Pengembangan Sistem Deteksi Dini Kebakaran Menggunakan Pendekatan *Transfer Learning* dengan Model ResNet50 Berbasis Website

Mega Intan Pratiwi[1], Salamun Rohman Nudin[2]

Manajemen Informatika, Universitas Negeri Surabaya
Jalan Ketintang, Ketintang, Kec. Gayungan, Surabaya, Jawa Timur 60231

[1]mega.20007@mhs.unesa.ac.id

[2]salamunrohman@unesa.ac.id

Abstrak*—Kebakaran merupakan peristiwa bencana alam yang dapat menyebabkan kerugian signifikan baik dari segi manusia maupun harta benda. Faktor-faktor seperti konsleting listrik, pembakaran sampah, dan ledakan tabung gas semakin meningkat di Indonesia, menambah potensi kebakaran. Deteksi dini kebakaran menjadi kunci untuk mengurangi kerugian. Sistem deteksi tradisional berbasis sensor skalar memiliki keterbatasan, seperti area jangkauan terbatas, khususnya jika dipasang di langit-langit dalam ruangan, sementara sistem vision menggunakan kamera yang dapat menjangkau area yang lebih luas. Transfer Learning, dengan fokus pada Convolutional Neural Network (CNN), menjadi pendekatan dalam penelitian ini untuk meningkatkan akurasi deteksi. Penelitian ini bertujuan untuk menyempurnakan sistem deteksi dini kebakaran, mengatasi keterbatasan sistem tradisional berbasis sensor skalar, dan memberikan solusi dalam melindungi nyawa dan harta benda dari risiko kebakaran. ResNet50 dipilih sebagai model utama untuk mendeteksi fire, smoke dan non-fire. Diharapkan bahwa pengembangan sistem ini akan memberikan respon lebih cepat dan akurat dalam mendeteksi kebakaran, terutama dalam area yang luas.

***Kata kunci*—Deteksi Dini Kebakaran, Confolutional Neural Network (CNN), Transfer Learning, ResNet50, Sistem Vision**

Abstract*—Fire is a natural disaster event that can cause significant losses in terms of both human lives and property. Factors such as electrical short circuits, waste burning, and gas cylinder explosions are increasing in Indonesia, adding to the potential for fires. Early fire detection is key to reducing losses. Traditional detection systems based on scalar sensors have limitations, such as limited coverage areas, especially if installed on indoor ceilings, while vision systems using cameras can cover larger areas. Transfer Learning, focusing on Convolutional Neural Networks (CNN), is the approach in this research to improve detection accuracy. This study aims to refine the early fire detection system, overcome the limitations of traditional sensor-based systems, and provide a solution to protect lives and property from fire risks. ResNet50 is selected as the main model to detect fire, smoke, and non-fire. It is expected that the development of this system will provide faster and more accurate responses in detecting fires, especially in large areas.

***Keywords*—Early Fire Detection, Confolutional Neural Network (CNN), Transfer Learning, ResNet50, Vision System**

I. PENDAHULUAN

Kebakaran merupakan salah satu peristiwa bencana alam yang dapat mengakibatkan cedera bahkan hilangnya nyawa serta harta benda seseorang (Dogan dkk., 2022) . Peristiwa ini bisa terjadi kapan saja dan dimana saja termasuk di hutan, rumah, dan gedung- gedung lainnya. Beberapa faktor yang akhir-akhir ini semakin marak terjadi di indonesia yang berpotensi menjadi sumber kebakaran yaitu disebabkan oleh beberapa faktor seperti konsleting listrik, pembakaran sampah, ledakan tabung gas, faktor alam seperti musim kemarau dengan paparan suhu tinggi disertai kecepatan angin yang tinggi hingga menimbulkan kebakaran hutan maupun lahan semakin cepat meluas, dan faktor penyebab kebakaran lainnya. Akibat dari bencana tersebut, masyarakat mengalami kerugian baik dari segi ekonomi, ekologi maupun manusianya sendiri.

Untuk menghindari kerugian akibat bencana kebakaran, penting untuk mendeteksi kebakaran secara otomatis sejak dini. Melakukan deteksi dini terhadap bencana kebakaran merupakan cara efektif untuk menyelamatkan nyawa. Beberapa metode deteksi dini kebakaran paling populer didasarkan pada sensor skalar dan vision. Sistem berbasis sensor skalar yang menggunakan sensor api, gas, dan suhu merupakan pendekatan tradisional untuk mendeteksi kebakaran. Biaya yang dikeluarkan untuk sensor ini lebih murah serta mudah digunakan, namun terdapat beberapa faktor yang membuat sensor skalar menjadi kurang dalam melakukan deteksi kebakaran misalnya area jangkauan yang terbatas dan sensor ini biasanya dipasang pada langit-langit sehingga hanya dapat digunakan di dalam ruangan saja (Yar dkk., 2023). Sedangkan, api dan asap membutuhkan waktu yang cukup lama untuk mencapai langit-langit tersebut sehingga sistem berbasis sensor skalar kurang cocok digunakan untuk deteksi dini pada kebakaran. Dengan demikian, sistem berbasis sensor vision dikembangkan untuk mengatasi hal tersebut. Sistem ini menggunakan pengaturan kamera yang terpasang untuk pengawasan, memiliki respon yang cepat dalam mendeteksi kebakaran serta dapat menjangkau area lebih luas.

Dengan menggunakan metode Transfer Learning sistem deteksi dini kebakaran yang akan penulis kembangkan berguna untuk mendeteksi api dan asap yang berpotensi menimbulkan kebakaran dalam kumpulan video yang memiliki kompleksitas waktu yang rendah. Dalam konteks ini, penggunaan teknologi pendeteksian citra, khususnya memanfaatkan Convolutional Neural Network (CNN), terbukti berhasil dalam pendeteksian objek, termasuk pendeteksian kebakaran. Dua arsitektur CNN yang terkenal yaitu ResNet50 menjadi pilihan utama untuk pengembangan sistem deteksi objek yang kompleks.

Namun, meskipun banyak penelitian yang berhasil mengimplementasikan metode deteksi ini, terdapat tantangan dan potensi pengembangan lebih lanjut, terutama dalam konteks deteksi dini kebakaran. Oleh karena itu, penelitian ini akan merancang sebuah metode deep learning yang lebih akurat, terkhusus menggunakan model ResNet50 untuk mendeteksi kebakaran secara langsung dalam kumpulan data video dengan kompleksitas waktu yang rendah.

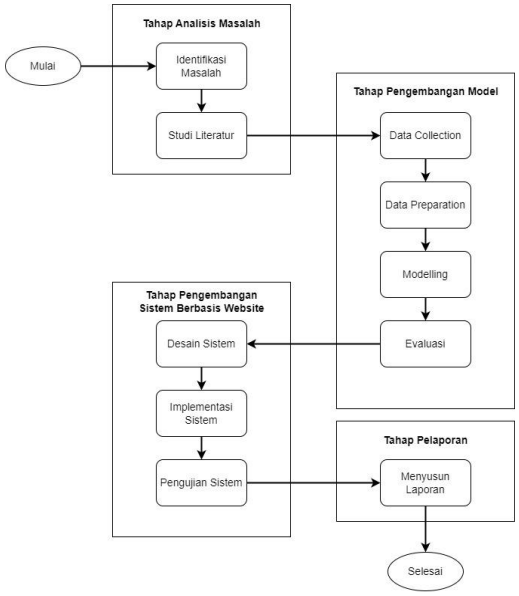
Pendekatan ini diharapkan dapat menjadi solusi untuk mengatasi keterbatasan sistem deteksi tradisional berbasis sensor skalar yang dalam penerapannya masih kurang efektif dalam melakukan deteksi kebakaran pada area yang luas dan memerlukan waktu yang cukup lama untuk merespon hal tersebut. Dengan pemanfaatan teknologi Convolutional Neural Network (CNN) dan model arsitektur yang lebih efektif seperti ResNet50 metode ini diharapkan dapat memberikan respon yang lebih cepat dan akurat dalam mendeteksi api dan asap kebakaran.

