JURNAL RESPONSIF, Vol. 3 No.2 Agustus 2021, pp. 199~206
E-ISSN**:** 2685-6964

KLASIFIKASI JENIS KENDARAAN RODA EMPAT MENGGUNAKAN *EXTREME LEARNING MACHINE*

Ina Najiyah[1], Salman Topiq[2]

[1]Universitas Adhirajasa Reswara Sanjaya
e-mail: inajiyah@ars.ac.id

[2]Universitas Adhirajasa Reswara Sanjaya
e-mail: salman@ars.ac.id

Abstrak

Kendaraan merupakan sebuah objek yang menjadi alat transportasi penduduk khususnya di Indonesia. Kendaraan roda empat saat ini sudah beranekaragam jenisnya mulai dari kendaraan kecil, sedang sampai kendaraan besar. Tujuan penelitian ini melakukan klasifikasi jenis kendaraan roda empat dengan bidang *Image Processing.* Metode yang dipilih adalah metode *Extreme learning machine*, dimana metode ini cukup baik dalam melakukan pemroresan gambar dan untuk klasifikasi. Penelitian ini mengklasifikasikan kedalam empat class yaitu class sedan, MVP, truck dan Bus dengan total masing-masing *dataset* 100 gambar. Hasil dari penelitian ini membuktikan bahwa metode *Extreme learning machine* baik untuk mengklasifikasikan kendaraan roda empat dengan akurasi baik yaitu 86% dan nilai precisionnya 82%.

Kata Kunci: Klasifikasi Jenis Kendaraa*n, Extreme learning machine, Image Processing.*

Abstract

Vehicle is an object that becomes a means of transportation for the population, especially in Indonesia. Currently, there are various types of four-wheeled vehicles ranging from small, medium to large vehicles. The purpose of this study is to classify the types of four-wheeled vehicles in the field of Image Processing. The method chosen is the Extreme learning machine method, where this method is quite good at processing images and for classification. This study classifies into four classes, namely sedan, MVP, truck and bus classes with a total of 100 images for each dataset. The results of this study prove that the Extreme learning machine method is good for classifying four-wheeled vehicles with a good accuracy of 92% and a precision value of 82%.

Keywords: *Classification of Vehicle Types, Extreme learning machine, Image Processing.*

1. Pendahuluan

Jumlah kendaraan di Indonesia khususnya mobil semakin hari semakin meningkat kuantitasnya (Priyambodo, 2018). Kondisi meningkatnya jumlah kendaraan tersebut maka pemerintah menugaskan pihak kepolisian untuk memantau kondisi kendaraan tersebut di lalu lintas guna meningkatkan keamanan. Kondisi yang ada saat ini, penggunaan kamera pengintai di lalu lintas cukup membantu pihak kepolisian dalam memantau mobil-mobil yang dicurigai, hanya saja pengecekan dilakukan masih secara manual. Pengecekan secara manual tersebut masih belum cukup baik, dikarenakan waktu yang dibutuhkan kurang efisien serta tingkat keakurasian yang masih terbilang rendah sehingga dibutuhkan sebuah system untuk membantu mendeteksi mobil-mobil tersebut secara otomatis guna membantu pihak kepolisian dalam mengintai mobil dan tetap menjaga keamanan secara efektif. Sistem tersebut dapat dibantu dengan bidang teknologi informasi. Bidang teknologi informasi saat ini sudah banyak digunakan untuk membantu mengatasi berbagai macam permasalahan, salah satunya yaitu bidang *Machine learning*.

