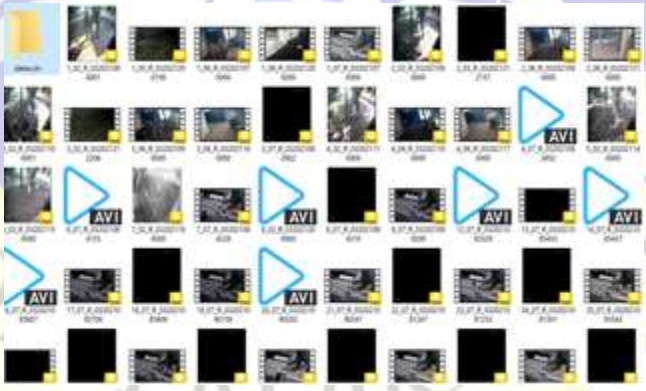
BAB IV
HASIL

4.1 Penyajian Data Uji Coba

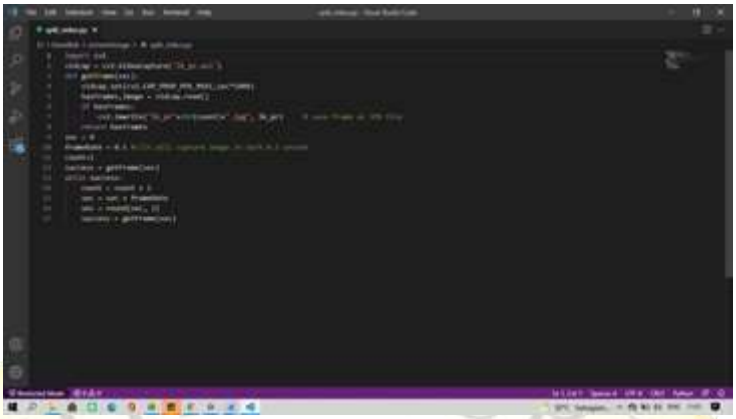
Pada sub bab ini akan dijelaskan lebih rinci mengenai rancangan penelitian Pada pembahasan sebelumnya, adapun hasil uji coba pengenalan jenis kelamin Pondok Pesantren Nurul Jadid yang telah diuji dengan menggunakan metode *Faster R-CNN* dan membandingkan dengan metode YOLO, berikut ini penjelasan lebih rinci mengenai pembahasan di atas.

4.2 Pengumpulan *Dataset*

Pada penelitian ini *dataset* yang digunakan berupa hasil video CCTV yang sudah di *convert* menjadi *image* menggunakan coding *python* lalu *dataset image* dipakai dengan jumlah 2000 data citra, dimana beberapa sumber data didapatkan dari hasil video CCTV keamanan Pondok Pesantren Nurul Jadid, adapun *image* laki laki dengan jumlah 1431 data, dan *image* perempuan dengan jumlah 1043 dan degan hasil *training* berkisar 2055, dan data *testing* 377. Data *testing* yang diuji cobakan yaitu beberapa *dataset* video CCTV Pondok Pesantren Nurul Jadid, sekaligus video dari halaman youtube beberapa Pondok Pesantren luar , Dan adapun deteksi *image* yang di buat untuk *testing* juga dia ada beberapa uji coba ambil di halaman web, Gambar 4.1. menunjukkan hasil video yang telah di *convect* dengan coding *python*.



Gambar 4.1 *Dataset* Video CCTV



Segmen Program 4.2 *Coding python split* data video ke *image*

Gambar 4.3 Hasil *Convert* Video ke *image*

4.3 *Pre-Processing*

Pada proses *Pre-Prosesing* Setelah *dataset* sudah diolah maka langkah selanjutnya yaitu *labeling* data format xml. Dimana data tersebut akan dibuat data *training* dan data *testing* lalu diimplementasikan ke metode *Faster R-CNN* dan YOLO. Adapun untuk mendapatkan hasil xml dengan *annotation* atau pelabelan data.

4.3.1 Annotation

Langkah yang akan dilakukan dengan split video atau meng *convert* ke *frame* lalu langkah selanjutnya menggunakan aplikasi tool *labeling* untuk menentukan *annotation* untuk *bounding box* pada masing-masing kelasnya. Hal ini dibutuhkan oleh *Faster R-CNN* sebagai sifatnya membutuhkan RoI *(Region of Interest)* yang dapat dipelajari sebagai dasar untuk mencari *Region* yang akan di konvolusikan. Dalam kasus yang dimiliki ada 2 kelas *annotation* dimana kelas-kelas seperti yang dijelaskan sebelumnya adalah jenis kelamin/ *gender* secara busana *full body* untuk 2 *class* ini akan dilakukan *training* untuk mendapatkan model yang diinginkan yaitu untuk

klasifikasi *gender* secara busana. Pelabelan citra adalah tahap pertmana bagian *dataset input* yang sudah diberikan label atau pengenal *class* dengan tujuan untuk menyimpan informasi citra ke selanjutnya disimpan dalam berkas PASCAL VOC atau YOLO dan menghasilkan hasil dengan format XML atau TXT . untuk tujuan konversi *dataset* ke berkas 'Annotation.txt atau xml'. Setelah proses konversi berkas XML atau YOLO dengan *output* hasil berupa file data CSV agar digunakan untuk *feeding* data pada proses *training*, Pelabelan dilakukan secara manual terhadap 2880 dataset *frame* menggunakan *tool labelling* seperti dapat dilihat pada gambar

