Reproducibility Companion Paper: Instance of Interest Detection

Fan Yu[1,3], Dandan Wang[1], Haonan Wang[1], Tongwei Ren[1,3,*]
Jinhui Tang[2], Gangshan Wu[1], Jingjing Chen[4], Michael Riegler[5]
[1]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
[2]School of Computer Science, Nanjing University of Science and Technology, Nanjing, China
[3]Shenzhen Research Institute of Nanjing University, Shenzhen, China
[4]Shanghai Key Lab of Intelligent Information Processing, School of Computer Science, Fudan University, Shanghai, China
[5]SimulaMet, Norway
yf@smail.nju.edu.cn,hedda_stone@hotmail.com,527725618@qq.com,rentw@nju.edu.cn
jinhuitang@njust.edu.cn,gswu@nju.edu.cn,chenjingjing.tju@gmail.com,michael@simula.no

ABSTRACT

To support the replication of "Instance of Interest Detection", which was presented at MM'19, this companion paper provides the details of the artifacts. Instance of Interest Detection (IOID) aims to provide instance-level user interest modeling for image semantic description. In this paper, we explain the file structure of the source code and publish the details of our IOID dataset, which can be used to retrain the model with custom parameters. We also provide a program for component analysis to help other researchers to do experiments with alternative models that are not included in our experiments. Moreover, we provide a demo program for using our model easily.

CCS CONCEPTS

• **Computing methodologies** → **Computer vision**.

KEYWORDS

Instance of interest; instance of interest detection; instance of interest annotation; instance extraction; interest estimation

ACM Reference Format:
Fan Yu[1,3], Dandan Wang[1], Haonan Wang[1], Tongwei Ren[1,3,] and Jinhui Tang[2], Gangshan Wu[1], Jingjing Chen[4], Michael Riegler[5]. 2020. Reproducibility Companion Paper: Instance of Interest Detection. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20), October 12–16, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3394171.3414811

*Corresponding author.

MM '20, October 12–16, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3414811

1 ARTIFACTS DESCRIPTION

1.1 Introduction

We proposed a novel task named Instance of Interest Detection (IOID) [9] to provide instance-level user interest modeling for image semantic description. IOID focuses on extracting the instances which are beneficial to represent image content, while other related tasks such as saliency analysis and instance segmentation extract the regions attracting visual attention or with a predefined category. The output result of IOID is named as Instance of Interest (IOI). To this end, we proposed a Cross-influential Network (CIN) for IOID, which integrates both visual saliency and semantic context.

The artifacts include the IOID dataset and the source code of the CIN model. The source code is available at https://github.com/yfraquelle/IOID, and the related download links are provided in the README file.

1.2 Dataset

We construct the first dataset for IOID based on the training set of MSCOCO 2017, which contains manually labelled captions and panoptic segmentation. The IOID dataset contains 45,000 images with IOIs in 133 categories, the same as the panoptic information in MSCOCO 2017, including 80 thing categories, such as person, ball and cow, and 53 stuff categories, such as wall, tree and mountain. The IOID dataset is split into the training set and the test set that contain 36,000 images and 9,000 images, respectively. The link for downloading the IOID dataset is provided in the README file. The data of images and annotations are stored in three json files, "class_dict.json", "train_images_dict.json" and "val_images_dict.json". The "train_images_dict.json" file and the "val_images_dict.json" file are in the same format and an example is shown in Figure 1(a). The format of the "class_dict.json" file is shown in Figure 1(b). It should be noted that the "category_id" is not contiguous in the category information provided by MSCOCO. Thus we give a contiguous "class_id" to each category.

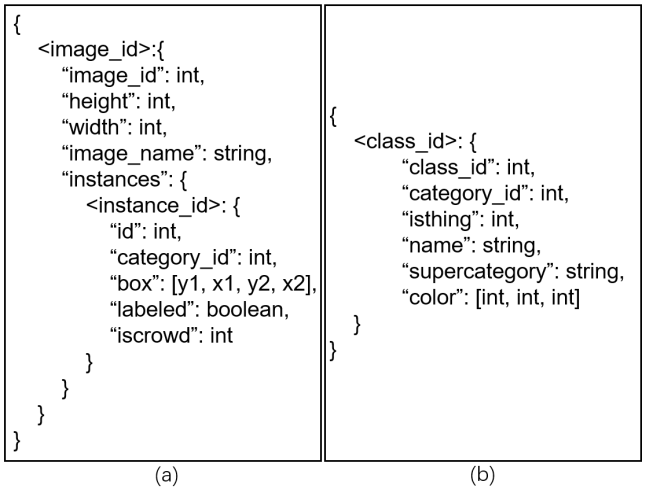

Figure 1: Data format in the json files. (a) The format of the "train_images_dict.json" and "val_images_dict.json" file, where the <image_id>, which represents the id of an image, and the <instance_id>, which represents the id of an instance, are both string values. (b) The format of the "class_dict.json" file, where the <class_id>, representing the "class_id" of an object category, is a string value.

