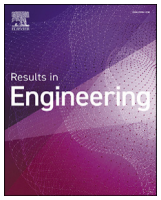
Results in Engineering 18 (2023) 101107

Contents lists available at ScienceDirect

Results in Engineering

journal homepage: www.sciencedirect.com/journal/results-in-engineering



Research paper

Automatic image captioning combining natural language processing and deep neural networks

Antonio M. Rinaldi *, Cristiano Russo, Cristian Tommasino

Department of Electrical Engineering and Information Technologies, IKNOS-LAB Intelligent and Knowledge Systems (LUPT), University of Naples Federico II, 80125 Via Claudio, 21, Napoli, Italy

ARTICLE INFO

Keywords:
Object detection
Image captioning
Deep neural networks
Semantic-instance segmentation

ABSTRACT

An image contains a lot of information that humans can detect in a very short time. Image captioning aims to detect this information by describing the image content through image and text processing techniques. One of the peculiarities of the proposed approach is the combination of multiple networks to catch as many distinct features as possible from a semantic point of view. In this work, our goal is to prove that a combination strategy of existing methods can efficiently improve the performance in the object detection tasks concerning the performance achieved by each tested individually. This approach involves using different deep neural networks that perform two levels of hierarchical object detection in an image. The results are combined and used by a captioning module that generates image captions through natural language processing techniques. Several experimental results are reported and discussed to show the effectiveness of our framework. The combination strategy has also improved, showing a gain in precision over single models.

1. Introduction

The image captioning topic has recently received great attention in the computer vision community. The goal of image captioning is to describe analyzed image content. This issue is interesting for its important practical applications and because it is a great challenge for computer vision to understand image contents.

Computer Vision (CV) and Natural Language Processing (NLP) were considered and treated as separate research areas in the past. Nowadays, with the huge amount of available data in multiple formats produced from several sources, such as mobile devices, IoT, and multimedia social networks [19], researchers can apply both techniques in conjunction to better understand multimedia contents and to extract knowledge more than information from a given visual content. Indeed, a fundamental task in image retrieval and computer vision is the automatic annotation of images, whose goal is to find proper, hence meaningful, words or phrases for images. The more such annotations are semantically rich, the more such mapping between the annotated text and the visual content is consistent. In this context, approaches based on artificial intelligence, which often exploits pre-trained models, are used to learn the mapping between low-level and semantic features and then generate annotations or captions for a given image. A common valid approach includes classification that assigns semantic classes to an image given in input visual features extracted from the image.

Generating a meaningful image content description in a natural language requires a high level of recognized object understanding. An important task in this context is recognizing the detected objects' semantic meanings. For these reasons, classification approaches have been used to assign recognized objects to semantic classes based on their visual features. The use of natural language processing techniques attempts to give a possible solution to this problem by trying to reduce the semantic gap [2,4,30,32] defined in [34] as "the lack of coincidence between the information that can be extracted from visual data and the interpretation that these same data have for a user in a given situation".

There are two general paradigms in image captioning: top-down and bottom-up. The top-down paradigm starts from the information given by an image and converts it into words; on the other hand, the bottom-up approach considers words that describe various aspects of an image and combines them to generate a sentence. In both approaches, linguistic models are used to generate coherent sentences. Recent top-down approaches follow an encoder-decoder framework to generate image captions. Convolutional neural networks are developed to encode visual images, and recurrent neural networks to decode this information. This paper presents a framework based on a top-down approach with a set

* Corresponding author.
E-mail addresses: antoniomaria.rinaldi@unina.it (A.M. Rinaldi), cristiano.russo@unina.it (C. Russo), cristian.tommasino@unina.it (C. Tommasino).

https://doi.org/10.1016/j.rineng.2023.101107
Received 5 September 2022; Received in revised form 6 April 2023; Accepted 16 April 2023
Available online 19 April 2023

of convolutional neural networks combined to encode the input images and a decoder of the detected information that generates a description of image contents. A distinctive feature of the proposed approach is the combination strategy. In our study, we consider multiple neural network models to exploit the best traits of each model in the final captioning task. This is a common and well-established technique used in machine learning under different conditions and use cases. The remainder of this paper is organized as follows. In Section 2 an overview of the related research is presented; Section 3 describes the architecture of the implemented system for image captioning together with our approach; In Section 4, we report several experimental results and, eventually, Section 5 concludes the paper and proposes future directions of our works.

2. Related works

A large number of methods and techniques based on different approaches have been applied to image captioning during years [14,35] and, the improvement of computer vision techniques [33,38] has led to the design and implementation of more efficient image captioning systems [9].

The current state of the art in image captioning can be divided into three categories: (i) top-down, (ii) bottom-up, and (iii) mixed approaches.

The bottom-up approach can be viewed as a classic methodology for image captioning in which the process starts with combining visual concepts, objects, attributes, words, and phrases into sentences using linguistic models. A model called Baby Talk is proposed in [13]. This name derives from the similarity with children's language used to generate image descriptions. It is based on detection modules used to detect objects. A set of classifiers processes each candidate object, and a Conditional Random Field (CRF) is built with graph labeling to generate captions. Fang et al. [7] proposed an architecture based on an image dataset and related descriptions. The system uses a weakly supervised learning approach to create detectors of common words used in the descriptions of the analyzed image. After this step, sentences are generated from this set of words and classified according to their similarity to image contents.

Top-down approaches define image captioning as a machine language translation problem. Instead of translating contents among different languages, these approaches translate visual information into natural language. The visual representation is derived from a CNN trained on large datasets; conversely, the caption is generated by an RNN that processes natural language. The differences between the various existing implementations often concern the type of RNN used. In [21], a multimodal RNN to generate image captions is used. It models a probability distribution to generate a previous word related to an image, and the captions are generated according to this distribution. The model consists of two subnetworks: a deep recurrent neural network for sentences and a deep convolutional neural network for images. In addition, the model is also applied for retrieval tasks able to find images or sentences. Vinyals et al. [37] present a model that uses RNN in a neural and probabilistic framework to generate image descriptions. Fixed-size, The description probability is modeled with an RNN, where the variable number of words is expressed as a fixed length memory updated after a new input using a non-linear function generated by a Long-Short Term Memory (LSTM) network. This type of model uses RNNs to encode the variable input length into a fixed-size vector, and it uses this representation to decode the desired output sentence.

In recent years, there has been a transition to mixed approaches that involve selecting and merging results generated by bottom-up and top-down approaches. An approach based on a semantic attention model is proposed in [40]. The first task analyzes the responses from a classification step by a CNN to build a global visual description. Later, attribute detectors are run to have a list of attributes or visual concepts that appear more often in the image. Each attribute corresponds to an entry in the system dictionary. All visual features are sent to an RNN for caption generation. Karpathy et al. [12] describe a multimodal Recurrent Neural Network architecture that uses the deep visual-Semantic alignments to learn to generate novel descriptions of image regions. They show their method can overcome state-of-the-art results. Anderson et al. [3] proposed an architecture that combines a bottom-up approach based on Faster R-CNN that recognizes image regions associated with a feature vector with a top-down approach to measuring the feature weights. The bottom-up approach returns a feature vector composed of the location of each object in an image together with the class to which the object belongs. This vector is weighed using two LSTMs to generate the image caption.

