Jurnal Teknik Informatika (JUTIF)
Vol. 2, No. 1, Juni 2021, hlm. 57-66

DOI: https://doi.org/10.20884/1.jutif.2021.2.2.77
p-ISSN: 2723-3863
e-ISSN: 2723-3871

PEOPLE COUNTING FOR PUBLIC TRANSPORTATIONS USING YOU ONLY LOOK ONCE METHOD

Tsabita Al Asshifa Hadi Kusuma*[1], Korediantо Usman[2], Sofia Saidah[3]

[1,2,3]Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom, Indonesia
[1]tsasabita@gmail.com, [2]korediantousman@telkomuniversity.ac.id, [3]sofiasaidahsfi@telkomuniversity.ac.id

(Naskah masuk: 26 Januari 2021, diterima untuk diterbitkan: 02 Februari 2021)

Abstract

People counting have been widely used in life, including public transportations such as train, airplane, and others. Service operators usually count the amount of passengers manually using a hand counter. Nowadays, in an era that most of human-things are digital, this method is certainly consuming enough time and energy. Therefore, this research is proposed so the service operator doesn't have to count manually with a hand counter, but using an image processing with You Only Look Once (YOLO) method. This project is expected that people counting is no longer done manually, but already based on computer vision. This Final Project uses YOLOv4 that is the latest method in detecting untill 80 classes of object. Then it will use transfer learning as well to change the number of classes to 1 class. This research was done by using Python programming language with various platforms. This research also used three training data scenarios and two testing data scenarios. Parameters measured are accuration, precision, recall, F1 score, Intersection of Union (IoU), and mean Average Precision (mAP). The best configurations used are learning rate 0.001, random value 0, and sub divisions 32. And the best accuration for this system is 69% with the datasets that has been trained before. The pre-trained weights have 72.68% of accuracy, 77% precision, and 62.88% average IoU. This research has resulted a proper performance for detecting and counting people on public transportations.

Keywords: *CNN, deep learning, IoU, mAP, people counting, YOLOv4*

PEOPLE COUNTING UNTUK TRANSPORTASI PUBLIK MENGGUNAKAN METODE *YOU ONLY LOOK ONCE*

Abstrak

People counting cukup banyak digunakan dalam kehidupan sehari-hari, diantaranya pada transportasi publik seperti kereta api, pesawat udara, dan transportasi umum lainnya. Umumnya petugas akan menghitung jumlah penumpang secara manual menggunakan *hand counter*. Cara tersebut tentunya cukup memakan waktu dan tenaga di era yang sudah serba *digital* seperti saat ini. Maka dari itu, penelitian ini disusun dengan tujuan dapat menghitung jumlah orang dengan pengolahan citra atau *image processing* menggunakan metode *You Only Look Once* (YOLO). Dengan ini, perhitungan jumlah orang tidak lagi dilakukan secara manual, tetapi sudah berbasis *computer vision*. Penelitian ini menggunakan metode YOLOv4 yang merupakan metode paling mutakhir dalam mendeteksi hingga 80 kelas dari berbagai objek. Kemudian dilanjutkan dengan melakukan *transfer learning* menjadi 1 *class*. Penelitian dirancang menggunakan bahasa pemrograman Python dengan *platform* Google Colab, Visual Studio Code serta Windows PowerShell. Selain itu, penelitian ini terbagi atas tiga skenario *training* data dan dua skenario pengujian. Parameter yang dianalisis pada penelitian ini diantaranya yaitu akurasi, presisi, *recall*, F1 *score*, IoU dan mAP. Selain itu, konfigurasi terbaik yang didapat pada penelitian ini adalah *learning rate* sebesar 0.001, *random value* bernilai 0, hingga *sub divisions* 32. Akurasi yang didapatkan untuk menghitung jumlah orang adalah sebesar 69% dengan menggunakan *pre-trained weights* yang telah dilatih sendiri. *Pre-trained weights* tersebut memiliki nilai mAP sebesar 72.68% dengan presisi 77% dan *average* IoU 62.88%. Penelitian ini menghasilkan performa yang cukup baik untuk mendeteksi maupun menghitung orang pada transportasi publik.

Kata kunci: *CNN, deep learning, IoU, mAP, people counting, YOLOv4*

1. PENDAHULUAN

Seiring dengan majunya zaman, teknologi digital sudah sangat berkembang hingga menjadi salah satu kebutuhan penting manusia saat ini. Dengan beragam variasi dan kecanggihan dari teknologi tersebut, sistem perhitungan jumlah orang pun turut dikembangkan.

Penerapan *people counting* juga berlaku pada transportasi publik. *People counting* dibutuhkan untuk mencocokkan data jumlah penumpang ataupun sebagai parameter untuk pembatasan jumlah penumpang dalam transportasi tersebut.

Transportasi publik yang cukup membutuhkan *people counting* diantaranya adalah pesawat udara dan kereta api antar kota ataupun antar provinsi. Hal ini dibutuhkan untuk menyesuaikan jumlah penumpang yang terdaftar di *database* dan jumlah penumpang yang telah berada di dalam pesawat atau kereta api. Terlebih kedua transportasi tersebut memiliki biaya dengan jumlah yang tidak sedikit.

Hanya saja, sebagian besar perusahaan transportasi di Indonesia saat ini masih menggunakan *hand counter* untuk menghitung jumlah penumpang. Untuk itu, dibutuhkan pengembangan lebih lanjut tentang *people counting* menggunakan *image processing*.