Machine learning merupakan salah satu bidang pembelajaran mesin dimana dapat membantu permasalahan seperti deteksi dan klasifikasi (Roihan et al., 2020). *Machine learning* dapat diusulkan dalam mengatasi permasalahan deteksi jenis

Naskah diterima 11 Agustus 2021; direvisi 15 Agustus 2021; diterbitkan 31 Agustus 2021

kendaraan khususnya mobil. Ranah *machine learning* pernah dilakukan dalam mendeteksi kendaraan misalnya penelitian yang melakukan deteksi kendaraan mobil dengan menggunakan metode *Region Of Interest* (ROI) (Pratomo et al., 2020) penelitian lain melakukan deteksi kendaraan dengan algoritma yang berbasis deep learning dengan metode *Radial Basis Function* (Pangestuti et al., 2016), serta penelitian lain melakukan deteksi kendaraan bermobil menggunakan *Klasifikasi Neural Network, Support Vector Machine,* Dan Algoritma *C4.5* (Purwaningsih, 2016).

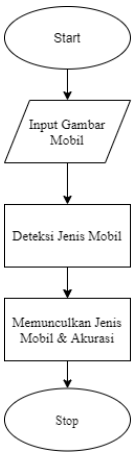
Pada penelitian ini, penulis mengusulkan metode *Extreme Machine learning* (EML) guna membantu melakukan deteksi kendaraan bermobil dengan empat class yaitu mobil truck, sedan, MPV dan bus. Algoritma *extreme learning machine* (ELM) adalah metode pembelajaran Single *Layer* Feedforward Neural Network (SLFNs) yang menyederhanakan proses inisialisasi dan pelatihan berat. *Extreme learning machine* (ELM) berbeda dari metode pembelajaran tradisional dengan memilih secara acak bobot *input* dan selanjutnya secara analitik menentukan bobot output dengan menggunakan solusi linear kuadrat terkecil. Oleh karena itu tidak diperlukan algoritma berdasarkan *gradient descent*, seperti algoritma *backpropagation*.

Algoritma *extreme learning machine* (ELM) memiliki kelebihan yaitu pelatihan sangat cepat, membutuhkan lebih sedikit pengaturan parameter, dan hasilnya menghasilkan kinerja generalisasi yang baik (Najar et al., 2018). Berdasarkan penelitian (Fadilla et al., 2018) yang menghasilkan nilai akurasi sebesar 96,7%. Dapat disimpulkan bahwa metode *Extreme learning machine* (ELM) cukup baik diimplementasikan untuk proses klasifikasi penyakit Chronic Kidney Disease (CKD).

Berdasarkan pemaparan tersebut, pada penelitian ini akan dilakukan klasifikasi jenis kendaraan roda empat meliputi sedan, MVP, bus dan truck dengan *dataset* berjumlah 400 data masing-masing 100 gambar.

2. Metode Penelitian

Pada sistem yang dibangun, skenario yang terjadi apabila pengguna mengakses sistem dan memasukan gambar mobil akan diketahui jenis mobil tersebut serta besaran akurasinya. Adapun prosedur dari sistem dapat dilihat pada gambar 1. berikut:



Gambar 1. Alur Prosedur Sistem Yang Dibangun

Adapun analisis sistem dari model *Extreme learning machine* yang dibangun seperti pada Gambar 2.

Gambar 2. Flowchart *Extreme learning machine* Yang Dibangun

Berikut ini adalah keterangan Flowchart *Extreme learning machine* Yang Dibangun :

1. *Input Dataset*
   Data yang dikumpulkan adalah data gambar yang di dalamnya terdapat satu objek dari jenis mobil sedan, MVP, truk dan bus. Data gambar tersebut adalah gambar yang berwarna dengan ukuran yang bervariasi. Data gambar ini nantinya akan dibagi ke dalam dua kelompok yaitu data *training* dan data *testing*.

