Reproducibility Companion Paper: Human Object Interaction Detection via Multi-level Conditioned Network

Yunqing He
State Key Laboratory for Novel Software Technology, Nanjing University
Nanjing, China
heyq@smail.nju.edu.cn

Xu Sun
State Key Laboratory for Novel Software Technology, Nanjing University
Nanjing, China
sunx@smail.nju.edu.cn

Hui Jiang
State Key Laboratory for Novel Software Technology, Nanjing University
Nanjing, China
huijiang@smail.nju.edu.cn

Tongwei Ren*
State Key Laboratory for Novel Software Technology, Nanjing University
Nanjing, China
rentw@nju.edu.cn

Gangshan Wu
State Key Laboratory for Novel Software Technology, Nanjing University
Nanjing, China
gswu@nju.edu.cn

Maria Sinziiana Astefanoaei†
Data-intensive Systems and Applications, IT University of Copenhagen
Copenhagen, Denmark
msia@itu.dk

Andreas Leibetseder†
Institute of Information Technology, Klagenfurt University
Klagenfurt, Austria
aleibets@itec.aau.at

ABSTRACT

To support the replication of "Human Object Interaction Detection via Multi-level Conditioned Network", which was presented at ICMR'20, this companion paper provides the details of the artifacts. Human Object Interaction Detection (HOID) aims to recognize fine-grained object-specific human actions, which demands the capabilities of both visual perception and reasoning. In this paper, we explain the file structure of the source code and publish the details of our experiments settings. We also provide a program for component analysis to assist other researchers with experiments on alternative models that are not included in our experiments. Moreover, we provide a demo program for facilitating the use of our model.

CCS CONCEPTS

• **Computing methodologies** → **Activity recognition and understanding**.

KEYWORDS

Human object interaction detection, conditioned network, multilevel visual representation, multimodal feature fusion, feature transformation

*Corresponding Author.
†Reproducibility Reviewers.

1 ARTIFACTS DESCRIPTION

1.1 Introduction

In our original paper [10], we proposed a novel multi-level conditioned network (MLCNet) that fuses extra spatial-semantic knowledge with visual features to enhance the reasoning capability of human object interaction detection (HOID) [9]. Specifically, we constructed a multi-branch CNN as the backbone for multi-level visual representation. We then encoded extra knowledge including human body structure and object context to dynamically influence the feature extraction of CNN by affine transformation and attention mechanism [11]. Finally, we fused the modulated multimodal features to distinguish the interactions. We trained and evaluated our method on HICO-DET dataset [1] and V-COCO dataset [5]. The artifacts include the source code of the MLCNet model, which is available at https://github.com/fraliphsoft/HOI-det.

1.2 Source Code Structure

The file structure of the source code is shown in Figure 1. Considering that there are mass of dependency files in our project, *e.g.*, pycocotools, only the important files that we modified or necessary for code running are listed here. Our work is based on the pytorch implementation of Faster R-CNN [8, 12] at https://github.com/jwyang/faster-rcnn.pytorch and a recently proposed HOID method TIN [6] at https://github.com/DirtyHarryLYL/Transferable-Interactiveness-Network. We use WSHP [3] at https://github.com/MVIG-SJTU/WSHP to locate human body part regions.
data: containing HICO-DET dataset and V-COCO dataset.
eval_hico: a toolkit for evaluation on HICO-DET dataset.
eval_vcoco: a toolkit for evaluation on the V-COCO dataset.
lib\dataset\hico2.py: pre-processing data from HICO-DET.

