PERANCANGAN SISTEM DETEKSI PELANGGARAN PENGGUNAAN HELM DENGAN METODE *DEEP LEARNING* MENGGUNAKAN YOLOV5 ULTRALYTIC

Zwingli Hilkia Batubara[1*)], Yosia Hamonangan Nainggolan[2*)] M. Arfan[3], Achmad Hidayatno[4]

[123]Program Studi Sarjana Departemen Teknik Elektro, Universitas Diponegoro
Jl. Prof. Sudharto, SH, Kampus UNDIP Tembalang, Semarang 50275, Indonesia
*) *E-mail: zwingli.hilkia@gmail.com*

Abstrak

Penegakan aturan terkait penggunaan helm pada sepeda motor dilingkungan kampus menjadi upaya yang dilakukan untuk mengurangi dampak kecelakaan yang fatal, namun nyatanya masih banyak pengendara yang tidak sadar dan tidak taat akan hal ini. Pihak keamanan kampus yang terbatas dibanding dengan pelanggaran yang banyak terjadi telah menjadi keluhan dalam usaha menciptakan lingkungan kampus yaang aman. Meningkatnya teknologi *machine learning* yang semakin berkembang dan mudah diakses dapat diterapkan menjadi sistem deteksi pelanggaran dalam upaya membantu pihak keamanan dalam *memonitoring* keamanan berkendara. Sistem deteksi pelanggaran penggunaan helm dirancang menggunakan YOLOv5 Ultralytic dengan memanfaatkan metode *deep learning* yang didasari dengan algoritma *convolutional neural network*. YOLOv5 memberikan model dengan penekanan pada kecepatan inferensi yang baik, sehingga memungkinkan penggunaan dalam aplikasi *real-time* atau *near-real-time*. Model akhir hasil *training* menggunakan yolov5 memiliki *mean average precision* (mAP) rata-rata sebesar 0,938 yang mengindikasikan model dapat melakukan deteksi objek dengan akurat dan konsisten. Sistem deteksi ini akan menangkap dan mengirimkan gambar pelanggaran ke database. Hasil pengujian akhir sistem deteksi dengan melakukan evaluasi *confusion matrix* yang didapat menunjukkan akurasi sistem sebesar 98,5%. Gambar tersebut nantinya akan dimonitoring pada aplikasi android.

Kata kunci : *Deteksi Helm*, YOLOv5 Ultralytic, *Deep Learning*, *mAP* (*mean Average Precision*), *convolutional neural network, Confusion Matrix*

Abstract

Enforcement of helmet use regulations for motorcycles on campus is an effort to reduce the impact of fatal accidents. However, there are still many riders who are not aware and not comply to these regulations. The limited number of campus security officers compared to the many violations that occur has become a complaint in the effort to create a safe campus environment. The development of machine learning technology that is increasingly accessible can be applied to a violation detection system to help security officers monitor driving safety. The helmet violation detection system is designed using YOLOv5 Ultralytic by utilizing the deep learning method based on the convolutional neural network algorithm. YOLOv5 provides a model with an emphasis on good inference speed, making it possible to use in real-time or near-real-time applications. The final model of the training results using yolov5 has an average mAP of 0.938, which indicates that the model can perform object detection accurately and consistently. This detection system will capture and send violation images to the database. The final test results of the detection system by evaluating the confusion matrix obtained showed a system accuracy of 98.5%. These images will be monitored in an Android application.

***Keywords** : Helmet Detection, YOLOv5 Ultralytic, Deep Learning, mAP (Average Precision), Convolutional Neural Network, Confusion Matrix*

1. Pendahuluan

Menurut data langsung dari web Korlantas Porli, pada tahun 2023 pada bulan november ini terdapat 158,90 juta kendaraan bermotor yang terdaftar di Indonesia, dan 83,50% diantaranya adalah sepeda motor [1]. Jumlah kendaraan bermotor yang tinggi di Indonesia juga memberikan dampak serius, yaitu tingkat kecelakaan lalu lintas yang semakin tinggi. Menurut data dari Badan Pusat Statistik, pada tahun 2021 terjadi sebanyak 103.645 kasus kecelakaan lalu lintas yang melibatkan kendaraan bermotor, dengan korban jiwa sebanyak 25.266 orang [2]. Jumlah kecelakaan tersebut didominasi oleh kendaraan bermotor sebanyak 73% [2].

Kondisi pengendara di lingkungan kampus Universitas Diponegoro (Undip) di Semarang masih menjadi perhatian penting, terutama terkait penggunaan helm. Meskipun Undip adalah salah satu pusat pendidikan terkemuka di Indonesia, faktanya masih banyak pengendara, baik mahasiswa maupun staf, yang tidak menggunakan helm saat berkendara di dalam kampus [3]. Perilaku tidak taat ini dapat mengancam keselamatan mereka sendiri dan orang lain di sekitar mereka. Undip dapat menciptakan lingkungan yang lebih aman dan mengutamakan keselamatan berkendara bagi mahasiswa dan stafnya dengan meningkatkan kesadaran akan pentingnya penggunaan helm dan memperkuat penegakan aturan terkait helm di lingkungan kampus.

Berdasarkan beberapa penelitian yang sudah ada sebelumnya, terdapat beberapa metode atau algoritma yang dapat digunakan untuk melakukan deteksi penggunaan helm pada pengguna sepeda motor seperti *Deep Learning*, *Haar Cascades*, Metode *Thresholding*, *Canny*, ataupun *Recurrent Neural Networks (RNN)*. Terdapat beberapa jenis atau arsitektur *Deep Learning* yang dapat digunakan untuk mendeteksi penggunaan helm pada pengendara sepeda motor. Arsitektur pertama adalah dengan menggunakan metode *Convolutional Neural Networks* (CNN). Arsitektur CNN sangat berguna untuk pengenalan gambar dan klasifikasi gambar, serta tugas-tugas visi komputer lainnya karena mereka dapat memproses data dalam jumlah besar dan menghasilkan prediksi yang sangat akurat. CNN dapat mempelajari fitur-fitur suatu objek melalui beberapa kali iterasi, sehingga tidak perlu lagi melakukan tugas-tugas rekayasa fitur secara manual seperti ekstraksi fitur [4]. Arsitektur kedua adalah dengan menggunakan *Region-Based Convolutional Neural Networks* (R-CNN). R-CNN menggabungkan proposal wilayah persegi panjang dengan fitur jaringan saraf konvolusi. Gabungan tersebut adalah algoritma deteksi dua tahap, tahap pertama mendefinisikan subset wilayah [5]. Sedangkan pada tahap kedua, setiap daerah akan diklasifikasikan sebagai artefak. Sehingga, R-CNN memiliki tiga varian untuk mengoptimalkan, mempercepat, atau meningkatkan hasil dari mendeteksi objek [5]. Selain itu, terdapat juga beberapa arsitektur deep learning yang dapat digunakan dalam mendeteksi suatu objek seperti RetinaNet, SSD, dan YOLO.

