Image Models for large-scale Object Detection and Classification

Jordan Kralev
Technical University, Sofia
jkralev@tu-sofia.bg

Svetla Koeva
Institute for Bulgarian Language, BAS
svetla@ddcl.bas.bg

Abstract

Recent developments in computer vision applications that are based on machine learning models allow real-time object detection, segmentation and captioning in image or video streams. The paper presents the development of an extension of the 80 COCO categories into a novel ontology with more than 700 classes covering 130 thematic subdomains related to Sport, Transport, Arts and Security. The development of an image dataset of object segmentation was accelerated by machine learning for automatic generation of objects' boundaries and classes. The Multilingual image dataset contains over 20,000 images and 200,000 annotations. It was used to pre-train 130 models for object detection and classification. We show the established approach for the development of the new models and their integration into an application and evaluation framework.

Keywords: image dataset, image models, object detection, object classification

1 Introduction

The shift of traditional data fusion methods challenged by multimodal big data motivates the creation of a new image corpus, the Multilingual Image Corpus, which is characterised by carefully selected images that illustrate thematically related domains and precise manual annotation for the segmentation and classification of objects in the images.

Recent developments in computer vision applications that are based on machine learning models allow real-time object detection, segmentation and captioning in image or video streams (Kasapbaşi et al., 2022; Cameron et al., 2019). We developed an image processing pipeline for object detection and object segmentation using pre-trained models. We also delivered a reliable service for automatic annotation of objects in images using advanced deep learning techniques (Michelucci, 2019) and some existing tools and machine learning frameworks.

The paper presents the development of an extension of the 80 COCO categories into a novel ontology with more than 700 classes covering 130 thematic subdomains related to Sport, Transport, Arts and Security (presented in Section 2). The development of an image dataset for object segmentation and classification (The Multilingual image Corpus, MIC21) was accelerated by machine learning for automatic generation of objects' boundaries and classes (presented in Section 3). The MIC21 image dataset containing more than 20,000 images and 200,000 annotations was used to pre-train 130 models for object detection and classification. We show the accepted approach for the development of machine learning models and their integration into a framework for the evaluation and running of models (in Section 4).

In other words, we will demonstrate the application of models for the prediction of object outlines and classes in images as part of the development of the Multilingual Image Corpus, and then we will show how the new dataset, in turn, can be used to pre-train existing models so that they predict large number of object classes.

2 Multilingual Image Corpus in brief

The Multilingual Image Corpus offers data to train models specialized in object identification, segmentation and classification by providing fully annotated objects within images with segmentation masks categorised according to an Ontology of Visual Objects. The Multilingual Image Corpus is distinguished by the following key features: a) large image collection containing thousands of images and annotations; b) an Ontology of visual objects specifically created for object classification;

c) preparatory automatic object segmentation and classification evaluated by experts; d) translation of object classes and attaching definitions of concepts in 25 languages.

The dataset contains images from 4 thematic domains (Sport, Transport, Arts and Security), which represent highly related objects such as Tennis player and Soccer player, Limousine and Taxi, Singer and Violinist, Fire engine and Police boat grouped in 130 subsets of images. The images in the dataset are collected from a range of repositories offering API: Wikimedia, Pexels, Flickr, Pixabay, Creative Commons Search. Each image is equipped with a metadata description in JSON format. The metadata include fields such as: the name of the sub-dataset, sub-dataset id, image author, author's web address, image original size, file name, image license, image source, last access to the source, source's web address, MIC21 project url, etc. (Koeva et al., 2022)

The selected classes for annotation are organized into an Ontology of visual objects (Koeva, 2021). The Ontology consist of 706 classes that describe visual objects, 147 classes that represent their hypernyms, 14 relations between concepts and axioms that make explicit claims about the relations between concept classes. The Ontology classes correspond (but are not limited) to WordNet concepts (Fellbaum, 1999; Miller et al., 1990) which can be represented by visual objects (almost half of the Ontology classes are not contained in the WordNet). Two of the relations and their properties are also inherited from WordNet.

For example, the dominant class **Accordionist** is represented in WordNet, while the dominant class **Handball player** – not. For new dominant classes the appropriate hypernym in the WordNet structure is determined, in this case – **Athlete**. The attribute classes for Handball player are: Handball referee, Handball court, Handball, Handball goal, Handball jersey, Handball pants, Handball shorts, Handball shoe, Handball sock, Race number, Knee pad, from which only Handball and Knee pad are part of the WordNet. Ontology relations between the dominant class and its attribute classes link depicted relations between visual objects, for example: Handball player is next to Handball referee, Handball player plays at Handball court, Handball player plays with Handball and so on.

The use of the Ontology of visual objects ensures the selection of mutually exclusive classes, the interconnectivity of classes by means of formal relations and an easy extension of the Ontology with more concepts corresponding to visual objects.

All Ontology classes have been translated into 25 languages using publicly available wordnets and BabelNet: English, Albanian, Bulgarian, Basque, Catalan, Croatian, Danish, Dutch, Galician, German, Greek, Finnish, French, Icelandic, Italian, Lithuanian, Polish, Portuguese, Romanian, Russian, Serbian, Slovak, Slovenian, Spanish, and Swedish (Koeva et al., 2022).

An image processing pipeline for object detection and object segmentation was developed. Two software packages – Yolact (Bolya et al., 2019) and Detectron2 (Wu et al., 2019), and Fast R-CNN (Girshick, 2015) models trained on the COCO dataset were used for the generation of annotation proposals. The COCO format is a commonly used format for the instance segmentation representation (Sun et al., 2022; Amo-Boateng et al., 2022; Conrady et al., 2022; Cui et al., 2022).

The task for the annotators was to correct, reject or create new polygons for individual objects in the image and to classify the objects against the classes from the predefined Ontology. Table 1 displays the Multilingual Image Corpus's current status.

