INSYST

Journal of Intelligent System and Computation

p-ISSN:2621-9220
e-ISSN:2722-1962

Volume 5 Nomor 1, April 2023

Published By **Lembaga Penelitan dan Pengabdian Masyarakat (LPPM)**
Institut Sains dan Teknologi Terpadu Surabaya (ISTTS)
formerly **Sekolah Tinggi Teknik Surabaya (STTS)**

Managed By
Departement of Informatics
Institut Sains dan Teknologi Terpadu Surabaya (ISTTS)

INSYST

Journal of Intelligent System and Computation

Volume 05 Nomor 01 April 2023

INSYST

Journal of Intelligent System and Computation

Volume 05 Nomor 01 April 2023

Reviewer:

INSYST

Journal of Intelligent System and Computation

Volume 05 Nomor 01 April 2023

Reviewer:

Windra Swastika, Ph.D
Universitas Ma Chung, Indonesia

Romy Budhi Widodo, Dr.Eng.
Universitas Ma Chung, Indonesia

Theresia Ratih Dewi Saputri, Ph.D.
Ciputra University

Richard Evan Sutanto, Ph.D.
Ciputra University

INSYST

Journal of Intelligent System and Computation

Volume 05 Nomor 01 April 2023

Daftar Isi

Ekstraksi Partitur Balok Monofonik Untuk Instrumen Flute dengan CRNN dan CRF

Stella Vania[1], Patrick Sutanto[1], Ricky Sutanto[1], dan Joan Santoso[1]
[1]Departemen Informatika, Fakultas Sains dan Teknologi, Institut Sains dan Teknologi Terpadu Surabaya, Surabaya, Indonesia

Corresponding author: Joan Santoso (e-mail: joan@stts.edu).

ABSTRACT Sheet music notation is not easy for beginners to read in the world of music. This is where Optical Music Recognition (OMR) can come into play. OMR is a field of research that investigates how to computationally read musical notation in documents. Music beginners can be helped in reading sheet music with a program that implements OMR and provides output in a format that is easily understood by users. This scientific work is made with a deep learning approach in several architectures. The dataset used is Camera-PrIMuS which consists of an image dataset in a line of sheet music and also the ground-truth per object in the image in question. The architectures used are CRNN, CRNN-CRF, and Attention. Of the three architectures, the best results were obtained for the Attention architecture with a symbol error rate (SER) of around 9%, followed by CRNN with a SER of around 84%, and CRNN-CRF which, based on test results, is not suitable for OMR because of uncoverging loss value. The Attention architecture broadly consists of encoder and decoder blocks. The encoder functions to receive image input and encode the image. The encoding results are then received by the decoder whose role is to perform the decoding and predict the next sequence based on the encoding results from the encoder. In its implementation, the program can receive input in the form of a slightly skewed full sheet sheet image, so the program will also perform skew-correction and crop the image per line so that input from the user can be processed by the model. The output of the model which is still in the form of predictive labels will be processed again to produce numeric notes and MIDI files which are relatively easier for users to understand.

KEYWORDS Artificial Neural Networks, Recurrent Neural Network.

ABSTRAK Notasi partitur balok bukanlah notasi yang mudah dibaca oleh pemula dalam dunia musik. Di sinilah *Optical Music Recognition* (OMR) dapat berperan. OMR merupakan sebuah pembelajaran mengenai komputer yang dapat mengenali objek dalam partitur balok. Dengan adanya program yang menerapkan OMR dan memberikan output dengan format yang mudah dipahami oleh pengguna, maka pemula dalam dunia musik dapat terbantu dalam membaca partitur not balok. Karya ilmiah ini dibuat dengan pendekatan *deep learning* dalam beberapa arsitektur. Dataset yang digunakan adalah Camera-PrIMuS yang terdiri dari dataset gambar sebaris partitur musik dan juga ground-truth per objek pada gambar yang bersangkutan. Arsitektur yang digunakan adalah CRNN, CRNN-CRF, dan *Attention*. Dari ketiga arsitektur tersebut, hasil terbaik diperoleh pada aristektur *Attention* dengan *symbol error rate* (SER) sekitar 9%, diikuti dengan CRNN dengan SER sekitar 84%, dan CRNN-CRF yang berdasarkan hasil uji coba tidaklah cocok untuk OMR dengan nilai loss yang tidak kunjung turun dalam proses training. Arsitektur *Attention* secara garis besar terdiri dari blok *encoder* dan *decoder*. *Encoder* berfungsi untuk menerima input gambar dan melakukan encoding terhadap gambar tersebut. Hasil encoding kemudian diterima oleh decoder yang berperan untuk melakukan *decoding* dan memprediksi *sequence* selanjutnya berdasarkan hasil encoding dari *encoder*. Dalam implementasinya program dapat menerima input berupa gambar selembar partitur penuh yang agak miring, maka program juga akan melakukan *skew-correction* dan pemotongan gambar per baris agar input dari pengguna dapat diproses oleh model. Output dari model yang masih berupa label-label prediksi akan diproses kembali agar menghasilkan not angka dan file MIDI yang relatif lebih mudah untuk dipahami oleh pengguna.

KATA KUNCI Artificial Neural Networks, Recurrent Neural Network

I. PENDAHULUAN

Pemula dalam dunia musik sangat memungkinkan untuk mengalami kesulitan dalam membaca partitur balok. Hal ini wajar mengingat notasi partitur balok tidak begitu mudah untuk dibaca. Dengan demikian, program yang mampu membaca partitur balok dan mengkonversikan hasil pembacaannya dalam bentuk output yang mudah dipahami oleh pengguna dapat membantu para pemula dalam membaca notasi partitur balok.

Optical Music Recognition (OMR) merupakan sebuah bidang penelitian yang bertujuan agar komputer dapat membaca partitur musik [1]. Hasil pembacaan tersebut kemudian akan disimpan ke dalam format yang dapat dibaca mesin. Setelah tersimpan secara digital, informasi musikal tersebut biasanya disimpan dalam format file yang umum sesuai kegunaannya. Sebagai contoh adalah file MIDI jika tujuan dari penerapan OMR itu adalah untuk memainkan kembali musik dari partitur yang bersangkutan.

OMR memiliki kemiripan dengan *Optical Character Recognition* (OCR), keduanya melakukan ekstraksi informasi dari gambar sehingga OMR sempat disebut sebagai music OCR. Walaupun demikian, OMR memiliki tantangan yang berbeda dibanding OCR. Dengan beberapa perbedaan yang ada antara OMR dan OCR, kurang tepat untuk dikatakan bahwa OMR adalah music OCR.

Salah satu perbedaan adalah walaupun notasi partitur balok pada dasarnya tersusun atas elemen-elemen tertentu (kepala not, tangkai not, bendera), posisi elemen tersebut dan kombinasinya terhadap elemen lain dapat mempengaruhi makna semantik dari elemen tersebut. Sebagai contoh, not yang terletak pada garis paling bawah dengan kunci G (nada E4) merepresentasikan nada yang berbeda dengan not pada posisi yang sama namun dengan kunci F (nada G2). Hal seperti demikian tidak ditemui pada permasalahan OCR.

OMR memiliki beberapa pendekatan yang dapat digunakan, salah satunya adalah pendekatan secara *deep learning* yang digunakan pada pembuatan karya ilmiah ini. Karya ilmiah ini dibuat dalam tiga arsitektur, yaitu CRNN, CRNN-CRF, dan *Attention*. Masing-masing arsitektur akan dievaluasi berdasarkan *symbol error rate* yang diperoleh dari perhitungan *Levenshtein distance*. Penggunaan metrik ini terinspirasi dari [2].

II. KAJIAN PUSTAKA

OMR memiliki beberapa pendekatan, baik pendekatan tanpa *machine learning* maupun pendekatan dengan *machine learning*. Secara general, OMR secara garis besar memiliki *pipeline* sebagai berikut, sesuai yang diutarakan [3] dan disempurnakan oleh [4]:

A. PREPROCESSING

Preprocessing merupakan tahapan untuk memanipulasi gambar agar lebih mudah diproses dalam tahapan-tahapan sebelumnya. Yang termasuk dalam tahapan ini antara lain adalah *skew-correction, binarization, noise removal*, dan sebagainya.

B. MUSIC OBJECT DETECTION

Tahapan ini bertujuan untuk menemukan dan mengklasifikasikan objek yang relevan dari gambar.

C. NOTATION ASSEMBLY

Tahapan ini bertujuan untuk memperoleh kembali informasi kontekstual semantik dari objek-objek yang berhasil ditemukan pada tahapan sebelumnya.

D. ENCODING

Tahapan ini bertujuan untuk mengkonversikan informasi kontekstual semantik dari tahapan sebelumnya ke dalam format yang umum sesuai tujuannya. Contohnya adalah format MIDI untuk memainkan musik tersebut, atau MusicXML untuk modifikasi lebih jauh pada program notasi music

Dengan adanya proses pengembangan OMR dengan *machine learning*, beberapa langkah yang biasanya ada dapat dihilangkan atau bahkan menjadi langkah yang lebih besar. Sebagai contoh, tahapan *music object detection* dulunya terdiri dari langkah segmentasi dan langkah klasifikasi. Pada tahapan segmentasi, dilakukan *staff-line removal* karena garis paranada (*staff-line*) dapat mengganggu proses klasifikasi karena garis paranada dapat mempersulit proses pemisahan dua objek yang terpisah dengan menggunakan metode *connected component analysis*. Walaupun demikian, dengan pendekatan *deep learning*, objek tetap dapat terdeteksi tanpa menghilangkan garis paranada, seperti pada [5], [6], [7], [8].

Pada pembuatan karya ilmiah ini, salah satu arsitektur yang dicoba adalah CRNN, seperti yang terdapat pada [2]. Digunakan pula arsitektur CRNN-CRF, CRF pada awalnya diharapkan untuk dapat mempelajari *constraint-constraint* yang diperoleh pada saat training sehingga label yang dipilih bukanlah label dengan probabilitas tertinggi. Dengan adanya CRF, label yang dipilih adalah label dengan probabilitas tertinggi yang memenuhi *constraint* yang dipelajari.

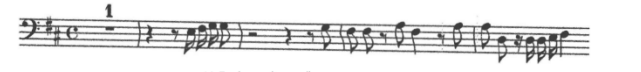
Arsitektur terakhir yang dicoba dalam pembuatan karya ilmiah ini adalah *Attention*. Percobaan arsitektur ini terinspirasi dari cara kerja *neural machine translation* [9][10][13][14]. Uji coba dalam arsitektur ini juga merupakan improvisasi dari arsitektur CRNN. Arsitektur ini dapat memberikan bobot lebih (*attention*) pada bagian dari partitur yang berkaitan dengan suatu label tertentu. Secara garis besar, arsitektur ini terdiri dari *encoder* dan *decoder*. Layer *encoder* terdiri dari layer CNN[11] dan Bi-LSTM. Layer *decoder* terdiri dari layer LSTM[15]. Jika diperhatikan, layer-layer yang digunakan pada arsitektur *Attention* memiliki kemiripan dengan layer CRNN yang terdiri dari layer CNN dan Bi-LSTM. Walaupun demikian,

perbedaan proses yang terjadi di antara layer-layer tersebut memberikan hasil akhir yang berbeda. Pada arsitektur *Attention*, output LSTM pada *decoder* tidak langsung digunakan untuk memprediksi, melainkan diproses dengan output Bi-LSTM dari *encoder* untuk memperoleh bobot *attention*. Dengan adanya mekanisme seperti ini, arsitektur ini dapat melakukan *aligning* antara gambar dengan label secara otomatis.

III. *DATASET*

Dataset yang digunakan dalam pengerjaan penelitian ini berjudul "*Camera Printed Images of Music Staves*" (Camera-PrIMuS). Dataset ini digunakan dan diterbitkan bersama dengan paper *Camera-PrIMuS: Neural End-To-End Optical Music Recognition on Realistic Monophonic Scores* [2]. Dataset ini diterbitkan dengan tujuan untuk memenuhi kebutuhan dataset dalam penelitian *Optical Music Recognition* (OMR) dengan pendekatan *machine learning*, dikarenakan seluruh penelitian berbasis *machine learning* membutuhkan dataset dalam jumlah besar dengan kualitas baik.

Dataset Camera-PrIMuS terdiri dari 87.678 awalan partitur not balok real yang direpresentasikan dalam enam jenis format file. Pada pembuatan karya ilmiah ini, hanya akan digunakan dua jenis format file dari keenam format yang ada, yaitu gambar partitur terdistorsi dengan format jpg dan representasi simbol musik dalam *semantic encoding* sebagai *ground-truth*. Contoh dari kedua format data yang digunakan dapat dilihat pada gambar 1.



(a) Gambar partitur musik

clef-F4 keySignature-DM timeSignature-C multirest-1 barline rest-quarter rest-eighth note-E3_sixteenth
note-F#3_sixteenth note-G3_sixteenth note-G3_eighth barline rest-half rest-quarter rest-eighth note-G3_eighth
barline note-F#3_eighth note-F#3_eighth rest-eighth note-A3_eighth note-F#3_quarter rest-eighth note-A3_eighth
barline note-A3_eighth note-D3_eighth rest-sixteenth note-D3_sixteenth note-D3_sixteenth note-E3_sixteenth
note-F#3_quarter

(b) Semantic encoding

GAMBAR 1. Contoh sampel dataset

Pada *semantic encoding* dataset Camera-PrIMuS, terdapat 1.781 label. Secara garis besar, 1.781 label tersebut terdiri dari 10 label *clef*, 14 label *key signature*, 39 label *time signature*, 1384 label note dengan *range* nada antara A2 hingga G5, 210 label *grace* note dengan *range* nada antara A2 hingga G5, 24 label *rest*, 98 label *multirest*, 2 label lainnya, yaitu *tie* dan *barline*.

Pada notasi partitur not balok real, seluruh objek-objek musikal tidak memiliki jumlah kemunculan yang sama. Karena dataset Camera-PrIMuS dibuat berdasarkan notasi musik *real*, persebaran kemunculan tiap label pun juga tidak merata. Sebagai contoh, label berjenis *note* jauh lebih sering muncul dibanding label jenis lainnya. Hal ini sangatlah wajar karena mayoritas bagian dari notasi musik tersusun atas note. Label *barline* menduduki urutan kedua, karena setiap data pasti memiliki beberapa barline.

A. PREPROCESSING GAMBAR

Walaupun gambar partitur pada dataset yang berformat jpeg memiliki tiga *color channel*, gambar akan diload secara grayscale. Proses *load* gambar secara grayscale ini dilakukan dengan pertimbangan bahwa partitur pada umumnya memiliki warna yang bersifat monokrom. Pixel-pixel gambar direpresentasikan dalam bentuk matrix dua dimensi. Setiap pixel memiliki range nilai antara 0 hingga 255, semakin terang warna yang direpresentasikan pixel tersebut, semakin besar pula nilainya. Nilai-nilai tersebut akan dibalik dengan mengurangkannya dengan 255. Nilai-nilai tersebut akan dinormalisasi sehingga menjadi berkisar antara 0 hingga 1.

Model membutuhkan input gambar dengan dimensi yang sama untuk setiap *batch*-nya, sementara gambar partitur pada dataset memiliki dimensi lebar dan tinggi yang bervariasi. Karena itulah proses *resize* perlu dilakukan. Walaupun demikian, proses *resize* tidak dapat dilakukan dengan menentukan ukuran yang pasti untuk setiap gambar, karena setiap gambar memiliki proporsi panjang dan tinggi yang berbeda. Jika dipatok ukuran yang sama untuk setiap gambar, maka akan terdapat gambar yang *stretch*, gambar seperti demikian dapat mengganggu proses training.

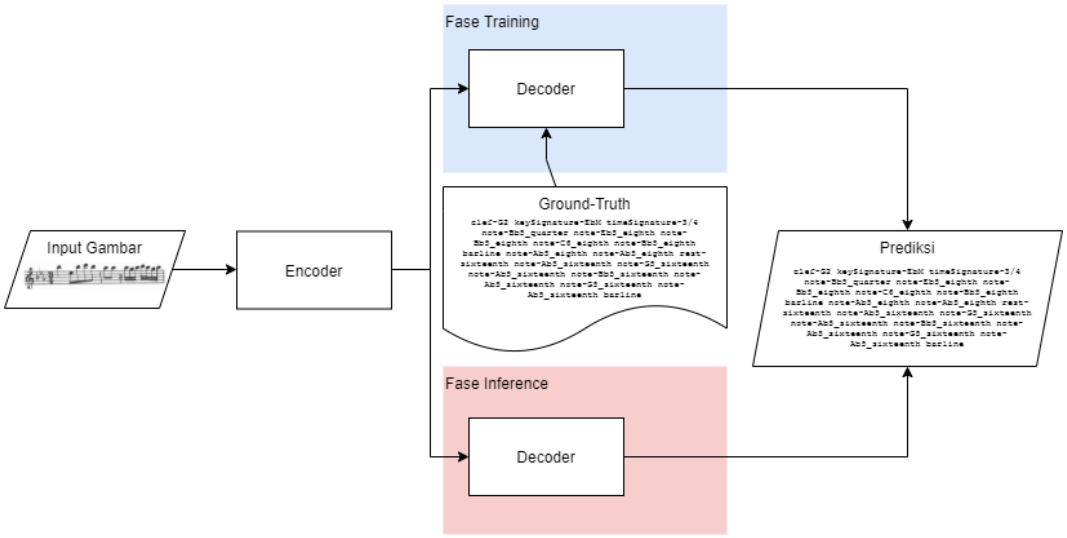
Agar gambar tidak stretch, proses resize yang dilakukan adalah sebagai berikut. Pertama, akan dicari terlebih dahulu dimensi terpanjang dan tertinggi pada batch yang diproses. Memperbesar gambar dari ukuran aslinya dapat memperburuk kualitas gambar. Gambar berukuran besar juga dapat memperlambat proses training karena jumlah parameter yang akan di-training juga semakin banyak. Dengan alasan ini, kedua nilai tersebut masing-masing diperkecil menjadi 60 persennya saja.

Dalam proses *resize*, tinggi gambar akan menjadi sesuai dengan tinggi maksimal gambar pada batch tersebut. Untuk menjaga proporsi gambar, panjang gambar setelah proses *resize* diperoleh dengan membagi panjang dengan tinggi gambar asli dan mengalikannya dengan dimensi tinggi maksimal. Agar dimensi panjang seluruh gambar pada *batch* juga seragam, maka gambar yang tidak mencapai dimensi panjang maksimal akan diberi padding sehingga panjangnya menjadi seragam, yaitu 60% dari panjang maksimal gambar pada batch yang diproses

B. PREPROCESSING GROUND-TRUTH

Pada dataset Camera-PrIMuS, *ground-truth* untuk setiap data gambar adalah berupa rangkaian label untuk setiap objek yang terdapat pada partitur tersebut. Setiap label pada suatu file *ground-truth* untuk masing-masing gambar dipisahkan dengan tab.

Preprocessing ground-truth diawali dengan penambahan token "<START>" pada awal setiap rangkaian ground-truth



GAMBAR 2. Arsitektur Model Secara Garis Besar

dan juga token "<END>" pada akhir setiap rangkaian ground-truth. Penambahan kedua token tersebut berfungsi sebagai penanda awal dan akhir sequence untuk model yang menerapkan konsep *attention*, hal ini dilakukan karena *decoder* dalam fase *training* pada *attention* membutuhkan kedua penanda tersebut

Sama seperti pada gambar, model juga membutuhkan *ground-truth* dengan panjang yang sama per *batch*-nya. Maka dari itu, akan dihitung rangkaian *ground-truth* terpanjang pada *batch* yang diproses. Setiap rangkaian *ground-truth* yang lebih pendek akan diberi token "<PAD>" seperlunya setelah token "<END>" hingga memiliki panjang yang sama dengan rangkaian *ground-truth* terpanjang

Pada akhir *preprocessing ground-truth*, seluruh *ground-truth* beserta token-token "<START>", "<END>", dan "<PAD>" akan diubah ke dalam bentuk *one-hot encoding*. Proses perubahan ke bentuk ini didasari dengan pemahaman bahwa model tidak dapat secara langsung menangani label-label tersebut sehingga label-label tersebut perlu diubah ke dalam bentuk representasi angka. Representasi angka dalam bentuk *one-hot encoding* dipilih karena dalam dataset Camera-PrIMuS, label-label yang ada tidak bersifat *ordinal* (tidak memiliki urutan khusus). Berbeda kasusnya jika label-label ground-truth memiliki urutan berdasarkan label-label itu sendiri. Contoh untuk kasus ini adalah label dingin, normal, dan panas.

Dalam kasus dengan label ini memiliki urutan tersendiri berdasarkan suhunya, representasi angka untuk label dapat digantikan dengan representasi angka biasa secara berurutan.

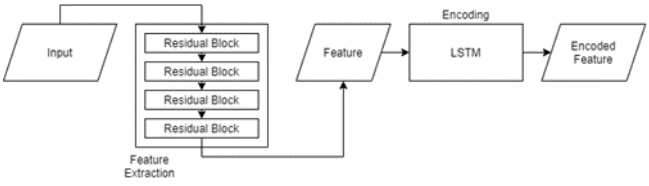
IV. METODE YANG DIGUNAKAN

Arsitektur yang yang dibuat pada karya ilmiah ini berbasiskan konsep *attention* yang secara garis besar terdiri atas *encoder* dan *decoder*. Gambaran mengenai arsitektur yang diajukan dapat dilihat pada **Error! Reference source not found.**. Tujuan dari arsitektur ini adalah untuk melakukan prediksi label suatu objek tertentu dari suatu bagian gambar berdasarkan probabilitas tertingginya.

$$\hat{y} = \arg\max P(y|x) \quad (1)$$

A. ENCODER

Blok *encoder* berperan dalam melakukan *embedding* terhadap gambar. *Encoder* yang dibuat pada penelitian ini secara garis besar terdiri dari empat *residual block*[12] dan sebuah layer LSTM. *Residual block* yang pertama menerima input berupa gambar yang telah melalui proses *preprocessing*. Output dari setiap *residual block* akan diteruskan ke *residual block* selanjutnya. Kegunaan utama dari *residual block* adalah untuk melakukan *feature extraction*. *Feature* yang diperoleh akan melalui proses *encoding* pada layer LSTM menjadi *encoded feature*. Untuk lebih jelasnya, dapat dilihat pada gambar 3.



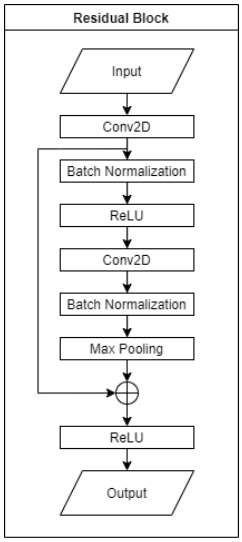
GAMBAR 3. Arsitektur Encoder

Encoder menerima input gambar per *batch* yang telah melalui *preprocessing*. Pada pengerjaan penelitian ini, jumlah data per batch adalah 16. Masing-masing gambar pada batch terdiri dari satu *channel* warna. Setiap gambar memiliki 2 dimensi, yaitu dimensi panjang dan tinggi

Walaupun ukuran panjang dan tinggi gambar pada suatu *batch* harus sama, masing-masing *batch* dapat memiliki dimensi panjang dan tinggi yang berbeda-beda. Penentuan

dimensi panjang dan tinggi gambar pada suatu batch dilakukan berdasarkan dimensi gambar terpanjang dan dimensi gambar tertinggi dalam batch yang bersangkutan.

Untuk memperoleh feature yang akan di-train pada proses *training*, perlu dilakukan *feature extraction* untuk mengambil *feature* dari gambar input. Proses *feature extraction* ini dilakukan oleh *residual block*. Pada penelitian ini, digunakan empat *residual block*, output suatu *residual block* akan menjadi input untuk *residual block* selanjutnya. Output dari *residual block* terakhir adalah berupa *feature* yang akan diumpankan ke layer selanjutnya pada *encoder*, yaitu layer LSTM. Arsitektur masing-masing *residual block* pada penelitian ini dapat dilihat pada.gambar 4



GAMBAR 4. Arsitektur Residual Block

Pada masing-masing residual block, input pada awalnya akan melalui *convolutional* layer 2 dimensi. Karena adanya distribusi data yang berbeda pada setiap *batch*, maka perlu dilakukan normalisasi. Tanpa adanya normalisasi, proses training akan menjadi kurang optimal karena karena model menjadi kurang stabil. Untuk itulah output *convolutional* 2 dimensi pertama melewati layer *batch normalization*[16].

Setelah melalui layer *batch normalization*, terdapat activation function ReLU[19][20] yang membiarkan nilai positif dan mengubah seluruh nilai negatif menjadi 0. Dengan adanya ReLU, tidak terdapat lagi nilai negatif pada output.

Nilai yang telah melalui activation function ReLU tersebut akan melalui *convolutional* layer 2 dimensi dan *batch normalization* layer lagi. Untuk mengurangi *feature* yang dihasilkan, digunakanlah layer *max pooling*[18].

Untuk menghindari permasalahan *vanishing gradient* ataupun *exploding gradient*[17] yang umum terjadi dengan banyaknya layer yang digunakan, output dari seluruh rangkaian sebelumnya ditambahkan dengan output dari *convolutional* layer 2 dimensi pertama. Dengan demikian, *feature-feature* yang mungkin mulai menghilang ataupun terlalu menonjol dapat distabilkan kembali. *Feature* tersebut akan diteruskan ke *residual block* selanjutnya untuk diproses. Pada *residual block* terakhir, *feature* akan diteruskan menuju layer LSTM.

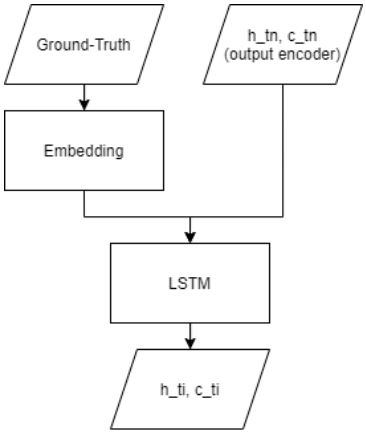
Layer LSTM pada *encoder* berperan dalam melakukan *encoding* terhadap *feature* yang dihasilkan *residual block* sehingga *encoded feature* tersebut dapat diproses lebih lanjut oleh *decoder*. Lebih spesifiknya, layer LSTM yang digunakan pada pengerjaan penelitian ini adalah Bi-LSTM.

Pada pengerjaan penelitian ini, digunakan hanya satu layer LSTM saja. Layer LSTM pada awalnya akan menerima input berupa *feature* dari *residual block*. *Hidden state* dan *cell state* pertama diinisialisasi dengan 0, nilai-nilai tersebut akan berubah seiring dengan proses yang berjalan. Output dari LSTM adalah *hidden state* dari masing-masing *timestep*, *hidden state* terakhir, dan *cell state* terakhir

B. DECODER

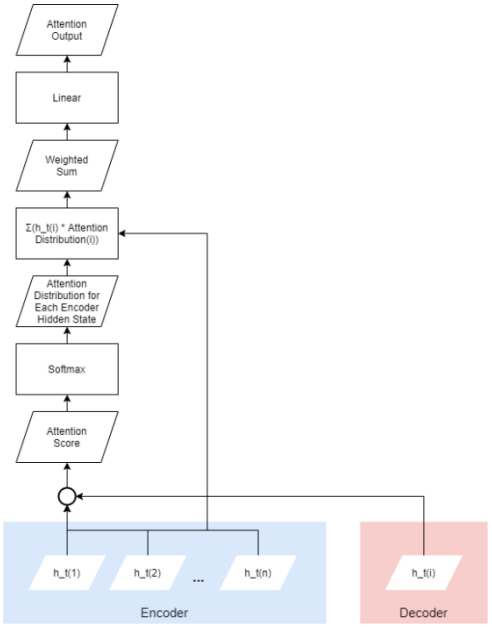
Decoder memiliki cara kerja yang berbeda pada fase *training* dan *inference*. Walaupun demikian, pada fase *training* maupun fase *inference*, *decoder* selalu menerima input yang merupakan output dari layer LSTM pada blok *encoder*. *Decoder* akan menerima *hidden state* dari setiap *timestep* pada LSTM *encoder, hidden state* terakhir, dan juga *cell state* terakhir. Sama seperti pada blok *encoder*, blok *decoder* juga melakukan prosesnya per *batch*.

Pada fase *training, decoder* tidak hanya menerima input berupa *hidden state* dan *cell state* terakhir dari blok encoder, *decoder* juga menerima rangkaian *ground-truth* yang akan diumpankan secara berurutan pada setiap *timestep*. *Ground-truth* yang diumpankan akan melalui proses *embedding* terlebih dahulu agar *ground-truth* dengan relasi yang erat memiliki representasi yang mirip. *Ground-truth* yang telah melalui proses embedding akan diumpankan ke layer LSTM bersama dengan *hidden state* dan *cell state* terakhir dari blok encoder. Output dari LSTM adalah *hidden state* dan *cell state* untuk *timestep* tersebut.



GAMBAR 5. Diagram Alur Proses Seputar LSTM pada Decoder Fase Training Dalam Satu Timestep

Hidden state yang dihasilkan pada suatu *timestep* pada LSTM *decoder* akan "dihubungkan" dengan masing-masing *hidden state* dari LSTM *encoder*. Proses penghubungan inilah yang menjadi inti dari *attention*. Proses penghubungan ini dilakukan dengan perkalian matrix. Hasil dari perkalian matrix ini adalah *attention score*, yaitu bobot yang menentukan seberapa besar keterkaitan antara *ground-truth* dengan *feature* yang terdapat pada *encoder*. *Attention score* ini akan dilewatkan ke *activation function softmax* sehingga diperoleh distribusi *attention* untuk masing-masing *hidden state encoder* dengan jumlah seluruh nilai distribusi tersebut adalah 1. *Hidden state* dari *encoder* dengan keterkaitan yang tinggi dengan *hidden state decoder* pada *timestep* tersebut akan memperoleh distribusi *attention* yang lebih banyak dibanding dengan *hidden state encoder* lainnya.



GAMBAR 6. Arsitektur Attention pada Decoder Fase Training Dalam Satu Semester

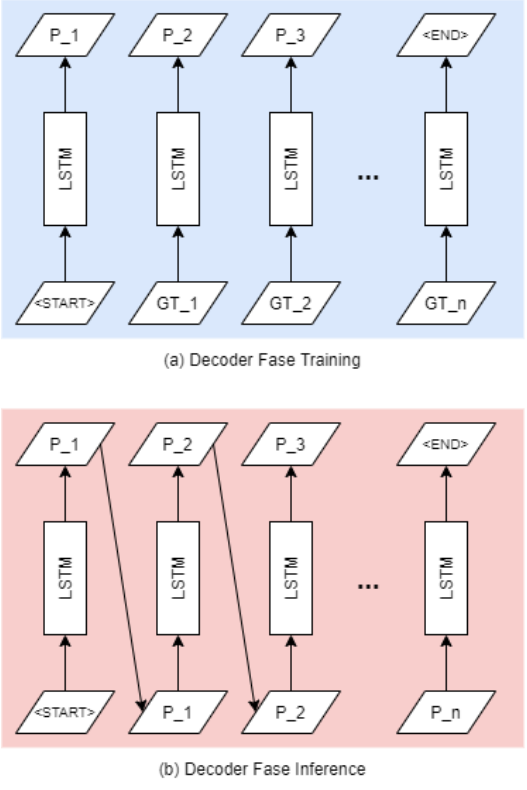
Attention akan diaplikasikan dengan mengalikan distribusi *attention* tersebut dengan masing-masing nilai *hidden state* yang bersangkutan. Dari sinilah akan diperoleh nilai *hidden state* dari setiap *timestep* pada fase *encoder* yang telah memiliki bobot (*weighted sum*). Nilai ini akan dilewatkan pada layer *linear* untuk melakukan *mapping* untuk menyesuaikan ukurannya tanpa menghilangkan informasi kontekstual yang dikandungnya. Hasil dari *linear* layer ini adalah *attention* output, yaitu prediksi untuk *timestep* yang bersangkutan. Prediksi dari masing-masing *timestep* akan dibandingkan dengan *ground-truth* pada perhitungan *loss*.

Pada pengerjaan penelitian ini, terdapat banyak label yang memungkinkan untuk masing-masing objek dengan masing-masing objek hanya dapat memiliki satu class saja. Dengan kasus seperti ini, *loss function* yang digunakan adalah *categorical cross-entropy*. Rumus untuk *categorical cross-entropy* adalah sebagai berikut:

$$Loss = -\sum_{i=1}^{n} y_i \cdot \log \hat{y}_i \qquad (2)$$

Dengan y_i adalah ground-truth dan $\hat{y}_i$ adalah output attention (hasil prediksi) pada *sequence* ke-i. Karena akan terdapat sequence sepanjang n, maka akan terbentuk vektor sepanjang n dengan masing-masing nilainya merupakan nilai *loss* per object. Untuk memperoleh nilai *loss* final, diambil nilai rata-rata dari seluruh loss pada vektor tersebut. *Decoder* pada fase inference masih menggunakan layer LSTM yang sama seperti LSTM pada *decoder* fase *inference*. Kegunaan dari layer LSTM pada decoder fase *inference* juga masih sama seperti pada fase *training*, yaitu untuk melakukan proses *decoding* dari *feature* yang diumpankan.

Pada fase *inference*, proses yang dilakukan tidak jauh berbeda. Hal yang membedakan adalah LSTM pada fase *inference* dari fase training menerima input yang berbeda. Layer LSTM pada *decoder* fase *inference* menerima input berupa prediksi dari *timestep* sebelumnya, *hidden state* terakhir dari *encoder*, dan *cell state* terakhir dari *encoder*. Prediksi dari *timestep* sebelumnya akan melalui proses *embedding* terlebih dahulu sebelum diproses lebih lanjut. *Embedding* prediksi *timestep* sebelumnya akan diumpankan ke layer LSTM bersama dengan *hidden state* dan *cell state* terakhir dari blok encoder. Output dari LSTM adalah *hidden state* dan *cell state* untuk *timestep* tersebut.



GAMBAR 7. Ilustrasi Perbandingan Alur Informasi pada Decoder Fase Training dengan Fase Inference

V. UJI COBA

A. EVALUATION METRICS

Untuk menilai apakah model yang dibuat sudah baik atau belum, perlu dilakukan pengukuran seberapa berbeda sequence hasil prediksi model dengan ground-truth-nya. Evaluation metrics yang cocok untuk permasalahan ini adalah *Levenshtein distance, Levenshtein distance* merupakan metrik untuk mengukur perbedaan antara dua sequence. Nilai *Levenshtein distance* adalah jumlah pengubahan (penyisipan, penghapusan, dan penggantian) minimal untuk mengubah suatu *sequence* agar menjadi serupa dengan sequence satunya. *Levenshtein distance* dirumuskan sebagai berikut:

$$lev_{(a,b)} = \begin{cases} max(i,j), & jika\ min(i,j) = 0 \\ min \begin{cases} lev_{(a,b)}(i-1,j)+1 \\ lev_{(a,b)}(i,j-1)+1 \\ lev_{(a,b)}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} \end{cases} \quad (3)$$

Dengan:

- a dan b merupakan masing-masing *sequence*
- i dan j merupakan posisi index pada masing-masing *sequence*

Nilai *Levenshtein distance* masih perlu diproses kembali untuk mengekspresikan seberapa baiknya performa model dalam bentuk persentase. Untuk itu, nilai Levenshtein *distance* akan dinormalisasi dengan panjangnya dan dirata-rata per batch untuk memperoleh *symbol error rate*

B. HASIL UJI COBA

Model dengan arsitektur *attention* pada penelitian ini di-*train* per *mini-batch*. Training dilakukan hingga 16.000 epoch, dalam setiap epoch, satu *mini-batch* berisi 16 data diumpankan. Hasil dari *training* model dapat dilihat pada tabel I

TABEL I
HASIL UJI COBA ARSITEKTUR ATTENTION PER 2.000 EPOCH

Model	Epoch	Loss	Symbol Error Rate (%)		
			Train	Test	Val
Attention-A	2.000	0.87	56.67%	57.11%	56.63%
Attention-B	4.000	0.46	26.29%	27.38%	26.79%
Attention-C	6.000	0.27	20.13%	21.59%	21.10%
Attention-D	8.000	0.28	15.50%	16.95%	16.79%
Attention-E	10.000	0.20	13.08%	14.72%	14.51%
Attention-F	12.000	0.13	11.58%	13.43%	13.18%
Attention-G	14.000	0.11	8.30%	10.14%	9.86%
Attention-H	16.000	0.07	7.94%	9.68%	9.57%

Perbedaan tiap arsitektur pada table 1 terletak pada jumlah training(epoch).Berdasarkan tabel di atas, dapat dilihat bahwa model Attention-A memiliki nilai loss yang besar, yaitu 0.87. Ketika model tersebut dievaluasi dengan digunakan untuk memprediksi data pada dataset, diperoleh hasil *symbol error rate* pada data di dataset training dengan nilai 56.67%. *Symbol error rate* yang diperoleh model Attention-A ketika memprediksi data pada *test set* sedikit lebih tinggi, yaitu 57.11%. Hal yang menarik ditemukan ketika model tersebut digunakan untuk memprediksi data pada *validation set*, karena *symbol error rate* yang diperoleh justru sedikit lebih rendah dibanding dengan *symbol error rate* pada data training.

Model Attention-B yang di-training dengan 4.000 epoch menghasilkan nilai *loss* yang jauh lebih rendah dibanding dengan model Attention-A. Loss yang dihasilkan mengecil hingga kurang lebih setengah dari *loss* pada model sebelumnya. Rata-rata *symbol error rate* pada model Attention-B pun juga mengecil hingga sekitar setengah dari Attention-A. Pada Attention-B, tetap terlihat bahwa hasil evaluasi pada test set memiliki *symbol error rate* tertinggi. Walaupun demikian, hasil evaluasi pada *train set* dalam Attention-B memiliki *error rate* yang lebih kecil dibanding pada *validation set*.

Model Attention-C di-training dengan 6.000 epoch dan menghasilkan nilai loss yang secara signifikan lebih rendah dibanding dengan Attention-A dan Attention-B. Ketika dievaluasi, *symbol error rate* juga ikut mengecil pada ketiga bagian dari dataset yang dicoba. Hasil evaluasi dengan train set pada Attention-C menghasilkan *symbol error rate* sekitar 1% lebih rendah dibanding dengan set lainnya.

Model Attention-D di-training dengan 8.000 epoch. Walaupun nilai *loss* yang dihasilkan sedikit lebih tinggi dibanding dengan Attention-C, *symbol error rate* yang muncul dalam tahap evaluasi tetap mengecil hingga sekitar 5% dari Attention-C. Tetap terlihat bahwa hasil evaluasi pada train set memiliki hasil terbaik dengan nilai *symbol error rate* terkecil dan hasil evaluasi pada *test set* memperoleh hasil terburuk.

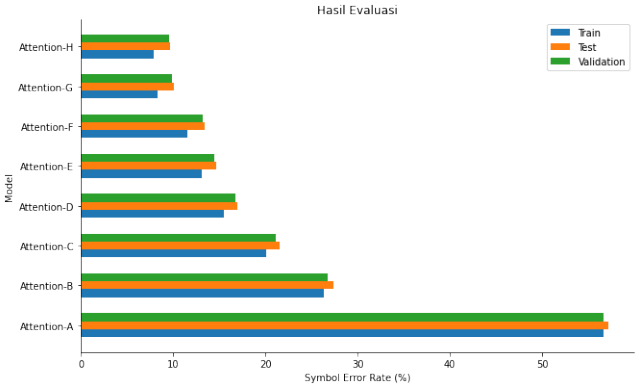
Pada model Attention-E yang di-training dengan 10.000 epoch, nilai loss terlihat mengecil. Hasil evaluasi pada train, test, dan validation set juga turut mengecil walaupun penurunan *symbol error rate* sudah tidak sesignifikan sebelumnya. Hasil evaluasi pada *train set* tetap mengungguli set lainnya, perbedaan tersebut menjadi semakin besar, yaitu sekitar 1,5% dibanding dengan *test set*.

Model Attention-F yang di-training dengan 12.000 epoch menghasilkan nilai *loss* yang semakin mengecil. Walaupun demikian, penurunan *symbol error rate* pada tahap evaluasi tidak begitu besar. Ketiga set rata-rata menghasilkan *symbol error rate* sekitar 12%. *Test set* tetap menghasilkan *symbol error rate* tertinggi, diikuti dengan *validation set* dan *train set.*

Model Attention-G menghasilkan *loss* yang agak mengecil dibanding dengan model sebelumnya setelah di-train dengan 14.000 epoch. Pada model Attention-G, hasil evaluasi yang diperoleh terlihat membaik dibanding model-model sebelumnya dengan rata-rata sekitar 9%.

Model Attention-H melalui proses training dengan 16.000 epoch. Karena *loss* yang dihasilkan dianggap sudah cukup kecil, yakni kurang dari 0,1, dan penurunan *loss* sudah tidak begitu signifikan yang berarti *loss* sudah cukup konvergen,

maka model inilah yang digunakan. Ketika dilakukan tahapan evaluasi, terlihat pula bahwa *symbol error rate* pada masing-masing set juga menurun dibanding dengan hasil evaluasi untuk model Attention G. Perbedaan *symbol error rate* antara *train set* dan *test set* terlihat semakin membesar pula, yakni hampir mencapai 2%



GAMBAR 8. Grafik Visualisasi Hasil Evaluasi untuk Masing-Masing Model Attention

Dari seluruh uraian di atas, dapat ditarik kesimpulan sementara bahwa performa model meningkat pada jumlah epoch yang lebih banyak. Di antara model Attention-A yang memiliki 2.000 epoch dan Attention-B yang memiliki 4.000 epoch, terdapat penurunan *symbol error rate* yang signifikan. Penurunan terus terjadi di antara model-model selanjutnya hingga Attention-H. Di antara model Attention-G dan Attention-H, masih terlihat adanya penurunan yang sangat kecil sehingga disimpulkan bahwa model sudah cukup konvergen. Hasil evaluasi pada train set hampir selalu memberikan *symbol error rate* terendah, hal ini wajar mengingat model di-*train* dengan menggunakan *train set*

VI. KESIMPULAN

Arsitektur CRNN menghasilkan *symbol error rate* yang sangat tinggi, yaitu 84,8%. Walaupun demikian, nilai *loss* pada awal training hingga akhir proses training mengalami penurunan yang signifikan.

