

**Laporan Tugas 4 IF4073 Interpretasi dan Pengolahan
Citra
Semester I Tahun 2023/2024**



Disusun Oleh:

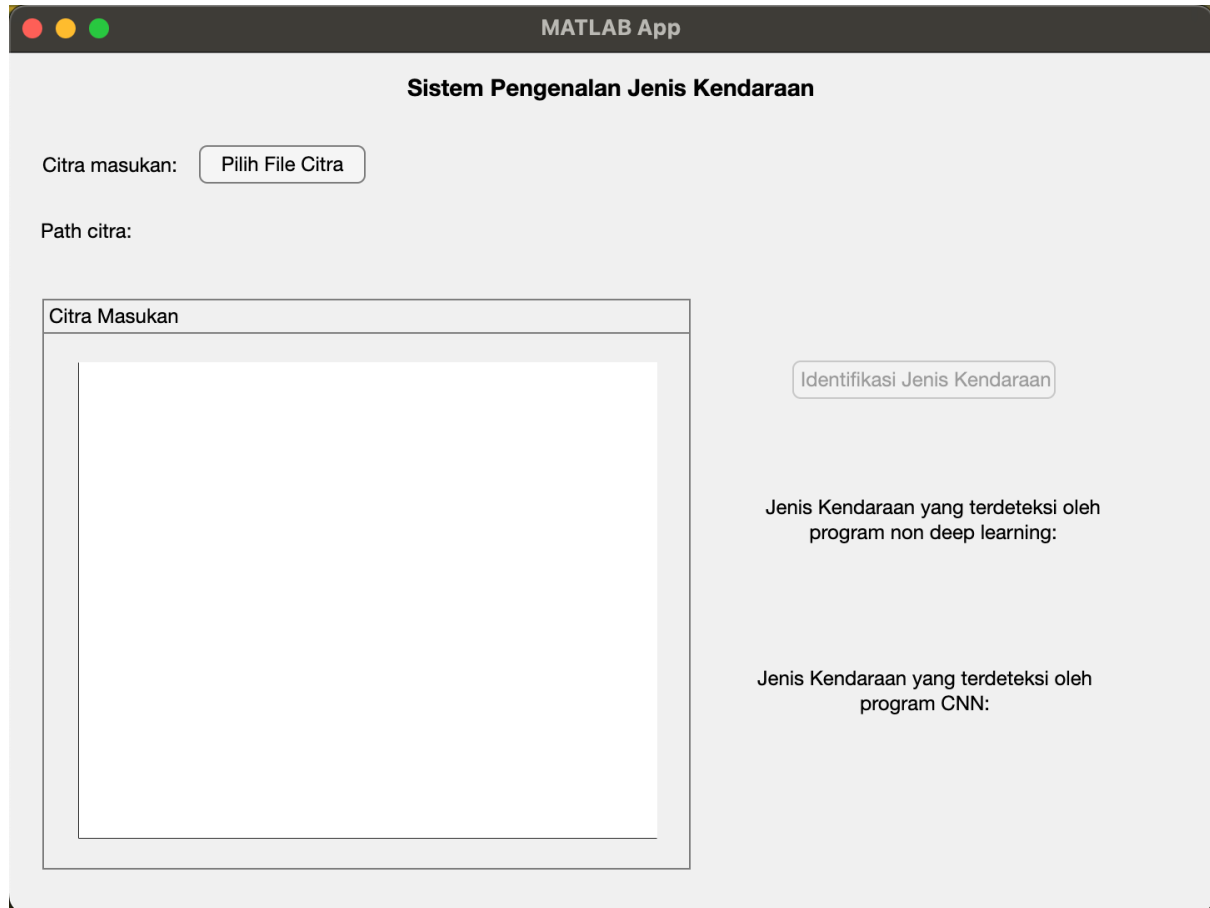
Adzka Ahmadya Zaidan	13520127
Fachry Dennis Herald	13520139
Zayd Muhammad Kawakibi Zuhri	13520144

