SEMI-GUIDED DATA ANNOTATION TOOL

Talha Bacak, Duygu Erduran
Bilgisayar Mühendisliği Bölümü
Yıldız Teknik Üniversitesi, 34220 Istanbul, Türkiye
{l1116038, l1116706}: @std.yildiz.edu.tr

***Özetçe* —Bu çalışmada, YOLO modelleri üzerinde manuel etiketleme ve otonom etiketleme sağlayan windows işletim sistemin üzerinde çalışan yarı güdümlü veri etiketleme aracı gerçekleştirilmesidir. Manuel olarak kullanıcı el ile dikdörtgen şeklinde etiketleme yapılabilirken; otonom etiketlemede ise, herhangi bir YOLO modeli import edebilir ya da uygulamadaki hazır modeller kullanılabilir. Hazır modellerde COCO veri setiyle eğitilmiş YOLOv5s ve YOLOv4-tiny esas alınmıştır. Etiketleme yöntemlerinin etiketleme süre bilgileri karşılaştırılmıştır.**

Anahtar Kelimeler—Otonom etiketleme, Manuel etiketleme, COCO veri seti, YOLOv5s, YOLOv4-tiny.

***Abstract*—In this study, It is the implementation of a semi-guided data labeling tool running on the windows operating system that provides manual labeling and autonomous labeling on YOLO models. Rectangular labeling can be done manually by the user; In autonomous labeling, any YOLO model can be imported or ready-made models in the application can be used. Ready models are based on YOLOv5s and YOLOv4-tiny trained with the COCO dataset. Labeling time information of labeling methods were compared.**

Keywords—Autonomous labeling, Manual labeling, COCO dataset, YOLOv5s, YOLOv4-tiny.

I. INTRODUCTION

Convolutional Neural Networks (CNN) algorithms are one of the most widely used methods in object detection and recognition applications. The feature of the YOLO model, which is one of the widely used CNN algorithms today, is that the "You Only Look Once" algorithm is fast enough to detect objects in one go. When the YOLO algorithm starts working, it detects the objects in the images or videos and the coordinates of these objects at the same time. . In this project, it is desired to make a semi-guided data labeling tool using the GUI running on the Windows operating system, aiming to detect and label objects using deep learning models.

Data labeling is designed to consist of 2 main methods. One is manual tagging and its value is autonomous tagging. In manual labeling, labeling can be done with the mouse. In autonomous labeling, YOLOv5s, YOLOv4-tiny models trained with the COCO dataset can be used or other YOLO models can be imported. The created tags can be saved in txt file in YOLO format or in xml file in Pascal format. These data can be imported from txt files in YOLO format or from xml files in Pascal format. It is aimed to expand the scope of the application so that all YOLO models can be used and imported. It is expected to determine which method is advantageous in labeling by comparing the time spent in labeling with different features.

II. DATASET

Coco dataset is widely used in deep learning studies. It is preferred because of the convenience feature of the dataset in semi-supervised data labeling operations. In this study, trained labeling is performed using the Coco dataset. [1]

III. DEEP LEARNING METHODS

The widespread use of deep learning algorithms in object detection and recognition applications in images and videos has become widespread. In recent years, object detection and recognition applications have found solutions to many problems in the fields of security, defense, natural disasters (flood, earthquake and fire, etc.), health (prevention of the spread of epidemics, etc.), agriculture and forestry. [2]

Convolutional Neural Networks (CNN) algorithms are one of the most widely used methods in object detection and recognition applications. One of the popular CNN algorithms, YOLO, "You Only Look Once", the answer to why this is the name of the algorithm is that it is fast enough to detect objects at once. detects. [3]

YOLOv5 is lightweight, extremely easy to use, works fast, infers fast, and performs well. In terms of Yolov5 performance, ease of use (installation and configurations) and versatility, the models provide more efficiency than other Yolo versions in terms of working on different devices.The test was carried out on Yolov5 models by keeping the package size constant. The data obtained as a result of the test were adapted on the graph. The obtained data are better interpreted.[4]

