ELKOMIKA | ISSN (p): 2338-8323 | ISSN (e): 2459-9638 | Vol. 7 | No. 3 | Halaman 570 – 584
DOI : http://dx.doi.org/10.26760/elkomika.v7i3.570 September 2019

Deep Learning RetinaNet based Car Detection for Smart Transportation Network

IRMA AMELIA DEWI[1], LISA KRISTIANA[1], ARSYAD RAMADHAN DARLIS [2], REZA FADILAH DWIPUTRA[1]

[1]Program Studi Informatika, Institut Teknologi Nasional Bandung
[2]Program Studi Teknik Elektro, Institut Teknologi Nasional Bandung
Email: irma_amelia@itenas.ac.id

Received 16 Agustus 2019 **|** *Revised* 31 Agustus 2019 **|** *Accepted* 7 September 2019

ABSTRAK

Deteksi objek yang merupakan salah satu bagian utama dari sistem Smart Transportasion Network (STN) diajukan pada penelitian ini. Penelitian ini menggunakan salah satu model STN yaitu Infrastructure-to-Vehicle (I2V), dimana sistem ini bekerja dengan mendeteksi kendaraan mobil menggunakan model arsitektur RetinaNet dengan backbone Resnet101 dan FPN (Feature Pyramid Network), kemudian hasil deteksi mentrigger VLC transmitter yang terpasang di lampu penerangan jalan mengirimkan sinyal informasi menuju VLC receiver yang dipasang di mobil. Pada tahap proses training, jumlah dataset mobil yang digunakan adalah sekitar 1600 image dan 400 validation image serta pengulangan proses sebanyak 100 epoch. Berdasarkan 50 kali pengujian pada image test, diperoleh nilai precision mencapai 86%, nilai recall mencapai 85% dan f1-score mencapai 84% .

***Kata kunci**: Object detection, RetinaNet, Resnet101, STN, VLC, I2V*

ABSTRACT

Object detection is one of the main part in Smart Transportation Network (STN) system proposed in this research. This research used one of the STN models, namely Infrastructure-to-Vehicle (I2V), a system works by detecting car using RetinaNet architecture model with ResNet 101 and FPN (Feature Pyramid Network) as backbone, then the detection result triggers VLC transmitter set up on the street lighting to transmit information signal to the VLC receiver which set up in the car. At the training process stage, the number of car datasets is approximately 1600 images, 400 validation images and repetition of processes about 100 epochs. Based on the 50 times testing process on a image test, it is obtained 86% of a precision value, by reaching 85% of recall value, and 84% of f1-score.

***Keywords**: Object detection, RetinaNet, Resnet101, STN, VLC, I2V*

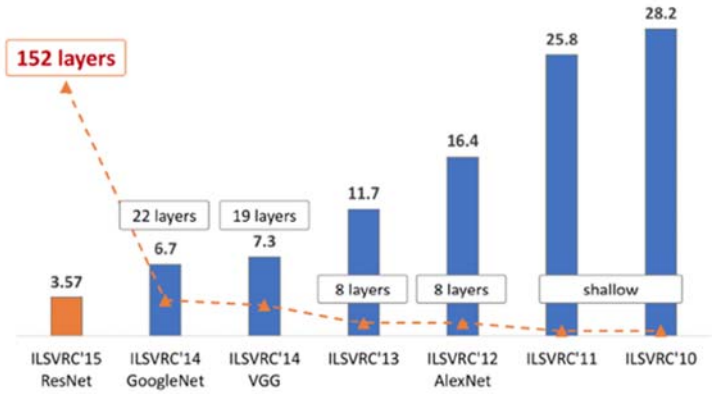
1. PENDAHULUAN

Smart Transportation Network (STN) merupakan teknologi yang memanfaatkan cahaya dari lampu infrastruktur dalam sebuah jaringan komunikasi menggunakan transportasi. Cahaya yang digunakan dapat diperoleh dari lampu penerangan jalan dan lampu yang terdapat pada mobil digunakan untuk mengirimkan informasi dari satu tempat ke tempat yang lain secara *broadcast*. Bentuk sistem komunikasi pada STN, dibagi menjadi tiga model, yaitu Model *Infrastructure to Vehicle* (I2V), Model *Vehicle to Vehicle* (V2V) dan model *Vehicle to Human* (V2H) **(Kristiana, Schmitt, & Stiller, 2017a) (Kristiana, Schmitt, & Stiller, 2017b) (Kristiana, Schmitt, & Stiller, 2017c) (Kristiana, Schmitt, & Stiller, 2017d).** Pada penelitian ini hanya membahas model I2V yang bekerja dengan mengoptimalkan infrastruktur, dimana kendaraan berupa mobil yang merupakan salah satu dari transportasi yang melintasi jalan, baik di dalam kota ataupun antar kota untuk menerima, membawa dan mengirimkan informasi secara *broadcast* dan *massive*. Model I2V terdiri dari proses deteksi objek, VLC (*Vehicle Light Communication*) *transmitter* dan VLC *receiver.* Proses diawali dengan mendeteksi objek khususnya adalah kendaraan mobil, hasil dari deteksi objek dikirimkan ke VLC *transmitter* yang terpasang pada lampu penerangan jalan dan VLC *receiver* yang terpasang di atap mobil pada kondisi lingkungan yang mendekati sebenarnya, baik dalam hal jarak, maupun lainnya. VLC adalah teknologi pada sistem komunikasi yang memanfaatkan *visible light*, yaitu radiasi elektromagnetik dengan panjang gelombang 380 nm sampai 750 nm yang dapat dilihat oleh mata manusia. Dalam beberapa penelitian membuktikan bahwa sistem komunikasi dapat memanfaatkan cahaya tampak **(Darlis, Lidyawati, & Nataliana, 2013) (Darlis, Lidyawati, Nataliana, & Wulandari, 2014) (Darlis, Lidyawati, & Jambola, 2018) (Darlis, Cahyadi, & Chung, 2018)**.

Fokus penelitian ini membahas mengenai *object detection* atau *car detection* sebagai objek yang dideteksi. Deteksi objek digunakan untuk mengklasifikasikan banyak objek dari *single image*, tidak hanya memberikan informasi berupa kelas dari masing-masing objek tetapi juga berhubungan dengan pelabelan oleh suatu *bounding box* beserta jenis objek untuk menunjukan keberadaanya **(Tang & Yuan, 2015) (Szeliski, 2011)** atau koordinat (x,y) dari setiap objek **(Foley & O'Reilly, 2015)**. Deteksi objek pada penelitian ini memanfaatkan teknik *deep learning* yang menggunakan jaringan hierarki *multilayer* untuk menghasilkan *feature map* yang mengoptimalisasikan kinerja pada *training data* **(Nguyen, Ross, Fookes, & Sridharan, 2017).** Dalam beberapa tahun terakhir telah banyak penelitian mengenai algoritma *object detection* yang menggunakan metode *Convolutional Neural Networks* (CNN) **(Ding, Lin, He, Wang, & Huang, 2018) (Foley & O'Reilly, 2015)** seperti *Faster* R-CNN **(Hsu, Huang, & Chuang, 2018) (Ren, He, Girshick, & Sun, 2016)**, R-FCN **(Dai, Li, He, & Sun, 2016)**, *multi-box single shot detectors* (SSD) **(Liu, et al., 2016) (Biswasa, Su, Wang, Stevanovic, & Wang, 2018)**, dan Yolo **(Redmon & Farhadi, 2017)**.

