Genetic Programming and Evolvable Machines (2023) 24:15
https://doi.org/10.1007/s10710-023-09464-0

Semantic segmentation network stacking with genetic programming

Illya Bakurov[1,2,3] · Marco Buzzelli[4] · Raimondo Schettini[4] · Mauro Castelli[1] · Leonardo Vanneschi[1]

Received: 6 September 2022 / Revised: 14 September 2023 / Accepted: 23 September 2023 /
Published online: 26 October 2023

Abstract

Semantic segmentation consists of classifying each pixel of an image and constitutes an essential step towards scene recognition and understanding. Deep convolutional encoder–decoder neural networks now constitute state-of-the-art methods in the field of semantic segmentation. The problem of street scenes' segmentation for automotive applications constitutes an important application field of such networks and introduces a set of imperative exigencies. Since the models need to be executed on self-driving vehicles to make fast decisions in response to a constantly changing environment, they are not only expected to operate reliably but also to process the input images rapidly. In this paper, we explore genetic programming (GP) as a meta-model that combines four different efficiency-oriented networks for the analysis of urban scenes. Notably, we present and examine two approaches. In the first approach, we represent solutions as GP trees that combine networks' outputs such that each output class's prediction is obtained through the same meta-model. In the second approach, we propose representing solutions as lists of GP trees, each designed to provide a unique meta-model for a given target class. The main objective is to develop efficient and accurate combination models that could be easily interpreted, therefore allowing gathering some hints on how to improve the existing networks. The experiments performed on the Cityscapes dataset of urban scene images with semantic pixel-wise annotations confirm the effectiveness of the proposed approach. Specifically, our best-performing models improve systems' generalization ability by approximately 5% compared to traditional ensembles, 30% for the less performing state-of-the-art CNN and show competitive results with respect to state-of-the-art ensembles. Additionally, they are small in size, allow interpretability, and use fewer features due to GP's automatic feature selection.

Keywords Genetic programming · Stacking · Semantic segmentation · Ensemble learning · Deep learning

Extended author information available on the last page of the article

1 Introduction

Semantic segmentation (SS) is a supervised machine learning (SML) technique that refers to the process of assigning a particular class to every pixel in an image. Recent advancements in the field of deep learning (DL) have also fostered the field of SS [19, 40]. The pioneering work of Long et al. [35] from 2015, where several convolutional neural networks (CNNs) were adapted and successfully applied to solve the SS task, has stimulated the scientific community towards exploration and sophistication of DL-based SS [13, 17, 53]. Current state-of-the-art DL-based SS neural networks are mainly built upon an encoder–decoder architecture. Typically, the encoder is a pre-trained deep CNN that downsamples the input images to feature maps. This technique allows for a reduction of the spatial resolution and, therefore, the memory usage while extracting relevant semantic features. The decoder part follows the encoder and performs gradual upsampling of the low-resolution feature maps to recover the original spatial resolution [19].

The street scenes' SS for automotive applications constitutes one of the most challenging tasks in the field of computer vision (CV) as several new imperative exigencies emerge. As the model needs to be executed continuously on self-driving vehicles to take fast decisions in response to constantly changing environmental events, they are not only required to operate accurately but also to process the input images fast enough to allow timely decision-making of the self-driving vehicle. In this context, several new efficiency-oriented architectures were developed having these requirements in mind [36, 38, 39, 43, 52]. However, these tendentiously obtain a gain of efficiency at the cost of accuracy's deterioration. Moreover, these networks exhibit different performances when evaluated on different target classes. All this suggests that their efficient and cautious combination could yield an effective performance improvement.

Given the increasing importance and the underlying complexity of street scenes' SS, we propose to use Genetic Programming (GP) [29, 46] as a meta-learning technique to stack different efficiency-oriented SS architectures in the context of street scene recognition for automotive applications. The objective of our study is fourfold: (i) design a SS system that is simultaneously efficient and accurate, and can be used in the context of fast street scenes' segmentation, (ii) understand the synergistic effect of each base SS network (the base learner) in the ensemble, (iii) obtain an interpretable fusion model that can potentially unfold new insights on the neural architecture design and (iv) perform time-complexity analysis of the proposed system.

Particularly, following the work of Mazzini and Schettini [38], we consider the following four efficient neural architectures as the base learners for our stacking model: ENet [43], ERFNet [52], ESPNet [39], and SSNet [36, 38]. We tested our approach on the Cityscapes dataset [14] - a popular and widely utilized dataset of urban scene images with pixel-wise annotations conceived for semantic understanding of urban street scenes. The original annotations include 30 different object classes, and only 19 are typically used for training and evaluation.

The paper is organized as follows. Section 2 provides the necessary background for this study covering both ensemble learning and semantic segmentation with efficient neural architectures. Section 4 presents the proposed approach for stacking efficiency-oriented networks for fast scene recognition using GP. Section 5 describes how the experiments were organized, the dataset, and the parameters used in this study. Section 6 presents and discusses the experimental findings in light of the experimental objectives. Section 7 concludes the work and proposes ideas for future research.

2 Background

2.1 Efficiency-oriented deep neural networks for semantic segmentation

Historically, the development of deep CNNs for SS mainly relied on advancements in the neural architectures' design for image classification and object recognition tasks. A breakthrough of immeasurable worth was achieved in 2015, when Long et al. [35] successfully adapted state of the art SOTA CNNs, originally conceived to solve image classification and object recognition tasks, to solve the task of predicting a label for each pixel in the image (aka dense prediction tasks). This scientific breakthrough re-oriented the scientific community's focus towards DL-based techniques for SS problem-solving. The authors proposed to transform fully connected layers into fully convolutional through a process called "convolutionization" enabling a given classification network, such as AlexNet [30], VGG [55] and GoogLeNet [57], to output a heatmap. Given that classification networks sequentially subsample the input image throughout the network to keep the filters small, and the computational requirements reasonable, their variants in the form of fully convolutional networks (FCNs) produced an output whose size is significantly reduced compared to the networks' input. In this sense, the authors proposed to stack several in-network upsampling layers to obtain a prediction for the whole input image. Finally, to refine predictions, the authors proposed combining deep, coarse, semantic information in the upsampling layers with the local appearance information from the respective downsampling layers.

The fully convolutional networks (FCN) paved the way for the next-generation DL-based SS systems such as PSPNet [65] and DeepLab [13], which currently constitute the state-of-the-art in the field. Although these networks achieve superior results in the reference semantic image segmentation tasks, such as the Cityscapes [14], they also require a gargantuan amount of learnable parameters and long inference times. Meanwhile, society's digital transformation dictates new standards for DL-based systems: the ever-growing number of battery-powered mobile devices and their applications (like, for instance, home-automation devices, augmented reality wearables, autonomous vehicles, and flying drones), require algorithms not only to operate reliably but also to fit in devices' limited memory, have low power consumption, and operate in real-time. As a matter of fact, the commonly accepted processing precondition for autonomous vehicles is at least 30 frames per second (FPS) [38, 52]; therefore, the large and complex DL-based SS systems are of no use in this context. To tackle this efficiency-accuracy trade-off, part of the scientific community focused on the so-called efficiency-oriented architectures, such as ENet [43], EDANet [34], ERFNet [52], and SSNet [36, 38], that will be discussed, in detail in the following Sections.

2.1.1 ENet

ENet was the first CNN architecture specifically optimized for fast inference and high accuracy in SS tasks. It was the first high-performance DL-based architecture able to operate in real-time applications [43].

ENet's architecture was strongly inspired by residual blocks, the building block of ResNets [22], and accommodated all the main achievements in the neural architecture design of its time. The authors proposed to heavily reduce input size during the early stages of the network (particularly in the first two blocks) to efficiently compress the spatially redundant visual information and, therefore, decrease the computational costs of the network. Following the work of [57], the authors conducted the pooling operation in parallel with a convolution of stride two and concatenated the resulting feature maps to avoid too aggressive dimensionality reduction that could hinder the information flow. In continuation, to avoid overly downsampling, the authors used dilated convolutions in the main convolutional layers, inside several bottleneck modules, in the stages that operate upon the smallest resolutions. Following the design principles of SegNet [3], ENet's authors decided to save max-pooling layers' indices to produce sparse upsampled maps in the decoder that are then convolved with a trainable decoder filter bank to produce dense feature maps. In such a way, the network can better retain boundary information in the extracted image representations without storing all the encoder feature maps in memory. Unlike SegNet, ENet's decoder is not an exact mirror of the encoder. Instead, ENet's architecture consists of a relatively large encoder and a small decoder. This aspect was motivated by the consideration that the decoder's role is to upsample the output of the encoder, only fine-tuning the details. Motivated by the findings of [21], the authors replaced all the Rectifying Linear Unit (ReLU) nonlinearities with parametric ReLUs (PReLUs), which allow one to learn the negative slope of non-linearities efficiently. Following the findings of [26], later ratified in [58], factorized (a.k.a. asymmetric) convolutions were employed to reduce the amount of potentially redundant parameters and speedup the inference times. Finally, to deal with relatively small segmentation datasets, the authors placed spatial dropout at the end of the convolutional branches as a regularization method. ENet's pioneering design choices greatly influenced the design standards of posterior efficiency-oriented networks.

2.1.2 ERFNet

Romera et al. [52], proposed a novel efficient residual factorized CNN for real-time SS, called ERFNet, that, similarly to ENet [43], was also inspired by ResNets's residual learning framework [22]. However, their architecture differs from ENet's in three main aspects. Firstly, the authors reduced the number of learnable parameters in the residual (non-bottleneck) modules, by rewriting the kernels with one-dimensional

(1D) factorized convolutions [25]. Concretely, the authors replaced $N \times N$ (which can be seen as matrices), as a product of two smaller 1D kernels (which can be seen as vectors of shapes $N \times 1$ and $1 \times N$, respectively). Such a simplification allows reducing the computational costs while retaining a similar accuracy compared to the traditional two-dimensional (2D) convolutions [23, 25, 32]. The proposed blocks were stacked sequentially, with varying dilation rates, to build the encoder segment of the network.

Second, similarly to ENet, ERFNet's architecture is an asymmetric encoder-decoder with a smaller decoder. However, the authors did not use max-pooling layers' indices during the upsampling phase; instead, the ERFNets' decoder uses simple deconvolution layers with stride 2 (a.k.a. transposed convolutions). The authors pointed out that deconvolutions can save on computational resources while obtaining a similar (or a slightly better) accuracy.

Third, to improve the efficiency of the downsampling operator, the authors proposed to downsample the input images and the subsequent feature maps by concatenating the parallel outputs of a single 3 ×3 convolution (with stride 2) and a max-pooling layer. This approach was taken from ENet; however, unlike in ENet where it was only used in the initial block, this approach was used in all the downsampling blocks and not only the first (initial) block. The experiments conducted on the publicly available Cityscapes data set [14] demonstrated that ERFNet could achieve comparable accuracy values to the state-of-the-art networks while being several orders of magnitude faster to compute.

2.1.3 EDANet

In [34], the authors proposed a novel CNN architecture for SS called efficient dense modules of asymmetric convolution (EDANet). The architecture followed the novel trends in the neural architectures' design, such as the densely connected structure articulated in [24]. Specifically, the authors proposed a novel structure based on a point-wise convolution layer and two pairs of Efficient Dense modules with Asymmetric convolutions (EDA). The overall architecture is composed of EDA modules' stacks, called EDA blocks. To accelerate the actual inference speed, the authors arranged the composite functions in the so-called post-activation manner: (i) convolution, (ii) batch normalization, and (iii) ReLU. The dense connectivity proposed in [24] was modified from layer-level to module-level, meaning that the output of each EDA module is the concatenation of its input and the newly produced features. Concatenating the features learned from each module that has a different receptive field individually allowed EDANet to gather multi-scale information together naturally. Intending to aggregate more contextual information, the authors used dilated convolutions at the second asymmetric convolution pair of every EDA module but the initial three modules. To enlarge the receptive field gradually, the dilation rates in the system sequentially grow in value. As a downsampling strategy, the authors adopted the approach of ERFNet: by concatenating the parallel outputs of a single 3×3 convolution (with stride 2) and a max-pooling layer. Contrary to ENet and ERFNet, EDANet's authors decided to discard the decoder structure. At the end of the last EDA block, a point-wise convolution was added as a projection layer to output C

feature maps (where *C* equals the number of target classes in a given SS task), followed by a bilinear interpolation to upsample feature maps by a factor of 8 to the size of input images. Although such a design choice slightly deteriorated the network's accuracy, it also allowed for a reduction in computational costs. As a result, the proposed architecture is one of the most accurate among networks that exceed the real-time threshold of 30 FPS [34].

