-squares solution of the above linear system by computing

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{Y} \tag{3.16}$$

where $\mathbf{H}^{\dagger}$ is the MoorePenrose generalized inverse [2] of matrix $\mathbf{H}$.

However, from the learning point of view, ELMs also aims to reach the smallest training error but also the smallest norm of output weights[68, 70]. This is motivated by the fact that, as demonstrated in [5, 4], the smaller the weights are, the better generalization performance a network tends to have. Thus, the final ELM objective is to find $\hat{\boldsymbol{\beta}}$ such that

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{\beta}\| + \|\mathbf{H}(\mathbf{R},b)\boldsymbol{\beta} - \mathbf{Y}\|. \tag{3.17}$$

It turns out that the solution to such a problem is still the one in eq.(3.16). Indeed, the Moore-Penrose generalized inverse produces the smallest norm least-squares solution to the given linear system, hence fully satisfies the ELM objective.

From this it can be shown that, by using the orthogonal projection method [137],

the hidden-to-output connection weights can be computed as

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^T \left(\mathbf{H}\mathbf{H}^T\right)^{-1} \mathbf{Y} \tag{3.18}$$

$$= \left(\mathbf{H}^T\mathbf{H}\right)^{-1} \mathbf{H}^T\mathbf{Y} \tag{3.19}$$

where eq(3.18) holds when $\mathbf{H}\mathbf{H}^T$ is nonsingular and eq(3.19) is valid when $\mathbf{H}^T\mathbf{H}$ is nonsingular.

In addition, to achieve a stabler solution and to obtain better generalization performance [71, 66], the ridge regression theory [60] can be exploited, and a positive value $1/C$ can be added to the diagonal elements of $\mathbf{H}\mathbf{H}^T$. By considering this, and substituting eq.(3.18) in eq.(3.5), the predicted output of a new sample can be computed as

$$\hat{\mathbf{y}} = \mathbf{h}(\mathbf{x})\mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1} \mathbf{Y}. \tag{3.20}$$

### Kernel ELM

Standard ELMs perform an initial projection onto the ELM random feature space by means of a linear mapping. To obtain a stable embedding that preserves the distance between original data points such a mapping is randomly generated according to any continuous sampling probability distribution.

However, this may still yield to: (i) lose of relevant data dimensions (if $L$ is too small); (ii) curse of dimensionality and high computational load issues (if $L$ is too large). This introduces the problem of selecting the proper number of hidden neurons for the considered task.

In particular, it should be noticed that the curse of dimensionality problem states that the difficulty of an estimation problem increases drastically with the dimension $L$ of the space. This is due to the fact that as a function of $L$, exponentially many patterns are required to properly sample the $L$-dimensional space. This well-known statement induces some doubts about whether it is a good idea to go to a high-dimensional feature space for learning.

On the other hand, statistical learning theory shows that the contrary holds as well. Indeed, learning in a very high dimensional space can be simpler if a low complexity model is adopted. In particular, this is true if a low complexity mapping accounting for the variability and richness of the data is used [173]. As demonstrated in [124], such a mapping can be defined by a *kernel function*.

In light of this, instead of computing the mapping to the ELM random feature space by means of a random projection matrix (i.e., by using $\mathbf{h}(\mathbf{x}) : \mathcal{X} \rightarrow \mathcal{H}$), kernel functions are exploited. Let $\phi(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ be a kernel function, like a

(i) **Gaussian RBF:** $\phi(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp\left(\left(-\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2\right)/2\sigma^2\right)$

(ii) **Polynomial:** $\phi(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \left(\langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle + \theta\right)^d$

(iii) **Sigmoidal:** $\phi(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \tanh\big(\kappa\langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)}\rangle + \theta\big)$

then, let $\mathbf{\Phi}$ be the kernel matrix such that $\mathbf{\Phi}_{i,j} = \phi(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$. By substituting the random projection matrix $\mathbf{H}$ with the kernel matrix $\mathbf{\Phi}$, the objective in eq.(3.17) can then be re-written as

$$\arg\min_{\boldsymbol{\beta}} \|\boldsymbol{\beta}\| + \|\mathbf{\Phi}\boldsymbol{\beta} - \mathbf{Y}\|. \tag{3.21}$$

As before, since the kernel projection is known, only the hidden-to-output connection weights have to be learned. The solution to the aforementioned problem is the same suggested in eq.(3.16), where the generalized pseudo inverse should be computed for $\mathbf{\Phi}$ instead of for $\mathbf{H}$.

As demonstrated in [71], by adopting kernels in ELM one can avoid to select the number of hidden neurons $L$, and the predicted output class of a new data sample $\hat{\mathbf{x}}$ can be computed as

$$\hat{\mathbf{y}} = \begin{bmatrix} \phi(\hat{\mathbf{x}}, \mathbf{x}^{(1)}) \\ \vdots \\ \phi(\hat{\mathbf{x}}, \mathbf{x}^{(n)}) \end{bmatrix}^T \left(\frac{\mathbf{I}}{C} + \mathbf{\Phi}\right)^{-1} \mathbf{Y} \tag{3.22}$$

where $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$ and $\mathbf{\Phi}$ are the training data samples and the training kernel matrix, respectively.

### ELM Committee

As a result of the ELM training procedure carried out with the kernel extension, the set of hidden-to-output layer connection weights are learned. Such weights are learned by exploiting the set $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^n$ of $n$ $d$-dimensional features. A feature vector $\mathbf{x}^{(i)}$ is typically computed by concatenating all the feature vectors deriving from different cues (e.g., color histograms, Histogram of Oriented Gradients, etc.). While such an approach is widely and successfully adopted, it may introduce several problems. Indeed, if many multiple features are considered to represent the input data, then the overall joint feature dimension may be very large. Thus, the computational load is increased. In addition, low dimensional features are usually dominated by high dimensional ones, hence these are somehow considered as more important for discriminating between input data patterns.

To address the aforementioned problems an approach that separately considers the different types of features is proposed. Such an approach is named ELM committee and works as follows. Let $\{(\mathbf{x}_*^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^n$ be the set of $n$ training samples pairs, where $\mathbf{x}_*^{(i)} = \{\mathbf{x}_k^{(i)} : \mathbf{x}_k^{(i)} \in \mathbb{R}^{d_k}, k = 1, \ldots, K\}$ denotes the set of $K$ different feature types (e.g., color histogram, Histogram of Oriented Gradients, etc.) extracted for the $i$-th training data sample and $d_k$ indicates the dimensionality of the $k$-the feature type. Thus, it is not necessary that, features of different type span the same space, i.e., $|\mathbf{x}_1^{(i)}|$ may be different to $|\mathbf{x}_2^{(i)}|$.

In the ELM committee approach, an ELM is required to learn the optimal hidden-to-output connection weights $\hat{\boldsymbol{\beta}}$ for each feature type separately. Therefore, $K$ different sets of connection weights are learned by solving

$$\hat{\boldsymbol{\beta}}_k = \arg\min_{\boldsymbol{\beta}_k} \|\boldsymbol{\beta}_k\| + \|\boldsymbol{\Phi}_k \boldsymbol{\beta}_k - \mathbf{Y}\| \tag{3.23}$$

where $\boldsymbol{\Phi}_k$ denotes the ELM kernel mapping for the $k$-th feature type.

Since kernel ELMs are separately trained on different features, the solution to their objectives produce $K$ separate predictors, each one denoted as

$$\hat{\mathbf{y}}_k = \begin{bmatrix} \Phi_k(\hat{\mathbf{x}}, \mathbf{x}_k^{(1)}) \\ \vdots \\ \Phi_k(\hat{\mathbf{x}}, \mathbf{x}_k^{(n)}) \end{bmatrix}^T \left( \frac{\mathbf{I}}{C} + \boldsymbol{\Phi}_k \right)^{-1} \mathbf{Y} \tag{3.24}$$

where $\mathbf{x}_k^{(1)}, \ldots, \mathbf{x}_k^{(n)}$ and $\boldsymbol{\Phi}_k$ are the training data samples of type $k$ and the corresponding training kernel matrix, respectively. Such a procedure is highly parallelizable and can easily scale to high-dimensional problems.

### 3.2.4 Committee Supervisor

In the previous sections a learning approach to separately model $K$ different types of features has been introduced. This reduces the computational loads and allows to better handle the problems of dominating high dimensional features. Despite such benefits, since an answer is given by each committee member (i.e., each single ELM), there should be an appropriate pooling procedure that collects them to provide a final decision. Such a task is accomplished by the committee supervisor.

The committee supervisor can be as simple as a pooling operator (e.g., average pooling, max pooling, etc.). The common output of the supervisor is a class label. However, we believe that when classification results have to be presented to users (as in the case of object recognition), a ranking could be more appropriate. Towards this objective, we introduce a supervisor that aims to learn the coefficients $\boldsymbol{\alpha}$ of the linear combination of the $k = 1, \ldots, K$ member answers $\hat{\mathbf{y}}_k$ such that an optimal ranking can be obtained. A Structural Support Vector Machine (i.e., the supervisor) has been selected for such a task. The overall architecture is shown in figure 3.3.

#### The Structural Supervisor Objective

Let $\mathcal{X}$ and $\mathcal{O}$ denote the input feature (i.e., $\mathbf{x} \in \mathcal{X}$) and the output (i.e., $\mathbf{o} \in \mathcal{O}$) spaces, respectively. The idea behind Structural SVM [169, 170, 78] is to discriminatively learn a scoring function $f : \mathcal{X} \times \mathcal{O} \to \mathbb{R}$ over input/output pairs, where the space of the outputs $\mathcal{O}$ is no longer restricted to contain only numbered labels (as in common classification problems), but it is a structured output space whose elements may be sequences, strings, lattices, etc. In SELC, the structured output space consists in a

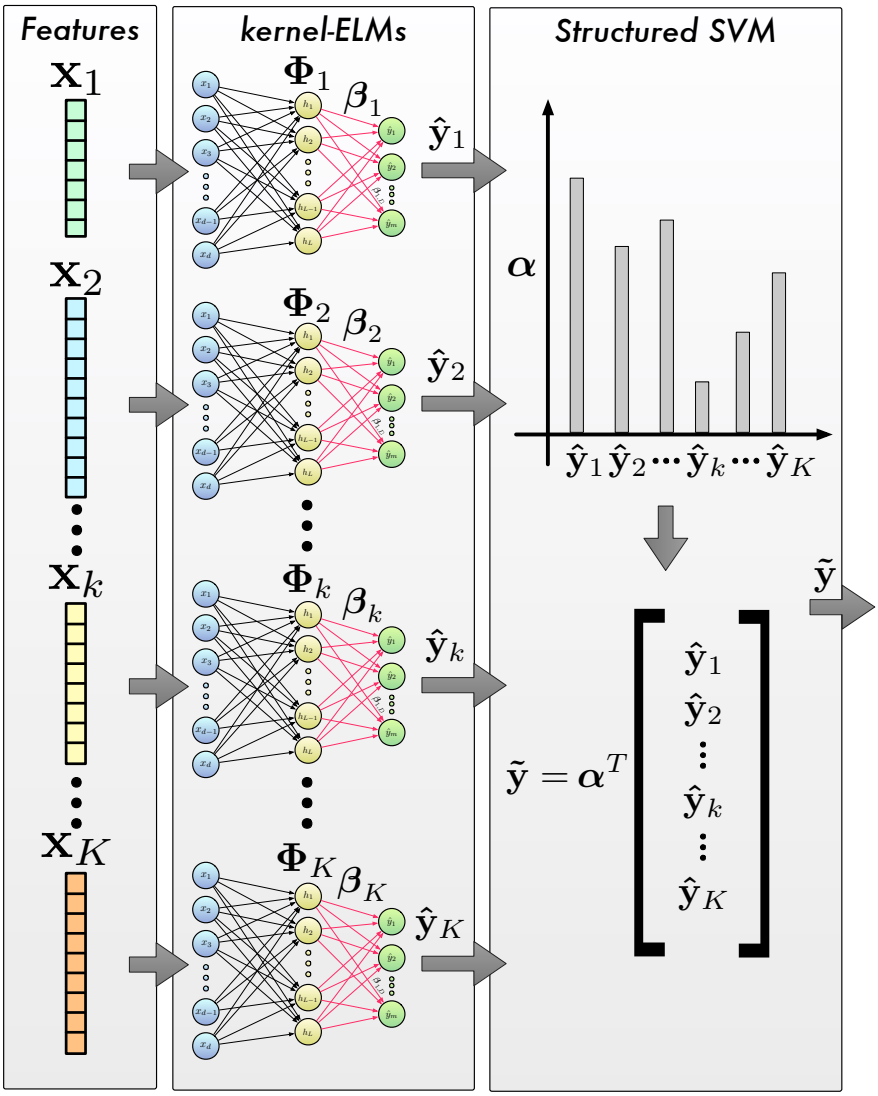



Figure 3.3: Architecture of the proposed Supervised Extreme Learning Committee approach.

ranking of the considered classes. Since the true class should be always ranked first, such structured output space also yields to optimal classification performance.

As before, let $\mathcal{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{n}$ be the input set and $c^{(i)} \in \mathcal{C} = \{1, \ldots, m\}$ denote the class of the $i$-th sample. For a given sample $\mathbf{x}^{(i)}$, the objective is to learn the coefficients $\boldsymbol{\alpha}$ that order relevant classes $\mathcal{C}^{(i)+} \subseteq \mathcal{C}$ (i.e., classes "similar" the same

class of the sample) before irrelevant ones $\mathcal{C}^{(i)-} \subseteq \mathcal{C}$ (i.e., classes "different" from the class of the sample).

However, in common classification problems there is only knowledge of the order between the relevant and irrelevant classes, but not of the order within relevant or irrelevant ones. To overcome such an issue, we consider the query-class set of a sample $\mathbf{x}^{(i)}$ as a partially ordered set, where the partial order $\mathbf{o}^{(i)}$ is defined as

$$\mathbf{o}^{(i)} = \{o^{(i)^+,(i)^-}\}, \quad o^{(i)^+,(i)^-} = \begin{cases} +1 & \text{if} \quad c^{(i)+} \prec c^{(i)-} \\ -1 & \text{if} \quad c^{(i)+} \succ c^{(i)-} \end{cases} \tag{3.25}$$

where $c^{(i)+} \prec c^{(i)-}$ indicates that a relevant class $c^{(i)+} \in \mathcal{C}^{(i)+}$ is ranked before an irrelevant one $c^{(i)-} \in \mathcal{C}^{(i)-}$, and after otherwise.

In a structural SVM model the objective function is defined by

$$f(\mathbf{x}; \boldsymbol{\alpha}) = \arg\max_{\mathbf{o} \in \mathcal{O}} F(\mathbf{x}, \mathbf{o}; \boldsymbol{\alpha}) \tag{3.26}$$

where $\boldsymbol{\alpha}$ denotes a parameters vector. It might be useful to think of $F$ as a family of cost functions –parametrized by $\boldsymbol{\alpha}$, which measure how well the output $\mathbf{o}$ matches the input of interest $\mathbf{x}$.

By letting $\Psi(\mathbf{x}, \mathbf{o})$ be a combined feature representation of inputs and outputs and assuming $F$ to be linear in such a feature space, i.e.

$$F(\mathbf{x}, \mathbf{o}; \boldsymbol{\alpha}) = \langle \boldsymbol{\alpha}, \Psi(\mathbf{x}, \mathbf{o}) \rangle \tag{3.27}$$

then, the appropriate margins and constraints can be introduced to formulate the structural SVM with slack rescaling objective [78] as

$$\begin{gathered} \min_{\boldsymbol{\alpha}, \xi \geq 0} \frac{1}{2}\|\boldsymbol{\alpha}\|^2 + \frac{\gamma}{n}\sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad \forall i, \forall \tilde{\mathbf{o}}^{(i)} \in \mathcal{O} \setminus \mathbf{o}^{(i)} : \\ \langle \boldsymbol{\alpha}, \Psi(\mathbf{x}^{(i)}, \mathcal{C}; \mathbf{o}^{(i)}) - \Psi(\mathbf{x}^{(i)}, \mathcal{C}; \tilde{\mathbf{o}}^{(i)}) \rangle \geq 1 - \frac{\xi_i}{\Delta(\mathbf{o}^{(i)}, \tilde{\mathbf{o}}^{(i)})} \end{gathered} \tag{3.28}$$

where $\gamma$ is a parameter that controls the trade-off between the norm of the coefficients $\boldsymbol{\alpha}$ and the average of the slack variables $\xi_i$. $\Psi(\cdot)$ is the combined feature representation (details in the following) and $\mathcal{O}$ is the space consisting of all possible partial orders. Within such a space, $\mathbf{o}^{(i)}$ denotes a correct partial order that ranks all relevant classes before irrelevant ones while $\tilde{\mathbf{o}}^{(i)}$ is an incorrect partial order that violates some of the pairwise relations. Finally, $\Delta(\mathbf{o}^{(i)}, \tilde{\mathbf{o}}^{(i)})$ is a suitable loss function that quantifies the loss associated with predicting a wrong partial order.

The constraints in eq.(3.28) state that, for each sample, the score $\langle \boldsymbol{\alpha}, \Psi(\mathbf{x}^{(i)}, \mathcal{C}; \mathbf{o}^{(i)}) \rangle$ of a correct order $\mathbf{o}^{(i)}$ must be greater than the score $\langle \boldsymbol{\alpha}, \Psi(\mathbf{x}^{(i)}, \mathcal{C}; \tilde{\mathbf{o}}^{(i)})) \rangle$ of all incorrect orders $\tilde{\mathbf{o}}^{(i)}$ by a required margin. This margin equals 1 in the slack-rescaling formulation.

### Supervisor Learning

As shown in eq.(3.28), three main components have to be defined in order to properly achieve the final objective:

(i) The combined feature representation for inputs and outputs: $\Psi(\cdot)$;
(ii) The function used to compute the loss between a wrong and a correct partial order: $\Delta(\cdot,\cdot)$;
(iii) A separation oracle to optimize the given objective by means of the cutting plane approach.

**The Combined Feature Representation:** The flexibility in designing $\Psi(\cdot)$ has strongly pushed the adoption of Structural SVMs to attack a wide plethora of problems like natural language parsing [169], object detection [194] and segmentation [10], just to name a few. Therefore, the choice of such function highly depends on the task that one wants to address.

Since in the current approach only relevant and irrelevant pairs relationships are known, a modification of the commonly adopted partial order feature [119, 77] is used. This, denoted $\Psi(\mathbf{x}^{(i)}, \mathcal{C}; \mathbf{o}^{(i)})$, is computed as

$$\sum_{i^+=1}^{|\mathcal{C}^{(i)+}|} \sum_{i^-=1}^{|\mathcal{C}^{(i)-}|} o^{(i)^+,(i)^-} \frac{(\boldsymbol{\psi}(\mathbf{x}^{(i)}, c^{(i)+}) - \boldsymbol{\psi}(\mathbf{x}^{(i)}, c^{(i)-}))}{|\mathcal{C}^{(i)+}||\mathcal{C}^{(i)-}|}. \tag{3.29}$$

In the proposed case, the order should be optimized over the committee members decisions, thus the feature $\boldsymbol{\psi}$ is represented by the output of the $K$ committee members

$$\boldsymbol{\psi}(\mathbf{x}^{(i)}, c^{(i)}) = \left[\hat{y}^1_{c^{(i)}}, \ldots, \hat{y}^K_{c^{(i)}}\right]^T \tag{3.30}$$

where $\hat{y}^k_{c^{(i)}}$ is the output computed by the $k$-th member with respect to class label $c^{(i)}$.

Such partial order feature is suitable for the proposed objective because it only depends on the difference between relevant and irrelevant pairs, not the entire list. By adding the differences of correct orders and subtracting that of incorrect orders, the partial order feature emphasizes the directions in feature space which are closely related to correct ordering.

**The Loss Function:** Similarly to the selection of a suitable combined feature representation, the choice of the loss function for Structural SVMs is also highly dependent on the task. Among all the possible loss functions that can be used, the area under curve (AUC) measure [77, 185] is the more appropriate for the proposed approach. Indeed, it allows to characterize the difference between relevant and and irrelevant pairs with only partial order available.

As shown in [77, 185], computing the AUC requires computing a ranking. This can be naturally obtained by ordering each example according to $\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\mathbf{x}^{(i)}, c^{(i)})\rangle$, for all $c^{(i)} \in \mathcal{C}$. From such a ranking, the partial ordering $\tilde{\mathbf{o}}^{(i)}$ can be computed and the

AUC loss calculated as

$$\Delta(\mathbf{o}^{(i)}, \tilde{\mathbf{o}}^{(i)}) = \sum_{i^+}^{|c^{(i)+}|} \sum_{i^-}^{|\mathcal{C}^{(i)-}|} \frac{\mathbf{1}_{\left(o^{(i)+,(i)-} \neq \tilde{o}^{(i)+,(i)-}\right)}}{|\mathcal{C}^{(i)+}||\mathcal{C}^{(i)-}|} \tag{3.31}$$

where $\mathbf{1}_{(\cdot)}$ is the indicator function. Thus, the AUC loss function tells, on average, how many incorrect orders are obtained with the current partial ordering $\tilde{\mathbf{o}}^{(i)}$.

**The Separation Oracle:** As shown in [77], learning a ranking function that optimizes an upper bound on the AUC loss on the training set requires a constraint for every possible wrong output $\tilde{\mathbf{o}}^{(i)}$. Unfortunately, the number of possible wrong outputs is exponential in the size of $\mathcal{C}$. Such a problem can be addressed by adopting a cutting plane algorithm which iteratively introduces constraints until the original problem is solved within a desired tolerance [170]. In such a case, one key step is to efficiently determine the separation oracle. Given a fixed $\boldsymbol{\alpha}$, for each example $\mathbf{x}^{(i)}$ the separation oracle aims to find the output $\dot{\mathbf{o}}^{(i)}$ associated with the most violated constraint, i.e.

$$\dot{\mathbf{o}}^{(i)} = \arg\max_{\dot{\mathbf{o}}^{(i)} \in \mathcal{O}} \langle \boldsymbol{\alpha}, \Psi(\mathbf{x}^{(i)}, \mathcal{C}; \mathbf{o}^{(i)}) \rangle + \Delta(\mathbf{o}^{(i)}, \dot{\mathbf{o}}^{(i)}). \tag{3.32}$$

For a fixed $\boldsymbol{\alpha}$, the argument maximizing eq.(3.32) can be found by sorting the committee answers by $\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\mathbf{x}^{(i)}, c^{(i)}) \rangle$ in descending order. This strongly reduces the computational complexity as the maximization objective in eq.(3.32) only requires $O(n \log n)$ processing time.

### 3.2.5 The Supervised Extreme Learning Committee Decision

In the preceding sections it has been shown how standard ELMs can be extended to include Kernels as well as how a committee of such Kernel-ELMs can be formed. Then, exploiting the idea that not every committee member should have the same power to take th final decision, a committee supervisor, based on structural SVM ranking, has been introduced.

Once the training procedure is completed, the learned parameters can be adopted to obtain the ranking for a new test data sample $\hat{\mathbf{x}}$. First, the committee members are asked to produce $K$ answers $[\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_K]$. Then, the learned supervisor coefficients $\boldsymbol{\alpha}$ are used to weights such answers. As a result, the output of the Supervised Extreme Learning Committee is computed as

$$\tilde{\mathbf{y}} = \boldsymbol{\alpha}^T \begin{bmatrix} \hat{\mathbf{y}}_1^T \\ \vdots \\ \hat{\mathbf{y}}_K^T \end{bmatrix}. \tag{3.33}$$

The ranking for the test data sample is finally obtained by sorting in descending order the elements in $\tilde{\mathbf{y}}$.