Domain	**Images**	**Annotations**
Sport	6,915	65,482
Transport	7,710	78,172
Arts	3,854	24,217
Security	2,837	35,916
MIC21	**21,316**	**203,797**

Table 1: The Multilingual Image Corpus in Numbers

The metadata of images, Ontology, object annotations and multilingual descriptions of Ontology classes are available to be downloaded, copied, modified, distributed, displayed and used in accordance with the Creative Commons Attribution-ShareAlike 4.0 International License.[1][2][3]

The Multilingual image dataset can be implemented in: automatic identification and annotation of objects in images (a prerequisite for effective search of images and (within) video content), automatic annotation of images with short descriptions in European languages.

[1] https://doi.org/10.57771/be1g-vm57
[2] https://doi.org/10.57771/hxe0-4826
[3] https://doi.org/10.57771/v36v-yb33

We developed a framework based on FiftyOne, Yolact and Detectron2, and implemented it over Mask R-CNN on Python3, Keras and TensorFlow. We pre-trained Fast R-CNN models using the Detectron2 framework with ground truth annotations, which resulted in 130 models that generate bounding boxes and segmentation masks for each instance of a particular object within an image. The framework maintains an API functionality for processing new images with any of the three models: Yolact, Detectron2 and MIC21. The MIC21 framework allows for evaluation and comparison of the grand truth and MIC21 models annotations as well as for running the models on new image datasets.[4]

We will present in more detail the integration of existing models in our Image Processing Pipeline in order to automatically predict the objects' boundaries in images and their classes, as well as the development of 130 models based on the Multilingual Image Corpus, which can be used for object recognition and classification and for future experiments.

3 Image Processing Pipeline

The Image Processing Pipeline contains a number of modules that support the work of annotators in several ways: by predicting object outlines and by managing images and annotations.

The Multilingual Image Corpus is organized into manageable in size datasets containing at least 100 images (in rare cases) and in the most common case – about 150 images. The initial datasets roughly correspond to the final thematic subdomains that will be formed. However, in the preliminary stage of the work, it was acceptable to have several datasets representing a single thematic subdomain, images classified in inappropriate datasets, etc. Therefore, the initial organization of the images into datasets is mostly with respect to the collection methodology and the decision on the size of the data. After the initial processing of the images and the manual annotation, some images are reorganized, if necessary, which reflects the final content of the thematic subdomains.

All input images are represented in raster image compression formats such as: Portable Network Graphic (PNG), Joint Photographic Experts Group (JPEG) or Tag Image File Format (TIFF). The size of images varies considerably between 2 to 11 megapixels. The small image dimensions may affect the quality of the annotated regions. On the other hand, when the image dimensions are too large, the amount of allocated memory, as well as the processing time, increases exponentially. Another requirement for the input image is its colour space format to be in red, green and blue (RGB) channels without additional or missing channels.

For an effective processing of images with a convolution model they have to be in proper dimensions and in the RGB color space. Hence, the first module of the pipeline examines each image and performs the necessary transformations (resizing and/or color space mapping); in case the transformations are not possible, the image is excluded from the dataset. As a result, the images are described by their attributes as JSON objects. The pre-processing step is automated by a Python script, which calls some of the OpenCV[5] routines for performing the operations over the images.

The open source Yolact (Bolya et al., 2019) provides several convolution neural network models for object detection and segmentation within the COCO domain (Lin et al., 2014). The model we have employed for the automatic annotation is Resnet50-FPN. The notation indicates that each convolution layer from the backbone component of the model includes a feed-through connection from the input to the output of the layer. Such structure is appropriate for training highly stacked models (as those used for computer vision tasks) in order to improve the numerical stability of the optimization procedure and to prevent over-fitting. The intuition behind this is that each convolution layer from the stack is approximating only the residual error between the target and the output of the previous layer. First the data is processed through a stack of input convolutional layers with decreasing resolution. Consequently, a stack of output convolutional layers with increasing resolution is applied. The dimension of each layer from an input stack is matched by a layer from the output stack. In addition, 1x1 convolutional connections are established between the corresponding layers from the input and the output stacks.

The result from dataset processing through Yolact software is the instance segmentation and classification within the COCO domain. The results are stored in MS COCO JSON format. The format provides two options for recording the bounding contour of each detected object – run-

[4] https://mic21.dcl.bas.bg

[5] https://github.com/opencv/opencv

length encoding (RLE) and point coordinates. The RLE is a compression format over the point coordinates and allows for more compact representation; however, not all systems are able to work with it directly. A useful tool for converting between formats is the Python library pycocotools. In some cases, more processing is required because the raw segmentation mask resulting from the convolution model is a binary mask. For the conversion of a binary mask to an object contour a useful routine from OpenCV library, findContours, is used.

As noted, the object detection model produces annotation data in the domain of the 80 COCO categories. The automatically obtained annotation data are imported in an open-source annotation software, the COCO-annotator (Brooks, 2019). The process is automated through a bash script, which connects to the database docker of the annotator, creates a new dataset, copies the images and imports the annotation data. The COCO-annotator features multi-user environment composed of a mongoDB database, Flask backend and Vue frontend employing a worker processing model. In the COCO-annotator, software images are organized into datasets and the front-end provides a tool for performing manual editing of annotation contours creating/deleting annotations or changing/assigning object labels.

To further accelerate manual annotation an automatic relabelling of the imported annotations in the coco-annotator database is implemented. It takes as an input a dictionary that states the relabelling rules specific for a sub-dataset. For example, the category 'Person' in the sub-dataset 'Basketball' is replaced with the class 'Basketball player' and the identifier of the new class replaces the identifier of all annotations 'Person' within the sub-dataset 'Basketball'.

