LAPORAN PENELITIAN
YANG DIAJUKAN KE LEMBAGA PENELITIAN DAN PENGABDIAN KEPADA MASYARAKAT

UNTAR
Universitas Tarumanagara

PENDETEKSIAN CITRA API PADA VIDEO DENGAN MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK

Disusun oleh:
Ketua Tim
Dra. Chairisni Lubis, M.Kom (0307096301/10393012)
Anggota
Zyad Rusdi ST, M.Kom (0315066602/10802017)

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS TARUMANAGARA
JAKARTA

2021

HALAMAN PENGESAHAN
LAPORAN PENELITIAN
Periode 1 / Tahun 2021

1. Judul Penelitian	:	Pendeteksian Citra Api pada Video dengan Menggunakan Convolutional Neural Network
2. Ketua Tim	:	
a. Nama dan gelar	:	Dra.Chairisni Lubis, M.Kom
b. NIDN / NIK	:	0307096301/10393012
c. Jabatan / gol	:	Lektor Kepala
d. Program Studi	:	Teknik Informatika
e. Fakultas	:	Teknologi Informasi
f. Bidang Keahlian	:	Artificial Intelligence
g. Alamat Kantor	:	Untar Kampus 1, Gedung R lt.10 Jl. S. Parman No.1
h. No. HP/Tlp/Email		087777344328/02154201427/chairisnil@fti.untar.ac.id
3. Anggota Tim Penelitian	:	
a. Jumlah Anggota	:	Dosen: 1 orang
b. Nama Anggota 1/Keahlian	:	Zyad Rusdi ST, M.Kom/ Sistem Informasi
c. Nama Anggota 1I/Keahlian	:	-
d. Nama Anggota 1II/Keahlian	:	-
e. Jumlah Mahasiswa	:	1 orang
f. Nama Mahasiswa/NIM	:	Byon Prawiradirdja/535170045
4. Lokasi Kegiatan Penelitian	:	Laboratorium FTI
5. Luaran yang Dihasilkan	:	Publikasi Internasional
6. Jangka Waktu Pelaksanaan	:	Periode I (Januari 2021 - Juni 2021)
7. Biaya yang disetujui LPPM	:	Rp.10.000.000,00

Jakarta, 31 Agustus 2021

Menyetujui Ketua LPPM	Ketua Tim
Jap Tji Beng, MMSI, Ph.D NIDN/NIK: 0323085501 / 10381047	Dra. Chairisni Lubis, M.Kom NIDN/NIK: 0307096301/10393012

RINGKASAN

Pada penelitian “Pendeteksian Citra Api pada Video dengan Menggunakan Convolutional Neural Network” ini, Metode *Convolutional Neural Network* (CNN) yang merupakan salah satu metode dalam *Deep Learning* digunakan untuk mendeteksi keberadaan api di suatu lokasi melalui video dan mendapatkan tingkat keberhasilan pendeteksian api yang tinggi dan ringan. Pada tahap pelatihan, input berupa citra yang mengandung api dan tidak. Diharapkan dari proses pelatihan didapat bobot keterhubungan yang tepat untuk dijadikan model CNN yang akan digunakan pada proses pengujian. Pada proses pengujian digunakan data uji yang berasal dari video yang berasal dari Kamera yang ditempatkan pad suatu lokasi. Dengan menggunakan Program Aplikasi Pendeteksi Api yang dirancang diharapkan sumber kebakaran dapat dideteksi dengan cepat berdasarkan informasi berupa video yang dikirimkan ke *User*. Untuk mengukur tingkat keberhasilan pendeteksiaan, digunakan nilai tingkat keakuratan yang merupakan perbandingan antara jumlah citra yang mengandung api dan tidak yang dapat dikenali dengan benar dengan jumlah seluruh citra pada video dalam data uji. Pada penelitian ini, tingkat akurasi pengenalan api dengan menggunakan model CNN sebesar 97%, kesalahan terjadi apabila kondisi videonya kurang baik

Kata Kunci: *Convolutional Neural Network*, *Deep Learning*, Pendeteksian Api

KATA PENGANTAR

Pada kesempatan ini, kami ingin melaporkan hasil penelitian kepada Lembaga Penelitian dan Pengabdian Kepada Masyarakat, Universitas Tarumanagara. Judul dari penelitian kami adalah “Pendeteksian Citra Api pada Video dengan Menggunakan Convolutional Neural Network”. Semoga penelitian ini dapat memperluas khasanah ilmu pengetahuan, terutama bidang teknologi informasi, khususnya bidang *Deep Learning*.