Gambar 4.4 *Tool annotation*

Gambar 4.5 Hasil *annotation* atau *lebeling* data xml

4.4 Hasil Implementasi Metode *Faster RCNN*

Pada sub bab ini akan dijelaskan hasil dari implementasi metode *Faster Region Neural Network* (*FASTER R-CNN*) dengan menggunakan Google Collab yang terdiri dari beberapa tahapan:

4.4.1 *Tensorflow*

Tensorflow adalah *framework* dalam pembelajaran *machine learning* dibuat dalam mendukung dari beragam bahasa pemrograman dan dimana mempermudah *developer* melakukan pembelajaran mesin (Devikar., 2016). *Tensorflow* juga memungkinkan sebuah *user* agar membuat model *machine learning* sendiri sesuai dengan kebutuhan yang dibutuhkan *developer*. Adapun tingkatan *toolkit tensorflow*.


```
[3]   %tensorflow_version 1.x
      TensorFlow 1.x selected.
```

Segmen Program 4.1 Instal Tensorflow

4.4.2 Build File Protobuf

Bertujuan untuk *encode* data sebelum dikirimkan ke *encode* dilakukan agar mencapai efisiensi pengiriman data. Karena menyusutkan ukuran blok data dan meningkatkan kecepatan saat pengiriman. Tujuan awalnya adalah untuk menyederhanakan protokol *request and response*. Karena memiliki format khusus (*proto*), *protobuf* juga sering digunakan sebagai basis untuk pembuatan kode otomatis untuk *serialize* and *deserialize*. Jadi *protobuf* adalah cara untuk membuat serialisasi data terstruktur menjadi aliran *biner* secara cepat dan efisien. Adapun perintah *cell* yaitu *mengenerate* (menghasilkan) file berisi *code python* yang dapat di *import* untuk membuat serialisasi data, *protoc* singkatan dari *protocol buffer compiler*, *protoc* memberikan kinerja tinggi dari pada standar lain seperti XML dan JSON. *protoc* mendukung kode yang dihasilkan dalam berbagai bahasa termasuk bahasa *python*.

```
!apt-get install protobuf-compiler python-pil python-lxml python-tk
!pip install Cython

%cd /content/gdrive/MyDrive/tensorflow/models/research/
!protoc object_detection/protos/*.proto --python_out=.
```

Segmen Program 4.2 Build File Protobuf

4.4.3 Konfigurasi PYTHONPATH

Konfigurasi *pythonpath* bertujuan untuk menambahkan *directory path* yang menjadi tempat dimana modul atau *package* yang dibutuhkan *python* berada adapun gambar coding dibawah ini:

```
[ ] import os
    os.environ['PYTHONPATH'] += ':/content/gdrive/MyDrive/tensorflow/models/research/:/content/gdrive/MyDrive/tensorflow/models/research/slim'
```

Segmen Program 4.3 Konfigurasi PYTHONPATH

4.4.4 Install dan *Build* semua diperlukan oleh System

Adapun Install dan *build* ini bertujuan agar dapat melakukan pengujian apakah semua yang telah disetting sebelumnya sudah berjalan dengan baik.

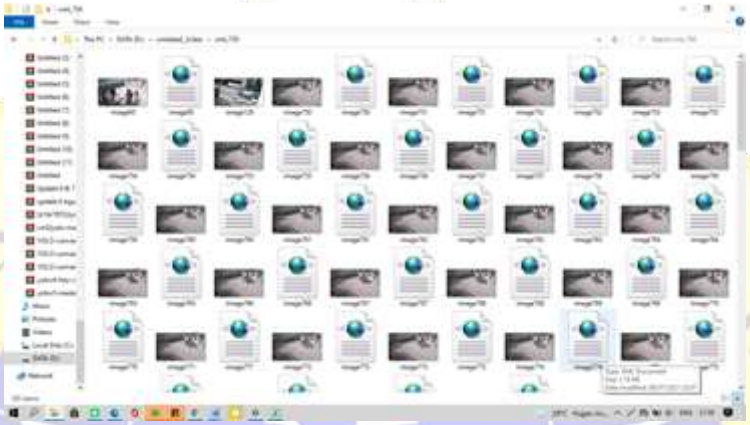
```
%cd /content/gdrive/MyDrive/tensorflow/models/research/slim/
!python setup.py build
!python setup.py install

/content/gdrive/MyDrive/tensorflow/models/research/slim
running build
running build_py
running egg_info
writing slim.egg-info/PKG-INFO
writing dependency_links to slim.egg-info/dependency_links.txt
writing requirements to slim.egg-info/requires.txt
writing top-level names to slim.egg-info/top_level.txt
reading manifest file 'slim.egg-info/SOURCES.txt'
writing manifest file 'slim.egg-info/SOURCES.txt'
running install
```

Segmen Program 4.4 Install dan Build

4.4.5 Konversi *Dataset*

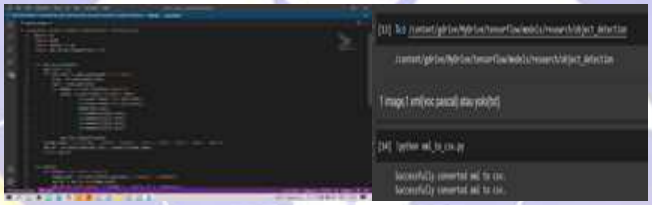
Adapun langkah selanjutnya yaitu mentraining data dari data *image* dan xml untuk mendapatkan data csv, dan adapun data *image* dan xml dibuat untuk *training* dan data *testing* digabung, lalu di *generate* ke format csv yang akan kedalam format *TF Record*. Bertujuan untuk

mengubah *dataset* ke dalam format *Biner* agar pelatihan itu semakin optimal. Data *biner* memakai lebih sedikit ruang pada *disk*, membutuhkan lebih sedikit waktu untuk menyalin dan dapat dibaca jauh lebih efisien. *TF Record* juga memungkinkan untuk menyimpan data *Sequence*, untuk konfigurasi *dataset* ke format *TFRecord*, ada kode yang harus disesuaikan yaitu *label*, atau *labelmap name* sesuai dengan *dataset* yang dimiliki dengan langkah dibawah.

