BAB IV

HASIL DAN PEMBAHASAN

4.1. Hasil Penyajian Data

Dalam tahapan sebelumnya telah dipaparkan terkait proses dari rancangan pelaksanaan tugas akhir ini, dimulai dari pengumpulan *dataset* hingga proses uji coba metode. Dalam bab ini akan memaparkan hasil dari setiap tahapan-tahapan yang telah dilakukan dalam penelitian ini, supaya peneliti lain memiliki gambaran dari setiap proses yang dilakukan dalam penelitian ini secara lebih rinci guna sebagai evaluasi dalam pengembangan penelitian selanjutnya.

4.1.1. Hasil Identifikasi Masalah dan Studi Literatur

Identifikasi masalah dalam sebuah penelitian dilakukan untuk mengetahui permasalahan yang menjadi pendorong penelitian dilakukan. Masalah dalam penelitian ini didasarkan pada latar belakang yang telah dipaparkan sebelumnya, yaitu rendahnya dalam penggunaan APD pada pekerja dan diperlukannya otomatisasi monitoring dengan cara deteksi objek dengan metode yang diusulkan. Studi literatur dilakukan untuk memberikan pemahaman terkait konsep, teori, dan metode yang digunakan dalam penelitian ini. Hal ini dapat memberikan pedoman atau pendukung dalam proses pemahaman penelitian yang dilakukan, dan hal tersebut telah dipaparkan dalam sub bab sebelumnya yang menjelaskan mengenai landasan teori penelitian dan penelitian yang berkaitan dengan penelitian, salah satunya adalah pemahaman mengenai metode *You Only Look Once* (YOLO).

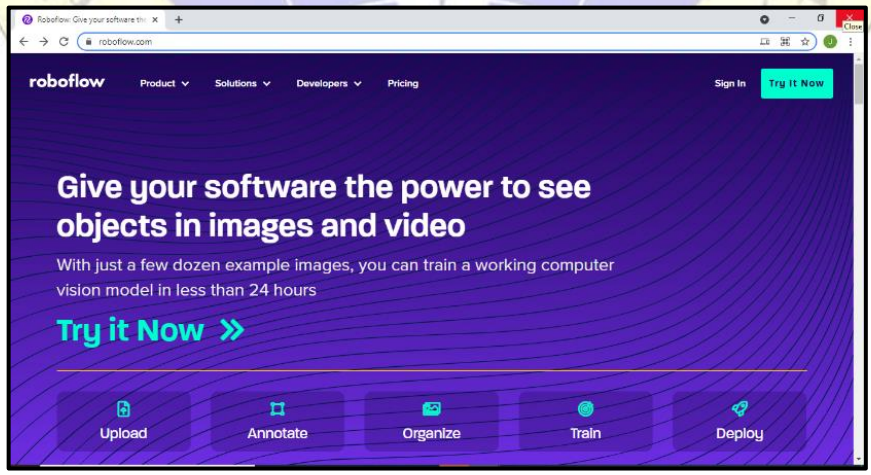
4.1.2. Hasil Analisis Permasalahan

Dari hasil identifikasi permasalahan sebelumnya, analisis permasalahan yang didapatkan telah dipaparkan pada rumusan masalah yang ada, yaitu mengimplementasikan metode usulan yaitu YOLO dalam mendeteksi objek head, helmet, vest, dan no_vest dan mengukur tingkat keberhasilan metode YOLO dalam mendeteksi objek tersebut. Dalam menyelesaikan permasalahan tersebut, beberapa kebutuhan penelitian diperlukan. Kebutuhan yang digunakan dalam penelitian yang dilakukan adalah sekumpulan data atau *dataset.* Data merupakan bahan utama dalam penelitian yang dapat memberikan sebuah informasi setelah adanya proses pengolahan data. Data yang digunakan merupakan sekumpulan data yang

merepresentasikan pekerja sedang menggunakan APD baik berupa *helmet* atau vest, serta pekerja yang tidak sedang menggunakan APD yang direpresentasikan pada objek *head* atau *no_vest*. *Google Colaboratory* digunakan untuk mengeksekusi *source code Python* menggunakan usulan metode penelitian yaitu YOLO agar dapat mengolah data dengan baik, sehingga menghasilkan informasi sesuai dengan yang diinginkan dalam penelitian yang dilakukan.

4.1.3. Hasil Pengumpulan *Dataset*

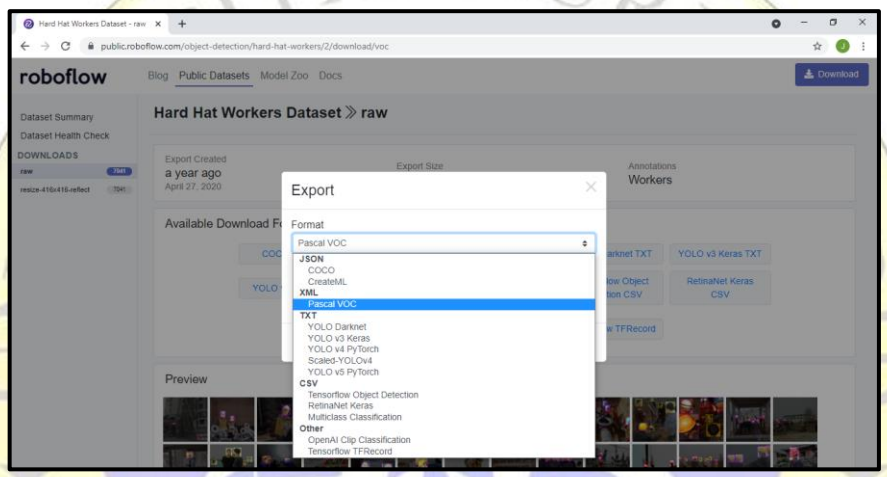
Pengumpulan *dataset* dalam penelitian ini tidak dilakukan secara mandiri dengan kata lain mendapatkan *dataset* tidak langsung terhadap objek yang dituju. Namun dalam penelitian ini memanfaatkan *dataset* yang tersedia oleh beberapa situs penyedia *dataset* secara *online* dan dapat diperoleh secara gratis sesuai kebutuhan data penelitian yang diinginkan. Sama halnya dengan yang telah dipaparkan sebelumnya, bahwasanya data tersebut didapatkan melalui https://roboflow.com/. Adapun cara yang dapat dilakukan dalam mendapatkan *dataset* tersebut tersusun seperti berikut ini. Halaman Utama merupakan halaman yang menyediakan beberapa info terkait cara mengakses aplikasi. Agar dapat mengakses situs *Roboflow*, maka pastikan kita memiliki akun terlebih dahulu untuk dijadikan sebagai hak akses dalam situs tersebut. Jika tidak memiliki akun maka daftarkan akun terlebih dahulu. Halaman utama dari *Roboflow* dapat dilihat dari Gambar 4.1.



Gambar 4.1. Halaman Utama *Roboflow*

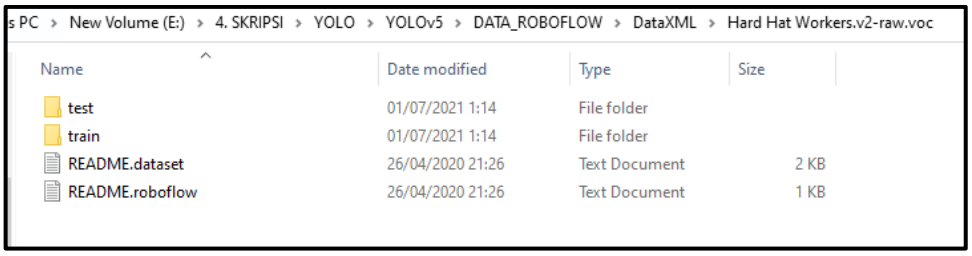
Dalam mendapatkan *dataset* yang sesuai, maka peneliti dapat mengakses kumpulan *dataset* melalui halaman *Public Datasets*. Didalam *public datasets*

terdapat 39 macam *dataset* yang dapat digunakan sesuai kebutuhan data yang diinginkan pada saat penelitian dilakukan. *Dataset* yang digunakan penelitian ini, dapat diakses melalui halaman berikut ini https://public.roboflow.com/object-detection/hard-hat-workers. *Dataset* terdiri dari 7.041 *dataset*, yang terdiri dari gambar dan anotasi. Terdapat dua pilihan *dataset* yang dapat diunduh, pertama data *raw* (data mentah/asli) dan yang kedua merupakan data hasil *resize* dengan ukuran 416 x 416 pixels. Dalam penelitian ini, menggunakan data *raw* sebagai *dataset* yang diunduh dalam format Pascal VOC (*Visual Object Classes*) atau *.xml. Gambar 4.2. merupakan halaman unduh data.



Gambar 4.2. Unduh Data *.xml

Setelah selesai dalam mengunduh *dataset* maka selanjutnya adalah *extract dataset*, karena hasil unduh *dataset* sebelumnya tersimpan dalam bentuk *.zip. Ekstrak *dataset* digunakan untuk mempermudah dalam melakukan *pre-processing* data. *Dataset* terdiri dari folder data *train*, *test*, dan dua data *Roboflow.txt* yang berisikan alamat *dataset* dan informasi dari *dataset* yang diunduh. *Raw dataset* memiliki beberapa dimensi ukuran gambar dari 179 x 270 pixels hingga 640 x 959 pixels yang tersimpan dalam format *.jpg. Dan *dataset* tersebut tersedia berupa gambar yang telah dilengkapi dengan hasil anotasi, berupa kelas *head*, *helmet*, dan *person*. Gambar 4.3. merupakan *dataset* yang berhasil diunduh yang selanjutnya dapat dijadikan sebagai sumber *dataset*.