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023**

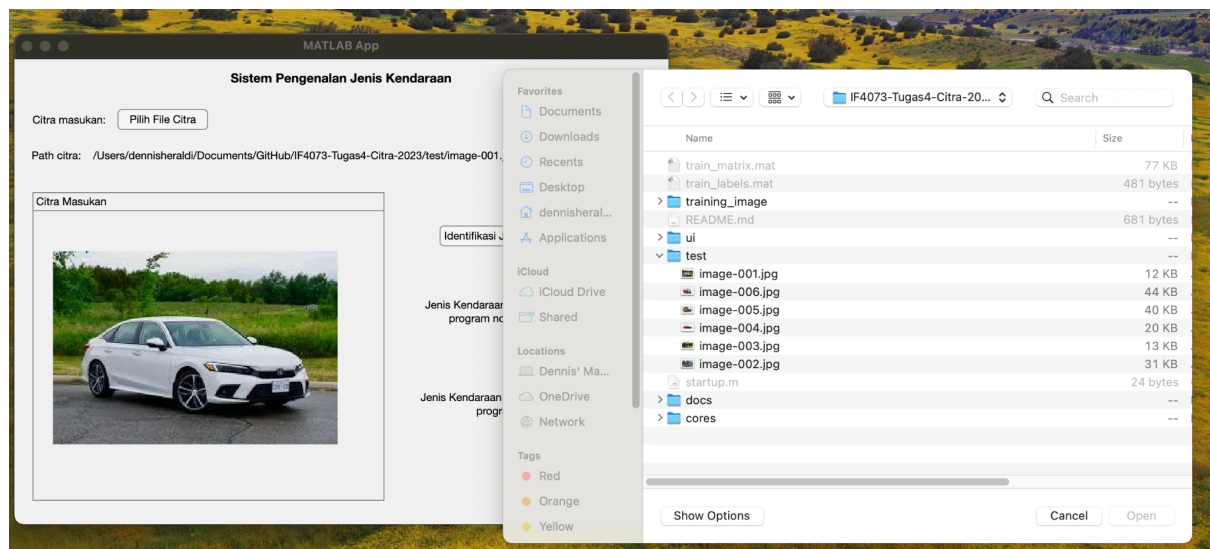
Screenshot GUI Program

Screenshot dari GUI program terlihat sebagai berikut.

