Deteksi Pejalan Kaki pada *Zebracross* untuk Peringatan Dini Pejalan Kaki menggunakan Mask R-CNN

1st Mauridhi Hery Purnomo
Department of Computer Engineering
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
hery@ee.its.ac.id

2nd Eko Mulyanto Yuniarno
Department of Computer Engineering
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
ekomulyanto@ee.its.ac.id

3rd Agung Wicaksono
Department of Computer Engineering
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
agung.17072@mhs.its.ac.id

***Abstract*—Dewasa ini, fitur keselamatan pada kendaraan roda empat atau mobil sudah sangat berkembang pesat. Hal tersebut terbukti dengan banyaknya produsen mobil yang menerapkan teknologi seat belt, air bag, adaptive cruise control, electronic stability control, autonomous emergency braking, blind spot monitoring dan lain sebagainya. Namun, fitur yang sudah disebutkan diatas dinilai masih kurang ramah bagi pejalan kaki. Terbukti menurut data dari WHO, terdapat 270.000 pejalan kaki meninggal dunia setiap tahun atau sekitar 22% dari seluruh korban meniggal akibat kecelakan di jalan. Berawal dari permasalahan tersebut, penulis akan melakukan penelitian mengenai pendeteksian pejalan kaki pada zebracross untuk peringatan dini pengendara mobil sebagai topik penelitian. Pada tugas akhir ini, terdapat 3 objek yang akan dideteksi yaitu pejalan kaki, zebracross dan pengendara motor dengan menggunakan metode Mask R-CNN. Hasil terbaik yang didapatkan adalah pada penggunaan ResNet-101 untuk backbone Mask R-CNN dengan skor mAP sebesar 76.605%, mAR sebesar 85.375% serta F1-Score sebesar 80.302% .**

***Index Terms*—Pejalan Kaki, Zebracross, Mask R-CNN, Pengolahan Citra**

I. Latar Belakang

MOBIL merupakan salah satu jenis kendaraan bermotor yang banyak terdapat di Indonesia. Pada tahun 2019 Badan Pusat Statistik mencatat terdapat 15.592.419 mobil penumpang yang berada di Indonesia. Dengan bertambahnya jumlah mobil di Indonesia dari tahun ke tahun, meningkatkan juga jumlah kecelakaan mobil. Fitur keselamatan dan keamanan pada mobil sangat penting bagi para pengendara dan penumpang, sehingga para produsen mobil berusaha meningkatkan teknologi keselamatan dan keamanan pada mobil buatannya. Sebagai contoh beberapa fitur keselamatan dan keamanan yang terdapat pada mobil antara lain, *adaptive cruise control*, *hill strat assist*, *blind spot monitoring*, *electronic stability control* dan lain sebagainya.

Menurut data dari WHO, terdapat 270.000 pejalan kaki meninggal dunia setiap tahun atau sekitar 22% dari seluruh korban meniggal akibat kecelakan di jalan [1]. Sedangkan untuk di Indonesia sendiri, mengutip dari laman *Global Road Safety Facility*, presentase kematian pejalan kaki akibat kecelakaan lalu lintas sebesar 38% dari total 31.282 kematian di jalan raya yang dilaporkan pada tahun 2016 [2]. Melihat kegiatan para pejalan kaki yang jarang berada di badan jalan, angka tersebut tentu cukup tinggi. Para pejalan kaki hanya menggunakan badan jalan ketika hendak menyebrang jalan lewat zebracross. Kelalaian dari pejalan kaki maupun pengendara mobil merupakan faktor utama mengapa angka kematian pejalan kaki cukup tinggi. Salah satu contoh kelalaian pejalan kaki adalah pada saat menyebrang jalan tidak memperhatikan kendaraan yang akan lewat dan atau mengabaikan rambu serta lampu lalu lintas. Di sisi pengendara mobil, kelelahan, kurangnya fokus saat berkendara dan tidak memperhatikan rambu maupun marka dapat berakibat fatal baik kepada pejalan kaki dan pengendara lain.