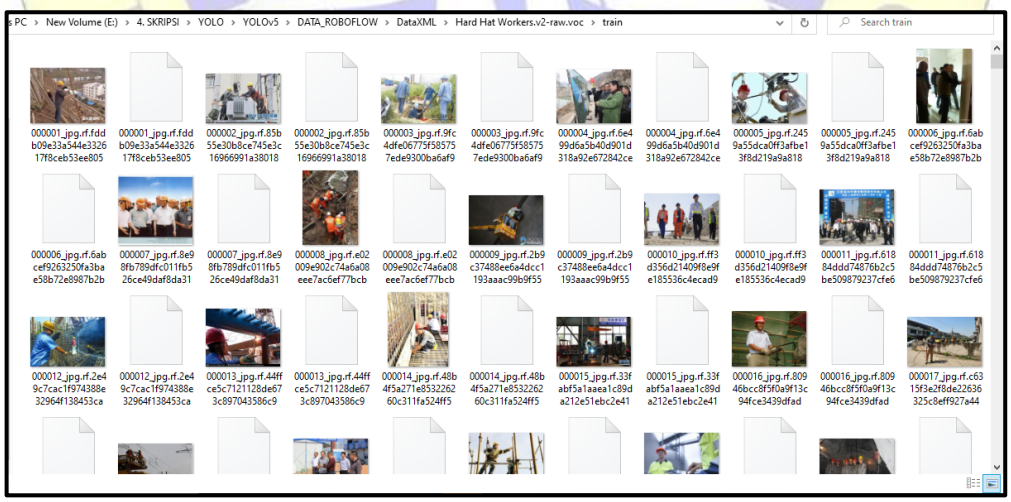
Gambar 4.3. *Raw Dataset*

4.1.4. Hasil *Pre-processing* Data

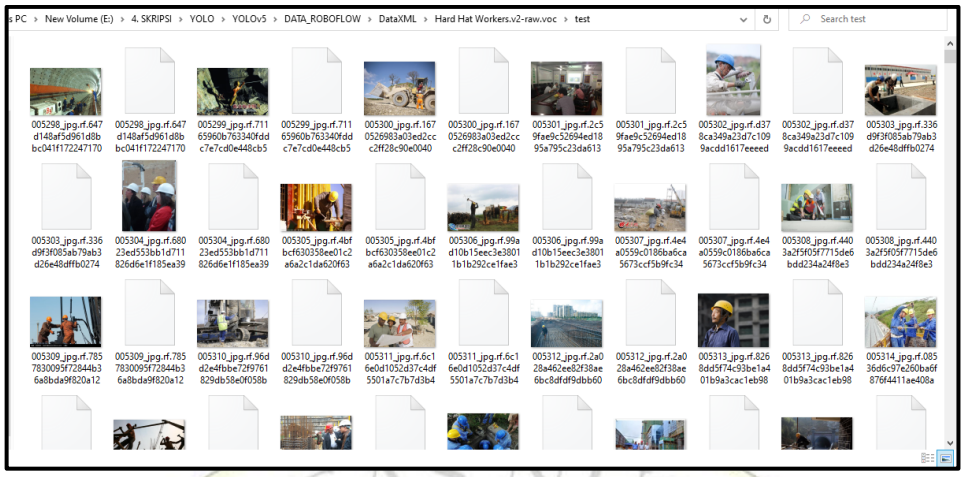
Sebelum proses implementasi *dataset* yang telah didapatkan sebelumnya kepada metode usulan yang digunakan dalam penelitian ini, maka tahapan *pre-processing* data sangat dibutuhkan agar proses pengolahan data berjalan dengan sistematis. *Pre-processing* data dilakukan dengan beberapa tahapan, dari pengelompokan data, *annotation* (anotasi), dan *splitting a dataset* (membagi *dataset*).

4.1.4.1. Pengelompokan Data

Dari beberapa penjelasan yang telah dipaparkan sebelumnya *dataset* yang didapatkan melalui *Roboflow* terdapat 7.041 data yang terdiri dari data gambar dan anotasi yang telah terbagi menjadi dua sub *folder train* dan *test*. Pemaparan tersebut terangkum pada Gambar 4.4 dan Gambar 4.5.

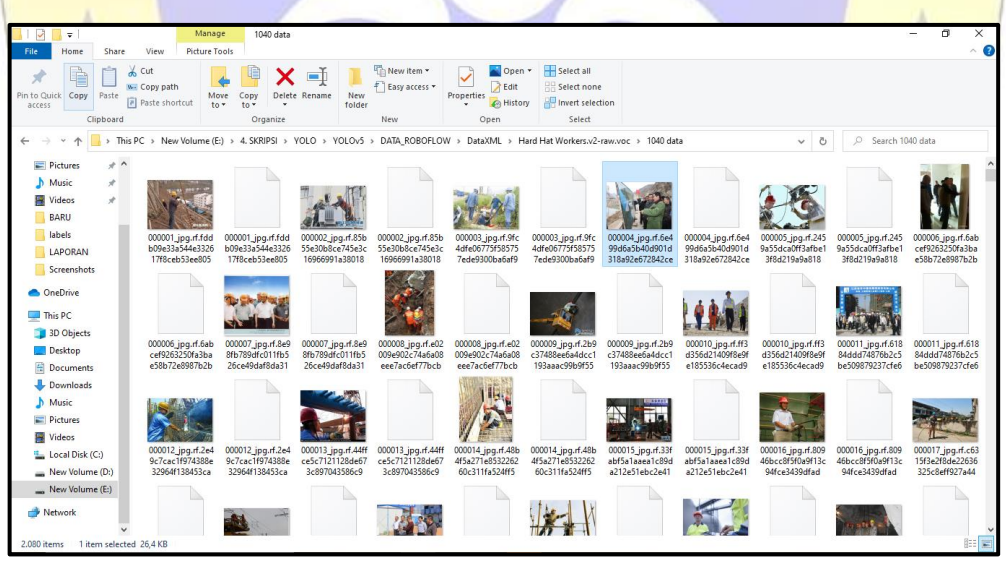


Gambar 4.4. Data *train*



Gambar 4.5. Data *test*

Dari jumlah *dataset* yang mencapai 7.041 data gambar dan anotasi, penelitian ini menggunakan 1.173 data gambar dan anotasi yang dipilih secara acak dalam *folder train*. Data anotasi/label gambar harus memiliki nama file yang sama dengan data gambar yang dipilih hal tersebut guna untuk mempermudah dan menghindari kesalahan proses anotasi objek. Gambar 4.6. merupakan kumpulan dari 1.173 *dataset* yang berisikan data gambar dan anotasi pekerja yang terlibat dalam bidang konstruksi dari beberapa kedudukan pekerjaan.



Gambar 4.6. *Dataset*

4.1.4.2. *Annotation*

Dalam proses objek deteksi, proses *annotation* atau anotasi diperlukan sebelum melakukan implementasi data terhadap metode yang digunakan, dengan

tujuan agar metode dapat dilatih dengan baik dari data anotasi yang ada sehingga, lebih mudah dalam mengenali objek yang diinginkan. Proses anotasi gambar dalam penelitian ini dilakukan menggunakan pemanfaatan aplikasi *labelling dataset* yaitu *LabelImg*. Dalam mengolah data menggunakan metode *You Only Look Once* (YOLO) format anotasi data yang digunakan adalah *.txt. Pada saat pengambilan *dataset*, format unduhan anotasi data yang dipilih adalah *.xml, hal ini dikarenakan untuk mempermudah proses penambahan anotasi untuk kelas *vest* dan *no_vest*, karena sebelumnya data telah tersedia dengan anotasi kelas *head*, *helmet*, dan *person*. Dengan menggunakan format *.xml kelas *vest* dan *no_vest* dapat langsung ditambahkan melalui aplikasi *LabelImg*. Dan kelas person yang ada sebelumnya dapat dihapus karena kelas tersebut tidak digunakan dalam penelitian ini. Gambar 4.7. merepresentasikan kelas yang digunakan dalam *dataset*.

a. *Head* (Kepala)

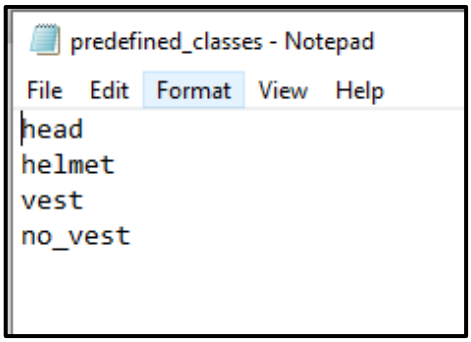
b. *Helmet* (Pelindung Kepala)

c. *Vest* (Rompi)

d. *No_Vest* (Tidak Berompi)