Certain manipulations during the manual annotation were performed to provide functionalities that are not implemented in the COCO-annotator software. We have developed a dedicated Python script performing such operations by connecting to the mongoDB engine of the annotation software using PyMongo Python library:

- Import images and annotations when creating a new dataset;
- List annotated images by class label and store them into a file on the disk;
- List hyperlinks to images in the database according to their thematic subdomain;
- List annotated images by thematic subdomain;
- Generate statistical reports for the annotated images;
- Merge two thematic sub-datasets into a single one;
- Move one or several images from one thematic sub-dataset to another, together with their associated annotations;
- Remove images, annotations or categories from a dataset based on various criteria;
- Export all dataset images and annotations into JSON COCO format;
- Remove images from the dataset marked as deleted in the COCO-annotator software;
- Replace labels in a thematic sub-dataset according to specific rules;
- Scan image paths in the database for inconsistency and fix them if necessary;
- Change classes of particular images or differentiate labels between two distinct thematic sub-datasets.

Python scripts[6] execute each of the described operations.

After manual annotation and database post-processing, the images from the resulting 130 thematic subdomains are exported together with their ground truth annotations represented in MS COCO format. The structure of the MIC21 dataset is as follows:

```
-thematic_field_name
  - data
    - image_1.jpg
    - image_2.png
    ...
  thematic_field_name_gt.json
```

The data sub-directory for the respective thematic subdomain contains the images in jpg, jpeg or png format. The `*gt.json` field is a COCO format JSON file describing the polygonal segmentation of objects in images.

[6]https://github.com/link_will_be_provided

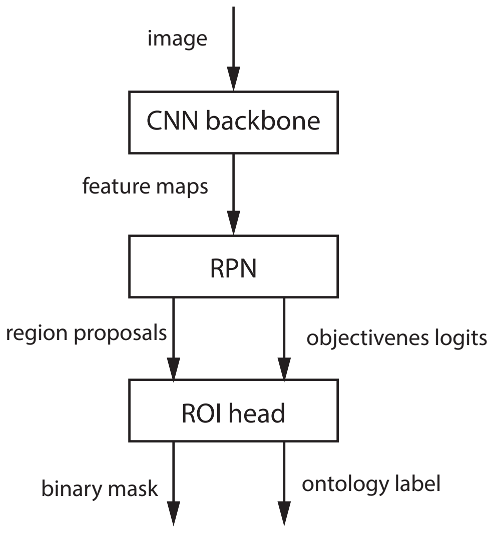

Figure 1: Structure of the model

4 MIC21 Models

We trained domain specific models, which are able to detect and label objects in an image with classes from the MIC21 ontology. The benefits of such models are twofold – first, they allow further acceleration of the manual annotation reflecting the MIC ontology of visual objects; second, they represent an extension of the standard COCO classes to 130 thematic domains. For the training of the domain specific models we use Detectron2 framework of the Facebook research group[7], which is an open-source Python software based on PyTorch library. The general structure of the object detection model is presented in Figure 1.

It is composed of a convolution neural network (CNN) backbone component, a proposal generator and a region of interest (ROI) head (Redmon et al., 2015). Detectron2 supports 3 backend structures - Resnet, Regnet and FPN. The backbone is represented as stacked convolution layers with different interconnections depending on the structure type.

In the Resnet structure, the residual building block has an option for a direct shortcut connection from the input of the layer to its output, i.e. projecting the input features into the output, and the actual network is keeping the difference between the input and the target. The output of the backend component comprises selected feature maps from the stack of layers depending on the network design. Usually, in addition to the output of the last layer, 3 to 4 of the output feature maps from the deeper layers are selected.

Region proposals in the modern object detection networks are generated through a region proposal convolution network (RPN), composed of a 3x3 convolution layer followed by 1x1 convolution layers for the generation of object box deltas and an objectiveness score for a box. As a basis for region proposals, a set of anchors is generated within the image, for example, by dividing the image into a grid of large boxes and putting an anchor point at the centre of each box. Then the anchor boxes are refined during the training of the network by fitting them to the ground truth. The result of the proposal generator layer is a list of box coordinates and an objectiveness score indicating whether the respective box contains an object or not.

The third stage in the object detection framework is a ROI head network, which iterates over each of the generated proposal boxes and performs per region classification and binary mask extraction. The prediction is based on the features from the backbone layer constrained to the current examined box.

Each of the trained models is characterised by input data, output data and parameters. The parameters represent the internal weights of the model obtained during training, which are specific for each thematic subdomain and have to be loaded through a `DetectionCheckpointer` class. The input for the model is a Python list structure `list[dict]`, where each element of the list is a dictionary field `image`, representing a 3-dimensional array with colours for each pixel from the image in RGB colour space, and also width and height attributes for the image in pixels. If the model weight is updated during the training, the input dictionary for each image has to include a field `instances` describing the coordinates of the bounding boxes for the ground truth objects in the image, as well as a class label for each object in the range `[0,num_categories]` and a ground truth binary mask for each object.

The training of the models is performed with the Detectron2 framework by inheriting the `DefaultTrainer` class. The training loop and each of the network layers are aligned with the PyTorch requirements for building neural network models. Each layer has to provide a `loss` function, which calculates the residual error given the training targets and network outputs, and additionally to provide a `forward` function, which calculates the

[7]https://github.com/facebookresearch/detectron2

layer outputs from inputs. To compute the gradients during the backward network pass the PyTorch features an `autograd` engine, which (when enabled) is able to track each arithmetic operation during the forward pass and to obtain the gradient of the residual error of the layer with respect to the parameters.

During the training, the model is evaluated periodically when a certain iteration count is passed by tracking the intersection over union (IoU) metric by category. The library `pycocotools` contains useful routines for comparing results from computer vision models either by using bounding boxes or binary masks. The IoU metric is defined as:

$$S_{IoU}(Z,T) = \frac{\mathcal{A}(Z \cap T)}{\mathcal{A}(Z \cup T)}, \qquad (1)$$

where Z is the model bounding box or mask detection, T is the corresponding ground truth instance from the same ontology class and $\mathcal{A}$ is the area calculating operator, which usually is expressed by the number of pixels in a region. In order to calculate this metric for multivariate models (with several ontology classes), first a correspondence between model outputs and ground truth targets has to be established by comparing the region overlap. When a model output region is matched to a ground truth region for a given IoU threshold, 4 metrics can be calculated:

- TP – true positive – when Z and T are from the same class;
- FP – false positive – when Z and T are matched, but the model is wrong for the class of Z;
- FN – false negative – no region Z is matched to a ground truth T;
- TN – true negative – an object that is not part of the ground truth is also left undetected by the model.

