Voice Alert Sebagai Alat Bantu Penglihatan di Lingkungan Rumah dan Jalanan Secara Umum Berbasis Android

Kevin Christian S, Liliana, Rolly Intan
Program Studi Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) – 8417658
E-Mail: kevinchristiansalim@gmail.com, lilian@petra.ac.id, rintan@petra.ac.id

ABSTRAK

Menurut penelitian yang dilakukan oleh *Governors Highway Safety Assocation* tahun 2017 menunjukan bahwa terdapat sekitar 6 ribu pejalan kaki terbunuh di Amerika karena kebiasaan penggunaan *smartphone* saat berjalan kaki. Penelitian lain dari Jeff Ronsen menunjukan bahwa otak mengalami kelebihan informasi dan tidak bisa berfungsi dengan baik ketika melakukan kedua aktivitas ini bersamaan. Berjalan sambil menggunakan *smartphone* membuat konsentrasi terhadap suasana jalan terpecah dan membuat pejalan kaki tidak berfokus pada jalan melainkan pada *smartphone*. Perilaku tersebut mengakibatkan resiko terjadi kecelakaan pada pejalan kaki meningkat. Salah satu solusi untuk mencegah terjadinya kecelakaan diatas adalah dengan memanfaatkan kamera *smartphone* untuk mengambil gambar yang berada didepan pengguna.

Kamera *smartphone* dapat digunakan untuk mengambil data *input* berupa gambar secara *real time* yang kemudian dilakukan proses deteksi objek dan mengeluarkan *alert* berupa suara yang menyebutkan nama objek yang terdeteksi kepada pengguna *smartphone*. Objek yang dideteksi adalah objek yang umumnya berada di lingkungan rumah dan jalanan seperti manusia, mobil, sepeda, sepeda motor, dan rambu stop. Objek deteksi menggunakan *SSD MobileNet* yang diterapkan *transfer learning* yang dilatih lebih lanjut dengan menggunakan *dataset google open image dataset v6*. Hasil dari *transfer learning* berupa *weight* yang digunakan untuk mendeteksi objek dari *input* kamera *android*.

Hasil pengujian menunjukan bahwa *SSD_MobileNet_V2* dengan *learning rate* 0.01 dan *steps 10.000* memiliki nilai *mAP* terbaik dengan 80% dalam mendeteksi objek. Aplikasi *SSD_MobileNet_V2* dapat mendeteksi objek dengan kecepatan *inference time* 80ms – 110ms secara *realtime* dalam *device* keadaan *standby*, dan *voice alert* dengan *instant* mengeluarkan *alert* ketika ada objek yang terdeteksi.

Kata Kunci: SSD, MobileNet, Voice Alert, Android, Deteksi Objek Real Time

ABSTRACT

According to research conducted by the Governors Highway Safety Assocation in 2017 showed that there are about 6 thousand pedestrians killed in America due to the habit of using smartphones while walking. Another study from Jeff Ronsen showed that the brain is overloaded and cannot function properly when performing these two activities at the same time. Walking while using a smartphone makes the concentration on the atmosphere of the road split and makes pedestrians not focus on the road but rather on the smartphone. Such behavior results in an increased risk of pedestrian accidents. One solution to prevent accidents above is to use the smartphone camera to take pictures in front of the user.

Smartphone cameras can be used to retrieve input data in the form of images in real time which is then carried out the process of object detection and issue alerts in the form of sounds that mention the name of the detected object to smartphone users. Detected objects are objects that are generally located in home and street environments such as humans, cars, bicycles, motorcycles, and stop signs. Object detection using SSD MobileNet applied transfer learning that is further trained by using google open image dataset v6 dataset. The result of transfer learning is weight used to detect objects from android camera input.

The test results showed that SSD_MobileNet_V2 with a learning rate of 0.01 and steps 10,000 has the best mAP value with 80% in detecting objects. The SSD_MobileNet_V2 can detect objects with an inference time speed of 80ms – 110ms in real time in a standby device, and voice alerts by instantly issuing alerts when an object is detected.

Keywords: *SSD, MobileNet, Voice Alert, Android, Real time object detection*

1. PENDAHULUAN

Banyak pengguna *smartphone* yang menggunakan *smartphone* disaat yang tidak tepat seperti sedang berjalan kaki. *Smartphone* sudah menjadi salah satu peralatan yang tidak bisa dilepaskan oleh banyak orang [1, 3, 5, 9, 14, 23]. *Smartphone android* merupakan perangkat keras dengan jumlah pengguna terbanyak [24, 27] dimana *android* memiliki *market share* sebesar 72.52% dibandingkan iOS yang hanya 26.8%. *Governors Highway Safety Association* [10] melakukan penelitian pada tahun 2017 menunjukan bahwa terdapat sekitar 6 ribu pejalan kaki terbunuh di Amerika Serikat dan menurut pihak polisi hal ini terkait dengan kebiasaan penggunaan *smartphone* saat berjalan kaki. Penelitian yang dilakukan oleh Jeff Ronsen [12] mengenai kinerja otak saat menggunakan *smartphone* dan berjalan kaki menunjukan bahwa 60% orang yang berjalan kaki sambil