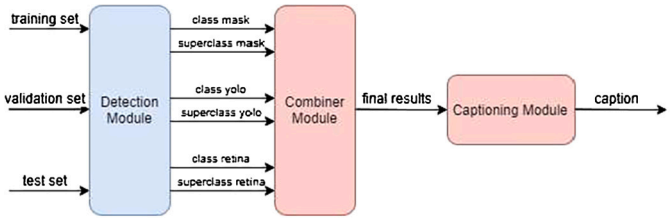


Fig. 1. System architecture.

A more recent approach based on the Transformer model is proposed in [42]. A Transformer is a deep learning model which allows using attention mechanisms without using RNNs. The authors use a knowledge graph to help the model generate captions and add a Kullback-Leibler (KL) divergence term to discriminate incorrect predictions. Another attention-based mechanism for image captioning is presented in [11]. Such a bio-inspired approach relies on the concept of divergent-convergent attention, combining visual and semantic information as a multi-perspective input to their model.

Likely in [23,39,15], the authors proposed to train a deep neural network by integrating textual and visual features.

In this work, we proposed a framework for image captioning based on an encoder step performed by combining three CNNs for image object detection of images. The single results of these CNNs are combined with having the best result without losing information due to the CNN characteristics. Moreover, a classification on two hierarchical levels is proposed to assign to the recognized objects a class or a "superclass" (a more generic concept from a semantic perspective) depending on the highest result scores. Finally, a decoder step is implemented, considering a combining strategy based on feature extraction to perform a natural language processing step for having a clear and exhaustive description of an image.

3. The implemented system and approach

The system architecture is shown in Fig. 1. From a general point of view, It is organized around three main modules: a detection, a combination, and a captioning module that generates the image caption. In this section, we describe the implemented module, the chosen DNN architectures, the combination strategy, and the techniques used for the image captioning generation.

3.1. Detection module

The detection module is based on three instance segmentation models: Mask R-CNN [8], YOLOv3 [26], and RetinaNet [16]. The choice of these models is due to their quasi-real-time performances as shown in [22,10], their relevance in several applications and subdomains such as point-based instance segmentation [25], single-pixel reconstruction [41], and ball detection [5]. In this work, we propose a two levels hierarchical classification. The instance segmentation is divided into regression for object bounding boxes and classification to recognize the object class. This strategy allows assigning a superclass for the recognized object if the model fails to determine a more specific subclass.

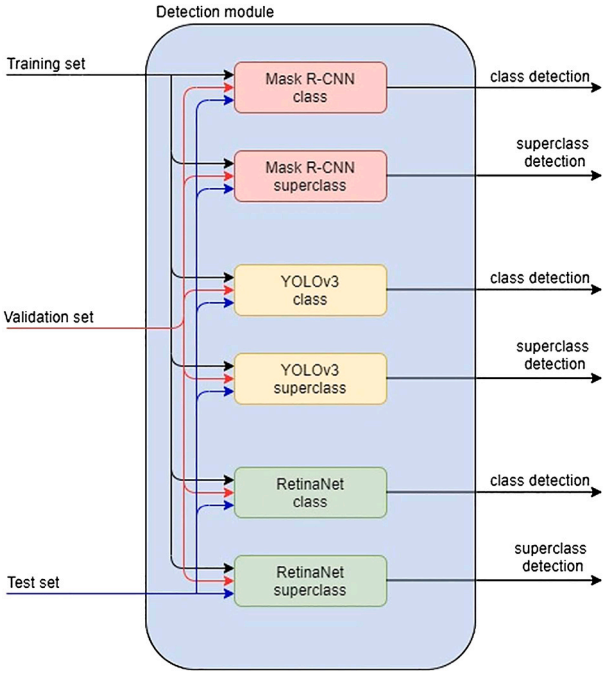


Fig. 2. Detection Module Architecture.

As seen in Fig. 2, each model is duplicated using one model for class detection and one for superclass detection. Each of these six modules receives in input a training set, a validation set, and a test set, and it returns as output a vector containing the features of the classified objects in the image test set.

Mask R-CNN is an extension of Faster R-CNN [27] for instance, segmentation problems. This DNN architecture adds a third branch that allows the mask prediction of an instance. This operation is performed in parallel with the bounding box regressor and the classifier in the Faster R-CNN. Specifically, Mask R-CNN generates a mask with size 28x28 for each region of interest (RoI) which will be later expanded to fit the corresponding bounding box size. For each identified instance, Mask R-CNN will return as output the recognized class, its bounding box, and a binary mask superimposed on it.

The presence of masks adds to the loss function of each extracted RoI an additional term as shown in Equation (1):

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{box} + \mathcal{L}_{mask} \tag{1}$$

Where $\mathcal{L}_{cls}$ and $\mathcal{L}_{box}$ are equal to the loss functions related to the bounding box classifier and regressor used in Faster R-CNN and Fast R-CNN. For $\mathcal{L}_{mask}$ only one ground truth mask is associated to each RoI and a sigmoid activation function is to be applied to each pixel of the mask. The branch associated with the generated mask prediction produces a binary mask with $m \times m$ size for each of the possible K classes. We ave $K \times m^2$ possible masks, each of them associated with a different class. $\mathcal{L}_{mask}$ is defined as the average between the binary cross-entropy loss functions which includes the k-th mask if the region is associated with the k-th ground truth mask as shown in Equation (2):

$$\mathcal{L}_{mask} = -\frac{1}{m^2} \sum_{1 \le i,j \le m} [y_{ij} log \hat{y}_{ij}^k + (1 - y_{ij}) log(1 - \hat{y}_{ij}^k)] \tag{2}$$

The second convolutional neural network used in this work is YOLO (acronym of You Only Look Once), in its latest version. This network is based on the Darknet53 image classifier, consisting of 53 convolutional layers, allowing the network to achieve good performance in a short time [26].

YOLO defines the bounding boxes using dimensional clusters like anchor boxes. It predicts t_x, t_y, t_w, t_h information for each bounding represented as x and y coordinates at the beginning of the box and its dimensions expressed as width and height. If the cell starts from the top left corner of the image (c_x, c_y) and the previous bounding box has width and height p_w, p_h, then the prediction is:

$b_x = \sigma(t_x) + c_x$
$b_y = \sigma(t_y) + c_y$
$b_w = p_w e^{t_w}$
$b_h = p_h e^{t_h}$

The width and height are predicted as offsets from the centroids of the cluster, while the center coordinates relative to the allocation of the filter application are defined using a sigmoid function.