Jakarta, September 2021
Peneliti

(Chairisni Lubis, M.Kom)

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Kebakaran merupakan kejadian yang sangat berbahaya dan dapat berakibat fatal. Pada umumnya kebakaran akan terjadi apabila beberapa kondisi terpenuhi, contohnya seperti minyak yang terkena api, kayu yang tersambar petir, dan sebagainya. Selain itu, kebakaran pada umumnya juga membutuhkan waktu dan memenuhi syarat-syarat lain sebelum mencapai tahap yang tidak terkendali. Karena itu, jika terdapat informasi bahwa adanya api yang tidak wajar pada suatu lokasi, kejadian tersebut dapat dihentikan sebelum memasuki tahap yang tidak terkendali. Dengan informasi bahwa adanya api tersebut, maka dapat diputuskan apakah api tersebut berbahaya atau tidak, apakah akan terjadi, telah terjadi, atau tidak terjadinya kebakaran.

Sekarang sangat mudah memasang kamera di berbagai tempat untuk melihat situasi di suatu lokasi tanpa harus ada pemantau yang berada di tempat tersebut. Dengan adanya Kamera, maka lokasi tersebut akan selalu dapat dipantau dengan mengirimkan informasi mengenai lokasi tempat terpasangnya kamera tersebut. Dengan informasi yang telah terkirim, dapat dibuat berbagai macam program yang berfungsi untuk mengolah informasi tersebut sesuai dengan yang diinginkan oleh *user*. Dengan menggunakan teknologi dan informasi yang telah tersedia, maka dapat dibuat program yang terus mengawasi apakah ada api pada lingkungan sekitar. Jika terdapat api pada lingkungan sekitar, maka program tersebut akan memberitahu *user* sehingga dapat diketahui dengan cepat jika terjadinya kebakaran. Dengan demikian walaupun telah terjadi kebakaran, kejadian tersebut dapat dihentikan sebelum memasuki tahap yang tidak terkendali.

Convolutional Neural Network atau CNN merupakan salah satu metode *deep learning* yang baik untuk pengenalan pola sehingga dalam penelitian ini dapat digunakan untuk mengetahui adanya api pada suatu citra yang dikirimkan dari suatu lokasi. Keunggulan model CNN memiliki arsitektur yang sangat dalam, dengan demikian arsitektur model CNN yang akan dibuat dapat lebih baik dalam mengenali pola. Pada penelitian ini, program aplikasi yang dirancang dengan menggunakan CNN dapat memberitahu *user* bahwa apakah ada api di suatu lokasi.

1.2 Tujuan Khusus

Tujuan penelitian ini adalah untuk mendeteksi adanya citra api pada video apakah terdapat citra api atau tidak dengan menggunakan *Convolutional Neural Network* (CNN). Selain itu dengan menggunakan program aplikasi dibuat dapat diukur tingkat keakuratan CNN dalam mendeteksi adanya api dalam suatu citra.

1.3 Pentingnya Penelitian yang direncanakan

Program Aplikasi Pendeteksian Citra Api pada Video dengan Menggunakan Convolutional Neural Network yang dihasilkan dari penelitian ini dapat digunakan untuk mendeteksi adanya api pada suatu lokasi berdasarkan informasi yang dikirim dari CCTV atau Webcam yang dipasang pada lokasi tersebut.

