MS 13. Implementation of deep learning for image description development

Version 0.2

Documentation Information

Contract Number	2018-EU-IA-0104
Project Website	https://saintgeorgeonabike.eu/
Contractual Deadline	31.05.2020
Nature	
Author	Cédric K. Bhihe (BSC), Artem Reshetnikov (BSC)
Contributors	
Reviewer	Antoine Isaac (EF), Maria-Cristina Marinescu (BSC)
Keywords	image description, caption generation, visual relationships

Co-financed by the Connecting Europe Facility of the European Union

Change Log

Version	Author	Description Change
V0.1	Artem Reshetnikov	First Iteration
V0.2	Maria Cristina Marinescu	Final Version

Table of Contents

1. Introduction

This document highlights key points of the implementation of the deep learning approach for caption generation in SGOAB, and therefore specific to the Cultural Heritage domain. It introduces an alternative approach to deep learning over aligned (image, description) pairs, which is the approach we are currently using as a result of the lack of (sufficiently large sets of) aligned pairs for training. Milestone documents MS6 “System and module-level architecture development”, MS8 “Database for the selected use cases with the description of the datasets” and MS4 ”Methodology for aligning the visual and textual resources" complement the present document. MS4 defines three main options for caption generation based on different approaches:

1. Obtain descriptions and align them to paintings
2. Generate descriptions by manual annotation
3. Generate descriptions automatically and test them manually

MS9 explains the first two options, while this document goes in depth into the third option.

2. Caption generation based on object detection and scene composition analysis

The state-of-art caption generation approaches were described in section 2.4 (“Caption generation based on attention mechanism”) of MS6 “System and module-level architecture development” and in MS14 “State-of-the-Art review for deep learning techniques to generate image descriptions”. However, implementing these approaches in our project is difficult due to training data issues. According to the article “Evaluating the effect of dataset size on predictive models using supervised learning techniques”[1], developing a deep learning model requires using a reasonable size dataset for modelling, but what constitutes a reasonable size of data remains vague. In some cases the use of small size of sample data is expected to improve performance of the model if the appropriate techniques are used. However, training a caption generation model requires a significant amount of clean and well-prepared data even when using transfer learning.

Our current approach is to create descriptions automatically based on the identified objects and the visual relations between each detected object pair - inferred based on object labels and bounding boxes for the objects identified in a painting. Given that the descriptions are automatically generated with no human in the loop, the results must be evaluated. These descriptions are simple in form and rather mimic the triple format that is generated, rather than being descriptions in more complex natural language. On the other hand, the deep neural networks used in options 1 and 2 could, at least theoretically, generate complex statements, if the training set contains this style of descriptions.

Images of cultural heritage objects - paintings in particular - often represent multiple objects that interact with each other. To understand a scene or the iconographic meaning of a painting, being able to recognize individual objects is generally not sufficient. The relationships among them also contain crucial messages. Thanks to the advances in deep learning, the past several years have witnessed remarkable progress in several key tasks in computer vision, such as object recognition, scene classification, and attribute detection. In our approach we first detect objects and their corresponding bounding boxes (BBx) by using object detection deep learning models. We then use heuristics based on the relative location, sizes and overlapping of the labeled BBx to generate triples in the form of (Object, Relationship, Object) or object attributes [4]. These triples can be seen as simple descriptions or (together) a form of a knowledge graph corresponding to the image. The following section describes some details of this pipeline.

Our current approach for caption generation has two steps. First we perform object detection, which includes the generation of the most probable object labels and the corresponding BBx; this task uses deep learning. Detailed technical aspects are described in Section 2.1 “Implementation of training process of object detection model”. The second algorithmic step consists of scene composition analysis, including the analysis of visual relationships between pairs of detected objects. Our approach is rule-based and the corresponding heuristics admit a small degree of specialization to iconographic content for the time-period and subject-matter of interest (sacred art produced between the XIV and the XVIII centuries). More detailed technical aspects are presented in section 2.2 of this report and in section 2.5 of MS6 “System and module-level architecture development”.

2.1 Implementation of model training for object detection

Object detection is a fundamental task based on which we generate visual relations. If the labels are incorrectly predicted or the BBx are incorrectly drawn, this can have a cascaded negative effect on the descriptions that we are able to generate. After review of many different deep learning architectures, the Mask R-CNN from Matterport [5] was chosen as the baseline architecture for SGoaB.

This is an implementation of the article “Mask R-CNN”[6] using Python 3, Keras, and TensorFlow. The model generates bounding boxes and segmentation masks for each instance of an object in the image. It is based on the Feature Pyramid Network (FPN) and a ResNet backbone.