The sum squared error is the loss function used for the training as shown in Equation (3):

$$SSE = \sum\_i = 1^n (x_i - \bar{x})^2 \tag{3}$$

Where n is the number of observations, x_i is the value of the i-th observation, and $\bar{x}$ is the real value.

The ground truth of the predicted coordinates is $\hat{t} *$, and the gradient is the value of the ground truth minus the prediction: $\hat{t} * - t *$. The value of the ground truth can be easily calculated by reversing the previous equations. YOLOv3 predicts a score for each bounding box using logistic regression. This score is one if the bounding box overlaps a ground truth object more than any other previous bounding box. If the considered bounding box is not the best but overlaps a ground truth object for no more than a threshold, the prediction is ignored. The threshold usually used is 0.5. YOLO assigns a bounding box for each ground truth object. Each box predicts the classes that the bounding box might contain using a multilabel classification. For this purpose, independent logistic classifiers are used.

The third used architecture is RetinaNet. The RetinaNet architecture is composed of three parts: a backbone, a feature pyramid network (FPN), and a back-end detection. An image is processed by the backbone. Usually, ResNet and, together with the FPN, a network similar to an encoder-decoder is implemented. The benefit of this architecture is that the features of consecutive layers are merged from the rawest to the finest layer with an effective feature propagation to different layers at different scales to the next layer. After this step, the multi-scale pyramid features move data to the back end, and it switches on two branches for the bounding box regression and object classification. In these last two branches, there is no weight sharing; the weights of each branch are shared through the pyramid features. The anchors used have an area ranging from 32^2 to 512^2 on the pyramid levels P3 to P7 respectively and 3 proportions $\{1:2, 1:1, 2:1\}$ are used. For a denser scale coverage, anchors of magnitude $\{2^0, 2^{\frac{1}{3}}, 2^{\frac{2}{3}}\}$ are added to each level of the pyramid having nine anchors for each level. For each anchor, there is a one-dimensional vector of length K (number of classes) of ranking targets and a 4-dimensional vector of box regression targets. Anchors are assigned to the ground-truth object boxes with an Intersection over union (IoU) threshold of 0.5 and to the background if the IoU is in the range [0, 0.4). Each anchor is assigned to a maximum of one object and sets the corresponding entry of the class to 1 and the other entries to 0 in the length vector K. If the anchor has an IoU in the range [0.4, 0.5), it is not assigned and ignored during the training. The regression box is calculated as the offset between the anchor and the assigned object or omitted if there is no assignment.

The classification subnet predicts the probability of the presence of an object in each spatial position for each of the A anchors and each K object class. The used network is a Fully Convolutional Network (FCN) that applies 4 3x3 convolutional layers. Each layer has C filters and is followed by a ReLU activation and a convolutional layer with $K \times A$ filters. (K classes, A = 9 anchors, and C = 256 filters). The subnet used for the box regression is an FCN for each of the pyramid levels to decrease the offset from each anchor box to the nearest ground-truth object.

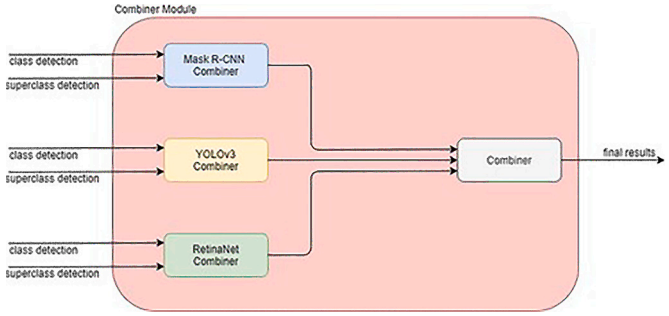


Fig. 3. Architecture of the combiner module.

3.2. Combiner module

This module combines the results obtained from the models in the detection module.

As we can see in Fig. 3, the first three nodes perform a combination of the results generated by each detection model to determine the object class or superclass. Later, the three results determined by these combinations are the input of a fourth combiner. The combiner first performs the combination between the results of Mask R-CNN and YOLOv3. The results of this combination are then compared with the results of RetinaNet, and the output vector containing the final results is provided, which then includes the combination of the results of all three models on two hierarchical levels of classes and superclasses.

The combiner output is a single feature vector that will be used to generate the caption of the analyzed image. The elements of the feature vector are:

- Image Id
- Bounding boxes of identified objects
- Class assigned to the object
- Probability of belonging to the class (i.e., score)

The combination step is performed by comparing all the elements of one vector with all the elements of the other one looking for the intersections between the bounding boxes. If one or more intersections are found, the maximum is chosen, and if it exceeds a threshold of 0.8 (i.e., overlap), we evaluate which of the two elements generating this intersection has the highest score of assignment to a predicted class, and we chose it as the best candidate. The elements of each vector that does not have a maximum intersection value will also be added to the final vector.

The metric we use for the evaluation is the well-established Intersection over Union, shown in Equation (4):

$$IoU = \frac{A \cap B}{A \cup B} \tag{4}$$

Let's clarify with an example of the combiner job. We suppose to have a vector with two elements, A and B, and the second vector with only one element C, with the following IoUs:

$IoU_{A,C} = 0.95$

$IoU_{B,C} = 0.45$

It means that the elements A and C refer to the same object in the image, having a very high IoU. We proceeded to compare the score of the assignment to the predicted class. Assuming that:

$score_A = 0.85$

$score_C = 0.95$

our combiner will choose C.

Since element B has an IoU lower than the IoU between the other two elements and lower than the threshold, it is not considered superimposed to the object defined in element C. However, it is added to the final vector because it identifies another object in the image. In this way, no information is lost during the different combination steps.

3.3. Captioning module

Image captioning is the process of generating textual descriptions of images using artificial intelligence techniques. The goal is to create a system that can accurately understand the contents of an image and produce a natural language description that captures the most important aspects of the image. The image captioning process typically involves a deep learning model that uses a combination of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to analyze the visual features of an image and generate a corresponding caption. The CNNs are used to extract relevant features from the image, while the RNNs are used to generate a sequence of words that form a coherent caption. Image captioning has a wide range of applications, including helping visually impaired individuals understand the contents of images, assisting with image retrieval and organization, and enhancing the user experience on social media platforms and e-commerce sites. This module inputs the combined results and describes the recognized objects in natural language to generate a caption of the whole image. The caption has the following syntax: "*The image X contains list of objects on the left; list of object on the right*". *Such rigid structure represents a limitation in the caption generation. However, we point out that a smooth and dynamic generation of captions is not the main goal of this study, which is focused instead on evaluating the combination strategy over individual neural network models.* In the actual implementation, it uses the spatial feature of the bounding boxes in the feature vector from the combiner module to identify the object's position in the image and understand if they are on the right or left side of the image, dividing them into two subvectors. After this step, the module recognizes the class of the objects using the predicted class in the combined vector feature, and it counts the number of occurrences of the same class in each subvector. The object name is derived from the class id (see Section 4). If we have more occurrences of the same object assigned to the same class, its name is pluralized, and the verbs are accordingly conjugated.