2.1.4 SSNet

Contrary to the research track defined in the ENet, ERFNet, and EDANet, other scientific community members focused their attention to improve the decoder's segment. Notably, Mazzini et al. introduced a novel upsampling module that effectively replaces the traditional operators like, for example, the bilinear interpolation and the nearest neighbors upsampling [36, 38]. In this work, we considered the authors' latest scientific achievement - the so-called spatial sampling network (SSNet) for fast scene understanding introduced in [38]. Given that SSNet constitutes a logical continuation of the research track paved by the so-called guided upsampling network (GUN) [36], we will first introduce the latter.

GUN [36] is a multi-resolution neural architecture built upon a pre-trained DRN-D-22 neural network introduced by Yu et al. [63] that jointly exploits high-resolution and large-context information. Besides efficiently adapting a novel multi-resolution encoder architecture, Mazzini et al. also redesigned the decoder segment by replacing the traditional operators with a novel guided upsampling module (GUM). This module enriches the upsampling by efficiently introducing a learnable transformation to improve semantic maps along objects' boundaries. Unlike the traditional upsampling operators (such as the bilinear interpolation and the nearest neighbors), which make use of a regular grid to sample pixels from the low-resolution images, GUM uses a warping grid, named guidance offset table (GOT), to correct the prediction map along objects' boundaries. Concretely, GOT contains two offset values for each pixel of the high-resolution feature map that shift the sampling coordinates of each element of the map in x and y dimensions, respectively. The offsets are predicted by a neural network branch named Guidance Module (GM). Therefore, the parameters are trainable by the backpropagation algorithm along with the whole neural network.

Further, Mazzini et al. [38] proposed a novel lightweight architecture for the encoder segment and an improved guided upsampling module (iGUM) that operates as a decoder segment. Unlike GUN, SSNet's encoder consisted of a single-resolution neural architecture built upon the previously presented ERFNet. Similar to ERFNet, SSNet uses several early downsampling layers to speed up the inference time. SSNet's authors were strongly inspired by ERFNet's non-bottleneck-1D module, originally developed as a more optimal solution to the so-called non-bottleneck modules proposed by [22]. However, instead of directly integrating the non-bottleneck-1D module from ERFNet in their architecture, Mazzini et al. redesigned it to speed up the inference time and allow the module to learn across channels by employing point-wise convolutions right before and after two asymmetric kernels. Recall that the non-bottleneck-1D module proposed by Romera et al. did not use point-wise convolutions to encourage cross-channel learning. Finally, instead of using traditional upsampling layers in the decoder segment, the authors use an iGUM layer as a decoder. The novel iGUN was proposed to improve the overall efficiency by reducing the number of learnable parameters required by GOT in the original GUM. In the original definition of GUM, each bidimensional coordinates vector of the regular sampling grid was summed with its corresponding bidimensional offset from GOT. In this sense, GM needed to learn $2 \times fN \times fM$ parameters, where f is the upsampling factor, and N and M are the spatial dimensions of the output probability map to be upsampled. The novelty of iGUM relies on the simplification of the warping grid's utilization. Specifically, the authors realized that a significant part of the offsets in GOT could be interpolated simply instead of being learned. Therefore, the authors proposed to learn a low-resolution GOT of size $2 \times N \times M$, reducing, as such, the number of learnable parameters and improving the system's efficiency.

2.2 Stacked generalization

Ensemble learning (EL) is a sub-field of machine learning (ML) inspired by humans' natural tendency to seek and weigh others' opinions prior to decision-making. Under this perspective, EL consists of combining several individual models, called base learners (BLs), in a way to produce an ensemble model which is expected to solve a given task better than any of the base learners in isolation [47, 50]. In general terms, EL methods differ in the way input data is represented and manipulated within the ensemble, whether the ensemble's BLs are trained independently or not, how the final prediction is performed, etc.

Stacked generalization (a.k.a. stacking) consists of training an ensemble from the combined outputs of several base learners. Specifically, it consists of two fundamental steps: (i) independently training the base learners to solve the underlying task, (ii) and then training a meta-learner from the base learners' predictions [62]. In other words, the predictions obtained from the base learners are used as inputs for a meta-learner. Consequently, stacking is expected to perform at least as well as (if not better than) the best base learner. The interpretability of a meta-learner model can be of high value:

1 Stacking/combination can aid the researcher's interpretation of the underlying models, by describing their corresponding qualities, such as the need for a refinement phase via max-pooling, akin to a noise reduction post-processing.
2 Interpretation can be also achieved in terms of the mutual relationship between underlying models. For example, as noted in [4], the addition operator may be assigned a probabilistic interpretation that translates to a generalization of the "OR" relationship, indicating that two models are complementary (and that a good segmentation result can be achieved by considering either of the two).
3 More generally, our stacking methodology could be extended for application to an arbitrary set of underlying models, including simple ones that provide a good starting point for their individual interpretability.

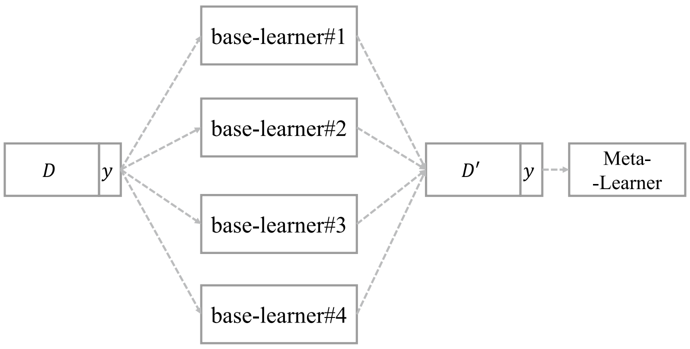


Fig. 1 Stacked generalization from four base learners (BLs)

Figure 1 illustrates a meta-learner combining predicted outputs of four BL models trained on the same input data D with target variable y. As one can see from the figure, the meta learner is trained on D', a dataset originated from the combined outputs of the four BLs, but the target variable is the same.

Theoretically, there are no restrictions nor precise recommendations regarding the definition of the meta-learning model and the BLs - these can belong to any known class of ML models. An overview of the scientific literature on stacked generalization for SML problem-solving does not suggest an agreement upon which models are to be used [50]. For example, some authors use regularized regression to combine predicted outputs from conceptually different BL models [49], while some use boosted ensembles [48]; others, use genetic algorithms (GAs) for stacking single-hidden-layer feed-forward networks after applying bagging on the training set [67]. Empirical evidence, however, shows that the most considerable performance improvement can be observed when stacking together more dissimilar BLs [6, 10]; when BLs are highly correlated in their outputs, stacking tends to overfit [49].

Due to its flexible representation and powerful inductive capabilities, GP can be seen as an effective meta-learner for stacking. With properly chosen operators and hyper-parameters, it can combine BLs in a highly non-linear fashion, better exploiting their outputs and achieving superior generalization ability. Moreover, GP intrinsically performs an automatic selection of the terminals (i.e., input features). Finally, the obtained GP tree can foster the interpretability of evolved solutions to some degree. Figure 2 shows a possible meta-learner evolved through GP: the terminal set consists of BLs predictions (white nodes), which are combined using mathematical operators (grey nodes).

3 Related works

3.1 Stacking with GP

To our knowledge, the first evidence of GP's usage in the context of stacked generalization comes from 2006 [27], when it was used as a meta-learner to combine

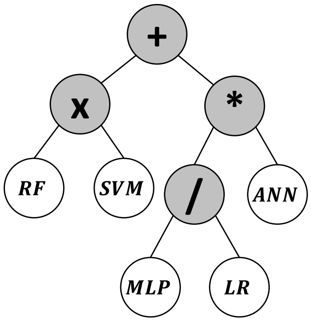


Fig. 2 Possible meta-learner evolved by means of GP. The nodes in grey represent the mathematical operators, whereas the nodes in white represent the outputs of different BLs

predicted outputs from ten feed-forward artificial neural networks (FF-ANNs) with different neural architectures (three had no hidden layer, five had one hidden layer and the remaining two had two hidden layers); the experimental evidence based on 22 publicly available datasets demonstrated the superiority of the method against other evaluated methods. In [12], an equivalent approach was compared against three other ensemble approaches based on GAs. The experimental results involving four synthetic and one real-world symbolic regression problem confirmed the predominance of GP-based ensembles not only against the best BL but also the three different types of GA-based ensembles. Zameer et al. [64] used GP as a meta-learner to combine four different types of artificial neural networks (ANNs) for mapping meteorological measures and wind power; the obtained results were compared with the recent artificial intelligence-based strategies on several measures and demonstrated the efficacy of the proposed ensemble scheme. Sharma et al. [54] used a multi-level stacking ensemble to forecast future incidences of conjunctivitis disease; the experimental result showed that stacking allowed to decrease various error metrics by a significant amount. Bakurov et al. [6] performed an extensive exploration of GP as a meta-learner for stacked generalization; the study assessed some of the recent advancements in the field of GP [7, 18, 31, 41, 60]. The experimental evidence based on seven synthetic and four real-world symbolic regression problems confirmed the effectiveness of GP as a meta-learning technique when compared against eight other SML methods.

The absolute majority of studies in the literature apply stacking to solve SML problems involving cross-sectional or time-series data. In this context, a single data instance in the training set x_i can be characterized as a vector in k-dimensional space with the corresponding output value y_i:$y_i : x_i = x_i^1, ..., x_i^k$. Such a data representation makes problem-solving more accessible and simple as high-level application programming interfaces (APIs) such as scikit-learn [11, 44] can be used; as a matter of fact, this API allows one to train a stacking ensemble using dozens of SML models in a few lines of code. This is not the case, however, when solving CV tasks, like the dense prediction task addressed in this work where one data instance can be seen as a tensor of four dimensions (batch size, number of classes, height, and width). Therefore, it is not surprising that there are significantly fewer works in the literature reporting the application of stacking in the field of CV. The next Section below enumerates some of the studies that use GP.

3.2 GP in image processing and computer vision

GP has been successfully applied to solve several image processing (IP) and CV applications. One of the earliest examples of GP's application for real-world problem-solving is the work presented by Tackett [59] of Hughes Missile Systems in late 1993 where GP was used to construct a binary classifier that combines feature vectors extracted from images using the Multi-function Target Acquisition Processor algorithm. The objective of the study was to identify whether a target (like, for instance, a tank, aircraft, etc.), was present in a given patch of infra-red images taken from a cluttered terrain (containing rocks, bushes, etc.). It was shown that GP can achieve higher performance and reduced computational complexity when compared with a binary tree classifier and a back-propagation neural network. In fact, one of the most successful evolved classifiers comprised just 25 program elements, used 62.5% of input features, employed just four simple arithmetic operators, and could have been written in a single line [45]. A similar approach was taken by Agnelli et al. [1], where GP was used to combine 12 low-level domain-specific feature detectors using essentially simple mathematical operators; the features were selected based on the authors' previous experience in the specific domain, and the efficiency-effectiveness trade-off. The approach was assessed in the scope of binary classification of image segments extracted from printed pages (including books, serials, and newspapers). Instead of capitalizing upon existing feature detectors (aka filters), Harris and Buxton [20] and shortly after Poli [45] proposed to discover optimal problem-specific filters using GP. While the former used GP to evolve edge detectors for 1-D signals and image profiles, the latter took a broader approach to obtain cost-effective filters capable of highly and selectively emphasizing the image characteristics for a given task. Further, Poli combined the evolved filters with simple thresholding strategies to detect features of interest and build pixel-classification-based (binary) segmentation algorithms. Roberts and Claridge [51] used GP to automatically evolve a skin lesion segmentation system from segmentation images provided by an expert clinician. The function set included imaging operators such as thresholds, morphological operations, logical operations, region intensity functions, edge filtering, merging, quantisation, etc. Interestingly, the authors were able to achieve high rates of generalization ability (assessed on 90 images) by training on just eight images. A similar approach to evolve a binary segmentation system by means of GP was taken by Sing et al. [56]. In their study, common arithmetical operators were coupled with a wide range of image filters, as well as morphological and enhancement operators. The representation was then converted into a sequential MATLAB binary segmentation program. The proposed approach was compared with a SOTA method based on GAs and reported not only superior performance but also simpler solutions. Al-Sahaf [2] proposed an end-to-end image classification framework called Two-Tier GP (2TGP) that simultaneously evolves the feature based on raw pixel input and the classification rules. To extract features, the authors use the so-called aggregation functions, which extract a given region of interest from the image and compute an aggregation statistic (mean, median, standard deviation, minimum and maximum). To derive the classification rules, traditional arithmetic operators and one decision rule are used. The proposed method was assessed on four binary classification tasks and was shown to outperform a traditional feature-based image classification by GP and another GP method which also aims to automatically extract image features. Some of the evolved solutions were capable of generating genuine features.