2. Praproses Data
   Agar dapat diproses oleh *Extreme learning machine*, maka data citra yang telah dikumpulkan harus melalui praproses terlebih dahulu. Praproses ini terdiri dari proses deteksi tepi, cropping, konversi citra warna ke grayscale, dan penyeragaman ukuran citra. Algoritma deteksi tepi yang digunakan adalah metode Canny. Setelah dilakukan deteksi tepi kemudian masing-masing tepi yang terdeteksi akan diberikan *bounding* box. Luas masing-masing *bounding* box tersebut kemudian dihitung dan dicari *bounding* box dengan luas terbesar untuk dilakukan proses cropping. Selanjutnya citra berwarna yang telah melalui proses cropping akan dikonversi menjadi citra grayscale dan dilakukan penyeragaman ukuran citra yaitu menjadi 50x50 piksel.
3. Pemodelan *Extreme learning machine*
   ELM mempunyai model matematis yang berbeda dari jaringan syaraf tiruan feedforward. Model matematis dari ELM lebih sederhana dan efektif. Dengan 75 jumlah sampel yang berbeda (xi, ti), jumlah hidden nodes sebanyak 1250 dan fungsi aktivasi $g(x)$ maka berdasarkan Persamaan didapatkan model matematis ELM pada penelitian ini sebagai berikut:

$\sum \beta i g i(\mathrm{x}j) = 1250\ i=1 \sum \beta i g(wi\ .\ \mathrm{x}j + bi) = oj$ 1250 i=1 , j = 1, 2, . . . , 75

di mana
- wi = [wi1, wi2, . . . , wi2500] T merupakan vektor bobot yang menghubungkan hidden node ke-i dan *input* nodes.
- βi = [βi1, βi2, βi3] T merupakan vektor bobot yang menghubungkan hidden node ke-i dan output nodes.
- bi merupakan bias dari hidden node ke-i.
- wi . xj merupakan inner product dari wi dan xj.

Berikut ini adalah penjabaran model pada setiap *layer* arsitektur jaringan:

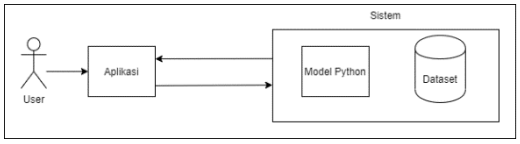
a. *Input Layer*
   Layer ini merupakan *layer* pertama pada jaringan. *Layer* ini terdiri dari 50x50 atau 2.500 node sesuai dengan ukuran citra *input* yang merupakan hasil dari praproses data. Satu piksel pada citra *input* mewakili satu node pada lapisan ini. *Input layer* dengan hidden *layer* dihubungkan oleh vektor bobot w dan bias yang nilainya ditentukan secara acak. Fungsi yang dipilih sebagai fungsi aktivasi pada *layer* ini adalah fungsi softsign yaitu:
   $f(x) = x\ 1 + |x|$
   Fungsi aktivasi ini memberikan batasan keluaran antara (-1,1). Berdasarkan Persamaan tersebut dapat dirumuskan suatu fungsi umum sebagai berikut:
   $Hi,j = g(wi\ .\ \mathrm{x}j + bi) = wi\ .\ \mathrm{x}j + bi\ 1 + |wi\ .\ \mathrm{x}j + bi\ |$
   Keterangan variabel dan indeks pada persamaan di atas adalah sebagai berikut:
   H : matriks output pada hidden *layer*
   w : vektor bobot yang menghubungkan hidden node dan *input* node.
   x : vektor *input*
   b : bias yang terhubung dengan hidden node
   i : indeks jumlah node pada hidden *layer*
   j : indeks jumlah citra *input* w.
   x : inner product dari w dan x

b. Hidden *Layer*
   Layer ini merupakan *layer* kedua dari jaringan dan terdiri dari 1250 buah node. Hidden *layer* dan output *layer* dihubungkan oleh vektor bobot β. Sehingga berdasarkan Persamaan dengan N = 75 dan $\tilde{N}$ = 1250 dapat dituliskan suatu persamaan sebagai berikut:
   $H\beta = T$,

c. Output *Layer*
   Setiap node yang ada di hidden *layer* dihubungkan dengan output *layer* melalui vektor bobot β. Jumlah node pada output *layer* disesuaikan dengan jumlah kelas dari objek yang akan dikenali. Output *layer* pada penelitian ini terdiri dari 4 buah node dan hasil keluaran dari output *layer* ini mewakili kelas dari citra *input*.

