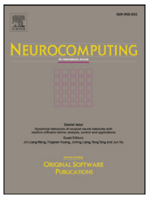
Neurocomputing 338 (2019) 418–431

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Visual content-based web page categorization with deep transfer learning and metric learning

Daniel López-Sánchez[a,*], Angélica González Arrieta[a], Juan M. Corchado[a,b]

[a] Department of Computer Science and Automation, University of Salamanca, Spain
[b] Osaka Institute of Technology, Japan

ARTICLE INFO

Article history:
Received 10 February 2018
Revised 29 June 2018
Accepted 10 August 2018
Available online 10 January 2019

Keywords:
Web page categorization
Metric learning
Transfer learning
Deep learning

ABSTRACT

The growing amounts of online multimedia content challenge the current search, recommendation and information retrieval systems. Information in the form of visual elements is highly valuable in a range of web mining tasks. However, the mining of these resources is a difficult task due to the complexity and variability of images, and the cost of collecting big enough datasets to successfully train accurate deep learning models. This paper proposes a novel framework for the categorization of web pages on the basis of their visual content. This is achieved by exploring the joint application of a transfer learning strategy and metric learning techniques to build a Deep Convolutional Neural Network (DCNN) for feature extraction, even when training data is scarce. The obtained experimental results evidence that the proposed approach outperforms the state-of-the-art handcrafted image descriptors and achieves a high categorization accuracy. In addition, we address the problem of over-time learning, so the proposed framework can learn to identify new web page categories as new labeled images are provided at test time. As a result, prior knowledge of the complete set of possible web categories is not necessary in the initial training phase.

1. Introduction

Over the last decade, there has been an exponential increase in the webpages available online. The proliferation of blog-hosting and free content management systems (CMS) such as WordPress, Blogger or Tumblr have contributed to this growth as they made it possible for users with no experience in managing digital systems to share a variety of contents. The nature of these hosting services promotes the publication of multimedia contents, especially images and videos. However, as a result of the democratization of the Internet, new and challenging problems have emerged. Specifically, it is becoming increasingly difficult for users to find the content that they are looking for without having to go through related but undesired results.

In the recent years, these challenges have attracted the attention of numerous researchers [1–7]. Web page classification methods try to categorize web pages according to their subject. This categorization can later be used to assist tasks such as information retrieval, recommendation, parental filtering, focused crawling or contextual advertising [8]. Unfortunately, most of the current web page categorization approaches found in the literature focus solely on mining textual information and meta-data (e.g., plain text, hyperlinks, HTML structures, etc.). As a consequence, these methods neglect the valuable discriminative information present as multimedia content in websites, and often become language-specific. This limitation can be partially explained because mining multimedia content is a highly complex problem that frequently demands intensive computations and extensive training datasets in order to build effective models. In addition, the set of possible web page categories that a system must handle depends on the specific application and may vary over-time, which supposes a barrier for the application of traditional classification methods.

On the other hand, the field of artificial vision and specifically the sub-field of visual object recognition have experienced a major breakthrough after the general adoption of the deep learning paradigm in recent years [9,10]. Deep learning models are composed of a previously intractable number of processing layers, which allows them to learn more complex representations of the data by taking into consideration multiple levels of abstraction. This eventually led to a dramatic improvement in the state-of-the-art of visual object recognition, object detection and other related domains [11].

This paper builds upon the ideas and results presented in [12], where the authors explored the applicability of deep learning

* Corresponding author.
E-mail address: lope@usal.es (D. López-Sánchez).

https://doi.org/10.1016/j.neucom.2018.08.086
0925-2312/© 2019 Elsevier B.V. All rights reserved.

techniques to the problem of web page classification by adopting a transfer learning strategy. In this paper, we extend the framework proposed in [12] by also studying the use of different state-of-the-art metric learning techniques to improve the performance of distance-based classification on features extracted by a pre-trained Deep Convolutional Neural Network (DCNN). In addition, the use of metric learning techniques enabled us to address the problem of over-time learning, evaluating the ability of the proposed framework to learn to identify new web page categories as new labeled samples are provided. This feature increases the applicability of the proposed framework by avoiding the requirement of knowing the final set of possible web categories during the initial training phase and making the system compatible with online-learning scenarios and changes in the distribution and number of categories. In this regard, we introduce a novel algorithm to control the fine-tuning of our feature-extraction DCNN as additional labeled samples arrive during test phase. We also extended the database collected for the experiments in [12], making the results more reliable.

2. Related work

This section summarizes the most notable works in the literature. We first review the current state of web page categorization and then outline the relevant proposals in the fields of transfer learning and metric learning for neural networks.

2.1. Web page categorization

As mentioned above, web page categorization refers to the process of classifying web pages according to the subject of the contents present on them. Given the increasing amount of online web pages and the potential benefits of having an effective method for web page categorization, numerous researchers have focused on solving this problem by applying very different techniques. For this reason, it is not possible to cover all of them in this section. Nevertheless, the following paragraphs provide an overview of the advances, challenges and current trends in the field of web categorization.

Early works in the field addressed the problem of web page categorization as a simple document classification problem, using only features derived from the textual content of web pages [13]. Typically, webs pages were represented by *n*-gram frequency vectors, and classical classifiers were applied to these representations. Later on, methods were developed to take advantage of more web-specific features such as HTML tags and structures [14,15]. For instance, Know and Lee [15] proposed a web page classification method based on the *k*-Nearest Neighbors algorithm, where terms within different HTML sections are weighted differently.

More sophisticated methods were later designed to use not only the information present in the web page being analyzed, but also the information present in the webs referenced by hyper-links. For instance, Utard and Furnkranz [16] proposed using a portion of the text present in parent pages (i.e., pages linking the target page) such as the anchor text, the neighborhood of the anchor text or the paragraph containing the link, as well as the text on the target web page. Studies on this topic suggest that the most relevant information for classification can be extracted from sibling web pages[1] [17].

The most recent work in the topic of web page categorization has focused on specific limitations of the existing techniques. For instance, Abidin and Ferdhiana [6] proposed and algorithm for updating n-grams word dictionaries in the context of web page categorization. Their algorithm can be used to refine the *n*-gram thesaurus as new web pages become available for training, without degrading the classification accuracy. In this manner, the *n*-gram dictionaries used to generate the feature representation of web pages can be updated dynamically, adapting the feature representation to changes in the language used in web pages.

Another important limitation of text-based web classification systems which has been recently addressed in the literature is the language-dependent nature of the solutions. In this regard, Wang et al. [7] proposed using the large number of available labeled English web pages to help classify web pages in other languages, thus proposing a cross-language web page classifier. Other recent works have focused on the scalability of web categorization systems [5,18], classification based solely on the distribution of web elements [19], or the application of modern techniques such as topic modeling [4,20] and deep belief networks [2].

Finally, some recent works have shown the potential of analyzing the visual contents of web pages for web page categorization. In [12], the authors addressed the problem of web categorization by using the features extracted by a pre-trained DCNN in conjunction with simple classifiers. Similarly, in [21] the authors proposed combining the pre-trained DCNN with a simple metric learning technique to improve the accuracy of the system in the context of the Case-Based Reasoning (CBR) methodology.

2.2. Deep transfer learning and metric learning

Over the past years, the field of computer vision has witnessed a major breakthrough thanks to the development of the Deep Learning paradigm [11]. Deep Learning models have drastically changed fields such as object detection [22], speech recognition [23] or activity recognition [24]. Deep learning relies on the availability of large training datasets to build very complex models (composed of millions of parameters) that effectively benefit form the volume of training data and achieve unprecedented accuracy rates on a range of tasks. However, collecting such datasets can be an expensive and not always feasible task. As a consequence, an alternative strategy called transfer learning has become increasingly popular by enabling researchers and practitioners to re-purpose the weights learned by other networks in their own application domains [23,25,26].

Metric Learning has also gained attention within the deep learning community during the recent years. Early work on this topic provided a means to train neural networks to produce a useful feature space rather than a final label prediction [27]. The representation of samples generated by the network can then be used with a distance-based classifier to classify test samples. Two of the most popular metric learning techniques for neural network training are DrLIM [27] and Triplet-Loss [28]. Originally presented in 2006, DrLIM has been the object of a renewed interest after the general adoption of the Deep Learning paradigm. In fact, the DrLIM method or slightly modified versions of it have been recently applied with great success in a range of domains including visual place recognition [29], clickbait detection [26] or product image retrieval [30] among others [31–33]. More recently proposed, Triplet-Loss also enjoys a notable popularity, and has been recently used for speaker change detection [34], image-based clothes recognition [35] and sketch-based image retrieval [36]. One major advantage of metric learning techniques for neural networks lies in their ability to learn new classes without the need to re-train the network, just by adding the representation of new samples generated by the network to the current sample database. In addition, the networks can be fine-tuned as new labeled samples become available to adapt themselves to changes in the classification problem [26].

Aside from neural networks, metric learning is also an active topic of research within the machine learning community

[1] In this context, the term sibling web page refers to a web page that is linked by a parent web page that also linked the target web page.

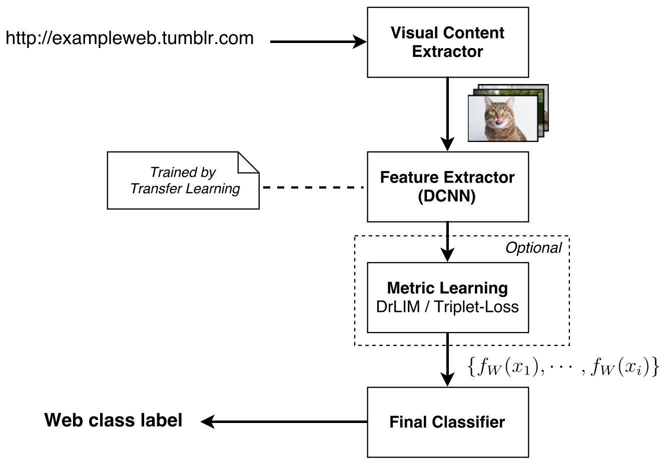

http://exampleweb.tumblr.com
Visual Content Extractor
Trained by Transfer Learning
Feature Extractor (DCNN)
Optional
Metric Learning DrLIM / Triplet-Loss
$\{f_W(x_1), \cdots, f_W(x_i)\}$
Web class label
Final Classifier

Fig. 1. Overview of the proposed framework's processing steps, from the input of the system (URL of a web page) to the output (predicted category).