With respect to the classification outcome for a given IoU threshold, three additional metrics for the model are commonly examined, which are model precision

$$P = \frac{N_{TP}}{N_{TP} + N_{FP}}, \qquad (2)$$

reflecting how good the model is in producing correct labelling for the detected regions, model recall,

$$R = \frac{N_{TP}}{N_{TP} + N_{FN}}, \qquad (3)$$

which is about how good the model is in detecting the correct objects from a category and general model detection accuracy expressed by

$$A = \frac{N_{TP} + N_{TN}}{N_{TP} + N_{FP} + N_{FN} + N_{TN}}, \qquad (4)$$

where $N_\bullet$ denotes the number of detections over the whole dataset from TP, FP, FN or TN category. Note that each of the metrics P, R and A are functions of the IoU threshold level $S_{IoU}(Z,T)$ used to perform the matching between the model detection and ground truth regions. By selecting different IoU thresholds we will get different model performance, hence to obtain a more complete picture of the model detection capabilities P and R are evaluated for the whole range of IoU values from 0 to 1, producing the so-called precision recall curve for a model.

5 Framework for running and evaluation the MIC21 models

For the purposes of presentation, comparison and evaluation the dataset is organized into a system of components, called 'MIC21 framework' (Figure 2). The framework is composed of a backend (processing service) and frontend (visualization service) component. The processing service is implemented as a Flask server implemented in Python, which is able to run the Yolact, Detectron2 and MIC21 pre-trained domain specific models (Figure 2). The processing service offers a set of Web APIs implemented over an HTTP, with the following functions:

- Prediction of annotations using the Yolact software;
- Prediction of annotations using the Detectron2 software;
- Prediction of annotations using the MIC21 trained models;
- Import of new images, their ground truth and predictions into FiftyOne framework (into a new or already existing dataset);
- Initial loading of datasets into FiftyOne;
- A simple interface to upload a new image to a dataset;
- Evaluation of predictions against the ground truth and storing the results into the FiftyOne framework. Print the evaluation statistics.

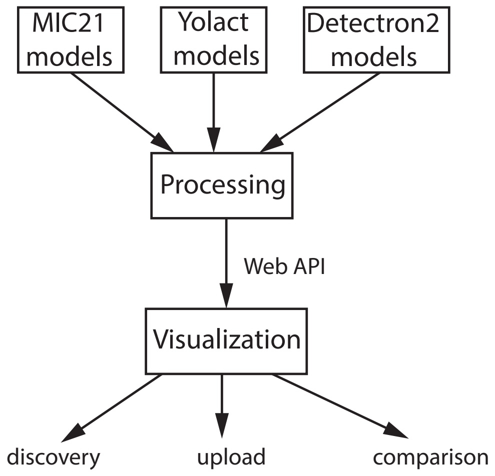

Figure 2: MIC21 framework components

The framework frontend service is based on the FiftyOne software (Moore and Corso, 2020), which is an open-source tool for building high-quality datasets and computer vision models. In the FiftyOne frontend service the Yolact, Detectron2 and MIC21 pre-trained models can be compared with the ground truth annotations for 130 subdomains in the Sport, Transport, Arts and Security thematic domains.

The repository for the framework is freely accessible at GitHub and includes: source code of Mask R-CNN built on FPN and ResNet101, training code for MS COCO, pre-trained weights for MS COCO and MIC21 classes, Jupyter notebooks for visualization the detection pipeline and evaluation routines for MS COCO metrics integrated in the FiftyOne.

The original FiftyOne code is extended with a function, which allows a user to upload a new image into a selected MIC21 sub-dataset. When uploaded, the image is automatically processed by the backend service and it is annotated independently by 3 different models (Yolact, Detectron2 and MIC21). The results can be compared in the FiftyOne.

6 Results and Evaluation

The MC21 object detection models produce the output in four components:

- Bounding box, described with its coordinates and its width and height;
- Polygon of points outlining the object contours;
- Class label of the detected object;
- Confidence score between 0 and 1 – how certain the model is about the predicted class.

The FiftyOne framework integrates functionalities to compare different object detection models (in our case MIC21 model outcomes and the ground trough annotations). As each image can represent many objects from different classes, the comparison shows the correct classification for a given object within an image. Overall, the results can be summaries as follows:

- If a model could find the object location, the object is assigned a class;
- The classes that have strong correlation with the COCO classes (i.e. *baseball player* vs. *human*; *soldier* vs. *human*, etc.) are recognized with a better precision, over 90 %, such classes represent 27 % from the MIC21 Ontology classes;
- Classes representing objects that are not categorised in the COCO dataset are recognized and classified with accuracy over 50 %. Figure 3 represents details for four randomly selected sub-datasets.

Category	**Accur.**	**Precis.**	**Recall**	**Support**
Sport	0.60	0.83	0.63	1663
Transport	0.59	0.84	0.63	1421
Arts	0.59	0.89	0.63	855
Security	0.20	0.39	0.24	1973
Total	0.5	0.76	0.53	1478

Table 2: Average metrics for the 130 MIC21 models

The results depend on the selected model parameters, number of training epochs, batch size and also on the structure of the train and the validation datasets. In our experiment, an initial training is performed with a fixed number of 1500 epochs. The resulting models can be re-trained further by the code templates provided within the framework. After the initial training, we have calculated average accuracy, precision and recall metrics for each Ontology class represented in the MIC21 dataset. The low accuracy and recall for some classes from the domains Security and Arts is due to the small number of the ground truth instances. Methods for training models with small datasets have already being developed (Chen et al., 2021; Hu et al., 2020).