menggunakan *smartphone* tidak bisa berjalan dengan lurus, otak mengalami kelebihan informasi dan tidak bisa berfungsi dengan baik saat melakukan aktivitas ini secara bersamaan. Berjalan kaki dengan menggunakan *smartphone* menyebabkan pengguna *smartphone* tidak fokus dan lebih rentan terkena kecelakaan. Berjalan sambil menggunakan *smartphone* membuat terpecahnya konsentrasi terhadap suasana jalan dan membuat pejalan kaki tidak berfokus pada jalan melainkan pada *smartphone* [6, 7, 19]. Perilaku tersebut mengakibatkan resiko terjadi kecelakaan pada pejalan kaki meningkat.

Salah satu solusi untuk mencegah terjadinya kecelakaan saat berjalan kaki sambil menggunakan *smartphone* adalah dengan memanfaatkan kamera *smartphone* untuk mengambil gambar yang berada didepannya. Gambar dapat dimanfaatkan untuk mendeteksi dan mengenali objek yang berada didepan *user*. Kamera *smartphone* dapat digunakan sebagai *input* gambar secara *real time* yang kemudian dilakukan proses deteksi objek dan mengeluarkan *alert* berupa suara yang menyebutkan nama objek yang terdeteksi.

Penelitian yang sudah dilakukan dalam mendeteksi dan pengenalan objek adalah "*The Open Image V4 Dataset"* [16] menggunakan *SSD* [17] dan *faster R-CNN* [21]. Hasil yang didapat menunjukan bahwa *SSD_MobileNet_V2* memiliki *inference time* yang lebih cepat (0.024 detik) dibandingkan dengan Faster-RCNN *with inception-ResNetV2* (0.454 detik).

Penelitian ini mengajukan sebuah sistem yang membantu *user* dengan menggunakan kamera untuk memperhatikan jalan dengan bantuan *voice alert* pada *smartphone android*. Objek yang dideteksi adalah objek yang sering ditemukan di lingkungan rumah dan jalanan seperti manusia, mobil, sepeda, sepeda motor, dan rambu stop. Objek deteksi menggunakan SSD MobileNet yang diterapkan *transfer learning* [29] dengan mempertahankan arsitektur dari model nya yang dilatih lebih lanjut dengan menggunakan *dataset google open image dataset v6*. Hasil dari *transfer learning* adalah *weight* yang digunakan untuk mendeteksi objek dari *input* kamera *android* [32]. Penelitian ini memanfaatkan *smartphone android* dimana kamera belakang sebagai *input* untuk mendeteksi objek yang kemudian dikeluarkan hasil dari pengenalan objek yang terdeteksi berbentuk *voice alert.*

2. TINJAUAN STUDI

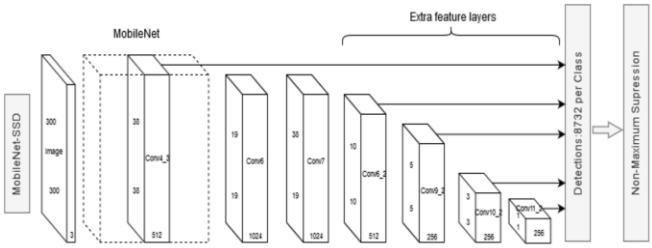
Pada penelitian yang dilakukan Kuznetsova et al.[16] dalam mengembangkan *open image dataset v4*, metode yang digunakan dalam penelitian ini adalah *Faster-RCNN* dengan *SSD_MobileNet_v2* dalam penelitian yang dilakukan dengan dataset *Open Image Dataset V4*. Hasil dari penelitian ini adalah performa dari 2 model terus meningkat jika dilatih lebih banyak data, hasil dari training data juga menunjukan bahwa models yang lebih ringan dapat dilatih dengan mudah oleh *open image dataset*. *Open image dataset* dapat mendorong penelitian kedepannya agar lebih baik. Hasil yang didapat menunjukan bahwa *SSD_MobileNet_V2* memiliki *inference time* yang lebih cepat (0.024 detik) dibandingkan dengan *Faster-RCNN with inception-ResNetV2* (0.454 detik). Penelitian ini menghasilkan program yang hanya bisa berjalan di komputer saja. Penelitian yang akan dilakukan akan mengembangkan objek deteksi dapat digunakan dalam *smartphone* secara *real time* yang dapat digunakan oleh *end user*.

