TTTC3413 ROBOT APPLICATION

SEMESTER 1 SESI 2021/2022

PROJECT 2

TITLE

HELMET TRACKING USING YOLOV5 & DEEPSORT

LECTURER

DR. ABDUL HADI BIN ABD RAHMAN

GROUP 3

	MEMBER NAME	MATRIC NO.
1.	NURFARIESYA FATIN BINTI AHMAD SAFUAN	A175116
2.	AMAL MAJIDA BINTI MUNIR	A174807

CONTENTS

1.0 INTRODUCTION

Players in rough sports such as rugby and American football are easily exposed to injuries whether light injuries or serious injuries. Therefore, the players are required to wear protective gear to reduce the chances of getting serious injuries that could affect their life. Some examples of these protective gears are headgear or helmet, mouth guards, gum shield and shoulder vest. However, a helmet is the most crucial protective gear that a player must wear since wearing it can avoid the player from having a serious head injury. As we know, the human head is the home to all the body's major sensory organs and the most important of these is the brain. It controls and coordinates human actions and reactions. Although the nose, ears, tongue, nerves and other parts are important, without a healthy brain, they would all be useless. A concussion or open skull fracture can occur as a result of brain trauma caused by an impact. Even minor head injuries that do not result in loss of consciousness can result in long-term behavioural and cognitive issues like memory loss, inability to focus, sleep disturbances, and, in some cases, permanent disability or death. This is why the National Football League (NFL) and Amazon Web Services (AWS) are teaming up to develop the best sports injury surveillance and mitigation program on Kaggle which would help accurately identify each player's "exposures" and helmet throughout a football play. Therefore our objective in this project is to develop a helmet tracking system using YOLOv5 and Deepsort by modifying a given dataset from the NFL and AWS competition on Kaggle.

2.0 PROBLEM STATEMENT

Wearing a helmet is essential for each player in the game to avoid serious injuries. However, keeping track of the helmet on each player during the game is almost impossible since they keep moving and sometimes the view is blocked by another player or there is occlusion. Therefore, having a system to track the helmet on each player's head is a crucial investment to make since it is highly necessary to avoid any dangerous situation happening.

3.0 TECHNICAL SOLUTION

We have obtained the dataset from Kaggle, NFL Health & Safety. This dataset contains helmets images and videos. There are 9947 images and 126 videos in this dataset. However after analyzing each image, we only used 1200 out of 9947 images for our dataset due the original dataset being too large which made the dataset difficult to be trained. We chose the images in different points of view and situations to ensure that there are various conditions in the dataset. We use a website application called makesense.ai for labeling the images since some of them are labeled inaccurately.

To develop a helmet tracking system on the player's, we decided to use the YOLOv5 and Deep Sort algorithm. We know that YOLOv5 is one of the best algorithms for object tracking. Compared to other prior releases of YOLO, YOLOv5 is the most recent and lightest version that has been implemented in Pytorch rather than a fork from the original Darknet. This makes it easier to train the dataset. Besides, some researchers have claimed that the YOLOv5 algorithm outperforms YOLOv3 and YOLOv4 in terms of accuracy. In addition, compared to other algorithms such as the RCNN algorithm, YOLOv5 is better at detecting smaller objects and runs faster than RCNN. The running speed of YOLOv5 is 52.8 FPS while the running speed of RCNN is 21.7FPS. YOLOv5 runs about 2.5 times faster than RCNN. To improve object tracking, we are using DeepSORT. DeepSORT allows us to track the object through longer periods of occlusions. Therefore, this makes DeepSORT able to reduce the number of identity switches.

4.0 DISCUSSION

The dataset we used in this project is from the NFL and AWS competition on Kaggle [1]. After analyzing through the dataset, we decided to only use 1200 images for weight training purposes. First, we trained the data to get the best weight for helmet detection by using YOLOv5. To train the data, we need to ensure that the number of data we have corresponds to the number of labels. The labels used in this project are in .csv format.

```
[ ] import pandas as pd
    import os
```

```
[ ] from google.colab import drive
    drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

```
helmet_pd = pd.read_csv("/content/gdrive/Shareddrives/Balance Unlimited/Robot Projek 2/Helmet/imageLabels.csv")
helmet_pd.head()
```

	image	label	left	width	top	height
0	57502_000480_Endzone_frame0495.jpg	Helmet	403	30	296	33
1	57502_000480_Endzone_frame0495.jpg	Helmet	421	27	335	41
2	57502_000480_Endzone_frame0495.jpg	Helmet	439	24	374	49
3	57502_000480_Endzone_frame0495.jpg	Helmet	457	21	413	57
4	57502_000480_Endzone_frame0495.jpg	Helmet	475	18	452	65

Figure 1.1:Read the label.

Then, we split the dataset with a ratio of 80:20 for training and validation. Thus we get 1080 images for training and 120 images for validation.