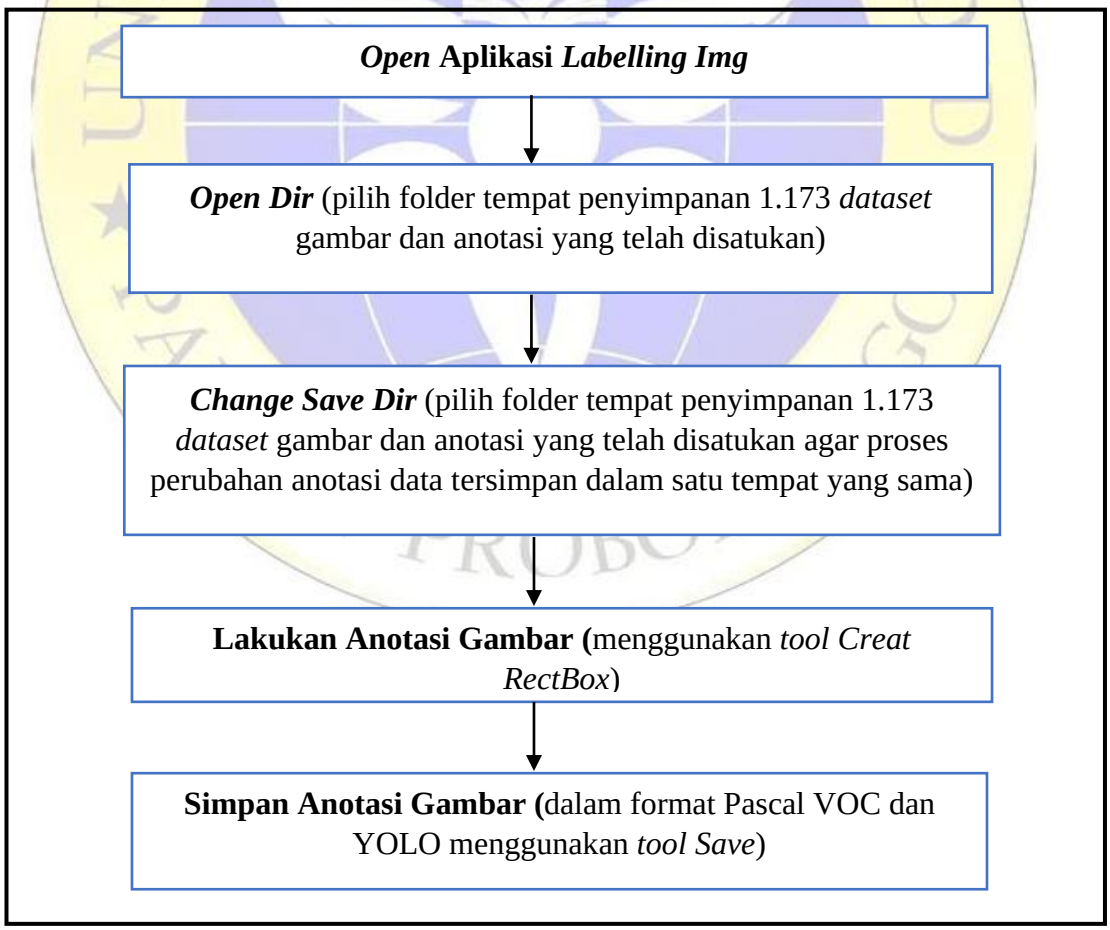
Gambar 4.7. Representasi Kelas

Sebelum proses penambahan anotasi kelas *vest* dan *no_vest*, nama kelas yang ada pada *folder* data *LabelImg* diubah terlebih dahulu menjadi *head*, *helmet*, *vest*, *dan no_vest* dengan tujuan agar pada saat anotasi gambar hanya fokus kedalam empat kelas yang telah ditentukan. Gambar 4.8. merupakan kelas data dalam anotasi gambar menggunakan *Labelling Img*.



Gambar 4.8. Kelas Anotasi Gambar

Setelah melakukan perubahan kelas di dalam *folder* data dalam aplikasi *LabelImg*, tahapan selanjutnya adalah melakukan anotasi pada 1.173 data gambar untuk menambahkan anotasi kelas *vest* atau *no_vest*, jika dari 1.173 gambar tersebut terdapat objek *vest* atau *no*_vest yang telah dipaparkan dalam Gambar 4.6. Proses anotasi gambar dengan *Labelling Img* dapat digambarkan melalui Gambar 4.9.



Gambar 4.9. Proses Anotasi Gambar

Setelah mengikuti langkah-langkah diatas, maka hasil dari proses anotasi tersebut dapat tergambarkan dalam Gambar 4.10. Ada beberapa alat yang dapat digunakan dalam proses anotasi dalam *LabelImg*, misalnya *Creat RectBox* yang digunakan untuk membuat kotak yang digunakan sebagai kotak pembatas dalam membuat anotasi sebuah objek.

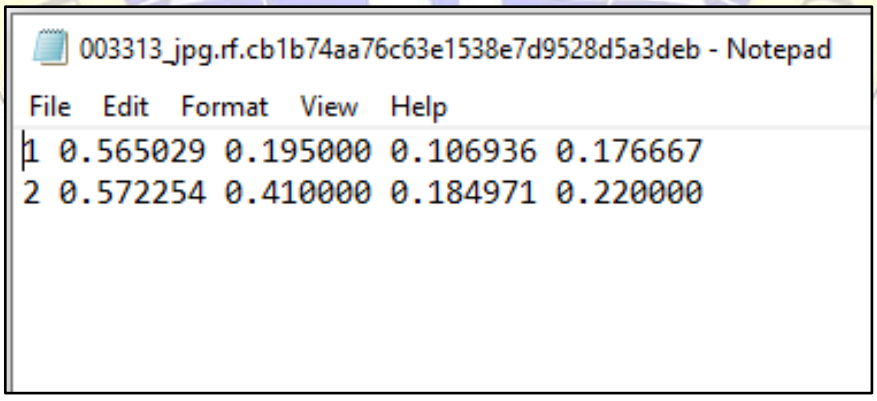
Gambar 4.10. Anotasi Gambar pada *LabelImg*

Kelas anotasi gambar sebelum adanya proses anotasi hanya memiliki 3 kelas yaitu *head, helmet*, dan *person* dan anotasi tersimpan dalam format *.xml. Setelah adanya proses ulang anotasi pada 1.173 *dataset* maka *dataset* dalam format *.xml telah berubah dengan memiliki empat kelas dan anotasi gambar juga tersimpan dalam format *.txt. Urutan anotasi berdasarkan nilai dalam format *.txt, dapat diartikan sebagai, 0 merupakan nilai kelas *head*, 1 merupakan nilai kelas *helmet*, 2 merupakan nilai kelas *vest*, dan 3 merupakan nilai kelas *no_vest*. Proses penyimpanan hasil anotasi gambar dari format *.xml ke *.txt, dilakukan karena metode You Only Look Once (YOLO) dapat membaca atau memproses data dalam format *.txt. Gambar 4.11. merupakan anotasi gambar dalam format *.xml yang sebelumnya masih terdiri dari 3 kelas yaitu *head, helmet*, dan *person.*

```
<annotation>
	<folder></folder>
	<filename>003313
_jpg.rf.cb1b74aa76c63e1538e7d9528d5a3deb.jpg</filename>
	<path>003313
_jpg.rf.cb1b74aa76c63e1538e7d9528d5a3deb.jpg</path>
	<source>
		<database>roboflow.ai</database>
	</source>
	<size>
		<width>346</width>
		<height>300</height>
		<depth>3</depth>
	</size>
	<segmented>0</segmented>
	<object>
		<name>helmet</name>
		<pose>Unspecified</pose>
		<truncated>0</truncated>
		<difficult>0</difficult>
		<occluded>0</occluded>
		<bndbox>
			<xmin>177</xmin>
			<xmax>214</xmax>
			<ymin>32</ymin>
			<ymax>85</ymax>
		</bndbox>
	</object>
</annotation>
```

Gambar. 4.11. Anotasi Gambar *.xml

Perbedaan antara format anotasi *.xml dan *.txt dapat dilihat berdasarkan isi atau susunan dari hasil anotasi yang dihasilkan. Anotasi dengan format *.xml memiliki hasil anotasi yang lebih kompleks dibandingkan dengan hasil anotasi *.txt yang cukup berisikan angka serta nilai dari x,y,w dan h dari hasil *bounding box*. Gambar 4.12. merupakan hasil anotasi dalam format *.txt.