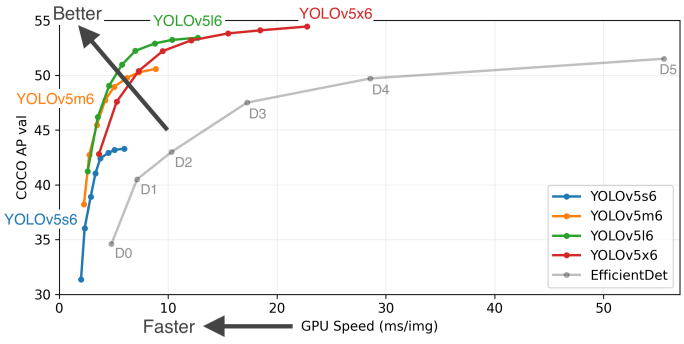

Figure 1 Performance values of YOLOv5 models [4]

IV. DEVELOPMENT ENVIRONMENT AND LIBRARIES

It is desired to make a semi-guided data labeling tool using the GUI running on the Windows operating

system, aiming to detect and label objects using deep learning models. Operating System Windows10 is preferred. Application Development Language is Python v3.8.5 because there are many ready-made libraries for image processing and deep learning, and it is rich in resources and documents. The Application Development Environment is Anaconda Spyder and Qtdesigner. Framework and Python Libraries Used:Torch, Sys and PyQt5.

V. METHODOLOGY

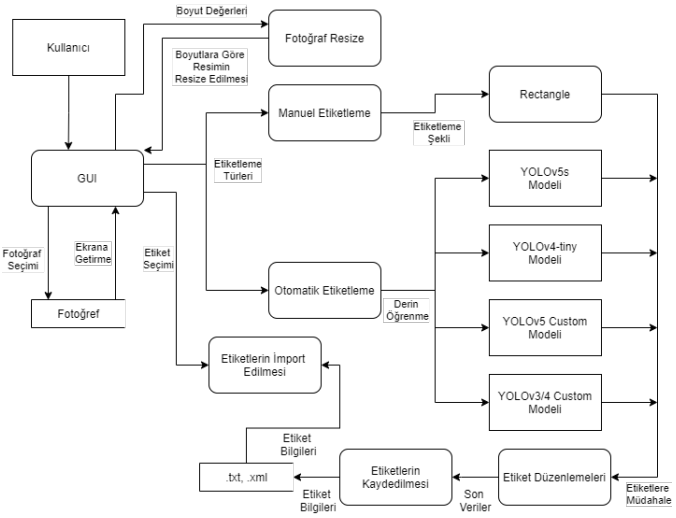

Figure 2 Data flow diagram

The application has 4 main tasks in the graphical interface. These are importing photos, resizing photos, importing tag information, and Tag Types. The dimensions of the photos are scaled according to the dimensions entered on the application. Label types are divided into 2 parts: Manual labeling and Autonomous Labeling. Manual labeling is done as a rectangle. If it is Autonomous Labeling, there are 4 part models. Two of these are the ready-labeled YOLOv5s model and the YOLOv4-tiny Model. Others are YOLOv5 Custom Model where the user can import the trained model and YOLOv3/4 Custom Model where you can use old versions of YOLO. Once the model is imported, it can be accessed as .txt or .xml. User tagging can also intervene. It can delete the tag and add the tag. Labeling can be saved in the desired format. .txt or .xml. The special feature of the application is that the recorded data can be imported again.

VI. DETECTION PROCESS

Manual tagging and autonomous tagging, which are the main uses of the application, will be discussed. In manual tagging, users created the tags and tag classes themselves. In autonomous tagging, tags and tag classes are created by the selected deep learning model. YOLOv5s trained with the COCO dataset was used as a deep learning model. 10 users were asked to first autonomously tag 5 photos and then manually tag them. All users have tagged the same 5 photos. Users tagged the objects mentioned in these 5 photos. The tag information of the photos is shown in the table 2. The times taken for these labeling operations are specified in the table 2. The specified times are the total time taken as a result of tagging 5 photos in order.

