Information Fusion 57 (2020) 89–101

Contents lists available at ScienceDirect

Information Fusion

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

Neural architecture search for image saliency fusion

Simone Bianco, Marco Buzzelli*, Gianluigi Ciocca, Raimondo Schettini

University of Milano – Bicocca, Department of Computer Sciences Systems and Communications, viale Sarca 336, Milan 20126, Italy

ARTICLE INFO

Keywords:
Saliency fusion
Evolutionary algorithms
Neural architecture search

ABSTRACT

Saliency detection methods proposed in the literature exploit different rationales, visual clues, and assumptions, but there is no single best saliency detection algorithm that is able to achieve good results on all the different benchmark datasets. In this paper we show that fusing different saliency detection algorithms together by exploiting neural network architectures makes it possible to obtain better results. Designing the best architecture for a given task is still an open problem since the existing techniques have some limits with respect to the problem formulation, to the search space, and require very high computational resources. To overcome these problems, in this paper we propose a three-step fusion approach. In the first step, genetic programming techniques are exploited to combine the outputs of existing saliency algorithms using a set of provided operations. Having a discrete search space allows us a fast generation of the candidate solutions. In the second step, the obtained solutions are converted into backbone Convolutional Neural Networks (CNNs) where operations are all implemented with differentiable functions, allowing an efficient optimization of the corresponding parameters (in a continuous space) by backpropagation. In the last step, to enrich the expressiveness of the initial architectures, the networks are further extended with additional operations on intermediate levels of the processing that are once again efficiently optimized through backpropagation.

Extensive experimental evaluations show that the proposed saliency fusion approach outperforms the state-of-the-art on the MSRAB dataset and it is able to generalize to unseen data of different benchmark datasets.

1. Introduction

According to [1], "*Visual salience (or visual saliency) is the distinct subjective perceptual quality which makes some items in the world stand out from their neighbors and immediately grab our attention*". The human vision system is able to efficiently detect salient areas in a scene and further process them to extract high-level information [2,3]. Visual saliency has been primarily studied by neuroscientists, cognitive scientists and recently has received attention from other research communities working in the fields of computer vision, computer graphics and multimedia e.g. [4]. In the area of multimedia and computer vision, visual saliency can be used to emphasize object-level regions in the scene that can serve as a pre-processing step for scene recognition [5,6], object detection [7,8], segmentation [9], and tracking [10]. It can also be exploited for image manipulation and visualization in applications such as image retargeting [11], image collage [12], and non-photorealistic rendering [13]. Moreover, in multimedia application saliency can be exploited for image and video summarization [14–16], enhancement [17], retrieval [18], and image quality or aesthetic assessment [19,20].

Saliency detection methods can be divided into two categories: bottom-up and top-down. Bottom-up methods are stimuli-driven [21]. The saliency is usually modeled by local or global contrast on hand-crafted visual features and knowledge about human visual attention is embedded in the model exploiting some heuristic priors such as background [22], compactness [23], or objectness [24]. With these methods no explicit information about the semantics of the salient regions is provided but it is indirectly embedded via prior assumptions that are made on the location, shape or visual properties of the salient regions to be detected. Bottom-up methods can be considered general purpose.

Top-down saliency methods are designed to find regions in the images that are relevant for a given task. They are often also referred to as task-driven approaches. These methods usually formulate the saliency detection as a supervised learning problem [25]. The rationale of top-down saliency methods is to identify image regions that belong to a pre-defined object category [26]. For this reason, these methods are theoretically more robust for identifying salient regions in cluttered backgrounds where bottom-up methods may fail. Top-down approaches rely on the use of training data to build the detection model. They can be very robust for the specific task on which they are trained but may not generalize well to other tasks.

In order to make the detection more robust and to improve the generalization capabilities, saliency methods often integrate different features

* Corresponding author.
E-mail addresses: simone.bianco@unimib.it (S. Bianco), marco.buzzelli@unimib.it (M. Buzzelli), gianluigi.ciocca@unimib.it (G. Ciocca), raimondo.schettini@unimib.it (R. Schettini).

https://doi.org/10.1016/j.inffus.2019.12.007
Received 25 March 2019; Received in revised form 1 October 2019; Accepted 9 December 2019
Available online 10 December 2019
1566-2535/

[27] that can be both hand-crafted or learned by Convolutional Neural Networks (CNNs) [28–30], or fuse saliency maps generated from different methods [31]. However, the feature definition and selection, and the combination strategies are usually empirically designed.

Since multiple observers may consider salient different regions in the scene depending on the scene context and/or on the observer's cultural background, saliency detection is an ill-posed problem [22,32]. Saliency detection methods proposed in the literature exploit different rationales, visual clues, and assumptions but as demonstrated by the experiments in [33], there is no best overall saliency detection algorithm that is able to achieve good results on all the different benchmark datasets.

In our previous works [34,35], we have exploited genetic programming (GP) to build the rationale with which to combine the binary outputs of several change detection algorithms. By using a-priori defined unary, binary and n-ary operators, the GP approach automatically combined the inputs using the provided operators and built an optimal, task-driven, solution (i.e. program) in the form of a hierarchical tree structure.

In this work we want to further investigate and extend this approach to combine graylevel saliency maps, a domain we first addressed in [36]. We first create a candidate solution for combining the saliency maps using GP with a set of operations whose parameters are a-priori fixed. To further improve this solution, we should also tune these parameters, but they cannot be easily (or efficiently) optimized within the GP framework. In order to optimize the parameters, we use the candidate solution obtained by the GP as a blueprint upon which to design the architecture of a backbone Convolutional Neural Network. Within the CNN optimization framework, it is now easier and much more efficient to search for the optimal parameters of the operations of the GP solution. Another important advantage of the implementation of the backbone CNN is that the proposed solutions can be evaluated and then we can easily and safely create deeper variants of the CNN by including other operations (e.g. post-processing) on intermediate results. These operations, initialized as identities, are further optimized or can be completely ignored by the CNN during training.

The extensive experiments on benchmark datasets, both qualitative and quantitative, validate the effectiveness of the proposed fusion strategy.

Finally, beyond the focus on saliency estimation for the scope of this paper, the proposed information fusion technique can be considered a general purpose method, with possible applications to other fields such as change detection [35] and semantic segmentation [37].

2. State of the art

2.1. Saliency detection algorithms

Borji et al. [33] benchmarked 41 different saliency detection algorithm each based on different assumptions and heuristics. For example, Li et al. [38] compute saliency from the perspective of image reconstruction error of background images generated at different level of details. A graph-based approach is used instead by Yang et al. [39]. Again, superpixels are the base for the saliency computation. Foreground and background region queries are used to rank each image regions using a graph-based manifold ranking. Other graph-based approaches are the ones presented by Jiang et al. [40] and Aytekin et al. [41]. In [40] the salient regions are detected as those regions that, in an absorbing Markov chain on an image graph model, require the most time to be absorbed in the background nodes. In [41] saliency is determined from a graph built on superpixels and by optimizing a criterion related to the image boundary, local contrast and area information. Zhu et al. [42] subdivide the images into patches, and an image patch is considered not salient only when the region it belongs to is heavily connected to the image boundary. Since computing superpixels is often time consuming, Zhang et al. [43] presented an alternative approach based on a fast Minimum Barrier Distance to measure a pixels connectivity to the image boundary that is able to perform saliency detection at 80 fps. The previous approaches exploit the boundary prior assumption for the saliency. The work by Cheng et al. [44], relies on the contrast assumption instead. The saliency of each image region is carried out by simultaneously evaluating global contrast differences and spatial coherence with nearby regions. Color is also an important cue for locating salient regions. Kim et al. [45] represented the saliency as a linear combination of high-dimensional color space where salient regions and backgrounds can be distinctively separated.

Multiple cues can be also exploited in the definition of salient regions. For example, Liu et al. [46] integrated global contrast, spatial sparsity, and object prior with regional similarities to generate initial saliency measure for image regions. Another approach that uses multiple features for saliency detection is presented by Wang et al. [47]. This approach considers different regional features (contrast, appearance, geometry) that are computed in a multi-level segmentation schema.

Compared with traditional approaches CNN-based ones are able to process images extracting information at different levels of details. They can automatically learn what is the relevant information within an image given a specific task. This make them most suitable and effective for designing top-down supervised saliency detection algorithms. Zhao et al. [48] designed an end-to-end CNN framework composed of two-branches network where one branch analyzes the image at a local level (local context), while the other analyzes it at a global level (global context). Similarly, a three-branches fully convolutional neural network is proposed by Li and Yu [49] where each branch processes an image segmented at a different scale.

One of the drawback of the previous methods is that they work on a pre-segmented image (usually using super-pixel approaches), and the resulting saliency maps are thus not pixel-precise. To mitigate this problem, Li and Yu [28] devised an end-to-end deep contrast network consisting in two streams that produce two saliency maps, one at pixel-level and the other at superpixel-level that are fused together. Li et al. [50] incorporated the objectness cue into a deep learning-based saliency model by designing a multi-task learning scheme to explore the correlations between saliency detection and semantic image segmentation. Liu and Han [51] propose to learn feature representations and various global structured saliency cues by concatenating two different networks: the first roughly localizes the relevant object in the image; the second implements a hierarchical recurrent convolutional neural network to refine the previous saliency map by incorporating local contexts. Lee et al. [52] propose to exploit both low level and high level features in a unified deep learning approach. The rationale is that hand-crafted features can provide complementary information to enhance performance of saliency detection algorithms.

Using features from different layers in the network architecture provides multi-scale feature maps that can be exploited for an efficient salient object detection. Example of algorithms using this approach are those proposed by Hou et al. [29], which uses a combination of side-outputs and short connections, and Bianco et al. [30] which uses a Fully Convolutional Network that builds a semantically aware internal representation of what is a salient object.

Recurrent network architectures can help reducing prediction errors by iteratively integrating contextual information which is important for saliency detection. For example Wang et al. [53] presented a recurrent fully convolutional network that is used to automatically learn to refine the saliency map by correcting its previous errors. Liu and Han [54] fine tuned a pre-trained CNN, on the ImageNet large scale dataset, on saliency data and long short-term-memory (LSTM) is applied to model the global context. A Recurrent Localization Network (RLN) is proposed by Wang et al. [55] where salient objects are localized exploiting contextual information in a weighted response map. Wang et al. [56] also propose a hierarchy of convolutional LSTMs to generate saliency maps using an Attentive Saliency Network that has learned to locate salient objects from detected fixation maps.