[37]. Most approaches in this field formulate the metric learning problem as learning pairwise real-valued metric of the form $d_{\mathbf{M}}(x, x') = \sqrt{(x - x')^\top \mathbf{M}(x - x')}$ where $\mathbf{M} \in \mathbb{R}^{d\times d}$ is a symmetric positive semi-definite matrix (i.e., a Mahalanobis distance). Note that one can express $\mathbf{M}$ as $\mathbf{L}^\top \mathbf{L}$ where $\mathbf{L} \in \mathbb{R}^{k\times d}$ and k is the rank of $\mathbf{M}$. As a consequence, it is possible to rewrite $d_{\mathbf{M}}(x, x')$ as follows:

$$\begin{aligned} d_{\mathbf{M}}(x, x') &= \sqrt{(x - x')^\top \mathbf{M}(x - x')} \\ &= \sqrt{(Lx - Lx')^\top (Lx - Lx')} \end{aligned} \quad (1)$$

Therefore, learning a Mahalanobis distance is equivalent to learning a linear projection of the data (defined by the projection matrix $\mathbf{L}$) and then using the Euclidean distance in the resulting feature space. The different Mahalanobis metric learning algorithms mainly differ in the way they learn the matrix $\mathbf{M}$. However, they are somehow limited by the linear nature of the transformation they learn. In this paper, we use a number of representative methods of this type as a baseline for comparison with our proposed model. In particular, we evaluate our approach against MLEV-G [38], KISSME [39], OASIS [40] and ITML [41]. Henceforth, we will use the term Mahalanobis metric learning to refer to these methods, in contrast to the other evaluated metric learning techniques specially designed to train neural networks.

3. Proposed framework

This section details the proposed framework. First, the global structure and elements of the processing pipeline are outlined, followed by a detailed description of each module in the framework. Our system is designed to perform the following task: given a URL, the system must (1) access that URL, extract all the images available on the web page, and filter those that do not contain any discriminative information; (2) extract a feature descriptor from each image such that the classification problem becomes easier over that feature space; and (3) analyze each feature descriptor and combine the results to emit a prediction concerning the category of the entire web page. As shown in Fig. 1, the proposed framework consists of four major processing modules:

1. **Visual content extractor**. This module deals with the extraction and pre-processing of the images present in the input URL to be classified.
2. **Feature extractor**: The feature extraction module, based on a DCNN, generates a high-level feature descriptor for each of the extracted images, easing the classification process.
3. **Metric learning** (optional): The optional metric learning module adapts the final layers of the feature extraction network to make the generated image-descriptors more suitable for distance-based classification, improving the accuracy of subsequent k-Nearest Neighbors (k-NN) classification.
4. **Final classification module**: this module utilizes different classification algorithms to analyze the feature descriptors extracted from the web page, assigning a class label to each descriptor and combining them to categorize the complete web page.

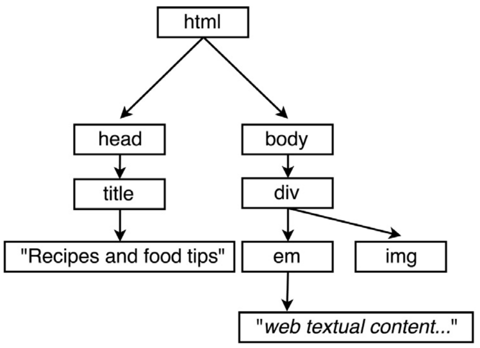


Fig. 2. Example of Document Object Model (DOM) tree representation for a very simple web page. The branches of the trees are explored recursively to find all the image elements.

3.1. Multimedia content extractor

The first module of the proposed framework is in charge of extracting all the images that are present on a web page and filtering the ones that do not contain useful information (i.e. navigation icons, advertisements, banners, etc.). To do this, a web-scrapping approach was used. In this approach, the system begins by downloading the HTML document that corresponds to the provided URL. Then, the Beautiful Soup library [42] is used to analyze the structure and the HTML hierarchy in the document. After this, the web page is represented by a tree structure called Document Object Model (DOM) whose leaves are the elements of the document (e.g. titles, links, images, etc.). Fig. 2 shows the DOM of a very simple web page.

Once a DOM has been built for a given web page, it is explored exhaustively in the search for images. The URLs of those images are stored and later used to download the pictures. Several criteria can be applied to filter the extracted images. For example, it is possible to discard the images that contain a specific group of keywords in the *alt* attribute (e.g., we might discard images that contain the word “advertisement” in its *alt* attribute). Another possible approach consists of rejecting the images whose dimensions are outside a specific range or aspect ratio. This is because images with extreme proportions do not usually contain discriminative information. For example, very small images might correspond to navigation icons, while those with an elongated shape tend to be advertisement banners. In our experiments, the system was configured to discard images whose height or width was lower than 100*px* or whose aspect ratio (i.e., height divided by width) was outside the range [0.5, 2].

3.2. Deep feature extractor

Once a number of images have been extracted from a web page, it would be possible to directly apply any classification method. However, the complexity of the image recognition problem that we are trying to solve demands a large number of training instances and a very complex model that can tackle the difficulty of the

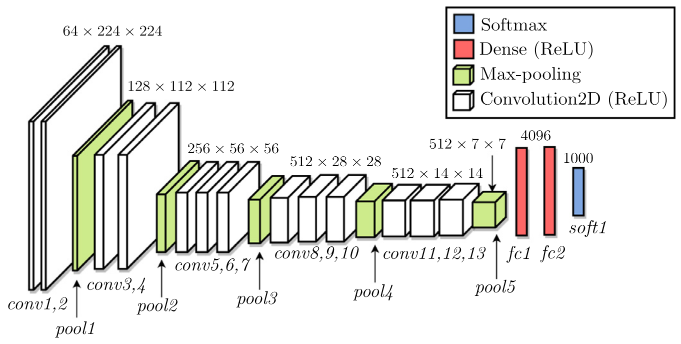


Fig. 3. Architecture of the VGG16 Deep Convolutional Neural Network and names of the layers. The shown layer sizes are use the channels-first format. The standard image input-size is $3 \times 224 \times 224$.

classification problem. This is mainly due to the high intra-class variability of the images from such artificial vision problems. Collecting such an extensive dataset can be a tedious and very time-consuming task. In addition, the time it requires to train such a classifier would be prohibitively long even if complex computation parallelization techniques and expensive specialized hardware were used. To overcome these limitations, we propose the use of a strategy known as transfer learning (see [43] for a recent survey on this strategy). The key idea of this approach is to solve a new problem using the knowledge gained from solving a previous interconnected problem. In fact, this approach has been applied with great success in the context of artificial neural networks and deep learning [25,44]. In the case of a neural network, some of its layers are initialized with the weights learned by another network that was trained to solve a different but somehow related problem. Then, the resulting network is used either as an off-the-shelf feature extractor to train simple classifiers [25], or as part of a bigger network that will be fine-tuned to solve the target problem [44].

In the context of our framework, we propose using a pre-trained model of DCNN as a feature extractor for simple classifiers. Specifically, the selected model is the VGG16 DCNN [45] developed by the Visual Geometry Group at the University of Oxford. The VGG16 model was originally trained on the ImageNet dataset, which consists of 14 million images belonging to 1000 categories. The VGG team scored a 92.7% top-5 test accuracy on this benchmark[2], achieving the second place in the classification task of the Large Scale Visual Recognition Challenge (ILSVRC) 2014 [9]. Although several models have outperformed VGG16 in the recent years (e.g., [47,48]) this model remains competitive with the state-of-the-art, and is often the model of choice for transfer learning (e.g., [49]). Mainly, we selected VGG16 for our experiments because of two reasons: (1) It maintains a good balance between computational cost and accuracy as compared to other popular architectures [50], and (2) Its architecture is remarkably simple as it contains only the widely used convolution/max-pooling blocks followed by some fully-connected layers and a final softmax layer, which makes VGG16 a paradigmatic example of the success of DCNNs. Fig. 3 shows the overall architecture of the network. We will refer to this figure to name each of the layers of the model in the experimental results section.

In the proposed framework, the network is cut at a specific layer and the activations of the neurons in that layer are used as a representation of the input image. A simpler classifier is then applied over that feature space. It has been proved that the outermost layers provide a more abstract and compact representation of the input images [44]. However, the final layers are more task-dependent and might not be appropriate if the target problem we want to address is very different from the problem the network was originally trained to solve. In general, the layer that produces the most suitable representation for a task must be determined empirically. To this end, the accuracy rates obtained with features from different layers and various classification methods are reported in Section 4. The next section describes the different classification methods we used for this task.

3.3. Selected classification algorithms

After the high level features have been extracted from images using the DCNN, a simpler classifier is in charge of emitting the final class prediction. Provided that the features are sufficiently abstract, the use of simple and efficient linear classifiers is adequate [25]. In addition, using simple classification models can prevent over-fitting, especially in scenarios where little training information is available. Nevertheless, we also evaluated a simple nonparametric classifier that is not strictly a liner classifier, namely the k-Nearest Neighbors (k-NN) algorithm. Additionally, k-NN was evaluated with $L2$ unit-length normalization of the feature descriptors extracted by the DCNN. The reason for which k-NN is included in our comparative is that, thanks to its lazy learning nature, it is naturally compatible with over-time learning as explained in the following sections. The following linear classifiers were evaluated:

1. Linear Support Vector Machine (L-SVM), as implemented in LIBLINEAR [51]. This classifier's decision function is:
$$h(x) = sgn(w^T x + b) \tag{2}$$
Where w is calculated by solving the primal optimization problem with $L2$ penalty:
$$\min_{w} \quad \frac{1}{2} w^T w + C \sum_{i=1}^{n} \max(1 - y_i(w^T x + b), 0)^2 \tag{3}$$
The multi-class support is obtained following a one-vs-the-rest scheme.
2. Logistic Regression (LR) as implemented in LIBLINEAR [51]. The decision function of this method is the following[3]:
$$h(x) = \frac{1}{1 + e^{-\mu}} \quad \text{where} \quad \mu = w^T x + b \tag{4}$$
Here, the multi-class support was obtained using a one-vs-the-rest scheme.
3. Perceptron with linear activation function as implemented by scikit-learn [52].

The majority of the web pages we analyzed, contained several images with relevant information. All the images extracted from a web page must be considered when making a prediction of the category of the web page as a whole. In our experiments, a simple vote aggregation approach is taken where the final prediction is the most common label among the images of the web. When a tie occurs, it is solved at random. While our experimental results show that this approach if very effective when using linear classifiers such as Support Vector Machines and Logistic Regression, we also obtained relatively poor results when using a distance-based classifier directly on the features extracted by VGG16, especially when the L2 normalization was not enabled. This motivated us to explore the applicability of metric learning techniques in improving the feature descriptors generated by the DCNN.