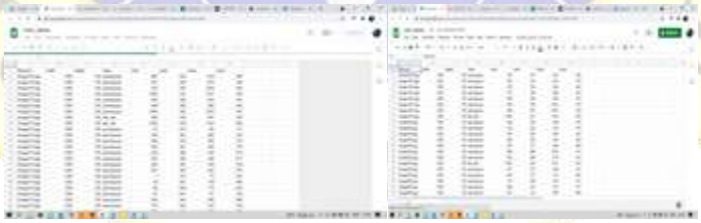


Gambar 4.6 Dataset *image* dan xml di folder *training* dan *testing*

Gambar 4.6. menunjukkan kumpulan dari *image* yang digunakan untuk proses *training* beserta file xml dari hasil pelafalannya. File ini nantinya akan dibuat menjadi file "train.record" yang akan dimasukkan kedalam sistem untuk dijadikan data *training system.*



Segmen Program 4.5 Code xml ke csv



Gambar 4.7 Hasil train_lebel.csv sama test_lebel. csv

4.4.6 Proses *Training*

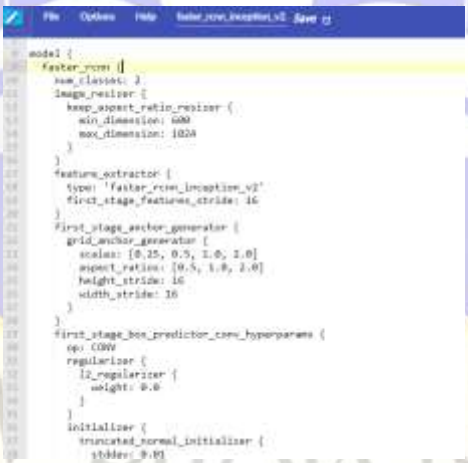
adapun proses *training* model yaitu dengan *generate* beberapa kode yang bisa menjalankan dan mengelola hasil *dataset* hingga menghasilkan sebuah data latih, adapun *step* selanjutnya dibawah ini

4.4.7 *Generate Tfrecord*

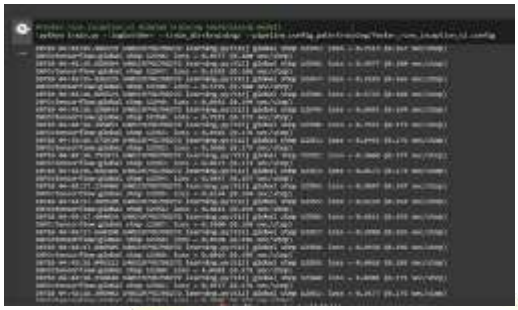
Setelah *generate* menjadi bagian file csv yaitu “train_labels.csv” dan “test_label.csv” yang berisi informasi semua dari *image* yang akan dijadikan data train.record dan test.record, contoh *generate* ini dapat dilihat pada Gambar 4.9.

Segmen Program 4.6. Hasil *generate*

Adapun setelah proses *generate* train.record dan test record maka langkah selanjutnya membuat file *labelmap.pbtxt.* dan konfigurasi model Faster_rcnn_inception_v2 di dalam folder file *training*



setelah mendapatkan hasil *training* yaitu langkah selanjutnya bagaimana coding untuk mentraining model adapun koding dibawah ini