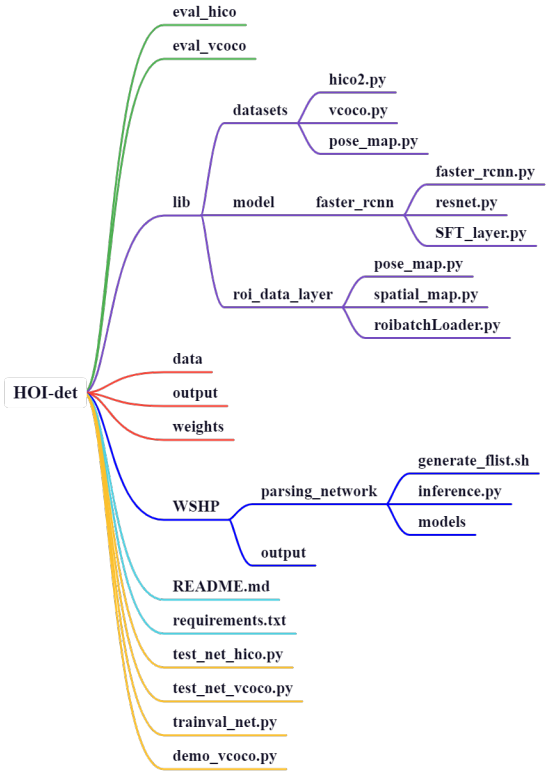

Figure 1: File structure.

lib\dataset\vcoco.py: pre-processing data from V-COCO.
lib\model\faster_rcnn\faster_rcnn.py: working as the main file to define the structure of our model.
lib\model\faster_rcnn\resnet.py: containing the backbone of our model.
lib\model\faster_rcnn\SFT_layer.py: working as the main file for the feature transform block.
lib\roi_data_layer\pose_map.py: generating body part mas-ks as input for phrase and spatial branch in our method.
lib\roi_data_layer\spatial_map.py: generating spatial feature as input for phrase and spatial branch in our method.
lib\roi_data_layer\roibatchLoader.py: working as the data loader for two datasets.
weights: containing pretrained models in HICO-DET data-set and V-COCO dataset, respectively.
WSHP\parsing_network\generate_flist.sh: generating filenames list from datasets.
WSHP\parsing_network\inference.py: generating human body part regions from datasets.
WSHP\models: containing pretrained models of WSHP.
test_net_hico.py: working as the main file to predict and evaluate results on the basis of our model in HICO-DET.
test_net_vcoco.py: working as the main file to predict and evaluate results on the basis of our model in V-COCO.
trainval_net.py: working as the main file to train our model.
demo_vcoco.py: working as a demo program in V-COCO.

2 EXPERIMENTS

2.1 Datasets

We use the same datasets as in our original paper, and also the same training and test splits. Here, we describe the two datasets in turn.

HICO-DET is constructed by augmenting HICO dataset [2] with instance annotations. It includes 47,776 images (38,118 for training and 9,658 for testing) with 80 object and 117 action categories that form 600 HOI categories. Over 150K HOI instances are provided by HICO-DET.

V-COCO is constructed by augmenting a subset of MS-COCO dataset [7] with interaction category annotations. It includes 10,346 images (2,533 for training and 2,867 for validation and 4,946 for testing) with 26 HOI categories. Over 16K HOI instances are provided by V-COCO.

2.2 Experiments Settings

We conduct our experiments in the following environment:

(1) Ubuntu 16.04 LTS with CPU i7-5930K, GPU GTX 1080 Ti, 32GB memory and 500GB free space.

(2) CUDA 8.0 and cuDNN 6.0.21.

(3) Python 2.7 with opencv-python=4.2.0.32, numpy=1. 16.6, torch=0.4.0, tensorboardX=2.0, tensorflow=1.1.0, matplotlib=2.2.5, scipy=1.2.3, easydict=1.9, Pillow=6.2.2.

(4) Matlab R2017a.

For the support of the latest runtime environment, we also reproduce our experiments in a renewed branch, which requires the following environment:

(1) Ubuntu 16.04 LTS with CPU Intel(R) Xeon(R) E5-2680 v4 @ 2.40GHz, GPU GeForce RTX 3090 @ 24GB, 64GB memory and 500GB free space.

(2) CUDA 11.1 and cuDNN 8.0.5.

(3) Python 3.8 with opencv-python=4.5.5.62, numpy=1. 21.5, torch=1.9.0, tensorboardX=2.4, tensorflow=2.6.0, matplotlib=3.5.1, scipy=1.6.2, easydict=1.9, Pillow=8.4.0.

(4) Matlab R2017a.