4. Experimental results

In this section, we propose and discuss several experimental results obtained from various executions of the presented system to put into evidence the performance of the proposed framework for image captioning. The neural networks used for the image object segmentation have been trained by means of a transfer learning approach starting from already trained weights. In the training phase, an NVIDIA GeForce 980 4 Gb GPU and NVIDIA CUDA and CUDNNN tools [24] have been used with the Python PyCharm development environment. The analysis of various models obtained with the different configurations defined in the training phase has been validated in two steps: (i) Graphic Analysis: using the Tensorboard tool [20], the loss function trends have been analyzed. The aim is to have a loss function related to the training data decreasing as much as possible at any epoch and a validation loss without a rich growing to avoid overfitting; (ii) Evaluation Metrics: when reasonable values for the loss functions have been obtained, we proceed with the related metrics calculation. The used metrics are IoU, Average Precision, and Average Recall. In the following of this section, we will present the used data set and our strategy for tuning an efficient model to have a good capacity to learn correctly the instances in training set having a good capacity for generalization of the learned knowledge. The results obtained from our framework and some examples of image caption are presented at the end of the section.

4.1. Dataset description

The entire COCO dataset [18] has been used. This dataset consists of 118287 images for the training set, 5000 for the validation set, and 40671 for the test set. It contains 80 object classes divided into 12 superclasses, as shown in Table 1.

Table 1
COCO dataset classes.

Superclass	Classes
Person	Person
Vehicle	bicycle, car, motorcycle, airplane, bus, train, truck, boat
Outdoor	Traffic light, fire hydrant, stop sign, parking meter, bench
Animal	Bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe
Accessory	Backpack, umbrella, handbag, tie, suitcase
Sports	Frisbee, skis, snowboard, sports ball, kit, baseball bat, baseball glove, skateboard, surfboard, tennis racket
Kitchen	Bottle, wine glass, cup, fork, knife, spoon, bowl
Food	Banana, apple, sandwich, orange, broccoli, carrot, hotdog, pizza, donut, cake
Furniture	Chair, couch, potted plant, bed, dining table, toilet
Electronic	Tv, laptop, mouse, remote, keyboard, cell phone
Appliance	Microwave, oven, toaster, sink, refrigerator
Indoor	Book, clock, vase, scissors, teddybear, hair drier, toothbrush

Table 2
Statistics of COCO classes used in the training set.

Class	Number of Images
Person	64115
Chair	12774
Car	12251
Dining Table	11837

Table 3
Statistics of COCO classes used in the validation set.

Class	Number of Images
Person	2693
Chair	580
Car	535
Dining Table	501

For our purposes, we have chosen to analyze the four most populated classes of images in the dataset. The statistics about the used classes for the training set and the validation set are shown in Table 2 and 3, respectively.

4.2. Mask-RCNN

The implementation of Mask R-CNN used for the creation and evaluation of the model [1] is implemented in Python language, and it uses TensorFlow and Keras libraries. It has been re-engineered for integration into our segmentation problem. Mask R-CNN has numerous parameters whose variation has a considerable impact on the final result. We have manually tested several combinations of parameters for each model. In this paper, we show the results for the ones which gave us the best accuracy. These parameters are shown in Table 4.

The parameters listed above are related to the model configuration, together with the number of training epochs and the used data augmentation pipeline. From a general point of view, we will have a decrement in the loss functions related to the training steps when epochs increase. Moreover, it is also necessary that we haven't an increment of the loss function related to the validation set (validation loss) if the epochs will increase. If this increment occurs, the model will be in overfitting.

In our implementation, the Mask-RCNN basic parameters have been set at the following maximum values:

- NUMBER OF_EPOCHS = 40
- STEPS_PER_EPOCH = 100
- VALIDATION_STEPS = 80
- TRAIN_ROIS_PER_IMAGE = 80

Table 4
MASK-RCNN parameters.

Parameter	Description
BACKBONE	Resnet 101 or resnet 50, where 50 and 101 are the number of layers of the network
GPU_COUNT	Number of used GPUs
IMAGES_PER_GPU	Number of images to analyze on each GPU
BATCH_SIZE	Not modifiable, equal to the product of the two previous ones
DETECTION_MIN_CONFIDENCE	Minimum score threshold to consider a classification
IMAGE_MIN_DIM	Minimum size of the scaled image
IMAGE_SHAPE	Image shape in [W H D] format
LEARNING_RATE	Step size at each iteration
MAX_GT_INSTANCES	Maximum number of ground truth instances to use in an image
NAME	Configuration name
NUM_CLASSES	Number of classes to be classified
NUMBER OF_EPOCHS	Number of epochs
STEPS_PER_EPOCH	Steps for each epoch
TRAIN_ROIS_PER_IMAGE	RoIs trained for each image
VALIDATION_STEPS	Number of steps for validation
WEIGHT_DECAY	Term in the weight update rule that causes the weights to exponentially decay to zero

- IMAGE_SHAPE = [512 512 3].
- IMAGE_PER_GPU = 1 (hence BATCH_SIZE = 1)

Different values have been used for the tuning of other parameters. In particular:

- LEARNING_RATE: between 0.001 and 0.01 in steps of 0.001
- WEIGHT_DECAY: between 0.0001 and 0.01 in steps of 0.0001
- DETECTION: between 0.0 and 0.9 in steps of 0.1
- TRAINED_ROIS_PER_IMAGE: between 10 and 80 in step of 10
- STEPS: between 10 and 100 in steps of 10
- VALIDATION_STEPS: between 10 and 80 in steps of 10
- DETECTION_MAX_INSTANCES: between 10 and 100 in steps of 10
- MAX_GT_INSTANCES: between 10 and 100 in steps of 10
- Number of training images: 1000, 10000 and all images in the dataset.

Moreover, we have to consider which network layers to train: (i) Heads: weights related only to fully-connected layers; (ii) 3+, 4+, 5+: weights for the first 3/4/5 layers of ResNet; (iii) All: weights for all layers. In this regard, several hybrid configurations have been tested with different strategies as a learning rate variation every x steps, a variation of the involved layers in the weight training process, and a combination of the above two strategies. The following pipelines have been used for image augmentation:

- Horizontal rotation
- Vertical rotation
- One of the two previous random choices
- One of the two rotations chosen randomly followed by a 60, 120 or 180 degree rotation always chosen randomly
- One of the two rotations randomly chosen followed by a Gaussian or an average blur
- Linear contrast
- From 0 to 2 operations random chosen between one of the two rotations, one of the three rotations of 60, 120 or 180 degrees and one of the two blurs

In general, with the same parameters, the ResNet101 backbone shows a better performance in terms of loss than ResNet50, as shown in Fig. 4. We decided to use a ResNet101 backbone for further experiments on image captioning.