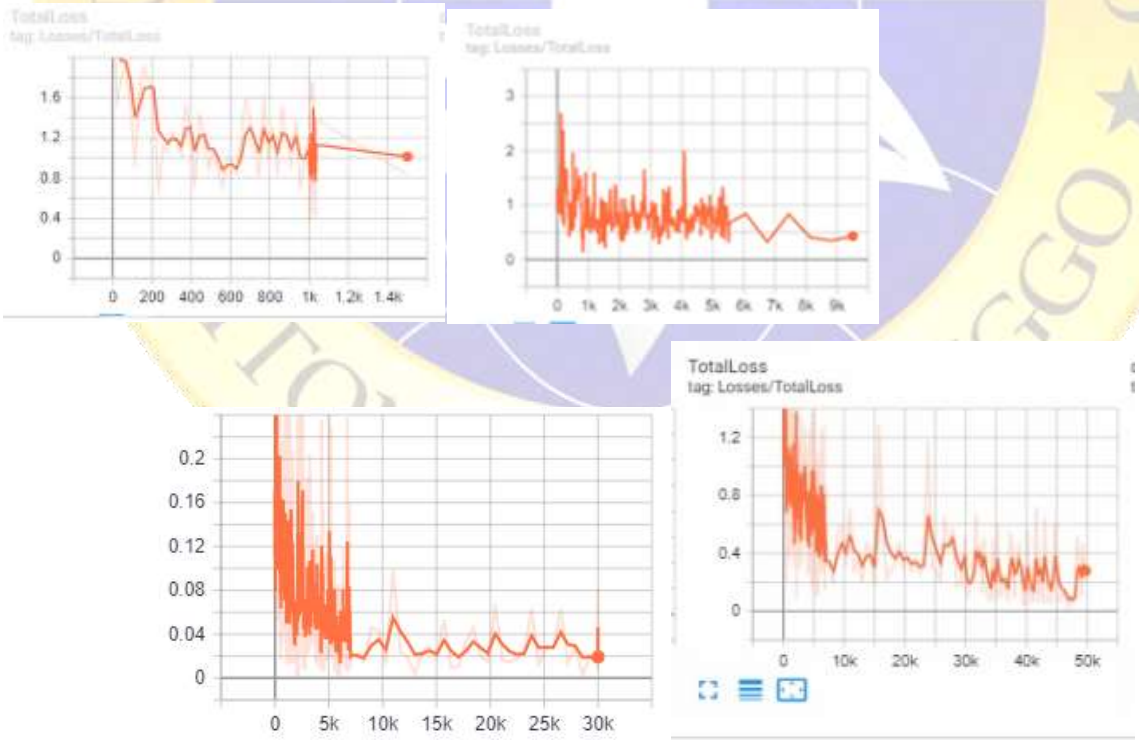


Segmen Program 4.7. *Iterasi training* data

Setelah model didapat maka selanjutnya bagaimana mendapatkan hasil model *training* dan di cekpoint dengan model.ckpt .adapun langkah terlebih dahu pengujian epoch

4.4.8 Pengujian *Epoch*

Setelah mentraining model maka tahap yang kerap dilakukan melihat hasil beberapa epoch yang dicoba untuk mendapatkan hasil optimal adapun beberapa uji coba epoch terhadap performa map dan loss dari model yang sudah dibuat di bagian file faster_rcnn_inception_v2 lalu di mengexport model.Export model training ini digunakan sebagai kesimpulan akhir dari model,dan model siapa digunakan yang mana akan menghasilkan output model format.pb yang berisi classifier untuk mendeteksi objek.hasil pengujiannya dibawah ini



Gambar 4.8 *Hasil grafik Total Loss nya*

Melihat grafik semakin tinggi *step* nya atau uji coba nya tinggi maka semakin sedikit proses *loss* nya. Selanjutnya tahapan uji coba per num_step

Tabel 4.2 Hasil uji coba *num step*

Num Step	Initial_learning rate	Batch_Size	Total loss
1000	**0.0002**	**1**	**0.5645**
2000	**0.0002**	**1**	**0.5441**
3000	**0.0002**	**1**	**0.4825**
5000	**0.0002**	**1**	**0.4367**
8000	**0.0002**	**1**	**0.4533**
10000	**0.0002**	**1**	**0.4409**
15000	**0.0002**	**1**	**0.2567**
20000	**0.0002**	**1**	**0.2576**
30000	**0.0002**	**1**	**0.0566**
50000	**0,002**	**1**	**0,1118**

Adapun uji coba *learning rate* dilakukan untuk menguji pengaruh *learning rate* terhadap performa *Map* dan *loss* dari model yang dibuat, adapun hasil pengujian *epoch* terhadap performa map dan *loss* model yang dibuat, semakin tinggi *num step* nya maka *error* atau kesalahan data semakin sedikit, dan semakin kecil learning rate nya maka akan overgan dan akan mendapatkan keakuratan data. Setelah menentukan *epoch* atau *num step* nya mendapatkan tingkat akurasi yang terbaik, langkah selanjutnya yaitu tahap uji coba dari *num-step* 50000 dari model *Faster R-CNN*, dari data *training* 2880 dan data *testing* 377, adapun data uji coba dari beberapa tempat dan beberapa video CCTV yang di extrak ke gambar. Dari 30 gambar dan beberapa CCTV dengan format jpg,

Gambar 4.9 Hasil Uji coba data *testing*

4.5 Uji Coba

4.5.1 Uji Coba *Faster RCNN*

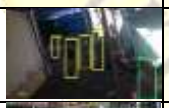
Setelah *dataset* disiapkan dan data uji coba yang dilakukan menggunakan *epoch* 30000 karena melihat ke erroran nya lebih sedikit dari uji coba *epoch* yang lain, adapun data *testing* dengan metode *Faster R-CNN* yang telah diimplementasikan, maka akan dilakukan uji coba kembali dengan menggunakan beberapa *dataset* Video CCTV Uji coba ini akan dilakukan pada perangkat prosesor Intel(R)Core(TM)i3-7020U3G CPU @2.30GHz, dengan memori 4096 MB RAM. Sistem operasi yang digunakan Windows 10 Pro 2019 64-bit. pada tahapan yang dilakukan untuk menentukan apakah metode berhasil, dari coding gambar g1. disajikan data beberapa frame dengan hasil tingkat akurasi pada tahapan implementasi dengan menggunakan metode *Faster RCNN*, adapun hasil sesuai dengan model dimana yang telah dibuat, kelas laki_laki dipresentasikan dengan nilai 0, dan model perempuan dengan nilai *array* 1,

```
import os
import cv2
import numpy as np
import tensorflow as tf
import sys


# This is needed since the notebook is stored in the object_detection folder.
sys.path.append("..")

# Import utilities
from utils import label_map_util
from utils import visualization_utils as vis_util
from google.colab.patches import cv2_imshow

# Name of the directory containing the object detection module we're using
MODEL_NAME = 'inference_graph'
IMAGE_NAME = '/content/gdrive/MyDrive/tensorflow/models/research/object_detection/testing/ujicoba/ground1.jpg'

# Grab path to current working directory
CWD_PATH = os.getcwd()

# Path to frozen detection graph .pb file, which contains the model that is used
# for object detection.
PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')

# Path to label map file
PATH_TO_LABELS = os.path.join(CWD_PATH,'training','labelmap.pbtxt')
```