YOLO adalah pendekatan baru untuk mendeteksi beberapa objek yang ada dalam sebuah gambar secara *real-time* dengan menggambar kotak pembatas di sekelilingnya [6]. YOLO meneruskan gambar melalui algoritma CNN hanya sekali untuk mendapatkan output. Arsitektur YOLO mirip dengan jaringan saraf konvolusi yang terinspirasi oleh model GoogLeNet untuk klasifikasi gambar. Lapisan awal jaringan pertama-tama mengekstrak fitur gambar, dan lapisan yang terhubung sepenuhnya memprediksi probabilitas dan koordinat keluaran. Model jaringan YOLO yang lengkap dibuat terdiri dari 24 lapisan konvolusi, dua lapisan yang terhubung penuh, lapisan reduksi 1x1, dan lapisan konvolusi 3x3 [7].

Sistem deteksi nantinya akan dirancang dengan menggunakan YOLOv5. YOLOv5 dibangun di PyTorch dan dikembangkan oleh Ultralytics dan diklaim memiliki beberapa peningkatan signifikan dibandingkan detektor YOLO yang ada. YOLOv5 tidak perlu mempertimbangkan kumpulan data apa pun untuk digunakan sebagai masukan, dan memiliki kemampuan untuk secara otomatis mempelajari kotak jangkar yang paling tepat untuk kumpulan data tertentu yang sedang dipertimbangkan, dan menggunakannya selama pelatihan [8].

2. Metode Penelitian

2.1 *Artificial Intelligence*

Artificial Intelligience (AI) atau kecerdasan buatan mengacu pada ilmu pengetahuan dan teknik di balik pembuatan perangkat cerdas, terutama program komputer cerdas dimana semua pekerjaan yang biasa dilakukan oleh manusia dapat digantikan oleh komputer [7]. Kualitas kognisi, ucapan, dan penglihatan yang dimiliki manusia dijadikan standar untuk membuat kecerdasan buatan [9].

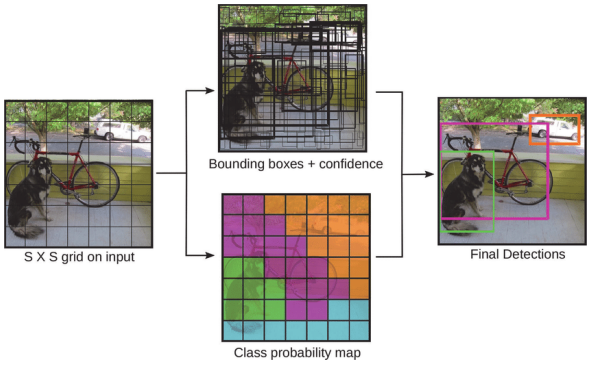
2.2 *Machine Learning*

Machine Learning adalah pemrograman komputer untuk mengoptimalkan kriteria-kinerja menggunakan data contoh atau pengalaman masa lalu [10] . Machine learning terbagi menjadi 3 kategori dasar, yaitu *supervised learning*, *unsupervised learning*, dan *reinforcement learning*. Sedangkan, *deep learning* adalah sub-bidang dari *machine learning* yang berfokus menggunakan *Artificial Neural Networks* (ANNs) untuk belajar dari data [11].

2.3 YOLO

You Only Look Once (YOLO) adalah pendekatan baru untuk mendeteksi beberapa objek yang ada dalam sebuah gambar secara *real-time* dengan menggambar kotak

pembatas di sekelilingnya [6]. YOLO meneruskan gambar melalui algoritma CNN hanya sekali untuk mendapatkan output. Model deteksi dari YOLO dapat dilihat pada Gambar 1 berikut.

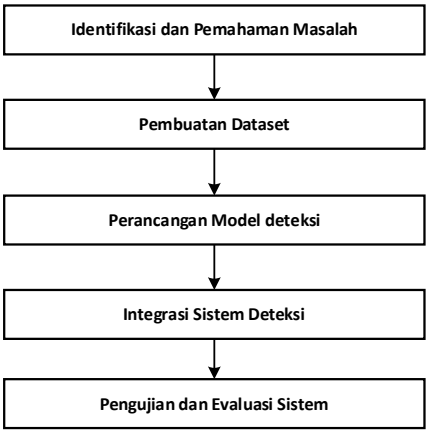


Gambar 1 Model Deteksi YOLO [12]

Gambar 1 memperlihatkan arsitektur YOLO mirip dengan jaringan saraf konvolusi yang terinspirasi oleh model GoogLeNet untuk klasifikasi gambar. Lapisan awal jaringan pertama-tama mengekstrak fitur gambar, dan lapisan yang terhubung sepenuhnya memprediksi probabilitas dan koordinat keluaran. Model jaringan YOLO yang lengkap dibuat terdiri dari 24 lapisan konvolusi, dua lapisan yang terhubung penuh, lapisan reduksi 1x1, dan lapisan konvolusi 3x3 [13].

2.4 Tahapan Perancangan Sistem Deteksi Penggunaan Helm

Perancangan model deteksi pada Subsistem ini akan diterapkan bersamaan dengan subsistem lainnya yaitu Aplikasi Android "Helmonzy" yang dimana nantinya akan digabungkan menjadi satu sistem yaitu Sistem Deteksi dan Pelaporan Tindak Pelanggaran Penggunaan Helm pada Pengendara Sepeda Motor di Lingkungan Kampus. Flowchart perancangan sistem dapat dilihat pada Gambar 2.