```
[ ] list_images = helmet_pd['image'].unique()
    len(list_images)
```

```
1200
```

```
[ ] from sklearn.model_selection import train_test_split

    lst_train, lst_valid = train_test_split(list_images, test_size=0.2, random_state=42)
    print(len(lst_train))
    print(len(lst_valid))
```

```
1080
120
```

Figure 1.2: Random Split with ratio 80:20 percentage.

Next is we install all the requirements and dependencies required for the YOLOv5 environment and configure the files and directories structure for training and validation.

```
[ ] %cd /content/gdrive/Shareddrives/Balance Unlimited/Robot Projek 2/Helmet
    !git clone https://github.com/ultralytics/yolov5  # clone repo
    %cd yolov5
    !git checkout v6.0
    !pip install -r requirements.txt
```

Figure 1.3: Installing the environment.

```
%%writefile NFL.yaml

train: /content/gdrive/Shareddrives/Balance Unlimited/Robot Projek 2/Helmet/images/train
val: /content/gdrive/Shareddrives/Balance Unlimited/Robot Projek 2/Helmet/images/valid

# number of classes
nc: 1

# class names
names: ['Helmet']
```

```
Writing NFL.yaml
```

Figure 1.4: Configure the files and directories structures.

Then we start training the weight.

```
!WANDB_MODE="disabled" python train.py --img 1280 --batch 8 --epochs 15 --data NFL.yaml --weights yolov5m6.pt
```

```
Using 2 dataloader workers
Logging results to runs/train/exp
Starting training for 15 epochs...

     Epoch   gpu_mem       box       obj       cls    labels  img_size
      0/14       11G    0.1407   0.08964         0       146      1280: 100% 135/135 [13:29<00:00,  5.99s/it]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 8/8 [00:18<00:00,  2.36s/it]
                  all        120       2506      0.357      0.377      0.294      0.069

     Epoch   gpu_mem       box       obj       cls    labels  img_size
     14/14     11.3G   0.06058   0.08012         0       226      1280: 100% 135/135 [13:27<00:00,  5.98s/it]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 8/8 [00:16<00:00,  2.08s/it]
                  all        120       2506      0.954        0.8      0.886      0.514

15 epochs completed in 3.432 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 71.4MB
Optimizer stripped from runs/train/exp/weights/best.pt, 71.4MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model Summary: 378 layers, 35248920 parameters, 0 gradients, 49.0 GFLOPs
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 8/8 [00:23<00:00,  2.95s/it]
                  all        120       2506      0.954        0.8      0.885      0.514
Results saved to runs/train/exp
```

Figure 1.5: Training progress and the accuracy we obtained.

The process to train the weight takes about 2 hours 15 minutes. The accuracy we obtained from training the weight is 0.885. This is the output we get for the helmet weight training.

Figure 1.6: Helmet Detection result.

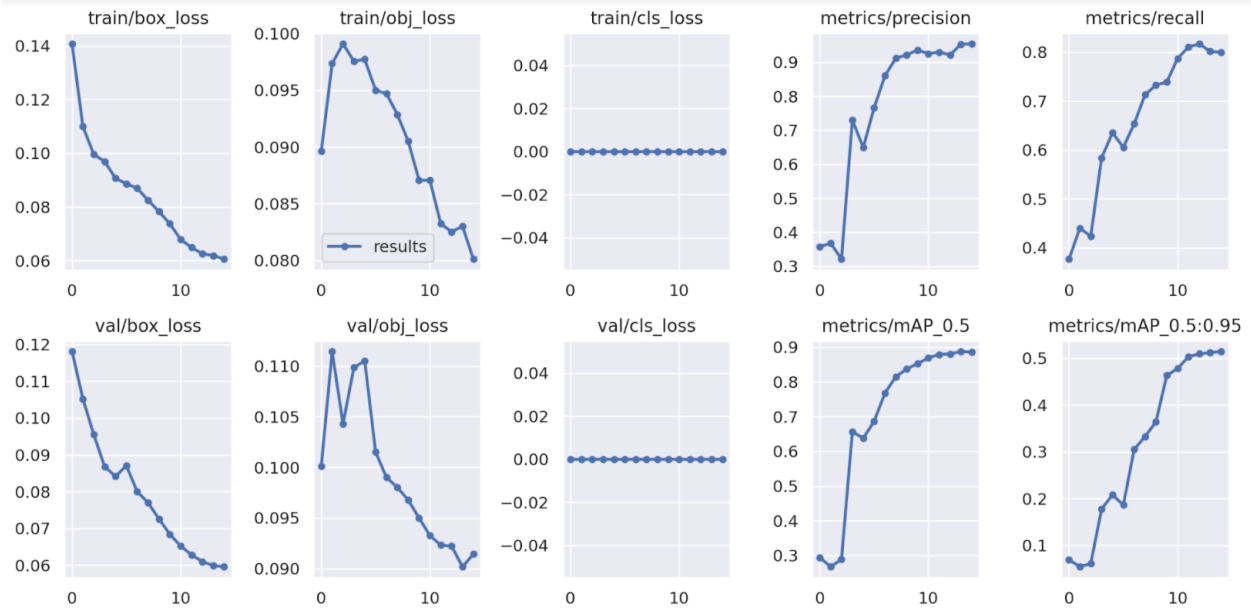

Figure 1.7: The results of training and validation.

After obtaining the weight training, best.pt. We continue to develop the tracking of the helmet. For this, we used YOLOv5 with Deep SORT. First, we install the requirements and dependencies of Yolov5_Deepsort_Pytorch as shown in the figure below.