3. Hasil dan Pembahasan

Analisis arsitektur sistem merupakan tahapan pengidentifikasian arsitektur pada sistem yang dibangun. Pada pengidentifikasian arsitektur ini diberikan gambaran dari arsitektur sistem yang dibangun. Adapun arsitektur sistem dapat dilihat pada gambar 3. berikut:



Gambar 3. Arsitektur Sistem Yang Dibangun

Berikut ini merupakan alur proses dari arsitektur sistem yang dibangun:

1. *client* mengakses aplikasi yang terinstall pada *handphone* android.
2. sistem melakukan manipulasi data yang dibutuhkan. Data diambil dari model yang dibuat.
3. data yang diambil dari *dataset* kemudian diolah agar sesuai dengan *request* dari *client*.
4. sistem menampilkan data sesuai *request* ke aplikasi user/client.

Teknologi Yang Digunakan

Analisis teknologi bertujuan untuk mengetahui teknologi apa saja yang akan digunakan sesuai dengan kebutuhan dalam sistem yang dibangun. Berikut merupakan teknologi yang digunakan pada sistem yang dibangun antara lain:

a. *Library Keras Python*

Ditulis dengan bahasa pemrograman Python, *Keras* adalah perpustakaan berbasis sumber terbuka yang dirancang untuk menyederhanakan model dari kerangka *Deep Learning*. Keras dapat dijalankan di atas *framework* (kerangka kerja) kecerdasan buatan seperti TensorFlow, Microsoft Cognitive Toolkit, dan Theano (Santoso & Ariyanto, 2018). Pada penelitian ini keras digunakan dalam tahap *training* untuk melatih *machine* mengenali jenis mobil dan tingkat akurasinya pada *dataset*.

Berikut merupakan *code library* keras:

```
from keras.utils import np_utils
from keras.models import Sequential, load_model
from keras.layers import Dense, Dropout, Activation, Conv2D,MaxPooling2D, Flatten
```

b. *Library* OpenCV

OpenCV adalah sebuah *library* (perpustakaan) yang digunakan untuk mengolah gambar dan video hingga kita mampu meng-ekstrak informasi didalamnya. OpenCV dapat berjalan di berbagai bahasa pemograman, seperti C, C++, Java, Python, dan juga support diberbagai platform seperti Windows, Linux, Mac OS, iOS dan Android. Pada penelitian ini openCV digunakan untuk mengolah data mobil dengan format jpg (Sidharta, 2017). Berikut *code* openCV:

```
import cv2
img = cv2.imread('dataset')
cv2.waitKey(0)
#Mendestroy windows
cv2.destroyAllWindows()
```

Proses penyelesaian masalah prediksi dengan *Extreme learning machine* (ELM) diawali dengan normalisasi data, kemudian melakukan proses *training*, lalu proses *testing*, selanjutnya dilakukan denormalisasi data dan perhitungan nilai error dengan menggunakan Mean Square Error (MSE).

Berikut merupakan proses pembahasan pada klasifikasi kendaraan.

3.1. Normalisasi Data

Normalisasi data dilakukan karena rentang nilai *input* tidak sama. *Input* akan diproses ke nilai output yang kecil sehingga data yang digunakan harus disesuaikan agar dapat diproses untuk mendapatkan nilai normalisasi yang kecil (Arifin, 2021). Dalam penelitian ini, data yang digunakan akan disesuaikan dengan cara menormalisasi data. *Dataset* yang digunakan pada penelitian ini yaitu image, maka proses normalisasi data adalah proses membagi gambar menjadi beberapa pixel dan menghapus bagian pixel yang tidak penting