BAB 2
STUDI PUSTAKA

Pada penelitian ini, akan digunakan Program Aplikasi Pendeteksi Api yang menggunakan metode *Convolutional Neural Network* (CNN) yang merupakan salah satu arsitektur dalam *Deep Learning* untuk mengenali apakah ada api pada suatu citra atau tidak dalam video yang dimasukkan. Perancangan Program Aplikasi akan dibangun dalam beberapa tahap yaitu tahap pengumpulan data, tahap pelatihan dan tahap pengujian.

Pada tahap pelatihan, data yang telah terkumpul akan di *resize* untuk mendapatkan ukuran yang sama sebelum dimasukkan ke dalam CNN, selain itu data juga akan dibersihkan dengan menggunakan penghilang *noise* agar data yang dimasukkan bersih. Setelah itu akan dilakukan proses pelatihan dengan memberikan data latih yang telah tersedia untuk membangun model CNN yang nantinya akan digunakan dalam proses Pengujian. Program Aplikasi Pendeteksi Api ini akan dibuat dengan menggunakan bahasa Python.

Berdasarkan penjelasan di atas, maka pada bab ini akan dijelaskan teori yang digunakan dalam membangun Program Aplikasi Pendeteksi Api ini.

2.1 Pendeteksi Api

Sebuah pendeteksi api dapat dianggap sebagai sebuah sistem untuk mendeteksi api. Sebuah pendeteksi api dapat menggunakan alat pendeteksi panas, alat pendeteksi asap, dan masih banyak lagi. Contoh alat pendeteksi api yang sekarang banyak digunakan adalah *sprinkler*. Jika *sprinkler* mendeteksi adanya asap maka alat tersebut akan mengeluarkan air karena api akan menghasilkan asap.

Pada penelitian *Computer Vision Based Fire Detection* pada University of California di San Diego dibuat algoritma untuk klasifikasi api dengan metode *multi-feature*-based. Pada penelitian tersebut dimasukkan ciri-ciri yang harus dicari pada suatu gambar untuk mengklasifikasikan bahwa sesuatu merupakan api atau bukan, seperti gerak-gerik api, warna pixel, analisis tekstur gambar, dan sebagainya. Hasil percobaan yang didapat dari penelitian ini adalah sebesar 80% benar dalam mengklasifikasikan adanya api atau tidak pada gambar. [1]

Pada penelitian *Video-Based Fire Detection Using Deep Learning Models* oleh Byongjun Kim dan Joonwhoan Lee digunakan metode R-CNN dan LSTM untuk mengidentifikasi api dari video. Hasil akurasi terbaik yang didapat dari penelitian ini adalah sebesar 95%. [2]

Arpit Jadon, Mohd. Omama, Akshay Varshney, Mohammad Samar Ansari, dan Rishabh Sharma membuat Perancangan Program Aplikasi Pendeteksi api yang ringan dan mudah dijalankan oleh perangkat keras biasa dengan CNN buatan sendiri. Hasil akurasi terbaik yang didapat dari penelitian ini adalah sebesar 96.53%.[]

Li-Wei Kang, I-Shan Wang, Ke-Lin Chou, Shih-Yu Chen, dan Chuan-Yu Chang melakukan penelitian *Image-based Real-Time Fire Detection Using Deep Learning with Data Augmentation for Vision-based Surveillance Applications* dengan menggunakan tiny-YOLOv3, yaitu algoritma pendeteksi objek yang sangat cepat. Penelitian ini dilakukan dengan memberikan data latih ke YOLO agar dapat mengidentifikasi api. Hasil keberhasilan tiny-YOLOv3 yang telah dilatih ini adalah sebesar 99.38% untuk mengidentifikasi api. [3]

Berdasarkan Penelitian di atas, akan dibuat Program Aplikasi Pendeteksi api dengan menggunakan *Convolutional Neural Network* (CNN) yang merupakan salah satu metoda *deep learning* untuk mendeteksi keberadaan api pada video yang diberikan. Program Aplikasi Pendeteksi Api yang dihasilkan diharapkan dapat mendeteksi api pada video lebih akurat dan ringan.