## 3.3 Experimental Results

To validate the proposed SELC approach, results on four benchmark datasets for food recognition have been computed. For each of them, an analysis of the performances of the selected features as well as on the benefits of the proposed approach with respect to standard ELMs is conducted first. Then, the advantages of the proposed supervisor with respect to other approaches are shown. Finally, comparisons with state-of-the-art methods are presented to show the superior performance of SELC.

As commonly performed in the evaluation of food recognition approaches [22, 32, 33], the achieved results are shown in terms of recognition accuracy and by means of the *top-n* criterion [32]. The *top-n* criterion defines the chance of obtaining a correct recognition within the first $n$ retrieved images.

The performances achieved by the existing methods have been taken from the corresponding works or have been directly provided by the authors.

### 3.3.1 Experimental Settings

To evaluate the performance of our approach, we have adopted the following settings for all the experiments. All the algorithm hyperparameters have been selected through 4-fold cross validation.

**Image Feature Representation:**

**Color:**
For every color channel a 32 bin histogram is extracted. Thus, the set of color histogram features consists of 5 histograms (i.e., one for each color space) each of which has dimensionality equal to 96.

**Shape:**

1. PHOG: features have been extracted from each color channel of a given image which has been projected onto the HSV color space first [110]. HOG quantized considering 9 bins have been extracted considering 3 levels of the spatial pyramid. The resulting PHOG feature vector consists of 2295 elements.
2. GIST: the same implementation settings used in [127] have been adopted to get the 512-D feature vector.

**Texture:**

1. LBP: the uniform rotation invariant descriptor [126] has been considered to extract the LBP descriptor using 8-neighbors and a radius of 1. The resulting vector consists of 59 elements.
2. LPQ: the basic LPQ version [136] with decorrelation and SIFT uniform $3 \times 3$ window for local frequency estimation has been used. The resulting 256-D vector contains the histogram of the LPQ codewords.

3. LCP: a feature vector of 81-D has been extracted considering 8 neighborhoods and a radius of 2.
4. BGP: The 216-D vector has been computed using the same settings described in [188].
5. PRICoLBP: The code released with [133] has been used to extract the 1770-D feature vector.
6. Textons: 300 Gaussian Mixtures have been considered to quantize the filter responses, hence to learn the codebook.

Apart from textons –which have been separately extracted from each channel of the RGB color space– all other considered texture feature representations have been extracted from grayscale images.

**Local:**

1. DSP-SIFT: The Domain Size Pooling-SIFT (DSP-SIFT) descriptor [30] introduces a simple modification of the original SIFT one. Specifically, gradient orientations of a grayscale image are pooled across different domain sizes in addition to the usual spatial locations. The descriptor computed for each detected keypoint lies in a 128-D space.
2. OppSIFT: The 384-D visual descriptor –computed for every detected interest point– describes all of the channels in the opponent color space. Due to the normalization of the SIFT descriptor, such a feature is invariant to changes in light intensity [172].
3. C-SIFT: In [172], authors showed that the O1 and O2 components of the opponent color space contain intensity information. To obtain a descriptor which is scale-invariant with respect to light intensity changes, the 384-D C-SIFT descriptor was proposed [172].

Each of these feature has been encoded using a BoW approach with 1000 codewords.

**Data-Driven:**
Following [139], the output for the last convolutional layer has been taken as the image feature representation. As a result each image is described by a 4096-D feature vector.

When jointly considered, the resulting feature vector lies in a 19770-D feature space.

### Kernels

When kernel-ELMs are used, their performances are evaluated using four different kernels, namely the:

1. linear kernel;
2. cosine kernel;
3. exponential $\chi^2$ kernel;
4. radial basis function kernel (with free parameter set to 1);

### Datasets

To validate the proposed method four publicly available benchmark datasets have been considered. The PFID [22], UNICT-FD889 [32], the UECFood100 [117] and the Food-101 [16] datasets have been selected because they provide different food recognition challenges:

1. The PFID dataset has images acquired under different lighting conditions and from different viewing angles. Therefore, it is useful to understand if the proposed method is robust to such challenges.
2. The UNICT-FD889 dataset has images of 889 different real food plates acquired by mobile devices in uncontrolled scenarios. Hence, results on this dataset provide an estimate on how well an algorithm scales to a real scenario.
3. The UECFood100 dataset contains about 14000 images, corresponding to 100 different food categories.
4. Similarly, the Food-101 dataset has images of 101 different foods. However, in such case 1000 images for each category are available. Due to the large number of images, these two datasets are well suited to evaluate the learning performance of the proposed approach.

More details regarding each dataset are given in the following.

### Experimental Scenarios

Three main different scenarios have been selected to analyze the performance of the proposed approach. These are the followings:

1. To see how single features perform on the food recognition task, performances obtained by separately exploiting each considered type of feature have been computed.
2. To demonstrate the benefits of kernel ELM over standard ELM, results will be also given for the case when kernel is not used. Under such a scenario, the number of hidden neurons has been set to $L = 1000$.
3. To demonstrate the benefits of the proposed kernel-ELM committee supervisor, the achieved performance are compared to those obtained by using common fusion schemes: (a) *low*-level consists in feature concatenation; (b) *mid*-level, where the kernels computed for different features are combined. Results have been obtained by kernel averaging [40], kernel product [40] and by exploiting the Sparse and Non-Sparse version of Multiple Kernel ELMs (MKELMs) [108]; (c) *high*-level, where committee members outputs are fused using the weights learned by means of score averaging, Lasso and Logistic Regression (our method belongs to such a category).

In all the following results, the performances shown for both the *mid* and *high*-levels fusion schemes (SELC included) have been computed using the $\chi^2 - exp$ kernel for every feature type. Notice that, the kernels could have been separately selected for each dataset to obtain better recognition performance. However, to provide a more general framework, the choice of the kernel has been kept fixed.

Figure 3.4: 15 randomly selected samples from the PFID dataset. Each column corresponds to a different type of food (i.e., a different class). Rows show three different instances. In the PFID dataset some food classes (i.e., items on the 5th, 6th and 7th columns) are very similar to each other.

### 3.3.2 Performance Evaluation

#### PFID Dataset

The Pittsburgh Fast-food Image Dataset[1] (PFID) was the first dataset build exclusively for food recognition [22]. The PFID contains data of three instances of 61 different food items which were purchased on different dates from the same restaurant or at different branches of the same fast food chain. Each instance has 6 images collected under different lighting conditions, with different background, and sensed from different viewing angles (see Figure 3.4). As a result, the whole dataset contains 1089 images of 61 different classes of food.

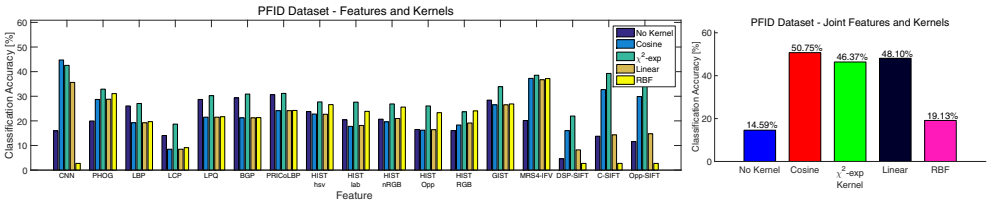
Following the protocol in [182, 33], performance evaluations have also been conducted by re-organizing the 61 PFID food categories into 7 major classes: Sandwiches, Salads&Sides, Chicken, Breads&Pastries, Donuts, Bagels, and Tacos. In both the cases, 3-fold cross-validation has been conducted using 12 images from two instances of each original class for training, and the 6 remaining images of the third instance of each original class for testing [182, 33].

**Performance Analysis:**

In Figure 3.5(a) the performances achieved by the single features using different kernels are shown. Let consider the case when no kernel is used. Under such scenario, results demonstrate that PRIcoLBP features are the best performing ones with an accuracy of 30.62%. In general, color histograms and texture features perform better than the CNN ones. Despite this, when the Cosine kernel is used, the opposite occurs and CNN features yield to the best performance with an accuracy improvement of about 30%.

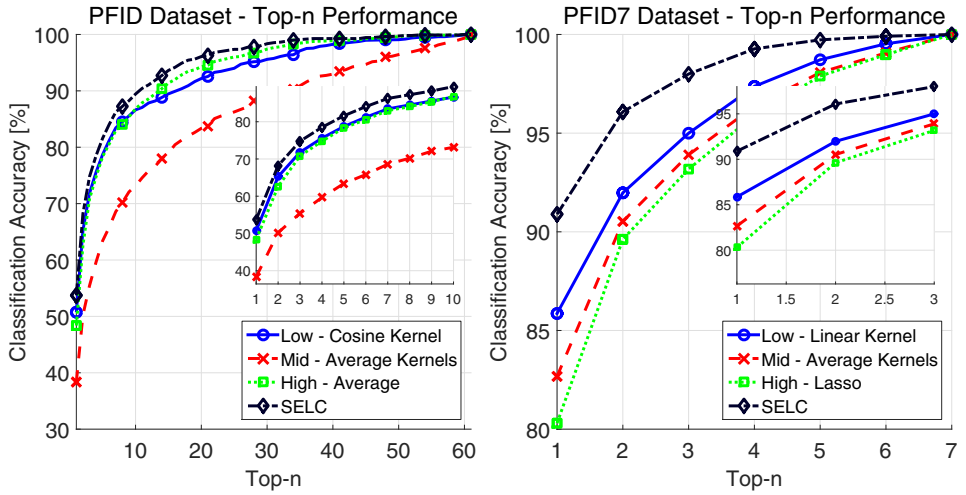
In Figure 3.5(b) the performances achieved by considering the joint feature space (i.e., low level fusion scheme) and using different kernels are shown. The depicted results show that, when no kernel is used, the accuracy is of 14.59%, which is even lower than the one achieved by using some features alone. If the Cosine kernel is

[1]Available at http://pfid.intel-research.net/




(a) (b)

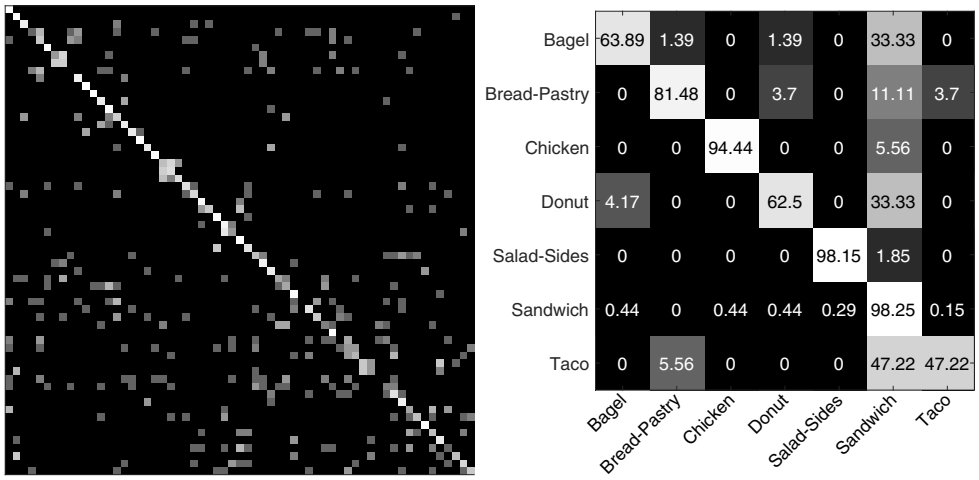
Figure 3.5: Accuracy performances obtained by the proposed method on PFID dataset. In ((a)) performances achieved by using single features with different kernels are given. In ((b)) performances achieved by considering the joint feature space and different kernels are shown.




(a) (b)

Figure 3.6: *Top–n* Performances on the PFID dataset using the proposed approach are compared to the results achieved considering the best performing low/mid/high level fusion schemes. In ((a)) results are computed considering all the 61 classes. In ((b)) results are computed considering only the 7 major classes. The inside pictures show the performance on a reduced range of *Top–n*.

adopted, performance reaches an accuracy of 50.75%, which is slightly better than the single performance achieved by CNN features only. This shows that, as for the single feature scenario, if a kernel is used performance will improve. However, dependently on the adopted kernel, the performances of the joint feature approaches are very close or even worse than the ones achieved by using the best single feature. This highlights the fact that, if jointly considered, there is no guarantee that adding more features to tackle the problem of food recognition improves the performance.




(a) (b)

Figure 3.7: Confusion matrices computed for both the standard PFID dataset ((a)) and its 7 major classes ((b)). The lighter the diagonal, the more effective the approach. In ((b)), the class labels and the correct classification percentages are shown.

To show the benefits of the proposed committee-based method, in Figure 3.6 the achieved *top–n* performances are compared to the ones obtained by the low/mid/high level fusion schemes having the best accuracy results on the considered dataset. Results have been computed considering both the 61 categories and the 7 major classes.

Results computed considering all the 61 categories (see Figure 3.6(a)) show that by using the proposed SELC approach the accuracy performance is of 53.73%. Thus, it SELC improves the best results obtained by considering the joint features and cosine kernel by about 3%. Such a gap increases more with larger values of $n$. A similar difference in performance is shown with respect to the average high level fusion scheme. Kernel averaging yields to a significant decrease in the performance. Indeed, in such a case the accuracy is of 38.36%, only.

When the 7 major classes are considered only, a similar behavior is achieved (see Figure 3.6(b)). Considering the accuracy reached by using the linear (85.86%) and the cosine (85.22%) kernels for the low level fusion scenario, the accuracy improves by 5.03% and by 5.67% respectively. When the best performing mid and high level fusion schemes used for comparisons are considered, SELC improves the corresponding accuracies by more than 7% and 10%, respectively.

To better analyze the performance of the proposed method, the confusion matrices shown in Figure 3.7 have been computed. The lighter the diagonal line, the more effective the approach, because it has a higher probability of classifying food of a given category as itself. When the 7 major classes are considered (Figure 3.7(b)), the correct classification percentages are shown together with the class labels.

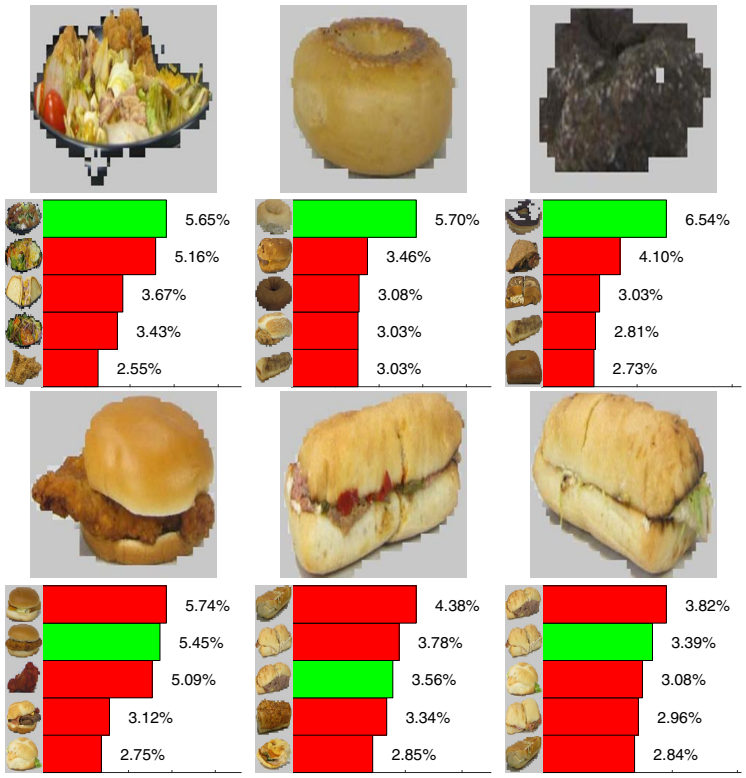



Figure 3.8: Performances achieved on the PFID datasets by the proposed method are shown for 6 query images (organized in two rows). The bar histograms show the score (in percentage) of the proposed approach for the true match (in green) and for the remaining top 4 matches (in red). On the y-axis of each bar histogram a randomly selected training image corresponding to the food class is depicted.

For both the scenarios, the diagonal elements show high probabilities, thus reflecting the capacities of the proposed approach in correctly classifying most of the plates. More interestingly, Figure 3.7(b) have a light vertical band showing that in a large number of cases a food is assigned to the Sandwich class. The motivation behind such behavior is that in the 7 major classes dataset, the majority of the samples belong to such a class. Hence the dataset is not well balanced among all the classes as it was in the case all the 61 categories were considered. Therefore, when the algorithm is trained with many Sandwich samples and very few Taco samples that shares similar characteristic as those, the classifier is not able to find a good decision boundary separating the two classes.

In Figure 3.8 the performances achieved by the proposed method are shown for 6 query images (see caption for additional details). The depicted results demonstrate that proposed approach is able to well capture the global appearance of the images and

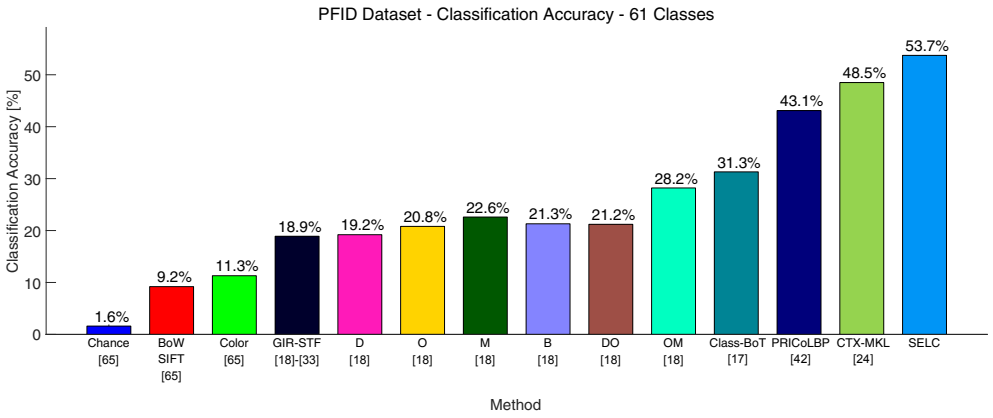



Figure 3.9: Comparisons with state-of-the-art methods computed considering all the 61 categories in the PFID dataset. Results are shown as classification accuracy.

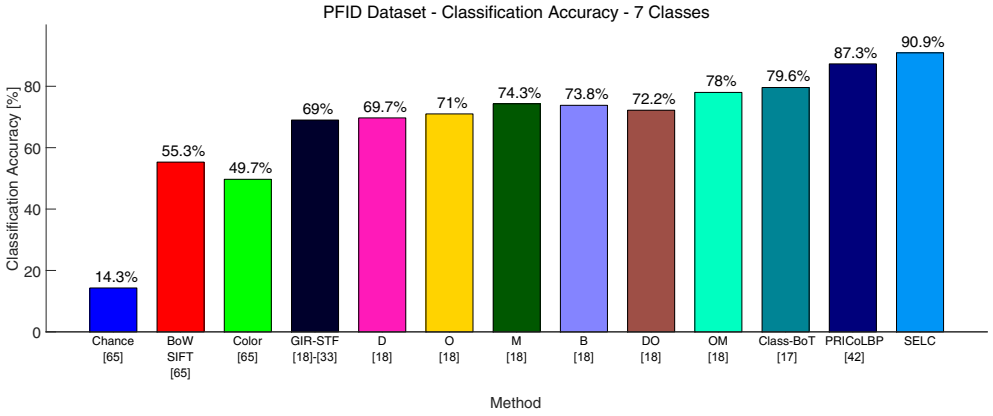



Figure 3.10: Comparisons with state-of-the-art methods on the PFID dataset considering only the 7 major classes. Results are shown as classification accuracy.

it also has the capacity to reliably find the true match under challenging conditions. When the query image is not correctly classified, or the considers cases are very challenging, the resulting scores are very close to each other, thus meaning there is uncertainty in the given answer.

**State-of-the-art Comparisons:** In Figure 3.9, the performance of the proposed SELC approach is compared to the state-of-the-art ones. The comparison is given with respect to the PFID dataset with all the 61 classes. Results demonstrate that the proposed approach improves the state-of-the-art performance of CTX-MKL [11] by more than 5.2% and outperforms recent approaches like Class-BoT [33] and OM [182] by more than 20%.

Table 3.1: Classification performances achieved by state-of-the-art methods on the 7 major classes of the PFID dataset. Since the number of images for the different classes is not balanced, for each class the per-class accuracy is given together with the corresponding number of images. Best results for each class are in boldface font.

| Images per class | Sandwich | Salad & Sides | Bagel | Donut | Chicken | Taco | Bread & Pastry |
|---|---|---|---|---|---|---|---|
| On each test run | 228 | 36 | 24 | 24 | 24 | 12 | 18 |
| Per class accuracy [%] (Number of Images) | Sandwich | Salad & Sides | Bagel | Donut | Chicken | Taco | Bread & Pastry |
| Color [22] | 69.0 (157.3) | 16.0 (5.8) | 13.0 (3.1) | 0.0 (0) | 49.0 (11.8) | 39.0 (4.7) | 8.0 (1.4) |
| BoW SIFT [22] | 75.0 (171) | 45.0 (16.2) | 15.0 (3.6) | 18.0 (4.3) | 36.0 (8.6) | 24.0 (2.9) | 3.0 (0.5) |
| GIR-STF [182]-[155] | 79.0 (180.1) | 79.0 (28.4) | 33.0 (7.9) | 14.0 (3.4) | 73.0 (17.5) | 40.0 (4.8) | 47.0 (8.5) |
| OM [182] | 86.0 (196.1) | 93.0 (33.5) | 40.0 (9.6) | 17.0 (4.1) | 82.0 (19.7) | 65.0 (7.8) | 67.0 (12.1) |
| Class-Based BoT [33] | 87.6 (199.7) | 84.3 (30.3) | **70.8 (17.0)** | 43.1 (10.3) | 66.7 (16.0) | **69.4 (8.3)** | 53.7 (9.7) |
| SELC | **98.25 (224.1)** | **98.15 (35.3)** | 63.89 (15.3) | **62.50 (15.0)** | **94.44 (22.7)** | 47.22 (5.7) | **81.48 (14.7)** |

In Table 3.1 and Figure 3.10 the accuracy performance comparison between the proposed approach and state-of-the-art ones on the 7 major classes of the PFID dataset are given. In addition, following in [33], to better understand the results in Table 3.1, the number of images belonging to the different classes are reported together with the per-class accuracy.

Results depicted in Figure 3.10 show that the accuracy performance of SELC (90.9%) outperforms all the considered algorithms, with PRI-CoLBP [133] being the closest ones with an accuracy of 87.3%.

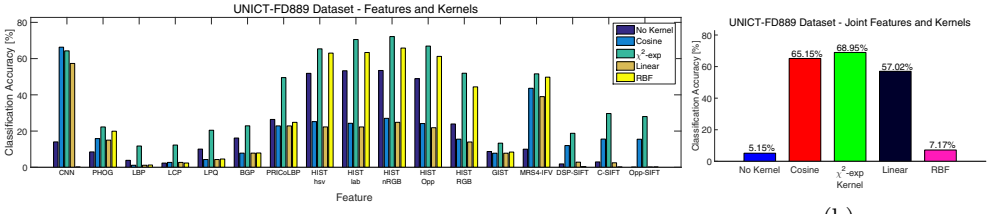
Results in Table 3.1 show that the main source of errors is the Taco food category. As already discussed, this is due to the imbalanced conditions of the dataset. Despite this, the proposed SELC approach performs better than existing ones in classifying 5 out of 7 categories. In the two remaining case, Class-based BoT [33] performs better. Notice that Class-based BoT requires that an encoding is computed for each different class of food. While such an approach could have been exploited in our work as well, we decided not to use it to limit the computational requirements.