```
[ ]  %cd /content/drive/Shareddrives/Balance Unlimited/Robot Projek 2/Helmet/Yolov5_DeepSort_Pytorch
     !pip install -r requirements.txt

     /content/drive/Shareddrives/Balance Unlimited/Robot Projek 2/Helmet/Yolov5_DeepSort_Pytorch
     Requirement already satisfied: matplotlib>=3.2.2 in /usr/local/lib/python3.7/dist-packages (from -r requir
     Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.7/dist-packages (from -r requiremer
```

Figure 1.8: Installing the requirements for YOLOv5 and Deepsort.

Then we run the track.py with the weight training we obtained earlier, best.pt. It took us about 4 minutes to get the result.

This is the output we achieved.

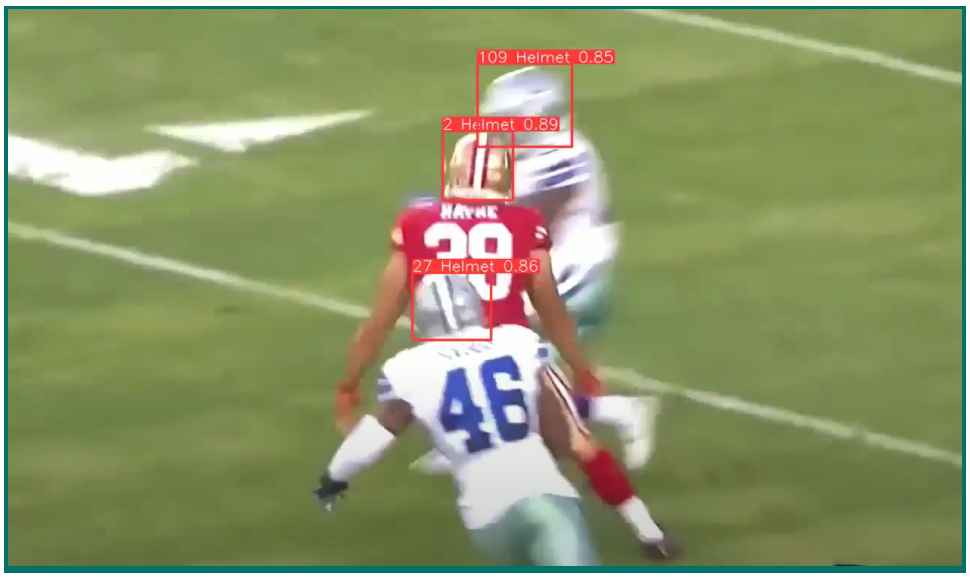

Figure 1.8: Output 1

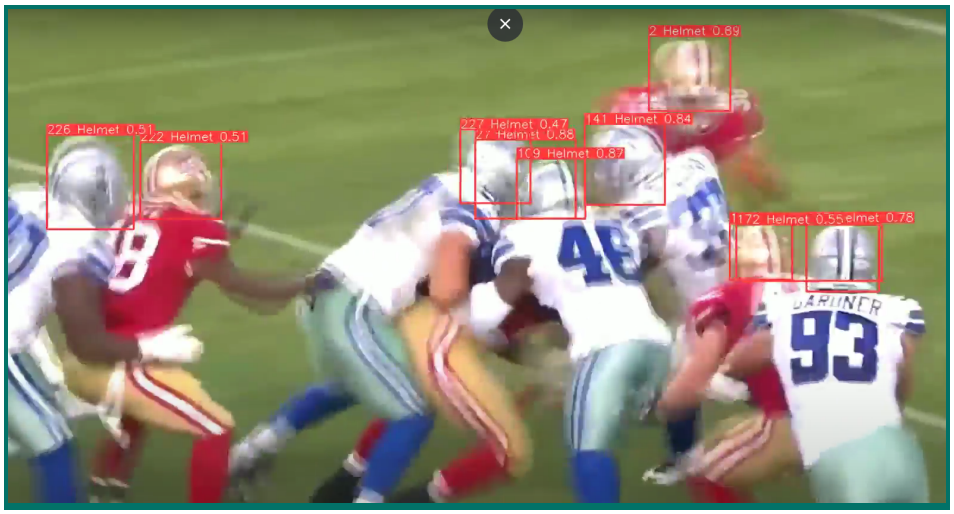

Figure 1.9: Output 2

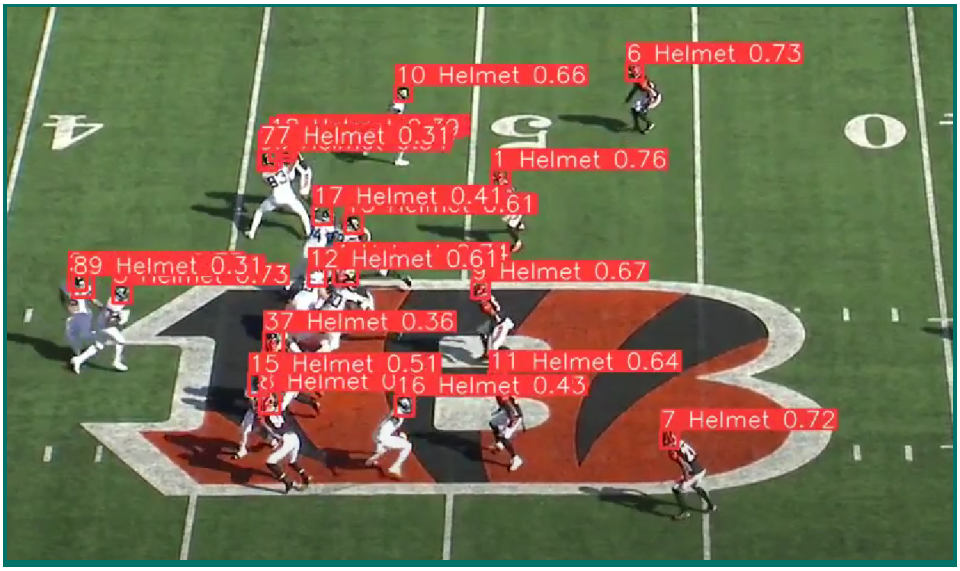

Figure 1.10: Output 1

This is the link for our video output.

1. https://drive.google.com/drive/folders/1bZ7STV2Ga1rHeAiXTcNsFjL0a1ylVc45?usp=sharing

5.0 CONCLUSION

In conclusion, we used YOLOv5 to perform the weight training for helmet detection and YOLOv5 with Deepsort to perform the helmet tracking. We also use Graphics Processing UInits (GPUs) in google colaboratory to make the runtime faster as a GPU is designed to quickly