Segmen program 4.8 untuk menghasil kan *output* gambar

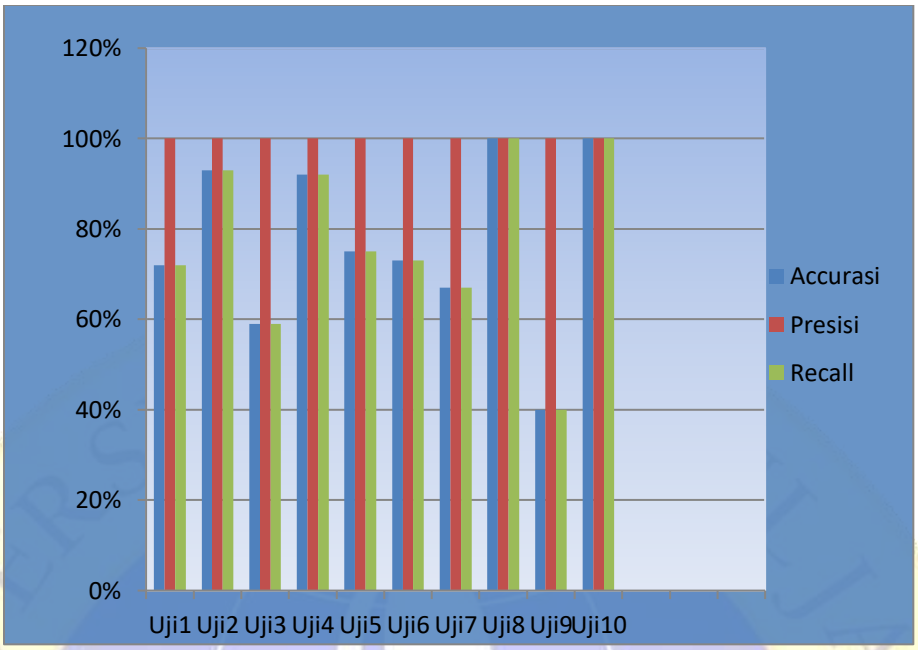
Dari data uji coba, metode masih belum terdeteksi dengan baik pada gambar keseluruhan, objek yang tidak terdeteksi kebanyakan adalah objek yang terjangkau jauh dari kamera CCTV dan tidak adanya model *training dataset* seperti halnya model kerudung dan juga sedikitnya *dataset* yang tidak memakai kopyah. Hasil uji *parameter* pada seluruh kategori data ada pada Tabel 4.2.

Tabel 4.3. Hasil Rincian kategori data gambar dari beberapa video

Nama Urut	Nama File	Total Objek	Nilai Prediksi				Akurasi	Presisi	*Recall*	Waktu Komputasi (*second*)
			TP	TN	FP	FN				
Uji1		7	5	0	0	2	72%	100%	72 %	0.98
Uji2		14	13	0	0	1	93%	100%	93%	0.71
Uji3		12	7	0	0	5	59%	100%	59%	0.80
Uji4		12	11	0	0	1	92%	100%	92%	0.53
Uji5		4	2	1	0	1	75%	100%	75%	0.35
Uji6		7	4	1	0	3	73%	100%	73%	0.70

Uji7		6	4	0	0	2	67%	100%	67%	0.50
Uji8		4	4	0	0	0	100%	100%	100%	0.45
Uji9		5	2	0	0	3	40%	100%	40%	0.32
Uji10		15	15	0	0	0	100%	100%	100%	0.80
Rata – Rata							77%	100%	77%	0,55

Pada Tabel 4.2 hasil uji coba dari metode *Faster* , pada semua data beberapa video CCTV dan beberapa keadaan menunjukan bahwa rata-rata tingkat akurasi yang dicapai sebesar 77%, tingkat presesi sebesar 100%, dan rata-rata *recall* yang dicapai sebesar 77% dan rata-rata komputasi sebesar 0,55%. Hasil data yang telah disebutkan maka peneleti menentukan, tingkatan *accuracy* mengalami *sensitive* atau menurun jika dalam keadaan sepi, dan objek yang didapat dari perempuan ada beberapa yang tidak kedeteksi seperti halnya anak perempuan yang masih kecil karena data perempuan yang tidak berkerudung di dataset training tidak, dan adapun hasil uji coba di tempat CCTV yang berbeda seperti halnya tempat sambang atau penerimaan tamu santri maka algoritma yang bekerja mendapatkan objek perempuan dengan gaya model yang berbeda seperti halnya kerudung,karna dataset gaya berbeda kerudung pun masih kurang maka hasil accurasinya menemukan tingkatan yang menurun,Adapun Metode *Faster RCNN* akan menghasilkan metode yang baik jika objek yang didapat sama dengan model didataset yang sudah ada didataset Gambar 4.12 menunjukkan grafik hasil dari *accuracy Faster R-CNN*.

