A Hybrid Framework for Matching Printing Design Files to Product Photos

A. KAPLAN and E. AKAGUNDUZ

***Abstract*— We propose a real-time image matching framework, which is hybrid in the sense that it uses both hand-crafted features and deep features obtained from a well-tuned deep convolutional network. The matching problem, which we concentrate on, is specific to a certain application, that is, printing design to product photo matching. Printing designs are any kind of template image files, created using a design tool, thus are perfect image signals. For this purpose, we create an image set that includes printing design and corresponding product photo pairs with collaboration of an actual printing facility. Using this image set, we benchmark various hand-crafted (SIFT, SURF, GIST, HoG) and deep features for matching performance. Various segmentation algorithms including deep learning based segmentation methods are applied to select feature regions. Results show that SIFT features selected from deep segmented regions achieves up to 96% product photo to design file matching success in our dataset. We propose a framework in which deep learning is utilized with highest contribution, but without disabling real-time operation using an ordinary desktop computer.**

Index Terms— **image matching, hand-crafted features, deep features, semantic segmentation, product image processing**

I. Introduction

IMAGE MATCHING is a broad title that covers or partially relates to various topics among a number of different computer vision problems, namely image-based localization, multi-view 3D reconstruction, structure-from-motion, image retrieval, tracking, just to name a few. This title may refer to finding a transformed version of an image [1], or may refer to a different version of the problem, such as finding an image with a similar semantic context [2]. Regardless of the problem definition, image matching boils down to a simple statement: finding a similarity model between (at least) two images, which would satisfy the pairings for a given image set.

The algorithms proposed under this title in recent years can be mainly split into two principle categories. The first category consists of approaches that utilize hand-crafted representations. Among these methods, the bag-of-visual-words (BoVW) algorithm [3] proved to be very successful, irrespective of the type of the hand-crafted feature used, and is still the state-of-the-art approach due to its flexibility, compactness and speed.

However, as a part of the growing wave of interest on deep-learning-based methods, a second category of approaches recently focus on image matching using convolutional neural networks (CNN) [4]. The strength of these methods comes from the abstract features that merge at the deeper layers of CNNs [5]. The earlier approaches of this category [6-9] performs particularly good at problems like image category classification, object detection and/or localization, mainly because of their capability to convolve abstract features into image categories or object definitions. There are also attempts with promising results, which aim at transferring pre-trained and well-tuned CNNs into image retrieval frameworks [10, 11]. Nonetheless, these network structures are not well-suited to match a given image to its pair, since they use fully connected layers that lead to a classification layer (such as soft-max). This final layer is used to classify the extracted abstract features into object categories. Therefore, their structure is not designed with the purpose of finding pairs.

Very recently a new CNN structure, namely the Siamese network (SN), has been proposed specifically for the problem of image matching [12]. SNs can learn feature spaces that map similar image pairs close to each other and dissimilar image pairs with a selective distance, by using labelled pairs. This approach has also been successfully applied to similar problems that require an image-to-image matching, such as face recognition [13] or aerial-to-ground image matching [14]. SNs improve matching performance dramatically, however they come up with two main drawbacks. Firstly, they necessitate the creation of a large-scale image set, because in order to span the entire space of possible transformations from the original image to the image to be matched, a massive number of image pairs are required. Such an image set collection and annotation effort is extremely expensive and usually, industrially impracticable. Secondly, these networks make a separate full forward deep CNN run for each candidate image in the image set, thus are slow even with dedicated hardware, such as a GPU.

Deep learning is a powerful tool. In less than a decade, nearly all vision problems shifted to CNN domain.

ALPER KAPLAN, is with Cognitive Science Program, Graduate School of Social Sciences, Yeditepe University, Istanbul, Turkey (e-mail: alperkaplan@outlook.com).

ERDEM AKAGUNDUZ, is with Department of Electrical and Electronics Engineering, Çankaya University, Ankara, Turkey, (e-mail: akagunduz@cankaya.edu.tr).

https://orcid.org/0000-0002-2306-6008

Manuscript received January 20, 2020; accepted April 16, 2020.
DOI: 10.17694/bajece.677326

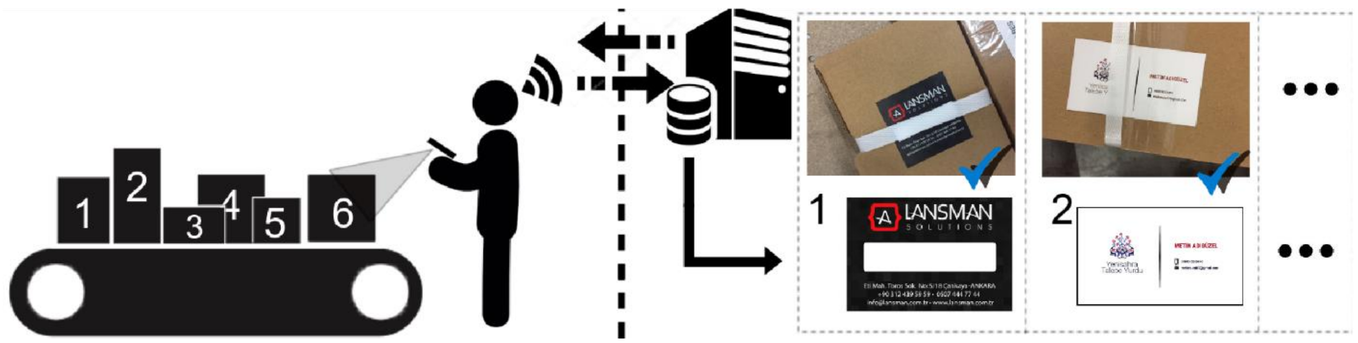


Fig.1. Pictorial representation of our operational problem definition

Nevertheless, it still comes with a price. As the layers of a CNN get deeper, the hardware requirements for real-time operation become more and more expensive. We still don't have a mobile solution, which may replace the high-cost and power-hungry GPUs that allow real-time deep learning operations. And when it comes to the problem of image matching, our best solution yet, namely the Siamese networks architecture, require massive training sets and forward-run for all possible candidate images. In conclusion we still need ingenious solutions for real-time, operation-specific and affordable image matching frameworks.

A. *Problem Definition and the Proposed Solution*

In this paper, we study a particular version of the image matching problem, in which we match printing design files to product photos. Printing designs are any kind of template image files, created using a design tool and used as templates for printing a flyer, banner, poster, etc. These files are computer generated, thus they possess no signal-based deficiencies like noise or optic blur. Most of them are in vector format, hence, are resolution-free.

On the other hand, photographs of a printed product suffer many unwanted effects, such as uncontrolled shooting angle, uncontrolled illumination, occlusions, printing deficiencies in colour, camera noise, optic blur, etc. Matching them to their original design files requires learning the unknown transformation that the photographing action creates. This transformation is not deterministic by nature and is affected by predominant uncontrolled factors, such as the photographer, the camera or the background.

A pictorial representation of our problem definition and system framework is depicted in Figure 1. In a sample scenario of our problem definition, an operator (or an automatic visualization system) shoots the photographs of some printed products by using a computation-limited device (such as a mobile phone, etc.). Then this device sends the product photo to a server machine, in which the photo is matched to its design file pair, in real-time.