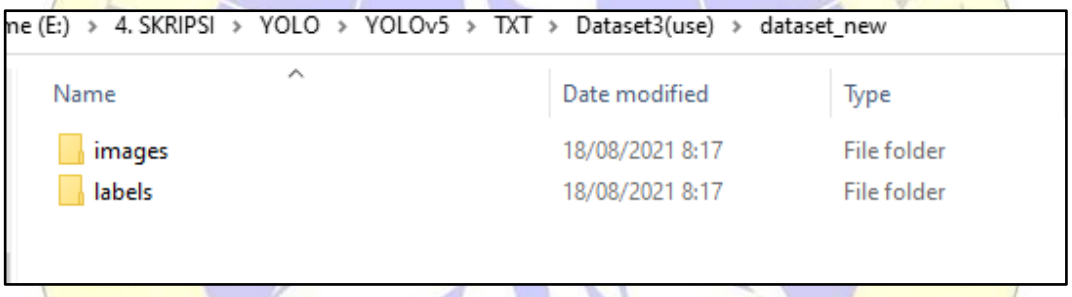


Gambar 4.12. Anotasi Gambar *.txt

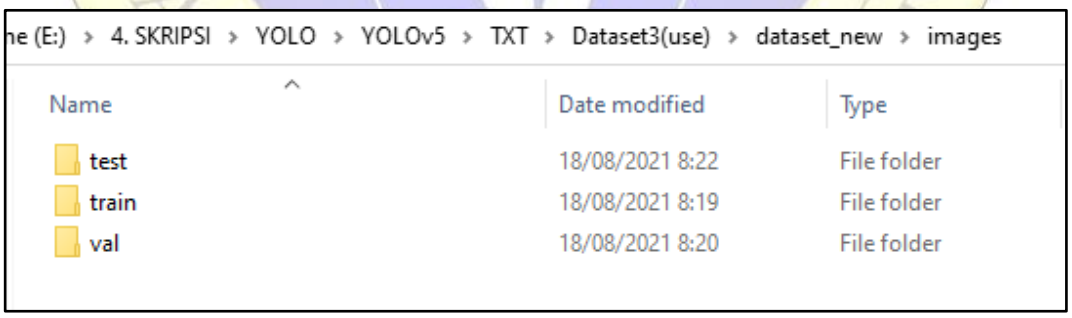
4.1.4.3. *Splitting a dataset*

Splitting a dataset merupakan *pre-processing* data yang bertujuan untuk membagi 1.173 *dataset* menjadi data *training* dan data *validation.* Data *training* dan data *validation* digunakan untuk melatih metode dengan tujuan metode

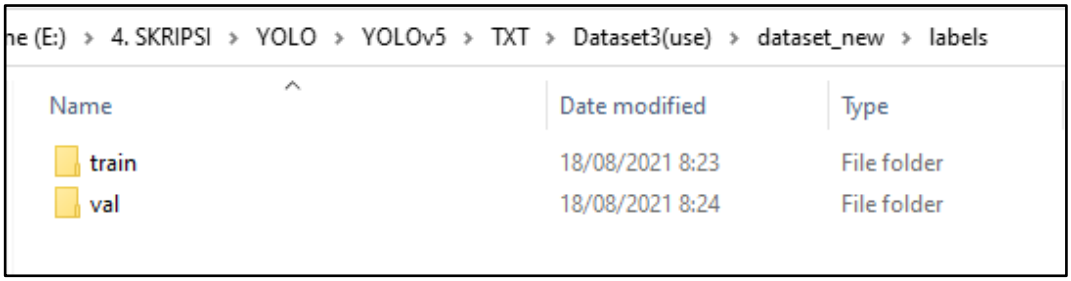
Y*ou Only Look Once* (YOLO) dapat mendeteksi keempat kelas yang telah disediakan sesuai dengan objek yang dituju. Pembagian data dilakukan dengan cara membagi 1.173 *dataset* ke dalam data *training* (*train*) sebesar 80% atau sebanyak 940 data gambar dan anotasi dan *validation* (*val*) sebesar 20% atau sebanyak 233 data gambar dan anotasi. Data tersebut disatukan dalam satu *folder* dengan nama *dataset_new*, dalam *folder* tersebut terdiri dari sub *folder images* untuk menyimpan data gambar dan *labels* untuk menyimpan data anotasi gambar. Dalam sub *folder images* terdiri dari 3 sub *folder train*, *val*, dan *test* dan *labels* terdiri dari 2 sub *folder train*, dan *val*. Gambar 4.13. merupakan gambar *folder* utama *dataset* dengan nama *dataset_new* dengan sub *folder images* dan *labels*, Gambar 4.14. merupakan gambar sub *folder images* yang berisikan *folder train*, *val*, dan test, sedangkan Gambar 4.15. merupakan sub *folder labels* yang berisikan *folder train*, dan *val*, karena data *test* tidak memerlukan anotasi gambar.



Gambar 4.13. *Folder dataset_new*

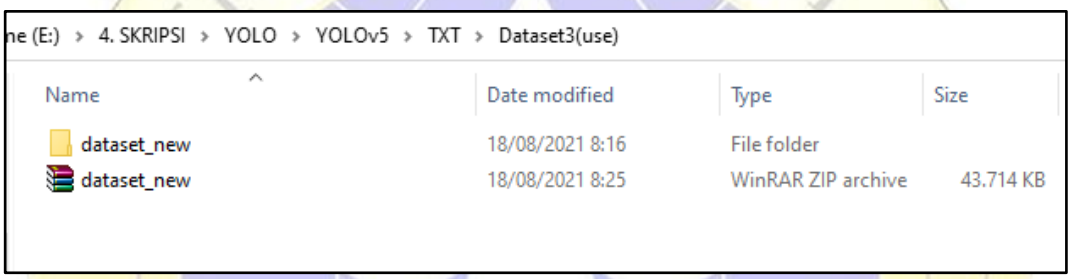


Gambar 4.14. Sub *Folder images*



Gambar 4.15. Sub *Folder labels*

Sebelum data tersebut diimplementasikan dengan metode *You Only Look Once* (YOLO), maka selanjutnya adalah *dataset* dengan nama *folder dataset_new* tersebut di ekstrak dalam format *.zip, sehingga *dataset* tersebut dapat direpresentasikan pada Gambar 4.16.



Gambar 4.16. Ekstrak *Dataset*

4.1.5. Hasil Implementasi Metode *You Only Look Once* (YOLO)

Tahapan ini merupakan tahapan yang memaparkan terkait pengolahan data yang telah diproses sebelumnya, ditujukan agar dapat mendeteksi objek sesuai dari keempat kelas yang telah ditentukan sebelumnya, yaitu *head*, *helmet*, *vest*, dan *no_vest*. Beberapa tahapan dalam implementasi *dataset* dengan metode *You Only Look Once* (YOLO) meliputi *setup YOLOv5*, memanggil *dataset*, melakukan proses *training dataset*, dan yang terakhir adalah melakukan proses *testing* data. Segala proses yang dilakukan dalam implementasi metode *YOLOv5* dilakukan melalui *Google Colaboratory*.

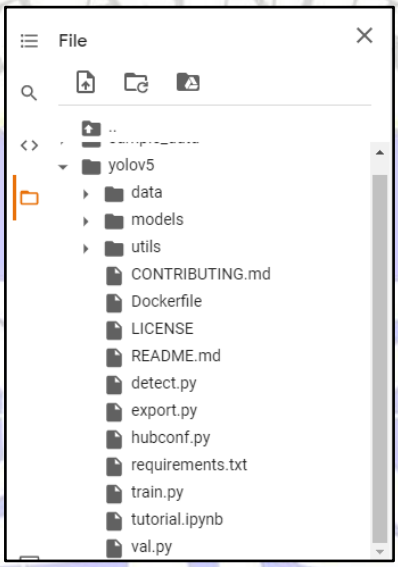
Dalam penelitian yang dilakukan, *dataset* diimplementasikan dengan menggunakan metode *YOLOv5*, sehingga langkah utama yang dapat dilakukan adalah setup atau mempersiapkan metode *YOLOv5* dengan melakukan proses *clone repository* atau penyimpanan *YOLOv5* yang dapat diakses melalui alamat berikut https://github.com/ultralytics/yolov5. Dengan menggunakan fungsi !git, maka

proses dalam klone penyimpanan dari YOLOv5 dapat dilakukan dengan mudah dan cepat. Segmen Program 4.1 merupakan proses dari klone dari *repository YOLOv5*.

Segmen Program 4.1. *Clone YOLOv5*

```
1. !git clone https://github.com/ultralytics/yolov5
```

Setelah proses klone dari penyimpanan *YOLOv5* berhasil, maka data file pada Google Colaboratory bertambah dengan adanya file baru yang bernama yolov5. Gambar 4.17. merupakan hasil klone dari YOLOv5.



Gambar 4.17. Hasil Klone *YOLOv5*

Setelah melakukan proses klone YOLOv5, agar dapat menggunakan fungsi dari file yolov5, maka diperlukan pemasangan atau instalasi *requirements.txt* pada *Google Colaboratory* seperti berikut ini. Beberapa source code dalam menyiapkan dalam penggunaan yolov5 juga dibutuhkan dimulai dari *import torch*, karena yolov5 diimplementasikan dalam *pytorch* dan juga mengatur performa kecepatan dalam memproses metode dengan penggunaan GPU dan CPU. Segmen Gambar 4.2. merupakan segmen program lanjutan dalam *setup YOLOv5*.

Segmen Program 4.2. *Setup YOLOv5*

```
%cd yolov5
%pip install -qr requirements.txt
import torch
from IPython.display import Image, clear_output
print(f"Setup complete. Using torch {torch.__version__} ({t
orch.cuda.get_device_properties(0).name if torch.cuda.is_av
ailable() else 'CPU'})")
```

Dataset yang merupakan bahan utama dalam penelitian ini sebelumnya telah tersimpan dalam *Google Drive* untuk mempermudah proses pemanggilan data yang diinginkan. Agar *dataset_new* dalam *Google Drive* dapat diakses dan digunakan dalam *Google Colaboratory*, maka diperlukan pemasangan *Drive* (penyimpanan) agar penyimpanan terpanggil dalam daftar isi file pada *Google Colaboratory* serta melakukan ekstrak dataset *.zip agar dataset yang tersimpan dalam file tersebut terbaca dalam proses pemanggilan data. Segmen Program 4.3. merupakan segmen program dalam pemanggilan *dataset_new*.