Recent works tackle the problem of saliency detection from a different perspective with respect to previous works. For example [57] introduces a game-theoretic approach by formulating the saliency detection problem as a non-cooperative game. Zeng et al. [58] trained a DNN as an embedding function to map pixels and the attributes of the salient/background regions of an image into the same metric space that are used in an iterative process to classify the pixels as salient/background. Finally, Azaza et al. [59] incorporated into an object proposal method the importance of the object's immediate context using a context proposal algorithm in order to improve saliency detection.

2.2. Saliency information fusion

In the literature some works specifically tackle the problem of information fusion in the context of saliency detection. Combining different information is a possible way to improve saliency detection.

Some works use different cues in order to capture the many aspects of a salient region. This can be considered as an early fusion information approach since it combines features for generating the saliency map. For example Xu et al. [60] proposed to generate a saliency map for video frames by a combination of three different maps together: a static saliency map, a motion saliency map and a top-down saliency map. A linear method with adaptive coefficients is used to fit different types of videos. In the context of high dynamic range videos, Banitalebi-Dehkordi et al. [61] exploited several hand-crafted saliency features such as motion, color, brightness intensity, and texture orientations to generate conspicuous maps that are then fused in a single saliency map with a Random Forest learning algorithm.

Also late fusion information, i.e. directly combining saliency maps generated by existing algorithms, could be exploited for improving the overall performance of the saliency detection algorithm. The idea is that the saliency detection algorithms often complement each other, and thus by combining their results we can leverage the different rationales behind these approaches. An example of this is the work by Mai et al. [31]. They propose a data-driven saliency aggregation approach that must be method-aware and individual image-aware in order to adapt it to the specificity of the available data. This is achieved by considering image similarity (similar images should have similar saliency maps) and training the model using a Conditional Random Field algorithm. A similar approach is also described in [62] where deep features are used for comparing images instead of hand-crafted ones. Given the set of similar images and their saliency maps, the final output is a weighted combination of these maps where the weights are set according to the accuracy of each saliency detection algorithm. Another way to evaluate the contribution of the different outputs of saliency detection algorithms is proposed by Jiang et al. [63]. A no-reference saliency map quality metric is first defined, and then used to assess the saliency maps to be combined. The final saliency map is constructed by weighted-averaging the best saliency maps according to the metric. Wei et al. [64] consider the saliency fusion problem as a statistics inference process and propose an unsupervised saliency detection algorithm. They combine saliency maps generated by state-of-the-art algorithms and structural information obtained by superpixels with a Dempster–Shafer theory based saliency fusion framework.

2.3. Optimizing the network architecture design

In recent years, with the increasing complexity of neural network architectures, many automatic approaches to design or optimize neural networks have been proposed, either in terms of internal architecture or with respect to the network hyper-parameters. Most of these approaches are based on evolutionary algorithms such as genetic algorithms. Benardos et al. [80] determined the best neural network architecture, for a given task, using genetic algorithms optimizing the performance and the complexity of the network. The complexity is expressed in terms of number of layers, the number of neurons in each layer, the activation function in each layer, and the optimization function. Stanley and Miikkulainen [65] designed the NEAT algorithm aimed at optimizing small recurrent networks by evolving their topology and weights. An improved version of the NEAT algorithm, named CoDeepNEAT, has been recently presented [66]. It is aimed at optimizing deep learning architectures by extending existing neuro-evolution methods to topology, components and hyper-parameters. Sugurama et al. [67] introduced CGP-CNN which exploits Cartesian Genetic Programming to encode and evolve the network architecture.

Another approach for the design of network architecture is Reinforcement Learning. Baker et al. [68] use Q-learning by training an agent whose goal is to discover CNN architectures that performs well on a given machine learning task. The learning agent sequentially picks layers of a CNN model to build the final model. Zoph and Le [69,70] defined a Neural Architecture Search algorithm based on reinforcement learning. The authors defined a search space of different types of convolutional and pooling layers with different possible settings for each and ways to combine them. They train a recurrent neural network (NASNet) to generate the model descriptions of neural networks maximizing the expected accuracy of the generated architectures for a given task.

Genetic and evolutionary algorithms are able to find network architectures by selecting the best solution among a finite set of possibilities. They can combine predefined blocks and parameters by optimizing an objective function. The discrete search space allows these algorithms to efficiently perform the search for the best solution. Algorithms based on reinforcement learning, also explore a discrete space when searching for the network topology, but are more powerful for parameters (i.e. weights) optimization of the network thanks to backpropagation. However, they suffer from very high requirements with respect to the computational resources. As an example, for the CIFAR-10 classification problem, the NASNet approach was trained across 500 P100 GPUs over 4 days resulting in 2000 GPU-hours and 20,000 networks evaluated. Similarly, a recent approach proposed by Real et al. [71], and based on the same search space but using genetic algorithm, required the power of 450 K40 GPUs working for 7 days to generate the final network architecture (AmoebaNet-A). These drawbacks prompted researchers to investigate more efficient approaches to accelerate the search for good CNN structures within the search space [72,73].

3. Proposed method

Our proposed saliency estimation approach aims at combining the advantages of Genetic Programming with those of Convolutional Neural Networks. With our approach, we design and optimize GP-generated solutions for saliency estimation in three steps. In the first step, Genetic Programming techniques are exploited to combine existing saliency maps using a set of provided operations. The output of this step is a fusion tree that encodes the optimal fusion strategy with respect to the defined objective function and operations. In the second step, the obtained fusion tree is converted into a backbone CNN where operations are all implemented with differentiable functions. The network's parameters (defined in a continuous space) are efficiently optimized using backpropagation. Finally, in the third step, the backbone CNN is extended with additional operations on intermediate layers. These additional operations, once again optimized through backpropagation, are introduced to enrich the expressiveness of the initial architecture.

3.1. Creation of the fusion tree

The first step of our method consists in finding the operations that produce the best fusion of the input saliency estimation algorithms considered exploiting GP [74]. The candidate solutions found by GP, are encoded as trees of operations built using a set of terminal symbols T

Table 1
The set of functional symbols used in GP with their name, their n-arity, the domain on which they operate, and their corresponding operators.

Name	Operation	Inputs	Domain	Effect
ERO	Erosion	1	Spatial	Morphological erosion with 5×5 square structuring element
DIL	Dilation	1	Spatial	Morphological dilation with 5×5 square structuring element
MF	Median	1	Spatial	2D median filter with 5×5 kernel
MIN	MinStack	≥ 2	Stack	1D minimum filter
MAX	MaxStack	≥ 2	Stack	1D maximum filter
MED	MedianStack	≥ 2	Stack	1D median filter
AVG	AverageStack	≥ 2	Stack	1D average filter
LOG	Quantization	1	Pixel	Generalized logistic non-linearity

and a set of non-terminal or functional symbols F. The solution space of all the possible fusion trees is explored by GP as follows:

1. Randomly create the initial population
2. Repeat until the maximum number of iterations is reached:
   (a) Calculate the fitness value of the individuals
   (b) Select individuals based on fitness value
   (c) Apply genetic operators to obtain new individuals
   (d) Replace the old population with the new one, eventually copying the fittest individual(s)
3. Return the best individual

More in detail, given a set of n input saliency estimation algorithms $S = \{S_k\}_{k=1}^{n}$, the candidate solutions evolved by GP are built using the set of functional (or non-terminal) symbols F and the set of terminal symbols $T = S$. The functional symbols correspond to operations performed on the inputs. We incorporate into the GP framework the list of operators given in Table 1 along with their functional symbols. They can be grouped on the basis of the support on which they operate: the first group operates in the 2D spatial neighborhood of the pixels belonging to the same saliency map; the second group operates on stacks of pixels across different saliency maps; the third one operates on individual pixels without considering any neighborhood.

Following the procedure defined in [29], we define the fitness function as a weighted average of two different measures, based on Mean Absolute Error (MAE) and F-measure (F_β):

$$MAE = \frac{1}{|I|}\sum_{i\in I}\frac{1}{|C|}\sum_{c\in C}\left|PR_{i,c} - GT_{i,c}\right| \tag{1}$$

$$F_\beta = \max_{t\in T}\frac{\left(1+\beta^2\right)\frac{1}{|I|}\sum_{i\in I} Precision_i(t)\cdot\frac{1}{|I|}\sum_{i\in I} Recall_i(t)}{\beta^2\cdot\frac{1}{|I|}\sum_{i\in I} Precision_i(t) + \frac{1}{|I|}\sum_{i\in I} Recall_i(t)} \tag{2}$$

Where I is the set of images, C the set of coordinates for every given image, and T the set of possible thresholds. PR and GT are, respectively, the saliency prediction and ground truth. Expansion of *Precision* and *Recall* is here omitted for brevity reasons. β is set to $\sqrt{0.3}$ as suggested in [29]. While MAE is computed as an average over all individual images, F_β is selected according to the best binarization threshold on precision and recall values that are in turn averaged across all images.

Mathematically the fitness function is defined as

$$f = w_1 \cdot \sqrt{F_\beta} + w_2 \cdot \mathrm{MAE}, \tag{3}$$

with $w_1 = 1$, and $w_2 = -0.01$. The weights are chosen with opposite signs since F_β should be maximized while MAE minimized: in this way fittest individuals have larger f values. The magnitudes of the weights have been empirically chosen on the basis of the difficulty to optimize the corresponding measures, and specifically with F_β driving the optimization process.

3.2. Creation of the backbone and extended CNN

The fusion tree generated from the genetic programming step involves different types of operations (Table 1). The intention is to further refine some of these operations by exploiting backpropagation optimization. In order to do so, it is necessary to build the corresponding backbone CNN architecture, by representing all tree nodes with differentiable functions, for the gradients to correctly flow to the input nodes. The resulting neural network can also be augmented with further operations, defining an extended CNN. Both backbone CNN and extended CNN are designed as a generalization of the behavior defined by GP, i.e. in their initialization state they produce the same output. Via backpropagation, then the CNNs can be optimized to generate more accurate solutions.

3.2.1. Conversion from fusion tree into CNN

Details of the mapping between operations in the fusion tree and the corresponding CNN are presented in the following.