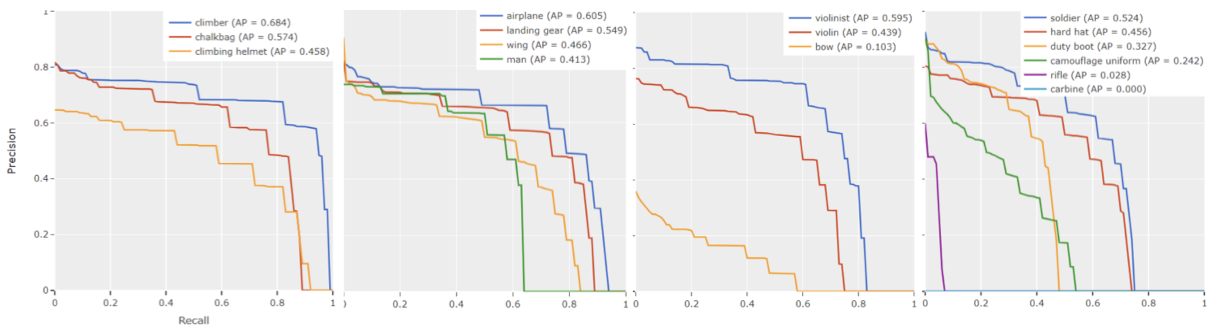

Figure 3: Precision-recall relationship for 4 randomly selected sub-datasets. From left to right: Climbing, Airplane, Violonist and Soldier

During the training we have used a fixed number of 1 500 epochs to generate results for the 130 models in reasonable time frame. However, training for a fixed number of iterations leads to a different performance of the model over the dataset. We summarize the results from the evaluation of the trained models over the target domains in figures 4 to 7. For this purpose we calculate average precision and recall metrics using the official COCO API library (Dollar and Lin, 2014). The library provides the class `COCOEval`, which takes ground truth and model detection arrays as inputs and evaluates each image and category in the dataset over specified surface area ranges. The matching between ground truth and detection masks is determined by a range of intersection-of-unions thresholds (IoT). In our evaluation scenario, we evaluate all area ranges, and the IoT range from 0.5 to 0.95 for both precision and recall. The per image metrics are then accumulated for the whole dataset.

To highlight the capabilities of the trained models, we have performed evaluation over a single class per dataset, which was selected as a dominant class for the particular subdomain. These classes are usually with high number of instances in the subdomain. For some of the subdomains the selection of a dominant class can be ambiguous. However, the main rule during that decision process was that the dominant class must uniquely identify the respective subdomain. Another rule we have observed was to use different sets of classes for different subdomains (to the possible extend). The resultant sets of classes can be tracked in the Figures 4 to 7.

In the Figure 4 we compare the subdomains from the domain Sport. We can see that for the most of the dominant classes we have reached average precision and recall of about 0.4. The highest average precision is reached for the category **Golf player** in the subdomain **Golf**, and the lowest precision and recall are for the class **Race driver** in the subdomain **Car racing** (not a dominant class), which is due to uncommon for COCO models pose of the object **Person** within the images and provided limited training epochs. Hence, if we deviate more from what is typical of the initial training of the model, we have to perform deeper changes in the layers with the additional training in order to preserve the level of fit to ground truth. It is noticeable that we have big difference between average precision and recall for some classes such as **Volleyball player**, **Soccer player**, **Hockey player**, **Cricketer**, etc. In all cases, the recall is about 30% lower than the precision, meaning that when the model detects the respective instance it is correctly classified. However, not all objects from a particular class have been detected. This can be related to how the ground truth objects are selected, in terms of a sufficient number of images depicting the object in a particular situation, or can be attributed to overlapping between two objects in some situations. Such events can lead to lower confidence score from the model. We evaluate all MIC21 models for a confidence score of 0.9, which is quite high, to highlight the differences between the models.

Figure 5 contains the results of evaluation over the domain Transport. The pre-trained models have reached a performance between 0.4 and 0.8 for average precision and recall for that domain. The highest result on precision is for the subdomain **Tram** of about 0.85 and the highest recall is for the subdomain **Convertible** of about 0.84. It is interesting to note that, while for the Sport subdomains we always have higher precision than recall, for the Transport subdomains we have many cases when recall is higher than the precision. This indicates that, while selected models are better at recognizing automobiles, it is more difficult to identify them than people. The lowest metrics for Transport are for the subdomain **Car transporter**, which can also be attributed to the untypical objects we try to identify using a model that was first trained for the 80 COCO classes.

Results from the evaluation of the models in the domain Arts are presented in the Figure 6. With some models targeting higher and some models aiming lower, the average precision and recall are about 0.4-0.5. We cannot see a precision over recall dominance as in Sport because the dominant class in those subdomains is once again Person. With recall and precision close to 0.85 and 0.83, the subdomain **Photographer** achieves the higher metrics.

Interesting finding is that as with Cellists, Ballet dancers, and Percussionists, lower recall levels unnecessarily reduce precision levels. In other words, if the model is good in detecting a particular object, it has good chances to properly classify it. This can be related to the fact that we modify only the ROI head sub-component of the model, without targeting the lower detection layers. In our dataset we have a few classes from the domain Security visualized in the Figure 7. Classes from the domain Security show similar metrics with other examined domains with both precision and recall ranging around 0.5.

7 Conclusion and Future Work

The Multilingual Image Corpus offers data to train models specialized in object detection, segmentation, and classification by providing fully annotated objects within images with segmentation masks, categorised according to an Ontology of Visual Objects. The Ontology of visual objects allows easy integration of annotated images in different datasets as well as learning the associations between objects in images.

Models trained on the COCO dataset were used for the generation of annotation proposals. We developed a framework based on FiftyOne, Yolact and Detectron2, and implemented it over Mask R-CNN on Python3, Keras and TensorFlow. We pre-trained Fast R-CNN models with the MIC 21 dataset, which resulted in 130 models that generate bounding boxes, segmentation masks and object classes.