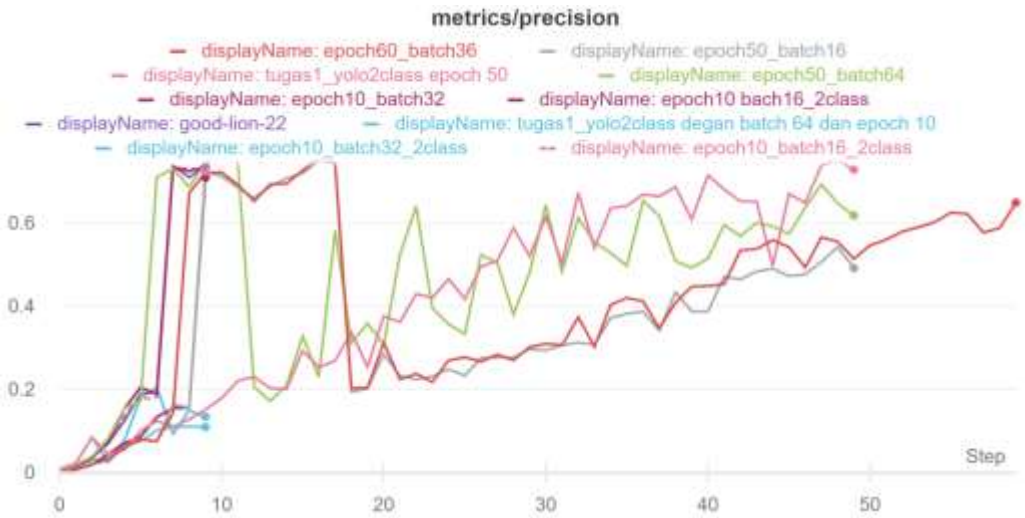


Gambar 4.10 Grafik hasil *accuracy Faster RCNN*

Setelah melihat jumlah rata-rata dari setiap komputasi dari metode *Faster Region Neural Network* (R-CNN) juga mengalami perubahan setiap kategori uji coba dari beberapa video CCTV, dari setiap kondisi yang ramai, sepi dan normal.

4.5.2 Uji Coba Metode *You Only Look Once* (YOLO)

Adapun Program YOLO yang akan dibandingkan mengacu pada penelitian yang dilakukan oleh (Jonathan Adiwibowo 2019) dengan *source code* yang dibuat oleh (https://github.com/ultralytics/yolov5). *Library* yang dibutuhkan untuk menjalankan program ini diantaranya adalah OpenCV 3.4, Python 3.6, Tensorflow-gpu 1.5.0 dan Keras 2.1.3. Program ini dijalankan dengan perangkat prosesor Intel(R)Core(TM)i3-7020U3G CPU @2.30GHz, dengan memori 4096 MB RAM. Sistem operasi yang digunakan Windows 10 Pro 2019 64-bit. pada tahapan yang dilakukan untuk menentukan apakah metode berhasil dipakai apa tidak maka,pada tahapan ini, metode *You Only Look Once (*YOLO) akan diuji dengan data yang sama, kemudian setelah proses uji coba selesai maka selanjutnya antara hasil metode keduanya *Faster R-CNN* dan YOLO akan dibandingkan parameter nya, adapun uji coba YOLO memakat epoch 60 dan batch 36. Contoh dari beberapa uji coba *https://wandb.ai/home* grafik dibawah ini



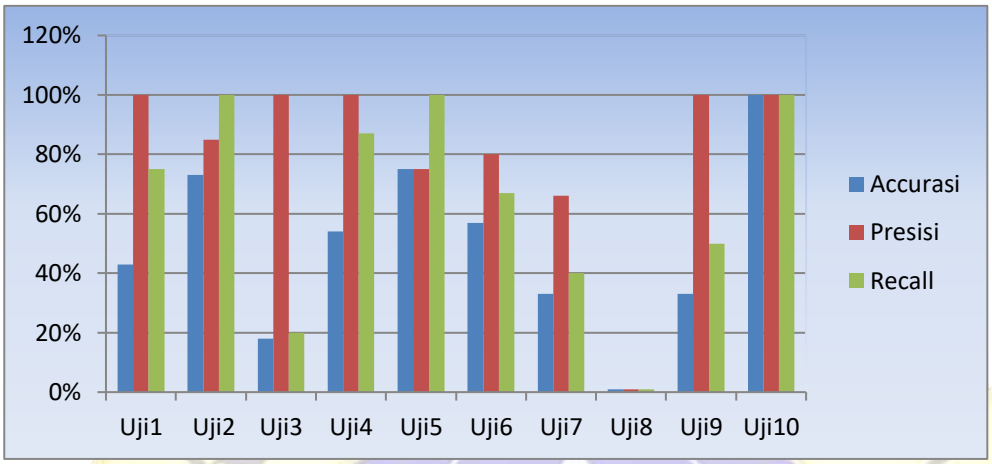
Gambar 4.11 Grafik hasil uji coba

Table 4.4. Hasil *parameter* uji coba metode YOLO

Nama Urut	Nama File	Total Objek	Nilai Prediksi				Akurasi	Presisi	*Recall*	Waktu Komputasi
			TP	TN	FP	FN				
Uji1		7	3	1	0	2	43%	100%	75%	0.50
Uji2		15	11	2	0	2	73%	85%	100%	0.56
Uji3		11	2	0	0	8	18%	100%	20%	0.30
Uji4		13	7	0	0	1	54%	100%	87%	1.05
Uji5		4	3	0	1	0	75%	75%	100%	0.35