### UNICT-FD889 Dataset

The UNICT-FD889 Dataset[2] has been recently introduced in [32]. The UNICT-FD889 dataset is the one that has the largest number of different classes to recognize. It comes with 3583 images related to 889 distinct food categories belonging to different

[2]Available at http://iplab.dmi.unict.it/UNICT-FD889

Figure 3.11: 15 randomly selected samples from the UNICT-FD889 dataset. Columns correspond to a different type of food (i.e., to a different class). Rows show the appearance variations between samples belonging to the same class.




(a) (b)

Figure 3.12: Accuracy performances on the UNICT-FD889 dataset achieved by: ((a)) using single features with different kernels and ((b)) considering the joint feature space and different kernels.

nationalities (e.g., Italian, English, Thai, Indian, Japanese, etc.). Images have been collected in a real and uncontrolled scenario (e.g., different backgrounds and light environmental conditions) by means of smartphones. Hence, the UNICT-FD889 dataset is a collection of food images acquired by users in real cases of meals. Each food belonging to a particular class has been acquired multiple times (four on average) to ensure geometric and photometric variabilities (see Figure 3.11 for a few examples).

To provide a fair comparison with existing methods, the following results have been computed by averaging the performance on the same three splits adopted in [32].

**Performance Analysis:** In Figure 3.12(a) the performances achieved by the single features using different kernels are shown. Differently from the results of single feature on the PFID dataset, in such a case the best classification accuracies are achieved using color histogram features. In particular, when no kernel is used, an accuracy of 53.44% is achieved using color histogram features extracted from the normalized RGB color space. CNN features do not perform well. Their classification accuracy obtained without using a kernel is of 14.03% only. However, when kernels are introduced, their performance increases significantly and a classification accuracy of 66.30% is reached using the Cosine kernel. Thus, as shown for the PFID dataset, results demonstrate that the choice of an appropriate kernel may strongly improve the recognition performance.

In Figure 3.12(b) the performances achieved by considering the joint feature space (i.e., low level fusion scheme) and using different kernels are shown. The depicted

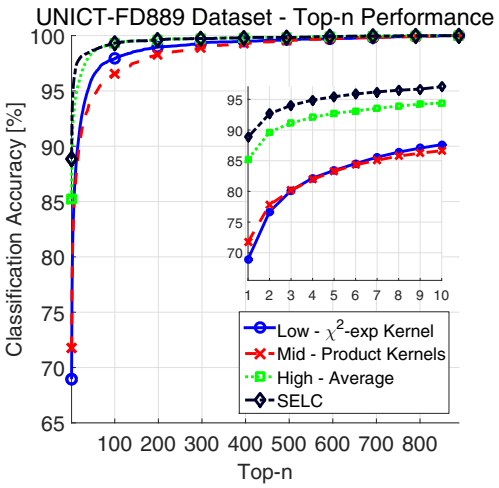



Figure 3.13: Performances on the UNICT-FD889 dataset using the proposed approach are compared to the results achieved considering the best performing low/mid/high level fusion schemes. Performance are given using the *top–n* criterion.

results show that, when no kernel is used on the joint feature space the obtained accuracy (5.15%) is much less than the one obtained by using color histogram features only. While the usage of a kernel drastically improves the accuracy performance, these are still on the same line as the ones achieved by color histograms. Indeed, using the $\chi^2 - exp$ kernel an accuracy of 68.95% is reached (which is very similar to the one obtained using the same kernel on normalized RGB histogram features, i.e. 72.16%).

In Figure 3.13 the *top–n* performance of the proposed SELC approach is compared to the ones achieved by the best performing low/mid/high level fusion schemes. Under such scenario, results show that significant benefits can be obtained by using the proposed high level fusion committee-based approach.

In particular, let us consider the results obtained by the low level fusion approach using the $\chi^2 - exp$ kernel. When $n = 1$, a classification accuracy of 68.95% is obtained using the joint feature space. SELC reaches a classification accuracy of 88.85%, thus yielding to a 20% performance improvement. Such a gap reduces to 1%, only when $n$=300. Similarly, when compared to the mid and high level fusion schemes, results show that a significant improvement is achieved. Specifically, an accuracy improvement of about 17% and 3.5% is obtained with respect to the Product Kernel [40] and the Average fusion schemes, respectively.

In Figure 3.14 the performances achieved by the proposed method are shown for 6 query images (organized in two rows). The reported cases show that the proposed approach is able to well capture both the global appearance of the images and the little details that differentiate two very similar classes (e.g., see the first query on the second row).

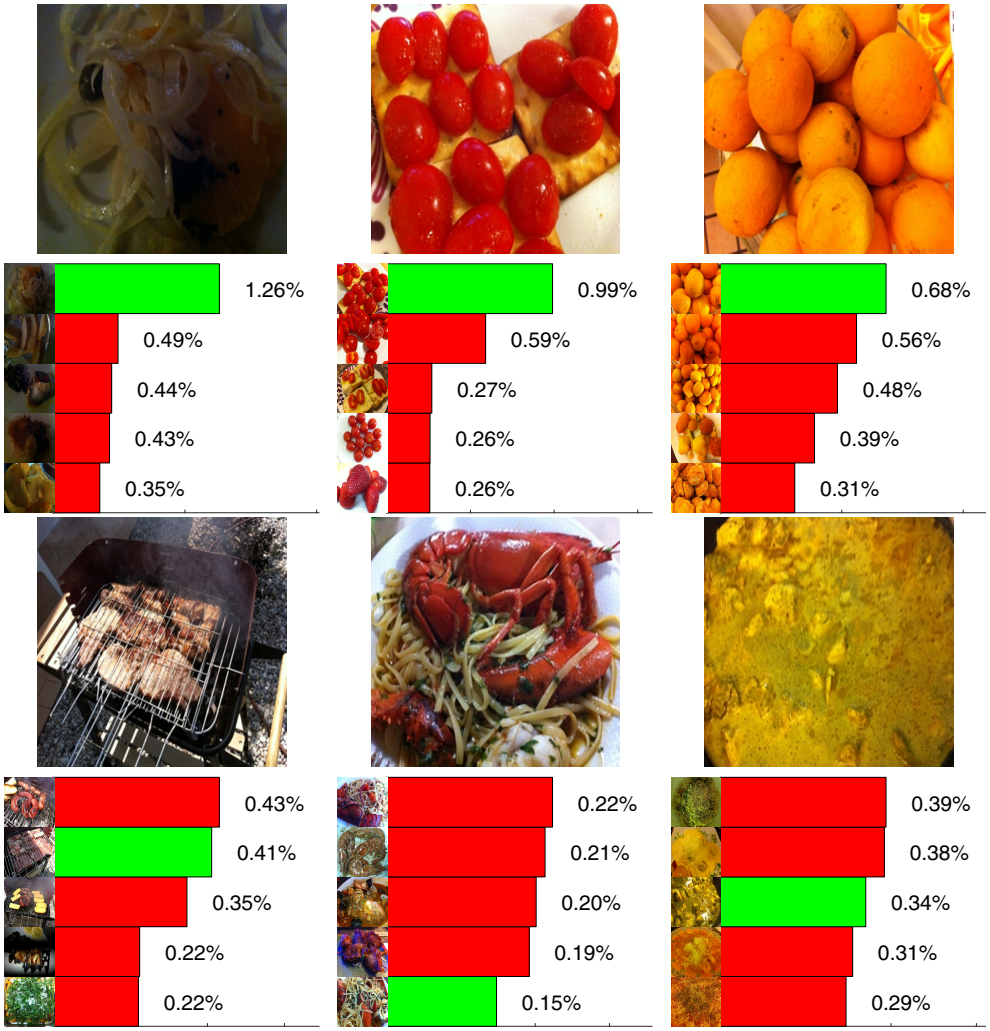



Figure 3.14: Performances achieved by the proposed method on the UNICT-FD889 dataset are shown for 6 query images (organized in two rows). At the bottom of each of those, the bar histograms show the score (in percentage) of the proposed approach for the true match (in green) and for the remaining top 4 matches (in red). On the y-axis of each bar histogram a randomly selected training image corresponding to the food class is depicted. *(Best viewed in color)*

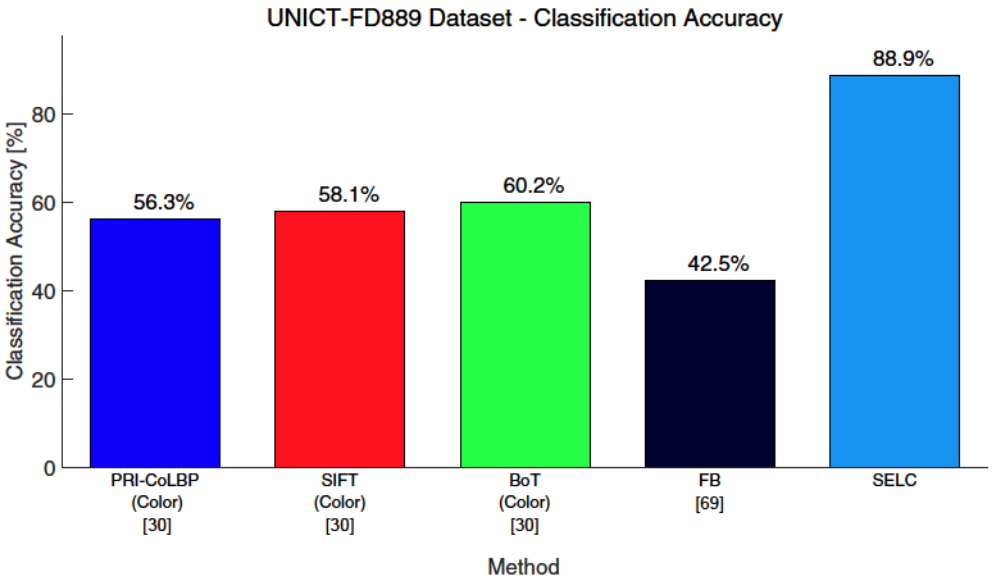



Figure 3.15: Comparisons with state-of-the-art methods on the UNICT-FD889 dataset. Results are shown in terms of classification accuracy.

**State-of-the-art Comparisons:** In Figure 3.15 the performance of the proposed SELC approach is compared to the state-of-the-art ones given in [32, 114]. The depicted results demonstrate that the proposed method strongly outperforms the existing ones by improving the best previous performance by more than 28%. In particular, the PRI-CoBP [133] approach that has similar performance to SELC on the PFID dataset, is getting the second worst accuracy on the UNICT-FD889 dataset.

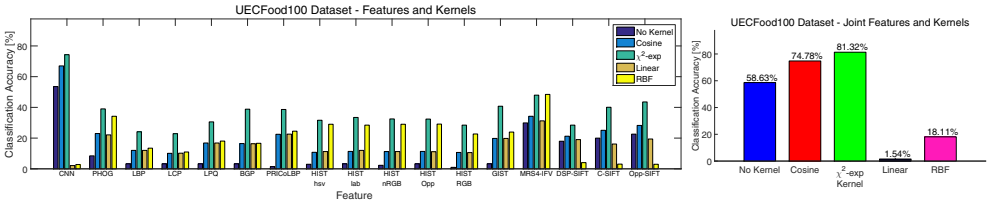
### UECFood100 Dataset

The UECFood100 Dataset [3] is one of the largest food recognition datasets [117]. This dataset contains approximately 14000 real-world food images belonging to 100 different categories. The UECFood100 dataset was built to implement a practical food recognition system [86] which was intend to be used in Japan. Because of this, it was collected in such a way that multiple food items were present in a single image, thus with the objective to perform both the detection and the recognition tasks. However, since the proposed system is designed to focus only on the recognition task, the given ground truth bounding boxes have been used to obtain a dataset of images containing single food items only (see Figure 3.16). Despite this, the same protocol in [117] has been followed to fairly compare the obtained performance with existing methods.

**Performance Analysis:** In Figure 3.17, the accuracy performance on the

[3] Available at http://foodcam.mobi/dataset100.html

Figure 3.16: 15 randomly selected samples from the UECFood100 dataset. Images of single food items were obtained by using the given ground truth bounding boxes. Columns correspond to a different type of food (i.e., to a different class). Rows show the appearance variations between samples belonging to the same class.




(a) (b)

Figure 3.17: Accuracy performances on the UECFood100 dataset obtained by single features and joint ones, both with different kernels. In ((a)) the results of single features with different kernels are depicted. In ((b)) the accuracy performances obtained by considering the joint feature space and different kernels are shown.

UECFood100 dataset are shown for single features (Figure 3.17(a)) and joint features (Figure 3.17(b)) both with different kernels.

Results in Figure 3.17(a) show the performances achieved by single features exploiting different kernels. When no kernel is used to model each single feature space, local features are well discriminating between the 100 categories. Data-driven features (i.e., CNN) yield to the best performance with a classification accuracy of about 53.54%. Color histogram features extracted from the normalized RGB color space achieve the lowest classification accuracy (i.e., 0.96%). As for the other two considered datasets, when kernels are used performances strongly improve. In particular, the CNN feature performance increases to 66.99% and to 74.30% when the cosine and the $\chi^2 - exp$ kernels are adopted, respectively. Using the same kernels, MRS4 encoded features yield to a classification accuracy of 34.21% and 47.96%. More interestingly, when the linear and the RBF kernels are used the CNN feature performance significantly drop down to 2.11% and 2.78%.

Performance achieved by considering the joint feature space and using different kernels are depicted in Figure 3.17(b). Results show that an accuracy of 58.63% is reached when no kernel is used to model the joint feature space. Performance improves by 16.15% if the cosine kernel is adopted. Such an improvement is more significant

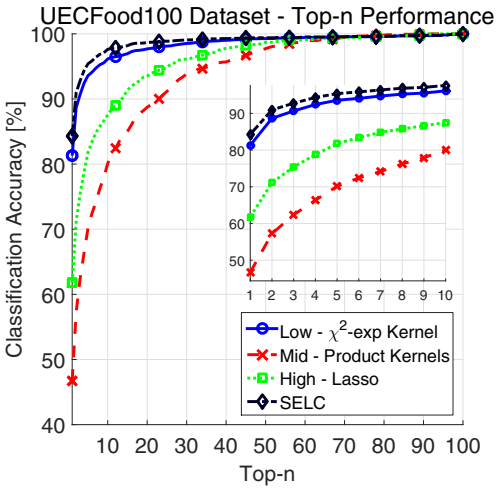



Figure 3.18: Results on the UECFood100 dataset using the proposed approach are compared to the best performing low/mid/high level fusion schemes. Performance are shown using the *top–n* criterion.

when the exponential $\chi^2$ kernel is used. Indeed, in that particular case performance increases by more than 20%.

In Figure 3.18 the *top–n* performance of the proposed SELC approach is compared to the ones achieved by the existing low/mid/high level fusion schemes that have the highest accuracy on the considered UECFood100 dataset. Results show that, for low values of $n$, the performance achieved by SELC is very close to the ones obtained by modeling the joint features space with the $\chi^2 - exp$ kernel. In particular, when $n$=1, the gap between the two approaches is of about 3%. Such a difference remains stable and reduces to 0.5% only when $n$=32. The mid and high level fusion schemes used for comparison, namely Product Kernels [40] and Lasso, have an accuracy of 46.71% and 61.80%, respectively. Thus, such methods are strongly outperformed by the proposed fusion scheme.

Qualitative performances achieved by the proposed SELC approach on the UECFood100 dataset are shown in Figure 3.19. Results are shown for 6 query images (organized in two rows). The depicted images demonstrate that the proposed approach is able to model the appearance of the 100 categories and can well generalize to even very challenging test samples (e.g., first row, 3rd query).

**State-of-the-art Comparisons:** In Figure 3.20 the performance of the proposed SELC approach is compared to the state-of-the-art ones both in terms of classification accuracy and by using the *top–n* criterion. Results are compared to the ones provided in [117] and [181]. Methods like Circle, JSEG, DCR, DPM, and Whole (all from [117]), use a detector to identify the location of the food, while GTBB [117] and PMTS [181] uses the same ground truth as SELC. The reported results demonstrate that state-of-

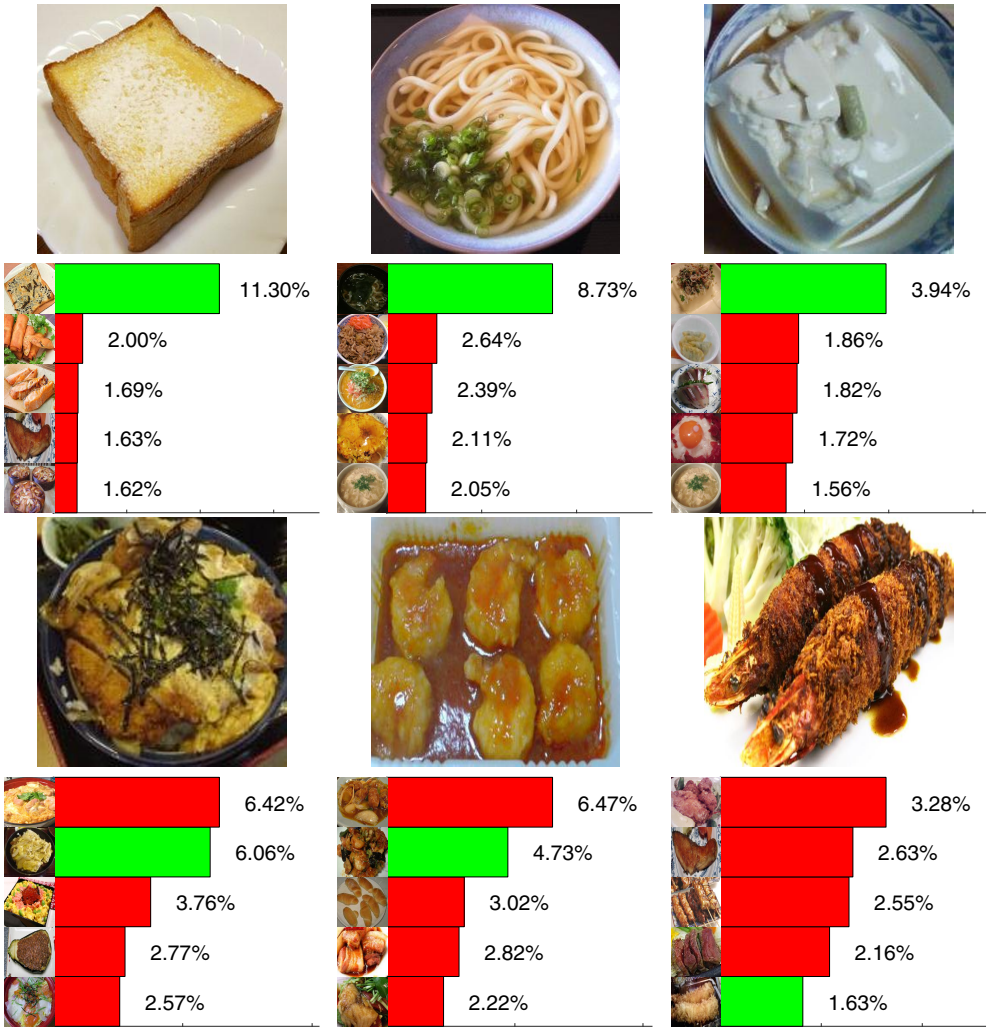
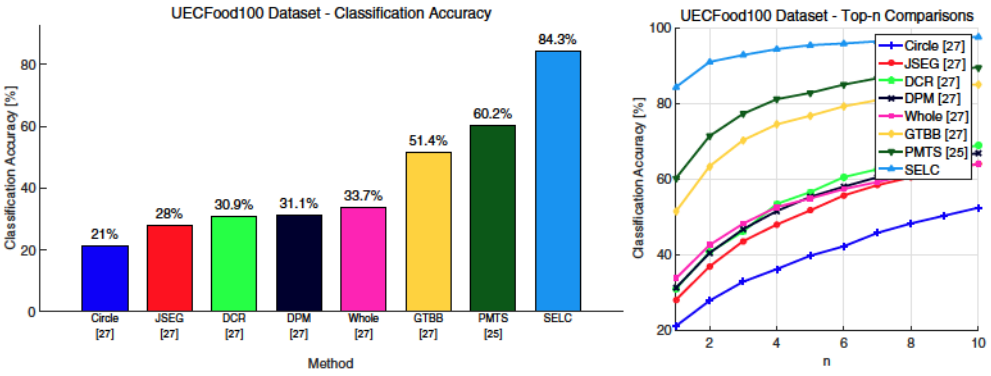



Figure 3.19: Performances achieved by the proposed method on the UECFood100 dataset are shown for 6 query images (organized in two rows). At the bottom of each of those, the bar histograms show the score (in percentage) of the proposed approach for the true match (in green) and for the remaining top 4 matches (in red). On the y-axis of each bar histogram a randomly selected training image corresponding to the food class is depicted. *(Best viewed in color)*




(a) (b)

Figure 3.20: Comparisons with state-of-the-art methods on the UECFood100 dataset. Results are shown in terms of classification accuracy.

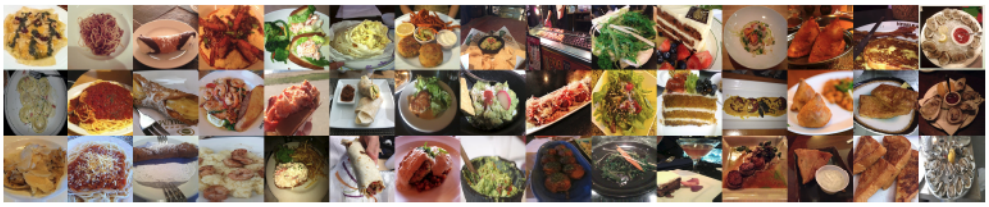


Figure 3.21: 15 randomly selected samples from the Food-101 dataset. Columns correspond to a different type of food (i.e., to a different class). Rows show the appearance variations between samples belonging to the same class.

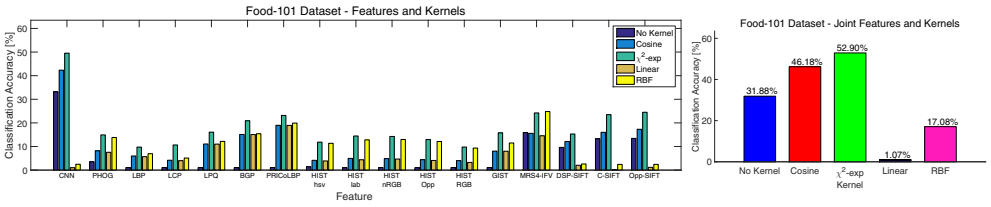
the-art performance are significantly improved from 60.2% (PMTS [181]) to 84.3%. Such a difference reduces little when $n$ increases. When $n$=5 the proposed method is the only one that achieves a classification accuracy of more than 95%.

To conclude, while results of other approaches that use the detector are not directly comparable, we can hypothesize that since GTBB uses the same features and learning algorithm as those to perform the classification, it is plausible to assume that SELC outperforms such methods as well if the same detector is used.

### Food-101 Dataset

The Food-101 Dataset[4] is the largest food recognition dataset [16]. It has been collected by downloading images from foodspotting.com The top 101 most popular and consistently named dishes were selected. Then, for each category 750 training and 250 test images were collected and manually cleaned. On purpose, the intense colors

[4]Available at http://www.vision.ee.ethz.ch/datasets/food-101




(a) (b)

Figure 3.22: Feature and kernel performance analysis on the Food-101 dataset. In ((a)) classification accuracies achieved by using single features with different kernels are shown. In ((b)) the accuracy performances obtained by considering the joint feature space (low-level fusion scheme) and different kernels are depicted.

and sometimes wrong labels included in the training images were not cleaned. As a result the dataset contains 101'000 real-world food images (see Figure 3.21).

The same splits introduced in [16] have been used to compute all the following results.

**Performance Analysis:** Results in Figure 3.22 show the accuracy performances on the Food-101 dataset of single features and joint features (i.e., low-level fusion scheme) with different kernels.