Teknologi *artificial intelligent* tentu dapat digunakan untuk mengatasi masalah yang sudah disebutkan pada penjelasaan sebelumnya. Melihat penerapan *artificial intelligent* pada teknologi keselamatan pengendara mobil seperti *adaptive cruise control*, *hill start assist* dan lain sebagainya. Bagian dari a*rtificial intelligent* yang sangat cocok untuk masalah seperti ini adalah *deep learning* dengan memanfaatkan pengolahan citra pada data berbentuk gambar tentu dapat digunakan untuk deteksi pejalan kaki di zebracross guna mengurangi jumlah korban akibat kecelakaan. Deteksi pejalan kaki ini kedepannya dapat digabungkan dengan *buzzer* dan atau LED sebagai komponen *output* untuk mengingatkan kepada pengendara bahwa ada pejalan kaki yang sedang menyebrangi jalan serta mengembalikan fokus untuk berkendara.

Penelitian mengenai deteksi pejalan kaki di jalan raya sudah ada sebelumnya seperti *Real-Time Pedestrian Detection With Deep Network Cascades* (Anelia Angelova et al.) [3], dimana pada penelitian ini dihasilkan average miss rate sebesar 26,2% yang berjalan secara real time. Selanjutnya terdapat penelitian yang berjudul *Pedestrian Detection: The Elephant In The Room* (Irtiza Hasan et al.). [4]. Pada penelitian ini memband-

ingkan antara detektor objek secara umum yaitu Cascade R-CNN dengan detektor objek khusus pejalan kaki pada dataset yang berbeda. Pada penelitian yang berjudul *Fast Vehicle and Pedestrian Detection Using Improved Mask R-CNN* (Chenchen Xu et al.) [5] dijelaskan mengenai deteksi terhadap kendaraan yang bergerak dan pejalan kaki. Penggunaan Restnet 86 untuk menggantikan Restnet-101 sebagai backbone. Pada *MASK R-CNN for Pedestrian Crosswalk Detection and Instance Segmentation* (Mon Arjay Malbog). [6] deteksi hanya dilakukan pada pedestrian crosswalk atau zebracross saja. *Training data* dilakukan dengan Mask R-CNN untuk deteksi objek serta Restnet-101 sebagai backbone.

II. Desain dan Implementasi Sistem

Pada *paper* ini dijelaskan mengenai penerapan metode *Deep Learning* yang bertujuan untuk mendeteksi dan mensegmentasikan adanya pejalan kaki yang sedang melewati *zebracross* maupun berada disekitarnya. Gambar 1 menampilkan blok diagram dari sistem kerja yang digunakan untuk mendeteksi pejalan kaki pada *zebracross*. Pada penelitian ini, digunakan dataset dari Caltech Pedestrian Dataset.

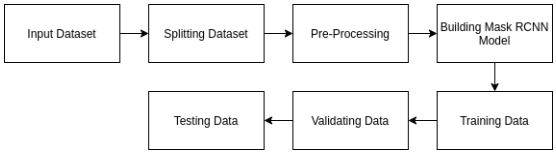


Fig. 1: Blok Diagram Sistem Kerja

A. Dataset yang Digunakan

Pada penelitian ini, *dataset* yang digunakan didapatkan dengan beberapa cara, antara lain:

1) *Caltech Pedestrian Database*, merupakan kumpulan gambar yang diambil dari sudut pandang pengendara mobil di California Amerika Serikat dengan ukuran 640 x 480 pixel. Terdapat sekitar 250.000 gambar dengan 350.000 *bounding boxes* dan sekitar 2.300 pejalan kaki dengan kriteria unik diberi tanda. Namun, pada *dataset* ini hanya pejalan kaki saja yang diberi label, sehingga perlu dilakukan proses pelabelan ulang sesuai kelas yang diinginkan. Tidak semua gambar pada *dataset* ini diambil untuk digunakan, gambar yang mempunyai objek berupa pejalan kaki dan *zebracross* saja yang akan digunakan. Gambar 2 merupakan contoh dari gambar yang terdapat pada *Caltech Pedestrian Database*.

Fig. 2: Contoh Gambar dari Caltech Pedestrian Database

2) Tangkapan layar dari beberapa video *online Youtube*. Pada cara ini, penulis mencari video yang berada pada salah satu *website video streaming* yaitu Youtube dengan persyaratan video diambil dari sudut pandang pengendara mobil yang berkendara pada jalan raya dengan ukuran gambar 1360x768 px. Pada *frame-frame* tertentu dilakukan *screenshot* dan disimpan untuk selanjutnya dilakukan proses pemberian label pada objek-objek yang diinginkan seperti pada Gambar 3.