Pada penelitian yang dilakukan Sutjiadi & Pattiasina[31] yaitu memberikan *alert* kepada *driver* ketika parkir menggunakan *dashboard camera* ketika ada mobil dekat. metode yang digunakan untuk objek deteksi dan proses *re-train* adalah *pre-trained quantized coco SSD MobileNet v1*. Hasil dari penelitian ini adalah sistem berhasil mendeteksi dan klarifikasi 3 jenis kendaraan bermotor yaitu mobil, bus, dan truck dengan tingkat akurasi yang baik. Selain itu sistem juga mampu memberikan peringatan secara visual dan alarm ketika kendaraan yang ada di depan sudah berada pada posisi yang cukup dekat. Perbedaan penelitian yang dilakukan dalam deteksi objek dengan menggunakan dashboard dengan penelitian ini adalah penelitian yang dilakukan disini menggunakan *dashboard camera* mobil untuk sistem peringatan lalu di kirimkan ke android untuk alert, sementara penelitian yang akan dilakukan tidak menggunakan tambahan *hardware* kamera dan langsung menggunakan kamera *smartphone android* dan mengeluarkan *voice alert* berupa suara.

Pada penelitian yang dilakukan Kang[13] yaitu mengembangkan objek deteksi dengan menggunakan metode baru yaitu *SSD* dengan *VGG16* untuk melakukan *real time object detection*. Hasilnya model yang digunakan dapat berjalan dengan kecepatan 42 FPS dalam *hardware XC7VX690T FPGA* dengan hasil mAP 78.13% dalam mendeteksi objek. Penelitian ini menghasilkan program yang dapat dijalankan dikomputer dengan kekuatan komputasi yang tinggi. Penelitian akan menggunakan *SSD MobileNet* dengan *feature extraction* yang lebih ringan sehingga dapat berjalan dalam *hardware* dengan *low computing power* seperti *smartphone*.

3. METODOLOGI

3.1 SSD MobileNet

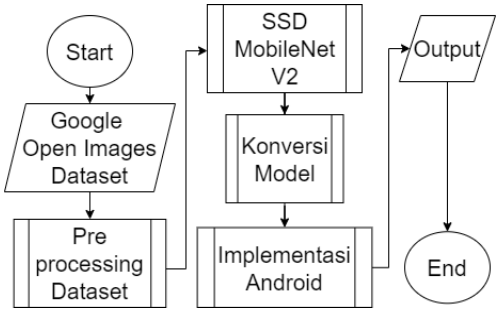


Gambar 1. Arsitektur Model SSD dengan MobileNet

SSD-MobileNet [2, 4, 8, 25] adalah *model* yang dibangun dengan menggunakan *MobileNet* untuk melakukan *feature extraction* dan *SSD* sebagai *classifier*. Keunggulan *SSD* [15, 18, 28] adalah memiliki kecepatan deteksi dan akurasi yang baik karena menggunakan *single deep neural network* [20]. *Single deep neural network* memiliki enam *feature layer* dengan jumlah *feature map* yang semakin sedikit setiap *layer* nya. Sedangkan *MobileNet* merupakan *model* yang dirancang khusus untuk berjalan di *smartphone* dengan struktur *model* yang lebih sederhana. *MobileNet* merubah *standard convolution gilter* menjadi *depth-wise* dan 1x1 *point-wise convolution*. *Point-wise convolution* digunakan untuk menggabungkan *output* dari *depthwise convolution* yang menggunakan sebuah *filter* dari setiap *input*, sehingga mengurangi ukuran dari model dan proses komputasi. *MobileNet* menggunakan 3x3 *depthwise convolutions* membuat proses komputasi berkurang menjadi 1/8 – 1/9 dari

convolution lainnya [11]. Penggabungan *SSD* dan *MobileNet* akan menghasilkan arsitektur yang lebih ringan karena dengan mengganti *network* menggunakan *MobileNet* dapat mengurangi parameter *feature extraction* dari *SSD* sehingga meningkatkan kecepatan deteksi dari *model* yang dibuat. Gambar 1 merupakan arsitektur dari *SSD_MobileNet*.

3.2 Desain Sistem



Gambar 2. Blok Diagram Sistem

Pre-processing

Proses pertama dari Gambar 2 adalah *pre processing dataset*. *Dataset* diambil dari *google open image dataset v6*. *Dataset* berisi 5 *class* yaitu manusia, mobil, sepeda, sepeda motor, rambu stop. Setelah itu data di resize menjadi ukuran 300x300.

Training menggunakan SSD_MobileNet_V2

Pada proses ini dilakukan *transfer learning* dengan cara menggunakan *pre-trained SSD_MobileNet_V2 model* yang sudah dilatih sebelumnya dan mempertahankan arsitektur model untuk mendapatkan *weight* baru dari *dataset* yang digunakan.

Konversi Model Android

Pada Proses konversi *weight* yang sudah disimpan agar dapat berjalan didalam *android* menggunakan *tensorflow lite*.