Environment requirements include:

- Python 3.x for x ≤7.0 along with the following packages:
  - numpy
  - scipy
  - Pillow
  - cython
  - matplotlib
  - scikit-image
  - tensorflow>=1.3.0 and <2.0
  - keras>=2.0.8
  - opencv-python
  - h5py
  - imgaug
  - horovod
- CUDA
- The NVIDIA CUDA Deep Neural Network library (cuDNN)

We additionally used the following techniques to improve our model beyond the state of the art models: metrics of the model and decrease the training time:

- **Transfer learning.** A pre-trained model is a saved network that was previously trained on a large dataset, typically on a large-scale object detection task. This model may be used as is - if the objects for which it was trained are the same as those for the current task - or use transfer learning to customize this model to a different task.

  The intuition behind transfer learning for object detection is that if a model is trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world. It is possible then to take advantage of these learned feature maps without having to start from scratch by training a large model on a large dataset.

In SGOAB we are using as a pre-trained model the object detection model trained on the MS COCO dataset[7].

- **Distributed deep learning.** The training time on the initial 4K dataset was small enough to fit on one or multiple GPUs within a cluster node; as the dataset grew to 8k, so did the training times, which sometimes took a week to complete on a single node of the cluster with two GPUs. To maintain short training times we switched to distributed training.

  In the article "Parallel and Distributed Deep Learning" [8], Vishakh Hegde described several parallel and distributed methods:

  - Local training: The model and data are stored on a single machine.
    - Multi-core processing: Here, we assume that the whole model and the data can fit into the memory of a single machine with multiple cores. These multiple cores share the available memory. There are two ways to use multiple cores to speed up the training process.
      - Use the cores to process multiple images at once, in each layer. This is an embarrassingly parallel process.
      - Use multiple cores to perform stochastic gradient descent of multiple mini-batches in parallel.
    - Use GPU for computationally intensive subroutines such as matrix multiplication.
    - Use both multi-core processing and GPU where all cores share the GPU and computationally intensive subroutines are pushed to the GPU.
  - Distributed training: When it is not possible to store the whole dataset or a model on a single machine, it becomes necessary to them across multiple machines.
    - Data parallelism: Data is distributed across multiple machines. This can be used in the case when the dataset is too large to be stored on a single machine, or to achieve faster training.
    - Model parallelism: If the model is too big to fit onto a single machine, it can be split across multiple machines. For example, a single layer can be fit into the memory of a single machine and forward and backward propagation involves communication of output from one machine to another in a sequential fashion.

Our solution implements data parallelism by using the Horovod Python package. We made some modifications to the original code of the Mask R-CNN architecture to be able to train the model on several nodes of the cluster. Concretely, we execute the following steps:

1) Run multiple copies of the training script and each copy:
   a) reads a chunk of the data
   b) runs it through the model
   c) computes model updates (gradients)
2) Average gradients among the multiple copies
3) Update the model
4) Repeat (from Step 1a)

We do not use model parallelism because our model is small enough to fit onto a single cluster node.

2.1 Implementation of scene composition analysis

Multiple objects can be successfully detected and labeled in an image (e.g. by R-CNN). However, part of the challenge inherent in building systems for automatic image captioning is that learning the visual relationships between detected objects in an image is not trivial. In this section, we describe how a custom implementation of bounding-box (bbx) analysis yields useful visual relationships between objects previously detected by R-CNN technology. We dubbed this Python-based implementation "*VIS-REL*".

Our code is applied to imagery representing sacred art produced between the 14th and the 18th centuries (both included). The context being that of sacred iconography, producing captions to enrich image annotations is a task that broadly corresponds to Panofsky's second level of interpretation[1] of cultural heritage imagery. An example of that would be for the image beholder or the image processing system to rightfully conclude that 13 men having supper with bread and wine (primary level of interpretation) represent the figure of Jesus Christ flanked by his 12 apostles in "The Last Supper" before his crucifixion in Jerusalem (secondary level of interpretation) as described in the New Testament of the Christian Bible. In the next subsections, we show how bbx analysis is instrumental in producing visual relationships at Panofsky's first level of interpretation.

2.1.1 Data model

Our input consists solely of bounding boxes, accompanying labels and the corresponding image size in pixels. Consider the color image of Figure 1 (left) representing the crucifixion of Jesus Christ. Figure 1 (right) also shows the obtained R-CNN processing output, against a gray-scale background so bbxes and labels are more easily distinguished.