Hal melakukan proses normalisasi data pada penelitian ini menggunakan *code* python. Berikut *code* python untuk melakukan proses normalisasi data.

```
import cv2
img = cv2.imread('mobil.jpg')
h,w = img.shape[:2]
new_h, new_w = int(h/2),int(w/2)
resizeImg = cv2.resize(img, (new_w,new_h))
 cv2.imshow('Original', img)
cv2.imshow('Resizing', resizeImg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

3.2 Proses *Training*

Proses *training* harus terlebih dahulu dilakukan sebelum proses prediksi.

Input dataset dilakukan dengan cara memanggil folder *dataset* melalui command

prompt. *Dataset* yang digunakan dijelaskan pada Tabel 1.

Tabel 1. Jumlah *Dataset*

No	Nama Label	Jumlah
1	Sedan	100
2	MVP	100
3	Bus	100
4	Truck	100
Jumlah *Dataset*		400

Tabel 1. menerangkan jumlah *dataset* sebanyak 400 gambar, pada proses *training* dan *testing*, *dataset* dibagi menjadi 80% untuk *training* dan 20% untuk *testing*. Model ELM di*training* sebanyak 4999 epoch.

Proses *training* bertujuan untuk mendapatkan nilai output weight. Langkah-langkah yang dilakukan dalam proses *training* adalah

Langkah pertama adalah menginisialisasi *input* weight dan bias. Nilai ini diinisialisasi secara acak dengan rentang nilai antara -1 hingga 1. 2.

Berikut merupakan *code* python pada proses pertama.

```
output_weights = np.dot(pinv2(hidden_nodes(X_train)), y_train)
```

Langkah selanjutnya adalah menghitung keluaran hidden layer (Hinit). Persamaan 2 berikut untuk menghitung keluaran di hidden layer.

$Hinit\ train = Xtrain.WT + b$

Berikut merupakan code python yang digunakan pada penelitian ini untuk menghitung keluaran hidden layers.

```
input_size = X_train.shape[1]
hidden_size = 1000
input_weights = np.random.normal(size=[input_size,hidden_size]) biases = np.random.normal(size=[hidden_size])
```

Langkah ketiga adalah memilih fungsi actiovation. Fungsi activation tedapat tiga fungsi, yaitu softmax, sigmoid dan relu. Penelitian ini menggunakan fungsi softmax. Berikut merupakan *code* python pada penelitian ini.

```
def relu(x): return np.maximum(x, 0, x)
def hidden_nodes(X): G = np.dot(X, input_weights) G = G + biases H = relu(G) return H
```

Menampilkan hasil akurasi metode *extreme learning machine.*

Saat model telah diimpelemntasikan, perlu dimunculkan besar akurasi yang didapat. Hal ini untuk membuktikan bahwa model *Extreme learning machine* baik untuk melakukan klasifikasi mobil.

Berikut merupakan coding untuk memunculkan akurasi.

```
test_accuracy, predictions = sess.run(
    [evaluation_step, prediction],
    feed_dict={bottleneck_input:
test_bottlenecks,
        ground_truth_input:
test_ground_truth})
  tf.compat.v1.logging.info("Hasil akurasi sebesar = %.1f%% (N=%d)' %
        (test_accuracy * 100, len(test_bottlenecks)))
```

Untuk menjalankan coding train akurasi, tuliskan *code* berikut pada terminal atau command prompt.

```
python scripts/retrain.py --output_graph=tf_files/retrained_graph.pb --output_labels=tf_files/retrained_labels.txt --image_dir=tf_files/flower_photos
```

akurasi pada proses *training* sebesar 92%

3.3 Menyimpan model

Model perlu disimpan agar proses *testing* atau prediksi dapat sesuai dengan model yang dibuat pada saat *training*. Berikut merupakan *code* python untuk menyimpan model.