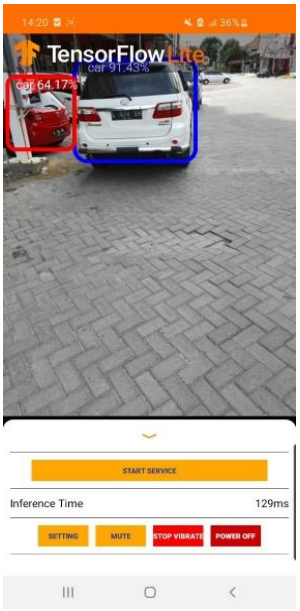
Implementasi Android

Pada proses ini kamera digunakan sebagai *input* gambar secara *real time*. *Input* gambar akan dideteksi menggunakan *model.tflite* dan hasil deteksi objek adalah *result* berisi *class*, *location*, dan *score* dari objek yang dideteksi. *Result* akan digunakan untuk mengeluarkan *output* berupa *bounding box* lokasi objek dan *voice alert* yang mengeluarkan suara dari nama *class* yang dideteksi. Proses ini berjalan secara terus menerus sampai sistem ditutup.

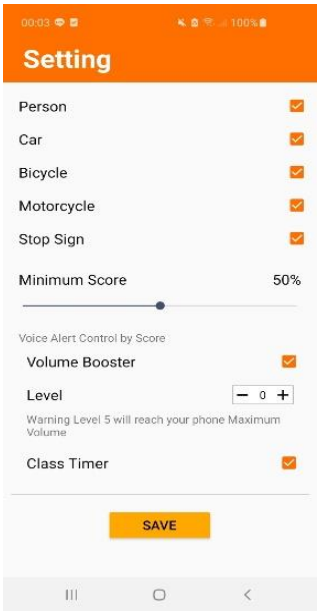
Foreground Service

Foreground service [30] merupakan *activity* yang berjalan sebagai *background service* dalam *smartphone*. *Foreground service* digunakan agar sistem yang sudah dibuat dapat berjalan sebagai *service application* atau *background apps*. *Foreground service* merupakan *activity* yang dibuat terpisah dengan *activity* biasa, sehingga dilakukan kembali beberapa proses seperti pada *Implementasi Android* seperti membuka kamera sampai dengan proses mengeluarkan result dari hasil deteksi. Aplikasi yang menggunakan *foreground service* ditandai dengan adanya notifikasi yang diketahui *user*. *Foreground service* berhenti berjalan ketika aplikasi kembali dibuka.

3.3 *User Interface*



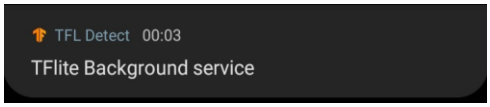
Gambar 3. Halaman Aplikasi



Gambar 4. Halaman Setting

Gambar 3 merupakan tampilan awal dari aplikasi pada saat dibuka. Aplikasi langsung menampilkan hasil input dari camera, dan secara *real time* melakukan deteksi objek yang berada di depan device camera. Objek yang dideteksi ditandai dengan kotak *result* yang berisi *class*, *location* dan *score* dari objek tersebut. *Button start service* merupakan *button* yang digunakan untuk memulai *foreground service*. *Inference time* merupakan kecepatan dari *device* dalam mendeteksi objek setiap *frame* nya dalam *milli second*. *Button setting* digunakan untuk membuka halaman *setting activity* dapat dilihat dalam Gambar 4 *setting activity*. *Button mute / unmute* digunakan untuk mengaktivasi *voice alert*. *Button start / stop vibrate* digunakan untuk mengaktivasi getaran dari *device* ketika mendeteksi objek. *Vibrate* digunakan pada saat berada ditempat yang ramai karena kemungkinan *voice alert* pada saat itu tidak terdengar.

Gambar 4 merupakan tampilan dari *setting Activity*. *Setting activity* mempunyai menu *class selector, minimum score, volume booster* dan *class timer*. *Class selector* digunakan untuk memilih *class* yang ingin dideteksi pada lingkungan tertentu. *Minimum score* berbentuk *slider* yang digunakan untuk mengatur minimal *score* dari objek yang dideteksi. *Minimal score* digunakan untuk mengurangi bias deteksi contohnya adalah bayangan yang dideteksi sebagai manusia. Pada penelitian ini penentuan *minimum score* 56% adalah yang terbaik untuk mengatasi masalah tersebut. Untuk deteksi *class* lainnya digunakan minimum *score* 50%. *Volume booster* digunakan untuk menambah *volume* dari *voice alert* yang dihasilkan oleh aplikasi. Aplikasi memperkeras *volume* seiring dengan meningkatnya *confidence score* dari objek yang dideteksi. *Class timer* digunakan untuk memberikan jeda dalam mengeluarkan *voice alert*. Jeda diberikan agar *voice alert* tidak keluar secara terus menerus karena proses deteksi berjalan secara terus menerus. Jeda akan semakin pendek ketika objek yang sama terdeteksi pada saat berikutnya.