Uji6		7	4	1	0	2	57%	80%	67%	0.72
Uji7		6	2	1	0	3	33%	66%	40%	0.26
Uji8		4	0	4	0	0	0%	0%	0%	0.46
Uji9		6	2	0	0	2	33%	100%	50%	0.37
Uji10		3	3	0	0	0	100%	100%	100%	0.50
Rata - Rata							60%	50%	28%	0,65

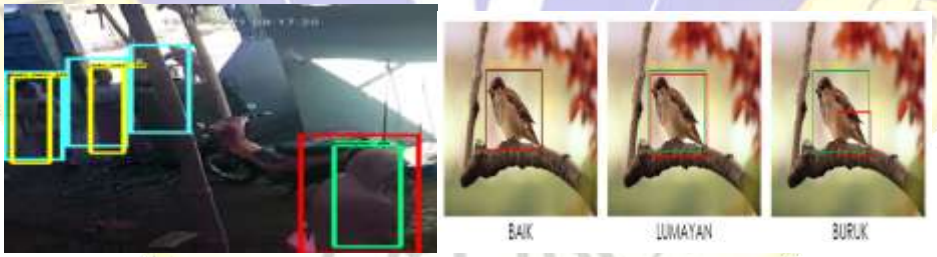
Pada Tabel 4.3 hasil uji coba dari metode YOLO, pada semua data beberapa video CCTV dan beberapa keadaan menunjukan bahwa rata-rata tingkat akurasi yang dicapai sebesar 60%, tingkat presesi sebesar 15% , dan rata-rata *recall* yang dicapai sebesar 71% dan rata-rata komputasi sebesar 0,28%. Dari data yang telah dijelaskan diatas maka peneleti menentukan tingkatan *accuracy* mengalami *sensitive* atau menurun jika dalam keadaan sepi, dan *dataset training* dan validasinya masih sedikit serta tidak seimbang dengan *dataset* Metode *Faster RCNN* namun metode YOLO akan menghasilkan *output* yang baik jika objek yang didapat sama dengan model *dataset* . Gambar 4.14 menunjukkan grafik hasil *accuracy* menggunakan metode YOLO