Segmen Program 4.3. Pemanggilan *Dataset*

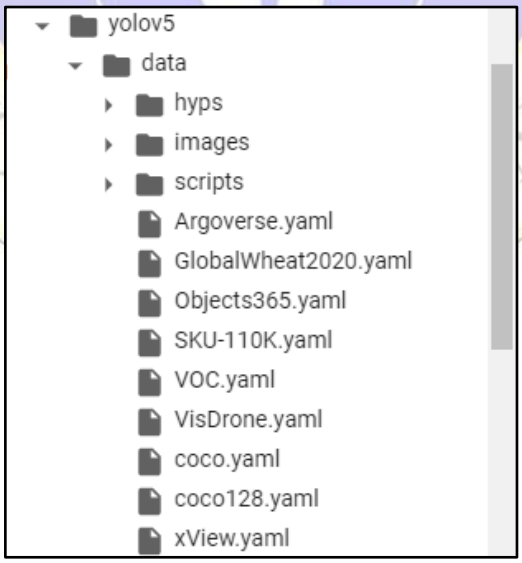
```
from google.colab import drive
drive.mount('/content/drive')
!unzip -
q /content/drive/MyDrive/Skripsi2021/Project/dataset_new.zip
 -d ../
```

Proses *training* merupakan proses dalam melatih metode *You Only Look Once* (YOLO) dengan menggunakan *dataset* yang telah dipersiapkan agar mendapatkan model yang dapat mendeteksi keempat kelas yang telah ditentukan sesuai objek yang dituju. Proses *training* dengan *YOLOv5* dilakukan dengan memanggil *file train.py* yang telah disediakan oleh *YOLOv5* hasil dari klon yang telah dilakukan sebelumnya, agar proses *training* data dapat dilakukan sebagaimana mestinya. Segmen Program 4.4. merupakan fungsi untuk menjalankan proses *training* melalui file *train.py*.

Segmen Program 4.4 *Training* Data

```
1. !python train.py --img 640 --batch 8 --epochs 50 --
   data coco128.yaml --weights yolov5s.pt --cache
```

Sebelum melakukan proses *training* data menggunakan Segmen Program 4.4. maka fungsi file dari *coco128.yaml* perlu diubah susunannya sesuai kebutuhan penelitian yang diinginkan, dimulai dari pemanggilan data, jumlah kelas yang ditentukan dan nama kelas yang juga telah ditentukan sebelumnya. *File coco128.yaml* ini berfungsi untuk pemanggilan data serta jumlah kelas dan nama kelas yang ingin diimplementasikan dengan menggunakan metode *YOLOv5*. Pemanggilan data disesuaikan dengan letak serta nama data yang telah disiapkan dalam *file dataset_new*, sedangkan jumlah kelas diubah menjadi 4 kelas, karena dalam penelitian ini hanya menggunakan 4 kelas yang dibutuhkan yang terdiri dari nama kelas *head, helmet, vest*, dan *no_vest*. Urutan dalam pemanggilan nama kelas dari data juga harus disesuaikan dengan urutan kelas yang dilakukan pada proses anotasi sebelumnya. Proses perubahan *file coco128.yaml* dapat dilakukan dengan cara mengakses *file coco128.yaml* yang berada dalam *file yolov5* yang terletak di dalam *file* data, setelah itu dapat kita buka *file coco128.yaml* dengan cara menekan dua kali *file* tersebut. Gambar 4.18. merupakan gambar dari letak *file coco128.yaml*, sedangkan Segmen Program 4.5. merupakan segmen program *dari coco128.yaml*.

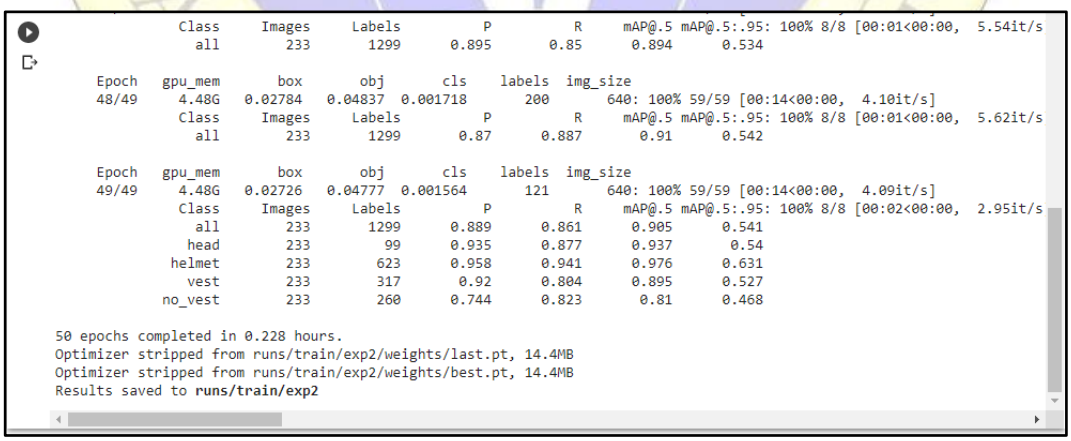


Gambar 4.18. *File coco128.yaml*

Segmen Program 4.5. *coco128.yaml*

```
1. train: /content/dataset_new/images/train/
2. val: /content/dataset_new/images/val/
3. nc: 4
4. names: [ 'head', 'helmet', 'vest', 'no_vest' ]
```

Pada saat melakukan proses training data dengan menggunakan metode *YOLOv5*, pengujian serta penentuan parameter juga harus dilakukan agar mendapatkan nilai akurasi sesuai yang diinginkan dalam penelitian untuk deteksi di setiap objek kelas yang ditentukan. Parameter yang digunakan dalam metode *YOLOv5* ini terdiri dari nilai *batch* serta jumlah *epoch* yang digunakan. *Batch* digunakan untuk menentukan proses pembagian data yang digunakan dalam proses *training* data, karena data yang digunakan memiliki jumlah yang cukup besar dan tidak memungkinkan untuk diproses dengan hanya sekali proses, sedangkan penentuan jumlah *epoch* dilakukan untuk menentukan berapa kali metode pembelajaran dilakukan dalam mengolah seluruh data yang tersimpan dalam *file train* dan *val*. Proses penentuan nilai dari parameter tersebut dapat dilihat dari Segmen Program 4.4. Ketika proses perubahan pada file *coco128.yaml* dan penentuan nilai dari parameter yang digunakan pada *YOLOv5* selesai dilakukan maka, selanjutnya proses *training* data dapat dilakukan, yang dirangkum pada Gambar 4.19.



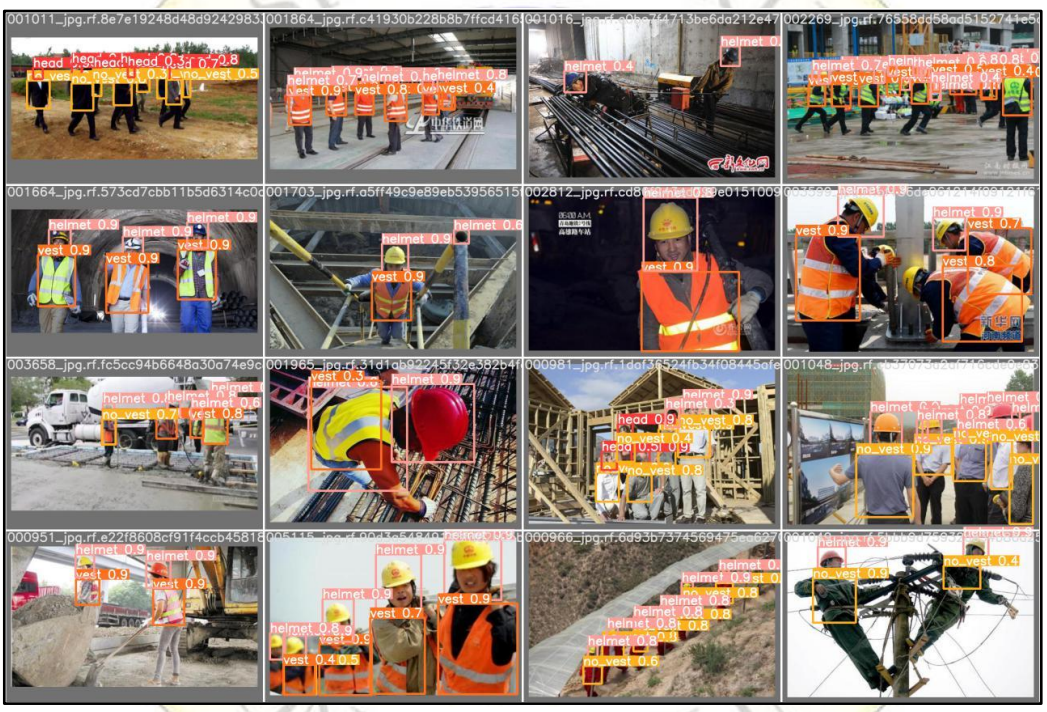
Gambar 4.19. Proses *Training*

Setiap hasil proses *training* data yang dilakukan secara otomatis tersimpan dalam *file* ***runs/train/exp,*** model pembelajaran secara otomatis akan tersimpan dalam format *best.pt* dan model tersebut yang akan digunakan dalam

proses uji coba. Hasil gambar dari proses *training* yang dilakukan pada data *train* dan *val* dapat dipanggil melalui Segmen Program 4.6. Setiap gambar yang di proses dalam *training dataset* juga dapat dilihat melalui Gambar 4.20.