The best performance was obtained with the following parameters:

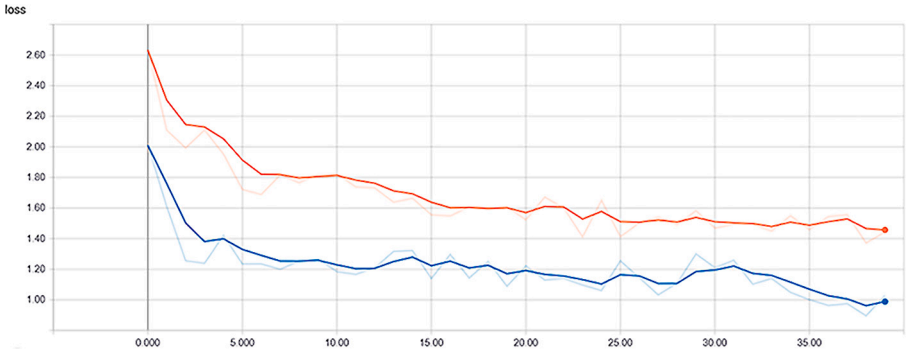


Fig. 4. Comparison between ResNet50 backbone (in red) and Resnet101 backbone (in blue).

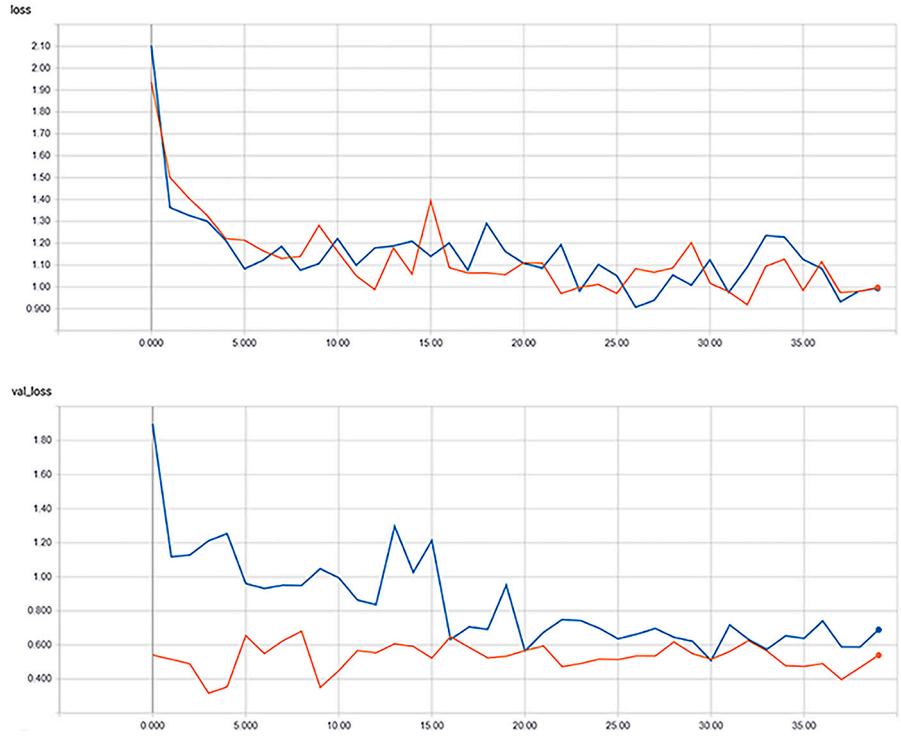


Fig. 5. Loss and Validation Loss Function for model 1 and 2.

- DETECTION MAX. 100
- DETECTION_MIN_CONFIDENCE 0.8
- LEARNING RATE 0.001
- MAX_GT_INSTANCES 100
- STEPS_FOR_EPOCH 100
- TRAIN FOR IMAGE 80
- VALIDATION_STEPS 50
- WEIGHT_DECAY 0.0001
- All images of the dataset
- Pipeline 1

We will show some examples of models resulting from different parameter settings. We analyze only the two best configurations to explain our evaluation process. This process has been used with all the other DNN models.

In Fig. 5 we show a comparison of model 1 and 2.

Model 1 is the one in blue, while model 2 is the one in red. We notice how they have a similar loss function at the end of the training epochs, while the validation loss in the case of model 1 decreases more quickly, and for model 2, it remains more or less constant for all the training epochs. The differences in configuration between these two models are (i) Epoch division: Model 1 has been trained for ten epochs on heads, ten on 4+, and 20 on all. Model 2 has been trained for five epochs on heads, five on 4+ and 30 on all; (ii) Learning rate: it is set to 0.0001 value after 20 epochs in model 1, it is set to 0.0001 after ten epochs for model 2. Due to the similarities in the loss trends, we calculate the reference metrics to choose the best model. The results are shown in Fig. 6. The best model is the number 1, and some examples of object segmentation using Mask-RCNN are in Fig. 7, where the three images shown are named “img16”, “img345” and “img1” respectively.

4.3. RetinaNet

The used RetinaNet implementation for the creation and evaluation of the model is the version present in [17]. It has been re-engineered to be integrated into our proposed instance segmentation pipeline.

This network has a simpler implementation compared with Mask-RCNN, and the tuning parameters for the training step are:

- Network Depth
- Learning Rate
- Number of epochs
- Batch size

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.2396099285619648
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.4291855028290235
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.23734280354674242
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.0714635481934158
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.2709373569995335
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.4454167692966118
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.18362969256519066
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.3194947396942907
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.32770891803288904
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.09815685104226254
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.3512924778147461
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.6007108171927852
```

(a) Model 1

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.2239329143542671
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.42126674067934544
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.21579821096805255
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.06610210888637648
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.2527725229045758
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.40827431788433244
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.17269793522449722
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.3054259869607087
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.3143172104355375
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.0968127207951655
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.33082079664629016
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.5759349172314538
```

(b) Model 2

Fig. 6. Evaluation metrics.