Gambar 2 *Flowchart* Perancangan Sistem Makalah

2.5 Identifikasi dan Pemahaman Masalah

Langkah awal dalam perancangan sistem deteksi ini adalah mengidentifikasi dan memahami masalah yang ada. Makalah ini akan berfokus dalam masalah pelanggaran penggunaan helm, sehingga dibutuhkan sistem deteksi yang dapat melakukan perbedaan antara pengguna sepeda motor yang menggunakan helm dan yang tidak menggunakan helm.

Identifikasi dapat mencakup identifikasi pengguna, penentuan lokasi implementasi alat pada lokasi yang sesuai, pengumpulan data, pemilihan model yang sesuai, penyesuaian spesifikasi sistem dengan infrastruktur yang tersedia, pemahaman terhadap cara kerja sistem, serta kebutuhan-kebutuhan yang diperlukan untuk memecahkan permasalahn yang ada dalam Makalah. Kebutuhan yang diperlukan seperti kebutuhan fungsional, kebutuhan perangkat keras, dan kebutuhan perangkat lunak.

2.5.1 Kebutuhan Fungsional

Deskripsi kebutuhan fungsional adalah gambaran mengenai fungsi-fungsi yang dapat dijalankan oleh model ini. Kebutuhan fungsional sistem deteksi mencakup:

1. Sistem dapat mendeteksi objek berupa pengguna sepeda motor yang menggunakan helm dan yang tidak menggunakan helm.
2. Sistem dapat menangkap gambar setelah dilakukan deteksi dan mengirimkannya ke database secara *real-time*.
3. Sistem dapat bekerja secara optimal pada jangka waktu panjang secara *real-time*.

2.5.2 Kebutuhan Perangkat Keras

Pembuatan model memerlukan perangkat keras yang mampu memproses data dengan kemampuan komputasi yang baik. Banyak data dan kesederhanaan model akan mempengaruhi cepatnya proses pelatihan. Kemampuan perangkat keras juga akan memengaruhi kecepatan pelatihan model. Perangkat keras yang digunakan untuk membuat model sistem deteksi objek dan menjalankan model tercantum pada Tabel 1.

Tabel 1 Kebutuhan perangkat keras untuk membuat model.

Komponen	Keterangan
CPU	Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
GPU	Nvidia® GTX™ 1650 8GB
GPU (Google Colab)	Tesla K80 GPU with 12GB Ram
RAM (2 single)	Kingston® Micron Technology 8GB DDR4-2666(1333MHz)
	Kingston® Nanya Technology 8GB DDR4-3200(1600MHz)
Storage	SSD : Kingston ® A400™ 480GB
	SSD : NVMe® SN530™ 512GB
Motherboard	CFL® Octavia_CFL

Perangkat keras berupa kamera yang digunakan untuk dapat menangkap gambar dan melakukan deteksi secara *real-time* pada alat dapat dilihat pada Tabel 2 sebagai berikut:

Tabel 2 Kebutuhan perangkat keras berupa Webcam

Spesifikasi	Keterangan
Video Resolution	2560 x 1440 pixel
Max Resolution	1440p/30fps - 720p/30fps
Image Sensor	COMS
Field of View	90
Plug Type	USB 2.0

2.5.3 Kebutuhan Perangkat Lunak

Pembuatan model memerlukan perangkat lunak juga seperti sistem operasi, *web browser*, *Dataset Tools*, *Integrated development environment* (IDE), bahasa pemrograman, *framework*, *object detection model*, dan *Library* untuk dapat membuat model yang baik. Perangkat lunak yang digunakan dalam Makalah ini dapat dilihat pada Tabel 3 berikut ini :

Tabel 3 Kebutuhan perangkat lunak

Perangkat Lunak	Keterangan
Sistem Operasi	Windows 10
Web Browser	Google Chrome
Dataset Tools	Roboflow
Integrated development environment (IDE)	Google Colab, PyCharm
Bahasa Pemrograman	Python
Framework	PyTorch
object detection model	YOLOv5
Library	Torch, Ultralytics, CUDA, OpenCV, Pip, Pandas, Numpy, Matplotlib, PyYAML, Pillow, Scipy, Torchvision, Roboflow

2.6 Pembuatan *Dataset*

Langkah pembuatan *dataset* dalam perancangan sistem deteksi pada Makalah ini memegang bagian terpenting dalam membangun model deteksi yang baik. Kualitas dataset akan memiliki dampak yang signifikan terhadap kinerja model dikarenakan *dataset* menyediakan model dengan contoh-contoh objek yang harus dideteksi. Tanpa *dataset*, model tidak akan memiliki cara untuk mempelajari seperti apa objek yang harus dideteksi.

Beberapa hal yang dapat dipertimbangkan dalam membuat dataset seperti keragaman dataset dan ukuran dataset. Keragaman dataset membantu model untuk menggeneralisasi data baru. Model yang dilatih dengan set data yang beragam akan dapat mendeteksi objek dalam berbagai skenario yang lebih luas daripada model yang dilatih dengan *set* data yang kurang beragam, Sedangkan ukuran *dataset* membantu mencegah *overfitting*. *Overfitting* terjadi ketika sebuah model belajar untuk menyesuaikan diri dengan data pelatihan dengan terlalu baik dan tidak dapat menggeneralisasi dengan baik untuk data baru. *Dataset* yang besar dapat membantu mencegah *overfitting* dengan memberikan lebih banyak contoh untuk dipelajari oleh model.