Nilai loss pada arsitektur CRNN-CRF dalam proses training sangat besar dan tidak mengalami penurunan setelah di-*train* dalam 4.000 epoch hingga *training* akhirnya dihentikan. Dari sini dapat disimpulkan bahwa arsitektur ini tidak cocok dalam menangani permasalahan ini.

Arsitektur *Attention* menghasilkan nilai loss terus yang menurun hingga batas tertentu pada proses training. Ketika dievaluasi, *symbol error rate* yang dihasilkan berkisar antara 10%. Dengan demikian dapat disimpulkan bahwa arsitektur *Attention* merupakan arsitektur yang paling cocok untuk menangani permasalahan ini dibanding dengan kedua arsitektur lainnya.

Skew-correction yang diterapkan untuk menangani input pengguna sebelum diproses oleh model dapat berjalan dengan baik jika warna kertas kontras dengan warna latar belakangnya. *Skew-correction* juga berjalan optimal apabila tidak terdapat motif pada latar belakang gambar yang dapat mengganggu proses deteksi kertas yang terdapat dalam *skew-correction*.

Pemotongan gambar partitur per baris dilakukan dengan melakukan dilasi berulang kali terhadap hasil *edge-detection* pada gambar. Proses ini tidak dapat berjalan optimal terdapat bagian dari partitur yang blur karena dapat mempengaruhi proses *edge-detection*. Selain itu, *noise* yang terdapat pada gambar juga dapat memperburuk performa dari pemotongan ini karena dapat ikut terdeteksi sebagai baris partitur setelah hasil *edge-detection* didilasi.

File MIDI yang dihasilkan oleh program dapat memainkan nada-nada dengan ketukan yang benar sesuai hasil prediksi. Hasil yang baik ini dapat diperoleh dengan mudah mengingat format label yang seragam sehingga mempermudah proses parsing dan konversi ke file MIDI.

Seperti halnya pada file MIDI, not angka yang dihasilkan juga sesuai dengan hasil prediksi model, output model setelah melalui *postprocessing* adalah label-label dengan format yang seragam sehingga mempermudah proses *parsing* dan konversi ke not angka

COPYRIGHT

DAFTAR PUSTAKA

[1] Jorge Calvo-Zaragoza, Jan Hajič Jr., and Alexander Pacha. 2020. Understanding Optical Music Recognition. ACM Comput. Surv. 53, 4, Article 77 (September 2020)

[2] Jorge Calvo-Zaragoza, & David Rizo. (2018). Camera-PrIMuS: Neural End-to-End Optical Music Recognition on Realistic Monophonic Scores. Proceedings of the 19th International Society for Music Information Retrieval Conference, 248–255

[3] David Bainbridge and Tim Bell. The challenge of optical music recognition. Computers and the Humanities, 35(2):95–121, 2001.

[4] Ana Rebelo, Ichiro Fujinaga, Filipe Paszkiewicz, Andre R.S. Marcal, Carlos Guedes, and Jamie dos Santos Cardoso. Optical music recognition: state-of-the-art and open issues. International Journal of Multimedia Information Retrieval, 1(3):173–190, 2012.

[5] J. Calvo-Zaragoza and D. Rizo. End-to-End Neural Optical Music Recognition of Monophonic Scores. Applied Sciences, 8(4):606–629, 2018.

[6] J. Hajic Jr. and P. Pecina. Detecting Noteheads ˇ in Handwritten Scores with ConvNets and Bounding Box Regression. Computing Research Repository, abs/1708.01806, 2017

[7] A. Pacha, K.-Y. Choi, B. Couasnon, Y. Ricquebourg, ¨ R. Zanibbi, and H. Eidenberger. Handwritten music object detection: Open issues and baseline results. In 13th IAPR Workshop on Document Analysis Systems, pages 163–168, 2018

[8] E. van der Wel and K. Ullrich. Optical music recognition with convolutional sequence-to-sequence models. In 18th International Society for Music Information Retrieval Conference, pages 731–737, 201

[9] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In International Conference on Learning Representations, San Diego, California, USA

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin.2017.Attention Is All You Need.In 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.
[11] Y. Lecun,L. Bottou,Y. Bengio,P. Haffner.1995.Gradient-based learning applied to document recognition.In Proceedings of the IEEE 86(11):2278 – 2324
[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun.2016.Deep Residual Learning for Image Recognition.2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
[13] Minh-Thang Luong, Hieu Pham, Christopher D. Manning.2015.Effective Approaches to Attention-based Neural Machine Translation.In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing
[14] Ilya Sutskever, Oriol Vinyals, Quoc V. Le.2014.Sequence to Sequence Learning with Neural Networks.In Neural Information Processing Systems (NIPS 2014)
[15] Sepp Hochreiter,Jurgen Schmidhuber.1997.Long Short-Term Memory.Neural Computation 9(8):1735-1780, 1997
[16] Sergey Ioffe, Christian Szegedy.2015.Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.In ICML'15: Proceedings of the 32nd International Conference on International Conference on Machine Learning pages 448-456
[17] Razvan Pascanu, Tomas Mikolov, Yoshua Bengio.2013.On the difficulty of training Recurrent Neural Networks.In ICML'13: Proceedings of the 30th International Conference on International Conference on Machine Learning
[18] Karen Simonyan, Andrew Zisserman.2014.Very Deep Convolutional Networks for Large-Scale Image Recognition.In International Conference on Learning Representations(ICLR) 2015
[19] Kunihiko Fukushima.1980.Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. In Biological Cybernetics volume 36, pages193–202 (1980)
[20] Vinod Nair, Geoffrey E. Hinton.2010.Rectified Linear Units Improve Restricted Boltzmann Machines. In ICML'10: Proceedings of the 27th International Conference on International Conference on Machine Learning

Deteksi Aspek Review E-Commerce Menggunakan IndoBERT Embedding dan CNN

Syaiful Imron[1], Esther Irawati Setiawan[1], dan Joan Santoso[1]
[1]Departemen Teknologi Informasi, Fakultas Sains dan Teknologi, Institut Sains dan Teknologi Terpadu Surabaya, Surabaya, Indonesia

Corresponding author: Esther Irawati Setiawan (esther@istts.ac.id)

ABSTRACT With the development of information technology, the term e-commerce appears in the business world. In e-commerce there is a review feature, customers can provide reviews in the form of text, images, and stars. These reviews are opinions from customers regarding the items purchased. But in most e-commerce, there are no category features related to reviews, this makes it difficult for potential buyers to analyze manually. Aspect-based sentiment analysis (ABSA) is a solution to this problem. ABSA has three tasks, one of which is Aspect Category Detection which aims to classify customer reviews into several aspects where the aspects have been predetermined. Quite a lot of research related to aspect category detection using machine learning. Of the several methods tested, the Convolutional Neural Network (CNN) is the best method. In addition, the use of BERT as word embedding produces better results than conventional word embedding. This research uses a dataset from Bukalapak e-commerce with 3114 reviews and 6 aspects (Accuracy, Delivery, Quality, Price, Packaging, and Service). Based on trials using IndoBERT as word embedding and CNN for aspect detection, an accuracy of 94.86% was obtained. Thus, the model can be used for detection aspects. In addition, the CNN method gets better results than the LSTM method.

KEYWORDS Aspect Category Detection, CNN, IndoBERT, Reviews.

ABSTRAK Dengan semakin berkembangnya teknologi informasi, maka muncul istilah *e-commerce* dalam dunia bisnis. Pada *e-commerce* ada fitur *review*, pelanggan dapat memberikan *review* berupa teks, gambar, dan bintang. *Review* tersebut merupakan opini dari pelanggan terkait barang yang dibeli. Tetapi pada kebanyakan *e-commerce* tidak ada fitur kategori terkait *review* hal ini membuat calon pembeli kesusahan dalam menganalisa secara manual. *Aspect-based sentiment analysis* (ABSA) merupakan solusi dari permasalahan tersebut. ABSA memiliki tiga tugas salah satunya *Aspect Category Detection* yang memiliki fungsi untuk menggabungkan *review* pelanggan menjadi beberapa aspek dimana aspek-aspek tersebut sudah didefinisikan terlebih dahulu. Cukup banyak penelitian terkait *Aspect Category Detection* dengan mengunakan *machine learning*. Dari beberapa metode yang diuji, *Convolutional Neural Network* (CNN) merupakan metode terbaik. Selain itu penggunaan BERT sebagai *word embedding* menghasilkan output yang bagus baik dari pada *word embedding* konvensional. Penelitian ini menggunakan dataset dari *e-commerce* Bukalapak dengan 3114 review dan 6 aspek (Akurasi, Pengiriman, Kualitas, Harga, Pengemasan, dan Pelayanan). Berdasarkan ujicoba dengan menggunakan IndoBERT sebagai *word embedding* dan CNN untuk deteksi aspek, maka didapatkan akurasi sebesar 94,86%. Dengan demikian model tersebut dapat digunakan untuk deteksi aspek. Selain itu, metode CNN mendapatkan hasil yang lebih baik dari pada metode LSTM.

KATA KUNCI Aspect Category Detection, CNN, IndoBERT, Review.

I. PENDAHULUAN

Teknologi informasi saat ini mudah diperoleh karena teknologi informasi telah merasuk ke seluruh lapisan masyarakat Indonesia. Teknologi informasi berkembang dan mengalami perubahan yang membawa banyak kemudahan bagi kehidupan masyarakat. Saat ini terjadi sinergi antara teknologi informasi dan bisnis yang tidak dapat dipisahkan dengan mudah, perpaduan antara teknologi informasi dan

bisnis memunculkan istilah *e-commerce* [1]. Pada sistem *e-commerce*, *review* pelanggan menjadi bagian penting bagi kelangsungan bisnis perusahaan. Oleh karena itu, mengembangkan dan mempertahankan konsumen merupakan cara yang menguntungkan perusahaan [2]. Tetapi *review* pelanggan, informasi yang diposting tidak lengkap, hanya berupa teks, gambar, video, bintang dan tidak ada kategori khusus. Untuk itu review pelanggan perlu dilakukan analisis sentimen.

Dalam *Natural Language Processing* (NLP) terdapat salah satu topik yang mempunyai tugas ekstrak dan deteksi teks guna menghasilkan polaritas dari teks tersebut. Dalam suatu teks, polaritas dapat berupa positif, negatif, atau netral. Topik yang dimaksud adalah analisis sentimen [3]. Analisis sentimen terbatas menentukan polaritas hanya pada satu aspek saja. Untuk menentukan aspek dan polaritas sentimen dapat menggunakan *aspect-based sentiment analysis* (ABSA) [4].

ABSA bertujuan untuk ekstrak sebuah kalimat untuk mengetahui aspek dan sentimen. Hal ini guna mendapatkan jumlah aspek dalam kalimat tersebut dan menentukan polaritas sentimen dari setiap aspek. Contoh *review* "barang bagus, tapi pengiriman lama", pada *review* tersebut kalimat "barang bagus" ini menunjukkan konteks "kualitas" yang bernilai "positif". Sedangkan pada kalimat "pengiriman lama" menunjukkan konteks "pengiriman" yang bernilai "negatif".

Penelitian tentang ABSA pada *Semantic Evaluation* 2014 [5], 2015 [6], dan 2016 [7], menjelaskan ada beberapa task yang digunakakan yaitu *Sentiment Polarity Classification*, *Opinion Target Expression*, dan *Aspect Category Detection*. *Aspect Category Detection* berfungsi untuk menggabungkan *review* pelanggan menjadi beberapa aspek dimana aspek-aspek tersebut sudah didefinisikan terlebih dahulu [8].

Cukup banyak penelitian terkait *Aspect Category Detection* dengan menggunakan pendekatan *machine learning*. Beberapa metode yang ada pada *machine learning* antara lain SVM, LSTM dan CNN. Berdasarkan penelitian [9] klasifikasi aspek dengan menggunakan CNN mendapatkan hasil yang paling baik dibandingkan dengan SVM dan LSTM.

BERT adalah teknik pra-pelatihan berbasis jaringan saraf untuk NLP [10]. Selain *Word2Vec* atau *GloVe* BERT dapat digunakan sebagai *word embedding*. Pada penelitian [11] [12] dengan menggunakan BERT sebagai *word embedding* memiliki hasil lebih baik dari pada menggunakan *Word2Vec* atau *GloVe*.

Berdasarkan latar belakang tersebut, belum ada penelitian tentang deteksi aspek dengan menggunakan BERT sebagai *word embedding* dan metode CNN. Oleh karena itu, penelitian ini mengangkat topik tentang deteksi aspek *review* e-commerce dengan menggunakan BERT sebagai *word embedding*, metode CNN untuk deteksi aspek, dan dataset bahasa Indonesia.

II. TINJAUAN PUSTAKA

Pada tahun 2018, [3] melakukan penelitian terkait ekstraksi aspek dengan menggunakan CNN. Model yang dibangun terdiri dari satu *convolutional layer* diikuti *non-linearity*, *max pooling*, dan *fully connected layer*. Untuk mencegah *overfitting* menggunakan *regularization*. Terdapat 2 jenis dataset yang digunakan, yaitu dataset kriket dan dataset restoran. Dataset kriket terdiri dari 6 kategori aspek dan 2958 review. sedangkan dataset restoran terdiri dari 5 aspek dan 2053 review. Penelitian ini juga membandingkan akurasi dari CNN dengan SVM, RF, dan KNN. Setelah dilakukan pengujian terhadap 2 jenis dataset, maka didapatkan hasil bahwa model CNN mendapatkan akurasi yang lebih baik dari SVM, RF, dan KNN.

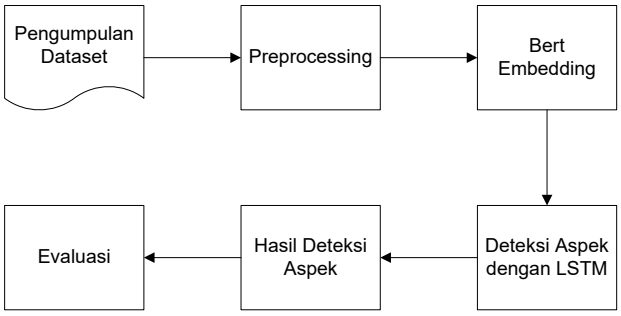
Penelitian tentang *opinion mining* dengan membandingkan dua metode, yaitu LSTM dan CNN dilakukan oleh [13] pada tahun 2018. Untuk model LSTM menggunakan *hidden state* sebanyak 600, sedangkan model CNN untuk *window size* sebanyak 5 dan *feature dimention* sebanyak 300. Kedua model menggunakan *L2 regularization* untuk mengatasi *overfitting*. Sedangkan *word embedding* menggunakan *GloVe2*. Penelitian ini menggunakan *SemEval* 2014 sebagai dataset dengan domain laptop dan restoran. Berdasarkan ujicoba didapatkan bahwa model CNN mendapatkan hasil lebih baik dari model LSTM. Dimana model CNN mendapatkan akurasi sebesar 77,48%. Sedangkan model LSTM sebesar 74,30%. Hal ini disebabkan *convolutional layer* pada CNN memiliki keunggulan dalam mengekstraksi fitur lokal suatu teks.

Selanjutnya penelitian tentang ekstraksi aspek dilakukan oleh [14]. Pada penelitian tersebut menggunakan CNN. Model yang dibangun terdiri dari dua *convolutional layer*, dua *max pooling*, dan *fully connected layer* dengan *softmax*. Penelitian ini juga membandingkan *word2vec* dan CBOW sebagai *word embedding*. *Part of speech tags* (POS) digunakan sebagai fitur tambahan. Fitur POS yang digunakan adalah *noun, verb, djective, adverb, preposition,* dan *conjunction.* Dataset yang digunakan adalah *SemEval* 2014. Hasil penelitian dengan menggunakan CBOW sebagai *word embedding*, mendapatkan akurasi yang lebih baik daripada menggunakan *word2vec*. Sedangkan model CNN yang menggunakan POS sebagai fitur tambahan mendapatkan akurasi yang lebih baik dari pada model CNN tanpa POS.

III. METODOLOGY

Pada tahap ini, menjelaskan arsitektur dan metode yang digunakan. Desain arsitektur pada penelitian deteksi aspek ini ada 6 proses yaitu pengumpulan dataset, *preprocessing*, *word embedding*, deteksi aspek, pengujian dan evaluasi. Proses pengumpulan data *review* dari *e-commerce* bukalapak dilakukan dengan cara *scrapping*. Selanjutnya dilakukan *preprocessing.* Tahap selanjutnya *word embedding* dengan menggunakan BERT. Proses selanjutnya yaitu deteksi aspek menggunakan CNN. Terakhir melakukan pengujian dan evaluasi menggunakan *confussion matrix*. Gambar 1

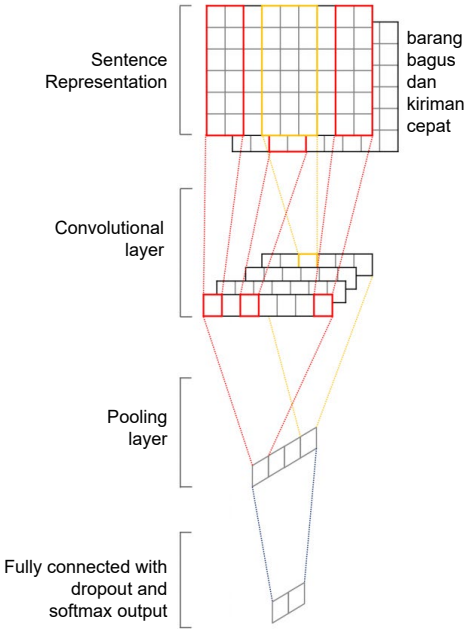
merupakan tahapan proses penelitian deteksi aspek dengan menggunakan BERT dan CNN.



GAMBAR 1. Arsitektur Penelitian Ekstraksi Aspek

A. CNN

Pada awalnya, *convolutional neural network* (CNN) sering dipergunakan untuk memproses gambar. Namun pada tahun 2015, [15] melakukan penelitian tentang penggunaan CNN untuk klasifikasi teks. Arsitektur CNN yang digunakan untuk klasifikasi teks antara lain representasi kalimat, *convolutional layer*, *max-pooling layer*, *fully connected layer, droupout* dan *softmax*. Untuk arsitektur CNN dapat dilihat pada Gambar 2.



GAMBAR 2. Arsitektur CNN

Berikut ada penjelasan terkait dengan arsitektur CNN dalam Gambar 2:

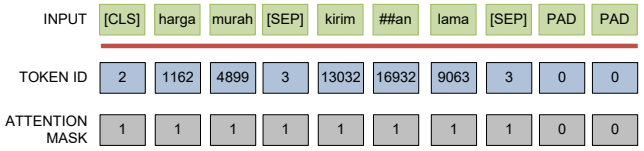
1) Representasi teks
   Pada tiap *input* CNN, kalimat harus dalam bentuk vektor. Untuk merubah menjadi sebuah vektor, maka diperlukan *word embedding*. Selanjutnya vektor tersebut dapat dipergunakan untuk *input* ke dalam *convolutional layer*.
2) *Convolutional layer*
   Dalam *convolutional layer* dilakukan proses *convolution* dengan menggunakan sebuah *filter* atau biasa disebut *slidding window*. Cara kerja *filter* atau *slidding window* adalah memecah representasi kata dari *matriks* vektor kedalam beberapa *window*. Dalam proses *convolutional layer* akan menghasilkan *feature map*.
3) *Pooling layer*
   Dalam proses *pooling layer* terdapat beberapa metode, salah satunya yaitu *max-pooling layer*. *Max-pooling* bekerja dengan cara mengambil nilai paling tinggi dari *feature map*.
4) *Fully connected layer*
   Pada *layer* ini semua hasil dari *max-pooling* dengan menggunakan *activation function* dan menghasilkan representasi baru dari kalimat.
5) *Dropout*.
   Droupout digunakan untuk mengatasi agar tidak terjadi *overfitting*.
6) *Optimization*
   Untuk menurunkan nilai *loss* dari sebuah model selama proses *training* maka diperlukan *optimization*. Ada beberapa jenis *optimizer* yang dapat dipakai antara lain, *Adadelta* [16], *Adam* [17], *Nadam* [18] dan *SGD* [19].
7) *Sigmoid*.
   Teknik ini digunakan apabila output hanya berupa 2 buah *class*.
8) *Softmax*.
   Teknik ini digunakan pada penelitian multiclass *classification*. Hasil dari *fully connected layer* diproses *softmax* dengan cara mengambil probabiltas terbesar untuk menentukan class.

B. WORD EMBEDDING

Dalam bidang NLP, ada sebuah teknik pembelajaran yaitu w*ord embedding*. Dalam proses *word embedding*, setiap kata akan dirubah menjadi kumpulan vektor angka. Pada penelitian ini *word embedding* menggunakan BERT. Untuk inputan dilakukan penambahan *special token* [CLS] di awal kalimat, sedangkan [Sep] dipenghubung dan diakhir kalimat.

Pertama kalimat *input* yang sudah ditambah *spesial token*, dilakukan tokenisasi. Tokenisasi adalah proses merubah kalimat input menjadi *token* berdasarkan *vocabulary*. Apabila terdapat kata yang tidak ada dalam *vocabulary*, kata tersebut akan diganti menjadi kata dasar yang terdapat pada *vocabulary* dan diberi tanda (#) pada bagian yang tidak diketahui [20].

Hasil tokenisasi selanjutnya dirubah menjadi *token id* dan *attention mask*. *Token id* dan *attention mask* perlu dirubah menjadi *tensor*. Untuk itu merubah data menjadi *tensor* pada penelitian ini menggunakan model *indoBERT-base-p1* yang didapat dari *library huggingface transformer* [21] [22]. Untuk contoh BERT *embedding* dapat dilihat pada Gambar 3.



GAMBAR 3. BERT Embedding

C. CONFUSSION MATRIX

Untuk mengetahui *performace* sebuah model maka perlu dilakukan evaluasi. Salah satu metode evaluasi adalah *confusion matrix*. Dalam *confusin matrix* terdapat beberapa metrix, antara lain akurasi, *recall*, *precision*, dan *F1-score*.

1) Akurasi.
Akurasi menggambarkan akurasi dari sebuah model dalam mengklasifikasikan dengan benar.

$$\text{Akurasi} = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

2) *Recall*.
Recall menggambarkan hasil prediksi yang benar dengan jumlah data asli yang bernilai sama.

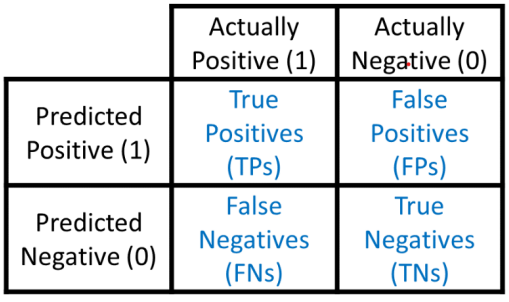
$$Recall = \frac{TP}{TP+FN} \quad (2)$$

3) *Precision*
Precision menggambarkan hasil prediksi yang benar dengan total jumlah hasil prediksi yang bernilai sama.

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

4) *F1-Score*
F1-Score menggambarkan perbandingan rata-rata precision dan recall yang dibobotkan.

$$F1\text{-}Score = \frac{2TP}{2TP+FP+FN} \quad (3)$$



GAMBAR 4. Confusion Matrix

IV. UJICOBA DAN PEMBAHASAN

Dalam tahap ini akan membahas proses terkait dengan penelitian, meliputi *prepocessing*, *word embedding*, *training* model, *testing* model dan *evaluasi*.

A. DATASET

Pada penelitian ini menggunakan dataset *review* dari e-*commerce* Bukalapak. Hanya menggunakan *review* yang berupa teks deskriptif dalam Bahasa Indonesia. Selanjutnya pengelompokan kategori aspek mengikuti penelitian [23]. Dimana ada enam aspek yaitu akurasi, kualitas, pelayanan, harga, pengemasan, dan pengiriman. Dataset yang didapat sebanyak 3,114 *review* dimana distribusi dataset tersbut dapat dilihat dalam Tabel I.

TABEL I
DISTRIBUSI DATASET

Aspek	Jumlah
Kualitas	1124
Pengiriman	592
Akurasi	504
Pengemasan	312
Harga	302
Pelayanan	280
Total	3114

Sebelum dilakukan proses labeling, ada beberapa *review* yang dihapus dari dataset dikarenakan *review* tersebut tidak terkait dengan barang yang dibeli oleh pelanggan. Sehingga *review* yang digunakan merupakan *review* yang terkait dengan barang yang dibeli oleh pelanggan. Sehingga *review* sesuai dengan kategori aspek yang ada. Contoh *review* yang dihapus dapat dilihat dalam Tabel II.

TABEL II
CONTOH REVIEW

Review
mantap om, makasih ya...
harga kualitas fungsi produk
awal nya takut dapat seperti yg ada di bintang 1 or 2
Belum dibuka

Dalam penelitian ini metode yang akan digunakan untuk melakukan deteksi aspek adalah *supervised learning*, untuk itu diperlukan proses labeling untuk dataset. Pelabelan aspek *review* dilakukan secara manual. Contoh *review* yang sudah diberi label aspek dapat dilihat dalam Tabel III.

TABEL III
CONTOH REVIEW

Review	Aspek
respon sangat cepat begitu pesan proses dikirim recomendasi baik seler	Pelayanan
Barang sudah diterima dengan packing yg rapi	Pengemasan
Barang baru dipake udah gabisa hidup lagi di ganti baterai lain juga tetep gabisa hidup	Kualitas
Harga murah tp bukan barang murahan	Harga
Seller kalau di chat balasnya sangat lama, bagaimana kalau mau komplain	Pelayanan
Barang yang dikirim sesuai dengan gambar yang ada	Akurasi

B. PREPROCESSING

Dataset sebelum diproses lebih lanjut, terlebih dahulu dilakukan *preprocessing*. Dalam tahap *preprocessing* ini

terdapat 4 proses, yaitu *case folding, punctuation removal, word normalization,* dan *stemming*.

1) *Case folding*
   Case folding merupakan proses merubah semua huruf menjadi huruf kecil atau huruf besar. Hanya huruf 'A' sampai huruf 'Z' yang diubah.
2) *Punctuation removal*
   Punctuation removal merupakan proses menghapus karakter selain huruf antara lain menghapus tag HTML, menghapus tanda baca, menghapus *non-ascii*, dan menghapus spasi berlebih.
3) *Word normalization*
   Word normalization dilakukan untuk normalisasi kata kata yang disingkat, kata yang mengalami kesalahan penulisan, dan kata yang tidak sesuai dengan KBBI.
4) *Stemming*
   Stemming proses untuk menghilangkan imbuhan, awalan, dan akhiran dari sebuah kata. *Stemming* dalam penelitian ini menggunakan library Sastrawi.

C. PEMBAGIAN DATA TRAINING DAN TESTING

Sebelum dilakukan *word embedding*, dataset terlebih dahulu dibagi menjadi data *training* dan data *testing*. Data *training* berfungsi menganalisa pola dalam membangun model. Data *testing* berfungsi untuk mengukur akurasi dari model yang telah dibangun. Untuk pembagian dataset, 80% data *training* dan 20% data *testing* dengan menggunakan *library sklearn*.

D. WORD EMBEDDING

Training *word embedding* menggunakan BERT sebagai *word embedding*. Penelitian ini menggunakan *IndoBERT* model *IndoBERT-base-p1* yang menggunakan indo4B sebagai dataset. Panjang *word embedding* dalam penelitian ini 128 token. Gambar 4 merupakan hasil *word embedding*. Hasil dari *word embedding* menggunakan BERT adalah *token id* dan *attention mask*.

```
[array([[   2, 5040,  972, ...,    0,    0,    0],
        [   2, 3511,   27, ...,    0,    0,    0],
        [   2, 8302, 2279, ...,    0,    0,    0],
        ...,
        [   2, 7424, 2659, ...,    0,    0,    0],
        [   2,  963,  786, ...,    0,    0,    0],
        [   2, 2885, 1107, ...,    0,    0,    0]],
        [1, 1, 1, ..., 0, 0, 0],
        [1, 1, 1, ..., 0, 0, 0],
        ...,
        [1, 1, 1, ..., 0, 0, 0],
        [1, 1, 1, ..., 0, 0, 0],
        [1, 1, 1, ..., 0, 0, 0]], dtype=int32)]
```

GAMBAR 5. Hasil IndoBERT

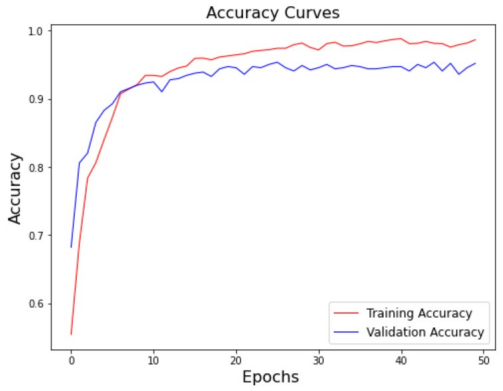
E. PENGUJIAN MODEL

Dalam tahap ini membahas pengujian model yang sudah dibangun. *Parameter* yang gunakan dapat dilihat pada Tabel IV. *Parameter* ini didapat dilakukan setelah melakukan beberapa kali ujicoba.

TABEL IV
PARAMETER

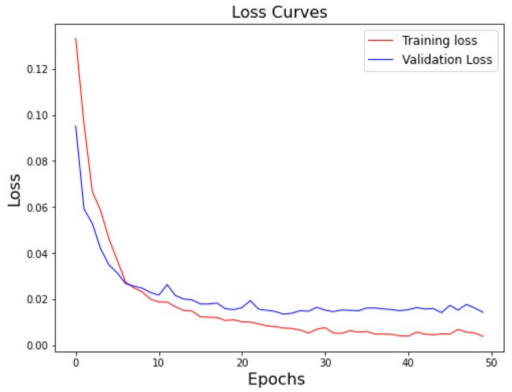
Parameter	Value
Filter Size	300
Feature Maps	5
Regularizer L1	0,1
Dropout	0,5
Epoch	50
Batch Size	32
Learning Rate	1e-3
Optimizer	Adam

Setelah dilakukan pengujian terhadap dataset *review* bukalapak dengan enam aspek (pengemasan, akurasi, kualitas, pengiriman, harga, dan pelayanan), deteksi aspek dengan model CNN mendapatkan akurasi sebesar 94,86%, *precision* 95,32%, *recall* 94,86%, dan *F1-score* 95,09%. Untuk akurasi, *precision, recall,* dan *F1-score* nilai yang digunakan adalah *weighted average*. Gambar 5 merupakan grafik validasi akurasi deteksi aspek menggunakan model CNN. Pengujian pada epoch pertama sampai epoch ke 10, akurasi masih dibawah 90%. Tetapi mulai epoch ke 10 sampai epoch ke 50, akurasi stabil di atas 90%.



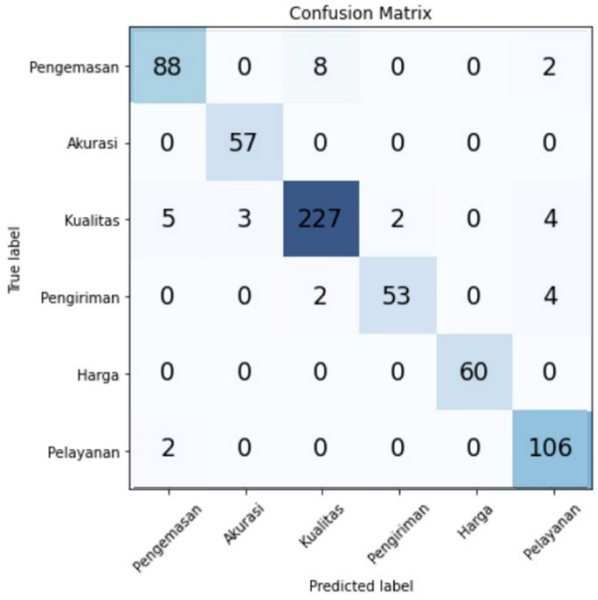
GAMBAR 6. Grafik Akurasi

Pada gambar 6, merupakan grafik validasi *loss* karena BERT *embedding* dan model CNN untuk deteksi aspek. Pada grafik valisasi *loss* deteksi aspek, nilai loss dari data *train* lebih kecil dari pada data *test*.



GAMBAR 7. Grafik Loss

Selanjutnya menggunakan *confusion matrix* guna melakukan pengecekan hasil pengujian. Berdasarkan pada gambar 7, hasil aspek pengemasan: 88, akurasi: 57, kualitas: 227, pengiriman: 53, harga: 60, dan pelayanan: 106. Ada beberapa data review yang tidak sesuai dengan aspeknya. Sebagai contoh, ada 2 review tentang aspek kualitas tetapi terdeksi sebagai pengiriman.



GAMBAR 8. Confusion Matrix

Setelah model selesai dibangun dan dilakukan evaluasi, selanjutnya dilakukan pengujian *review*. Hal ini bertujuan untuk mengetahui *review* tersebut diprediksi secara benar atau salah oleh model. Untuk hasil pengujian *review* dapat dilihat pada Tabel V. Ada beberapa *review* yang salah prediksi, hal ini karena ada beberapa kata yang mirip atau sama antar aspek.

TABEL V
HASIL PENGUJIAN REVIEW

Review	Aspek	Aspek Prediksi
barang baru dipake udah gabisa hidup lagidi ganti baterai lain juga tetep gabisa hidup	Kualitas	Kualitas
respon sangat cepat begitu pesan proses dikirim	Pelayanan	Pengiriman
Namun akan lebih baik apabila pengiriman dipercepat	Pengiriman	Pengiriman
harga murah tp bukan barang murahan	Harga	Harga
barang bagus sdh diterima dgn packing yg rapi	Pengemasan	Kualitas

Selain itu penelitian ini juga membandingkan metode CNN dan metode LSTM. Hasil pengujian perbandingan metode CNN dan metode LSTM dapat dilihat pada Tabel VI. Berdasarkan pengujian bahwa metode CNN mendapatkan hasil lebih baik daripada metode LSTM. Deteksi aspek dengan menggunakan metode CNN mendapatkan akurasi sebesar 94,86% sedangkan menggunakan metode LSTM mendapatkan akurasi sebesar 88,92%. Kedua model, untuk nilai yang digunakan adalah *weighted average.*

TABEL VI
PERBANDINGAN METODE

Aspek	Precision	Recall	F1-Score	Akurasi
CNN	95,32%	94,86%	95,09%	94,86%
LSTM	90,01%	88,20%	88,90%	88,92%

V. KESIMPULAN

Setelah dilakukan pengujian dan evaluasi, penelitian ekstraksi aspek dengan menggunakan BERT sebagai word embedding dan metode CNN untuk ekstraksi aspek mendapatkan hasil yang sangat baik, yaitu akurasi sebesar 94,86%. Sedangkan deteksi aspek dengan menggunakan metode LSTM, hanya mendapatkan akurasi sebesar 88,92%. Untuk itu dapat diambil kesimpulan bahwa metode CNN lebih baik dari pada metode LSTM untuk deteksi aspek.

Pada penelitian ini penyebaran dataset antar aspek tidak merata, hal ini dapat mempengaruhi akurasi sebuah model. Diharapkan pada penelitian yang akan datang penyebaran dataset antar aspek agar merata. Disamping itu, penelitian yang akan datang dapat menggunakan BERT sebagai ekstraksi aspek, tidak hanya sebagai word embedding. Penelitian yang dilakukan penulis adalah *single label classification*, diharapkan dimasa mendatang dapat menggunakan penelitian *multi label classification*.

DAFTAR PUSTAKA

[1] D. Hendarsyah, "E-Commerce Di Era Industri 4.0 Dan Society 5.0," Iqtishaduna, vol. 8, no. 2, pp. 171–184, Dec. 2019, 10.46367/iqtishaduna.v8i2.170.

[2] R. P. Ananda and A. Yuniawan, "Studi Empiris Kepuasan Pelanggan E-Commerce Secara Global," SLJIL, vol. 6, no. 7, p. 3499, Jul. 2021, 10.36418/syntax-literate.v6i7.3541.

[3] Md. A. Rahman and E. Kumar Dey, "Aspect Extraction from Bangla Reviews using Convolutional Neural Network," in 2018 Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), Kitakyushu, Japan, Jun. 2018, pp. 262–267, 10.1109/ICIEV.2018.8641050.

[4] T. Alvarez-López, J. Juncal-Martínez, M. Fernández-Gavilanes, E. Costa-Montenegro, dan F.J. González-Castã, "SVM and CRF for Aspect Detection and Unsupervised Aspect-Based Sentiment Analysis," Proc. 10th Int. Work. Semant. Eval., 2016, hal. 306–311.

[5] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, dan S. Manandhar, "SemEval-2014 Task 4: Aspect Based Sentiment Analysis," Proc. 8th Int. Work. Semant. Eval., 2015, hal. 27–35.

[6] H. Papageorgiou, I. Androutsopoulos, D. Galanis, M. Pontiki, dan S. Manandhar, "SemEval-2015 Task 12: Aspect Based Sentiment Analysis," Proc. 9th Int. Work. Semant. Eval., 2015, hal. 486–495.

[7] M. Pontiki, D. Galanis, H. Papageorgiou, dkk., "SemEval-2016 Task 5: Aspect Based Sentiment Analysis," Proc. 10th Int. Work. Semant. Eval., 2016, hal. 19–30.

[8] S. Movahedi, E. Ghadery, H. Faili, dan A. Shakery, "Aspect Category Detection via Topic-Attention Network," arXiv Prepr. arXiv1901.01183, hal. 1–9, 2019.

[9] H. H. Do, P. Prasad, A. Maag, and A. Alsadoon, "Deep Learning for Aspect-Based Sentiment Analysis: A Comparative Review," Expert

Systems with Applications, vol. 118, pp. 272–299, Mar. 2019, 10.1016/j.eswa.2018.10.003.
[10] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ArXiv:1810.04805 [Cs].
[11] P. R. Amalia and E. Winarko, "Aspect-Based Sentiment Analysis on Indonesian Restaurant Review Using a Combination of Convolutional Neural Network and Contextualized Word Embedding," Indonesian J. Comput. Cybern. Syst., vol. 15, no. 3, p. 285, Jul. 2021. 10.22146/ijccs.67306.
[12] R. Man and K. Lin, "Sentiment Analysis Algorithm Based on BERT and Convolutional Neural Network," in 2021 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC), Dalian, China, pp. 769–772, Apr. 2021. 10.1109/IPEC51340.2021.9421110.
[13] W. Quan, Z. Chen, J. Gao, and X. T. Hu, "Comparative Study of CNN and LSTM based Attention Neural Networks for Aspect-Level Opinion Mining," in 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, Dec. 2018, pp. 2141–2150. 10.1109/BigData.2018.8622150.
[14] S. Poria, E. Cambria, and A. Gelbukh, "Aspect extraction for opinion mining with a deep convolutional neural network," Knowledge-Based Systems, vol. 108, pp. 42–49, Sep. 2016, 10.1016/j.knosys.2016.06.009.
[15] KIM, Y., Convolutional neural networks for sentence classification. In: EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference. hal.1746-1751. 2014.
[16] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method." arXiv, Dec. 22, 2012.
[17] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization." arXiv, Jan. 29, 2017.
[18] T. Dozat, "INCORPORATING NESTEROV MOMENTUM INTO ADAM," 2016.
[19] S. Ruder, "An overview of gradient descent optimization algorithms." arXiv, Jun. 15, 2017.
[20] M. M. Abdelgwad, "Arabic aspect based sentiment classification using BERT." arXiv, Nov. 27, 2021.
[21] M. O. Ibrohim and I. Budi, "Multi-label Hate Speech and Abusive Language Detection in Indonesian Twitter," in Proceedings of the Third Workshop on Abusive Language Online, Florence, Italy, pp. 46-57, 2019. 10.18653/v1/W19-3506.
[22] B. Wilie et al., "IndoNLU: Benchmark and Resources for Evaluating Indonesian Natural Language Understanding," p. 15. arXiv, Oct. 08, 2020.
[23] M. T. Ari Bangsa, S. Priyanta, dan Y. Suyanto, "Aspect-Based Sentiment Analysis of Online Marketplace Reviews Using Convolutional Neural Network," Indonesian J. Comput. Cybern. Syst., vol. 14, no. 2, p. 123, Apr. 2020

Perbandingan Akurasi Deteksi Emosi Pada Suara Menggunakan Multilayer Perceptron, Random Forest, Decision Tree dan K-NN

Windra Swastika[1], Alvin A. Oepojo[1], dan Paulus L. T. Irawan[1]
[1]Program Studi Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Ma Chung, Malang, Indonesia

Corresponding author: Windra Swastika (e-mail: windra.swastika@machung.ac.id).

ABSTRACT This study aims to compare the accuracy of emotion recognition through sound using several types of classifiers. The four basic emotions to be recognized are happy, sad, neutral, and angry. The research methodology began with obtaining sound datasets from the RAVDESS database, consisting of 24 actors with 60 sound samples per actor. However, only 28 sounds were selected from each actor, resulting in a total of 672 sounds used in this study. Three techniques, namely mel frequency cepstral coefficient (MFCC), Chroma, and Mel Scale, were used to extract features from the sound dataset. Four types of classifiers, Multilayer Perceptron Classifier (MLPC), Decision Tree, Random Forest, and K-NN, were then used to create the models. The dataset was divided into training and testing data for each classifier in three trials, which were 85% training – 25% testing, 80% training – 25% testing, and 75% training – 25% testing. The results of the study showed that the model using the Random Forest Classifier had the highest accuracy, which was 79% with an 80% training – 20% testing dataset division. Meanwhile, the model using the Decision Tree Classifier had the lowest accuracy, which was 57% with a 75% training – 25% testing dataset division. In this study, the feature extraction techniques used, which were MFCC, Chroma, and Mel Scale, were proven effective in producing sound dataset features. Furthermore, the study results also showed that the Random Forest Classifier was superior in recognizing emotions through sound compared to other types of classifiers.