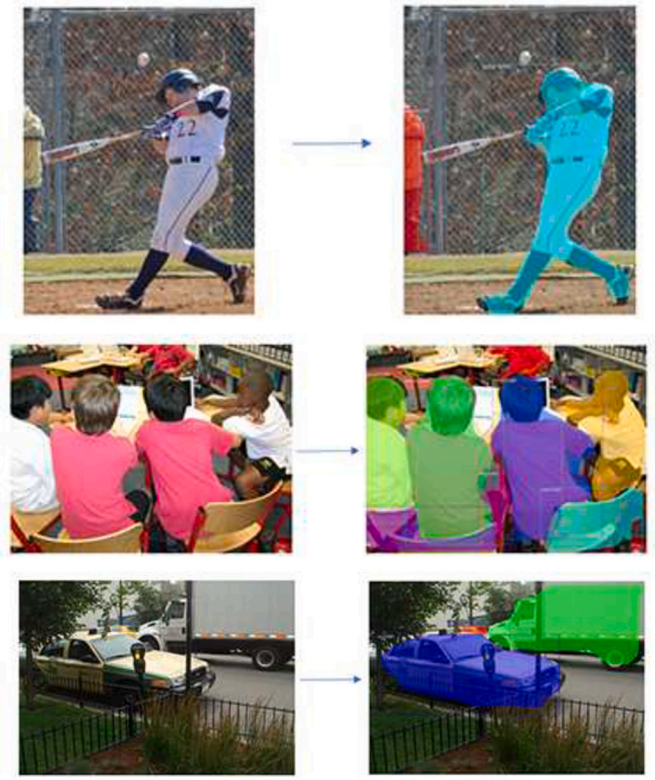


Fig. 7. Examples of object segmentation using Mask R-CNN.

ResNet101 has been chosen as the backbone due to the optimal trade-off between execution speed and mAP. The figure The experimental results obtained using different parameters values Fig. 8 shows the trend of the best model loss function trained with the following parameter values.

- 50 epochs
- Learning rate = 0.0001
- Batch size = 4

This configuration has been chosen by experimental results obtained with different parameter values, and the evaluation results are in Fig. 9.

Some examples of object segmentation are in Fig. 10.

In the first figure, the baseball player is detected as a person with a very high score and a very precise bounding box. In the second figure, only 4 of the children present and one of the three chairs in the foreground are detected, with variable scores and bounding boxes. In the third figure, the yellow car in the background is detected with a very high score (0.999) and a very accurate bounding box. On the other hand, the car in the foreground has a lower score with a partial bounding box.

4.4. YOLOv3

The implementation of YOLOv3 used for the creation and evaluation of the model is the version present in the repository [36] written in Python language, with Pytorch extension, and then reengineered the original project to the proposed instance segmentation problem.

The training parameters are similar to the ones listed in the previous models:

- Number of epochs
- Learning rate
- Learning momentum
- Weight decay
- Image size
- Multiscale (changing the image size during training)

The image augmentation has been performed with 3 operations:

- contrast variation
- saturation variation
- brightness variation

After several experiments, the best model for this network has been defined with the following configuration and its loss trend is shown in Fig. 11.

- 100 epochs
- Learning rate = 0.001
- Learning Momentum = 0.9
- Weight decay = 0.005

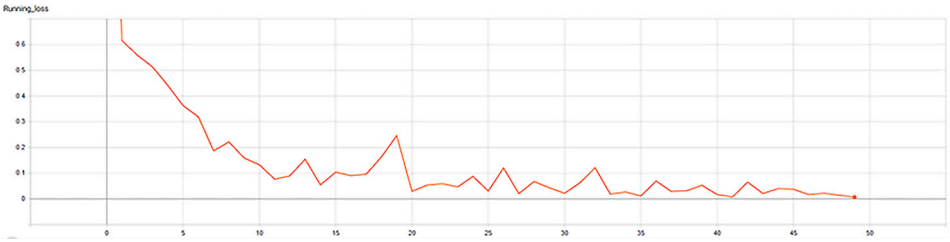


Fig. 8. Loss trend of used Retinanet model.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.206107281682046l7
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.3838564794850509
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.19474855082659062
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.07519024168462406
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.24162710516360S6
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.35029945020025804
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.1504817082774304
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.2771978683481594
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.29381586483858174
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.1210161253684731
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.3282138391068793
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.4703948455328598
```

Fig. 9. Model Evaluation.

Fig. 10. Examples of object detection using Retinanet.

- Image size = 512x512
- Multiscale not used

The evaluation results for the model are in Fig. 12.

Some examples of object segmentation are in Fig. 13.

We can notice that the model gives optimal performances on *person* class. In fact, in the first image, it detects the baseball player, while in the second image, are recognized four *persons* where 3 of them have a very high score. In the third image, the *car* hasn't a good detection both from the score value and bounding boxes.

4.5. Classes/superclass combiner

In some cases, the object's visual similarity in a first-class classification step could induce a misclassification because they represent different objects but in the same hierarchy. We try to solve this fine-grain classification error by implementing a strategy to combine the first results with the ones obtained from a direct superclass. For the detection of superclasses, the pre-trained weights used to perform the transfer learning have been used. In the following of this subsection, we show the results performed using this strategy for the three used

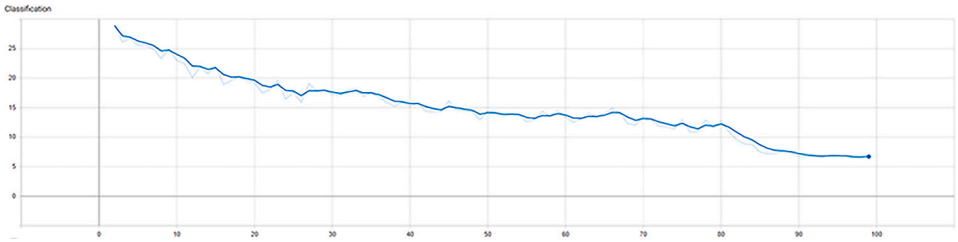


Fig. 11. Loss trend of used model for YOLOv3.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.24225536752290982
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.3872306450193302
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.21886503241671821
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.13472229871381819
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.23044727504688539
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.38280464752258555
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.23307365151558428
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.30848965910263942
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.311993506450405
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.14978792185745758
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.29231499757138534
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.51856368201787336
```

Fig. 12. Model evaluation.

Fig. 13. Examples of object detection using YOLOv3.

DNNs. The performance evaluation of the combined results obtained by the best-trained model on the classes and superclasses for Mask-RCNN is shown in Fig. 14.

As we can see, better results have been obtained in terms of mAP. Fig. 15 shows the improvement of MASK-RCNN classification using superclasses.

In fact, in addition to the two classified *people* in the first image, the *baseball bat* has been classified as “sports” superclass improving the overall detection. In the second image, instead, we notice that the *chairs*, previously classified as *dining table*, are now identified as *furniture*. In the first object classification of the third image, the *truck* has been recognized as *car* while it is now classified as a *vehicle* superclass with a higher score than the previous classification.

Fig. 16 shows the performance of the combined results obtained by the best-trained model on classes and superclasses for Retinanet.