Seiring dengan berkembang dan beragamnya aplikasi *computer vision*, maka dikembangkan model arsitektur membuat CNN lebih dalam dan kuat dalam kinerjanya. Salah satu model arsitektur yang menggunakan *one-stage detector* dari CNN yaitu RetinaNet. Arsitektur RetinaNet memiliki keakurasian melampai *two-stage-detector* dan diatas rata-rata dari *one-stage detector* dalam *focal loss* pada *train data* **(Lin, Goyal, Girshick, He, & Doll´ar, 2018)** seperti juga beberapa penelitian yang telah dilakukan sebelumnya **(Milton, 2018) (Wang, Wang, Zhang, Dong, & Wei, 2019)**. RetinaNet dapat bekerja dengan beragam *backbone* CNN seperti ResNet, (ResNet50, ResNet101), DenseNet, VGG net-16, dan VGG net-19 **(Hoang, Nguyen, Truong, Lee, & Park, 2019)**. Berdasarkan acara tahunan ILSVRC (*ImageNet Large Scale Visual Recognition Challenge*) yang dijadikan kegiatan untuk mengenalkan kinerja beragam model *backbone*. Hasil evolusi pemenang ILSVRC dari 2010

sampai dengan 2015 arsitektur yang mencapai *error rate* terendah sekitar 3.5% adalah model detektor RetinaNet dengan backbone arsitektur ResNet seperti pada Gambar 1. Resnet 101 yang telah berhasil memenangkan kontes seperti ILSVRC dan COCO (*Common Objects in Context)* 2015 (deteksi, segmentasi dan klasifikasi) **(Arcos-Garcia, Alvarez-Garcia, & Soria-Morillo, 2018) (Ding, Lin, He, Wang, & Huang, 2018).**



Gambar 1. Evolusi Pemenang ILSVRC dari tahun 2010-2015 (Nguyen, Ross, Fookes, & Sridharan, 2017)

Dikarenakan deteksi objek pada sistem STN ini merupakan bagian yang utama, maka membutuhkan pemilihan metode yang tepat dalam memilih *object detector*. Berdasarkan dari beberapa tinjauan pustaka tersebut, maka pada penelitian ini melakukan deteksi objek mobil untuk memperoleh hasil pendeteksian yang optimal dari segi presisi, *recall* dan akurasi sistem dengan menggunakan model arsitektur RetinaNet dan ResNet 101+FPN sebagai *backbone*.

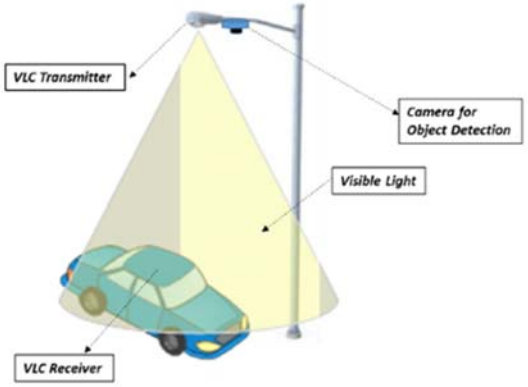
Cara Kerja Sistem STN

Pada penelitian sistem STN model I2V ini bekerja diawali dengan mendeteksi kendaraan mobil dengan menggunakan kamera sebagai input proses deteksi objek. Kemudian hasil dari deteksi objek menghasilkan output sinyal 1 atau *High* diterima oleh VLC *transmitter* yang diposisikan di lampu penerangan jalan. Pada penelitian ini, sistem dibuat dalam mode satu arah (*directional*) dari lampu penerangan kepada mobil, sinyal informasi berupa audio dikirimkan dari lampu penerangan menuju mobil sebagai media transportasi memanfaatkan cahaya tampak (*Visible Light*), seperti yang diilustrasikan sistem STN model I2V pada Gambar 2. Sehingga dalam mengimplementasikan STN, pada dasarnya blok dibagi menjadi dua buah blok, yaitu Blok *Object Detection* dan *Visible Light Communication* (VLC) seperti pada blok diagram STN Gambar 3. Model I2V berupa *transmitter* yang meliputi sistem *Object Detection* dan VLC yang terpasang pada lampu penerangan jalan dan *receiver* yang terpasang pada atap mobil pada kondisi lingkungan yang mendekati sebenarnya, baik dalam hal jarak, maupun lainnya.

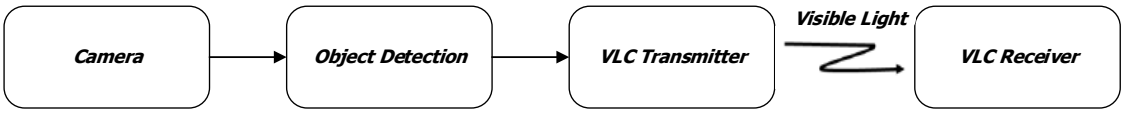
Arsitektur Model RetinaNet

Pada proses pendeteksi objek menggunakan aristektur RetinaNet. Aristektur RetinaNet memiliki tiga komponen yaitu jaringan *backbone* untuk ekstraksi fitur dan dua *subnetwork*, satu diantanya untuk *classification* dan lainnya untuk *box regression* seperti pada Gambar 4 **(Lin, Goyal, Girshick, He, & Doll´ar, 2018).** Untuk mengatasi skala objek maka arsitektur RetinNet menggunakan *Feature Pyramid Network* (FPN) sebagai *feature extractor*. Keuntungan menggunakan FPN adalah penerapan hierarki fitur piramidal pada jaringan *deep convolutional* untuk merepresentasikan *multi-scale objects*. Arsitektur model RetinaNet biasanya menggunakan jaringan *backbone* salah satunya seperti *Residual Network* (ResNet)

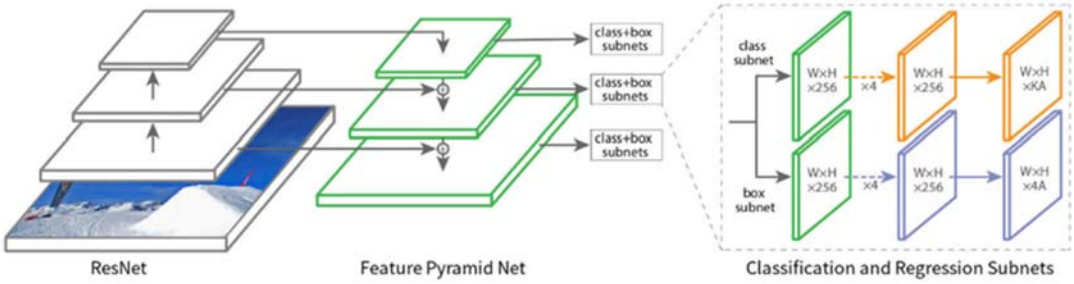
(He, Zhang, Ren, & Sun, 2015), untuk mengekstraksi semantik tertinggi *feature map* dan menerapkan FPN untuk mengekstraksi fitur dimensi yang sama dengan beragam skala.