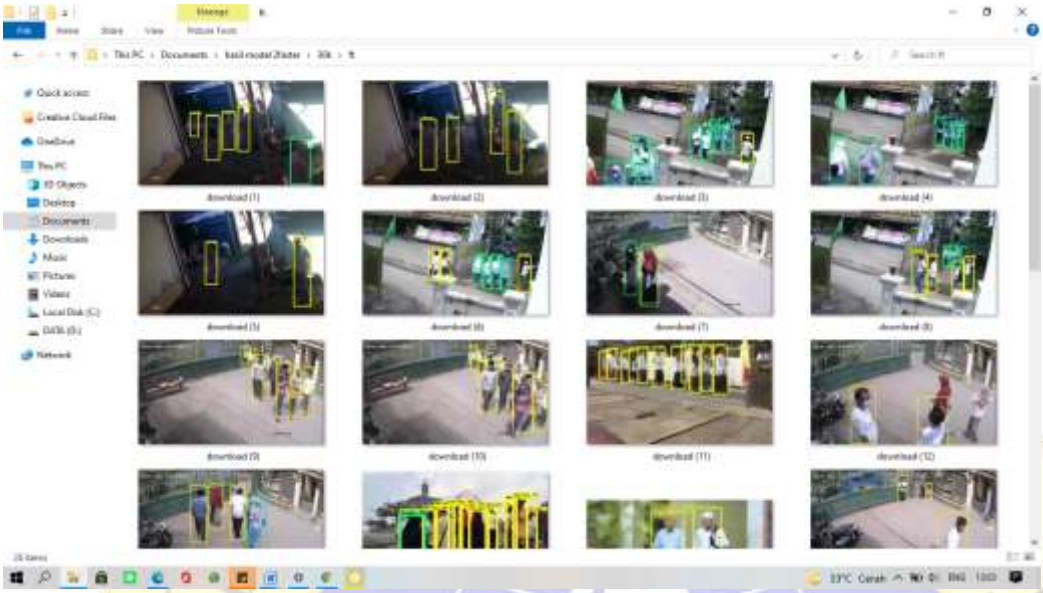
Gambar 4.12 *Accurasi* YOLOV5

4.6 Analisa Data

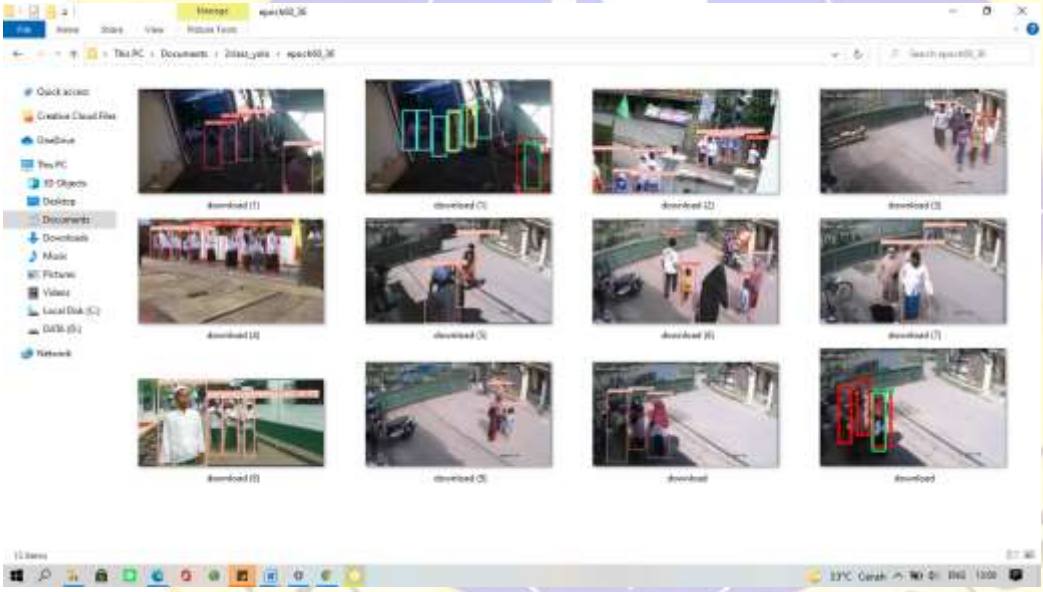
Pada tahapan ini, data hasil uji coba akan akan dianalisis untuk mengetahui tingkat keberhasilan, terdapat dua *focus* analisis yang akan dilakukan yaitu, analisis hasil uji coba metoda *Faster Region Neural Network* dan menbandingkan uji coba Metode *You Only Look Once* (YOLO). Model tersebut memprediksi *Bounding Box* dari objek yang terdeteksi diduga bahwa kotak prediksi tidak akan sama denga *Ground truth bounding box* sedangkan yang berwarna hijau dan kuning prediksi *bounding box* nya dari keduanya sedangkan *ground truth* nya berwarna merah biru dari itu peneliti bisa melihat model yang dibuat merupakan prediksi yang baik denga skor kecocokan yang tinggi.



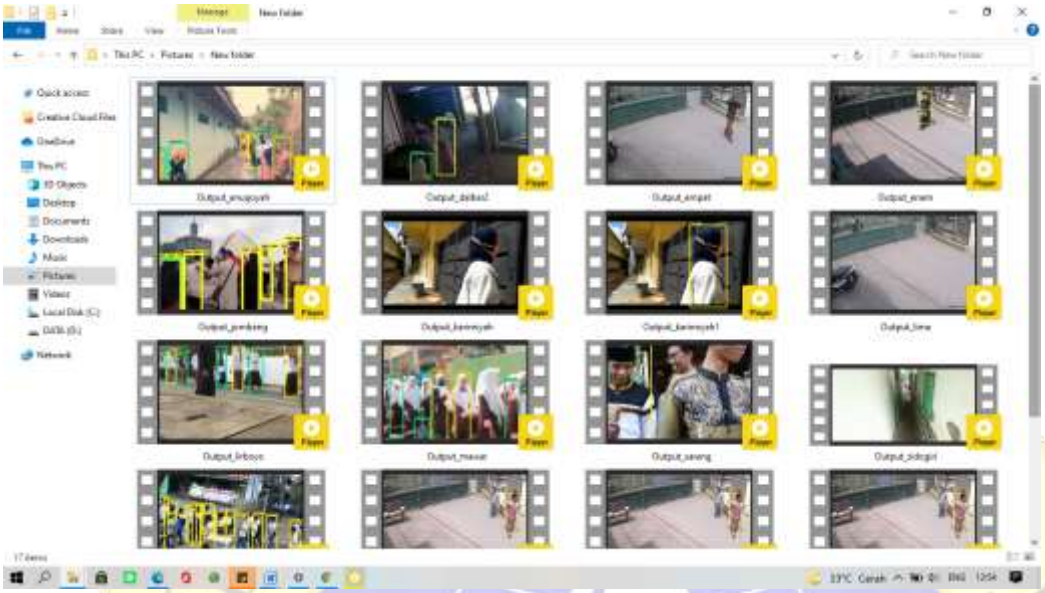
Gambar 4.13. Hasil Uji Coba *Faster R-CNN* dan YOLO



Gambar 4.13. Hasil uji coba metode *Faster Region Neural Network*



Gambar 4.14. Hasil Uji coba YOLO



Gambar 4.15. Uji coba video CCTV

Berdasarkan hasil analisis rata-rata tingkat akuras serta sensitivitas metode *Faster Region Neural Network* dan *You Only Look Once*(YOLO) ketika uji coba dalam metode *faster region neural netwok* dari hasil rata-rata 77% ketika dari bagian uji coba gambar pertama *accurasi* 80% karena *dataset* yang ada sama degan hasil *training* dan adapun hasil dari uji coba ke 6 hanya mendapatkan hasil *accurasi* 25% karna *dataset* perempuan kekurangan model gaya kerudung. Adapun hasil uji coba ke 9 mendapatkan hasil 100% meski dari hasil vodeo CCTV berbeda namun *dataset* laki-laki lebih banyak dari pada perempuan. Namun metode *faster* akan menghasilkan akurasi yang baik jika uji coba *step* nya lebih dan model yang digunakan lebih banyak.

Adapun metode YOLO nilai rata-rata 60% merupakan hasil *accuracy* lebih besar dari metode *faster*. Karena *dataset* untuk metode YOLO masih sekitar 500 dari pada metode *faster*, namun metode ini cukup baik dalam mendeteksi dan *training* model cukup cepat dari pada metode faster. Dari hasil kedua metode diatas bisa digunakan untuk uji coba pada video CCTV lain, namun hasil durasi dalam *testing* video cukup lambat dari metode faster RCNN dan untuk hasil *testing* Video CCTV Dari metode keduanya juga bisa diuji cobakan pada video *real time* dengan CCTV lain dan hasil rata-rata sekitar 80% untuk nilai kebenaranya