KEYWORDS Decision Tree, K-NN, Multilayer Perceptron, Random Forest, Speech Emotion Recognition

ABSTRAK Penelitian ini bertujuan untuk membandingkan akurasi pengenalan emosi melalui suara dengan menggunakan beberapa jenis classifier. Emosi dasar yang akan dikenali ada 4, yaitu senang, sedih, neutral dan marah. Metodologi penelitian dimulai dengan memperoleh dataset suara dari database RAVDESS, yang terdiri dari 24 aktor dengan jumlah suara sebanyak 60 per aktor. Namun, hanya 28 suara yang dipilih dari setiap aktor, sehingga total ada 672 suara yang digunakan dalam penelitian ini. Untuk mengekstraksi fitur dari dataset suara, digunakan tiga teknik yaitu mel frequency cepstral coefficient (MFCC), Chroma, dan Skala Mel. Kemudian, empat jenis classifier digunakan dalam pembuatan model yaitu Multilayer Perceptron Classifier (MLPC), Decision Tree, Random Forest, dan K-NN. Dataset dibagi menjadi data train dan data test dalam 3 uji coba untuk masing-masing classifier, yait 85% train – 25% test, 80% train – 25% test, dan 75% train dan 25% test. Hasil penelitian menunjukkan bahwa model dengan menggunakan Random Forest Classifier memiliki akurasi tertinggi yaitu sebesar 79% dengan pembagian dataset 80% train - 20% test. Sedangkan, model dengan Decision Tree Classifier memiliki akurasi terendah sebesar 57% dengan pembagian dataset menjadi 75% train - 25% test. Dalam penelitian ini, teknik ekstraksi fitur yang digunakan yaitu MFCC, Chroma, dan Skala Mel, yang terbukti efektif dalam menghasilkan fitur dari dataset suara. Selain itu, hasil penelitian juga menunjukkan bahwa Random Forest Classifier lebih unggul dalam mengenali emosi melalui suara jika dibandingkan dengan jenis classifier yang lain.

KATA KUNCI Decision tree, Deteksi Emosi Pada Suara, K-NN, Multilayer Perceptron, Random Forest

I. PENDAHULUAN

Emosi merupakan fenomena yang tidak dapat dipisahkan dari kehidupan manusia. Emosi dapat muncul secara verbal maupun nonverbal, dan salah satu bentuk ekspresi emosi yang paling umum adalah melalui suara. Suara yang dipenuhi emosi dapat dengan mudah diidentifikasi oleh orang lain, terutama jika suara tersebut mengalami perubahan yang signifikan dari suara normal. Deteksi emosi suara merupakan salah satu bidang yang sedang berkembang dalam penelitian teknologi suara.

Deteksi emosi suara memiliki berbagai aplikasi praktis dalam kehidupan sehari-hari [1]. Salah satu aplikasi yang paling umum adalah dalam industri telekomunikasi, di mana teknologi deteksi emosi suara dapat digunakan untuk meningkatkan layanan pelanggan. Selain itu, deteksi emosi suara juga dapat digunakan dalam bidang kesehatan mental untuk mengidentifikasi perubahan emosi yang mungkin merupakan tanda-tanda awal dari masalah kesehatan mental. Aplikasi lain dari deteksi emosi suara adalah dalam pembuatan film, di mana teknologi ini dapat digunakan untuk membantu menciptakan suasana yang tepat sesuai dengan skenario.

Meskipun deteksi emosi suara memiliki banyak aplikasi praktis, namun teknologi ini juga merupakan tantangan yang cukup sulit. Emosi manusia bisa bervariasi secara cepat dan tidak terlalu terprediksi, sehingga sulit untuk mengidentifikasi emosi dengan tepat hanya dengan menggunakan suara saja. Selain itu, ada banyak faktor yang bisa mempengaruhi emosi, seperti latar belakang, pengalaman, dan kondisi fisik, yang semuanya harus dipertimbangkan dalam proses deteksi emosi suara.

Penelitian terdahulu telah menggunakan berbagai metode untuk mencoba meningkatkan akurasi deteksi emosi suara. Salah satu metode yang sering digunakan adalah pemodelan suara, di mana algoritma dibuat untuk mengenali pola-pola suara yang biasa muncul saat seseorang merasa cemas, sedih, marah, atau senang. Selain itu, pemodelan gerakan tubuh juga sering digunakan sebagai tambahan untuk membantu dalam proses deteksi emosi suara [2]. Meskipun metode-metode ini telah menunjukkan hasil yang cukup baik, namun masih terdapat banyak tantangan yang harus diatasi untuk meningkatkan akurasi deteksi emosi suara.

Salah satu tantangan utama dalam deteksi emosi suara adalah perbedaan individu dalam ekspresi emosi [3]. Setiap orang memiliki cara yang berbeda dalam mengekspresikan emosi, sehingga sulit untuk menentukan pola-pola suara yang pasti akan muncul saat seseorang merasa cemas, sedih, marah, atau senang. Selain itu, ada juga perbedaan kultural dalam ekspresi emosi yang perlu dipertimbangkan dalam proses deteksi emosi suara. Penelitian terdahulu telah menunjukkan bahwa teknologi deteksi emosi suara masih belum cukup akurat untuk digunakan secara luas. Namun, dengan terus melakukan penelitian dan pengembangan, diharapkan teknologi deteksi emosi suara akan semakin maju dan dapat memberikan manfaat yang lebih besar bagi kehidupan sehari-hari.

Pada penelitian ini, akan dilakukan perbandingkan akurasi deteksi emosi pada suara dengan menggunakan 4 metode classifier, yaitu Multilayer Perceptron [4], Random Forest [5], Decision Tree [6] dan KNN. Metode-metode tersebut juga sering digunakan di penelitian lain untuk menyelesaikan permasalahan klasifikasi [7][8]. Dari ke-4 metode tersebut diharapkan bisa diketahui metode manakah yang dapat menghasilkan model dengan akurasi yang paling baik dalam melakukan deteksi emosi pada suara serta parameter yang sesuai untuk mendapatkan akurasi yang terbaik.

II. TINJAUAN PUSTAKA

A. MFCC

MFCC (*Mel Frequency Cepstral Coefficient*) merupakan suatu cara ekstraksi fitur yang biasanya digunakan untuk *speech recognition* [9]. Guna MFCC adalah sebagai bentuk saluran vokal menunjukkan dirinya dalam amplop spektrum daya waktu singkat. Menghitung MFCC memiliki 6 tahapan yaitu:

1. Melakukan pre-emphasis pada sinyal suara. Pre-emphasis adalah proses meningkatkan amplitudo sinyal suara pada frekuensi tinggi untuk mengurangi distorsi harmonik.
2. Menyamakan sinyal suara dengan menggunakan Fast Fourier Transform (FFT) untuk mengkonversi dari domain waktu ke domain frekuensi.
3. Menentukan filter bank mel-frequency dengan menentukan beberapa titik pemisah pada skala mel dan mengaplikasikan triangular filter pada setiap titik pemisah tersebut.
4. Melakukan dot-product antara hasil FFT dengan setiap filter dalam filter bank, yang akan memberikan dengan log-energi pada setiap filter.
5. Mengaplikasikan Discrete Cosine Transform (DCT) pada log-energi untuk mengkonversi ke domain cepstrum.
6. Koefisien cepstral dari hasil DCT yang dianggap penting akan dipilih sebagai fitur yang digunakan untuk klasifikasi suara.

B. CHROMA

Chroma adalah salah satu metode untuk mengekstraksi fitur suara yang digunakan dalam pengolahan suara dan pengenalan suara [10]. Chroma adalah representasi musik dari sinyal suara yang memfokuskan pada distribusi energi frekuensi dari sinyal suara pada setiap nada musik.

Cara untuk menghitung Chroma adalah:

1. Melakukan analisis Short-Time Fourier Transform (STFT) pada sinyal suara untuk mengkonversi dari domain waktu ke domain frekuensi.

2. Membuat filter bank yang terdiri dari 12 filter yang mewakili setiap nada musik dalam skala chroma.
3. Melakukan dot-product antara hasil STFT dengan setiap filter dalam filter bank, yang akan memberikan kita dengan energi pada setiap filter.
4. Mapping dari energi filter bank kedalam notasi musik untuk menentukan distribusi energi frekuensi pada setiap nada musik dari sinyal suara.

Chroma feature digunakan dalam musik analysis, pengenalan musik, and content-based retrieval. Salah satu keunggulan Chroma feature adalah dapat memisahkan suara dari berbagai macam instrumen musik yang memainkan notasi yang sama. Beberapa Chroma-based feature yang umum digunakan diantaranya Chroma Vector, Chroma Energy Normalized, Chroma Deviation.

C. SKALA MEL

Skala Mel adalah skala yang digunakan dalam proses transformasi frekuensi pada pengolahan suara [11]. Skala Mel adalah skala logaritmik yang lebih cocok dengan cara kerja pendengaran manusia. Pada skala Mel, perbedaan frekuensi yang sama akan diwakili oleh jarak yang sama pada skala Mel, sementara pada skala linear, perbedaan frekuensi yang sama akan diwakili oleh jarak yang berbeda.

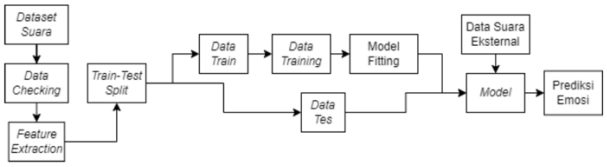
Dalam pengolahan suara, skala Mel digunakan dalam *mel frequency cepstral coefficient* (MFCC) untuk mengekstrak fitur suara. Dalam MFCC, sinyal suara dikonversi dari domain waktu ke domain frekuensi menggunakan *Fast Fourier Transform* (FFT), kemudian dikonversi lagi dari domain frekuensi linear ke domain frekuensi Mel menggunakan filter bank mel-frequency.

Pengkonversian dari domain frekuensi linear ke domain Mel dilakukan dengan melakukan transformasi (1).

$$f(mel) = 2595\log(1 + f(Hz)/700) \qquad (1)$$

III. METODOLOGI PENELITIAN

A. RANCANGAN SISTEM



GAMBAR 1. Rancangan sistem penelitian

Gambar 1 menunjukkan rancangan sistem yang akan dibuat. Hal pertama yang perlu disiapkan adalah dataset suara. Dataset suara berisi berbagai macam suara aktor dan / atau orang dengan gender yang berbeda yang berbicara dengan berbagai macam dialog dengan bahasa yang sama. Format dari *file* suara berupa .wav atau setidaknya memiliki format yang sama dan kualitas tiap *file* suara harus mono. Tiap *file* juga harus memiliki nama yang berbeda dengan format penamaan tertentu. Contohnya bila suara emosi neutral adalah 01 dengan actor 01 yang berbicara, maka file bisa dinamai 01-01. Jumlah suara harus minimal ratusan agar saat model dilatih akan memiliki banyak referensi untuk membedakan emosi.

Lalu, dataset suara tersebut akan dicek di *Data Checking*. *Data Checking* adalah sebuah tahapan untuk memastikan data bisa dipakai. Hal yang dicek seperti apakah ciri-ciri untuk tahapan selanjutnya *feature extraction* bisa diambil atau tidak dari *file* suara tersebut dan apakah tiap file memiliki kualitas *mono*. Bila ada beberapa file yang tidak bisa diambil ciri-cirinya, maka *file* tersebut tidak akan digunakan untuk tahap selanjutnya. Setelah dataset sudah dicek, tahap selanjutnya adalah *feature extraction* dimana data dilihat ciri-cirinya untuk membedakan tiap emosi. Ciri-cirinya bisa merupakan frekuensi dari suara, penamaan *file* dan lain-lainnya.

Lalu data akan dibagi dengan *train-test split* untuk memisahkan dataset suara menjadi dua data yaitu data latih untuk melatih model yang akan digunakan untuk prediksi dan data tes untuk mengetes prediksi dari data latih. Pembagian data latih dan data tes bisa dipisah sesuai keinginan. Pada penelitian ini, akan dilakukan 3 jenis *train-test split*, yaitu *85% train – 15% test, 80% train – 20% test* dan *75% train – 25% test.*

Sebelum data bisa dimasukkan, model harus dibuat dahulu. Model ini berisi sebuah algoritma yang digunakan untuk membedakan data-data menjadi kategori-kategori tertentu seperti membedakan emosi senang, sedih, neutral dan marah. Model juga diatur sesuai algoritmanya seperti jumlah iterasi, jumlah *hidden layer,* dan berbagai macam pengaturan lainnya. Untuk penelitian ini, ada empat algoritma model yang akan digunakan yaitu MLPClassifier (*Multi-Layer Perceptron Classifier*), *Decision Tree Classifier*, *Random Forest Classifier*, dan *K-Nearest Neighbors Classifier*.

Bila data sudah dipisah, maka data latih dilatih dengan mempelajari tiap file yang memililki fitur yang berbeda dan penamaan file yang berbeda dari emosi tertentu. Akhirnya, data dimasukkan ke dalam sebuah model yaitu sebuah file yang sudah dilatih untuk mengerti perbedaan dari tiap emosi untuk melakukan prediksi dengan data tes. Prediksi dilakukan dengan membandingkan hasil latihan yang akan dibandingkan dengan emosi sebenarnya. Akurasi dari model akan tergantung dari total jumlah benar dan salah dari prediksi model.

B. FEATURE EXTRACTION

Feature extraction berguna untuk melihat ciri-ciri / fitur dari sebuah file untuk membedakan perbedaan dari satu file dengan yang lain sesuai yang diinginkan. Ciri-ciri atau fitur yang dibedakan ada empat:

1. Nama file,
2. MFCC (*Mel Frequency Cepstral Coefficient*) yaitu spektrum daya suara jangka pendek,
3. Chroma yang berkaitan dengan 12 kelas *pitch* yang berbeda,

4. Mel (*Mel Spectrogram Frequency*) untuk mengetahui frekuensi suara.

File suara akan dicek fitur atau ciri suara yang dijelaskan di atas ada atau tidak kecuali nama file dan bila ada, akan mengambil beberapa hal untuk membantu membedakan fitur-fitur satu file dengan file lain. MFCC akan mengambil *sample rate* dan jumlah MFCC dari file suara, Chroma mengambil STFT (*Short-time Fourier transform*) dan *sample rate* dan Mel hanya memerlukan *sample rate*. Hasil dari tiap fitur akan direpresentasikan dengan satuan angka.

Setelah tiap ciri atau fitur sudah didapat, maka tiap fitur dan / atau ciri-ciri akan disimpan dalam sebuah array untuk membedakan tiap file yang ada. Dalam penelitian ini, hal yang ingin dibedakan adalah emosi. Ada berbagai macam emosi dan juga emosi memiliki cara mengekspresikannya yang berbeda seperti intensitas kemarahan, intensitas kesenangan dan lain-lain berbasis dari empat fitur diatas.

Dari banyak emosi yang bisa diprediksi, penelitian ini akan menggunakan empat emosi umum yaitu neutral, senang, sedih dan marah. Untuk mengetahui bagaimana mengerti emosi yang berbeda, maka perlu membuat *dictionary* untuk emosi tersebut. *Dictionary* dibuat dengan membuat array dimana string di nama file akan menunjukkan emosi tertentu, contohnya seperti '01' menunjukkan emosi neutral. Setelah itu, file suara mulai diekstrak satu per satu untuk tiap aktor dan tiap emosi yang ada.

C. DATASET SUARA

Untuk penelitian ini, ada dua dataset suara yang akan digunakan. Dataset suara pertama berasal dari website Kaggle dengan dataset RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song) Emotional Speech Audio [12] yang memiliki 24 aktor (12 laki-laki dan 12 perempuan) dan setiap aktor memiliki 60 suara tetapi hanya 28 suara dari tiap aktor yang dipakai dengan total 672.

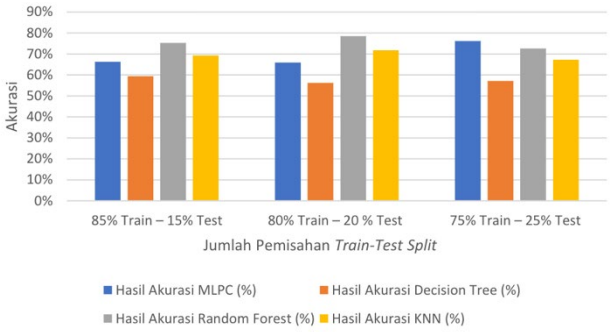
Ada tambahan 8 file suara yang berasal dari *website* Freesound dengan 4 suara dari aktor laki-laki yang berbeda dan 4 suara aktor perempuan yang berbeda-beda yang digunakan untuk mengetes model dengan data eksternal. Penamaan dari data suara eksternal tidak disamakan dengan format nama dari dataset RAVDESS.

Dataset RAVDESS memiliki penamaan spesifik untuk tiap *file* dan dikode dengan nomor. Berikut adalah ciri-ciri dari penamaan file:

1. Modalitas (01 = *video* dan *audio*, 02 = *video* saja, 03 = *audio* saja),
2. Lagu atau monolog (01 = monolog, 02 = lagu).
3. Emosi (01 = *neutral*, 02 = tenang, 03 = senang, 04 = sedih, 05 = marah, 06 = ketakutan, 07 = merasa jijik, 08 = kaget),
4. Intensitas emosi (01 = normal, 02 = kuat). Catatan: Neutral tidak memiliki intensitas emosi,
5. Kata (01 = "*Kids are talking by the door*", 02 = "*Dogs are sitting by the door*", 03 = Lain-lain 1, 04 = Lain-lain 2),
6. Repetisi (01 = repetisi pertama, 02 = repetisi kedua),
7. Aktor (Nomor ganjil adalah laki-laki, nomor genap adalah perempuan).

IV. HASIL DAN DISKUSI

Model akan dibuat dengan berbagai macam classifier yaitu MLPC, *Decision Tree Classifier*, *Random Forest Classifier* dan KNN *Classifier*.



GAMBAR 2. Hasil akurasi untuk masing-masing model

Gambar 2 menunjukkan hasil akurasi dari 4 *classifier* dan masing-masing classifier diukur menggunakan 3 jenis kombinasi *train-test*.

Rata-rata akurasi dari ke-4 *classifier* adalah sebagai berikut: untuk 85% train – 15% test adalah 67.6%, rata-rata akurasi untuk 80% train – 20% test adalah 68.2% sedangkan untuk 75% train - 25% test memiliki rata-rata akurasi 68.3%. Sedangkan untuk masing-masing classifier, rata-rata akurasi yang didapatkan adalah sebagai berikut: untuk MLPC, rata-rata akurasi yang didapatkan adalah 69.5%, untuk rata-rata akurasi yang didapatkan dari *Decision Tree* adalah 57.6%, untuk rata-rata akurasi yang didapatkan dari *Random Forest* adalah 75.5%, dan rata-rata akurasi yang didapatkan KNN adalah 68.2%.

Secara umum, hasil akurasi tertinggi didapatkan untuk jenis 80% train - 20% test dengan *Random Forest Classifier* yaitu dengan akurasi sebesar 79%. Sementara akurasi terendah adalah model dengan 75% train - 25% test yang menggunakan *Decision Tree Classifier* yaitu dengan akurasi 57%.

Tabel I, II, III dan IV merupakan hasil *confusion matrix* dari masing-masing classifier.

TABEL I
CONFUSION MATRIX PREDIKSI EMOSI MODEL MLPC

Emosi	Neutral (Prediksi)	Happy (Prediksi)	Sad (Prediksi)	Angry (Prediksi)
Neutral (Asli)	47.3%	30.1%	29.1%	11%
Happy (Asli)	3.1%	67.7%	17.7%	11.6%
Sad (Asli)	4.2%	18%	73%	7.3%
Angry (Asli)	0%	10.1%	5%	84.9%

TABEL II
CONFUSION MATRIX PREDIKSI EMOSI MODEL DECISION TREE CLASSIFIER

Emosi	Neutral (Prediksi)	Happy (Prediksi)	Sad (Prediksi)	Angry (Prediksi)
Neutral (Asli)	43.7%	16.4%	34.5%	5.4%
Happy (Asli)	8.5%	53.8%	13.1%	24.6%
Sad (Asli)	23.3%	21.7%	50.8%	4.2%
Angry (Asli)	2%	7.1%	2%	81.9%

TABEL III
CONFUSION MATRIX PREDIKSI EMOSI MODEL RANDOM FOREST CLASSIFIER

Emosi	Neutral (Prediksi)	Happy (Prediksi)	Sad (Prediksi)	Angry (Prediksi)
Neutral (Asli)	49.1%	12.7%	38.2%	0%
Happy (Asli)	1.5%	69.2%	17%	12.3%
Sad (Asli)	4.2%	15%	77.5%	3.3%
Angry (Asli)	0%	5%	2%	93%

TABEL IV
CONFUSION MATRIX PREDIKSI EMOSI MODEL KNN CLASSIFIER

Emosi	Neutral (Prediksi)	Happy (Prediksi)	Sad (Prediksi)	Angry (Prediksi)
Neutral (Asli)	76%	11%	13%	0%
Happy (Asli)	4%	61%	13%	22%
Sad (Asli)	17%	14%	62%	7%
Angry (Asli)	2%	7.1%	4%	86.9%

Dari tabel I, II, III dan IV, terlihat bahwa model yang prediksi emosi yang paling akurat adalah model yang menggunakan *Random Forest Classifier* dengan rata-rata akurasi 72.2%. Emosi yang paling mudah diprediksi adalah emosi *Angry* atau marah dengan akurasi di atas 93%.

V. KESIMPULAN

Dari keseluruhan model *classifier* yang diuji untuk melakukan prediksi emosi berdasarkan suara, didapatkan bahwa *Random Forest Classifier* memiliki akurasi yang tertinggi, yaitu 75,5%. Hal ini dikarenakan *classifier* ini menggabungkan beberapa pohon keputusan (*decision tree*) untuk menghasilkan prediksi akhir. Ketika beberapa pohon keputusan digabungkan, akan terbentuk "hutan" yang dapat memberikan hasil yang lebih akurat dan stabil. Dalam hal ini, *Random Forest Classifier* dapat mengatasi *overfitting* yang sering terjadi pada *decision tree*, serta dapat mengatasi masalah *high variance* yang biasa terjadi pada *MLPC* dan *K-NN*.

Selain itu, Random Forest Classifier juga dapat melakukan seleksi fitur secara otomatis dan memilih fitur-fitur yang paling relevan dalam mengklasifikasikan data. Hal ini sangat penting dalam pengenalan emosi melalui suara karena suara memiliki banyak fitur yang kompleks dan tidak semuanya relevan dalam menentukan emosi yang diungkapkan.

Dari *Confusion matrix* untuk *Random Forest Classifier*, didapatkan bahwa nampak bahwa emosi marah (*angry*) merupakan emosi yang paling akurat untuk dideteksi, sedangkan emosi netral merupakan emosi yang paling tidak akurat. Faktor jumlah dataset untuk emosi netral tidak sebanyak dibandingkan emosi yang lainnya memberikan pengaruh terhadap akurasi deteksi.

PERAN PENULIS

Windra Swastika: mendesain metodologi penelitian dan evaluasi
Alvin Andrius Oepojo: implementasi algoritma ekstraksi fitur; pengolah data dan ekstraksi fitur
Paulus Lucky Tirma Irawan: Pengujian hasil dan analisis

COPYRIGHT

DAFTAR PUSTAKA

[1] Zhang, Hongli, Alireza Jolfaei, and Mamoun Alazab. "A face emotion recognition method using convolutional neural network and image edge computing." *IEEE Access* 7 (2019): 159081-159089.
[2] Gunes, Hatice, and Maja Pantic. "Automatic, dimensional and continuous emotion recognition." *International Journal of Synthetic Emotions (IJSE)* 1.1 (2010): 68-99.
[3] Khalil, Ruhul Amin, et al. "Speech emotion recognition using deep learning techniques: A review." *IEEE Access* 7 (2019): 117327-117345.
[4] Alnuaim, A. A., Zakariah, M., Shukla, P. K., Alhadlaq, A., Hatamleh, W. A., Tarazi, H., ... & Ratna, R. (2022). Human-computer interaction for recognizing speech emotions using multilayer perceptron classifier. Journal of Healthcare Engineering, 2022.
[5] Yan, S., Ye, L., Han, S., Han, T., Li, Y., & Alasaarela, E. (2020, June). Speech interactive emotion recognition system based on random forest. In *2020 International Wireless Communications and Mobile Computing (IWCMC)* (pp. 1458-1462). IEEE.
[6] Sun, L., Fu, S., & Wang, F. (2019). Decision tree SVM model with Fisher feature selection for speech emotion recognition. *EURASIP Journal on Audio, Speech, and Music Processing*, *2019*(1), 1-14.
[7] L. Alwi, A. T. Hermawan, and Y. . Kristian, "Identifikasi Biji-Bijian Berdasarkan Ekstraksi Fitur Warna, Bentuk dan Tekstur Menggunakan Random Forest", INSYST, vol. 1, no. 2, pp. 92–98, Dec. 2019.
[8] J. A. Septian, T. M. Fachrudin, and A. Nugroho, "Analisis Sentimen Pengguna Twitter Terhadap Polemik Persepakbolaan Indonesia Menggunakan Pembobotan TF-IDF dan K-Nearest Neighbor", *INSYST*, vol. 1, no. 1, pp. 43–49, Aug. 2019.
[9] Zheng, Fang, Guoliang Zhang, and Zhanjiang Song. "Comparison of different implementations of MFCC." *Journal of Computer science and Technology* 16.6 (2001): 582-589.

[10] Er, Mehmet Bilal, and Ibrahim Berkan Aydilek. "Music emotion recognition by using chroma spectrogram and deep visual features." *International Journal of Computational Intelligence Systems* 12.2 (2019): 1622-1634.
[11] Gowdy, John N., and Zekeriya Tufekci. "Mel-scaled discrete wavelet coefficients for speech recognition." *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)*. Vol. 3. IEEE, 2000.
[12] Steven R. Livingstone, & Frank A. Russo. (2019). *RAVDESS Emotional speech audio* [Data set]. Kaggle. https://doi.org/10.34740/KAGGLE/DSV/256618

Perbandingan Implementasi Evolutionary Algorithm (EPO, FHO, dan CFA) pada Kasus Travelling Salesman Problem untuk Tempat Pariwisata di Surabaya

Christian T.S.L. Chen[1], David Cahyadi[1], Jonathan A. Bevan[1], Williandy Takhta[1], Ariel P. Lesmana[1], Christopher D.A. Poernomo[1], dan Widean Nagari[1]
[1]Departemen Informatika, Fakultas Sains dan Teknologi, Institut Sains dan Teknologi Terpadu Surabaya, Surabaya, Indonesia

Corresponding author: Christian T.S.L. Chen (e-mail: christian.t20@mhs.istts.ac.id).

ABSTRACT Traveling is a rapidly growing business around the world, and Indonesia is no exception. In Indonesia, especially Surabaya, the travel and tourism industry has been on the rise in recent years, and it is expected to continue to grow in the coming years. With these improvements, route search for tourism must be efficient and fast, one of the popular solutions right now is Evolutionary Algorithms (EA). Evolutionary algorithms are a type of optimization technique that mimics the process of natural evolution to find solutions to complex problems. One such problem that can be effectively solved using evolutionary algorithms is the Traveling Salesman Problem (TSP). This problem involves visiting a specific set of cities and finding the shortest route back to the starting point. Several evolutionary algorithms have been proposed to solve TSP, such as Cuttlefish Algorithm (CFA), Emperor Penguin Optimizer (EPO), and Fire Hawk Optimizer (FHO). The cuttlefish algorithm is based on the behavior of wild cuttlefish, the EPO is inspired by the huddling behavior of emperor penguins, whereas FHO uses the principle of fire propagation. These algorithms have the potential to solve TSP using their own uniqueness. Our conclusion for all those algorithms that are used in this research is that EPO manages to find the best solution while CFA and FHO solutions being slightly below EPO. From our research, EPO is 39.97% better than CFA and 14.75% better than FHO. EPO also has better computation time which is 69.59% faster than CFA and 178.34% faster than FHO.

KEYWORDS Cuttlefish Algorithm, Emperor Penguin Optimizer, Fire Hawk Optimizer, Travelling Salesman Problem

ABSTRAK Traveling merupakan bisnis yang tumbuh pesat di seluruh dunia, dan Indonesia tidak terkecuali. Di Indonesia, khususnya Surabaya, industri pariwisata telah mengalami peningkatan dalam beberapa tahun terakhir, dan diharapkan akan terus tumbuh dalam beberapa tahun ke depan. Dengan peningkatan tersebut, pencarian rute untuk pariwisata harus efisien dan cepat, salah satu solusi yang populer saat ini adalah *Evolutionary Algorithms* (EA). Algoritma evolusi adalah jenis teknik optimisasi yang meniru proses evolusi alami untuk menemukan solusi terhadap masalah yang kompleks. Salah satu permasalahan yang dapat diselesaikan dengan efektif menggunakan algoritma evolusi adalah *Traveling Salesman Problem* (TSP). Permasalahan tersebut melibatkan pengunjungan pada beberapa kota dan menemukan rute terpendek untuk kembali ke titik awal. Beberapa algoritma evolusi telah dicadangkan untuk menyelesaikan TSP, seperti algoritma Cuttlefish (CFA), Emperor Penguin Optimizer (EPO) dan Fire Hawk Optimizer (FHO). Algoritma sotong didasarkan pada perilaku sotong liar, EPO terinspirasi oleh perilaku berkerumun dari penguin kaisar, sedangkan FHO menggunakan prinsip propagasi api. Semua algoritma yang telah disebutkan tadi memiliki potensi untuk menyelesaikan TSP dengan keunikannya masing-masing. Kesimpulan kami untuk semua algoritma yang digunakan dalam penelitian ini adalah bahwa EPO berhasil menemukan solusi terbaik diikuti dengan solusi dari CFA dan FHO. Berdasarkan hasil percobaan kami, didapatkan EPO menghasilkan solusi 39.97% lebih baik dari CFA serta 14.75% lebih baik dari FHO secara rata-rata. Serta EPO juga memiliki waktu komputasi rata-rata lebih cepat (69.59% lebih cepat dari CFA dan 178.34% lebih cepat dari FHO).

KATA KUNCI *Cuttlefish Algorithm*, *Emperor Penguin Optimizer*, *Fire Hawk Optimizer*, *Travelling Salesman Problem*

I. PENDAHULUAN

Indonesia merupakan negara yang memiliki banyak sekali budaya dan alam. Selain Bali, Jakarta, maupun Yogyakarta, Surabaya juga tidak kalah hebat. Surabaya melalui program non-profit pemerintah yang bergerak dalam bidang branding marketing dengan nama Sparkling Surabaya, melakukan publikasi tentang tempat pariwisata yang terdapat di Surabaya.

Dengan berkembangnya jumlah tempat pariwisata di Surabaya maka proses penemuan jalan yang terbaik dan efisienpun juga perlu untuk diteliti lebih lanjut. Algoritma–algoritma untuk menemukan jalan yang terbaik dan efisien tersebut juga mulai berkembang dengan sangat cepat, salah satu perkembangan dari algoritma–algoritma tersebut adalah Evolutionary Algorithm.

Evolutionary Algorithm merupakan algoritma pencari solusi yang dilakukan secara stokastik yang biasanya ditemukan atau diimplementasikan dari evolusi biologis[1]. Sedangkan menurut Sloss, EA itu merupakan Algoritma Evolusi yang menyediakan kerangka kerja dimana kerangka tersebut dapat digunakan kembali di berbagai domain, mereka sebagian besar merupakan algoritma yang terinspirasi dari biologi yang berada sebagai cabang dari Intelijen Buatan[2]. Bartz-Beielstein mengatakan bahwa Algoritma Evolusi dipahami sebagai algoritma pencarian stokastik berbasis populasi yang dalam beberapa hal menirukan proses evolusi alami yang terjadi di alam. Proses tersebut dapat digunakan untuk menemukan solusi optimal dari suatu masalah yang diberikan. EA dapat digunakan untuk menyelesaikan berbagai masalah yang beragam, mulai dari masalah yang memiliki struktur yang sangat kompleks hingga masalah yang relatif sederhana [3].

Dari pernyataan-pernyataan diatas dapat disimpulkan bahwa Algoritma Evolusi merupakan algoritma pencarian solusi stokastik yang didasarkan pada proses evolusi alami yang terjadi di alam. Terinspirasi dari ilmu biologi, Algoritma Evolusi dapat digunakan untuk menyelesaikan berbagai masalah yang memiliki struktur yang kompleks atau yang relatif sederhana. Dengan menggunakan kerangka kerja tersebut, solusi optimal untuk masalah yang diberikan dapat ditemukan. EA telah banyak digunakan dalam berbagai domain, terutama di bidang Intelijen Buatan.

Evolusi biologis merupakan salah satu hal yang sangat menunjukkan bagaimana mahluk hidup dapat mengoptimasi dirinya dengan perubahan agar dapat menjadi mahluk hidup yang bisa bertahan hidup sampai sekarang. Optimasi–optimasi diimplementasikan dalam sebuah algoritma algoritma tersebut yang diharapkan dapat mencari solusi dari sebuah problem yang lebih kompleks seperti Travelling Salesman Problem.

Travelling Salesman Problem (TSP) merupakan salah satu masalah yang sangat mudah untuk dijelaskan namun sangat susah untuk disesaikan. Problem dari TSP adalah bagaimana cara agar travelling salesman dapat mengunjungi semua kota yang ada pada permasalahan dan dapat kembali ketempat awal dengan jarak terdekat atau dengan efisien[4].

Travelling Salesman Problem menurut Hashim merupakan masalah optimasi kombinatorial klasik yang berurusan dengan menemukan jarak atau waktu terpendek untuk tur tertutup di situasi kota n, di mana setiap kota dikunjungi tepat sekali sebelum kembali ke titik awal[5] dan menurut Nurdiawan bahwa Travelling Salesman Problem itu adalah optimasi kombinatorial[6].

II. PENELITIAN TERDAHULU

Travelling Salesman Problem (TSP) merupakan problem mudah untuk dijelaskan namun sangat sulit untuk diselesaikan. Tujuan dari TSP adalah untuk menemukan rute terpendek atau paling efisien yang mengunjungi setiap kota dalam masalah tersebut tepat sekali dan kembali ke titik awal. Hashim mengatakan bahwa TSP harus memiliki setiap kota dikunjungi sekali sebelum kembali ke titik awal, dan Nurdiawan mengatakan itu adalah optimasi kombinatorial. Masalah yang dialami TSP sangat sulit untuk diselesaikan karena seringkali memerlukan menemukan rute paling efisien dengan jarak terpendek.

Beberapa penelitian mengenai TSP telah dilakukan oleh beberapa ahli seperti Hashim yang melakukan penelitian TSP untuk melakukan optimasi kepada perjalanan untuk mengunjungi tempat wisata di Langkawi, salah satu distrik/pulau di Malaysia. Hashim menggunakan software LINGO versi 12.0 yang merupakan software matematika yang dapat membantu menyelesaikan problem TSP[5].

Nurdiawan et al., sebelumnya juga pernah melakukan penelitian mengenai permasalahan TSP untuk melakukan proses penjadwalan paket touring menggunakan Genetic Algorithm. Nurdiawan menggunakan Genetic Algorithm untuk menyelesaikan permasalahan TSP[6]. Penelitian tersebut berhasil menemukan rute paling optimal untuk pariwisata antar kota Indramayu, Cirebon, Majalengka, dan Kuningan. Nurdiawan et al. kemudian menyimpulkan bahwa Genetic Algorithm merupakan algoritma yang sangat cocok untuk menyelesaikan TSP, namun sangat dependen terhadap mekanisme persilangan dan mutasi.

Penelitian lain yang menggunakan Genetic Algorithm untuk menyelesaikan TSP pernah dilakukan oleh Hanif Khan et al.[7]. Permasalahan TSP dengan Time Windows merupakan salah satu bentuk pengembangan masalah TSP, di

mana pada permasalahan tersebut ditambahkan beberapa hal yang menyusahkan penyelesaian masalah, yaitu unsur waktu. Contohnya adalah ditambahkannya waktu pergi dari 1 tempat ke tempat lainnya dan waktu dari tempat wisata yang akan dikunjungi. Unsur waktu juga harus menjadi faktor pertimbangan komputasi GA untuk mendapatkan hasil terbaik. Juwairah et al[8]. menyimpulkan bahwa permasalahan TSP yang dimodifikasi dengan penambahan Time Windows pada rute tempat wisata D.I. Yogyakarta masih dapat diselesaikan menggunakan GA.

Sembiring et al., mengimplementasi TSP dengan menggunakan Ant Colony Optimization (ACO) [9]. Implementasi TSP tersebut digunakan untuk mendapatkan rute terpendek transportasi bahan mentah. ACO menggunakan konsep dari semut ketika mencari makanan dalam suatu colony. Salah satu yang mempengaruhi semut adalah pheromone yang juga digunakan sebagai parameter dalam algoritma ACO. Algoritma ACO terbukti untuk membuat rute terpendek sehingga menghemat jarak tempuh sebesar 37%.

Beberapa penelitian diatas dapat menunjukkan bahwa penelitian sejenis TSP belum pernah menggunakan dengan tiga metode EA yang akan dibahas. Kasus pada tempat wisata Surabaya sangatlah penting untuk diselesaikan agar turis dapat memiliki . Salah satu cara untuk mendapatkan hasil yang efisien dalam permasalahan TSP yaitu adalah dengan menggunakan Evolutionary Algorithm. Penelitian ini akan menggunakan tiga algoritma EA yaitu Cuttlefish Algorithm (CFA)[10], Emperor Penguin Optimizer (EPO)[11], dan Fire Hawk Optimizer (FHO)[12].

Tujuan dari penelitian ini adalah membandingkan tiga algoritma tersebut dan mencari algoritma terbaik yang dapat menghasilkan solusi terbaik untuk Travelling Salesman Problem (TSP) yang diimplementasikan kepada tempat Pariwisata di Surabaya.

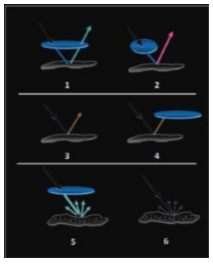
III. ALGORITMA CUTTLEFISH (CFO)

Cuttlefish atau sotong merupakan hewan yang hidup di perairan seperti danau, sungai, dan juga laut[13]. Binatang tersebut memiliki kemampuan untuk mengubah warna dari kulitnya agar dapat melakukan kamuflase dengan lingkungan yang ada disekitarnya[14]. Algoritma cuttlefish dibuat dengan inspirasi bagaimana sebuah sotong melakukan proses mimikri atau mengubah warna kulitnya sesuai dengan tempat atau lingkungannya[15]. Sotong akan melakukan mimikri dengan cara menyesuaikan cahaya yang masuk ke dalam tubuhnya lalu merefleksikannya agar sesuai dengan lingkungannya. Algoritma cuttlefish ditemukan oleh ilmuan bernama Adel Sabry Eesa, Adnan Moshin Abdulazeez Brifcani dan Zeynep Orman[10].

GAMBAR 1. Sotong

Proses kerja dari algoritma mengikuti proses kerja dari sel-sel yang dimiliki oleh kulit sotong, sel-sel yang terdapat pada kulit sotong tersebut ada 3 yaitu Chromatophores, Iridophores, Leucophores dimana sel-sel tersebut akan bekerja secara independen untuk melakukan proses perubahan warna sesuai dengan lingkungannya atau mimikri tersebut, kombinasi dari ketiga sel tersebut akan merefleksikan cahaya yang masuk ke dalam tubuh sotong tersebut. Proses tersebut dapat dinamakan Global Optimum Solution[14].

Algoritma cuttlefish memiliki tujuan untuk mendapatkan bentuk dengan melihat mekanisme perubahan warna kulit yang dilakukan oleh sotong. Algoritma cuttlefish diimplementasikan dengan cara dibagi menjadi 6 bagian yang disebut kasus yang akan dikumpulkan dalam 4 grup. Kasus tersebut adalah perwakilan dari mekanisme perubahan warna kulit dari hewan sotong. Kasus tersebut memiliki 2 proses penting, yaitu reflection dan visibility. Reflection merupakan proses representasi dari mekanisme yang digunakan sotong untuk merefleksikan cahaya yang masuk. Sedangkan visibility adalah representasi dari hasil simulasi bentuk dan pola yang ditunjukkan ke dalam lingkungannya.



GAMBAR 2. Kasus pada Algoritma Cuttlefish

Inisialisasi awal adalah langkah yang harus dilakukan pada algoritma Cuttlefish, inisialisasi yang harus dilakukan adalah mengisi populasi dengan solusi acak, dan juga melakukan input nilai pada variabel r1, r2, v1, dan v2. Perhitungan untuk mendapatkan solusi baru akan dilakukan setelah proses input nilai variabel yang telah dilakukan, lalu populasi-populasi tersebut akan dibagi menjadi 4 grup sesuai dengan kasus-kasus yang terdapat pada Gambar 2. Rumus yang digunakan untuk menemukan solusi baru ($newp$) adalah dengan

menjumlahkan hasil dari $reflection$ dan $visibility$ seperti pada (1).

$$newp = reflection + visibility \quad (1)$$

Grup 1 (kasus 1 dan kasus 2) adalah bagian awal perjalanan algoritma, grup 1 adalah proses interaksi antara sel chromatophores dan sel iridophores yang melakukan proses relaksasi dan kontraksi agar dapat melakukan proses refleksi dan relaksasi sesuai dengan lingkungan yang ada di sekitarnya, proses akan direpresentasikan dalam bentuk rumus sesuai dengan (2) dan (3).

$$reflection = R * G_1[i].Points[j] \quad (2)$$

$$visibility_j = V * Best.Points[j] - G_1[i].Points[j] \quad (3)$$

Chromatophores adalah sel yang disimulasikan pada (2) dan (3). R pada (2) merepresentasikan untuk menghitung jarak dari peregangan yang dilakukan oleh otot pada sel tersebut dalam kontraksi atau relaksasi dan V pada (3) merepresentasikan derajat terlihatnya pola tersebut pada hasil akhirnya. Nilai dari R dan V adalah sebagai berikut:

$$R = random() * (r_1 - r_2) + r_2 \quad (4)$$

$$V = random() * (v_1 - v_2) + v_2 \quad (5)$$

Random di sini adalah angka acak antara (0,1). Variabel r_1 dan r_2 merupakan konstanta yang digunakan untuk menentukan interval regangan dari sel chromatophors, sedangkan v_1 dan v_2 adalah konstanta yang digunakan untuk menentukan derajat dari hasil akhir pola yang dimiliki. Nilai dari R atau V pada (4) dan (5) dapat langsung bernilai 1 atau dapat dikalkulasikan.

