Image Recognition System using Generic YOLOv5 Weight Model and Roboflow for Pre-processed Dataset

Suchada Sitjongsataporn[1*] and Kornika Moolpho[2]

[1]Department of Electronic Engineering, School of Electrical and Electronic Engineering (SEE)
Mahanakorn University of Technology, Nongchok, Bangkok, Thailand, ssuchada@mut.ac.th*
[2]Department of Mechatronic Engineering, Mahanakorn Institute of Innovation (MII)
Mahanakorn University of Technology, Nongchok, Bangkok, Thailand, kornika@mut.ac.th

Abstract

This paper presents the generic Yolov5 weight model for image recognition system by using pre-processed dataset. We focus on the YOLOv5 model and Roboflow for case studies. First, YOLOv5 model is used for importance of preprocessing training sets, and examples of COCO in the 128 data set. Second, Roboflow model is presented for pre-processed data. By the utilization of Roboflow, an advanced computer vision platform, to differentiate between raccoons and prairie dogs. The implementation of YOLOv5 and Roboflow in these cases that demonstrate the potential of AI technologies in differentiating image recognition.

Keywords: YOLOv5 model, Roboflow, Image recognition, COCO128 dataset

1. Introduction

Image Classification is a fundamental task in vision recognition that aims to understand and categorize an image as a whole under a specific label [1]. Unlike object detection, which involves classification and location of multiple objects within an image, image classification typically pertains to single-object images.

First of all, Roboflow is a Computer Vision developer framework for better data collection to preprocessing, and model training techniques [2]. Roboflow has public datasets readily available to users and has access for users to upload their own custom data also. Roboflow accepts various annotation formats. In data preprocessing, there are steps involved such as image orientations, resizing, contrasting, and data augmentations. YOLOv5, as one of the most advanced real-time object detection frameworks, is the cornerstone of this project. The YOLO (You Only Look Once) series occupies an important position in the field of object detection with its excellent speed and accuracy [3]. YOLOv5 further optimizes the performance and flexibility of the model, enabling it to process complex image data quickly and accurately, making it suitable for a wide range of practical application scenarios.

Secondly, data preprocessing plays a crucial role in our project. In order to enable the YOLOv5 model to effectively learn and recognize objects, we carefully preprocessed the training data set. This includes steps such as image resizing [4], format standardization [5], data enhancement [6], and more. Through these preprocessing operations, we ensure that the data received by the model reflects the diversity of the real world and meets the technical requirements of the training process.

Finally, the COCO128 dataset plays a key role in the examples. COCO (Common Objects in Context) [7] is a widely used large-scale image data set that contains a variety of object categories and is very suitable for training deep learning models. In our project, we used the COCO128 subset, which is a streamlined version of the COCO dataset and contains 128 selected images and their corresponding annotations. This data set not only provides rich and varied training samples, but also helps us efficiently train and optimize our models with limited resources.

In summary, the efficiency of the YOLOv5 model, coupled with the carefully designed pre-processing process and the diversity of the COCO128 data set, together form the core of this image recognition project. We explore the utilization of Roboflow, an advanced computer vision platform, to differentiate between raccoons and prairie dogs for experiments. The implementation of Roboflow in this case study demonstrates the potential of AI technologies in differentiating different species of animals. Through the combined use of these technologies and resources, we can effectively achieve high-precision image recognition, laying a solid foundation for subsequent application development

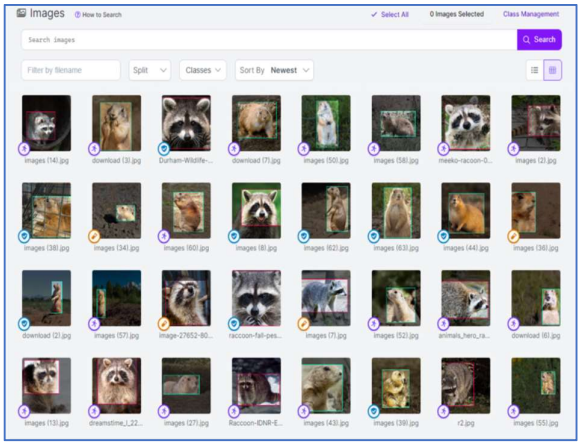

Fig. 1 Roboflow universe [8]