Results, in Figure 3.22(a) show that the performances of single features are similar to the ones obtained for the other datasets with CNN ones largely dominating the others. Such features achieve a 49.54% accuracy when the $\chi^2 - exp$ kernel is used. The second runner up is the MRS4-IFV feature with an accuracy that is less than half of the aforementioned one (i.e., 24.80%). Regardless the considered kernel, color histograms, LBP and LCP barely achieve an accuracy higher than 15%. Thus, these are the worst performing ones for such a dataset.

When all such features are jointly considered (see Figure 3.22(b)) the performances improve and an accuracy of 52.90% is obtained using the $\chi^2 - exp$ kernel. In all the other cases, the performances considerably reduce, even with respect to single features. In particular, when the RBF and linear kernels are used the accuracy never gets higher than 20%. Thus, showing that CNN and MRS4-IFV features used alone yield to better performance.

In Figure 3.23, the *top–n* performance achieved by SELC is compared to the ones obtained by using different fusion schemes. Specifically, the results are shown for the low/mid/high level fusion schemes that have the highest accuracy on this dataset. Results show that the proposed approach obtains the highest accuracy (i.e., 55.89%) and it also yields to the best ranking with respect to other methods. More interestingly, the best high fusion scheme, i.e., Lasso, has the worst performance both in terms of accuracy and ranking.

**State-of-the-art Comparisons:** In Table 3.2, the accuracy performance of

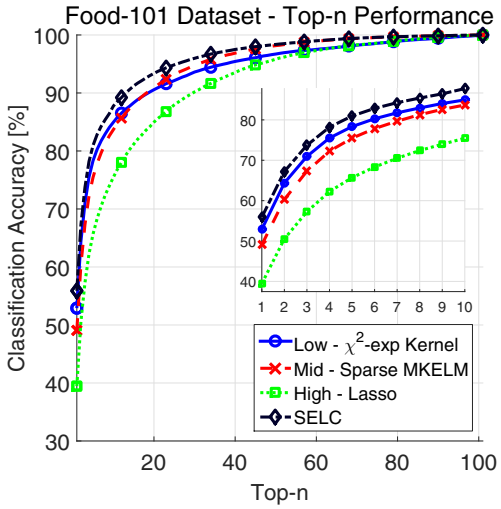



Figure 3.23: *Top–n* performances on the Food-101 dataset. The results achieved by the proposed fusion scheme are compared to the ones obtained by using the best low/mid/high ones.

SELC is compared to the ones obtained by existing methods on the Food-101 dataset. Results show that, with an accuracy of 56.40%, the best performing approach on such dataset is achieved by employing a convolutional neural network which has been trained using the AlexNet architecture. Such results reflects the performance achieved by our single CNN features which achieves an accuracy of 49.54%. However, notice that in such a case the adopted OverFeat [153] has been trained on natural images and not on this dataset specific samples. Despite this, the performance of SELC is only 0.51% less than the best existing one. All the other approaches are significantly outperformed. In particular, the performance of the very recent RFDC [16] work is improved by more than 5%.

### Computational Performance

To show the computational performance of the proposed approach, we have computed the results in Table 3.3 and Figure 3.24.

Results in Table 3.3 report on the classification accuracy and the processing times required to run a non optimized MATLAB implementation of the proposed approach on a Intel Xeon E5-v2660 machine equipped with 256GB of RAM. The results are shown for all the datasets and for the different fusion schemes. The processing reported for PFID training has been average over all the three trials.

Results demonstrate that the low level fusion schemes are less demanding in terms of computational times. In particular, when no kernel is exploited, only 1 second is

Table 3.2: Accuracy performance achieved by state-of-the-art methods on the Food-101 dataset. Best results is highlighted in boldface font.

| Method | Accuracy [%] |
|---|---|
| HoG [16] | 8.85 |
| SURF BoW-1024 [16] | 33.47 |
| SURF BoW-1024 + Color BoW-256 [16] | 38.83 |
| SURF IFV-64 [16] | 44.79 |
| Color IFV-64 [16] | 14.24 |
| SURF IFV-64 + Color Bow-64 [16] | 49.40 |
| BoW [16] | 28.51 |
| IFV [16] | 38.88 |
| AlexNet-CNN [16] | **56.40** |
| RF [16] | 37.72 |
| RCF [16] | 28.46 |
| MLDS [16] | 42.63 |
| RFDC [16] | 50.76 |
| SELC | 55.89 |

required for training. When kernels are introduced the processing time increases and reaches a maximum of about 16 seconds ($\chi^2 - exp$ kernel).

Mid and high level fusion schemes have similar performances both in terms of accuracy as well as in processing times. Specifically, it should be noticed that the SELC approach requires more processing time than Average and Product Kernels [40], and the Average high fusion methods only. These only require a sum or a product over kernels or committee answers. Thus, the proposed fusion scheme not only produces the best accuracy but it is also competitive in terms of computational performance.

Finally, it is a matter of fact that nowadays, food recognition algorithms are very attractive for mobile devices. As regards a possible deployment of the SELC approach on such devices, we can state the following. As shown in Table 3.3, the kernel computation is computationally demanding, especially if the training set is very large. On the contrary, the classification and fusion operations can be performed in fractions of a second. Despite this, the proposed approach uses many different features which requires more than a couple of second to be extracted.

To verify if all the proposed features are necessary for a correct classification we have performed the following experiment on the Food-101 dataset. We have run the proposed approach by subsequently eliminating a single committee member output from the recognition phase (i.e., the fusion weight assigned to the feature has been zeroed). The process is conduced by eliminating the features following an ordering given by the sorted (ascending) supervisor weights. Thus, the feature assigned with the lowest weight is eliminated first. Results in Figure 3.24 show that by eliminating 12 features out of 17, the performance decreases by about 1%. If two more features are

Table 3.3: Computational and accuracy performances [%] of the proposed method. The PFID training time performances [s] are shown in the 7th column. The given values have been averaged over the three considered trials. The last column shows the time required to classify a single image (averaged over all the datasets). Best results are highlighted in boldface font.

| | | PFID | PFID7 | UNICT-FD889 | UECFood100 | Food-101 | PFID Training Time [s] | Average Test Time [s] |
|---|---|---|---|---|---|---|---|---|
| Low | No Kernel | 14.59 | 33.22 | 5.15 | 58.63 | 31.88 | **0.99** | **0.01** |
| | Cosine | 50.75 | 85.22 | 65.15 | 74.78 | 46.18 | 2.97 | 0.08 |
| | $\chi^2 - exp$ | 46.37 | 81.22 | 68.95 | 81.32 | 52.90 | 15.48 | 0.29 |
| | Linear | 48.10 | 85.86 | 57.02 | 1.54 | 1.07 | 2.69 | 0.07 |
| | RBF | 19.13 | 18.33 | 7.17 | 18.11 | 17.08 | 2.91 | 0.07 |
| Mid | Average Kernels [40] | 38.36 | 82.67 | 59.74 | 11.06 | 37.26 | 68.75 | 1.67 |
| | Product Kernels [40] | 36.99 | 78.03 | 71.77 | 46.71 | 38.61 | 68.07 | 1.64 |
| | Sparse MKELM [108] | 35.63 | 80.31 | 57.33 | 10.96 | 49.07 | 74.82 | 1.71 |
| | Non-Sparse MKELM [108] | 35.81 | 80.49 | 57.44 | 12.25 | 39.88 | 74.65 | 1.73 |
| High | Average | 48.38 | 78.76 | 85.24 | 52.40 | 31.09 | 69.11 | 1.74 |
| | Lasso | 35.54 | 80.31 | 57.29 | 61.80 | 39.46 | 72.86 | 1.76 |
| | Logistic-Regression | 38.47 | 68.94 | 55.86 | 51.71 | 32.35 | 102.40 | 1.77 |
| | SELC | **53.73** | **90.89** | **88.85** | **84.31** | **55.89** | 70.57 | 1.76 |

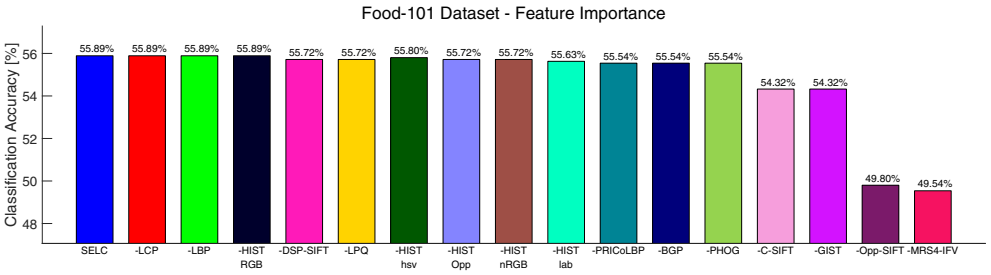



Figure 3.24: Accuracy performance achieved by the proposed approach with subsequent elimination of a committee member. First bar shows the SELC approach performance. Following ones show the results obtained by eliminating a particular feature and all the other ones appearing on its left, from the test phase. The last column shows the accuracy performance obtained by eliminating all the features other than CNN ones.

removed, such a reduction is more evident and the accuracy goes below 50%. Results demonstrate that only few features are very discriminative and the supervisor is able to capture such information. Thus, to reduce the computational complexity, features that are assigned a low fusion weight can be excluded from the test phase without significant performance loss.

To summarize, the proposed method has significant complexity both in terms of memory and time for the computation of the considered features. However, this additional computational burden is justified by the better performances that are obtained with respect to other fusion schemes and approaches in the literature.

### 3.3.3 Discussion

On the basis of the results obtained for the four considered datasets we can state the following considerations.

**Features:**
Results on single features have shown a large inconsistency across the different datasets. For instance, color histogram features yield to excellent performance on the UNICT-FD889 dataset while they perform very poorly on the UECFood100 and Food-101. The opposite occurs when CNN features are considered. In addition, despite a common belief, texture features generally yield to unsatisfying results over all the datasets. Such a behavior is largely driven by the intrinsic challenges of each specific dataset.

More specifically, let us consider the UNICT-FD889 dataset. Images of a same category are acquired by taking picture of the same dish under different conditions (e.g. by rotating the plate, zooming in, etc.). Since the considered food images are highly textured and generally present similar gradients, features considering such information (local features included) tend to perform poorly, especially if color information is not considered. Results for the other datasets follow similar motivations.

**Kernels:**
Results have shown that using a kernel function instead of computing a random mapping between the input and hidden ELM neurons has significant benefits in terms of classification accuracy. In particular, for low values of $n$ the choice of an appropriate kernel matters and can significantly affect the performance. However, different kernels yield to different improvements which also depend on the type of considered feature. Such difference in the results is driven by the specific kernel computation and its parameters.

For instance, when computing the RBF kernel we have set the free parameter to 1 (optimal for all features/datasets on average). However, since this controls the radius of influence of each sample and depends on the magnitude of the considered feature components, it is reasonable that for some specific features the kernel computation yields to a new feature space which is highly separable into the specific food categories.

A similar reasoning could be extended to the joint feature cases. Despite this,

it should be noticed that the cosine and the $\chi^2 - exp$ kernels generally improve the recognition performances.

**Supervisor:**
The deep comparison with other low/mid/high-level fusion scheme has shown that for every dataset the proposed supervisor yields to better performance than other approaches. This substantiate the benefits of the proposed committee-based approach. More specifically, results have demonstrated that the Structural SVM is able to correctly capture the feature importance and can exploit this information to produce better results both in terms of classification accuracy as well as in terms of ranking performance. Moreover, it has negligible impact on the computational burden.

**Overall Performance:**
The results obtained conducting an extensive analysis and the comparisons with state-of-the-art approaches have shown that, regardless of the considered dataset, the proposed SELC approach can be successfully applied for food recognition purposes. It also scales very well to real-world widely different scenarios. Finally, the computational analysis and the feature importance evaluation have shown that SELC can be easily extended for mobile devices usage.

## 3.4 Conclusion

In this chapter, a system for image recognition based on a learning committee has been introduced. The committee-based approach has been conceived with the idea that existing ad-hoc image representations based on *a priori* knowledge of the problem might not be sufficient to correctly handle the task. Therefore, a system that uses as many different features as possible but exploits only a subset of those to perform the image classification task has been proposed. The approach has been named Supervised Extreme Learning Committee (SELC). In SELC, each ELM is presented a particular feature type only, hence it highly specializes on classifying images by using a certain feature type. The classification results obtained by the committee members are later fused into a single output by means of a Structural SVM. This produces an optimal plausibility rank.

To show the benefits of the proposed SELC approach extensive evaluations on four benchmark datasets have been conducted. These demonstrated that SELC has superior performance to the single members taken separately, as well as to other existing fusion schemes. Comparisons with existing methods have shown that SELC is able to outperform the state-of-the-art results on all the considered datasets.

# 4

# Going Deeper and Wider with Wide-Slice Residual Networks

*The recent success of DNN for image classification tasks generated a enormous amount of research which generally exploited off-the shelf deep architectures to classify images. The chapter grounds on the idea that better results can be obtained if the deep architecture is defined with respect to an analysis of the image composition. Following such an intuition, a new deep scheme is designed to handle the image structure. Extensive evaluations on three benchmark datasets demonstrate that our solution shows better performance with respect to existing approaches.*

## 4.1 Introduction

The recent advent of deep learning technologies has achieved successes in many visual perception tasks such as object and action recognition, image segmentation, visual question answering, etc. [158, 166, 54, 34, 26, 189, 195, 56, 64]. Yet the status quo of computer vision and pattern recognition is still far from matching human capabilities especially when it comes to classify an image whose intra-category appearance might present more differences than its inter-category counterparts.

Specifically, in the previous chapters we have seen that the image recognition problem can be addressed by exploiting hand-crafted image representations based on *a priori* knowledge of the problem (e.g., [117, 33, 133]). Such a solution yielded to encouraging results (e.g., [33, 16, 115] obtained considering combinations of different features (e.g., color, shape, spatial relationships, etc.).

The solutions included in the aforementioned categories consider that the manually designed feature representation are optimal for the specifically addressed tasks. While this is appealing, it might not be the case and it also contrast the neuroscience discoveries which showed that the human brain is organized in a hierarchical

Figure 4.1: The food recognition problem is characterized by severe intra-class variations. However, some dishes have a particular structure which has not been considered.

fashion with large circuit modularity and substantial reuse of general sub-circuits. This feature might indicate that in the hierarchical brain model, initial processing layers act as feature detectors producing information that is general, yet exploitable in many specific classification problems. Then, by going deeper in the hierarchical architecture, more task-specific information is captured. It has been showed that such a feature can be included in a hierarchical learning architecture by aggregating simple features into more and more complex patterns as the structure becomes deeper.

Following such results, there have been a surge of effort in investigating the possibility of learning specific image representation (e.g., [87, 107, 48, 21, 125]). However, such approaches generally obtained better performance thanks to a mere application of off-the-shelf DNN solutions to the problem. Thus, existing approaches neglected the design of a proper architecture which considers the specific problem challenges. This motivates the development of a novel architecture that is defined following an analysis of the image composition.

Specifically, in this chapter we introduce a novel WIde-Slice Residual Network (WISeR) that has been designed to capture the structural information that is present in some food dishes (i.e., images). Our key intuition is that, regardless the exploited ingredients and the final presentation, many dishes are largely characterized by vertical food layers. For instance, 15% of the whole data in the Food-101 benchmark dataset is represented by food images that have such traits (see Figure 4.1 for few examples). Thus, we propose to leverage such a vertical structure to introduce the *slice convolution layer*. However, since not all the food dishes present such a structure, we also exploit a large residual learning architecture to obtain a generic food representation. This, together with the representation obtained from the slice convolution, is enclosed in single architecture to emit the food classification.

**Contributions::** Concretely, our contributions are: (i) We propose a novel convolutional layer that captures the vertical structure of food dishes; (ii) By combining the features detected through such layer with a stack of residual learning blocks we obtain a good representation for food dishes which do not show a specific structure;

(iii) Significantly enlarge the number of feature maps per convolution layer to tackle the diminishing feature reuse issue in deep residual networks [162], thus to improve the representational power of the learned feature detectors.

Results on three benchmark datasets show that, by combining such three ingredients together, our approach performs better than existing works in the field.

## 4.2 Wide-Slice Residual Networks

Our goal is to take a single-shot of an image and output the corresponding food category. The proposed model aims to achieve such an objective by combining a slice convolution layer with residual learning.

### 4.2.1 Architecture

As shown in Figure 4.2, the model consists of a single deep network with two main branches: a residual network branch (Sec. 4.2.2), and a slice network branch with a slice convolutional layer (Sec. 4.2.3). The residual network encodes generic visual representations of food images. The slice network specifically captures the vertical food layers. Features extracted from the two branches are then concatenated and fed to the fully connected layers that emit a classification prediction.

### 4.2.2 Residual Network Branch

#### Residual Learning

Since the breakthrough paper on extremely deep neural networks first appeared in [52] and later published in [54] –which won the ILSVRC and MSCOCO 2015 competitions, a surge of effort has been dedicated on exploring residual learning in such architectures. The idea behind residual learning is very simple yet has been shown to be extremely effective [54] in solving optimization issues that affects the process of learning the parameters of very deep neural networks (e.g., with more than 20 layers).

Everything starts from the assumption that given an input $\mathbf{x}$, a shallow network with few stacked non-linear layers can approximate a mapping function $\mathcal{M}(\mathbf{x})$. On the basis of such an assumption, it is reasonable to hypothesize that a network with the same structure can approximate the residual function $\mathcal{F}(\mathbf{x}) = \mathcal{M}(\mathbf{x}) - \mathbf{x}$ (given that the input and output have the same dimensionality). While, either learning an approximation of the mapping function $\mathcal{M}(\mathbf{x})$ or the residual function $\mathcal{F}(\mathbf{x})$ is feasible, the ease of such a process is significantly different. Indeed, as demonstrated in [54], deep networks trained to approximate the mapping function $\mathcal{M}(\mathbf{x})$ suffer from a degradation that does not appear in networks trained on approximating the residual function $\mathcal{F}(\mathbf{x})$. This opens to the success of residual learning for very deep networks.

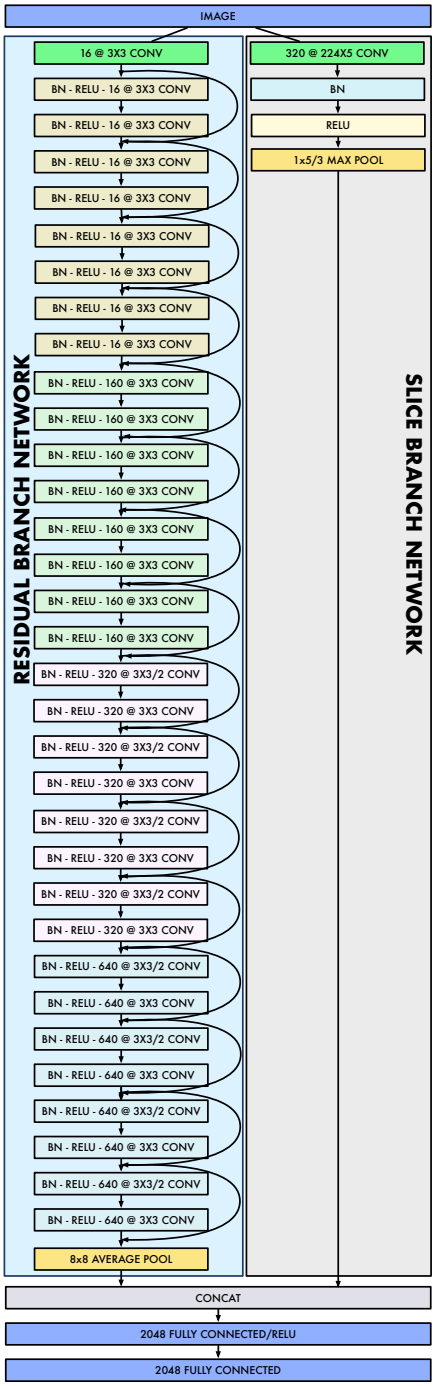
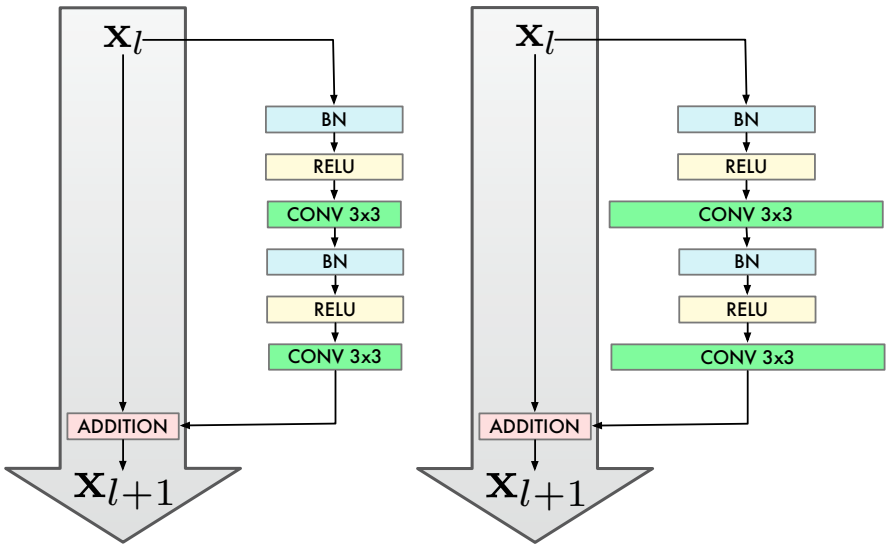



Figure 4.2: Proposed WISeR architecture consisting of two branches: a residual network branch (Sec.4.2.2), and a slice branch network with slice convolutional layers (Sec.4.2.3). The output of the two branches in fused via concatenation, then fed to the two fully connected layers to emit the food classification prediction.




(a) (b)

Figure 4.3: Graphical representation of (a) Basic Residual Blocks and (b) Wide Residual Blocks. By expanding the number of convolution kernels (i.e., widening), the number of parameters to learn increases, hence the networks has more capacity.

### Wide Residual Blocks

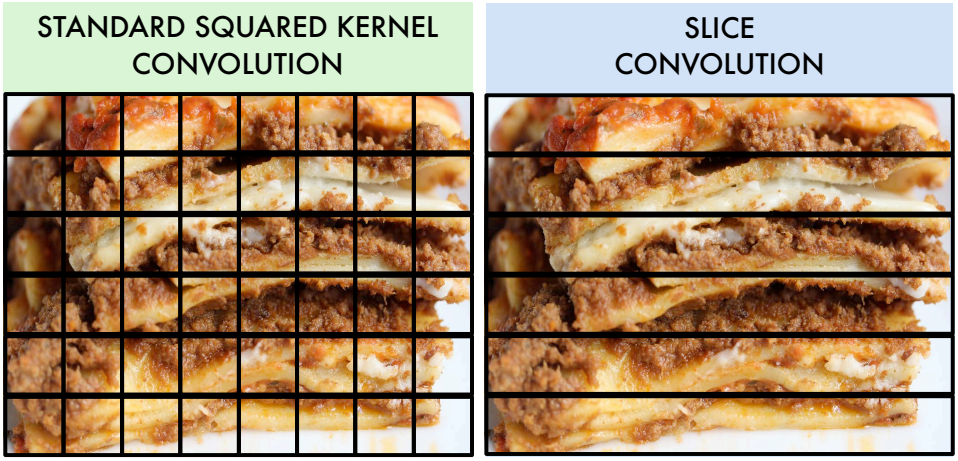
Following the methodology in [54], we exploit residual learning every few stacked layers. Formally, we let a residual block with identity mapping [55] be represented as

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l) \tag{4.1}$$

where $\mathbf{x}_l$, $\mathbf{x}_{l+1}$ and $\mathcal{W}_l = \{\mathbf{W}_{l,k} | k = 1, \dots, K\}$ represent the input, the output and the set of parameters associated with the $l$-th residual block, respectively. $K$ denotes the number of layers in a residual block ($K = 2$, in our case). The residual learning objective is to find the parameters $\mathcal{W}_l$ that best approximate the function $\mathcal{F}(\mathbf{x}_l, \mathcal{W}_l)$.