[2] The 92.7% accuracy obtained by the VGG team at the ILSVRC 2014 classification challenge was in fact the result of using an ensemble of 7 VGG16-like networks (see [45] for details). The single, pre-trained VGG16 network instance used in our experiments achieves a 90.1% top-5 accuracy on the ImageNet validation dataset [46].

[3] Note that despite the nonlinearity of the decision function, the decision boundary $\{x : h(x) = 0.5\}$ is a hyperplane and therefore this classifier is considered to be linear.

3.4. Application of metric learning methods

As briefly described in Section 2.2, the term Metric Learning (ML) in the context of deep learning refers to the set of techniques that enable the training of neural networks to output a useful representation of samples rather than class-label predictions. In particular, most neural network metric learning techniques try to learn a feature representation where Euclidean distances become a meaningful similarity measure for distance-based classification or retrieval.

In the context of our framework, we propose the application of metric learning in conjunction with transfer learning to improve the performance of distance-based classification over the features generated by the DCNN. Specifically, metric learning is applied to fine-tune the final layers of the pre-trained DCNN. The reason for using ML to adjust the final layers of our pre-trained DCNN is twofold. The first reason is that training the final layers of the DCNN will improve the discriminative information present in the generated descriptors, thus increasing the accuracy obtained by a k-NN classifier trained on them. Secondly, as explained in the next section, the use of metric learning to adapt the final layers will preserve the natural compatibility of our feature-extraction DCNN with over-time learning. Therefore, in a real application environment the system could be trained on an initial set of categories by collecting a relatively small training set (in our experiments we use ~ 130 images per class) and then new classes may be learned over-time by providing the system with new labeled samples at test time. Conveniently, since only a few layers at the end of the network's architecture will be trained, it is not necessary to have a large number of training samples and training times are in the order of minutes instead of hours or days. For our experiments, we selected two of the most popular metric learning techniques to train neural networks, namely DrLIM [27] (often referred to as the Siamese network method) and the Triplet-Loss method [28].

Even though the original DrLIM method was proposed more than a decade ago [27], it has gained in popularity with the irruption of the deep learning paradigm. Moreover, nowadays it remains as one of the preferred choices for metric learning with neural networks [31,53]. With DrLIM, a parametrized function is optimized in such a way that samples with the same class label are placed nearby in the output representation, whereas samples with different class labels are pushed far away from each other. As a consequence, samples in the output representation get arranged in clusters according to their class label, easing the classification based on Euclidean distances. To achieve this, DrLIM performs the training on a set of sample-pairs that are generated by pairing samples from the original training dataset. Particularly, let the initial training dataset be represented by a set of sample/label pairs $\{(x_1, y_1), \dots, (x_n, y_n)\}$. DrLIM then works by choosing sample pairs from the training dataset alongside a pair-label y, which takes the value of one when both samples in the pair share the same class label and zero otherwise. Originally, the authors proposed generating the complete space of pairs by exhaustively pairing each sample with all the other samples in the training dataset, resulting in $\binom{n}{2}$ pairs for a training set with n samples [27]. However, a more efficient training strategy can be adopted by using Mini-Batch Stochastic Gradient Descent (MB-SGD) and generating pairs online to complete each bath [31]. In our experiments, the following steps were taken to form each batch: (1) Select a sample at random from the training dataset, (2) Couple the selected sample with another randomly selected sample with the same class label, adding the pair to the batch with a pair-label $y = 1$, (3) Couple the selected sample with another randomly selected sample that has a different class label, adding the pair to the batch with a pair-label $y = 0$, and (4) Repeat steps 1–4 until the desired batch size is reached. By generating the batches in this manner, we ensure that they contain a balanced number of similar/dissimilar pairs, with could otherwise cause the output representation to collapse. Formally, a training batch consists of a set containing b pairs with their corresponding pair-labels:

$$batch = \{(x_1, x_2, y)^{(1)}, \dots, (x_1, x_2, y)^{(b)}\} \tag{5}$$

To train a network $f_W(\cdot)$ parametrized by the set of weights W, DrLIM begins by conceptually duplicating the network to simultaneously process the two samples of a pair from the batch. Note that, even though the network is duplicated the weights of both copies are tied. The Euclidean distance is then computed between the resulting representations $f_W(x_1)$ and $f_W(x_2)$:

$$D_W(x_1, x_2) = ||f_W(x_1) - f_W(x_2)||_2 \tag{6}$$

For convenience, $D_W(x_1, x_2)$ is henceforth abbreviated as D_W. Based on this distance measure, DrLIM defines the Contrastive Loss:

$$\begin{aligned} L(W, (x_1, x_2, y)) = \; & (1 - y)\frac{1}{2}(D_W)^2 \\ & + (y)\frac{1}{2}\{max(0, \alpha - D_W)\}^2 \end{aligned} \tag{7}$$

Intuitively, this loss function penalizes distances between the output representations of samples from the same class. Similarly, it rewards up to a certain margin α the distances between the representations of samples with dissimilar class labels. The total batch loss is computed as the sum of all the pairs's losses in the batch:

$$\mathcal{L}(W) = \sum_{i=1}^{b} L(W, (x_1, x_2, y)^{(i)}) \tag{8}$$

The use of this loss to optimize the weights of the network produces the above mentioned class-clustering effect. Fig. 4.a provides a schematic overview of the training process of a network $f_W(\cdot)$ with DrLIM and MB-SGD.

More recently proposed, the Triplet-Loss method [28] tries to ensure that the representation of each training sample is closer to the representations of all other samples of the same class than it is to the representation of any sample from a different class. To this end, the training is performed on triplets of samples rather than pairs. Each triplet contains an anchor sample x^a, a sample x^p that comes from the same class as the anchor, and a sample x^n from a different class to the anchor (see Fig. 5). x^p and x^n are commonly referred to as the positive and negative samples in the triplet. Again, rather than exhaustively generating all possible triplet combinations from the training dataset, the authors proposed using an online generation method combined with MB-SGD. First, the training dataset is sampled at random to select b anchor samples, where b is the desired batch size. Then, instead of choosing the positive/negative samples at random, hard negative/positive samples are selected from within the batch. In particular, for each selected anchor x^a, the corresponding positive sample x^p is selected from within the batch such that:

$$\arg\max_{x^p} ||f_W(x^a) - f_W(x^p)||_2^2 \tag{9}$$

Similarly, the negative pairs are selected such that:

$$\arg\max_{x^n} ||f_W(x^a) - f_W(x^n)||_2^2 \tag{10}$$

Intuitively, this selects positive samples that are too far away from the anchor, and negative samples that are too close to the anchor. Note that this selection process involves processing the samples in the batch with the most recent version of the network $f_W(\cdot)$, thus being significantly more expensive than the pair selection process of DrLIM. Once the batch has been built, it can be represented as:

$$batch = \{(x_p, x_a, x_n)^{(1)}, \dots, (x_p, x_a, x_n)^{(b)}\} \tag{11}$$

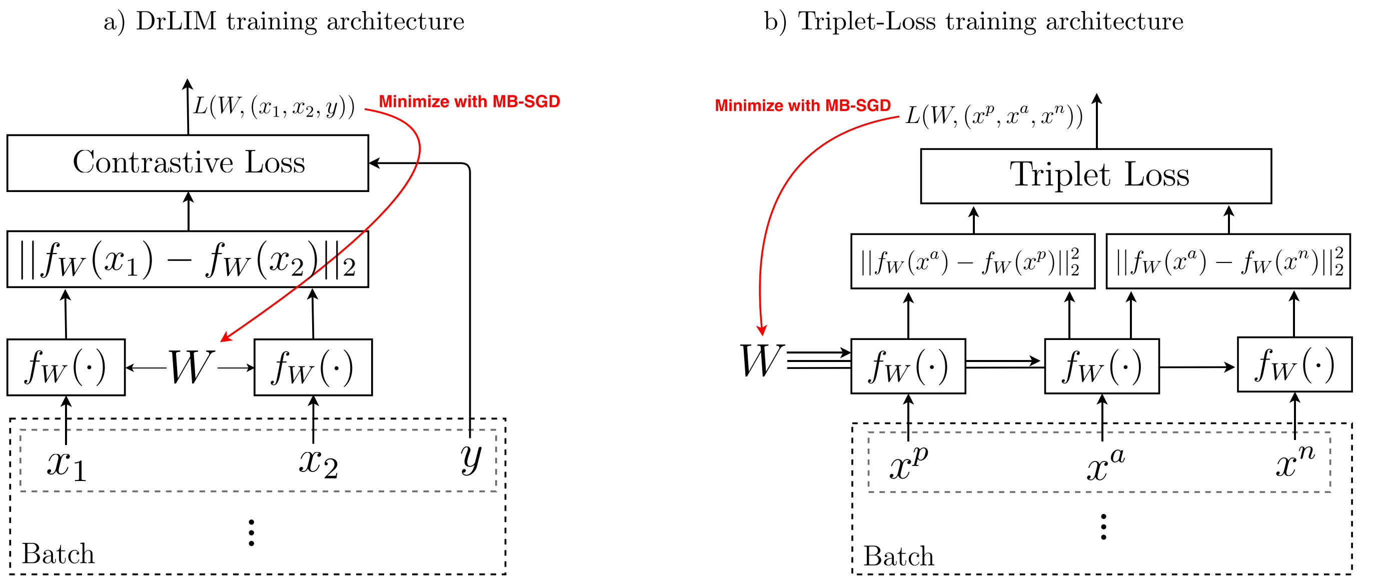


Fig. 4. Schematic view of the training process for a network $f_W(\cdot)$ with DrLIM (left) and the Triplet-Loss method (right) using Mini-Batch Stochastic Gradient Descent (MB-SGD). The total batch loss is calculated as the loss-average for all the training pairs/triplets in the batch.

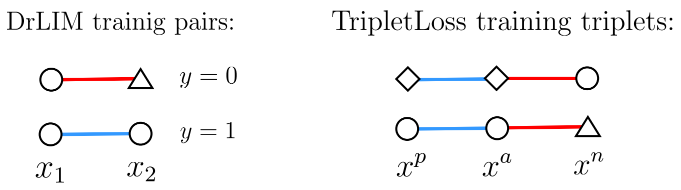


Fig. 5. Schematic view of training pairs for DrLIM (left) and training triplets for Triplet-Loss (right). The shapes of the samples indicate the corresponding class labels, and the color of links the similarity/dissimilarity in class labels.