2.2 Video

Video adalah rangkaian sinyal elektronik yang digunakan untuk meghasilkan sumber gambar diam yang stabil yang mensimulasikan gerakan. Video dapat menggunakan grafik, gambar, atau teks, dan digunakan untuk hiburan, pendidikan, atau tujuan lain [4]. Satuan yang digunakan dalam pengukuran suatu video adalah *Frame Per Second* atau FPS yang digunakan untuk mengukur *frame rate*, yaitu banyaknya citra layar penuh yang ditampilkan secara berurutan setiap detik. Biasanya digunakan pada saat menangkap video dan menampilkannya Kembali, dan digunakan untuk mengukur performa video [5].

2.3 Noise

Noise yang dimaksud pada penelitian ini adalah *noise* pada gambar. *Noise* gambar biasanya memanifestasikan dirinya sebagai bintik acak pada permukaan yang halus dan dapat sangat mempengaruhi kualitas gambar. Kualitas gambar yang terpengaruhi *noise* ini terkadang dapat dibantu dengan meningkatkan ketajaman gambar.

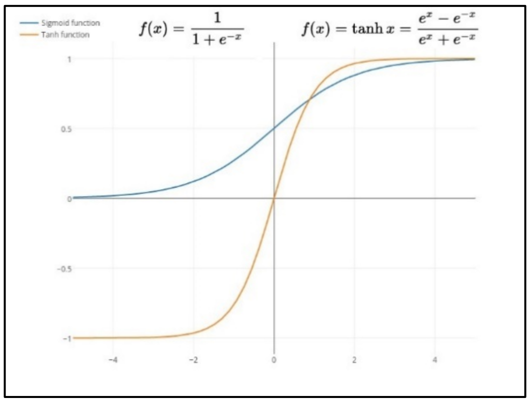
Tingkat *noise* biasanya meningkat tergantung pada panjang eksposur, suhu fisik, dan pengaturan sensitivitas kamera. Jumlah jenis *noise* gambar tertentu yang ada pada pengaturan tertentu bervariasi untuk model kamera yang berbeda dan terkait dengan teknologi sensor [6]. Pada beberapa *library* pembuatan perangkat lunak seperti cv2 terdapat fungsi yang akan digunakan untuk menghilangkan *noise* gambar pada perancangan ini.

2.4 Neural Network

Neural network adalah model yang terinspirasi oleh bagaimana neuron dalam otak manusia bekerja. Tiap neuron pada otak manusia saling berhubungan dan informasi mengalir dari setiap neuron tersebut. Tiap neuron menerima input dan melakukan operasi perkalian dengan sebuah bobot keterhubungan antar neuron, menjumlahkannya dan menambahkan bias. Hasil dari operasi ini akan dijadikan parameter dari fungsi aktivasi yang akan merupakan output dari suatu neuron [7]. Terdapat beberapa fungsi aktivasi yang biasa digunakan dalam Neural Network :

1. Sigmoid and Tanh Function

Fungsi aktivasi sigmoid mempunyai rentang antara 0 hingga 1 sedangkan rentang dari Tanh adalah -1 hingga 1. Kedua fungsi ini biasanya digunakan untuk klasifikasi 2 class atau kelompok data. Tanh memiliki kekurangan yaitu dapat mematikan gradient, tetapi kelebihannya adalah output yang dimiliki Tanh merupakan zero-centered. Dalam praktiknya Tanh lebih menjadi pilihan jika dibandingkan dengan Sigmoid [7]. Grafik dapat dilihat pada **Gambar 1** .



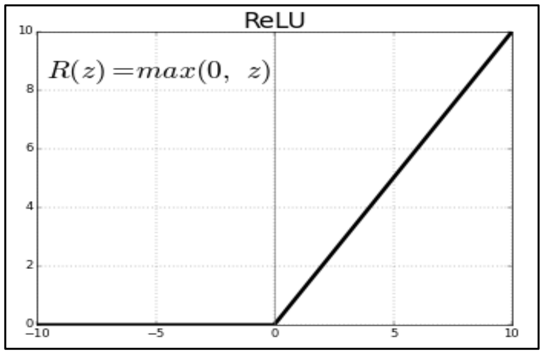
Gambar 1 Sigmoid Function

Sumber: Samuel Sena, Pengenalan Deep Learning Part 1 : Neural Network, https://medium.com/@samuelsena/pengenalan-deep-learning-8fbb7d8028ac , 11 Februari 2021