Sistem perhitungan jumlah orang sudah pernah diteliti oleh beberapa peneliti pendahulu. Salah satunya yaitu pada penelitian berjudul "*Design of people counting system using Matlab*" [1]. Penelitian ini menggunakan algoritma Viola Jones dalam menghitung jumlah orang. Metode ini menghasilkan akurasi tertinggi sebesar 89% dengan data uji menggunakan format video. Kelemahan dari penelitian ini, nilai *sensitivity* akan turun ketika mendeteksi orang yang berbeda menggunakan warna baju yang mirip.

Selain itu, metode *Convolutional Neural Network* (CNN) [2] juga telah diteliti dan diterapkan untuk deteksi objek.

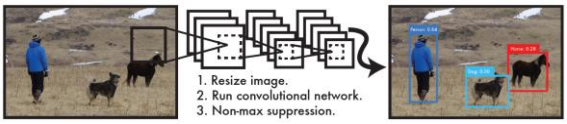
Setelah itu, metode CNN pun terus berkembang menjadi *Region*-CNN (R-CNN) [3], *Fast* R-CNN, *Faster* R-CNN [4], hingga *You Only Look Once* (YOLO) [5]. Pada R-CNN, algoritma *selective search* digunakan untuk mendeteksi objek [4]. Sistem harus melakukan konvolusi untuk setiap *region* yang jumlahnya dapat mencapai ribuan. Hal tersebut tentunya membutuhkan waktu yang cukup lama dan menjadikan R-CNN sebagai metode yang lebih akurat, namun memiliki waktu komputasi yang lambat [4]. *Fast* R-CNN pun diusulkan untuk mengatasi masalah tersebut. Setelah melalui algoritma *selective search*, sistem hanya membutuhkan sekali konvolusi untuk mendeteksi objek. Kemudian metode tersebut dikembangkan lagi menjadi *Faster* R-CNN yang tidak lagi menggunakan algortima *selective search*, melainkan *Region Proposal Network* (RPN) [6]. Namun, dari sekian metode tersebut masih belum dapat mendeteksi objek secara *real-time* dengan optimal [4].

Maka dari itu, pada penelitian ini diusulkan metode *You Only Look Once* (YOLO) yang memiliki performansi yang lebih tinggi dari metode sebelumnya. YOLO hanya melalui *convolutional neural network* dalam sekali saja untuk mendeteksi objek [5]. Hal ini tentunya menjadikan YOLO sebagai metode paling optimal dalam mendeteksi sekaligus menghitung objek secara *real-time* ataupun tidak *real-time*.

2. METODE PENELITIAN

2.1. YOLOv4

YOLO (*You Only Look Once*) adalah salah satu metode deteksi objek menggunakan *single convolutional neural network* yang memprediksikan *bounding box* serta *probability class* secara langsung dalam sekali evaluasi [5].



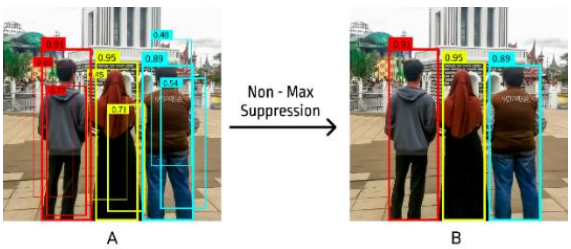
Gambar 1. Sistem Deteksi YOLOv4 [4]

Dari Gambar 1., setelah mendapatkan citra *input*, sistem melakukan *resize* terhadap citra menjadi 416 x 416 yang kemudian diproses dengan *single convolutional neural network* [4]. Setelah itu, dilakukan *non-max suppression* [7] untuk menghasilkan *bounding box* yang menentukan kelas dari tiap objeknya.

YOLOv4 [8] merupakan pengembangan dari versi sebelumnya, yaitu YOLOv3 [9]. Yang membedakan YOLOv3 dari versi-versi sebelumnya adalah arsitektur darknet yang digunakannya, yaitu Darknet-53. Arsitektur ini berperan sebagai fitur pengekstraksi citra *input* yang terdiri atas 53 *convolutional layers* sebagai *feature extraction* dan 53 *layers* lain yang berperan sebagai pendeteksi objek [9]. YOLOv4 juga masih menggunakan Darknet-53, tepatnya CSPDarknet-53 atau dapat disebut sebagai CSPNet, yaitu *backbone* baru yang dapat meningkatkan *learning capability* dari CNN [10].

2.2. *Non-Max Suppression*

Dalam algoritma deteksi objek, ada kemungkinan terdapat lebih dari satu *bounding box* mendeteksi objek yang sama. Maka dari itu, diperlukan *Non-Max Suppression* yang memiliki peran penting dalam memilih *bounding box* dengan nilai *confidence score* yang lebih tinggi [7].



Gambar 2. *Non-Max Suppression*

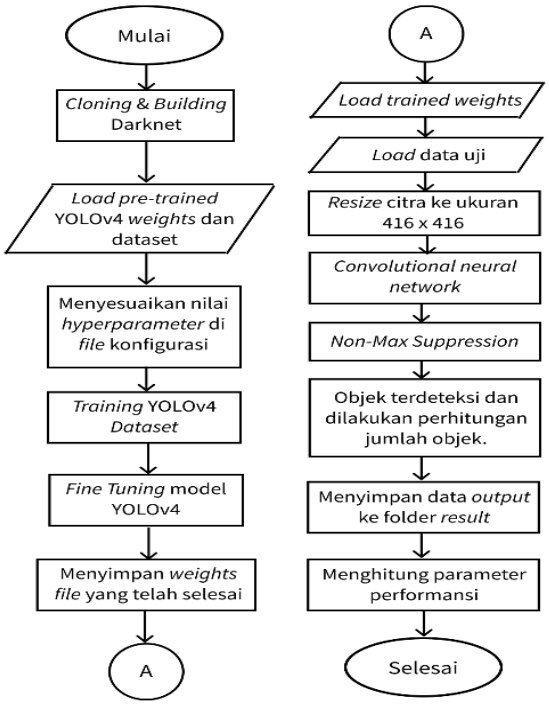
Seperti yang terlihat pada Gambar 2., terdapat tiga *bounding box* yang mendeteksi objek yang sama, namun dengan nilai *confident score* yang berbeda-beda. Pada *Non-Max Suppression*, sistem pada

awalnya menyeleksi nilai *confident score* yang tertinggi di antara tiga *bounding box* tersebut sebelum kemudian menghapus *bounding box* lain yang memiliki nilai *confidence score* yang lebih rendah [11].