Our data model (as detailed later in Table 1) is based on the representation of Figure 2, where the origin of pixel coordinates is traditionally situated top left and the ordinate axis points down.

---

1 Erwin Panofsky's three levels of interpretation applied to artworks are:

- **1st level of interpretation**: primary or natural subject matter understanding of pure forms. For instance: "I see three men standing next to a seated woman. A fourth figure rides a horse left of them. The woman sits in the gallery of a house. I see trees and clouds in the background."

- **2nd level of interpretation**: secondary or conventional subject matter understanding of iconography. For instance: "13 men having supper is the Last Supper", or "A haloed man on a cross surrounded by lamenting men and woman, in the presence or not of two more crucified figures is the Crucifixion of Christ."

- **3rd level of interpretation**: tertiary or intrinsic meaning (iconology) draws from the fact that artistic representations are not isolated occurrences but artifacts caught in the flow of time, and as such are strongly influenced by their historical environment and the prevailing habitus. It seeks to answer the question: "why did the artist elect this artistic modality, those symbols and other aspects of the composition to represent the scene?"

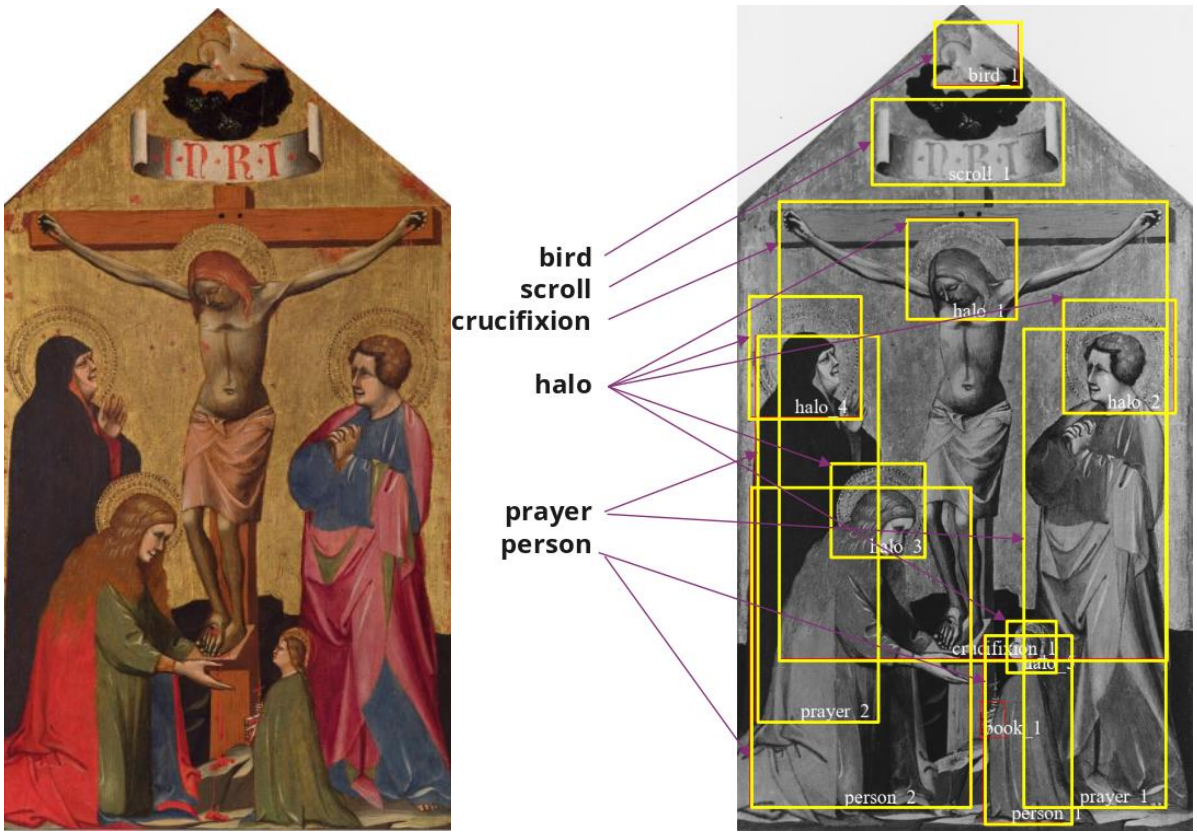

Fig.1: Crucifixion, by Jacopo di Paolo around 1400 (public domain).