```
save_graph_to_file(sess, graph, FLAGS.output_graph) with gfile.FastGFile(FLAGS.output_labels, 'w') as f: f.write('\n'.join(image_lists.keys()) + '\n')
```

4. Proses *Testing*

Proses *testing* harus atau prediksi dilakukan untuk menguji model yang telah kita buat. Berikut merupakan langkah saat melakukan proses *testing*.

a. *Input* Gambar Mobil

Gambar mobil yang akan di*input* disimpan didalam satu folder dengan coding. Berikut merupakan *code* pyhton untuk memasukan gambar :

```
def evaluate_graph(graph_file_name):
    with
load_graph(graph_file_name).as_default()
as graph:
        ground_truth_input = tf.placeholder(
            tf.float32,          [None,          5],
name='GroundTruthInput')

        image_buffer_input                    =
graph.get_tensor_by_name('input:0')
        final_tensor                          =
graph.get_tensor_by_name('final_result:0'
)
        accuracy,                          _  =
retrain.add_evaluation_step(final_tensor,
ground_truth_input)

        logits                                =
graph.get_tensor_by_name("final_training
_ops/Wx_plus_b/add:0")
        xent                                  =
tf.reduce_mean(tf.nn.softmax_cross_entr
opy_with_logits(
            labels = ground_truth_input,
            logits = logits))

    image_dir = 'tf_files/f=gambarsedan
    testing_percentage = 10
    validation_percentage = 10
    validation_batch_size = 100
    category='testing'

    image_lists                               =
retrain.create_image_lists(
        image_dir, testing_percentage,
        validation_percentage)
    class_count = len(image_lists.keys())

    ground_truths = []
    filenames = []
Memanggil model yang telah dibuat
    for label_index, label_name in
enumerate(image_lists.keys()):
        for image_index, image_name in
enumerate(image_lists[label_name][categ
ory]):
        image_name                            =
retrain.get_image_path(
            image_lists, label_name,
image_index, image_dir, category)
        ground_truth = np.zeros([1,
class_count], dtype=np.float32)
        ground_truth[0, label_index] = 1.0
        ground_truths.append(ground_truth)
        filenames.append(image_name)
    accuracies = []
    xents = []
    with tf.Session(graph=graph) as sess:
        for filename, ground_truth in
zip(filenames, ground_truths):
            image                             =
Image.open(filename).resize((224,224),Im
age.ANTIALIAS)
            image = np.array(image,
dtype=np.float32)[None,...]
            image = (image-128)/128.0

            feed_dict={
                image_buffer_input: image,
                ground_truth_input:
ground_truth}

            eval_accuracy,    eval_xent    =
sess.run([accuracy, xent], feed_dict)


accuracies.append(eval_accuracy)
            xents.append(eval_xent)
    return                 np.mean(accuracies),
np.mean(xents)
```

Mendeteksi jenis mobil

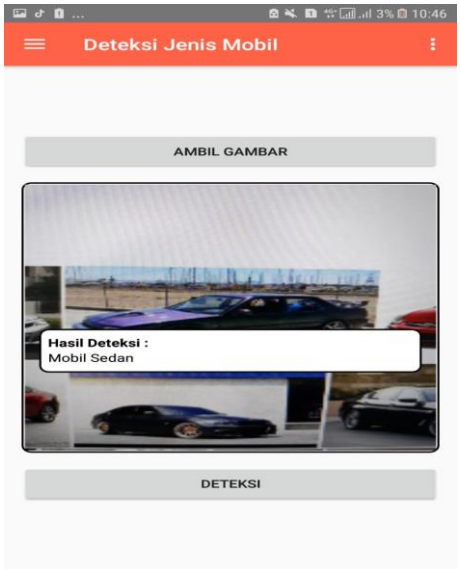
Setelah memanggil model, maka proses selanjutnya adalah deteksi mobil. Berikut merupakan *code* untuk mendeteksi jenis mobil