3. ALUR KERJA PENELITIAN

3.1. Algoritma *People Counting*

Dari Gambar 3., proses (a) merupakan proses *training* terhadap data latih dan data validasi yang telah dikumpulkan.



Gambar 3. Alur Kerja Penelitian

Sebelum melakukan *training*, CSPDarknet-53 di-*cloning* dan dibangun terlebih dahulu sebagai arsitektur YOLOv4. Kemudian, dilakukan penyesuaian *hyperparameter* dalam *configuration file* dan setelah itu *training data* dapat dimulai. *Weights file* yang merupakan *output* dari proses *training* langsung tersimpan di *storage* begitu *training* selesai.

Sementara itu, proses (b) pada Gambar 3., menunjukkan alur dari sistem perhitungan jumlah orang. *Weights file* dari hasil *training* memiliki peran penting dalam keberhasilan sistem. Jika performansi dari *weights file* tidak begitu bagus, maka sistem tidak dapat mendeteksi objek dengan akurat.

Setelah melakukan *load weights file* pada sistem, data uji dapat dimasukkan. Lalu sistem melakukan deteksi objek sekaligus menghitung jumlah objek yang terdeteksi tiap data ujinya. Begitu proses deteksi dan perhitungan selesai, maka parameter performansi pun dapat dikalkulasikan.

3.2. *Transfer Learning Pre-Trained* YOLOv4

Model arsitektur YOLOv4 saat ini telah memiliki *weights file* resmi yang dilatih menggunakan MS COCO *dataset* dan dapat mendeteksi hingga 80 kelas objek [12]. Karena penelitian ini hanya mendeteksi satu kelas saja pada sistemnya, maka *transfer learning* [13] pun perlu dilakukan.

Proses ini dilakukan dengan melakukan *training data* dengan model keluaran *weights file* yang harus sesuai dengan format YOLOv4.

Output dari *transfer learning* ini adalah *weights file* baru yang telah dilatih sendiri menggunakan *dataset* yang telah dikumpulkan. *Weights file* kemudian digunakan untuk melakukan proses deteksi dan perhitungan objek menggunakan YOLOv4.

3.3. Konfigurasi *Hyperparameter*

3.3.1 *Batch Size*

Batch size merupakan variabel yang menentukan seberapa banyak *image* atau *training data* yang dimasukkan saat *training*.

Semakin kecil nilai batch size yang digunakan, maka proses *training* semakin cepat. Sementara semakin besar nilai *batch size*, proses *training* pun akan memakan waktu yang cukup lama karena membutuhkan kapasitas *storage* yang lebih banyak.

Hal ini juga turut mempengaruhi akurasi pada sistem. Jika nilai *batch size* yang digunakan semakin besar, maka akurasinya pun akan semakin tinggi, karena akan semakin banyak fitur yang dipelajari oleh sistem [14].

3.3.2 *Subdivisions*

Sub divisions membagi nilai *batch* menjadi lebih kecil lagi, dan dapat disebut dengan *mini-batch*. Jika menggunakan nilai *batch* sebesar 64 dan dibagi dengan 8 *sub divisions*, maka menghasilkan nilai 8 yang berarti dilakukan proses *training* untuk 8 *images* tiap *mini-bacth*-nya. Proses ini berlangsung selama delapan kali hingga proses *training* pada satu *batch* tersebut selesai. Kemudian, sistem memproses *batch* selanjutnya yang juga bernilai 64.

Proses *sub divisions* bertujuan untuk mempercepat proses training sekaligus meningkatkan akurasi dengan bantuan GPU.

3.3.3 *Channels*

Nilai *channel* menentukan kedalaman citra dari data yang digunakan pada proses *training*. Jika data yang di-*training* menggunakan citra RGB, maka variabel *channel* harus bernilai tiga. Sementara jika menggunakan citra *greyscale*, maka *channel* menggunakan nilai satu.

3.3.4 *Learning Rate*

Learning rate merupakan penentu seberapa banyak *weight* yang diperbarui dalam proses *backpropagation*. *Learning rate* juga menentukan kecepatan pada iterasi sehingga dapat mencapai *loss function minimum*. Proses *training* berjalan semakin cepat jika nilai *learning rate* semakin tinggi [15]. Namun, nilai *learning rate* yang terlalu tinggi juga dapat menyebabkan nilai *loss function* turun-naik tidak menentu, sehingga dibutuhkan beberapa kali percobaan untuk mendapatkan nilai *learning rate* yang optimal [16].