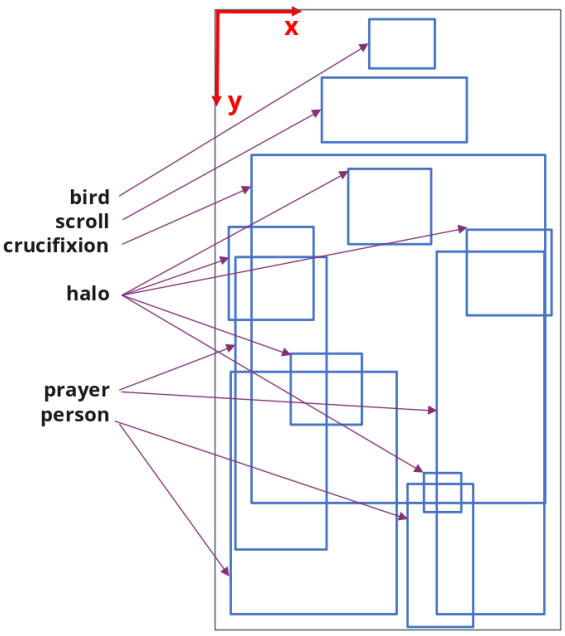

Fig.2: Cartesian coordinate system with origin located at top-left corner of image.

Parameter name	***Expanded name***	***Description***
label	object label	detected object's class label obtained from R-CNN
label_id	label identifier	used to distinguish between various detected objects belonging to the same class in an image.
uniq_label	unique label	results from the concatenation of label and label_id
cpt	bbx center point	(x,y) tuple, representing a bbx' center point. Values [2] are in $\mathbb{N}^+$ x $\mathbb{N}^+$.
oloc	object location	location of the bbx' center point on the image plane according to a 3x3 grid defining 9 regions; horizontally, then vertically from the top left corner: tl, tc, tr, lc, cc, rc, bl, bc, and br where t=top, c=center, b=bottom, l=left, r=right
rsa	relative surface area	surface area of the bbx relative to that of the whole image. Values are in [0,1].
off	orientation and form factor	composite parameter with values in]-1,1[, indicating: ***(i)*** the orientation of the bbx rectangle, whether vertical (positive values), horizontal (negative values) or neither (for a square bbx, with off=0), ***(ii)*** the form factor or aspect ratio of the bbx, where high absolute values indicate an elongated shape, whereas 0 indicates a perfect square.
dcpt	distance between bbxes' center points	computed distance (in pixels, $\in \mathbb{N}^+$) between any 2 bbxes' center points.
cpix	closest pixels	closest distance (in pixels, $\in \{-999\} \cup \mathbb{N}^+$ between any 2 bbxes. A value of 0 indicates touching bbxes. Overlapping bbxes have a singular cpix value of -999.
bro_1_2	relative overlap between bbxes 1 and 2	parameter overlapping surface area between any 2 bbxes (1 and 2) , calculated relative to the first bbx (1). Values are in [0,1].
rpos_1_2	relative position of bbx 1 and 2	relative position of bbxes relative to one another, expressed clockwise as NN, NE, EE, SE, SS, SW, WW, NW, where N=north, E=east, S=south, W=west
xmin	minimum bbx abscisa	unit in pixel. Values are in $\mathbb{N}^+$.
ymin	minimum bbx	unit in pixel. Values are in $\mathbb{N}^+$.

2 $\mathbb{N}^+$ denotes the set of positive integers, 0 included.

	ordinate	
xmax	maximum bbx abscisa	unit in pixel. Values are in $\mathbb{N}^+$.
ymax	maximum bbx ordinate	unit in pixel. Values are in $\mathbb{N}^+$.

Table 1: Data model parameters

The parameters of Table 1 are computed per image, for all bbxes, taken singly and in pairs. They are organized in 2-D square matrices where diagonal components characterize one single bbx, whereas off-diagonal components characterize bbxes pairwise. To load and access those parameters efficiently, each image's square matrix is persisted in binary format, as an ndarray using the Python package `*numpy*`. In practice, those arrays are suffixed with `*.vra*`.

2.1.2 Depth of field and perspective induced size effects

In order to assess whether any two detected objects, belonging to any two arbitrary object classes (e.g. a person and a horse), are in the same image view-plane, that is to say, at the same field-depth in an image, one needs a base-reference of pairwise proportions between objects of every trained class.