Gambar 2. Ilustrasi Sistem *Infrastructure-to-Vehicle (I2V)* pada *Smart Transportation Network (STN)*



Gambar 3. Blok Diagram *Smart Transportation Network* (STN)



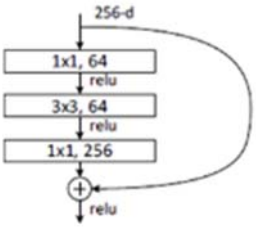
Gambar 4. Arsitektur RetinaNet menggunakan *Backbone* ResNet (Lin, Goyal, Girshick, He, & Doll´ar, 2018)

Arsitektur RetinaNet Pada Gambar 4 bagian yang berwarna hijau adalah fitur piramidal yang diekstraksi oleh FPN, bagian yang berwarna orange merupakan *subnet* klasifikasi dan bagian yang berwarna ungu merupakan *subnet bounding box regression*.

Feature Pyramid Network (FPN) dan *Backbone Residual Network (*ResNet)

FPN bertindak sebagai fitur ekstraktor yang memiliki dua jalur, yaitu jalur *bottom-up* dengan menerapkan jaringan konvolusi untuk mengekstraksi fitur dari input resolusi tinggi dan jalur *top-down* digunakan mengkontruksi layer *high-resolution multi-scale* dari layer paling atas di jalur *bottom-up.* Adapun pada jalur *top-down*, pertama-tama mengadaptasi konvolusi 1 x 1 untuk mereduksi jumlah *feature map channel* ke 256 dan menggunakan koneksi lateral untuk mengkombinasikan *feature map* yang berhubungan dan merekonstruksi layer untuk membantu memprediksi lokasi. Model *ResNet* merupakan suatu model yang membuat suatu

fungsi *shortcut* pada suatu jaringan operasi konvolusi yang diterapkan di jalur *bottom-up.* seperti yang ditunjukan oleh Gambar 5.



Gambar 5. *Building Block* untuk ResNet-50/101/152 (He, Zhang, Ren, & Sun, 2015)

Terdapat 4 macam arsitektur ResNet yang disesuaikan berdasarkan jumlah layernya yang terdiri dari 18 *layer*, 34 *layer*, 50 *layer*, 101 *layer*, dan 152 *layer* **(He, Zhang, Ren, & Sun, 2015).** Proses yang terjadi didalamnya pun berbeda-beda sesuai dengan jumlah *layer* tersebut, adapun detailnya bisa dilihat pada tabel 1.

Tabel 1. Modul Residual *Backbone* ResNet (He, Zhang, Ren, & Sun, 2015)

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

2. METODE PENELITIAN

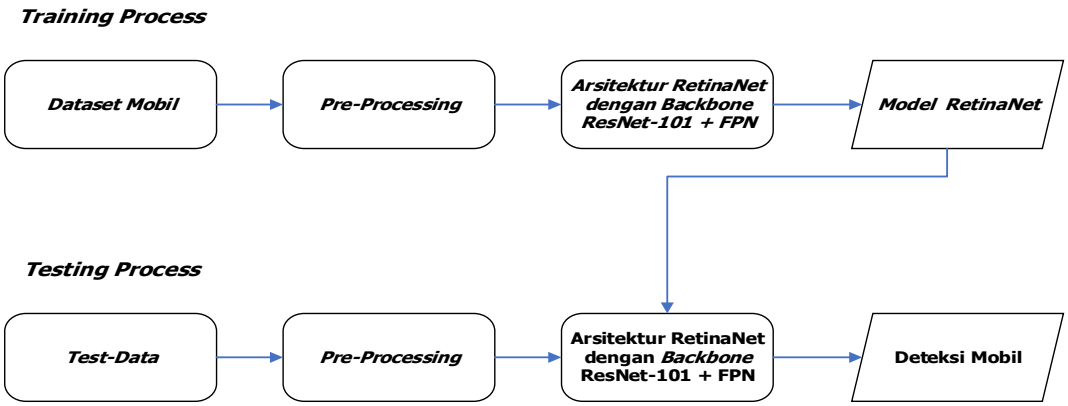
2.1. Sistem Deteksi Objek

Pada sistem *object detection* dibagi menjadi dua proses utama yaitu *training process* dan *testing process,* seperti pada Gambar 6. Pada proses *training process* dilakukan pembuatan model berdasarkan arsitektur RetinaNet terhadap file citra. Dataset terdiri dari *ground-truth boxes* yang menandakan keberadaan objek yang sebenarnya pada *training image* yang ditandai dengan *bounding box* (koordinat *xmin, xmax, ymin, ymax*) dan kelas objek yang disimpan pada file *.csv. Pada penelitian ini menggunakan data latih yang telah disediakan oleh *Udacity annotated driving dataset* **(Udacity, 2018).**

Proses pelatihan diawali dengan *preprocessing,* yaitu pembentukan variasi *training image* berdasarkan operasi *random transform* yang terdiri dari *rotation, flip, scaling,* dan *shear*. Selanjutnya proses ekstrasti *feature map* menggunakan model *Feature Pyramid Network* (FPN) dan jaringan *backbone* ResNet-101. Selama proses *training*, total *focal loss* dikomputasi sebagai jumlah dari keseluruhan 100.000 *anchor* setiap *image* yang dinormalisasi oleh jumlah *anchor* pada *ground-truth box*. Proses tersebut dilakukan sebanyak 100 *epoch*. Nilai *focal loss* dihitung dengan menggunakan Persamaan (1). Hasil keluaran proses *training* adalah model arsitektur RetinaNet yang disimpan dengan ekstensi file *.h5.

$$FL(p_t) = -\alpha(1-p_t)^{\gamma}\log(p_t) \quad (1)$$

Dimana, α adalah nilai antara 0 dan 1 untuk menyeimbangkan *sampel* berlabel positif dan *sampel* berlabel negatif untuk menyeimbangkan kelas dan γ adalah nilai skala positif. Bobot $\alpha(1-p_t)^{\gamma}$ adalah *cross entropy loss* yang bergantung terhadap nilai p_t, jika p_t lebih besar maka bobot lebih kecil dan sebaliknya.