3.3.5 *Max Batches*

Max batches merupakan banyaknya iterasi pada proses *training data*. Semakin tinggi nilai *max batch*, maka sistem akan semakin banyak mempelajari *data training*. Jumlah *training data* tidak boleh lebih dari jumlah *max batches*. Nilai *max batches* perlu disesuaikan dengan jumlah kelas dari objek yang akan dideteksi.

$$\textit{Max batches} = \textit{number of classes} \times 2000 \quad (1)$$

3.4. Parameter Performansi

3.4.1 Akurasi

Akurasi merupakan parameter uji yang menentukan kehandalan sistem dalam mengklasifikasikan objek [17]. Parameter akurasi berfungsi untuk menentukan kebenaran sistem dalam mendeteksi orang terhadap data uji. Akurasi dapat ditentukan menggunakan persamaan (2).

$$A = \frac{\Sigma B}{\Sigma n} \quad (2)$$

$$A = \frac{TP+TN}{TP+TN+FP+FN} \times 100\% \quad (3)$$

Dalam persamaan (2), variabel A adalah akurasi, variabel B merupakan jumlah objek yang terdeteksi dengan benar, dan variabel n adalah total keseluruhan data atau objek pada citra. Selain itu, perumusan akurasi dapat dikalkulasikan dengan menggunakan terminologi dari *confusion matrix* seperti persamaan (3).

3.4.2 Presisi

Presisi merupakan rasio dari jumlah objek yang terdeteksi dengan benar atau *True Positive* dibandingkan dengan seluruh data yang diprediksi positif. Dari persamaan (4), dapat diketahui bahwa semakin besar nilai *False Positive*, maka presisi semakin rendah dan begitu pun kebalikannya.

$$\text{Presisi} = \frac{TP}{TP+FP} \times 100\% \quad (4)$$

3.4.3 *Recall*

Recall merupakan rasio dari jumlah objek yang terdeteksi dengan benar atau *True Positive* dibandingkan dengan seluruh data yang positif. Nilai *recall* yang tinggi menunjukkan bahwa sistem dapat mengklasifikasikan kelas objek dengan benar. *Recall* dapat dikalkulasikan dengan persamaan (5).

$$\textit{Recall} = \frac{TP}{TP+FN} \times 100\% \quad (5)$$

3.4.4 F1 *Score*

F1 *Score* merupakan perbandingan rata-rata dari nilai presisi dan *recall*. F1 *Score* memiliki nilai tertinggi sebesar 1 dan terendah sebesar 0. Nilai F1 *Score* yang semakin mendekati 1, menunjukkan bahwa kinerja sistem telah baik. Secara matematis, nilai F1 *Score* dapat dilihat pada persamaan (6).

$$\text{F1 } \textit{Score} = 2 \times \frac{\text{Recall x Presisi}}{\text{Recall+Presisi}} \quad (6)$$

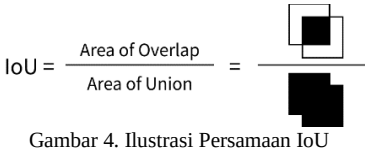
3.4.5 *Intersection over Union* (IoU)

IoU merupakan metrik yang mengevaluasi keakuratan sistem dalam mendeteksi objek pada *dataset* yang telah dilatih [18]. IoU membandingkan *ground-truth* atau objek pada citra dengan *predicted bounding box* dari model. Nilai IoU pada deteksi objek berperan sebagai nilai *threshold*

Terdapat dua nilai *threshold* IoU yang umum digunakan, yaitu 0.5 dan 0.75. Pada penelitian ini, nilai *threshold* yang digunakan adalah 0.5. Jika nilai *threshold* IoU $\geq$ 0.5, maka objek terdeteksi sebagai *True Positive* (TP). Dan ketika *threshold* IoU < 0.5, maka objek terdeteksi sebagai *False Positive* (FP).

IoU dapat diilustrasikan pada Gambar 4. dan Gambar 5. Secara matematis, IoU dapat didefinisikan pada persamaan (7).

$$\text{IoU} = \frac{A \cap B}{A \cup B} \quad (7)$$



Gambar 4. Ilustrasi Persamaan IoU

Gambar 5. Ilustrasi IoU pada citra

Gambar 5 menjelaskan persamaan dari Gambar 4. Pada Gambar 5, kotak berwarna merah merupakan *ground truth*, atau objek yang seharusnya dideteksi pada citra. Sementara kotak kuning merupakan *predicted bounding box* yang dihasilkan. Besar dari perpotongan antara kedua kotak yang ditandai dengan warna hijau tersebut menentukan besar dan kecilnya nilai IoU. Semakin jauh *bounding box* dari *ground truth*, maka semakin kecil nilai IoU yang dihasilkan.

3.4.6 *mean Average Precision* (mAP)

Nilai mAP merupakan metrik yang mengevaluasi performansi dari model deteksi objek. Sederhananya, mAP merupakan rata-rata dari nilai *average precision* (AP) dan mengukur seberapa bagus performansi dari *weights file* hasil *training data*. Sebelum mengkalkulasikan mAP, perlu dilakukan penyesuaian nilai *threshold* pada IoU untuk menentukan validasi dari objek yang dideteksi.

Penelitian ini menggunakan nilai *threshold* IoU sebesar 0.5, sehingga model menghasilkan mAP50 yang dapat juga disebut mAP@0.5.

4. HASIL DAN PEMBAHASAN

Pada penelitian ini, proses *training* dan pengujian dilakukan menggunakan laptop dengan spesifikasi seperti pada Tabel 1, dan spesifikasi *software* pada Tabel 2.