*Corresponding Author

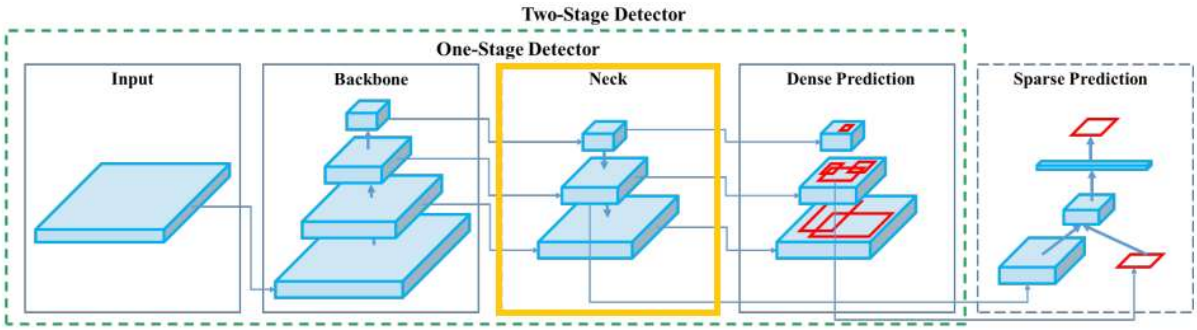

Fig. 2 YOLO architecture [9]

2. Roboflow universe

Roboflow Universe [8] is a repository of more than 110,000 open-source datasets in Fig.1 that you can use in your projects at https://universe.roboflow.com/. All datasets are containing everything from annotated cracks in concrete to plant images with disease annotations. Preparing a custom dataset using Roboflow as follows:

Step 1: Creating project

Before you start, you need to create a Roboflow account. Once you do that, you can create a new project in the Roboflow dashboard. Keep in mind to choose the right project type. In our case, Object Detection.

Step 2: Uploading images.

Next, add the data to your newly created project. You can do it via API or through our web interface. If you drag and drop a directory with a dataset in a supported format, the Roboflow dashboard will automatically read the images and annotations to ether.

Step 3: Labeling

If you only have images, you can label them in Roboflow Annotate.

Step 4: Generate new dataset version

Now that we have our images and annotations added, we can generate a Dataset Version. When Generating a new data version, you may elect to add preprocessing and augmentations. This step is completely optional, however, it can allow you to significantly improve the robustness of your model.

Step 5: Exporting dataset

Once the dataset version is generated, we have a hosted dataset we can load directly into our notebook for easy training. Click Export and select the YOLO v5 PyTorch dataset format.

3. Methodology

3.1 YOLOv5 model

YOLOv5 is a kind of real-time object detection model of computer vision models. YOLOv5 has been implemented in PyTorch and it has known for its high inference speed and accuracy, making it suitable for various real-time applications such as surveillance, autonomous vehicles, and image recognition.

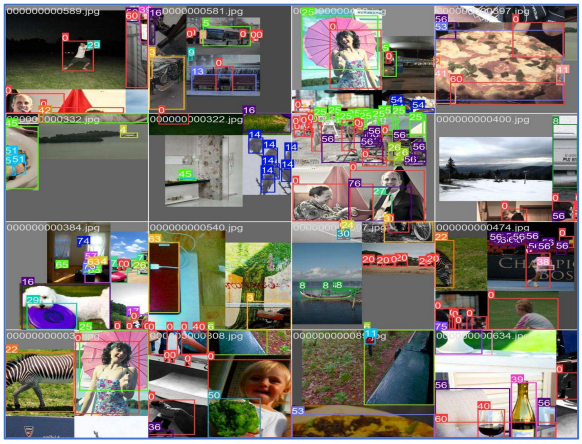

Fig. 3 Example of dataset from YOLOv5 model

Architecture of YOLOv5 [9] consists of three main parts named as backbone, neck and head following in Fig. 2 as follows. First, CSPDarknet model (Backbone) is used to extract features from the input image. Then, PANet (Neck) is blended features from different scales to enhance detection capabilities for various sizes of object. Finally, YOLO Layer (Head) is used to predict bounding boxes, objectness scores, and class probabilities. An example of dataset from YOLOv5 model is presented in Fig. 3.