Then, the Triplet Loss is applied to each triplet in the batch:

$$L(W,(x_p,x_a,x_n)) = \left[||f_W(x^a)-f_W(x^p)||_2^2 - ||f_W(x^a)-f_W(x^n)||_2^2+\alpha\right]_+ \qquad (12)$$

This loss forces the distance between the anchor and the negative sample to exceed the distance between the anchor and the positive sample by at least a margin α. As with DrLIM, the total batch loss is the sum of the losses for each triplet in the batch:

$$\mathcal{L}(W)=\sum_{i=1}^{b} L(W,(x_p,x_a,x_n)^{(i)}) \qquad (13)$$

Thus far, we have described the selected metric learning methods. However, we still need to define how these methods will be applied in conjunction with the transfer learning strategy adopted by our framework. Usually, metric learning methods such as Triplet-Loss and DrLIM are used to train an entire deep network from scratch. However, in the context of the proposed framework we decided to use a pre-trained model to avoid the need for a large training dataset and excessive training times. Moreover, trying to train from scratch or fine-tune a complete DCNN with very scarce data would most likely result in over-fitting. To avoid these issues and preserve the advantages of both transfer and metric learning, we just adapt the final layers of the pre-trained model. Concretely, we compare two modes of adapting the final layers of the pre-trained VGG16 model used in our framework:

- Mode A. The pre-trained VGG16 model is chopped at the selected layer. Then, a new fully-connected layer is appended at the end of the resulting network. Only the final fully-connected layer is trained by the selected metric learning method, while all the previous layers remain fixed in all experiments.
- Mode B. The pre-trained VGG16 model is chopped at the selected layer. Then, a new fully-connected layer is appended at the end of the resulting network. The last two layers of the network are trained by the selected metric learning method, while all the previous layers remain fixed in all experiments.

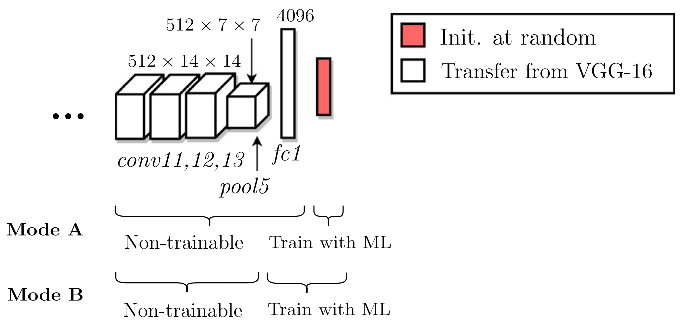


Fig. 6. Illustration of modes A and B for adapting the final layers of the pre-trained VGG16 model by using Metric Learning (ML) training algorithms.

Fig. 6 illustrates both network adaptation modes when selecting $fc1$ as the cut-off layer.

Once the feature extraction network $f_W(\cdot)$ has been trained, it is used to generate the feature representation for the training samples and a k-Nearest Neighbors classifier (k-NN) is initialized with those descriptors. Formally, given a training dataset of labeled images $DB=\{(x_1,y_1),\dots,(x_n,y_n)\}$ the neighbor database of k-NN is initialized as $DB_f=\{(f_W(x_1),y_1),\dots,(f_W(x_n),y_n)\}$. Even though we characterize the neighbor database as a set of descriptor/label pairs, any space-partitioning data structure (e.g., k-d trees, ball trees, etc.) can be applied to reduce query times.

When a test image x_{test} is presented to the system, it is first processed by the network yielding $f_W(x_{test})$, and the k-nearest neighbors to $f_W(x_{test})$ are found in DB_f. Then, the predicted class label for x_{test} is the majority class among the k retrieved neighbors. As when using other classifiers, the predicted class label for a complete web page is the most frequent class label among the images extracted from that page.

3.5. Over-time learning

As mentioned before, one of the main advantages of metric learning techniques used to train neural networks is that, since the network generates feature descriptors rather than final label predictions, new labeled samples can be incorporated to the k-NN database dynamically. This enables the system to learn over-time, adapting itself to changes in the classification problem [26] such as the emergence of new classes.

Usually, metric learning-based systems deal with over-time learning by simply adding the representation of new labeled samples to the k-NN database. This approach relies on the assumption that the feature representation learned by the network on the training dataset will be meaningful enough to correctly characterize new classes which arise over time. This assumption might be partially true for problems such as handwritten digit recognition [31], where all the classes can be characterized by different configurations of a set of common base features (e.g., straight lines, curves, circles, etc.). For instance, the feature representation learned to classify handwritten letters might be sufficiently discriminative to classify handwritten digits without further tuning of the feature-extraction network. However, in our target problem, each class is characterized by a specific set of features and there is strong intra-class variance. As a consequence, doing some fine-tuning of the network as new labeled samples become available may benefit the accuracy of the system.

In this section, we describe the over-time learning mechanisms of the proposed framework. First, we introduce over-time learning without fine-tuning of the feature extraction network, and then we present an algorithm to control the fine-tuning of the network to keep computational overheads low.

3.5.1. Over-time learning without fine-tuning

The simplest way to enable over-time learning with a metric-learning neural network consists in adding the feature descriptors generated by the network for new labeled samples to the k-NN database (DB) without any adaptation or fine-tuning of the network's weights after the initial training phase. At a particular point in time, our web categorization system can be fully characterized by the feature-extraction neural network $f_W(\cdot)$, the available labeled images:

$$DB = \{(x_1, y_1), \ldots, (x_n, y_n)\} \tag{14}$$

and the k-NN database generated by passing the available labeled images through the network:

$$DB_f = \{(f_W(x_1), y_1), \ldots, (f_W(x_n), y_n)\} \tag{15}$$

To save storage space, images are not stored even if we plan to perform fine-tuning latter. Instead, it suffices to store the representation of those images at the last non-trainable layer of the feature-extraction network.

Supposing that a new sample x_{new} and its class label y_{new} are provided to the system, it will be able to learn this sample by simply incorporating the descriptor generated from this image to the k-NN database:

$$DB_f \leftarrow DB_f \cup \{(f_W(x_{new}), y_{new})\} \tag{16}$$

Even if x_{new} belongs to a new, previously unseen class, the system will from then on be potentially able to recognize images from that class. Of course, the accuracy for a new category will most likely increase as more images of that class are added to the k-NN database. In must also be noted that this learning process of new classes comes with a cost. Intuitively, the overall accuracy of the system will decrease as new classes are included in the classification problem.

3.5.2. Over-time learning with fine-tuning

One way to mitigate the impact of learning to recognize new classes in the overall accuracy is to fine-tune the weights of the feature representation network as new labeled samples become available. In this way, the overlapping between the new learned classes and the existing ones will be lessened, so the negative impact in the overall accuracy will be lower. In addition, as shown by the experimental results presented in Section 4.4, the fine-tuning control Algorithm presented in this section enables higher recognition accuracies in the over-time learned classes.

A naive fine-tuning methodology would be to fine-tune the final layers of $f_W(\cdot)$ by executing a fixed number of training iterations as new samples arrive. However, executing even a few training iterations each time the system is presented with a new sample/label pair might be too expensive and potentially lead to over-fitting. To avoid this, we propose an algorithm that limits the fine-tuning to the cases where the new sample contains new discriminative information. In particular, the proposed algorithm triggers the fine-tuning process of the network when an incoming sample is misclassified, and limits the process to the number of iterations needed to ensure a correct classification of the new sample. This also avoids the problem of fixing the number of iterations manually. The proposed fine-tuning triggering algorithm is described in Algorithm 1.

Algorithm 1 Over-time fine-tuning control algorithm.

Require: The current feature-extraction network $f_W(\cdot)$, the database of samples and class labels DB, and the descriptor database DB_f. The new training sample and its correct label (x_{new}, y_{new}).

```
1:  kNN ← fit_kNN($DB_f$)
2:  $y_{pred}$ ← kNN.predict_class($f_W(x_{new})$)
3:  while $y_{pred} \neq y_{new}$ do
4:      Execute one training iteration for $f_W(\cdot)$
        over $DB \cup \{(x_{new}, y_{new})\}$
5:      $DB_f \leftarrow \{(f_W(x_1), y_1), \ldots, (f_W(x_n), y_n)\}$
6:      kNN ← fit_kNN($DB_f$)
7:      $y_{pred}$ ← kNN.predict_class($f_W(x_{new})$)
8:  end while
9:  $DB_f \leftarrow DB_f \cup \{(f_W(x_{new}), y_{new})\}$
10: $DB \leftarrow DB \cup \{(x_{new}, y_{new})\}$
```

We also noted that, to reduce the number of iterations needed to fine-tune the DCNN and learn to classify new classes with Algorithm 1, it helps forcing all the available samples from class y_{new} to be present in each training batch. Section 4.4 presents experimental results comparing the capabilities of our framework to learn a new class over-time with and without fine-tuning enabled.

4. Experimental results

This section presents experimental results evidencing the ability of the proposed system to effectively classify web pages based on their visual content. First, we describe the dataset collected for our experiments. After that, the classification accuracy of the different algorithmic alternatives presented in Section 3 is evaluated for both individual image classification and complete web page categorization tasks. We then try to provide additional insight into the suitability of the representations generated by the feature-extraction DCNN by using a 2D data visualization technique. Finally, we evaluate the ability of our framework to learn a new category without degrading the classification accuracy for the already known classes, as new labeled images are provided.

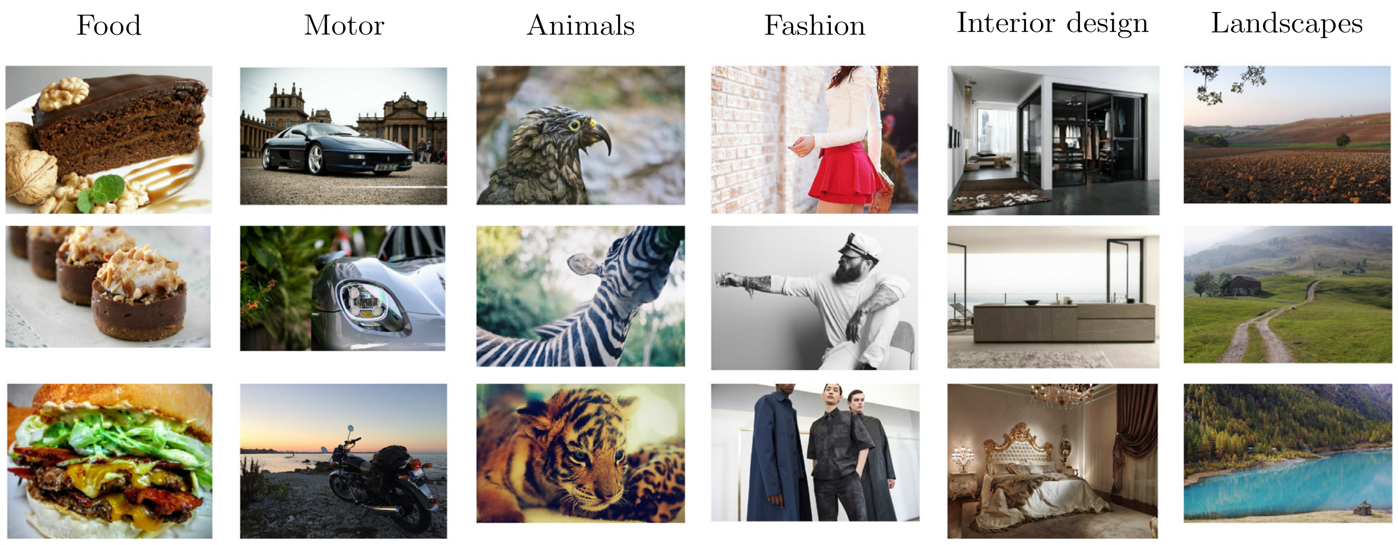


Fig. 7. Some sample images from each class. As expected, the artificial vision problem that emerged was of significant complexity, as the dataset exhibits a high intra-class variability. Also, the extracted images display in a wide range of resolutions and proportions.