Tabel 1. Spesifikasi Laptop

Aspek	Spesifikasi
Processor	Intel® Core™ i7-9570 CPU @ 2.60 GHz
RAM	8 GB SODIMM DDR4 2667 MHz
Storage	SSD 512 GB
Graphics Card	Nvidia GeForce GTX 1650 4GB
Operating System	Windows 10 Home Single Language 64-bit

Tabel 2. Spesifikasi *Software*

Aspek	Software
Bahasa Pemorgraman	Python versi 3.9.1
Software Training Data	Google Colab
Software Testing Data	Microsoft Visual Studio Code
Command Prompt	Windows Powershell dan Git Bash

4.1. Skenario *Training Data*

Hasil dari *training data* pada sistem sangat mempengaruhi keberhasilan model YOLOv4 dalam mendeteksi objek. Dari proses training data, akan didapatkan parameter akurasi, presisi, recall, F1 score, IoU, dan mAP. *Input* dari proses ini adalah sejumlah data latih dan data validasi, sementara output dari proses ini yaitu *weights file* yang akan digunakan saat pengujian sistem.

Sebagian besar dataset yang digunakan diambil dari internet. Hal ini disebabkan kurang kondusifnya situasi luar untuk mengambil data langsung di transportasi publik.

Data latih dan data validasi sepenuhnya diambil dari *Open Image Datasets* yang dikelola oleh *Google*. Sementara untuk data uji, hanya sebagian kecil yang diambil langsung di lapangan dan selebihnya dari internet.

Training data dilakukan menggunakan skenario yang dapat dilihat pada Tabel 3.

Tabel 3. Skenario *Training Data*

Training Data	*Validation Data*	*Max Batches*	*Class*
3000	600	4000	*Person*
1500	300	4000	*Person*
500	100	6000	*Human face*

4.1.1 Skenario Pertama

Training data pada skenario pertama digunakan 3000 data latih dan 600 data validasi. Data yang dilatih berasal dari kelas *person*, di mana sistem akan mendeteksi orang mulai dari kepala hingga ujung kaki. Konfigurasi sistem yang diatur adalah sebagai berikut:

1. *Batch size* = 64
2. *Sub divisions* = 16
3. *Channels* = 3
4. *Learning rate* = 0.001
5. *Max batches* = 4000
6. *Random value* = 1

Namun, hasil dari *training* pada skenario ini tidak berhasil disebabkan oleh nilai dari *sub divisions* dan *random value* memakan memori cukup banyak saat *training* sehingga proses menjadi tersendat dan membuat program harus di-*running* berulang-ulang. Hal ini mengakibatkan grafik mAP pada training menjadi terputus-putus seperti yang dapat dilihat pada Gambar 6.

Dari *training* pada skenario pertama ini, karena proses tidak berjalan dengan lancar dan optimal maka parameter performansi yang dihasilkan pun menjadi sangat rendah seperti yang terlihat pada Tabel 4 dan Tabel 5.

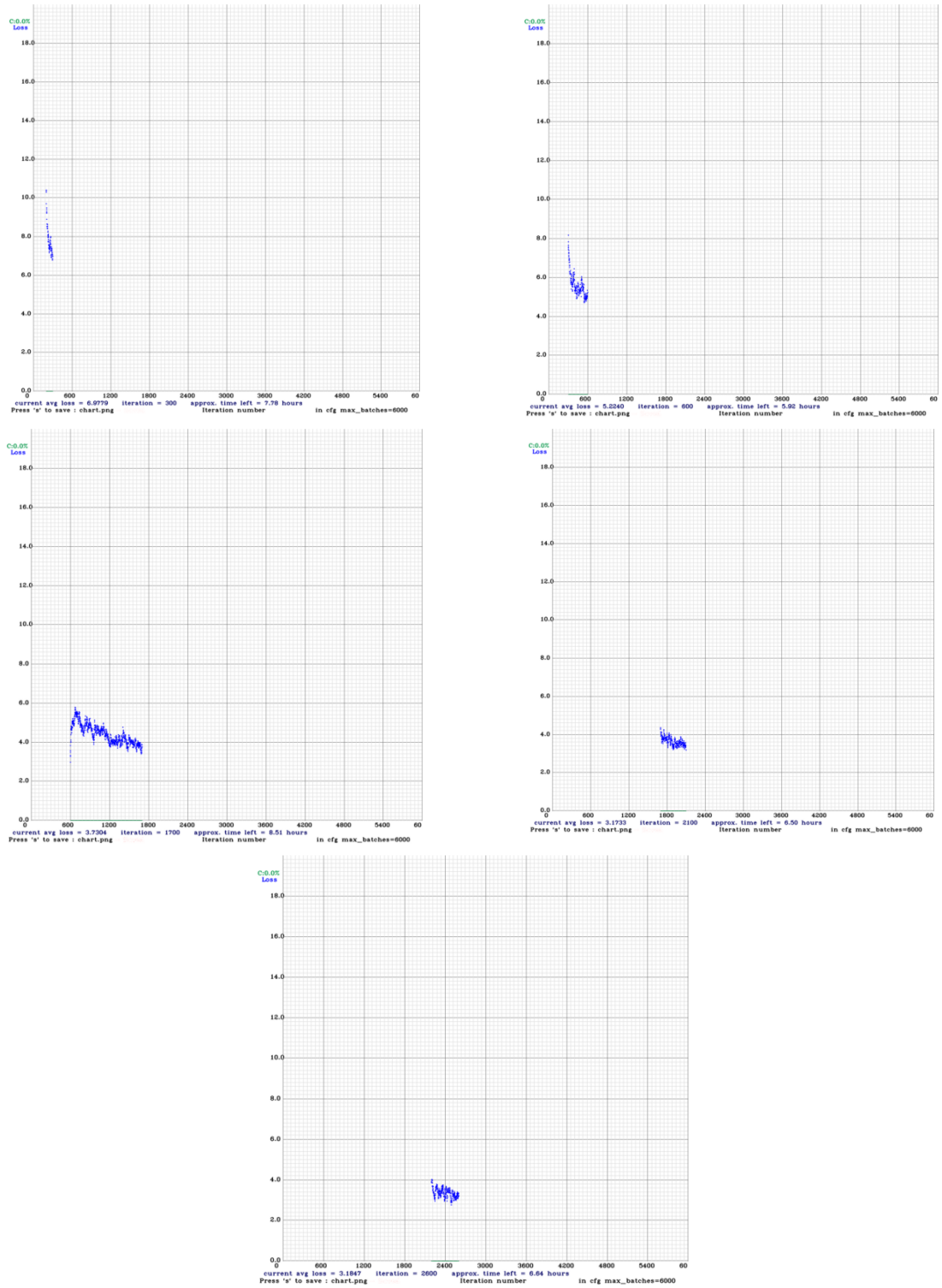
Tabel 4. *Confusion Matrix* Skenario Pertama