3.2 Roboflow model

We gathered various pictures featuring raccoons and prairie dogs from different locations. We utilized Roboflow to annotate and prepare these images for training. Subsequently, we fed this data into Roboflow's machine learning engine, making adjustments to guide the model in distinguishing between raccoons and prairie dogs. We repeated this iterative process until achieving the desired accuracy. Essentially, we relied on technology to ensure this model could effectively discern which species of animals.

```
import numpy as np
import torch
import torch.distributed as dist
import torch.nn as nn
import yaml
from torch.optim import lr_scheduler
from tqdm import tqdm

os.environ["GIT_PYTHON_REFRESH"] = "quiet"
os.environ['KMP_DUPLICATE_LIB_OK']='True'

FILE = Path(__file__).resolve()
ROOT = FILE.parents[0]  # YOLOv5 root directory
if str(ROOT) not in sys.path:
```

Fig. 4 YOLOv5 setup

3. Data preparation and Model training

3.1 Data preparation

Data preparation for simulation comprises with three steps as

1) Data Sourcing: Description of the dataset used, including source, size, and nature of the images.
2) Preprocessing Steps: Outline of image preprocessing techniques like resizing, normalization, and augmentation.
3) Dataset Splitting: Information on how the dataset is divided into training, validation, and testing sets.

3.2 Model training

1) Parameter Setting: Details of the model parameters, such as learning rate, batch size, and number of epochs.
2) Training Process: A step-by-step description of the training procedure, including any fine-tuning or optimization techniques used.
3) Graphs and Visuals: Reference to inserted graphs or charts that display the training progress, such as loss curves and accuracy metrics over epochs.

4. Simulation Results

4.1 Case study# 1: COCO128 dataset

The following datasets shown in Fig. 3 are some pictures and video results of preprocessing data and training using general model weights from https: / / www. bilibili. com/ video/ BV1NN4 y127NQ/ ? vd_source= edb9ce84d85f6b820f3362577477b9c1. For the COCO128 data set, you can decompress from https://ultralytics.com/assets/coco128.zip.

Due to timeliness, YOLOv5 will cause some problems when running this version. The following is a method to solve the problem by Setting the two environment variables as shown in Fig.4.

1) os.environ["GIT_PYTHON_REFRESH"] = "quiet"

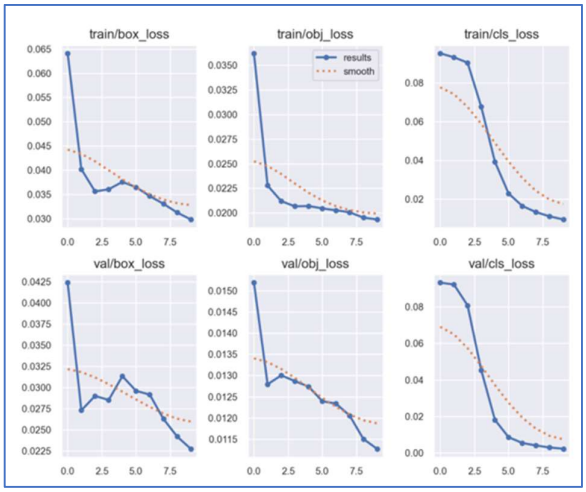

Fig. 5 Results of loss function from YOLOv5

2) os.environ['KMP_DUPLICATE_LIB_OK'] = 'True' solves different problems as follows.
   2.1) os. environ[" GIT_PYTHON_REFRESH"] = "quiet"
      1) This environment variable is for the GitPython library. GitPython is a Python library for interacting with Git repositories.
      2) Setting this variable to "quiet" can prevent GitPython from printing unnecessary log information during use, allowing it to run more quietly in some cases without interfering with the main program output.
   2.2) os.environ['KMP_DUPLICATE_LIB_OK'] = 'True'
      1) This environment variable is related to Intel's Math Kernel Library (MKL). MKL is a high-performance library commonly used for scientific computing.
      2) This environment variable is an unofficial solution for handling library conflicts that may arise in certain situations.