- *MaxStack* and *MinStack*:
  MaxStack is implemented as 3-dimensional max-pooling with kernel of spatial size 1×1, and depth D equals to the number of input channels.
  MinStack is implemented through 3D min-pooling, i.e. by changing the sign to the input and output of MaxStack as just defined.
- *Dilation* and *rosion*:
  Greyscale dilation with a squared structural element corresponds to 2-dimensional max-pooling. The kernel has spatial size 5×5 as in the genetic programming setup.
  Erosion is obtained through 2D min-pooling, i.e. by changing the sign to the input and output of dilation.
- *AverageStack*:
  Stacked-channel average is implemented as a convolutional layer with one kernel of size 1×1, depth D equal to the number of input channels determined by GP, and bias initialized at 0. For the extended CNN, two different behaviors are imposed, depending on the location of the node:
  - Input node:
    Convolutional layer with a bank of M filters with size $N \times N$, and depth S equal to the number of all input saliency algorithms. The filters are initialized to all zeros, except the middle value of the D kernel channels selected by GP. The layer is followed by a ReLU non-linearity, and by a compacter module that maps M channels to 1, which is the expected size for the subsequent operation. The compacter is based on a channel-wise average, minimum, or maximum operation. The actual final operation is experimentally determined.
  - Intermediate node:
    Convolutional layer with one kernel of spatial size $N' \times N'$, and depth D. The best value for N is defined experimentally.

  The hyperparameters of the input and internal nodes have been determined through a greedy search algorithm, as reported in Section 4.2.
  All the weights are optimized via backpropagation.
- *Quantization*:

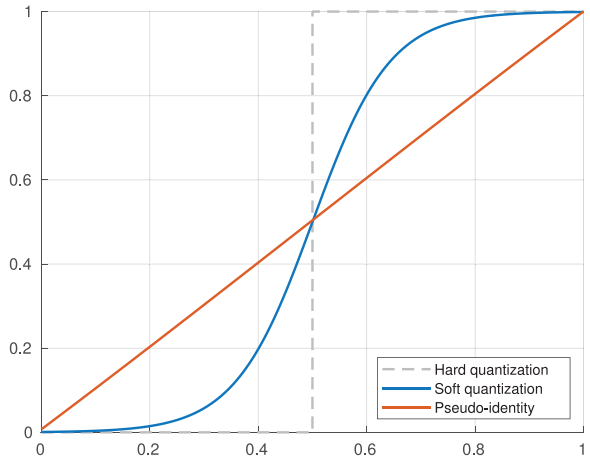

Fig. 1. Two specialization of the 7-parameter logistic function described in Eq. (4) (soft quantization and pseudo-identity) compared to a hard quantization function.

Quantization is approximated and generalized with a logistic function in its 7-parameter formulation:

$$y = (K - A) \cdot (Q \cdot \exp^{-B \cdot (x-G)} + C)^{(-1/V)} + A \tag{4}$$

Hard quantization would impair the gradient flow during backpropagation, as its derivative is always null where defined. It is therefore replaced with a soft quantization implemented as a sigmoid centered in 0.5, defined to cover the same co-domain of hard quantization, without ever reaching completely horizontal or vertical slope:

$$\begin{array}{llll} A = 0.0 & B = 14.0 & V = 1.0 & C = 1.0 \\ K = 1.0 & Q = 1.0 & G = 0.5 & \end{array}$$

The effect of this approximation is shown in Fig. 1, and the whole set of parameters is then optimized through backpropagation.

- *MedianStack*:
  Directly implemented as the median operation applied across the depth dimension: the value at each image coordinate is determined as the median at the corresponding coordinates from input channels.
- *Median*:
  The median filter is implemented in a two-step processing. First, the input is unfolded once per each spatial dimension, i.e. sliding windows are explicitly replicated using unitary stride and kernel size 5×5. Then, the copies are unrolled along an extra dimension, where the median operation is applied, effectively producing one value per sliding window location.

At the end of the tree, a final non-linearity is introduced using a logistic function in the form of Eq. (4), followed by a clamping of the values between 0 and 1. In this case, the logistic is initialized as an approximation of the identity function:

$$\begin{array}{llll} A = -1.557 & B = 0.666 & V = 0.354 & C = 0.817 \\ K = 1.165 & Q = 19.747 & G = -5.856 & \end{array}$$

The goal of this configuration, shown in Fig. 1, is to provide a neutral additional operation, which if needed can be exploited and adapted to a non-linearity by the CNN optimization. Clamping replicates the eventual cut-off given by saving the image to file before final application or evaluation. It also allows the logistic curve to create strong distortions inside the [0,1] range, preventing instead the loss function from penalizing values outside of it.

In the extended CNN, the same "logistic + clamping" block is introduced at each node of the computation (i.e. after each operation).

An example application of the mapping from fusion tree to backbone and extended CNNs is provided in Fig. 5.

Table 2
Comparison of traditional statistics computation (True Positives *TP*, False Positives *FP*, False Negatives *FN*, True Negatives *TN*) with threshold set to 0.5, and continuous statistics computation. Highlighted in boldface the different behavior between the two approaches.

PR	*GT*	*TP*	*FP*	*FN*	*TN*	*TP′*	*FP′*	*FN′*	*TN′*
0.0	0	0	0	0	1	0.0	0.0	0.0	1.0
0.4	0	0	**0**	0	**1**	0.0	**0.4**	0.0	**0.6**
0.6	0	0	**1**	0	**0**	0.0	**0.6**	0.0	**0.4**
1.0	0	0	1	0	0	0.0	1.0	0.0	0.0
0.0	1	0	0	1	0	0.0	0.0	1.0	0.0
0.4	1	**0**	0	**1**	0	**0.4**	0.0	**0.6**	0.0
0.6	1	**1**	0	**0**	0	**0.6**	0.0	**0.4**	0.0
1.0	1	1	0	0	0	1.0	0.0	0.0	0.0

3.2.2. Loss function

The final objective is to optimize both Mean Absolute Error (MAE) and *F*-measure (F_β). F_β is the weighted harmonic mean between precision and recall, therefore relying on a hard thresholding step for computation. MAE is instead evaluated directly on the raw prediction, without requiring any binarization. While MAE can be employed in its original form as a loss function, F_β is not suitable for backpropagation: according to the procedure defined in [29], in fact, the specific value for threshold is chosen by performing binarization at different levels, and selecting the one that eventually produces the best performance. We, therefore, propose learning through a continuous version of True Positives (*TP*), False Positives (*FP*), and False Negatives (*FN*), in order to avoid both the need for a hard threshold, as well as the process of its selection. F_β can be adapted as a loss function, here called F_β^{CC}, as follows:

1. We consider the continuous variants of *TP, FP* and *FN*:

$$TP'_{i,c} = PR_{i,c} \cdot GT_{i,c} \tag{5}$$

$$FP'_{i,c} = PR_{i,c} \cdot (1 - GT_{i,c}) \tag{6}$$

$$FN'_{i,c} = (1 - PR_{i,c}) \cdot GT_{i,c} \tag{7}$$

   Where PR is the saliency prediction on each pixel, and GT the corresponding ground truth. The effect of this formulation can be observed in Table 2.
2. We sum the above measures over all coordinates c in each image i, and compute the continuous variants of precision and recall as:

$$Precision'_i = \frac{\sum_{c \in C} TP'_{i,c}}{\sum_{c \in C} TP'_{i,c} + \sum_{c \in C} FP'_{i,c}} = \frac{\sum_{c \in C} PR_{i,c} \cdot GT_{i,c}}{\sum_{c \in C} PR_{i,c}} \tag{8}$$

$$Recall'_i = \frac{\sum_{c \in C} TP'_{i,c}}{\sum_{c \in C} TP'_{i,c} + \sum_{c \in C} FN'_{i,c}} = \frac{\sum_{c \in C} PR_{i,c} \cdot GT_{i,c}}{\sum_{c \in C} GT_{i,c}} \tag{9}$$

3. We compute a continuous variant of F_β from these values using fixed $\beta = \sqrt{0.3}$ as suggested in [29], and we complement it in order to effectively obtain a measure related to error:

$$F_\beta^{CC} = 1 - \frac{\left(1 + \beta^2\right) \frac{1}{|I|} \sum_{i \in I} Precision'_i \cdot \frac{1}{|I|} \sum_{i \in I} Recall'_i}{\beta^2 \frac{1}{|I|} \sum_{i \in I} Precision'_i + \frac{1}{|I|} \sum_{i \in I} Recall'_i} \tag{10}$$

MAE and F_β^{CC} (complemented continuous F_β) are then combined into a unique loss by means of a weighted average. Since we don't want, in principle, to have one measure overweighting the other, the weight is determined by bringing the two losses to the same value at the beginning of the backpropagation optimization process, i.e. in the sub-optimal configuration proposed by the Genetic Programming step.

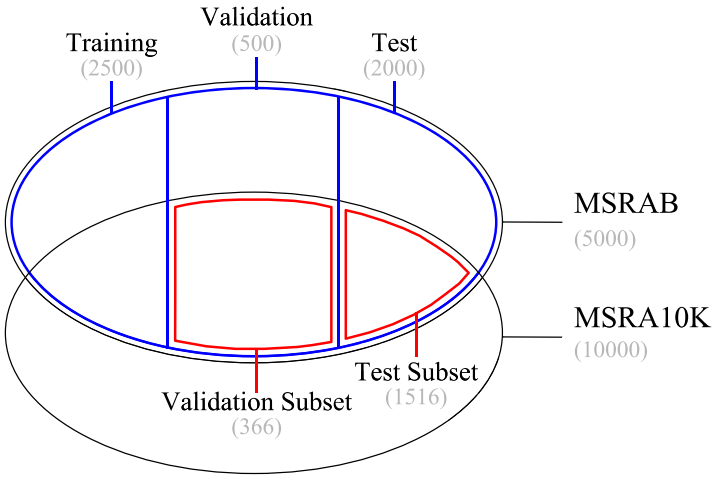

Fig. 2. Partitions and cardinalities of the MSRAB dataset, and intersection with the MSRAB10K. The Validation Subset has been used to train our fusion optimization, which has been then tested on the Test Subset.

4. Experiments

In this section, we first describe the experimental setup, by introducing the input saliency estimation algorithms, the datasets that have been adopted at different phases of the optimization, and the evaluation metrics. We then present the following experiments: we select different fusion trees from the Genetic Programming phase, generate the corresponding CNNs, and evaluate them on various datasets for a comparison with the input algorithms.

4.1. Setup

The proposed method is a general purpose algorithm that can operate with an arbitrary set of input data, in this case focused on optimizing a fusion tree for saliency estimation. As the final quality of saliency fusion is highly dependent on the quality of the chosen input saliency estimation methods, particular care has been given to conduct an appropriate selection of such methods. We wanted to investigate if it is possible to further improve the results of saliency estimation algorithms through their fusion. We focused our attention on two families of algorithms as described in Section 2: those based on hand-crafted features, and those based on deep learning techniques. Specifically, for the hand-crafted family we selected the ten best performing methods from the benchmark conducted by Borji et al. [33]. For deep learning we adopted all the methods described in the state of the art comparison from Hou et al. [29], with the addition of more recent architectures (DRCN [54] and MFCN [30]).