Segmen Program 4.6. Memanggil Hasil Gambar *Training*

```
1. Image(filename='/content/yolov5/runs/train/exp2/train_batch0
   .jpg', width=800)
2. Image(filename='/content/yolov5/runs/train/exp2/val_batch0_l
   abels.jpg', width=800)
3. Image(filename='/content/yolov5/runs/train/exp2/val_batch0_p
   red.jpg', width=800)
```

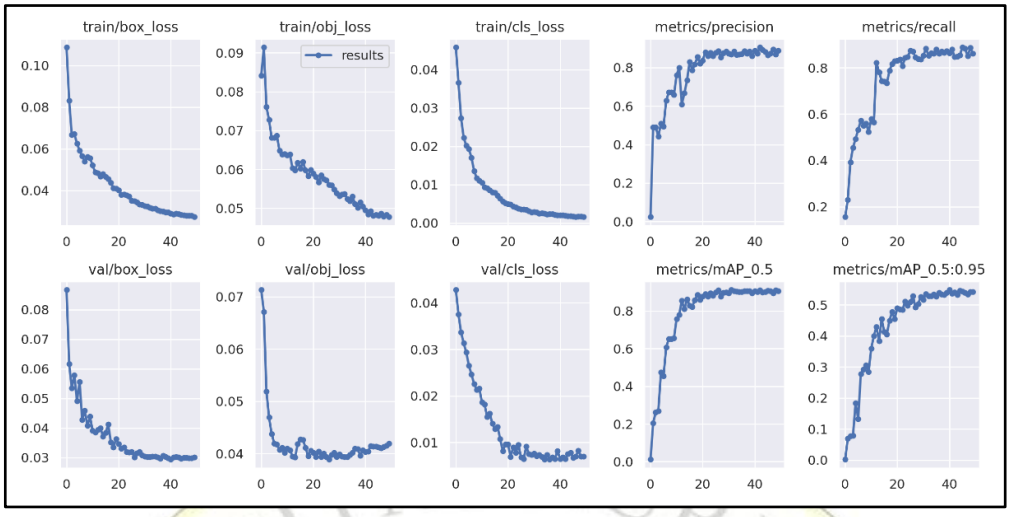


Gambar 4.20. Hasil *Training* Gambar

Visualisasi data dalam bentuk grafik yang dihasilkan dari proses *training* data dapat dipanggil dengan menggunakan Segmen Program 4.7. dengan hasil gambar yang ditunjukkan pada Gambar 4.21.

Segmen Program 4.7. Menampilkan Grafik Hasil *Training* Data

```
1. from utils.plots import plot_results
2. Image(filename='/content/yolov5/runs/train/exp2/results.png'
   , width=800)
```



Gambar 4.21. Hasil Grafik *Training*

4.1.6. Hasil Uji Coba

Uji coba atau *testing* data dilakukan untuk mengukur keberhasilan metode dalam mendeteksi objek berdasarkan nilai akurasi yang didapatkan setiap kelas objek yang dideteksi. Uji coba dilakukan pada 12 data gambar dan 1 data video yang dijadikan sebagai hasil dalam deteksi objek secara *real time*. Data gambar yang diuji tersimpan dalam *file test* dalam *dataset_new*. Data video tidak tersimpan dalam data *test* dikarenakan data video langsung dipanggil melalui alamat *youtube* dengan kata kunci "konstruksi" dan alamat tersebut dapat diakses melalui https://www.youtube.com/watch?v=vdk47BJ5gX4. Proses uji coba dilakukan dengan cara memanggil *file best.pt* yang didapatkan melalui proses *training* data sebelumnya. Segmen Program 4.8 merupakan fungsi untuk melakukan uji coba data *test* yang telah disiapkan baik berupa gambar dan video.

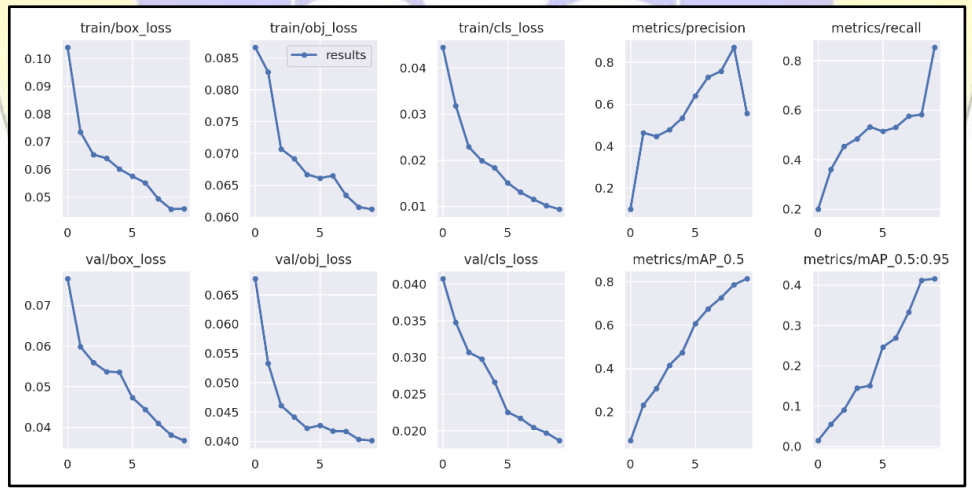
Segmen Program 4.8. *Testing* Data

```
1.  # Test Gambar
2.  !python detect.py --
    weights /content/yolov5/runs/train/exp2/weights/best.pt --
    img 640 --conf 0.60 --
    source /content/dataset_new/images/test

3.  # Test Video
4.  !python detect.py --
    weights /content/yolov5/runs/train/exp2/weights/best.pt --
    img 640 --conf 0.60 --source 'https://youtu.be/vdk47BJ5gX4'
```

Seluruh hasil uji coba tersimpan dalam **runs/detect/exp.** Pemanggilan data uji coba dapat menguji beberapa gambar yang tersimpan dalam 1 folder atau dapat diuji dengan cara pemanggilan satu persatu gambar yang ingin diuji coba. Dalam penelitian ini uji coba dilakukan terhadap beberapa gambar yang sudah disatukan dalam dengan nama *test*. Pemanggilan hasil deteksi pada uji coba gambar *test* sama dengan pemanggilan hasil dari *training* data yang dapat menggunakan perintah sama seperti pada Segmen Program 4.6. Tetapi dalam penggunaan segmen program tersebut juga harus disesuaikan dengan lokasi penyimpanan gambar dari hasil uji coba.

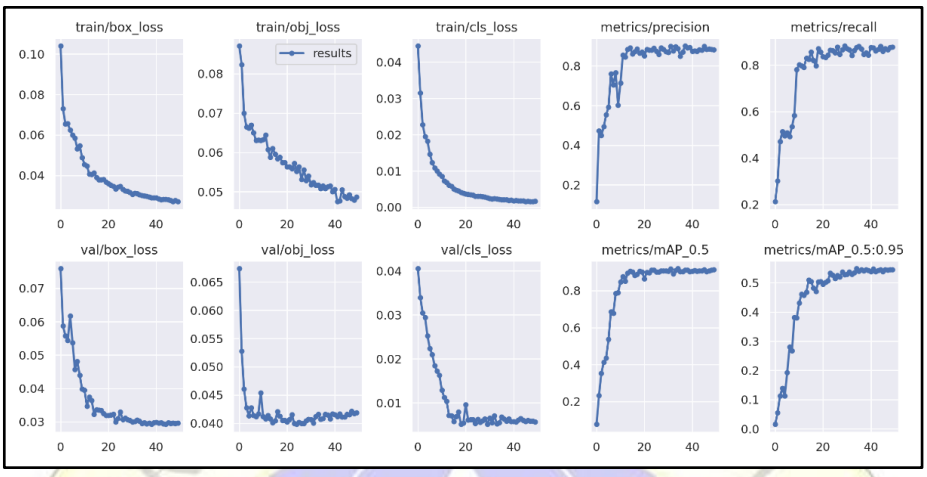
Dalam mendapatkan hasil akurasi yang didapatkan melalui implementasi data menggunakan metode *You Only Look Once* (YOLO) adalah dengan cara menguji data *test* yang ada. Untuk mendapatkan hasil akurasi yang diinginkan, penelitian ini membandingkan hasil akurasi dari setiap model yang dihasilkan dari beberapa nilai parameter yang diuji coba. Uji coba pertama dilakukan pada parameter *batch* = 8 dengan *epoch* = 10 yang menghasilkan grafik hasil uji coba seperti pada Gambar 4.22.