2. Rectified Linear Unit

ReLu merupakan salah satu fungsi aktivasi yang non linear. ReLU atau *Rectified Linear Unit* akan menampilkan nilai secara langsung jika hasil positif dan akan menghasilkan nol jika negatif. ReLU menjadi fungsi aktivasi default untuk banyak jenis Neural Network karena model yang menggunakannya menjadi lebih mudah untuk dilatih dan sering mencapai kinerja yang lebih baik. Namun ReLU memiliki kekurangan dapat membuat suatu unit menjadi "mati" jika learning rate yang digunakan terlalu tingg[1][7]. Fungsi aktivasi ReLU akan digunakan pada rancangan ini sebagai fungsi aktivasi neuron di *hidden layer*. Grafik ReLU dapat dilihat pada **gambar 2.**

[1] Ibid



Gambar 2 Relu Activation

Sumber: Samuel Sena, Pengenalan Deep Learning Part 1 : Neural Network, https://medium.com/@samuelsena/pengenalan-deep-learning-8fbb7d8028ac , 11 Februari 2021

Rumus perhitungan fungsi aktivasi ReLU sebagai berikut :

$$\boldsymbol{f(x) = \begin{cases} x, & if\ x \geq 0 \\ 0, & if\ x < 0 \end{cases}} \tag{1}$$

Dengan :

f(x) = Nilai keluaran dari aktivasi ReLU

x = Nilai masukkan aktivasi ReLU

3. Softmax

Softmax merupakan fungsi yang memberikan probabilitas untuk setiap kemungkinan kelas dalam model klasifikasi banyak kelas. Probabilitasnya berjumlah tepat 1. Misalnya, softmax menentukan bahwa probabilitas gambar tertentu adalah anjing sebesar 0,9, kucing 0,02, dan kuda 0,02 [8]. Fungsi aktivasi *softmax* akan digunakan pada rancangan ini sebagai fungsi aktivasi *output* pada lapisan keluaran. Rumus perhitungan fungsi aktivasi *softmax* sebagai berikut :

$$\boldsymbol{\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}} \tag{2}$$

Dengan :

$\boldsymbol{\sigma(z)_i}$: Vektor yang semua isinya antara 0 dan 1 yang jika semuanya dijumlahkan bernilai 1