Grup 2 (kasus 3 dan kasus 4) adalah simulasi dari iridophores yang akan melakukan refleksi cahaya dari luar, dan akan merefleksikan warna tertentu. Iridophores membantu proses menutupi oragan dan direpresentasikan sebagai best solution. Rumus yang digunakan pada grup 2 memiliki perubahan pada reflection dimana rumus tersebut dituliskan menjadi seperti berikut:

$$reflection_j = R * Best.Point[j] \quad (6)$$

Rumus R akan menjadi 1, sedangkan V akan dihitung ulang. Grup 2 akan melakukan proses local search dan menggunakan perbedaan dari best solution dan hasil terbaik sekarang untuk menemukan interval disekitar solusi terbaik sebagai area baru yang terjadi setelah rumus (6).

Grup 3 (kasus 5) adalah representasi dari sel leucophores, sel akan bekerja sebagai cermin. Sel Kemudian akan melakukan refleksi pada cahaya lingkungannya yang masuk ke dalam sel. Untuk menemukan warna yang akan direfleksikan, maka dapat diasumsikan bahwa warna yang datang adalah solusi terbaik (best). Interval yang digunakan adalah interval diantara best untuk menentukan visibility. Rumus yang digunakan pada grup 3 merupakan hasil modifikasi dari (2) dan (3) yaitu sebagai berikut:

$$reflection_j = R * Best.Points[j] \quad (7)$$

$$visibility = V * (Best.Points[j] - AV_{Best}) \quad (8)$$

AV_{Best} merupakan nilai dari rata-rata best points. Nilai dari R pada rumus adalah 1, sedangkan V akan dilakukan kalkulasi ulang. Hasil rumus modifikasi dari (2) adalah (7) dan hasil modifikasi dari (3) adalah (8).

Grup 4 (kasus 6) adalah grup terakhir dari algoritma cuttlefish yaitu sel leucophores akan melakukan refleksi cahaya dari lingkungannya. Hal ini akan membantu proses sotong untuk mengubah warna dari kulitnya agar sesuai dengan lingkungannya. Simulasi yang dapat diasumsi dari kasus 4 adalah representasi dari solusi acak yang baru. Inisialisasi nilai acak akan digunakan pada kasus 4.

IV. ALGORITMA EMPEROR PENGUIN OPTIMIZER (EPO)

Emperor Penguin Optimizer adalah sebuah algoritma optimasi yang terinspirasi dari perilaku emperor penguin [11]. Algoritma EPO menggunakan kombinasi dari eksplorasi dan eksploitasi untuk melakukan pencarian solusi optimal pada sebuah masalah yang sedang dihadapi. Algoritma ini menggunakan sebuah grup artificial penguin, yang setiap penguinnya adalah hasil representasi dari sebuah potensi dari solusi masalah tersebut. Simulasi perilaku dari penguin dilakukan dengan meniru etika penguin dalam melakukan pencarian makanan pada kondisi ekstrim Antartika. Algoritma EPO dapat melakukan adaptasi pada kondisi dan juga dapat melakukan proses hingga konvergen untuk mendapatkan solusi terbaik dengan lebih cepat. Sehingga mendapatkan posisi penguin (solusi) yang sesuai dengan memiliki nilai fitness yang tinggi.

GAMBAR 3. Emperor Penguin

Algoritma EPO akan dibagi menggunakan fase-fase yang disesuaikan dengan perilaku emperor penguin saat melakukan proses bertahan hidup di kondisi ekstrim yang terdapat pada pedalaman antartika. Fase-fase yang akan dilakukan saat melakukan kerumunan adalah membuat dan menentukan

batasannya, lalu melakukan kalkulasi dari batasannya, menentukan jarak antara penguin, dan mengubah posisi dari effective mover. Fitur yang penting dalam proses perilaku gerombolan adalah setiap penguin mendapatkan kesempatan yang sama untuk mendapatkan kehangatan.

Algoritma EPO adalah salah satu implementasi dari hasil model matematika pada perilaku gerombolan emperor penguin. Pertama, para emperor penguin akan melakukan inisialisasi dari jumlah gerombolan secara acak. Lalu, temperatur dari gerombolan tersebut akan dihitung. Jarak antara emperor penguin juga akan dikalkulasikan agar dapat membantu proses eksplorasi dan juga eksploitasinya. Effective mover akan didapatkan setelah semua proses tersebut telah dijalankan dan optimal solution tersebut ditemukan dan akan melakukan proses perhitungan ulang dari batasan gerombolan dan juga posisi dari emperor penguin yang sudah diperbaharui apabila memiliki kehangatan yang lebih baik dari sebelumnya. Proses-proses implementasi mengenai EPO akan dibahas lebih jauh pada subbab berikutnya.

Emperor penguin melakukan proses bergerombol agar dapat menyimpan energi dan memaksimalkan temperatur dari gerombolan tersebut. Implementasi model matematis dari proses adalah melakukan asumsi bahwa temperatur $T = 0$ saat radius dari poligon $R > 0.5$ dan temperatur $T = 1$ saat radius menjadi $R \leq 0.5$. Temperatur T mengatur proses eksplorasi dan ekploitasi pada emperor penguin saat lokasinya berbeda. Rumus temperatur saat berada di dalam gerombolan T'adalah sebagai berikut:

$$T' = \left(T - \frac{Max_{iteration}}{x - Max_{iteration}}\right) \quad (9)$$

$$T = \begin{Bmatrix} 0, \ if \ R > 0.5 \\ 1, \ if \ R <= 0.5 \end{Bmatrix}$$

Di mana x pada (9) adalah iterasi saat ini, $Max_{iteration}$ merupakan nilai iterasi maksimal, R adalah radius, dan T adalah waktu untuk menentukan solusi optimal dalam search space.

Perhitungan jarak antar penguin dan perhitungan untuk mendapatkan optimal solusi terbaik dilakukan setelah mendapatkan batasan dari gerombolan. Solusi optimal saat ini adalah solusi dari nilai fitness yang paling tinggi atau posisi dengan temperatur yang paling hangat. Emperor penguin yang lain akan melakukan proses perpindahan sesuai dengan solusi optimal terbaik dengan rumus berikut:

$$\vec{D}_{ep} = Abs\left(S(\vec{A}).\vec{P}(x) - \vec{C}.\vec{P}_{ep}(x)\right) \quad (10)$$

$\vec{D}_{ep}$ adalah representasi dari jarak antara penguin dan penguin yang memiliki nilai fitness terbaik, sedangkan x pada (10) adalah iterasi saat ini. Vektor $\vec{A}$ dan $\vec{C}$ digunakan untuk menghindari tabrakan antara sesama penguin. Vektor $\vec{P}$ menotasikan emperor penguin dengan solusi paling optimal (fitness tertinggi). Terakhir, vektor $\vec{P}_{ep}$ merupakan vektor posisi pada iterasi x dari setiap emperor penguin. Fungsi$S()$ merepresentrasikan nilai dorongan yang dimiliki oleh emperor penguin untuk bergerak mendekati emperor penguin yang memiliki solusi terbaik. Fungsi S() dapat dihitung dengan rumus (14). Vektor$\vec{A}$ dan $\vec{C}$ dapat dihitung dengan rumus berikut:

$$\vec{A} = \left(M * \left(T' + P_{grid}(Accuracy)\right) * Rand()\right) - T' \quad (11)$$

$$P_{grid}(Accuracy) = Abs\left(\vec{P} - \vec{P}_{ep}\right) \quad (12)$$

$$\vec{C} = Rand() \quad (13)$$

Pada rumus (11), M adalah parameter yang menggambarkan nilai pergerakan untuk menjaga jarak antara penguin agar tidak terjadi tabrakan. Nilai M akan diisi sebesar 2. T' adalah nilai temperatur di sekitar gerombolan. $P_{grid}(Accuracy)$ adalah akurasi dari poligon yang dapat dihitung dengan melakukan perbandingan lokasi setiap emperor penguin dengan emperor penguin paling optimal. Perbandingan tersebut dirumuskan pada rumus (12). $Rand()$ pada rumus (13) adalah nilai acak antara [0, 1].

$$S\left(\vec{A}\right) = \left(\sqrt{f \, . e^{-x/l} . - e^{-x}}\right)^2 \quad (14)$$

Pada rumus (14), e adalah fungsi ekspresi. Sedangkan notasi f dan l adalah parameter kontrol untuk mendapatkan hasil eksplorasi dan eksploitasi yang baik. f dan l masing-masing memiliki yang berada pada rentang [2, 3] dan [1.5, 2] secara berurutan. Nilai tersebut telah terbukti untuk mendapatkan nilai yang lebih baik dalam menemukan solusi.

Fase terakhir dalam algoritma EPO adalah melakukan relokasi dari emperor penguin yang dilakukan berdasarkan hasil kalkulasi sebelumnya dan disesuaikan dengan hasil terbaik dari solusi optimal. Emperor penguin yang terbaik akan bertanggung jawab untuk melakukan perubahan posisi dari emperor penguin lainnya. Rumus yang digunakan untuk melakukan perubahan adalah sebagai berikut:

$$\overrightarrow{P_{ep\_new}} = \vec{P}(x) - \vec{A} \, . \vec{D}_{ep} \quad (15)$$

$$\overrightarrow{P_{ep}}(x+1) = \begin{Bmatrix} \overrightarrow{P_{ep\_new}}, \ if \ \overrightarrow{P_{ep}}(x) < \overrightarrow{P_{ep\_new}} \\ \overrightarrow{P_{ep}}(x), \ if \ \overrightarrow{P_{ep}}(x) \geq \overrightarrow{P_{ep\_new}} \end{Bmatrix} \quad (16)$$

Di mana rumus (16) merepresentasikan posisi terbaru dari emperor penguin yang telah melakukan perubahan. Menghitung Pep baru dapat dilakukan dengan menggunakan rumus (15). Pada rumus (15), nilai P(x) didapatkan dari penguin terbaik pada iterasi sebelumnya, sedangkan A

bersumber dari rumus (11), dan Dep dari rumus (10). Posisi terbaru dari emperor penguin ditentukan dari hasil fitness emperor penguin dengan posisi lama dibandingkan dengan yan baru. Posisi emperor penguin akan terubah hingga iterasi mencapai maksimal iterasi.

V. ALGORITMA FIRE HAWK OPTIMIZER (FHO)

Fire Hawk merupakan algoritma metaheuristik yang dibuat dengan inspirasi perilaku berburu dari beberapa burung seperti whistling kites, black kites, dan brown falcons. Burung – burung tersebut dinamai Fire Hawk dikarenakan terdapat aksi spesifik yang dilakukan dalam melakukan proses pencarian mangsa yang dilakukan di habitatnya, yang disebut dengan menyalakan api [12]. Algoritma Fire Hawks dengan sengaja mencoba untuk menyalakan api dengan membawa ranting yang terbakar pada paruhnya, yang dilaporkan sebagai fenomena penghancuran di alam.

GAMBAR 4. Fire Hawk Disekitar Api

Sebagai mekanisme untuk melakukan kontrol dan mendapatkan mangsanya, burung dapat mengambil ranting untuk membakar sebagian kecil dari lahan untuk menakuti mangsanya seperti tikus, ular, dan hewan lainnya. Hewan yang kabur akan dengan tergesa-gesa berlarian yang membuat hewan-hewan tersebut sangat mudah ditangkap oleh hawks. Fire Hawk Optimizer akan melakukan inisialisasi dengan cara menentukan jumlah kandidat solusi yang akan menentukan posisi vektor dari mangsa dan fire hawks. Inisialisasi acak akan dilakukan saat menentukan posisi awal dari vektor-vektor tersebut.

$$X = \begin{bmatrix} X_1 \\ X_i \\ X_N \end{bmatrix} = \begin{bmatrix} X_1^1, X_1^2, X_1^j \dots X_1^d \\ X_i^1, X_i^2, X_i^j \dots X_i^d \\ X_N^1, X_N^2, X_N^j \dots X_N^d \end{bmatrix}, \begin{Bmatrix} i = 1, 2, \dots, N. \\ j = 1, 2, \dots, d. \end{Bmatrix} \quad (17)$$

Rumus yang digunakan untuk mencari X pada (17) adalah sebagai berikut:

$$X_i^j(0) = X_{i,min}^j + rand.\left(X_{i,max}^j - X_{i,min}^j\right), \begin{Bmatrix} i = 1, 2, \dots, N. \\ j = 1, 2, \dots, d. \end{Bmatrix} \quad (18)$$

Pada rumus (18), X_i merupakan solusi ke-i dalam lingkup pencarian; variable d merupakan representasi dari dimensi dalam masalah yang sedang dicari; N adalah total dari solusi yang merupakan kandidat dalam lingkup pencarian; X_i^j merupakan variabel penentu ke-j dari solusi ke-i; $X_{i,max}^j$ dan $X_{i,min}^j$ merupakan batasan maksimal dan minimal dari variabel penentu ke-j dan kandidat solusi ke-I; $rand$ adalah nilai acak yang memiliki rentang antara 0 sampai 1.

Untuk mendapatkan lokasi dari Fire Hawk yang terdapat pada lingkup pencarian, Representasi dari Fire Hawk akan digunakan jika kandidat solusi yang memiliki fungsi objektif yang lebih baik, sedangkan representasi dari mangsa adalah semua selain Fire Hawk tersebut.

Setelah melakukan inisialisasi tersebut, maka perhitungan untuk menentukan jumlah dari jarak antara Fire Hawk dan mangsa-mangsanya. Hasil dari perhitungan tersebut adalah menemukan mangsa terdekat dari setiap burung, yang akan membantu untuk menemukan jangkauan efektif dari burung-burung tersebut. Rumus yang digunakan untuk mencari jarak adalah sebagai berikut:

$$D_k^l = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}, \begin{Bmatrix} l = 1,2, \dots, n. \\ k = 1,2, \dots, m. \end{Bmatrix} \quad (19)$$

Pada rumus (19), D_k^l merupakan total jarak antara Fire Hawk ke-l dan mangsa ke-k; m adalah jumlah dari semua mangsa yang ada dalam lingkup pencarian; n adalah jumlah dari semua fire hawk yang terdapat pada lingkup pencarian; dan (x_1, y_2) dan (x_2, y_2) adalah representasi dari koordinat Fire Hawks dan juga mangsa yang terdapat pada lingkup pencarian.

Setelah menemukan jarak dari Fire Hawks dan juga mangsanya, proses selanjutnya adalah untuk melakukan pengumpulan ranting yang terbakar, dengan cara mengambil api dari titik yang menjadi pusat dari api agar dapat melakukan pembakaran pada tempat yang telah ditentukan. Pada proses ini burung akan melakukan pengambilan ranting terbakar lalu menjatuhkannya ditempat yang telah ditentukan yang akan membuat mangsa kabur secara tergesa-gesa. Rumus yang digunakan untuk melakukan update posisi pada proses ini ialah sebagai berikut:

$$FH_l^{new} = FH_l + (r_1 * GB - r_2 * FH_{Near}), l = 1,2,..,n \quad (20)$$

Pada rumus (20), FH_l^{new} adalah representasi dari posisi baru dari Fire Hawk ke-l; GB merupakan solusi terbaik atau Global Best dari lingkup pencarian; FH_{Near} adalah salah satu Fire Hawk yang terdapat pada lingkup pencarian; dan r_1 dan $r_2$adalah nulai acak yang berada pada nilai antara (0, 1) untuk menentukan pergerakan Fire Hawks kearah pusat api dan kearah Fire Hawk lainnya.

Proses selanjutnya dari algortima FHO adalah pergerakan mangsa yang berada pada wilayah dari setiap Fire Hawk, hal ini merupakan hal yang penting dari mangsa untuk melakukan perubahan posisi. Saat Fire Hawk melakukan pembakaran dengan menjatuhkan ranting disekitar wilayah mangsa, mangsa tersebut dapat memilih untuk bersembunyi, kabur,

atau berlari kearah Fire Hawk karena panik. Aksi tersebut dapat dirumuskan dalam bentuk sebagai berikut:

$$PR_q^{new} = PR_q + (r_3 * FH_l - r_4 * SP_1), \begin{Bmatrix} l = 1,2,\dots,n. \\ q = 1,2,\dots,r. \end{Bmatrix} \quad (21)$$

Pada rumus (21), PR_q^{new} merupakan posisi terbaru dari mangsa ke-q yang dikelilingi oleh Fire Hawk (FH_l); SP_1 merupakan tempat yang aman saat dikelilingi oleh Fire Hawk ke-l; r_3 dan r_4 merupakan nilai acak yang berada diantara (0, 1) untuk menentukan pergerakan dari mangsa tersebut.

Mangsa memiliki kemungkinan untuk melakukan perpindahan menuju ke wilayah Fire Hawk lainnya. Berikut merupakan rumus yang menggambarkan perubahan tempat mangsa:

$$PR_q^{new} = PR_q + (r_5 * FH_{Alter} - r_6 * SP), \begin{Bmatrix} l = 1,2,\dots,n. \\ q = 1,2,\dots,r. \end{Bmatrix} \quad (22)$$

Pada rumus (22), PR_q^{new} merupakan posisi terbaru dari mangsa ke-q; FH_{Alter} merupakan Fire Hawk lainnya yang terdapat pada lingkup pencarian ini; dan r_5 dan r_6 merupakan nilai acak yang berada pada nilai antara (0, 1) untuk menentukan pergerakan dari mangsa tersebut.

Safe Place adalah tempat yang aman yang dapat dikunjungi oleh mangsa dan dalam algoritma SP_1 dan SP dirumuskan sebagai berikut:

$$SP_1 = \frac{\sum_{q=1}^{r} PR_q}{r}, \begin{Bmatrix} q = 1,2,\dots,r. \\ l = 1,2,\dots,n. \end{Bmatrix} \quad (23)$$

$$SP = \frac{\sum_{k=1}^{m} PR_k}{m}, k = 1,2,\dots,m. \quad (24)$$

Pada rumus (23, 24), PR_q adalah mangsa ke-q yang sedang dikelilingi oleh Fire Hawk ke-l; PR_k adalah mangsa ke-k yang berada pada lingkup pencarian.

VI. DATA

Data yang digunakan dalam penelitian adalah lokasi tempat wisata yang berada pada Surabaya, serta beberapa hotel bintang empat dan lima yang ditambahkan sebagai tempat penginapan. Data tersebut didapatkan dengan menggunakan Open Road Service, dimana website tersebut menyediakan API yang dapat digunakan untuk membantu proses pathfinding atau penemuan jalan dari satu titik ke titik yang lain, lalu untuk data gambar map yang digunakan didapatkan dengan menggunakan Open Street Map. Data lain yang digunakan pada penelitian lain adalah data hotel bintang empat dan lima. Data tersebut didapatkan dengan library bantuan library bantuan bernama Beautiful Soup, yang merupakan sebuah library yang dapat digunakan dengan bahasa pemrograman python untuk melakukan proses parsing data HTML atau XML. Lokasi-lokasi pariwisata merupakan data yang terpenting dalam penelitian, data tersebut didapatkan dengan menggunakan website yang telah disediakan oleh pemerintah yaitu website tourism Surabaya.

Data lokasi tempat wisata yang telah didapatkan tersebut akan melalui proses pembersihan data, dikarenakan beberapa tempat wisata yang tidak relevan untuk penelitian. Data preprocessing akan melakukan proses seperti membersihkan data-data yang kurang relevan seperti "bangunan kosong", "apotek", "CIMB NIAGA". Karena tidak sesuai dengan tema pada penelitian yaitu tempat wisata maka dari-data yang dikumpulkan dibersihkan terlebih dahulu sebelum digunakan dalam program. Data yang dikumpulkan sebelum pembersihan data atau preprocessing adalah 447 data lokasi tempat wisata, lalu setelah dilakukan proses pembersihan data, jumlah data berkurang menjadi 123 data lokasi wisata.

VII. IMPLEMENTASI ALGORITMA

Algoritma-algoritma diatas akan diimplementasikan pada program yang telah dibuat khusus untuk memecahkan masalah dengan bantuan hasil representasi dari data yang telah disediakan, algoritma-algortima akan mengambil 3 hasil terbaik dengan nilai yang terendah. Implementasi dari program-program tersebut akan dijelaskan lebih lanjut pada subbab-subbab selanjutnya.

GAMBAR 5. Representasi Solusi

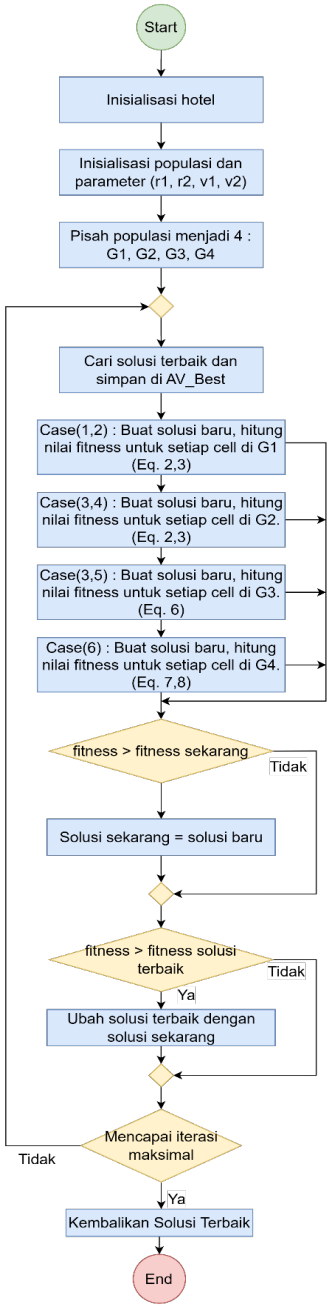
A. REPRESENTASI SOLUSI

Pada implementasi tiga algoritma yang dijelaskan pada bab ini akan menggunakan representasi solusi yang sama. Representasi digambarkan pada Gambar 5. Representasi berupa sebuah array satu dimensi dimana setiap elemennya merupakan suatu ID dari lokasi wisata. Jumlah dari elemen array merupakan jumlah semua tempat wisata yang ada yaitu 123 tempat wisata yang terletak pada surabaya. Index pada array akan merepresentasikan urutan tempat wisata yang harus dikunjungi. Ketika fitness dihitung, array akan dibagi berdasarkan jumlah hari dan kunjungan tempat lokasi wisata per hari.

Pada awalnya pembagi dari array akan ditentukan berdasarkan jumlah kunjungan per hari. Namun ketika jumlah pengalian dari hari dan kunjungan per hari lebih dari jumlah tempat wisata maka pembagi akan didapatkan dari pembagian jumlah tempat wisata dengan kunjungan per hari. Namun jumlah hari pada input program tidak dapat melebihi dari jumlah tempat wisata pada data dikarenakan hanya akan

mengurutkan tempat wisata kota yang terdekat dari tempat tinggal dimana hal tersebut kurang sesuai dengan tujuan penelitian.

Representasi yang berupa array satu dimensi dapat divisualisasikan sebagai array dua dimensi yang merupakan alat bantu untuk memudahkan ketika menghitung nilai fitnessnya. Dimensi pertama merupakan jumlah hari yang berasal dari input parameter. Dimensi kedua merupakan lokasi wisata yang hendak dikunjungi. Pada implementasinya akan memperhitungkan hotel sebagai perantara setiap harinya dengan anggapan bahwa wisatawan akan selalu berangkat dari hotel dan pulang ke hotel setiap harinya. Cara kerja dari fitness function akan dijelaskan lebih lanjut pada bagian E.



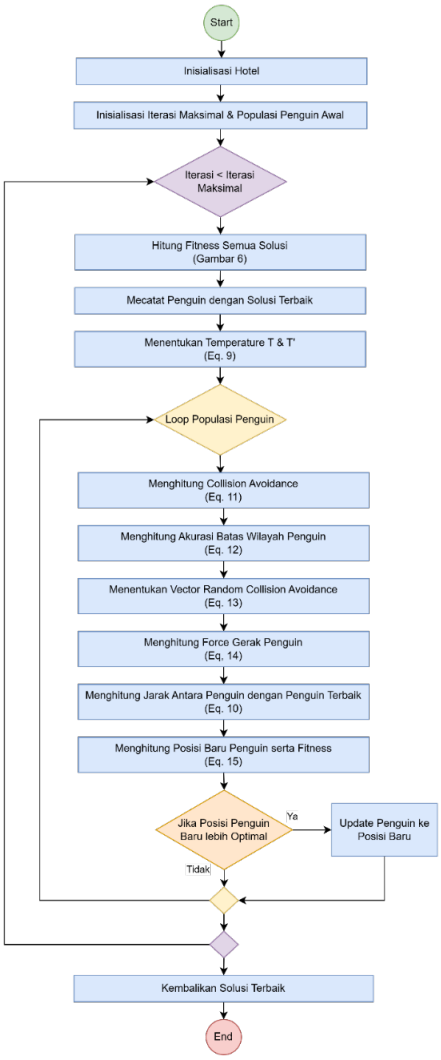
GAMBAR 6. Diagram Alur Cuttlefish Algorithm

B. CUTTLEFISH ALGORITHM

Algoritma cuttlefish diinisialisasi dengan mengatur semua solusi yang dimiliki untuk dibagi ke dalam kelompok-kelompok yang telah ditentukan, yaitu kelompok 1, kelompok 2, kelompok 3, dan kelompok 4. Setelah menentukan grup-grup tersebut lalu parameter-parameter yang diperlukan seperti r1, r2, v1 dan v2. Kelompok 1 akan menjadi representasi dari proses sotong untuk melakukan proses refleksi dari cahaya yang masuk, kelompok 2 adalah proses visibility dimana pada kelompok ini menentukan apakah cahaya akan diserap kembali atau dipantulkan. Kelompok 3 adalah proses mendapatkan mendapatkan reflection dari cahaya yang terdapat pada area sekitar best solution, dan kelompok 4. Loop akan dilakukan selama stopping condition belum terpenuhi, hal yang pertama terjadi saat melakukan proses looping adalah mencari average best dari semua solusi yang ada, lalu proses selanjutnya adalah melakukan proses-proses tertentu sesuai dengan kelompok yang telah dibagi pada proses awal. Program akan berhenti setelah stopping condition tercapai dan akan mengambil 3 nilai terendah untuk menjadi hasil terbaik sebagai output algoritma cuttlefish. Visualisasi dalam bentuk diagram terdapat pada Gambar 6.

C. EMPEROR PENGUIN OPTIMIZER

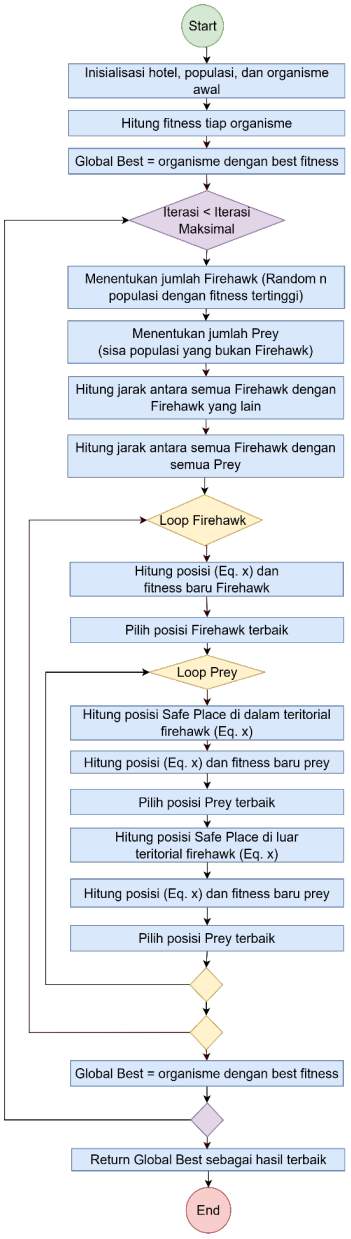
EPO diawali dengan mengatur beberapa parameter yang dibutuhkan, seperti maksimal iterasi, maksimal penguin, konstanta f, M dan l serta batas bawah dan batas atas. Pada EPO, seekor penguin merepresentasikan sebuah solusi yang merupakan sebuah array satu dimensi yang berisi sejumlah lokasi wisata yang akan dikunjungi. Algoritma akan berjalan selama iterasinya belum menyentuh batas maksimal iterasi yang ditentukan. Setiap iterasinya diawali dengan menghitung fitness untuk setiap penguin yang ada dan mencari penguin dengan fitness terbaik. Setelah itu loop untuk setiap penguin yang ada dan hitung fitness baru untuk penguin tersebut. Apabila fitness yang baru lebih baik daripada yang lama, posisi penguin akan tergantikan dengan yang baru sesuai dengan fitness yang terbaik. Posisi penguin juga perlu disesuaikan mengikuti batas bawah dan batas atas yang merupakan parameter di awal program dimana jika melebihi akan digantikan dengan nilai pada parameter tersebut. Setelah algoritma sudah berjalan sebanyak jumlah maksimal iterasi, penguin dengan nilai fitness tinggi akan digunakan sebagai solusi. Visualisasi proses EPO dalam bentuk diagram terdapat pada Gambar 7.



GAMBAR 7. Diagram Alur Emperor Penguin Optimizer

D. FIREHAWK OPTIMIZER

Algoritma FHO diimplementasi dengan menggunakan sejumlah parameter di dalam alur programnya yaitu seed, maksimal iterasi, besar populasi, daftar solusi, daftar lokasi wisata, daftar hotel, solusi pertama, firehawk terbaik, dan configs. Seed merupakan seed untuk random. Maksimal iterasi merupakan jumlah maksimal iterasi. Besar populasi merupakan jumlah kandidat solusi. Daftar solusi digunakan untuk menampung 3 solusi terbaik yang akan dihasilkan. Daftar lokasi wisata digunakan untuk menyimpan daftar tempat parawisata. Daftar hotel digunakan untuk menyimpan daftar hotel yang ada yaitu berjumlah 40 data. Solusi pertama digunakan untuk menampung inisialisasi dari solusi awal. Firehawk terbaik digunakan untuk menampung firehawk terbaik. Visualisasi dalam bentuk diagram terdapat pada Gambar 8.



GAMBAR 8. Diagram Alur Fire Hawk Optimizer

Pertama-tama akan dilakukan inisialisasi posisi dari setiap kandidat solusi dengan fitness awal adalah 0. Kemudian akan dilakukan pengurutan kandidat solusi berdasarkan fitness dengan nilai fitness tertinggi di paling depan. Kandidat solusi berupa sebuah array satu dimensi yang berisi sejumlah lokasi wisata yang akan dikunjungi. Setelah itu akan dilakukan kalkulasi fitness terhadap setiap kandidat solusi. Global best awal akan ditentukan. Bagian-bagian selanjutnya ini akan diiterasikan sebanyak maksimal iterasi. Jumlah firehawks akan diacak dalam rentang 1 hingga besar populasi-1 dan sisanya adalah jumlah preys. Kemudian dari daftar solusi akan ditentukan mana yang merupakan firehawks dan sisanya

merupakan preys. Setelah itu akan dilakukan kalkulasi jarak antara firehawks dan preys. Selanjutnya akan dilakukan kalkulasi jarak antar firehawks. Wilayah yang terdampak firehawks akan ditentukan. Kemudian akan dilakukan pembaruan posisi dan fitness setiap firehawk. Setelah itu untuk setiap prey yang terdampak di wilayah firehawk, dilakukan kalkulasi safe place. Posisi dan fitness setiap prey akan diperbarui. Selanjutnya untuk setiap prey yang diluar wilayah firehawk, dilakukan kalkulasi safe place. Posisi dan fitness setiap prey akan diperbarui. Setelah itu apabila fitness dari firehawk terbaik pada iterasi tersebut lebih baik dari pada global best saat ini, maka global best akan diperbarui. Tiga solusi terbaik kemudian akan didapatkan.

E. FITNESS FUNCTION

Pada bagian ini akan dijelaskan mengenai fitness function yang akan digunakan pada semua algoritma. Fitness function ini digunakan untuk menghitung seberapa dekat antara solusi dan spektrum neutron referensi[16]. Fitness function dihitung dengan menjumlahkan jarak dari hotel yang ditempati menuju ke tempat wisata tujuan. Di awal hari akan menghitung jarak dari hotel ke tempat wisata tujuan dan di akhir hari akan menghitung jarak dari tempat wisata terakhir di hari tersebut ke hotel. Lalu dari total jarak mulai dari hotel pada hari pertama hingga kembali ke hotel pada hari terakhir akan dikali dengan -1 agar semakin besar fitness score semakin baik. Tujuan dari fitness function yang digunakan adalah mendapatkan jarak yang tempuh yang paling rendah untuk mengunjungi tempat pariwisata. Pseudocode untuk rumus fitness function terdapat pada Gambar 9.

```
procedure fitness function ()
  inisialisasi total = 0
  for l=1:n (semua tempat wisata tujuan)
    if l adalah tempat wisata pertama di suatu hari
      total = total + jarak(hotel ke l)
    if l adalah tempat wisata terakhir di suatu hari
      total = total + jarak(l-1 ke l)
      total = total + jarak(l ke hotel)
    else
      total = total + jarak(l-1 ke l)
  end
  total = total * -1
  return total
end procedure
```

GAMBAR 9. Pseudocode Fitness Function

VIII. UJI COBA

Pada bagian ini dijelaskan hasil dari perbandingan implementasi beberapa algoritma untuk menyelesaikan masalah tersebut. Terdapat beberapa variasi uji kasus yang dicoba untuk penelitian dimana terdapat parameter input jumlah hari dan kunjungan per hari, dimana pada kasus pertama menggunakan 30 hari yang setiap harinya mengunjungi 4 tempat, kasus kedua menggunakan 14 hari yang setiap harinya mengunjungi 5 tempat, kasus ketiga menggunakan 7 hari yang setiap harinya mengunjungi 5 tempat, dan kasus keempat menggunakan 3 hari yang setiap harinya mengunjungi 6 tempat. Uji coba pada penelitian ini menggunakan salah satu hotel yang didapatkan yaitu hotel Sheraton yang terletak pada Jl. Embong Malang No. 25-31. Pemilihan hotel tersebut dikarenakan 2 alasan yaitu pertama hotel yang dipilih merupakan hotel bintang lima karena umumnya akan menjadi pilihan bagi banyak turis, alasan kedua, hotel yang dipilih adalah hotel yang berada di daerah tengah kota untuk menjangkau lokasi pariwisata yang lebih beragam. Penelitian ini juga melakukan uji coba terhadap parameter khusus untuk masing-masing algoritma yang ditentukan dan digunakan pada awal algoritma dijalankan.

Pengujian parameter digunakan agar dapat menghasilkan fitness score terbaik dengan kombinasi paramater yang digunakan. Fitness score semakin baik bila nilai yang tertulis semakin besar. Terdapat 4 variasi uji kasus dilakukan pada penelitian seperti yang telah dijelaskan sebelumnya dimana setiap uji kasus memiliki jumlah hari dan jumlah kunjungan per hari yang berbeda. Jumlah kunjungan per hari menunjukkan berapa jumlah tempat yang dikunjungi setiap harinya sedangkan jumlah hari menunjukkan berapa lama perjalanan pariwisata tersebut dilakukan. Pada akhirnya jumlah tempat yang dikunjungi adalah jumlah hari dikali jumlah kunjungan per hari, seperti pada kasus pertama 30 hari dengan 4 kunjungan perhari memiliki arti bahwa jumlah tempat yang akan dikunjungi berjumlah 120. Keempat uji kasus dijelaskan pada Tabel I.

TABEL I.
VARIASI UJI KASUS

No	Jumlah Hari	Jumlah kunjungan per hari
1	30	4
2	14	5
3	7	5
4	3	6

Tentunya terdapat beberapa pengaturan parameter untuk algoritma-algoritma yang dibandingkan pada penelitian. Faktor parameter pada algoritma EPO telah dicoba berdasarkan dengan kombinasi dari nilai konstan f, M, dan l. Algoritma CFA juga memiliki beberapa parameter yang dapat mempengaruhi hasil fitness yaitu r1, r2, v1, dan v2. Sedangkan algoritma FHO tidak memiliki pengaturan parameter yang

dapat dilakukan uji coba. Rentang nilai parameter untuk setiap algoritma dijelaskan pada Tabel II.

TABEL II.
RENTANG NILAI PARAMETER

No	Algoritma	Parameter
1	Emperor Penguin Optimizer (EPO)	*f: [2-3]; M: [1-3]; l: [1.5-2]*
2	Firehawk Optimizer (FHO)	-
3	Cuttlefish Algorithm (CFA)	*r1: [-2-2]; v1: [-2-2]; r2: [-2-2]; v2: [-2-2]*

Pada setiap uji kasus, akan dilakukan menggunakan 250, 500, 1000, dan 1500 iterasi. Namun khusus uji coba pada algoritma FHO, pada percobaan iterasi 1000 dan 1500 hanya disimpan hasil yang memiliki nilai fitness terbaik, sehingga hasil yang memiliki nilai fitness kurang baik tidak tersimpan.

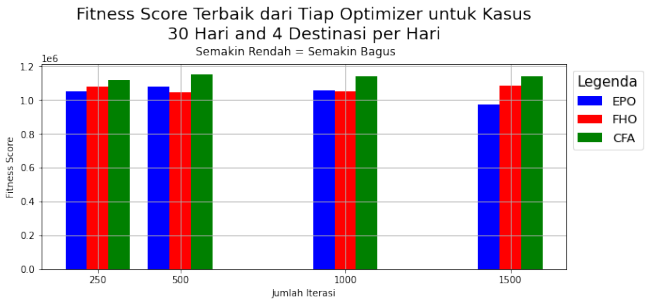
A. UJI KASUS #1

Uji kasus ini akan membahas untuk hari berjumlah 30 dan terdapat maksimal 4 kunjungan per hari. Nilai fitness terbaik yang didapatkan adalah -970963.64 yang diperoleh menggunakan algoritma EPO. Tabel III adalah tabel yang menunjukkan nilai fitness terbaik untuk setiap algoritma pada uji kasus 30 hari dan maksimal 4 kunjungan per hari.

TABEL III.
FITNESS TERBAIK UJI KASUS #1

Algoritma	No	Fitness	Iterasi	Pop ulasi	Parameter
EPO	1	-970963.64	1500	100	*f*: 2.6; *M*: 1; *l*: 2.1
	2	-1053594.97	250	100	*f*: 3; *M*: 1; *l*: 1.9
	3	-1055009.62	1000	100	*f*: 3; *M*: 2; *l*: 1.5
FHO	1	-1043266.04	500	50	-
	2	-1053691.75	1000	50	
	3	-1078990.4	500	50	
CFA	1	-1142067.97	1000	100	*r1*: -2; *r2*: -2; *v1*: 1; *v2*: -2
	2	-1140326.61	1500	100	*r1*: -2; *r2*: -1; *v1*: 1; *v2*: 1
	3	-1145331.48	1500	100	*r1*: -2; *r2*: -1; *v1*: 2; *v2*: 0

Data pada Tabel III dapat disimpulkan menjadi suatu grafik yang dapat dilihat pada Gambar 10. Gambar 10 memberikan visualisasi data mengenai perbandingan nilai fitness setiap algoritma berdasarkan jumlah iterasi yang dilakukan. Optimisasi dengan menggunakan EPO memiliki fitness paling baik pada 250 iterasi dan 1500 iterasi, sedangkan optimisasi dengan menggunakan FHO lebih baik pada 500 dan 100 iterasi. Pada uji kasus #1, EPO memiliki performa yang lebih baik sebesar 17,44% dari CFA dan 7,45% lebih baik dari FHO.



GAMBAR 10. Hasil Uji Coba Kasus #1

B. UJI KASUS #2

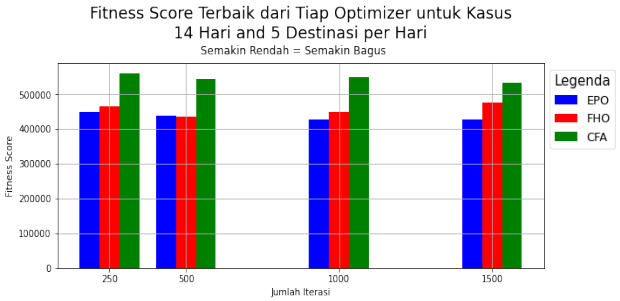
Uji kasus kedua akan membahas kasus di mana jumlah hari sebanyak 14 dan terdapat maksimal 5 kunjungan dalam satu hari. Nilai fitness terbaik yang didapatkan pada uji kasus kedua adalah -426475.63, nilai fitness tersebut diperoleh menggunakan algoritma EPO. Tabel IV merupakan tabel yang menunjukkan nilai fitness terbaik yang diperoleh untuk setiap algoritma pada uji kasus kedua.

TABEL IV.
FITNESS TERBAIK UJI KASUS #2

Algoritma	No	Fitness	Iterasi	Pop ulasi	Parameter
EPO	1	-426475.63	1000	100	*f*: 2.1; *M*: 2; *l*: 2.1
	2	-427561.85	1500	100	*f*: 2.4; *M*: 1; *l*: 1.7
	3	-427859.84	1500	100	*f*: 3; *M*: 1; *l*: 1.7
FHO	1	-435565.46	500	50	-
	2	-449880.12	1000	50	
	3	-458713.13	500	50	
CFA	1	-531663.69	1000	100	*r1*: -2; *r2*: -1; *v1*: -1; *v2*: 1
	2	-539349.36	1000	100	*r1*: -2; *r2*: -1; *v1*: 2; *v2*: -2
	3	-539349.36	500	100	*r1*: -2; *r2*: -2; *v1*: 1; *v2*: 0

Data pada Tabel IV dapat disimpulkan menjadi suatu grafik yang dapat dilihat pada Gambar 11. Pada uji coba kasus #2, ditemukan bahwa optimisasi menggunakan EPO unggul pada 250, 1000, dan 1500 iterasi dibandingkan dengan algoritma optimalisasi lainnya. Optimisasi menggunakan FHO unggul pada percobaan dengan 500 iterasi dengan selisih yang kecil. Pada uji kasus #2, EPO menghasilkan performa 24,66% lebih

baik dari CFA dan 2,13% lebih baik daripada FHO. Namun hasil uji coba kedua memperlihatkan bahwa FHO memiliki nilai fitness yang lebih baik jika dibandingkan dengan uji kasus yang pertama.