Recent works report successful usage of GP for image enhancement in real-estate marketing [16] - a complex scenario where the overall aesthetics and technical aspects of the image must be adjusted to deliver a realistic, credible, and attractive result for customers. The authors proposed a generic framework to conceive effective image enhancement pipelines combining SOTA image processing filters and GP. When choosing the set of terminals, the authors decided to focus on five main aspects of image enhancement: contrast adjustment, brightness adjustment, colour balance, noise removal, and sharpening. Also, the authors introduce an "if-then-else" function that, depending on image-related features that capture characteristics of the perceived quality of the image applies one of two image enhancement branches of the GP tree. The GP-based system was trained to jointly optimize three metrics that reflect the perceived technical image quality, aesthetics, and commercial attractiveness of the processed images. Later, the approach was extended with conditional adversarial networks for image-to-image translation to improve even further the overall image quality [15]. It was shown that the framework was capable to achieve better performance than the SOTA image enhancement tools, including those based on generative adversarial networks. One of the largest contributions of this work consists of universality, as most of the SOTA approaches are primarily non-modular and problem-specific. This paper complements the previous studies on the use of GP in the context of image analysis. In the existing research, Bakurov et al. [4, 5] proposed a framework for full-reference image quality measures' (FR-IQAMs) formulation through GP in two phases, characterized by different mutation strategies. Specifically, the authors derived terminal sets from the building blocks of the so-called structural similarity at different levels of abstraction. The empirical evidence from a cross-dataset validation proved the method's superiority compared to traditional FR-IQAMs. Moreover, it was shown that the obtained solutions are competitive with more complex deep image quality measures. To complement the work presented in [4] and later extended in [5], in this paper, we propose to solve the problem of street scenes' segmentation for automotive applications. Thus, we show that GP can be used not only to design competitive image quality measures but also to address even more complex tasks, such as dense prediction.

The most similar works to the one presented in this paper are [9] and [8]. Bianco et al. [9] applied GP as a meta-learner for the predicted outputs of nine video change detection algorithms. It was observed that no single algorithm was able to achieve superior performance. Instead, different algorithms are best suited to different problems. For this matter, in order to create a robust ensemble leveraging the algorithms' peculiarities, the authors proposed to combine their predicted outputs (binary foreground/background masks) using a set of unary, binary, and n-ary functions (e.g., logical AND and logical OR), as well as post-processing operators (e.g., filters for noise removal) to polish the final output. The experimental evidence on a dedicated dataset composed of different types of video sequences (ChangeDetection.net 2014 challenge) demonstrated a significant superiority of the proposed approach. To the best of our knowledge, this is the first work that uses GP to select and combine different video change detection algorithms. A more recent study reports the use of GP to combine the outputs of existing saliency detection algorithms (which are essentially binary masks), using a set of provided operations [8], in the context of a three-step neural architecture search for image saliency fusion. The authors used three groups of functional operators for GP, each operating on different domains of pixels: (i) 2D spatial neighborhood of the pixels belonging to the same saliency map, (ii) stacks of pixels across different saliency maps, and (iii) individual pixels without considering any neighborhood. Similarly to the study in [9], no state-of-the-art algorithm reported superior performance on the majority of datasets and problem domains. The experimental evaluations showed that the proposed saliency fusion approach could successfully outperform SOTA methods on a popular image saliency benchmark.

A deep analysis of the literature suggests that GP was not used yet for stacking deep CNNs' output for semantic segmentation with more than two classes. The most similar works perform stacking of binary masks [8, 9]. The presence of several classes (in our problem there are 19) makes the task particularly challenging while, at the same time, providing more degrees of freedom for the proposed approach, discussed in Sect. 4.

4 Proposed method

The SS networks generate a high-dimensional logit tensor, representing the non-normalized per-pixel pseudo-probability distribution over the problem classes. The shape of the logits tensor is $[bs, C, H, W]$ where bs stands for the batch size, C corresponds to the number of target classes and $H \times W$ represents the spatial dimensions of the target image. To produce the final segmentation mask, this tensor is first subject to the softmax activation function, which normalizes the logits to a proper probability distribution; then the index of the largest value across the channel dimension is chosen at each pixel of the mask to represent the predicted class. In abstract terms, one can think of logits as a network's internal representation (conceptualization) of the visual scenes before making the final decision; as such, they can simultaneously encode uncertainty about some of the classes and certainty regarding others.

Considering one is interested in combining predictions from several CNNs for SS, applying a simple voting classifier on the networks' predicted segmentation masks might hinder the rich representational potential encoded in the logits. We hypothesize that a properly designed ensemble model should capitalize upon the *concepts* generated by different CNNs for SS, reduce uncertainty and improve the overall confidence (resulting in a better segmentation). Such a model should take into account both spatial and cross-channel relationships between different values of logits across different networks. Moreover, the final ensemble model should be simple enough to enable its deployment for real-time applications (ideally, also discarding some of the networks altogether). To the best of our knowledge, GP is the only tool that can be used to stack the multidimensional tensors of logits from the four efficiency-oriented CNNs for SS and fulfil the aforementioned requirements. Specifically, the flexibility introduced by GP's tree-based representation allows manipulation and non-linear combination of logits at different levels through a wide range of operators, including those widely used in the field of CV and IP (like, for example, convolution and pooling). Moreover, GP allows for direct interpretation of candidate solutions which can be useful to extract additional valuable information. Finally, thanks to its ability to perform automatic feature selection and to generate highly non-linear models, we expect GP to evolve solutions that are both efficient and effective. The proposed approach for stacking SS networks approach is explored into two variants: Sect. 4.1 presents the so-called multi-class variant (MC), whereas Sect. 4.2 shows the so-called single-class variant (SC). Each implies a different perspective over the solutions' representation and, consequently, a different search procedure.

4.1 Multi-class stacking approach

In this variant of GP stacking, the prediction at each output class is obtained through the same evolved meta-model. From now on, it will be called the multi-class stacking GP variant (MC-S-GP). The terminal set for the MC variant is composed of the logits[1] obtained from the four high-performance SS neural architectures described in Sect. 2.1. In this sense, one terminal element is a 4-dimensional tensor with sizes $[bs, C, H, W]$, where bs stands for the batch size, $C = 19$ corresponds to the number of target classes in the Cityscapes dataset's instance and $H \times W$ represents the spatial dimensions of the input images. The traditional arithmetic operators were used along with CV-specific operators (like pooling, smoothing, and edge detection). A complete enumeration and description of the operators can be found in Sect. 5.4. We used standard GP with sub-tree mutation and sub-tree crossover to explore the search space of all possible stacking models. To evaluate a given candidate solution - a GP tree representing a stacking model - one needs to (i) pass individual networks' logits through the tree to obtain the stacking model's logits (a tensor with sizes $[bs, C, H, W]$), (ii) apply the softmax activation function followed by the argmax function, and (iii) compute the evaluation function between the predicted segmentation maps and the respective targets (see Sect. 5.1 for a detailed description of the fitness function).

Figure 3 illustrates a potential candidate solution produced by MC-S-GP. Specifically, it is possible to see how a candidate solution can be represented both in terms of the Polish prefix notation (the equation on the left-hand side) and as a LISP tree (the hierarchical scheme on the right-hand side). The internal nodes of the tree (the circles) represent the primitive functions, whereas multi-colored stacks of rectangles represent the terminals. Given that a particular terminal corresponds to a multidimensional tensor of the SS network's logits, each uniquely colored rectangle in the stack regards a specific target class of an output segmentation map. In the context of our study, the logits tensor has shape $bs \times 19 \times 256 \times 256$, where bs stands for batch size, 19 corresponds to the number of target classes in the Cityscapes dataset's instance, and 256×256 represents the spatial dimensions of the input images.

[1] The logits can be defined as the predicted feature maps before the softmax activation layer.

```
Let $[BLs]$ be the base learners' set, $[P_{BLs}]$ the respective parameters, $X_{train}$ and $X_{test}$
the input training and test data, respectively, and $y_{train}$ and $y_{test}$ the corresponding
target annotations:

1. create two empty lists, $X'_{train}$ and $X'_{unseen}$, to store the base learners'
   predicted logits;
2. $for\ (bl, p_{bl})\ in\ ([BLs],\ [P_{BLs}])$:
   (a) using $X_{train}$ and $y_{train}$, train $bl$ with parameters specified in $p_{bl}$;
   (b) extract the predictions of $bl$ on $X_{train}$ and $X_{unseen}$ and append them to
       $X'_{train}$ and $X'_{unseen}$, respectively;
3. compose the set of terminals $T$ for the GP algorithm as a stack of $len([BLs])$
   input features;
4. execute the GP algorithm with $X'_{train}$ and $y_{train}$ as the training data, and
   $X'_{test}$ and $y_{test}$ as the test data;
```

Algorithm 1 Pseudo-code for the proposed MC-S-GP method.

Algorithm 1 shows the pseudo-code for the proposed MC-S-GP method, following the nomenclature defined in [6]. Notice that the very same stacked generalization is applied to combine networks' predicted outputs at every target class (1, 2, 3,..., 19).

4.2 Single-class stacking approach

The dictate of a unique stacking model across different target classes may limit the system's "degrees of freedom" and hinder its potential to find a highly accurate solution. That is, it might be the case that the learned stacking model could be

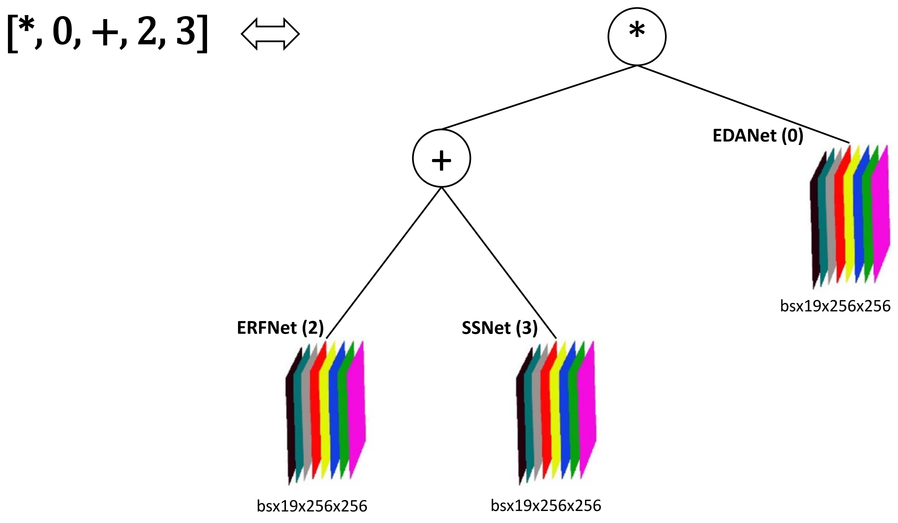


Fig. 3 A potential MC-S-GP model. The figure shows how a given GP individual can be represented as a list of program elements (on the left), and a LISP tree (on the right)

sub-optimal when combining networks' predictions for some classes (such as "car", "road", and "pedestrian"). In this sense, inspired by the work of Muni et al. [42], we conceive and explore an alternative representation for candidate solutions that allows GP to jointly develop stacking models tailored to each target class. Specifically, we propose to represent a given candidate solution as a C-dimensional list of trees $[T] = [T_0, T_1, ..., T_{C-1}]$, where $C = 19$ and stands for the number of target classes in the Cityscapes dataset's instance; each tree at a given position in the list consists of a potentially unique stacking model for a given class. Under this perspective, the MC-S-GP variant can be seen as a special case where all the C trees in $[T]$ are the same. From now on, this second alternative of the proposed approach will be called the single-class stacking GP (SC-S-GP) variant.

Given the flexibility of the new representation, we decided to give it the possibility to utilize all the possible information coming from the feature maps predicted by the four networks. Specifically, when searching a stacking model for a given target class, the system was provided access to other classes' feature maps, although with lower probability. Note that this is different from restricting the terminal set of a given tree in $[T]$ to class-specific feature maps predicted by the four BLs, something which implicitly happened in the MC-S-GP variant. This decision implies that the terminal set for the SC-S-GP variant is obtained by concatenating the four networks' logits on the channel dimension. In this sense, one terminal element is a 4-dimensional tensor with sizes $[bs, 4 \times C, H, W]$, where $4 \times C$ represents the concatenation of the four networks' logits on the channel dimension (given that $C = 19$, there will be 4x19=76 output channels).

This strategy was motivated by the importance of taking into account the global context of the scenes in order to improve the local predictions. Specifically, we wanted to allow the resulting model to make use of more information from the scene's context when trying to differentiate between classes like, for instance, a pedestrian and a rider in a road scene. It happens that simply learning to identify people in a scene is not enough - the context in which they happen to appear might be crucial for the posterior decision-making process. In this sense, we believe that properly "mixing" the feature maps from different classes may allow the system to achieve superior performance.