Selain itu, penggunaan teknologi pendeteksian citra dengan fokus pada deep learning, seperti CNN dapat memberikan wawasan yang lebih luas mengenai proses yang kompleks dan dapat meningkatkan kemampuan sistem untuk mengidentifikasi objek-objek dalam sebuah video. Hal ini menjadi relevan terutama dalam pencegahan kerugian secara signifikan dengan menggunakan deteksi dini kebakaran.

Dengan demikian, penelitian ini bertujuan untuk menyempurnakan metode deteksi dini kebakaran, mengatasi kendala-kendala yang terdapat dalam sistem sensor skalar, dan memberikan solusi yang lebih efektif dalam melindungi nyawa dan harta benda dari risiko kebakaran.

II. METODOLOGI PENELITIAN

Pada bab ini, akan dijelaskan alur atau tahapan yang akan dilakukan selama proses penelitian berlangsung. Adapun alur tahapan yang dilakukan dapat dilihat pada Gambar 1:



Gambar 1 Alur Perencanaan Penelitian

A. Tahap Analisis Masalah

1) Identifikasi Masalah

Secara umum, identifikasi masalah yang digunakan penulis sebagai sebab utama dalam melakukan penelitian ini adalah mengenai maraknya berita-berita kebakaran menunjukkan adanya potensi kerusakan serius pada properti, kehilangan nyawa, dan kerugian ekonomi.

2) Studi Literatur

Setelah melakukan identifikasi masalah, penelitian dilanjutkan dengan mencari studi literatur melalui jurnal penelitian produk serupa. Dari hasil identifikasi tersebut peneliti membuat interpretasi awal kebutuhan pengguna terhadap sistem deteksi dini kebakaran yang akan dikembangkan.

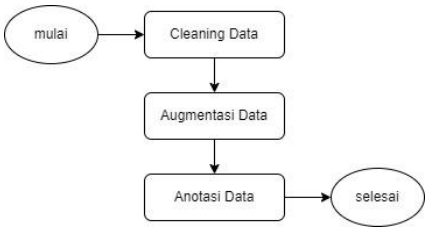
B. Tahap Pengembangan Model

1) Data Collection

Tahap yang pertama dalam pengumpulan data untuk sistem deteksi dini kebakaran adalah mengambil dataset gambar dari dataset yang dapat diakses melalui Kaggle. Gambar-gambar ini mencakup berbagai situasi kebakaran dan kondisi lingkungan yang berbeda, yang memungkinkan model untuk mengidentifikasi dengan baik bagaimana tanda-tanda awal kebakaran. Selanjutnya, data ini diproses dan dianalisis untuk melatih model deteksi, yang kemudian dapat digunakan untuk melakukan klasifikasi potensi kebakaran sekunder.

2) Data Preparation

Setelah melalui tahap pengumpulan data maka langkah selanjutnya adalah melakukan pre-processing data seperti pada Gambar 2.

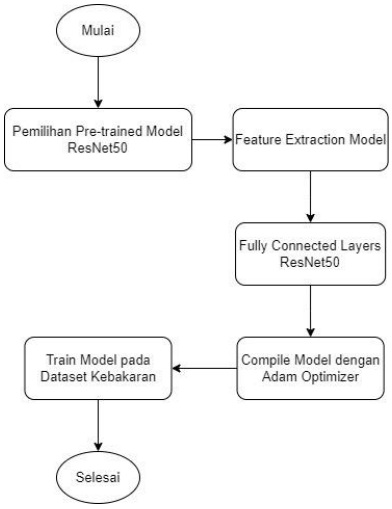


Gambar 2 Alur Pre-processing Data

Gambar 2 merupakan langkah-langkah dalam melakukan pre-processing. Tahap pertama yang dilakukan yaitu melakukan pembersihan data atau cleaning data. Tahap berikutnya dalam pre-processing yaitu augmentasi data. Augmentasi data adalah proses menghasilkan variasi tambahan pada dataset pelatihan dengan melakukan transformasi seperti rotasi, pergeseran, shear, zoom, dan flip horizontal. Fungsi yang akan digunakan yaitu 'ImageDataGenerator' dari TensorFlow untuk melakukan augmentasi data tersebut. Selain itu, tahap anotasi data juga penting dalam konteks deteksi kebakaran. Anotasi data melibatkan penambahan label atau informasi lainnya pada objek atau area tertentu dalam gambar.

Seluruh proses pre-processing ini adalah langkah penting dalam mempersiapkan dataset untuk pelatihan model deteksi kebakaran. Data yang bersih, teraugmentasi dengan baik, dan teranotasi dengan benar akan membantu model untuk memahami dan mengenali pola dengan lebih baik saat dilatih. Keberhasilan model dalam tugas deteksi kebakaran dapat sangat dipengaruhi oleh pre-processing yang cermat.

3) Modeling



Gambar 3 Alur Training Data

Pada Gambar 3 diatas menjelaskan terkait pemrosesan pelatihan data yang dilakukan menggunakan Google Colab. Proses akan dilakukan dengan menggunakan metode transfer learning dengan model ResNet50 dengan format file .ipynd yang kemudian disimpan kedalam Google Drive untuk menghindari terjadinya kehilangan data setelah pelatihan ketika terjadi error.

4) Evaluasi Data

Setelah tahap training data selesai, dilakukan evaluasi performa model menggunakan *confusion matrix* dan *classification report*. Dengan menggunakan *confusion matrix* dan *classification report*, menghasilkan informasi lebih lanjut tentang sejauh mana model dapat membedakan kelas-kelas dan performa model secara keseluruhan serta memberikan wawasan tentang *trade-off* antara mengidentifikasi positif dan menghindari *false positive* atau *false negative*. Hasil ini dapat membantu memahami kekuatan dan kelemahan model dalam klasifikasi. Misalnya, akurasi yang dihitung dengan menggunakan persamaan (1), nilai precision dihitung menggunakan persamaan (2), recall dihitung dengan menggunakan persamaan (3), dan f1-score dihitung menggunakan persamaan (4).

(1) Akurasi = $\frac{TP+TN}{TP+TN+FP+FN}$ …. Persamaan (1)

(2)

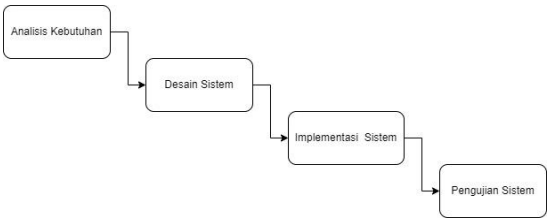
(3) Precision = $\frac{TP}{TP+FP}$ ……. (2)

(4) Recall = $\frac{TP}{TP+FN}$ …….. (3)

(5) F1 – Score = 2x$\frac{precision\ x\ recall}{precision+recall}$ ……. (4)

Suatu kondisi dikatakan *True Positive* (TP) atau *True Negative* (TN) ketika suatu observasi dikenali dengan benar, sedangkan False Positive (FP) dan *False Negative* (FN) merupakan kondisi suatu observasi salah diidentifikasi (Maharil, 2022).