In order to solve this problem, we propose a real-time image matching framework, which is hybrid in the sense that it uses both hand-crafted features and deep features obtained from a well-tuned very deep CNN [6]. The hand-crafted or deep features are extracted only from a region that is designated by a fine-tuned deep CNN. In our framework, this feature region segmentation operation is the only "deep" operation that is applied on the product photo, thus we avoid running a deep CNN for each possible pair in the image set, as it is done for Siamese networks. This also prevents us from using expensive deep learning hardware (a GPU), but still permits us to provide real-time operation. By using the hand-crafted or deep features extracted from the deep segmented region, a BoW framework is utilized in order to find the correct image pair.

In following section, we provide the details of the image set that includes printing design files and corresponding product photo pairs. Section 3 explains the deep learning experiments, which aim at solving feature region segmentation problem. Section 4 represents the BoVW framework, in which different hand-crafted and/or deep feature extraction, and product segmentation methods are benchmarked for optimal performance. Section 5 presents the experimental results, whereas the final section concludes the paper and gives directions for future work.

II. Design File - Product Photo Pairs Image Set

The existing image sets [15-18] prepared for matching or retrieval problems in the literature are very diverse in category. They deal with different problems such as retrieving RAW images, medical images, outdoor images, or even satellite images. Consequently, for each image set the problem definition is different. That's why, in order to provide a solution for our specific problem definition, we need to create a specific image set that includes printing design files and corresponding product photos.

As a consequence, an image set creation effort was carried out. To this end, an operator took the photographs of 2000 products at the production line. Product photos are images of a sample product (e.g. a flyer) usually stitched over a cargo box, which carries the other printed samples (Figure 2). The idea is to recognize this product (i.e. its ID) by matching the image of the sample on the cargo box with the design file at the server.

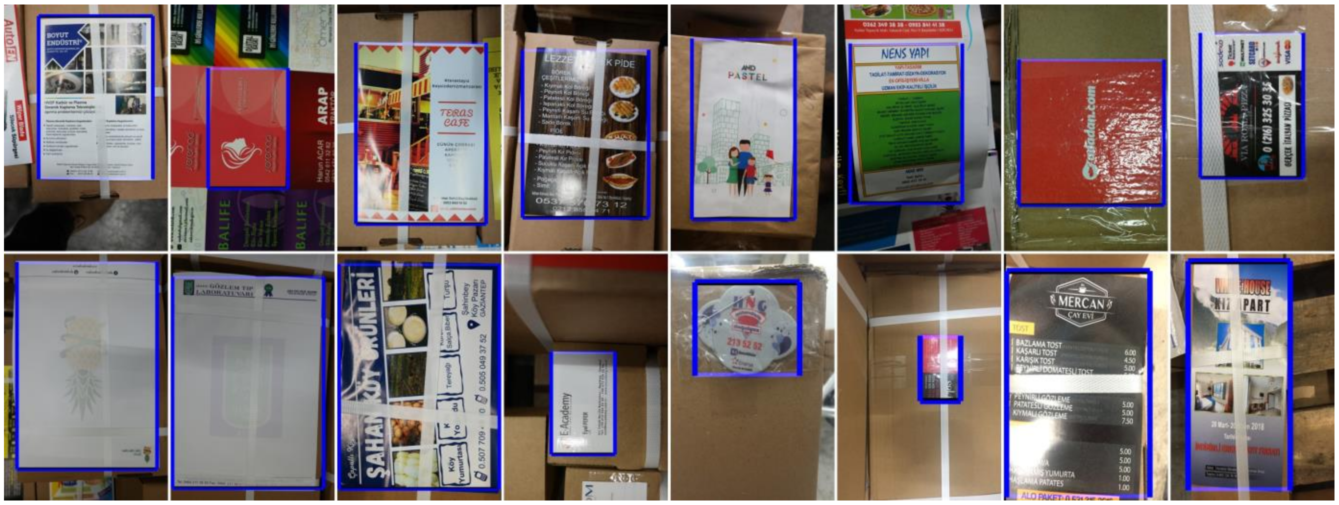


Fig.2. Sample product photos from the image set

For some of the products, there exists more than a single design file. A good example is a business card (Figure 1), which usually has information on both sides, and thus has two separate design files. In these cases, the problem definition is to match the product photo (which could be any face of the card) to one of the design files in the image set. Consequently, for the photographed 2000 product samples, 3458 design files were added to the image set. The operators were advised to shoot the product with a perpendicular angle so that the product (usually, but not necessarily rectangular in shape) would fit the image with uniform margins. However, this weak protocol was not successfully applied to all images, mainly because of human-errors, and it is difficult to say that the image set is rotation or scale controlled (please see Figures 1 and 2).

In addition to the product shooting and design file labelling efforts, an annotation effort was also carried out. For each photographed product the rectangle that encapsulates the product sample was annotated on the images (depicted as blue rectangles on Figure 2). These annotations will later be used as ground truth to our deep learning framework in the following section.

In Figure 2, several examples from the image set are provided. As it can be seen from this figure, the set includes various types of background clutter, occlusions caused by packaging (tapes, chords, etc.), unwanted flash light reflections, non-uniform illumination, folding of the sample product and such. Thus, it is important that the matching solution we propose, must be robust to these types of effects.

III. Deep Product Segmentation

As mentioned in the introduction section, we apply a deep-segmentation supported bag-of-visual-words method to match product photos to design files. This framework, with rigorous benchmarking, is provided in the next section. However, before we get into the details of our matching framework, in this chapter we present some methods for segmenting the product region in product photos using different deep CNN (DCNN) architectures.

Finding the pixels that belong to a specific object category is known as semantic segmentation in the literature. The reader may refer to various surveys on this problem [19-26]. The literature involves hundreds of different approaches to semantic segmentation. The most common component among these approaches is undoubtedly the utilization of the abstract features of pre-trained DCNNs, by fine-tuning or transfer learning.

In this paper, in order to segment the pixels of a product photo using deep learning, we adapt three different architectures: “FCN32s”, “FCN8s” and finally the proposed “VGG-Regression-Net”, as we name it.

TABLE I
VGG-REGRESSION-NET ARCHITECTURE

Input Layer	224×224×3 RGB Image. (VGG default input image size)	
VGG Layers	Pre-trained VGG layers (1 to N)	
New Layers	(all fully connected)	
layer no.	Number of Weights	Activation Vector Size
N+1	j×k×d×256	1×1×256
N+2	1×1×256×256	1×1×256
N+3	1×1×256×900	1×1×900
Output	900x1 vector (30x30 Segmentation Result)	

FCN32s and FCN8s are well-known fully convolutional semantic segmentation networks, designed specifically for this problem [27]. They are originally trained for 21 different pixel labels. The only difference between these two architectures is that FCN8s includes skip connections that allow feature concatenation between different hierarchies within the DCNN. In order to adapt these networks to our problem, the final deconvolutional layers are set to 2 labels depth (as “product” and “background”), and while this layer is being learned from scratch, all other convolutional layers in the networks are fine-tuned during training.