Gambar 6. Blok Diagram *Car Detection*

Pada proses *testing* seperti pada Gambar 7, citra hasil *capture* kamera di*preprocessing* dengan mengkonversikan citra RGB menjadi BGR kemudian dilakukan operasi pengurangan terhadap *filter mode Caffe* seperti pada Persamaan (2).

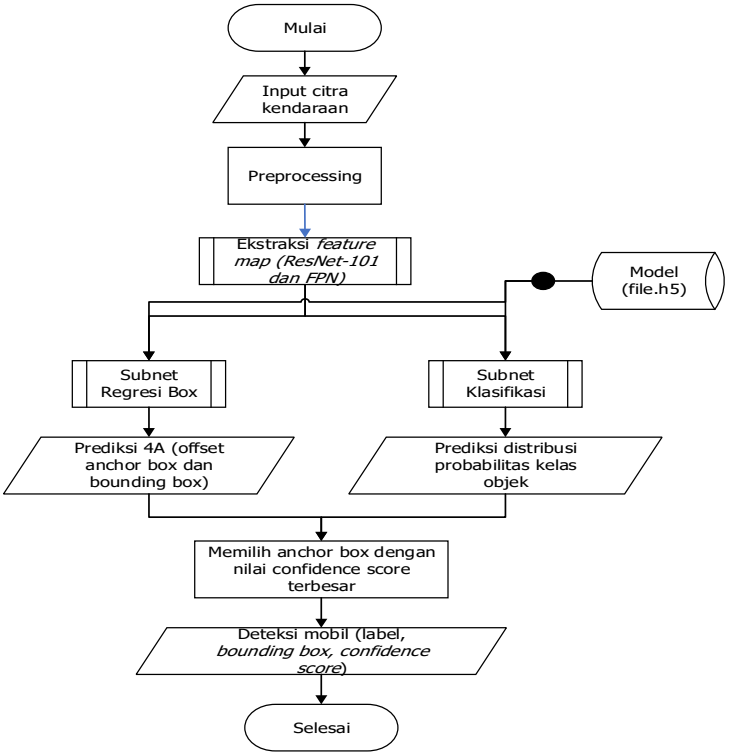
$$\begin{bmatrix} B^1 \\ G^1 \\ R^1 \end{bmatrix} = \begin{bmatrix} B \\ G \\ R \end{bmatrix} - \begin{bmatrix} 103{,}939 \\ 116.779 \\ 123.68 \end{bmatrix} \quad (2)$$

Proses selanjutnya adalah ekstraksi *feature map* dengan FPN dan jaringan *backbone* ResNet101 dengan langkah seperti pada *flowchart* Gambar 8. Proses *feature map* diawali dengan membuat *image pyramidal* seperti ilustrasi pada pada Gambar 10, semakin ke atas spasial resolusi semakin berkurang.

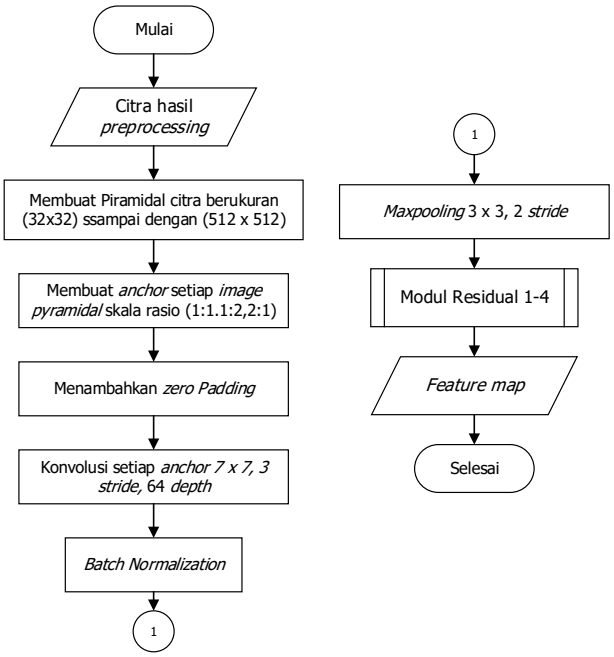
Setiap level piramida terdapat *anchor* dengan area 32^2, 64^2, 128^2, 256^2, 512^2 dari P3 sampai P7 dan aspek rasio *anchor* {1:2, 1:1, 2:1}, sehingga terdapat 15 *anchor* per level, seperti pada Gambar 11. Masing-masing *anchor box* bertanggungjawab mendeteksi keberadaan objek di area yang ditutupi tersebut. Area yang ditunjukan oleh *anchor* dikonvolusikan dengan matriks kernel 7x7, 2*strides* dengan jumlah kedalaman sebesar 64, ilustrasi *stride* konvolusi dapat dilihat pada Gambar 13. Namun sebelum proses konvolusi, hasil citra yang telah di*preprocessing* dilakukan *zero padding* dengan menambahkan nilai 0 pada setiap sisi matriks piksel, seperti contoh matriks piksel Tabel 2 yang dicuplik berukuran 3 x 2 menjadi 5 x 4. Hal ini dilakukan agar keseluruhan piksel 8 x 8 dapat dikonvolusikan.

Tabel 2. Matriks Piksel *Zero Padding*

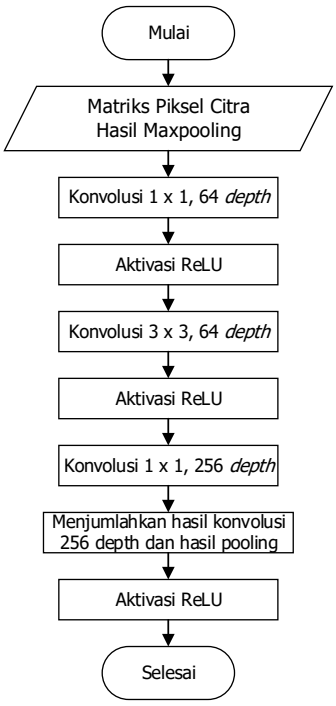
0	0	0	0	0
0	93	33	66	0
0	96	45	56	0
0	0	0	0	0