Fig. 3: Contoh Pembuatan *dataset* dari *Screenshot Youtube*

3) Pengambilan gambar secara mandiri menggunakan kamera *smartphone* yang diambil dari sudut pandang pengendara motor dengan ukuran gambar yang diambil sebesar 1280x720 px. Pengambilan gambra dilakukan di jalan-jalan Surabaya. Setelah dilakukan pengambilan gambar, proses selanjutnya adalah pemberian label pada objek-objek yang ingin dideteksi.

Pada *Machine Learning* dataset dibagi menjadi 3 bagian, yaitu dataset untuk keperluan *training*, *validation* dan *testing*.

1) *Training Set* digunakan pada proses *learning* untuk melatih model dari algoritma yang sudah dibuat sebelumnya.
2) *Validatin Set* digunakan untuk mengukur kinerja dari model yang sudah dibuat dan menghindari terjadinya *overfitting*
3) *Testing Set* digunakan untuk menguji keberhasilan dari model yang sudah dibuat.

Pembagian rasio dataset pada *paper* ini sebesar, 70% untuk *training*, 20% untuk *validation* dan 10% untuk *testing*.

B. Pemisahan Data

Dalam *machine learning* pemisahan data ke beberapa *subset* merupakan suatu hal yang sangat penting. Hal ini dikarenakan setiap *subset* memiliki fungsi masing-masing. Gambar 4 merupakan rasio pembagian data ke masing-masing subset.

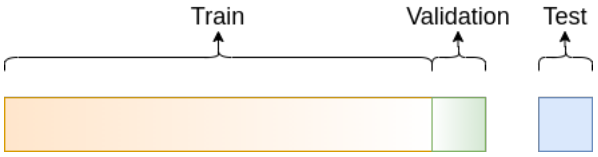


Fig. 4: Visualisasi Pembagian Data

1) *Training Sets*
Training Sets merupakan sampel data yang digunakan untuk melatih model yang sudah kita buat, dalam bidang *Neural Network* bisa disebut juga bobot dan bias. Model yang sudah kita buat mempelajari pola masukan dan keluaran dari data ini.

2) *Validation Sets*
Validation Sets merupakan sampel data yang digunakan untuk mengevaluasi model yang sudah dilatih menggunakan *training sets*. Selain itu, data ini digunakan untuk memperbarui dan menyempurnakan hyperparameter dari model ke tingkat yang lebih tinggi.
3) *Test Sets*
Test Sets merupakan sampel data yang digunakan untuk mengevaluasi model akhir setelah melalui proses *training dan validation*. Apabila pengujian model pada data ini sudah sesuai dengan yang diinginkan, maka proses *learning* sudah selesai. Namun apabila pengujian tidak sesuai dengan yang diharapkan maka diperlukan pengaturan ulang mulai dari proses *training*.

C. Pre-Processing

Pada tahap ini, gambar-gambar dari *dataset* akan mengalami proses penyesuaian sebelum masuk ke proses *data training*. Setiap gambar yang akan dijadikan bahan pembelajarnan model harus memiliki dimensi dan kedalaman yang sama. Tujuan dari *pre processing* adalah perbaikan data gambar dengan menekan distorsi yang tidak diinginkan atau meningkatkan beberapa fitur gambar yang relevan untuk pemrosesan lebih lanjut. Gambar 5 merupakan tahapan dari *pre-processing* gambar *dataset* yang dilakukan.

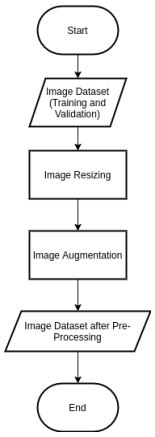


Fig. 5: Diagram Alir *Pre Processing*

Berikut merupakan penjelasan mengenai tahapan *pre-processing* yang dilakukan pada penelitian kali ini:

1) *Image resizing*
Langkah awal dari proses *pre-processing* adalah memastikan semua gambar dalam *dataset* kita memiliki ukuran yang sama. Selain itu, sama seperti sebagian besar model dari *neural network* lainnya, metode yang dilakukan penulis juga mengasumsikan gambar *input* berbentuk persegi. Jadi diperlukan pemeriksaan gambar di awal, apakah gambar sudah berbentuk persegi atau belum. Berbeda dari metode *image resizing* pada model *neural network* lainnya yang menggunakan teknik *cropping* untuk membuat aspek rasio gambar input menjadi persegi, penulis menggunakan metode yang sudah terdapat pada *Mask R-CNN*.
Ukuran gambar yang penulis pilih pada penelitian kali ini adalah 512x512 pixel. Pemilihan ukuran gambar ini dilakukan untuk mengurangi beban dan waktu saat *training data*. Apabila terdapat gambar pada *dataset* dengan ukuran baik panjang maupun lebar lebih dari 512 pixel. maka gambar akan di *down scaling* sampai ukuran 512 pixel. Sebaliknya, apabila ada gambar pada *dataset* dengan ukuran lebih kecil dari 512 pixel maka akan dilakukan *up scaling* sampai gambar berukuran 512 pixel. Aspek rasio gambar yang sudah melalui proses *scaling* tetap dipertahankan, namun diperlukan penambahan *zero padding* untuk membuat gambar *input* menjadi persegi seperti yang diinginkan.

Fig. 6: Contoh *Image Resizing*

Gambar 6 merupakan salah satu contoh *image resizing* yang dilakukan. Gambar *input* (gambar sebelah kiri) mempunyai ukuran 768x1360 dengan kedalaman 3 atau mempunyai format warna RGB. Setelah mengalami *image resizing* (gambar sebelah kanan) ukuran gambar menjadi 290x512. Namun untuk membuat gambar memiliki aspek rasio 1:1 (berbetuk persegi) maka diperlukan penambahan *zero padding* pada bagian atas gambar sebesar 111 pixel dan pada bagian bawah gambar sebesar 111 pixel. Dengan penambahn *padding* seperti itu membuat gambar *input* berbentuk persegi namun tidak mengurangi informasi gambar.
2) *Image Augmentation*
Langkah selanjutnya pada *pre-processing* adalah *image augmentaion*. Proses augmentasi yang dilakukan pada penelitian ini adalah rotasi dan transformasi. Tujuan dari penggunanaan *image augmentation* adalah untuk mengekspos *neural network* ke berbagai variasi, agar dapat mengenali fitur yang akan dilakukan pada proses *training*. Hal tersebut akan sangat membantu *neural network* untuk mengenali variasi yang tidak terdapat pada dataset. Seperti yang terdapat pada Gambar 7, tanpa ada augmentasi maka *neural network* hanya mengenali satu kondisi saja. Jika memakai augmentasi gambar seperti transformasi, maka setidaknya *neural network* akan dapat mengenali 2 kondisi. Semakin banyak augmentasi yang digunakan semakin banyak pula kondisi yang bisa dikenali oleh *neural network*. Namun semakin banyak kondisi yang dikenali, semakin lama dan berat proses *training data* yang dilakukan.

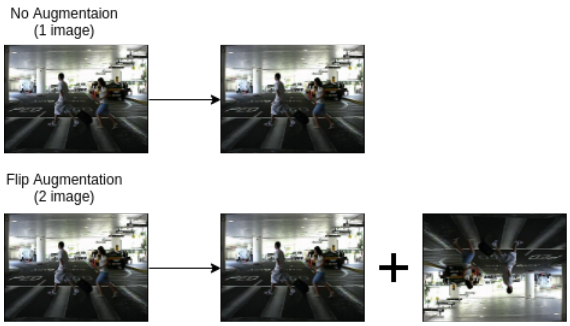


Fig. 7: Contoh *Image Augmentation*

D. Arsitektur Mask R-CNN

Mask R-CNN merupakah salah satu metode *deep lerning* yang dikembangkan dari Faster R-CNN dengan menambahkan satu cabang di tahap akhir untuk menghasilkan *mask* dari objek yang dideteksi [7]. Pengembangan tersebut dilakukan untuk memecahkan masalah *instance segmentation* yang terjadi dalam *machine learning* dan pengolahan citra. Dengan kata lain, *mask r-cnn*, dapat memisahkan objek yang berbeda walaupun dalam satu kelas yang sama pada gambar atau video. Selain memberikan hasil berupa *bounding box* dan klasifikasi objek seperti kebanyakan algoritma *object detection* lainnya, *mask r-cnn* juga memberikan *mask* dimana hal ini sangat bermanfaat pada segmentasi objek.