Gambar 4.22. Grafik Percobaan *Batch* 8 *Epoch* 10

Uji coba tidak dapat dilakukan dengan hanya sekali dalam menentukan nilai parameter yang baik. Beberapa nilai parameter dapat dicoba agar mendapatkan nilai akurasi terbaik dengan juga memperhatikan nilai *loss* dari data *train* dan *val* yang dilatih. Maka untuk membandingkan hasil grafik percobaan yang dari data latih, maka dilakukan perubahan nilai parameter yaitu dengan nilai *batch* = 8

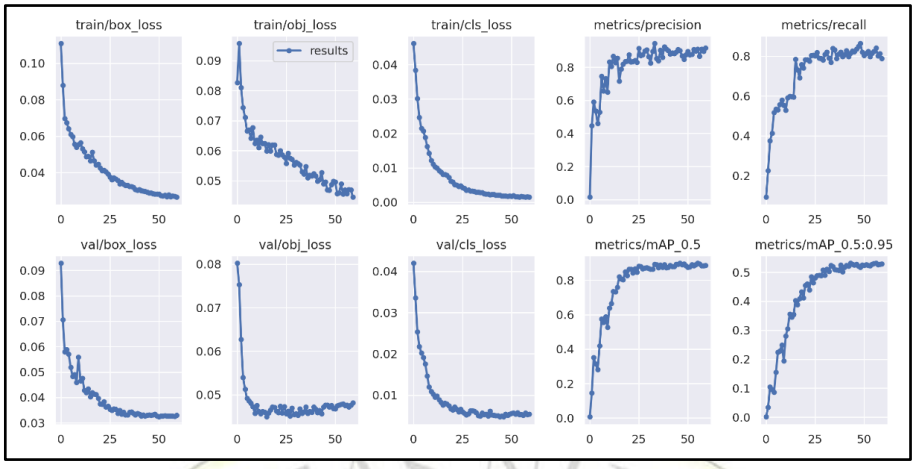
dengan *epoch* = 50 di percobaan kedua. Gambar 4.23. merupakan hasil dari grafik percobaan kedua.



Gambar 4.23. Grafik Percobaan *Batch* 8 *Epoch* 50

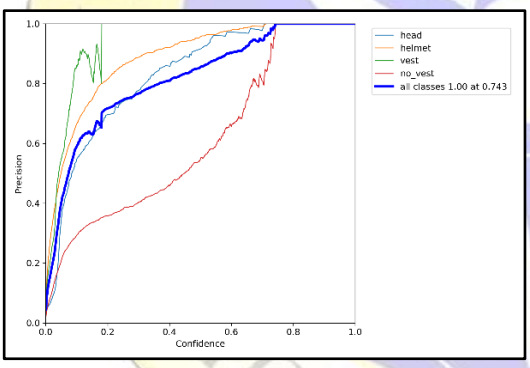
Apabila dibandingkan dengan hasil grafik percobaan pada *batch* 8 dengan *epoch* 10, maka hasil presisi dari penggunaan nilai parameter *batch* 8 dengan *epoch* 50 memiliki tingkat presisi yang lebih tinggi. Selain itu, nilai *loss* pada setiap data *train* dan *val* lebih kecil daripada percobaan menggunakan *batch* 8 dengan *epoch* 10 dari segi *box_loss* (kesalahan pada kotak), *obj_loss* (kesalahan pada object), dan *cls_loss* (kesalahan pada kelas objek). Dengan hal tersebut menandakan bahwasanya jumlah *epoch* sebanyak 50 yang dilatih maka, semakin besar hasil dari tingkat presisi dalam deteksi objek dan semakin rendah tingkat kesalahan dari setiap deteksi objek yang dilakukan.

Adapun yang dilakukan dalam percobaan ketiga adalah merubah nilai dari parameter batch dengan nilai 16 namun tetap dengan menggunakan nilai *epoch* sebesar 50, hal ini dikarenakan untuk menguji konsistensi dari nilai presisi yang dihasilkan dalam proses *training* menggunakan nilai *epoch* 50, karena hasil presisi dengan *epoch* 50 telah menghasilkan presisi yang cukup tinggi. Gambar 4.24. merupakan hasil dari grafik percobaan *batch* 16 dengan nilai *epoch* sebesar 50.

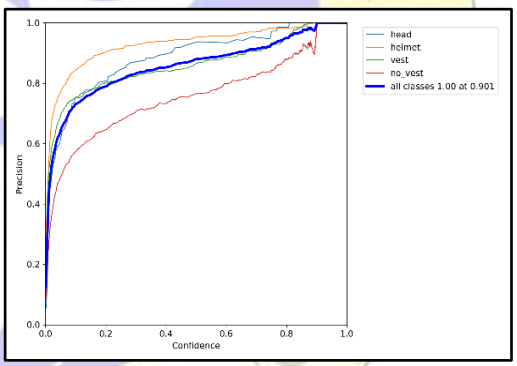


Gambar 4.24. Grafik Percobaan *Batch* 16 *Epoch* 50

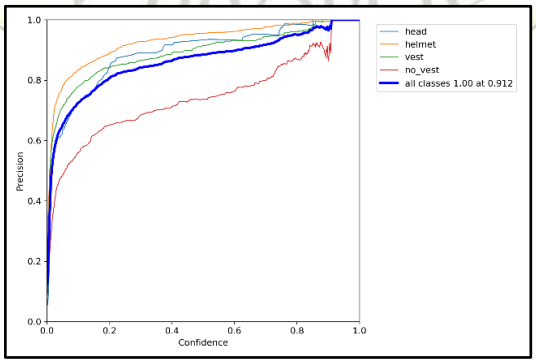
Perbandingan dari hasil presisi yang dilakukan dalam ketiga percobaan diatas juga dapat dibandingkan dengan melihat perbandingan grafik presisi dan *confidence* (kepercayaan) di bawah ini dari setiap kelas dan keseluruhan yang dideteksi. Gambar 4.25. merupakan perbandingan grafik presisi yang dihasilkan dari uji coba nilai parameter dari ketiga percobaan sebelumnya.



Batch 8 *Epoch* 10



Batch 8 *Epoch* 50



Batch 16 *Epoch* 50

Gambar 4.25. Perbandingan Presisi dari 3 Uji Parameter

Selanjutnya juga perlu dilakukan dalam membandingkan nilai kesalahan (*loss*) dari keseluruhan data objek, *box*, dan kelas dari data *train* dan *val* yang digunakan dalam proses *training*. Perbandingan tersebut disajikan dalam Tabel 4.3. Hasil perbandingan dari nilai kesalahan didapatkan bahwa nilai kesalahan paling kecil didapatkan dari parameter *Batch* 8 dengan *Epoch* 50.

Tabel 4.1. Perbandingan Nilai Kesalahan dari Uji 3 Parameter

No.	**Parameter**	**Nilai Kesalahan (*Loss*)**		
		Training		
		Objek	*Box*	Kelas
1.	Batch 8 Epoch 10	0.061208	0.045778	0.0093492
2.	Batch 8 Epoch 50	0.048642	0.027061	0.0016707
3.	Batch 16 Epoch 50	0.047774	0.02726	0.0015638
No.	**Parameter**	**Nilai Kesalahan (*Loss*)**		
		Validation		
		Objek	*Box*	Kelas
1.	Batch 8 Epoch 10	0.040106	0.036708	0.018646
2.	Batch 8 Epoch 50	0.041838	0.029613	0.0056703
3.	Batch 16 Epoch 50	0.041965	0.030118	0.0070332

Data uji yang digunakan dalam penelitian ini berjumlah 12 data gambar. Dalam data 12 gambar tersebut, terdiri dari 86 objek yang terdiri dari objek head, helmet, vest dan no_vest, dan 1 objek bernilai negatif dari setiap 12 data gambar. Data negatif didapatkan dengan cara membaca 1 objek bernilai negatif pada 12 data gambar yang tersedia. Data negatif tersebut juga digunakan untuk mengetahui hasil akurasi, presisi, dan sensitifitas pada model YOLOv5 yang telah dihasilkan pada

uji parameter sebelumnya. Rincian dari jumlah objek tersebut dipaparkan pada Tabel 4.2.

Tabel 4.2. Jumlah Keseluruhan Objek pada 12 Gambar Data Uji

No.	Objek	Jumlah Objek
1.	*Head*	13
2.	*Helmet*	24
3.	*Vest*	10
4.	*No_Vest*	27
5.	Objek Bernilai Negatif	12
Jumlah		**86**

Selanjutnya dalam menentukan parameter terbaik dari metode yang digunakan untuk mendeteksi objek APD dan Non APD, maka hasil dari uji coba 12 Gambar dihitung berdasarkan persamaan 3.1, 3.2, dan 3.3 untuk mendapatkan nilai akurasi, presisi, dan sensitifitas. Tabel 4.3. merupakan hasil dari data uji 12 gambar dengan menggunakan parameter *batch* 8 dan *epoch* 10. Tabel 4.4. merupakan hasil dari data uji 12 gambar dengan menggunakan parameter *batch* 8 dan *epoch* 50. Tabel 4.5. merupakan hasil dari data uji 12 gambar dengan menggunakan parameter *batch* 16 dan *epoch* 50.