A more complex and versatile solutions representation required us to adapt the search algorithm accordingly. In this sense, for the SC-S-GP variant, we propose to use a local search (LS) strategy where the GP's sub-tree mutation plays the role of the neighborhood function. Specifically, the proposed LS strategy consists of exploring the neighborhood of a given size for every successive tree (T_{C_i}) in $[T]$ (starting from T_{C_0} up to $T_{C_{18}}$). To conclude one iteration, it is necessary to explore the neighborhood of every tree in the C-dimensional list $[T]$. Every candidate neighbor of T_{C_i} is evaluated in the context of the class it represents; that is, each neighbor of T_{C_i} is assessed in terms of its ability to improve the system's fitness on class i. To evaluate $[T]$ (i.e., a given candidate-solution), one needs to (i) pass the corresponding networks' logits to each tree T_{C_i} in the list $[T]$ to obtain the stacking model's logits at each class, (ii) concatenate the resulting C tensors along the channel dimension to form a tensor with sizes $[bs, C, H, W]$), (iii) apply the softmax activation

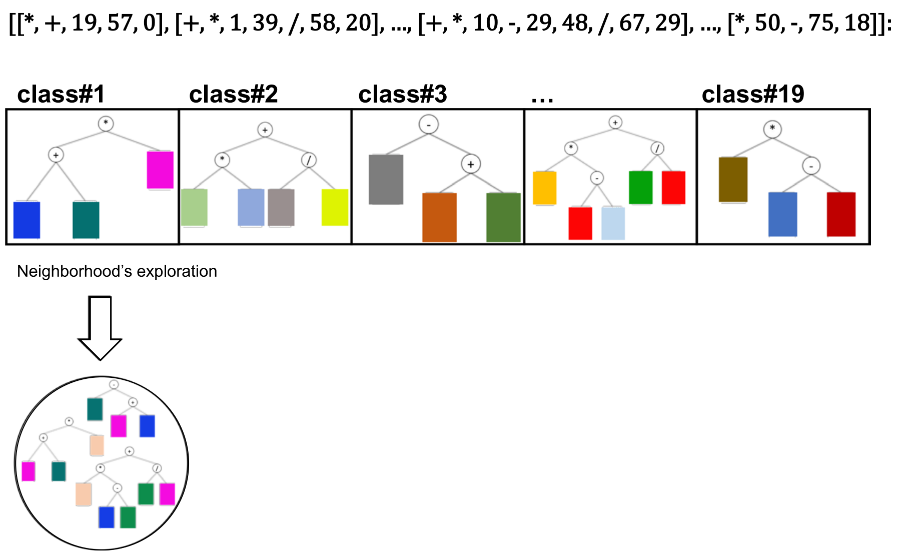

[[*, +, 19, 57, 0], [+, *, 1, 39, /, 58, 20], ..., [+, *, 10, -, 29, 48, /, 67, 29], ..., [*, 50, -, 75, 18]]:

Fig. 4 A potential SC-S-GP model. The figure shows how a C-dimensional list of LISP trees (in above) can be used to represent a joint SC-S-GP model where the predictions for each class are obtained by a unique GP tree (i.e., sub-model). The circle below shows a potential neighborhood of the tree T_{C_0}

function followed by the argmax function, and (iv) compute the evaluation function between the predicted segmentation maps and the respective targets (see Sect. 5.1 for a detailed description of the fitness function).

Figure 4 illustrates a potential candidate solution (i.e., a stacking model), in the scope of the SC-S-GP variant. From the figure, a candidate solution $[T]$ stores C trees, each meant to combine networks' logits at a given target class C_i. For example, the tree T_{C_0} regards the first target class and it combines the respective output logits from the four different networks. Contrarily to the MC-S-GP variant, the input terminals for a given tree T_{C_i} have shape $[bs, H, W]$ as they regard only one dimension of the output logits. In this sense, the number of distinct terminals comprising a GP tree at a given class is 76 (four SS networks $\times$ 19 classes). To construct the trees, we decided to use the same functions' set as in 4.1.

Algorithm 2 shows the pseudo-code for the proposed SC variant of the S-GP method. Note that here we follow the nomenclature defined in [6].

5 Experimental environment

5.1 Fitness function

Intuitively, a successful SS system is one that maximizes the overlap between the predicted and true pixel classes. To measure the quality of a given stacking model on the underlying SS task, we use the mean of class-wise intersection over union

```
Let $[BLs]$ be the base learners' set, $[P_{BLs}]$ the respective parameters, $X_{train}$ and $X_{test}$
the input training and test data, respectively, and $y_{train}$ and $y_{test}$ the corresponding
target annotations:

1. create two empty lists, $X'_{train}$ and $X'_{unseen}$, to store the base learners'
   predicted logits;
2. $for\ (bl, p_{bl})\ in\ (BLs,\ P_{BLs})$:
   (a) using $X_{train}$ and $y_{train}$, train $bl$ with parameters specified in $p_{bl}$;
   (b) extract the predictions of $bl$ on $X_{train}$ and $X_{unseen}$ and append them to
       $X'_{train}$ and $X'_{unseen}$, respectively;
3. compose the set of terminals $T$ for the LS-based algorithm as a stack of
   $len([BLs])$x$C$ input features;
4. execute the LS-based algorithm with $X'_{train}$ and $y_{train}$ as the training data,
   and $X'_{test}$ and $y_{test}$ as the test data:
   (a) generate and evaluate an initial candidate solution $[Tr]$;
   (b) $for\ Tr_{C_i}\ in\ [Tr]$:
       (i) $for\ i\ in\ range(pixelneighborhood\ size)$:
               (A) using GP's sub-tree mutation operator as the neighborhood
                   function, generate one neighbor of $T_{C_i}$ called $T'_{C_i}$;
               (B) compute the similarity between $[Tr]$ and the target segmen-
                   tation map with $T'_{C_i}$ as the stacking model for class $C_i$ in
                   $[Tr]$;
               (C) if using $T'_{C_i}$ improves the similarity between $[Tr]$ and the
                   target segmentation map, then $T_{C_i} = T'_{C_i}$.
```

Algorithm 2 Pseudo-code for the proposed single-class variant of the S-GP.

(mIoU). It is computed as the class-wise mean of the intersection over union (IoU), also known as the Jaccard coefficient:

$$Jaccard = IoU = \frac{\|A \cap B\|}{\|A \cup B\|} = \frac{TP}{TP + FP + FN}, \tag{1}$$

where A and B stand for the predicted and target segmentation masks for a given class, and TP, FP, and FN stand for true positives, false positives, and false negatives, respectively. Typically, SS models include a background class in addition to the target classes. In simple terms, the background class is used to represent a *none of the above* class, which serves as an adjunct to all the other classes; therefore, if we want to identify C classes in an image, in practice, there will be $C + 1$ classes. However, when objects of interest occupy a relatively small part of the image, even a naive model predicting background everywhere would have good MIoU for the background class. When calculating metrics, one typically is more interested in target classes rather than the background. Intuitively, a good identification of the objects will translate into a good identification of the background as well. To deal with the aforementioned limitations of the quality measurement, it is common to ignore the background class before taking the mean over IoUs [19, 36, 38, 40].

5.2 Dataset

Cityscapes is a popular and widely-utilized dataset of high-resolution urban scenes. It consists of 5000 annotated images, which are provided in three sets: 2975 training images, 500 validation images, and 1525 testing images. From the 5000 annotated images, only 3475, which regard the training and validation sets, are fine-annotated; the remaining 1525 from the test partition are coarse-grained annotations. For this reason, we only used the train and validation sets of the data. The original annotations include 30 different object classes; out of these, 19 were used in our study following the common practice adopted by the scientific community [34, 36, 38, 43, 52]: road, pole, sky, bus, sidewalk, traffic light, person, train, building, traffic sign, rider, motorcycle, wall, vegetation, car, bicycle, fence, terrain, truck, and background. The images were collected in 50 different European cities, in different parts of the year, with a large variability of weather and illumination conditions, making this dataset highly convenient for developing and benchmarking solutions designed for real-world automotive applications. The images were cropped down to 256×256 in a random fashion to foster the experiments.

5.3 Base learners' hyper-parameters

All models were trained following the procedure described in [33], minimizing the cross-entropy loss between the ground truth and the predicted classes for each pixel. We adopted Adam as a stochastic optimizer [28], using an initial learning rate of 5×10^{-4}, updated through the following polynomial decay learning rate policy:

$$LR(epoch) = \left(1 - \left(\frac{epoch}{total\_epochs}\right)^{0.9}\right) \cdot LR_0, \tag{2}$$

where *epoch* is the 0-based index of the current epoch, LR_0 is the initial learning rate, and *total_epochs* is set to 150 for all experiments, after preliminary evaluation. We used batch size = 6 in order to fit memory constraints. The images were pre-processed by subtracting the mean and dividing by the standard deviation computed on ImageNet [30], and data-augmented by random scale augmentation sampling the scaling factor from a uniform distribution with the interval [0.5, 2].

Figure 5a and b present the correlation heatmaps between the four networks calculated on the training and test data. The correlation between a given pair of networks was estimated as the average MIoU between their predicted feature maps at a given data partition. From the figure, it becomes clear that network outputs are not significantly correlated. This aspect means that the candidate base learners in the stacked ensemble can be said to be heterogeneous, and a significant improvement in the task can be potentially obtained if using them together in an ensemble [10].

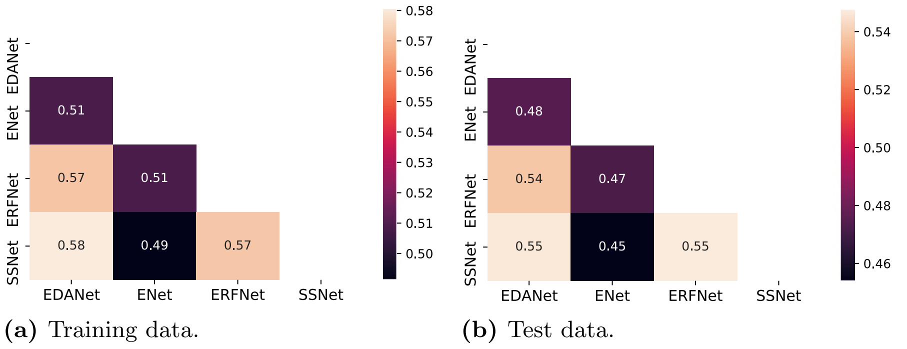


Fig. 5 Correlation heat-maps between the four efficiency-oriented neural networks. The correlation is calculated as the MIoU between the predicted SS maps on the training and test data (left and right sub-figures, respectively)

Table 1 Enumeration of S-GP's hyper-parameters

S-GP's variant	Parameter	Value
MC & SC	Training batch size	150
	Runs	10
	Function set	$\{FS_{basic}, FS_{CV}\}$
MC	Terminals	{ENet, ERFNet, EDANet, SSNet}
	(Population x generations)	(285 x 40)
	Initialization	RHH
	Selection	Tournament with pool size of 5%
	Crossover	Swap, P(C)=0.7
	Mutation	Sub-tree, P(M)=1-P(C)=0.3
SC	Terminals	{ENetx19, ERFNetx 19, EDANetx19, SSNetx19}
	(Neighborhood x generations)	{(60x10), (40x15)}
	Initialization	Grow
	Neighborhood function	Sub-tree
	Control	1.0
	Update-rate	0.95

5.4 S-GP's hyper-parameters

Table 1 provides a complete enumeration of the experimental parameters for the proposed S-GP system. The column S-GP's variant divides the table's rows across the two S-GP variants: the multi-class (MC) and the single-class (SC); whenever the parameters across the two variants overlap, we use the notation "MC & SC". The following paragraphs provide a detailed discussion of the selected parameters and values.

To train the system, we relied on batch processing. Specifically, we used a batch size of 150 images to evaluate the population/neighborhood at the end of each generation. Preliminary experiments showed that smaller batch sizes tend to deteriorate the system's performance. Individuals' fitness was computed as the mean intersection over union (MIoU) between the predicted segmentation maps and the respective target annotations. Considering the algorithms' stochastic nature, we repeated our experiments ten times (runs), each with a different seed for the pseudo-random numbers generator to initialize and execute the search. We guaranteed an equal computational effort for each variant of the S-GP system, measured in terms of fitness evaluations per run - 11685. In this sense, we decided to execute the MC variant of the S-GP system for 40 generations with a population size of 285 candidate solutions. The SC variant was studied in two configurations: (i) 15 generations and neighborhood size of 40 candidate solutions, and (ii) 10 generations and neighborhood size of 60 candidate solutions; in this sense, we wanted to assess the trade-off between the exploitation and the exploration of the proposed LS-based approach.