Gambar 4. Tampilan Notifikasi dari *Foreground Service*

Gambar 4 merupakan tampilan dari notifikasi yang ditampilkan kepada *user* saat *foreground service* sedang berjalan. Notifikasi bertujuan agar *user* mengetahui jika aplikasi sedang berjalan di *service*. Notifikasi dapat dapat digunakan untuk menutup *service* dan langsung membuka aplikasi kembali seperti Gambar 2 Halaman Awal Aplikasi.

4. PENGUJIAN

Pada bagian ini membahas tentang pengujian tingkat akurasi dengan menggunakan perhitungan *confusion matrix* [22, 26] terhadap *SSD_MobileNet_V2*. Model dengan nilai mAP tertinggi akan digunakan untuk mendeteksi objek dalam *android*. Nilai mAP digunakan untuk mengurangi *false positive* dari hasil deteksi, sehingga *model* tidak memberikan *alert* yang salah kepada *user*. Pengujian dilanjutkan dengan menguji kecepatan dari aplikasi untuk mendeteksi objek. Kemudian menguji kecepatan dari *respond alert* ketika mendeteksi objek.

4.1 Pengujian SSD dengan *Validation Dataset*

Pengujian *model SSD_MobileNet_V2* untuk mendeteksi gambar menggunakan *google open image dataset v6 validation dataset* berjumlah 100 gambar setiap *class* nya.

Tabel 1. Pengujian SSD terhadap *Validation Dataset*

LR	Steps	mAP	mAR	Accuracy	F1score
0.01	10000	**0.800**	0.607	0.840	0.690
	12000	0.768	0.616	0.837	0.684
	14000	0.798	0.593	0.832	0.681
0.001	10000	0.759	**0.651**	**0.846**	**0.701**
	12000	0.767	0.635	0.842	0.695
	14000	0.757	0.647	0.844	0.697
0.0001	10000	0.651	0.631	0.810	0.641
	12000	0.672	0.633	0.817	0.652
	14000	0.682	0.639	0.821	0.660

Tabel 1 Pengujian *SSD_MobileNet_V2* terhadap *validation dataset*. Pengujian ini digunakan untuk mencari konfigurasi terbaik dari *model* yang sudah dibuat. Hasil dari pengujian tersebut *model* dengan *learning rate* dengan 0.01 dengan *steps* 10000 menghasilkan *mAP* yang paling tinggi yaitu 0.8. Berdasarkan pengujian yang sudah dilakukan *model* yang digunakan dalam aplikasi *android* adalah *SSD_MobileNet_V2* dengan *learning rate* 0.01 dengan 10000 *steps* karena memiliki nilai mAP tertinggi.

4.2 Pengujian SSD dalam Android

Pengujian *model SSD_MobileNet_V2* yang sudah dikonversi menjadi *model* yang dapat berjalan dalam *android*. Pengujian *model SSD_MobileNet_V2* dilakukan menggunakan 10 gambar setiap *class* nya yang diletakan didepan *smartphone camera*.

Tabel 2. Pengujian SSD digunakan dalam *android*

mAP	mAR	F1 score	Average Accuracy	Score Rata Rata Tp	Score Rata rata Total
97%	96%	96,3%	98,5%	85,2%	82,6%

Tabel 2 Pengujian *SSD_MobileNet_V2* untuk mendeteksi benda dengan menggunakan *android* menghasilkan *mAP* hingga 97% dengan *score* rata rata yang dikatakan benar adalah 85,8%. Sementara rata rata dari total semua deteksi dari yang benar sampai dengan tidak terdeteksi adalah 82,6% untuk total 50 gambar.

4.3 Pengujian Kecepatan Inference Time

Pengujian dilakukan menggunakan *device* yang berbeda dengan melihat *inference time* atau kecepatan *device* dalam mendeteksi objek setiap *frame*.

Tabel 3. *Inference device* melakukan *realtime detection*

Device	Inference Time (ms)	Frekuensi Angka paling sering (ms)	Average (200 Inference pertama)
Samsung S6	150 - 300	170 - 240	201ms
Redmi Note 7	100 - 185	110 - 150	122ms
Samsung S10	50 - 120	80 - 110	83ms
Samsung A9	80 - 140	100 - 130	118ms

Hasil Tabel 3 dapat dilihat bahwa *SSD_MobileNet_V2* dapat berjalan dengan kecepatan dibawah 1 detik setiap *inference time*. *Device* ke 3 mempunya kecepatan *inference time* tercepat dengan rata rata *inference time* 83ms dalam 200 *inference* pertama.

4.4 *Response Alert* saat *Open Apps*

Pengujian dilakukan dengan memperhatikan kecepatan *voice alert* & *vibrate* muncul ketika mendeteksi objek saat *open apps* untuk melihat bagaimana kecepatan *response alert* ketika aplikasi mendeteksi *class*. Pengujian menggunakan Output:

- *Instant*: ketika alert keluar saat itu juga tanpa proses komputasi.
- *Delay*: ketika Alert keluar beberapa detik karena melakukan komputasi.