Most of the data-driven methods behind the exploited input saliency maps were originally trained on the training set of the MSRAB dataset [27] or its variants, as reported in [29]. The process of optimizing the best fusion trees, instead, has been here performed on a subset of the validation set of MSRAB. The prediction quality of the input methods, in fact, is expected to be slightly lower on “new” images, so we want to emulate, during the optimization process, the actual input distribution that will be obtained at inference time. Precomputed saliency maps for the input algorithms are available, in some cases, only on the MSRA10K dataset [44], which shares 3756 input images with MSRAB. We therefore defined the MSRAB Validation Subset (366 images) and MSRAB Test Subset (1516 images), based on the intersection between the original splits of MSRAB and the entire MSRA10K dataset, in order to define a common dataset among all input algorithms. Fig. 2 shows the definition of the MSRAB Subsets. The Validation Subset has been used for optimization, and the Test Subset for performance comparison against the input algorithms.

Table 3 reports the availability of already computed saliency maps for different algorithms on different datasets. Special attention has been given to deep learning methods, which are further tested on datasets different than the MSRAB, namely: DUTOMRON [39] (5166 images), ECSSD [75] (1000 images), HKU-IS [49] (1447 images), PASCALS [24] (850 images) and SOD [76] (300 images). Whenever precomputed saliency maps for a given deep learning method are not available for any one of the involved datasets, we resort to computing them all with official code implementations. We do not want, in fact, to risk exposing the model to different behaviors across experiments.

All solutions have been evaluated according to MAE and F_β as defined in Section 3, using the evaluation code provided with [29].

4.2. Hyperparameters search

We have conducted a greedy search to determine, in order, the optimal values for the hyperparameters that define the backbone and extended CNNs introduced in Section 3.2.1:

- Input kernel size (N)
- Input channel size (M)
- Compacter module
- Internal kernel size (N')

For an input node, the best values for M and N are found to be, respectively, 3 and 1, suggesting that keeping a pixel-precise processing is preferable at such an early stage. The best compacter is experimentally determined to be a channel-wise average, in a comparison against the minimum and maximum operations. For an internal node, the best value for N' is experimentally found to be 5. The entire set of values was determined through experiments conducted on the hand-crafted fusion tree HC-f on the MSRAB Validation Subset.

4.3. Optimization

We run two different experiments: in the first one we consider as input for Genetic Programming only saliency estimation algorithms based on hand-crafted features; in the second one we consider input maps coming from algorithms based on deep learning. The aim of the first experiment is to assess how much our proposal can improve over individual methods, and how it compares with deep learning solutions. The second experiment aims to assess if the proposed method is able to improve also the results of state of the art algorithms.

For the first experiment, we analyze the fittest individual at the end of the GP optimization. The fusion tree found, named HC-f, is reported in Fig. 3. An analysis of the tree shows that the optimization process selected only five out of the ten input algorithms available. The results obtained by the HC-f fusion tree, HC-f backbone CNN and HC-f extended CNN on the MSRAB Validation Subset are reported in Table 4. From

Table 3
Input saliency algorithms on the various datasets that have been used at different phases of optimization and evaluation. The symbols denote, respectively, ○: unavailability of official saliency maps, ◐: only partial availability (as prediction was performed on the MSRA10K dataset), ●: full availability. The last column indicates whether saliency maps have been recomputed (on all datasets) on grounds of the observed availability.

Datasets	M [27]		Computed	Datasets	M [27]		D	E	H	P	S	Computed
Methods (HC)	Val	Test	Locally*	Methods (DL)	Val	Test	[39]	[75]	[49]	[24]	[76]	locally*
DRFI [47]	◐	◐		DCL [28]	○	●	●	●	●	○	○	✓
DSR [38]	◐	◐		DHS [51]	○	○	○	●	○	●	○	✓
EQC [41]	○	○	✓	DS [50]	○	●	●	●	○	●	●	✓
GMR [39]	◐	◐		DRCN [54]	○	○	○	○	○	○	○	✓
HDCT [45]	○	○	✓	DSS [29]	○	●	●	●	●	●	●	✓
MB+ [43]	○	○	✓	ELD [52]	○	○	●	●	○	●	○	✓
MC [40]	◐	◐		MDF [49]	○	●	●	●	●	●	○	✓
RBD [42]	○	○	✓	MFCN [30]	●	●	●	●	●	●	●	
RC [44]	◐	◐		RFCN [53]	○	○	○	○	○	○	○	✓
ST [46]	○	○	✓	SC [48]	○	○	○	○	○	○	○	✓

* Recomputed locally using official code implementations

Table 4
Results for fusion trees of hand-crafted saliency algorithms at different levels of optimization on the MSRAB Validation Subset. Lower MAE is better, higher F_β is better.

	Optimization	MAE	F_β
HC-f	Fusion tree	0.1143	0.9040
	Backbone CNN	0.0494	0.9039
	Extended CNN	0.0471	0.9074

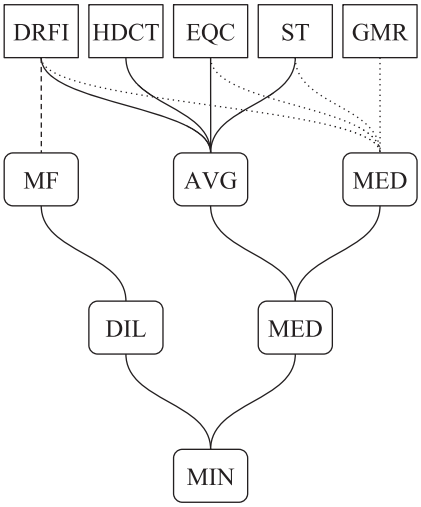

Fig. 3. Fusion tree based on hand-crafted input saliency algorithms and generated through Genetic Programming.

these results it is possible to see how the backbone CNN version of HC-f is able to reduce the MAE by almost 57% with a negligible reduction of F_β. The extended CNN further reduces the MAE of the backbone CNN by almost 5% at the same time increasing the F_β.

Table 5 reports the results of the extended CNN on the MSRAB Test Subset, in a comparison with the input saliency estimation algorithms. HC-f outperforms all other hand-crafted methods, reducing MAE on the second best (MB+) by 50% and improving F_β of DRFI by 4%.

For the second experiment, we analyze three individuals within the population at the last iteration of the GP optimization: the fittest individual, named DL-f; the individual with the lowest MAE, named DL-MAE; the individual with the maximum F_β, named DL-F_β. The corresponding fusion trees are depicted in Fig. 4. The analysis of the corresponding

Table 5
Comparison of the optimized fusion trees with input hand-crafted saliency algorithms on the MSRAB Test Subset. Lower MAE is better, higher F_β is better. Column-specific ranking is indicated as a pedix. Best results (ranking 1) are highlighted in boldface.

	MAE	F_β
DRFI [47]	0.1201_{5}	0.8655_{2}
DSR [38]	0.1227_{6}	0.8237_{11}
EQC [41]	0.1150_{4}	0.8612_{3}
GMR [39]	0.1251_{8}	0.8375_{7}
HDCT [45]	0.1477_{11}	0.8340_{10}
MB+ [43]	0.1095_{2}	0.8390_{6}
MC [40]	0.1435_{10}	0.8404_{5}
RBD [42]	0.1120_{3}	0.8367_{8}
RC [44]	0.1378_{9}	0.8354_{9}
ST [46]	0.1251_{7}	0.8583_{4}
HC-f (Extended CNN)	$\mathbf{0.0544}_{1}$	$\mathbf{0.8971}_{1}$

trees reveals that all three solutions selected only six out of the ten input algorithms available.

Focusing on DL-MAE, which has the largest variety of operators used, we report in Fig. 5 the initial fusion tree, its conversion into backbone CNN, and extended CNN. It is possible to observe how the genetic programming phase autonomously created a pseudo-morphological opening, by selecting erosion and dilation in this specific order, although separated by further processing. This allows the tree to exclude spurious elements coming from any of the input saliency maps, at the expense of a slightly less precise final estimation. Fig. 6 shows the step-by-step visualization of the three versions of DL-MAE on one example input. The backpropagation-based optimization exploits the minimum operation to severe the right-most branch of the tree. This is obtained by producing maps with overall high values (it can be easily seen in the extended CNN, and less obviously in the backbone CNN), which are then completely ignored by the minimum operation itself.

The results obtained by the fusion tree, backbone CNN and extended CNN of the DL-f, DL-MAE and DL-F_β on the MSRAB Validation Subset are reported in Table 6. From these results it can be noticed that the backbone CNNs produce further improvements of MAE with respect to the corresponding fusion trees, the most relevant being DL-F_β, which sees a drop in error by almost 72%. F_β presents a small average improvement across all backbone CNNs. On the contrary, the extended CNNs tend to produce overall slightly lower values for F_β. We, however, consider this an acceptable compromise to reach a further level of reduction for MAE.

In a comparison with the input saliency estimation algorithms on the MSRAB Test Subset (Table 7), the three deep learning fusion trees once again present similar performance with respect to each other,

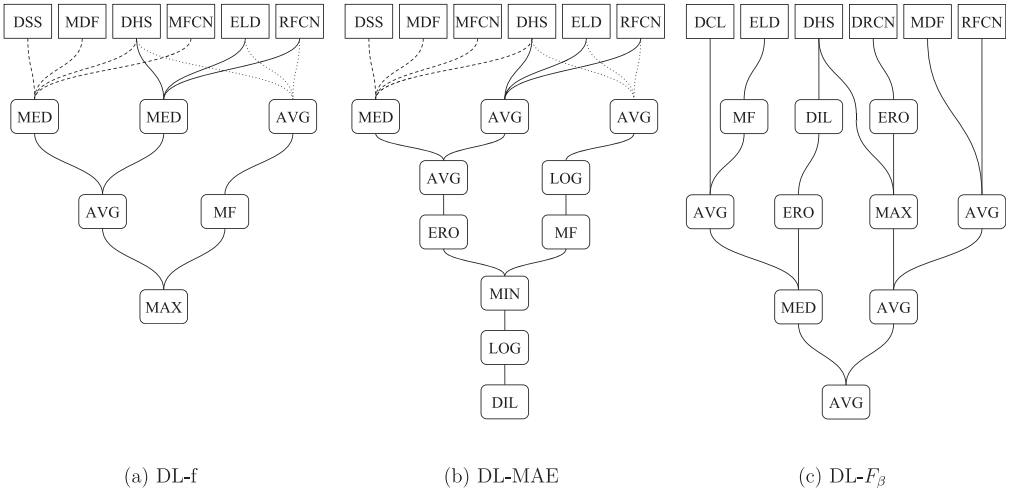

Fig. 4. Fusion trees based on deep learning saliency algorithms and generated through Genetic programming.