The generic YOLOv5 weighted parameter is used to specify the path to the model weight file. This is usually a pre-trained model or the output of a previous training cycle. So, we use the universal model weights downloaded before running. For general models, only 100 iterations are needed to achieve good results. .

Performance metrics of YOLOv5 model are shown in Figs. 5-6. About loss function (loss in Fig. 5), after the 10th iteration, the 'box_loss', 'obj_loss' and 'cls_loss' of training and verification show an overall downward trend. In the 20th iteration, the loss function starts out high, but then also drops significantly. In particular, ' cls_loss' decreases significantly in both training and validation, indicating that the model has greatly improved in classification.

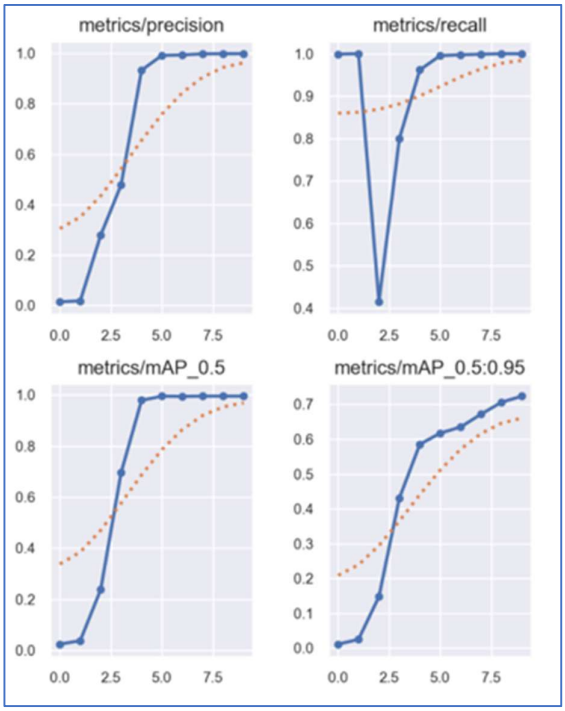

Fig. 6 Results of precision and recall from YOLOv5

Performance of accuracy metrics of YOLOv5 model are presented in Fig. 6 in term of precision, recall, mAP. After the 10th iteration, the precision and recall indicators show certain fluctuations on the training set, but the overall trend is upward. The mAP indicator also shows an upward trend. After the 20th iteration, precision performance stabilized and remained at a high level, while recall declined at some points, but eventually rebounded. Both ‘mAP_0.5’ and ‘mAP_0.5:0.95’ show a clear upward trend, indicating that the overall detection performance of the model has been improved.

4.2 Case study# 2: Roboflow

We utilized Roboflow to annotate and prepare these images for training. Subsequently, we fed this data into Roboflow's machine learning engine, making adjustments to guide the model in distinguishing between raccoons and prairie dogs shown in Fig.7.

The model demonstrated a noteworthy accuracy range, with precision, recall, and F1 score metrics consistently falling within the 70- 80% range. These metrics confirm the model's effectiveness in distinguishing between raccoons and prairie dogs, providing a reliable performance benchmark in Fig. 8.

Despite achieving commendable accuracy within the specified range, it's crucial to address the variability in results. The model's performance, while consistent, may still exhibit diversity that could benefit from further refinement to enhance overall reliability in species differentiation.