Our source code is composed of python. For fair comparison, we follow the same code as in iCAN [4] and TIN [6], instead of making new implementation of the original evaluation metrics. In this case, Matlab runtime environment is needed. The inherited evaluation functions are stored in 'eval_hico' and 'eval_vcoco' directories respectively.

2.3 Quick Start

We provide a demo program to facilitate the practice of our method. As shown in Figure 2, the visualization results can be obtained by running the following script:

```
python demo_vcoco.py
```

The '*–im_id*' parameter is also customized to choose an image and the '*–show_category*' parameter is used to decide whether to show interaction and object categories or not.

2.4 Test and Evaluation

The "test_net_hico.py" and "test_net_vcoco.py" file can be implemented with the following script to evaluate the performance of our model:

Table 1: Important parameters that can be customized.

Parameter	Description	Default Value
dataset	choose the dataset to use.(“hico_full" or “vcoco_full")	“vcoco_full"
net	choose the backbone to use.(“res101" or “vgg16")	“res101"
start_epoch	number of start epoch.	1
epochs	number of epochs to train.	6
disp_interval	number of iterations to display.	100
save_dir	directory to save models.	weights
mGPUs	whether to use multiple GPUs.	False
bs	batch_size.	1
o	training optimizer.	sgd
lr	starting learning rate.	0.00001
lr_decay_step	step to do learning rate decay, unit is epoch.	1
lr_decay_gamma	learning rate decay ratio.	0.1
r	resume checkpoint or not.	False
checksession	checksession to load model.	1
checkepoch	checkepoch to load model.	6
checkpoint	checkpoint to load model.	91451
use_tfb	whether use tensorboard.	True

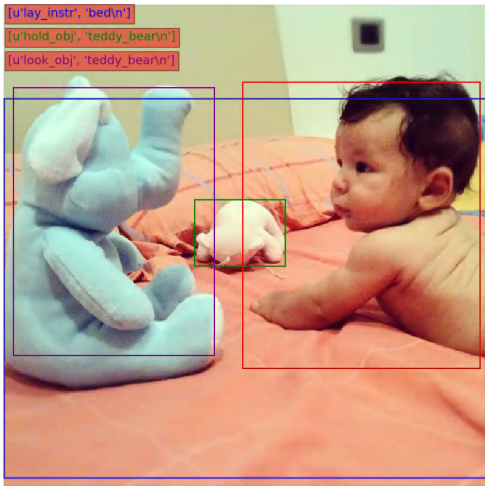

Figure 2: An example of the visualization result.

python test_net_hico.py

or

python test_net_vcoco.py

In order to reduce the time in evaluation, once the inference finished, the test results will be saved to ‘output’ directory. We also provide our pre-computed test results with links by the README.md file in our code.

The evaluation results will be saved to a text file. For HICO-DET dataset, the file is ‘./output/hico_full/eval_result.txt’. For V-COCO dataset, the file is ‘./output/vcoco_full/all_hoi_detections_eval.txt’, and the ‘role mAP’ metric illustrated in original paper is shown in the last text line. Following the settings from previous work, we select role mAP in Scenario 2 as the comparable result. It is notable that the experiment results in V-COCO dataset generated by our provided pretrained model are slightly higher than those in the original paper, while we did nothing but simply increase some training epochs.

2.5 Training for Replication

The model parameters defined in the “trainval.py” file can be adjusted for custom training, and some important parameters as well as their description are shown in Table 1, and most of them are inherited from Faster-RCNN. To repeat our training progress, please run:

python trainval_net.py dataset hico_full checkepoch 6 –checkpoint 91451

or

python trainval_net.py dataset vcoco_full checkepoch 18 –checkpoint 10051

It takes about 3 days and 1 day for training on HICO-DET dataset and V-COCO dataset respectively.

There are two backbone networks available for our model, VGG16 and ResNet101. In our experiments, we decide ResNet101 as our backbone network and initialize the parameters with a pretrained ResNet101 model before training. Both pretrained ResNet101 model and VGG16 model are provided with links in the README.md file in our code.