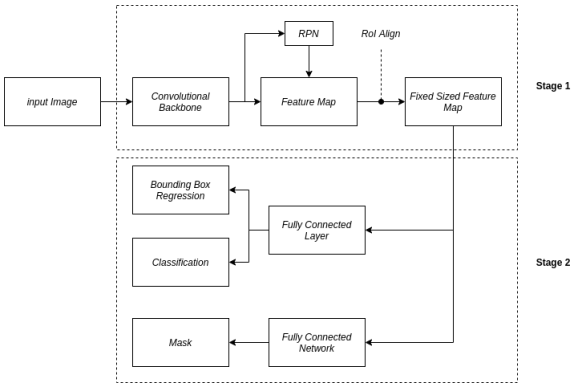


Fig. 8: Blok Diagram Alur Mask R-CNN

Seperti yang ditampilkan pada Gambar 8, pada proses *training* menggunakan *mask r-cnn* dibagi menjadi 2 tahapan. Tahap pertama adalah tahap untuk menghasilkan proposal tentang area di mana mungkin ada objek berdasarkan gambar *input*. Lalu tahap kedua adalah tahap untuk memprediksi kelas objek, memperbaiki *bounding box* dan menghasilkan *mask* di tingkat piksel objek berdasarkan proposal tahap pertama. Kedua tahap terhubung ke struktur *backbone*.

Backbone adalah *deep neural network* yang memiliki struktur seperti FPN (*Feature Pyramid Network*). *Backbone* terdiri dari *bottom-up pathway*, *up-bottom pathway* dan *lateral connection*. *Bottom-up pathway* dapat berupa berbagai jenis *Convolutional Network*, biasanya berupa *ResNet* atau *VGG*, yang mengekstrak fitur dari *raw images*. *Up-bottom pathway* menghasilkan *Feature Map Pyramid* yang ukurannya mirip dengan *bottom-up pathway*. *Lateral connection* adalah operasi konvolusi dan penjumlahan antara dua *pathway* dengan tingkat yang sesuai. FPN mempunyai kinerja yang lebih baik dari ConvNet tunggal lainnya terutama karena FPN dapat mempertahankan fitur semantik yang sangat baik pada berbagai skala resolusi

E. Training Process

Proses *training data* dilakukan setelah pembuatan model telah selesai dan *dataset* sudah melalui proses *pre-processing*. Pada saat pertama kali menjalankan proses *training*, bobot awal diambil dari *pre-trained weight* yang sudah tersedia pada *mask r-cnn*. Hal ini bisa dilakukan dengan menerapkan metode *transfer learning*. *Transfer learning* sendiri adalah teknik yang sangat efisien untuk melakukan proses *training* atau *retrain* pada *neural network*. Penggunaan *transfer learning* mempunyai keuntungan diantara lain proses *training* pada data baru memakan waktu yang lebih cepat daripada memulai dari awal serta masalah dapat dipecahkan dengan menggunakan training data yang lebih sedikit daripada membangun model dari awal.

Ada beberapa hal yang perlu diperhatikan dalam melakukan pengaturan saat akan menjalankan proses *training* antara lain :

1) *Iteration* adalah banyaknya proses yang dilakukan untuk melakukan *forward* dan *backward pass*. *forward pass* adalah proses dimana *output value* dari *neural network* didapatkan setelah *input value* dari *input-neuron* telah selesai diproses. Sedangkan *backward pass* adalah proses mengkalkulasikan bobot dari *neural network* mulai dari *output neuron* ke *input neuron* untuk mendapatkan *loss* dari setiap *neuron*. Pada *mask r-cnn iteration* bisa desebut juga dengan *step_per_epoch* sesuai dengan yang tercantunm pada *file* pengaturan *mask r-cnn*.
2) *Epoch*, ketika seluruh dataset sudah melalui proses *training* pada *neural network* sampai dikembalikan ke awal untuk sekali putaran. Sebagia contoh apabila kita menggunakan *iteration* sebanyak 10 kali, maka satu *epoch* sebanyak 10 *iteration* dan kelipatannya. Pada penelitian kali ini penulis menggunakan *epoch* sebanyak 100.

F. Validating Data

Setelah proses *training* dilakukan, perlu dilakukan apakah model yang dibuat sudah memiliki tingkat akurasi sesuai yang kita inginkan dengang menggunakan teknik validasi.Pada proses inilah, *dataset* yang telah dipisahkan pada proses sebelumnya akan berperan. Evaluasi memungkinkan pengujian model terhadap data yang belum pernah dilihat dan digunakan untuk pelatihan dan dimaksudkan untuk mewakili bagaimana model dapat menyelesaikan permasalahan tersebut. Tahapan pada validasi akan membantu untuk menemukan parameter terbaik untuk model prediktif dan mencegah dari *overfitting*.