C. Tahap Pengembangan Sistem Berbasis Website



Gambar 4 Hirarki Metode Waterfall

Pada Gambar 4 menjelaskan bahwa Sistem Deteksi Dini Kebakaran merupakan sistem yang bertujuan untuk mendeteksi potensi kebakaran secara tepat. Model Waterfall (model air terjun) dipilih dalam pengembangan sistem deteksi ini. Metode Waterfall memiliki langkah-langkah yang sangat terstruktur dalam mengembangan suatu sistem, tahapan yang dilakukan adalah sebagai berikut :

1) Analisis Kebutuhan

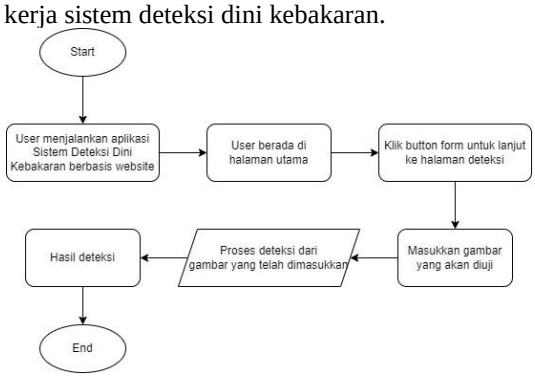
Analisis sistem diperlukan untuk mengetahui kebutuhan dan konfigurasi yang diperlukan untuk membuat produk penelitian berdasarkan permasalahan dan metode yang digunakan. Berikut merupakan analisis sistem yang diperlukan :

a. Kebutuhan untuk spesifikasi komputer minimal :
- CPU Intel Core i5 generasi ke 10
- RAM 8 GB
- Storage 1 GB
- OS (Windows/Linux/MacOS)
- VGA NVDIA

b. Aplikasi dan library yang diperlukan :
- Google Colab
- Visual Studio Code
- Python
- Pandas
- TensorFlow
- Numpy
- MatplotLib
- Keras
- Flask

2) Desain Sistem

Desain sistem untuk pengembangan sistem deteksi dini menggunakan diagram alur untuk memberikan gambaran alur kerja sistem deteksi dini kebakaran.



Gambar 5 Diagram Alur Pengguna

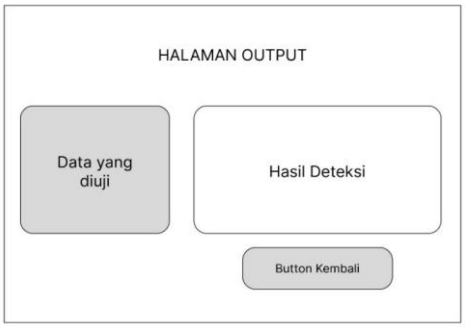
Gambar 5 merupakan desain sistem untuk pengembangan sistem deteksi dini menggunakan diagram alur untuk memberikan gambaran alur kerja sistem deteksi dini kebakaran. Selain desain sistem, terdapat desain antarmuka (User Interface) berupa wireframe yang akan dikembangkan menjadi website dengan tampilan yang user friendly diharapkan pengguna dapat dengan mudah menggunakan website ini, berikut tampilannya :

Gambar 6 Wireframe Halaman Utama Website

Gambar 6 diatas adalah wireframe perancangan halaman utama pada website sistem deteksi dini kebakaran yang akan dikembangkan. Dimana pada halaman utama website terdapat penjelasan dan form login atau button untuk menuju halaman deteksi.

Gambar 7 Wireframe Halaman Deteksi Website

Gambar 7 merupakan wireframe halaman deteksi untuk perancangan website sistem deteksi dini kebakaran.



Gambar 8 Wireframe Halaman Output Website

Gambar 8 merupakan wireframe halaman result untuk perancangan website sistem deteksi dini kebakaran.

3) Implementasi Sistem

Setelah melakukan desain sistem, kegiatan yang dilakukan untuk Pengembangan Sistem Deteksi Dini Kebakaraan adalah membuat sebuah program berbasis web yang dapat diakses dengan internet melalui URL berdasarkan rancangan sistem yang telah dibuat.

4) Pengujian Sistem

Pengujian produk deteksi dini kebakaran adalah langkah kritis dalam memastikan bahwa sistem tersebut dapat berfungsi dengan baik dan dapat diandalkan saat digunakan di lingkungan dunia nyata. Produk sistem deteksi dini kebakaran ini akan dilakukan pengujian dengan menggunakan metode *black-box*, dimana pengujiannya berfokus pada kode program yang telah dibuat menggunakan bahasa pemrograman python untuk menguji seberapa akurat hasil pelatihan data pada sistem ini. Pengujian dilakukan dengan memanfaatkan dataset berupa gambar dengan 3 kelas yaitu fire, smoke, non-fire, sehingga menghasilkan output deteksi dini kebakaran.

D. Tahap Pelaporan

Tahap terakhir dalam pengembangan sistem deteksi dini kebakaran adalah pelaporan, yang merupakan tahapan penting untuk menyampaikan informasi secara efektif

kepada pengguna, pemangku kepentingan, dan pihak terkait.

III. HASIL DAN PEMBAHASAN

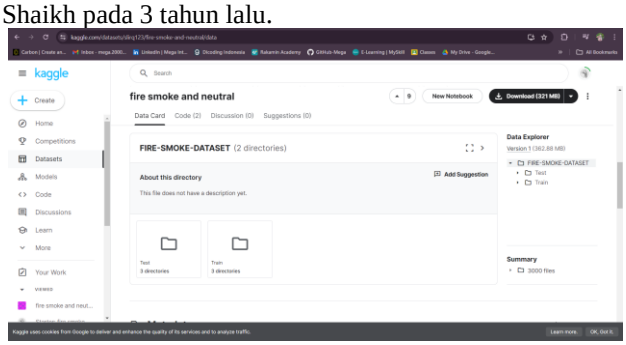
Pada bab ini, peneliti akan membahas hasil dari pengembangan model dan pengembangan sistem berbasis website.

A. Pengembangan Model

Dalam penelitian ini, proses pengembangan model dikembangkan dengan langkah-langkah yang sistematis mulai dari tahap data collection, data preparation, modeling, hingga evaluasi data yang diuraikan sebagai berikut:

1) Data Colletion

Pada proses data collection, peneliti menggunakan dataset yang diunduh dari Kaggle dengan judul dataset "fire smoke and neutral" yang diunggah oleh Abdullah Shaikh pada 3 tahun lalu.