Tabel 4. Pengujian Kecepatan Alert keluar

Device	*Voice Alert*	*Vibrate*
1	*Instant*	*Instant*
2	*Instant*	*Instant*
3	*Instant*	*Instant*
4	*Instant*	*Instant*

Hasil dari Tabel 4 adalah semua *device* langsung mengeluarkan *voice* / *vibrate* dengan *instant* setelah aplikasi mendeteksi objek. Pengujian ini menunjukan bahwa sistem akan secara langsung memberikan *voice alert* kepada *user* ketika sistem mendeteksi adanya objek yang terlihat dari kamera.

4.5 *Response Alert* saat *Foreground Service*

Pengujian *foreground service* dilakukan dengan kondisi *device standby* dan *device* sedang menjalankan aplikasi lainnya bersamaan seperti aplikasi video *youtube*, *social media line*, *game mobile legend*. Pengujian dilakukan dengan mendeteksi objek yang diletakan didepan *device* agar aplikasi mengeluarkan *result* hasil deteksi objek secara terus menerus, agar terlihat pengaruh dari *service* terhadap *device*. *Output* dari pengujian *Voice Alert* ini adalah:

Aplikasi:
- Stabil: berjalan normal seperti tidak ada *foreground service*.
- Melambat: mengalami penurunan peforma, tetapi fitur utama aplikasi tersebut berjalan dengan baik.
- Sangat Lambat: mengalami peforma yang kurang pada semua bagian.
- *Device* Tidak kuat: Aplikasi tidak mampu terbuka.

Alert dibandingkan dengan *open apps*:
- Stabil: *Alert* berjalan normal seperti membuka *open app* dan masih dibawah 1 detik.
- Melambat: *Alert* mengalami perubahan lebih dari 1 detik.
- Sangat Lambat: *Alert* mengalami perubahan lebih dari 3 detik.
- *Device* Tidak kuat: *Service* tidak mengeluarkan *alert*.

Tabel 5. Inferensi Device dalam *Foreground Service*

Device	*Device* Standby	Video		Social Media		Game	
		Aplikasi	Alert	Aplikasi	Alert	Aplikasi	Alert
1	Stabil	Stabil	Stabil	Stabil	Stabil	Sangat Lambat	Melambat
2	Stabil	Stabil	Stabil	Stabil	Stabil	Sangat Lambat	Stabil
3	Stabil	Stabil	Stabil	Stabil	Stabil	Melambat	Stabil
4	Stabil	Stabil	Stabil	Stabil	Stabil	Sangat Lambat	Melambat

Tabel 5 menunjukan bahwa *service* dapat digunakan dengan baik pada aplikasi *video* dan *social media* tanpa mengganggu peforma dari *device* maupun *aplikasi*. *Service* mulai mengganggu jalannya *device* ketika digunakan dalam aplikasi berat. Aplikasi yang berjalan pada *device* 3 melambat dan *device* 1, 2, dan 4 menjadi sangat lambat. *Service* yang berjalan juga mengalami penurunan kecepatan pada *Device* 1 dan 4 melambat dalam memberikan *alert*, sementara *service* pada *device 2 dan 3* masih dapat mengeluarkan *alert* dengan stabil. Tidak ada *device* yang memberikan hasil *device* tidak kuat pada aplikasi dan *service* yang berjalan bersamaan.

5. KESIMPULAN

Berdasarkan pengamatan dan pengujian pada sistem dapat disimpulkan beberapa hal sebagai berikut:

- *Pre-trained SSD_MobileNet_V2* yang dilakukan *transfer learning* dengan *learning rate* 0.01 dan *steps* 10000 memiliki mAP sebesar 80% dan merupakan mAP tertinggi yang didapatkan dengan menggunakan perhitungan *confusion matrix* dibandingkan dengan *learning rate* dan *steps* yang lainnya.
- *SSD_MobileNet_V2* dapat berjalan dengan sangat cepat dalam *realtime* di *device* dengan *inference time* tercepat 50ms – 120ms. Dengan frekuensi *inference time* angka terbanyak 80ms - 110ms dan *average inference time* dalam 200 *frame* pertama adalah 83ms saat *device standby*.
- Berdasarkan hasil pengujian sistem yang dilakukan *foreground service* dalam keadaan menggunakan aplikasi berat seperti Game, service mengganggu performa dari aplikasi yang sedang berjalan.
- Berdasarkan pengujian aplikasi yang dilakukan, aplikasi ini masih belum bisa menjawab permasalahan yang ada dalam latar belakang permasalahan. Aplikasi terlalu beresiko untuk digunakan karena arah dari smartphone dengan posisi dibawah -30 derajat secara *vertical* membuat kamera menghadap kebawah sehingga tidak dapat mendeteksi kendaraan yang berada didepan, kecepatan dari kendaraan yang tinggi membuat aplikasi berbahaya untuk digunakan.