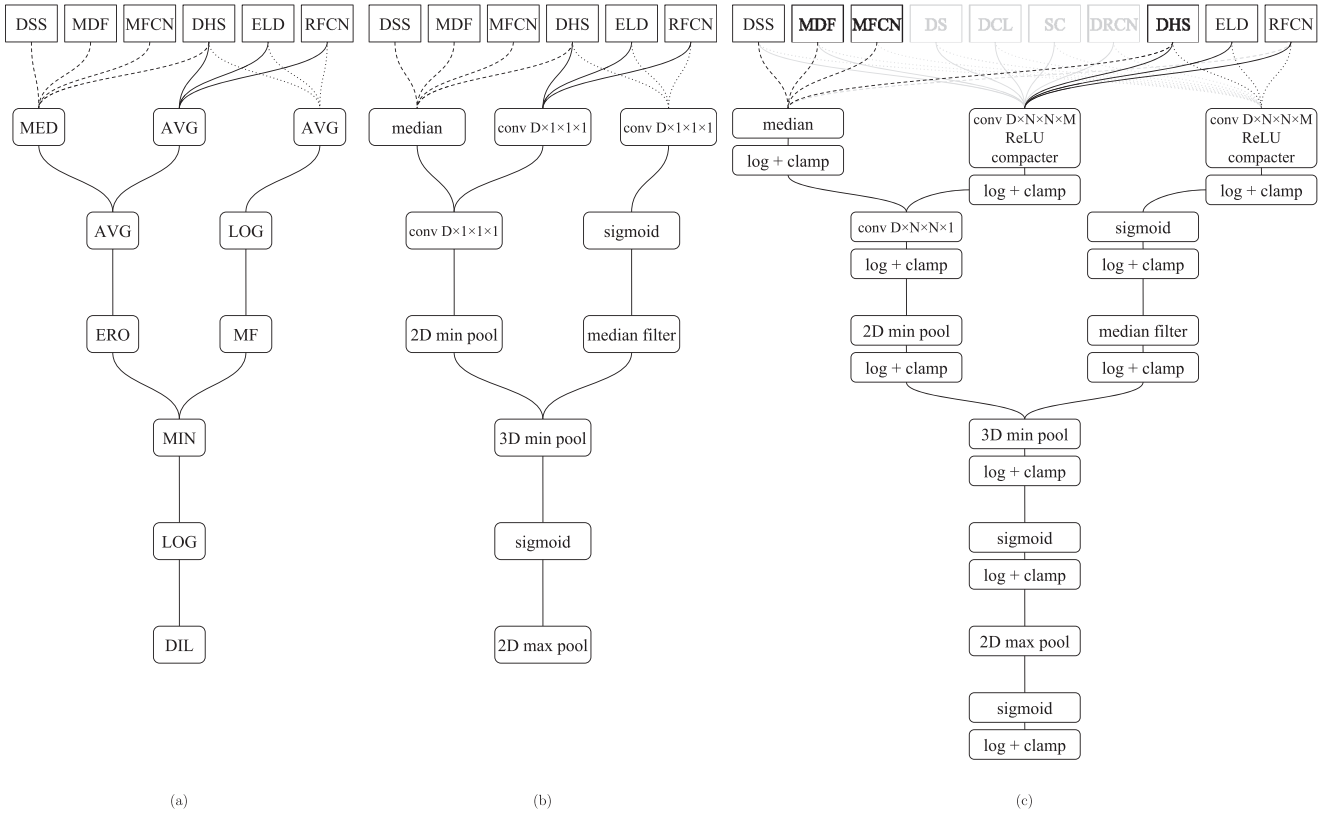

Fig. 5. Different implementation stages of the automatically designed DL-MAE: (a) fusion tree, (b) backbone CNN, and (c) extended CNN. Convolutional layers are described with D × N × N × M, where D is the filter depth (number of input channels), N denotes the filter spatial size, and M is the cardinality of filter bank (number of output channels).

and produce the overall best results in a comparison with the input algorithms. DL-MAE in particular outperforms the second best deep learning method (DHS) in reducing MAE by 23%, and improving F_β by less than 1%. This is a further manifestation of the difficulties encountered in significantly improving the F_β measure.

In a cross-domain evaluation, the extended CNN based on HC-f results in better performance than 7 out of 10 deep learning methods, according to MAE, and better than three of them according to F_β. This suggests to what extent it is possible to reach deep-learning-level performance by proper fusion of hand-crafted solutions.

Fig. 7 shows the precision and recall curves for input deep learning saliency algorithms, compared with different fusions produced by our optimization. Due to the close-to-binary nature of our predictions, the points forming our precision–recall curves are concentrated in a small area near the optimal 1.0–1.0 corner of the plot.

4.4. Comparison with other information fusion techniques

As a further experiment, we compare our proposed approach against two reference fusion algorithms. The first algorithm is the Simultaneous Truth And Performance Level Estimation (STAPLE) [77]. It is based on an Expected Maximization strategy and it was devised for estimating the "ground truth" segmentation from a group of experts' segmentations in the context of medical imaging. STAPLE takes different segmentations

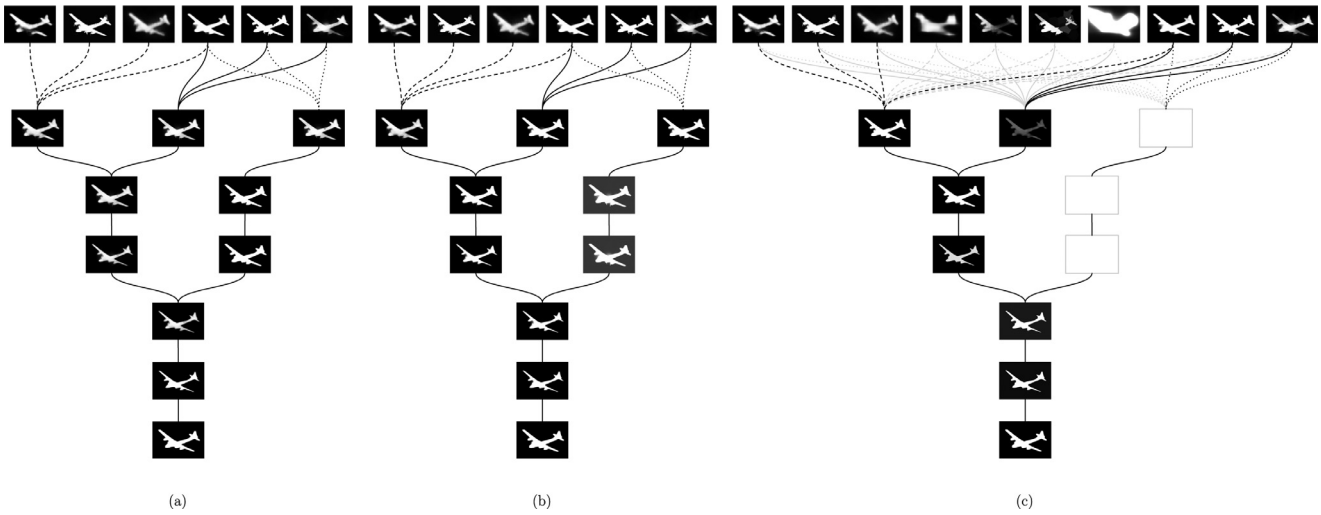

Fig. 6. Step-by-step visualization of the three levels of optimization of DL-MAE, namely: (a) fusion tree, (b) backbone CNN, and (c) extended CNN.

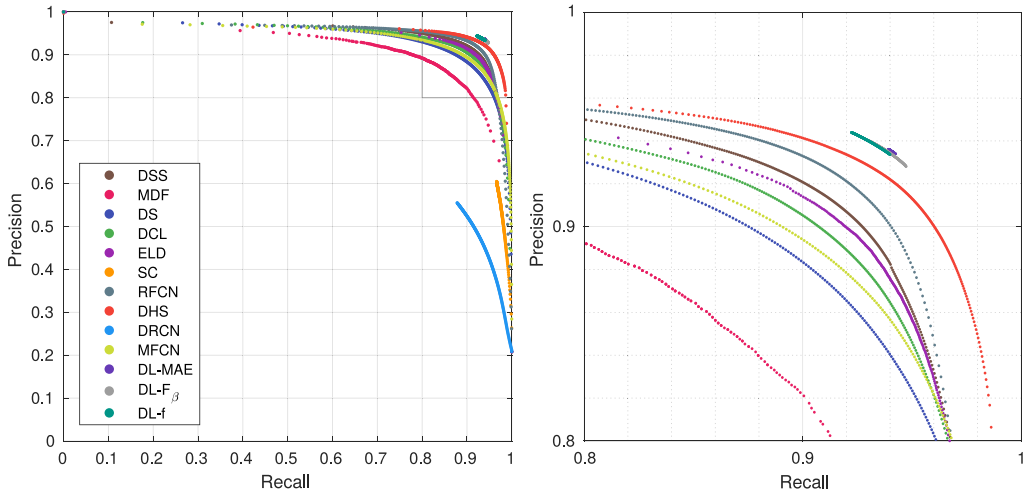

Fig. 7. Precision and recall curves on the MSRAB Test set. Our extended CNNs based on the fusion of input saliency algorithms (DL-MAE, DL-F_β, DL-f) are the closest to the optimal 1.0–1.0 corner of the plot.

Table 6
Results for fusion trees of deep learning saliency algorithms at different levels of optimization on the MSRAB Validation Subset. Lower MAE is better, higher F_β is better.

	Optimization	MAE	F_β
DL-f	Fusion tree	0.0412	0.9457
	Backbone CNN	0.0259	0.9454
	Extended CNN	0.0260	0.9439
DL-MAE	Fusion tree	0.0284	0.9447
	Backbone CNN	0.0258	0.9457
	Extended CNN	0.0253	0.9446
DL-F_β	Fusion tree	0.0903	0.9464
	Backbone CNN	0.0267	0.9467
	Extended CNN	0.0252	0.9448

Table 7
Comparison of the optimized fusion trees with input deep learning saliency algorithms on the MSRAB Test Subset. Lower MAE is better, higher F_β is better. Column-specific ranking is indicated as a pedix. Best results (ranking 1) are highlighted in boldface.