In practice *VIS_REL* computes pairwise-proportions based on common-sense measures and proportions translated as relative surface area proportions between bbxes. Those pairwise proportions between detectable objects are meant to reflect a common-sense representation of realistic pictorial proportions in paintings. However one may object that they also should be tuned to variations in styles and techniques by various artists across centuries. For instance, the proportions of animals between themselves and with respect to other object-classes can vary widely between artists, and over time between the 14th and the 18th centuries. For that reason, the SGoaB implementation allows for those proportions to be modified. For the time being however, pairwise-proportions are applied equally to all processed images, across all artistic periods.

Pairwise-proportions are kept in a 2-D asymmetric square matrix, with diagonal components equal to 1. They are persisted as an ndarray using Python `*numpy*`.

If the baseline of two bbxes, e.g. delimiting human-like figures, are similar, indicating commonality of view-plane or field-depth, but their sizes are out of proportions, our implementation may conclude that one of those two figures is either a child, a dwarf, or otherwise positioned in a way that makes it appear smaller, while the other is adult-sized. Figures 3 and 4 illustrate this case.

Fig. 3: Illustration from the book of Tobit (anonymous, date unknown), showing Tobit's son, Tobias, accompanied by the angel Raphael on their way to the ancient city of Rhages. Both figures have similar bbx' baselines (parameter "ymax") and are therefore situated on the same view plane. *VIS-REL* will detect both human-like figures as "standing", based on their "off" parameter values. Comparing the sizes (parameter "rsa") and baselines of objects "angel_1" and "person_1", it will infer that "person_1" and "angel_1" are positioned next to one another and that "person_1", having a significantly smaller size than "angel_1", is either a child or a dwarf, located to the onlooker's right hand of "angel_1" (parameter, rpos_2_1="EE", where index 1 and 2 denotes respectively "angel_1" and "person_1").

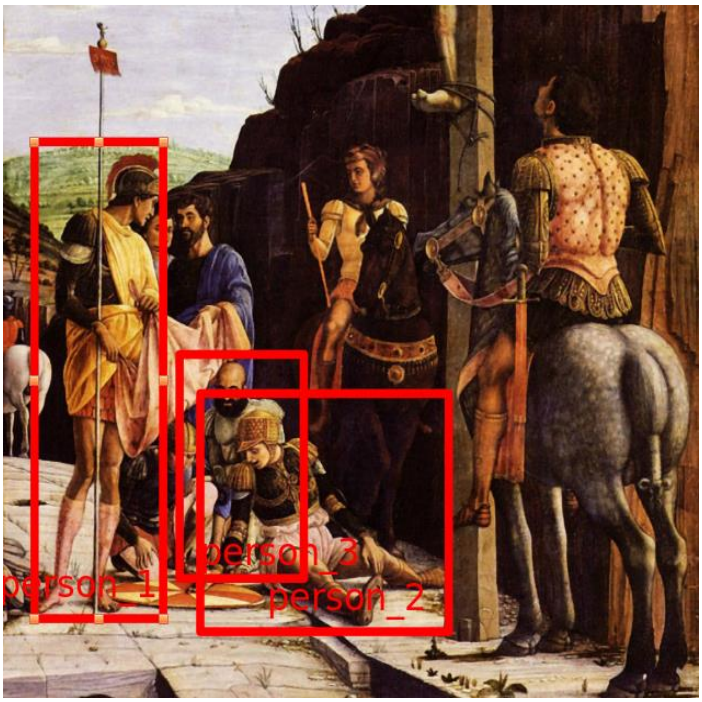

Fig. 4: R-CNN output, , here showing only 3 bbxes, each delimiting human figures. “person_1” and “person_2” have very similar bbx’ baselines (parameter “ymax”) and are therefore identified as scene-objects belonging to the same view-plane. *VIS-REL* will detect “person_1” as standing and person_2 as either sitting, bending, kneeling, squatting or crouching, based on their respective “off” and “rsa” parameters’ values. *VIS-REL* also detects that “person_3” is either sitting, bending, kneeling, or crouching, and situated slightly behind “person_1” and “person_2”.

Detail of “Crucifixion” (tempera on panel, center piece of the predella of the Verona San Zeno altar polyptych) by Andrea Mantegna (ca. 1458).

In the absence of a common bbx baselines for two detected objects, a smaller-than- expected object_1 in a depicted scene (as would be suggested by our common-sense pairwise-proportions) may indicate that it is located in the background of object_2 to whose size it is compared. Figures 5, 6 illustrate three different cases of scene compositions involving human-like figures with different baselines and different conclusions as to the relative location of figures with respect to one another, as inferred by the VIS-REL bbx analysis.