Penulis membuat *Dataset* pada Makalah dengan menggunakan *web tools* Roboflow, dimana proses pembuatan *dataset* dapat dengan efektif dilaksanakan dan dataset yang telah jadi dapat dipanggil langsung dengan *api* yang disediakan oleh Roboflow saat proses pembuatan model. Langkah-langkah dalam pembuatan *dataset* pada roboflow. yaitu:

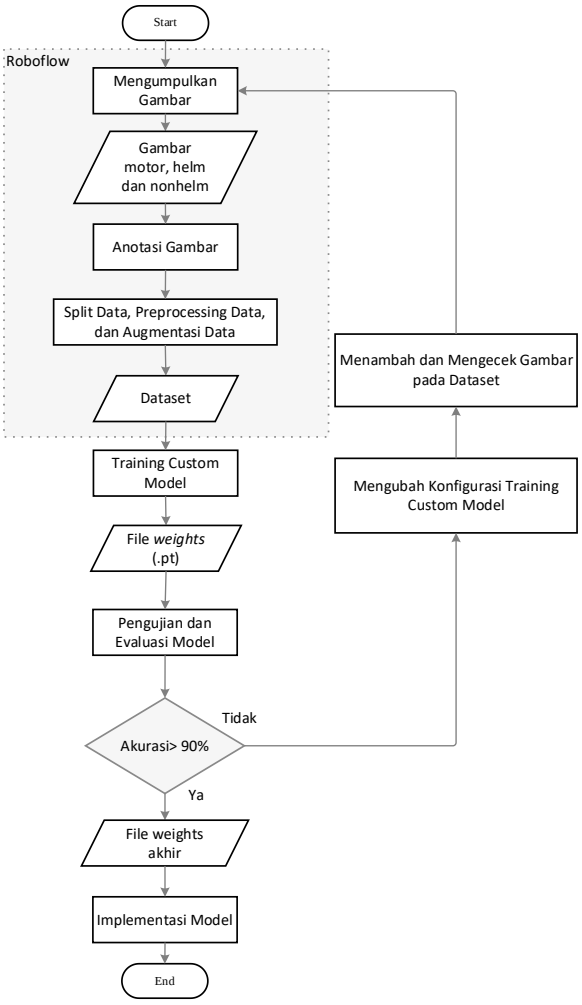
1. Pengumpulan gambar: Data berupa gambar yang nantinya akan diproses lebih lanjut pada *web* roboflow. Total gambar yang dikumpulkan sebanyak 1588 gambar.
2. Data *annotation*: Proses menambahkan label pada taip data. Makalah ini menetapkan 3 label pada dataset yaitu "helm", "motor", dan "nonhelm".
3. *Train/Test Split*: proses pembagian gambar yang telah dianotasi ke dalam set pelatihan, pengujian, dan *test* yang terpisah. Data dibagikan menjadi 1056 gambar *train* set, 302 gambar *valid* set dan 149 gambar sebagai *test* set.
4. *Preprocessing* dan *Augmentation* : Merupakan proses untuk mengubah ukuran seluruh gambar pada dataset dan melakukan augmentasi gambar. *preprocessing* yang dilakukan pada dataset Makalah yaitu *Auto-orient* dan *Resize*. Sedangkan augmentasi yang dilakukan dengan variasi *flip horizantal* dan *Saturation*.
5. *Generate* dan *Export Dataset* : Proses akhir dalam membuat *dataset* pada roboflow dengan melakukan *generate* pada dataset. Dataset yang telah di-*generate* akan di-*export* dalam bentuk format YOLOv5-Torch

2.7 Perancangan Model Deteksi

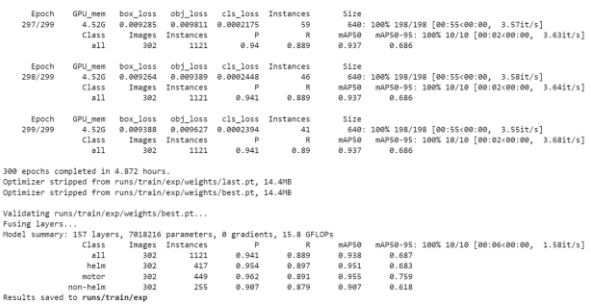
Dataset yang sudah di-*export* dari roboflow akan digunakan pada proses perancangan model deteksi. Beberapa tahapan pada diagram alir berikut digunakan untuk membantu memahami proses perancangan model deteksi.

Perancangan diawali dengan pembuatan dataset pada roboflow seperti pada proses sebelumnya. Dataset yang sudah di-*export* akan di-*training* dengan melakukan *clone* pre-model YOLOv5 dari github *official* [14] dan dilakukan penginstalan *package* yang diperlukan. Model di-*training* dengan memkai google colab dan menggunakan GPU Tesla T4 untuk melakukan *training*. Dataset roboflow sebelumnya akan di-*import* dan dilakukan *training* model dengan memanggil `train.py` dan mengisi parameter yang diperlukan.

Parameter pada train.py terdiri atas `img` untuk menentukan *size image*, *size image* yang digunakan adalah 640. Parameter `batch` digunakan untuk menentukan *batch size*, *batch size* yang digunakan secara normal adalah 16. `Epoch` untuk menentukan banyaknya pelatihan yang dilakukan algoritma model, *epoch* yang digunakan adalah 300. `Data` merupakan parameter untuk lokasi dataset. `Weights` merupakan parameter untuk lokasi dari model yang ingin digunakan untuk memulai *transfer learning*. Model weights yang dipakai adalah model yolov5s. Parameter terakhir yaitu `cache` merupakan parameter untuk melakukan *cache* pada gambar untuk mempercepat proses *training*. Berikut merupakan *snipset* akhir dari proses *training* setelah `train.py` dijalankan. *Flowchart* terkait hal ini dan proses akhir *Training* dapat dilihat pada Gambar 3 dan Gambar 4 berikut ini.



Gambar 3 Diagram Alir Perancangan Model Deteksi



```
     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
297/299      4.52G  0.009285  0.009811 0.0002175        59       640: 100% 198/198 [00:55<00:00,  3.57it/s]
               Class      Images  Instances          P          R      mAP50   mAP50-95: 100% 10/10 [00:02<00:00,  3.63it/s]
                 all         302        1121       0.94      0.889      0.937      0.686

     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
298/299      4.52G  0.009264  0.009389 0.0002448        46       640: 100% 198/198 [00:55<00:00,  3.58it/s]
               Class      Images  Instances          P          R      mAP50   mAP50-95: 100% 10/10 [00:02<00:00,  3.64it/s]
                 all         302        1121       0.941      0.889      0.937      0.686

     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
299/299      4.52G  0.009388  0.009627 0.0002394        41       640: 100% 198/198 [00:55<00:00,  3.55it/s]
               Class      Images  Instances          P          R      mAP50   mAP50-95: 100% 10/10 [00:02<00:00,  3.68it/s]
                 all         302        1121       0.941       0.89      0.937      0.686

300 epochs completed in 4.872 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.4MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7018216 parameters, 0 gradients, 15.8 GFLOPs
               Class      Images  Instances          P          R      mAP50   mAP50-95: 100% 10/10 [00:06<00:00,  1.58it/s]
                 all         302        1121       0.941      0.889      0.938      0.687
                helm         302         417       0.954      0.897      0.951      0.683
               motor         302         449       0.962      0.891      0.955      0.759
            non-helm         302         255       0.907      0.879      0.907      0.618
Results saved to runs/train/exp
```