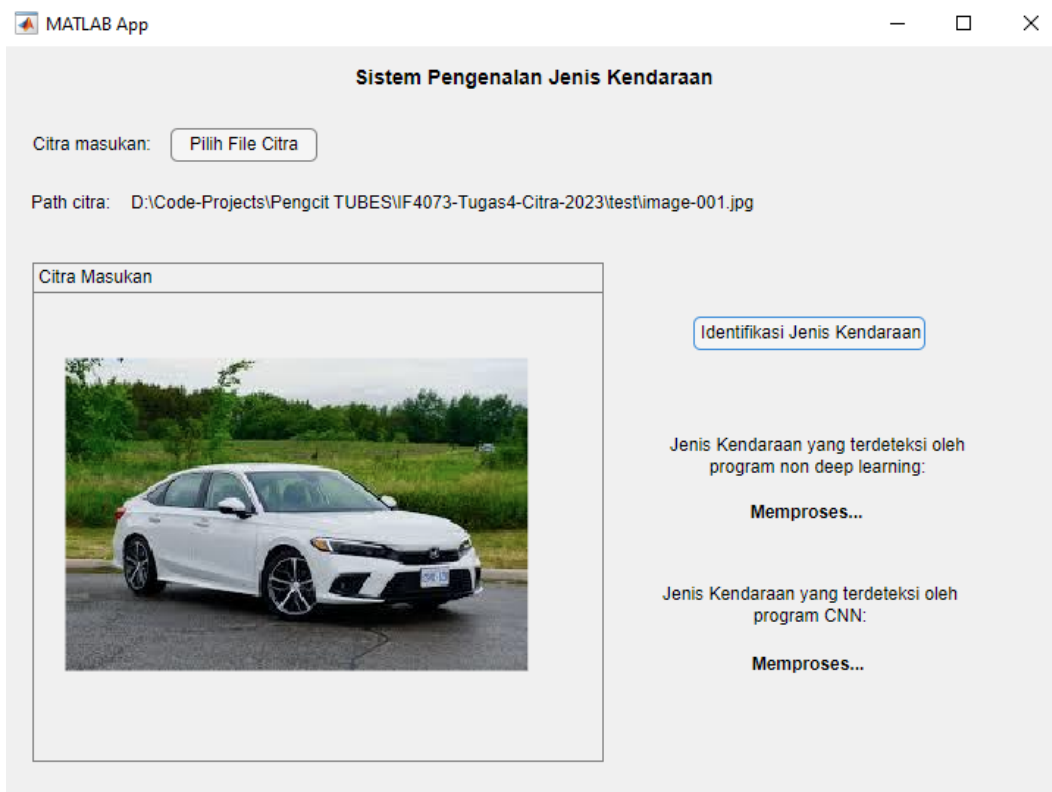
1. Tampilan awal program



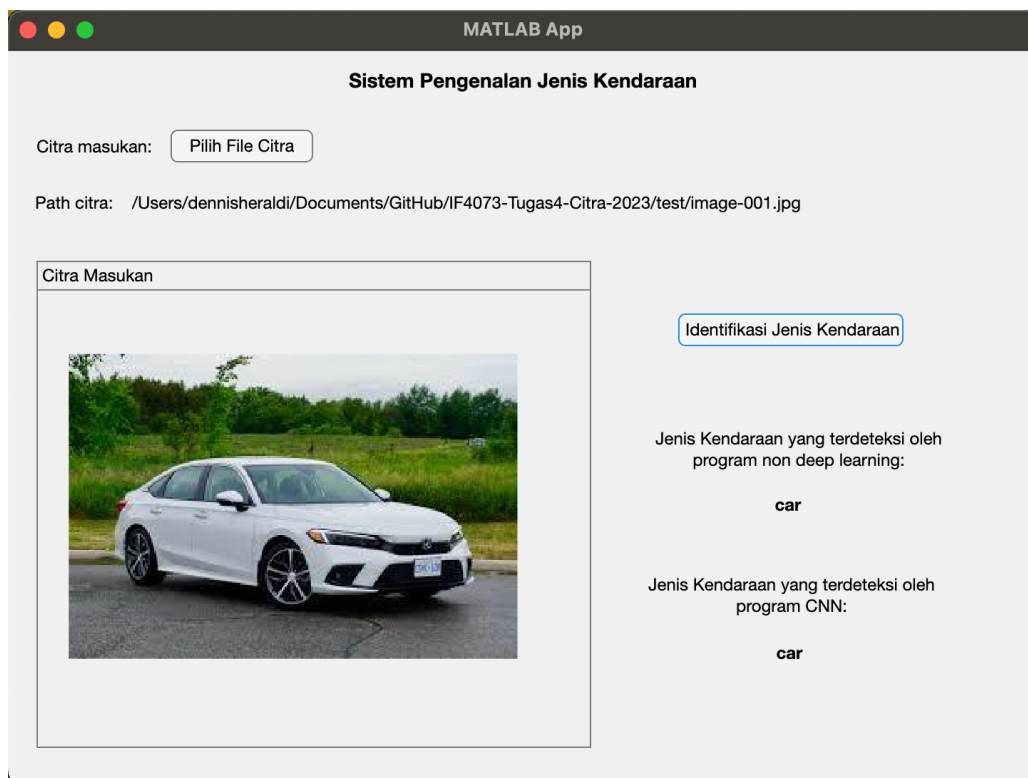
2. Tampilan program ketika memilih file citra masukan



3. Tampilan program saat memproses citra

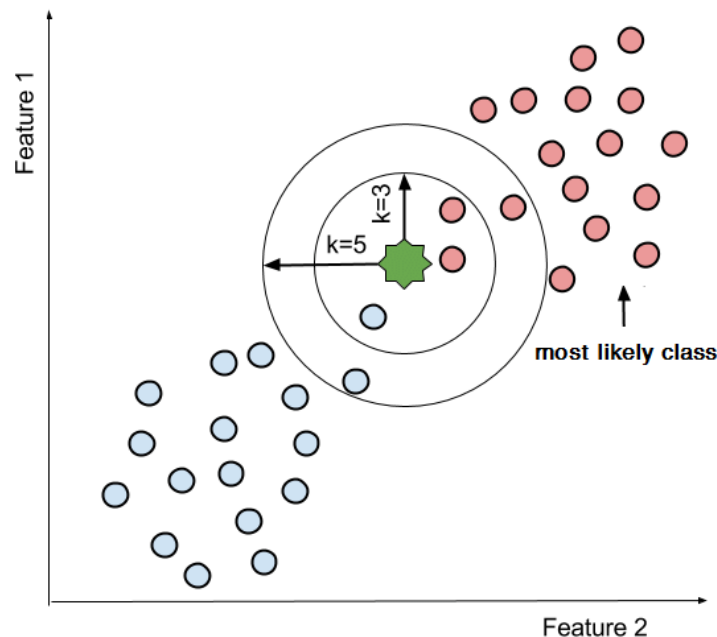


4. Tampilan program setelah proses identifikasi citra



Kode Program dan Analisis Cara Kerja Fungsi Program

a. KNN



K-nearest neighbors (KNN) adalah sebuah algoritma pembelajaran mesin yang dapat digunakan untuk mengklasifikasikan gambar kendaraan berdasarkan kemiripan dengan gambar-gambar lain yang sudah diketahui kelasnya. Algoritma ini bekerja dengan cara menghitung jarak antara gambar yang ingin diklasifikasikan dengan k gambar terdekat yang ada di data latih, lalu memilih kelas yang paling sering muncul di antara k gambar tersebut. Untuk dapat menghitung jarak antara gambar-gambar, kita memerlukan vektor fitur yang merepresentasikan ciri-ciri penting dari gambar tersebut. Vektor fitur adalah sebuah vektor numerik yang menggambarkan karakteristik suatu objek, seperti warna, bentuk, tekstur, dan sebagainya. KNN dipilih karena mudah dikembangkan dan dapat digunakan dalam MATLAB dengan menggunakan fungsi `fitcknn` untuk membuat model klasifikasi KNN dari data latih. Berikut kode MATLAB untuk pembuatan fitur, pembentukan matrix terlatih, dan inferensi model KNN untuk klasifikasi gambar:

```
function label = knn(img)
% First, we need a function to extract features (a vector representation) from
the image
```

```

function features = getFeatures(img)
    if size(img, 3) ~= 3
        img = repmat(im2uint8(img), [1, 1, 3]); % Duplicate grayscale into
3 channels
    end
    % convert to grayscale
    img_gray = rgb2gray(img);
    % resize to 64x64
    img_gray_64 = imresize(img_gray, [64, 64]);
    % convert to feature vector
    g = graycoprops(graycomatrix(img_gray_64));
    features = [g.Contrast, g.Correlation, g.Energy, g.Homogeneity];
    % add other features
    % firstly, a histogram of the image (colored)
    img_128 = imresize(img, [128, 128]);
    img_128 = double(img_128);
    % divide into 4x4 blocks, make sure to split channels
    img_128 = mat2cell(img_128, [64 64], [64 64], 3);
    % calculate histogram for each block
    hist = [];
    for i = 1:2 %#ok<*FXUP>
        for j = 1:2
            % reduce histogram to 4 bins
            binned_hist = histcounts(img_128{i, j}, 4);
            hist = [hist, binned_hist'];
        end
    end
    % flatten everything
    hist = hist(:)';
    % add to features
    features = [features, hist];
    % next feature are hough lines
    img_edge = edge(img_gray, 'canny');
    [H, T, R] = hough(img_edge);
    P = houghpeaks(H, 5);
    lines = houghlines(img_edge, T, R, P);
    % get the angles
    angles = [];
    rhos = [];
    for i = 1:length(lines)
        angles = [angles, lines(i).theta];
        rhos = [rhos, lines(i).rho];
    end
    % get the mean and std
    mean_angle = mean(angles);
    std_angle = std(angles);
    mean_rho = mean(rhos);
    std_rho = std(rhos);
    % add to features
    features = [features, mean_angle, std_angle, mean_rho, std_rho,
length(lines)];
    % normalize features
    features = features ./ max(features);
end
% Train KNN if trained matrix files don't exist
% we read from the train folder, each image will be extracted to a feature
vector
% and then we will store all the feature vectors in a matrix