3 REPRODUCIBILITY EFFORTS

In the replication, we reorganize and clean up the code files, and clarify every file that is working in our method. In the clean-up process, about 125 code and dependencies files that are not necessary for code running are removed or simplified. We re-analyze the complete process from the original data to the final results to integrate them into a fine-designed code warehouse. The refactored code can be accessed on github with a pipeline of operating manual. We clean out the running environments and dependencies to reduce the installation costs. We upgrade the codebase dependencies for

the support of the latest runtime environment. Finally, we provide a demo program to visualize our test results.

4 REVIEWING PROCESS

As indicated above, two external reviewers audited the published work in terms of reproducibility. This process was conducted in periodic communication and consultation with the main authors. As the tools utilized for the original research were rather outdated, the review focused on ensuring better compatibility with more modern hardware. The authors accomplished this elegantly by creating a new repository branch `pytorch1.x`, which supports recent versions of implementation-critical Python libraries such as PyTorch and Tensorflow. Moreover, the required library versions are obtainable using the conda package manager of the Anaconda Python distribution.

During the review, several problems arose, mostly concerning the setup of the training and testing environment. In particular, the biggest issue was the dependency on Matlab causing unpredictable complications on different operating systems. Ultimately, this led to using a Linux (Ubuntu 18.04) machine, as recommended by the authors. After a successful setup, the training process for both datasets—HICO and VCOCO—took several days on a workstation with the following hardware specs: Intel Core i7-6800K CPU @ 3.40GHz × 6, 64 GiB DDR4 @ 2666 MHz, Nvidia GeForce RTX 3090.

In summary, all original research experiments can approximately be recreated and the work can thus be rated as reproducible. We further commend the authors for quickly tending to all emerging problems as well as providing extensive explanations and solutions.

5 CONCLUSION

In this paper, we provided the details of the artifacts of the paper "Human Object Interaction Detection via Multi-level Conditioned Network". The artifacts contain the source code for experiments in the paper. Taking advantage of the source code, the experiments can be operated and customized.

ACKNOWLEDGMENTS

This work is supported by National Science Foundation of China (62072232), Natural Science Foundation of Jiangsu Province (BK2019 1248) and Collaborative Innovation Center of Novel Software Technology and Industrialization.

REFERENCES

[1] Yu-Wei. Chao, Yunfan. Liu, Xieyang. Liu, Huayi. Zeng, and Jia. Deng. 2018. Learning to Detect Human-Object Interactions. In *WACV*.
[2] Yu-Wei Chao, Zhan Wang, Yugeng He, Jiaxuan Wang, and Jia Deng. 2015. HICO: A Benchmark for Recognizing Human-Object Interactions in Images. In *ICCV*.
[3] Hao-Shu Fang, Guansong Lu, Xiaolin Fang, Jianwen Xie, Yu-Wing Tai, and Cewu Lu. 2018. Weakly and Semi Supervised Human Body Part Parsing via Pose-Guided Knowledge Transfer. In *CVPR*.
[4] Chen Gao, Yuliang Zou, and Jia-Bin Huang. 2018. iCAN: Instance-Centric Attention Network for Human-Object Interaction Detection. In *BMVC*.
[5] Saurabh Gupta and Jitendra Malik. 2015. Visual Semantic Role Labeling. *CoRR* (2015).
[6] Yong-Lu Li, Siyuan Zhou, Xijie Huang, Liang Xu, Ze Ma, Hao-Shu Fang, Yanfeng Wang, and Cewu Lu. 2019. Transferable Interactiveness Knowledge for Human-Object Interaction Detection. In *CVPR*.
[7] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *ECCV*.
[8] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *NeurIPS*.
[9] Xu Sun, Yunqing He, Tongwei Ren, and Gangshan Wu. 2021. Spatial-Temporal Human-Object Interaction Detection. In *ICME*.
[10] Xu Sun, Xinwen Hu, Tongwei Ren, and Gangshan Wu. 2020. Human Object Interaction Detection via Multi-Level Conditioned Network. In *ICMR*.
[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *NeurIPS* (2017).
[12] Jianwei Yang, Jiasen Lu, Dhruv Batra, and Devi Parikh. 2017. A Faster Pytorch Implementation of Faster R-CNN. *https://github.com/jwyang/faster-rcnn.pytorch* (2017).