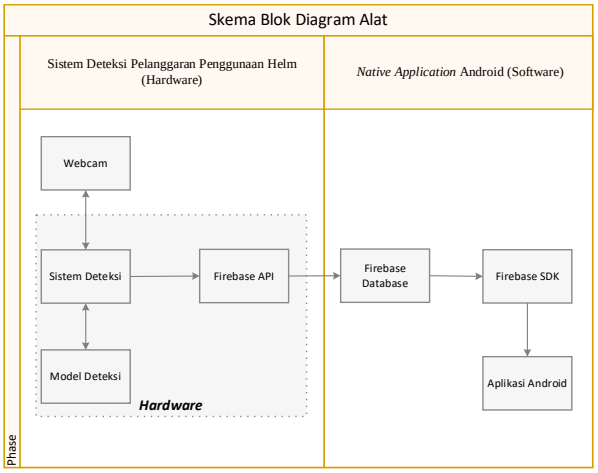
Gambar 4 Tampilan Proses Akhir Training Custom Model

Hasil *training* dari Gambar 4 tersebut sebanyak 300 epoch dengan total waktu training selama 4,872 jam. Model yang

dihasilkan dari training ini sebesar 14,4 mb dalam format weights (.pt). Hasil model seperti akurasi, grafik dan evaluasi lainnya akan lebih lanjut dijelaskan pada bab berikutnya.

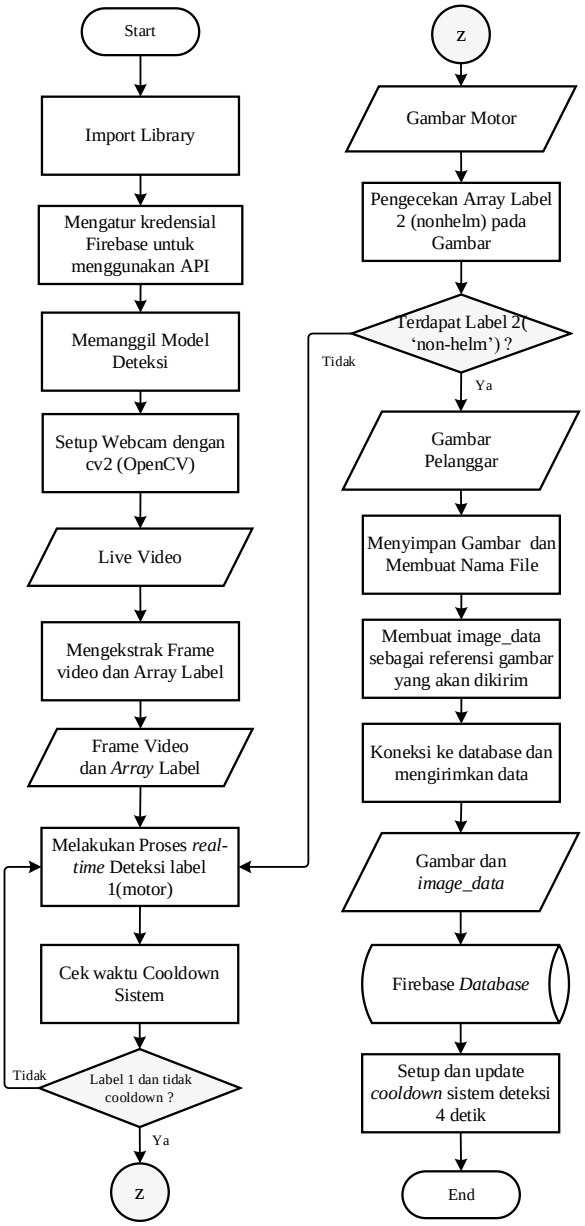
2.8 Perancangan Model Deteksi

Model hasil *custom training* YOLOv5 akan diintegrasikan ke sistem deteksi secara keseluruhan untuk dapat melakukan *real-time detection*, menangkap gambar secara otomatis dan mengirimkan gambar tersebut ke Firebase *database*. Sistem deteksi yang sudah jadi nantinya akan disambungkan ke database untuk dapat dilakukan uji coba alat secara keseluruhan. Skema blok diagram alat pada Gambar 5 berikut akan memudahkan melihat tahapan dan cara kerja alat.



Gambar 5 Skema Diagram Blok Alat Deteksi Pelanggaran Penggunaan Helm

Gambar 5 menunjukkan integrasi sistem, dimana sistem deteksi, model deteksi dan firebase api (API untuk menggubungkan sistem ke *database*) menjadi 1 sistem pada *hardware*. Sistem deteksi pelanggaran penggunaan helm akan mengirimkan hasil akhir berupa gambar dan referensi gambar ke database untuk nantinya diolah oleh aplikasi android sebagai aplikasi *monitoring*. Berikut merupakan diagram alir dari sistem deteksi secara keseluruhan.



Gambar 6 Diagram Alir Sistem Deteksi Pelanggaran Penggunaan Helm

Diagram alir pada Gambar 6 menunjukkan alur sistem deteksi pelanggaran helm dari awal *import* library hingga mengirimkan gambar pelanggar ke database.

Sistem berjalan yang diawali dengan melakukan *import library* yang sudah diinstal sebelumnya. Beberapa library seperti `torch`, `time`, `cv2`, `PIL`, `numpy`, dan `pathlib` akan digunakan langsung pada *main code*. Program selanjutnya akan mengatur kredensial agar program dapat mengakses API dan *database* firebase. Data kredensial merupakan file *json* yang nantinya diakses program ketika ingin mengirimkan data ke database seperti yang terlihat pada Gambar 6.