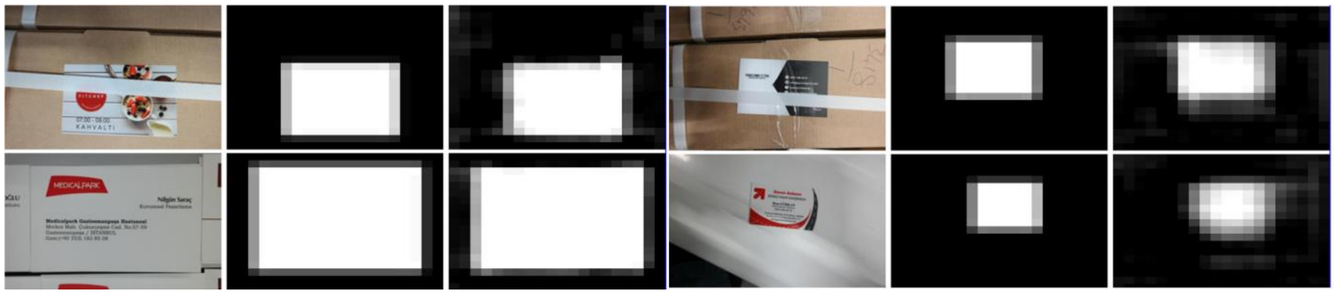


Fig.3. Segmentation results for VGG-Regression-Net Layer 13, namely Conv5[1] are depicted. For four different samples, the image (left), the annotated ground truth (middle) and the DCNN output (right) are shown.

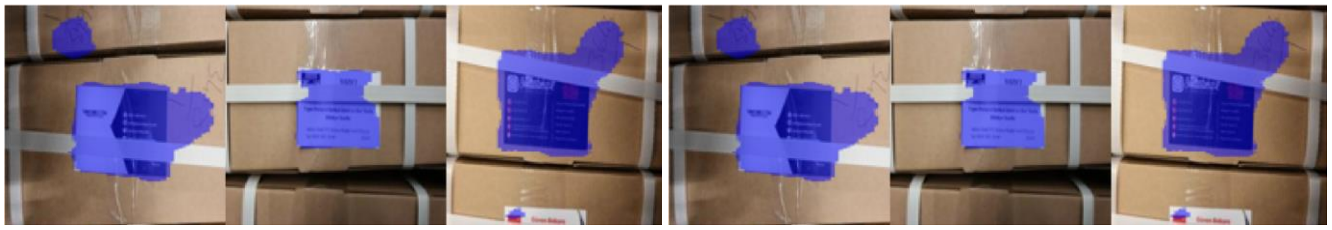


Fig.4. Segmentation results, as blue regions over the image, for FCN8s (leftmost three images) and FCN32s (rightmost three images) are depicted. FCNs may create segmentation regions with disconnected blobs, since their fully convolutional nature has no means to prevent such an output

A. *VGG-Regression-Net Architecture*

In addition to these fully convolutional architectures, a regression network is also proposed for the same problem. To that end, the convolution layers of a pre-trained deep network, namely VGG-VD-19L [6] are transferred to the VGG-Regression-Net (VGGRN) architecture. In VGGRN, fully connected layers are replaced and retrained. The aim is to assess whether a DCNN with fully connected layers, this time trained for regression of the segmentation mask, performs better than fully convolutional architectures, such as FCN32s or FCN8s. We hypothesize that the fully connected layers can provide inference for a global composition of the image, in which the FCNs could fail to achieve.

TABLE II
VGG-REGRESSION-NET EXPERIMENTS

Layer	NCC mean	NCC std.
VGG-Regression-Net Layer 09: Conv4^1	0.84	±0.11
VGG-Regression-Net Layer 10: Conv4^2	0.85	±0.12
VGG-Regression-Net Layer 11: Conv4^3	0.87	±0.15
VGG-Regression-Net Layer 12: Conv4^4	0.87	±0.13
VGG-Regression-Net Layer 13: Conv5^1	0.92	±0.06
VGG-Regression-Net Layer 14: Conv5^2	0.91	±0.09
VGG-Regression-Net Layer 15: Conv5^3	0.90	±0.14
VGG-Regression-Net Layer 16: Conv5^4	0.89	±0.12
VGG-Regression-Net Layer 16: Conv5^4	0.89	±0.12
FCN32s	0.87	±0.10
FCN8s	0.85	±0.09

The detailed architecture of the VGG-Regression-Net is provided in Table 1. Selected N number of pre-trained convolutinoal layers of VGG-VD-19L are transferred to this new architecture. The input layer of the original VGG-VD-19L receives 244×224 pixels RGB images. So any product photo that is fed to this DCNN is first down-sampled into 244×224 pixels resolution.

In order to find the most suitable layer, multiple transfer learning experiments are run. Each separate experiment corresponds to creating a new DCNN by transferring "some" VGG-VD-19L layers and replacing new fully connected decision layers, so that we create the segmentation mask for the product in the product image. The weights of the transferred layers are frozen during training.

The ground truth of these masks are obtained by using the annotation mentioned in the previous section (please see the blue rectangles in Figure 2). For each product photo, a ground truth mask for which the pixels inside the rectangle region are 1 and the rest (i.e. the background) 0, is created and used for training. The output of the final fully-connected layer consists of 900 components, which is actually the 30×30 pixels-sized, down-sampled version of the segmentation mask.

B. *Training the Deep Segmentation Architectures*

Although training the FCNs is a subject of semantic segmentation, training the proposed VGG-Regression-Net architecture is a regression problem. In order to train this DCNN, L1-norm operator is implemented as a loss function. Stochastic gradient descent (SGD) with momentum is utilized and a batch size of 16 images[1] is used for batch normalization.

[1] Stochastic Gradient Descent (SGD) algorithm with momentum is employed, Initial Learning rate: 0.001, Weight Decay: 0.0004, Momentum: 0.91. MatConvNet [28] library is used for training the VGG-Regression-Net, while MATLAB Deep Learning Toolbox is utilized for training the FCNs.

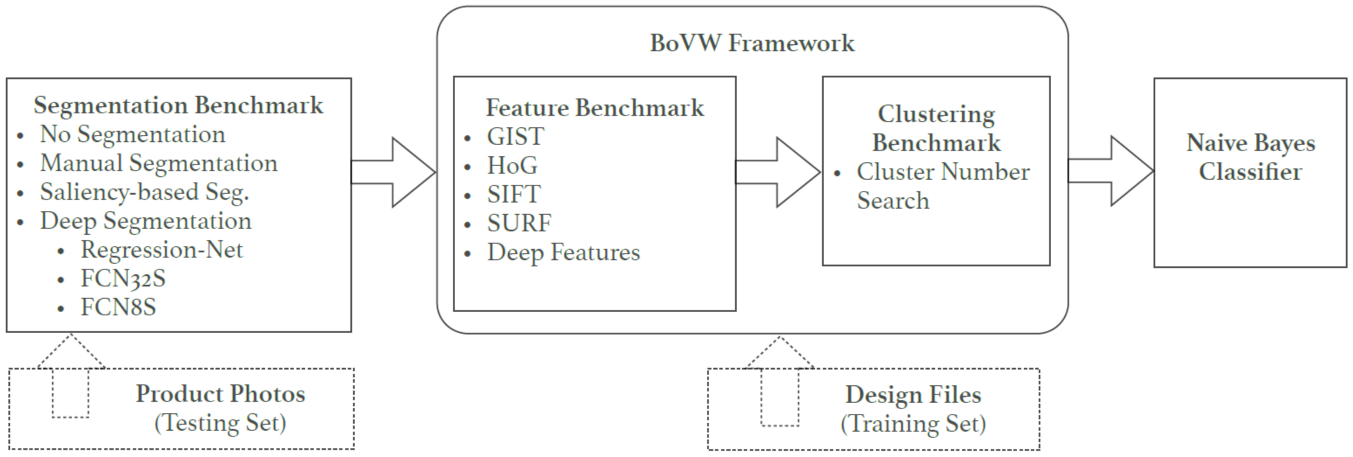


Fig.5. The overall matching framework is depicted. The framework consists of four main blocks, namely segmentation, feature extraction, clustering and classification.