z : Vektor yang berisi nilai yang didapatkan dari *fully-connected layer*

K : Banyaknya kelas yang ingin di klasifikasi

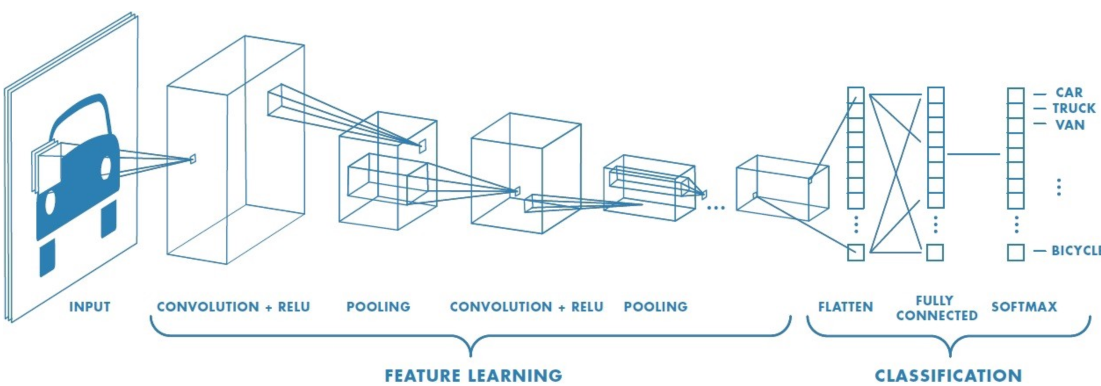
2.5 Supervised vs Unsupervised Learning

Pendekatan *supervised learning* mempuyai input dan output yang dapat dibuat menjadi suatu model hubungan matematis sehingga mampu melakukan prediksi dan klasifikasi berdasarkan data yang telah ada sebelumnya. Contoh algoritma yang termasuk dalam *supervised learning adalah :* Regresi linier berganda*,* Analisis deret waktu*,* Decision Tree dan Random Forest*,* Naïve Bayes Classifer*,* Nearest Neighbor Classifier*,* Artificial Neural Network*, dan* Support Vector Machine. [9]

Pendekatan *unsupervised learning* tidak menggunakan data latih atau data training untuk melakukan prediksi maupun klasifikasi. Berdasarkan model matematisnya, algoritma ini tidak memiliki target. Salah satu tujuan dari algoritma ini adalah mengelompokkan objek yang hampir sama dalam suatu area tertentu. Beberapa algoritma *unsupervised learning* adalah : K-Means, Hierachial Clustering, DBSCAN, Fuzzy C-Means, dan Self-Organizing Map. [9]

2.6 Convolutional Neural Network

Convolutional Neural Network (CNN) adalah salah satu jenis neural network yang biasa digunakan untuk pengenalan pola. CNN bisa digunakan untuk mendeteksi dan mengenali objek pada suatu gambar [10]. Gambaran arsitektur CNN dapat dilihat pada **Gambar 3** .



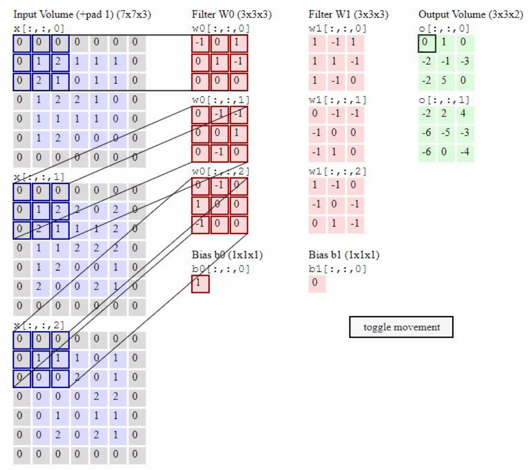
Gambar 3 Arsitektur Convolutional Neural Network

Sumber : Samuel Sena, Pengenalan Deep Learning Part 7 : Convolutional Neural Network (CNN), https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94, 11 Februari 2021

Arsitektur CNN terdiri dari banyak lapisan. Setiap lapisan CNN mengubah satu volume aktivasi ke volume lain melalui suatu fungsi. Dengan cara ini, CNN mengubah gambar asli lapis demi lapis dari nilai piksel asli ke nilai kelas akhir. Parameter CNN akan dilatih sehingga nilai kelas yang didapat konsisten sesuai dengan label yang diberikan saat pelatihan setiap gambar.[11]

2.2.5.1 Convolution Layer

Convolution layer terdiri dari neuron yang tersusun sedemikian rupa sehingga membentuk sebuah filter dengan panjang dan tinggi piksel tertentu. Filter akan digeser keseluruh bagian dari gambar. Setiap pergeseran akan melakukan operasi "dot" antara input dan nilai dari filter tersebut sehingga menghasilkan sebuah *output* atau biasa disebut sebagai *activation map* atau *feature map* yang dapat dilhat pada **gambar 4. [12]**



Gambar 4 Contoh *Activation Map*

Sumber : Cs231n , <u>Convolutional Neural Networks for Visual Recognition</u>, https://cs231n.github.io/convolutional-networks/

Activation map yang telah didapat akan dimasukkan ke dalam neuron di *hidden layer* melalui suatu fungsi aktivasi, agar fungsi aktivasi yang digunakan dapat mengeluarkan hasil yang dapat digunakan pada lapisan berikutnya. Fungsi aktivasi neuron di *hidden layer* yang digunakan pada rancangan ini adalah ReLU.