GAMBAR 11. Hasil Uji Coba Kasus #2

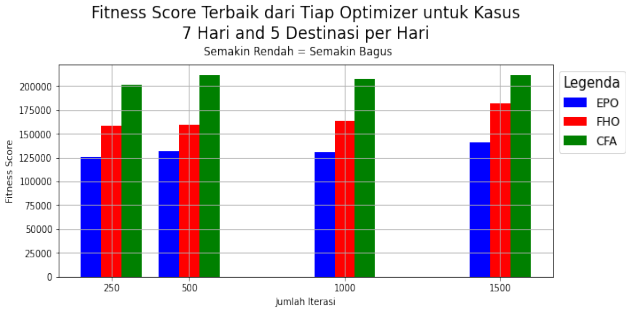
C. UJI KASUS #3

Pada uji kasus ketiga akan membahas di mana jumlah hari sebanyak 7 dan terdapat maksimal 5 kunjungan yang dilakukan dalam satu hari. Nilai fitness terbaik yang didapatkan adalah -125116.57, nilai fitness yang diperoleh tersebut dihasilkan dengan algoritma optimalisasi dengan menggunakan algoritma EPO. Tabel V merupakan tabel yang menunjukkan nilai fitness terbaik untuk setiap algoritma pada uji kasus ketiga.

TABEL V.
FITNESS TERBAIK UJI KASUS #3

Algoritma	No	Fitness	Iterasi	Pop ulasi	Parameter
EPO	1	-125116.57	250	100	*f*: 2.8; *M*: 1; *l*: 1.5
	2	-130636.13	1000	100	*f*: 2.1; *M*: 2; *l*: 1.8
	3	-131709.49	500	100	*f*: 2.9; *M*: 1; *l*: 1.8
FHO	1	-158777.89	250	50	-
	2	-159058.52	500	50	
	3	-163292.29	1000	50	
CFA	1	-200979.13	250	100	*r1*: -2; *r2*: -2; *v1*: 2; *v2*: -1
	2	-207344.54	1000	100	*r1*: -2; *r2*: -2; *v1*: -1; *v2*: 2
	3	-208092.88	1000	100	*r1*: -2; *r2*: -2; *v1*: -1; *v2*: 1

Data pada Tabel V dapat disimpulkan menjadi suatu grafik yang dapat dilihat pada Gambar 12. Pada grafik tersebut, optimisasi menggunakan EPO unggul pada seluruh percobaan iterasi (60% lebih baik dari CFA dan 26,9% lebih baik dari FHO). Grafik pada Gambar 12 juga menunjukkan bahwa hasil nilai fitness dari algoritma CFA memiliki hasil yang buruk



GAMBAR 12. Hasil Uji Coba Kasus #3

yaitu nilai fitness functionyang melebihi 500000 pada kasus uji coba ketiga.

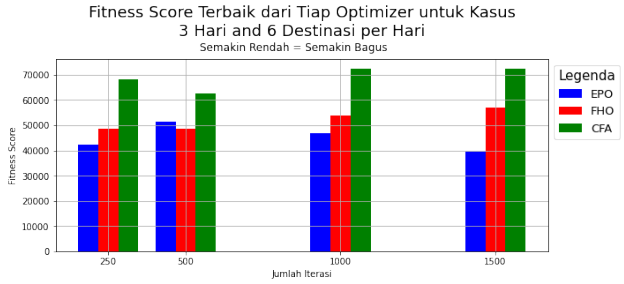
D. UJI KASUS #4

Pada uji kasus keempat akan membahas untuk hari berjumlah 3 dan terdapat maksimal 6 kunjungan dalam satu hari. Nilai fitness terbaik yang didapatkan adalah -39590.81, nilai fitness tersebut diperoleh dengan menggunakan algoritma EPO. Tabel VI merupakan tabel yang menunjukkan nilai fitness terbaik untuk setiap algoritma pada uji kasus keempat.

TABEL VI.
FITNESS TERBAIK UJI KASUS #4

Algoritma	No	Fitness	Iterasi	Populasi	Parameter
EPO	1	-39590.81	1500	100	*f*: 2.2; *M*: 2; *l*: 2.1
	2	-42403.18	250	100	*f*: 2; *M*: 1; *l*: 1.7
	3	-44399.97	250	100	*f*: 2.8; *M*: 1; *l*: 2.1
FHO	1	-48501.43	250	50	-
	2	-48643.86	500	50	
	3	-50825.39	500	50	
CFA	1	-62467.32	500	100	*r1*: -2; *r2*: -1; *v1*: 2; *v2*: 2
	2	-68228.72	250	100	*r1*: -2; *r2*: -2; *v1*: 1; *v2*: -2
	3	-68955.56	500	100	*r1*: -2; *r2*: -2; *v1*: 1; *v2*: -1

Data pada Tabel VI dapat disimpulkan menjadi suatu grafik yang dapat dilihat pada Gambar 13. Percobaan dengan menggunakan EPO memiliki hasil yang jauh lebih baik dibandingkan dengan algoritma lainnya pada 250, 1000, dan 1500 iterasi. Algoritma FHO memiliki hasil yang lebih baik dari optimisasi lainnya pada 500 iterasi. Pada uji kasus keempat, algoritma optimasi EPO memiliki performa 57,78% lebih baik dari CFA dan 22,51% lebih baik dibandingkan dengan algoritma FHO.



GAMBAR 13. Hasil Uji Coba Kasus #4

E. HASIL UJI KASUS

Pada setiap uji kasus yang dilakukan, terdapat pencatatan berapa lama program berjalan dari awal hingga akhir iterasi. Data tersebut dirata-rata dan dimasukkan ke dalam sebuah grafik pada Gambar 15. Kemudian untuk membuktikan hasil rute yang didapatkan berdasarkan fitness score terbaik, maka telah dibuat visualisasi rute pada peta. Seperti yang tergambar pada Gambar 14, dapat terlihat rute pada hari yang sama sudah terkumpul di area yang berdekatan.

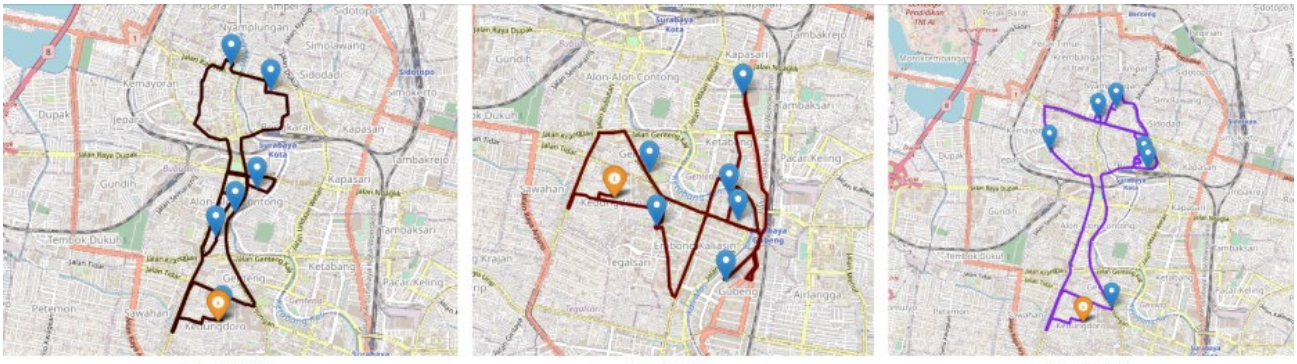
Di dalam grafik pada Gambar 15, memiliki pola yang sama pada setiap uji coba dan iterasi. Uji Coba dengan FHO memiliki waktu yang paling lama dari algoritma yang lainnya, terutama pada percobaan menggunakan iterasi dan populasi yang besar. Setelah FHO, CFA adalah algoritma dengan waktu jalan yang lebih lama daripada EPO, dan EPO adalah algoritma dengan waktu yang paling cepat daripada kedua algoritma lainnya. Selain pencatatan waktu, dilakukan pula pencatatan distribusi nilai fitness terhadap iterasi untuk setiap uji coba yang dilakukan. Grafik tersebut dapat dilihat pada Gambar 16 hingga Gambar 19.

Grafik distribusi menunjukkan bahwa setiap algoritma memiliki nilai fitness yang jarak maksimal dan minimal yang bervariasi. Persebaran fitness yang diperoleh menggunakan CFA cenderung memiliki range yang kecil dibandingkan dengan hasil fitness dengan menggunakan algoritma lain. Algoritma EPO dapat menghasilkan fitness terbaik dengan memberikan solusi yang memiliki fitness yang beragam. Sedangkan algoritma FHO memiliki distribusi fitness yang tidak tersebar merata melainkan terpusat pada fitness tertentu terutama pada iterasi 1000 dan 1500 dikarenakan percobaan hanya dilakukan untuk mencari fitness terbaik saja sehingga hasil yang kurang baik tidak tersimpan.

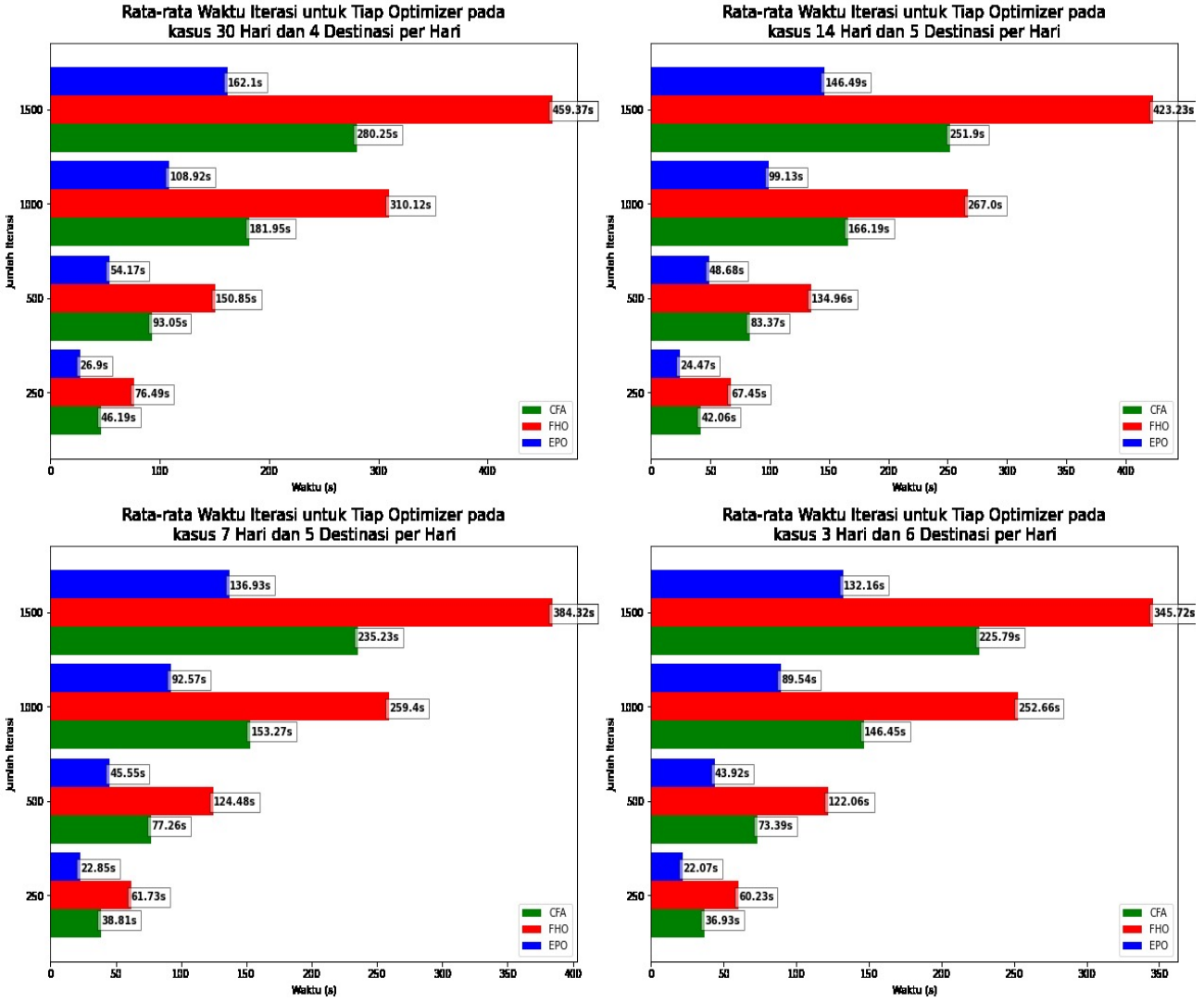
IX. KESIMPULAN

Berdasarkan hasil penelitian dan uji coba yang dilakukan, dapat terlihat EPO merupakan algoritma yang menghasil nilai fitness terbaik. Baik ditelaah dari sisi jumlah iterasi serta jumlah populasi ataupun melihat dari kasus yang dilakukan. Dimana pada setiap kasus tetap EPO menghasilkan fitness terbaik dengan nilai masing-masing, pada uji kasus pertama sebesar -970963.64, pada uji kasus kedua sebesar -426475.63, pada uji kasus ketiga sebesar -125116.57, dan uji kasus keempat sebesar -39590.81. Sedangkan dari sisi waktu komputasi, EPO tetap memegang waktu terbaik, sedangkan CFA memiliki waktu komputasi rata-rata terbaik kedua (69.59% lebih lambat dari EPO) serta FHO yang terlama dengan perbandingan 178.34% lebih lambat dari EPO. Maka dari semua hasil yang kami peroleh, baik dari sisi efisiensi komputasi dan hasil fitness score, maka algoritma EPO menjadi yang terbaik pada studi kasus ini.

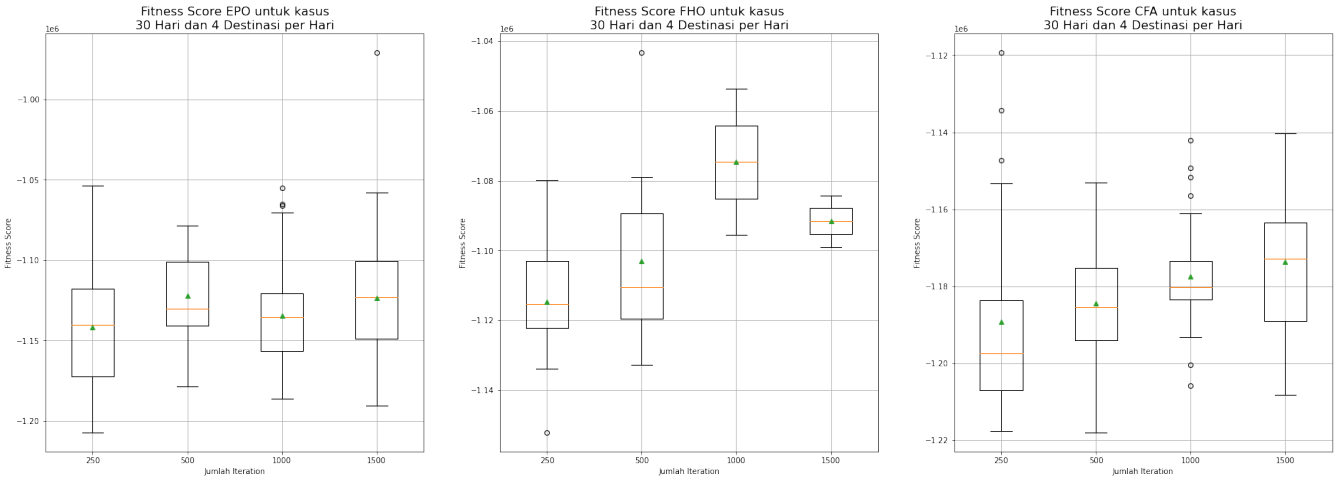
Terdapat beberapa hal yang masih dapat ditingkatkan untuk penelitian berikutnya. Jenis uji coba yang kami lakukan masih belum memiliki variasi yang memadai. Dimana bila dicoba dengan jumlah iterasi, jumlah penguin, serta parameter algoritma yang bervariasi memiliki kemungkinan untuk mendapatkan hasil fitness score yang lebih baik. Titik hotel yang digunakan pada penelitian ini hanya dari satu lokasi saja, akan lebih baik untuk menggunakan lebih banyak titik hotel untuk mendapatkan hasil yang lebih valid. Selain itu, data yang digunakan dapat ditingkatkan termasuk melakukan sortir spot pariwisata yang dimasa depan dapat berubah pula.



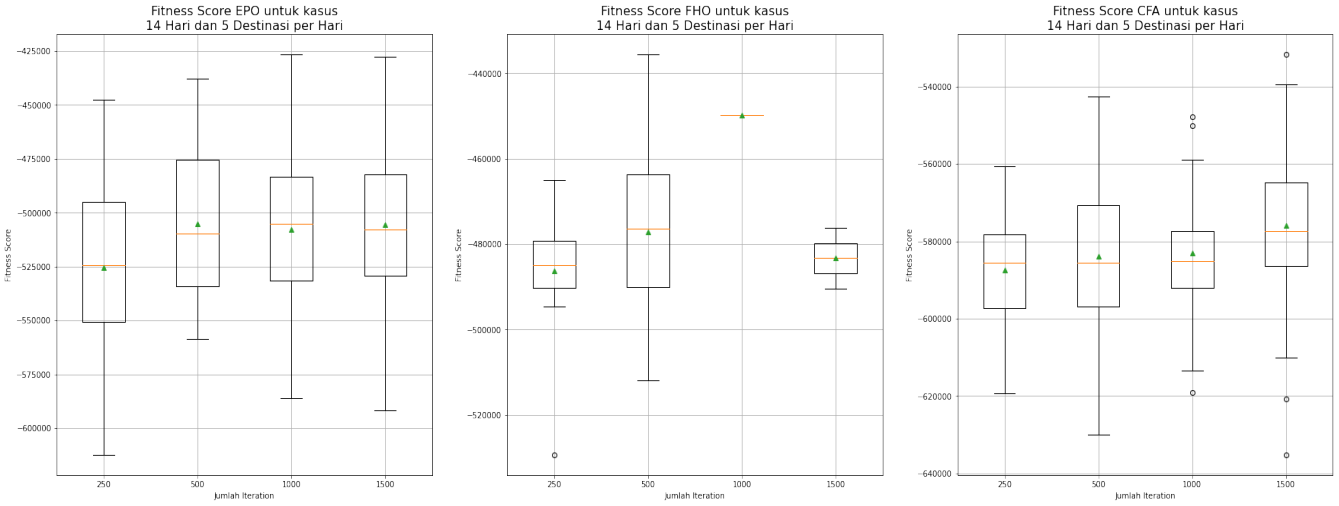
GAMBAR 14. Visualisasi Rute Hasil Uji Kasus #4



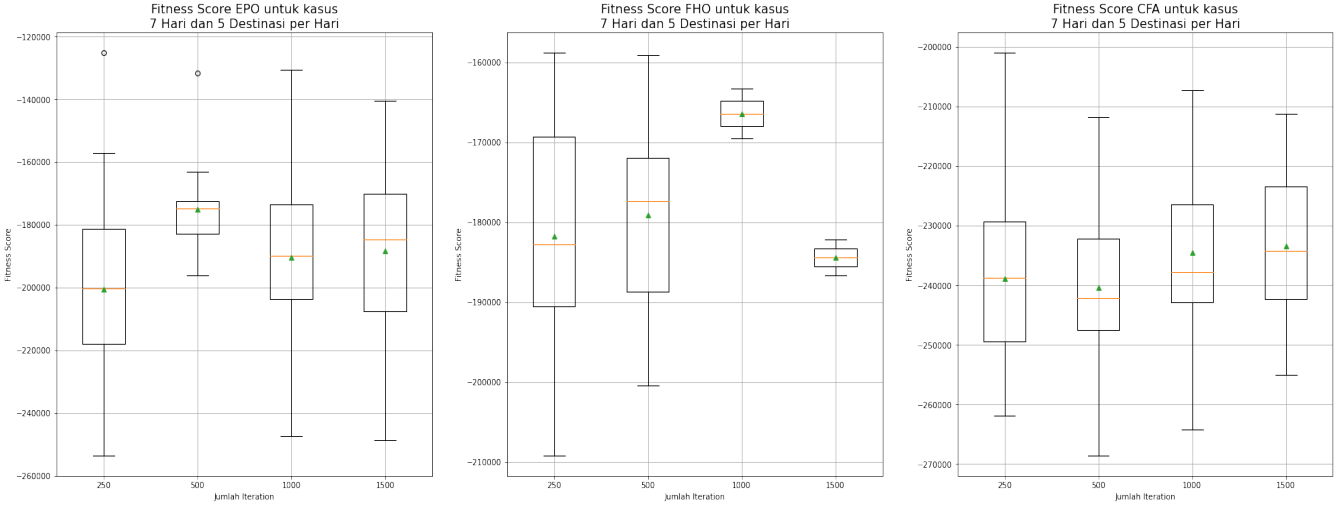
GAMBAR 15. Rata–rata Waktu untuk Setiap Algoritma



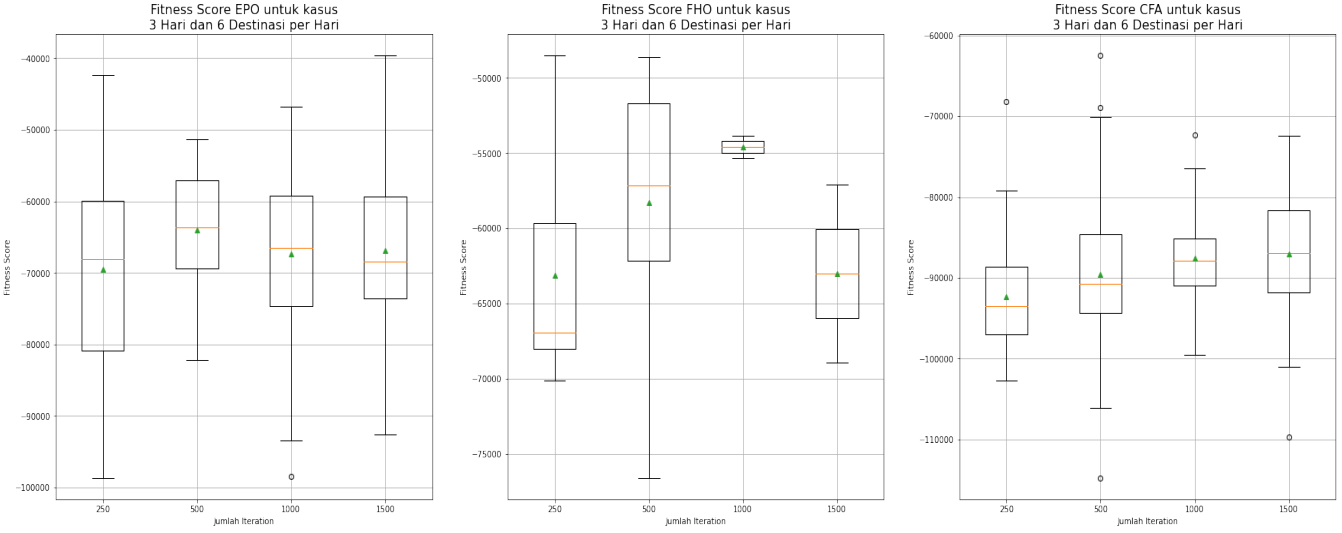
GAMBAR 16. Perbandingan Distribusi Fitness Antar Algoritma Pada Uji Kasus #1



GAMBAR 17. Perbandingan Distribusi Fitness Antar Algoritma Pada Uji Kasus #2



GAMBAR 18. Perbandingan Distribusi Fitness Antar Algoritma Pada Uji Kasus #3



GAMBAR 19. Perbandingan Distribusi Fitness Antar Algoritma Pada Uji Kasus #4

PERAN PENULIS

Christian Trisno Sen Long Chen: Konseptual, Persiapan Data, Implementasi Sistem, Visualisasi dan Uji Coba;
David Cahyadi: Pencarian dan Kurasi Data, Implementasi Sistem, Visualisasi;
Jonathan Arelio Bevan: Kurasi Data, Penyusunan Draf Asli, Penulisan Review dan Penyuntingan;
Williandy Takhta: Konseptual, Kurasi Data, Implementasi Sistem, Visualisasi dan Uji Coba;
Ariel Pratama Lesmana: Penyusunan Draf Asli, Penulisan Review dan Penyuntingan;
Christopher Davin Agus Poernomo: Implementasi Sistem, Visualisasi dan Uji Coba, Penyusunan Draf Asli;
Widean Nagari: Implementasi Sistem, Uji Coba, Penyusunan Draf Asli;

COPYRIGHT

DAFTAR PUSTAKA

[1] A. E. Eiben and G. Rudolph, "Theory of evolutionary algorithms: a bird's eye view," 1999. [Online]. Available: www.elsevier.com/locate/tcs
[2] A. N. Sloss and S. Gustafson, "2019 Evolutionary Algorithms Review," Jun. 2019, [Online]. Available: http://arxiv.org/abs/1906.08870
[3] T. Bartz-Beielstein, J. Branke, J. Mehnen, and O. Mersmann, "Evolutionary Algorithms," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 4, no. 3. Wiley-Blackwell, pp. 178–195, 2014. doi: 10.1002/widm.1124.
[4] K. L. Hoffman and M. Padberg, "Traveling salesman problem," in *Encyclopedia of Operations Research and Management Science*, New York, NY: Springer US, 2001, pp. 849–853. doi: 10.1007/1-4020-0611-X_1068.
[5] Z. Hashim and W. R. Ismail, "Applications of Travelling Salesman Problem in Optimizing Tourist Destinations Visit in Langkawi," in *Regional Conference on Science, Technology and Social Sciences (RCSTSS 2014)*, Springer Singapore, 2016, pp. 265–273. doi: 10.1007/978-981-10-0534-3_25.
[6] O. Nurdiawan, F. A. Pratama, D. A. Kurnia, Kaslani, and N. Rahaningsih, "Optimization of Traveling Salesman Problem on Scheduling Tour Packages using Genetic Algorithms," in *Journal of Physics: Conference Series*, 2020, vol. 1477, no. 5. doi: 10.1088/1742-6596/1477/5/052037.
[7] F. Hanif Khan, N. Khan, and S. Inayatullah, "SOLVING TSP PROBLEM BY USING GENETIC ALGORITHM," *Article in International Journal of Basic & Applied Sciences*, vol. 28, p. 170, 2010, [Online]. Available: https://www.researchgate.net/publication/236009452
[8] J. Juwairiah, D. Pratama, H. C. Rustamaji, H. Sofyan, and D. B. Prasetyo, "Genetic Algorithm for Optimizing Traveling Salesman Problems with Time Windows (TSP-TW)," *International Journal of Artificial Intelligence & Robotics (IJAIR)*, vol. 1, no. 1, pp. 1–8, Nov. 2019, doi: 10.25139/ijair.v1i1.2024.
[9] M. Tryana Sembiring and S. Chailes, "Ant Colony Optimization Implementation on Traveling Salesman Problem to Achieve the Shortest Logistic Route," in *IOP Conference Series: Materials Science and Engineering*, Dec. 2020, vol. 1003, no. 1. doi: 10.1088/1757-899X/1003/1/012045.
[10] A. Sabry Eesa, A. Mohsin Abdulazeez, Z. Orman, A. Mohsin, and A. Brifcani, "A Novel Bio-Inspired Optimization Algorithm Analysis of Health Data with Heuristic Learning Methods View project A New Dimensional Reduction Approach Based on Cuttlefish Algorithm and K-Nearest Neighbor for Gene Expression Data View project Cuttlefish Algorithm-A Novel Bio-Inspired Optimization Algorithm," *Article in International Journal of Scientific and Engineering Research*, vol. 4, no. 9, 2013, [Online]. Available: http://www.ijser.org
[11] G. Dhiman and V. Kumar, "Emperor penguin optimizer: A bio-inspired algorithm for engineering problems," *Knowl Based Syst*, vol. 159, pp. 20–50, Nov. 2018, doi: 10.1016/j.knosys.2018.06.001.
[12] M. Azizi, S. Talatahari, and A. H. Gandomi, "Fire Hawk Optimizer: a novel metaheuristic algorithm," *Artif Intell Rev*, 2022, doi: 10.1007/s10462-022-10173-w.
[13] Dr. James B. Wood, "http://www.thecephalopodpage.org/."
[14] H. Armanto, R. Kevin, and C. Pickerling, "Perencanaan Perjalanan Wisata Multi Kota dan Negara dengan Algoritma Cuttlefish," *INSYST*, vol. 1, no. 2, pp. 99–109, Dec. 2019.
[15] A. S. Eesa, Z. Orman, and A. M. A. Brifcani, "A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems," *Expert Syst Appl*, vol. 42, no. 5, pp. 2670–2679, Apr. 2015, doi: 10.1016/j.eswa.2014.11.009.
[16] R. Li, J. Yang, X. Tuo, and R. Shi, "Research on Fitness Function of Two Evolution Algorithms Using for Neutron Spectrum Unfolding," 2021.

Studi Klasifikasi Gerakan Semaphore menggunakan Fuzzy Mamdani dari Data IMU Sensor

Tobias N. Budimartono[1] dan Romy B. Widodo[1]
[1]Human-Machine Interaction Research Center, Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Ma Chung, Malang, Indonesia

Corresponding author: Romy B. Widodo (e-mail: romy.budhi@machung.ac.id).

ABSTRACT Often found in communication between ships and scouted the use of semaphore flags for long-distance communication media. Semaphore learning is generally done manually and using detection with digital images. However, both have limitations, namely the availability of experts/tutors and lighting problems. Given the importance of semaphores for long-distance communication, semaphore experts need a guidance system that can assess exercise movements. This study used fuzzy logic with the Mamdani inference system. The method used is to install inertial sensors on both wrists to obtain data on the angle of movement of the hands. The inertial sensor used produces three orientation angles, namely roll, pitch, and yaw; But this study used roll angles. The semaphore movement in this study pivoted on only one axis so that the roll angle data was used for analysis. Inertial sensors wirelessly deliver data to computers that already contain fuzzy machines. Semaphore movement has eight angular points, but in this study, a new method has been found using five areas to group the angles of both hands. Grouping five areas are beneficial for the creation of fuzzy rules. Fuzzy output is in the form of aligned or unaligned with the reference value. The experiment used five subjects who were asked to perform eight characters representing the letters A-Z, and spaces. The test results showed that the system successfully detected movement with an accuracy of 67.5%. The results of the analysis feedback showed that in the future a sensor was needed that could detect the tilt of the torso to compensate for the less upright posture when the subject experimented.

KEYWORDS Fuzzy Mamdani, Inertial Measurement Unit, Semaphore

ABSTRAK Sering dijumpai di komunikasi antar kapal dan dikepramukaan penggunaan bendera semaphore untuk media komunikasi jarak jauh. Pembelajaran semaphore umumnya dilakukan secara manual dan menggunakan pendeteksian dengan citra digital. Namun keduanya memiliki keterbatasan yaitu ketersediaan tenaga ahli/tutor dan permasalahan pencahayaan. Mengingat pentingnya semaphore untuk komunikasi jarak jauh maka ahli semaphore memerlukan *guidance system* yang dapat menilai gerakan latihan. Penelitian ini menggunakan fuzzy logic dengan mamdani inference system. Metode yang digunakan adalah memasang sensor inersial pada kedua pergelangan tangan untuk mendapatkan data sudut gerakan tangan. Sensor inersial yang digunakan menghasilkan tiga sudut orientasi yaitu roll, pitch, dan yaw; namun dalam penelitian ini digunakan sudut roll. Gerakan semaphore pada penelitian ini berporos hanya pada satu sumbu sehingga data sudut roll yang digunakan untuk analisis. Sensor inersial secara nirkabel memberikan data ke komputer yang telah berisi perangkat lunak fuzzy system. Pada dasarnya gerakan semaphore memiliki delapan titik sudut, namun dalam penelitian ini telah ditemukan metode baru menggunakan lima area untuk mengelompokkan sudut kedua tangan. Pengelompokan pada lima area bermanfaat untuk penciptaan rule fuzzy. Output fuzzy berupa skor sesuai atau tidak sesuai. Eksperimen menggunakan lima orang subjek yang diminta melakukan delapan karakter yang mewakili huruf A-Z, dan spasi. Hasil pengujian menunjukkan sistem berhasil mendeteksi gerakan dengan akurasi 67.5%. Hasil umpan balik analisis menunjukkan bahwa kedepannya diperlukan sebuah sensor yang dapat mendeteksi kemiringan torso untuk mengkompensasi postur tubuh yang kurang tegak saat subjek melakukan eksperimen.

KATA KUNCI Fuzzy Mamdani, Semaphore, Sensor Inersial

I. PENDAHULUAN

Arti kata komunikasi menyatakan pengiriman dan penerimaan pesan atau berita antara dua orang atau lebih sehingga pesan yang dimaksud dapat dipahami; juga berarti hubungan atau kontak. Komunikasi secara garis besar dapat dibagi menjadi dua berdasarkan cara menyampaikan pesan yaitu komunikasi secara verbal dan non-verbal. Komunikasi verbal bercirikan penggunaan kata-kata sebagai perantaranya baik secara lisan maupun tulisan. Sedangkan komunikasi non-verbal merupakan cara komunikasi yang tidak menggunakan kata-kata sebagai perantaranya melainkan menggunakan isyarat tubuh [1]–[5]. Adapun kendala dalam berkomunikasi secara non-verbal adalah memungkinkan perbedaan persepsi dalam mengartikan informasi berbentuk isyarat antara pengirim dan penerima. Akan tetapi ada komunikasi non-verbal yang telah ditentukan artinya secara luas sehingga tidak akan memungkinkan perbedaan arti antara pengirim dan penerima seperti isyarat tangan polisi dalam mengatur lalu lintas dan semaphore.

Semaphore merupakan salah satu komunikasi non-verbal yang menggunakan gerakan tangan atau bendera sebagai sarana penyampaian informasi [6]. Setiap kombinasi gerakan semaphore telah ditentukan sebelumnya sehingga siapapun dapat mengerti informasi yang disampaikan dengan baik bila gerakan tersebut dilakukan dengan benar. Adapun semaphore saat ini banyak digunakan dalam kegiatan kepramukaan sebagai alat komunikasi jarak jauh di alam terbuka. Penggunaan semaphore sangat dibutuhkan karena dapat memberikan informasi secara tepat dimana tidak memungkinkan untuk melakukan komunikasi menggunakan suara. Maka dari itu semaphore masih digunakan dalam kegiatan pramuka hingga saat ini. Dari hal tersebut diatas, perlunya gerakan semaphore dilakukan dengan benar. Pelatihan penggunaan semaphore kepada calon penggunanya adalah ujung tombak supaya menghasilkan pengguna yang benar-benar sempurna gerakan semaphore-nya.

Terdapat beberapa penelitian yang telah dilakukan untuk menerjemahkan gerakan semaphore. Klasifikasi gerakan semaphore dengan Kinect pada [7], [8]. Demikian juga pemanfaatan metode CNN pada data sensor IMU untuk klasifikasi semaphore [9]. Pemanfaatan pemrosesan gambar digital untuk meningkatkan pendeteksian semaphore dilakukan pada [10]. Pada penelitian ini permasalahan yang ingin dipecahkan adalah penerjemahan semaphore yang sebelumnya menggunakan gambar dari kamera, akan digantikan dengan sensor IMU yang bersifat wearable di tubuh penggunanya. Hal tersebut dikarenakan penerjemahan semaphore menggunakan kamera seperti pada Kinect memiliki beberapa kelemahan yaitu sensor tersebut dipengaruhi oleh jarak dan cahaya. Namun penggunaan IMU tidak dipengaruhi cahaya dan tidak membutuhkan konfigurasi ruangan khusus. Penggunaan IMU sesuai untuk deteksi gerakan semaphore sebab pengukuran gerakan semaphore tergantung dari besarnya sudut lengan terhadap poros torso, di sisi yang sama IMU dapat mengukur sudut orientasi.

Penelitian ini menawarkan solusi klasifikasi gerakan semaphore menggunakan data sensor IMU yang dipasang pada lengan. Metode yang digunakan adalah penelitian eksperimental dengan *Mamdani-type fuzzy-logic inference engine* untuk menentukan huruf berdasarkan karakteristik gerakan semaphore.

II. KAJIAN LITERATUR

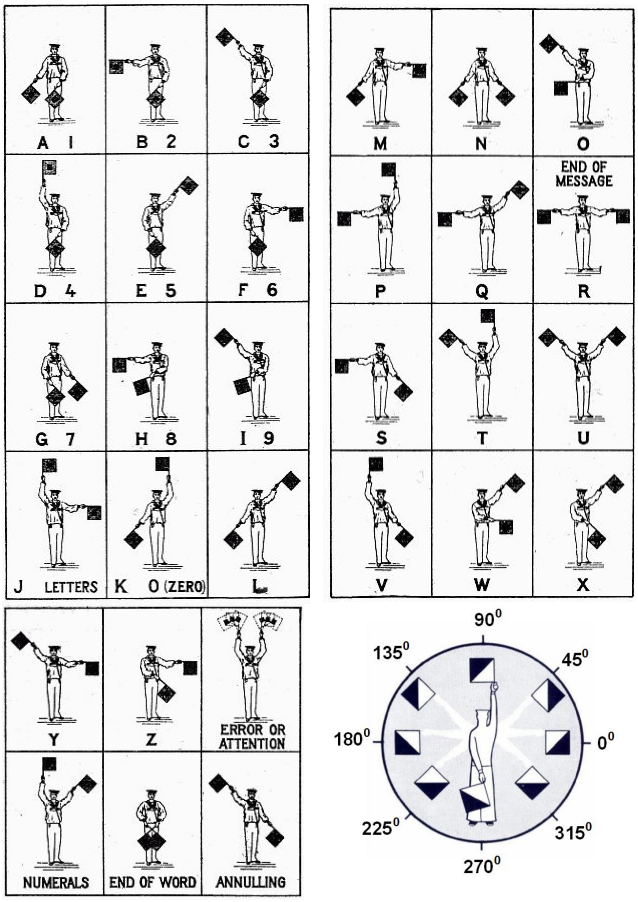
A. SISTEM FUZZY

Pemanfaatan *fuzzy* mamdani untuk penentuan atau klasifikasi jenis ikan dilakukan pada [11] berdasarkan karakteristik lahan. Penggunaan metode tersebut dapat diterapkan pada penelitian ini untuk menentukan huruf berdasarkan karakteristik gerakan semaphore. Selain itu pemanfaatan fuzzy mamdani untuk klasifikasi disertai penggabungannya dengan algoritma lain. Pemanfaatan fuzzy mamdani untuk klasifikasi tingkat tekanan darah manusia dapat dijumpai pada [12] dimana memiliki keberhasilan klasifikasi 99.4%. Penggunaan fuzzy mamdani untuk klasifikasi sentiment data pada NLP berhasil menunjukkan hasil yang akurat mengklasifikasikan ke dalam tiga kelas, yaitu netral, positif, dan negatif [13]. Klasifikasi sakit kepala dilakukan juga dengan fuzzy mamdani berhasil meningkatkan akurasi sebesar 88% [14]. Klasifikasi untuk penentuan kesalahan pada *double-circuit transmission lines* menggunakan fuzzy mamdani telah dilakukan pada [15]. Demikian juga aplikasi klasifikasi gambar binatang bergerak menggunakan gabungan fuzzy mamdani dan CNN model berhasil dikembangkan dengan akurasi 98% pada [16].

Metode Mamdani sering disebut sebagai metode Min-Max dikarenakan pada operasinya menggunakan operasi minimum dan maksimum [17]. Terdapat empat tahapan untuk mendapatkan *output* dari metode ini, empat tahapan tersebut adalah sebagai berikut: 1) Pembentukan himpunan fuzzy (*fuzzyfikasi*); 2) Aplikasi fungsi implikasi; 3) Komposisi aturan (aggregasi), yang terdiri atas tiga metode yaitu: metode max, metode penjumlahan (sum), dan metode probabilistik OR; 4) Penegasan (*defuzzyfikasi*), beberapa metodenya adalah metode *centroid* (*composite moment*), metode bisektor, metode *mean of maximum* (MOM), metode *largest of maximum* (LOM), dan metode *smallest of maximum* (SOM).

B. SEMAPHORE

Semaphore merupakan suatu metode pengiriman sinyal, pesan, atau informasi jarak jauh yang diciptakan oleh pendeta asal Perancis bernama Claude Chappe pada tahun 1790. Seiring perkembangan zaman penggunaan semaphore telah bergeser dari penggunaannya untuk komunikasi antar menara menjadi komunikasi antar kapal. Sehingga masih dipakai hingga saat ini. Bendera semaphore harus berbentuk persegi dengan ukuran 45 cm x 45 cm yang dipasangkan



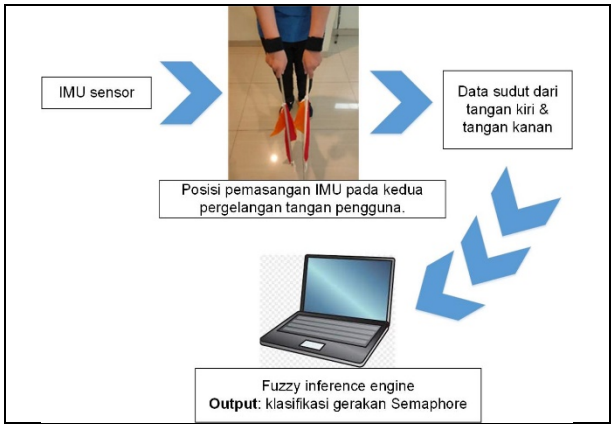
GAMBAR 1. **Gerakan semaphore dan delapan titik sudut semaphore [18]**

pada tongkat berukuran 55 cm. Pada umumnya bendera berwarna merah dan kuning yang terbagi secara diagonal, penempatan warna merah berada di atas warna kuning. Pemilihan kedua warna ini dikarenakan warna tersebut dapat dilihat dengan mudah. Gambar 1 menunjukkan huruf-huruf dalam semaphore yang diklasifikasikan, yaitu A-Z [18]. Adapun untuk mempermudah dalam mempelajari gerakan semaphore adalah dengan mengetahui prinsip arah jarum jam. Sehingga pada semaphore akan memiliki 8 titik sudut pasti, yaitu 0º, 45º, 90º, 135º, 180º, 225º, 270º, dan 315º.