For all architectures, data augmentation is applied by mirroring (×2), zooming (×2) and rotating (×4) the product photos, thus enlarging the image set by 16. For training the architectures 75% and for validation 15% of the image set are used. For this reason, each experiment is run 10 times, using a different subset for testing, which contains a separate 10% of the whole image set.

Regarding the VGG-Regression-Net architecture, in order to find the layer that provides the best abstract features for segmentation, 8 different structures are trained, by cutting the VGG-VD-19L at 8 different layers, namely conv41, conv42, conv43, conv44, conv51, conv52, conv53 and conv54, which are the 9th to 16th layers of VGG-VD-19L. Thus for 8 layers of VGG-Regression-Net, FCN32s and FCN8s, separately for 10 test sets, a total of 100 learning experiments are run.

In order to benchmark the success of different VGG-Regression-Nets and the FCNs architectures, normalized-cross correlation (NCC) of the test results with the ground truth are calculated and averaged over the entire set, respectively for each experiment. In Table 2, for each experiment the mean and the standard deviation of the segmentation accuracy (i.e. normalized-cross correlation of test results with the ground truth) are calculated. The best results are obtained using the 13th layer of VGG-VD-19L for fine-tuning experiments, with an average of 0.92 normalized cross-correlation. FCNs both perform poor.

We believe that this is mainly because of the fact that, the problem we solve here is not exactly semantic segmentation. The product photo to be segmented, a poster, a flyer et cetera is a composition of objects, not a single object. Our problem is about learning the pixels of a product inside an image, as a composition of objects. We believe that the reason why the VGG-Regression-Net architecture performed better compared to FCNs is mainly because, fully connected layers can learn the global composition of abstract features, whereas FCNs, with limited receptive fields, search for objects in local regions. Therefore, as seen in Figure 4, even disconnected blobs as segmentation results can be obtained for FCNs.

In the rest of this paper, we examine the effect of these three different segmentation methods to our matching performance. For this purpose, each matching method is deep segmented by the two adapted FCNs and the best VGG-Regression-Net experiment, which is obtained by using the abstract features from layer 13 (namely conv5[1]).

IV. Image Matching Framework

As previously mentioned in the introductory sections, the aim of this study is to find a real-time solution to product photo and design file matching problem. For this purpose, we propose a framework with various benchmarking experiments. The proposed framework is hybrid in the sense that it fuses a conventional pattern recognition method that uses hand-crafted features with a deep segmentation technique. And while doing this, the study presents benchmarking of different methods, in order to correctly acknowledge the best performance for the given framework.

In Figure 5, the high level depiction of our framework can be seen with the benchmarking processes we utilize. The matching is accomplished by using the BoVW method [3] together with a Naïve Bayes classifier. The main advantage of using BoVW is that it provides a fixed-length representation of the image, regardless of the number or type of features obtained from that image. Moreover, BoVW + Naïve Bayes online operation (testing) is extremely fast. It requires a relatively slower offline training phase, in which the features obtained from training set is clustered into N sets. But needless to say, this does not affect the real-time operation in our framework. Within the BoVW framework, two principle benchmarking efforts are carried out, first being the benchmarking for selection of the hand-crafted or deep features of BoVW and second being the benchmarking for the optimal number of clusters for BoVW.

BoVW relies on the features obtained from a test image, which is a product photo in our case. The product photo does not only include the “product” but considerable background as well; ergo, it may be crucial to select the features only from the product region in the photo. For this purpose, as seen in Figure 5, another benchmarking effort for finding the product region is also carried out, using different segmentation methods including the deep segmentation techniques explained in the previous section.

In the following subsections, we explain each benchmarking effort separately, following their process order in the framework. Thus, we start with the segmentation benchmark. Then we delve into our results and carry out discussions on the optimum method.

A. Segmentation Benchmark

As mentioned above, selecting the features only from the product regions may dramatically affect the matching performance of a BoVW model. For this reason, in our experiments we utilize five product segmentation methods out of three categories and compare their results within the complete framework. We also include the case where no segmentation is carried out, so that we can clearly assess the contribution of a tested segmentation method.

We categorize our tested segmentation strategies in three titles, namely manual segmentation, unsupervised segmentation and supervised segmentation.

1) Manual Segmentation

Manual segmentation is accomplished by the operator. As seen in Figure 1, the operator shoots the products with a mobile device and at this very moment, can manually segment the product region in the photo using the same mobile device. This is an unwanted scenario because it increases the operation time, which contradicts with the general purpose of the proposed system. However, we still choose to utilize this segmentation method in our framework, so as to see the effect of "perfectly" segmenting the product in a photo to our overall matching success and we regard this method as a ground truth for the segmentation step. In our image set we already have these manual annotations (please see Section 2 and Figure 2).

2) Unsupervised Segmentation: Visual Saliency

The first automatic segmentation method we utilize is an unsupervised method to find the product region in a photo. Unsupervised segmentation had been a very hot topic [29] before deep learning overwhelmingly manipulated the field with the idea of employing large-scale data to any problem. Since, for the sake of cheap and fast operation, we try to avoid deep operation as much as we can, we select a visual saliency-based method as our unsupervised segmentation method.

For this purpose, we use Graph-based Visual Saliency (GBVS) method, which is a bottom-up visual saliency model. By creating Markov chains among image pixels, the GBVS algorithm calculates saliency values from equilibrium distributions over pixel map locations [30]. Although it is not a direct segmentation technique, visual saliency is being used as an objectness measure and is utilized for segmenting objects in an image [31]. In Figure 6, a sample result on a product photo can be seen. Firstly, the saliency heat map is calculated. Then by using a constant threshold (0.11 in our experiments), the object region is segmented. In the same figure, the segmented object can be seen in the rightmost image.

There are various methods to segment an object from an image without any prior information, in other words in an unsupervised manner. The reason we choose to use GBVS is simply because of its speed-accuracy trade-off [31]. In an extended study, it is possible to search for the most optimum method to segment an object in an unsupervised manner, however we find this effort beyond the scope of this study.

3) Supervised Segmentation: Deep Learning

Supervision is simply utilizing domain-specific data. Thus, compared to any unsupervised method, it is more susceptible to over-fitting. However, if there is sufficient training data, supervised methods are preferable most of the time. In order to segment the product in a supervised manner, we utilize the three deep segmentation methods, as explained in Section 3.

B. Image Features Benchmark

The general idea of BoVW is very simple: "representing an image as a fixed-length set of features". The so-called features consist of keypoints and descriptors. Keypoints denote the salient locations in the image, ideally invariant to transformations. Descriptor is the description "around" the keypoint. BoVW use both keypoints and descriptors to construct vocabularies and represent each image as a frequency histogram of features that are in the image. Similarity measures to a test image can be calculated using these frequency histograms, and thus a classification can be performed.