Stride

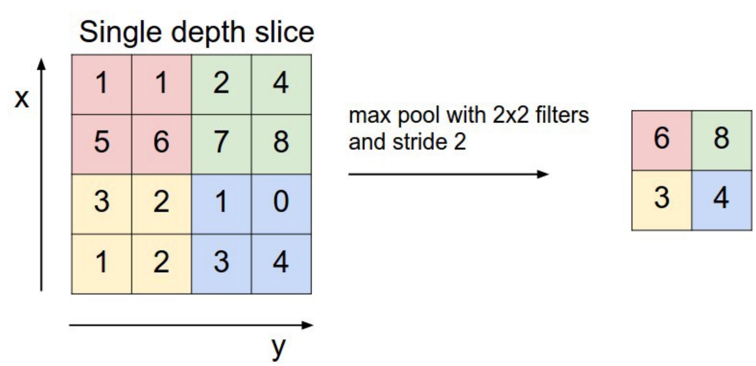
Stride adalah parameter yang menentukan berapa jumlah pergeseran filter. Jika nilai stride adalah 1, maka filter pada convolutional layer akan bergeser sebanyak 1 piksel secara horisontal lalu vertikal. Semakin kecil stride maka akan semakin detail informasi yang akan didapatkan dari sebuah input, namun membutuhkan komputasi yang lebih jika dibandingkan dengan stride yang besar [12].

Padding

Padding atau zero padding adalah parameter yang menentukan jumlah piksel (berisi nilai 0) yang ditambahkan di setiap sisi dari input. Hal ini digunakan dengan tujuan untuk memanipulasi dimensi output dari convolutional layer (Feature Map). Tujuan dari penggunaan padding adalah untuk mengatur dimensi output agar tetap sama seperti dimensi input atau setidaknya tidak berkurang secara drastis. Sehingga dapat menggunakan convolutional layer yang lebih dalam agar lebih banya ciri/fitur yang berhasil diekstrak. Hal ini meningkatkan performa dari model karena convolutional filter akan fokus pada informasi yang sebenarnya yaitu berada diantara zero padding tersebut.[12]

2.2.5.2 Pooling Layer

Pooling layer biasanya berada setelah convolutional layer. Pooling layer terdiri dari filter dengan ukuran dan stride tertentu yang akan bergeser pada seluruh area feature map. Tujuan dari penggunaan pooling layer adalah mengurangi dimensi dari feature map, sehingga mempercepat komputasi karena parameter yang ada semakin sedikit dan mengatasi overfitting. Pooling yang biasa digunakan adalah max pooling dan average pooling. Max pooling akan memilih pixel terbesar dari dalam filter setiap pergeseran filter, sedangkan average pooling akan memilih nilai rata-ratanya [12]. Contoh max pooling 2x2 dengan stride 2 dapat dilihat pada **gambar 5.**



Gambar 5 Contoh Max Pooling

Sumber : Samuel Sena, Pengenalan Deep Learning Part 7 : Convolutional Neural Network (CNN), https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94, 13 November 2017

2.2.5.3 Fully-Connected Layer

Feature map yang dihasilkan dari convolutional dan pooling layer berbentuk multidimensional array, sehingga harus melakukan "flatten" atau reshape feature map menjadi sebuah vektor agar dapat kita gunakan sebagai input fully-connected layer. Fully-connected layer memiliki beberapa hidden layer, activation function, output layer, dan loss function [12]. *Fully-connected layer merupakan tempat* sistem mencari dan menghasilkan nilai kelas akhir yang digunakan untuk mengklasifikasi citra. Fungsi aktivasi *hidden layer* yang digunakan pada rancangan ini adalah ReLU, dan fungsi aktivasi *output* yang akan digunakan pada perancangan ini adalah softmax.