```

```

% also store the labels in a vector, labels are from the file names
% inference code is at the bottom
dataset_path = 'training_image/';
if ~exist('train_matrix.mat', 'file') || ~exist('train_labels.mat', 'file')
    train_matrix = [];
    train_labels = {};
    % need to read all files and not just follow the naming
    % first, list all file names
    file_names = dir(dataset_path);
    % iterate over all files
    for i = 1:length(file_names)
        disp('Training KNN... (' + string(i) + '/' + string(length(file_names))
+ ')')
        % get the file name
        file_name = file_names(i).name;
        % check if it is a valid file
        if length(file_name) > 4 && (strcmp(file_name(end-3:end), '.jpg') ||
strcmp(file_name(end-3:end), '.png') || strcmp(file_name(end-4:end), '.jpeg'))
&& ~contains(file_name, 'Ambulance')
            % read the image
            img = imread(strcat(dataset_path, file_name));
            % get the features
            feats = getFeatures(img);
            % append to train matrix
            train_matrix = [train_matrix; feats];
            % read the label
            % label can be [car, bus, truck]
            % its taken from file name, but can be in various formats
            if contains(file_name, 'car')
                label = 'car';
            elseif contains(file_name, 'BUS')
                label = 'bus';
            elseif contains(file_name, 'Truck')
                label = 'truck';
            else
                label = '';
                disp(file_name);
            end
            % append to train labels
            train_labels = [train_labels; {label}];
            % close files
            close all;
        end
        clc; % clear console
    end

    save('train_matrix.mat', 'train_matrix');
    save('train_labels.mat', 'train_labels');
end

% load the train matrix and labels
train_matrix = load('train_matrix.mat');
train_labels = load('train_labels.mat');
% inference code
model = fitcknn(train_matrix.train_matrix, train_labels.train_labels);
% get features from image
feats = getFeatures(img);
% predict

```

```
label = predict(model, feats);  
end
```

Kode di atas adalah sebuah fungsi yang dapat digunakan untuk mengklasifikasikan gambar kendaraan menggunakan KNN. Berikut adalah penjelasan kode tersebut dalam konteks pembuatan model KNN:

- Pertama, kita perlu membuat sebuah fungsi bantuan untuk mengekstraksi fitur dari gambar. Fitur adalah sebuah vektor numerik yang merepresentasikan ciri-ciri penting dari gambar, seperti warna, bentuk, tekstur, dan sebagainya. Fitur ini digunakan untuk mengukur kemiripan antara gambar-gambar yang ingin diklasifikasikan.
- Fungsi bantuan tersebut bernama `getFeatures` dan menerima sebuah gambar sebagai input. Fungsi ini melakukan beberapa langkah untuk mendapatkan fitur dari gambar, yaitu:
 - Jika gambar tidak berwarna, maka gambar tersebut diubah menjadi gambar berwarna dengan menggandakan saluran abu-abu menjadi tiga saluran warna.
 - Gambar diubah menjadi gambar abu-abu dengan menggunakan fungsi `rgb2gray`.
 - Gambar diubah ukurannya menjadi 64×64 piksel dengan menggunakan fungsi `imresize`.
 - Gambar diubah menjadi vektor fitur dengan menggunakan fungsi `graycoprops` dan `graycomatrix`. Fungsi ini menghitung empat nilai statistik dari matriks korelasi abu-abu, yaitu kontras, korelasi, energi, dan homogenitas. Nilai-nilai ini menggambarkan tekstur gambar.
 - Selain empat nilai tersebut, fungsi ini juga menambahkan fitur lain, yaitu:
 - Histogram dari gambar berwarna. Histogram adalah sebuah vektor yang menghitung frekuensi kemunculan nilai intensitas warna dalam gambar. Fungsi ini menggunakan

fungsi `histcounts` untuk menghitung histogram dari setiap blok 64×64 piksel dalam gambar berwarna yang telah diubah ukurannya menjadi 128×128 piksel. Fungsi ini juga mengurangi jumlah bin histogram menjadi empat untuk menghemat ruang vektor. Histogram menggambarkan distribusi warna dalam gambar.