Photograph	Number of Labels	Number of Label Classes
1	4	2
2	1	1
3	4	1
4	8	3
5	5	1

Table 1 Tags of Photograph

Person	Autonomous Labeling Time (second)	Manual Labeling Time(second)
1	27	123
2	22	104
3	21	94
4	22	96
5	29	102
6	32	95
7	33	86
8	25	104
9	27	89
10	24	97

Table 2 Labeling time information

VII. EXPERIMENTAL RESULTS

The time spent was considered as the evaluation criterion. The data to be evaluated is in the table 2. The difference in time will be evaluated by taking the average of the autonomous tagging and manual tagging times spent by 10 people. The average time spent for autonomous tagging is **26.2** seconds. The average time spent for manual tagging is **99** seconds. As you can see, manual tagging took **3,77** times that of autonomous tagging. In this experiment, each user created a total of 22 tags and 8 tag classes for 5 photos. As the number of photos, tags, or tag classes increase, the time spent between autonomous tagging and manual tagging will continue to increase.

Autonomous tagging is more advantageous than manual tagging when tagging a small number, and it will continue to increase its advantage as the data to be tagged increases. The disadvantage that may occur in autonomous labeling is that it cannot make all labels error-free. The error rate here will vary according to the model to be used.

VIII. PERFORMANCE ANALYSIS

There are 2 basic labeling methods in our application, namely manual labeling and autonomous labeling. In manual labeling, individual rectangular labels can be made by the user. In autonomous labeling, any YOLO model can be imported or ready-made models in the application can

be used. For the experimental results, the YOLOv5s model trained with the COCO dataset available in the application was used. The time taken while importing the models is not included in the times specified in the table 2. If there are models that need to be imported or downloaded, these will take a short time depending on the hardware and internet speed. Since the ready-to-use models in the application are YOLOv5s and YOLOv4-tiny, the models work very fast. While labeling with imported models, the labeling speed may change depending on the model structure.

IX. CONCLUSION

In this project, semi-guided data tagging tool running on windows operating system that provides manual tagging and autonomous tagging on YOLO models. Rectangle labeling can be done manually by the user; In autonomous labeling, any YOLO model can be imported or ready-made models in the application can be used. YOLOv5s and YOLOv4-tiny trained with COCO dataset can be used in ready models. The advantages and disadvantages of autonomous tagging were examined by comparing autonomous tagging and manual tagging. The creation of data sets takes a lot of time and incorrect labeling is frequently encountered. An approach that can reduce these problems has been tried to be put forward.

According to the results obtained in this study, autonomous tagging works much faster than manual tagging. Autonomous tagging can make wrong or incomplete tagging in object detection. Therefore, after autonomous labeling in the application, the labels can be intervened manually. The created tags can be saved in txt file in YOLO format or in xml file in Pascal format.

While working on this project, we found that there were occasional errors in the previously labeled data and many errors in the datasets. For this reason, a feature has been added to the application where pre-labeled data can be loaded and checked. These data can be imported from txt files in YOLO format or from xml files in Pascal format. The scope of the application has also been expanded so that all YOLO models can be used and imported. In order for users to see objects in low-resolution photos more clearly, the feature of resizing the photo has been added and the errors that this may cause in the tag information have been prevented.

More user-friendly solutions can be offered when manipulating tags in the project. Its scope can be expanded by developing it to use different deep learning models besides YOLO. A development can be made so that it can also be applied to non-Windows platforms.

REFERENCES

[1] H. S. DIKBAYIR and H. İ. BÜLBÜL, "Derin öğrenme yöntemleri kullanarak gerçek zamanli araç tespiti," *TÜBAV Bilim Dergisi*, vol. 13, no. 3, pp. 1–14.

[2] M. Burgaz, "Derin öğrenme algoritmaları kullanarak insansız hava araçları ile silah tespiti," Master's thesis, Batman Üniversitesi Fen Bilimleri Enstitüsü, 2020.

[3] M. Çetinkaya and T. Acarman, "Trafik işaret levhası tespiti için derin öğrenme yöntemi," *Akıllı Ulaşım Sistemleri ve Uygulamaları Dergisi*, vol. 3, no. 2, pp. 140–157.

[4] Ultralytics. (2020) Yolov5. [Online]. Available: https://github.com/ultralytics/yolov5