Dengan kesimpulan yang dilakukan diatas, ada beberapa hal yang dijadikan sebagai saran dalam penelitian selanjutnya antara lain:

- Aplikasi yang dikembangkan lebih cocok di implementasikan untuk menghitung kendaraan yang lewat sesuai dengan jenis kendaraan yang lewat seperti sepeda, sepeda motor, mobil, truck, bus termasuk pejalan kaki yang lewat dengan menggunakan *object tracking*.
- Menambahkan class untuk dideteksi terhadap objek yang lebih sering ditabrak oleh user contohnya batu, cone, rambu jalan bolong, rambu jalan basah, bolongan di jalan, tembok, dan pintu kaca.

6. DAFTAR PUSTAKA

[1] Ambeth Kumar, V.D. 2018. Precautionary measures for accidents due to mobile phone using IOT. *Clinical eHealth*. 1, 1 (2018), 30–35. DOI:doi.org/10.1016/j.ceh.2018.12.001.

[2] Biswas, D., Su, H., Wang, C., Stevanovic, A. and Wang, W. 2019. An automatic traffic density estimation using Single Shot Detection (SSD)and MobileNet-SSD. *Physics and Chemistry of the Earth*. 110, (2019), 176–184. DOI:doi.org/10.1016/j.pce.2018.12.001.

[3] Botzer, A., Musicant, O. and Perry, A. 2017. Driver behavior with a smartphone collision warning application – A field study. *Safety Science*. 91, (2017), 361–372.

DOI:doi.org/10.1016/j.ssci.2016.09.003.

[4] Cao, M.T., Tran, Q.V., Nguyen, N.M. and Chang, K.T. 2020. Survey on performance of deep learning models for detecting road damages using multiple dashcam image resources. *Advanced Engineering Informatics*. 46, April (2020), 101182. DOI:doi.org/10.1016/j.aei.2020.101182.

[5] Chen, P.L. and Pai, C.W. 2018. Pedestrian smartphone overuse and inattentional blindness: An observational study in Taipei, Taiwan. *BMC Public Health*. 18, 1 (2018), 1–10. DOI:doi.org/10.1186/s12889-018-6163-5.

[6] Crowley, P., Madeleine, P. and Vuillerme, N. 2019. The effects of mobile phone use on walking: A dual task study. *BMC Research Notes*. 12, 1 (2019), 1–6. DOI:doi.org/10.1186/s13104-019-4391-0.

[7] Feld, J.A. and Plummer, P. 2019. Visual scanning behavior during distracted walking in healthy young adults. *Gait and Posture*. 67, August 2018 (2019), 219–223. DOI:doi.org/10.1016/j.gaitpost.2018.10.017.

[8] Gievska, S. and Madjarov, G. 2019. *Detection of Toy Soldiers Taken from a Bird's Perspective Using Convolutional Neural Networks*.

[9] Hamazima, M., Murayama, T., Yamasaki, H., Nakano, T. and Yamada, M. 2020. Study on Simultaneous-Action Discrimination Method Using Deep Learning. *International Journal of Intelligent Transportation Systems Research*. (2020). DOI:doi.org/10.1007/s13177-019-00216-y.

[10] Hedlund, J. 2017. Pedestrian Traffic Fatalities by State. *Governors Highway Safety Assocation*. 1, (2017).

[11] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H. 2017. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv*. (2017).

[12] Jeff Rossen 2017. *Rossen to the Rescue: Secrets to Avoiding Scams, Everyday Dangers, and Major Catastrophes*. Flatiron Books.

[13] Kang, H.-J. 2020. Real-Time Object Detection on 640x480 Image With VGG16+SSD. (2020), 419–422. DOI:doi.org/10.1109/icfpt47387.2019.00082.

[14] Kujala, T. and Mäkelä, J. 2018. Naturalistic study on the usage of smartphone applications among Finnish drivers. *Accident Analysis and Prevention*. 115, November 2017 (2018), 53–61. DOI:doi.org/10.1016/j.aap.2018.03.011.

[15] Kumar, A. and Srivastava, S. 2020. Object Detection System Based on Convolution Neural Networks Using Single Shot Multi-Box Detector. *Procedia Computer Science*. 171, 2019 (2020), 2610–2617. DOI:doi.org/10.1016/j.procs.2020.04.283.

[16] Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T. and Ferrari, V. 2020. The Open Images Dataset V4: Unified Image Classification, Object Detection, and Visual Relationship Detection at Scale. *International Journal of Computer Vision*. 128, 7 (2020), 1956–1981. DOI:doi.org/10.1007/s11263-020-01316-z.

[17] Li, J., Hou, Q., Xing, J. and Ju, J. 2020. SSD Object Detection Model Based on Multi-Frequency Feature Theory. *IEEE Access*. 8, (2020), 82294–82305. DOI:doi.org/10.1109/ACCESS.2020.2990477.