For a BoVW framework, the most important question is obviously "which feature/descriptor to use". Depending on the problem definition, imaging modality, performance requirements and computational budget, different methods can be used. For a comparison of local feature detectors and descriptors for visual object categorization, the reader may refer to [32]. In this study we employ 5 popular features, namely, GIST [33], histogram of gradients (HoG) [34], SIFT [35], SURF [36], and deep features, which are obtained using a special CNN layer, namely the Spatial Pyramid Pooling (SPP) Layer [37].

Among the aforementioned five features types, SIFT [35] and SURF [36] are, by definition, local; thus they are well-suited for BoVW. For HoG [34], the locality should be pre-defined, i.e. provided by the user for local a region with a fixed area. We use blocks of 16x16 on a uniform grid and obtain HoG features individually from each block.

GIST [33], on the other hand, is a global descriptor, more than a feature. It literally catches a "gist" of the scene by using multi-scale low level features. Hence it is incompatible for a BoVW model, and accordingly it is implemented out of the BoVW framework. We calculate the GIST descriptors for each training and test data. In consequence, by calculating the Euclidean distances between the GIST descriptors, we match a product photo to an image design file.

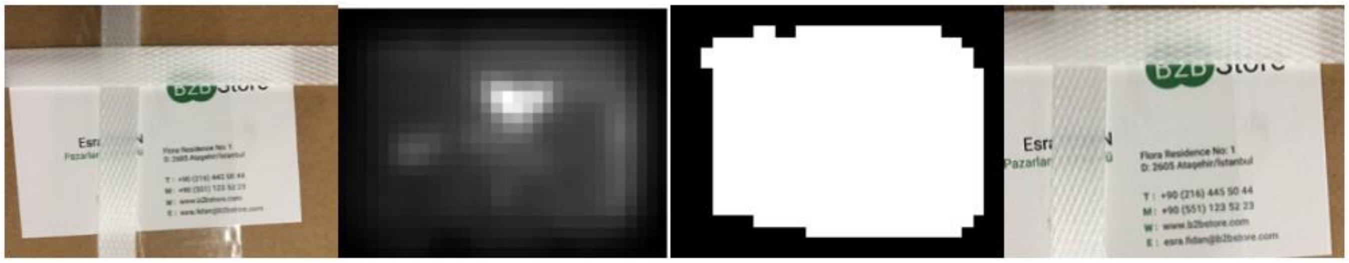


Fig.6. The objects in the product photos are segmented using the GBVS algorithm [30], which is selected as the unsupervised segmentation method for our segmentation benchmark.

Similarly, to GIST descriptor, the output of an SPP layer [37] does not need a histogramisation effort. This layer's output is already fixed-length. SPP collects activations from layers of different hierarchies, and concatenates them in a single fixed-length vector. For this purpose, we have utilized combinations of activations from different layers of the FCN32s network as an input to the SPP layer. By calculating the (weighted[2]) Euclidean distances between these vectors, matching is performed.

C. Hyper-Parameter Optimization

The proposed framework includes different methods with benchmarking of various intermediate steps (segmentation, feature extraction etc.). Thus there are many hyper-parameters that may affect the system performance. In this study, we optimize only the cluster number of the BoVW framework. We believe that this is the most important hyper-parameter, mainly because it is independent of the utilized segmentation, feature extraction or the classification steps. The number of clusters is the vocabulary size and thus the heart of a BoVW model. Accordingly, in the next section, we also provide results for different cluster numbers, thus showing the effect of vocabulary size on performance.

D. Classifier

The final block of the proposed framework is classification. BoVW provides a fixed length histogram representation for any image, and classification within this vector space is another step, which serves as the final decision of the system. In our framework, the final goal is to find the design file that matches the given product photo. Various classification methods can be employed for a BoVW system and the reader may refer to [38] for a detailed comparison.

BoVW concept is an adaptation of the bag of words (BoW) idea from natural language processing. In BoW, the so-called words are calculated by clustering the entire training set features into K number of subsets. Then, the number of each "word" in a document is counted, and a frequency histogram is created by using the word occurrences. We have the same concept in BoVW, but instead of words, we use image features. Image features can be anything, like salient regions in an image. We normalize frequency histograms to obtain probability distribution functions that represent the possibility of having a given visual word in an image. At this point we utilize Naïve Bayes algorithm. Occurrence of a word is assumed as an independent event (which is the Naïve part) and all feature probabilities are multiplied to find the matching probability of an image to another.

We have chosen Naïve Bayes algorithm as our classifier mainly because of two reasons. Firstly, it is fast and compatible with real-time processing. And secondly training requires a small amount of samples to estimate the model parameters. This is why it has always been an optimal [39] partner for BoVW. We believe that with another, maybe mathematically more complex classifier, our results may further be improved. For the sake of computation speed we employ Naïve Bayes for all BoVW experiments in this study, and leave the benchmarking of different classifiers to a future study.

V. Experimental Results

Before we present our comparative results with rigorous discussions in this section, the details of the experimental parameters are provided below.

A. Experimental Setup

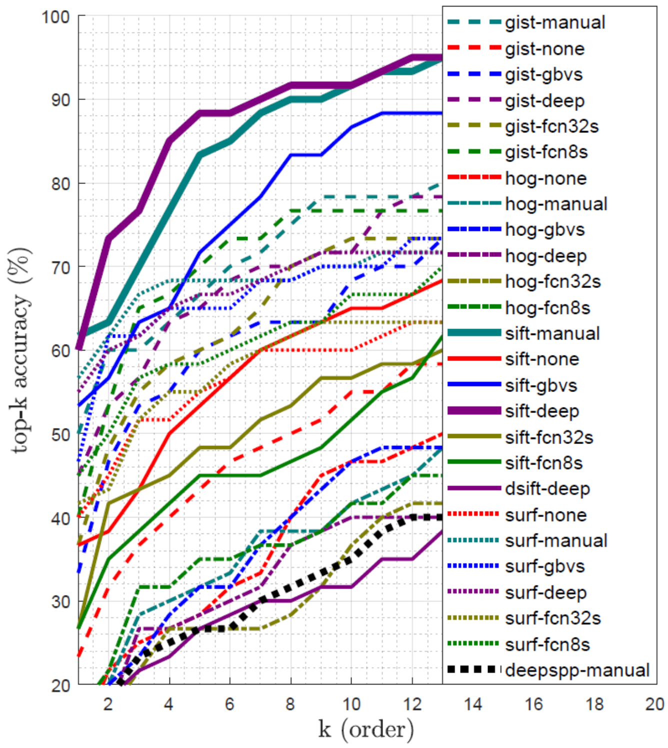
The main objective of our experiments is to find an optimal method to product photo and design file matching. The absolute value of matching success depends on the number of design files to be compared in the image set. If there are only, for example, 10 design files to match, regardless of the benchmarked methods, the success would be relatively higher compared to a case in which thousands of possible design file candidates exist.

In our experiments, we have selected the number of product photos to match as 60[3]. Moreover, we have selected the number of design files as 100, so that the 60 product photos will match some of the design files in this 100 element set, whereas the rest are just fillers[4].

[2] In an SPP layer [37], the activations are weighted by the size of the pooling layer.

[3] This parameter can be optimized with further experimentation. However, this would require running thousands of experiments with 26 different methods, which we have chosen leave to a future study.