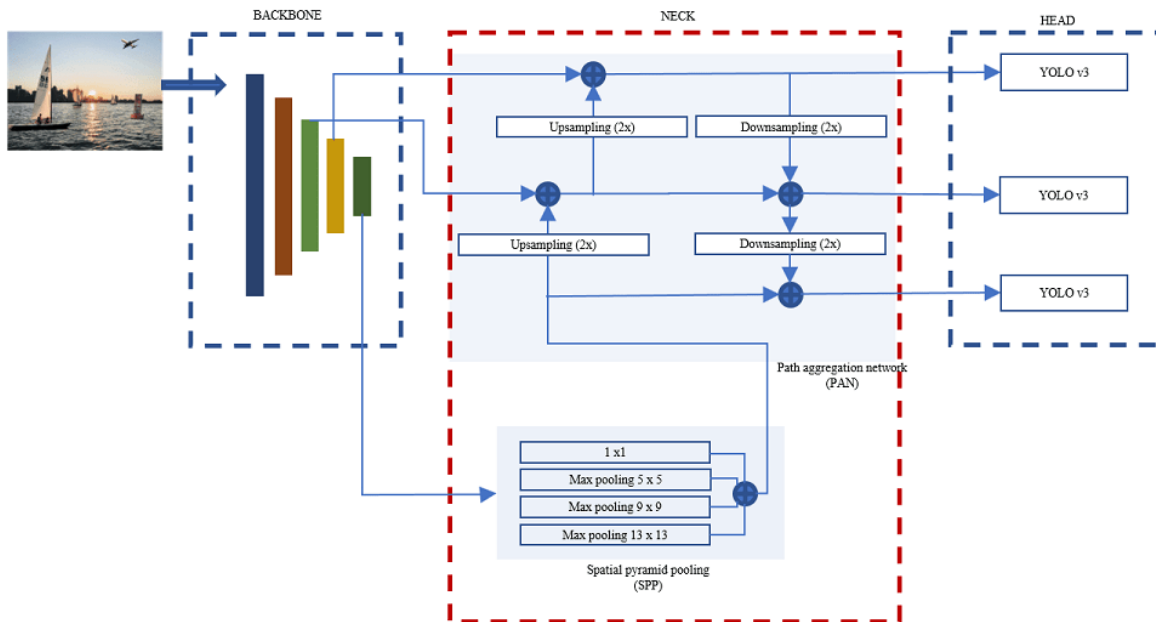
- Garis Hough dari gambar abu-abu. Garis Hough adalah sebuah metode untuk mendeteksi garis lurus dalam gambar. Fungsi ini menggunakan fungsi `edge`, `hough`, `houghpeaks`, dan `houghlines` untuk mendapatkan lima garis Hough terbaik dari gambar abu-abu yang telah diubah menjadi gambar tepi dengan menggunakan metode Canny. Fungsi ini juga menghitung rata-rata dan standar deviasi dari semua sudut dan jarak garis Hough, serta jumlah garis Hough yang dideteksi. Nilai-nilai ini menggambarkan bentuk gambar.
 - Fungsi ini mengembalikan vektor fitur yang terdiri dari 25 elemen, yaitu 4 nilai statistik, 16 nilai histogram, dan 5 nilai garis Hough. Fungsi ini juga menormalisasi vektor fitur dengan membaginya dengan nilai maksimumnya.
- Kedua, kita perlu melatih model KNN dari data latih yang sudah diberi label kelas. Data latih adalah kumpulan gambar kendaraan yang sudah diketahui jenisnya, seperti mobil, bus, atau truk. Label kelas adalah sebuah vektor yang berisi nama jenis kendaraan untuk setiap gambar.
- Fungsi utama tersebut bernama `knn` dan menerima sebuah gambar sebagai input. Fungsi ini melakukan beberapa langkah untuk melatih model KNN, yaitu:
 - Jika file `train_matrix.mat` atau `train_labels.mat` tidak ada, maka fungsi ini akan membuatnya. File-file ini berisi matriks fitur dan vektor label kelas dari data latih.
 - Fungsi ini membaca semua gambar yang ada di folder `training_image/` dan memeriksa apakah gambar tersebut valid.

Gambar yang valid adalah gambar yang memiliki ekstensi `.jpg`, `.png`, atau `.jpeg` dan tidak mengandung kata `Ambulance`.