[18] Li, Y., Dong, H., Li, H., Zhang, X., Zhang, B. and Xiao, Z. 2020. Multi-block SSD based on small object detection for UAV railway scene surveillance. *Chinese Journal of Aeronautics*. 33, 6 (2020), 1747–1755. DOI:doi.org/10.1016/j.cja.2020.02.024.

[19] Lin, M.I.B. and Huang, Y.P. 2017. The impact of walking while using a smartphone on pedestrians' awareness of roadside events. *Accident Analysis and Prevention*. 101, (2017), 87–96. DOI:doi.org/10.1016/j.aap.2017.02.005.

[20] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. and Berg, A.C. 2016. SSD: Single Shot MultiBox Detector Wei. *ECCV*. 1, (2016), 398–413. DOI:doi.org/10.1007/978-3-319-46448-0_2.

[21] Liu, Y. 2018. An Improved Faster R-CNN for Object Detection. *Proceedings - 2018 11th International Symposium on Computational Intelligence and Design, ISCID 2018*. 2, (2018), 119–123. DOI:doi.org/10.1109/ISCID.2018.10128.

[22] Luque, A., Carrasco, A., Martín, A. and de las Heras, A. 2019. The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*. 91, (2019), 216–231. DOI:doi.org/10.1016/j.patcog.2019.02.023.

[23] Mourra, G.N., Sénécal, S., Fredette, M., Lepore, F., Faubert, J., Bellavance, F., Cameron, A.F., Labonté-LeMoyne, É. and Léger, P.M. 2020. Using a smartphone while walking: The cost of smartphone-addiction proneness. *Addictive Behaviors*. 106, August 2019 (2020), 106346. DOI:doi.org/10.1016/j.addbeh.2020.106346.

[24] Novac, O.C., Novac, M., Gordan, C., Berczes, T. and Bujdoso, G. 2017. Comparative study of Google Android, Apple iOS and Microsoft Windows Phone mobile operating systems. *2017 14th International Conference on Engineering of Modern Electric Systems, EMES 2017*. (2017), 154–159. DOI:doi.org/10.1109/EMES.2017.7980403.

[25] Pinto de Aguiar, A.S., Neves dos Santos, F.B., Feliz dos Santos, L.C., de Jesus Filipe, V.M. and Miranda de Sousa, A.J. 2020. Vineyard trunk detection using deep learning – An experimental device benchmark. *Computers and Electronics in Agriculture*. 175, May (2020), 105535. DOI:doi.org/10.1016/j.compag.2020.105535.

[26] Sai Srinath, N.G.S., Joseph, A.Z., Umamaheswaran, S., Priyanka, C.L., Malavika Nair, M. and Sankaran, P. 2020. NITCAD - Developing an object detection, classification and stereo vision dataset for autonomous navigation in Indian roads. *Procedia Computer Science*. 171, 2019 (2020), 207–216. DOI:doi.org/10.1016/j.procs.2020.04.022.

[27] Sarkar, A., Goyal, A., Hicks, D., Sarkar, D. and Hazra, S.

2019. Android Application Development: A Brief Overview of Android Platforms and Evolution of Security Systems. *Proceedings of the 3rd International Conference on I-SMAC IoT in Social, Mobile, Analytics and Cloud, I-SMAC 2019*. (2019), 73–79. DOI:doi.org/10.1109/I-SMAC47947.2019.9032440.

[28] Shakeel, M.F., Bajwa, N.A., Anwaar, A.M., Sohail, A., Khan, A. and Haroon-ur-Rashid 2019. *Detecting Driver Drowsiness in Real Time Through Deep Learning Based Object Detection*. Springer International Publishing.

[29] Sheng, T.J., Islam, M.S., Misran, N., Baharuddin, M.H., Arshad, H., Islam, M.R., Chowdhury, M.E.H., Rmili, H. and Islam, M.T. 2020. An Internet of Things Based Smart Waste Management System Using LoRa and Tensorflow Deep Learning Model. *IEEE Access*. 8, (2020), 148793–148811. DOI:doi.org/10.1109/ACCESS.2020.3016255.

[30] Späth, P. 2018. Services. *Pro Android With Kotlin*. Apress, Berkeley, CA. 27–42.

[31] Sutjiadi, R. and Pattiasina, T.J. 2020. Deteksi Objek Menggunakan Dashboard Camera Untuk Sistem Peringatan Pencegah Kecelakaan Pada Mobil. 7, 2 (2020), 427–434. DOI:doi.org/10.25126/jtiik.202072520.

[32] Xu, M., Lin, F.X., Liu, J., Liu, Y., Liu, Y. and Liu, X. 2019. A first look at deep learning apps on smartphones. *The Web Conference 2019 - Proceedings of the World Wide Web Conference, WWW 2019*. 2018, (2019), 2125–2136. DOI:doi.org/10.1145/3308558.3313591.