Model deteksi selanjutnya akan dipanggil dengan menggunakan `torch.hub.load` dan memasukkan parameter berupa repo dari YOLOv5 dan *path* dari model deteksi yang sudah dilatih sebelumnya yaitu `helmonzy_fixed.pt` . Model yang sudah dipanggil akan melanjutkan program untuk memanggil `cv2` yang akan melakukan *setup* dari *webcam* sebagai *real-time input*. *Frame* dan *array_label* akan terlebih dahulu dipanggil sebelum webcam dapat bekerja sebagai *input* agar *window* dari *webcam* dapat terlihat

Fungsi *Array* digunakan untuk memuat label dari *object* yang dideteksi akan ditampilkan dan terbaca nantinya.

Array label akan memuat 3 label *object* yaitu 0 sebagai helm, 1 sebagai 'motor' dan 2 sebagai 'non-helm'. Sistem selanjutnya akan melakukan *real-time* deteksi dan model deteksi akan selalu mengecek label 1(motor) dari *array* tersebut. Frame yang terdeteksi label 1 akan dicek waktu cooldown sistem terlebih dahulu sebelum frame tersebut ditangkap dan dikirimkan ke fungsi deteksi berikutnya.

Proses yang terjadi pada real time deteksi dilakukan penambahan `cm2_frame < x_center < cm1_frame` sebagai area deteksi pada tengah video. Penambahan fungsi area deteksi ini akan dijelaskan pada bab berikutnya. Frame yang dideteksi akan diteruskan ke fungsi `process_image` dalam bentuk *screenshot* gambar yang telah berhenti.

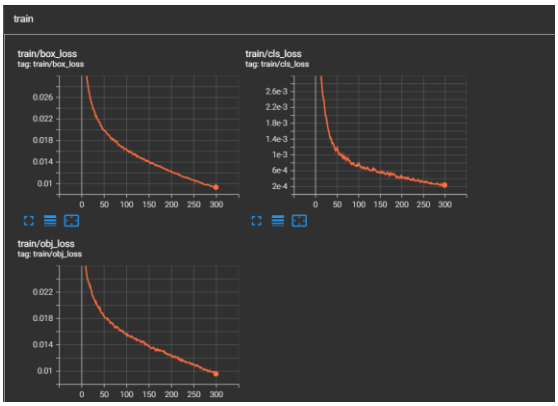
Gambar yang diteruskan ke fungsi `process_image` akan disimpan sementara didalam format .jpg untuk nantinya dilakukan deteksi pada label 2(nonhelm). Sistem yang mengecek adanya label 2 akan langsung menyimpan gambar tersebut ke lokal dan membuat nama gambar sesuai dengan *timestamp frame* waktu ditangkap.

Gambar yang telah disimpan dilokal akan dikirimkan ke firebase storage dan sistem juga membuat *image_data* yang merupakan referensi data agar *realtime* database nantinya dapat digunakan untuk membaca data waktu dan url dari gambar yang telah dikirimkan sistem ini. Sistem akan *mengupdate* waktu *cooldown* setelah gambar dan referensi data berhasil dikirimkan ke firebase database.

3 Pengujian dan Evaluasi

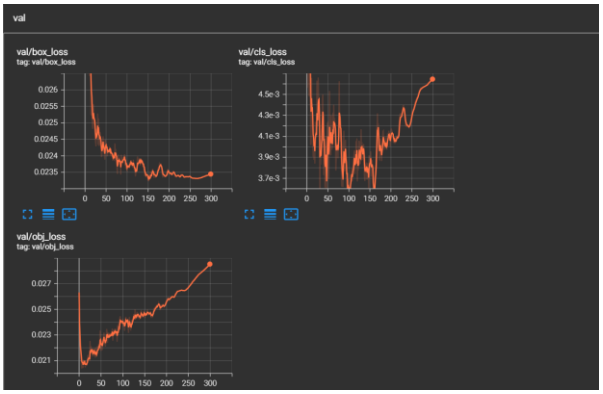
3.1 Pengujian dan Evaluasi Kerugian (*loss*) dan Peformance hasil Training Model Deteksi dengan Data Validasi

Evaluasi kerugian (*loss*) dan *peformance* dari hasil *training* model dilakukan dengan menjalankan *script build-in* yang tersedia pada YoLoV5 Ultralytic dan Tensor Board. Evaluasi *training loss* dapat dilihat pada Gambar 7.



Gambar 7 Tampilan Tensor Board pada Evaluasi *training loss*

Evaluasi dari *training loss* pada Gambar 7 menunjukkan *loss* yang terjadi pada *training* model dengan dataset sebanyak 3160 data *training*. Grafik menunjukkan *loss* hasil *training* pada epoch 300 (Epoch terakhir) sangat rendah yaitu 0,009388 *box loss*, 0,000239 *class loss*, dan 0,00962 *object loss*. Tampilan untuk evaluasi *validation loss* dapat dilihat pada Gambar 8.

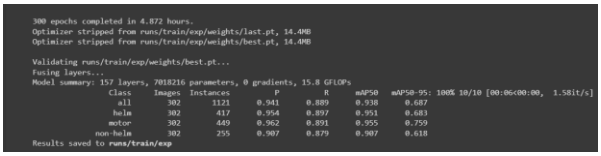


Gambar 8 Tampilan Tensor Board pada Evaluasi *validation loss*

Evaluasi dari *validation loss* pada Gambar 8 menunjukkan *loss* yang terjadi pada *validation* dengan dataset sebanyak 302 data *validation*. Grafik menunjukkan *loss* hasil *validation* pada epoch 300 (*Epoch* terakhir) sangat rendah yaitu 0,0234 *box loss*, 0,000465 *class loss*, dan 0,0285 *object loss*.

Berdasarkan grafik *loss training* dan *validation* tersebut, dapat diambil kesimpulan bahwa *training* model mempelajari pola-pola dalam set pelatihan dengan baik serta memiliki kinerja model yang stabil dan baik pada data baru. Hal ini terlihat dari perbedaan *loss* pada *training* dan *validation* yang sangat rendah (dibawah 0,1).

Pengujian dan evaluasi berupa *Precision*, *Recall*, mAP, dan F1. Berikut merupakan hasil pengujian serta hasil dari evaluasi *peformance training* dapat dilihat pada Gambar 9.