III. METODOLOGI

Penelitian menggunakan metode penelitian eksperimental. Data sudut gerakan tangan diambil dari sensor inersial (Inertial Measurement Unit / IMU) yang dipasang pada pergelangan tangan kanan dan kiri. Sensor IMU yang digunakan adalah Xsens jenis MTwAwinda. Sensor IMU menghasilkan tiga sudut yaitu roll, pitch, dan yaw. Pada penelitian ini sudut yang diambil adalah roll, yang merupakan sudut paling dominan sesuai arah putaran tangan. Gambar 2 menunjukkan diagram kerja penelitian ini. Data sudut roll dari sensor IMU yang dipasang pada kedua pergelangan tangan menjadi input bagi program fuzzy.

Adapun Gambar 3 menunjukkan lokasi pemasangan sensor. Data diambil menggunakan *software* Xsens MT Manager, dengan output berupa sudut. Sampling rate adalah 100 Hz.

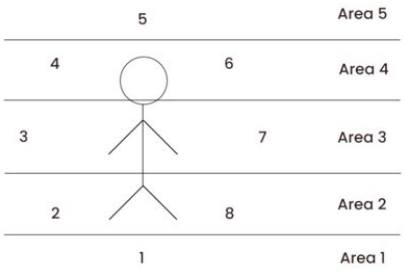


GAMBAR 2. **Diagram kerja penelitian.**

Klasifikasi pada gerakan semaphore menggunakan 5 area sebagai patokan posisi tangan seperti diilustrasikan di Gambar 4. Pembagian lima area ini merupakan kebaruan metode yang

GAMBAR 3. **Posisi sensor pada pergelangan tangan.**

ditemukan oleh peneliti dalam memecahkan masalah. Area pertama merupakan area paling bawah, area kedua merupakan hasil penggabungan titik sudut ke-2 dan 8, area ketiga merupakan hasil penggabungan titik sudut ke-3 dan 7, area keempat merupakan hasil penggabungan titik sudut ke-4 dan 6, dan yang terakhir area kelima merupakan area paling atas. Semua area tersebut berlaku pada setiap tangan baik tangan kiri maupun tangan kanan. Pembagian 8 titik semaphore menjadi 5 area didasarkan oleh pengamatan terhadap sudut *roll* yang berada pada rentang -180° sampai 180° atau putaran penuh. Oleh karena itu total keseluruhan area adalah 180° yang dibagi menjadi 5 yaitu masing-masing area memiliki rentang 36°. Semua huruf A-Z dan spasi; memiliki area



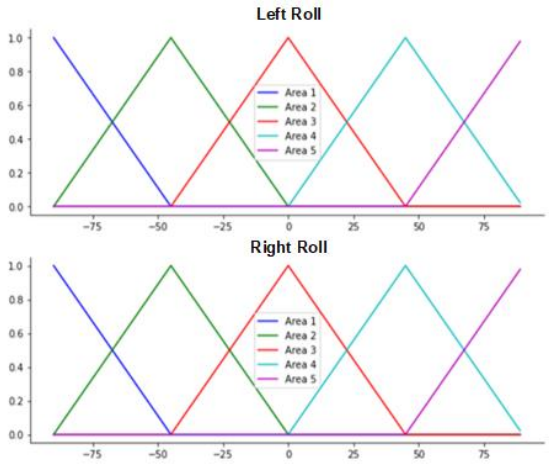
GAMBAR 4. Pembagian area gerakan semaphore untuk penentuan perancangan sistem fuzzy.

berdasarkan posisi tangan pada kelima area tersebut. Pembentukan rule juga berdasarkan kelima area tersebut.

A. HIMPUNAN INPUT-OUTPUT

Himpunan *input* diperoleh dari dua masukan yaitu sensor di pergelangan tangan kanan dan pergelangan tangan kiri. Setiap variabel *fuzzy* memiliki jumlah dan nilai himpunan yang sama yaitu kelima area semaphore yang telah ditentukan pada Gambar 4. Akan tetapi nilai-nilai dari kelima area sebelumnya masih bersifat tegas (*crisp*) sehingga perlu untuk mengkonversikan nilai-nilai crisp menjadi nilai-nilai *fuzzy*. Proses konversi tersebut dilakukan dengan cara mengambil nilai sudut yang menjadi pusat pada setiap area. Sehingga nilai-nilai pusat pada setiap area adalah masing-masing -90^0, -45^0, 0^0, -45^0, dan 90^0. Hasil dari kedua himpunan *input* dapat dilihat pada Gambar 5.

Setelah menentukan himpunan pada variabel *input* maka selanjutnya adalah menentukan himpunan *output*. Pada himpunan *output* yaitu hasil klasifikasi dalam bentuk huruf A-

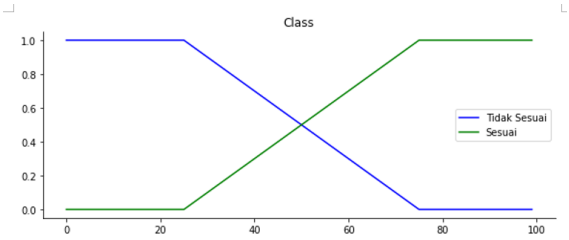


GAMBAR 5. Himpunan input.

Z dan spasi tidak memungkinkan dalam satu variabel dikarenakan nilainya berjenis nominal yang kurang sesuai digunakan pada himpunan *output*. Sehingga perlu dilakukan pengubahan nilai-nilai *output* menjadi nilai yang berjenis ordinal. Pengubahan nilai tersebut dilakukan dengan memisah setiap karakter berdasarkan kesesuaiannya pada gerakan semaphore. Sehingga akan terdapat 27 karakter yang masing-masing memiliki nilai kesesuaian terhadap data *input*. Variabel himpunan *output* memiliki dua label yaitu 'sesuai' dan 'tidak sesuai' dan nilai semesta pembicaraan pada rentang 0 sampai dengan 100, seperti diilustrasikan pada Gambar 6.

B. RULE

Rule merupakan aturan-aturan yang digunakan untuk



GAMBAR 6. Himpunan output.

menentukan himpunan *output* berdasarkan nilai keanggotaan. Pada penelitian ini *rule* akan dituliskan sebagai "*IF* Tangan Kanan *is* Area X *AND* Tangan Kiri *is* Area Y *THEN* [Sesuai / Tidak Sesuai]." Sehingga setiap karakter memiliki *rule* yang berbeda satu dengan yang lainnya. Setiap karakter memiliki 25 *rule* yang berarti terdapat 675 *rule* secara keseluruhan. Rule yang dibuat divalidasi melalui diskusi dengan tim peneliti, *tuning* rule juga dilakukan pada saat eksperimen.

Hasil penentuan *rule* merujuk pada area posisi sensor semaphore seperti pada Gambar 4. Posisi gerakan semaphore yang mendasari pembuatan rule seperti pada Tabel 1.

TABEL I
POSISI AREA GERAKAN SEMAPHORE

Karakter	Tangan kanan	Tangan kiri	Karakter	Tangan kanan	Tangan kiri
Spasi	1	1	N	2	2
A	2	1	O	4	3
B	3	1	P	3	5
C	4	1	Q	3	4
D	5	1	R	3	3
E	1	4	S	3	2
F	1	3	T	4	5
G	1	2	U	4	4
H	3	2	V	5	2
I	4	2	W	3	4
J	5	3	X	2	4
K	2	5	Y	4	3
L	2	4	Z	2	3
M	2	3			

C. DEFUZZIFIKASI

Defuzzyfikasi merupakan tahap terakhir pada pembentukan metode *fuzzy*. Pada tahap ini telah dihasilkan sebuah nilai tegas dalam range 0 sampai 100. Penelitian ini menggunakan metode *centroid* untuk mendapatkan titik pusat yang digunakan sebagai hasil *output*. Nilai yang dihasilkan tersebut merepresentasikan nilai kesesuaian gerakan semaphore

GAMBAR 7. Lingkungan pengujian.

terhadap karakter yang bersangkutan. Sehingga terdapat 27 nilai kesesuaian secara keseluruhan. Karakter yang memiliki nilai kesesuaian terbesar merupakan hasil akhir dari klasifikasi gerakan semaphore.

D. PENGUJIAN

Gerakan semaphore yang diuji kepada lima subjek adalah kata MA CHUNG, dengan karakternya yaitu: M, A, spasi, C, H, U, N, dan G. Dasar pengambilan kata tersebut adalah dikarenakan setiap karakternya tidak terdapat duplikasi karakter. Selain itu 8 karakter tersebut mencakup huruf dan karakter spasi yang gerakannya bisa digunakan untuk mewakili karakter lainnya; pemilihan 8 karakter ini juga mereduksi kelelahan Subjek. Sebelum analisis hasil pengujian, dilakukan pemrosesan awal data. Dimana 200 data di awal dan 200 data di akhir setiap percobaan diabaikan, dengan asumsi Subjek masih belum stabil kondisi tangannya pada dua detik di awal dan akhir suatu sesi pengambilan data. Pengujian dilakukan dengan memasukkan data yang didapatkan ke sistem *fuzzy* yang telah dibentuk. Sehingga data bersih yang didapatkan pada setiap huruf adalah sebanyak 600 data. Total keseluruhan data yang digunakan setelah dilakukan pengurangan tersebut adalah sebanyak 24.000 data.

Subjek diberi alat bantu berupa kertas yang bergambarkan gerakan semaphore yang harus dicoba. Kertas tersebut diletakan di depan kaki subjek sehingga dapat dilihat dengan mudah. Ilustrasi lebih jelas mengenai lingkungan pengambilan data dapat dilihat pada Gambar 7. Sedangkan Gambar 8 menunjukkan salah satu bentuk postur tubuh Subjek yang kurang sempurna, pembahasan postur ini dapat dilihat pada bagian diskusi.

IV. HASIL DAN DISKUSI

A. HASIL

Pada tahap ini pengujian dilakukan dengan memasukan data yang didapatkan ke sistem *fuzzy* yang telah dibentuk. Akan tetapi tidak semua data digunakan melainkan terdapat

GAMBAR 8. Postur tubuh subjek yang kurang sempurna.

penghapusan data sebanyak 200 setiap sisi tangan pada awal dan belakang. Tujuan dari penghapusan data ini adalah untuk menghindari kerusakan data diawal dan diakhir akibat kurangnya persiapan subjek. Jadi, 2 detik awal dan akhir digunakan waktu persiapan subjek. Sehingga data bersih yang didapatkan pada setiap huruf adalah sebanyak 300 data. Total keseluruhan data yang akan digunakan setelah dilakukan pengurangan tersebut adalah sebanyak 12.000 data.

Pengujian metode *fuzzy* Mamdani dilakukan pada seluruh data yang dimiliki pada masing-masing subjek. Tabel II menampilkan hasil pengujian untuk lima subjek dan delapan karakter yang dipilih oleh peneliti. Pemilihan delapan karakter tersebut sudah mewakili area-area pada Gambar 4. Kedelapan karakter tersebut adalah M, A, spasi, C, H, U, N, G. Pada Tabel II disertakan nomor rule yang diakses oleh sistem fuzzy pada saat ada input sudut.

TABEL II
HASIL KLASIFIKASI SEMAPHORE TIAP SUBJEK

	Prediksi								Rule
	M	A	Spasi	C	H	U	N	G	
Subjek 1	M (300)	A (300)	Spasi (300)	B (300)	H (300)	R (300)	G (300)	G (300)	1, 2, 6, 7
Subjek 2	M (300)	A (300)	Spasi (300)	C (300)	H (300)	R (300)	N (300)	G (300)	6, 7, 11, 12
Subjek 3	F (300)	A (300)	Spasi (300)	B (300)	H (300)	R (176) Q (124)	N (300)	G (300)	1, 2, 6, 7
Subjek 4	M (300)	A (300)	G (300)	H (300)	H (300)	U (300)	N (300)	G (300)	1, 2, 6, 7
Subjek 5	M (300)	N (300)	G (165) N (135)	I (300)	H (300)	U (300)	N (300)	N (300)	1, 2, 6, 7

Secara keseluruhan hasil klasifikasi gerakan semaphore dari Tabel II, diringkas pada Tabel III.

B. DISKUSI

Secara keseluruhan sistem berhasil melakukan klasifikasi terhadap gerakan semaphore dengan *score* akurasi sebesar 67.5%. Pengujian dilakukan dengan memilih delapan karakter M, A, spasi, C, H, U, N, dan G. Akurasi tersebut didapatkan berdasarkan hasil prediksi benar pada setiap karakter

dibandingkan dengan total jumlah seluruh karakter. Setiap subjek juga dihitung nilai akurasinya dan didapatkan bahwa subjek 2 memiliki gerakan semaphore yang paling baik yaitu sebesar 87.5% sedangkan subjek 5 memiliki gerakan semaphore yang kurang baik, dengan akurasi sebesar 50%. Dari hasil pengamatan, foto, dan video, penyebab kesalahan klasifikasi pada gerakan semaphore adalah gerakan subjek yang kurang baik. Seperti yang terlihat pada Gambar 8, postur tubuh salah satu subjek tidak tegak lurus melainkan terlihat sedikit miring sehingga menyebabkan sudut yang terbentuk lebih rendah dari yang seharusnya. Pada saat pembentukan gerakan tersebut sudut yang terbaca adalah sekitar 18° sampai 19°. Pada penelitian [9], digunakan CNN untuk klasifikasi data IMU pada semaphore; diperoleh akurasi 76.65%. Penggunaan CNN memberikan nilai akurasi lebih tinggi namun ada *trade off* yaitu proses pembelajaran dan pembuatan model yang memerlukan usaha lebih dibandingkan dengan metode fuzzy.

TABEL III
HASIL KLASIFIKASI SEMAPHORE

Subjek	Prediksi Benar	Akurasi
Subjek 1	5/8	62.5%
Subjek 2	7/8	87.5%
Subjek 3	5/8	67.5%
Subjek 4	6/8	75%
Subjek 5	4/8	50%
Total		67.5%

V. KESIMPULAN

Berdasarkan penelitian dapat disimpulkan bahwa pembagian delapan titik semaphore menjadi lima area berhasil digunakan untuk melakukan klasifikasi menggunakan logika fuzzy. Penerapan metode fuzzy Mamdani dapat digunakan untuk klasifikasi gerakan semaphore. Hal tersebut dapat dibuktikan melalui nilai akurasi yang telah diperoleh yaitu sebesar 67.5%. Namun kedepannya beberapa catatan peneliti untuk pengembangan penelitian adalah sebagai berikut: penambahan jumlah sensor pada bagian tubuh misalnya punggung dapat mendeteksi kemiringan torso dari subjek. Data kemiringan torso dapat digunakan untuk kompensasi rule fuzzy. Saran lain adalah pengoptimalan metode fuzzy Mamdani dengan melakukan tuning terhadap komponen-komponen fuzzy. Diharapkan dengan pengaturan yang lebih mendalam dapat menambah keakuratan metode dalam melakukan klasifikasi.

PERAN PENULIS

Penulis Tobias Nagata Budimartono: Desain eksperimen, pelaksanaan eksperimen, pemrograman, analisis hasil.
Penulis Romy Budhi Widodo: Koordinasi penelitian, penyiapan sensor, desain eksperimen, diskusi metode penelitian, penulisan artikel.

COPYRIGHT

DAFTAR PUSTAKA

[1] E. Padmalatha, S. Sailekya, R. Ravinder Reddy, C. Anil Krishna, and K. Divyarsha, "Sign language recognition," *Int. J. Recent Technol. Eng.*, vol. 8, no. 3, pp. 2128–2137, 2019, doi: 10.35940/ijrte.C4565.098319.

[2] S. P. Goswami, A. R. Ggr, and K. Sharma, "Introduction of indian sign language in inclusive education," *Disabil. CBR Incl. Dev.*, vol. 30, no. 4, pp. 96–110, 2019, doi: 10.5463/dcid.v30i4.796.

[3] M. Porta-Lorenzo, M. Vázquez-Enríquez, A. Pérez-Pérez, J. L. Alba-Castro, and L. Docío-Fernández, "Facial Motion Analysis beyond Emotional Expressions," *Sensors*, vol. 22, no. 10, 2022, doi: 10.3390/s22103839.

[4] R. C. Chen, W. E. Manongga, and C. Dewi, "Recursive Feature Elimination for Improving Learning Points on Hand-Sign Recognition," *Futur. Internet*, vol. 14, no. 12, pp. 1–18, 2022, doi: 10.3390/fi14120352.

[5] R. H. Abiyev, M. Arslan, and J. B. Idoko, "Sign language translation using deep convolutional neural networks," *KSII Trans. Internet Inf. Syst.*, vol. 14, no. 2, pp. 631–653, 2020, doi: 10.3837/tiis.2020.02.009.

[6] T. Juliatmojo and E. Ariwibowo, "Pembelajaran Sandi Morse Dan Sandi Semaphore Dalam Bentuk Simulasi Berbasis Multimedia," *J. Sarj. Tek. Inform.*, vol. 1, no. 1, pp. 129–139, 2013.

[7] R. Aisuwarya, N. Alfitri, and H. Wahyudi, "Sistem Penerjemah Sandi Semaphore Menggunakan Sensor Kinect dengan Pengenalan Pola Delapan Titik," in *Seminar Nasional Sains dan Teknologi*, 2017, pp. 1–6.

[8] M. Fuad and E. Prasetya, "Pengenalan Gestur Semaphore Menggunakan Sensor Kinect," in *Seminar Nasional Aplikasi Teknologi Informasi*, 2014, pp. 266–270.

[9] T. N. Budimartono, R. B. Widodo, and P. L. T. Irawan, "Perancangan Aplikasi Realtime Berbasis Desktop dengan Sensor IMU pada Klasifikasi Gerakan Semaphore Menggunakan Metode CNN," *Pros. Semin. Nas. Univ. Ma Chung*, pp. 75–88, 2022.

[10] K. A. Spenkov, O. R. Nikitin, I. E. Zhigalov, I. R. Dubov, and A. D. Pozdnyakov, "Using Image Processing to Improve Semaphore Communication," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 862, no. 5, pp. 1–6, 2020, doi: 10.1088/1757-899X/862/5/052021.

[11] A. Wirawan and A. Azhari, "Implementasi Metode Fuzzy-Mamdani untuk Menentukan Jenis Ikan Konsumsi Air Tawar Berdasarkan Karakteristik Lahan Budidaya Perikanan," *Bimipa*, vol. 24, no. 1, pp. 29–38, 2014.

[12] J. C. Guzmán, I. Miramontes, P. Melin, and G. Prado-Arechiga, "Optimal genetic design of type-1 and interval type-2 fuzzy systems for blood pressure level classification," *Axioms*, vol. 8, no. 1, 2019, doi: 10.3390/axioms8010008.

[13] F. Es-Sabery, A. Hair, J. Qadir, B. Sainz-De-Abajo, B. Garcia-Zapirain, and I. Torre-DIez, "Sentence-Level Classification Using Parallel Fuzzy Deep Learning Classifier," *IEEE Access*, vol. 9, pp. 17943–17985, 2021, doi: 10.1109/ACCESS.2021.3053917.

[14] M. Khayamnia, M. Yazdchi, A. Heidari, and M. Foroughipour, "Diagnosis of common headaches using hybrid expert-based systems," *J. Med. Signals Sens.*, vol. 9, no. 3, pp. 174–180, 2019, doi: 10.4103/jmss.JMSS_47_18.

[15] A. N. Kumar, C. Sanjay, and M. Chakravarthy, "A single-end directional relaying scheme for double-circuit transmission line using fuzzy expert system," *Complex Intell. Syst.*, vol. 6, no. 2, pp. 335–346, 2020, doi: 10.1007/s40747-020-00131-w.

[16] H. R. Mohammed and Z. M. Hussain, "Hybrid mamdani fuzzy rules and convolutional neural networks for analysis and identification of animal images," *Computation*, vol. 9, no. 3, 2021, doi: 10.3390/computation9030035.
[17] S. Kusumadewi and H. Purnomo, *Aplikasi Logika Fuzzy untuk Pendukung Keputusan*. Yogyakarta: Graha Ilmu, 2004.
[18] C. Curnow and E. Emmy, *Visual Signaling [EBook #43515]*. Signal Corps United States Army, 2013. [Online]. Available: https://www.gutenberg.org/files/43515/43515-h/43515-h.htm

Implementasi Algoritma Evolusi FHO, MVPA, dan HHO pada TSP di Tempat Pariwisata Pulau Bali

Christian B. Sabdana[1], Bryan Christopher[1], Jason G. Sutanto[1], Lawrence P. Sianto[1], Lukky Hariyanto[1], dan Nickolas Hartono[1]
[1]Departemen Informatika, Fakultas Sains dan Teknologi, Institut Sains dan Teknologi Terpadu Surabaya, Surabaya, Indonesia

Corresponding author: Christian B. Sabdana (e-mail: christian.b20@mhs.istts.ac.id).

ABSTRACT Vacation activities are essential for both individuals and families. The creation of travel route is a rather difficult task and must be thought of as a whole. In computer science terms, finding the optimal route in a *graph* is known as Traveling Salesman Problem. In case of creating good and well-defined route, we need specialized algorithm that can perform better in evaluating the route and provide an overall good result. In this research, we examine 4 algorithms, namely HHO (Harris Hawk Optimization), FHO (Fire Hawk Optimization), MVPA (Most Valuable Player Algorithm) and Modified MVPA along with trying the best hyperparameters for TSP problems in 55 tourist location across Bali Island. After conducting several experiments, we found that within the same hyperparameter, HHO perform well enough and have consistently much better fitness but with much longer runtime. Meanwhile FHO with the worst fitness has a runtime that is much faster than HHO and MVPA. Our modified MVPA algorithm was able to provide better results, although not as good as HHO. Qualitatively, the HHO algorithm provided better travel results with relatively consistent distances each day. This helps tourists make the most of their time in enjoying tourist locations compared to travel time.

KEYWORDS Evolutionary Computation, Optimization Algorithm, Travelling Salesman Problem

ABSTRAK Kegiatan berlibur merupakan kegiatan yang diperlukan baik perseorangan maupun bersama keluarga. Pembuatan rute perjalanan yang optimal dari banyak wisata liburan terkadang menjadi permasalahan rumit dan perlu dipikirkan rute optimalnya secara keseluruhan. Dalam ilmu komputer, permasalahan mencari rute optimal pada sebuah jaringan ini dikenal dengan *Traveling Salesman Problem*. Untuk mendapatkan rute yang baik, diperlukan algoritma khusus yang mampu mengevaluasi rute perjalanan dan memberikan hasil perjalanan yang cukup optimal. Di dalam penelitian ini, 4 algoritma *Evolutionary Computation* yaitu HHO (Harris Hawk Optimization), FHO (Fire Hawk Optimization), MVPA (Most Valuable Player Algorithm) dan modifikasi dari algoritma MVPA dibandingkan untuk menyelesaikan permasalahan TSP pada 55 lokasi wisata di Pulau Bali. Setelah dilakukan beberapa percobaan, HHO merupakan algoritma dengan nilai *fitness* terbaik dan konsisten tetapi dengan waktu eksekusi yang lebih lama. Sementara algoritma FHO memiliki waktu eksekusi yang lebih cepat tetapi nilai *fitness* yang lebih buruk dibandingkan dengan HHO dan MVPA. Algoritma MVPA yang telah dimodifikasi dapat memberikan hasil yang lebih baik meskipun masih belum bisa sebaik HHO. Secara kualitatif, algoritma HHO memberikan hasil perjalanan yang lebih baik dengan jarak tempuh tidak terlalu bervariasi setiap harinya. Hal ini membantu pelaku wisata agar dapat memanfaatkan waktu lebih banyak dalam menikmati lokasi wisata dibandingkan waktu perjalanan yang terbuang.

KATA KUNCI Algoritma Optimisasi, Evolutionary Computation, Travelling Salesman Problem

I. PENDAHULUAN

Traveling Salesman Problem (TSP) merupakan permasalahan yang sering dihadapi secara umum dan sering digunakan sebagai bahan uji coba terhadap algoritma optimisasi. Permasalahan ini berada pada domain teori graf dan termasuk ke dalam salah satu dari beberapa problem *NP-hard*[1]. Karena hal tersebut, tidak ada algoritma dengan kompleksitas waktu polinomial yang dapat menemukan solusi eksak permasalahan ini dengan efektif, dan kebenaran solusinya tidak dapat dibuktikan dengan mudah. Di sisi lain, algoritma optimisasi dikembangkan dengan tujuan mendapatkan solusi aproksimasi dalam waktu yang jauh lebih singkat dibandingkan teknik *brute-force*[2].

Algoritma optimisasi merupakan algoritma yang berfokus pada pencarian solusi yang mengoptimalkan sebuah fungsi objektif[3]. Proses optimisasi akan dilakukan secara berulang-ulang (iteratif) dan di setiap iterasinya, akan dilakukan evaluasi terhadap solusi-solusi yang telah ditemukan. Iterasi ini dilakukan terus menerus hingga didapatkan solusi yang cukup optimal. Pada algoritma optimisasi, terdapat *hyperparameter* yang nantinya akan mengatur bagaimana algoritma tersebut meminimalkan fungsi objektif yang diberikan. *Hyperparameter* yang berbeda membuka kemungkinan untuk mendapatkan solusi yang berbeda dan memberikan nilai objektif yang lebih baik.

Algoritma optimisasi terbagi menjadi banyak bagian seperti *stochastic optimization*, *constraint satisfaction*, *meta-heuristic*, dll. *Evolutionary computing* termasuk kedalam kelompok algoritma *meta-heuristic* dengan mekanisme yang terinspirasi oleh teori evolusi Darwin dan perilaku makhluk hidup di alam[4]. Algoritma *evolutionary computing* cukup efektif dalam melakukan pencarian solusi dikarenakan biaya komputasinya yang berkembang secara linear dengan ukuran permasalahan, dan metode pencarian menggunakan populasi yang mempercepat pencarian dan membuka peluang untuk diparalelisasi[4]. Pada *evolutionary computing*, fungsi objektif disebut juga dengan *fitness function* dan pada setiap proses evaluasi, setiap kandidat solusi akan dibandingkan dengan mengoptimalkan nilai *fitness*.

Pada penelitian ini, digunakan beberapa algoritma *Evolutionary Computing* dalam menyelesaikan *Traveling Salesman Problem* yang telah divariasikan mendekati dengan permasalahan yang dihadapi ketika perjalanan wisata sebenarnya. Algoritma yang digunakan yaitu *Harris Hawk Optimization* (HHO) [2], *Fire Hawk Optimization* (FHO) [5], dan *Most Valuable Player Algorithm* (MVPA) [6], serta algoritma modifikasi dari MVPA. Pengujian dilakukan dengan melakukan perbandingan kuantitatif deskriptif dari skalabilitas algoritma ditinjau dari perkembangan jumlah populasi, rata-rata perkembangan nilai *fitness* di setiap *epoch*, serta perbandingan kecepatan eksekusi algoritma di setiap *epoch*.

Representasi solusi yang digunakan merupakan sebuah barisan dari indeks lokasi yang dikunjungi. Representasi ini digunakan dikarenakan permasalahan *Travelling Salesman Problem* merupakan *sequencing-type problem*[7]. Selain itu, dikarenakan operasi fungsi pada algoritma yang diuji berada pada domain dan *range* bilangan real, digunakan *random-keys representation*[8] untuk mendapatkan permutasi lokasi yang dikunjungi.

II. TINJAUAN PUSTAKA

A. TRAVELING SALESMAN PROBLEM

Permasalahan ini dapat dideskripsikan secara sederhana sebagai berikut[9]: Diberikan n *vertex* dan $n(n-1)/2$ nilai yang menyatakan jarak antar pasangan *vertex*, tentukan rute yang mengunjungi semua *vertex* tersebut dengan jarak tempuh minimal. Permasalahan ini memiliki hubungan erat dengan *hamiltonian cycle*[10], yaitu sebuah *graph* memiliki rute TSP jika dan hanya jika *graph* tersebut memiliki minimal 1 *hamiltonian cycle*. *Hamiltonian cycle* didefinisikan sebagai sebuah *cycle* yang mengunjungi setiap *vertex*, sehingga dapat disimpulkan bahwa TSP merupakan permasalahan mencari *hamiltonian cycle* dengan total *weight* minimum.

TSP memiliki banyak variasi permasalahan, seperti: *Asymmetric* dan *Symmetric* TSP[11] yang masing-masing merupakan rute minimum pada graf berarah dan graf tak-berarah, *Euclidean* TSP[9] ketika *vertex* berupa titik pada bidang planar dan nilai *edge* antar dua *vertex* berupa jarak *euclidean* antara kedua *vertex* tersebut, dan beberapa variasi lainnya.

Pada penelitian ini, variasi permasalahan yang diteliti adalah *Asymmetric* TSP dengan nilai *edge* antar *vertex*/lokasi merupakan jarak lintasan/jalur antar lokasi. Terdapat tambahan kompleksitas problem yaitu total hari berkunjung dan maksimal jarak yang dapat ditempuh per harinya. Teknis pengambilan jarak antar lokasi dijelaskan pada bagian selanjutnya.

B. HARRIS HAWK OPTIMIZATION

Harris Hawk Optimization (HHO) adalah sebuah teknik optimasi *population-based* dan *gradient-free* yang terinspirasi dari perilaku berburu Elang Harris. Dalam melakukan penyerangan, sekelompok elang Harris akan bersama-sama menyergap mangsa dari berbagai arah. Dikarenakan kemampuan mangsa untuk bergerak dan berpindah tempat, maka dilakukan beberapa variasi saat melakukan penyerangan[2].

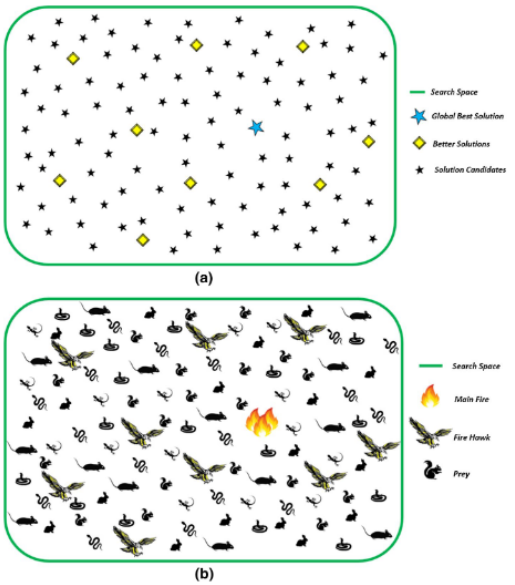
Dalam HHO, elang Harris direpresentasikan sebagai kandidat solusi dan kandidat solusi yang memiliki nilai *fitness* terbaik akan dianggap sebagai mangsa (*rabbit*). Proses pada HHO dibagi menjadi 2 fase, ekplorasi dan eksploitasi. Fase eksplorasi bertujuan untuk mengobservasi dan mengisolasi manga, bergantung terhadap energi mangsa (E). Pada beberapa kasus tertentu, energi yang dimiliki oleh mangsa dapat digunakan untuk berpindah tempat sehingga memaksa elang Harris untuk tetap melakukan eksplorasi dan mencari kandidat mangsa baru, dengan nilai *fitness* yang lebih baik. Ketika energi tidak cukup besar, maka elang Harris akan mengisolasi mangsa dan berpindah menuju tahap eksploitasi. Pada tahapan ini, elang Harris akan berfokus untuk menyerang mangsa secara lebih agresif.

Penyerangan pada fase eksploitasi dibagi menjadi *soft besiege* dan *hard besiege*. Kedua penyerangan ini memiliki variasi dengan ditambahnya perilaku *rapid dives*. Ketika melakukan eksploitasi, terdapat kemungkinan mangsa berhasil kabur. Saat mangsa mampu untuk kabur, maka elang Harris akan memadukan penyerangan dengan *rapid dives*.

Perhitungan energi mangsa pada setiap iterasi didesain sesuai dengan indeks iterasi sedemikian rupa sehingga pada awal iterasi, elang Harris akan lebih banyak melakukan eksplorasi dibandingkan eksploitasi kandidat solusi. Hal ini memberikan kesempatan bagi algoritma untuk menemukan kandidat solusi yang lebih baik sebelum berpindah menuju tahap eksploitasi.

C. FIRE HAWK OPTIMIZER

Fire Hawk Optimizer (FHO) merupakan *novel metaheuristic algorithm* yang terinspirasi dari perilaku mencari makanan dari *whistling kites, black kites,* dan *brown falcons*[5]. Ketiga elang tersebut menggunakan ranting kayu yang terbakar untuk membuat kebakaran kecil yang mengakibatkan mangsa terpaksa keluar dan kabur dari daerah tersebut sehingga mempermudah elang untuk menangkap mangsa. Perilaku tersebut yang membuat ketiga elang tersebut dinamakan *fire hawks*.



GAMBAR 1. Ilustrasi Fire Hawk Optimizer[5]**. Gambar (a) merupakan tahapan inisialisasi. Gambar (b) merupakan pembagian *Fire hawk* dengan mangsa di teritorinya.**

Dalam FHO, sejumlah kandidat solusi akan diasumsikan sebagai posisi vektor dari elang dan mangsa. Posisi awal dari kandidat solusi akan diinisialisasikan dengan proses acak yang dibatasi oleh daerah pencarian yang merepresentasikan batasan maksimum dan minimum, seperti yang ditampilkan pada Gambar 1a. Dari beberapa vektor yang diinisialisasikan, akan ada beberapa solusi yang memiliki nilai *fitness* lebih baik. Solusi-solusi inilah yang dijadikan sebagai *Fire hawks* sedangkan yang lainnya akan menjadi *preys* (mangsa), pada Gambar 1b. Selanjutnya, dilakukan perhitungan jarak *euclidean* untuk setiap pasangan elang dengan mangsa. Hal ini digunakan untuk menentukan teritori setiap elang dengan mencari nilai jarak rata-rata dari mangsa di sekitarnya.

Setiap elang selanjutnya akan bergerak mengambil dan memindahkan ranting terbakar ke teritorinya. Selain itu mangsa juga akan ikut bergerak di dalam daerah teritori setelah elang-elang tersebut menaruh apinya. Terdapat *safe place* di dalam teritori *Fire hawk* yang merupakan tempat aman bagi mangsa untuk berlindung dari *Fire hawk*. *Safe place* dapat dihitung dengan mengambil nilai rata-rata dari mangsa yang ada di daerah tertentu. Mangsa dapat memilih untuk berjalan sesuai perilaku binatang di dalam teritori *Fire hawk*.

Pada akhir iterasi, akan dilakukan perhitungan nilai *fitness* untuk setiap elang dan mangsa, serta akan ditentukan kandidat solusi *global best* (GB) atau solusi terbaik dari semua populasi. Algoritma ini akan terus berjalan hingga mencapai angka iterasi maksimum, dan mengembalikan *global best* sebagai aproksimasi solusi optimal.

D. MOST VALUABLE PLAYER ALGORITHM

Most Valuable Player Algorithm adalah sebuah *metaheuristic algorithm* yang terinspirasi dari sistem kompetisi olahraga. Pada sebuah kompetisi olahraga, sebuah populasi (*population*) dari pemain (P) akan bersaing secara tim (T) untuk memenangkan kompetisi serta mereka juga akan bersaing secara individual untuk memperebutkan gelar MVP (*Most Valuable Player*)[6].

Pada algoritma ini terdapat banyak *players*, banyak tim, dan banyak *fixture* atau pertandingan. Pada setiap *fixture*, akan terdiri dari 4 fase utama, yakni *Competition*, *Application of Greediness*, *Application of Elitism*, dan *Remove Duplicate*.

1) FASE *COMPETITION*

Fase ini akan dilakukan untuk setiap *team* dan terdiri dari 2 fase utama, yakni fase *Individual Competition* dan *Team Competition*. Pada fase *Individual Competition*, setiap *player* dari sebuah tim akan bersaing untuk menjadi *player* terbaik di timnya (*Franchise Player*). Sedangkan pada *Team Competition*, setiap *player* akan bersaing untuk menjadi yang terbaik pada sebuah *competition*.

Setelah melakukan *Individual Competition*, setiap tim (T_i) akan melakukan *Team Competition*. Pada fase ini akan dipilih satu tim lain (T_j) untuk dilawankan dengan tim T_i. Mekanisme pemilihan tim yang menang menggunakan probabilitas proporsional berdasarkan nilai *fitness* kedua tim.

2) FASE *APPLICATION OF GREEDINESS*

Fase ini dilakukan setelah semua tim melewati fase *Individual Competition*. Pada fase ini, semua player akan dibandingkan dengan dirinya sendiri sebelum melewati fase *Team Competition*. Jika player baru lebih baik, maka akan digunakan skill baru dari player tersebut, namun jika player lama lebih baik, maka tetap digunakan skill lama dari player tersebut.

3) FASE *APPLICATION OF ELITISM*

Elitism yang dimaksud pada fase ini adalah mengganti beberapa *player* yang terburuk dan digantikan dengan *player* yang terbaik. Pada MVPA, satu per tiga *player* terbaik akan menggantikan satu per tiga *player* terburuk.

4) FASE *REMOVE DUPLICATE*

Pada fase ini, *player* yang kembar dan bersebelahan, salah satunya akan digantikan dengan *player* terbaik pada urutan setelah player yang menggantikan pada fase *elitism*. Prosedur ini sama dengan yang dijelaskan pada penelitian yang dilakukan oleh Elsayed[12].

Setelah semua *fixture* telah dilakukan atau *stopping condition* telah tercapai, maka solusi terbaik bisa didapatkan dengan mengambil *player* yang memiliki nilai *fitness* terbaik.

III. METODE YANG DIAJUKAN

Metode penyelesaian permasalahan menggunakan algoritma optimisasi HHO, FHO, MVPA, dan algoritma MVPA yang telah dimodifikasi. Permasalahan yang dipilih adalah permasalahan TSP yang telah disesuaikan untuk mencari rute optimal dalam mengunjungi tempat wisata di Pulau Bali.

A. DATA

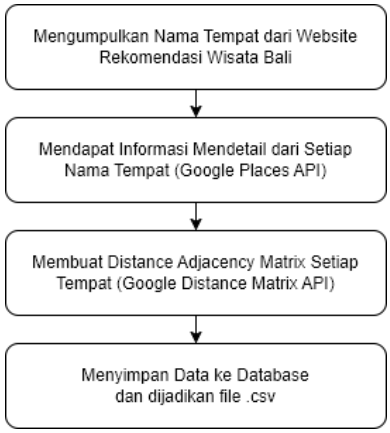
Data yang digunakan dalam penelitian ini adalah data lokasi tempat pariwisata yang berada pada Pulau Bali saja. Pengumpulan data serta detailnya akan menggunakan bantuan Google Places API[13] dan Google Distance Matrix API[14].

Untuk daftar lokasi yang dipilih, dilakukan pengumpulan data dari media online sebanyak 55 lokasi pariwisata populer di Pulau Bali. Lokasi-lokasi tersebut kemudian dimasukkan ke dalam Google Places API untuk mendapatkan informasi seperti nama lengkap, nama jalan, *latitude* dan *longitude*, dan *url* google map. Google Places API akan mengembalikan beberapa tempat yang kemungkinan mirip dengan *query* nama yang telah diinputkan. Dari beberapa hasil, akan diambil hasil yang paling relevan. Setelah itu data lokasi disimpan ke dalam basis data MySQL. Visualisasi proses ini ditampilkan pada Gambar 2.

Dikarenakan pada permasalahan ini setiap *vertex* terhubung satu sama lain, maka diperlukan *distance matrix* dari setiap titik ke titik lainnya dengan ukuran 55×55. Untuk mendapatkan jarak dari satu titik ke titik lainnya, digunakan Google Distance Matrix API. Jarak yang didapatkan dari Google Place API bukanlah hasil perhitungan *euclidean distance*, melainkan dari hasil perhitungan jalan yang sesungguhnya dapat ditempuh oleh kendaraan bermotor yang tentunya bukan merupakan garis lurus. Jarak antar pasangan lokasi-pun juga disimpan ke dalam basis data pada tabel yang berbeda.

B. PERMASALAHAN DAN REPRESENTASI

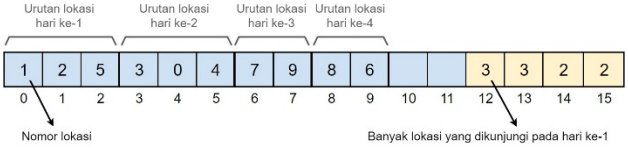
Permasalahan akan dipusatkan pada pencarian rute optimal pada 55 tempat wisata di Pulau Bali. Variabel yang menjadi *input* parameter antara lain jumlah hari berlibur (λ) di Bali dan jumlah lokasi maksimal yang dapat dikunjungi tiap harinya (ϕ). Jumlah lokasi maksimal (M) yang dapat dikunjungi bisa didapat dari hasil perkalian λ dan ϕ. Pada permasalahan ini, terdapat sedikit perbedaan dengan TSP, yakni tidak perlu kembali ke lokasi pertama setelah selesai mengunjungi lokasi lain.



GAMBAR 2. Alur Pencarian Data Lokasi Wisata di pulau Bali

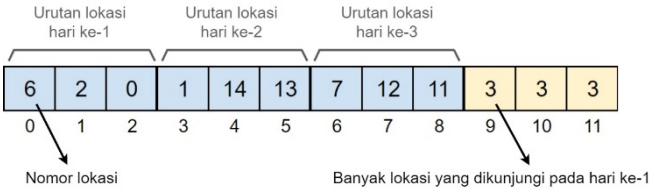
Akan terdapat 2 skenario yang dapat terjadi pada permasalahan ini. Yang pertama, bila jumlah lokasi (n) kurang dari atau sama dengan M, maka setiap lokasi pasti dikunjungi dan akan terdapat hari di mana lokasi yang dikunjungi tidak maksimal atau bahkan tidak mengunjungi lokasi sama sekali. Yang kedua, bila n lebih dari M, maka setiap hari jumlah lokasi yang dikunjungi pasti sejumlah ϕ dan ada kemungkinan terdapat satu atau lebih lokasi wisata yang tidak dikunjungi sama sekali.