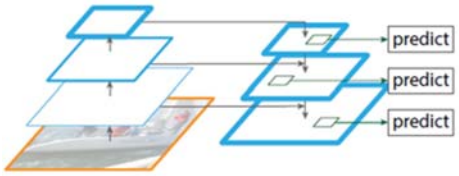
Gambar 7*. Flowchart* Sistem



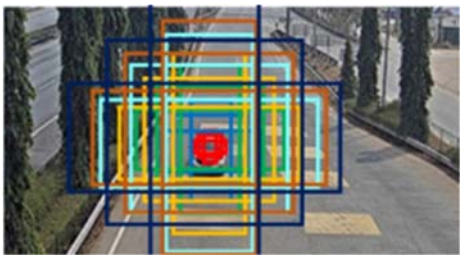
Gambar 8*. Flowchart* Proses *Feature Pyramidal Network* (FPN)



Gambar 9*. Flowchart* Proses Modul Residual (Modul Residual 1)



Gambar 10. *Feature Pyramid* (Lin, Dollar, & Girshick, 2017)



Gambar 11. *Anchor Box* pada *Image*

Proses konvolusi dilakukan dengan menggunakan Persamaan (3) setiap *stride* **(Munir, 2004)**:

$$h(x,y) = f(x,y) * g(x,y) = \sum_{a=-\infty}^{\infty} \sum_{b=-\infty}^{\infty} f(a,b) * g(x-a, y-b) \quad (3)$$

Proses konvolusi diilustrasikan seperti pada Gambar 12 yang dilakukan terhadap tiga *channel* warna BGR. Jika mengimplementasikan Persamaan (3), maka diperoleh perhitungan seperti pada Persamaan (4).

$$f(x,y) = Ap_1 + Bp_2 + Cp_3 + Dp_4 + Ep_5 + Fp_6 + Gp_7 + Hp_8 + Ip_9 \quad (4)$$

Kemudian dilanjutkan proses *Batch Normalization* (BN). Proses BN digunakan untuk meningkatkan kecepatan, performasi dan kestabilan pada jaringan dengan langkah-langkah sebagai berikut **(Ioffe & Szegedy, 2015)**:

1. Menghitung *mini batch mean* menggunakan Persamaan (5)

$$\mu_B \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i \quad (5)$$

2. Menghitung *Mini Batch Variance*

$$\sigma_B^2 \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2 \quad (6)$$

   Piksel citra yang diperoleh sebelumnya dikurangi dengan *mini batch mean,* selanjutnya dinormalisasi dengan mengkuadratkan nilai piksel. Setelah itu, merata-ratakan setiap kolom matriks citra.

3. Menghitung Normalisasi

$$\widehat{x_i} \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad (7)$$

   Hasil dari proses *mini batch variance* diakarkan, kemudian menghitung pembagian setiap baris pada matriks piksel

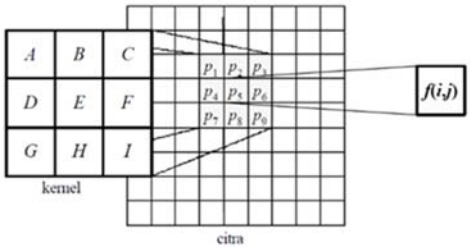
4. Menghitung *Scale and Shift*

$$y_i \leftarrow \gamma \hat{x_i} + \beta = BN_{\gamma,\beta}(x_i) \quad (8)$$

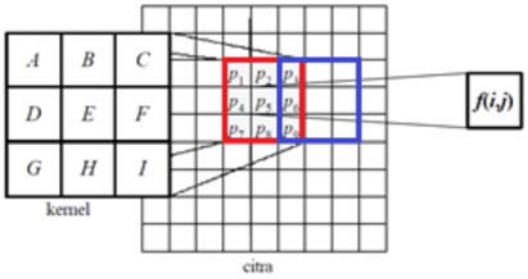
Dimana
γ = 1 dan β = 0
$\epsilon = 1e - 8$

Hasil dari proses BN dilanjutkan dengan proses aktivasi ReLU dengan menerapkan Persamaan (9).

$$ReLu(x) = \max(0, x) \quad (9)$$



Gambar 12. Ilustrasi Konvolusi citra (Munir, 2004)



Gambar 13. Ilustrasi *Stride* Konvolusi (2 *stride*)

Berikutnya adalah dilakukan proses *maxpooling,* Tabel 3 mengilustrasikan proses *maxpooling* 2 x 2, 2 *stride*.

Tabel 3. Ilustrasi Proses *Maxpooling* 2 x 2, S*tride*=2

49	50	34	55
60	55	45	66
56	48	50	70
34	78	67	87

⇒

60	66
78	87

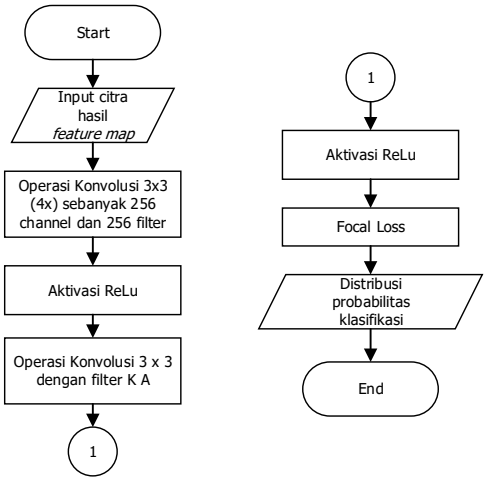
Hasil proses *maxpooling* akan diproses pada modul Residual ResNet. Proses Residual dilakukan sebanyak 4 modul, pada Gambar 9 merupakan residual modul pertama berdasarkan pada Tabel 1 layer ResNet 101, sehingga pada modul residual 2 sampai dengan 4 dilakukan proses yang sama seperti pada *flowchart,* namun dengan kernel konvolusi, kedalaman dan iterasi yang berbeda. Hasil ekstraksi fitur berupa *feature map* diproses selanjutnya secara paralel di *subnet classification* dan *subnet regression.*

Pada *subnet* klasifikasi seperti pada Gambar 14, *feature map* hasil ekstraksi dikonvolusikan sebanyak 4 kali terhadap matriks 3x3 dengan jumlah *channel* dan *filter* sebanyak 256. Kemudian hasil operasi tersebut diaktivasi menggunakan fungsi aktivasi ReLU. Selanjutnya hasil aktivasi ReLU dikonvolusikan kembali dengan *filter K A* (K=kelas, A=*anchor*), Proses berikutnya adalah penghitungan nilai *focal loss* pada setiap *anchor* di setiap tingkatan piramidal sebagai fungsi *loss*. Kemudian untuk mendapatkan nilai *focal loss* dari citra uji tersebut, jumlah *focal loss* pada keseluruhan *anchor* yang kira kira berjumlah 100.000 dijumlahkan dan dinormalisasi berdasarkan jumlah *anchor* yang ditetapkan pada *ground-truth box.* Untuk mendapakan hasil yang maksimal nilai γ yang digunakan pada *focal loss* sebesar 2 dan nilai α yang digunakan sebesar 0,25 **(Lin, Goyal, Girshick, He, & Doll´ar, 2018)**.