	MAE	F_β
DSS [29]	$0.0461_{\ 6}$	$0.9226_{\ 7}$
MFCN [30]	$0.0738_{\ 11}$	$0.9138_{\ 8}$
DCL [28]	$0.0590_{\ 8}$	$0.9133_{\ 9}$
DHS [51]	$0.0326_{\ 4}$	$0.9408_{\ 4}$
DS [50]	$0.0665_{\ 10}$	$0.9069_{\ 10}$
DRCN [54]	$0.2065_{\ 13}$	$0.7326_{\ 13}$
ELD [52]	$0.0400_{\ 5}$	$0.9229_{\ 6}$
MDF [49]	$0.0793_{\ 12}$	$0.8765_{\ 12}$
RFCN [53]	$0.0605_{\ 9}$	$0.9365_{\ 5}$
SC [48]	$0.0572_{\ 7}$	$0.8871_{\ 11}$
DL-MAE (Extended CNN)	$\mathbf{0.0250}_{\ 1}$	$\mathbf{0.9467}_{\ 1}$
DL-F_β (Extended CNN)	$0.0253_{\ 3}$	$0.9462_{\ 3}$
DL-f (Extended CNN)	$0.0253_{\ 2}$	$0.9467_{\ 2}$

and simultaneously estimates the final segmentation and the sensitivity and specificity parameters characterizing the performance of each expert. The second algorithm is the Probabilistic Rand Index Fusion (PRIF) [78]. The fusion strategy defined in PRIF is based on a Markovian Bayesian fusion procedure, and the fusion is guided by the Probabilistic Rand Index [79]. This index measures the agreement of one segmentation result to multiple ground truth segmentations, in a quantitative and perceptual way. We choose these two fusion strategies since they work taking into account the saliency maps only, and due to the

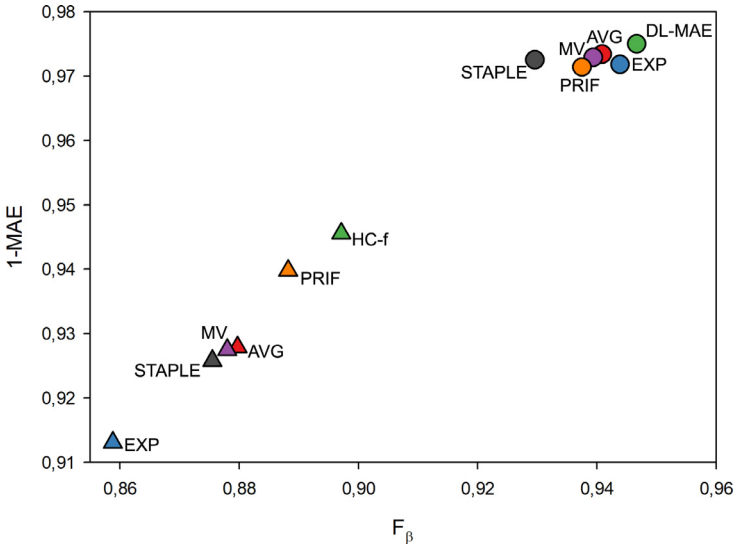

Fig. 8. Performance comparison with other fusion techniques on both deep learning and hand-crafted solutions, on the MSRAB Test Subset. The optimal solution is in the top-right corner of the plot.

Table 8
Comparison of deep learning fusion trees and input saliency algorithms on various datasets. Lower MAE is better, higher F_β is better. Column-specific ranking is indicated as subscript. Best results (ranking 1) are also highlighted in boldface.

	MSRAB test [27]		DUTOMRON [39]		ECSSD [75]		HKU-IS [49]		PASCALS [24]		SOD [76]	
	MAE	F_β	MAE	F_β	MAE	F_β	MAE	F_β	MAE	F_β	MAE	F_β
DSS [29]	0.0447 $_6$	0.9182 $_6$	0.0740 $_5$	0.7605 $_6$	0.0647 $_5$	0.9062 $_4$	0.0502 $_4$	0.9003 $_4$	0.1016 $_5$	0.8228 $_7$	0.1259 $_2$	0.8364 $_2$
MFCN [30]	0.0775 $_{12}$	0.9024 $_9$	0.1626 $_{12}$	0.7024 $_{10}$	0.1181 $_{12}$	0.8645 $_{10}$	0.1153 $_{12}$	0.8518 $_9$	0.1613 $_{12}$	0.7826 $_9$	0.1821 $_{12}$	0.7693 $_{10}$
DCL [28]	0.0578 $_7$	0.9094 $_8$	0.0944 $_9$	0.7334 $_9$	0.0800 $_6$	0.8958 $_8$	0.0634 $_6$	0.8934 $_6$	0.1146 $_7$	0.8071 $_8$	0.1308 $_6$	0.8331 $_5$
DHS [51]	0.0347 $_4$	0.9315 $_4$	**0.0271** $_1$	**0.9090** $_1$	0.0621 $_4$	0.9052 $_5$	0.0524 $_5$	0.8918 $_7$	0.0918 $_4$	0.8275 $_6$	0.1279 $_4$	0.8247 $_7$
DS [50]	0.0672 $_{10}$	0.8981 $_{10}$	0.0835 $_6$	0.7735 $_5$	0.0802 $_7$	0.9002 $_6$	0.0789 $_9$	0.8664 $_8$	0.1087 $_6$	0.8300 $_4$	0.1267 $_3$	0.8320 $_6$
DRCN [54]	0.3533 $_{13}$	0.6066 $_{13}$	0.4118 $_{13}$	0.4167 $_{13}$	0.3431 $_{13}$	0.6252 $_{13}$	0.3373 $_{13}$	0.5783 $_{13}$	0.3294 $_{13}$	0.5950 $_{13}$	0.3421 $_{13}$	0.6189 $_{13}$
ELD [52]	0.0429 $_5$	0.9139 $_7$	0.0923 $_7$	0.7381 $_8$	0.0816 $_8$	0.8652 $_9$	0.0722 $_7$	0.8430 $_{10}$	0.1216 $_9$	0.7739 $_{10}$	0.1545 $_8$	0.7645 $_{11}$
MDF [49]	0.0767 $_{11}$	0.8695 $_{12}$	0.0982 $_{11}$	0.6970 $_{11}$	0.1137 $_{11}$	0.8313 $_{11}$	0.0952 $_{10}$	0.8241 $_{11}$	0.1449 $_{10}$	0.7636 $_{11}$	0.1641 $_{10}$	0.7863 $_9$
RFCN [53]	0.0620 $_9$	0.9258 $_5$	0.0940 $_8$	0.7470 $_7$	0.0972 $_9$	0.8978 $_7$	0.0779 $_8$	0.8950 $_5$	0.1176 $_8$	0.8287 $_5$	0.1611 $_9$	0.8072 $_8$
SC [48]	0.0602 $_8$	0.8712 $_{11}$	0.0980 $_{10}$	0.6755 $_{12}$	0.1056 $_{10}$	0.8158 $_{12}$	0.0978 $_{11}$	0.7715 $_{12}$	0.1483 $_{11}$	0.7124 $_{12}$	0.1819 $_{11}$	0.7039 $_{12}$
DL-MAE	**0.0261** $_1$	0.9368 $_3$	0.0419 $_3$	0.8435 $_4$	0.0524 $_2$	0.9195 $_2$	0.0386 $_2$	0.9125 $_2$	0.0811 $_2$	0.8490 $_3$	0.1279 $_5$	0.8355 $_3$
DL-F_β	0.0265 $_2$	0.9379 $_2$	0.0411 $_2$	0.8503 $_2$	**0.0517** $_1$	**0.9209** $_1$	**0.0380** $_1$	**0.9151** $_1$	**0.0791** $_1$	**0.8549** $_1$	**0.1256** $_1$	**0.8415** $_1$
DL-f	0.0269 $_3$	**0.9389** $_1$	0.0425 $_4$	0.8441 $_3$	0.0545 $_3$	0.9187 $_3$	0.0410 $_3$	0.9107 $_3$	0.0832 $_3$	0.8511 $_2$	0.1315 $_7$	0.8332 $_4$

availability of the implementation codes of the algorithms. We further include some baseline fusion techniques based on cross-channel operations: MV (majority voting, implemented with the median operator), AVG (average operator), and EXP (average operator over maps that are distorted with an exponential operator). All baselines are followed by hard quantization at mid-gray level. As noted in [30], in fact, MAE is essentially a direct comparison between prediction and (binary) ground truth, so with a continuous-valued prediction there is always going to be some residual difference on "true positive" and "true negative" areas. Such differences, however small, would accumulate over the whole image and result in sub-optimal performance according to this particular evaluation measure.

Fig. 8 shows the performance obtained by these fusion algorithms on the MSRAB Test Subset, compared with our proposed solution for both the hand-crafted (HC) saliency algorithms and the deep learning (DL) saliency algorithms. This results in the formation of two distinct groups of points. For each group, the fusion proposed in this paper (represented respectively by HC-f and DL-MAE) dominates all other methods. PRIF appears to perform better than STAPLE in both groups, but while it also outperforms baseline solutions MV, AVG and EXP for hand-crafted methods, it presents inferior performance in the context of deep learning methods. Due to the nature of this approach, the reason is probably to be found in the different distribution of the input saliency maps from the two groups. It is worth noting that differently from our method, both PRIF and STAPLE require an examination of the saliency map of each image. That is, they change the fusion strategy based on the map contents, while our method applies the same learned fusion operations for all the inputs.

4.5. Extension to other datasets

With the intent of focusing on the most effective solutions, the deep learning input algorithms, and the resulting fusion CNNs, have been also evaluated on other standard datasets for saliency estimation, namely DUTOMRON [39], ECSSD [75], HKU-IS [49], PASCALS [24], and SOD [76]. Each of these datasets has been annotated with potentially different criteria from the other ones, as noted in [29], so they constitute an interesting benchmark to assess the generalization capabilities of the proposed fusion strategy. Results are reported in Table 8, along with results on the full Test set from MSRAB.

Although the three optimized CNNs exhibit comparable performance, there is a subtle yet consistent advantage of DL-F_β over DL-MAE, differently from what was observed in the previous experiments.