Before going further it is important to emphasize a few relevant ingredients regarding eq.(4.1):

i) neither extra parameter nor computation complexity is introduced (except for the negligible addition performed on feature maps, channel by channel);
ii) the function $\mathcal{F}(\mathbf{x}_l, \mathcal{W}_l)$ is very flexible and can represent both fully connected and convolutional layers;
iii) if the dimensionalities of $\mathbf{x}_l$ and $\mathcal{F}(\mathbf{x}_l, \mathcal{W}_l)$ are different (e.g., when varying the number of feature maps), a linear projection $\mathbf{P}$ can be exploited by the shortcut connections to match the dimensions (i.e., eq.(4.1) becomes $\mathbf{x}_{l+1} =$




(a) (b)

Figure 4.4: (a) Standard squared convolutional kernel commonly used in deep learning architectures for food recognition. (b) Proposed slice convolutional kernel aiming to capture the vertical layer structure of some food dishes.

$$\mathbf{P}\mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l)).$$

Armed with eq.(4.1), as shown in Figure 4.3, we followed the recommendations in [55] and adopted the batch normalization (`BN`) and ReLU (`ReLU`) layers as "pre-activations" for the convolutional layers (`Conv`). Then, to increase the representational power of a residual block we shared the same idea as [186] and widen the convolutional layers by significantly increasing the number of feature maps. This has been shown to be able to tackle the diminishing feature reuse problem [162] and improve performance of residual networks compared to increasing their depth.

### 4.2.3 Slice Network Branch

#### Slice Convolution

Common deep learning architectures (e.g., [158, 166, 54, 55]) exploit squared kernels to detect and extract relevant image features (see Figure 4.4(a)). Same occurs when such architectures are applied to the food recognition task (e.g., [87, 107, 48, 21]). By doing this, existing approaches do not directly consider the vertical traits of some food dishes. We believe that, while such a vertical structure can be captured via the deep hierarchy, a specific food layer detector can be extremely useful when it comes to classify food dishes that present such a peculiarity.

For such a purpose, we propose to exploit a slice convolution (see Figure 4.4(b)). It will learn the parameters of a convolution kernel that has the same width as the

(a) (b) (c)

Figure 4.5: Five different food dishes appearing in the (a) UECFood100, (b) UECFood256, and (c) Food-101 datasets. The three rows highlight the strong intra-class variations for the same food dish. *(Best viewed in color)*

input image. In such a way, it will act as a vertical layer feature detector.

#### Slice Pooling

For a specific vertically structured food category, it is not guaranteed that the vertical layers appears in the same position. Thus, the output of the slice convolution might be different depending on the location of such vertical traits. To tackle this issue we perform max pooling on vertically elongated windows. As a result we expect that a specific food layer is detected within a certain vertical location.

## 4.3 Experimental Results

First, we describe the selected datasets and the evaluation protocol. This is followed by a discussion of experimental and design selections. Then, we present the comparisons with existing methods to demonstrate the superior performance of WISeR, followed by concluding remarks.

### 4.3.1 Datasets

To validate the proposed WISeR approach, results on three benchmark datasets for food recognition have been computed. These have been selected on the basis of the different challenges they carry. See Figure 4.5 for a few samples.

**UECFood100**[1]**:** The UECFood100 dataset [117] contains 100 different food categories for a total of approximately 14'000 images. Images acquired by mobile cameras contain the most popular Japanese food dishes. Since the dataset has been conceived to address a real-world challenge, the acquired pictures may contain more than a single food dish. Therefore, this dataset is useful to understand if the approach

[1] http://foodcam.mobi/dataset100

is able perform food localization before classification, hence if it focuses on the relevant image details.

**UECFood256**[2]**:** The UECFood256 [88, 89] is a newly-constructed food image dataset, which has been build on the idea that number of food categories in existing datasets is not enough for practical use. Authors exploited knowledge on food of other countries and leveraged on existing categories to extend the UECFood100 dataset. The so obtained dataset contains 256 different foods dishes which are represented in about 32'000 images. As for the UECFood100 dataset, multiple food dishes can appear in a same image. With this dataset we aim to evaluate our approach on classifying a large number of challenging classes.

**Food-101**[3]**:** The food-101 dataset [16] consists of real-pictures of the 101 most popular dishes that appeared on foodspotting.com. On purpose, the images have not been selected and checked by human operator, hence the training set contains 75'750 images with intense colors and sometimes wrong labels. Additionally, 250 test images have been collected for each class, and have been manually cleaned. The dataset has a total of 101'000 real world images, including very diverse but also visually and semantically similar food classes. This allows us to validate our approach on a large dataset build with weakly labeled data.

### 4.3.2 Evaluation Protocol

Evaluation of food recognition approaches (e.g., [117, 86, 33, 11]) is generally performed by showing the *Top-1* recognition accuracy. In addition to that, we also report on the *top-5* criterion as generally considered when providing the results achieved by deep neural networks.

For the UECFood256 and Food-101 datasets, we used the provided splits. Since the UECFood100 dataset does not come with such a feature, we evaluated the performance of our approach using the same protocol in [87, 48], hence randomly partitioned the dataset into two subsets using 80% of the images for training and the rest for testing.

Notice that, the performance achieved by the existing methods have been taken from the corresponding works or have been directly provided by the authors.

### 4.3.3 Experimental and Implementation Settings

Existing food recognition deep net-based approaches tweak the network hyperparameters to the specific dataset (e.g., [87, 48]). In our evaluation, we have decided not to specifically adjust them to provide a generic framework.

Following the common recipe adopted by existing approaches [87, 21, 11], we

[2] http://foodcam.mobi/dataset256
[3] http://www.vision.ee.ethz.ch/datasets/food-101/

did not train our architecture from scratch since it required more food images than all the ones that are currently available in any dataset. We started from a WRN architecture [186] pre-trained on the ImageNet 2012 (ILSVRC2012) classification dataset [143].Then, we added the slice convolution branch and fine-tuned the whole architecture on the selected food recognition datasets. We initialized the new slice convolution units adopting the scheme proposed in [53]. To balance the learning rate of existing units and the new ones, we forced the learning rate of the latter to be $5\times$ the learning rate of the former ones.

**Data:** During the fine-tuning process we augment the number of dataset samples by taking $224\times224$ random crops from images resized such that the smaller dimension is of 256 pixels. We also exploited horizontal flipping with the scale and aspect ratio augmentation technique proposed in [166]. In addition, we applied photometric distortions [63] and the AlexNet-style color augmentation [54]. In testing, we considered the standard 10-crop testing [92].

For the UECFood100 and UECFood256 datasets we provide the results considering the whole images as well as the performances obtained with ground-truth cropped pictures. We adopted the ground-truth regions included in the publicly available versions of the two aforementioned datasets.

**Optimization:** Model training was performed via stochastic gradient descent with mini-batches containing 24 samples. The initial learning rate has been set to 0.01, then updated to 0.002 and 0.0004, after 50k and 90k iterations respectively. Momentum has been set to 0.9 and a weight decay penalty of 0.0005 had been applied to all layers. Training has been stopped after 100k iterations.

### 4.3.4 Performance Analysis

#### State-of-the-art Comparisons

In the following, the performances of our approach are compared to the state-of-the-art ones on the three considered benchmark datasets.

**UECFood100:** Table 4.1 shows the results achieved by existing methods and compares our approach with the top performer [186] on the UECFood100 leaderboard. Considering ground-truth cropped images, our architecture improves the *Top–1* performance of existing works specifically designed for food recognition (i.e., [113, 21]) by more than 7%. Such a gap reduces to about 2.5 percentage points if comparison is given with respect to [186]. The WISeR architecture is the only one that surpasses the 99% recognition accuracy at *Top–5*.

Notably, our solution shows a significant improvement over [107] (i.e., about 20%) when the considered images are not cropped to contain the ground-truth only, but exhibit more food dishes appearing at the same time.

**UECFood256:** Table 4.2 lists the best existing results available for the UECFood256 dataset. The depicted results show that our solution obtains the best

Table 4.1: *Top–1* and *Top–5* performance on the UECFood100 dataset. First 3 rows show the results achieved by using methods adopting hand-crafted features. Next 11 rows show the performance obtained by deep learning-based approaches on the ground-truth cropped images. Last 2 rows depict the results obtained considering images having more than a single food class (i.e., no ground truth is exploited). Best results is highlighted in boldface font.

| **Method** | **Top-1** | **Top-5** | **Publication** |
|---|---|---|---|
| MKL | 51.6 | 76.8 | COST2016 [107] |
| FC7 | 58.03 | 83.71 | ACMMM2016 [21] |
| SELC | 84.3 | 95.2 | CVIU2016 [113] |
| DeepFoodCam | 72.26 | 92.00 | UBICOMP2014 [87] |
| AlexNet | 75.62 | 92.43 | ACMMM2016 [21] |
| DeepFood | 76.3 | 94.6 | COST2016 [107] |
| FV+DeepFoodCam | 77.35 | 94.85 | UBICOMP2014 [87] |
| DCNN-FOOD | 78.77 | 95.15 | ICME2015 [180] |
| VGG | 81.31 | 96.72 | ACMMM2016 [21] |
| Inception V3 | 81.45 | 97.27 | ECCVW2016 [48] |
| Arch-D | 82.12 | 97.29 | ACMMM2016 [21] |
| ResNet-200 | 86.25 | 98.91 | CVPR2016 [54] |
| WRN | 86.71 | 98.92 | BMVC2016 [186] |
| WISeR | **89.58** | **99.23** | Proposed |
| DeepFood | 57.0 | 83.4 | COST2016 [107] |
| WISeR | 79.46 | 97.46 | Proposed |

performances by surpassing the 83% and 95% recognition accuracies at *Top–1* and *Top–5*, respectively.

More interesting are the performances obtained when no-ground truth is considered to locate the food dish within the image. In such a case, we outperform the previous best result and obtain the overall third best *Top–5* recognition accuracy, thus achieving better performance than recent methods which consider ground-truth cropped images (e.g., [48, 54]).

Such an outcome, together with the results shown in Table 4.1, might indicate that our proposed solution is able to focus only on the relevant portion of an image to perform the classification task. As shown in Figure 4.6, visual inspection of the obtained performance substantiate this hypothesis. It also demonstrate that the proposed solution has gained an high-level knowledge of the food dishes by giving high scores to food plates that are very similar to each other, or contain more than a single food class (e.g., $2^{\text{nd}}$ to $5^{\text{th}}$ sample). To obtain more detailed insights on such results, an analysis of the visual attention performed by the architecture has been conducted (see Sec. 4.3.5).

**Food-101:** A comparison with existing methods on the Food-101 dataset is shown in Table 4.3. Results demonstrates that our solution outperforms the best

Table 4.2: *Top–1* and *Top–5* performance on the UECFood 256 dataset. First 3 rows show the results obtained by using methods adopting hand-crafted features. Next 7 rows show the performance obtained by deep learning-based approaches on the ground-truth cropped dataset images. Last 2 rows depict the results obtained considering input images having more than a single food class (i.e., no ground truth is exploited). Best result is highlighted in boldface font.

| **Method** | **Top-1** | **Top-5** | **Publication** |
|---|---|---|---|
| RootHOG-FV | 36.46 | 58.83 | UBICOMP2014 [87] |
| Color-FV | 41.60 | 64.00 | UBICOMP2014 [87] |
| Color-FV+HOG-FV | 52.85 | 75.51 | UBICOMP2014 [87] |
| DeepFoodCam | 63.77 | 85.82 | UBICOMP2014 [87] |
| DeepFood | 63.8 | 87.2 | COST2016 [107] |
| DCNN-FOOD | 67.57 | 88.97 | ICME2015 [180] |
| Inception V3 | 76.17 | 92.58 | ECCVW2016 [48] |
| ResNet-200 | 79.12 | 93.00 | CVPR2016 [54] |
| WRN | 79.76 | 93.90 | BMVC2016 [186] |
| WISeR | **83.15** | **95.45** | Proposed |
| DeepFood | 54.7 | 81.5 | COST2016 [107] |
| WISeR | 72.71 | 93.78 | Proposed |

Table 4.3: *Top–1* and *Top–5* performance on the Food-101 dataset. First 9 rows show the results obtained by using methods adopting hand-crafted features. Last 7 rows show the performance obtained by deep learning-based approaches. Best results is highlighted in boldface font.

| **Method** | **Top-1** | **Top-5** | **Publication** |
|---|---|---|---|
| SURF BoW-1024 | 33.47 | - | ECCV2014 [16] |
| SURF IFV-64 | 44.79 | - | ECCV2014 [16] |
| BoW | 28.51 | - | ECCV2014 [16] |
| IFV | 38.88 | - | ECCV2014 [16] |
| RF | 37.72 | - | ECCV2014 [16] |
| RCF | 28.46 | - | ECCV2014 [16] |
| MLDS | 42.63 | - | ECCV2014 [16] |
| RFDC | 50.76 | - | ECCV2014 [16] |
| SELC | 55.89 | - | CVIU2016 [113] |
| AlexNet-CNN | 56.40 | - | ECCV2014 [16] |
| DCNN-FOOD | 70.41 | - | ICME2015 [180] |
| DeepFood | 77.4 | 93.7 | COST2016 [107] |
| Inception V3 | 88.28 | 96.88 | ECCVW2016 [48] |
| ResNet-200 | 88.38 | 97.85 | CVPR2016 [54] |
| WRN | 88.72 | 97.92 | BMVC2016 [186] |
| WISeR | **90.27** | **98.71** | Proposed |

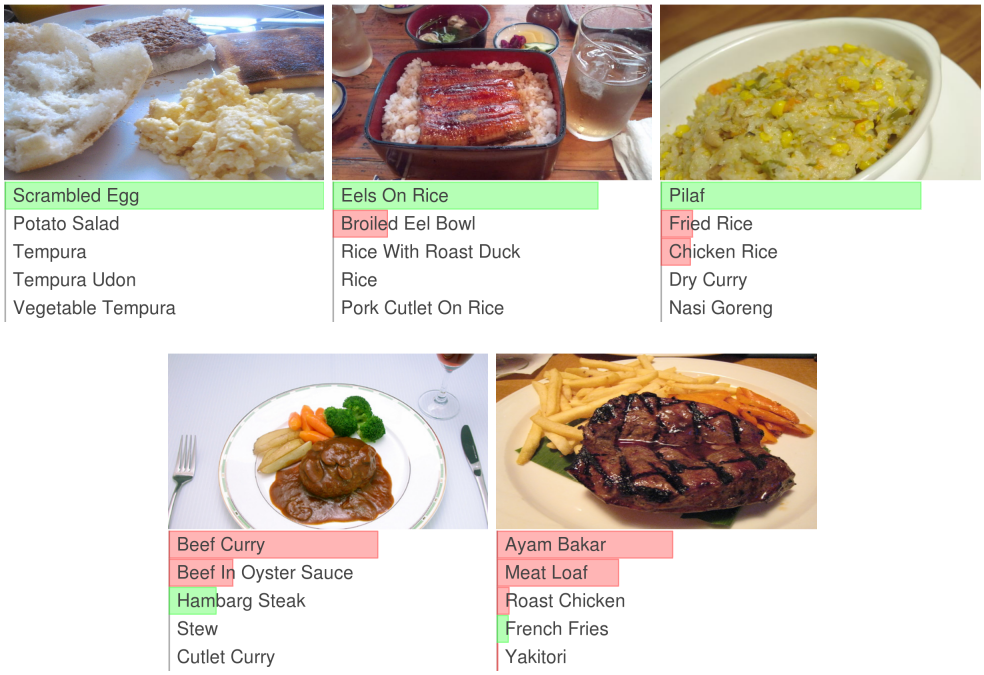



Figure 4.6: *Top–5* WISeR predictions on 5 image samples from the UECFood256 dataset (with no cropped ground-truths). Test image are shown at the top. In the bar plots, predictions are ranked from top (most likely class) to bottom (less likely class). The true match class is represented by a green bar. False matches are shown with red bars. *(Best viewed in color)*

results obtained by considering hand-crafted features [113]. The proposed architecture performs better than all the existing deep learning-based ones by achieving a *Top-1* accuracy of more than 90%. Such a result shows that our solution is able to learn good representations even from weakly labeled data. This substantiates the fact that residual learning-based networks are not suffering from the diminishing feature reuse problem [162].

### Ablation Analysis

**Branch Performance:** To better understand the source of our performance, Table 4.4 shows results for ablation experiments analyzing the contributions of the two architecture branches. Specifically: (i) *slice@WISeR* removes the residual network branch. Classification is obtained by considering only the slice convolution branch features; (ii) *residual@WISeR* performs the opposite. Food recognition is achieved via classification of residual features.

Table 4.4: *Top–1* and *Top–5* performance achieved by separately exploiting the two proposed network branches on the UECFood100, UECFood256 and Food-101 datasets. Slice@WISeR shows the results obtained using only the slice convolution branch. Residual@WISeR shows the performance achieved via the residual learning branch.

| Dataset | Model | Top-1 | Top-5 |
|---|---|---|---|
| UECFood100 | slice@WISeR | 41.72 | 66.15 |
| | residual@WISeR | 86.71 | 98.92 |
| UECFood256 | slice@WISeR | 30.56 | 57.65 |
| | residual@WISeR | 79.76 | 93.90 |
| Food-101 | slice@WISeR | 46.17 | 63.57 |
| | residual@WISeR | 88.72 | 97.92 |

Results show that for all the three datasets, the residual learning branch (i.e., *residual@WISeR*) largely outperforms the slice one. We hypothesize that the reason behind such a result is due to the following facts: (i) The residual learning branch has been pre-trained considering a very large set of natural images (i.e., ImageNet), while the weights of the slice branch are learned from scratch. Features extracted from a network trained on ImageNet –without fine-tuning– have shown to be highly discriminative *per se* for many visual recognition tasks [153, 81] (food recognition included [113]). Thus, it is reasonable to believe that such a pre-training introduces significant priors for the network weights. This is confirmed by the fact that training the whole WISeR architecture from scratch results in a recognition rate of 78.12%, 68.37%, and 79.45% on the three considered datasets, respectively. (ii) Capturing only vertical layer features excludes learning of other distinctive traits that are not specific of vertically structured food dishes.

**Model Capacity:** Improved recognition performance of WISeR over standalone WRN can be attributed to the increased model capacity on target tasks rather than slice convolution. To verify such a possibility, starting from the same ImageNet pre-trained WRN network, we exploited the transfer learning scheme proposed in [128] by including an additional 2048-units fully connected layer before the output one, then trained on the specific food dataset. Results show that using the transfer learning scheme with no slice branch, *Top1* results on all the datasets improve little over standalone WRN (about 1% on average). Thus, performances are lower than the ones obtained with the proposed solution.

**Per Category Performance:** Since the vertical pattern might not be generic, to see for which categories the slice convolution is helpful and for which categories it instead hurts we have conducted the computed the results in Figure 4.7. Results show that, on 14 out of 15 food categories presenting a vertical structure, performances are improved. Degradation on food dishes that do not present vertical traits is limited (i.e.,about 0.7% on average) and occurs only in 23 cases over the remaining 86 categories.

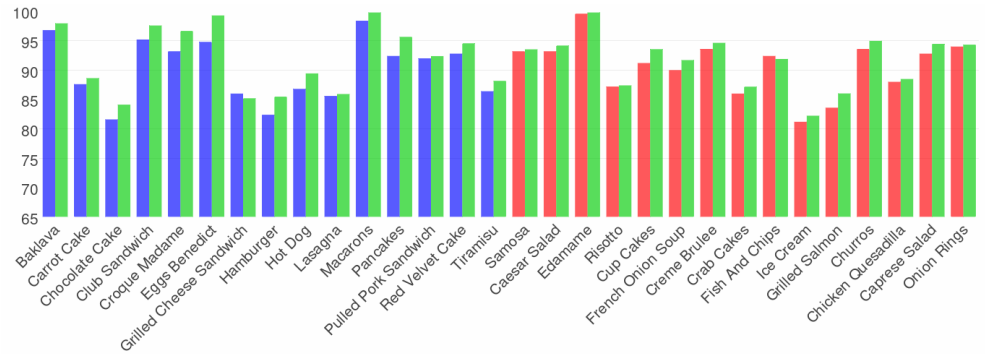



Figure 4.7: Per-category *Top–1* recognition percentage on 30 categories from the Food-101 dataset. Blue and red bars represent the results obtained by residual@WISeR on food categories that either present or not the vertical traits, respectively. The green bars represent the results obtained by the proposed WISeR approach.

### 4.3.5 Visual Attention

Results on the UECFood100 and UECFood256 datasets show that our approach can extract the useful information for classification from the relevant image regions only. To have a richer grasp on this outcome, we have conducted an analysis of the visual attention performed by the architecture. Towards this end, we have exploited the recent Grad-CAM approach [152]. It allows to obtain a coarse localization map regarding the important regions in the image which are considered for classification.

As show in Figure 4.8, when more than a single food dish is present in the image, the WISeR architecture is able to focus only on the image portion that contains the object of interest. This is substantiated by the fact that features are not extracted from other non-relevant food/non-food objects (e.g., the plate containing green leaves, the spoon, the paper glass, etc.).

### 4.3.6 Discussion

**Outcomes:** Results obtained for the three datasets demonstrate that: (i) our solution shows better performance with respect to existing approaches either based on hand-crafted features or on deep learning schemes; (ii) while the residual learning branch brings most of the classification power, by combining the so extracted features with the vertical layer traits discovered through the slice convolution branch the best achievements are attained; (iii) the proposed WISeR architecture is able to self-discover the image portion that should be considered to extract the features, hence to emit the classification.

This shows that our approach is able to well address the many non-trivial challenges in food recognition and is not designed to tackle the specific problems brought

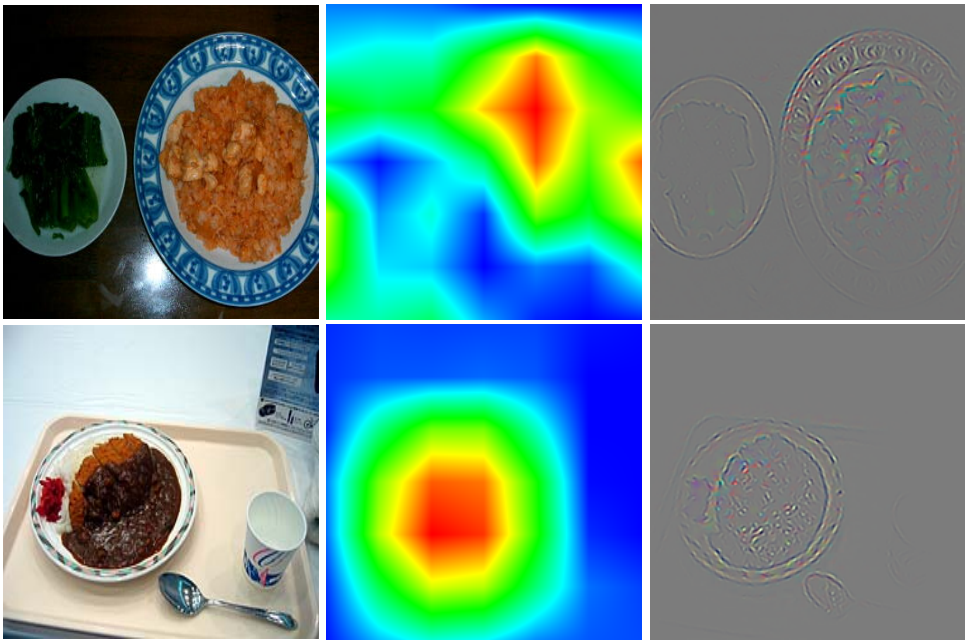


Figure 4.8: Analysis of the visual attention obtained through our architecture on two randomly selected images from the UECFood100 dataset. First column is the input image, second column shows the visual attention with a color-coded plot (blue means lower attention, red higher). Last column depicts the gradient computed with respect to the input image, showing that features are extracted only from the relevant image region. Results obtained through Guided Grad-CAM visual explanation [152]. *(Best viewed in color)*

in by a single dataset.