Pada penelitian kali ini, digunakan salah satu jenis teknik validasi menggunakan *Cross Validation*. Pada *Cross Validation*

data akan dibagi menjadi K lipatan, dimana setiap lipatan akan diambil satu data sebagai *validation data* dan sisanya akan digunakan untuk *training data*. Pemilihan *validation data* dilakukan secara menyilang, dengan ketentuan apabila *training* terjadi pada iterasi ke *k* maka data yang dipilih untuk validasi adalah data *k* juga. Gambar 9 merupakan visualisasi dari *K Fold Cross Validation*.

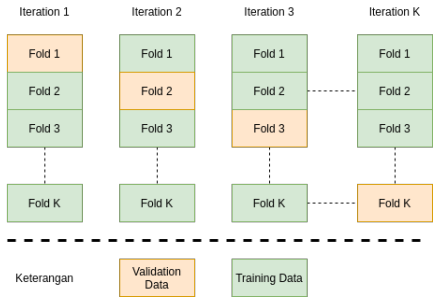


Fig. 9: Visualisasi *K Fold Cross Validation*

G. Testing Process

Testing data merupakan tahap akhir dalam algoritma *machine learning* secara umum. Pada tahap ini model akan diuji untuk didapatkan keakuratan untuk mendeteksi objek. Data uji yang digunakan pada tugas akhir kali ini adalah data yang berbentuk video. Jadi diperlukan *pre-processing* yang berbeda dengan *training data* dan *validation data*. Data video akan dipecah atau dipotong-potong menjadi format gambar dengan ketentuan 30 *frame per second*. Ketika *data test* sudah dikonversi menjadi bentuk gambar, maka proses deteksi bisa dilakukan. Gambar hasil deteksi berupa gambar asli yang sudah ditambah dengan *bounding box, classification, mask*. Lalu gambar-gambar tersebut disatukan lagi menjadi format video dengan ketentuan sama seperti saat pemotongan menjadi format gambar, yaitu 30 *frame per second*. Gambar 10 merupakan diagram alir dari proses *testing* dari model yang sudah dihasilkan dari proses *training dan validation*.

III. Testing and Result

A. Hasil Training dan Validasi

Setelah dilakukan serangkaian proses training didapatkan *output* berupa *model file* dengan format *h5*. *Training loss* terendah yang berhasil dicapai dengan menggunakan *backbone* Resnet-50 adalah 0.4061 dengan rincian *training bounding box loss* sebesar 0.04083, *training classification loss* sebesar 0.02268 serta *training mask loss* sebesar 0.139 (dimana $L = L_{bbox} + L_{cls} + L_{mask}$). Sedangkan *training loss* terendah yang berhasil dicapai dengan menggunakan *backbone* Resnet-101 adalah 0.3933 dengan rincian *training bounding box loss* sebesar 0.04164, *training classification loss* sebesar 0.0247 serta *training mask loss* sebesar 0.1403. Serta *training loss* terendah yang berhasil dicapai dengan menggunakan *backbone* Mobilenet-V1 (pada *epoch* ke 46) adalah 1.604 dengan rincian

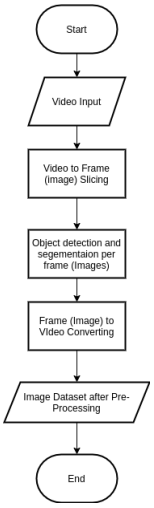


Fig. 10: Diagram Alir Proses *Testing*

training bounding box loss sebesar 0.2322, *training classification loss* sebesar 0.07858 serta *training mask loss* sebesar 0.6729. Gambar 11 merupakan grafik yang menunjukkan perubahan *training loss, training bounding box loss, training classification loss,* serta *training mask loss* dari *epoch* 1 sampai 300.