The MIC 21 framework supports web-based visualization, evaluation and comparison of different models together with the ground truth annotations.

We can provide a number of alternatives for completing multimodal tasks using the created datasets, including automatic image caption generation aligning sentences with images in various multimodal documents and visual question answering. Interpreting an image and the brief text that goes with it, such as a caption, a question or a description of the objects in the image, can be a supporting task.

Prospective developments also include: automatic extension of the dataset using the pre-trained models, which will considerably accelerate manual annotation in the target thematic domains; training models for automatic image captioning (Li et al., 2020) or question answering (Wu et al., 2021; Liu et al., 2021); adaptation of the pre-trained models for video processing (Zhao et al., 2021); identification (automatic generation) of images representing particular objects or particular textual descriptions; application in motion analysis systems.

Acknowledgments

The Multilingual Image Corpus (MIC21) project was supported by the European Language Grid project through its open call for pilot projects. The European Language Grid project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 825627 (ELG).

References

Mark Amo-Boateng, Nana Ekow Nkwa Sey, Amprofi Ampah Amproche, and Martin Kyereh Domfeh. 2022. Instance segmentation scheme for roofs in rural areas based on Mask R-CNN. *The Egyptian Journal of Remote Sensing and Space Science*.

Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. 2019. Yolact: Real-time instance segmentation. In *ICCV*.

Justin Brooks. 2019. COCO Annotator.

James A.D. Cameron, Patrick Savoie, Mary E. Kaye, and Erik J. Scheme. 2019. Design considerations for the processing system of a CNN-based automated surveillance system. *Expert Systems with Applications*, 136:105–114.

Tingkai Chen, Ning Wang, Rongfeng Wang, Hong Zhao, and Guichen Zhang. 2021. One-stage CNN detector-based benthonic organisms detection with limited training dataset. *Neural Networks*, 144:247–259.

Christopher R. Conrady, Şebnem Er, Colin G. Attwood, Leslie A. Roberson, and Lauren de Vos. 2022. Automated detection and classification of southern African Roman seabream using mask R-CNN. *Ecological Informatics*, 69:101593.

Fan Cui, Muwei Ning, Jiawei Shen, and Xincheng Shu. 2022. Automatic recognition and tracking of highway layer-interface using faster R-CNN. *Journal of Applied Geophysics*, 196:104477.

Piotr Dollar and Tsung-Yi Lin. 2014. Microsoft COCO Toolbox. Version 2.0.

Christiane Fellbaum, editor. 1999. *WordNet: an Electronic Lexical Database*. MIT Press, Cambridge, MA.

Ross Girshick. 2015. Fast R-CNN.

Xiaodong Hu, Xinqing Wang, Fan-jie Meng, Xia Hua, Yu-ji Yan, Yu-yang Li, Jing Huang, and Xue-mei Jiang. 2020. Gabor-CNN for object detection based on small samples. *Defence Technology*, 16:1116–1129.

Ahmed Kasapbaşi, Ahmed Eltayeb Ahmed Slbushra, Omar Al-Hardanee, and Arif Yilmaz. 2022. DeepASLR: A CNN based human computer interface for American Sign Language recognition for hearing-impaired individuals. *Computer Methods and Programs in Biomedicine Update*, 2:100048.

Svetla Koeva. 2021. Multilingual Image Corpus: Annotation Protocol. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 701–707, Held Online. INCOMA Ltd.

Svetla Koeva, Ivelina Stoyanova, and Jordan Kralev. 2022. Multilingual Image Corpus – Towards a Multimodal and Multilingual Dataset. In *Proceedings of the Language Resources and Evaluation Conference*, pages 1509–1518, Marseille, France. European Language Resources Association.

Ruifan Li, Haoyu Liang, Yihui Shi, Fangxiang Feng, and Xiaojie Wang. 2020. Dual-CNN: A Convolutional language decoder for paragraph image captioning. *Neurocomputing*, 396:92–101.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. 2014. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision (ECCV)*, pages 740–755, Zürich.

Yun Liu, Xiaoming Zhang, Qianyun Zhang, Chaozhuo Li, Feiran Huang, Xianghong Tang, and Zhoujun Li. 2021. Dual self-attention with co-attention networks for visual question answering. *Pattern Recognition*, 117:107956.

Umberto Michelucci. 2019. *Advanced Applied Deep Learning: Convolutional Neural Networks and Object Detection*. Apress.

George A. Miller, Richard Beckwith, Christiane. Fellbaum, Derek Gross, and Katherine Miller. 1990. Introduction to Wordnet: an on-line lexical database. *International journal of lexicography*, 3(4):235–244.

Brian Moore and Jason Corso. 2020. Fiftyone. *GitHub*.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2015. You Only Look Once: Unified, Real-time Object Detection.

Yuxin Sun, Li Su, Yongkang Luo, Hao Meng, Wanyi Li, Zhi Zhang, Peng Wang, and Wen Zhang. 2022. Global Mask R-CNN for marine ship instance segmentation. *Neurocomputing*, 480:257–270.

Yirui Wu, Yuntao Ma, and Shaohua Wan. 2021. Multi-scale relation reasoning for multi-modal Visual Question Answering. *Signal Processing: Image Communication*, 96:116319.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. Detectron2.

Guoping Zhao, Mingyu Zhang, Yaxian Li, Jiajun Liu, Bingqing Zhang, and Ji-Rong Wen. 2021. Pyramid regional graph representation learning for content-based video retrieval. *Information Processing & Management*, 58(3):102488.

Appendix A Evaluation over the classes in the four main domains: Sport, Transport, Arts, Security

.