**Limitations:** It is a matter of fact that nowadays food recognition algorithms are very attractive for mobile platforms. Our solution requires substantial memory loads as well as significant computational efforts to process a single datum, thus denying a possible deployment of our approach on these devices. A possible solution to such a problem would be to compress the network to obtain a shallower architecture [47] or to exploit binary weights [138]. We demand this study to future works.

## 4.4 Conclusion

In this chapter, a system for automatic image recognition based on a deep learning solution specifically designed for the considered task has been proposed. The WISeR architecture combines features extracted from two main network branches. The

residual learning branch provides a deep hierarchy which is able to capture the general traits that are shared with the majority of the existing categories. The slice convolution branch captures the structural information (i.e., vertical layers) of the images which present such a peculiarity. The features extracted from these branches are fused then exploited to emit the classification.

To demonstrate the benefits of the proposed solution, evaluations on three benchmark datasets have been conducted. Comparisons with existing methods have shown that by exploiting both the architecture branches together better performance than state-of-the-art approaches are achieved regardless the considered dataset. The visual attention analysis has shown that the network is able to self-identify the relevant portions of the image that should be considered for classification.

# 5

# Extreme Deep Learning Trees: The Evolution of Neural Learning Systems

*This chapter introduces a novel hierarchical learning architecture that aims to overcome the major problem which previously presented solutions suffer from: the need to manually design the whole architecture pipeline. First, a brief review of the most common biologically-inspired learning schemes which are the basis for the proposed learning model is presented. The description of the proposed solution follows with preliminary experimental evaluations conducted on three datasets having different image classification tasks.*

## 5.1 Introduction

We, as humans, are inherently able to complete a plethora of different and complex task in fractions of a second. In particular, in every instant of our chaotic and crowded lives we are exposed to a myriad of data acquired through many different sensors, but are somehow able to catch the relevant aspects of such data in a way that allows for their future use. Most of this capability we embody comes from the robustness and the efficiency of our brain, which is one of the reasons why mimicking its behavior has been a widely explored research area in artificial intelligence.

In response to this call, previous chapters have introduces hierarchical learning architectures that ultimately aimed to emulate our own visual abilities. However, current performance still remains a long way from the goal; and, although, in the last few years, deep learning architectures have taken huge steps towards achieving such a goal, they still lack several features that limit their use.

To push the progress towards cloning human capability with innovative techniques, in this chapter we start by providing a brief overview of the mainstream brain-inspired architectures and research directions proposed over the past decade. At the outset, however, it is important to emphasize that each architecture has strengths and weaknesses, depending on the application and context in which it is being used. Artificial Neural Networks (ANNs), Neural Trees (NTs), Convolutional Neural Networks (CNNs) and Extreme Learning Machines (ELMs) (and their respective variations) are the primary focus because they are well established in the field and have just showed great promise for future work. After such an analysis a novel architecture exploiting the strengths of the current methods is proposed. Preliminary results demonstrate that it is able to achieve state-of-the-art results in a more efficient way.

## 5.2 ANNs: The Basis of Neural Learning

In the last 30 years, different hierarchical learning architectures and models have been proposed to mimic the human brain. By far, the most widely adopted is the Artificial Neural Network (ANN). An ANN introduces layers of artificial computational units (i.e., neurons) wired in such a way that they resemble the human brain neural connections.

### 5.2.1 Artificial Neural Networks

As shown in Figure 5.1, ANNs are composed of a number of basic computational units, called *neurons*. These, organized in *layers*, perform linear or nonlinear transformations of their inputs. ANN structures usually contain one *input layer*, one *output layer* and possibly one or more *hidden layers*. The neurons of two adjacent layers are connected by synaptic *weights*. Every neuron computes a *transfer function*, e.g. a weighted sum of the outputs generated by the previous layer, then applies an *activation function*. The outputs are then connected to the next layer. The procedure ends when the output layer is reached.

An ANN without a hidden layer is a linear discriminant method known as single-layer feedforward neural network (SLFN). If nonlinearity is intrinsic in the classification problem, one or more hidden layers should be considered to define a multilayer ANN (e.g., a Multilayer perceptron (MLP) [142]).

ANNs can be classified into two classes depending on how the neurons are connected. This connection characterizes the flow of the input information and gives rise to two main kinds of network: feedforward ANNs and recurrent ANNs. In *feedforward neural networks*, input information flows only in one direction from one layer to another and there is no feedback loop in any other layer of the network. In *recurrent neural networks* at least one feedback loop exists in the network structure.

In both the cases, the purpose of training the network is to compute the optimal weights that minimize a suitable criterion function known as *loss function* or *cost*

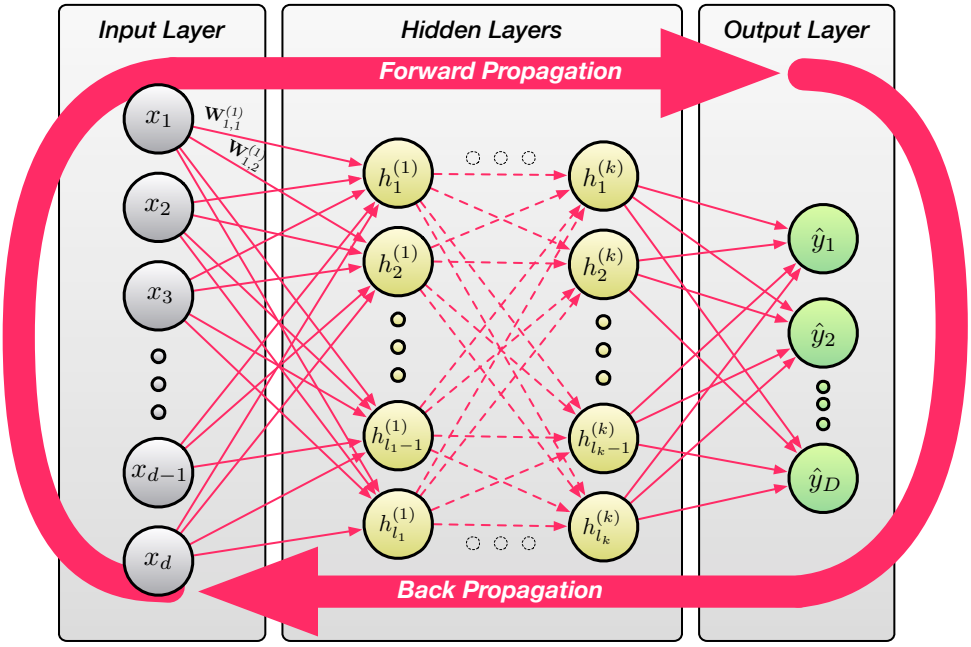



Figure 5.1: An ANN with $k$ hidden layers. The input column vector $\mathbf{x} \in \mathbb{R}^d$ represents the input layer. The hidden nodes at hidden layer $i$ compute the transfer function $h_j^{(i)} = \mathbf{x}^T\mathbf{W} + \mathbf{b}$ for all $j = 1, \cdots, l_i$, where $l_i$ is the number of hidden nodes at layer $i$, $\mathbf{W} \in \mathbb{R}^{d \times l_i}$ are the connection weights and $\mathbf{b} \in \mathbb{R}^{l_i}$ is the bias term. Then, a nonlinear transformation $g(\cdot)$, known as the activation function, is applied to each transfer function to produce the output which serves as the input for the next layer. The same operations are repeated for all the layers in the network. The output of the last layer is finally input to a suitable cost/loss function which should be minimized. The error computed through such a function is used by the BP algorithm to learn the optimal weights and biases.

*function*. For classification, optimal weights should reduce misclassification [187] while correct prediction of the real-valued output is expected for regression problems [160]. The sum-of-square error and the cross entropy are the two most adopted criteria for cost functions [90].

Recently, extensions to standard ANNs were proposed by considering generative models (e.g., Deep Belief Networks (DBNs) [57] and Deep Boltzmann Machines (DBMs) [144]) and unsupervised pre-training (e.g., Stacked Auto Encoders (SAEs) [176] and Statcked Denoising Auto Encoders (SDAEs) [176]) to initialize the connection weights.

### 5.2.2 Backpropagation for Neural Learning

By far the most popular training method for multi-layer ANNs is the backpropagation (BP) algorithm [142]. It is based on two phases that are iterated till convergence. A first phase fed forward the patterns using the connection weights (which, for the first are usually randomly initialized). The output of the last layer is compared with the expected value to compute the training error. Then, such error is backpropagated, i.e. fed back, to the neural network and used to adjust the weights using a gradient descent strategy. A complete forward-backward propagation is called an *epoch*. The training stops after a certain stopping criterion is satisfied. This can be a predefined number of training epochs, cross validation [163] and early stopping [122]. Later, derivations of BP including momentum [46], Nesterov's Accelerated Gradient [165] and second order information [20] were introduced.

### 5.2.3 What are the limits of ANNs?

Due to the non convexity objective function of the ANN, the BP algorithm may get stuck in a local minimum. Therefore, several training runs with different initial weights should be performed to find a good solution. This further increases the computational effort [49].

In addition, there is no fixed a priori rule to decide, for any given problem, the architecture (i.e., number of hidden layers and nodes in each of these layers) of an ANN. This can only be done by evaluating the network performance using different architectures and selecting the one that produces the best results.

Last but not least, ANNs learn how to transform the input patterns into the desired outputs. Usually, input patterns derive from an extraction process that involves the knowledge of the problem and supposes that certain features best represent the raw data for the task. It often happens that bad performance of the ANNs is related to the wrong selection of the features adopted to represent the data rather than to the discriminative incapacity of the ANN process.

To summarize, ANNs at present have drawbacks and limitations including: (i) slow learning speed, (ii) trivial human tuned parameters, (iii) non-optimum learning algorithms, (iv) *a priori* knowledge for proper representation of the input data.

## 5.3 NTs: An Hybrid Neural Architecture

To overcome some of these limitations, novel hybrid architectures originating from decision trees (DTs) and simple perceptrons have been proposed. The main motivation for the development of this new architecture came from the search for a training algorithm able to learn the structure of the architecture rather than requiring its upfront definition.

**Algorithm 2:** NT Training

**Input :** The training set $\mathbf{X}$
**Output:** The trained NT model

Set $Q_v \leftarrow \{v_0\}$ and $Q_X \leftarrow \{\mathbf{X}\}$;
**while** $Q_v$ **do**
  $v \leftarrow \texttt{Pop}(Q_v)$ and $\mathbf{X}' \leftarrow \texttt{Pop}(Q_X)$;
  `TrainNode` $(v, \mathbf{X}')$;
  $(Q_{\hat{v}}, Q_{\hat{X}}) \leftarrow \texttt{Classify}(v, \mathbf{X}')$;
  **while** $Q_{\hat{X}}$ **do**
    **if** $\mathbf{X}^* \leftarrow \texttt{Pop}(Q_{\hat{X}})$ *is homogeneous* **then**
      $v^* \leftarrow \texttt{Pop}(Q_{\hat{v}})$ is set to leaf;
    **else**
      $\texttt{Push}(Q_v, \texttt{Pop}(Q_{\hat{v}}))$;
      $\texttt{Push}(Q_X, X^*)$;
    **end**
  **end**
**end**

where $v_0$ represents the root node, $\mathbf{X} \in \mathbb{R}^{d \times n}$ is the training set consisting of $n$ $d$-dimensional feature vectors. $\mathbf{X}' \subset \mathbf{X}$ is the local training set (LTS) at a given node $v$. Let $Q_v$ and $Q_X$ be the queues holding the nodes to train and corresponding LTSs. **TrainNode** is the procedure used to train an ANN. It eventually substitutes the ANN node with a more appropriate classification scheme (e.g., a split node). **Classify** is the procedure that, given a node $v$, classifies the related LTS and produces the list of child nodes $Q_{\hat{v}}$ together with list of corresponding patterns $Q_{\hat{X}}$. The homogeneity property of a training set is a set of rules that define the correct classification of a LTS. Finally, **Pop** and **Push** are the usual queue related procedures.

### 5.3.1 From Decision Trees to Neural Trees

The DT is one of the most popular models for machine learning because it is able to make decisions with fewer computations and provides a more comprehensible model. Literature in extending standard DTs to improve their generalization ability is countless [91], however following [192], existing works can be generally categorized depending on the in-node models: (i) *multivariate approaches* consider multiple features which are exploited by linear [3] or non-liner [191] partitioning hypersurfaces; (ii) *multinode approaches* jointly consider many nodes to make decisions (e.g., Fuzzy DT [164]); (iii) *multitree approaches* adopt ensembles methods to improve single model performance (e.g., Random Forest [17]).

When the in-node model of a DT is an ANN perceptron the obtained architecture is called neural tree (NT) [159]. The first architecture resembling an NT can be dated back to 1984, when linear threshold units were used as in-node models of a DT [18]. Even though such a tree does not exploit ANNs, it outlined the upcoming research

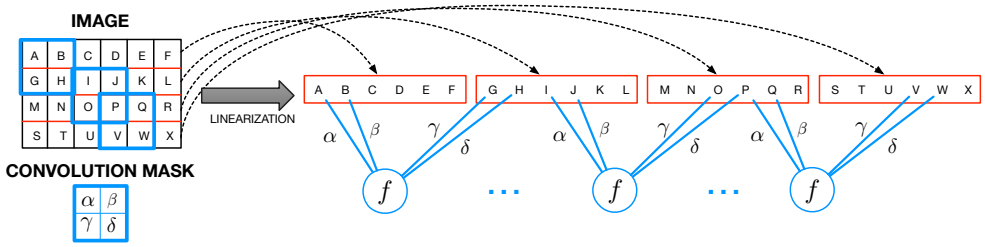



Figure 5.2: An example of convolution layer. The convolution of a 2D signal with a kernel can be defined as a particular ANN with two layers. The input neurons consist of signal values. The second layer neurons are connected only with a subset of the input ones. The connection weights from the input to the second layer (i.e., $\alpha, \beta, \gamma, \delta$) are the same for every neuron. Finally, the weighted input is processed by the function $f$, which is generally the activation function of an ANN.

on NTs.

### 5.3.2 Neural Tree

The first NT was proposed in 1988 by Utgoff [171]. Such a tree, called a perceptron tree, is composed of attribute tests as internal in-node models and perceptrons as leaves.

**Training:**

As standard NTs training algorithms, the perceptron tree training procedure generates the tree in a recursive manner by partitioning the training set. Such a procedure involves three steps: training internal nodes, then determining leaf nodes and labeling them.

For a generic NT, having heterogeneous in-node models (e.g., splits, ANNs, etc.), the training procedure can be written as in Algorithm 2. Starting from root:

i) The **TrainNode** procedure is used to train the in-node model.
ii) The training patterns are later classified by means of the **Classify** procedure, hence the training set is partitioned.
iii) If a partition satisfies the *homogeneity* condition, then a leaf node is created and labeled with the corresponding class. Under particular conditions, the in-node model can be replaced by a more appropriate one (e.g., in the perceptron tree, if the training patterns of a node are not linearly separable, the trained perceptron is substituted by a split node).
iv) If a partition is not *homogeneous*, a new child node is added to the tree. Such a node will be trained using the patterns of the associated partition.
v) The training stops when all nodes become leaves.

The differences between alternative NT schemes rely on the **TrainNode** procedure and on the homogeneity definitions. The former decides which type of in-node model has to be considered, while the latter defines the training stop criteria.

**Classification:**

The decision making procedure in the NT is governed by the **Classify** function. The class of a test pattern is obtained by traversing the tree in a top-down fashion starting from the root node. At each node the **Classify** function classifies the pattern, hence selects the children to which the pattern should be presented next. The procedure stops when the pattern reaches a leaf node representing the class.

### 5.3.3 What are the limits of NTs?

While NTs solve the problem of defining a priori the ANN architecture, they still have a number of drawbacks: (i) there is a lack of control on the depth of the tree; (ii) convergence is not guaranteed for all the schemes; (iii) training in-node models on small LTSs limit the generalization; (iv) as ANNs, NTs work on features representing the data.

While in recent years plentiful work was successfully conducted to address the aforementioned issues [121], as for ANNs, the success or failure of NTs for pattern classification still depends on the raw data representations. These are usually built by means of hand-crafted features extraction algorithms fit to the considered problem. This highlights the essence of new learning architectures in inducing discriminative raw data representation that is not task-dependent.

## 5.4 Learning Data Representations

To solve the raw data representation problem, in the last few years, several different systems able to automatically learn proper data representations have been investigated [7, 9]. However, much of the success of such new deep architectures should be awarded to a quite old [100] special kind of ANN able to capture signal spatio-temporal dependencies: the convolutional neural networks (CNNs).

To make the introduction to CNNs more intuitive, in the following, we make use of images instead of generic signals. Nevertheless, the discussion can be generalized to any type of signal.

### 5.4.1 Convolutional Neural Networks

In pattern recognition problems a vector is usually used to represent the real signal. In case of images, this vector becomes a matrix. Within such a matrix, there exist a

group of pixels that share local (spatio-temporal) properties, which CNNs are able to capture [100]. A CNN is a particular type of ANN where precise rules are defined to introduce the:

i) Convolution Layer,
ii) Local Contrast Normalization Layer,
iii) Pooling/Subsampling Layer, and
iv) Fully Connected Layer.

### The Convolution Layer

allows to discover the local properties in the data. It performs a *convolution* of the given image with a *kernel* by adopting a particular layer connection rule (see Figure 5.2). In particular, the kernel weights define the weights connecting a subset of the input neurons with a hidden one. As the network is trained, the weights of the kernel are adapted to extract the optimal local feature. However, if we use a single kernel, the layer will be highly specialized, hence its generalization capabilities are limited. To get round this, multiple different kernels are randomly initialized, thus different kernels will be learned. In other words, for a given *input feature map*, e.g. the input image, we can have a convolution layer that produce multiple *output feature maps*.

### The Local Contrast Normalization Layer

limits the effects of intensity variations over the different feature maps. Indeed, when looking at similar images, it may happen that, after the convolution operation the resulting feature maps span a different feature space. To deal with this problem a local contrast normalization (LCN) layer [74] is stacked at the output of the convolution layer. The layer, inspired by computational neuroscience models [131, 28], adopts a similar connection rule to the convolution layer. It performs local subtractive and divisive normalizations, enforcing a sort of local competition between adjacent features in a feature map, and between features at the same spatial location in different feature maps [74].

### The Pooling/Subsampling Layer

adds robustness to small shifts of the input data by looking at groups of input neurons that can either come from a single input feature map or from multiple feature maps [74]. While many different kinds of pooling layers, e.g. average pooling, L2 pooling, etc. can be found in the literature [74, 9], the max-pooling layer [9] is currently extremely popular. It tells us if a feature is present in the considered group, but not precisely where.

**The Fully Connected Layer**

is the common ANN layer where every input neuron is connected to each layer neuron. While can be used everywhere in the CNN architecture, such a layer is generally the last one [100, 93]. This is motivated by the fact that the fully connected layer acts as a classifier by separating the input data space.

### 5.4.2 Learning Features by Backpropagation

The layers of a CNN define particular connection rules which constrain the neurons to consider local information only. The fully connected one is not subject to such restrictions and is used to produce the final classification. Since CNNs are a particular type of ANNs, the classification error can be backpropagated to learn the connection weights of each CNN layer. In particular, the convolution layer has weights that define the convolution kernel. As the network is trained, these are adapted to extract the optimal local features.

### 5.4.3 What are the limits of CNNs?

CNNs are a special kind of ANNs, so they suffer from similar problems, BP above all. Most importantly, CNNs demand a huge amount of labeled data for training. This boils down to weak generalization power when the number of training data is small and the number of free parameters is large, i.e. when the network is very deep. This is due to the fact that, in such a case, the CNNs is prone to overfitting, or, in other words, the net faces a case of over-parameterization. In addition, as for ANNs, there is no fixed rule to define *a priori* how many hidden layers and neurons should be employed to obtain optimal performance.

To summarize, CNNs have apparent drawbacks and limitations including: (i) BP learning speed and generalization performance for limited training data sets; (ii) tweaks and tricks should be adopted to define the architecture parameters.

## 5.5 Extreme Learning Machines

ANNs and CNNs have been widely explored as architectures to handle complex and challenging tasks. While new learning techniques have been proposed by the community, almost all of them derive from the BP algorithm. Hence, such networks lack faster learning procedures. It is not surprising to see that it may take hours [25], days [93], and even longer to train a single ANN/CNN by using traditional BP-based methods.

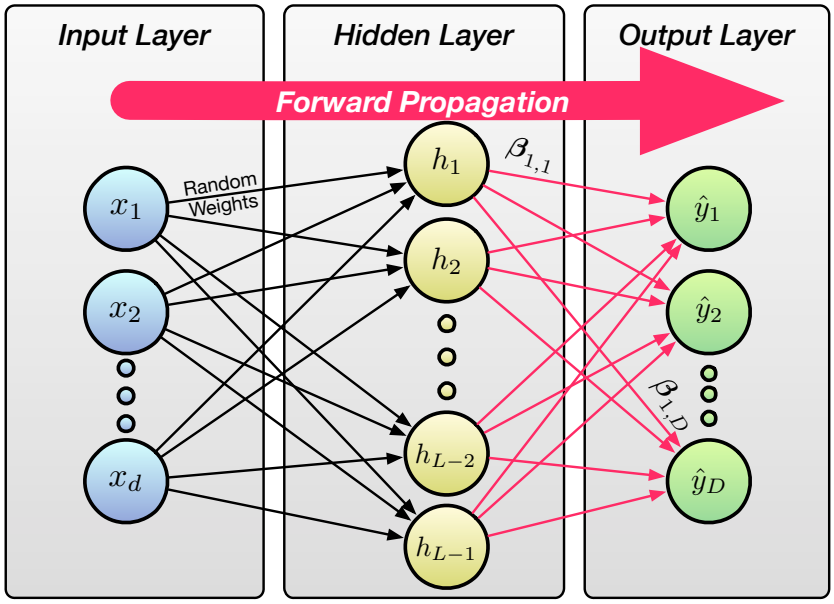



Figure 5.3: Architecture of an Extreme Learning Machine. Unlike previously seen neural architectures, there is no backpropagation in the learning process. With ELM the input-to-hidden weights are randomly picked. Only the set of hidden-to-output connection weights $\boldsymbol{\beta}$ are learned.

### 5.5.1 Biological-Inspired Learning

Extreme Learning Machines (ELMs) [70] were inspired by biological learning and proposed to overcome the issues faced by BP-based learning algorithms. ELMs were devised following the idea that some part of the human brain systems have random neurons whose parameters are independent of their environment. Indeed, all the hidden nodes in an ELM are independent of the training data as well as independent of each other (see Figure 5.3). Hidden nodes need not be tuned and the input to hidden weights can be randomly generated before seeing the training data. The solution to learn the optimal hidden to output weights (in order to obtain the correct input-output mappings) resides on the Moore-Penrose generalized inverse [2].