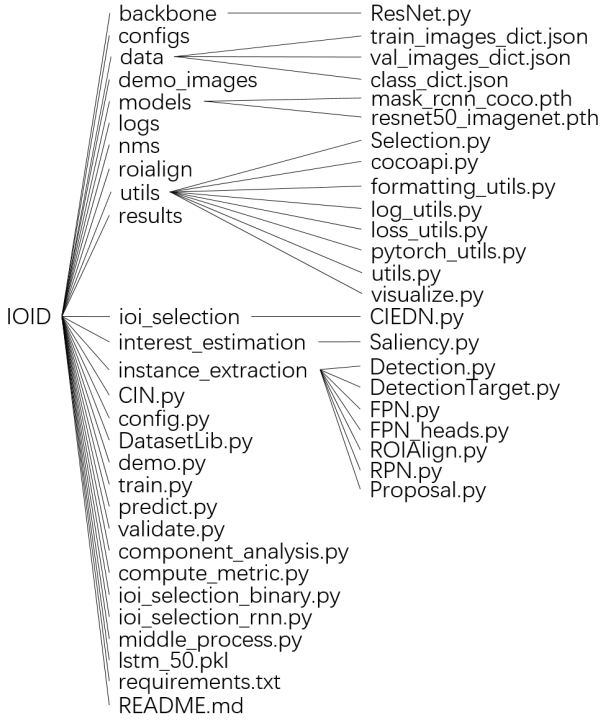

Figure 2: File structure.

1.3 Source code structure

The file structure of the source code is shown in the Figure 2. Our work is based on the implement of Mask RCNN at https://github.com/multimodallearning/pytorch-mask-rcnn and PiCANet at https://github.com/Ugness/PiCANet-Implementation.

backbone: containing the backbone of the CIN model.

configs: containing the configuration files.

data: containing the json files of the IOID dataset.

demo_images: containing some images for testing and visualization.

models: containing some pretrained models.

logs: saving model parameters during training.

nms: containing files for non-maximum suppression.

roialign: containing files for region of interest alignment.

utils: containing python files for assistance.

results: saving result files generated during testing.

ioi_selection: containing the python file used in the IOI selection module.

interest_estimation: containing the python file used in the interest estimation module.

instance_extraction: containing the python files used in the instance extraction module.

CIN.py: working as the main file for the CIN model.

config.py: working as the configuration file with default values.

DatasetLib.py: loading data for training and validation.

demo.py: working as the main file for testing and visualization.

train.py: working as the main file for training the CIN model.

predict.py: working as the main file for predicting the final or intermediate results based on the CIN model.

validate.py: working as the main file for evaluating the performance of the CIN model.

component_analysis.py: working as the main file for component analysis.

compute_metric.py: computing metrics.

ioi_selection_binary.py: implementing a simple model as a variant of the IOI selection module in the CIN model.

ioi_selection_rnn.py: implementing an rnn model as a variant of the IOI selection module in the CIN model.

middle_process.py: implementing data processing for component analysis.

lstm_50.pkl: saving the parameters of the pretrained model for the "ioi_selection_rnn.py".

requirements.txt: listing the python dependencies of the code.

README.md: working as the description for the source code and related information.

The CIN model is mainly built in the "CIN.py" file, using the "ResNet.py" file in the "backbone" folder as the backbone, and the files in the "ioi_selection", "interest_estimation" and "instance_extraction" folders work as the three modules in our CIN model. The files in the "instance_extraction" folder depend on the files in the folders named "nms" and "roialign". The "train.py", "predict.py", "demo.py" and

"validate.py" files perform training, inference or evaluation mainly depending on the "CIN.py" file. The "component_analysis.py", "ioi_selection_binary.py" and "ioi_selection_rnn.py" files are used for component analysis. Before component analysis, the "predict.py" and "middle_process.py" should be performed to generate required data.

2 EXPERIMENTS

2.1 Environment Installation

Our source code is tested in the following environment.