```
if __name__ == "__main__":

os.environ['TF_CPP_MIN_LOG_LEVEL']
= '2'
    accuracy,xent                             =
evaluate_graph(*sys.argv[1:])
    print('Accuracy: %g' % accuracy)
    print('Cross Entropy: %g' % xent)
```

3.5. Hasil Penelitian

Hasil penelitian ini adalah aplikasi deteksi jenis mobil menggunakan metode *Extreme learning machine* dengan *user interface* pada Gambar 4.



Gambar 4. User *Inteface* Deteksi Jenis Mobil

Dimana terlihat hasil deteksi dari gambar adalah mobil sedan. Hal ini tentu sesuai dengan gambar yang telah diinput.

4. Kesimpulan

Kesimpulan dari penelitian ini yaitu terbukti bahwa metode *Extreme learning machine (ELM)* menghasilkan akurasi yang baik dalam mengklasifikasikan jenis kendaraan dengan total akurasi 92% dengan melalui tahap 3999 epoch dalam proses *training*.

Referensi

Arifin, M. H. R. (2021). Deep Learninng. In *Institut teknologi nasional*.

Fadilla, I., Adikara, P. P., & Perdana, R. S. (2018). Klasifikasi Penyakit Chronic Kidney Disease (CKD) Dengan Metode *Extreme learning machine* (ELM). *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *2*(10), 3397–3405. https://www.researchgate.net/profile/Rizal_Perdana/publication/323365845_Klasifikasi_Penyakit_Chronic_Kidney_Disease_CKD_Dengan_Menggunakan_Metode_Extreme_Learning_Machine_ELM/links/5a9023c5aca2721405618881/Klasifikasi-Penyakit-Chronic-Kidney-Disease-CKD-

Najar, A. M., Magister, P., Matematika, D., Matematika, F., Dan, K., & Data, S. (2018). *Penerapan Metode Extreme learning machine Untuk Prediksi Tingkat Berdasarkan Keadaan Cuaca (Studi Kasus : Wilayah Dki Jakarta)*.

Pangestuti, G. W., Usman, K., & Purnama, B. (2016). Klasifikasi Kendaraan Roda Empat Dengan Ekstraksi Ciri Hybrid Berbasis Jaringan Syaraf Tiruan. *EProceedings of Engineering*, *3*(Fakultas Teknik Elektro, Universitas Telkom), 1619–1627.

Pratomo, A. H., Kaswidjanti, W., & Mu'arifah, S. (2020). Implementasi Algoritma Region of Interest (ROI) Untuk Meningkatkan Performa Algoritma Deteksi Dan Klasifikasi Kendaraan. *Jurnal Teknologi Informasi Dan Ilmu Komputer (JTIIK)*, *7*(1), 155–162. https://doi.org/10.25126/jtiik.202071718

Priyambodo, P. (2018). Analisis Korelasi Jumlah Kendaraan dan Pengaruhnya Terhadap PDRB di Provinsi Jawa Timur. *Warta Penelitian Perhubungan*, *30*(1), 59. https://doi.org/10.25104/warlit.v30i1.634

Purwaningsih, E. (2016). Seleksi Mobil Berdasarkan Fitur Dengan Komparasi Metode Klasifikasi Neural Network, Support Vector Machine, Dan Algoritma C4.5. *Jurnal Pilar Nusa Mandiri*, *XII*(2), 153–160. https://ejournal.nusamandiri.ac.id/index.php/pilar/article/view/269

Roihan, A., Sunarya, P. A., & Rafika, A. S. (2020). Pemanfaatan *Machine learning* dalam Berbagai Bidang: Review paper. *IJCIT (Indonesian Journal on Computer and Information Technology)*, *5*(1), 75–82. https://doi.org/10.31294/ijcit.v5i1.7951

Santoso, A., & Ariyanto, G. (2018). Implementasi Deep Learning Berbasis Keras Untuk Pengenalan Wajah. *Emitor: Jurnal Teknik Elektro*, *18*(01), 15–21. https://doi.org/10.23917/emitor.v18i01.6235

Sidharta, H. A. (2017). *INTRODUCTION TO OPEN CV*. Binus University. https://binus.ac.id/malang/2017/10/introduction-to-open-cv/