DL-F_β is the fusion tree that best optimizes F_β, by proper selection and combination of input algorithms. From this starting point, it is relatively easy to further optimize for MAE via backpropagation, for example by operating on logistic curves to regulate the confidence of estimated saliency, and thus reducing the gap with the binary ground truth. On the other hand, if the initial fusion tree is sub-optimal with respect to F_β, due for example to low recall, it becomes hard or impossible to further increase performance on such metric. This might require, in fact, involving areas that were completely excluded by the input

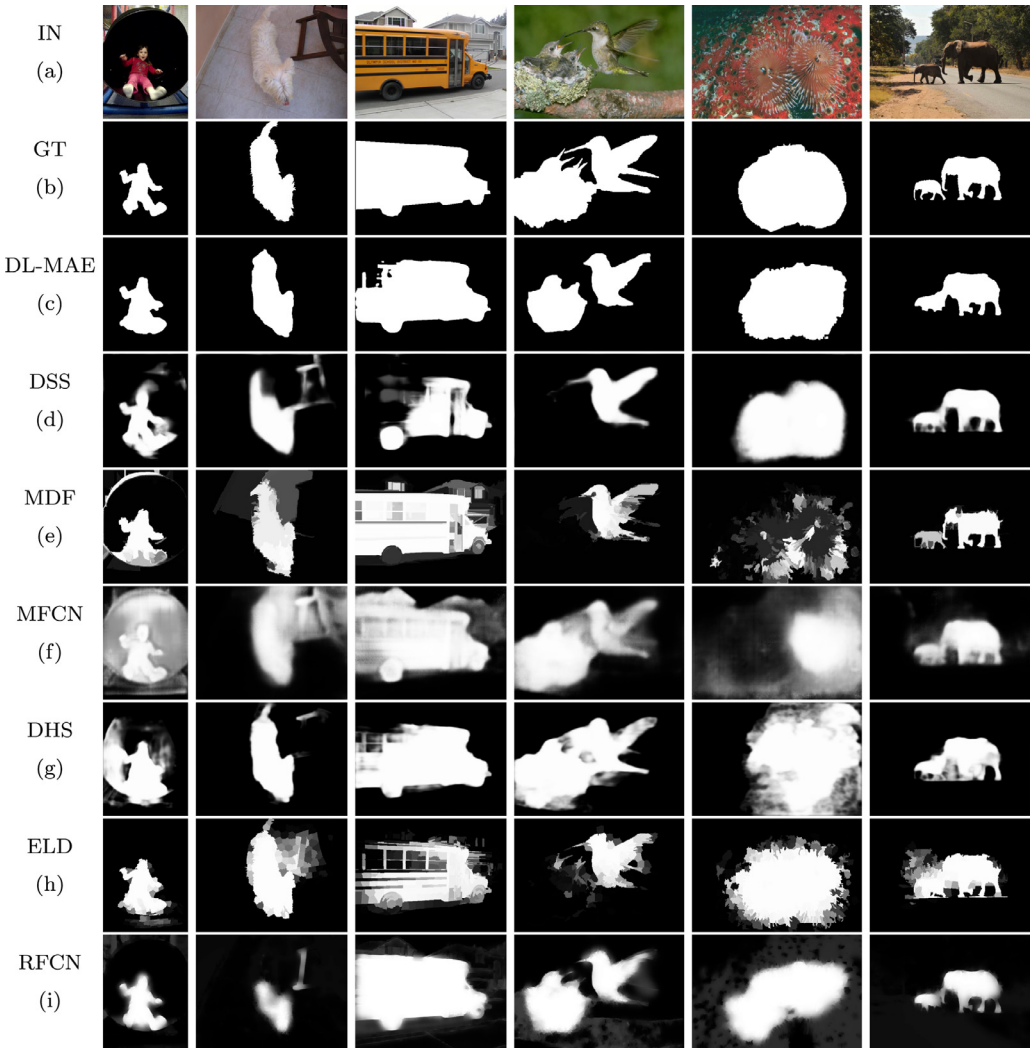

Fig. 9. Visual results of the extended CNN from DL-MAE. Rows (a) and (b) show, respectively, the starting image and the corresponding ground truth annotation. (c) is the output of DL-MAE. Rows (d)–(i) are the input saliency maps used by DL-MAE, namely: (d) DSS, (e) MDF, (f) MFCN, (g) DHS, (h) ELD, (i) RFCN.

saliency algorithms, in order to increase the recall level. DL-MAE has therefore slightly overfitted on the annotation choices that are specific to the MSRAB dataset, while DL-F_β produced a better generalization. Overall, the CNNs obtained through the optimization process appear to improve on the input algorithms on all datasets, with the exception of DUTOMRON, where DHS stands out from all other reported performance values. However, by further inspection, according to [51] and [29] DHS was trained on both the MSRA10K dataset and the DUTOMRON dataset itself, thus explaining the observed outlying performance.

Fig. 9 shows the effect of DL-MAE on different sample images, along with the exploited input saliency maps coming from the corresponding saliency estimation algorithms. It can be observed that the saliency estimations obtained with our fusion technique are extremely sharp, as a result of including MAE in the optimization function. As previously noted, in fact, this measure would be negatively affected by residual differences between a continuous prediction and a binary ground truth.

5. Conclusions

We have proposed a general purpose neural architecture search strategy, with a focus on the estimation of image saliency. Specifically, we have devised a three-step optimization process that combines the output of existing algorithms for saliency estimation.

First, a fusion tree is generated through genetic programming, working on a set of predefined operators. The discrete search space of the operators to be used and combined is efficiently handled by the evolutionary algorithm. This initial solution is then converted into a neural network following two different generalizations (backbone CNN and extended CNN). Backpropagation is eventually used to efficiently fine-tune the continuous parameters that characterize the operators chosen in the first step.

We have evaluated our solution by training on a fixed dataset and testing on multiple datasets adopted by the state of the art. The resulting model improves upon all the input saliency estimation algorithms, and outperforms other compared fusion techniques. By manual inspection of some fusion trees generated with our method, we have observed the emergence of interesting behaviors, such as the implementation of morphological opening by the genetic programming step, and the inhibition of entire branches of the tree by the backpropagation steps.

Experiments have been carried out starting from two independent sets of saliency estimation methods: hand-crafted and deep learning solutions. As a direction for future work we will consider jointly fusing both kinds of methods, as well as breaking-up the input saliency estimation algorithms into logical blocks. By applying our architecture search over such blocks, we aim at reducing any possible redundancies.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research. The research leading to these results has received funding from TEINVEIN: TEcnologie INnovative per i VEicoli Intelligenti, CUP (Codice Unico Progetto – Unique Project Code): E96D17000110009 – Call "Accordi per la Ricerca e l'Innovazione", cofunded by POR FESR 2014–2020 (Programma Operativo Regionale, Fondo Europeo di Sviluppo Regionale – Regional Operational Programme, European Regional Development Fund).