Gambar 9 Dataset Fire, Smoke, and Neutral

Gambar 9 merupakan tampilan situs kaggle yang menunjukkan dataset yang akan digunakan peneliti. Dataset yang dipilih oleh peneliti berjumlah 3600 file gambar yang terbagi menjadi data train, data validation, dan data testing dengan perbandingan 60:20:20. Alasan peneliti memilih dataset ini karena dataset tersebut menyediakan data yang sesuai dengan penelitian ini, dimana dataset ini berisi sebuah gambar yang menunjukan situasi kebakaran (fire), adanya asap (smoke), dan situasi tidak terjadi kebakaran maupun asap (Non-fire). Berikut merupakan tabel jumlah dataset:

TABEL I
JUMLAH DATASET

Jenis Dataset		Total Dataset
Train	Dataset Fire	720
	Dataset Smoke	720
	Dataset Non-fire	720
Validation	Dataset Fire	240
	Dataset Smoke	240
	Dataset Non-fire	240
Testing	Dataset Fire	240
	Dataset Smoke	240
	Dataset Non-fire	240
TOTAL DATASET		**3600**

2) Data Preparation

Pada tahap data preparation, peneliti membagi proses menjadi 3 tahapan yaitu :

a. Proses Data Cleaning

```
1  def remove_duplicate_images(input_dir,
   output_dir):
2      if not os.path.exists(output_dir):
3          os.makedirs(output_dir)
4
5  remove_duplicate_images(input_directory,
   output_directory)
```

Gambar 10 Kode Program Proses Data Cleaning

Gambar 10 merupakan kode program data cleaning. Data cleaning adalah suatu proses penghapusan atau perbaikan pada dataset yang berfungsi untuk meningkatkan kualitas dataset. Proses pada data cleaning membutuhkan waktu yang cukup lama dikarenakan data yang dibersihkan memiliki sifat non-numeric. Dataset yang digunakan oleh peneliti berupa citra gambar, sehingga perlu dilakukan cleaning data seperti mengidentifikasi dan menghapus gambar duplikat yang mungkin ada didalam dataset, serta data cleaning dilakukan untuk mengantisipasi hasil yang bias pada proses training data.

b. Proses Augmentasi Data

Proses augmentasi bertujuan untuk membantu meningkatkan keanekaragaman data pelatihan, sehingga model dapat belajar pola yang lebih umum dan lebih baik dalam mengatasi overfitting. Untuk proses ini, peneliti menggunakan kode program dengan bahasa python berikut ini:

```
1  train_datagen = ImageDataGenerator(
2      rescale=1./255,
3      shear_range=0.2,
4      zoom_range=0.2,
5      horizontal_flip=True,
6      rotation_range=20,
7      width_shift_range=0.2,
8      height_shift_range=0.2
9  )
10
11 validation_datagen =
   ImageDataGenerator(rescale=1./255)
```

Gambar 11 Kode Program Proses Augmentasi Data

Pada Gambar 11 merupakan kode program augmentasi data. Peneliti menggunakan `'ImageDataGenerator'` dari TensorFlow atau Keras agar dapat menghasilkan generator data untuk pelatihan model. Parameter yang diberikan pada `'ImageDataGenerator'` tersebut adalah:

- `'rescale=1./255'` digunakan untuk mengubah nilai piksel gambar menjadi rentang yang lebih kecil untuk membantu proses pelatihan model.
- `'shear_range=0.2'` parameter ini digunakan untuk mengontrol peregangan (shear) gambar, sehingga objek didalamnya menjadi sedikit terdistorsi.

- ‘zoom_range=0.2’ parameter ini digunakan untuk mengontrol zoom-in dan zoom-out pada gambar, dengan nilai yang ditentukan yaitu 0.2 atau sama dengan 20% sehingga gambar dapat di zoom hingga 20%.
- ‘horizontal_flip=True’ parameter ini berfungsi agar gambar dalam dataset akan teracak diputar secara horizontal, sehingga gambar akan terlihat seperti cermin horizontal.
- ‘rotation_range=20’ parameter ini digunakan untuk menetukan seberapa jauh gambar dapat diputar dalam satu arah atau sebaliknya.
- ‘width_shift_range=0.2’ digunakan untuk mengatur rentang geseran (shift) horizontal untuk augmentasi data.
- ‘height_shift_range=0.2’ digunakan untuk mengatur rentang geseran (shift) horizontal untuk augmentasi data.

c. Proses Anotasi Data

Proses anotasi data adalah proses pemberian label pada dataset. Proses anotasi dilakukan dengan memberikan label pada masing-masing class (Fire, Smoke, Non-Fire). Berikut kode program yang digunakan untuk proses anotasi data:

```
1  def annotate_data(image, label)
2      annotated_image = cv2.putText(image,
       f'Label: {label}', (10, 30),
       cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255,
       255), 2, cv2.LINE_AA)
3
4      return annotated_image
```

Gambar 12 Kode Program Anotasi Data

Gambar 12 merupakan kode program anotasi data. Dari proses tersebut maka dihasilkan gambar dengan label yang sesuai dengan masing masing classnya.

3) Modeling

Proses modeling yang dilakukan peneliti menggunakan model arsitektur ResNet50, peneliti menjalankan prosesnya dengan menggunakan Google Colab agar dapat memanfaatkan GPU gratis yang telah di sediakan oleh Google Colab sendiri serta dapat menghubungkannya dengan Google Drive sebagai media penyimpanannya.

a. Pemilihan Pre-trained Model ResNet50

Langkah pertama yang dilakukan adalah melakukan instalasi library atau modul yang digunakan pada proses modeling.

```
1  import numpy as np
2  import pandas as pd
3  import os
4  import cv2
5  import tensorflow as tf
6  from tensorflow.keras.applications import ResNet50
7  from tensorflow.keras.layers import Dense,
   GlobalAveragePooling2D
8  from tensorflow.keras.models import Model
   from tensorflow.keras.preprocessing.image import
   ImageDataGenerator
```

Gambar 13 Kode Program Import Library

Gambar 13 merupakan kode program yang dibutuhkan pada instalasi library atau modul. Kemudian setelah modul terinstal pada file notebook, maka proses yang dilakukan selanjutnya adalah menghubungkan dengan Google Drive menggunakan fungsi ‘drive.mount()’ seperti pada Kode Program dibawah ini:

```
1  from google.colab import drive
2  drive.mount('/content/drive')
```

Gambar 14 Kode Program Connected Google Drive

Gambar 14 merupakan kode program untuk memberikan izin akses pada penyimpanan akun Google Drive agar dapat terhubung dengan Google Colab.

b. Feature Extraction Model

Pada tahap feature extraction, peneliti menggunakan model ResNet50 yang telah dilatih sebelumnya sebagai ekstraktor fitur. Berikut merupakan kode programnya:

```
1  base_model = ResNet50(weights='imagenet',
   include_top=False, input_shape=(224, 224, 3))
2
3  for layer in base_model.layers[:-10]:
4      layer.trainable = False
```

Gambar 15 Kode Program Feature Extraction

Gambar 15 diatas merupakan kode program yang menjelaskan bahwa peneliti memuat model ResNet50 yang telah dilatih sebelumnya dengan menggunakan ‘ResNet50’ dari TensorFlow/Keras, model yang telah dilatih pada dataset sebelumnya seperti imagenet untuk mengekstraksi fitur-fitur umum dari gambar, seperti tepi, tekstur, atau bentuk objek yang ada dalam gambar. Dengan ‘include_top=False’ membuat lapisan pengklasifikasian penuh atau fully connected tidak disertakan pada ujung model karena peneliti akan menambahkan custom layers di atasnya. Selanjutnya semua lapisan pada model ResNet50 dibekukan agar model tersebut tidak dilatih ulang selama proses training model sehingga, memungkinkan penggunaan representasi fitur yang sudah dipelajari sebelumnya oleh ResNet50 tanpa mempengaruhi bobot lapisannya.

c. Fully Connected Layers

Setelah melalui tahap feature extraction maka langkah selanjutnya yaitu menambahkan fully connected layers di atas untuk melakukan pengklasifikasian kebakaran. Berikut merupakan kode programnya:

```
1  x = base_model.output
2  x = GlobalAveragePooling2D()(x)
3  x = Dense(256, activation='relu')(x)
4  predictions = Dense(num_classes,
   activation = 'softmax')(x)
```

```
5  x = base_model.output
```