Permasalahan ini akan diselesaikan dengan representasi perjalanan kota selama λ. Representasi kandidat solusi berupa array 1 dimensi sepanjang $M+\lambda$. Contohnya, apabila n adalah 55, λ adalah 5 hari dan ϕ adalah 11, maka representasinya adalah sebuah array 1 dimensi sepanjang 60 elemen. M elemen pertama (berjumlah 55) akan berisi nomor lokasi dan melambangkan urutan kunjungan lokasi. Sedangkan, λ elemen terakhir (berjumlah 5) melambangkan banyak lokasi yang akan dikunjungi untuk tiap harinya. Informasi ini digunakan untuk memastikan jumlah lokasi yang dikunjungi perhari tidak melebihi parameter ϕ. Pada skenario pertama, akan terdapat kolom array yang tidak digunakan pada akhir dari M-elemen pertama.



GAMBAR 3. Representasi pada Skenario Pertama

Pada contoh Gambar 3, parameter yang diberikan adalah 4 untuk λ, 3 untuk ϕ dan 10 tempat wisata (n). Melalui perhitungan yang telah dijelaskan sebelumnya, nilai M adalah 12. Karena n kurang dari M, contoh ini termasuk kedalam skenario pertama. M-elemen pertama (indeks ke-0 hingga ke-11) mewakilkan urutan tempat yang dikunjungi, sedangkan λ-elemen sisanya (indeks ke-12 hingga 15) mewakilkan jumlah tempat yang dikunjungi per harinya. Sebagai contoh, elemen indeks ke-12 memiliki nilai 3 yang artinya pada hari pertama, banyak lokasi yang dikunjungi adalah 3 lokasi dengan urutan indeks lokasi 1, 2, dan 5.



GAMBAR 4. Representasi pada Skenario Kedua

Pada contoh Gambar 4, parameter yang diberikan adalah 3 untuk λ, 3 untuk ϕ, dan dengan 15 lokasi wisata (n). Berdasarkan parameter yang diberikan, dapat ditentukan bahwa nilai M adalah 9. Karena nilai n lebih dari atau sama dengan M, contoh ini termasuk dalam skenario kedua. M-elemen pertama (indeks ke-0 hingga 8) mewakilkan urutan tempat yang dikunjungi, sedangkan λ-elemen sisanya (indeks ke-9 hingga 11) mewakilkan jumlah tempat yang dikunjungi per harinya. Sebagai contoh, elemen indeks ke-9 memiliki nilai 3 yang artinya pada hari pertama, banyak lokasi yang dikunjungi adalah 3 lokasi dengan urutan indeks lokasi 6, 2, dan 0.

C. FITNESS FUNCTION

Terdapat 3 fungsi objektif yang akan digunakan sebagai *fitness function*.

1) GLOBAL DISTANCE FUNCTION

Fungsi ini akan digunakan untuk meminimalkan total jarak yang ditempuh selama perjalanan mengunjungi lokasi wisata. Fungsi ini merupakan *inverse* dari total jarak pada kandidat solusi[15], yang selanjutnya akan di-*scaling* untuk mengatur jangkaukan keluaran fungsi. Secara matematis, fungsi ini didefinisikan pada formula (1).

$$F_1 = \frac{c_1}{\sum_{i=2}^{n} d_{s_{i-1}}^{s_i}} \quad \forall\, s_i \in S \tag{1}$$

n = banyaknya lokasi
S = himpunan terurut total dari kota-kota pada perjalanan
d_a^b = jarak antara kota dengan indeks a menuju b
c_1 = konstanta

Himpunan S pada formula (1) yang diberikan merupakan M-elemen pertama dari representasi solusi. Terdapat konstanta c_1 yang digunakan sebagai *scaling factor* bagi fungsi objektif agar tidak memberikan nilai *fitness* yang terlalu berjauhan jika data yang digunakan berbeda. Nilai konstanta c_1 dapat dihitung pada formula (2).

$$c_1 = n \sum_{i=1}^{n} d_{\max}(i) \tag{2}$$

n = banyaknya lokasi
$d_{\max}(i)$ = nilai maksimum ke-i dari matriks jarak d

Perhitungan nilai konstanta pada formula (2) tidak melibatkan nilai dari kandidat solusi, hal ini mengimplikasikan bahwa perhitungan konstanta dapat dilakukan pada tahapan inisialisasi. Dengan demikian, bobot komputasi pada setiap *epoch*-nya dapat dikurangi.

2) LOCATION LIMIT FUNCTION

Fungsi objektif ini digunakan untuk memberikan *punishment* ketika lokasi yang dikunjungi pada suatu hari melebihi batasan ϕ. *Punishment* (p) untuk hari ke-j didefinisikan sebagai formula (3).

$$p(j) = \begin{cases} n_j - \phi, & n_j \geq \phi \\ 0, & n_j < \phi \end{cases} \tag{3}$$

n_j = banyaknya lokasi yang dikunjungi pada hari ke-j

Sistem *punishment*[16] ini dapat dilihat sebagai sebuah *rectifier*. *Penalty* akan diberikan pada hari dengan jumlah lokasi yang dikunjungi melebihi batasan ϕ, tetapi tidak ada *reward* ketika kandidat solusi dapat memenuhi batasan. Nilai akhir yang dikembalikan oleh fungsi objektif ini berupa jumlahan dari seluruh *punishment* masing-masing hari, seperti yang ditunjukkan pada formula (4).

$$F_2 = -1 \sum_{i=1}^{\lambda} p(i) \tag{4}$$

3) DISTANCE VARIANCE PER DAY

Fingsi objektif ini digunakan untuk meminimalisir perbedaan total jarak perhari. Pada perhitungan fungsi objektif ini, digunakan nilai standard deviasi pada data jarak yang ditempuh setiap harinya. Standard deviasi digunakan karena kemampuannya dalam menilai persebaran data[17], sehingga dapat diketahui seberapa jauh perbedaan total jarak yang ditempuh perhari. Fungsi ini didefinisikan sebagai formula (5).

$$F_3 = \frac{c_2}{\sigma(D)+1} \tag{5}$$

$D = \{d_{s_{i-1}}^{s_i} | s_i \in S,\ i \geq n\}$
σ = standard deviasi
c_2 = konstanta
d_a^b dan S referensi menuju formula 1

Terdapat pula sebuah konstanta c_2 yang berperan sebagai *scaling factor* dan dapat dihitung dengan formula (6).

$$c_2 = \frac{d_{\max} - d_{\min}}{2} \tag{6}$$

Nilai d_{max}, dan d_{min} pada formula (6) masing-masing merupakan nilai maksimum dan nilai minimum pada matriks jarak *d*.

Setelah ketiga fungsi objektif pada formula (1), (4), dan (5) digunakan untuk memberikan evaluasi, dilakukan agregasi pada ketiga nilai tersebut untuk memberikan nilai *fitness* tunggal bagi kandidat solusi. Agregasi yang dilakukan menggunakan *weighted sum*[7] dengan formula (7).

$$F = F_1 + F_2 + 0.2F_3 \quad (7)$$

Kegunaan *scaling factor* pada formula (1) dan (5) untuk menyeimbangkan peran masing-masing fungsi pada perhitungan *fitness* di formula (7). Fungsi F_1 pada formula (1) memiliki nilai minimum mendekati *n* yang terjadi apabila total jarak pada kandidat solusi sama dengan total *n* jarak maksimum pada *distance matrix*. Sedangkan, fungsi F_2 pada formula (4) dan F_3 pada formula (5) masing-masing memiliki nilai maksimum 0 dan c_2 (lihat formula (6)). Fungsi F_2 yang bersifat sebagai *punishment*, memiliki bobot yang seimbang dengan fungsi F_1 sebagai objektif utama. Kondisi ini mengharuskan algoritma untuk mencari kandidat solusi yang dapat menyesuaikan dengan batasan, dikarenakan batasan ϕ dianggap sebagai *hard constraint* (termasuk kedalam salah satu input parameter). Sedangkan, fungsi F_3 pada formula (5) yang mengatur persebaran jarak, diberikan bobot yang lebih kecil dan konstanta *scaling factor* yang dinamis menyesuaikan dengan data, menjaga agar fungsi objektif ini tidak terlalu mendominasi fungsi lainnya.

D. MODIFIED MVPA

Selain melakukan uji coba menggunakan tiga algoritma metaheuristik yang telah disebutkan sebelumnya, dilakukan juga uji coba dengan menggunakan hasil modifikasi dari *Most Valuable Player Algorithm*. Modifikasi yang diberikan berada pada bagian *Individual Competition* dan pada fase *Application of Elitism* dan *Application of Greediness*. Perubahan pada Individual Competition ini bertujuan untuk memberikan porsi bagi setiap player melakukan improvisasi mengikuti *Franchise Player* dan *MVP*. Formula modifikasi pada tahapan *Individual Competition* didefinisikan sebagai formula (8).

$$T_i = T_i + (1-\alpha) \times r_1 \times (FP_i - T_i) + \alpha \times r_2 \times (MVP - T_i) \quad (8)$$
$$\alpha = hyperparameter$$

Modifikasi didasarkan pada mutasi *pattern-search-type* seperti formula perubahan kecepatan partikel pada PSO (Particle Swarm Optimization)[7][18]. *Franchise Player* dan *MVP* digunakan sebagai faktor pembanding (seleksi/improvisasi) dengan pemain terkait. Yang menjadi perbedaan dengan formula kecepatan pada PSO adalah, penggunaan α sebagai *control variable*, yang mengatur kecenderungan improvisasi *player*. Nilai α yang mendekati 0 akan membuat *player* lebih cenderung melakukan improvisasi mengikuti *Franchise Player*. Sebaliknya, nilai α yang mendekati 1 akan membuat *player* cenderung melakukan improvisasi mengikuti *MVP*. Bilangan acak r_1 dan r_2 dipertahankan sebagai unsur ketidakpastian.

Sedangkan untuk perubahan pada fase *Application of Elitism* dan *Application of Greediness* digunakan konsep yang mirip dengan penentuan pemenang pada *Team Competition*, yakni dengan melihat probabilitas berdasarkan *fitness* untuk menentukan apakah pada *player* tersebut perlu dilakukan seleksi menggantikan kandidat solusi saat ini. Dengan demikian, akan ada kemungkinan *player* tidak terseleksi pada fase *Greediness* atau *Elitism*. Probabilitas yang digunakan pada *Greediness* dan *Elitism* didapatkan dengan menggunakan formula (9).

$$Prob_{P_j}^{P_i} = \frac{F(P_i)}{F(P_i)+F(P_j)} \quad (9)$$

$Prob_{P_b}^{P_a}$ = probabilitas *player a* menggantikan *player b*
$F(P_a) = fitness\ score$ dari kandidat solusi *player a*

Pada fase *Greediness*, P_i merupakan kandidat solusi *player* sesudah fase *competition*, dan P_j merupakan kandidat solusi *player* yang sama setelah fase *competition*. Sedangkan pada fase *Elitism*, P_i mewakilkan player yang dipilih secara terurut menurun dari satu per tiga player terbaik, dan P_j mewakilkan player yang dipilih secara terurut menaik dari satu per tiga player terburuk.

Perhitungan probabilitas pada formula (9) didasari oleh motivasi untuk memberikan peluang proporsional bagi sebuah solusi yang dianggap lebih buruk untuk bisa *survive*. Pemberian peluang proporsional digunakan untuk menghindari *strong elitism* yang dapat membuat algoritma menuju fase konvergensi terlalu dini[7]. Hal ini membantu populasi untuk tidak terjebak pada *local maxima* dan bisa lebih tersebar untuk mencari solusi dengan nilai *fitness* yang lebih baik.

IV. UJI COBA

Uji coba yang dilakukan berupa pengujian beberapa parameter yang dapat mempengaruhi dan menghasilkan kandidat solusi terbaik. Kota yang akan dikunjungi pertama bersifat acak, dan berdasarkan representasi yang digunakan, algoritma optimisasi akan menentukan kota pertama yang dikunjungi agar dapat memberikan hasil *fitness* yang lebih baik. Uji coba ini akan dilakukan pada layanan *cloud service* dengan *environment* yang sama dan memanfaatkan library *meta-heuristic evolution algorithm MealPy*[19]. Dari ketiga algoritma yang diuji di paper ini, implementasi yang tersedia pada library terkait hanya algoritma FHO dan HHO. Algoritma MVPA dan modified-MVPA diimplementasikan menggunakan bantuan *optimizer class* sebagai *custom optimizer* pada library *MealPy*.

Uji coba yang dilakukan meliputi pengujian skenario seperti yang telah dijelaskan sebelumnya, pengujian *population size* yang berbeda, dan khusus untuk *Modified* MVPA juga dilakukan pengujian nilai *alpha* berbeda. Secara

keseluruhan 7 algoritma yang diuji coba yaitu FHO, HHO, *Original* MVPA, *Modified* MVPA (*alpha* 0.2), *Modified* MVPA (*alpha* 0.4), *Modified* MVPA (*alpha* 0.6), dan *Modified* MVPA (*alpha* 0.8).

Di setiap uji coba, setiap algoritma pada masing-masing konfigurasi ukuran populasi dan jumlah iterasi akan diuji sebanyak 4 kali dan dari ke-empat percobaan tersebut, akan diambil nilai *fitness* terbaik (*Best F*), rata-rata nilai *fitness* (*Mean*), standard deviasi (*Std*), dan rata-rata waktu eksekusi setiap *epoch* (*Avg Time*). Khusus pada algoritma MVPA (*original* maupun modifikasi), terdapat *hyperparameter* tambahan berupa jumlah tim pada populasi (N Tim) dan jumlah *player* yang ada pada satu tim. Jumlah *player* pada satu tim didapat dengan membagi ukuran populasi uji coba dengan jumlah tim.

Uji Coba 1

Pada semua model algoritma, akan dibandingkan beberapa nilai populasi dan iterasi dengan parameter **11 hari *Vacation Duration*** (λ) dan **6 *Limit Location per Day*** (ϕ). Uji coba 1 ini merepresentasikan skenario pertama, sehingga terdapat kemungkinan satu atau beberapa hari, jumlah lokasi yang dikunjungi tidak mencapai ϕ. Hasil kuantitatif dari uji coba ini ditunjukkan pada tabel I.

Uji Coba 2

Pada semua model algoritma, akan dibandingkan beberapa nilai populasi dan iterasi dengan parameter **11 hari *Vacation Duration*** (λ) dan **3 *Limit Location per Day*** (ϕ). Uji coba ini merepresentasikan skenario kedua, sehingga memungkinkan terdapat beberapa lokasi yang tidak dikunjungi sama sekali. Hasil kuantitatif dari uji coba ini ditunjukkan pada tabel II.

Uji Coba 3

Pada semua model algoritma, akan dibandingkan beberapa nilai populasi dan iterasi dengan parameter **30 hari *Vacation Duration*** (λ) dan **6 *Limit Location per Day*** (ϕ). Uji coba ini merepresentasikan skenario pertama dengan durasi berlibur yang panjang, sehingga memberikan opsi cukup banyak bagi algoritma. Hasil kuantitatif dari uji coba ini ditampilkan pada tabel III.

TABEL I
PERFORMA UJI COBA 1 FHO, HHO, *ORIGINAL* MVPA, DAN *MODIFIED* MVPA (11 λ DAN 6 ϕ)

N Tim	Pop	Iter	FHO				HHO				*ORIGINAL* MVPA			
			Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)
12	120	500	23254.8	20866.6	1380.3	0.00454	73529.5	62021.8	9480.81	0.20621	50751.9	39917.9	6800.0	0.25787
12	120	1000	21987.8	20339.8	1007.3	0.00675	72427.1	52996.2	**12951.9**	0.15898	68102.1	53064.2	9606.4	0.24606
12	120	5000	21399.9	20457.9	963.4	0.00571	111870.1	87039.4	21112.0	0.19563	49322.5	43947.8	**3132.0**	0.24655
40	600	500	23736.7	22929.9	650.6	0.02970	98037.8	82469.7	18176.5	1.11382	69879.3	46710.2	14427.8	1.22115
40	600	1000	21767.6	21262.1	**312.1**	0.03468	102490.2	77432.1	15045.6	1.31281	74497.1	51681.9	13868.5	1.24702
40	600	5000	23064.7	22175.2	795.8	0.04065	129357.0	95177.5	29538.9	1.31517	63531.5	50517.7	8139.9	1.34217
100	2000	500	24081.6	22453.9	1227.5	0.13779	**212255.2**	119114	57094.1	5.84633	55192.7	47333.2	5228.6	4.42658
100	2000	1000	**25632.2**	23810.4	1750.9	0.12889	128425.6	88635.6	26122.5	7.49057	**83055.1**	**65325.8**	15785.9	4.72104
100	2000	5000	24462.3	**22943.1**	1003.2	0.11438	190605.0	**134313**	35215.6	6.59766	76942.6	61504.6	14480.4	4.36209

MODIFIED MVPA ALPHA 0.2				*MODIFIED* MVPA ALPHA 0.4				*MODIFIED* MVPA ALPHA 0.6				*MODIFIED* MVPA ALPHA 0.8			
Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)
72536.8	55240.2	10147.8	0.23207	53458.4	48162.7	**3405.3**	0.26262	44938.7	41547.3	**2670.6**	0.20980	41348.2	39250.5	**1626.1**	0.22076
65273.6	48067.3	10124.8	0.21265	74683.8	57595.2	14256.0	0.22506	64729.5	52499.7	8200.9	0.24742	44110.6	39939.5	2451.9	0.22659
64554.1	62096.9	**1942.5**	0.22554	65736.9	56417.2	8055.7	0.22127	62633.9	53504.5	6816.6	0.23949	49862.7	42260.7	5074.6	0.25021
90203.0	77489.3	9885.0	1.10030	90326.2	68316.3	14355.2	1.15466	66181.3	54488.3	7871.9	1.15329	54378.0	45575.9	5117.6	1.04789
88287.4	76866.6	7804.0	1.16666	75389.7	66269.2	7700.8	1.23143	69166.8	59864.9	6476.5	1.28540	72734.8	58917.6	8570.2	1.23640
122483.6	99455.7	18050.4	1.17583	102471.1	80370.2	13306.2	1.25028	81951.6	68653.7	13195.2	1.23039	**73929.0**	56216.7	10251.9	1.23013
140616.5	119350.3	17638.5	4.15141	99912.2	80408.9	13031.3	3.94342	84169.8	65806.7	11457.9	4.33358	69401.3	57225.8	7050.1	4.02652
142138.7	**122547.6**	22082.6	3.81548	91748.1	85118.1	4667.3	4.74443	**86830.6**	69695.5	13114.7	4.03688	62076.4	53802.5	4958.3	4.47950
157275.3	129580.4	18355.7	3.94974	**117414.2**	**91153.5**	24085.3	4.65814	86661.5	**73295.2**	9946.9	4.09069	76949.7	**61768.7**	9248.6	4.39873

TABEL II
PERFORMA UJI COBA 2 FHO, HHO, *ORIGINAL* MVPA, DAN *MODIFIED* MVPA (11 λ DAN 3 ϕ)

N Tim	Pop	Iter	FHO				HHO				*ORIGINAL* MVPA			
			Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)
12	120	500	20182.3	19675.9	480.5	0.00125	60730.1	49099.3	8052.5	0.04120	43229.3	37793.3	5145.0	0.05484
12	120	1000	23658.8	21276.8	1496.2	0.00119	84771.9	78159.5	8225.5	0.04166	49345.2	43570.3	3727.6	0.05381
12	120	5000	21609.6	20916.7	**423.2**	0.00121	88116.9	70816.2	11800.9	0.04234	46638.9	40929.8	4621.6	0.05373
40	600	500	24954.4	22279.0	1733.0	0.00685	90238.0	77971.8	**7917.8**	0.26466	54678.2	53962.4	**796.6**	0.25779
40	600	1000	21539.5	20960.7	784.7	0.00633	165196.8	109512	37951.1	0.27745	61485.7	54413.8	8421.1	0.27378
40	600	5000	22980.7	22187.4	542.7	0.00648	159967.9	81916.7	46863.5	0.25955	56805.2	44442.8	7703.4	0.26776
100	2000	500	24566.9	22953.8	1083.9	0.02139	**179087.2**	**131829.0**	48202.9	1.48002	**89137.1**	**60221.0**	16897.8	0.87154
100	2000	1000	**26898.0**	**24804.0**	1398.7	0.02543	154855.4	131773.7	20239.3	1.37753	68950.2	59262.8	10778.7	0.91145
100	2000	5000	23483.4	22918.9	539.0	0.02641	167967.0	109978.1	34725.8	1.31055	68613.9	58171.2	6377.1	0.94613

MODIFIED MVPA ALPHA 0.2				*MODIFIED* MVPA ALPHA 0.4				*MODIFIED* MVPA ALPHA 0.6				*MODIFIED* MVPA ALPHA 0.8			
Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)
56180.8	44647.6	8134.3	0.04877	67611.7	53248.8	8817.98	0.04709	53802.3	47458.4	4236.7	0.05042	46190.5	39847.2	3773.7	0.05032
64682.5	54547.6	7717.0	0.04746	53975.8	48856.5	**5035.2**	0.05039	48223.5	45337.2	**2312.2**	0.04843	59186.2	47141.6	8131.9	0.04806
76093.8	65997.5	10017.8	0.04727	79443.9	62020.5	10498.5	0.04909	52300.3	49105.2	2752.6	0.04948	66807.6	53527.9	8448.9	0.05073
81788.8	77798.5	4895.5	0.24521	83682.3	68196.2	10336.1	0.25872	64319.1	55569.2	6260.5	0.24371	53575.2	50630.7	**2384.9**	0.23929
89923.3	86868.1	**3135.2**	0.25771	93283.2	75364.5	12865.1	0.23777	75488.6	61661.4	9585.7	0.24697	55818.6	47274.6	5217.9	0.25559
112262.0	102961.3	9337.1	0.23838	107949	93835.4	9084.7	0.23996	75661.6	61240.5	8359.5	0.24979	**76584.4**	**59641.1**	10386.9	0.24276
123783.1	108507.1	10926.6	0.80637	95494.3	86973.4	5929.7	0.83956	76761.6	72552.8	3357.1	0.82310	73362.7	52977.0	12684.7	0.80323
160794.6	**132293.8**	18371.0	0.81349	97866.3	89023.9	7827.7	0.85697	**105925.1**	**83982.8**	14458.3	0.81924	62470.7	55498.5	5064.8	0.88925
174693.8	127636.3	27575.1	0.83819	**132054.9**	**94923.1**	22858.6	0.81763	78444.1	70247.6	5560.2	0.81518	57825.3	53345.9	3471.7	0.93155

TABEL III
PERFORMA UJI COBA 2 FHO, HHO, *ORIGINAL* MVPA, DAN *MODIFIED* MVPA (30 λ DAN 6 ϕ)

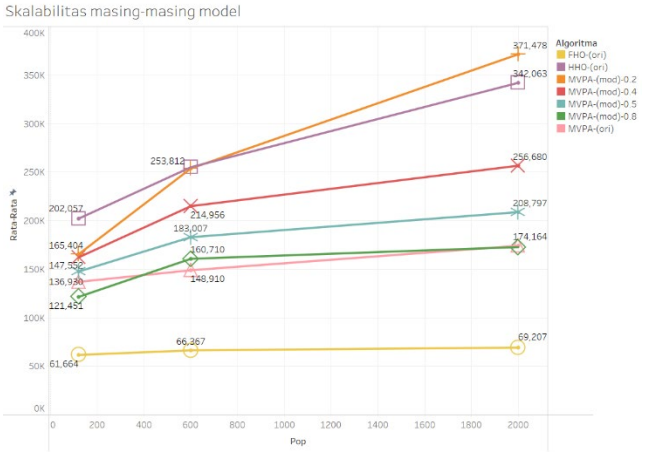
N Tim	Pop	Iter	FHO				HHO				ORIGINAL MVPA			
			Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)
12	120	500	21390.1	20514.3	799.5	0.00133	72778.7	47983.2	14445.5	0.06644	48175.1	40180.1	7597.8	0.08560
12	120	1000	23764.6	20549.7	1866.7	0.00114	72903.3	57560.8	14129.2	0.06342	61021.7	45682.5	9417.6	0.07980
12	120	5000	22846.4	20467.5	1411.8	0.00116	78227.5	66657.4	7103.4	0.08026	54243.2	50688.2	4596.8	0.08229
40	600	500	24891.2	22079.1	1649.7	0.00601	79468.8	71060.2	**6821.2**	0.36980	62881.8	52579.6	9761.9	0.42174
40	600	1000	22449.3	21411.0	680.7	0.00616	107027.2	69691.2	22461.1	0.42771	52096.4	48311.6	5238.7	0.52387
40	600	5000	23469.7	21624.8	1440.1	0.00583	142207.7	103224.9	33539.0	0.39870	58048.5	53832.9	5493.1	0.39482
100	2000	500	**26153.5**	**24580.8**	1220.3	0.02326	161722.8	116387.0	36553.4	1.69491	61089.5	53046.3	6550.4	1.54096
100	2000	1000	23725.3	23143.3	**480.6**	0.02077	162456.9	111941.1	39577.7	1.80135	**70616.0**	53025.1	11241.1	1.32373
100	2000	5000	24413.0	23161.9	781.1	0.02448	**165908.0**	**128842.1**	38582.3	1.82289	59410.2	**57354.8**	**2886.7**	1.50740

MODIFIED MVPA ALPHA 0.2				*MODIFIED* MVPA ALPHA 0.4				*MODIFIED* MVPA ALPHA 0.6				*MODIFIED* MVPA ALPHA 0.8			
Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)
58617.5	51153.1	5403.08	0.07168	56780.5	50497.2	4390.6	0.07279	47894.6	45183.3	3050.6	0.07458	51874.6	42902.9	6807.6	0.07235
61491.3	54185.1	6346.73	0.07533	79204.6	59217.7	14016.3	0.08520	53450.0	44712.9	5788.0	0.07475	53814.9	43882.5	8252.9	0.07712
55958.9	50556.5	**3446.3**	0.07431	51719.7	47015.1	**2820.7**	0.07447	55468.9	52317.8	**2927.6**	0.08208	53324.4	46953.9	4996.0	0.07298
100778.1	84449.8	12940.8	0.37020	76568.6	67638.1	8930.3	0.38438	72232.4	61296.2	11414.9	0.42565	53307.8	49914.5	2916.1	0.40868
105898.6	94876.4	11296.4	0.39416	86341.1	77362.8	6502.1	0.36441	67039.3	59003.1	4758.8	0.38766	56114.4	48436.2	5184.1	0.36654
111706.4	98819.5	12070.1	0.37378	100472	79271.2	13010.7	0.38631	69841.5	61339.7	8079.4	0.43045	**88925.3**	62525.4	15344.1	0.39344
127801.2	111408.6	12151.3	1.25682	92958.4	77523.4	12096.7	1.35042	78329.1	67477.9	6483.0	1.35722	67694.1	52471.3	10504.4	1.28920
116886.8	104600.3	9857.0	1.22911	91118.9	78875.1	8438.9	1.23464	86841.4	70643.1	10367.6	1.28184	70527.3	54459.5	9867.2	1.25031
156474.0	**128452.9**	19576.1	1.29607	**112146.3**	**102599.0**	6098.6	1.41489	**100306.3**	**88753.4**	10385.2	1.26889	66344.5	**64219.1**	**1919.5**	1.36447

Berdasarkan uji coba yang telah dilakukan, berikut analisis efektifitas masing-masing algoritma jika dibandingkan dari skalabilitas, performa eksplorasi, dan hasil kualitatif:

1) PERBANDINGAN SKALABILITAS MASING-MASING ALGORITMA

Skalabilitas yang akan dibahas adalah seberapa besar pengaruh kenaikan populasi terhadap nilai *fitness* dari solusi yang diberikan untuk setiap algoritma. Berikut adalah grafik dari rata-rata *fitness* dari setiap algoritma terhadap perubahan ukuran populasi dari data hasil uji coba tabel I, tabel II, dan tabel III.



GAMBAR 5. Grafik rata-rata *fitness* tiap model untuk berbagai ukuran populasi dengan jumlah *epoch* 500

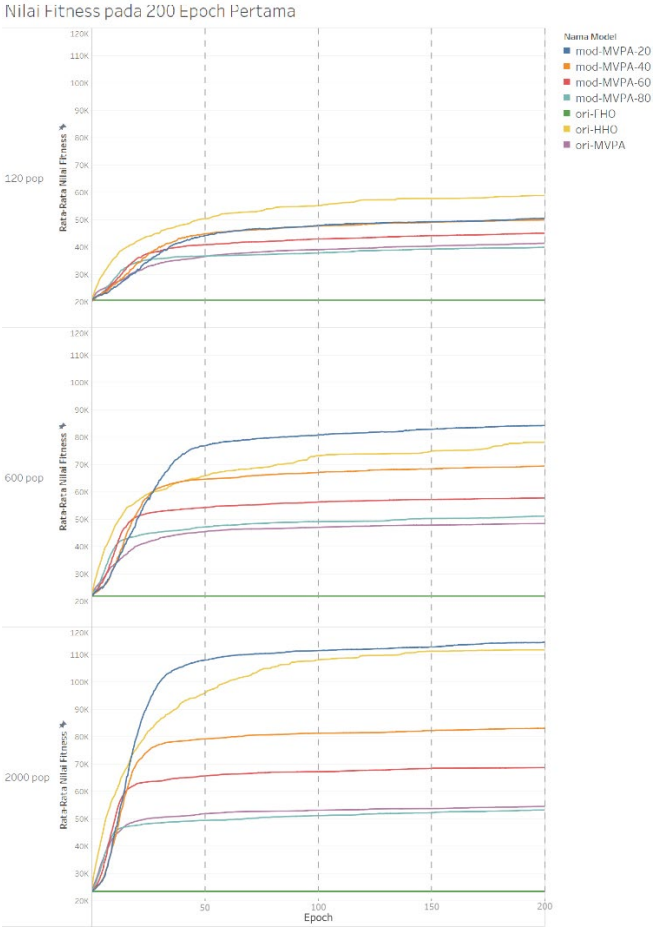
Dari grafik Gambar 5 dapat dilihat bahwa HHO dan *Modified* MVPA dengan *alpha* 0,2 memiliki skalabilitas yang cukup baik dibandingkan dengan algoritma yang lain. Selain itu, semakin kecil *alpha* pada *Modified* MVPA, semakin baik pula skalabilitasnya. Oleh karena itu, dapat disimpulkan nilai *alpha* 0,2 merupakan nilai terbaik untuk *Modified* MVPA.

Jika diperhatikan, skalabilitas yang ditunjukkan oleh *Modified* MVPA cukup baik. Namun, untuk ukuran populasi yang lebih kecil, algoritma *Modified* MVPA kesulitan dalam mendapatkan solusi yang baik. Hal ini menunjukkan bahwa *Modified* MVPA cukup bergantung pada ukuran populasi. Sedangkan algoritma HHO dapat memberikan hasil yang baik walaupun ukuran populasi yang diberikan relatif lebih kecil.

2) GRAFIK PERUBAHAN NILAI *FITNESS* BERBANDING DENGAN *EPOCH*

Gambar 6 menunjukkan perubahan rata-rata nilai *fitness* berbanding pada *epoch* untuk setiap populasi. Dari grafik yang ditunjukkan, dapat dilihat bahwa secara umum, ukuran populasi yang lebih besar akan mengurangi jumlah *epoch* yang dibutuhkan untuk mencapai titik konvergen, karena individu yang terlibat pada pencarian kandidat solusi terbaik lebih banyak. Dari keseluruhan percobaan, algoritma HHO dan *Modified* MVPA dengan alpha 0.2 adalah dua algoritma yang dapat menghasilkan solusi dengan nilai *fitness* yang cukup baik jika dibandingkan dengan FHO dan MVPA biasa.

Dari hasil uji coba, dua algoritma yang memberikan hasil lebih baik secara kuantitatif yaitu *Modified* MVPA dan HHO. Jika diperhatikan lebih lanjut, algoritma HHO memiliki kemampuan eksplorasi lebih tinggi, ditunjukkan dengan kemampuan algoritma dalam menghindari konvergensi dini serta menemukan kandidat solusi baru walaupun kandidat solusi terbaik saat ini memiliki nilai *fitness* yang sangat baik. Hal ini merupakan salah satu efek dari penggunakan fungsi *Lévy flights* yang cukup efektif pada tahapan eksplorasi[7].



GAMBAR 6. Grafik *Average Fitness* pada 200 *Epoch* pertama untuk tiap ukuran populasi

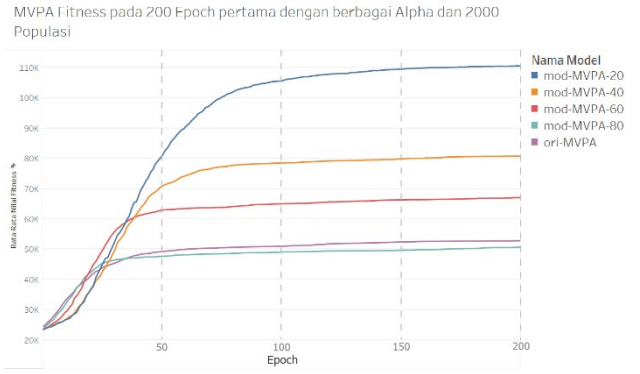
Pada Gambar 6, dapat dilihat perbedaan kecenderungan nilai *fitness* yang dihasilkan algoritma HHO dan *Modified* MVPA. Untuk jumlah populasi yang lebih sedikit, HHO memberikan hasil yang lebih superior dibandingkan *Modified* MVPA. Algoritma *Modified* MVPA juga terlihat kesulitan dalam melakukan eksplorasi lebih ketika memasuki fase konvergensi yang berakibat jika algoritma sudah cukup konvergen, penambahan jumlah *epoch* tidak memberikan pengaruh signifikan pada kualitas kandidat solusi yang diberikan. Berbeda halnya dengan algoritma HHO yang tetap mampu melakukan eksplorasi dan

menemukan kandidat solusi lebih baik meskipun telah memasuki fase konvergensi.

3) PERBANDINGAN MVPA DENGAN MODIFIED MVPA

Gambar 7 memberikan grafik perbandingan nilai *fitness* antara algoritma MVPA dan *Modified* MVPA. Pada grafik yang ditampilkan, secara umum terlihat bahwa algoritma *Modified* MVPA memberikan nilai *fitness* yang lebih baik dibandingkan dengan algoritma *original*, terutama pada pemberian *hyperparameter alpha* 0.2. Selain itu, dapat pula dilihat korelasi antara perubahan nilai *alpha* dengan performa algoritma. Nilai *alpha* yang lebih kecil cenderung memberikan hasil yang lebih baik.

Seperti yang ditunjukkan pada formula 8, nilai *alpha* berpengaruh pada kecenderungan improvisasi *player* pada algoritma *Modified* MVPA. Nilai *alpha* yang lebih kecil membuat *player* lebih cenderung melakukan improvisasi mengikuti *Franchise Player* pada tim terkait. Dominasi pengaruh *Franchise Player* yang lebih signifikan dibandingkan MVP, memberikan kesempatan *player* pada tim untuk dapat bergerak secara berkelompok pada area pencarian, namun tetap memasukkan faktor kandidat solusi terbaik (MVP) pada proses mutasi kandidat solusi.



GAMBAR 7. Grafik Perbandingan MVPA dan Mod MVPA dengan berbagai Alpha

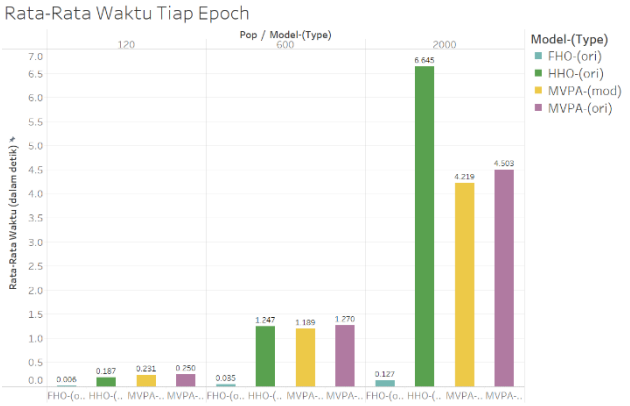
Pada Gambar 7, dapat pula terlihat fenomena *cold start* pada algoritma *Modified* MVPA. Jika diperhatikan kembali pada Gambar 6, fenomena ini dipengaruhi salah satunya oleh ukuran populasi. Fenomena ini terjadi dikarenakan fase *greediness* dan fase *elitism* yang tidak melakukan seleksi secara deterministik (formula 9 sebagai fungsi probabilitas). Solusi yang dinilai lebih buruk tetap memiliki kesempatan untuk bertahan pada kompetisi selanjutnya. Sedangkan, inisialisasi populasi dilakukan secara acak sehingga kebanyakan kandidat solusi memiliki nilai *fitness* yang relatif buruk. Hal ini mengakibatkan pada *epoch* awal, solusi yang memiliki nilai lebih baik tidak cepat untuk mendominasi seleksi dan fase *Team Competition*, serta membantu algoritma dalam memperluas fase eksplorasi sebelum akhirnya mencapai tahap konvergensi yang ditandai dengan mendominasinya kandidat solusi yang memiliki nilai *fitness* baik pada fase *Team Competition*.

Kelemahan dari algoritma *Modified* MVPA terletak pada kemampuan eksplorasinya di *epoch* menengah-akhir. Pada Gambar 7, terlihat algoritma *Modified* MVPA mengalami kesulitan dalam melakukan eksplorasi lebih ketika memasuki *epoch* ke-150. Diperlukan modifikasi yang meningkatkan daya jelajah algoritma agar mampu menemukan kandidat solusi yang lebih baik.

Meskipun hasil uji coba menunjukkan algoritma *Modified* MVPA konsisten memberikan hasil yang lebih baik dibandingkan algoritma MVPA, tetap diperlukan penelitian komperhensif terhadap modifikasi yang dilakukan. Pengujian konvergensi dan uji coba dengan berbagai permasalahan diperlukan untuk memberikan perbandingan akurat dan mengukur keberhasilan modifikasi pada algoritma MVPA.

4) PERBANDINGAN *EPOCH* TIME MASING-MASING ALGORITMA

Gambar 8 menunjukkan perbandingan rata-rata waktu yang dibutuhkan di setiap *epoch*.

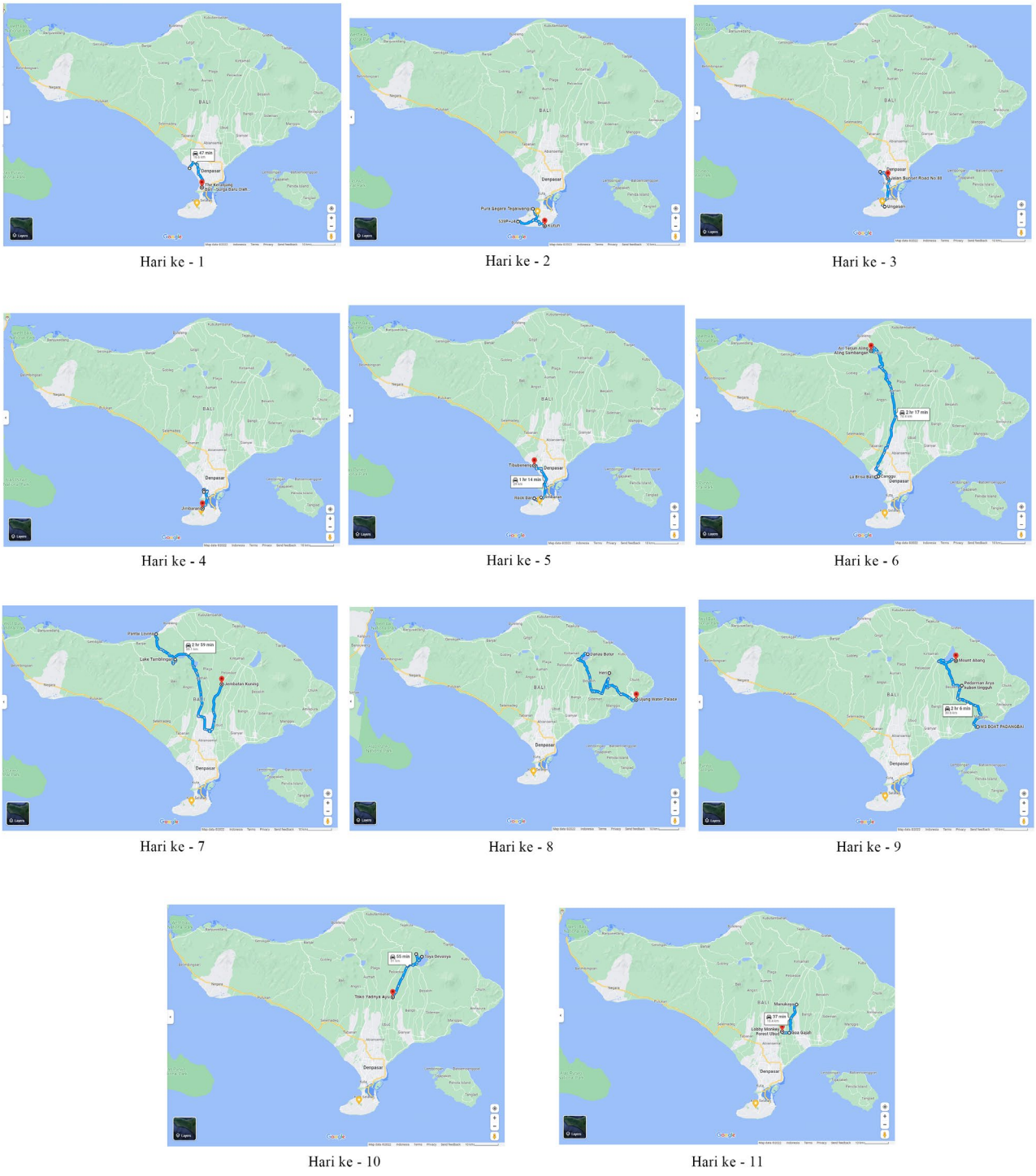


GAMBAR 8. Rata-rata *Epoch* time untuk masing-masing algoritma dan ukuran populasi

Dari data uji coba pada Gambar 8, ditunjukkan bahwa FHO memiliki *epoch* time yang lebih cepat dibandingkan dengan algoritma lainnya. Sedangkan, HHO memiliki *epoch* time yang lebih lama dibanding dengan algoritma lainnya. Algoritma MVPA dan *Modified* MVPA tidak memberikan perbedaan waktu yang cukup signifikan. Pada ukuran populasi 600, HHO dan MVPA memiliki *epoch* time yang relatif sama, namun pada ukuran populasi 2000, HHO memiliki *epoch* time sekitar 50% lebih tinggi dibandingkan dengan MVPA. Perbedaan waktu yang ditunjukkan oleh algoritma HHO bisa jadi dipengaruhi oleh variasi eksplorasi yang melibatkan perhitungan kompleks. Namun, berdasarkan Gambar 6, algoritma HHO mampu memberikan hasil yang baik dengan ukuran populasi yang lebih kecil. Sehingga, efektifitas algoritma HHO dapat diperoleh dengan menggunakan ukuran populasi minimal dan jumlah *epoch* yang lebih banyak.