Also in this case we obtained better results in terms of mAP. The improvements of Retinanet classification using superclasses are in Fig. 17.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.2299845214564768
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.3770683626404508
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.2450299911040622
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.05379272013331396
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.2668221261340843
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.46132700209019456
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.20543066254958037
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.27901015475573815
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.28236581104837494
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.06454831465419548
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.3316632842029505
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.5537499158694938
```

Fig. 14. Combined superclasses results with MASK-RCNN.

Fig. 15. Combined segmentation with MASK-RCNN.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.26730998877632123
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.4170233326250495
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.2814295586771677
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.14362295010209342
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.32851755347788597
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.4024341232241004
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.23809201000023206
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.3680240796118967
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.3978013924777029
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.2307991478433268
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.47303378335069196
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.5423296578483783
```

Fig. 16. Combined superclasses results with RetinaNet.

In the first image, in addition to the *baseball player* identified as *person*, the *ball* has been detected and classified as *sports*. In the second one, the detections of three other *persons*, a *book* on the right side identified as *indoor*, and the armchair on the top left side identified as *furniture* have been added to the previous detections. In the third image, the detection of the yellow *car* in the background has been kept, while the one in the foreground has been replaced by a detection of the *vehicle* superclass more generic than the first classification but with a higher score and a much more precise bounding box. In addition, the detection of the *truck* identified as a *vehicle* and the *parking meter*, identified as *outdoor* superclass, have been added. The YOLOv3 performance for the best model trained on classes and superclasses is in Fig. 18.

Also in this case we have obtained better results in terms of mAP. The classification results of this combination are shown in Fig. 19.

We observe that in the first image, another *person* is added to the detection results together with the *baseball* identified as “sports”. In the second image, two more *persons*, two chairs classified as *furniture*, the *notebook* classified as *electronic*, and the *book* classified as *indoor* have been added to the objects previously detected. In the third image, the detected *cars* have been replaced by the superclass *vehicle*.

Fig. 17. Combined segmentation with RetinaNet.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.29656512261829193
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.4184777545123143
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.32637518694639717
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.11639283853739593
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.348051450662698
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.4624143797010757
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.23677719011953055
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.3331850921809242
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.33740839147042656
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.13853710364012398
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.3914960693994586
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.512796062369978
```

Fig. 18. Combined Metrics of YOLO model.

4.6. Final results and captioning

The previous examples show that each model can have different precision in the object detection task. For this reason, instead of generating an image caption from the best result obtained by the models, we propose an approach based on the combination of results from these models in a single results vector using the strategy discussed in the subsection 3.2. The overall model evaluation results are shown in Fig. 20

Fig. 21 clearly state that we have better results using our combined strategy with respect to each single model.

Some detection examples using this strategy are in Fig. 22.

We explicitly point out that all objects have been detected and classified using our combined approach. For example, in the first image, the *baseball* is detected and classified with a maximum score of 0.856 given by YOLOv3. The same object has been detected only by two of the three used models.

After the detection of the combiner module, the image caption is generated according to the approach presented in subsection 3.3.

The image captions generated for our examples are shown in Table 5.

Table 5
Captions for test images.

The image img16 contains 1 person and 1 sports object on the left; 1 person and 1 sports object on the right.
The image img345 contains 2 electronic objects, 3 people, 1 furniture and 1 chair on the left; 3 people, 1 furniture and 3 indoor objects on the right.
The image img1 contains 1 outdoor object and 2 cars on the left; 1 vehicle on the right.

5. Conclusion and future work

In this work, we have proposed a system for the automatic generation of image captions. Our main contribution is the development of a framework based on the combined use of several convolutional neural networks (CNN). We have also exhaustively investigated such networks by evaluating several models based on them. The weaknesses and strengths of each model have been highlighted and discussed. The classification results have been used by means of a captioning module to generate a description of the image content in natural language, exploit-

Fig. 19. Combined segmentation with YOLOv3.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.3146357404481844
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.43814702665145083
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.3025196579904964
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.12371738106712983
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.3329712474727528
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.48560482815702954
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.2432050067807311
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.3760482411824377
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.389323758029102
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.1666958927717814
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.4567158661391417
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.6326420248083963
```

Fig. 20. Metrics for the combination of the three architectures.

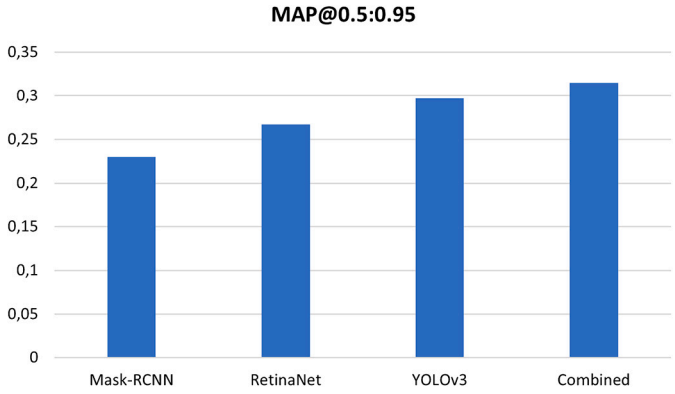


Fig. 21. Combined strategy comparison.

ing both the spatial and lexical information provided by the networks. The combination of the results gives us an impressive improvement regarding the classifications and detection steps, as shown by experiments, which indicates the goodness of our proposed approach and the effectiveness of the combined task. A straight research direction would look at comparing our framework with previous state-of-the-art works. Further improvements might be obtained from an ablation study to better evaluate each model contribution since several neural network architectures are combined. Focused crawlers [6] could be integrated into our framework for collecting images of specific domains of knowledge. Further improvements can be considered to improve the performance of the caption module using multimedia-semantic networks [29,28] or knowledge graphs [31]. Different image features can be used to have a better description of images together with semantic relations among the detected objects.

CRediT authorship contribution statement

All authors who contributed substantially to the study's conception and design were involved in the preparation and review of the manuscript until the approval of the final version. Antonio M. Rinaldi, Cristiano Russo and Cristian Tommasino were responsible for the literature search, manuscript development, and testing. Furthermore, Antonio M. Rinaldi, Cristano Russo and Cristian Tommasino actively contributed to all parts of the article, including interpretation of results,

Fig. 22. Combined strategy detection.

review and approval. In addition, all authors contributed to the development of the system for the performance of the system tests. All authors have read and agreed to the published version of the manuscript.

Declaration of competing interest

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

Data availability

The used data set is public and reference has been cited during the text manuscript.

References

[1] W. Abdulla, Mask r-cnn for object detection and instance segmentation on keras and tensorflow, https://github.com/matterport, Mask _RCNN.

[2] M.W. Akram, M. Salman, M.F. Bashir, S.M.S. Salman, T.R. Gadekallu, A.R. Javed, A novel deep auto-encoder based linguistics clustering model for social text, Trans. Asian Low-Resource Lang. Inf. Process. (2022).