(1) Operating system Ubuntu 16.04 LTS with CPU i7-8086K, GPU TITAN V, 64GB memory and 1TB free space.

(2)CUDA 9.0 and cuDNN 7.0.

(3)Python 3.5.6 with opencv_python==3.4.3.18, numpy==1.16.2, scikit_image==0.14.2, torchvision==0.2.1, torch==0.4.1, scipy==1.1.0, matplotlib==3.0.0, Pillow==7.0.0, skimage==0.0, tensorboardX==2.0, PyYAML==3.13 and cffi==1.12.2.

We provide a docker image for all software dependencies, which is available for download in the README file.

2.2 Test on the pretrained model

To visualize the result of the instance of interest detection, the "demo.py" file can be performed with the following script:

```
python demo.py --img <image path> --config
<configuration file path>
```

The README file provides the download link to the pretrained parameters of our CIN model. An example of the visualization result is shown in Figure 3

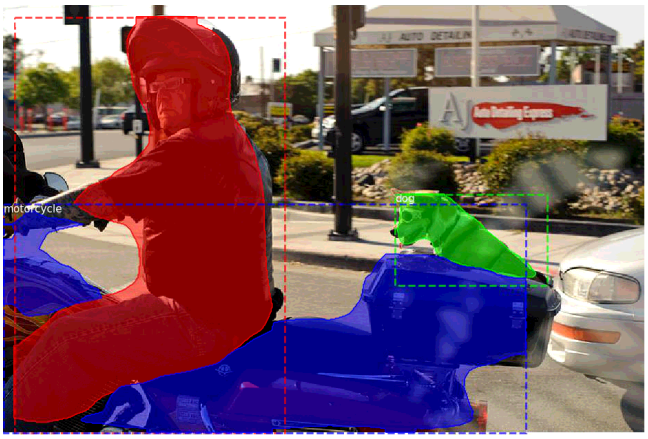

Figure 3: An Example of the visualization result of "demo.py".

2.3 Custom training and inference

The model parameters defined in the "config.py" file can be adjusted by a custom configuration file, and some important parameters as well as their description are shown in Table 1.

To train the CIN model, we use the pretrained Mask R-CNN model, which works well for the instance segmentation task. We divide the training process into four steps: "semantic" (stuff extraction), "p_interest" (interest estimation), "selection" (IOI selection) and "all" (the whole model). The first three steps train the parameters in different modules with the same backbone parameters. These steps can be performed in custom order, but we suggest to perform them in the recommended order. To specify the training order and the related attributes, we use a sequence as a parameter, containing the training steps with the corresponding learning rate and the number of epochs, such as "semantic,0.01,34,p_interest,0.01,44,selection,0.01,100". The "train.py" file can be performed with the following script:

```
python train.py --setting <setting sequence>
--config <configuration file path>
```

During inference, the three modules can also be separated and generate the corresponding intermediate results. In this case, the three modules can be replaced with other models for component analysis. To extract the intermediate results of different modules in the CIN model, the "predict.py" file can be performed with the following script in a custom mode: "instance" (generating results of instance extraction), "p_interest" (generating results of interest estimation), "insttr" (generating results of instance extraction and interest estimation) or "selection" (generating results of IOI selection):

```
python predict.py --mode <mode> --subset
<performing on which dataset> --config
<configuration file path>
```

2.4 Experiments in the original paper

The "validate.py" file can be performed with the following script to evaluate the performance of the CIN model:

```
python validate.py --config <configuration
file path>
```

For component analysis, the three modules can be replaced with some related models. Taking advantage of the flexible intermediate results extraction, the outputs of the instance extraction module and the interest estimation module can be extracted and work as the input of the alternative to the IOI selection module. Meanwhile, the results of these two modules can also be replaced with the results of the related models to work as the inputs of the IOI selection module. We use the "component_analysis.py" file to perform component analysis. The folder name of the panoptic segmentation results, the semantic segmentation results and the interest estimation results should be provided. Those results should be similar to the intermediate results of the first two modules. We provide the intermediate results generated by some related models [1–8] in our experiments, and the download link is provided in the README file. As the alternative to the IOI selection module, two simple models are implemented in the "ioi_selection_binary.py" file and the "ioi_selection_rnn.py" file, and their alias "binary" and "rnn" work as the optional values for the "sel_ext" parameter.

```
python component_analysis.py --ins_ext
```

Table 1: Important parameters that can be customized.