4.1. Dataset creation

For our experiments, we extended the web page classification dataset used in [12] by collecting additional web pages. In particular, the extended dataset consists of a total of 365 web pages distributed uniformly among six categories, namely "*Food and cooking*", "*Motor and races*", "*Interior design*", "*Animals*", "*Fashion*" and "*Landscapes*". Fig. 7 shows some sample images from each class. At the time of being collected, these web pages contained a total of 4027 images. Both web pages and individual images were manually labeled by a human annotator. For our experiments, a random train/test split was arranged, resulting in 71 training web pages containing 784 images and 294 test web pages containing 3243 images (i.e., approximatelly 20/80%).

4.2. Classification accuracy results

Our first set of experiments evaluates the classification accuracy of both individual images and complete web pages for the different algorithmic alternatives described in Section 3. First, as explained in Section 3.3, different simple classification algorithms were fitted on the images extracted from the training web pages, using the features at various depth levels of the pre-trained VGG16 model. The image-level classification accuracy was then evaluated on the images extracted from the test web pages, and the web page-level test accuracy was also determined. The resulting accuracies are shown in Table 1. For completeness, we also evaluated the selected classifiers when trained on two state-of-the-art hand-crafted feature descriptors, rather than using the DCCN for feature extraction. These descriptors where Fisher Vector (FV) [54] and the Vector of Aggregated Local Descriptors (VLAD) with power-law normalization [55]. For both FV and VLAD descriptors the selected vocabulary size was 10, resulting in a descriptor of size of 771 and 1280 respectively.

Table 2 displays the image-level and web page-level classification accuracies for our framework using the metric learning techniques as described in Section 3.4. Note that Mode B accuracies are only provided for *fc2* and *fc1* layers. This is because the other considered layers (i.e., *pool5-2*) have no trainable parameters, so when cutting the DCNN at them Mode A is equivalent to Mode B. Also note that the hyper-parameter out_n denotes the number of neurons in the final layer of the network. For al the models evaluated in this section, the free hyper-parameters were selected by a grid search process over typical values with 5-fold cross-validation over the training images. For convenience, the tables only report the selected hyper-parameters for the image-level accuracy results, but note that these are the same for the web page-level accuracy results. As a baseline for comparison, Table 3 shows the accuracies obtained with a number of representative Mahalanobis metric learning algorithms, namely MLEV-G [38], KISSME [39], OASIS [40] and ITML [41]. As in [39], we used PCA to reduce the dimensionality of the feature descriptors extracted by the DCNN prior to metric learning, so the computations would remain tractable.

As we can see in Table 1, the best performing method when ML adaptation of the final layers of the DCNN was not enabled was Logistic Regression trained on *fc2* features. This method achieved a 91.45% accuracy for single image categorization and a 98.29% accuracy for web page categorization. As in [12], the *k*-NN classifier exhibited a poor performance when trained directly on the features extracted from the DCNN. However, a significant improvement was registered by normalizing the image descriptors to L2 unit-length. Nevertheless, the accuracy of *k*-NN even after normalization was far below that of simple linear classifiers such as L-SVM and Logistic Regression. This justifies our proposal of applying ML techniques to adapt the final layers of the DCNN to make it more suitable for distance-based classification.

As expected, hand-crafted feature descriptors performed rather poorly. The best accuracy on web page categorization with hand-crafted feature extractors was achieved by using L-SVM with VLAD features, resulting in a mere 73.46% accuracy. This poor performance can be partially explained by the fact that, as opposed to our DCNN-based approach, FV and VLAD cannot take advantage of the transfer learning strategy, thus relying completely on the scarce amount of provided training data. Moreover, hand-crafted feature descriptors have been consistently outperformed by deep learning models in the recent literature [25].

Looking at Table 2, we can see that our proposal of using ML techniques for adapting the final layers of the DCNN consistently improved the accuracy of distance-based classification on the generated descriptors. Moreover, using the Triplet-Loss method with Mode B at *fc2* resulted in the best accuracy registered in our experiments, with a 91.85% accuracy for individual image categorization and a 98.97% accuracy for web page categorization. In contrast

Table 1
Accuracy rates on image-level classification and web categorization for different simple classifies, trained on features extracted at different depths of the pre-trained VGG16 DCNN model and on two state-of-the-art hand-crafted feature descriptors. (*) indicates $L2$ normalization of the image-descriptors prior to classification.

Features	Image-level accuracy					Webpage-level accuracy				
	L-SVM	LR	Percep.	k-NN	k-NN*	L-SVM	LR	Percep.	k-NN	k-NN*
fc2	91.24%	**91.45%**	90.81%	82.60%	90.28%	98.29%	**98.29%**	97.27%	88.77%	97.27%
fc1	90.90%	91.24%	90.22%	57.29%	87.97%	97.95%	97.27%	97.61%	63.94%	96.25%
pool5	90.16%	90.19%	90.16%	16.80%	90.47%	97.95%	97.61%	98.29%	14.96%	97.61%
pool4	84.11%	84.55%	83.47%	16.89%	81.89%	94.84%	95.23%	95.57%	15.64%	94.55%
pool3	74.12%	74.93%	69.75%	25.31%	64.63%	91.15%	90.47%	86.39%	29.93%	78.23%
pool2	62.04%	62.84%	62.53%	21.21%	29.07%	82.31%	85.03%	83.33%	25.17%	26.53%
Param.	$C = 0.01$	$C = 0.1$	$iter = 5$	$k = 5$	$k = 5$					
FV [54]	50.97%	48.87%	37.00%	40.54%	40.54%	72.10%	68.02%	42.85%	55.44%	55.44%
VLAD [55]	50.53%	51.37%	49.39%	41.07%	41.07%	73.46%	72.44%	70.74%	58.50%	58.50%
Param.	$C = 1$	$C = 1$	$iter = 5$	$k = 5$	$k = 5$					

Table 2
Accuracy rates on image-level classification and web categorization for a k-NN classifier, trained on features extracted at different depths of the pre-trained VGG16 DCNN with the final layers adapted by means of ML, as described in Section 3.4.

Cut VGG16 at	Image-level accuracy				Webpage-level accuracy			
	DrLIM Mode A	DrLIM Mode B	Triplet Mode A	Triplet Mode B	DrLIM Mode A	DrLIM Mode B	Triplet Mode A	Triplet Mode B
fc2	90.87%	90.25%	91.27%	**91.85**%	97.95%	98.63%	97.61%	**98.97**%
fc1	89.39%	86.67%	87.94%	91.48%	97.61%	96.93%	96.59%	97.95%
pool5	89.30%	–	85.66%	–	97.27%	–	96.25%	–
pool4	78.10%	–	79.24%	–	92.51%	–	93.19%	–
pool3	54.79%	–	69.56%	–	73.80%	–	88.77%	–
pool2	46.99%	–	60.99%	–	65.64%	–	80.33%	–
Param.	$lr = 0.005$ $batch = 100$ $iter = 1000$ $out_n = 100$	$lr = 0.005$ $batch = 100$ $iter = 1000$ $out_n = 100$	$lr = 0.001$ $batch = 200$ $iter = 750$ $out_n = 200$	$lr = 0.0005$ $batch = 200$ $iter = 750$ $out_n = 300$				

Table 3
Accuracy rates on image-level classification and web categorization for Mahalanobis metric-learning algorithms followed by k-NN classification, trained on features extracted at different depths of the pre-trained VGG16 DCNN model. $L2$ normalization of the image-descriptors generated by VGG16 was performed in all cases, followed by PCA dimensionality reduction to 500 features prior to metric learning.

Features	Image-level accuracy				Webpage-level accuracy			
	MLEV-G	KISSME	OASIS	ITML	MLEV-G	KISSME	OASIS	ITML
fc2	90.25%	91.11%	**91.64**%	91.52%	97.61%	97.95%	**98.29**%	97.61%
fc1	86.83%	90.96%	91.51%	90.56%	95.91%	97.61%	97.61%	97.27%
pool5	88.89%	90.44%	91.36%	90.84%	96.93%	96.93%	97.27%	97.27%
pool4	66.01%	84.73%	84.92%	82.70%	78.23%	95.23%	94.55%	94.89%
pool3	51.77%	74.39%	75.11%	71.01%	67.34%	89.45%	90.81%	89.11%
pool2	25.40%	64.87%	64.87%	46.37%	24.14%	85.03%	86.73%	51.36%
Param.	$\lambda = 10^{-6}$ $p = 450$	–	C=1 make_psd=*true*	$\gamma = 1$				

to the general criterion that deeper layers are more task-specific and less transferable [44], features extracted at deeper levels of the pre-trained network performed better in our experiments. We believe this is a consequence of the relative similarity between the original problem on which VGG16 was trained and our target problem. For instance, while the Imagenet dataset on which VGG16 was trained does not contain an “Interior design” class, it does include classes such as “folding chair”, “pillow” or “table lamp”. Lastly, by looking at both Tables 1 and 2, we can see that, even though a very simple voting scheme was used to aggregate individual image predictions for web page categorization, the web page-level accuracy was consistently greater than that of single image classification. Therefore, we can conclude that our framework benefits from the presence of multiple images in every web page.