[3] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, L. Zhang, Bottom-up and top-down attention for image captioning and visual question answering, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[4] M.F. Bashir, H. Arshad, A.R. Javed, N. Kryvinska, S.S. Band, Subjective answers evaluation using machine learning and natural language processing, IEEE Access 9 (2021) 158972–158983.

[5] M. Buric, M. Pobar, M. Ivasic-Kos, Ball detection using yolo and mask r-cnn, in: 2018 International Conference on Computational Science and Computational Intelligence (CSCI), IEEE, 2018.

[6] A. Capuano, A.M. Rinaldi, C. Russo, An ontology-driven multimedia focused crawler based on linked open data and deep learning techniques, in: Multimedia Tools and Applications, Springer, 2019, pp. 1–22.

[7] H. Fang, S. Gupta, F. Iandola, R.K. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J.C. Platt, et al., From captions to visual concepts and back, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[8] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in: Proceedings of the IEEE International Conference on Computer Vision, 2017.

[9] M.Z. Hossain, F. Sohel, M.F. Shiratuddin, H. Laga, A comprehensive survey of deep learning for image captioning, ACM Comput. Surv. 51 (6) (2019) 1–36.

[10] P. Hurtik, V. Molek, J. Hula, M. Vajgl, P. Vlasanek, T. Nejezchleba, Poly-yolo: higher speed, more precise detection and instance segmentation for yolov3, arXiv preprint, arXiv:2005.13243.

[11] J. Ji, Z. Du, X. Zhang, Divergent-convergent attention for image captioning, Pattern Recognit. 115 (2021) 107928.

[12] A. Karpathy, L. Fei-Fei, Deep visual-semantic alignments for generating image descriptions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[13] G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Choi, A.C. Berg, T.L. Berg, Baby Talk: Understanding and Generating Image Descriptions Proceedings of the 24th Cvpr, 2011.

[14] A. Kumar, S. Goel, A survey of evolution of image captioning techniques, Int. J. Hybrid Intell. Syst. 14 (3) (2017) 123–139.

[15] C.-W. Kuo, Z. Kira, Beyond a pre-trained object detector: cross-modal textual and visual context for image captioning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022.

[16] Y. Li, F. Ren, Light-weight retinanet for object detection, arXiv preprint, arXiv:1905.10011.

[17] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollàr, Pytorch implementation of retinanet object detection, https://github.com/yhenon/pytorch-retinanet, 2020.

[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft coco: common objects in context, in: European Conference on Computer Vision, Springer, 2014.

[19] K. Madani, A.M. Rinaldi, C. Russo, A semantic-based strategy to model multimedia social networks, in: Transactions on Large-Scale Data-and Knowledge-Centered Systems XLVII, Springer, 2021, pp. 29–50.

[20] D. Mané, et al., Tensorboard: Tensorflow's Visualization Toolkit, 2015.

[21] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, A. Yuille, Deep captioning with multimodal recurrent neural networks (m-rnn), arXiv preprint, arXiv:1412.6632.

[22] E. Mohamed, A. Shaker, H. Rashed, A. El-Sallab, M. Hadhoud, Insta-yolo: real-time instance segmentation, arXiv preprint, arXiv:2102.06777.

[23] V.-Q. Nguyen, M. Suganuma, T. Okatani, Grit: faster and better image captioning transformer using dual visual features, in: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVI, Springer, 2022.

[24] NVIDIA, P. Vingelmann, F.H. Fitzek, Cuda, release: 10.2.89, https://developer.nvidia.com/cuda-toolkit, 2020.

[25] L. Qi, Y. Wang, Y. Chen, Y.-C. Chen, X. Zhang, J. Sun, J. Jia, Pointins: Point-based instance segmentation, IEEE Trans. Pattern Anal. Mach. Intell. (2021).
[26] J. Redmon, A. Farhadi, Yolov3: an incremental improvement, arXiv preprint, arXiv: 1804.02767.
[27] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: towards real-time object detection with region proposal networks, in: Advances in Neural Information Processing Systems, 2015.
[28] A.M. Rinaldi, C. Russo, K. Madani, A semantic matching strategy for very large knowledge bases integration, Int. J. Inf. Technol. Web Eng. 15 (2) (2020) 1–29, publisher: IGI Global.
[29] A.M. Rinaldi, C. Russo, C. Tommasino, A knowledge-driven multimedia retrieval system based on semantics and deep features, Future Internet 12 (11) (2020) 183, publisher: Multidisciplinary Digital Publishing Institute.
[30] C. Russo, K. Madani, A.M. Rinaldi, Knowledge construction through semantic interpretation of visual information, in: International Work-Conference on Artificial Neural Networks, Springer, 2019.
[31] C. Russo, K. Madani, A.M. Rinaldi, Knowledge acquisition and design using semantics and perception: a case study for autonomous robots, Neural Process. Lett. (2020) 1–16.
[32] C. Russo, K. Madani, A.M. Rinaldi, An unsupervised approach for knowledge construction applied to personal robots, IEEE Trans. Cogn. Dev. Syst. 13 (1) (2020) 6–15.
[33] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, T. Poggio, Robust object recognition with cortex-like mechanisms, IEEE Trans. Pattern Anal. Mach. Intell. 29 (3) (2007) 411–426.
[34] A.W. Smeulders, M. Worring, S. Santini, A. Gupta, R. Jain, Content-based image retrieval at the end of the early years, IEEE Trans. Pattern Anal. Mach. Intell. 22 (12) (2000) 1349–1380.
[35] G. Srivastava, R. Srivastava, A survey on automatic image captioning, in: International Conference on Mathematics and Computing, Springer, 2018.
[36] Ultralytics, Pytorch implementation of yolov3, "YOLOv3 in PyTorch" https://github.com/ultralytics/yolov3, 2020.
[37] O. Vinyals, A. Toshev, S. Bengio, D. Erhan, Show and tell: a neural image caption generator, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.
[38] C. Wang, K. Huang, How to use bag-of-words model better for image classification, Image Vis. Comput. 38 (2015) 65–74.
[39] Y. Wang, B. Xiao, A. Bouferguene, M. Al-Hussein, H. Li, Vision-based method for semantic information extraction in construction by integrating deep learning object detection and image captioning, Adv. Eng. Inform. 53 (2022) 101699.
[40] Q. You, H. Jin, Z. Wang, C. Fang, J. Luo, Image captioning with semantic attention, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.
[41] J. Yu, J. Yao, J. Zhang, Z. Yu, D. Tao, Sprnet: single-pixel reconstruction for one-stage instance segmentation, IEEE Trans. Cybern. 51 (4) (2020) 1731–1742.
[42] Y. Zhang, X. Shi, S. Mi, X. Yang, Image captioning with transformer and knowledge graph, Pattern Recognit. Lett. 143 (2021) 43–49.