To explore the search space of possible stacking models in the MC variant, we used standard GP with sub-tree mutation and sub-tree crossover with probabilities of 30% and 70%, respectively; the initial population was generated by employing the Ramped Half-and-Half (RHH) initialization technique with a maximum depth of 5 levels; the selection was a tournament with a tournament pool size of 5%. In the SC variant, we used the GP's sub-tree mutation as the neighborhood function and grow initialization technique with a maximum depth of 5 levels; the initialization was repeated 15 times for each of the 19 trees to match the number of RHH's fitness evaluations; in this sense, the best tree on the training data was selected for constructing the initial candidate solution. The LS-based approach for the SC variant was explored in two forms: (i) with standard hill climbing (HC) and (ii) with simulated annealing (SA); the control and the update-rate parameters for the latter were set to 1.0 and 0.95, respectively. In particular, the control parameter corresponds to the temperature parameter of SA, while the update-rate parameter governs the cooling strategy responsible for decreasing the temperature. Note that both S-GP variants were artificially seeded base-learners' genotypes in the initial population/neighborhood. This was performed to allow the search algorithms to start at a *good* point in the search space and foster their convergence. In particular, the base learners were provided in the form of a single-node tree for the MC variant of the proposed S-GP method. For the SC variant, given the fact it corresponds to a list of stacking models for each target class, the initial seed consists of a list made of 19 single-node trees. It is worth noting that we preferred to rely on the standard GP search process due to its simplicity, to maintain the computational effort manageable, and to provide compact solutions. For instance, we could have considered geometric semantic GP [41] as the stacking method, but this would have produced solutions whose size and complexity would not be acceptable for the underlying requirements of the application under analysis.

The terminal set for the MC variant of the S-GP system was built from the four SS neural networks' output logits; in this sense, a given terminal can be seen as a tensor of shape $bs \times 19 \times 256 \times 256$, where bs stands for the batch size, 19 represents the number of target classes and 256×256 represents the spatial dimension of

Table 2 Detailed list of the two considered function sets

Functions	FS_{basic}	FS_{CV}
Traditional (+, -, *, /)	✓	✓
Trigonometric (sine, cosine)	✓	✓
Pixelwise (minimum, maximum, average)	✓	✓
Convolutional edge (Sobel vertical/horizontal)		✓
Convolutional filter (Laplacian, Gaussian)		✓
Pooling (average, maximum)		✓

the networks' output; in total, there are four distinct terminals. The terminal set for the SC's variant was also built from the SS neural networks' output logits, but with a difference. Since the SC's variant discriminates between networks' output feature maps channels, the feature maps were flattened across the channel dimension and detached; therefore, a given terminal in the SC variant can be seen as a tensor with sizes $bs \times 256 \times 256$, and there are $4 \times 19 = 76$ distinct terminals.

Besides the traditional operators (such as {+, -, *, /}), we also used pixel-wise minimum, maximum, mean, and some of the most popular convolution and pooling operators from the field of image processing and computer vision. Specifically, as for the convolution operators, we used the vertical and the horizontal Sobel kernels, the discrete approximations of the Laplacian and the Gaussian filters; as for the pooling operators, we used average and maximum pooling with stride and padding set to one; for both convolution and pooling operators, we used a 3×3 sized window/kernel. We ran the experiments with (function set FS_{CV}) and without (function set FS_{basic}) the computer-vision-specific operators following Table 2, in order to verify their utility for the underlying problem. Moreover, we also considered the maximum, the minimum, and the average operators, applied pixel-wise between the two input terminals. Finally, we added the sine and the cosine functions to the functions' set to foster non-linearities' modeling.

5.5 Other ensemble methods

To ensure the viability of the proposed approach, S-GP will be compared against two traditional ensemble methods such as simple (i.e., unweighted average) and voting, alongside more complex weighted average ensembles. To implement the latter, we relied upon GAs. The purpose of this section is to provide a brief description of these ensemble methods and their parameters.

To implement the voting ensemble, we generated the predicted segmentation masks for each of the considered SS networks and applied a voting scheme classifier as described in [66]. The simple (i.e., unweighted average) average ensemble, instead, was implemented at the logits' level. That is, first, we stack the four SS networks' non-normalized per-pixel pseudo-probability distributions over the classes, which yields a tensor with sizes $bs \times 4 \times 19 \times 256 \times 256$; then, we apply the average across the second dimension of this tensor, which yields a tensor with sizes $bs \times 19 \times 256 \times 256$; finally, this tensor is converted into the output segmentation map by passing it through the softmax activation layer and taking the index of the maximum value on the second dimension (representing, at this stage, the likelihood of a given pixel belonging to some target class).

Two variants of GAs were implemented for the weighted average ensemble, to establish compatibility with each of the proposed S-GP methods and to allow for a fair comparison. The first variant, named MC-GA, follows the concept of the proposed MC variant of S-GP. Specifically, candidate solutions with four constrained parameters are evolved through GA to implement a weighted average as a linear combination of the form $w_0 \times X_0 + w_1 \times X_1 + w_2 \times X_2 + w_3 \times X_3$, where w_i represents a given parameter and X_i the corresponding logits' tensor of the SS network i. To evaluate candidate solutions, a similar procedure is taken to that described for the simple average ensemble with the exception that logits are weighted. To establish compatibility with the MC variant of S-GP, the GA was used with the same parameters as GP when possible (see Table 1 for a complete overview of the parameters). In particular, the same selection method, mutation and crossover probabilities, number of generations, population size, and training batch size were used. The second variant, named SC-GA, corresponds to the proposed SC variant of S-GP. Given that S-GP aims at evolving a separate stacking ensemble model for each class, the SC-GA variant follows the same approach by simultaneously evolving a combination of 19 class-specific weighted-average ensembles. Each class-specific ensemble is a weighted average of the four logits, obtained from the SS networks, that were sliced on the underlying class. This approach is repeated for all target classes, which yields candidate solutions with $4 \times 19 = 76$ parameters. Once the weights are applied, the procedure to evaluate the fitness is equivalent to the aforementioned MC-GA variant. Similarly to the aforementioned MC-GA, the SC-GA variant was used with the same parameters as GP.

Several other ensemble methods were proposed [50]. The fundamental reason why these were not included in this work is directly related to the specifics of the machine learning task addressed in this work (semantic segmentation) and therefore the representation of one training instance - in our task, it is a multidimensional tensor and not a vector, as it happens in traditional machine learning applications. For example, in the case of the MC variant of S-GP, one training instance is a tensor with shape $[bs, C, H, W]$, and the produced output by the stacking model must be equal in size. This is clearly different from D-dimensional vectors that one can usually find in traditional machine learning applications where other SOTA ensemble methods exhibit preeminent results. On the other hand, evolutionary-based approaches, such as GA and GP, are more flexible to handle more complex data representations.

6 Results and discussion

This section presents the experimental results and discusses the main findings. It is divided into four sub-sections, following the experimental objectives:

1. design a stacking model by means of GP that efficiently leverages the synergistic effect of efficient neural architectures' combination, and attains levels of precision superior to any of the networks in isolation;
2. assess networks' relative importance (worth);
3. obtain a set of *small* and *human-interpretable* models that simultaneously achieve high levels of precision on the underlying CV task;
4. assess systems' time complexity.

6.1 Performance

Figure 6 presents the performance of the MC variant of S-GP and compares it against the corresponding MC GA-based stacking, the simple (i.e., unweighted) average ensemble, and the best base learner. Note that the figure intentionally excludes the voting ensemble as it reports significantly lower accuracy when compared to the simple average. Specifically, the averages calculated across the elite individuals observed at every generation of each run are reported for GP and GA. The proposed S-GP is presented in two configurations that differ in terms of the feature set (visit Sect. 5.4 for the details related to the feature sets): MS-FS_{basic} in blue and MC-FS_{CV} in red. The corresponding MC-GA weighted average is presented in green. The simple (i.e., unweighted) average ensemble and the best base learner (SSNet [36]) are depicted as black and grey straight lines, respectively. In a top-down manner, the figure depicts: (i) the training fitness, calculated as MIoU on batches of the training set, (ii) the test fitness, (iii) and the trees' length calculated as the number of program elements.

From the elites' aggregated training fitness (the sub-figure at the top), we can see that S-GP with the two feature sets seems to exhibit a comparable performance until generation 16/17. From that moment on, MC-FS_{CV} stagnates while MC-FS_{basic} continues to exhibit fitness improvement (blue and red lines, respectively), although at a lower rate of change. When compared with MC-GA, both S-GP variants exhibit notably higher training fitness. Although the average line of the MC-GA approach exceeds that of SSNet (in black), it also shows a substantial amount of instability during the training.

Since systems' generalization ability is of great concern, we focus our attention on the elites' aggregated test fitness (the sub-figure in the middle). To begin with, it is paramount to mention that both configurations of MC-GP and MC-GA outperform the best base learner regardless of the feature set, and the differences are statistically significant after Wilcoxon's signed-rank test for related paired samples with a significance level of 5%. Regarding the difference between the two feature sets of the proposed S-GP approach: MC-FS_{basic} (in blue) generalizes slightly better than its analogue with CV operators, however, the difference is not statistically significant. Moreover, similarly to what was observed with the training fitness, the test fitness of the two feature sets stagnates after generation 16/17, making this more noticeable for the MC-FS_{CV} configuration (in red). Similarly to what was observed from the curves on the training data, the average test fitness of the MC-GA approach is

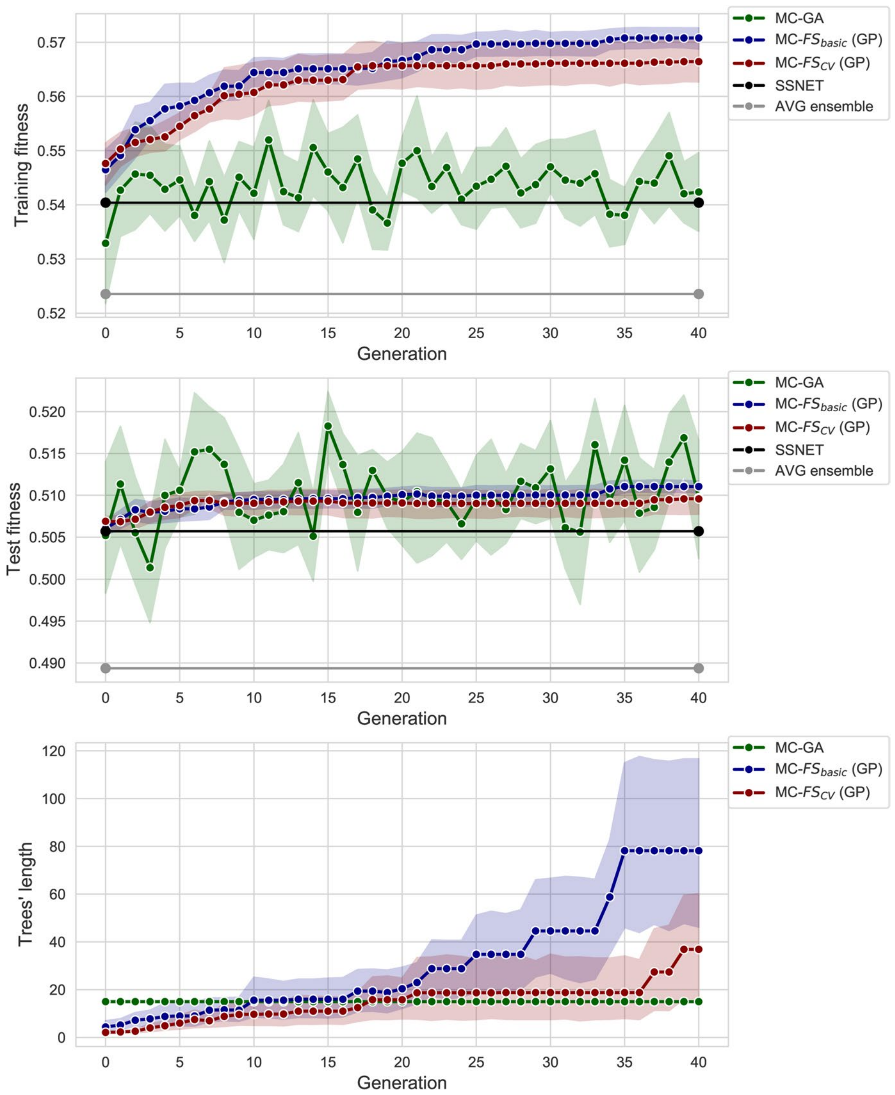


Fig. 6 Learning curves for the MC variant of the S-GP system (MC-FS_{basic} in blue and MC-FS_{CV} in red), and the corresponding MC-GA ensemble (MC-GA in green). The figure also includes the simple (i.e., unweighted) average ensemble and the best base learner in grey and black, respectively (Color figure online)

notably less stable when compared to S-GP, although a light-increasing trend can be observed. The differences between MC-GA and the two proposed MC-GP variants in terms of the test fitness were not found to be statistically significant after Wilcoxon's signed-rank test for related paired samples with a significance level of 5%.

The simple average ensemble (AVG), depicted as a grey line in both training and test fitness plots, is shown to be the least-performing model, which can be explained by the fact that the best base learner (SSNet) is substantially better than the other three models, that exhibit comparable performance. In this sense, giving equal weight to all the models deteriorates the performance of the ensemble with respect to the SSNet.