Gambar 16 Kode Program Fully Connected Layers Custom

Dari Gambar 16 dapat diketahui bahwa peneliti menambahkan fully connected custom layers di atas representasi fitur yang diperoleh dari ResNet50. Pertama, `'GlobalAveragePooling2D'` digunakan untuk meratakan representasi fitur menjadi vektor. Kemudian, peneliti menambahkan lapisan Dense dengan 256 neuron serta fungsi aktivasi ReLU. Lalu, pada lapisan Dense terakhir menggunakan `'Dense(3,activation='softmax')'` dengan maksud bahwa 3 merupakan multiclass yang menunjukkan jumlah kelas yang berbeda , yaitu fire, smoke, dan neutral. Selanjutnya, fungsi aktivasi softmax diatur agar menghasilkan distribusi probabilitas untuk setiap kelas sehingga model dapat melakukan klasifikasi multiclass dengan benar. Setelah itu, agar dapat menghasilkan model yang dapat melakukan klasifikasi yang lebih baik maka peneliti menggabungkan model dasar ResNet50 dengan custom layers yang telah ditambahkan sebelumnya dengan menggunakan `'model'` untuk pelatihan dan evaluasi sistem.

d. Compile Model dengan Adam Optimizer

Pada tahap ini, peneliti menentukan optimizer, fungsi loss, dan evaluation metrics untuk compile modelnya. Berikut merupakan kode programnya:

```
1  model = Model(inputs=base_model.input,
   outputs = predictions)
2
3  model.compile(optimizer = optimizer,
   loss='categorical_crossentropy',
   metrics=['accuracy'])
```

Gambar 17 Kode Program Compile Model

Dari Gambar 17 diketahui bahwa untuk pembuatan objek optimizer adam, learning rate (laju pembelajaran) yang digunakan dalam proses optimisasi diatur dengan nilai 0.001.

e. Train Model

Pada proses pelatihan data, peneliti menggunakan model arsitektur ResNet50 dan menentukan fungsi-fungsi yang diperlukan. Berikut kode programnya:

```
1  model_checkpoint =
   ModelCheckpoint('best_model.h5',
   monitor='val_accuracy',
   save_best_only=True, mode='max',
   verbose=1)
2
3  early_stopping =
   EarlyStopping(monitor='val_accuracy',
   patience=5, mode='max', verbose=1)
```

Gambar 18 Kode Program Memanggil Fungsi Callback

Dari Gambar 18 diatas, fungsi Callback 'ModelCheckpoint' dipanggil untuk menyimpan model terbaik berdasarkan kinerja terbaik dari akurasi validasi, fungsi tersebut digunakan untuk menghindari kehilangan model yang telah dilatih apabila terjadi kegagalan saat memuat kembali model terbaik untuk penggunaan lebih lanjut. Fungsi 'EarlyStopping' digunakan untuk menghentikan pelatihan apabila tidak ada peningkatan dalam metrik tertentu seperti tidak adanya peningkatan pada akurasi validasi setelah beberapa epoch. Pemanggilan fungsi 'EarlyStopping' juga dapat membantu agar tidak terjadi overfitting dan mengoptimalkan penggunaan sumber daya komputasi.

```
1  # Pelatihan Model
2  history = model.fit(
3      train_generator,
4
   steps_per_epoch=train_generator.samples
       // train_generator.batch_size,
5
   validation_data=validation_generator,
6
   validation_steps=validation_generator.
       samples // validation_generator.
       batch_size,
7      epochs=50,
8      callbacks=[early_stopping,
       model_checkpoint]
9  )
```

Gambar 19 Kode Program Training Model

Gambar 19 diatas merupakan kode program untuk melakukan pelatihan data dengan menggunakan 'fit' method serta menggunakan data generator untuk pelatihan dan validasi. Jumlah epoch yang ditentukan adalah 50, lalu memanggil Callback 'EarlyStopping' dan 'ModelCheckpoint' untuk memantau proses pelatihan data. Saat pelatihan data berlangsung, fungsi Callback 'EarlyStopping' bekerja, sehingga saat peneliti memberi epoch nilai 50 dan saat pelatihan data berlangsung di epoch 18/50 tidak ada peningkatan dalam metrik maka fungsi tersebut menghentikan pelatihan. Setelah proses pelatihan data selesai, langkah selanjutnya adalah menyimpan model deep learning yang telah dilatih sebelumnya agar dapat digunakan kembali tanpa perlu melatih ulang setiap kali melakukan deteksi atau saat aplikasi dijalankan.

4) Evaluasi Data

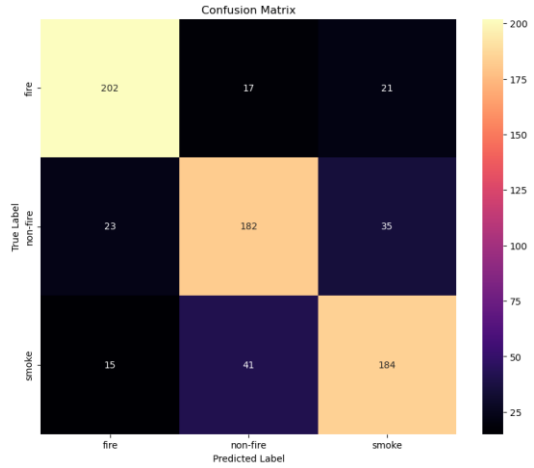
Setelah melakukan proses pelatihan data yang telah berlangsung selama kurang lebih 4 jam maka, perlu adanya evaluasi performa menggunakan data validasi.

Classification Report:

	precision	recall	f1-score	support
Fire	0.84	0.84	0.84	240
Non-fire	0.76	0.76	0.76	240
Smoke	0.77	0.77	0.77	240
Accuracy			0.79	720
Macro Avg	0.79	0.79	0.79	720
Weighted Avg	0.79	0.79	0.79	720

Gambar 20 Classification Report

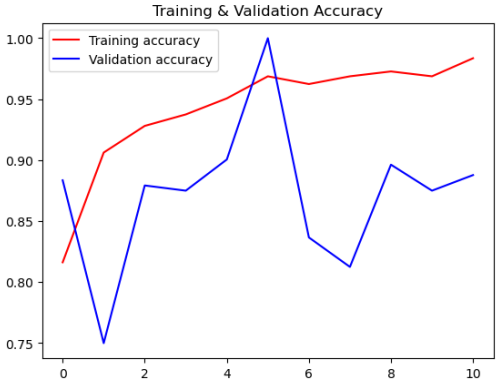
Gambar 20 diatas merupakan classification report untuk evaluasi data. Selain classification report, untuk mendapatkan pemahaman terkait performa model, peneliti juga memvisualisasikan confusion matrix dengan heatmap. Berikut adalah visualisasi heatmap confusion matrix:



Gambar 21 Visualisasi Heatmap Confusion Matrix

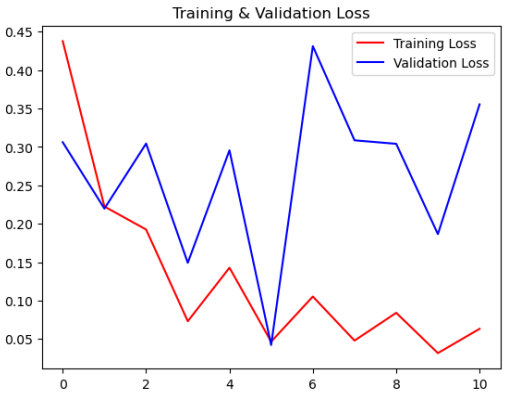
Gambar 21 merupakan heatmap confusion matrix yang menunjukkan performa dari model klasifikasi untuk tiga kelas yaitu Fire, Smoke, dan Non-fire. Baris pada heatmap meunjukkan label sebenarnya (*True Label*) dari data training, sedangkan kolom menunjukkan label yang di prediksi (*Predict Label*) oleh model. Warna dalam sel heatmap yang membentuk diagonal utama mengidentifikasi jumlah data yang diprediksi dengan nilai *true*, sedangkan diagonal lainnya bernilai *false*. Warna kuning lebih terang menunjukkan jumlah yang lebih tinggi dalam sel matriks, sedangkan warna yang lebih gelap menunjukkan jumlah yang lebih rendah. Dari gambar diatas dapat disimpulkan bahwa model memiliki kinerja yang cukup baik untuk mendeteksi gambar, namun terdapat beberapa kesalahan yang signifikan, dimana beberapa data diklasifikasikan kurang tepat oleh model. Kesalahan klasifikasi yang ditunjukkan pada baris dan kolom *False Positives* dan *False Negatives* dapat membantu mengidentifikasi area dimana model yang harus ditingkatkan lagi. Sehingga, dengan menggunakan *confusion matrix* peneliti mendapatkan informasi yang mendalam mengenai performa model dalam mengklasifikasikan setiap kelas.

Selain melihat heatmap *confusion matrix* untuk evaluasi data, peneliti dapat melihat performa pelatihan data yang telah dilakukan sebelumnya dengan melihat grafik berikut ini:



Gambar 22 Grafik Training & Validation Accuracy

Gambar 22 diatas merupakan grafik performa akurasi data training dan validation saat pelatihan berlangsung selama 10 epoch. Sumbu horizontal (x) menunjukkan jumlah epoch, yaitu iterasi pelatihan model, sedangkan sumbu vertikal (y) menunjukkan nilai akurasi yang berkisar antara 0.75 sampai 1.00. Kurva merah (training accuracy) menunjukkan akurasi model pada data penelitian, sedangkan kurva biru (validation accuracy) menunjukkan akurasi model pada data validasi. Dari grafik diatas maka dapat disimpulkan bahwa model menunjukkan peningkatan akurasi yang signifikan pada data pelatihan akan tetapi model menunjukkan performa akurasi yang kurang konsisten pada data validasi. Dari grafik diatas, didapatkan nilai akurasi tertinggi pada data training sebesar 0.98 dan nilai akurasi tertinggi pada dataset validation sebesar 1.00.



Gambar 23 Grafik Training & Validation Loss

Gambar 23 diatas merupakan grafik performa loss data training dan validation saat pelatihan berlangsung selama 10 epoch. Sumbu horizontal (x) menunjukkan jumlah epoch, yaitu iterasi pelatihan model, sedangkan sumbu vertikal (y) menunjukkan nilai loss yang berkisar dari 0.05 hingga 0.45. Kurva merah (training loss) menunjukkan loss model pada data penelitian, sedangkan kurva biru (validation loss) menunjukkan loss model pada data validasi. Dari grafik diatas maka dapat disimpulkan bahwa model menunjukkan penurunan loss yang signifikan pada data pelatihan akan tetapi model menunjukkan performa loss yang kurang konsisten pada data validasi. Dari grafik

diatas, didapatkan nilai loss terendah pada data training sebesar 0.05 dan nilai loss terendah pada dataset validation sebesar 0.04. Model ini nantinya akan digunakan untuk melakukan klasifikasi image.

B. Pengembangan Sistem Berbasis Website

Setelah melalui tahap pengembangan model, maka langkah selanjutnya yang dilakukan peneliti adalah pengembangan sistem berbasis website. Pengembangan sistem berbasis website dibagi menjadi 2 tahap yaitu tahap implementasi sistem dan tahap pengujian sistem, berikut adalah tahap yang dilakukan:

1. Implementasi Sistem

```
sistem_deteksi/
|__ static/
|   |__ css/
|   |   |__ style.css
|   |
|   |__ js/
|       |__ script.js
|
|__ templates/
|   |__ index.html
|   |__ result.html
|
|__ app.py
|__ best_model.h5
|__ util.py
```

Gambar 24 Struktur Folder Website Sistem Deteksi

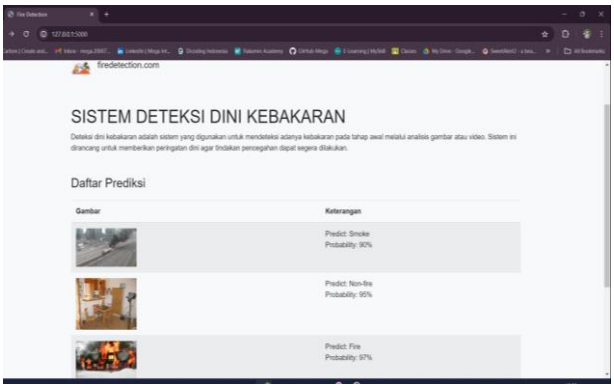
Gambar 24 merupakan penggambaran struktur folder untuk membangun website sistem deteksi dini kebakaran. Pada tahap implementasi sistem, file utama yang dipersiapkan adalah app.py yang menggunakan bahasa pemrograman python untuk mengatur logika aplikasi website, routing, mengelola data, dan integrasi model deep learning. Setelah mendefinisikan logika kode program app.py maka langkah selanjutnya adalah membuat tampilan pengguna untuk halaman utama, halaman deteksi, dan halaman output dari website.

2. Pengujian Sistem

Setelah melalui tahap implementasi sistem, langkah selanjutnya yaitu melakukan pengujian sistem tersebut. Dalam pengujian ini, sistem akan dilakukan pengujian fitur apakah sistem dapat menerima inputan dari pengguna, melakukan klasifikasi, menampilkan hasil klasifikasi, dan yang terakhir pengujian terhadap tingkat akurasi model dalam mendeteksi kebakaran. Berikut merupakan skenario penggunaan website sistem deteksi dini kebakaran untuk pengujian pada sistem :

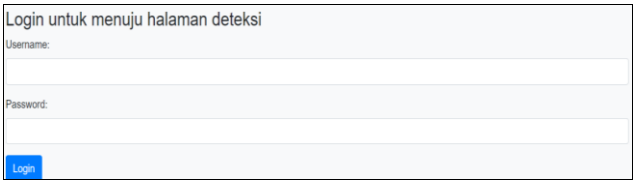
a. Mengunjungi Halaman Utama Website

Pada saat pertama kali mengunjungi website, pengguna akan melihat halaman utama website seperti pada gambar di bawah ini.



Gambar 25 Halaman Utama Website

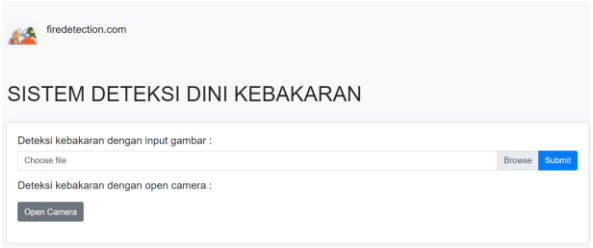
Gambar 25 adalah tampilan halaman utama website, apabila pengguna mengunjungi website maka pengguna akan diarahkan ke halaman ini. Selain itu, pengguna yang ingin menuju ke halaman deteksi harus melakukan login terlebih dahulu.



Gambar 26 Tampilan Halaman Login

Gambar 26 adalah tampilan halaman login, pengguna yang ingin mengunjungi halaman deteksi harus login terlebih dahulu dengan memasukkan username dan password, akan tetapi pengguna yang dapat.