[4] As explained in Section 2, the number design files that match a set of 60 product photos vary. A single page flyer has a single corresponding design file, whereas a two-sided business card has two design files matches for both sides. Thus the exact number of fillers (i.e. randomly selected non-pair design files) in a set of 100 design files changes according to the set of product photos.



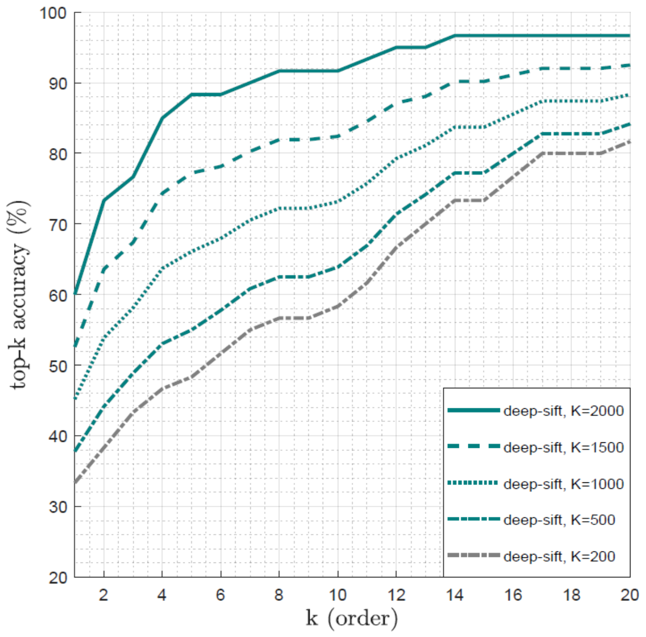


Fig.7. Top-k accuracy curves are depicted. In these curve x-axis denotes the order of match, i.e. the order of similarity of the actual pain in the training set. The y-axis denotes the average success for that order value. a) (left) Top-k accuracy curves for all method. b) (right) Top-k accuracy curves for *sift-deep* under varying number of clusters in BoVW.

Accordingly, we have created 1000 different randomly selected 60 product photo - 100 design file sets, using the total 2000 product photos and 3458 design files. For each benchmarked method, 1000 experiments are run using these 1000 different test cases. At each experiment, for each product photo, the order of match is recorded. For this purpose, when a product photo is matched to 100 design files for an experiment, the Naïve Bayes probabilities (or the Euclidean distances) are calculated and sorted. The rank of the probability of the corresponding design file is recorded as the order of that product photo's matching performance[5].

B. Results

For benchmarking, we employ 26 different methods using a combination of 6 different segmentation methods and 5 different features as presented in Section 4. For instance, the case in which SIFT features, that are obtained from only the product region segmented using VGG-Regression-Net, is referred to as "sift-deep". Or, the case, in which GIST global features are obtained from a manual segmented object region in a product photo, is called "gist-manual".

All 26 methods are tested for the same 1000 experiment sets and for each product photo in each experiment, the order of match is recorded. Using this order measure, we also calculate the average order of being matched. For example, for a specific method, the percentage of product photos that have smaller or equal order "k" is recorded as the top-k accuracy. For example, if the top-k accuracy for order k=10 is 95%, this shows that by only checking best 10 possible match results, it is possible to match the correct design file with 0.95 probability.

In Figure 7.a, the top-k accuracy for all methods are depicted. The numerical values are provided in Table III. The best accuracy is obtained with "sift-deep" method, for which SIFT features that are obtained from only the product region segmented via the VGG-Regression-Net, are fed to the BoVW framework. The sift-deep method also slightly outperforms sift-manual method, for which the segmentation is performed by the human operator. This indicates that human operators can make mistakes in product region annotation, whereas deep learning-based segmentation method can generalize these errors and perform much better.

After observing the success of SIFT, we have applied another version of the SIFT descriptor, namely the "Dense SIFT" (DSIFT) [40]. DSIFT is the same algorithm as the SIFT but it is run on a denser grid of locations. That is why DSIFT provide, on average, 10 times higher number of keypoints, compared to SIFT. In Figure 7.a, the performance of DSIFT, which is quite poor, can also be seen. This is, we believe, because of the fact that, increasing the number of keypoints does not support representation, but conversely creates more false alarm matches between feature clusters.

[5] For example, for a single product photo, we check the similarities to the given 100 design files in that experiment. The actual design file that matches the given product photo has the order 4, when all Naïve Bayes probabilities (or the Euclidean distances) are sorted. Then the order for this product photo in this experiment is simply 4.

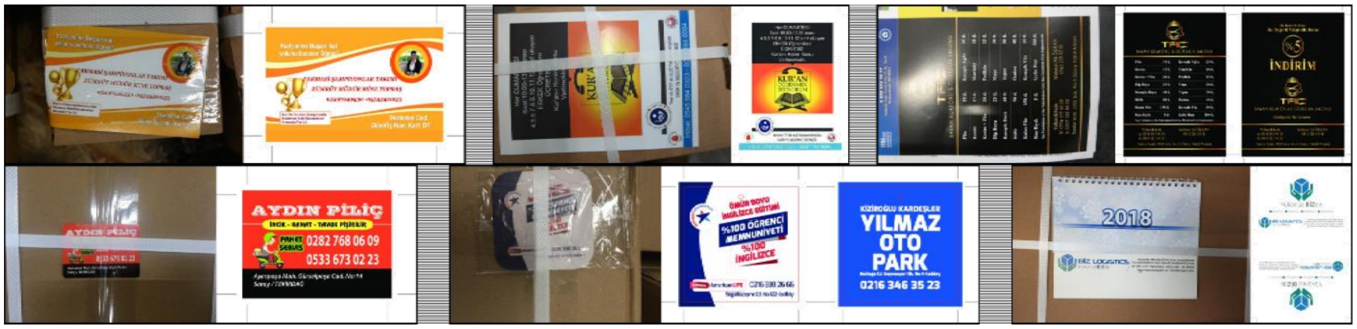


Fig.8. Sample results for the sift-deep method are seen. The samples on the top row are found with a perfect hit. The samples in the bottom row are matched with orders 8, 70 and 11 from left to right, respectively.

TABLE III
NUMERICAL RESULTS FOR EACH METHOD

Method / Order	1st	5th	10th	15th	20th
gist-manual	50.0%	66.7%	78.3%	81.7%	86.7%
gist-none	23.3%	43.3%	55.0%	63.3%	65.0%
gist-gbvs	33.3%	60.0%	68.3%	73.3%	75.0%
gist-deep	45.0%	65.0%	71.7%	80.0%	88.3%
gist-fcn32s	36.7%	60.0%	73.3%	76.7%	80.0%
gist-fcn8s	40.0%	70.0%	76.7%	76.7%	78.3%
hog-none	13.3%	28.3%	46.7%	50.0%	55.0%
hog-manual	15.0%	31.7%	41.7%	51.7%	58.3%
hog-gbvs	16.7%	31.7%	46.7%	50.0%	58.3%
hog-deep	11.7%	28.3%	40.0%	45.0%	55.0%
hog-fcn32s	8.3%	26.7%	36.7%	45.0%	53.3%
hog-fcn8s	16.7%	35.0%	41.7%	46.7%	50.0%
sift-manual	61.7%	83.3%	91.7%	95.0%	95.0%
sift-none	36.7%	53.3%	65.0%	76.7%	83.3%
sift-gbvs	53.3%	71.7%	86.7%	88.3%	90.0%
sift-deep	**60.0%**	**88.3%**	**91.7%**	**96.7%**	**96.7%**
sift-fcn32s	26.7%	48.3%	56.7%	65.0%	80.0%
sift-fcn8s	26.7%	45.0%	51.7%	66.7%	80.0%
dsift-deep	8.3%	26.7%	31.7%	45.0%	45.0%
surf-none	40.0%	55.0%	60.0%	66.7%	70.0%
surf-manual	56.7%	68.3%	70.0%	71.7%	71.7%
surf-gbvs	46.7%	65.0%	70.0%	76.7%	76.7%
surf-deep	55.0%	66.7%	71.7%	73.3%	76.7%
surf-fcn32s	41.7%	55.0%	63.3%	63.3%	63.3%
surf-fcn8s	45.0%	58.3%	66.7%	70.0%	73.3%
deepspp-manual	18.3%	26.7%	35.0%	40.0%	43.3%