Fig. 5: Two detected objects with vastly different baselines: ymax(person_1) ≫ ymax(person_2). How- ever person_2 has a relative overlap with person_1 greater than or equal to 80% (bro_2_1≥0.80). In iconographical terms it translates as person_2 being ***in_front_of*** person_1. The smaller size and location of person_2 relative to that of person_1 (parameters "rsa" and oloc_2_1="NN") combined with the attributes of person_1 (halo and crown), also detected by R-CNN although their respective bbxes are not shown here, constitute the first steps in the disambiguation of the scene as a possible Madonna representation. In this case, the resulting caption-seeds for that scene are:

- "person_1 stands", or "person_1 is_vertical"
- "person_1 wears crown"
- "person_1 has halo" => "person_1 is saint"
- "person_2 kneels/crouches/sits/squats/bends"
- "person_2 has halo" => "person_2 is saint"
- "person_2 is child/dwarf"
- "person_1 holds person_2"

"Madonna" or "Virgin and Child" by Matthew Paris, illustration of Historia Anglorum (ca. 1255, British Library Coll.).

2.1.3 Heuristic determination of visual relationships

Inference of visual relationships between co-occurring objects is rule-based. It allows for the:

- elucidation of relative positions of pairs of detected objects
- detection of bbx overlaps attributable to:
    - either perspective (2 objects' bbxes may overlap but the corresponding objects may correspond to different view-planes and thus may not be close to one another,
    - or proximity and direct interaction, when the 2 corresponding objects share the same view-plane,
- general ordering of objects in the composition, according to a list of positional predicates:
    - is_next_to

- is_behind
- is_before
- is_to_left
- is_to_right
- is_below/is_beneath/is_under
- is_above
- is_on, implying direct contact
- is_with

The methodology of bbx-analysis is quantitative whenever possible, self-consistent and algorithmically *independent of the absolute size* of detected objects in a scene.

Partial departure from quantitativeness in object ordering occurs when a concept such as that of the bbx' baseline cannot be reliably utilized to ascertain their view-plane. An example is the class label "crucifixion", whose bbx does not usually include the base of the cross, but rather limits itself to the body of the crucified figure. In this instance, as in Figure 6 below, bbxes' baselines are not instrumental in determining where the view-plane of various detected objects are with respect to one another. Instead we must rely purely on the pairwise proportions of co-occurring objects and therefore tolerate a less precise measure of view-plane depth.