True Positive (TP)	*False Positive* (FP)	*False Negative* (FN)
715	1012	504

Tabel 5. Parameter Performansi Skenario Pertama

Konfigurasi	Nilai
Presisi (%)	41%
Recall (%)	59%
F1 *Score*	0.49
Average IoU (%)	31.58%
mAP (%)	40%
Average Loss	± 3

Setelah ditelusuri, 16 *sub divisions* yang juga dapat disebut dengan *mini-batch* ini ternyata kurang mampu untuk diproses pada spesifikasi laptop yang digunakan. *Batch size* sebesar 64 dibagi oleh 16 *sub divisions* menjadi 4. Dalam kata lain, sistem akan melakukan *training* terhadap empat data latih tiap 16 *mini-batch*. Selain itu, nilai *random value* juga turut berpengaruh dalam kegagalan proses training ini.



Gambar 6. Grafik *Training Data* Skenario Pertama

4.1.2 Skenario Kedua

Training data pada skenario kedua menggunakan data latih berjumlah 1500 dan data validasi sebesar 300 dengan data latih masih menggunakan kelas *person*. Skenario ini menggunakan nilai *sub divisions* dan *random value* yang berbeda dari skenario pertama.

1. *Batch size* = 64
2. *Sub divisions* = 32
3. *Channels* = 3
4. *Learning rate* = 0.001
5. *Max batches* = 4000
6. *Random value* = 0

Training data menghasilkan grafik mAP seperti pada Gambar 7.

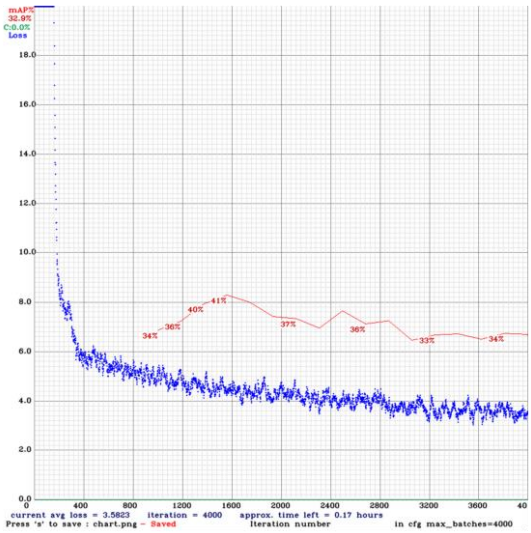
Hasil *training data* pada skenario ini tidak terlalu bagus. Dapat dilihat angka *True Positive* jauh lebih rendah dari *True Negative*, di mana sistem lebih banyak *error* dalam mendeteksi objek pada data latih. Hal ini menyebabkan parameter performansi menjadi rendah pada Tabel 6 dan Tabel 7.

Tabel 6. *Confusion Matrix* Skenario Kedua

True Positive **(TP)**	***False Positive*** **(FP)**	***False Negative*** **(FN)**
389	636	217

Tabel 7. Parameter Performansi Skenario Kedua

Konfigurasi	**Nilai**
Presisi (%)	38%
Recall (%)	64%
F1 *Score*	0.48
Average IoU (%)	29.38%
mAP (%)	42.83%
Average Loss	± 3



Gambar 7. Grafik *Training Data* Skenario Kedua

4.1.3 Skenario Ketiga

Training data pada skenario kedua menggunakan data latih berjumlah 500 dan data validasi sebesar 100. Jumlah data diturunkan karena setelah melalui beberapa *training* yang gagal sebelumnya, jumlah 1500 terlalu besar untuk diproses sistem. Kemudian, kelas dari data latih yang digunakan juga berbeda, yaitu *human face*.

Sebagian besar penumpang pada transportasi publik berada dalam kondisi duduk, sehingga yang terlihat paling jelas adalah wajah. Maka dari itu, *training data* kali ini dicoba menggunakan kelas *human face*.

Selain itu, *max batches* yang digunakan pada skenario ini lebih besar dari skenario sebelumnya, yaitu sebesdar 6000. *Average loss* pada skenario sebelumnya masih berada di atas 3 saat iterasi telah mencapai 4000. Agar sistem bisa mempelajari data latih dengan lebih lama lagi, besar *max batches* pun dinaikkan.