Parameter	Description	Default Value
GPU_COUNT	The number of GPUs.	1
IMAGES_PER_GPU	The number of images to train with on each GPU.	1
STEPS_PER_EPOCH	The number of training steps per epoch.	1000
NUM_CLASSES	The number of classification classes (including background).	134
LEARNING_RATE	Learning rate.	0.001
LEARNING_MOMENTUM	Learning momentum.	0.9
IMAGE_PATH	The path of the images related files.	../data/
JSON_PATH	The path of the json files.	data
WEIGHT_PATH	The path of the default model weights.	models/CIN_ooi_all.pth
IMAGE_SIZE	The size of image after resizing and padding.	1024
MAP_IOU	The iou threshold when mapping prediction to the ground truth.	0.5
STUFF_THRESHOLD	The threshold when filtering small stuff.	1000
THING_NUM_CLASSES	The number of things (including background).	81
STUFF_NUM_CLASSES	The number of stuff.	53
SELECTION_THRESHOLD	The threshold when selecting IOIs.	0.4

```
<panoptic segmentation path> --sem_ext
<semantic segmentation path> --p_intr_ext
<interest estimation path> --sel_ext
<IOI selection method> --config
<configuration file path>
```

3 REPRODUCIBILITY EFFORTS

In the original paper, the connections among different files were not clearly described, and when the reviewers tried to run the scripts for the experiments, they found that some of the software dependencies such as "PyYAML" and "cffi" were missing in the given list and the "import" sentences were not updated in the original code. The authors quickly fixed the problems by offering a detailed explanation at the end of the Section 1.3. The missing dependencies also have been added in the Section 2.1 and the README file in the code repository has been updated. Moreover, a docker image containing all the software dependencies has been provided in response to the reviewers' advice, making it more feasible for researchers in the area.

The reviewers acknowledge the efforts of the original authors to provide necessary corrections during the reviewing process, including simplifying the format of scripts parameters and fixing minor bugs in the code, as well as careful proofreading.

In conclusion, two reviewers and the authors worked together for this companion paper. The revised code now enables third-party researchers to reproduce the experiments in the original paper and is customizable for further research on visual relation detection.

4 CONCLUSION

In this paper, we provided the details of the artifacts of the paper "Instance of Interest Detection" for replication. The artifacts contain the IOID dataset and the source code for experiments in the paper. Taking advantage of the source code, the experiments can be operated and customized.

ACKNOWLEDGEMENT

This work is supported by Natural Science Foundation of Jiangsu Province (BK20191248), National Science Foundation of China (61732007), Science, Technology and Innovation Commission of Shenzhen Municipality (JCYJ20180307151516166), and Collaborative Innovation Center of Novel Software Technology and Industrialization.

REFERENCES

[1] Liangchieh Chen, George Papandreou, Iasonas Kokkinos, Kevin P Murphy, and Alan L Yuille. 2018. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets and Atrous Convolution and and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018), 843–848.

[2] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross B Girshick. 2017. Mask R-CNN. IEEE Conference on Computer Vision and Pattern Recognition.

[3] Qibin Hou, Mingming Cheng, Xiaowei Hu, Ali Borji, Zhuowen Tu, and Philip H S Torr. 2017. Deeply Supervised Salient Object Detection with Short Connections. IEEE Conference on Computer Vision and Pattern Recognition.

[4] Guanbin Li, Yuan Xie, Liang Lin, and Yizhou Yu. 2017. Instance-Level Salient Object Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[5] Nian Liu, Junwei Han, and Minghsuan Yang. 2018. PiCANet: Learning Pixel-Wise Contextual Attention for Saliency Detection. IEEE Conference on Computer Vision and Pattern Recognition.

[6] Zhiming Luo, Akshaya Kumar Mishra, Andrew Achkar, Justin A Eichel, Shaozi Li, and Pierremarc Jodoin. 2017. Non-local Deep Features for Salient Object Detection. IEEE Conference on Computer Vision and Pattern Recognition.

[7] Junting Pan, Cristian Cantonferrer, Kevin Mcguinness, Noel E Oconnor, Jordi Torres, Elisa Sayrol, and Xavier Giro I Nieto. 2017. SalGAN: Visual Saliency Prediction with Generative Adversarial Networks. IEEE Conference on Computer Vision and Pattern Recognition.

[8] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. International Conference on Machine Learning.

[9] Fan Yu, Haonan Wang, Tongwei Ren, Jinhui Tang, and Gangshan Wu. 2019. Instance of Interest Detection. ACM International Conference on Multimedia.