Tabel 4.3. Uji Coba Gambar dengan *Batch* 8 dan *Epoch* 10

Nama	Jumlah Objek	Nilai				Keterangan	Akurasi	Presisi	*Recall*
		TP	FP	FN	TN				
coba1.jpg	3	1		1	1	1 helmet 1 negatif objek	67%	100%	100%
coba2.jpg	5	4			1	2 helmet 2 no vest 1 negatif objek	100%	100%	100%

Tabel 4.3. Lanjutan

Nama	Jumlah Objek	Nilai				Keterangan	Akurasi	Presisi	Recall
		TP	FP	FN	TN				
coba3.jpg	17	14		2	1	8 helmet 6 no_vest 1 negatif objek	88%	100%	87%
coba4.jpg	3	1		1	1	1 helmet 1 negatif objek	67%	100%	50%
coba5.jpg	7	3		3	1	3 helmet 1 negatif objek	57%	100%	50%
coba6.jpg	7	3		3	1	3 helmet 1 negatif objek	57%	100%	50%
coba7.jpg	3	2			1	1 head 1 no_vest 1 negatif objek	100%	100%	100%
coba8.jpg	9	8			1	4 head 4 no_vest 1 negatif objek	100%	100%	100%
coba9.jpg	15	12		2	1	7 head 5 no_vest 1 negatif objek	87%	100%	86%
coba10.jpg	5	2		2	1	1 helmet 1 no_vest 1 negatif objek	60%	100%	50%
coba11.jpg	5	3		1	1	2 helmet 1 no_vest 1 negatif objek	80%	100%	75%
coba12.jpg	7	4		2	1	3 helmet 1 no_vest 1 negatif objek	71%	100%	67%
Jumlah	**86**	**57**	**-**	**17**	**12**	**-**	**80%**	**100%**	**77%**

Tabel 4.4. Uji Coba Gambar dengan *Batch* 8 dan *Epoch* 50

Nama	Jumlah Objek	Nilai				Keterangan	Akurasi	Presisi	*Recall*
		TP	FP	FN	TN				
coba1.jpg	3	1		1	1	1 helmet 1 negatif objek	67%	100%	50%
coba2.jpg	5	4			1	2 helmet 2 no vest 1 negatif objek	100%	100%	100%
coba3.jpg	17	15		1	1	8 helmet 6 no_vest 1 negatif objek	94%	100%	94%
coba4.jpg	3	2			1	1 helmet 1 vest 1 negatif objek	100%	100%	100%
coba5.jpg	7	6			1	3 helmet 3 vest 1 negatif objek	100%	100%	100%
coba6.jpg	7	5		1	1	3 helmet 2 vest 1 negatif objek	86%	100%	83%
coba7.jpg	3	2			1	1 head 1 no_vest 1 negatif objek	100%	100%	100%
coba8.jpg	9	8			1	4 head 4 no_vest 1 negatif objek	100%	100%	100%
coba9.jpg	15	13		1	1	7 head 6 no_vest 1 negatif objek	93%	100%	93%
coba10.jpg	5	4			1	1 helmet 1 head 2 no_vest 1 negatif objek	100%	100%	100%

Tabel 4.4. Lanjutan

Nama	Jumlah Objek	Nilai				Keterangan	Akurasi	Presisi	Recall
		TP	FP	FN	TN				
coba11.jpg	5	4			1	2 helmet 1 no_vest 1 vest 1 negatif objek	100%	100%	100%
coba12.jpg	7	6			1	3 helmet 2 vest 1 no_vest 1 negatif objek	100%	100%	100%
Jumlah	**86**	**70**	**-**	**4**	**12**	**-**	**95%**	**100%**	**94%**

Tabel 4.5. Uji Coba Gambar dengan *Batch* 16 dan *Epoch* 50

Nama	Jumlah Objek	Nilai				Keterangan	Akurasi	Presisi	*Recall*
		TP	FP	FN	TN				
coba1.jpg	3	2			1	1 helmet 1 no_vest 1 negatif objek	100%	100%	100%
coba2.jpg	5	4			1	2 helmet 2 no vest 1 negatif objek	100%	100%	100%
coba3.jpg	17	15		1	1	8 helmet 7 no_vest 1 negatif objek	94%	100%	94%
coba4.jpg	3	2			1	1 helmet 1 vest 1 negatif objek	100%	100%	100%
coba5.jpg	7	6			1	3 helmet 3 vest 1 negatif objek	100%	100%	100%

Tabel 4.5. Lanjutan

Nama	Jumlah Objek	Nilai				Keterangan	Akurasi	Presisi	Recall
		TP	FP	FN	TN				
coba6.jpg	7	5		1	1	3 helmet 2 vest 1 negatif objek	86%	100%	83%
coba7.jpg	3	2			1	1 head 1 no_vest 1 negatif objek	100%	100%	100%
coba8.jpg	9	8			1	4 head 4 no_vest 1 negatif objek	100%	100%	100%
coba9.jpg	15	14			1	7 head 7 no_vest 1 negatif objek	100%	100%	100%
coba10.jpg	5	3		1	1	1 helmet 1 head 1 no_vest 1 negatif objek	80%	100%	75%
coba11.jpg	5	4			1	2 helmet 1 no_vest 1 vest 1 negatif objek	100%	100%	100%
coba12.jpg	7	6			1	3 helmet 2 vest 1 no_vest 1 negatif objek	100%	100%	100%
Jumlah	**86**	**71**	**-**	**3**	**12**	**-**	**96%**	**100%**	**96%**

Dari hasil data uji diatas didapatkan bahwa nilai akurasi, presisi, dan sensitifitas terkecil berdasarkan dari 12 data gambar didapatkan dari parameter *batch* 8 *epoch* 10, dengan hasil akurasi 80%, presisi 100%, dan sensitifitas 77%. Sedangkan hasil akurasi, presisi, dan sensitifitas terbesar didapatkan dari parameter

batch 16 *epoch* 50 mendapatkan hasil akurasi 96%, presisi 100%, dan sensitifitas 96%.

4.2. Analisis Hasil Uji Coba

Tahapan analisa digunakan untuk menganalisa data yang digunakan untuk mengetahui keberhasilan metode yang digunakan dalam mendeteksi keempat kelas yang meliputi kelas *helmet* atau *vest* yang merepresentasikan penggunaan Alat Pelindung Diri (APD), kelas *head* atau *no_vest* merepresentasikan tidak sedang menggunakan Alat Pelindung Diri (APD). Uji coba dilakukan terhadap 12 gambar data yang diambil secara acak dari dataset *Hard Hat Workers Datasets* yang tersedia di *Roboflow* di dalam *test*. Data *test* dipilih berdasarkan jumlah orang yang ada pada gambar, yaitu terdiri dari 1 orang, 2 orang, dan lebih dari 2 orang. Tabel 4.6. merepresentasikan pengelompokan data berdasarkan jumlah orang.

a. 1 Orang

b. 2 Orang

c. >= 2 Orang

Gambar 4.26. Kategori Data Berdasarkan Jumlah Orang Dalam Data *Test*

Keseluruhan data uji terangkum pada Gambar 4.27. yang digunakan dalam pengujian data. Terdiri dari 12 data gambar yang digunakan dalam uji coba model yang dihasilkan dari uji coba parameter sebelumnya.

Gambar 4.27. Data Uji

Jumlah objek dari setiap gambar dilakukan dan ditentukan secara mandiri, hal tersebut guna untuk mengetahui seberapa besar tingkat akurasi yang didapatkan oleh setiap model yang dihasilkan dalam percobaan parameter sebelumnya. Gambar 4.28. merupakan proses uji coba berdasarkan posisi *ground truth* (kebenaran dasar) dan hasil deteksi objek. Pada Gambar 4.28. *ground truth* direpresentasikan dengan garis berwarna hijau dan merah. Pewarnaan *ground truth* dibedakan menurut kelas dan dibedakan dengan warna hasil deteksi agar dapat dibandingkan dengan jelas. *Ground truth* berfungsi untuk mengetahui letak dari objek yang dituju untuk dideteksi agar mempermudah proses perhitungan dari hasil deteksi objek pada data uji untuk menentukan hasil akurasi, presisi, dan sensitifitas.

Gambar 4.28. Hasil *Ground Truth* dan Deteksi Objek

Berdasarkan hasil uji coba yang dilakukan dalam menentukan akurasi, presisi, dan sensitifitas terbaik, maka hasil akurasi, presisi, dan sensitifitas terbesar didapatkan dari parameter *Batch* 16 *Epoch* 50 sebesar akurasi 96%, presisi 100%, dan sensitifitas 96%. Dan akurasi, presisi, dan sensitifitas terkecil didapatkan dari

parameter *Batch* 8 *Epoch* 10 sebesar akurasi 80%, presisi 100%, dan sensitifitas 77%. Hasil kesalahan terkecil dalam deteksi objek, *box*, dan kelas didapatkan dari parameter *Batch* 8 *Epoch* 50. Hasil akurasi, presisi, dan sensitifitas dari parameter *Batch* 8 *Epoch* 50 juga memiliki nilai yang baik, hanya selisih ± 1% dengan nilai akurasi, presisi dan sensitifitas yang didapatkan oleh parameter *Batch* 16 *Epoch* 50. Hasil akurasi, presisi, dan sensitifitas dari *Batch* 8 *Epoch* 50 masing-masing sebesar akurasi 95%, presisi 100%, dan sensitifitas 94%. Sehingga, model terbaik yang dihasilkan dalam pengujian data dengan menggunakan metode *You Only Look Once* (YOLO) versi 5 didapatkan dari parameter *Batch* 8 dengan *Epoch* 50, karena memiliki keunggulan lainnya dalam memiliki nilai kesalahan (*loss*) terkecil. Nilai kesalahan (*loss*) juga merupakan nilai yang dipertimbangkan dalam menentukan model terbaik dengan menggunakan metode YOLOv5. Selanjutnya model dari parameter *Batch* 8 *Epoch* 50 tersebut, diimplementasikan dalam pengujian video untuk dapat melihat hasil deteksi objek dari keempat kelas. Gambar 4.29. merupakan rangkuman dari hasil uji coba *Batch* 8 *Epochs* 50. Implementasi model terbaik yang telah dihasilkan hanya untuk mengukur tingkat keberhasilan metode dalam mendeteksi keempat objek dalam video secara tepat atau sesuai dengan objek yang dituju.

Gambar 4.29. Hasil Uji Coba Video pada Parameter *Batch* 8 *Epoch* 50