- Fungsi ini mengubah setiap gambar menjadi vektor fitur dengan menggunakan fungsi bantuan `getFeatures` dan menambahkannya ke matriks fitur. Fungsi ini juga membaca label kelas dari nama file gambar dan menambahkannya ke vektor label kelas. Label kelas dapat berupa `car`, `bus`, atau `truck`.
- Fungsi ini menyimpan matriks fitur dan vektor label kelas ke file `train_matrix.mat` dan `train_labels.mat` dengan menggunakan fungsi `save`.
- Terakhir, untuk kode inferensinya sendiri, jika kita sudah memiliki `train_matrix.mat` dan `train_labels.mat` maka kedua matriks akan dibaca dari file dan dimasukkan ke dalam model KNN dari MATLAB dengan `fitknn`. Lalu, gambar yang menjadi input dari fungsi `knn(img)` ini, diubah menjadi vektor dengan `getFeatures` dan diprediksi labelnya oleh model menggunakan `predict`. Output dari fungsi ini adalah string label prediksi gambar tersebut.

b. CNN

Model CNN yang digunakan adalah model *pretrained* berbasis arsitektur YOLOv4 yang dikembangkan oleh MATLAB sehingga dapat digunakan langsung dengan add-ons Deep Learning Toolbox dan Computer Vision Toolbox. Model dapat langsung digunakan tanpa perlu ada pelatihan ulang. Arsitektur dari YOLOv4 adalah sebagai berikut.



YOLOv4 adalah *one-stage object detection network* yang terdiri atas tiga bagian: *backbone*, *neck*, dan *head*.

- Bagian *backbone* dapat berupa *pretrained* CNN seperti VGG16 atau CSPDarkNet53 yang dilatih dengan dataset COCO atau ImageNet. Bagian ini bertindak sebagai jaringan yang mengekstraksi fitur sehingga didapatkan *feature maps* dari gambar input
- Bagian *neck* menghubungkan *backbone* dengan *head*. Bagian *neck* berisi modul spatial pyramid pooling (SPP) dan path aggregation network (PAN). Bagian ini menggabungkan *feature maps* dari layer yang berbeda dan hasil penggabungannya dikirim ke bagian *head*.
- Bagian *head* mengagregasi fitur dan memprediksi *bounding boxes*, *objectness scores*, dan *classification scores*. Bagian *head* dari YOLOv4 berisi one-stage object detectors YOLOv3.

Berikut adalah kode fungsi yang mengimplementasikan YOLOv4.

```
function label = cnn(img)
    addpath cores/pretrained-yolo-v4/
    addpath cores/pretrained-yolo-v4/src/
    addpath cores/pretrained-yolo-v4/models/
    modelName = 'YOLOv4-coco';
    model = helper.downloadPretrainedYOLOv4(modelName);
    net = model.net;

    % Read test image.
```

```

image = img;

% Get classnames of COCO dataset.
classNames = helper.getCOCOClassNames;

% Get anchors used in training of the pretrained model.
anchors = helper.getAnchors(modelName);

% Detect objects in test image.
executionEnvironment = 'auto';
[~, scores, labels] = detectYOL0v4(net, image, anchors, classNames,
executionEnvironment);

label_score = [string(labels) string(scores)];

% Kelas valid
valid_classes = ["bus", "car", "truck"];

% Filter untuk hanya kelas yang valid
valid_scores = [];
for i = 1:size(label_score, 1)
    if any(valid_classes == label_score{i, 1})
        valid_scores = [valid_scores; label_score(i, :)];
    end
end

% Inisialisasi variabel untuk menyimpan kelas dengan skor tertinggi
max_score = -1;
max_class = "car"; % Default

% Cari kelas dengan skor tertinggi
for i = 1:size(valid_scores, 1)
    current_score = str2double(valid_scores{i, 2});
    if current_score > max_score
        max_score = current_score;
        max_class = valid_scores{i, 1};
    end
end

% Output kelas dengan skor tertinggi atau default
label = max_class;
end

function [bboxes, scores, labels] = detectYOL0v4(dlnet, image, anchors,
classNames, executionEnvironment)
% detectYOL0v4 runs prediction on a trained yolov4 network.
%
% Inputs:
% dlnet          - Pretrained yolov4 dlnetwork.
% image          - RGB image to run prediction on. (H x W x 3)
% anchors        - Anchors used in training of the pretrained model.
% classNames      - Classnames to be used in detection.
% executionEnvironment - Environment to run predictions on. Specify cpu,
%                  gpu, or auto.
%
% Outputs:
% bboxes         - Final bounding box detections ([x y w h]) formatted as
%                  NumDetections x 4.

```

```

% scores      - NumDetections x 1 classification scores.
% labels      - NumDetections x 1 categorical class labels.
% Copyright 2021 The MathWorks, Inc.
% Get the input size of the network.
inputSize = dlnet.Layers(1).InputSize;
% Apply Preprocessing on the input image.
[img, scale] = helper.preprocess(image, inputSize);
% Convert to dlmarray.
dlInput = dlmarray(img, 'SSCB');
% If GPU is available, then convert data to gpuArray.
if (executionEnvironment == "auto" && canUseGPU) || executionEnvironment ==
    "gpu"
    dlInput = gpuArray(dlInput);
end
% Perform prediction on the input image.
outFeatureMaps = cell(length(dlnet.OutputNames), 1);
[outFeatureMaps{:}] = predict(dlnet, dlInput);
% Apply postprocessing on the output feature maps.
[bboxes,scores,labels] = helper.postprocess(outFeatureMaps, anchors, ...
    inputSize, scale, classNames);
end