Gambar 9 Pengujian dan Evaluasi Model dengan Data Validasi

Gambar 9 menunjukkan pengujian dari hasil *training custom model* terbaik yang diujikan dengan data *validation* sebanyak 302 gambar. Hasil menunjukkan nilai mAP rata-rata sebesar 0,938 dimana untuk label 'helm', 'motor', dan 'non-helm' secara berurut sebesar 0,951 ; 0,955 ; 0,907. *Mean Average Precision* (mAP) untuk non-helm memiliki perbedaan yang signifikan dibandingkan dengan label lain, ini disebabkan oleh label 'non-helm' memiliki objek yang sangat bervariasi. Kepala pria, wanita, anak-anak, orang memakai topi, juga ukuran rambut menjadi alasan utama mengapa akurasi non-helm tidak sebaik yang lainnya.

Pertimbangan dari *range* data variasi yang banyak tersebut, menunjukkan akurasi diatas 90% untuk 'non-helm' sudah sangat baik dan dapat melakukan deteksi dengan lancar.

3.2 Pengujian Model Deteksi dengan Variasi Jarak dan Kecepatan

Pengujian model deteksi dilakukan berdasarkan variasi jarak 1,2, dan 3 meter dengan kecepatan 10 km/jam, 20 km/jam, 30 km/jam dan 40km/jam. Pengujian ini akan melihat dan menilai akurasi serta *confidence* model berdasarkan variasi jarak dan kecepatan tersebut. Hasil pengujian terdapat pada Tabel 4, Tabel 5, Tabel 6, Table 7, Tabel 8 dan Tabel 9 berikut ini.

Tabel 4 Data *Confidence* Hasil Pengujian Jarak 1 Meter dengan Variasi Kecepatan

Objek	*Confiedence*, Kecepatan			
	Confidence (%)			
	10 km/jam	20 km/jam	30 km/jam	40 km/jam
Motor	94,81%	93,20%	91,65%	91,01%
Helm	94,68%	94,40%	93,04%	92,11%
Non-helm	94,81%	93,20%	91,65%	91,01%

Tabel 5 Data *Accuracy* Hasil Pengujian Jarak 1 Meter dengan Variasi Kecepatan

Objek	*Accuracy*, Kecepatan			
	Accuracy (%)			
	10 km/jam	20 km/jam	30 km/jam	40 km/jam
Motor	100%	100%	90%	80%
Helm	100%	100%	90%	90%
Non-helm	100%	90%	90%	90%

Tabel 6 Data *Confidence* Hasil Pengujian Jarak 2 Meter dengan Variasi Kecepatan

Objek	*Confiedence*, Kecepatan			
	Confiedence (%)			
	10 km/jam	20 km/jam	30 km/jam	40 km/jam
Motor	96,61%	96,01%	95,13%	94,64%
Helm	95,38%	94,93%	93,66%	92,65%
Non-helm	94,27%	93,71%	93,09%	92,46%

Tabel 7 Data *Accuracy* Hasil Pengujian Jarak 2 Meter dengan Variasi Kecepatan

Objek	*Accuracy*, Kecepatan			
	Accuracy (%)			
	10 km/jam	20 km/jam	30 km/jam	40 km/jam
Motor	100%	100%	100%	100%
Helm	100%	100%	100%	90%
Non-helm	100%	100%	100%	90%

Tabel 8 Data *Confidence* Hasil Pengujian Jarak 3 Meter dengan Variasi Kecepatan

Objek	*Confiedence*, Kecepatan			
	Confiedence (%)			
	10 km/jam	20 km/jam	30 km/jam	40 km/jam
Motor	95,97%	95,32%	94,96%	94,45%
Helm	95,04%	94,14%	93,38%	92,47%
Non-helm	94,09%	93,62%	92,84%	92,06%

Tabel 9 Data *Accuracy* Hasil Pengujian Jarak 3 Meter dengan Variasi Kecepatan

Objek	*Accuracy*, Kecepatan			
	Accuracy (%)			
	10 km/jam	20 km/jam	30 km/jam	40 km/jam
Motor	100%	100%	100%	100%
Helm	100%	100%	100%	90%
Non-helm	100 %	100%	100%	90%

Berdasarkan hasil pengujian pada Tabel 4, Tabel 5, Tabel 6, Table 7, Tabel 8 dan Tabel 9, model memiliki rata-rata *confidence* pada pengujian jarak 1 meter, 2 meter dan 3 meter dengan kecepatan diantara 0 hingga 40km/jam secara berurutan adalah 92,67%, 94,38%, dan 94,03%. Rata-rata akurasi pada pengujian jarak 1 meter, 2 meter dan 3 meter dengan kecepatan diantara 0 hingga 40km/jam secara berurutan adalah 93,33%, 98,33%, dan 98,33%. Hasil ini menunjukkan kecepatan mempengaruhi *confidence* dan akurasi pada masing-masing jarak namun tidak terlalu jauh perbedaannya. Jarak kerja optimal model adalah jarak 2 hingga 3 meter dengan kecepatan 10 hingga 30 km /jam.

3.3 Pengujian Integrasi Model Deteksi Secara *Real-Time*

Pengujian ini dilakukan dengan model yang telah diintegrasikan ke program keseluruhan untuk menangkap dan mengirimkan data berupa gambar pelanggaran penggunaan helm ke database *firebase*. Pengujian dilakukan secara *real-time* pada lingkungan kampus Universitas Diponegoro, yaitu pos keamanan pintu masuk bundaran Universitas Diponegoro. Pengujian dilakukan selama 3 hari dengan waktu 1 jam per hari yang dimulai dari jam 10:30 hingga 11:30. Total kendaraan yang dideteksi hingga akhir pengujian sebanyak 1200 sepeda motor. Hasil dari pengujian integrasi model deteksi secara *real-time* dapat dilihat pada tabel 10 berikut ini.

Tabel 10 Hasil Analisis dari Pengujian Integrasi Model Deteksi secara *Real-time*

Jadwal	Jumlah Kendaraan	Pelanggar yang terkirim	Pelanggar yang dilewatkan	Pelanggar yang salah
Hari ke-1	290	93	3	1
Hari ke-2	483	149	5	1
Hari ke-3	427	146	6	2
Total	**1200**	**388**	**14**	**4**

Pengujian ini berfokus pada label nonhelm yang dianggap sebagai pelanggar yang dilakukan dengan mencatat data aktual dan membandingkannya dengan data hasil prediksi. *Confusion Matrix* sebagai *validasi* yang didapat dari membandingkan data tersebut dapat dilihat pada Gambar 10 sebagai berikut.