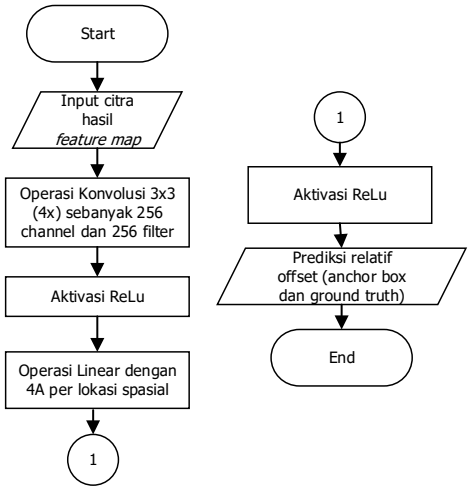
Pada subnet *box-regression* seperti pada Gambar 15*,* dilakukan proses yang sama seperti pada klasifikasi untuk setiap *output pyramid* dengan tujuan meregresi setiap kelebihan nilai pada

anchor box ke sekitar objek *ground-truth* jika terdapat objek. Pada masing-masing *subnet classification* dan *subnet regression* dilakukan perhitungan fungsi *loss* pada setiap *anchor*.

RetinaNet memilih lebih dari 1.000 *anchor box* yang memiliki *confidence score* tertinggi (probabilitas prediksi setiap kelas) dari masing-masing level FPN, setelah dilakukan *thresholding score* 0,05. Selanjutnya objek pada *image* dapat diprediksi oleh lebih dari satu *anchor box/bounding box*, sehingga untuk menguranginya dilakukan *Non-Maximum Suppression* (NMS) yaitu memilih *anchor box/bounding box* dengan *confidence score* tertinggi dan menghapus *bounding box* yang bertumpuk dengan *Intersection of Union* lebih besar dari 0.5, sehingga menghasilkan satu *bounding box* dengan nilai *confidence score* tertinggi.



Gambar 14. Alur Proses *Subnet* Klasifikasi

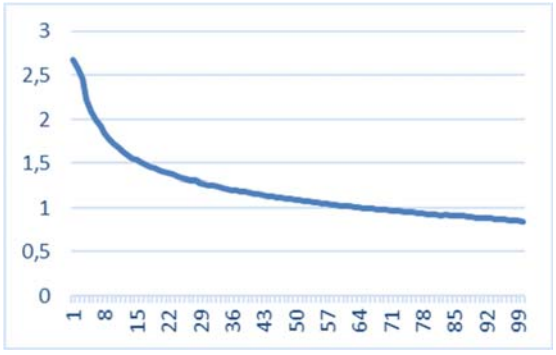


Gambar 15. Alur Proses *Subnet box-regression*

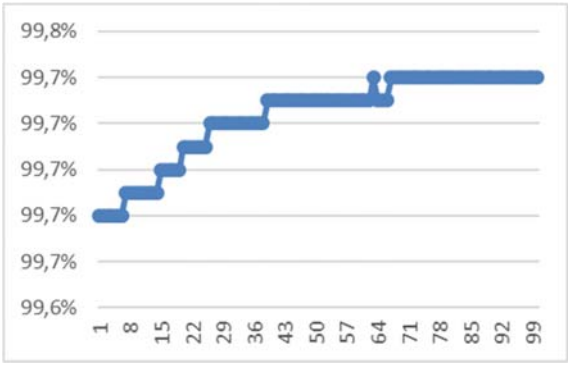
3. HASIL DAN PEMBAHASAN

Pada proses *training* telah dilakukan dengan memanfaatkan *Google Colabs Reasearch* dimana *engine* spesifikasi *hardware engine* adalah GPU Nvidia Tesla T4 16 GB Memory (16 GB Mem-100 GB SSD). Dengan spesifikasi hardware tersebut dilakukan *training* pada *image* yang sudah dianotasikan dalam bentuk format .csv berasal dari *Udacity annotated driving dataset* **(Udacity, 2018)**. *Dataset* yang digunakan di*split* dengan perbandingan 80:20 dari 2000 image, sehingga diperoleh 1600 *train image* dan 400 *validation image*. Proses *training* dilakukan sebanyak 100 *epoch* dengan waktu *training* sekitar 300s *per epoch*, sehingga diperoleh 30.000s atau sekitar 8.3 jam.

Berdasarkan hasil proses *training* menggunakan RetinaNet *backbone* ResNet-101 diperoleh nilai *loss* dan *accuracy* pada proses *regression* dan *classification*, dimana pada proses regresi menentukan koordinat atau lokasi objek berdasarkan *bounding box* dan proses klasifikasi menentukan ada tidaknya suatu objek. Nilai *loss* digunakan untuk mengestimasi *error* dan membandingkan serta mengukur seberapa baik hasil prediksi dengan menggunakan fungsi *focal loss*. Pada Gambar 16 merupakan kurva *loss* dalam 100 kali *epoch*, berdasarkan kurva diperoleh bahwa terjadi penurunan *loss* rata-rata mencapai 1.21.

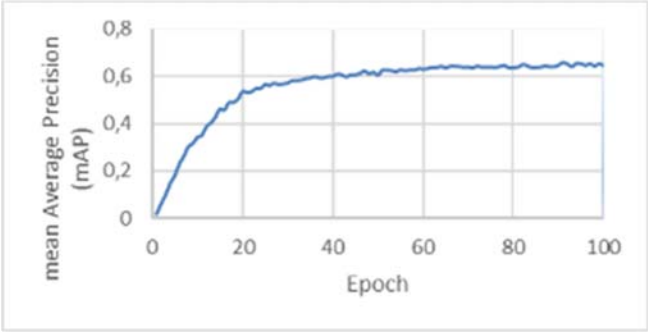


Gambar 16. Kurva *Loss* Proses *Training*



Gambar 17. Kurva *Accuracy* Proses *Training*

Sementara pada Gambar 17 merupakan hasil membandingkan performasi model berdasarkan keakurasian pada *training set* dan *validation set* dalam 100 kali *epoch*, diperoleh nilai akurasi proses klasifikasi mencapai 99.7%.