```

Kode di atas memperlihatkan penggunaan model YOLOv4 untuk deteksi objek dalam gambar. Mula-mula Model YOLOv4, yang telah dilatih pada dataset COCO, diunduh dan dimuat. Gambar input, diberikan sebagai argumen fungsi, kemudian diolah menggunakan model ini. Proses ini melibatkan pengambilan nama-nama kelas dan *anchor boxes* yang digunakan selama pelatihan model. Setelah itu, fungsi `detectYOLOv4` dipanggil untuk mendeteksi objek dalam gambar. Fungsi ini mengubah ukuran gambar agar sesuai dengan input model, melakukan prediksi menggunakan YOLOv4, dan mengaplikasikan pasca-pemrosesan untuk mendapatkan bounding boxes, skor, dan label kelas. Bounding boxes tidak dipergunakan karena fungsi di atas hanya mengembalikan label kelas yang relevan dengan skor tertinggi. Kembali ke fungsi utama, hasil deteksi difilter untuk hanya memasukkan beberapa kelas tertentu (seperti bus, mobil, dan truk) dan kemudian memilih kelas dengan skor tertinggi dari kelas-kelas yang valid. Terakhir, fungsi mengembalikan label kelas dengan skor tertinggi sebagai output.

Contoh Hasil Eksekusi


MATLAB App

Sistem Pengenalan Jenis Kendaraan

Citra masukan:

Path citra: D:\Code-Projects\Pengcit TUBES\IF4073-Tugas4-Citra-2023\test\image-001.jpg

Citra Masukan



Jenis Kendaraan yang terdeteksi oleh program non deep learning:

car

Jenis Kendaraan yang terdeteksi oleh program CNN:

car


MATLAB App

Sistem Pengenalan Jenis Kendaraan

Citra masukan:

Path citra: D:\Code-Projects\Pengcit TUBES\IF4073-Tugas4-Citra-2023\test\image-002.jpg

Citra Masukan



Jenis Kendaraan yang terdeteksi oleh program non deep learning:

bus

Jenis Kendaraan yang terdeteksi oleh program CNN:

bus

Sistem Pengenalan Jenis KendaraanCitra masukan:

Path citra: D:\Code-Projects\Pengcit TUBES\IF4073-Tugas4-Citra-2023\test\image-003.jpg

Citra Masukan

Jenis Kendaraan yang terdeteksi oleh
program non deep learning:**bus**Jenis Kendaraan yang terdeteksi oleh
program CNN:**bus****Sistem Pengenalan Jenis Kendaraan**Citra masukan:

Path citra: D:\Code-Projects\Pengcit TUBES\IF4073-Tugas4-Citra-2023\test\image-004.jpg

Citra Masukan

Jenis Kendaraan yang terdeteksi oleh
program non deep learning:**bus**Jenis Kendaraan yang terdeteksi oleh
program CNN:**car**

Sistem Pengenalan Jenis KendaraanCitra masukan:

Path citra: D:\Code-Projects\Pengcit TUBES\IF4073-Tugas4-Citra-2023\test\image-005.jpg

Citra Masukan

Jenis Kendaraan yang terdeteksi oleh
program non deep learning:**truck**Jenis Kendaraan yang terdeteksi oleh
program CNN:**truck****Sistem Pengenalan Jenis Kendaraan**Citra masukan:

Path citra: D:\Code-Projects\Pengcit TUBES\IF4073-Tugas4-Citra-2023\test\image-006.jpg

Citra Masukan

Jenis Kendaraan yang terdeteksi oleh
program non deep learning:**truck**Jenis Kendaraan yang terdeteksi oleh
program CNN:**truck**

Sistem Pengenalan Jenis KendaraanCitra masukan:

Path citra: D:\Code-Projects\Pengcit TUBES\IF4073-Tugas4-Citra-2023\test\image-007.jpg

Citra Masukan

Jenis Kendaraan yang terdeteksi oleh
program non deep learning:

car

Jenis Kendaraan yang terdeteksi oleh
program CNN:

car

Sistem Pengenalan Jenis KendaraanCitra masukan:

Path citra: D:\Code-Projects\Pengcit TUBES\IF4073-Tugas4-Citra-2023\test\image-008.jpg

Citra Masukan

Jenis Kendaraan yang terdeteksi oleh
program non deep learning:

car

Jenis Kendaraan yang terdeteksi oleh
program CNN:

truck

Sistem Pengenalan Jenis Kendaraan

Citra masukan:

Path citra: D:\Code-Projects\Pengcit TUBES\IF4073-Tugas4-Citra-2023\test\image-009.jpg

Citra Masukan



Jenis Kendaraan yang terdeteksi oleh program non deep learning:

bus

Jenis Kendaraan yang terdeteksi oleh program CNN:

bus

Sistem Pengenalan Jenis Kendaraan

Citra masukan:

Path citra: D:\Code-Projects\Pengcit TUBES\IF4073-Tugas4-Citra-2023\test\image-010.jpg

Citra Masukan



Jenis Kendaraan yang terdeteksi oleh program non deep learning:

truck

Jenis Kendaraan yang terdeteksi oleh program CNN:

truck

Sistem Pengenalan Jenis KendaraanCitra masukan:

Path citra: D:\Code-Projects\Pengcit TUBES\IF4073-Tugas4-Citra-2023\test\image-011.jpg