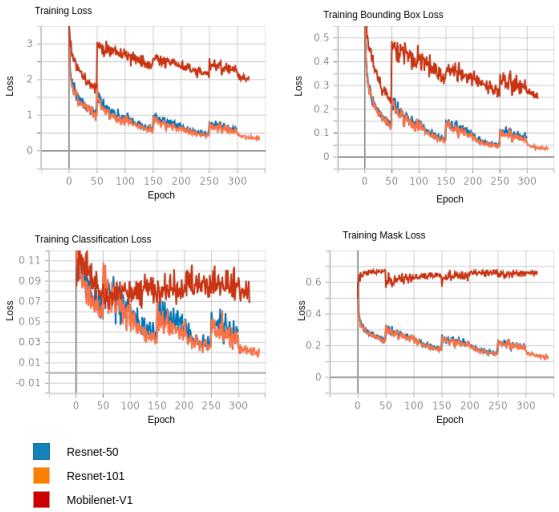


Fig. 11: Grafil Perubahan *Training Loss*

Sedangkan pada saat proses *validation* sendiri *Loss* terendah yang berhasil dicapai menggunakan Resnet-50 adalah 0.3653 dengan rincian *validation bounding box loss* sebesar 0.4556, *validation classification loss* sebesar 0.01912 serta *validation mask loss* sebesar 0.1519. Sedangkan *validation* sendiri *Loss* terendah yang berhasil dicapai menggunakan Resnet-101 adalah 0.299 dengan rincian *validation bounding box loss* sebesar 0.03867, *validation classification loss* sebesar 0.01246 serta *validation mask loss* sebesar 0.1461. Serta *validation* sendiri *Loss* terendah yang berhasil dicapai menggunakan Mobilenet-V1 adalah 1.624 dengan rincian *validation bounding box loss* sebesar 0.2088, *validation classification loss* sebe-

sar 0.04799 serta *validation mask loss* sebesar 0.6001. Gambar 12 merupakan grafik yang menunjukkan perubahan *validation loss, validation bounding box loss, validation classification loss,* serta *validation mask loss* dari *epoch* 1 sampai 300.

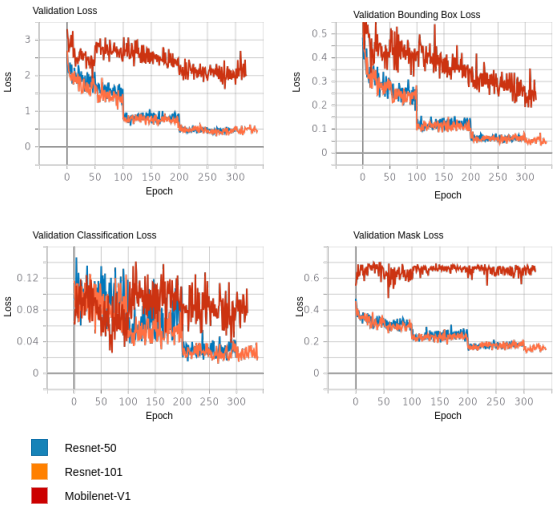


Fig. 12: Grafil Perubahan *Validation Loss*

Selain menggunakan *Loss Function* untuk mengukur peforma hasil *training* yang sudah dilakukan, digunakan juga *mean Average Precision (mAP). Precision* sendiri merupakan fungsi untuk menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Maka, *precision* merupakan rasio prediksi benar positif (TP) dibandingkan dengan keseluruhan hasil yang diprediksi positif (TP dan FP). Rumus untuk mencari *Precision* adalah sebagai berikut :

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

Perhitungan *mAP* pada penelitian ini dilakukan setiap 5 *epoch* sekali, karena jika dilakukan setiap *epoch* akan memerlukan *training time* yang lebih lama serta *resource hardware* yang diperlukan lebih besar. Nilai *mAP* tertinggi didapatkan pada dengan menggunakan Resnet-50 adalah sebesar 94.92. Sedangkat pada Resnet-101 *mAP* tertinggi yang berhasil dicapai sebesar 96.21 dan pada Mobilenet-V1 sebesar 26.04. Gambar 13 merupakan grafik yang menunjukan perubahan *validation mean Average Precision* dari *epoch* 1 sampai 300.

Tabel I adalah tabel perbandingan dari *output file size*, *mAP*, *training time* dari total 3 model dengan *backbone* berbeda yang diuji.