### 5.5.2 ELMs: Advantages and Shortcomings

ELMs are extremely fast learning algorithms for SLFN and have the following important properties: (i) Minimum training error: the special solution for learning the hidden to output layer connection weights, derived from exploiting the properties of the network structure, as well as the Moore-Penrose generalized inverse matrix [2], is one of the least-square solutions of a linear system. (ii) The smallest norm of

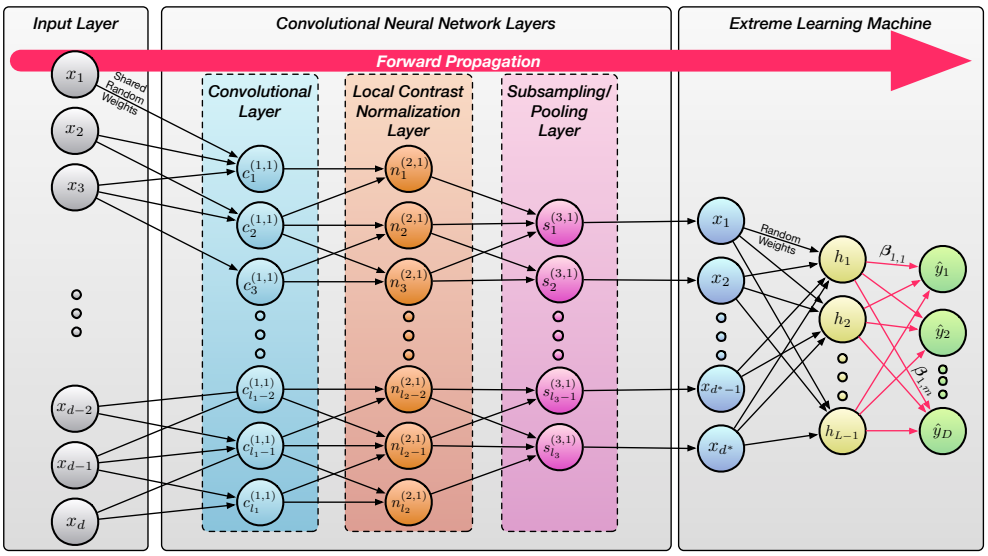



Figure 5.4: Structure of an Extreme Deep Learning Tree in-node model. The input to such model is the raw data. This is convolved with a convolution kernel having random weights. The output of the convolution goes through the local contrast normalization and the pooling layers. The output of the such last layer is used as the input of an ELM. The ELM performs a random mapping and learns the hidden-to-output weights.

weights and best generalization performance: Bartletts neural network generalization theory [4] for feedforward neural networks asserts that the smaller the norms of weights are, the better generalization performance the networks tend to have. ELMs have such a property, indeed, the solution has the smallest norm among all the least-square solutions. (iii) Unique minimum norm least-square solution: the minimum norm solution for learning the hidden to output weights is unique.

While ELMs have such valuable properties, their success or failure, like ANN and NT architectures, hinges on the adopted data representation.

## 5.6 The Future of Brain-Inspired Learning Architectures: Extreme Deep Learning Trees

Considered the weak and strong points that each of the previously presented architectures has, we propose a novel architecture that borrows their strengths and combines them into a unique system. We named such an architecture Extreme Deep Learning Tree (EDLT).

### 5.6.1 EDLT Structure

An EDLT exploits an NT architecture whose in-node models are composed of an ELM stacked at the output of a three-layer CNN (i.e., a CNN composed of a convolutional, local contrast normalization and pooling layers). See Figure 5.4 for the in-node model architecture.

#### The Tree: Less Prior Knowledge.

One of the main current issues of ANNs is to determine the number and arrangement of hidden layers and neurons correctly before training starts. A common approach is to train many different networks (each one having a different structure), and then adopt the configuration of the one that has the minimum validation error. But, even if the optimal configuration for a particular task can be found, it is not certain that the picked configuration will be the best one. Indeed, the possible configurations of a single network with only one hidden layer are infinite (we can have one layer with infinite hidden neurons). The tree structure of an NT, together with the adoption of appropriate split nodes, allows us to solve such a problem.

#### CNN Layers: Random Feature Extraction.

We have shown that CNNs are able to learn meaningful feature extractors only by looking at the data. However, this process requires a network to perform several forward/backward propagations over the data. As well as being time consuming, such a process has been shown to be barely effective [147, 74] if compared to CNNs with random weights. In particular, in [147], it has been shown that a CNN-based architecture with random weights is *frequency selective* as well as *translation invariant*.

In response to this, in our EDLT structure, we use a three-layer CNN with random weights to discover local properties of the data. In particular, for each CNN, hence for each node of the EDLT, we perform a convolution with a single kernel. While this disregards the basic CNN rationale for which subsequent layers learn higher level features, it speeds up the computation (no BP training procedure is used) and does not require the number of kernels to be specified before the training starts.

#### ELM: Fast and Accurate Classification.

One of the biggest problems in current ANN is BP. ELMs can be used to overcome this problem. In particular, we stack an ELM at the output of the CNN pooling layer. Then, the randomly convolved-contrast normalized-subsampled features are projected onto the random feature space which preserves the distance between all pairs of original features (see the Johnson-Lindenstrauss lemma [80] for details). After such a projection, the hidden to output connection weights are learned by computing the Moore-Penrose generalized inverse matrix [70].

### 5.6.2 EDLT Advantages

To summarize, the EDLT architecture has the following strengths: (i) The NT architecture allows us to perform a "divide and conquer" approach, thus inherently reducing big challenging problems to small and easier ones. At the same time, it removes the problem of finding the best number of layers/units in an ANN/CNN. (ii) The random weights CNN architecture avoids the requirement of adopting a hand-crafted feature extractor to represent the given input signal. (iii) The ELM architecture allows us to avoid using BP, thus considerably reducing the learning speed.

### 5.6.3 Performance Evaluation

To evaluate the performance of our method with respect to state-of-the-art deep architectures we considered three datasets: (i) The MNIST dataset [100][1] is probably the most widely adopted dataset to evaluate deep architecture performance. It consists of two datasets with $28 \times 28$ images of handwritten digits, one for training (60k images) and one for testing (10k images). (ii) The LeafSnap dataset [95][2] has 7719 images of leaves taken by mobile devices in outdoor environments. (iii) The ORL Face dataset[3] has $64 \times 64$ images of 40 distinct subjects acquired at different times, with lighting variations and changing facial expressions.

Comparisons are given with respect to Deep Belief Networks (DBNs) [58], Deep Boltzmann Machines (DBMs) [144], Stacked Auto Encoders (SAEs) [176], Statcked Denoising Auto Encoders (SDAEs) [176], standard ELMs with random feature projections, Gaussian kernel ELMs [85], multi-layer ELM [85], Random Forest of Decision Trees [17] and Support Vector Machines (SVMs) with Radial Basis Function (RBF) kernel. For all the datasets, raw pixels' intensities are the input to the such systems.

We conducted the experiments on a Desktop PC with an i7 3770 3.4-GHz core, 16 Gbytes of RAM running MATLAB 2014a. Gaussian-kernel ELMs and SVMs required more than 16 Gbytes of RAM, so we ran it on a high-performance cluster with 2-GHz processors and 128 Gbytes of RAM running MATLAB 2013a.

In Table 5.1 we report on the performance of EDLT using the original MNIST (without any affine or elastic distortion [156]), the LeafSnap and the ORL Face datasets. MNIST and ORL Face common splits have been adopted. For LeafSnap, 5-fold cross validation has been conducted on images downsampled to $32 \times 32$ pixels and converted to grayscale.

Results show that, for every dataset, EDLT meets the performance of state-of-the-art architectures both in terms of recognition performance and in terms of computational training times. Indeed, while the EDLT achieves the best recognition rates on all the three datasets, performance vary little between the adopted approaches. In particular, it is worth noticing that, for the MNIST dataset, the final EDLT structure

[1] Available at http://yann.lecun.com/exdb/mnist/
[2] Available at http://leafsnap.com/dataset/
[3] Available at http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

Table 5.1: Performance comparison of EDLT with state-of-the-art deep architectures. Comparisons have been carried out on the unmodified/unprocessed MNIST, LeafSnap and ORL Face Datasets. Best results are in boldface font. Results for Gaussian-kernel ELMs and SVMs have been computed using a faster machine. It is worth noticing how, on a huge dataset (like MNIST), by exploiting GPU parallelism the proposed approach has lower training time than ELMs.

| Dataset | MNIST | | LeafSnap | | ORL Face | |
|---|---|---|---|---|---|---|
| Algorithm | Acc. % (Std.Dev.) | Training Time (s) | Acc. % (Std.Dev.) | Training Time (s) | Acc. % (Std.Dev.) | Training Time (s) |
| EDLT | **99.06** (± 0.03) | **339.72** | **47.61** (± 1.03) | 129.21 | **95.10** (± 1.52) | 52.01 |
| Multi-layer ELM [85] | 99.02 (± 0.04) | 464.51 | 36.19 (± 0.93) | 58.43 | 94.23 (± 0.04) | 37.65 |
| ELM random features | 97.39 (± 0.1) | 389.39 | 32.04 (± 0.87) | **27.80** | 93.90 (± 1.08) | **14.89** |
| Gaussian kernel ELM [85] | 98.75 (± 0.09) | 790.96 | 40.91 (± 1.02) | 407.21 | 94.41 (± 1.21) | 259.11 |
| Deep belief network [58] (DBN) | 98.87 (± 0.06) | 20,580 | 33.12 (± 1.21) | 1,566 | 94.79 (± 1.89) | 1,066 |
| Deep Boltzmann machine [144] (DBM) | 99.05 (± 0.07) | 68,246 | 35.43 (± 1.30) | 5,420 | 94.03 (± 1.67) | 2,109 |
| Stacked auto-encoder [176] (SAE) | 98.61 (± 0.07) | 9,891 | 22.90 (± 2.12) | 931.77 | 81.80 (± 3.38) | 480.94 |
| Stacked denoising auto-encoder [176] (SDAE) | 98.72 (± 0.06) | 13,707 | 24.19 (± 1.77) | 1,047 | 82.31 (± 2.90) | 600.67 |
| Random Forest (1000 trees) [17] | 98.47 (± 0.08) | 17,746 | 46.37 (± 1.04) | 1,289 | 90.10 (± 2.38) | 810.28 |
| SVM with RBF Kernel | 96.60 (± 0.09) | 29,438 | 25.76 (± 1.52) | 2,797 | 84.20 (± 2.97) | 305.50 |

consists of 364 nodes out of which only 29 have reached depth 16. Since the EDLT borrows the NT structure, depth is determined during training by the *homogeneity* condition. In our case, a partition is *homogeneous*, hence the node becomes a leaf, if all its patterns belong to the same class. To control depth/overfitting, homogeneity property can be modified following [121]. This shows that the model is capable of handling challenging tasks without requiring too deep branches (e.g., overfitted data).

For every considered dataset, our proposed EDLT strongly outperforms the widely adopted deep architectures, like DBNs and DBMs, in terms of computational training times. On average, training an EDLT requires about 3 minutes, while training such architectures require more than 2 and 7 hours respectively.

## 5.7 Conclusion

In this chapter we have given a brief overview of the most widely adopted brain-inspired learning architectures, namely, NTs, CNNs and ELMs. In response to such

an analysis we have proposed a novel architecture that combines the strengths of each one. By using such an architecture we want to overcome the problems related to: (i) the a priori definition of an ANN architecture, (ii) the design of hand crafted discriminative data representations, and (iii) the usage of the slow and tricky BP algorithm. The EDLT architecture borrows the tree structure from NT, and introduces a new in-node architecture consisting of a CNN with an ELM classifier stack at the end. Results on three datasets have shown that state-of-the-art performance have been met at a low computational effort.

# 6

# Concluding Remarks and Future Works

*This last chapter closes the Thesis by drawing the conclusion which emerged from the empirical evaluations of the presented hierarchical learning solutions. Speculations on future works in the field are finally presented.*

In this Thesis, we have addressed the problem of hierarchical learning and proposed four novel solutions which have been empirically evaluated considering the image classification task.

This Thesis largely drew inspiration from the fascinating biological brain organisms and its ability to learn simple concepts as well as complex notions from a few samples. In particular, it leverages on the recent neuroscience discoveries which showed that the human brain is organized in a hierarchical fashion with large circuit modularity and substantial reuse of general sub-circuits. However, delivering a solution to the ultimate AI goal is not possible yet. Thus, the presented content focused on the vision problem which has been recently shown to be one of the most promising windows into human intelligence.

Following such key insights, we started by conducting an empirical evaluation study that aimed to tie the discrimination and invariance properties of the biological vision with functional modules and computer vision theory of wavelets. Then, we continued by following the recent neuroscience discoveries highlighting the evidence for massive parallel operations that are performed by the mammalian visual systems such that the aggregation of different visual clues can be instantaneously performed. Such assumptions have been validated by studying the effects of different visual representations within a framework consisting of shallow hierarchical architectures.

These solutions, however, neglected the deep hierarchy of computations present in the primate visual system. Indeed, it has been shown that in the human brain hierarchical model the initial processing layers act as generic feature detectors which

produce information that is exploitable in the context of many specific classification problems, while by going deeper in the hierarchy, more task-specific information is captured. Pivoting on this argument, we capitalized on the deep learning wave of enthusiasm and introducing a hierarchical learning model whose architecture design depends on the final classification task. Such argument, however, contrasts the model adaptability which the human brain is embodied. It is not new, that the brain functional networks does not dwell on any external source/knowledge/decision to construct/modify its structure, but autonomously adapts through evolution. This strong evidence motivated the investigation of a hierarchical architecture that follows the "divide and conquer" approach which is highly connected with naturally occurring phenomena. As a result, we presented a hierarchical structure automatically adapts itself to the problem complexity by introducing new computational units when needed.

The following were the main contributions of the Thesis.

**A Forest of Random Trees for the Optimal Selection of Feature Encodings**

In chapter 2, we have performed an in-depth analysis of widely adopted hand-crafted visual features that aim to resemble the process of feature extraction observed in the mammalian brain. The idea was to throughly investigate the performance of existing approaches on the image classification task to have a better grasp on their advantages and limitations. The evaluation has been carried out considering a classical hierarchical learning solution that proved to be extremely efficient in capturing only the discriminative features among the pool of available ones. The obtained results showed that considered techniques have poor performance per se, but can be useful if a proper classification scheme is adopted. The chapter defined the basis for the whole Thesis.

**Learning to Rank with a Committee of Shallow Neural Networks**

Results from the empirical evaluations conducted in chapter 2 demonstrated that image classification is a problem with many challenges which cannot be addressed only by considering simple filter-response features and a unique classification solution. Despite the successful stories achieved by considering methods belonging to such a category, it is often the case that the designed ad-hoc image representations based on some *a priori* knowledge of the problem are not sufficient to correctly handle all the problem challenges. In chapter 3, a possible solution to sidestep the aforementioned problems has been introduced. It brings to the limits the idea that any possible hand-crafted visual feature representation could be relevant, but the optimal result is obtained if a proper fusion scheme is exploited. Following this idea, we prosed a classification system based on a committee of shallow neural networks each of which separately consider a specific visual characteristic. Then, an optimal ranking is delivered by fusing the committee classifications via a Structural Support Vector Machine. Comparisons with the state-of-the-art have demonstrated the benefits of

the proposed architecture.

**Going Deeper and Wider with Wide-Slice Residual Networks**

In chapter 4 we have built upon the results of the approach discussed in chapter 3. In particular, we have focused on the limitations of hand-crafted visual features. Inspired by this, we aimed to understand how feature representations can be learned from data. Differently from state-of-the-art methods, we have not considered a learning solution that exploits off-the shelf schemes, thus neglecting the design of a proper architecture which considers the specific problem challenges. Indeed, we have proposed a novel architecture that is defined following an analysis of the image composition. We have shown that our approach significantly improves the existing performance on three benchmark datasets. The analysis of the source of performance demonstrated that robust feature representations are learned and, more interestingly, that such solution is able to automatically focus on the image portion that contains the object of interest to perform its classification.

**New Directions– Extreme Deep Learning Trees: The Evolution of Neural Learning Systems**

In chapter 5 we have introduced a novel direction of research. As the large majority of the existing literature in the field, all the approaches discussed in the preceding chapters hinge on the manual specification of multiple hyperparameters (e.g., the number of features to consider, the number of hidden layers/neurons in an ANN, etc.). It is a matter of fact that the selection of such hyperparameters is a tedious "trial-and-error" approach which does not guarantee that the obtained values are optimal for the task. The lack of a solution aiming to address all the discussed hierarchical learning issues, plus the problem of hyperparameters selection, motivated the design of a novel NT-based hierarchical scheme that yields to three main achievements: i) does not require the specification of any of the common ANN hyperparameters; ii) very fast learning and inference are performed by solving the optimization problems in an analytical fashion; iii) does not hinge on hand-crafted visual features but obtains optimal image representations by means of a CNN-driven approach. The preliminary analysis conducted on different image classification tasks has demonstrated the benefits of the solution, both in terms of accuracy as well as with regard to the computational performance.

# Future Work

Mimicking the human brain to achieve human-level cognition performance has been a core challenge in artificial intelligence research for decades. Humans are very efficient in capturing the most important information while being exposed to a plethora of different stimuli, a capability that is used to represent and understand their surroundings in a concise fashion. Machine learning research has made considerable progress towards cloning such human capability with innovative techniques like deep and feature learning, incremental learning, etc. The approaches proposed in this Thesis to address

the visual image classification challenges can also be extended to other tasks. So, our work opens up to a very wide range of applications in numerous other areas. Among of them we'll outline two directions which can lead to future works, one dealing with the process of learning visual feature representations, the other with applications on different domains.

### Learning Representations

Having features invariant to pose, illumination changes, viewpoint variations, rotations,etc. is the holy grail of computer vision as it can lead to the solution of an enormous variety of problems like recognition, tracking, etc.. Despite the huge effort put by the community, such kind of features have not been discovered yet. With the last two chapters of the Thesis, we introduced two solutions that aimed to mimic the human brain in terms of its vision-classification capabilities. In particular, in chapter 4, we introduced an architecture able to automatically focus on the image portion that contains the object of interest and extract the most relevant features to perform its classification. While, in the last chapter we mainly explored the usage of features that correspond to the visual characteristics considered by the V1 area of the occipital cortex [94]. We believe that being able to introduce the process of learning more abstract and high-level representations in the EDLT structure will open to new and relevant discoveries that can have a strong impact towards the ultimate human brain mimicking goal.

### Applications

The concept and the ideas that have been discussed in this thesis can be used and applied in many other domains different from the vision one. For instance, the process of separately considering different features, then fusing them to provide an optimal ranking, can also be used to present the web-surfer a ranking of pages that best respond to a given search. This is just an example of such applications, but we can also think about other domain, like sound understanding, language processing, medical analysis, etc..

# Bibliography


[1] M. F. Amasyali and O. Ersoy. Cline: A new decision-tree family. *IEEE Transactions on Neural Networks*, 19(2):356–363, 2008. 12

[2] K. S. Banerjee. Generalized Inverse of Matrices and Its Applications. *Technometrics*, 15(1):197–197, feb 1973. 51, 108

[3] R. C. Barros, P. a. Jaskowiak, R. Cerri, and A. C. P. L. F. De Carvalho. A framework for bottom-up induction of oblique decision trees. *Neurocomputing*, 135:3–12, 2014. 103

[4] P. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536, mar 1998. 51, 109

[5] P. L. Bartlett. The Network of the Future-4WARD. In *Advances in Neural Information Processing Systems*, pages 134—-140, 1996. 51

[6] D. S. Bassett, N. F. Wymbs, M. A. Porter, P. J. Mucha, J. M. Carlson, and S. T. Grafton. Dynamic reconfiguration of human brain networks during learning. *Learning*, 108(18):7641–7646, 2010. 3

[7] Y. Bengio. Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127, 2009. 8, 105

[8] Y. Bengio and Y. Lecun. Scaling Learning Algorithms towards AI. *Large Scale Kernel Machines*, (1):321–360, 2007. 7

[9] Y. Bengio, A. Courville, and P. Vincent. Representation learning: a review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–828, aug 2013. 8, 105, 106

[10] L. Bertelli, T. Yu, D. Vu, and B. Gokturk. Kernelized structural SVM learning for supervised object segmentation. *International Conference on Computer Vision and Pattern Recognition*, pages 2153–2160, 2011. 57

[11] V. Bettadapura, E. Thomaz, A. Parnami, G. D. Abowd, and I. Essa. Leveraging Context to Support Automated Food Recognition in Restaurants. In *Winter Conference on Applications of Computer Vision*, 2015. 47, 66, 90

[12] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1st edition, 1996. ISBN 978-0198538646. 7

[13] L. Bobrowski. Learning processes in multilayer threshold nets. *Biological Cybernetics*, 31(1):1–6, 1978. 6

[14] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *International Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008. 47

[15] A. Bosch, A. Zisserman, and X. Munoz. Image Classification using Random Forests and Ferns. In *International Conference on Computer Vision*, pages 1–8. Ieee, 2007. 46

[16] L. Bossard, M. Guillaumin, and L. Van Gool. Food-101 Mining Discriminative Components with Random Forests. In *European Conference Computer Vision*, 2014. 29, 40, 61, 75, 76, 77, 78, 83, 90, 93

[17] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001. 18, 25, 103, 111, 112

[18] L. Breiman, J. Friedman, C. J. Stone, and R. Olshen. *Classification and Regression Trees*. Chapman and Hall, 1st edition, 1984. 10, 103

[19] R. P. Brent. Fast training algorithms for multilayer neural nets. *IEEE transactions on neural networks*, 2(3):346–54, jan 1991. 12

[20] W. L. Buntine and A. S. Weigend. Computing second derivatives in feed-forward networks: a review. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 5(3):480–8, jan 1994. 102

[21] J. Chen and C.-W. Ngo. Deep-based Ingredient Recognition for Cooking Recipe Retrieval. In *ACM Multimedia*, pages 1—-6, 2016. 84, 88, 90, 91, 92

[22] M. Chen, K. Dhingra, W. Wu, L. Yang, R. Sukthankar, and J. Yang. PFID: Pittsburgh Fast-food Image Dataset. *International Conference on Image Processing*, pages 289–292, 2009. 27, 59, 61, 62, 67

[23] N. Chomsky. *Rules and Representations*. Columbia University Press, 2005. ISBN 978-0231132718. 1

[24] K. J. Cios and N. Liu. A machine learning method for generation of a neural network architecture: a continuous ID3 algorithm. *IEEE transactions on neural networks*, 3(2):280–91, jan 1992. 12

[25] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column Deep Neural Networks for Image Classification. In *International Conference on Computer Vision and Pattern Recognition*, pages 3642–3649, 2012. 8, 107

[26] Z. Deng, A. Vahdat, H. Hu, and G. Mori. Structure Inference Machines: Recurrent Neural Networks for Analyzing Relations in Group Activity Recognition. In *International Conference on Computer Vision and Pattern Recognition*, 2016. 8, 83

[27] E. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. In *Advances in Neural Information Processing Systems*, pages 1–10, 2015. 9

[28] J. J. DiCarlo, D. Zoccolan, and N. C. Rust. How does the brain solve visual object recognition? *Neuron*, 73(3):415–34, feb 2012. 3, 106

[29] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell, T. Eecs, and B. Edu. DeCAF : A Deep Convolutional Activation Feature for Generic Visual Recognition. In *International Conference on Machine Learning*, volume 32, 2014. 8

[30] J. Dong and S. Soatto. Domain-Size Pooling in Local Descriptors: DSP-SIFT. In *International Conference on Computer Vision and Pattern Recognition*, 2015. 47, 60

[31] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative Unsupervised Feature Learning with Convolutional Neural Networks. *arXiv preprint arXiv: ...*, pages 1–13, 2014. 8

[32] G. M. Farinella, D. Allegra, and F. Stanco. A Benchmark Dataset to Study the Representation of Food Images. In *European Conference Computer Vision Workshops*, 2014. 27, 28, 40, 46, 47, 59, 61, 67, 68, 71

[33] G. M. Farinella, M. Moltisanti, and S. Battiato. Classifying Food Images Represented as Bag of Textons. In *International Conference on Image Processing*, pages 5212—-5216, 2014. 27, 43, 46, 47, 59, 62, 66, 67, 83, 90