The "deepspp" method, in which deep CNN features are fed to a SPP layer, performs poor, even with manual (perfect) segmentation of the product. Deep features carry abstract information, which may fill the semantic gap of any vision problem. Consequently, this poor performance of deep features was intriguing for us. For this reason, we have carried out extensive deep visualization experiments to uncover this issue. SPP creates activations from all (thousands even when only the deepest layer is used) neurons from the selected layers, most of which are unfortunately noise and are usually dropped out within the deep CNN. An SPP layer does not have the ability to select deep features according to their quality. That is why this method performs unsurprisingly poor within a BoVW framework, compared to a more selective and scale-invariant descriptor method, such as the SIFT. We have utilized different combinations of activations from various layers of the FCN32s network. The best performance was obtained when only the activations from the final max-pooled convolutional layer (pool5 - 13×13×512) was utilized. Only the resulting curve for this case is depicted in Figure 7.a. Our visualisation experiments clearly show that, regardless of the layer the deep features are obtained, selective activations are always overwhelmingly outnumbered by noisy, unselective and insignificant activations, which cannot lead to any semantic decision.

Consistent with our observations presented in Section 3, segmenting with FCN32s and FCN8s does not contribute the BoVW matching success positively. Semantic segmentation of product photos as if they are plain objects, is apparently not helping the feature selection operation enough.

In Figure 7.b, a parameter optimization effort for cluster numbers is depicted. As the number of clusters in BoVW increases, so as the success rates. In our tests, we tested up to 2000 clusters, which is the best case. This number can further be increased for higher success with a price of dramatically increasing our training time. The top-k accuracy for all methods that utilize BoVW in Figure 7.a are calculated using 2000 clusters, which is our optimal case.

In Figure 8, some sample results for the sift-deep method can be seen. The three samples on the top row are found with order 1, i.e. with a perfect hit. The samples in the bottom row are matched with orders 8, 70 and 11 from left to right, respectively. The mismatch cases are usually because of strong clutter or impaired design files.

The actual implementation of the system shows that the average "end-to-end" matching time for a product photo, using the sift-deep method in a regular, no-GPU desktop computer is less than 4 seconds, including communication delays[6]. This is a feasible duration for the operation considering that it is much faster than the operator manually searching for the product id, which takes about a minute for a single product photo. Still, the computation duration is open

[6] Image upload: 0.65s) + (deep segmentation: 2.15s) + (SIFT extraction: 0.69s) + (matching 0.23s) + (downloading the results 0.025s) = (TOTAL 3.74s on average). Feature extraction for design files are performed offline.

to improvement with better hardware and further software optimization.

VI. Conclusions

The real-time image matching framework we propose in this paper is hybrid in the sense that it uses both hand-crafted features and deep features obtained from a well-tuned DCNN. We concentrate on a specific application, that is to say, printing design to product photo matching. Since photographs of a printed product suffer many unwanted effects, such as uncontrolled shooting angle, uncontrolled illumination, occlusions, printing deficiencies in color, camera noise, optic blur, et cetera, we benchmark different hand-crafted and deep features to choose an optimal performance and propose a framework, in which deep learning is utilized with highest contribution.

Our results show that a deep segmentation supported BoVW method gives satisfactory results for the proposed operational concept. What is more, hand-crafted features, when deep segmented from a region of interest may lead to better results, compared to deep features, which may include overwhelming number of noisy and unselective activations.

Like all current problems in computer vision, image matching problem is also moving to the DCNN domain. On the other hand, DCNNs require millions of data and expensive hardware. That’s why we still need ingenious, practical and cheap industrial solutions until deep CNN hardware becomes standard in the following years.

In the meantime, we continue our studies on deep CNN structures, specifically on Siamese networks. We are currently building a Siamese network which can learn similarities between a product photo and design file pair. In order to train such a Siamese network, our analyses show that the current dataset must be significantly larger, compared to the dataset utilized in this study. Thus, we first focus our studies on enlarging our image set for training of such a system. Furthermore, a Siamese network will bring higher computation burden. For that matter, we are also studying embedded deep learning solutions that will utilize system-on-chip solutions for real-time operations.

Acknowledgment

This research was partially supported by the National Science Council of Turkey (TUBITAK - TEYDEB), with the project title “Customer-Information Matching, Invoicing and Barcoding of Post-Cut Products in Custom Printed Products using Image Processing Methods”, under the 1507 program and with the project number “7170364”. The authors would like to thank the owner of the project, Şans Printing Industries (bidolubaski.com) for their support and hard-work.

References

[1] T. Dharani, I. L. Aroquiaraj, "A survey on content based image retrieval," International Conference on Pattern Recognition, Informatics and Mobile Engineering, Tamilnadu, India, pp 485-490, 2013.

[2] Y. Liu, D. Zhang, G. Lu, W.Y. Ma, "A survey of content-based image retrieval with high-level semantics," Pattern Recognition, vol. 40. 1, 2007, pp 262 - 282.

[3] J. Sivic, A. Zisserman, "Video Google: a text retrieval approach to object matching in videos," International Conference on Computer Vision, 9th IEEE, Nice, France, vol. 2, pp 1470-1477, 2003.

[4] H. Wang, Y. Cai, Y. Zhang, H. Pan, W. Lv, H. Han, "Deep learning for image retrieval: What works and what doesn't," International Conference on Data Mining Workshop, Washington, DC, US, pp 1576-1583, 2015.

[5] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, H. Lipson, "Understanding neural networks through deep visualization," Deep Learning Workshop, International Conference on Machine Learning, Lille 2015, pp 2015.

[6] K. Simonyan, A. Zisserman, "Very deep convolutional networks for large-scale image recognition" International Conference on Learning Representations Workshops, San Diego, CA, US, 2015.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, "Going deeper with convolutions," IEEE Conference on Computer Vision and Pattern Recognition, ,Boston, MA, US, pp 1-9, June 2015.

[8] A. Krizhevsky, I. Sutskever, G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Advances in Neural Information Processing Systems (NIPS), Lake Tahoe, NV, US, pp 1097-1105, 2012.

[9] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," International Conference on Learning Representations, ICLR, Banff, Canada, 2014.