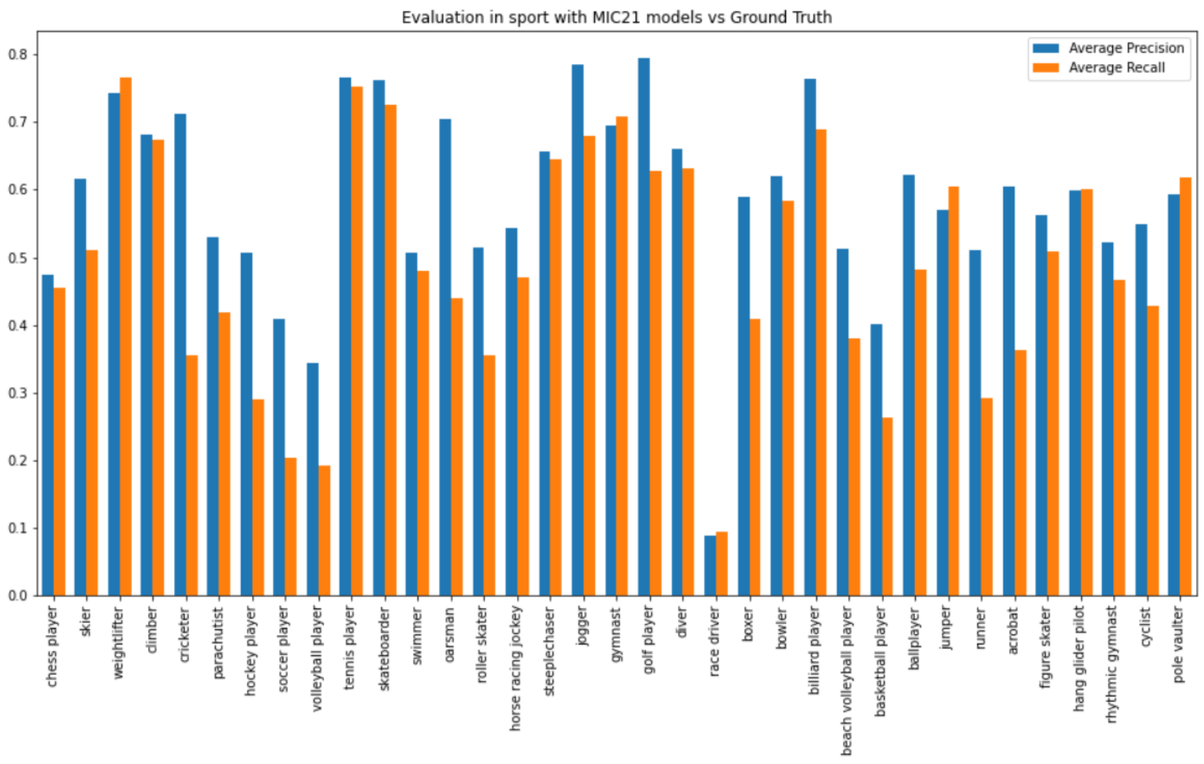

Figure 4: Evaluation over Sport categories

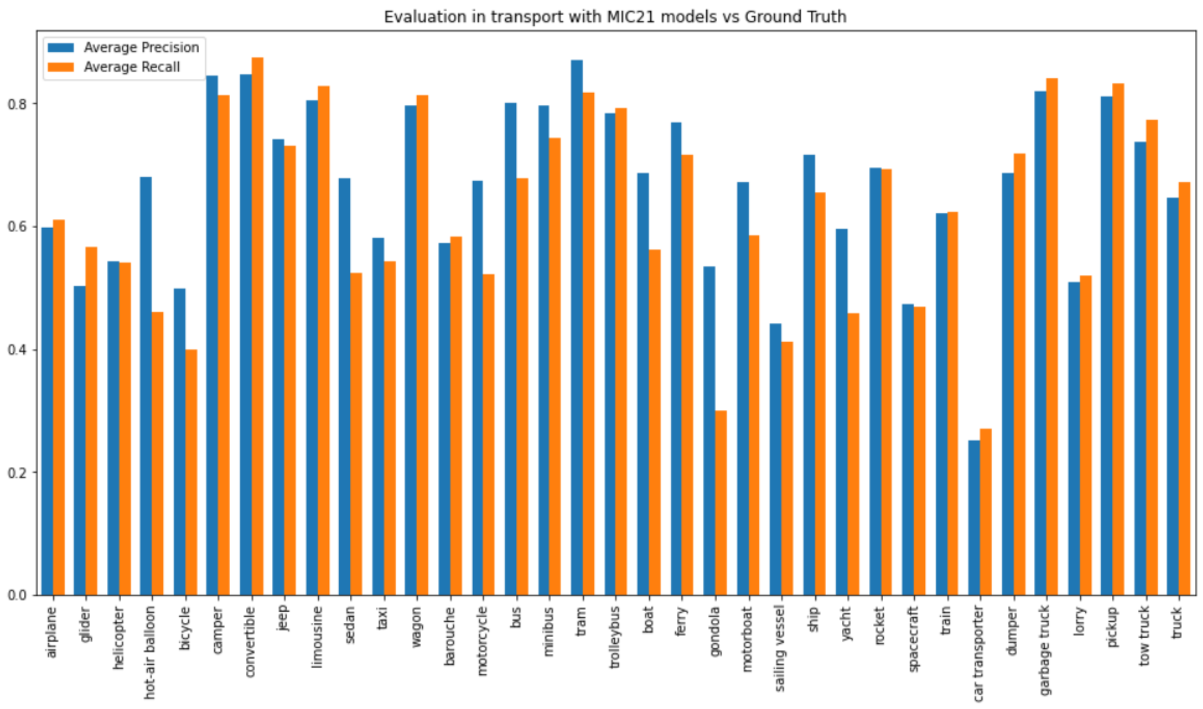

Figure 5: Evaluation over Transport categories