No	*Backbone*	*Output File Size*	*mAP*	*Training Time*
1	Resnet-50	170.9 MB	94.92%	03:40:24
2	Resnet-101	244 MB	96.21%	04:09:16
3	Mobilenet-V1	83.3 MB	26.04%	03:18:44

TABLE I: Tabel Perbandingan Hasil *Training*

B. Hasil Testing

Proses *testing* dilakukan menggunakan data dengan format gambar dan video dalam tiga kondisi yaitu pagi, siang dan malam hari. Pada *testing* dengan menggunakan *input* berupa gambar seperti yang tertampil pada Gambar 14 didapatkan hasil seperti pada Tabel II

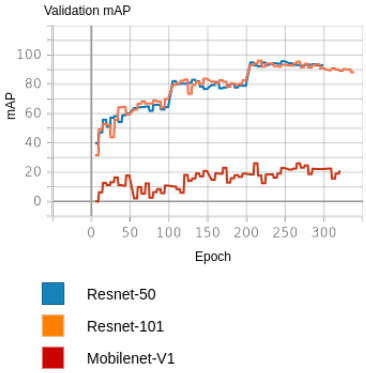
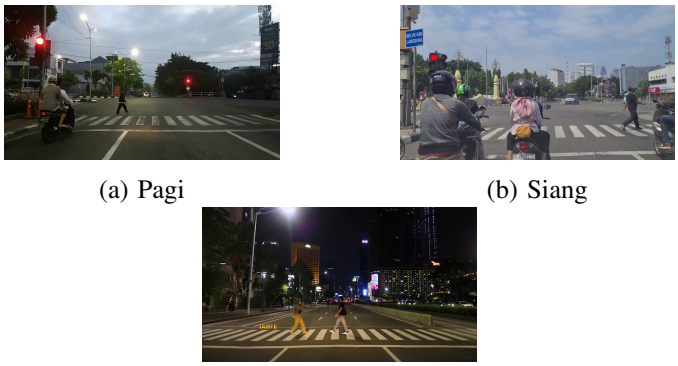


Fig. 13: Grafil Perubahan *Validation Mean Average Precision*



(a) Pagi (b) Siang

(c) Malam

Fig. 14: Gambar *Testing* pada Kondisi yang Berbeda

Backbone	*mAP (%)*	*mAR (%)*	*F1 Score (%)*
ResNet-50	66.785	77	71.529
ResNet-101	76.605	85.375	90.302
MobileNet-v1	25	Nan	Nan

TABLE II: Perbandingan Evaluasi Setiap Model

IV. Conclusion

Berdasarkan hasil pengujian yang telah dilakukan, penulis dapat menyimpulkan bahwa dalam penelitian ini telah diimplementasikan dengan baik proses deteksi dan segementasi pejalan kaki dan zebracross dengan menggunakan Mask R-CNN, dengan *mean Average Precision* sebesar 76.62%. Backbone ResNet-101 memiliki hasil akurasi yang lebih baik dibanding dengan *backbone* lainnya dengan peforma lebih tinggi sebesar 12.82% dibanding ResNet-50 dan 67.36% dibanding MobileNet-v1. Waktu yang dibutuhkan dalam proses pendektesian akan semakin lama jika objek yang berada pada gambar semakin banyak.

REFERENCES

[1] World Health Organization. More than 270 000 pedestrians killed on roads each year. [Online]. Available: https://www.who.int/news/item/02-05-2013-more-than-270-000-pedestrians-killed-on-roads-each-year

[2] The Global Road Safety Facility. Indonesia, indonesia's road safety country profile. [Online]. Available: https://www.roadsafetyfacility.org/country/indonesia

[3] A. Angelova, A. Krizhevsky, V. Vanhoucke, A. Ogale, and D. Ferguson, "Real-time pedestrian detection with deep network cascades," 2015.

[4] I. Hasan, S. Liao, J. Li, S. Akram, and L. Shao, "Pedestrian detection: The elephant in the room," 03 2020.

[5] C. Xu, G. Wang, S. Yan, J. Yu, B. Zhang, S. Dai, Y. Li, and L. Xu, "Fast vehicle and pedestrian detection using improved mask r-cnn," *Mathematical Problems in Engineering*, vol. 2020, 2020.

[6] M. A. Malbog, "Mask r-cnn for pedestrian crosswalk detection and instance segmentation," pp. 1–5, 2019.

[7] I. Matterport, "Mask r-cnn for object detection and segmentation," https://github.com/matterport/Mask_RCNN#mask-r-cnn-for-object-detection-and-segmentation, 2019, acessed : 2020-11-4.