Citra Masukan

Jenis Kendaraan yang terdeteksi oleh
program non deep learning:**truck**Jenis Kendaraan yang terdeteksi oleh
program CNN:**car**

Diskusi dan Analisis Hasil Program

Pendeteksian kendaraan menggunakan arsitektur KNN dapat mengklasifikasi kendaraan secara benar sebanyak 7 dari 11 gambar uji, yaitu akurasi sebesar 63.6%. Arsitektur ini mungkin bias mengklasifikasi kendaraan sebagai mobil untuk gambar uji dengan pencahayaan yang minim karena data dengan pencahayaan yang gelap biasanya adalah gambar mobil. Arsitektur KNN melakukan klasifikasi kendaraan berdasarkan fitur-fitur yang diekstrak dari gambar. Namun, keberhasilan model sangat tergantung pada jumlah dan representativitas dataset latih. Dalam hal ini, ukuran dataset latih relatif kecil, yang dapat mempengaruhi kemampuan model untuk melakukan generalisasi dengan baik terhadap data uji yang lebih kompleks. Oleh karena itu, arsitektur KNN belum dapat menghasilkan performa yang optimal untuk mengklasifikasikan jenis kendaraan dari gambar uji.

Sementara itu, pendekatan pendeteksian menggunakan arsitektur CNN berbasis YOLOv4 menunjukkan kinerja yang sangat baik pada hasil pengujian program. Dengan akurasi mencapai 100% pada pengujian 11 gambar uji, metode ini berhasil mendeteksi gambar uji dengan presisi tinggi. Kemampuannya tidak hanya terbatas pada gambar dengan kondisi pencahayaan yang optimal, tetapi juga efektif dalam situasi dengan pencahayaan yang minim. Kinerja YOLOv4 sangat baik berkat kompleksitas arsitekturnya yang canggih dan ketersediaan data latih yang luas dan beragam, memungkinkan model untuk mengklasifikasikan jenis kendaraan dengan akurat dalam berbagai skenario. Keunggulan ini mencerminkan bagaimana YOLOv4 secara efektif menerapkan pembelajaran mendalam untuk menghasilkan deteksi yang sangat akurat dan andal, sehingga YOLOv4 dapat dipertimbangkan sebagai metode pendeteksian objek yang sangat baik dan sangat mungkin untuk dikembangkan pada kasus lainnya.

Kesimpulan

Dalam pengembangan sistem pengenalan kendaraan menggunakan teknik pengolahan citra, dua pendekatan berbeda telah diimplementasikan. Program pertama menggunakan arsitektur KNN dengan teknik-teknik pengolahan citra digital yang telah dipelajari, sedangkan program kedua menggunakan algoritma CNN dengan arsitektur YOLOv4 sebagai pendekatan deep learning.

Program pertama, yang mengadopsi arsitektur KNN, menunjukkan akurasi sebesar 63.6% dalam mengklasifikasikan jenis kendaraan dari 11 gambar uji. Program ini masih memiliki keterbatasan terutama pada gambar dengan pencahayaan minim karena kemungkinan untuk mengklasifikasikan kendaraan sebagai mobil cenderung lebih tinggi pada gambar yang gelap. Hal ini dapat disebabkan oleh kekurangan jumlah dan representativitas dataset latih yang relatif kecil.

Sementara itu, program kedua dengan pendekatan CNN menggunakan YOLOv4 menghasilkan kinerja yang sangat baik dengan akurasi 100% pada 11 gambar uji. YOLOv4 memiliki kemampuan dalam mendeteksi objek pada gambar dengan presisi tinggi, bahkan dalam kondisi pencahayaan yang minim. Kompleksitas arsitektur CNN dan ketersediaan dataset latih yang luas memungkinkan model untuk mengklasifikasikan jenis kendaraan dengan akurat.

Komentar dan Refleksi Terhadap Tugas

Setelah mengimplementasikan program sesuai dengan spesifikasi dan melakukan eksperimen untuk menguji kebenaran program, ditemukan beberapa hal berikut untuk memperbaiki program untuk lebih baik ke depannya:

- Pada model non deep learning (KNN), perlu dilakukan eksplorasi metode ekstraksi fitur yang lebih efektif. Fitur-fitur yang berhasil diekstraksi dengan baik adalah yang mampu menunjukkan perbedaan yang signifikan antara berbagai jenis kendaraan, namun tetap mampu menggambarkan kesamaan di antara jenis kendaraan yang sama. Semakin baik fitur diekstraksi, maka metode dapat mengidentifikasi lebih banyak jenis kendaraan.
- Pada model non deep learning (KNN), model perlu dilatih dengan lebih banyak data sehingga memiliki kemampuan generalisasi yang lebih baik.
- Perlu eksplorasi model non deep learning selain KNN dan kemudian dibandingkan untuk mengetahui model non deep learning terbaik dalam mendeteksi jenis kendaraan.
- Model dengan deep learning (CNN) menunjukkan kinerja yang superior karena lebih baik dalam mengekstraksi fitur dibandingkan model dengan metode non deep learning.

Alamat GitHub Program

Alamat GitHub: <https://github.com/dennisheraldi/IF4073-Tugas4-Citra-2023>