Total Keseluruhan			
Jumlah Data : 1200		Predicted Values	
		Negative	Positive
Actual Values	FALSE	798	4
	TRUE	14	384

Gambar 10 *Confusion Matrix* Pelanggaran Penggunaan Helm pada Sistem

Berdasarkan data yang didapat dari *confusion matrix* pada Gambar 10, didapatkan gambar yang masuk ke database sebanyak TP + FP (384 + 4 = 388). Perhitungan tersebut sesuai dengan pengecekan gambar yang terkirim oleh sistem dari validasi database. Gambar pelanggar yang seharusnya dikirimkan oleh model adalah TP + TN yaitu sebanyak 398 gambar, sehingga dari hasil pengujian ini dapat dihitung tingkat akurasi, *precision*, *recall*, dan *F1-score* dari sistem yang telah dibuat yaitu sebagai berikut :

a. *Accuracy* [15]

$$Accuracy = \frac{TP+FN}{TP+TN+FP+FN}$$

$$Accuracy = \frac{384+798}{384+14+4+798} = 98{,}5\ \%$$

b. *Precision* [15]

$$Precision = \frac{TP}{TP+FP}$$

$$Precision = \frac{384}{384+4} = 98{,}97\%$$

c. *Recall* [15]

$$Recall = \frac{TP}{TP+FN}$$

$$Recall = \frac{384}{384+14} = 96{,}48\%$$

d. *F-1 Score* [15]

$$F-1\ Score = 2 * \frac{Precision*Recall}{Precision+Recall}$$

$$F-1\ Score = 2 * \frac{0{,}98*0{,}96}{0{,}98+0{,}96}$$

$$F-1\ Score = 2 * \frac{0{,}94}{1{,}94} = 97{,}71\%$$

Hasil dari perhitungan pengujian ini menunjukkan bahwa sistem dapat bekerja dengan baik dengan akurasi yang tinggi dengan kesalahan deteksi yang minimal.

4 Kesimpulan

Training model mempelajari pola-pola dalam set pelatihan dengan baik serta memiliki kinerja model yang stabil dan baik pada data baru. Hal ini terlihat dari perbedaan *loss* pada *training* dan validation yang sangat rendah (dibawah 0,1). Hasil evaluasi dari model terbaik menunjukkan nilai *mean average precision* (mAP) rata-rata sebesar 0,938 dimana untuk label 'helm', 'motor', dan 'non-helm' secara berurut sebesar 0,951; 0,955; 0,907. Tingkat akurasi, *precision*, *recall*, dan *F1-score* dari integrasi sistem keseluruhan secara berurutan adalah 98,5%, 98,97%, 96,48%, dan 97,71%. Hasil pengujian menunjukkan akurasi sistem sebesar 98,5%. Hasil dari perhitungan pengujian ini menunjukkan bahwa sistem dapat bekerja dengan baik dengan akurasi yang tinggi dengan kesalahan deteksi yang minimal.

Referensi

[1] Korlantas Polri, "Grafik Data Kendaraan," Korlantas Polri. Accessed: Aug. 18, 2023. [Online]. Available: http://rc. korlantas. polri.go. id:8900/eri2017/laprekappolres.php?kdpolda=18 &poldanya=LAMPUNG

[2] Badan Pusat Statistik, "Badan Pusat Statistik," Badan Pusat Statistik. Accessed: Aug. 18, 2023. [Online]. Available: https://archive.bps.go.id/ indicator/17/513/1/jumlah-kecelakaan-korban-mati-luka-berat-luka-ringan-dan-kerugian-materi. html

[3] A. K. Berlalu, W. Ady, and D. B. Susantono,

"ANALISIS KESELAMATAN BERLALU LINTAS DI LINGKUNGAN KAMPUS UNDIP," *Tek. PWK (Perencanaan Wil. Kota)*, vol. 3, no. 4, pp. 693–707, Oct. 2014, doi: 10.14710/ TPWK. 2014.6724.

[4] Seth Weidman, *Deep Learning from Scratch*. O'Reilly Media, Inc., 2019.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 580–587, Nov. 2013, doi: 10.1109/CVPR.2014.81.

[6] U. Handalage and L. Kuganandamurthy, *Real-Time Object Detection Using YOLO: A Review*. 2021. doi: 10.13140/RG.2.2.24367.66723.

[7] S. Punia, "Artificial Intelligence and Intelligence Amplification," in *Nerds on Wall Street*, no. May, Wiley, 2012, pp. 149–158. doi: 10.1002/9781119201113.part3.

[8] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, "A survey of modern deep learning based object detection models," *Digit. Signal Process.*, vol. 126, p. 103514, Jun. 2022, doi: 10.1016/j.dsp.2022.103514.

[9] Y. Baştanlar and M. Ozuysal, *Introduction to Machine Learning Second Edition*, vol. 1107. 2014. doi: 10.1007/978-1-62703-748-8_7.

[10] J. Hurwitz and D. Kirsch, *Machine Learning For Dummies*, IBM Limite. John Wiley & Sons, Inc., 2018.

[11] E. Trivizakis and K. Marias, "Deep Learning Fundamentals," 2023, pp. 101–131. doi: 10.1007/978-3-031-25928-9_6.

[12] R. Santiago Teles de Menezes, R. Marrocos Magalhaes, and H. Maia, "Object Recognition Using Convolutional Neural Networks," in *Recent Trends in Artificial Neural Networks - from Training to Prediction*, IntechOpen, 2020. doi: 10.5772/intechopen.89726.

[13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.

[14] Ultralytics, "Comprehensive Guide to Ultralytics YOLOv5," Ultralytics YOLOv8 Docs. Accessed: Dec. 22, 2023. [Online]. Available: https://docs.ultralytics.com/yolov5/

[15] H. M and S. M.N, "A Review on Evaluation Metrics for Data Classification Evaluations," *Int. J. Data Min. Knowl. Manag. Process*, vol. 5, no. 2, pp. 01–11, Mar. 2015, doi: 10. 5121 /ijdkp.2015. 5201.