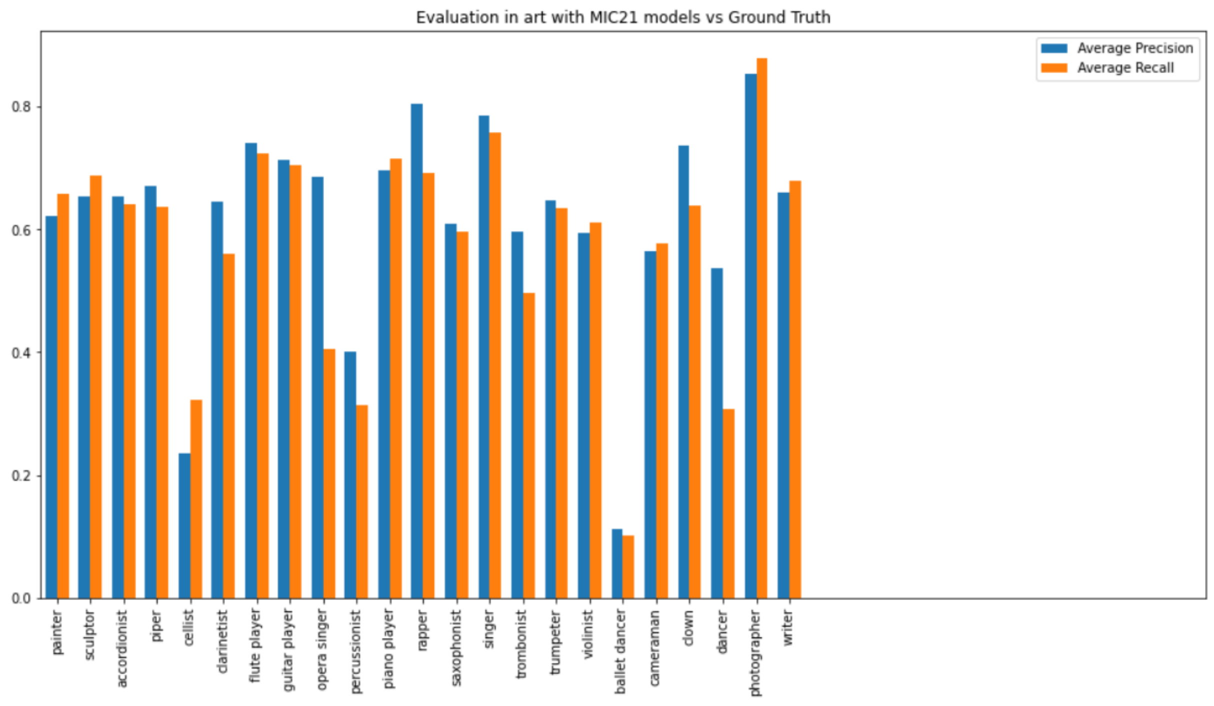

Figure 6: Evaluation over Arts categories

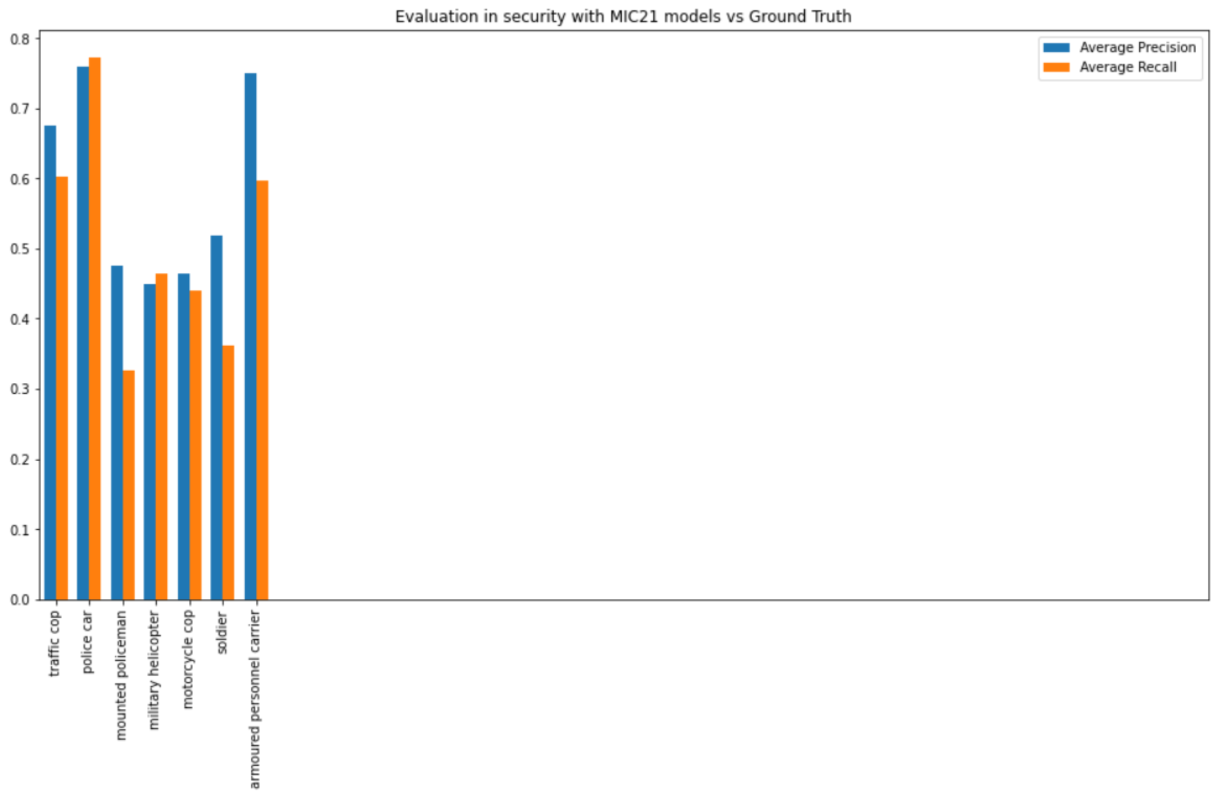

Figure 7: Evaluation over Security categories