Finally, the Mahalanobis metric learning algorithms compared in Table 3 performed reasonably well with the exception of MLEV-G. KISSME, OASIS and ITML managed to improve the distance-based classification accuracy significantly for features extracted at various depths of the DCNN. The best performing method of this class was OASIS, whit a 98.29% web page classification accuracy when running on features extracted at layer *fc*2 of the DCNN. However, these methods were outperformed by the best neural metric learning approach, Triplet-Loss, as highlighted before. We believe the fact that the compared Mahalanobis metric learning algorithms are restricted to learning a linear projection is somehow limiting their performance in this particular application scenario.

4.3. Feature representation visualization

In this section, we try to provide some insight into the suitability of the features at different depths of the network to solve the proposed problem. To do this, we use the metric Multidimensional Scaling (MDS) algorithm as implemented in the scikit-learn

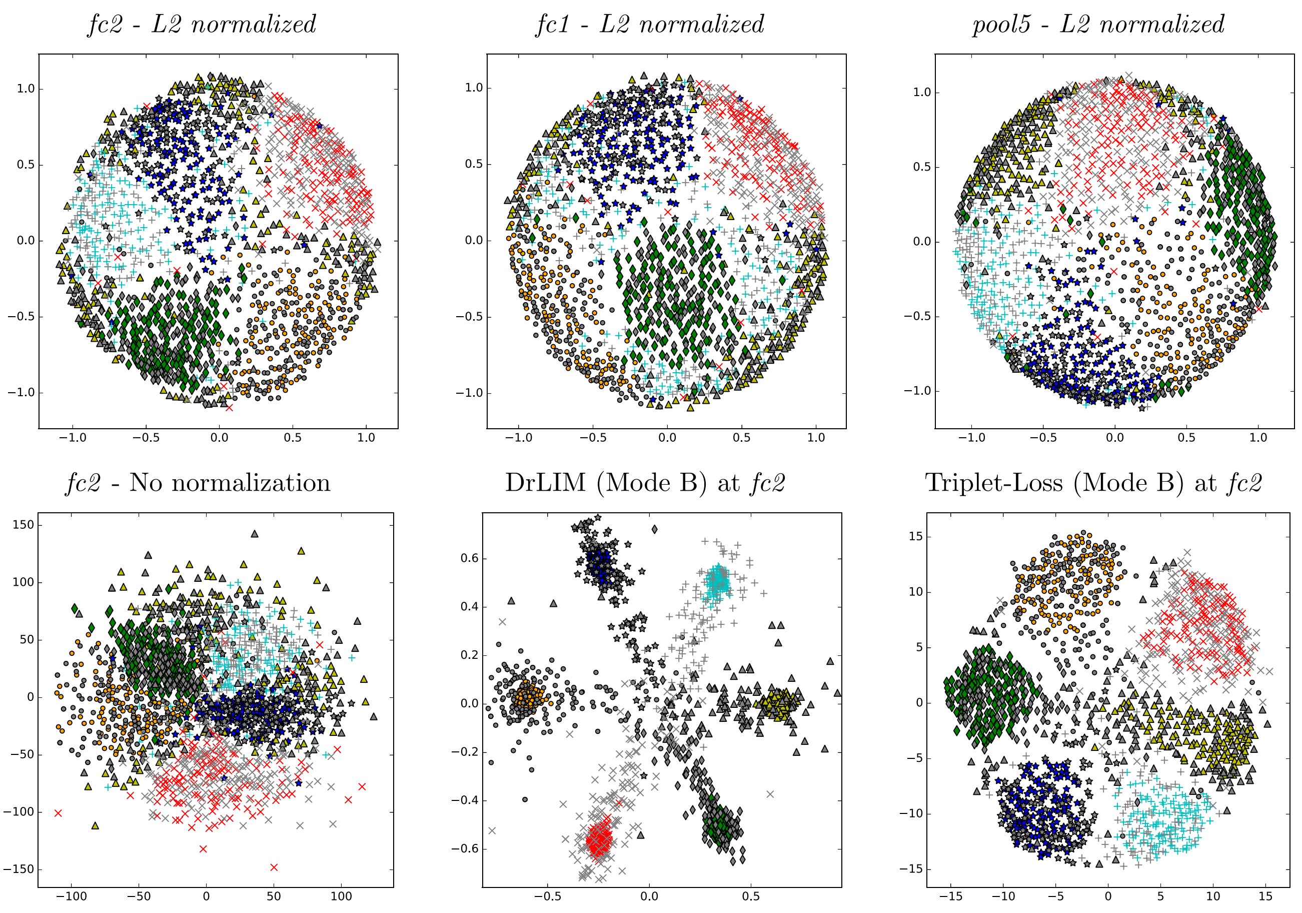


Fig. 8. MDS embedding of features extracted with different configurations of the feature extraction DCNN. Training samples are depicted in color and test samples in gray. The shape of the markers indicates the corresponding class label: *Food* (×), *Motor* (●), *Interior design* (★), *Animals* (▲), *Fashion* (+) and *Landscapes* (♦).

library [52]. MDS is a popular dimensionality reduction algorithm mainly used in data visualization contexts. It works by generating a low-dimensional representation of the high-dimensional input data such that pairwise distances between data points are preserved. As opposed to other popular dimensionality reduction techniques such as t-SNE [56], which focuses on preserving local structure, MDS confers the same importance to the preservation of small and large distances. As a consequence, MDS will enable us to get a intuitive visualization of the structure and distribution of samples in the *k*-NN database when using different feature-extraction network options. Fig. 8 shows a 2D representation of train and test images generated by using MDS. Training samples are depicted in color while test samples are represented in gray[4]. Plotting training and test samples separately will enable us to detect over-fitting in the representations. The class label of each sample is indicated by its shape.

The first three scatter-plots correspond to L2 normalized descriptors extracted at layers *fc*2, *fc*1 and *pool*5. As we can see, a significant level of class-clustering exists in these representations. However, margins are not clearly defined and some overlapping occurs. As a consequence, test samples falling in the boundaries of clusters will likely be misclassified. This problem becomes even more apparent when looking at the scatter-plot corresponding to *fc*2 without normalization. Here, the different classes exhibit a high degree of overlapping, explaining the poor performance registered when using distance-based classification on this representation.

[4] To ease the visualization, only a random subset of the test images were included in the figures.

The final two plots correspond to the representation of samples generated by adapting the final layers of the DCNN with DrLIM and Triplet-Loss, with Mode B at *fc*2 (see Section 3.4). These figures clearly illustrate the nature of both techniques and the effect of their corresponding loss functions. First, we see that in both cases training samples have been accurately clustered, with more defined boundaries than in the previous plots. This explains the improvement in the classification accuracy of *k*-NN when trained on these representations, as even test points falling in the border of clusters will be surrounded mostly by training samples of the correct class. However, we see a fundamental difference in the clusters generated by DrLIM and Triplet-Loss, which can be explained by their corresponding loss functions. In the case of DrLIM, the clusters seem to be more compact and of lower variance, as if all the training samples of each class had been forced to collapse into one point. This is a consequence of the loss function used by DrLIM, which forces samples from the same class to be as close as possible to each other in the output representation, collapsing the intra-class variance present in the input images. Unfortunately, the visualization suggests that this behavior led to some degree of over-fitting, as a significant number of test samples fall far from the training clusters, being more prone to misclassification.

Conversely, whereas the clusters in the scatter-plot of Triplet-Loss are well separated, they preserve a certain degree of intra-class variability. In other words, while the network has leveraged the discriminative information present in images to arrange them in non-overlapping class clusters, it has not collapsed the information present in the form of intra-class variability. In ad-

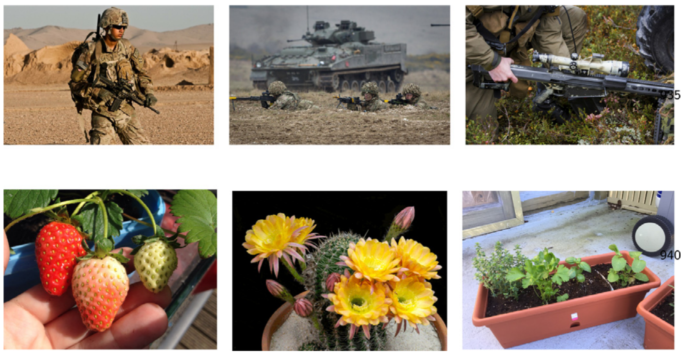


Fig. 9. Some sample images from the new classes "*Military forces*" (top) and "*Gardening*" (bottom), collected to evaluate the over-time learning capabilities of the proposed framework.

dition, the representation of test samples is more consistent with that of training samples, which suggests the absence of over-fitting. These two factors explain the superior performance of Triplet-Loss in the experiments of this paper.

4.4. Over-time learning results

Finally, we present experimental results concerning the claimed over-time learning capabilities of the proposed framework. As explained in Section 3.5, over-time learning can be implemented by simply incorporating new labeled samples to the k-NN database, or it can additionally include fine-tuning of the feature-extraction network (see Algorithm 1). Here we evaluate the performance of both approaches by monitoring the class-specific accuracy of the framework as labeled samples from new classes are provided to the system. For simplicity, we focus solely on single image classification.

For this set of experiments, we collected images from web pages of two new categories. Specifically, the new categories were "*Military forces*" and "*Gardening*". The former category contains images of soldiers, weapons, ammunition, military vehicles such as tanks and other military-related items. The latter consists of images related to gardening and agriculture, such as pots, flowers, cactuses and other plants (see Fig. 9). A total of 790 images were extracted for the class "*Military forces*", while 782 were collected for "*Gardening*". Out of each of these new classes, 50 images were