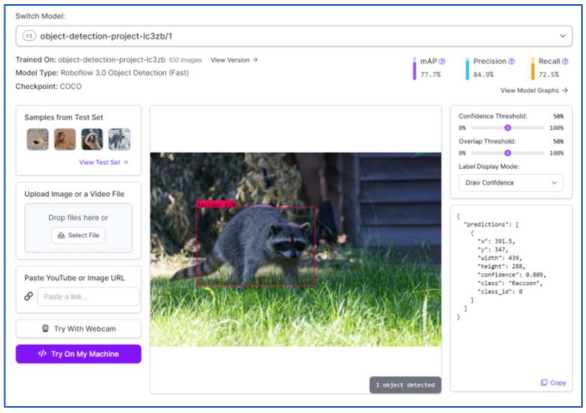

Fig 7 Results and performance of Roboflow model

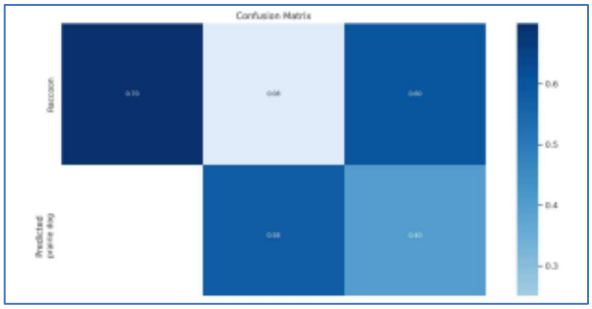

Fig 8 Confusion matrix from Roboflow model

5. Discussion and Challenge

The model faces challenges in adapting to diverse environments, particularly when there are changes in lighting and overall surroundings as shown in Fig. 8. Despite achieving commendable accuracy within the specified range, there are instances where it might still act a bit different in the real world.

This is evident in instances where the AI highlighted the ground, mistakenly identifying it as a prairie dog. It just goes to show how tricky dealing with wildlife can be and emphasizes the importance of continually refining the model to ensure optimal performance across various conditions.

6. Conclusions

In conclusion, the application of Roboflow in distinguishing raccoons and prairie dogs highlights the potential of AI technologies. The methodology, which involves diverse image collection and repetitive training, proves effective. Despite commendable accuracy (70-80%), challenges emerged, particularly in the model's adaptation to varied environments, as seen in misidentifying the ground as a prairie dog. It's essential to address these challenges for improved reliability in real world situations. Continuous refinement is crucial going forward. This case study reveals both the capabilities and challenges of AI. As technology progresses tackling real world challenges is essential to maximize the advantages of AI recognition.

References

[1] L. Cao, Z. Ma, Q. Hu and Z. Xia, "Small Sample Image Classification for Synthetic Aperture Sonar Based on Super-Resolution Reconstruction and Improved Self-supervised Contrastive Learning," *IEEE Sensors Journal*, July 2025.

[2] Roboflow, “Everything you need to build and deploy computer vision applications.” Available from https://roboflow.com/, Access on 9 August 2025.

[3] PyTorch, “YOLOv5”, Available from https://pytorch.org/hub/ultra_lytics_yolov5/, Access on 9 August 2025.

[4] A. K. Akhmedova, I. A. Gavrilov, A. N. Puziy, V. A. Gubenko and A. R. Radik, "Efficiency Estimation of Video Compression with a Bidirectional TV Images Resizing," 2024 IEEE 25th International Conference of Young Professionals in Electron Devices and Materials (EDM), Altai, Russian Federation, pp. 2390-2394, 2024.

[5] B. Hyseni and L. A. Bexheti, "The Impact of Open Data Standardization on the Successful Management of e-Government," 2023 12th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, pp. 1-8, 2023.

[6] S. Shao and C. Xiao, "A Data Enhancement Method for Non-Autoregressive Data Models Based on Joint Multi-Intent Detection and Slot Filling," 2024 International Conference on Electronics and Devices, Computational Science (ICEDCS), Marseille, France, pp. 504-509, 2024.

[7] COCO, “Common Object in Context”, Available from https://cocodataset.org/#home, Access on 9 August 2025.

[8] Roboflow, “Explore the Roboflow Universe: The world's largest collection of open source computer vision datasets and APIs.”, Available from https://universe.roboflow.com/, Access on 9 August 2025.

[9] A. Bochkovskiy, C.Y. Wang, H. Y. Mark Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection”, arXiv:2004.10934v1 [cs.CV] 23 Apr 2020, Available from https://arxiv.org/pdf/2004.10934, Access on 9 August 2025.