[34] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional Two-Stream Network Fusion for Video Action Recognition. In *International Conference on Computer Vision and Pattern Recognition*, pages 1933–1941, 2016. 8, 83

[35] G. Foresti and T. Dolso. An Adaptive High-Order Neural Tree for Pattern Recognition. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, 34(2):988–996, apr 2004. 12

[36] G. L. Foresti and C. Micheloni. Generalized neural trees for pattern classification. *IEEE Transactions on Neural Networks*, 13(6):1540–7, jan 2002. 12

[37] G. L. Foresti and G. G. Pieroni. Exploiting Neural Trees in Range Image Understanding. *Pattern Recognition Letters*, 19(9):869–878, 1998. 12

[38] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980. 6

[39] C. F. Gauss. *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium*. 1809. 5

[40] P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *International Conference on Computer Vision*, 2009. 61, 69, 73, 78, 79

[41] J.-M. Geusebroek, R. van den Boomgaard, A. Smeulders, and H. Geerts. Color invariance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1338–1350, 2001. 46

[42] F. Girosi and T. Poggio. Networks and the best approximation property. *Biological Cybernetics*, 63(3):169–176, jul 1990. 7

[43] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *International Conference on Computer Vision and Pattern Recognition*, pages 1 —- 8, 2013. 8, 9

[44] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014. 9

[45] Y. Guo, G. Zhao, and M. Pietikäinen. Texture Classification using a Linear Configuration Model based Descriptor. *Procedings of the British Machine Vision Conference 2011*, pages 119.1–119.10, 2011. 46

[46] M. Hagiwara. Theoretical derivation of momentum term in back-propagation. In *International Joint Conference on Neural Networks*, volume 1, pages 682–686. IEEE, 1992. 102

[47] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations*, pages 1–13, 2016. 97

[48] H. Hassannejad, G. Matrella, P. Ciampolini, I. De Munari, M. Mordonini, and S. Cagnoni. Food Image Recognition Using Very Deep Convolutional Networks. In *European Conference Computer Vision Workshops and Demonstrations*, pages 41–49, 2016. 84, 88, 90, 92, 93

[49] S. O. Haykin. *Neural Networks and Learning Machines*. Prentice Hall, 3rd edition, 2008. 102

[50] K. He and R. Girshick. Mask R-CNN. *arXiv:1703.06870*, 2017. 8, 9

[51] K. He, X. Zhang, S. Ren, and J. Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *arXiv preprint arXiv ...*, cs.CV, 2014. 8, 9

[52] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *ArXiv e-prints*, 1512.03385, 2015. 85

[53] K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *ArXiv e-prints*, abs/1502.0, 2015. 91

[54] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *International Conference on Computer Vision and Pattern Recognition*, pages 770—-778, 2016. 2, 8, 83, 85, 87, 88, 91, 92, 93

[55] K. He, X. Zhang, S. Ren, and J. Sun. Identity Mappings in Deep Residual Networks. In *European Conference on Computer Vision*, pages 630–645, 2016. 87, 88

[56] L. A. Hendricks, S. Venugopalan, M. Rohrbach, R. Mooney, K. Saenko, and T. Darrell. Deep Compositional Captioning: Describing Novel Object Categories without Paired Training Data. In *International Conference on Computer Vision and Pattern Recognition*, pages 1–10, 2016. 8, 83

[57] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(July):504–507, 2006. 7, 101

[58] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006. 111, 112

[59] S. Hochreiter. *Untersuchungen zu dynamischen neuronalen Netzen*. PhD thesis, Technical University of Munchen, 1991. 6

[60] A. E. Hoerl and R. W. Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1):55–67, 1970. 52

[61] J. M. Hopf, M. Bader, M. Meng, and J. Bayer. Is human sentence parsing serial or parallel?: Evidence from event-related brain potentials. *Cognitive Brain Research*, 15(2):165–177, 2003. 2

[62] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, jan 1991. 2

[63] A. G. Howard. Some Improvements on Deep Convolutional Neural Network Based Image Classification. *ArXiv e-prints*, 1312.5402, 2013. 91

[64] R. Hu, H. Xu, M. Rohrbach, J. Feng, K. Saenko, and T. Darrell. Natural Language Object Retrieval. In *International Conference on Computer Vision and Pattern Recognition*, pages 4555–4564, 2016. 8, 83

[65] G. B. Huang. Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE Transactions on Neural Networks*, 14(2):274–281, 2003. 50

[66] G.-B. Huang. An Insight into Extreme Learning Machines: Random Neurons, Random Features and Kernels. *Cognitive Computation*, apr 2014. 52

[67] G. B. Huang and H. A. Babri. Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. *IEEE Transactions on Neural Networks*, 9(1):224–9, jan 1998. 7

[68] G.-b. Huang, Q.-y. Zhu, and C.-k. Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *IEEE International Joint Conference on Neural Networks*, volume 2, pages 985–990. IEEE, 2004. 51

[69] G.-B. Huang, L. Chen, and C.-K. Siew. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Transactions on Neural Networks*, 17(4):879–92, jul 2006. 50

[70] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1-3):489–501, dec 2006. 47, 48, 49, 51, 108, 110

[71] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang. Extreme learning machine for regression and multiclass classification. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics*, 42(2):513–29, apr 2012. 44, 52, 53

[72] D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148:574–91, oct 1959. 2, 5

[73] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems*, pages 487–493, 1999. 23, 47

[74] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *International Conference on Computer Vision*, pages 2146–2153. IEEE, sep 2009. 106, 110

[75] H. Jegou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation.pdf. In *International Conference on Computer Vision and Pattern Recognition*, 2010. 24

[76] H. J. Jerison. *Evolution of the Brain and Intelligence*. Academic Press, 2012. ISBN 978-0124121959. 3

[77] T. Joachims. A Support Vector Method for Multivariate Performance Measures. *International Conference on Machine Learning*, 440:377–384, 2005. 57, 58

[78] T. Joachims, T. Finley, and C. N. J. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009. 54, 56

[79] J. Johnson, A. Karpathy, and L. Fei-Fei. Densecap: fully convolutional localization networks for dense captioning. In *International Conference on Computer Vision and Pattern Recognition*, 2016. 8

[80] W. B. Johnson, J. Lindenstrauss, and G. Schechtman. Extensions of lipschitz maps into Banach spaces. *Israel Journal of Mathematics*, 54(2):129–138, jun 1986. 49, 110

[81] A. Joulin, L. van der Maaten, A. Jabri, and N. Vasilache. Learning Visual Features from Large Weakly Supervised Data. In *European Conference on Computer Vision*, pages 67–84, 2016. 8, 95

[82] T. Joutou and K. Yanai. A food image recognition system with Multiple Kernel Learning. *International Conference on Image Processing*, 2009. 46

[83] B. Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290:91–97, 1981. 46

[84] A. Karpathy and T. Leung. Large-scale Video Classification with Convolutional Neural Networks. In *International Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014. 8

[85] L. L. C. Kasun, H. Zhou, G.-B. Huang, and C.-M. Vong. Representational Learning with ELMs for Big Data. *IEEE Intelligent Systems*, 28(6):30–59, nov 2013. 49, 111, 112

[86] Y. Kawano and K. Yanai. FoodCam: A real-time mobile food recognition system employing Fisher Vector. *Multimedia Tools and Applications*, pages 369–373, 2014. 71, 90

[87] Y. Kawano and K. Yanai. Food Image Recognition with Deep Convolutional Features. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, pages 589–593, 2014. 84, 88, 90, 92, 93

[88] Y. Kawano and K. Yanai. Automatic expansion of a food image dataset leveraging existing categories with domain adaptation. In *European Conference Computer Vision Workshops and Demonstrations*, pages 3–17, 2014. 90

[89] Y. Kawano and K. Yanai. FoodCam-256: A Large-scale Real-time Mobile Food RecognitionSystem employing High-Dimensional Features and Compression of Classifier Weights. In *ACM International Conference on Multimedia*, pages 761–762, 2014. 90

[90] D. M. Kline and V. L. Berardi. Revisiting squared-error and cross-entropy functions for training neural network classifiers. *Neural Computing and Applications*, 14(4):310–318, jul 2005. 101

[91] S. B. Kotsiantis. Decision trees: A recent overview. *Artificial Intelligence Review*, 39(4):261–283, 2013. 103

[92] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, pages 1097—-1105, 2012. 2, 8, 91

[93] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, pages 1106–1114, 2012. 107

[94] N. Krüger, P. Janssen, S. Kalkan, M. Lappe, and A. Leonardis. Deep Hierarchies in the Primate Visual Cortex: What Can We Learn for Computer Vision? *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1847–1871, 2013. 3, 4, 17, 118

[95] N. Kumar, P. N. Belhumeur, A. Biswas, D. W. Jacobs, W. J. Kress, I. C. Lopez, and V. B. Soares. A Computer Vision System for Automatic Plant Species Identification. In *European Conference on Computer Vision*, pages 1–14, 2012. 111

[96] K. I. Laws. Rapid Texture Identification. In T. F. Wiener, editor, *Image Processing for Missile Guidance*, pages 376–381, dec 1980. 17, 20

[97] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2:2169–2178, 2006. 46

[98] Q. V. Le, M. A. Ranzato, M. Devin, G. S. Corrado, and A. Y. Ng. Building High-level Features Using Large Scale Unsupervised Learning. In *International Conference on Machine Learning*, 2012. 8

[99] N. Le Roux and Y. Bengio. Representational Power of Restricted Boltzmann Machines and Deep Belief Networks. *Neural Computation*, 20(6):1631–1649, jun 2008. 8

[100] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 7, 105, 106, 107, 111

[101] A. M. Legendre. *Nouvelles methodes pour la determination des orbites des cometes*. Didot F., 1805. 5

[102] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, jan 1993. 2

[103] T. Leung and J. Malik. Representing and Recognizing the Visual Appearance of Materials using Three-dimensional Textons. *International Journal of Computer Vision*, 43(1):29–44, 2001. 18, 20

[104] M. Lin, Q. Chen, and S. Yan. Network In Network. *arXiv preprint*, page 10, 2013. 8

[105] S. Linnainmaa. *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors*. PhD thesis, University of Helsinki, 1970. 6

[106] S. Linnainmaa. Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics*, 16(2):146–160, jun 1976. 6

[107] C. Liu, Y. Cao, Y. Luo, G. Chen, V. Vokkarane, and Y. Ma. Deepfood: Deep learning-based food image recognition for computer-aided dietary assessment. In *IEEE International Conference on Smart Homes and Health Telematics*, volume 9677, pages 37–48, 2016. 84, 88, 91, 92, 93

[108] X. Liu, L. Wang, G.-B. Huang, J. Zhang, and J. Yin. Multiple kernel extreme learning machine. *Neurocomputing*, 226(2012):63–69, 2015. 61, 79

[109] P. Maji. Efficient design of neural network tree using a new splitting criterion. *Neurocomputing*, 71(4-6):787–800, jan 2008. 12

[110] N. Martinel and C. Micheloni. Re-identify people in wide area camera network. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 31–36, Providence, RI, jun 2012. IEEE. 59

[111] N. Martinel, C. Micheloni, and C. Piciarelli. Learning pairwise feature dissimilarities for person re-identification. In *International Conference on Distributed Smart Cameras*, pages 1–6, Palm Springs, CA, oct 2013. IEEE. 25

[112] N. Martinel, C. Micheloni, and G. L. Foresti. The Evolution of Neural Learning Systems: A Novel Architecture Combining the Strengths of NTs, CNNs, and ELMs. *IEEE Systems, Man, and Cybernetics Magazine*, 1(3):17–26, jul 2015. 44

[113] N. Martinel, C. Piciarelli, C. Micheloni, and G. L. Foresti. A Structured Committee for Food Recognition. In *International Conference on Computer Vision Workshops*, 2015. 91, 92, 93, 94, 95

[114] N. Martinel, C. Piciarelli, C. Micheloni, and G. L. Foresti. On Filter Banks of Texture Features for Mobile Food Classification. In *International Conference on Distributed Smart Cameras*, pages 11–16, Seville, Spain, 2015. 18, 40, 71

[115] N. Martinel, C. Piciarelli, and C. Micheloni. A supervised extreme learning committee for food recognition. *Computer Vision and Image Understanding*, 148:67–86, 2016. 83

[116] Y. Matsuda and K. Yanai. Multiple-Food Image Recognition Considering Co-occurrence. In *International Conference on Pattern Recognition*, pages 1724–1730, 2013. 43, 46

[117] Y. Matsuda, H. Hoashi, and K. Yanai. Recognition of multiple-food images by detecting candidate regions. In *International Conference on Multimedia and Expo*, pages 25–30, 2012. 28, 29, 40, 61, 71, 73, 83, 89, 90

[118] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943. 5

[119] B. Mcfee and G. Lanckriet. Metric Learning to Rank. *International Conference on Machine Learning*, pages 775–782, 2010. 57

[120] D. Meunier, R. Lambiotte, A. Fornito, K. D. Ersche, and E. T. Bullmore. Hierarchical modularity in human brain functional networks. *Frontiers in Neuroinformatics*, 3(October):37, 2009. 1

[121] C. Micheloni, A. Rani, S. Kumar, and G. L. Foresti. A balanced neural tree for pattern classification. *Neural Networks*, 27:81–90, 2012. 105, 112

[122] N. Morgan and H. Bourlard. Continuous speech recognition using multilayer perceptrons with hidden Markov models. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 413–416. IEEE, 1990. 102

[123] J. S. Morris, A. Ohman, and R. J. Dolan. A subcortical pathway to the right amygdala mediating "unseen" fear. *Proceedings of the National Academy of Sciences*, 96(4):1680–1685, 1999. 2

[124] K. R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, mar 2001. 7, 52

[125] A. Myers, N. Johnston, V. Rathod, A. Korattikara, A. Gorban, N. Silberman, S. Guadarrama, G. Papandreou, J. Huang, and K. Murphy. Im2Calories : towards an automated mobile vision food diary. In *International Conference on Computer Vision*, 2016. 84

[126] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, jul 2002. 46, 59

[127] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42 (3):145–175, 2001. 46, 59

[128] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks. In *International Conference on Computer Vision and Pattern Recognition*, 2014. 95

[129] Y. Park. A comparison of neural net classifiers and linear tree classifiers: Their similarities and differences. *Pattern Recognition*, 27(11):1493–1503, nov 1994. 12

[130] F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6314 LNCS(PART 4):143–156, 2010. 23, 47

[131] N. Pinto, D. D. Cox, and J. J. DiCarlo. Why is real-world visual object recognition hard? *PLoS computational biology*, 4(1):e27, jan 2008. 106

[132] T. Poggio and F. Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247(4945):978–982, 1990. 7

[133] X. Qi, R. Xiao, C.-G. Li, Y. Qiao, J. Guo, and X. Tang. Pairwise Rotation Invariant Co-occurrence Local Binary Pattern. *IEEE Transactions on Pattern Analysis and Machine Iintelligence*, 36(11):2199 – 2213, 2014. 40, 46, 60, 67, 71, 83

[134] J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1:81–106, 1986. 12

[135] A. Radford, L. Metz, and S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *International Conference on Learning Representations*, pages 1–15, 2016. 9

[136] E. Rahtu, J. Heikkilä, V. Ojansivu, and T. Ahonen. Local phase quantization for blur-insensitive image analysis. *Image and Vision Computing*, 30(8):501–512, aug 2012. 46, 59

[137] C. R. Rao and S. K. Mitra. Generalized Inverse of a Matrix and Its Applications. In *Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 601–620, 1972. 51

[138] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In *European Conference on Computer Vision*, pages 525–542, 2016. 97

[139] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN Features off-the-shelf : an Astounding Baseline for Recognition. In *Computer Vision and Pattern Recognition Workshops*, 2014. 47, 60

[140] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *International Conference on Computer Vision and Pattern Recognition*, 2016. 9

[141] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–408, 1958. 5

[142] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, oct 1986. 6, 100, 102

[143] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, dec 2015. 8, 91

[144] R. Salakhutdinov and G. Hinton. Deep Boltzmann Machines. In *International Conference on Artificial Intelligence and Statistics*, volume 5, pages 448–455, 2009. 101, 111, 112

[145] J. Sanchez, F. Perronnin, T. Mensink, and J. Verbeek. Image Classification with the Fisher Vector: Theory and Practice. *International Journal of Computer Vision*, 105(3):222–245, 2013. 23, 47

[146] A. Sankar and R. J. Mammone. Growing and pruning neural tree networks. *IEEE Transactions on Computers*, 42(3):291–299, 1993. 11

[147] A. M. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Y. Ng. On Random Weights and Unsupervised Feature Learning. In *International Conference on Machine Learning*, pages 1–9, 2011. 110

[148] C. Schmid. Constructing models for content-based image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–39–II–45, Montbonnot, France, 2001. IEEE Comput. Soc. 18, 20

[149] J. Schmidhuber. Deep Learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. 7

[150] B. Schölkopf, C. J. C. Burges, and A. J. Smola. *Advances in Kernel Methods: Support Vector Learning*. The MIT Press, 1998. ISBN 978-0262194167. 7

[151] A. Schwaighofer and V. Tresp. The Bayesian Committee Support Vector Machine. In *International Conference on Artificial Neural Networks*, pages 411–417, 2001. 44

[152] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra. Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization. *ArXiv e-prints*, 1610.02391, 2016. x, 8, 96, 97

[153] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated Recognition , Localization and Detection using Convolutional Networks. In *International Conference on Learning Representations*, pages 1–15, 2014. 8, 47, 77, 95

[154] I. K. Sethi and J. H. YOO. Structure-driven induction of decision tree classifiers through neural learning. *Pattern Recognition*, 30(11):1893–1904, nov 1997. 12

[155] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, jun 2008. 46, 67

[156] P. Simard, D. Steinkraus, and J. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *International Conference on Document Analysis and Recognition*, volume 1, pages 958–963. IEEE Comput. Soc, 2000. 111

[157] H. a. Simon. The Architecture of Complexity: Hierarchic Systems. In *Proceedings of the American Philosophical Society*, volume 106, pages 467–482, 1962. 1

[158] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations*, pages 1–14, 2015. 2, 8, 9, 83, 88

[159] J. A. Sirat and J.-P. Nadal. Neural Trees: a new tool for classification. *Neural Network*, 1(4):423–448, 1990. 11, 103

[160] D. F. Specht. A general regression neural network. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 2(6):568–76, jan 1991. 101

[161] O. Sporns. Structure and function of complex brain networks. *Dialogues in Clinical Neuroscience*, 15(3):247–262, 2013. 2

[162] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway Networks. *ArXiv e-prints*, 1505.00387, 2015. 8, 9, 85, 88, 94

[163] M. Stone. Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36(2):111–147, 1974. 102

[164] a. Suarez and J. Lutsko. Globally optimal fuzzy decision trees for classification and regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(12):1297–1311, 1999. 103

[165] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147, 2013. 102

[166] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *International Conference on Computer Vision and Pattern Recognition*, pages 1–9. IEEE, jun 2015. 2, 8, 9, 83, 88, 91

[167] V. Tresp. A Bayesian Committee Machine. *Neural Computation*, 12:2719–2741, 2000. 44

[168] V. Tresp. Committee Machines. *Handbook for Neural Network Signal Processing*, pages 1–21, 2001. 44

[169] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. *International conference on Machine Learning*, page 104, 2004. 54, 57

[170] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large Margin Methods for Structured and Interdependent Output Variables. *Journal of Machine Learning Research (JMLR)*, 6:1453–1484, 2005. 14, 44, 54, 58

[171] P. E. Utgoff. Perceptron Trees: A Case Study in Hybrid Concept Representations. *Connection Science*, 1(4):377–391, jan 1989. 11, 104

[172] K. E. a. van de Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–96, sep 2010. 46, 47, 60

[173] V. N. Vapnik. *The Nature of Statistical Learning Theory*, volume 8. Springer - Verlag, New York, 1995. ISBN 0387987800. 7, 52

[174] M. Varma and A. Zisserman. Texture classification: are filter banks necessary? *International Conference on Computer Vision and Pattern Recognition*, 2, 2003. 18, 20

[175] M. Varma and A. Zisserman. A Statistical Approach to Texture Classification from Single Images. *International Journal of Computer Vision*, 62:61–81, 2005. 46, 47

[176] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *Journal of Machine Learning Research*, 11 (3):3371–3408, 2010. 101, 111, 112

[177] X. Wang and A. Gupta. Generative Image Modeling using Style and Structure Adversarial Networks. In *European Conferece on Computer Vision*, 2016. 9

[178] T. Weldon, W. Higgins, and D. Dunn. Efficient Gabor filter design for texture segmentation. *Pattern Recognition*, 29(12):2005–2015, 1996. 18, 20

[179] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Science*. PhD thesis, Harvard University, 1974. 6

[180] K. Yanai and Y. Kawano. Food image recognition using deep convolutional network with pre-training and fine-tuning. In *International Conference on Multimedia & Expo Workshops*, pages 1–6. IEEE, jun 2015. 92, 93

[181] K. Yanai, T. Kaneko, and Y. Kawano. Real-Time Photo Mining from the Twitter Stream: Event Photo Discovery and Food Photo Detection. *International Symposium on Multimedia*, pages 295–302, 2014. 73, 75

[182] S. Yang, M. Chen, D. Pomerleau, and R. Sukthankar. Food recognition using statistics of pairwise local features. *International Conference on Computer Vision and Pattern Recognition*, pages 2249–2256, 2010. 43, 46, 62, 66, 67

[183] C. T. Yildiz and E. Alpaydin. Omnivariate decision trees. *IEEE Transactions on Neural Networks*, 12(6):1539–46, jan 2001. 12

[184] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo. Image Captioning with Semantic Attention. In *International Conference on Computer Vision and Pattern Recognition*, page 10, 2016. 8

[185] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A Support Vector Method for Optimizing Average Precision. In *ACM SIGIR conference on Research and development in information retrieval*, pages 271–278, 2007. 57

[186] S. Zagoruyko and N. Komodakis. Wide Residual Networks. In *British Machine Vision Conference*, 2016. 8, 9, 88, 91, 92, 93

[187] G. P. Zhang. Neural Networks for Classification : A Survey. *IEEE Transactions on System, Man and Cybernetics Part C - Applications and Reviews*, 30(4):451–462, 2000. 101

[188] L. Zhang, Z. Zhou, and H. Li. Binary Gabor pattern: An efficient and robust descriptor for texture classification. In *International Conference on Image Processing*, pages 81–84. Ieee, sep 2012. 46, 60

[189] Z. Zhang, S. Fidler, and R. Urtasun. Instance-Level Segmentation with Deep Densely Connected MRFs. In *International Conference on Computer Vision and Pattern Recognition*, pages 669–677, 2015. 8, 83

[190] J. Zhao, M. Mathieu, and Y. LeCun. Energy-based Generative Adversarial Network. In *International Conference on Learning Representations*, pages 1–15, 2017. 9

[191] Q. Zhao. Evolutionary design of neural network tree-integration of decision tree, neural network and GA. In *IEEE Congress on Evolutionary Computation*, volume 1, 2001. 103

[192] Q. Zhao. Inducing NNC-Trees With the R^4 Rule. *IEEE Transactions on Systems, Man and Cybernetics Part B*, 36(3):520–533, 2006. 103

[193] Z.-H. Zhou and Z.-Q. Chen. Hybrid decision tree. *Knowledge-Based Systems*, 15(8):515–528, nov 2002. 12

[194] L. Zhu, Y. Chen, A. Yuille, and W. Freeman. Latent hierarchical structural learning for object detection. In *International Conference on Computer Vision and Pattern Recognition*, pages 1062–1069, 2010. 57

[195] Y. Zhu, O. Groth, M. Bernstein, and L. Fei-Fei. Visual7W: Grounded Question Answering in Images. In *International Conference on Computer Vision and Pattern Recognition*, pages 4995–5004, 2016. 8, 83