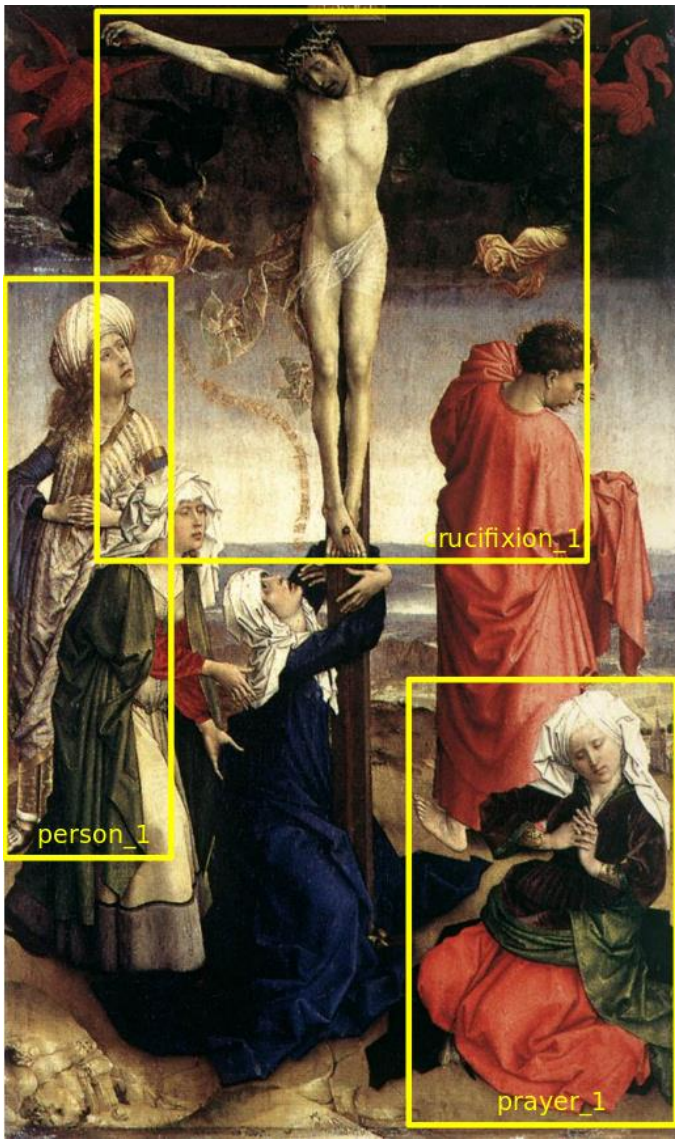

Fig. 6: Illustration of objects' view-plane disambiguation based on the analysis of baselines (parameter "ymax"), size and relative proportions (parameter "rsa" and "ppa_1_2", not shown in Table 1), orientation and form factor (parameter "off") as well as relative location (parameter "rpos_1_2").

VIS-REL identifies:
- "person_1" as ***standing*** and located ***behind*** the crucified figure of "crucifixion_1".

- "prayer_1" as either sitting, bending, kneeling, squatting or crouching and located ***in_front_of*** the crucified figure in "crucifixion_1"

"Crucifixion" by Rogier van der Weyden, ca. 1425~30 (oil on panel, Gemäldegalerie, Berlin).

A short example of pseudocode is provided in Figure 7 for illustrative purposes. It relies on parameters "rsa", "rpos_1_2", "off", "xmin", "ymin", "xmax", "ymax" and reflects pre-existing domain-specific iconological knowledge on two main aspects:

- a human-like figure adorned with a halo is a ((saint)) or a ((martyr)).
- a scene in which a bird appears high above a special object is the sign that that object is singled out by the ((holy_spirit)).
- special objects tend to be represented front and center and to be bigger than others in the represented scene.

In *VIS-REL* special objects are generally main topic candidates and are detected relying on parameters "bro_1_2", "oloc", "rsa" and "ymax".

The pseudocode below applies to pairs of bbxes in Figure 1. We chose to concentrate on labels "crucifixion_1" (bbx_1), "halo_1" (bbx_2)) and "bird_1" (bbx_3).

```
if  ( bbx_1= 'crucifixion_1' and  bbx_1 is_vertical
      and ( bbx_2 = 'halo_1'
            and  overlap with bbx_1
            and is_in_upper_region of bbx_1
            and pairwise-proportions are respected )
    )
then  "'crucifixion_1'((person)) is ((saint))"

if  (bbx_1= 'crucifixion' and  bbx_1 is_vertical
     and ( bbx_3 = 'bird'
           and is_north_of 'crucifixion_1'
           and baseline_3 is_above topline_1
           and pairwise-proportions are respected )
    )
then  "'crucifixion_1'((person)) is ((jesus_christ))"
```

Fig. 7: Example of simplified pseudocode applicable in the case of Figure 1 above. Double parentheses are used to represent labels of concepts or entities not present in our list of class labels and inferred by *VIS-REL*.

Although rudimentary this rule-based logic allows us to determine that the scene is that of the crucifixion of ((jesus_christ)).

2.2.4. Results and discussion

Generally *VIS-REL* yields caption-seeds that take the form of triples *(subject, predicate, object)*, as in:

(person_2, rides/is_on, horse_1), or

(person_3, wields, lance), or

(angel_1, is_with, lily), or

(person_6, is_next_to, person_4)

The bbx analysis may also yield pairs, where only an object attribute-property is signified, as in:

(person_1, stands/is_vertical), or

(shield, is_oblique/is_horizontal)

This is meant to be compatible with a simple semantic representation of visual relationships or of object representational attributes.

In light of the examples provided in subsections 3-2-1 and 3-2-2, difficulties become immediately apparent. In particular the representation of partially obfuscated objects in paintings or computer-inferred bbxes with incomplete surface extension around any detected object make using *VIS-REL* difficult to use.

The performance of *VIS-REL* is completely conditioned by the quality of the R-CNN's output. If the neural network object detection and label classification yield erroneous labels and/or coarsely drawn bbxes, *VIS-REL* may be rendered ineffectual.