5) VISUALISASI SOLUSI YANG DIHASILKAN

Rute yang ditampilkan pada visualisasi ini adalah hasil terbaik dari semua percobaan yang telah dilakukan. Hasil ini memiliki nilai *fitness* sekitar 212.000 dan didapatkan dengan algoritma HHO dengan ukuran populasi 2000 dan jumlah *epoch* 500 serta parameter λ=11 dan ϕ=3. Visualisasi solusi ditampilkan pada Gambar 9.



GAMBAR 9. **Visualisasi Rute Hasil Dari Algoritma HHO**

V. DISKUSI, KESIMPULAN, DAN BATASAN

Dari ketiga algoritma yang telah dicoba, HHO merupakan algoritma dengan hasil yang terbaik pada permasalahan TSP. Meskipun dengan waktu eksekusi yang lebih lama dari algoritma lain, hasil yang didapatkan berbeda cukup jauh jiak dibandingkan dengan kedua algoritma lainnya. Sedangkan FHO merupakan algoritma dengan hasil yang terburuk dari ketiga algoritma lainnya, tetapi dengan waktu eksekusi yang paling cepat. Algoritma MVPA yang telah dimodifikasi memiliki hasil yang lebih baik jika dibandingkan dengan MVPA biasa, tetapi masih belum bisa sebaik HHO. Untuk pemilihan algoritma, jika *environment* dan *resource* yang dimiliki mumpuni serta waktu bukan

menjadi penghalang, maka HHO adalah yang paling disarankan karena dapat menghasilkan solusi yang paling baik. Tetapi, ketika *environment* dan *resource* yang dimiliki terbatas serta tidak memiliki cukup banyak waktu, maka MVPA adalah algoritma yang disarankan karena dapat menghasilkan solusi yang cukup baik.

Uji coba yang dilakukan sangat terpusat pada penyelesaian permasalahan spesifik yang didefinisikan pada penelitian ini. Kurangnya variasi permasalahan menjadi salah satu kelemahan dalam melakukan perbandingan efektifitas antar algoritma. Namun, hal ini berada diluar fokus penelitian ini karena yang menjadi fokus penelitian hanyalah perbandingan performa algoritma pada permasalahan spesifik. Sedangkan, hal yang menjadi batasan penelitian pada algoritma modifikasi ialah, tidak adanya uji konvergensi dan variasi uji coba dengan berbagai permasalahan yang lebih komperhensif. Hal ini diperlukan untuk memvalidasi kebenaran dan hasil positif yang diberikan agar tidak bias pada salah satu permasalahan saja.

PERAN PENULIS

Christian Budhi Sabdana: Konseptualisasi, Metodologi dan Implementasi Sistem, Uji Coba, Penyusunan Draf Asli, Penulisan Review & Editing, Visualisasi;
Bryan Christopher: Konseptualisasi, Penyusunan Draf Asli, Visualisasi;
Jason Gerald Sutanto: Konseptualisasi, Metodologi dan Implementasi Sistem, Penyusunan Draf Asli, Penulisan Review & Editing, Visualisasi;
Lawrence Patrick Sianto: Kurasi Data, Sumber Daya, Administrasi Proyek, Implementasi Sistem, Uji Coba, Penyusunan Draf Asli, Penulisan Review & Editing;
Lukky Hariyanto: Konseptualisasi, Penyusunan Draf Asli, Visualisasi;
Nickolas Hartono: Konseptualisasi, Implementasi Sistem, Uji Coba, Penyusunan Draf Asli, Penulisan Review;

COPYRIGHT

DAFTAR PUSTAKA

[1] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, *The traveling salesman problem: A computational study*. 2011.
[2] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Generation Computer Systems*, vol. 97, pp. 849–872, Aug. 2019, doi: 10.1016/j.future.2019.02.028.
[3] A. P. Engelbrecht, "Appendix A: Optimization Theory," *Comput Intell*, pp. 551–579, 2007, doi: 10.1002/9780470512517.app1.
[4] B. Galvan, G. D., P. J., M. Sefrioui, and G. Winter, "Parallel Evolutionary Computation for Solving Complex CFD Optimization Problems : A Review and Some Nozzle Applications," in *Parallel Computational Fluid Dynamics 2002*, Elsevier, 2003, pp. 573–604. doi: 10.1016/B978-044450680-1/50072-3.
[5] M. Azizi, S. Talatahari, and A. H. Gandomi, "Fire Hawk Optimizer: a novel metaheuristic algorithm," *Artif Intell Rev*, vol. 56, no. 1, pp. 287–363, Jan. 2023, doi: 10.1007/s10462-022-10173-w.
[6] H. R. E. H. Bouchekara, "Most Valuable Player Algorithm: a novel optimization algorithm inspired from sport," *Operational Research*, vol. 20, no. 1, pp. 139–195, Mar. 2020, doi: 10.1007/s12351-017-0320-y.
[7] *Nature-Inspired Optimization Algorithms*. Elsevier, 2014. doi: 10.1016/C2013-0-01368-0.
[8] J. C. Bean, "Genetic Algorithms and Random Keys for Sequencing and Optimization," *ORSA Journal on Computing*, vol. 6, no. 2, pp. 154–160, May 1994, doi: 10.1287/ijoc.6.2.154.
[9] C. H. Papadimitriou, "The Euclidean travelling salesman problem is NP-complete," *Theor Comput Sci*, vol. 4, no. 3, pp. 237–244, Jun. 1977, doi: 10.1016/0304-3975(77)90012-3.
[10] S. S. Skiena, *The Algorithm Design Manual*. 1998.
[11] E. Guanabara, K. Ltda, E. Guanabara, and K. Ltda, *The Traveling Salesman Problem and Its Variations*, vol. 12. in Combinatorial Optimization, vol. 12. Boston, MA: Springer US, 2007. doi: 10.1007/b101971.
[12] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "A new genetic algorithm for solving optimization problems," *Eng Appl Artif Intell*, vol. 27, pp. 57–69, 2014, doi: 10.1016/j.engappai.2013.09.013.
[13] "Google Maps Platform Documentation | Places API | Google Developers." https://developers.google.com/maps/documentation/places/web-service (accessed Apr. 20, 2023).
[14] "Google Maps Platform Documentation | Distance Matrix API | Google Developers." https://developers.google.com/maps/documentation/distance-matrix (accessed Apr. 20, 2023).
[15] C. Fu, L. Zhang, X. Wang, and L. Qiao, "Solving TSP problem with improved genetic algorithm," 2018, p. 40057. doi: 10.1063/1.5039131.
[16] T. V Maia, "Avoidance Learning," in *Encyclopedia of the Sciences of Learning*, Boston, MA: Springer US, 2012, pp. 403–406. doi: 10.1007/978-1-4419-1428-6_968.
[17] J. M. Bland and D. G. Altman, "Statistics notes: Measurement error," *BMJ*, vol. 312, no. 7047, p. 1654, Jun. 1996, doi: 10.1136/bmj.312.7047.1654.
[18] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, IEEE, pp. 1942–1948. doi: 10.1109/ICNN.1995.488968.
[19] N. Van Thieu and S. Mirjalili, "MEALPY: a Framework of The State-of-The-Art Meta-Heuristic Algorithms in Python." zenodo, Jun. 22, 2022.

Segmentasi Citra Area Tumpukan Sampah Dengan Memanfaatkan Mask R-CNN

Moch. Rizal Budi Utomo[1], Reddy Alexandro Harianto[1],Endang Setyati[1]
[1]Magister Teknologi Informasi, Fakultas Sains dan Teknologi, Institut Sains dan Teknologi Terpadu Surabaya, Surabaya, Indonesia

Corresponding author: Moch. Rizal Budi Utomo (e-mail: budirizal3@gmail.com).

ABSTRACT This study proposes the use of object detection for trash in images to assist sanitation workers in addressing the problem of garbage accumulation in rivers, vacant lots, and roads. The method employed is Mask R-CNN, a two-stage object detection approach that not only identifies objects using bounding boxes but also performs object segmentation through masking. The dataset consists of 700 training data and 100 validation data captured using a camera. The objects detected are trash items. Testing was conducted by comparing the detection results of the Mask R-CNN model with manual human calculations for 25 test data. The research findings indicate that the Mask R-CNN model delivers better performance with an accuracy rate of 80.16%. With the implementation of this program, sanitation workers can leverage object detection technology to aid in selecting areas that need to be cleaned, thereby reducing the time and effort required for monitoring and removing trash. Additionally, with a satisfactory level of accuracy, this program can serve as an effective tool for identifying trash in images, helping enhance the operational efficiency of sanitation workers and mitigating the negative impacts caused by garbage accumulation in the environment. Therefore, this research makes a significant contribution to the development of technological solutions for addressing garbage accumulation issues and improving the effectiveness of cleanup efforts in rivers, vacant lots, and roads.

KEYWORDS Garbage Segmentation, IOU, Mask R-CNN

ABSTRAK Penelitian ini mengusulkan penggunaan konsep deteksi objek sampah dalam gambar untuk membantu petugas kebersihan dalam mengatasi masalah penumpukan sampah di sungai, lahan kosong, dan jalan. Metode yang digunakan adalah Mask R-CNN, sebuah pendekatan two-stage object detection yang tidak hanya mengidentifikasi objek dengan bounding box, tetapi juga melakukan segmentasi objek dengan masking. Dataset terdiri dari 700 data latihan dan 100 data validasi yang diambil menggunakan kamera. Objek yang dideteksi adalah sampah. Pengujian dilakukan dengan membandingkan hasil deteksi dari model Mask R-CNN dengan perhitungan manual oleh manusia untuk 25 data uji. Hasil penelitian menunjukkan bahwa model Mask R-CNN memberikan performa yang lebih baik, dengan tingkat akurasi sebesar 80.16%. Dengan adanya program ini, petugas kebersihan dapat memanfaatkan teknologi deteksi objek untuk membantu dalam memilih lokasi yang harus dibersihkan, mengurangi waktu dan upaya yang diperlukan untuk memantau dan membersihkan sampah. Selain itu, dengan tingkat akurasi yang memadai, program ini dapat menjadi alat yang efektif untuk mengidentifikasi sampah dalam gambar, membantu meningkatkan efisiensi operasional petugas kebersihan, dan mengurangi dampak negatif yang disebabkan oleh penumpukan sampah di lingkungan. Dengan demikian, penelitian ini memberikan kontribusi yang signifikan dalam pengembangan solusi teknologi untuk mengatasi masalah penumpukan sampah dan meningkatkan efektivitas upaya pembersihan di sungai, lahan kosong, dan jalan.

KATA KUNCI IOU, Mask R-CNN, Segmentasi Sampah

I. PENDAHULUAN

Dengan perkembangan ekonomi, konsumsi masyarakat sangat meningkat, dan jumlah sampah terjadi kenaikan yang sangat signifikan. Terdapat permasalahan tersebar yang masih sering terjadi pada wilayah di Indonesia, yaitu terdapat penumpukan sampah. Lokasi penumpukan sampah yang sering terjadi pada jalan, lahan kosong, sungai, pantai, dan lain-lain. Penumpukan sampah tersebut mempersulit petugas kebersihan untuk mengetahui lokasi yang terdapat tumpukan sampah. Dalam rangka menangani masalah tumpukan sampah, petugas kebersihan saat ini perlu mengunjungi lokasi yang terdampak. Namun, pendekatan pemantauan manual tidak efektif karena tidak dapat mencakup area kerja yang luas dan memerlukan banyak tenaga kerja. Oleh karena itu, solusi yang efektif diperlukan dengan memanfaatkan teknologi guna mengatasi permasalahan ini.. Teknologi yang akan dibuat digunakan untuk mendeteksi tumpukan sampah, agar dapat mempermudah petugas kebersihan untuk memantau sampah pada pintu air. Serta dapat memberikan peringatan untuk petugas kebersihan jika terdapat penumpukan sampah sudah pada level tertentu. Pemantauan kondisi menggunakan CCTV yang mudah dipasang pada area yang sulit.

Dalam era perkembangan teknologi kecerdasan buatan saat ini, masalah deteksi sampah seperti yang dijelaskan sebelumnya dapat diatasi dengan pendekatan teknologi. Salah satunya adalah melalui deteksi objek, di mana sistem dapat mengidentifikasi tumpukan sampah. Proses deteksi ini melibatkan penggunaan gambar dari kamera dan analisis frame menggunakan metode yang ditentukan untuk menemukan objek yang diinginkan. Deteksi objek bekerja dengan mencari keberadaan objek berdasarkan kelas yang ada dalam gambar. Pengembangan teknologi deteksi objek dilakukan dengan menerapkan pembelajaran mesin pada sub-konsep tertentu, seperti Deep Learning [7]. Dalam konsep Deep Learning, komputer akan melatih dirinya untuk memetakan fitur-fitur dari kumpulan gambar, seperti kanal warna dan ukuran objek, guna mengenali pola-pola tertentu yang berguna dalam skema deteksi objek.

Dalam beberapa tahun terakhir, dengan pesatnya pengembangan *deep learning*, peneliti pada bidang *computer vision* telah membuat terobosan dalam deteksi objek. Saat ini, algoritma deteksi objek didasarkan pada model *deap learning,* yang dibagi menjadi dua kategori: (1) pertama menghasilkan *region proposals*, lalu melakukan proses klasifikasi pada area. Contoh umum algoritma kategori pertama adalah algoritma R-CNN berdasarkan *region proposals,* seperti R-CNN[2], Fast-R-CNN[5], Faster R-CNN[1], Mask-RCNN[5],dll; (2) algoritma yang kedua menggunakan proses satu tahap, tidak memerlukan tahap *region proposals,* secara langsung menghasilkan *class probability* dan posisi koordinat objek tersebut, dan membandingkan algoritma seperti YOLO[3] and SSD[4]. Namun, metode ini kurang terpat karena hanya dapat mendeteksi objek yang spesifik, sedangkan pada penelitian ini dibutukan algoritma yang dapat mendeteksi kumpulan objek.

Dalam penelitian ini, Mask R-CNN digunakan untuk deteksi objek area penumpukan sampah karena dapat melakukan proses segmentasi[6]. Untuk menghindari hilangnya presisi karena terlalu banyak koordinat, diperlukan membuat koordinat sebagai penanda area sungai. Sedangkan algoritma untuk menghitung persentase kebedaraan sampah pada area yang sudah ditandai menggunakan *Intersection Over Union (IOU)*.

II. TINJAUAN PUSTAKA

Peneliti melakukan studi literatur pada algoritma yang akan digunakan. Penelitian terkait deteksi objek sampah dilakukan Shuijing Li, Ming Yan [8] dan Aparna Iyer [9]. Penelitian Shuijing Li membahas permasalahan mengklasifikasi sampah. Dengan tujuan membantu proses mengklasifikasi sampah secara akurat dan mengurangi pemborosan waktu. Proses pembuatan kumpulan dataset berdasarkan kriteria klasifikasi sampah yang terdapat pada kota, dan proses training data menggunakan Mask Scoring RCNN. Mask Scoring RCNN menambah struktur penilaian berdasarkan Mask RCNN untuk mendapatkan struktur yang lebih akurat. Dalam *instance segmentation,* kualitas *mask* di *Mask RCNN* umumnya tidak terkait dengan klasifikasi. Oleh karena itu, Mask Scoring RCNN menambahkan modul MaskIou pada basis asli, dan dimasukkan skor prediksi dan karateristik ROI yang diperoleh setelah *Mask* di dalam lapirsan volume dan *full connection layer,* untuk mendapatkan skor modelnya. Keakuratan Mask Scoring RCNN mencapai 65.8% karena proses pengumpulan data dibagi menjadi empat kategori. Algoritma Mask Scoring RCNN lebih sesuai untuk mengenali beberapa objek sekaligus.

Aparna Iyer membahas tentang penggunaan Mask RCNN untuk membantu proses pembuatan profil sampah. Penelitian ini digunakan untuk mengidentifikasi sampah plastic dari kumpulan sampah. Jumlah gambar yang dibutuhkan untuk pembuatan dataset sebanyak 4000 gambar. Setelah itu dilakukan proses *labeling* pada objek sampah plastik untuk mendapatkan hasil model yang tepat. Fokus utama dari system yang diusulkan adalah untuk mengidentifikasi volume pada kandungan plastic. Visual Geometry Group (VGG) adalah alat yang diguanakan untuk pelabelan gambar dan disimpan dalam format JSON. Setelah dataset telah diberi label dan pra-proses, dilanjukan ke pembuatan model menggunakan Mask R-CNN. Hasil pembuatan model diberikan nilai Mean Average Precision atau dikenal dengan MAP yang menunjukkan keakuratan model dalam mengklasifikasi. Selanjutnya, TensorFlow digunakan untuk membangun model Mask RCNN, dan diterapkan di situs web menggunakan Flash. Setelah peneliti membandingkan beberapa model, ternyata Mask R-CNN menghasilkan hasil

yang paling akurat. Hasilnya 75% akurasi pada proses training dan 65% pada proses testing.

Dalam penelitian yang dilakukan oleh Hao Wu, et.al [12] dan JunNian Gou, et.al [13], mereka menggunakan Mask RCNN dengan backbone resnet-101 untuk deteksi objek. Wu menggunakan model pretrained dari dataset mscoco untuk mendeteksi sambungan solder pada papan sirkuit. Data pelatihannya mencakup gambar

dengan komponen yang salah, komponen yang benar, komponen yang diganti, komponen yang terlepas, dan tanpa komponen. Mask R-CNN mencapai performa 100% dalam mendeteksi sambungan solder. Di sisi lain, Gou melakukan deteksi objek kerusakan hasil cetak pada gambar hasil computer tomography (CT). Data gambar yang digunakan mencakup kerusakan seperti gelembung, kerak, dan patahan. Pada cetakan tersebut terdapat faktor noise yang membuat kontras rendah, derau garis keabu-abuan, dan warna kabur pada tepi citra, yang menyulitkan identifikasi objek. Pengujian pada data gambar cetakan CT menunjukkan nilai mAP sebesar 98% untuk kemampuan deteksi objeknya.

III. METODOLOGI PENELITIAN

GAMBAR 1. Metodologi Penelitian

Penelitian ini mengadopsi metodologi yang terdiri dari lima tahap, yaitu pengumpulan dataset, pembentukan model deteksi, pelatihan model deteksi, pengujian, dan evaluasi model. Peneliti akan menyajikan alur metodologi penelitian melalui Gambar 1 guna memberikan pemahaman yang lebih jelas.

A. PENGUMPULAN DATASET

Lokasi yang dipilih untuk pengumpulan dataset sangat penting dalam pembuatan model deteksi sampah. Dalam penelitian ini, lokasi yang dipilih adalah lokasi yang terdapat penumpukan sampah, seperti pintu air sungai, lahan kosong, pasar, dan lain-lain. Pengumpulan data pada lokasi ini akan memastikan bahwa dataset yang diambil sesuai dengan objek yang akan dideteksi. Selain itu, lokasi ini juga memiliki beberapa jenis lokasi dan kondisi, seperti pagi, siang, dan malam, yang memastikan variasi data yang diambil.

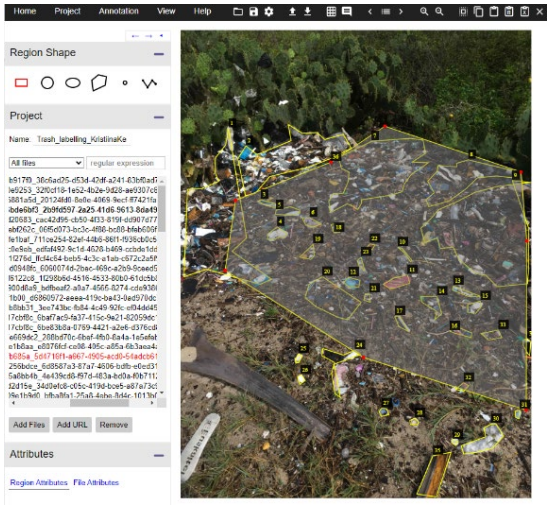
Cara pengumpulan dataset juga sangat penting dalam pembuatan model deteksi sampah. Dalam penelitian ini, dataset diambil menggunakan smartphone. Menggunakan smartphone memastikan bahwa dataset yang diambil dapat diproses dengan cepat dan mudah. Selain itu, smartphone juga memiliki kualitas gambar yang baik, sehingga data yang diambil akan memiliki kualitas yang baik pula.

Dalam penelitian ini, jumlah dan distribusi dataset memiliki peranan penting dalam pelatihan model deteksi sampah. Terdapat total 27.963 gambar yang digunakan sebagai dataset. Pembagian dataset dilakukan dengan alokasi 70% untuk data training dan 30% untuk data testing. Langkah ini penting untuk memastikan bahwa model yang dikembangkan memiliki akurasi yang tinggi dan mampu memberikan hasil yang baik saat melakukan deteksi sampah..

GAMBAR 2. Contoh Data Tumpukan Sampah Pada Sungai dengan kondisi siang hari

GAMBAR 3. Contoh Data Tumpukan Sampah Pada Sungai dengan kondisi malam hari

Contoh pengambilan gambar untuk pembuatan dataset ditampilkan dalam Gambar 2 dan 3. Pengambilan data dilakukan menggunakan kamera digital dengan resolusi 2000x1333 piksel dan menghasilkan berkas berukuran 2-4MB. Setelah proses akuisisi, ukuran file gambar akan dikurangi menjadi sekitar 600 hingga 800KB, tetapi tetap mempertahankan ukuran piksel yang sama. Selanjutnya, gambar-gambar tersebut akan dilabeli. Dalam kasus ini, proses pelabelan untuk Mask R-CNN menggunakan format data JSON yang dibuat oleh peneliti dengan bantuan aplikasi *VIA*.



GAMBAR 4. Proses Labeling

Proses pelabelan objek untuk Mask R-CNN dilakukan dengan menggunakan poligon yang mengikuti bentuk objek tersebut. Sebagai contoh, dalam Gambar 4, peneliti menggunakan aplikasi VIA untuk melakukan pelabelan objek.

Bentuk format yang akan dihasilkan oleh aplikasi VIA adalah JSON. Setiap objek yang dibentuk pada setiap gambar akan disimpan dengan format JSON.

B. PEMBENTUKAN MODEL DETEKSI

Pada tahap ini, dilakukan konfigurasi parameter untuk algoritma Mask R-CNN dan penjelasan mengenai pemilihan model pre-trained dan backbone. Konfigurasi parameter model Mask R-CNN akan disesuaikan dengan kemampuan komputasi dan kapasitas memori pada kartu grafis yang digunakan. Hal ini bertujuan agar kartu grafis dapat mengolah gambar dengan baik, mengingat Mask R-CNN membutuhkan sumber daya komputasi yang tinggi [10]. Sebagai contoh, peneliti Matija Buric, et.al [11] menggunakan model Mask R-CNN untuk mendeteksi objek bola. Mereka menggunakan gambar masukan dengan ukuran 1024x1024 piksel untuk diberikan kepada model. Selanjutnya, peneliti melakukan pembentukan model dengan mengkonfigurasi parameter yang dijelaskan dalam Tabel 1.

TABEL I
KONFIGURASI MODEL MASK R-CNN

Parameter	Nilai
Backbone	Resnet101
Backbone_strides	[4, 8, 16, 32, 64]
Batch_size	1
Detection_min_confidence	0.7
Detection_nms_threshold	0.3
FBN_fc_layer_size	1024
Image_pex_gpu	1
Image_max_dim	1024
Image_min_dim	800
Learning_rate	0.001
Num_classes	1
RPN_nms_threshold	0.7
Steps_per_epoch	1000
Validation_steps	50
Weight_decay	0.0001

C. PELATIHAN MODEL DETEKSI

Pada tahap pelatihan model deteksi, digunakan algoritma Mask R-CNN. Awalnya, model ini menggunakan pre-trained model dari mscoco karena dataset ini memiliki jumlah data yang besar dan mencakup 80 kelas objek yang beragam.

Meskipun Mask R-CNN memiliki banyak varian backbone yang dapat dipilih, pada penelitian ini dipilih backbone Restnet-101. Pelatihan model Mask R-CNN dilakukan melalui beberapa epoch, dan hasilnya disimpan dengan ekstensi .h5.

Dengan melatih model Mask R-CNN menggunakan data dari mscoco dan menggunakan backbone Restnet-101, diharapkan hasil yang diperoleh akan lebih optimal dan memuaskan. Oleh karena itu, tahap pelatihan model deteksi ini sangat penting untuk menghasilkan model yang mampu melakukan deteksi objek dengan baik dan akurat.

D. PENGUJIAN MODEL

Pada tahap ini, model dengan akurasi terbaik selama pelatihan dipilih sebagai model utama untuk pengujian. Model yang memiliki performa yang baik dan stabil penting untuk memastikan hasil deteksi yang akurat. Setelah mendapatkan model terbaik, dilakukan proses pengujian. Pengujian gambar dilakukan dengan menggunakan model terbaik yang telah dipilih sebelumnya. Gambar yang digunakan berasal dari dataset yang telah dikumpulkan sebelumnya. Setelah dilakukan pengujian menggunakan model Mask R-CNN, hasil deteksi akan memberikan kotak penanda dan lapisan masking yang melingkupi objek sesuai dengan kelas objek yang terdeteksi. Pada akhir proses pengujian, hasil deteksi akan menghasilkan perhitungan objek yang terdeteksi sesuai dengan kelas yang diprediksi oleh model Mask R-CNN. Hasil akhir ini akan membantu

peneliti dalam mengevaluasi performa model yang digunakan dan juga sebagai dasar untuk perbaikan model dan proses pelatihan di masa depan.

E. EVALUASI MODEL

Setelah proses pengujian model deteksi sampah selesai dilakukan, hasilnya akan dianalisis dan diberikan evaluasi terhadap model tersebut. Evaluasi ini penting dilakukan untuk menentukan tingkat akurasi dan keandalan model deteksi sampah yang dibangun. Dalam proses evaluasi ini, akan menggunakan metode-metode tertentu untuk memastikan bahwa model yang dibangun dapat mencapai performa yang baik dan memenuhi harapan dari penelitian ini.

Dalam evaluasi model Mask R-CNN menggunakan metrik IoU (Intersection over Union), langkah-langkahnya dimulai dengan mempersiapkan dataset yang telah diannotasi dengan label piksel untuk objek yang ada dalam gambar. Kemudian, menjalankan model pada gambar-gambar dari dataset pengujian untuk menghasilkan prediksi segmentasi piksel. Selanjutnya, dilakukan perhitungan untuk mengukur sejauh mana area segmentasi piksel yang diprediksi oleh model dan area segmentasi piksel yang sebenarnya tumpang tindih. Dengan menggunakan rumus IoU, IoU dihitung dengan membagi luas tumpang tindih oleh luas gabungan.

IV. HASIL DAN PEMBAHASAN

Dalam tahap ini, peneliti akan menjelaskan tentang hasil pembentukan dataset, inisialisasi perangkat yang digunakan dalam penelitian, hasil pelatihan model deteksi, hasil uji coba model, dan evaluasi model. Penjelasan akan disajikan dalam beberapa poin berikut.

A. HASIL PEMBENTUKAN DATASET

Pada proses pembentukan dataset pada penelitian ini, pengumpulan gambar dilakukan dengan memilih lokasi-lokasi yang memiliki tumpukan sampah sebagai objek yang akan dideteksi. Dataset yang dikumpulkan menggunakan smartphone dan diambil pada beberapa jenis lokasi seperti pintu air sungai, lahan kosong, pasar, dll dan pada beberapa kondisi seperti pagi, siang, dan malam. Dataset yang digunakan terdiri dari 27.963 gambar yang akan dipisahkan menjadi 70% untuk data training dan 30% untuk data testing.

Hasil pembentukan dataset pada penelitian ini menunjukkan bahwa dataset yang dikumpulkan memiliki variasi yang cukup baik dalam hal objek sampah dan kondisi lingkungan. Variasi objek sampah dapat membantu model deteksi untuk mengenali berbagai jenis sampah dan tidak hanya terpaku pada satu jenis objek saja. Variasi kondisi lingkungan juga membantu model untuk mengenali objek sampah pada berbagai kondisi, seperti pada saat terik matahari, pada malam hari, atau pada kondisi hujan.

Dalam pembentukan dataset untuk Mask R-CNN, digunakan teknik anotasi berbentuk garis polygon. Konsep anotasi ini melibatkan penggunaan kumpulan titik untuk mengelilingi objek dan membentuk lapisan di area objek tersebut. Proses anotasi ini akan diterapkan pada semua gambar, baik yang digunakan untuk pelatihan maupun pengujian.

Proses anotasi dilakukan dengan memilih objek yang ingin dideteksi dan mengelilinginya dengan garis polygon. Setiap objek yang dianotasi akan diberikan label sesuai kelas objek tersebut. Misalnya, objek burung dianotasi sebagai kelas burung, objek mobil sebagai kelas mobil, dan seterusnya.

Ketelitian dalam proses anotasi akan mempengaruhi akurasi model deteksi. Oleh karena itu, dibutuhkan kesabaran dan kehati-hatian dalam melakukan proses anotasi. Jika anotasi dilakukan dengan benar, maka hasil deteksi objek pada gambar akan semakin akurat dan memuaskan. Namun, jika terjadi kesalahan pada proses anotasi, maka hasil deteksi objek pada gambar akan kurang akurat.

GAMBAR 5. Contoh Hasil Anotasi

Semua gambar training dan testing, hasil akhirnya akan diberikan dalam bentuk file json (javascript object notation) yang memiliki format sesuai dengan konsep yang diterapkan pada COCO dataset. Dalam Gambar 5 yang ada, hasil proses anotasi yang dilakukan terlihat jelas dengan menghasilkan file *json (javascript object notation)* yang sesuai dengan format COCO dataset.

B. UJI COBA DAN EVALUASI

Peneliti melakukan tahap uji coba dan evaluasi pada model Mask R-CNN dengan menggunakan berkas pembobotan hasil pelatihan yang memiliki akurasi terbaik. Tahap uji coba dan evaluasi ini merupakan bagian penting dalam pembuatan model deteksi, karena dapat mengukur performa model tersebut. Peneliti memuat berkas pembobotan dengan nilai loss terendah, menunjukkan bahwa model memiliki akurasi yang baik.

Setelah memuat berkas pembobotan, proses deteksi dimulai dengan memproses gambar uji. Hasil deteksi menghasilkan proposal wilayah dalam bentuk bounding box dengan empat titik koordinat, serta lapisan atau layer yang mewakili area objek yang terdeteksi. Waktu yang dibutuhkan

untuk proses deteksi berkisar antara 3 hingga 5 detik, tergantung pada ukuran dan kompleksitas gambar yang digunakan. Gambar 6 dan Gambar 7 menunjukkan contoh keluaran hasil deteksi yang menggambarkan performa model Mask R-CNN pada tahap uji coba dan evaluasi.

GAMBAR 6. **Deteksi Objek Kondisi Pagi hari dengan Mask R-CNN**

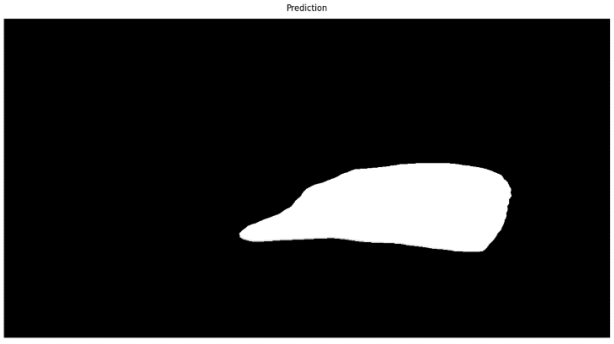
GAMBAR 7. **Deteksi Objek Kondisi Malam hari dengan Mask R-CNN**

Pada gambar di atas, dapat dilihat hasil pengujian deteksi yang ditandai dengan area berwarna merah. Selanjutnya dilakukan proses menghitung persentase hasil deteksi menggunakan rumus IOU. IOU (Intersection over Union) adalah salah satu metrik evaluasi yang umum digunakan dalam bidang Computer Vision, khususnya dalam tugas segmentasi gambar dan deteksi objek. IOU dapat digunakan untuk mengukur seberapa baik sebuah model dapat memprediksi lokasi atau segmentasi objek pada gambar dengan membandingkan hasil prediksi model dengan nilai ground truth atau nilai yang sebenarnya.

Pada dasarnya, IOU mengukur seberapa besar area overlap antara dua region (dalam kasus segmentasi gambar) atau dua bounding box (dalam kasus deteksi objek), dibandingkan dengan total area dari kedua region atau bounding box. Nilai IOU berkisar dari 0 hingga 1, di mana 0 menunjukkan tidak adanya overlap antara dua region atau bounding box, dan 1 menunjukkan kedua region atau bounding box saling tumpang tindih secara sempurna.

$$IOU = \frac{Intersection}{Union} \quad (1)$$

IOU dapat dihitung dengan membagi luas intersection (area overlap) antara dua region atau bounding box dengan luas union (total area) dari kedua region atau bounding box. Disajikan pada Persamaan (1) adalah rumus perhitungan dari IOU.



GAMBAR 8. **Merubah gambar sebelum proses menghitung IOU**

Sebelum proses perhitungan IOU dilakukan proses mengubah gambar menjadi hitam putih, untuk area hasil deteksi dan area deteksi dirubah menjadi warna putih, diluar area tersebut dirubah menjadi warna hitam seperti Gambar 8. Sehingga proses perhitungan IOU memandingkan Gambar Hasil deteksi dan Gambar Area Deteksi. Peneliti telah menghasilkan tabel yang menunjukkan hasil pengujian terhadap 25 gambar uji menggunakan R-CNN. Tabel 2 berikut ini memperlihatkan hasil pengujian tersebut:

TABEL II
PENGUJIAN DETEKSI SAMPAH

Gambar	IOU
1.jpg	88%
2.jpg	79%
3.jpg	83%
4.jpg	83%
5.jpg	84%
6.jpg	81%
7.jpg	73%
8.jpg	84%
9.jpg	85%
10.jpg	86%
11.jpg	74%
12.jpg	70%
13.jpg	81%
14.jpg	80%
15.jpg	74%
16.jpg	72%
17.jpg	83%
18.jpg	85%
19.jpg	93%
20.jpg	79%
21.jpg	75%
22.jpg	73%
23.jpg	80%

24.jpg	83%
26.jpg	76%

Pada pengujian Mask R-CNN terhadap 25 data gambar uji, hasilnya menunjukkan deteksi yang baik. Rentang hasil deteksinya meliputi jumlah yang sesuai dengan yang sebenarnya maupun jumlah yang berbeda jauh. Dalam beberapa kasus, terdapat kesalahan deteksi pada objek. Hasil akhir menunjukkan nilai rata-rata Intersection over Union (IOU) untuk objek sampah sebesar 80.16%.

V. KESIMPULAN

Dari hasil uji coba yang dilakukan, dapat disimpulkan bahwa Mask R-CNN dapat diimplementasikan dengan baik untuk deteksi objek sampah pada gambar. Berbagai variasi data pengujian dalam penelitian ini menunjukkan bahwa Mask R-CNN memberikan performa deteksi yang lebih baik. Mask R-CNN mampu mencapai nilai mean average precision sebesar 80.16% dalam mendeteksi semua kelas objek pada gambar. Meskipun menghadapi variasi skenario gambar yang beragam, seperti sudut pengambilan dan kondisi pencahayaan yang berbeda, Mask R-CNN mampu mengatasi tantangan tersebut dalam deteksi objek, seperti yang terlihat pada tabel hasil pengujian.

Rekomendasi untuk penelitian selanjutnya adalah mempertimbangkan pengolahan kondisi saat hujan dan melakukan data augmentasi di luar proses algoritma untuk meningkatkan variasi data dan jumlah data pelatihan. Perbaikan tahap preprocessing, seperti pelabelan dan perbaikan citra, juga dapat dipertimbangkan untuk meningkatkan kualitas data penelitian dan performa deteksi. Mengingat alokasi sumber daya perangkat keras dan waktu komputasi yang dibutuhkan oleh kedua algoritma, alternatif framework, library, atau model yang lebih ringan dari segi arsitektur dan beban komputasi dapat dijajaki.

PERAN PENULIS

Moch. Rizal Budi Utomo: Konseptualisasi, Metodologi, Perangkat Lunak, Validasi, Investigasi, Sumber Daya, Kurasi Data, Penyusunan Draft Asli, Visualisasi;

Reddy Alexandro: Konseptualisasi, Metodologi, Perangkat Lunak, Validasi, Analisis Formal, Investigasi, Penyusunan Draft Asli, Peninjauan Dan Penyuntingan, visualisasi, pengawasan, administrasi proyek;

Endang Setyati: Validasi, Analisi Formal, Investigasi, Peninjauan Dan Penyuntingan, Pengawasan, Administrasi Proyek;

COPYRIGHT

DAFTAR PUSTAKA

[1] Ren, Shaoqing, et al. "Faster R-CNN: towards real-time object detection with region proposal networks." *International Conference on Neural Information Processing Systems* , pp. 91-99,2015.

[2] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580-587, 2014.

[3] Redmon, Joseph, et al. "You only look once: Unified, realtime object detection." *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779-788, 2016.

[4] Liu, Wei, et al. "Ssd: Single shot multibox detector." *European conference on computer vision*. Springer, Cham, pp. 21-37, 2016.

[5] Girshick, Ross. "Fast r-cnn." *Proceedings of the IEEE international conference on computer vision*, pp. 1440-1448, 2015.

[6] He, Kaiming, et al. "Mask R-CNN." *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 99, pp. 1-1, 2017.

[7] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 2, pp. 386–397, 2020.

[8] Shuijing Li, Ming Yan, "Garbage object recognition and classification based on Mask Scoring RCNN", IEEE International Symposium on Multimedia, 2021.

[9] Aparna Iyer, et al. "A Garbage Profiling System Using Mask R-CNN Deep Learning Algorithm", IEEE International Symposium on Multimedia, 2022.

[10] Setyaningsih., E.R., et.al, "YOLOv4 dan Mask R-CNN Untuk Deteksi Kerusakan Pada Karung Komoditi", *TEKNIKA, Volume 11(1),* Maret 2022.

[11] M. Buric, M. Pobar, and M. Ivasic-Kos, "Ball detection using Yolo and Mask R-CNN", *International Conference on Computational Science and Computational Intelligence (CSCI),* 2018.

[12] H. Wu, W. Gao, and X. Xu, "Solder joint recognition using mask R-CNN method," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 10, no. 3, pp. 525–530, 2020.

[13] J. N. Gou, X. Y. Wu, and L. Liu, "Detection and Segmentation of Defects in Industrial CT Images Based on Mask R-CNN," *Journal of Computers*, vol. 31, no. 6, pp. 141–154, 2020.

INSYST

Journal of Intelligent System and Computation

Volume 05 Nomor 01 April 2023

Author Guidelines

- Manuscript should be written in Indonesia and be submitted online via journal website. Online Submission will be charged at no Cost
- Manuscript should not exceed 15 pages including embedded figures and tables, without any appendix, and the file should be in Microsoft Office (.doc/.docx). download template
- Title should be less than 15 words
- Abstracts consists of no more than 200 words, contains the essence of the article and includes a brief background, objectives, methods and results or findings of the study. Abstract is written in one paragraph.
- Keywords are written in Indonesia three to five words/phrases, separated with coma and consist of important words/phrases from the article.
- Author's name, affiliation, affiliation address and email. State clearly and include country's name on your affiliation address.
- The main text of the writing should be consists of: Introduction, Method, Result and Discussion, and Conclusion; followed by Acknowledgment and Reference
- Introduction State adequate background, issues and objectives, avoiding a detailed literature survey or a summary of the results. Explain how you addressed the problem and clearly state the aims of your study.
- Used method is the scientific in the form of study of literature, observation, surveys, interviews, Focus Group Discussion, system testing or simulation and other techniques commonly used in the world of research. It is also recommended to describe analysis techniques used briefly and clearly, so that the reader can easily understand.
- Results should be clear, concise and not in the form of raw data. Discussion should explore the significance of the results of the work, not repeat them. Avoid extensive citations and discussion of published literature. INSYST will do the final formatting of your paper.
- Conclusion should lead the reader to important matter of the paper. Authors are allowed to include suggestion or recommendation in this section. Write conclusion, suggestion and/or recommendation in narrative form (avoid of using bulleting and numbering)
- Acknowledgments. It is highly recommended to acknowledge a person and/or organizations helping author(s) in many ways. Sponsor and financial support acknowledgments should be included in this section. Should you have lots of parties

to be acknowledged, state your acknowledgments only in one paragraph. Avoid of using bulleting and numbering in this section

- The number of references are not less than 10 with at least 8 primary references. Primary references are include journal, thesis, disertation and all kinds of research reports. All refferences must come from source published in last 7 years.
- Figure and table should be in black and white, and if it is made in color, it should be readable when it is later printed in black and white.
- Figure and table should be clearly readable and in a proportional measure to the overall page.

Tim Redaksi

Journal of Intelligent System and Computation

Departement of Informatics

Institut Sains dan Teknologi Terpadu Surabaya

Jl. Ngagel Jaya Tengah 73-77 Surabaya

Email: insyst@istts.ac.id

Website: https://jurnal.stts.edu/index.php/INSYST/index