1. *Batch size* = 64
2. *Sub divisions* = 32
3. *Channels* = 3
4. *Learning rate* = 0.001
5. *Max batches* = 6000
6. *Random value* = 0

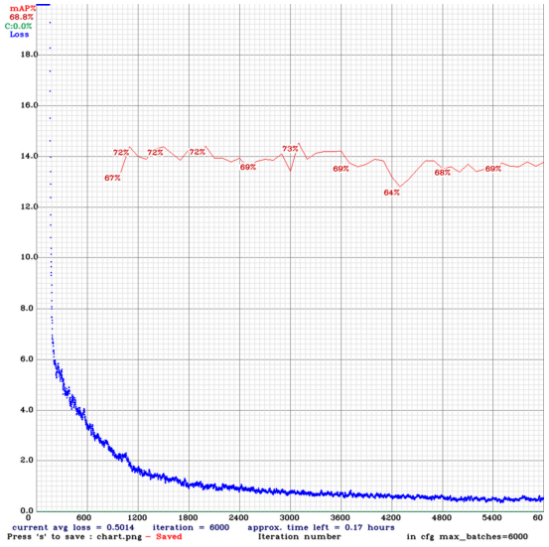
Hasil *training* dapat dilihat pada Gambar 8, dan parameter performansi terdapat di Tabel 8 dan Tabel 9.

Tabel 8. *Confusion Matrix* Skenario Ketiga

True Positive **(TP)**	***False Positive*** **(FP)**	***False Negative*** **(FN)**
141	41	45

Tabel 9. Parameter Performansi Skenario Ketiga

Konfigurasi	**Nilai**
Presisi (%)	77%
Recall (%)	76%
F1 *Score*	0.77
Average IoU (%)	62.88%
mAP (%)	72.68%
Average Loss	± 0.6



Gambar 8. Grafik *Training Data* Skenario Ketiga

Hasil *training* pada skenario ketiga telah jauh lebih baik dari sebelumnya. Jumlah *True Positive* pada skenario ini terlihat jauh lebih besar dari jumlah *True Negative*. Hal ini mencerminkan sistem telah dapat mendeteksi objek dengan benar dari dataset yang telah dilatih.

Nilai presisi pada *training data* menggunakan kelas *human face* adalah 77% dengan mAP sebesar 72.68%. *Average loss* yang terlihat pada Gambar 8 juga telah jauh berada di bawah 2, sehingga *weights file* pada skenario ini sudah jauh lebih baik dari yang sebelumnya dan dapat langsung diujikan untuk mendeteksi dan menghitung jumlah orang.

4.2. Skenario Pengujian Sistem

Setelah mendapatkan *pre-trained weights* yang telah dilatih menggunakan satu kelas objek, data uji bisa langsung diujikan menggunakan program yang telah disisipkan *weights file* baru. Program kemudian di-*running* untuk melihat seberapa akurat *pre-trained weights* dapat mendeteksi dan menghitung jumlah orang pada data uji.

Parameter yang dihitung pada skenario pengujian sistem ini hanyalah akurasi sistem dalam menghitung orang. Dalam pengujian sistem, akan ada dua skenario yang dilakukan.

4.2.1 Skenario Pertama

Pada skenario ini, program dijalankan menggunakan *pre-trained weights* dengan kelas *human face* yang telah dilatih pada skenario *training data*. Hasilnya, akurasi rata-rata sistem dalam menghitung jumlah objek pada satu citra sebesar 69%, dengan akurasi tertinggi sebesar 95%. Citra uji yang mencapai akurasi tertinggi terlihat pada Gambar 9.

Gambar 9. Akurasi Terbaik Skenario Pertama

Faktor yang sangat berpengaruh dalam keberhasilan sistem ini di antaranya adalah aksesoris wajah penumpang, posisi duduk penumpang dan seberapa jauh penumpang tersebut duduk. Dari pengujian skenario pertama, penumpang yang menggunakan topi dan masker tidak terdeteksi oleh sistem meskipun wajahnya terlihat paling jelas.

4.2.2 Skenario Kedua

Skenario ini menjalankan program menggunakan *pre-trained weights* YOLOv4 yang telah dilatih oleh MS COCO *dataset*. Nilai mAP dari *weights file* ini adalah 62.8%, sedikit lebih rendah dari mAP yang didapatkan pada proses *training data*. Penyebabnya, MS COCO melatih *dataset* untuk 80 kelas. Sehingga nilai mAP tersebut masih bisa terbilang tinggi jika dibandingkan dengan *pre-trained weights* untuk satu kelas.

Tujuan dari skenario kedua ini adalah untuk mengetahui akurasi menghitung orang menggunakan *pre-trained weights* dari MS COCO. Karena pada *weights* ini tidak terdapat kelas *human face*, maka digunakan kelas *person*.

Hasilnya, akurasi rata-rata pada sistem *people counting* menggunakan *weights file* dari MS COCO adalah sebesar 62%, dengan akurasi terbaik sebesar 90%. Citra dengan akurasi tersebut dapat dilihat pada Gambar 10.

Gambar 10. Akurasi Terbaik Skenario Kedua

Berbeda dari skenario sebelumnya, faktor yang cukup berpengaruh dalam keberhasilan sistem mendeteksi objek pada skenario ini adalah posisi badan penumpang. Karena sebagian besar penumpang dalam kondisi duduk, sehingga hanya bagian kepala yang terlihat. Selain itu, skenario ini sulit mendeteksi dan menghitung jumlah penumpang yang berdempetan.

4.3. Perbandingan Hasil Pengujian

Setelah menyelesaikan pengujian terhadap kedua skenario, dapat dibandingkan parameter performansinya. Di mana *weights file* yang dilatih untuk satu kelas memiliki akurasi yang sedikit lebih tinggi dibandingkan dengan *weights file* yang dilatih untuk 80 kelas.

Kedua skenario tersebut memiliki keunggulan dan kelemahannya masing-masing, seperti yang terlihat pada Gambar 11.

Gambar 11. Perbandingan Data Uji dengan Akurasi Terbaik dari Skenario Pertama

Gambar bagian atas menunjukkan citra dengan akurasi tertinggi menggunakan *pre-trained weights* yang telah dilatih untuk satu kelas. Program yang menggunakan *weights file* ini dapat mendeteksi dan menghitung 21 orang dalam citra, sementara program yang menggunakan *weights file* dari MS COCO hanya dapat menghitung hingga 7 orang.