Another pivotal aspect to take into account is the solutions' complexity. This becomes particularly important when solutions' interpretability is required (which is our case). In this sense, it is useful to analyze the length of the elite trees obtained through S-GP (the sub-figure at the bottom). From the figure, we can see that, at the end of 40 generations, MC-FS_{basic} (in blue) tends to produce twice as long trees as MC-FS_{CV} (in red); interestingly, the difference starts to increase significantly after the 20th generation - a few generations after the aforementioned stagnation point. The MC-GA approach (in green) is shown as a straight line given the fact the candidate solutions in GAs are of fixed size. We fix the length of the solutions obtained by MC-GA to 15, as it corresponds to the length of a binary tree encoding the weighted average as a linear combination of the form $w_0 \times X_0 + w_1 \times X_1 + w_2 \times X_2 + w_3 \times X_3$, where X_i represents the output of a specific neural model.

From the perspective of trees' length, MC-GA clearly provides the simplest solution. In light of the generalization ability, however, the MC-GA does not appear as the most prominent approach given its relative under-performance and notable instability (both across generations and runs). Instead, the proposed MC-FS_{basic} method appears to be the most accurate and stable across runs and generations.

Figure 7 presents the SC variant of S-GP and compares it against the corresponding GA weighted ensemble, the simple (i.e., unweighted) average ensemble, and the best base learner. Note that the figure intentionally excludes the voting ensemble as it reports significantly lower accuracy when compared to the simple average. Similarly to Fig. 6, the lines represent the average fitness and length values calculated across the elite individuals observed at every generation of each run, reported for GP and GA. Given the abundance of considered configurations in the SC variant of S-GP, we divide the figure into two columns, each regarding a different feature set: SC-FS_{basic} on the left and SC-FS_{CV} on the right. In a top-down manner, the figure is divided into three sub-figures (following the configuration of Fig. 6): (i) the training fitness calculated as MIoU on batches of the training set, (ii) the test fitness, (iii) and the trees' length calculated as the number of program elements. The four colored lines represent the performance of the two different LS approaches studied in two configurations of neighborhood size *versus* generations. Notice that each configuration requires an equal number of fitness evaluations by run ($40 \times 15 = 60 \times 10 = 600$). The same number of fitness evaluations was used for the SC-GA method. Specifically, the figure presents:

- *In sky blue* HC with a neighborhood size of 40 executed for 15 generations (HC_40×15)
- *In dark blue* HC with a neighborhood size of 60 executed for ten generations (HC_60×10)
- *In dark red* SA with a neighborhood size of 40 executed for 15 generations (HC_40×15)

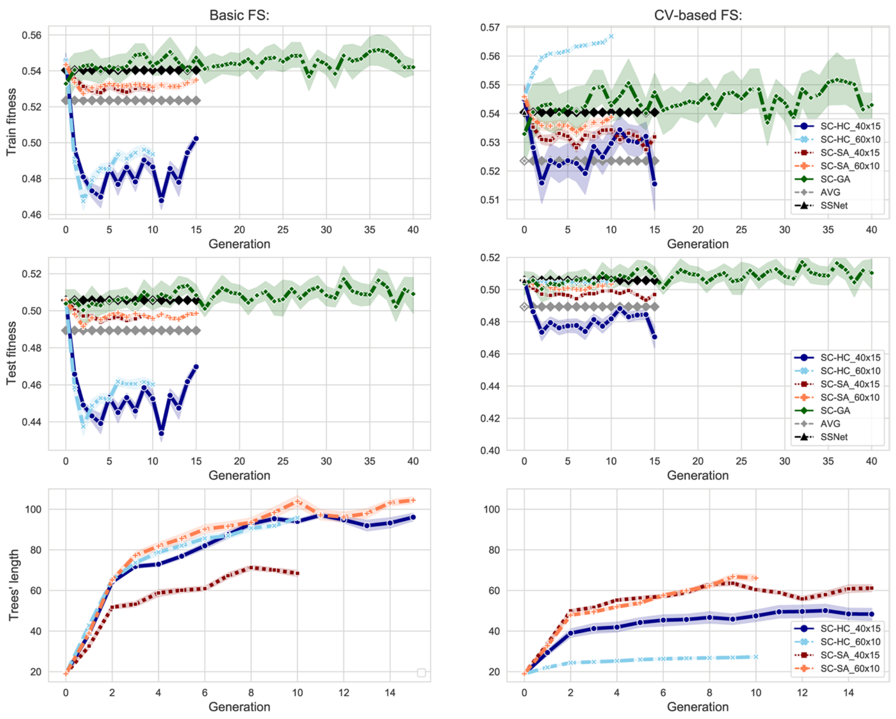


Fig. 7 Learning curves for the SC variant of the S-GP system with different configurations and the corresponding GA-weighted ensemble (SC-GA, in green). The figure also includes the simple (i.e., unweighted ensemble) average ensemble and the best base learner in grey and black, respectively. The first column regards the so-called basic feature set, whereas the second column regards the feature set including CV-based operators (Color figure online)

- *In coral* SA with a neighborhood size of 60 executed for ten generations (HC_60×10).
- *In green* the GA-weighted ensemble where each of the 19 channels is a weighted ensemble of the four SS networks
- *In grey* the simple (i.e., unweighted) average ensemble of the four SS networks' logits
- *In black* the best base learner (SSNet [36])

From the proposed SC S-GP configurations only HC_60×10 using the FS_{CV} feature set was able to learn from the training data. However, in terms of generalization ability, none could report results statistically superior to the best base learner (SSNet [36]), except SC-GA. The differences between SC-GA and the proposed SC-GP variants in terms of the test fitness were found to be statistically significant after Wilcoxon's signed-rank test for related paired samples with a significance level of 5%, except for the HC_40×15 variant with CV-based operators.

In fact, the behavior of SC-GA is similar to what has been observed in Fig. 6 with MC-GA: negligible superiority over the best base learner and notably high

Table 3 Summary table comprising all the methods (algorithms) involved in this study

Approach	Algorithm	*Test fitness*	*Train fitness*	*Length*
CNN for SS	EDANet	0.4497	0.4849	1
	ENet	0.3885	0.4250	1
	ERFNet	0.4557	0.4880	1
	SSNet	0.5057	0.5404	1
Simple ensemble	Average	0.4894	0.5235	9
	Voting	0.4676	0.5030	5
GA ensemble	MC-GA	**0.5099**	0.5446	15
	SC-GA	**0.5107**	0.5450	285
GP ensemble	MC-GP-basic	**0.5111**	0.5708	78.2
	MC-GP-CV	**0.5096**	0.5664	36.9
	SC-HC-basic-40x15	0.4683	0.5000	95.5
	SC-HC-basic-60x10	0.4564	0.4916	98.2
	SC-SA-basic-40x15	0.4978	0.5293	102.2
	SC-SA-basic-60x10	0.4961	0.5316	70.8
	SC-HC-CV-40x15	0.4998	0.4816	50
	SC-HC-CV-60x10	0.5045	0.5671	27.4
	SC-SA-CV-40x15	0.4979	0.5389	59.3
	SC-SA-CV-60x10	0.5032	0.5388	67

variability between the generations and across the runs. Concerning the solutions' complexity, the HC_60×10 configuration was the one producing the smallest trees (in average terms). The length of the SA-GA approach is not present in the figure because it is disproportionally large when compared to the proposed S-GP variant: assuming a binary tree representation, the length of the SC-GA method is $15 \times 19 = 285$.

A closer inspection of the results showed that the HC_60×10 configuration often converged to the best learner with slight modifications at some of the target classes, for example, applying a max pooling operator over the S-GP model regarding the background class. The presented results refute our assumption about the benefits of the joint evolution of tailored S-GP for each class. We argue that this happens for the following two reasons. First, the usage of a single stacking model across all target classes, as in the MC variant, introduces an implicit form of regularization for the S-GP system; contrarily and, for us unexpectedly, fine-tuning a stacking model for each target class introduces too many "degrees of freedom" and, instead of fostering S-GP's performance, hinders its potential from finding a good solution. Second, we speculate that the larger amount of "degrees of freedom" introduced by the SC variant also demands more computational resources (like, for instance, the neighborhood's size and the number of generations). Given the noticeable under-performance of the SC variant, in addition to its excessive computational load (see Sect. 6.4), it was decided to leave this research track, and we focus instead on the MC variant only.

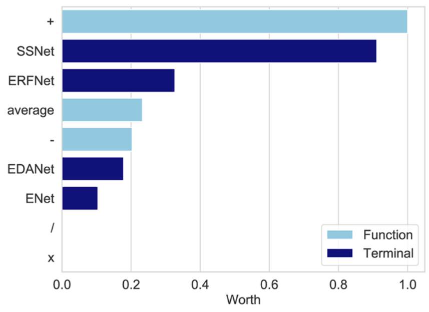
Table 3 provides a summary of performance across the proposed MC and SC variants of S-GP, the GA-based counterparts, traditional ensemble methods, and the base learners. Specifically, the table contains the average test (*Test fitness*) and training (*Train fitness*) fitness values, calculated for each algorithm (identified by the column *Algorithm*). The column *Approach* serves as a macro category to group different algorithms. The values for the proposed S-GP ensemble and the corresponding GA-weighted were obtained by averaging the results obtained from the elite candidate solutions at the end of the runs. The column *Length* represents the average solutions' length proposed by different algorithms. To provide a fair comparison in terms of the solutions' length, we use a binary tree encoding for the two GA-weighted average ensemble methods. Specifically, in MC-GA the tree length corresponds to the length of a binary tree encoding the weighted average as a linear combination of the form $w_0 \times X_0 + w_1 \times X_1 + w_2 \times X_2 + w_3 \times X_3$; in SC-GA the length of the binary trees encoding the weighted average as an equivalent linear combination for each of 19 channels is, therefore, $19 \times 15 = 385$. The voting ensemble, however, can be seen as an exception as it consists of the four SS networks and a single voting operator with arity four (which yields a length of five). A statistical analysis was carried out to assess the differences in terms of the methods' generalization ability. In particular, we tested whether the differences in terms of average test fitness between the best base learner (SSNet) and different methods were significant after Wilcoxon's signed-rank test for related paired samples with a significance level of 5%; the algorithms reporting statistically better results are reported in bold.

From the summary table, it becomes clear that the only approaches that were able to achieve a statistically significant improvement over the baseline were the proposed MC variant of S-GP with the basic and CV-based feature sets and the corresponding GA-based ensembles. The largest accuracy was observed for the MC variant of S-GP with the basic feature set but at the cost of generating deeper trees. Section 6.3, however, shows that manual simplification of the trees generated through MC-GP-basic can produce small and interpretable models.

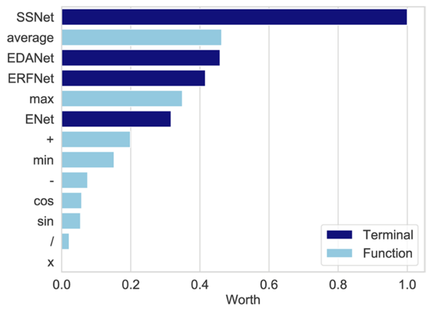
6.2 Worth analysis

Figure 8 shows the estimated worth of program elements for the MC and SC variants of the S-GP system, with function sets FS_{basic} and FS_{CV}. Following the common practice in GP [61], the worth was estimated as the program elements' frequency of the best individuals' genotype (the elites), observed at the end of the last generation of every run. In this sense, the figure reports the frequency counts of program elements from the 10 runs. The most frequent program elements are also assumed to be the most relevant. The program elements were ranked in descending order according to their estimated worth and colored based on whether they belonged to the functions' or terminal set (brighter and darker tones, respectively).

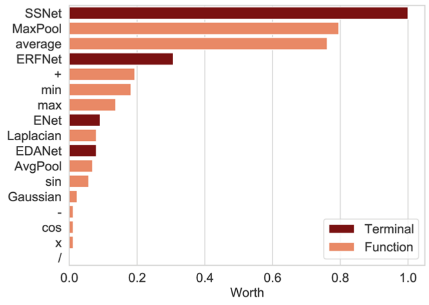
When looking at the terminals' worth (these correspond to the darker tones of the figure's bars), one can observe that within each variant of S-GP, the worth of terminals follows the same order, regardless of the feature set. Independently on the variant,



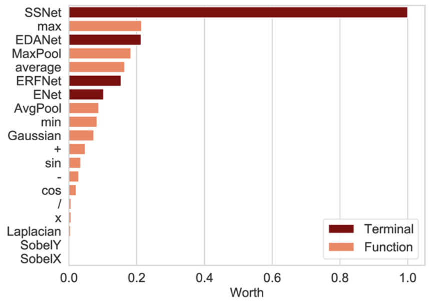
(a) Program elements' worth for the MC-FS_{basic} variant of S-GP



(b) Program elements' worth for the SC-FS_{basic} variant of S-GP



(c) Program elements' worth for the MC-FS_{CV} variant of S-GP.