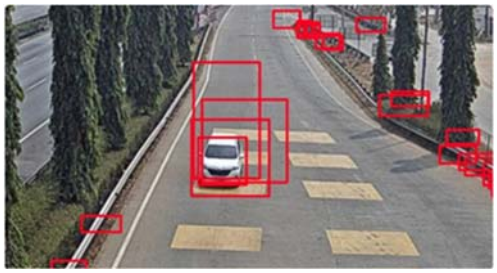
Gambar 18. mAP *Validation Data Set* terhadap *Train Data*

Pada proses *training* dilakukan pengujian terhadap seberapa presisi *train data* yang digunakan untuk mencegah kondisi *overfit* yang menyebabkan penurunan kinerja pada *test set* dengan menggunakan *validation set*. Dari hasil *training* diperoleh nilai *mean Average Precision* (mAP) sampai dengan *epoch* ke-100 mencapai 65% seperti yang ditunjukkan pada Gambar 18.

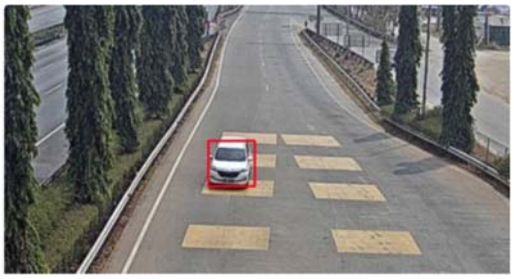
Untuk pengujian sistem telah dilakukan pada 50 data uji. Citra uji diinputkan seperti pada Gambar 15 pada sistem kemudian dilakukan proses *preprocessing* mengkonversi citra RGB menjadi BGR dan dikurangi menggunakan *mode Caffe* untuk mengatasi permasalahan warna dan menghasilkan citra seperti pada Gambar 20. Kemudian dilanjutkan dengan proses ekstraksi ciri menghasilkan *feature map* melalui proses FPN dan dilanjutkan diproses di *subnet regression* dan *classification*. Jika dilihat pada Gambar 21 menghasilkan banyak *box* di area gambar, maka untuk menilai ketepatan objek pada gambar dilakukan meminimalisir *box* pada nilai *confidence score* dengan menerapkan nilai *threshold* 0.5, sehingga *bounding box* dengan nilai *confidence score* kurang dari 0.5 dihapus. Setelah itu, semua *bounding box* dengan nilai *confidence score* diatas 0.5 diambil nilai *Intersection over Union* (IoU) yang terbesar, sehingga menghasilkan satu *bounding box* yang dianggap sebagai objek yang diprediksi seperti yang ditunjukan pada Gambar 22. Hasil dari proses ini, menghasilkan satu *bounding box* pada setiap objek yang terdeteksi. Hasil akhir proses berupa pendeteksian objek berupa *bounding box* pada lokasi *image* dengan label nama kelas dan *confidence score*.

Gambar 19. Citra Uji

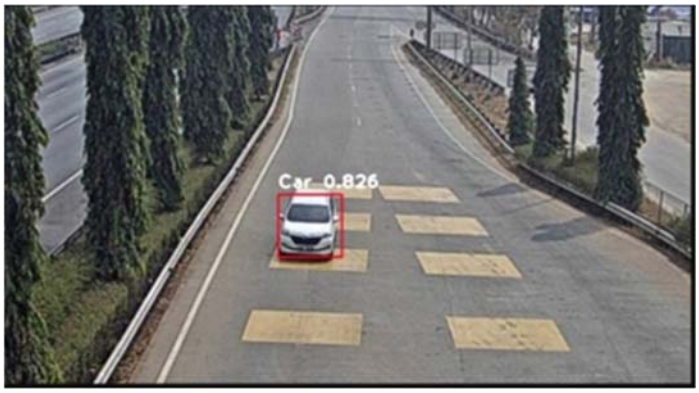
Gambar 20. Citra Hasil *Preprocessing*



Gambar 21. Citra Hasil *Feature Map*



Gambar 22. Citra Hasil Proses Regresi



Gambar 23. Citra Hasil Proses klasifikasi

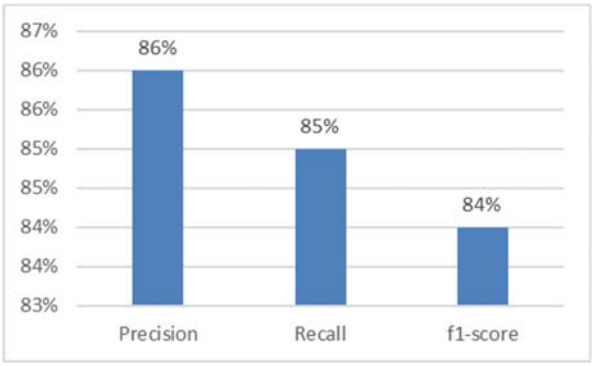
Untuk mengukur kinerja sistem dilakukan proses perhitungan *Precision, Recall*, *f1-Score* dan *Acuracy* dari proses pengujian *dataset test*. Dalam perhitungan parameter kinerja sistem dibutuhkan nilai dari *Ground Truth* (GT), *True Positif* (TP), *False Positif* (FP), *False Negatif* (FN). Dikarenakan pada penelitian ini hanya mendeteksi satu objek sehingga tidak perlu mencari nilai *True Negatif* (TN). Nilai GT diperoleh dengan menentukan jumlah objek sebenarnya yang terdeteksi, pada penelitian ini dilakukan dengan menggunakan data *test-set* dari Udacity **(Udacity, 2018)**. Nilai TP diperoleh, jika diasumsikan bahwa Positif (P) adalah observasi positif dalam penelitian ini yaitu objek mobil, maka Negatif (N) adalah observasi *non-positive* /negatif yang menandakan objek selain mobil. Maka nilai TP adalah observasi positif dan diprediksi positif sebagai mobil, sementara FP adalah observasi negatif tetapi diprediksi postif atau jumlah data positif namun terklasifikasi salah oleh sistem, dan FN adalah jumlah data negatif, dan terklasifikasi salah oleh sistem. Sehingga untuk mengukur keakurasian sistem dilakukan perhitungan terhadap nilai *precision, recall* dan *f1-score*. Nilai presisi sistem deteksi dihitung menggunakan Persamaan (10).

$$Precision = \frac{TP}{FP+TP} * 100\% \quad (10)$$

Sementara untuk memperoleh nilai *Recall* sistem dilakukan perhitungan menggunakan Persamaan (11) dan *f1-score* dihitung menggunakan Persamaan 12.

$$Recall = \frac{TP}{FN+TP} * 100\% \quad (11)$$

$$F - Measure = \frac{2*Recall*Precision}{Recall+Precision} \quad (12)$$



Gambar 24. Grafik Kinerja Sistem

Dengan menentukan nilai presisi/akurasi dapat menunjukkan seberapa presisi model dalam menentukan objek positif yang sesuai dengan aktualnya. Berdasarkan hasil pengujian terhadap 50 citra uji diperoleh presisi sistem mengklasifikasikan objek dengan tepat mencapai 86% seperti yang dapat dilihat pada grafik Gambar 24. Sementara *Recall mengukur* kemampuan model untuk menemukan semua objek positif/mobil, berdasarkan pengujian sistem mampu melakukan *recall* sebesar 85%. Pada pengujian juga dilakukan pengukuran dua parameter nilai *precision* dan *recall* pada saat bersamaan dikarenakan terdapat data pengujian dengan nilai *precision* tinggi dan *recall* rendah atau sebaliknya, sehingga dilakukan pengukuran nilai *f1-score* atau *F-Measure* menggunakan *Harmonic Mean* menggunakan Persamaan (12). Pada pengujian sistem untuk menentukan keseimbangan nilai *precision* dan *recall* diperoleh nilai *f1-score* mencapai 84%.