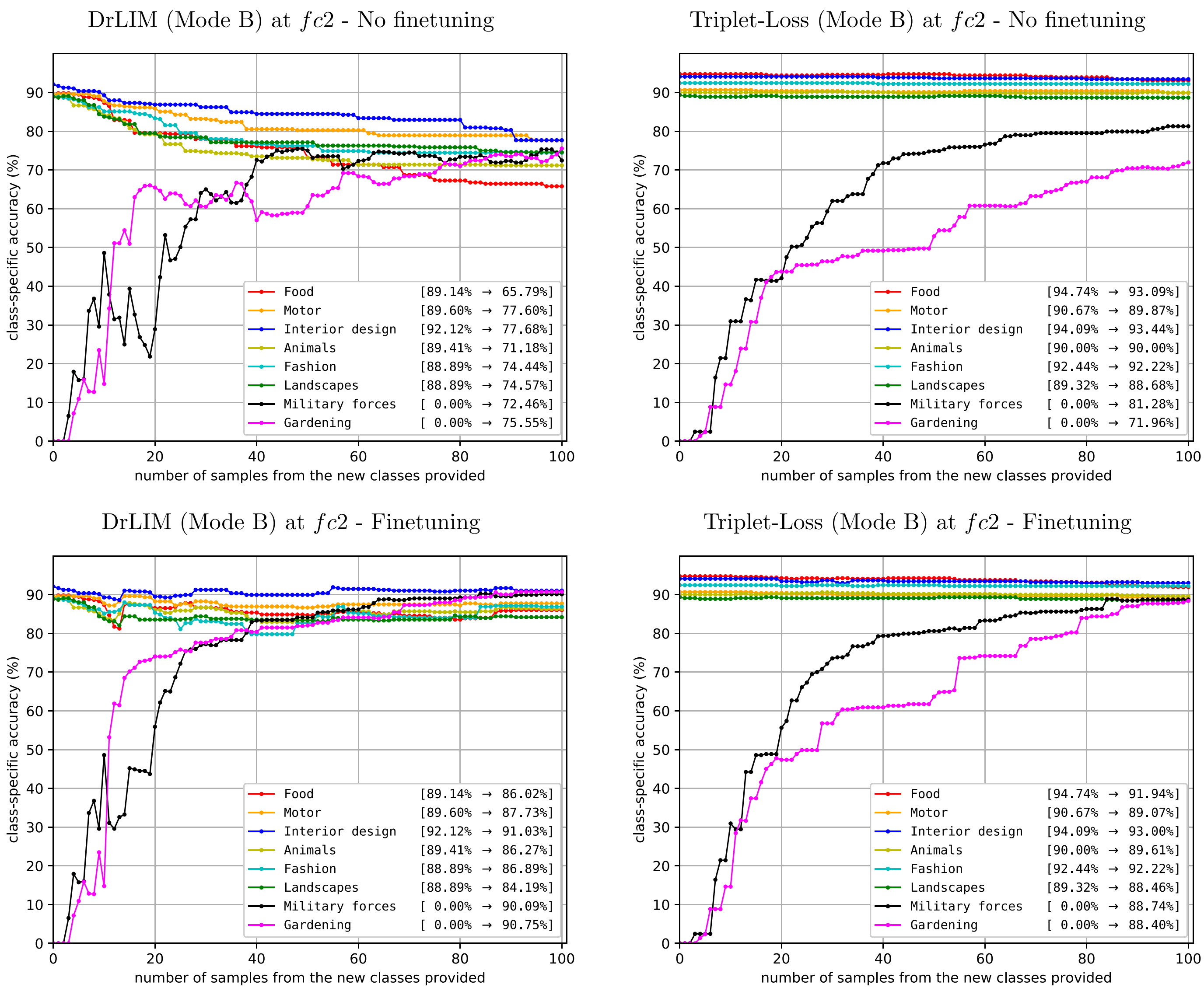


Fig. 10. Evolution of class-specific accuracies as labeled sames from the new classes "*Military forces*" and "*Gardening*" are provided to our framework, instantiated with different configuration options.

set aside to be provided to the framework for over-time learning. The remaining images were added to the test set described in Section 4.1.

To evaluate the over-time learning capabilities of our framework, we simulated an online learning scenario by first training the system in the standard, six-classes training dataset used in the previous sections. Afterwards, the 100 "*Military forces*" and "*Gardening*" labeled images that we set aside were sequentially and in a random order provided to the framework, which incorporated them into its *k*-NN database as described in Section 3.5. At every step, the accuracy of the framework was measured using the test set, which now included the "*Military forces*" and "*Gardening*" test images. The accuracy was computed independently for each class, so we could assess whether learning the new class decreased the performance of the system in recognizing the original classes. This experimental protocol was repeated for different configuration options of our framework. Namely, we experimented with DrLIM and Triplet-Loss ML techniques, Mode B (see Section 3.4) and fine-tuning with Algorithm 1 enabled/disabled. The resulting learning curves are shown in Fig. 10.

The first plot in Fig. 10 corresponds to our framework with DrLIM as the ML algorithm and fine-tuning disabled. In other words, new labeled samples were just included into the *k*-NN database without altering the weights of the feature-extraction network. As we can see, the classification accuracy for samples of the new classes starts at zero (since at the beginning the *k*-NN database contains no samples of those classes) and grows rapidly as new labeled samples are provided. By the time the 100 over-time learning samples had been incorporated into the system, the accuracies for the new classes reached 72.46% and 75.55% respectively. Unfortunately, this learning process also caused a noticeable drop in the accuracy for the original classes. We believe this effect is related to the intra-class variance collapse caused by DrLIM as described in the previous section. This lack of variability might be causing the learned samples from the new class to fall near or inside the clusters of the other classes, leading the system to incorrectly classify some of the test samples as members of the new learned class. The final overall accuracy with this configuration was 78.38%.

The second set of curves corresponds to the use of Triplet-Loss also with fine-tuning disabled. As in the previous case, the accuracies for the new classes started at zero and grew up to 81.28% and 71.96%. Whereas in this case the accuracy of the new class stabilized at a modest accuracy, over-time learning did not cause a decrease in the accuracy for other classes. As a consequence, this option can be considered as a conservative approach, which ensures that new classes will be classified with a reasonable accuracy once enough samples have been incorporated into the *k*-NN database, while preserving almost intact the recognition accuracy for the already-known classes. The final overall accuracy with this configuration was 87.71%.

The plot in the bottom left corner of Fig. 10 corresponds to the instance of our framework using DrLIM as the ML technique and fine-tuning controlled by Algorithm 1. In this case, the accuracies for the original classes were affected considerably during the initial learning stages. However, once the number of learned samples exceeded 70, the accuracy in recognizing the original classes partially recovered and the accuracy of the system in recognizing the new classes began to approach them. At the end of the over-time learning simulation, the accuracies of the system in recognizing the new classes were 90.09% and 90.75%. The final overall accuracy with this configuration was 88.07%.

Lastly, the final set of curves in Fig. 10 corresponds to the use of Triplet-Loss with fine-tuning enabled. As shown in the figure, this configuration provided the best results by both approximately preserving the accuracy for the initial classes and successfully learning to recognize images from the new classes with accuracies of 88.74% and 88.40%. The final overall accuracy with this configuration was 90.34%.

5. Conclusions and future work

In this paper, a novel framework for image-based web page categorization has been proposed. The system is able to classify web pages based on their visual content rather than their textual information. This makes the proposed technique very suitable for modern website analysis where visual elements have a dominant role. As opposed to traditional web page categorization techniques, our approach is language independent, thus being more widely applicable. The major drawback of the proposed approach is that it relies on the presence of images in web pages. As a consequence, it is unable to deal with web pages that only contain textual information. This limitation could be overcome by combining our visual content-based categorization approach with existing text-based web categorization techniques to create a more robust hybrid categorization model, as both approaches are compatible and complementary.

The major contribution of this paper is the application of transfer learning techniques in conjunction with metric learning to the problem of visual-content based web page categorization. Our experimental results show that this approach enables the construction of very accurate classifiers even when the artificial vision task is noticeably complex, and little training data is available. The experiments presented in Section 4.2 show that high classification accuracies can be achieved by using simple linear classifiers on features extracted from a pre-trained DCNN, greatly outperforming state-of-the-art handcrafted feature descriptors. Our results also revealed the importance of L2 normalization for distance-based classification of the features extracted by a DCNN. The results presented in Section 4.4 evidence that metric learning techniques can be applied to adapt the final layer of the feature-extraction DCNN and push the accuracy of distance-based classification even further. The results also suggest that Triplet-Learning is the most appropriate metric learning technique for this task. The visual analysis and discussion presented in Section 4.3 provided insight into the nature of the compared feature representations, and a better understanding of their applicability to the problem being addressed. Finally, the results presented in Section 4.4 suggest that our framework is capable of learning new classes over-time, with little impact on the accuracy of the originally known classes if Triplet-Loss is used in combination with our fine-tuning control algorithm.

Nevertheless, the proposed approach could be further improved. First, a more sophisticated method to filter advertisements and other non-relevant visual content from web pages could be implemented. Also, it would be interesting to reproduce our experiments with different pre-trained DCNN models, investigating whether the higher performance of features extracted at deeper layers and the superiority of Triplet-Loss as compared to DrLIM is independent of the selected pre-trained model. Another interesting research line would be trying to address the problem of imbalanced training datasets with methods like Condensed Nearest-Neightbours or Near-miss [57], as a prior stage to metric learning. Finally, a more advanced way of combining individual image predictions to categorize web pages could be applied, including techniques such as ensemble classification and mixture of experts [58]. In the future, we will also intend to apply the proposed framework to more specific tasks such as visual content-based parental filtering.

Acknowledgment

This work has been supported by project MOVIURBAN: Máquina social para la gestión sostenible de ciudades inteligentes: movilidad urbana, datos abiertos, sensores móviles. SA070U 16. Project co-financed with Junta de Castilla y León, Consejería de Educación and FEDER funds. In addition, the research of Daniel López-Sánchez has been financed by the University Faculty Training (FPU) program, reference number FPU15/02339.

References

[1] K. Mahmood, H. Takahashi, A. Raza, A. Qaiser, A. Farooqui, Semantic based highly accurate autonomous decentralized URL classification system for web filtering, in: Proceedings of the IEEE Twelfth International Symposium on Autonomous Decentralized Systems (ISADS), IEEE, 2015, pp. 17–24.

[2] S. Sun, F. Liu, J. Liu, Y. Dou, H. Yu, Web classification using deep belief networks, in: Proceedings of the IEEE 17th International Conference on Computational Science and Engineering (CSE), IEEE, 2014, pp. 768–773.

[3] A. Sun, E.-P. Lim, W.-K. Ng, Web classification using support vector machine, in: Proceedings of the 4th International Workshop on Web Information and Data Management, ACM, 2002, pp. 96–99.

[4] M. Gu, F. Zhu, Q. Guo, Y. Gu, J. Zhou, W. Qu, Towards effective web page classification, in: Proceedings of the International Conference on Behavioral, Economic and Socio-Cultural Computing (BESC), IEEE, 2016, pp. 1–2.

[5] Y. Zheng, C. Sun, C. Zhu, X. Lan, X. Fu, W. Han, LWCS: a large-scale web page classification system based on anchor graph hashing, in: Proceedings of the 6th IEEE International Conference on Software Engineering and Service Science (ICSESS), IEEE, 2015, pp. 90–94.

[6] T.F. Abidin, R. Ferdhiana, Algorithm for updating n-grams word dictionary for web classification, in: Proceedings of the International Conference on Informatics and Computing (ICIC), IEEE, 2016, pp. 432–436.

[7] H. Wang, H. Huang, F. Nie, C. Ding, Cross-language web page classification via dual knowledge transfer using nonnegative matrix tri-factorization, in: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2011, pp. 933–942.