[10] A. Babenko, A. Slesarev, A. Chigorin, V. Lempitsky, "Neural codes for image retrieval," European Conference in Computer Vision, Zurich, Switzerland, pp 584-599, 2014.

[11] V. Chandrasekhar, J. Lin, O. Morère, H. Goh, A. Veillard, "A practical guide to CNNs and fisher vectors for image instance retrieval," Signal Processing, vol. 128, pp 426-439, 2016.

[12] I. Melekhov, J. Kannala, and E. Rahtu, "Siamese network features for image matching," International Conference on Pattern Recognition (ICPR), Cancún, Mexico, pp 378-383, 2016.

[13] Y. Taigman, M. Yang, M. Ranzato, L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, US, pp 1701-1708, 2014.

[14] T. Lin, Y. Cui, S. Belongie, J. Hays, "Learning deep representations for ground-to-aerial geolocalization," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, US, pp 5007-5015, 2015.

[15] D. Cai, X. Gu, C. Wang, "A revisit on deep hashings for large-scale content based image retrieval," ArXiv.CoRR, vol. abs/1711.06016, pp 1-11, 2017.

[16] R. Datta, J. Li, J. Z. Wang, "Content-based image retrieval: Approaches and trends of the new age”, ACM SIGMM International Workshop on Multimedia Information Retrieval, New York, NY, USA, pp. 253-262, 2005.

[17] P. Clough, H. Müller, T. Deselaers, M. Grubinger, T. Martin Lehmann, J. R. Jensen, W. Hersh, "The CLEF 2005 Cross-Language Image Retrieval track," International Conference of the Cross-Language Evaluation Forum for European Languages, Vienna, Austria, vol. 1171, pp. 535-557, 2005.

[18] G. Schaefer, "UCID-RAW - a colour image database in raw format," European Congress on Computational Methods in Applied Sciences and Engineering, Porto, Portugal, pp 179-184, 2017.

[19] T. Ahmad, P. Campr, M. Cadik, G. Bebis, "Comparison of semantic segmentation approaches for horizon/sky line detection," International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, US, pp 4436-4443, 2017.

[20] F. Jiang, A. Grigorev, S. Rho, Z. Tian, Y. Fu, W. Jifara, A. Khan, S. Liu, "Medical image semantic segmentation based on deep learning," Neural Computing and Applications, vol 29. 5, pp 1257–1265, 2018.

[21] M. Siam, S. Elkerdawy, M. Jägersand, S. Yogamani, "Deep semantic segmentation for automated driving: Taxonomy, roadmap and challenges," IEEE International Conference on Intelligent Transportation Systems, Yokohama, Japan, pp. 1-8, 2017.

[22] I. Ulku, E. Akagunduz, "A Survey on Deep Learning-based Architectures for Semantic Segmentation on 2D images," ArXiv.Corr, vol. abs/1912.10230, pp 1-20, 2019.

[23] M. H. Saffar, M. Fayyaz, M. Sabokrou, M. Fathy, "Semantic video segmentation: A review on recent approaches," ArXiv.Corr, vol. abs/ 1806.06172, pp 1-24, 2018.

[24] H. Yu, Z. Yang, L. Tan, Y. Wang, W. Sun, M. Sun, Y. Tang, "Methods and datasets on semantic segmentation: A review," Neurocomputing, vol. 304, pp. 82 - 103, 2018.

[25] Y. Guo, Y. Liu, T. Georgiou, M. S. Lew, "A review of semantic segmentation using deep neural networks," International Journal of Multimedia Information Retrieval, vol. 7, pp. 87-93, Jun 2018.
[26] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. G. Rodríguez, "A review on deep learning techniques applied to semantic segmentation," ArXiv.Corr, vol. abs/1704.06857, pp 1-19, 2017.
[27] E. Shelhamer, J. Long, T. Darrell, "Fully convolutional networks for semantic segmentation," IEEE Transactons on Pattern Analysis and Machine Intelligence, vol. 39, pp. 640-651, Apr. 2017.
[28] A. Vedaldi, K. Lenc, "MatConvNet: Convolutional neural networks for Matlab," ACM International Conference on Multimedia, Brisbane Australia, pp. 689-692, 2015.
[29] H. Zhang, J. E. Fritts, and S. A. Goldman, "Image segmentation evaluation: A survey of unsupervised methods," Computer Vision and Image Understanding, vol. 110. 2, pp. 260 - 280, 2008.
[30] J. Harel, C. Koch, and P. Perona, "Graph-based visual saliency," Advances in Neural Information Processing Systems, Vancouver, Canada, pp 545-552, 2006.
[31] Y. Xu, J. Li, J. Chen, G. Shen, Y. Gao, "A novel approach for visual saliency detection and segmentation based on objectness and top-down attention," International Conference on Image, Vision and Computing, Chengdu, China, pp 361-365, 2017.
[32] J. Lankinen, V. Kangas, J. Kamarainen, "A comparison of local feature detectors and descriptors for visual object categorization by intra-class repeatability and matching," International Conference on Pattern Recognition, Tsukuba, Japan, pp 780-783, 2012.
[33] A. Oliva, A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," International Journal of Computer Vision, vol. 42, pp. 145-175, 2001.
[34] N. Dalal, B. Triggs, "Histograms of oriented gradients for human detection," IEEE Computer Vision and Pattern Recognition, San Diego, CA, US, pp. 886-893, 2005.
[35] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, vol. 60, pp. 91-110, 2004.
[36] H. Bay, T. Tuytelaars, L. Van Gool, "SURF: Speeded up robust features," European Conference on Computer Vision, Graz, Austria, pp 404-417, 2006.
[37] K. He, X. Zhang, S. Ren, J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," Arxiv.CoRR, vol. abs/1406.4729, pp 1-13, 2014.
[38] C. Hentschel, H. Sack, "Does one size really fit all?: Evaluating classifiers in bag-of-visual-words classification," International Conference on Knowledge Technologies and Data-driven Business, New York, NY, USA, pp. 7:1-7:8, 2014.
[39] L. I. Kuncheva, "On the optimality of naïve bayes with dependent binary features," Pattern Recognition Letters, vol. 27, pp. 830-837, 2006.
[40] K. Lenc, A. Vedaldi, "Understanding image representations by measuring their equivariance and equivalence," IEEE Conference On Computer Vision and Pattern Recognition, Boston, MA, US, pp. 991-999, 2015.

BIOGRAPHIES

ALPER KAPLAN was graduated from Yeditepe University, Computer Engineering Department in 2016. Currently, he is a research assistant and a master's student at Cognitive Science Program of Yeditepe University. He continues his studies by focusing on his thesis about machine-generated music. In addition to his academic studies, he works as a software developer in an e-commerce company. His interests consist of machine learning, machine-generated art/music, music theory, natural language processing, image processing.

ERDEM AKAGÜNDÜZ received his Ph.D. degree in Middle East Technical University, Electrical and Electronics Engineering Department, Ankara, Turkey, in 2011. He worked a research and teaching assistant with the METU EEE CVIS Lab. from 2001 to 2008. Between 2009-2016 he worked as an algorithm design engineer with ASELSAN. Before starting his academic career in Turkey, as a post-doctoral research associate, we visited the University of York, UK, in 2016. He is currently an Assistant Professor in Çankaya University, Electrical and Electronics Engineering Department, Ankara, Turkey. His research interests include deep learning, computer vision and sound processing.