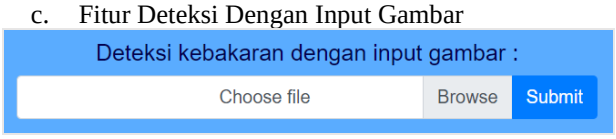
b. Halaman Deteksi



Gambar 27 Tampilan Halaman Deteksi

Gambar 27 merupakan tampilan dari halaman deteksi. Halaman ini dirancang untuk memberikan akses mudah kepada pengguna dalam melakukan deteksi dini kebakaran melalui dua metode utama yaitu dengan input gambar atau menggunakan kamera langsung.

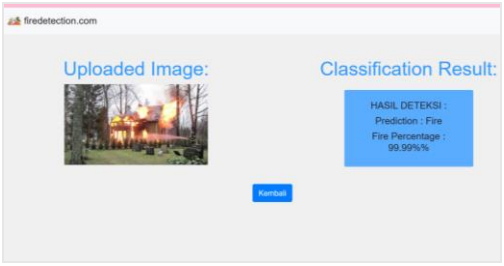
c. Fitur Deteksi Dengan Input Gambar



Gambar 28 Tampilan Fitur Deteksi Kebakaran dengan Input Gambar

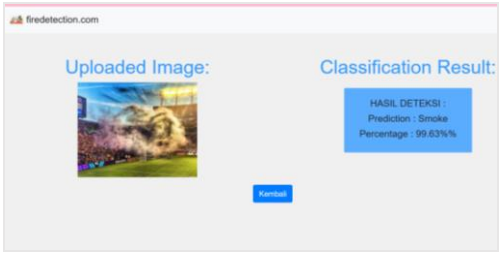
Gamabr 28 merupakan tampilan fitur deteksi dini kebakaran dengan input gambar. Pada fitur deteksi dengan input gambar, pengguna dapat menginputkan gambar yang akan dideteksi dengan langkah-langkah sebagai berikut :

- Langkah 1 : Pengguna dapat memilih opsi “Deteksi dengan Input Gambar” dengan mengklik tombol “Browse”.
- Langkah 2 : Setelah tombol “Browse” diklik, pengguna akan diarahkan untuk memilih gambar yang tersimpan di perangkat lokal.
- Langkah 3 : Setelah gambar dipilih, pengguna harus mengklik tombol “Submit” untuk memulai proses klasifikasi.
- Langkah 4 : Setelah proses klasifikasi selesai, pengguna akan diarahkan ke halaman "Classification Result". Di halaman ini, hasil klasifikasi atau prediksi akan ditampilkan bersama dengan persentase prediksinya seperti pada gambar dibawah ini :



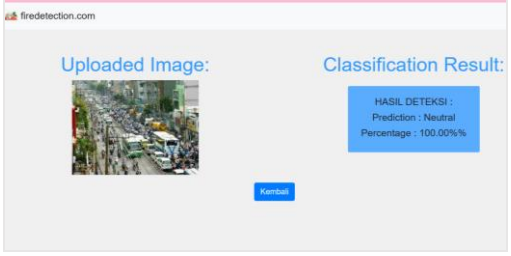
Gambar 29 Halaman Classification Result (Fire)

Gambar 29 merupakan situasi yang menunjukkan hasil deteksi kebakaran (Fire) dengan persentase gambar terdeteksi sebesar 99.99%.



Gambar 30 Halaman Classification Result (Smoke)

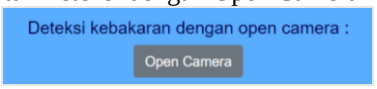
Gambar 30 merupakan situasi yang menunjukkan hasil deteksi adanya asap (Smoke) dengan persentase gambar terdeteksi sebesar 99.63%.



Gambar 31 Halaman Classification Result (Non-fire)

Gambar 31 merupakan situasi yang menunjukkan hasil deteksi tidak adanya kebakaran (Non-fire) dengan persentase gambar terdeteksi sebesar 100.00%.

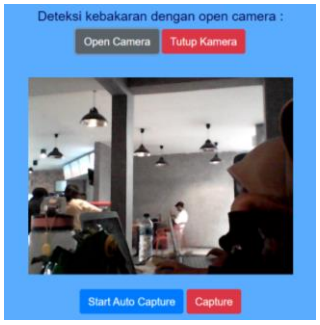
d. Fitur Deteksi dengan Open Camera



Gambar 32 Tampilan Fitur Deteksi Kebakaran dengan Open Camera

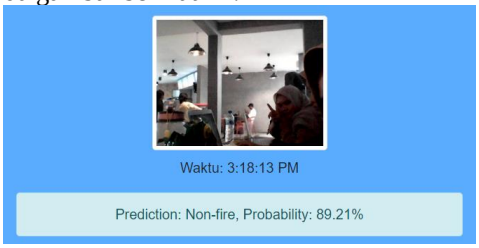
Gambar 32 merupakan tampilan fitur deteksi dini kebakaran dengan menggunakan open camera atau kamera secara langsung. Pada fitur deteksi dengan open camera, pengguna dapat langsung mendeteksi dengan menyalakan kamera secara dengan langkah-langkah sebagai berikut :

- Langkah 1 : Untuk melakukan deteksi dengan kamera secara langsung, pengguna dapat memilih opsi “Deteksi dengan Open Camera” dengan mengklik tombol “Open Camera”.
- Langkah 2 : Setelah tombol “Open Camera” diklik, maka kamera perangkan akan terbuka seperti pada gambar 33.



Gambar 33 Tampilan Ketika Open Camera

- Langkah 3 : Untuk mengambil gambar secara manual, pengguna dapat mengklik tombol "Capture" yang terletak di bagian bawah layar. Hasil klasifikasi akan muncul segera setelah gambar diambil, menampilkan prediksi dan persentasenya seperti pada gambar berikut ini.



Gambar 34 Hasil Deteksi dengan Situasi Non-fire

Gambar 34 merupakan hasil deteksi pada situasi yang menunjukkan tidak terjadi kebakaran dengan persentase gambar terdeteksi sebesar 89.21%.

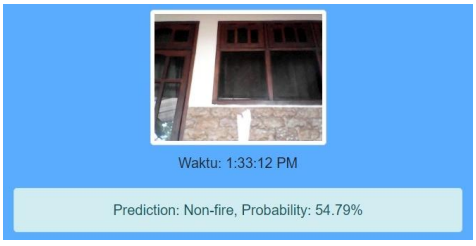
Gambar 35 Hasil Deteksi dengan Situasi Fire

Gambar 35 merupakan hasil deteksi pada situasi yang menunjukkan terjadi kebakaran (Fire) dengan persentase gambar terdeteksi sebesar 48.63%.

- Langkah 4 : Jika pengguna ingin mengambil gambar secara otomatis, pengguna dapat mengklik tombol "Start Auto Capture". Fitur ini akan mengambil gambar dan mendeteksi secara berkala. Pada uji coba website kali ini akan diatur pengambilan situasi setiap 10 detik dalam 1 menit dan menampilkan hasil klasifikasi beserta persentasenya untuk masing-masing gambar yang diambil, nilai persentase probability diambil dari output model prediksi yang menunjukkan tingkat keyakinan model terhadap kelas yang diprediksi. Sehingga, output yang ditampilkan setiap 10 detik dalam durasi 1 menit menampilkan hasil seperti dibawah ini.