References

[1] L. Itti, Visual saliency, Scholarpedia 2 (9) (2007) 2237.
[2] R.M. Shiffrin, W. Schneider, Controlled and automatic human information processing: ii. perceptual learning, automatic attending and a general theory., Psychol. Rev. 84 (2) (1977) 127.
[3] W. Schneider, R.M. Shiffrin, Controlled and automatic human information processing: I. Detection, search, and attention., Psychol. Rev. 84 (1) (1977) 1.
[4] L. Itti, C. Koch, Computational modelling of visual attention, Nat. Rev. Neurosci. 2 (3) (2001) 194.
[5] D. Gao, S. Han, N. Vasconcelos, Discriminant saliency, the detection of suspicious coincidences, and applications to visual recognition, IEEE Trans. Pattern Anal. Mach. Intell. 31 (6) (2009) 989–1005.
[6] Z. Ren, S. Gao, L.-T. Chia, I.W.-H. Tsang, Region-based saliency detection and its application in object recognition., IEEE Trans. Circuits Syst. Video Technol. 24 (5) (2014) 769–779.
[7] S. Mitri, S. Frintrop, K. Pervölz, H. Surmann, A. Nüchter, Robust object detection at regions of interest with an application in ball recognition, in: Proceedings of the IEEE International Conference on Robotics and Automation, 1, IEEE; 1999, 2005, p. 125.
[8] V. Navalpakkam, L. Itti, An integrated model of top-down and bottom-up attention for optimizing detection speed, in: Proceedings of the 2006 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2006, pp. 2049–2056.
[9] Q. Li, Y. Zhou, J. Yang, Saliency based image segmentation, in: Proceedings of the 2011 International Conference on Multimedia Technology, 2011, pp. 5068–5071.
[10] V. Mahadevan, N. Vasconcelos, Saliency-based discriminant tracking, in: Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 1007–1013.
[11] S. Avidan, A. Shamir, Seam carving for content-aware image resizing, ACM Trans. Graph. 26 (3) (2007).
[12] R. Margolin, L. Zelnik-Manor, A. Tal, Saliency for image manipulation, Vis. Comput. 29 (5) (2013) 381–392.
[13] D. DeCarlo, A. Santella, Stylization and abstraction of photographs, ACM Trans. Graph. 21 (3) (2002) 769–776.
[14] N. Ouerhani, J. Bracamonte, H. Hügli, M. Ansorge, F. Pellandini, Adaptive color image compression based on visual attention, in: Proceedings of the 11th International Conference on Image Analysis and Processing (ICIAP), 11, Institute of Electrical and Electronics Engineers (IEEE), 2001, pp. 416–421.
[15] S. Corchs, G. Ciocca, R. Schettini, Video summarization using a neurodynamical model of visual attention, in: Proceedings of the 6th IEEE Workshop on Multimedia Signal Processing, IEEE, 2004, pp. 71–74.
[16] C. Guo, L. Zhang, A novel multiresolution spatiotemporal saliency detection model and its applications in image and video compression, IEEE Trans. Image Process. 19 (1) (2010) 185–198.
[17] F. Gasparini, S. Corchs, R. Schettini, Low quality image enhancement using visual attention, Opt. Eng. 46 (2007) 040502(1–3).
[18] Y. Gao, M. Shi, D. Tao, C. Xu, Database saliency for fast image retrieval, IEEE Trans. Multimed. 17 (3) (2015) 359–369.
[19] A. Li, X. She, Q. Sun, Color image quality assessment combining saliency and FSIM, in: Proceedings of the Fifth International Conference on Digital Image Processing (ICDIP 2013), 8878, International Society for Optics and Photonics, 2013, p. 88780I.
[20] L. Wong, K. Low, Saliency retargeting: an approach to enhance image aesthetics, in: Proceedings of the 2011 IEEE Workshop on Applications of Computer Vision (WACV), 2011, pp. 73–80.
[21] L. Itti, C. Koch, E. Niebur, A model of saliency-based visual attention for rapid scene analysis, IEEE Trans. Pattern Anal. Mach. Intell. 20 (11) (1998) 1254–1259.
[22] Y. Wei, F. Wen, W. Zhu, J. Sun, Geodesic saliency using background priors, in: Proceedings of the European Conference On Computer vision, Springer, 2012, pp. 29–42.
[23] F. Perazzi, P. Krähenbühl, Y. Pritch, A. Hornung, Saliency filters: contrast based filtering for salient region detection, in: Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2012, pp. 733–740.
[24] Y. Li, X. Hou, C. Koch, J.M. Rehg, A.L. Yuille, The secrets of salient object segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 280–287.
[25] H. Jiang, J. Wang, Z. Yuan, Y. Wu, N. Zheng, S. Li, Salient object detection: a discriminative regional feature integration approach, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 2083–2090.
[26] H. Cholakkal, D. Rajan, J. Johnson, Top-down saliency with locality-constrained contextual sparse coding., in: Proceedings of the 2015 BMVC, 1, 2015, p. 5.
[27] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, H.-Y. Shum, Learning to detect a salient object, IEEE Trans. Pattern Anal. Mach. Intell. 33 (2) (2011) 353–367.
[28] G. Li, Y. Yu, Deep contrast learning for salient object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 478–487.
[29] Q. Hou, M. Cheng, X. Hu, A. Borji, Z. Tu, P.H.S. Torr, Deeply supervised salient object detection with short connections, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3203–3212.
[30] S. Bianco, M. Buzzelli, R. Schettini, Multiscale fully convolutional network for image saliency, J. Electron. Imaging 27 (5) (2018) 051221.
[31] L. Mai, Y. Niu, F. Liu, Saliency aggregation: a data-driven approach, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 1131–1138.
[32] M. Amirul Islam, M. Kalash, N.D. Bruce, Revisiting salient object detection: simultaneous detection, ranking, and subitizing of multiple salient objects, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7142–7150.
[33] A. Borji, M. Cheng, H. Jiang, J. Li, Salient object detection: a benchmark, IEEE Trans. Image Process. 24 (12) (2015) 5706–5722.
[34] S. Bianco, G. Ciocca, R. Schettini, How far can you get by combining change detection algorithms? in: Proceedings of the International Conference on Image Analysis and Processing – ICIAP 2017, in: Lecture Notes in Computer Science, 10484, Springer, 2017, pp. 96–107.
[35] S. Bianco, G. Ciocca, R. Schettini, Combination of video change detection algorithms by genetic programming, IEEE Trans. Evol. Comput. 21 (6) (2017) 914–928.
[36] M. Buzzelli, S. Bianco, G. Ciocca, Combining saliency estimation methods, in: Proceedings of the International Conference on Image Analysis and Processing – ICIAP 2019, Springer International Publishing, Cham, 2019, pp. 326–336.
[37] D. Mazzini, M. Buzzelli, D.P. Pau, R. Schettini, A CNN architecture for efficient semantic segmentation of street scenes, in: Proceedings of the 8th IEEE International Conference on Consumer Electronics – Berlin (ICCE-Berlin), IEEE, 2018, pp. 1–6.
[38] X. Li, H. Lu, L. Zhang, X. Ruan, M. Yang, Saliency detection via dense and sparse reconstruction, in: Proceedings of the 2013 IEEE International Conference on Computer Vision, 2013, pp. 2976–2983.
[39] C. Yang, L. Zhang, H. Lu, X. Ruan, M. Yang, Saliency detection via graph-based manifold ranking, in: Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 3166–3173.
[40] B. Jiang, L. Zhang, H. Lu, C. Yang, M.-H. Yang, Saliency detection via absorbing Markov chain, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 1665–1672.
[41] Ç. Aytekin, E.C. Ozan, S. Kiranyaz, M. Gabbouj, Visual saliency by extended quantum cuts, in: Proceedings of the 2015 IEEE International Conference on Image Processing (ICIP), 2015, pp. 1692–1696.
[42] W. Zhu, S. Liang, Y. Wei, J. Sun, Saliency optimization from robust background detection, in: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2814–2821.
[43] J. Zhang, S. Sclaroff, Z. Lin, X. Shen, B. Price, R. Mech, Minimum barrier salient object detection at 80 fps, in: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1404–1412.
[44] M. Cheng, N.J. Mitra, X. Huang, P.H.S. Torr, S. Hu, Global contrast based salient region detection, IEEE Trans. Pattern Anal. Mach. Intell. 37 (3) (2015) 569–582.
[45] J. Kim, D. Han, Y. Tai, J. Kim, Salient region detection via high-dimensional color transform and local spatial support, IEEE Trans. Image Process. 25 (1) (2016) 9–23.
[46] Z. Liu, W. Zou, O.L. Meur, Saliency tree: a novel saliency detection framework, IEEE Trans. Image Process. 23 (5) (2014) 1937–1952.
[47] J. Wang, H. Jiang, Z. Yuan, M.-M. Cheng, X. Hu, N. Zheng, Salient object detection: a discriminative regional feature integration approach, Int. J. Comput. Vis. 123 (2) (2017) 251–268.
[48] R. Zhao, W. Ouyang, H. Li, X. Wang, Saliency detection by multi-context deep learning, in: Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1265–1274.
[49] G. Li, Y. Yu, Visual saliency based on multiscale deep features, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5455–5463.
[50] X. Li, L. Zhao, L. Wei, M. Yang, F. Wu, Y. Zhuang, H. Ling, J. Wang, Deepsaliency: multi-task deep neural network model for salient object detection, IEEE Trans. Image Process. 25 (8) (2016) 3919–3930.
[51] N. Liu, J. Han, DHSNet: deep hierarchical saliency network for salient object detection, in: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 678–686.
[52] G. Lee, Y.-W. Tai, J. Kim, Deep saliency with encoded low level distance map and high level features, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 660–668.
[53] L. Wang, L. Wang, H. Lu, P. Zhang, X. Ruan, Salient object detection with recurrent fully convolutional networks, IEEE transactions on pattern analysis and machine intelligence 41 (7) (2018) 1734–1746.
[54] N. Liu, J. Han, A deep spatial contextual long-term recurrent convolutional network for saliency detection, IEEE Trans. Image Process. 27 (7) (2018) 3264–3274.

[55] T. Wang, L. Zhang, S. Wang, H. Lu, G. Yang, X. Ruan, A. Borji, Detect globally, refine locally: a novel approach to saliency detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3127–3135.
[56] W. Wang, J. Shen, X. Dong, A. Borji, Salient object detection driven by fixation prediction, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1711–1720.
[57] Y. Zeng, M. Feng, H. Lu, G. Yang, A. Borji, An unsupervised game-theoretic approach to saliency detection, IEEE Trans. Image Process. 27 (9) (2018) 4545–4554.
[58] Y. Zeng, H. Lu, L. Zhang, M. Feng, A. Borji, Learning to promote saliency detectors, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1644–1653.
[59] A. Azaza, J. van de Weijer, A. Douik, M. Masana, Context proposals for saliency detection, Comput. Vis. Image Underst. 174 (2018) 1–11, doi:10.1016/j.cviu.2018.06.002.
[60] J. Xu, S. Yue, Fusion of saliency maps for visual attention selection in dynamic scenes, Int. J. Adv. Res. Artif. Intell. 2 (4) (2013) 48–58, doi:10.14569/IJARAI.2013.020408.
[61] A. Banitalebi-Dehkordi, Y. Dong, M.T. Pourazad, P. Nasiopoulos, A learning-based visual saliency fusion model for high dynamic range video (LBVS-HDR), in: Proceedings of the 23rd European Conference on Signal Processing Conference (EUSIPCO)„ IEEE, 2015, pp. 1541–1545.
[62] T.V. Nguyen, M. Kankanhalli, As-similar-as-possible saliency fusion, Multimed. Tools Appl. 76 (8) (2017) 10501–10519.
[63] Q. Jiang, Z. Wu, C. Tian, T. Liu, MSR: a simple and effective metric for visual saliency map fusion, in: Proceedings of the 8th International Symposium on Computational Intelligence and Design (ISCID), 2, IEEE, 2015, pp. 432–435.
[64] X. Wei, Z. Tao, C. Zhang, X. Cao, Structured saliency fusion based on Dempster–Shafer theory, IEEE Signal Process. Lett. 22 (9) (2015) 1345–1349.
[65] K.O. Stanley, R. Miikkulainen, Evolving neural networks through augmenting topologies, Evol. Comput. 10 (2) (2002) 99–127.
[66] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, et al., Evolving deep neural networks, in: Artificial Intelligence in the Age of Neural Networks and Brain Computing, Elsevier, 2019, pp. 293–312.
[67] M. Suganuma, S. Shirakawa, T. Nagao, A genetic programming approach to designing convolutional neural network architectures, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '17, ACM, 2017, pp. 497–504.
[68] B. Baker, O. Gupta, N. Naik, R. Raskar, Designing Neural Network Architectures Using Reinforcement Learning, arXiv:1611.02167 (2016).
[69] B. Zoph, Q.V. Le, Neural Architecture Search with Reinforcement Learning, arXiv:1611.01578 (2016).
[70] B. Zoph, V. Vasudevan, J. Shlens, Q.V. Le, Learning transferable architectures for scalable image recognition, in: Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 8697–8710.
[71] E. Real, A. Aggarwal, Y. Huang, Q.V. Le, Regularized Evolution for Image Classifier Architecture Search, arXiv:1802.01548 (2018).
[72] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, K. Murphy, Progressive neural architecture search, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 19–34.
[73] T. Hinz, N. Navarro-Guerrero, S. Magg, S. Wermter, Speeding up the hyperparameter optimization of deep convolutional neural networks, Int. J. Comput. Intell. Appl. (2018). 1850008
[74] J.R. Koza, Genetic programming as a means for programming computers by natural selection, Stat. Comput. 4 (2) (1994) 87–112.
[75] Q. Yan, L. Xu, J. Shi, J. Jia, Hierarchical saliency detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 1155–1162.
[76] D. Martin, C. Fowlkes, D. Tal, J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, in: Proceedings of the Eighth IEEE International Conference on Computer Vision, ICCV, 2, IEEE, 2001, pp. 416–423.
[77] S.K. Warfield, K.H. Zou, W.M. Wells, Simultaneous truth and performance level estimation (STAPLE): an algorithm for the validation of image segmentation, IEEE Trans. Med. Imaging 23 (7) (2004) 903–921.
[78] M. Mignotte, A label field fusion Bayesian model and its penalized maximum rand estimator for image segmentation, IEEE Trans. Image Process. 19 (6) (2010) 1610–1624.
[79] M. Mignotte, Segmentation by fusion of histogram-based-means clusters in different color spaces, IEEE Trans. Image Process. 17 (5) (2008) 780–787.
[80] P.G. Benardos, G.C. Vosniakos, Optimising feedforward artificial neural network architecture, Engineering Applications of Artificial Intelligence 20 (3) (2007) 365–382, doi:10.1016/j.engappai.2006.06.005.