4. KESIMPULAN

Penggunaan *validation set* pada proses *training* dalam pembentukan model *deep learning* menggunakan arsitektur RetinaNet dengan ResNet 101 dan FPN sebagai *backbone* dapat mencegah terjadinya kondisi *overfit.* Pada proses *training* diperoleh *regression loss* mencapai 1.21 dan akurasi pada klasifikasi mencapai 99,7% sementara untuk *mean Average Precision* (mAP) mencapai 65%. Proses pengujian dilakukan untuk mengukur kinerja sistem diperoleh *precision* mencapai 86%, nilai *recall* mencapai 85% dan dihitung juga nilai *f1-score* untuk mengukur keseimbangan *precision* dan *recall* sistem, maka diperoleh nilai *f1-score* sebesar 84%. Hal-hal yang mempengaruhi tingkat kinerja sistem adalah dipengaruhi jumlah *training data* dan *validation data* serta jumlah *epoch*.

UCAPAN TERIMA KASIH

Penelitian ini didanai penuh oleh Kementrian Riset, Teknologi, Dan Pendidikan Tinggi Republik Indonesia dalam hibah Penelitian Terapan Unggulan Perguruan Tinggi (PTUPT) 2019.

DAFTAR RUJUKAN

Arcos-Garcia, A., Alvarez-Garcia, J., & Soria-Morillo, L. (2018). Evaluation of Deep Neural Networks for traffic sign detection systems. *Elsevier*, (pp. 332-344).

Biswasa, D., Su, H., Wang, C., Stevanovic, A., & Wang, W. (2018). An automatic traffic density estimation using Single Shot Detection (SSD). *Elsevier Ltd*.

Dai, J., Li, Y., He, K., & Sun, J. (2016). R-FCN: Object Detection via Region-based Fully Convolutional Networks. *arXiv:1605.06409v2 [cs.CV]*.

Darlis, A. R., Cahyadi, W. A., & Chung, Y.-H. (2018). Shore-To-Undersea Visible Light Communication. *Wireless Personal Communications, 99*(2), 681–694.

Darlis, A. R., Lidyawati, L., & Jambola, L. (2018). Color Filter Identification For Bidirectional Visible Light Communication. *Elkomika: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika, 6*(2), 303.

Darlis, A. R., Lidyawati, L., & Nataliana, D. (2013). Implementasi Visible Light Communication (VLC) Pada Sistem Komunikasi. *Elkomika: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika, 1*(1),13 – 25.

Darlis, A. R., Lidyawati, L., Nataliana, D., & Wulandari, N. (2014). Implementasi Sistem Komunikasi Video menggunakan Visible Light Communication (VLC). *Jurnal Reka Elkomika, 2*(3), 160 – 173.

Ding, X., Lin, Z., He, F., Wang, Y., & Huang, Y. (2018). A Deeply-Recursive Convolutional Network for Crowd Counting. arXiv:1805.05633v1.

Foley, D., & O'Reilly, R. (2015). An Evaluation of Convolutional Neural Network Models for Object Detection in Images on Low-End Devices. *CEUR Workshop Proceedings, Vol-2259.* Paris.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *CVPR*, pg.770-778.

Hoang, T. M., Nguyen, P. H., Truong, N. Q., Lee, Y. W., & Park, K. R. (2019). Deep RetinaNet-Based Detection and Classification of Road Markings by Visible Light Camera Sensors. *MDPI-Sensors, 19, 281*.

Hsu, S.-C., Huang, C.-L., & Chuang, C.-H. (2018). Vehicle Detection using Simplified Fast R-CNN. Chiang Mai: IEEE.

Kristiana, L., Schmitt, C., & Stiller, B. (2017a). Evaluation of inter-vehicle connectivity in three-dimensional cases. *Wireless Days, 2017.* Porto, Portugal: IEEE.

Kristiana, L., Schmitt, C., & Stiller, B. (2017b). Application of an enhanced V2VUNet in a complex three-dimensional inter-vehicular communication scenario. *IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob).* Bandung, Indonesia: IEEE.

Kristiana, L., Schmitt, C., & Stiller, B. (2017c). The evaluation of a predictive forwarding scheme in three-dimensional vehicular communication scenarios. *International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT).* Avignon, France: IEEE.

Kristiana, L., Schmitt, C., & Stiller, B. (2017d). The Evaluation of the V2VUNet Concept to Improve Inter-vehicle Communications. *IFIP International Conference on Autonomous Infrastructure, Management and Security.* Springer.

Lin, T.-Y., Dollar, P., & Girshick, R. (2017). Feature Pyramid Networks for Object Detection. *arXiv:1612.03144v2 [cs.CV]*.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Doll´ar, P. (2018). Focal Loss for Dense Object Detection. arXiv:1708.02002v2.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, Alex, C.-Y., & Berg, A. (2016). SSD: Single Shot MultiBox Detector. Springer.

loffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv:1502.03167v3 [cs.LG].

Milton, M. A. (2018). Towards Pedestrian Detection Using RetinaNet in ECCV 2018 Wider Pedestrian Detection Challenge. *arXiv:1902.01031*, 225-228. Diambil kembali dari arXiv.org.

Munir, R. (2004). Konvolusi dan Transformasi Fourier. Dalam *Pengolahan Citra Digital* (pp. 61). Bandung: Informatika.

Nguyen, K., Ross, A., Fookes, C., & Sridharan, S. (2017). Iris Recognition With Off-the-Shelf CNN Features: A Deep Learning Perspective. 6 .

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. Honolulu: IEEE.

Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection. *arXiv:1506.01497*.

Szeliski, R. (2011). Computer vision algorithms and applications. (pp. 10–17). London: Spinger.

Tang, S., & Yuan, Y. (2015). Object Detection based on Convolutional Neural Network. *Stanford University.* California.

Udacity. (2018). Self Driving Car. Mountain View: Github.

Wang, Y., Wang, C., Zhang, H., Dong, Y., & Wei, S. (2019). Automatic Ship Detection Based on RetinaNet Using Multi-Resolution Gaofen-3 Imagery. *MDPI-Remote Sensing, 11*.