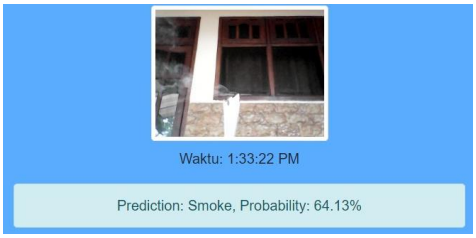
Gambar 36 Situasi Pada 10 Detik Pertama

Gambar 36 merupakan situasi di 10 detik pertama yang menunjukkan hasil deteksi tidak terjadi kebakaran (Non-fire) dengan persentase gambar terdeteksi sebesar 61.21%.



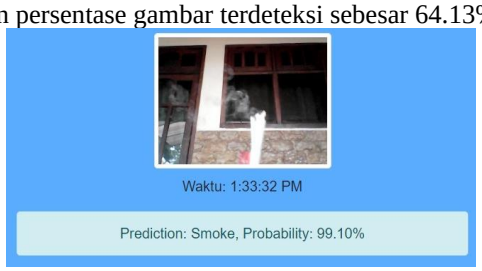
Gambar 37 Situasi Pada 10 Detik Kedua

Gambar 37 merupakan situasi di 10 detik kedua yang menunjukkan hasil deteksi tidak terjadi kebakaran (Non-fire) dengan persentase gambar terdeteksi sebesar 54.79%.



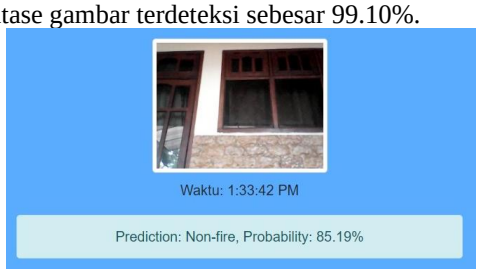
Gambar 38 Situasi Pada 10 Detik Ketiga

Gambar 38 merupakan situasi di 10 detik ketiga yang menunjukkan hasil deteksi tidak adanya asap (Smoke) dengan persentase gambar terdeteksi sebesar 64.13%.



Gambar 39 Situasi Pada 10 Detik Keempat

Gambar 39 merupakan situasi di 10 detik keempat yang menunjukkan hasil deteksi adanya asap (Smoke) dengan persentase gambar terdeteksi sebesar 99.10%.



Gambar 40 Situasi Pada 10 Detik Kelima

Gambar 40 merupakan hasil deteksi pada situasi yang menunjukkan tidak terjadi kebakaran (Non-fire) dengan persentase gambar terdeteksi sebesar 85.19%.

Gambar 41 Situasi Pada 10 Detik Keenam

Gambar 41 merupakan hasil deteksi pada situasi yang menunjukkan terjadi kebakaran dengan persentase gambar terdeteksi sebesar 52.87%.

- Langkah 5 : Ketika pengguna telah selesai mengambil gambar atau tidak lagi membutuhkan akses kamera, pengguna dapat mengklik tombol "Tutup Kamera" untuk menonaktifkan kamera perangkat pengguna. Dengan menutup kamera, pengguna tidak hanya menghentikan proses pengambilan gambar, tetapi juga menjaga privasi

dan mengurangi penggunaan daya baterai perangkat. Tombol tersebut memastikan bahwa pengguna dapat dengan mudah kembali ke halaman utama atau beralih ke fitur lainnya.

Meskipun sistem deteksi dini kebakaran ini dirancang untuk memberikan hasil yang akurat, ada beberapa situasi di mana sistem mungkin tidak dapat mendeteksi kebakaran dengan sempurna. Beberapa alasan utama yaitu termasuk kualitas kamera yang kurang baik, sehingga dapat mengurangi ketajaman dan kejelasan gambar yang diambil, serta kualitas cahaya yang tidak memadai, yang dapat mempengaruhi kemampuan sistem untuk menganalisis gambar secara efektif. Selain itu, jarak antara kamera dan objek yang terlalu jauh juga dapat mempengaruhi akurasi deteksi.

IV. KESIMPULAN

Setelah melalui semua tahapan penelitian, berikut ini adalah kesimpulan yang didapatkan:

1. Penelitian ini mengimplementasikan metode transfer learning menggunakan model ResNet50 untuk Sistem Deteksi Dini Kebakaran Berbasis Website. Pengembangan model dilakukan melalui langkah-langkah yang sistematis mulai dari data collection, data preparation, modeling, hingga evaluasi data. Dari hasil uji coba yang telah dilakukan oleh peneliti, Model ResNet50V2 (ResNet50 Versi 2) dipilih karena menghasilkan akurasi yang lebih tinggi dibandingkan dengan ResNet50. Dengan menggunakan ResNet50V2 sebagai model pre-training dan dioptimalkan dengan fully connected maka evaluasi model pada data training menunjukkan performa ketepatan sistem deteksi dini kebakaran dengan mencapai nilai akurasi tertinggi pada data training sebesar 0.98 dan nilai loss terkecil sebesar 0.06, pada data validasi sebesar 1.00 dan nilai loss terkecil sebesar 0.04, pada data testing sebesar 0.79. Model ini mampu mendeteksi gambar yang diunggah maupun yang diambil secara langsung menggunakan kamera.
2. Sistem deteksi dini kebakaran berbasis website dikembangkan dengan menggunakan framework Flask. Proses pengembangan sistem dibagi menjadi dua tahap utama yaitu implementasi dan pengujian. Dari tahap implementasi dan pengujian yang dilakukan, dapat disimpulkan bahwa antarmuka website yang dibangun dengan framework Flask berhasil mengintegrasikan model dari deep learning yang telah dilatih sebelumnya untuk deteksi dini kebakaran, pemroses dan memberikan hasil klasifikasi, serta menampilkan hasil klasifikasi dengan antarmuka pengguna yang mudah digunakan.

REFERENSI

[1] S. M. Metev and V. P. Veiko, *Laser Assisted Microtechnology*, 2nd ed., R. M. Osgood, Jr., Ed. Berlin, Germany: Springer-Verlag, 1998.

[2] J. Breckling, Ed., *The Analysis of Directional Time Series: Applications to Wind Speed and Direction*, ser. Lecture Notes in Statistics. Berlin, Germany: Springer, 1989, vol. 61.

[3] S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok, "A novel ultrathin elevated channel low-temperature poly-Si TFT," *IEEE Electron Device Lett.*, vol. 20, pp. 569–571, Nov. 1999.

[4] M. Wegmuller, J. P. von der Weid, P. Oberson, and N. Gisin, "High resolution fiber distributed measurements with coherent OFDR," in *Proc. ECOC'00*, 2000, paper 11.3.4, p. 109.

[5] R. E. Sorace, V. S. Reinhardt, and S. A. Vaughn, "High-speed digital-to-RF converter," U.S. Patent 5 668 842, Sept. 16, 1997.

[6] (2002) The IEEE website. [Online]. Available: http://www.ieee.org/

[7] M. Shell. (2002) IEEEtran homepage on CTAN. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/supported/IEEEtran/

[8] *FLEXChip Signal Processor (MC68175/D)*, Motorola, 1996.

[9] "PDCA12-70 data sheet," Opto Speed SA, Mezzovico, Switzerland.

[10] A. Karnik, "Performance of TCP congestion control with rate feedback: TCP/ABR and rate adaptive TCP/IP," M. Eng. thesis, Indian Institute of Science, Bangalore, India, Jan. 1999.

[11] J. Padhye, V. Firoiu, and D. Towsley, "A stochastic model of TCP Reno congestion avoidance and control," Univ. of Massachusetts, Amherst, MA, CMPSCI Tech. Rep. 99-02, 1999.

[12] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std. 802.11, 1997.