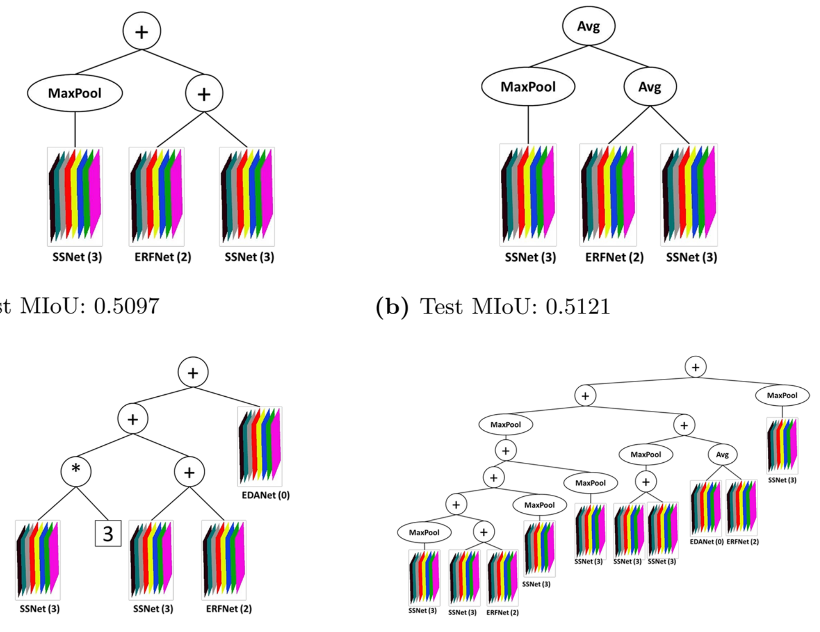


(d) Program elements' worth for the SC-FS_{CV} variant of S-GP.

Fig. 8 Estimate of program elements' worth via frequency plots. The figure contains both multi and single-class variants of the proposed S-GP approach explored using two types of operators (basic and CV-based). The evolution for MC with FS_{basic} in **a** automatically excluded trigonometric operators sine and cosine, as well as pixel-wise minimum and maximum. The evolution for MC with FS_{CV} in **c** excluded convolutional-edge filters based on the Sobel operator

SSNet is shown to be by far the most relevant efficiency-oriented SS network in the S-GP ensembles. When considering the MC variant of S-GP - the most prominent stacking method - the ERFNet is reported as the second most relevant terminal. Such an order was expected since SSNet's encoder is essentially a slight improvement over the ERFNet's. Moreover, SSNet uses a more sophisticated decoder which was shown to improve the segmentation maps along objects' boundaries efficiently. Surprisingly, EDANEt's worth is estimated to be roughly the same as ENet's: in MC-FS_{basic}, EDANet's worth is slightly above ENet's, whereas in MC-FS_{CV}, the opposite happens. In our opinion, this happens because, under the perspective of the accuracy-speed trade-off that reigns the neural architectures' design, these two networks headed more towards speed at the cost of accuracy. In a nutshell, besides obtaining network worth estimates, one can conclude that these estimates happen to be robust with regard to the empirical findings from the literature.

When looking at the functions' worth (these correspond to lighter tones of the figure, one can observe that pixel-wise average, addition, maximum, and minimum



(a) Test MIoU: 0.5097 (b) Test MIoU: 0.5121

(c) Test MIoU: 0.5115 (d) Test MIoU: 0.5122

Fig. 9 Candidate stacking models evolved utilizing S-GP (multi-class variant)

happen to be the most relevant across operators regardless of the variant and feature set. The subtraction, division, multiplication, and trigonometric functions are notably the least relevant functions. This fact suggests the small utility of these functions for the underlying problem instance.

When looking at the coral bars of Fig. 8c and d both regard FS_{CV} configurations for the MC and SC variants of S-GP), one can observe that the MaxPool is the most frequently used CV-based operator. Other operators can be found, but at a significantly lower rate, which suggests their reduced utility in the scope of the underlying problem instance: the AvgPool, the Laplacian filter, the AvgPool, and the Gaussian kernel.

6.3 Inspection of stacking models

Figure 9a, b, c, and d present four example candidate solutions of the multi-class stacking variant, which achieved a fair performance-complexity trade-off. In fact, their compact representation allows a closer inspection of the underlying trees.

It is fundamental to remind the reader that our stacking procedure was applied before the SoftMax operation (i.e., our generated solutions directly process the logit data). This experimental choice, whose superiority over post-SoftMax processing was determined via preliminary experiments, has the advantage of operating on the raw network outputs before any non-linear compression of the data range that might hinder the ability to capture nuances in the predictions or even lose information due to numerical precision issues. Furthermore, in this context, the data can be re-purposed and combined without the need to preserve their characteristics as probability distributions since those will be enforced as a post-processing SoftMax step if necessary.

One potential drawback of working with logits is that there is no inherent control over the original data distribution since different networks (or even different training runs of the same network) will produce outputs in arbitrary ranges. This factor is possibly the reason for the behavior shown in Fig. 9c, where the SSNet terminal is multiplied by a factor of 3 before combination with other branches. Notice that this tree has been mathematically simplified for ease of interpretation and that the underlying operation was a sequence of sums involving the same network multiple times.

An alternative explanation for increasing the values of one network output before the further combination is boosting its importance relative to other networks, as the optimization procedure implicitly learned that it is more reliable and thus should be given greater relevance in a consensus-based decision. This hypothesis is corroborated by the fact that SSNet specifically has the highest accuracy among our set of base learners. The actual consensus can then be implemented by combining multiple branches through a pixel-wise sum operation (or, equivalently, an average) as displayed in solutions Fig. 9a and b. The final class decision for each pixel is, in fact, determined by the ArgMax operator across all class values. The relative order of these values can be overturned if enough branches agree on a particular class distribution, simply by summing (or averaging) their outputs.

Another often selected function, as also highlighted by the worth analysis, is the MaxPool operator. This operation is hypothesized to work as a spatial denoising operator, as it effectively removes isolated low-level single-pixel values. The origin of these sparse elements can be potentially attributed to artifacts of the dilated convolutions and the fast upsampling modules, implemented in many efficient networks for semantic segmentation. The preference for max pooling also supports this denoising interpretation as opposed to average pooling, which would instead have the effect of spreading the impact of such sparse elements.

Finally, Fig. 9d shows a solution that combines all the aforementioned elements: reinforcing a single network by summing it with itself, conducting a consensus process via either sums or averages, and widespread use of max pooling, potentially as a denoising operator.

By taking into consideration the solutions' performance in terms of MIoU on test data, as well as the level of complexity of the corresponding stacking trees, the candidate in Fig. 9b can be considered as an ideal compromise. This solution effectively exploits the SSNet and ERFNet models from the pool of investigated base learners, running respectively at 113.1FPS and 61.0FPS on a TitanX (Pascal) Graphic Processing Unit (GPU), as reported by [38]. Their stacking will therefore operate on up to 39FPS, minus the necessary computation of pooling and averaging operations, effectively matching the commonly-adopted criterion of 30FPS for real-time performance [37, 38, 52]. Figure 10 shows an example of output segmentation maps produced by the stacking model of Fig. 9b.

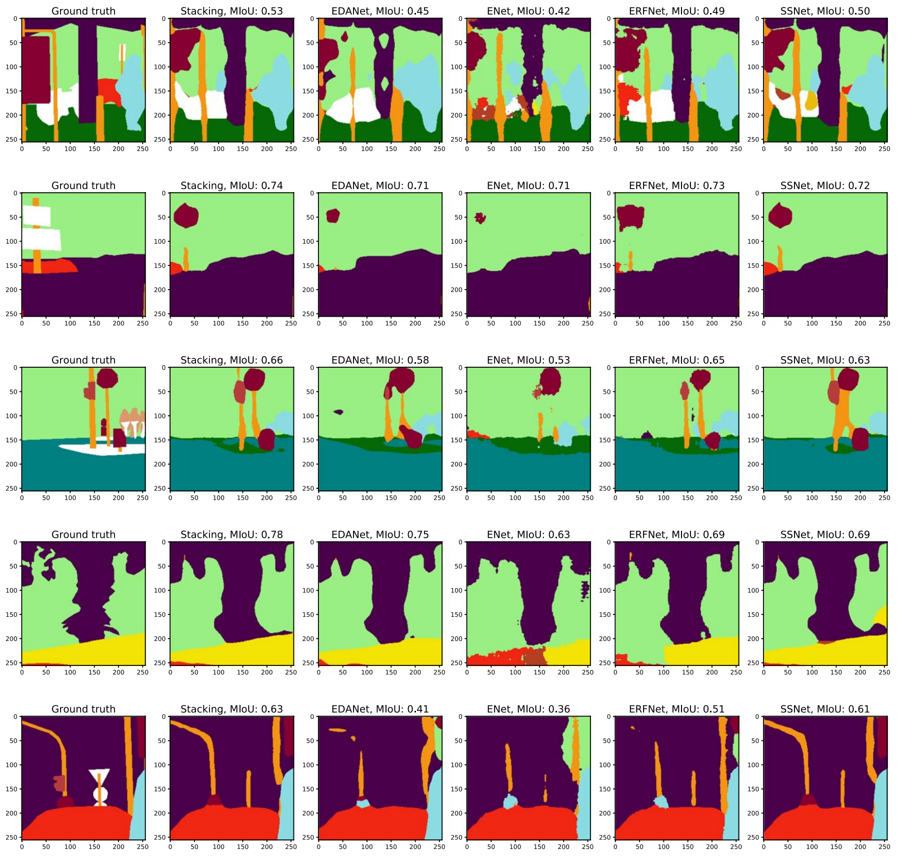


Fig. 10 Example of output segmentation maps produced by the stacking model in Fig. 9b

6.4 Time complexity

Figure 11 presents experiments' processing times calculated as the average number of hours to execute one run. We can see that the MC variants happen to be by far the most time-efficient experiments - a fact that is directly related to the solutions' simplicity of representation and, consequently, processing-time usage. As for the functions' set, one can see that S-GP with CV-based operators generally requires slightly larger processing times. This outcome happens because the CV-based operators (like, for example, the convolution or the pooling) are applied between the underlying kernel and a kernel-sized patch of the input image, for every pixel in the padded input image.

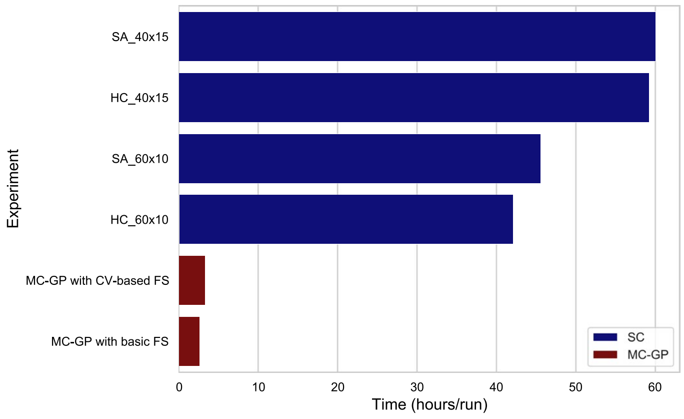


Fig. 11 Experiments' processing times in hours per run

7 Conclusion

This paper presents a study of genetic programming (GP) in the context of stacked generalization for semantic segmentation (SS) neural architectures. More specifically, we explored GP's role as the meta-learning algorithm that combines preliminary outputs of four different efficiency-oriented semantic segmentation architectures for fast recognition of urban scenes, such as ENet, ERFNet, EDANet, and SSNet in an evolutionary fashion. The contribution of this work is three-fold. First, we generated stacking models able to overcome the performance of each of the state-of-the-art neural architectures considered in this study (measured as mean intersection over union calculated on the test set). Second, we were able to assess networks' relative importance in the context of the stacking model. Third, we evolved stacking models that are of small size and human-interpretable.

Further research in the field is still in demand, and we consider deepening the cross-fertilization between the fields of evolutionary computation and ensemble learning. For example, stacking multiple training instances of the underlying neural models, or breaking the search-space into disjoint sub-spaces that would be better handled by carefully evolved stacking models focused on a given sub-space. Another valuable research direction could involve other sources of information when generating stacking models like, for example, the depth or attention maps.

Author Contributions IB: Data curation, Formal Analysis, Investigation, Methodology, Software, Validation, Visualization, Writing - original draft, Writing - review & editing. MB: Conceptualization, Formal Analysis, Investigation, Methodology, Project administration, Software, Supervision, Validation, Visualization, Writing - original draft, Writing - review & editing. RS: Conceptualization, Methodology, Resources, Supervision, Writing - review & editing. MCi: Funding acquisition, Methodology, Supervision, Writing - review & editing. LV: Funding acquisition, Methodology, Supervision, Writing - review & editing.

Funding Open access funding provided by FCT|FCCN (b-on). This work was supported by national funds through the FCT (Fundação para a Ciência e a Tecnologia) by the projects GADgET (DSAIPA/

DS/0022/2018), AICE (DSAIPA/DS/0113/2019), UIDB/04152/2020 - Centro de Investigação em Gestão de Informação (MagIC)/NOVA IMS, and by the grant SFRH/BD/137277/2018.