[8] X. Qi, B.D. Davison, Web page classification: Features and algorithms, ACM Comput. Surv. (CSUR) 41 (2) (2009) 12.

[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252.

[10] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436.

[11] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, M.S. Lew, Deep learning for visual understanding: a review, Neurocomputing 187 (2016) 27–48.

[12] D. López-Sánchez, A.G. Arrieta, J.M. Corchado, Deep neural networks and transfer learning applied to multimedia web mining, in: Proceedings of the International Symposium on Distributed Computing and Artificial Intelligence, Springer, 2017, pp. 124–131.

[13] D. Mladenic, Turning yahoo into an automatic web-page classifier, in: Proceedings of the 13th European Conference on Artificial Intelligence (ECAI), Wiley, 1998, pp. 473–474.

[14] K. Golub, A. Ardö, Importance of HTML structural elements and metadata in automated subject classification, in: Proceedings of the International Conference on Theory and Practice of Digital Libraries, Springer, 2005, pp. 368–378.

[15] O.-W. Kwon, J.-H. Lee, Text categorization based on k-nearest neighbor approach for web site classification, Inf. Process. Manag. 39 (1) (2003) 25–44.

[16] H. Utard, J. Fürnkranz, Link-local features for hypertext classification, in: Proceedings of the Semantics, Web and Mining, Springer, 2006, pp. 51–64.

[17] L. Nie, B.D. Davison, X. Qi, Topical link analysis for web search, in: Proceedings of the 29th annual international ACM SIGIR conference on Research and Development in Information Retrieval, ACM, 2006, pp. 91–98.

[18] S.T. Marath, M. Shepherd, E. Milios, J. Duffy, Large-scale web page classification, in: Proceedings of the 47th Hawaii International Conference on System Sciences (HICSS), IEEE, 2014, pp. 1813–1822.

[19] G. Heinrich, Evaluation of a distribution-based web page classification, in: Digital Transformation in Journalism and News Media, Springer, 2017, pp. 55–68.

[20] Y. Wei, W. Wang, B. Wang, B. Yang, Y. Liu, A method for topic classification of web pages using LDA-SVM model, in: Proceedings of the Chinese Intelligent Automation Conference, Springer, 2017, pp. 589–596.

[21] D. López-Sánchez, J.M. Corchado, A.G. Arrieta, A cbr system for image-based webpage classification: case representation with convolutional neural networks, in: Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference (FLAIRS), AAAI, 2017, pp. 483–488.

[22] W. Chu, D. Cai, Deep feature based contextual model for object detection, Neurocomputing 275 (2018) 1035–1042.

[23] Z. Huang, S.M. Siniscalchi, C.-H. Lee, A unified approach to transfer learning of deep neural networks with applications to speaker adaptation in automatic speech recognition, Neurocomputing 218 (2016) 448–459.

[24] X. Wang, L. Gao, J. Song, X. Zhen, N. Sebe, H.T. Shen, Deep appearance and motion learning for egocentric activity recognition, Neurocomputing 275 (2018) 438–447.

[25] A.S. Razavian, H. Azizpour, J. Sullivan, S. Carlsson, CNN features off-the-shelf: an astounding baseline for recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE, 2014, pp. 512–519.

[26] D. López-Sánchez, J.R. Herrero, A.G. Arrieta, J.M. Corchado, Hybridizing metric learning and case-based reasoning for adaptable clickbait detection, Appl. Intell. (2018) 2967–2982.

[27] R. Hadsell, S. Chopra, Y. LeCun, Dimensionality reduction by learning an invariant mapping, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2, IEEE, 2006, pp. 1735–1742.

[28] F. Schroff, D. Kalenichenko, J. Philbin, FaceNet: a unified embedding for face recognition and clustering, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 815–823.

[29] Z. Chen, A. Jacobson, N. Sünderhauf, B. Upcroft, L. Liu, C. Shen, I. Reid, M. Milford, Deep learning features at scale for visual place recognition, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2017, pp. 3223–3230.

[30] S. Bell, K. Bala, Learning visual similarity for product design with convolutional neural networks, ACM Trans. Graph. (TOG) 34 (4) (2015) 98.

[31] G. Koch, Siamese neural networks for one-shot image recognition, Ph.D. thesis. University of Toronto, 2015.

[32] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, F. Moreno-Noguer, Discriminative learning of deep convolutional feature point descriptors, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), IEEE, 2015, pp. 118–126.

[33] Y. Sun, Y. Chen, X. Wang, X. Tang, Deep learning face representation by joint identification-verification, in: Proceedings of the Advances in Neural Information Processing Systems, 2014, pp. 1988–1996.

[34] H. Bredin, TristouNet: triplet loss for speaker turn embedding, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2017, pp. 5430–5434.

[35] Z. Liu, P. Luo, S. Qiu, X. Wang, X. Tang, DeepFashion: powering robust clothes recognition and retrieval with rich annotations, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1096–1104.

[36] T. Bui, L. Ribeiro, M. Ponti, J. Collomosse, Compact descriptors for sketch-based image retrieval using a triplet loss convolutional neural network, Comput. Vis. Image Underst. 164 (2017) 27–37.

[37] A. Bellet, A. Habrard, M. Sebban, A survey on metric learning for feature vectors and structured data, arXiv preprint, 2013, arXiv:1306.6709 .

[38] D. Li, Y. Tian, Global and local metric learning via eigenvectors, Knowl. Based Syst. 116 (2017) 152–162.

[39] M. Koestinger, M. Hirzer, P. Wohlhart, P.M. Roth, H. Bischof, Large scale metric learning from equivalence constraints, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2012, pp. 2288–2295.

[40] G. Chechik, V. Sharma, U. Shalit, S. Bengio, Large scale online learning of image similarity through ranking, J. Mach. Learn. Res. 11 (2010) 1109–1135.

[41] J.V. Davis, B. Kulis, P. Jain, S. Sra, I.S. Dhillon, Information-theoretic metric learning, in: Proceedings of the 24th International Conference on Machine Learning, ACM, 2007, pp. 209–216.

[42] V.G. Nair, Getting Started with Beautiful Soup, Packt Publishing Ltd, 2014.

[43] K. Weiss, T.M. Khoshgoftaar, D. Wang, A survey of transfer learning, J. Big Data 3 (1) (2016) 9.

[44] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks? in: Proceedings of the Advances in Neural Information Processing Systems, 2014, pp. 3320–3328.

[45] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition,arXiv preprint, 2014, arXiv:1409.1556.

[46] F. Chollet, et al., Keras, 2015, (https://github.com/keras-team/keras).

[47] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826.

[48] F. Chollet, Xception: deep learning with depthwise separable convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1800–1807.

[49] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, SSD: single shot multibox detector, in: Proceedings of the European Conference on Computer Vision, Springer, 2016, pp. 21–37.

[50] A. Canziani, A. Paszke, E. Culurciello, An analysis of deep neural network models for practical applications, arXiv preprint, 2016, arXiv:1605.07678.

[51] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, LIBLINEAR: a library for large linear classification, J. Mach. Learn. Res. 9 (2008) 1871–1874.

[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[53] D. Jayaraman, R. Gao, K. Grauman, Unsupervised learning through one-shot image-based shape reconstruction, arXiv preprint, 2017, arXiv:1709.00505.

[54] J. Sánchez, F. Perronnin, T. Mensink, J. Verbeek, Image classification with the fisher vector: theory and practice, Int. J. Comput. Vis. 105 (3) (2013) 222–245.

[55] J. Delhumeau, P.-H. Gosselin, H. Jégou, P. Pérez, Revisiting the VLAD image representation, in: Proceedings of the 21st ACM International Conference on Multimedia, ACM, 2013, pp. 653–656.

[56] L.v.d. Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (2008) 2579–2605.

[57] G. Lemaître, F. Nogueira, C.K. Aridas, Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning, J. Mach. Learn. Res. 18 (17) (2017) 1–5. http://jmlr.org/papers/v18/16-365.html.

[58] S. Masoudnia, R. Ebrahimpour, Mixture of experts: a literature survey, Artif. Intell. Rev. 42 (2) (2014) 275–293.

Daniel López-Sànchez (M.Sc.) obtained a Computer Science degree (highest GPA award) in 2015 and an M.Sc. in Artificial Intelligence in 2016 at the University of Salamanca (Spain). In 2016, he received the FPU grant from the Spanish Government and started pursuing a Ph.D. at the BISITE Research Group. His research interest focuses in the field of machine learning, especially in the subfields of dimensionality reduction and kernel methods. He is also interested in developing novel applications of the Deep Learning paradigm. He has participated as a co-author in papers published in recognized international conferences and journals.

Angélica Gonzàlez-Arrieta (Ph.D.) Received a Ph.D. in Computer Science from the University of Salamanca in 2000. She is currently a Lecturer in Salamancas University Department of Computer Science and has attended several Masters courses. She is further a professor and tutor for UNED (Universidad Espaola de Educacin a Distancia, Spains Open University). In the past, she carried out relevant administrative tasks, such as Academic Secretary of the Science Faculty (19962000) and Chief of Staff for the University of Salamanca (20002003). From 1990, she has cooperated with the Home Ministry, and from 2008 with the Home and Justice Counsel of the local government (Junta de Castilla y Len). She is a member of the research group BISITE (http://bisite.usal.es) and has lead several research projects sponsored by both public and private institutions in Spain. She is the coauthor of several works published in magazines, workshops, meetings, and symposia.

Juan M. Corchado (Ph.D.) Received a Ph.D. in Computer Science from the University of Salamanca in 1998 and a Ph.D. in Artificial Intelligence (AI) from the University of Paisley, Glasgow (UK) in 2000. At present, he is Vice President for Research and Technology Transfer since December 2013 and a Full Professor with Chair at the University of Salamanca. He is the Director of the Science Park of the University of Salamanca and Director of the Doctoral School of the University. He has been elected twice as the Dean of the Faculty of Science at the University of Salamanca. He has led several Artificial Intelligence research projects sponsored by Spanish and European public and private sector institutions and has supervised seven Ph.D. students. He is the coauthor of over 230 books, book chapters, journal papers, technical reports, etc. Since January 2015, Juan Manuel Corchado is Visiting Professor at Osaka Institute of Technology. He is also member of the Advisory group on Online Terrorist Propaganda of the European Counter Terrorism Centre (EUROPOL). Juan Manuel is also editor and Editor-in-Chief of Specialized Journals like ADCAIJ (Advances in Distributed Computing and Artificial Intelligence Journal), IJDCA (International Journal of Digital Contents and Applications) and OJCST (Oriental Journal of Computer Science and Technology).