In some images, pairs of overlapping bbxes may cause *VIS-REL* to conclude that certain object-pairs entertain a visual relationship when they do not. It is the case of the scene representation in Figure 1 where the object "crucifixion_1" overlaps significantly with the bbxes of three out of five detected halos ("halo_1", "halo_2", and "halo_4") Only "halo_1" is genuinely related to the object "crucifixion_1" while the two others are related to the human figures left and right of the cross and close to it. Non disambiguated caption seeds are highlighted in burgundy below, for "crucifixion_1+halo_2" and "crucifixion_1+halo_4":

```
"crucifixion_1+prayer_1": [
        "crucifixion_1((person)) is_on ((cross))",
        "prayer_1 is_vertical",
        "prayer_1 is_near crucifixion_1((person))/((cross))",
        "prayer_1 is_below crucifixion_1((person))/((cross))",
        "prayer_1 is_right_of crucifixion_1((person))/((cross))"],
"crucifixion_1+person_1": [
        "crucifixion_1((person)) is_on ((cross))",
        "person_1 is_vertical",
        "person_1 is_behind crucifixion_1((person))/((cross))",
        "person_1 is_beneath crucifixion_1((person))/((cross))"],
"crucifixion_1+person_2": [
        "crucifixion_1((person)) is_on ((cross))",
        "person_2 is_vertical",
        "person_2 is_near crucifixion_1((person))/((cross))",
```

```
            "person_2 is_beneath crucifixion_1((person))/((cross))"],
"crucifixion_1+prayer_2": [
            "crucifixion_1((person)) is_on ((cross))",
            "prayer_2 is_vertical",
            "prayer_2 is_near crucifixion_1((person))/((cross))",
            "prayer_2 is_below crucifixion_1((person))/((cross))",
            "prayer_2 is_left_of crucifixion_1((person))/((cross))"],
"crucifixion_1+halo_1": [
            "crucifixion_1((person)) is_vertical",
            "crucifixion_1((person)) is_on ((cross))",
            "crucifixion_1((person)) is_with halo_1",
            "crucifixion_1((person)) is ((saint))"],
"crucifixion_1+halo_2": [
            "crucifixion_1((person)) is_horizontal/is_inclined",
            "crucifixion_1((person)) is_on ((cross))",
            "crucifixion_1((person)) is_with halo_2",
            "crucifixion_1((person)) is ((saint))"],
"crucifixion_1+halo_4": [
            "crucifixion_1((person)) is_horizontal/is_inclined",
            "crucifixion_1((person)) is_on ((cross))",
            "crucifixion_1((person)) is_with halo_4",
            "crucifixion_1((person)) is ((saint))"],
"prayer_1+person_1": [
            "person_1 stands",
            "prayer_1 stands",
            "prayer_1 is_behind person_1",
            "person_1 is ((child))/((infant))/((dwarf))_(222)"],
"prayer_1+halo_2": [
            "prayer_1 stands",
            "prayer_1 is_with halo_2",
            "prayer_1 is ((saint))/((martyr))"],
"person_1+halo_5": [
            "person_1 stands",
            "person_1 is_with halo_5",
            "person_1 is ((saint))/((martyr))"],
"person_2+prayer_2": [
            "prayer_2 stands",
            "person_2 stands"],
"person_2+halo_3": [
```

```
            "person_2 stands",
            "person_2 is_with halo_3",
            "person_2 is ((saint))/((martyr))"],
    "prayer_2+halo_4": [
            "prayer_2 stands",
            "prayer_2 is_with halo_4",
            "prayer_2 is ((saint))/((martyr))"
```

The appropriate disambiguation for “halo_2” and “halo_4” will be carried out elsewhere (i.e. in a separate algorithmic treatment) based on the two visual relationships “prayer_1 + halo_2” and "prayer_2+halo_4" satisfactorily represented in the above output.

3. Code

The code is available in the GitLab repository of the project, at the following address: https://gitlab.bsc.es/sgoab/object-detection.

4. References

[1] *A.R.Ajiboye, R.Abdullah-Arshah, H.QinandH.Isah-Kebbe,* “Evaluating the effect of dataset size on predictive model using supervised learning technique”, IJSECS, 2015

[2] *Samuel Lipping, Konstantinos Drossos, and Tuomas Virtanen,* “Crowdsourcing a dataset of audio captions”, Detection and Classification of Acoustic Scenes and Events, 2019

[3] *Yin Cui, Guandao Yang, Andreas Veit, Xun Huang, Serge Belongie*, “Learning to Evaluate Image Captioning”, 2018

[4] *Bo Dai, Yuqi Zhang, Dahua Lin,* “Detecting Visual Relationships with Deep Relational Networks”, CVF, 2017

[5] https://github.com/matterport/Mask_RCNN

[6] *Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick*, “Mask-RCNN”, 2018

[7] *Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár* Microsoft COCO: Common Objects in Context. 2014

[8] *Vishakh Hegde, Sheema Usmani,* “Parallel and Distributed Deep Learning”