Kemudian pada Gambar 12, terlihat bahwa citra dengan akurasi tertinggi dari skenario kedua, memiliki akurasi yang sedikit lebih rendah jika menggunakan skenario pertama.

Gambar 12. Perbandingan Data Uji dengan Akurasi Terbaik dari Skenario Kedua

Skenario pertama tidak dapat mendeteksi penumpang yang berada dalam kondisi berbalik badan, dan skenario kedua tidak dapat menghitung penumpang yang hanya terlihat kepalanya saja.

5. KESIMPULAN

Berdasarkan hasil simulasi dan analisis penelitian yang telah dilakukan sebelumnya, dapat ditarik kesimpulan bahwa konfigurasi terbaik untuk menghasilkan *weights file* yang bagus pada penelitian ini adalah dengan menggunakan *max batches* sebesar 6000, *sub divisions* sebesar 32, dan *random value* yang bernilai 0. Dari skenario *training* data, akurasi terbaik didapat sebesar 72.68%. Sementara untuk skenario pengujian data, akurasi rata-rata sistem dalam mendeteksi dan menghitung orang menggunakan *pre-trained weights* yang telah dilatih sendiri adalah sebesar 69%, dengan akurasi terbaik sebesar 95%. Kemudian akurasi rata-rata sistem dalam mendeteksi dan menghitung orang menggunakan *pre-trained weights* resmi oleh COCO *dataset* adalah sebesar 62%, dengan akurasi terbaik sebesar 90%. Selain itu, pengujian menggunakan skenario pertama cukup sulit mendeteksi wajah penumpang yang menggunakan aksesoris seperti masker, topi, dan lainnya. Sementara pengujian menggunakan skenario kedua cukup sulit mendeteksi ,,mn b

penumpang yang hanya terlihat kepalanya saja.

DAFTAR PUSTAKA

[1] S. Saxena and D. Songara, "Design of people counting system using MATLAB," in *2017 Tenth International Conference on Contemporary Computing (IC3)*, 2017, pp. 1–3, doi: 10.1109/IC3.2017.8284344.

[2] S. Mane and S. Mangale, "Moving Object Detection and Tracking Using Convolutional Neural Networks," in *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2018, pp. 1809–1813, doi: 10.1109/ICCONS.2018.8662921.

[3] H. Zhang, Y. Sun, L. Liu, X. Wang, L. Li, and W. Liu, "ClothingOut: a category-supervised GAN model for clothing segmentation and retrieval," *Neural Comput. Appl.*, vol. 32, no. 9, pp. 4519–4530, 2020.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017, doi: 10.1109/TPAMI.2016.2577031.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[6] K. Shih, C. Chiu, and Y. Pu, "Real-time Object Detection via Pruning and a Concatenated Multi-feature Assisted Region Proposal Network," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 1398–1402, doi: 10.1109/ICASSP.2019.8683842.

[7] A. Mrutyunjay, P. Kondrakunta, and H. Rallapalli, "Non-max Suppression for Real-Time Human Localization in Long Wavelength Infrared Region," in *Advances in Decision Sciences, Image Processing, Security and Computer Vision*, Springer, 2020, pp. 166–174.

[8] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv Prepr. arXiv2004.10934*, 2020.

[9] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv Prepr. arXiv1804.02767*, 2018.

[10] C. Wang, H. Mark Liao, Y. Wu, P. Chen, J. Hsieh, and I. Yeh, "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 1571–1580, doi: 10.1109/CVPRW50498.2020.00203.

[11] L. V. Magnusson and R. Olsson, "Improving the Canny Edge Detector Using Automatic Programming: Improving Non-Max Suppression," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, 2016, pp. 461–468.

[12] H. Caesar, J. Uijlings, and V. Ferrari, "COCO-Stuff: Thing and Stuff Classes in Context," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1209–1218, doi: 10.1109/CVPR.2018.00132.

[13] L. Yang, L. Jing, J. Yu, and M. K. Ng, "Learning Transferred Weights From Co-Occurrence Data for Heterogeneous Transfer Learning," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 27, no. 11, pp. 2187–2200, 2016, doi: 10.1109/TNNLS.2015.2472457.

[14] P. M. Radiuk, "Impact of training set batch size on the performance of convolutional neural networks for diverse datasets," *Inf. Technol. Manag. Sci.*, vol. 20, no. 1, pp. 20–24, 2017.

[15] Y. Wu *et al.*, "Demystifying Learning Rate Policies for High Accuracy Training of Deep Neural Networks," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 1971–1980, doi: 10.1109/BigData47090.2019.9006104.

[16] J. Konar, P. Khandelwal, and R. Tripathi, "Comparison of Various Learning Rate Scheduling Techniques on Convolutional Neural Network," in *2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS)*, 2020, pp. 1–5, doi: 10.1109/SCEECS48394.2020.94.

[17] N. D. Miranda, L. Novamizanti, and S. Rizal, "Convolutional Neural Network Pada Klasifikasi Sidik Jari Menggunakan Resnet-50," *J. Tek. Inform.*, vol. 1, no. 2, pp. 61–68, 2020.

[18] G. Zhang, L. Ge, Y. Yang, Y. Liu, and K. Sun, "Fused Confidence for Scene Text Detection via Intersection-over-Union," in *2019 IEEE 19th International Conference on Communication Technology (ICCT)*, 2019, pp. 1540–1543, doi: 10.1109/ICCT46805.2019.8947307.