render high-resolution images and videos. Therefore, we successfully obtained an accuracy of 0.885 which is considered as a good result for the helmet weight. After implementing the helmet-weight with the YOLOv5 and Deepsort, we also achieved a good result as the helmet tracking did track the helmet on each player during their movement and can also detect the helmet even after being blocked or collided by another player.

6.0 REFERENCE

1. *NFL Health & Safety - Helmet Assignment | Kaggle*. (2018). Kaggle.com. https://www.kaggle.com/c/nfl-health-and-safety-helmet-assignment
2. C K. (2020, August 7). *How to track football players using Yolo, SORT and Opencv.* Medium; Towards Data Science. https://towardsdatascience.com/how-to-track-football-players-using-yolo-sort-and-opencv-6c58f71120b8
3. *Helmet detection error analysis in football videos using Amazon SageMaker | Amazon Web Services*. (2021, March 18). Amazon Web Services. https://aws.amazon.com/blogs/machine-learning/helmet-detection-error-analysis-in-football-videos-using-amazon-sagemaker/
4. Chamidu Supeshala. (2020, August 23). *YOLO v4 or YOLO v5 or PP-YOLO? Which should I use? | Towards Data Science*. Medium; Towards Data Science. https://towardsdatascience.com/yolo-v4-or-yolo-v5-or-pp-yolo-dad8e40f7109