Declarations

Conflict interests No financial or non-financial conflicts of interests.

Ethical approval No human or animal participants were involved in this study.

References

1. D. Agnelli, A. Bollini, L. Lombardi, Image classification: an evolutionary approach. Pattern Recognit. Lett. **23**(1), 303–309 (2002). https://doi.org/10.1016/S0167-8655(01)00128-3
2. H. Al-Sahaf, A. Song, K. Neshatian, M. Zhang, Two-tier genetic programming: towards raw pixel-based image classification. Expert Syst. Appl. **39**(16), 12291–12301 (2012). https://doi.org/10.1016/j.eswa.2012.02.123
3. V. Badrinarayanan, A. Kendall, R. Cipolla, Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **39**(12), 2481–2495 (2017). https://doi.org/10.1109/TPAMI.2016.2644615
4. Bakurov, I., Buzzelli, M., Castelli, M., Schettini, R., Vanneschi, L.: Genetic programming for structural similarity design at multiple spatial scales. in Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '22, p. 911-919. Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3512290.3528783
5. I. Bakurov, M. Buzzelli, R. Schettini, M. Castelli, L. Vanneschi, Full-reference image quality expression via genetic programming. IEEE Trans. Image Process. **32**, 1458–1473 (2023). https://doi.org/10.1109/TIP.2023.3244662
6. I. Bakurov, M. Castelli, O. Gau, F. Fontanella, L. Vanneschi, Genetic programming for stacked generalization. Swarm Evolut. Comput. **65**, 100913 (2021). https://doi.org/10.1016/j.swevo.2021.100913
7. Bakurov, I., Vanneschi, L., Castelli, M., Fontanella, F.: Edda-v2–an improvement of the evolutionary demes despeciation algorithm. in International Conference on Parallel Problem Solving from Nature, pp. 185–196. Springer (2018)
8. S. Bianco, M. Buzzelli, G. Ciocca, R. Schettini, Neural architecture search for image saliency fusion. Inform. Fusion **57**, 89–101 (2020)
9. S. Bianco, G. Ciocca, R. Schettini, Combination of video change detection algorithms by genetic programming. IEEE Trans. Evol. Comput. **21**(6), 914–928 (2017)
10. L. Breiman, Stacked regressions. Mach. Learn. **24**, 49–64 (1996). https://doi.org/10.1007/BF00117832
11. Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., Varoquaux, G.: API design for machine learning software: experiences from the scikit-learn project. in ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pp. 108–122 (2013)
12. Bukhtoyarov, V., Semenkina, O.: Comprehensive evolutionary approach for neural network ensemble automatic design. pp. 1–6 (2010). https://doi.org/10.1109/CEC.2010.5586516

13. L.C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Trans. Pattern Anal. Mach. Intell. **40**(4), 834–848 (2018). https://doi.org/10.1109/tpami.2017.2699184
14. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 3213–3223 (2016)
15. J. Correia, N. Rodriguez-Fernandez, L. Vieira, J. Romero, P. Machado, Towards automatic image enhancement with genetic programming and machine learning. Appl. Sci. (2022). https://doi.org/10.3390/app12042212
16. J..a Correia, D. Lopes, L. Vieira, N. Rodriguez-Fernandez, A. Carballal, J. Romero, P. Machado, Experiments in evolutionary image enhancement with elaine. Genet. Progr. Evolvable Mach. **23**(4), 557–579 (2022). https://doi.org/10.1007/s10710-022-09445-9
17. Girshick, R.B., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. in 2014 IEEE Conference on Computer Vision and Pattern Recognition pp. 580–587 (2014)
18. Gonçalves, I., Silva, S., Fonseca, C.M., Castelli, M.: Unsure when to stop?: Ask your semantic neighbors. in Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '17, pp. 929–936. ACM, New York, NY, USA (2017). https://doi.org/10.1145/3071178.3071328
19. S. Hao, Y. Zhou, Y. Guo, A brief survey on semantic segmentation with deep learning. Neurocomputing **406**, 302–321 (2020). https://doi.org/10.1016/j.neucom.2019.11.118
20. Harris, C., Buxton, B.F.: Evolving edge detectors with genetic programming (1996). in Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University. The MIT Press. https://doi.org/10.7551/mitpress/3242.003.0044
21. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. in 2015 IEEE International Conference on Computer Vision (ICCV) pp. 1026–1034 (2015)
22. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 770–778 (2016)
23. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR (2017). arXiv: org/abs/1704.04861
24. Huang, G., Liu, Z., Weinberger, K.Q.: Densely connected convolutional networks. in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 2261–2269 (2017)
25. Jaderberg, M., Vedaldi, A., Zisserman, A.: Speeding up convolutional neural networks with low rank expansions. ArXiv arXiv: abs/1405.3866 (2014)
26. Jin, J., Dundar, A., Culurciello, E.: Flattened convolutional neural networks for feedforward acceleration. CoRR arXiv: abs/1412.5474 (2015)
27. Johansson, U., Löfström, T., König, R., Niklasson, L.: Building neural network ensembles using genetic programming. in The 2006 IEEE International Joint Conference on Neural Network Proceedings, pp. 1260 – 1265 (2006). https://doi.org/10.1109/IJCNN.2006.246836
28. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv: 1412.6980 (2014)
29. J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (MIT Press, London, 1992)
30. A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks. Commun. ACM **60**, 84–90 (2012)
31. La Cava, W., Spector, L., Danai, K.: Epsilon-lexicase selection for regression. in Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO '16, p. 741-748. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2908812.2908898
32. Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I.V., Lempitsky, V.S.: Speeding-up convolutional neural networks using fine-tuned cp-decomposition. in Y. Bengio, Y. LeCun (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015). arXiv: org/abs/1412.6553
33. Leonardi, M., Mazzini, D., Schettini, R.: Training efficient semantic segmentation cnns on multiple datasets. in International Conference on Image Analysis and Processing, pp. 303–314. Springer (2019)
34. Lo, S.Y., Hang, H., Chan, S., Lin, J.J.: Efficient dense modules of asymmetric convolution for real-time semantic segmentation. in Proceedings of the ACM Multimedia Asia (2019)

35. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3431–3440 (2015). https://doi.org/10.1109/CVPR.2015.7298965
36. Mazzini, D.: Guided upsampling network for real-time semantic segmentation. in: British Machine Vision Conference 2018, BMVC 2018, Newcastle, UK, Sept 3-6, 2018, p. 117. BMVA Press (2018)
37. Mazzini, D., Buzzelli, M., Pau, D.P., Schettini, R.: A cnn architecture for efficient semantic segmentation of street scenes. in: 2018 IEEE 8th International Conference on Consumer Electronics-Berlin (ICCE-Berlin), pp. 1–6. IEEE (2018)
38. Mazzini, D., Schettini, R.: Spatial sampling network for fast scene understanding. in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) pp. 1286–1296 (2019)
39. Mehta, S., Rastegari, M., Caspi, A., Shapiro, L., Hajishirzi, H.: Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. ArXiv arXiv: abs/1803.06815 (2018)
40. S. Minaee, Y.Y. Boykov, F. Porikli, A.J. Plaza, N. Kehtarnavaz, D. Terzopoulos, Image segmentation using deep learning: A survey. IEEE Trans. Pattern Anal. Mach. Intell. (2021). https://doi.org/10.1109/TPAMI.2021.3059968
41. Moraglio, A., Krawiec, K., Johnson, C.G.: Geometric semantic genetic programming. in: International Conference on Parallel Problem Solving from Nature, pp. 21–31. Springer (2012)
42. D. Muni, N. Pal, J. Das, A novel approach to design classifiers using genetic programming. IEEE Trans. Evol. Comput. **8**(2), 183–196 (2004). https://doi.org/10.1109/TEVC.2004.825567
43. Paszke, A., Chaurasia, A., Kim, S., Culurciello, E.: Enet: A deep neural network architecture for real-time semantic segmentation. ArXiv arXiv: abs/1606.02147 (2016)
44. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
45. R. Poli, Genetic programming for feature detection and image segmentation, in *Evolutionary Computing*. ed. by T.C. Fogarty (Springer, Heidelberg, 1996), pp.110–125
46. Poli, R., Langdon, W.B., McPhee, N.F.: A field guide to genetic programming. Published via and freely available at http://www.gp-field-guide.org.uk (2008). (With contributions by J. R. Koza) http://lulu.com
47. R. Polikar, Ensemble based systems in decision making. Circuits Syst. Mag. IEEE **6**, 21–45 (2006). https://doi.org/10.1109/MCAS.2006.1688199
48. S.S.M. Rahman, T. Islam, M.I. Jabiullah, Phishstack: Evaluation of stacked generalization in phishing urls detection. Procedia Comput. Sci. **167**, 2410–2418 (2020). https://doi.org/10.1016/j.procs.2020.03.294
49. S. Reid, G. Grudic, Regularized linear models in stacked generalization, in *Multiple Classifier Systems*. ed. by J.A. Benediktsson, J. Kittler, F. Roli (Springer, Heidelberg, 2009), pp.112–121
50. Y. Ren, L. Zhang, P. Suganthan, Ensemble classification and regression-recent developments, applications and future directions [review article]. IEEE Comput. Intell. Mag. **11**(1), 41–53 (2016). https://doi.org/10.1109/MCI.2015.2471235
51. M.E. Roberts, E. Claridge, An artificially evolved vision system for segmenting skin lesion images, in *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2003*. ed. by R.E. Ellis, T.M. Peters (Springer, Heidelberg, 2003), pp.655–662
52. E. Romera, J. Álvarez, L.M. Bergasa, R. Arroyo, Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. IEEE Trans. Intell. Transp. Syst. **19**, 263–272 (2018)
53. O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015*. ed. by N. Navab, J. Hornegger, W.M. Wells, A.F. Frangi (Springer, Cham, 2015), pp.234–241
54. N. Sharma, M. Mangla, S.N. Mohanty, C.R. Pattanaik, Employing stacked ensemble approach for time series forecasting. Int. J. Inf. Technol. **13**, 2075–2080 (2021). https://doi.org/10.1007/s41870-021-00765-0
55. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2015)
56. Singh, T., Kharma, N., Daoud, M., Ward, R.: Genetic programming based image segmentation with applications to biomedical object detection. in: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, GECCO '09, p. 1123-1130. Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1569901.1570052
57. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9 (2015). https://doi.org/10.1109/CVPR.2015.7298594

58. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 2818–2826 (2016)
59. Tackett, W.A.: Genetic programming for feature discovery and image discrimination. in: Proceedings of the 5th International Conference on Genetic Algorithms, p. 303-311. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
60. Vanneschi, L., Bakurov, I., Castelli, M.: An initialization technique for geometric semantic gp based on demes evolution and despeciation. in 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 113–120 (2017). https://doi.org/10.1109/CEC.2017.7969303
61. L. Vanneschi, R. Poli, *Genetic Programming - Introduction, Applications Theory and Open Issues.* (Springer, Heidelberg, 2012)
62. D.H. Wolpert, Stacked generalization. Neural Netw. **5**, 241–259 (1992)
63. Yu, F., Koltun, V., Funkhouser, T.: Dilated residual networks. in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 636–644 (2017). https://doi.org/10.1109/CVPR.2017.75
64. A. Zameer, J. Arshad, A. Khan, M.A.Z. Raja, Intelligent and robust prediction of short term wind power using genetic programming based ensemble of neural networks. Energy Conv. Manag. **134**, 361–372 (2017). https://doi.org/10.1016/j.enconman.2016.12.032
65. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6230–6239 (2017). https://doi.org/10.1109/CVPR.2017.660
66. Zhou, Z.H.: Ensemble Methods: Foundations and Algorithms, (2012). https://doi.org/10.1201/b12207
67. Zhou, Z.H., Wu, J.X., Jiang, Y., Chen, S.F.: Genetic algorithm based selective neural network ensemble. in Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'01, p. 797-802. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)

Authors and Affiliations

Illya Bakurov[1,2,3] · **Marco Buzzelli**[4] · **Raimondo Schettini**[4] · **Mauro Castelli**[1] · **Leonardo Vanneschi**[1]

✉ Illya Bakurov
ibakurov@novaims.unl.pt; bakurov1@msu.edu

Marco Buzzelli
marco.buzzelli@unimib.it

Raimondo Schettini
raimondo.schettini@unimib.it

Mauro Castelli
mcastelli@novaims.unl.pt

Leonardo Vanneschi
lvanneschi@novaims.unl.pt

[1] Information Management School, Universidade Nova de Lisboa, Campus de Campolide, 1070-312 Lisboa, Lisbon, Portugal

[2] BEACON Center of Evolution in Action, Michigan State University, East Lansing, MI, USA

[3] Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA

[4] Department of Informatics, Systems and Communication, University of Milano – Bicocca, Viale Sarca 336, 20126 Milan, Italy