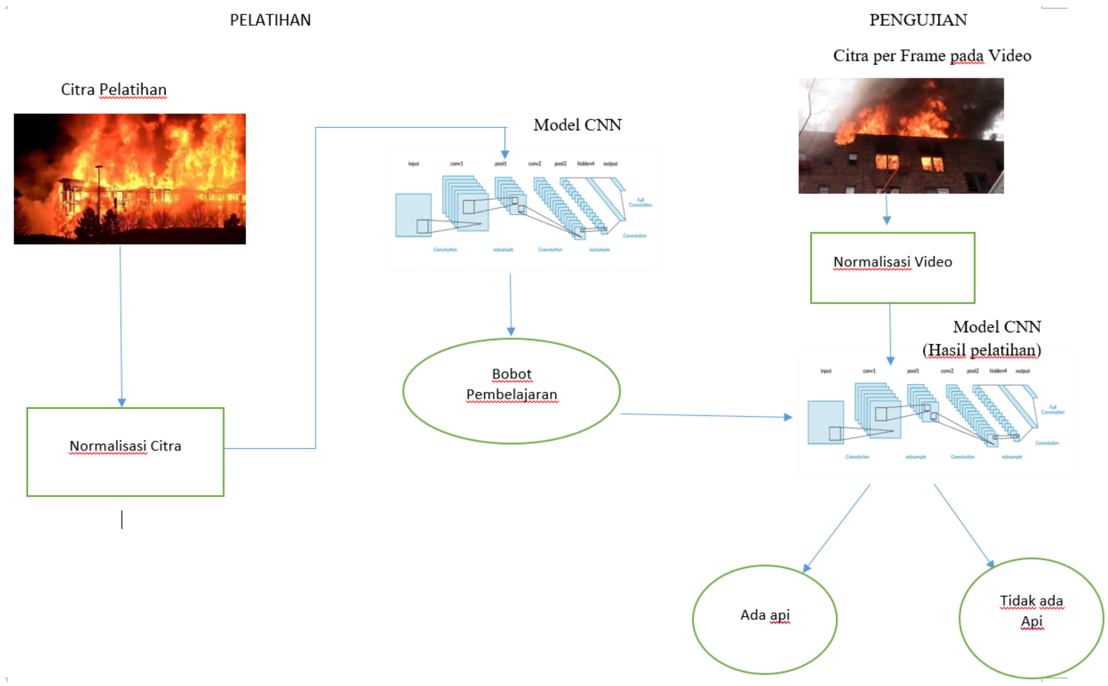
BAB 3
METODE PENELITIAN

3.1 Metode yang digunakan

Pada penelitian “Pendeteksian Citra Api pada Video dengan Menggunakan Convolutional Neural Network” ini, akan dirancang suatu Program Aplikasi yang dapat mendeteksi adanya citra api pada video. Program aplikasi penedeteksi api ini dirancang dengan menggunakan metode *Convolutional Neural Network* (CNN) dan dikembangkan dengan menggunakan bahasa pemogramaan Python.

Pada tahap awal Program Aplikasi akan menerima masukan data latih berupa citra yang mengandung api dan tidak. Sistem akan melakukan proses normalisasi citra dengan membuat semua ukuran citra yang dimasukkan sama dan melakukan proses *de-noise* agar data citra yang dimasukkan semua bersih. Citra yang telah dinormalisasi akan dimasukkan ke dalam arsitektur CNN untuk dilatih agar dapat menghasilkan bobot keterhubungan yang akan disimpan dalam model. Model yang sudah dilatih tersebut kemudian akan digunakan dalam proses pengujian dengan menggunakan data uji. Diagram alur ide rancangan dapat dilihat pada **gambar 6.**

.



Gambar 6. Diagram alur ide rancangan.

Program Aplikasi Pendeteksian Api yang dirancang menggunakan metode System Development Life Cycle (SDLC) yang merupakan pola pengembangan perangkat lunak yang terdiri dari: tahap perencanaan, tahap analisis, tahap perancangan, tahap implementasi, tahap pengujian dan tahap pemeliharaan. Pada bab ini akan dibahas: tahap perencanaan, tahap analisis, tahap perancangan.

3.1.1 Tahap Perencanaan

Pada tahap awal perancangan Program Aplikasi Pendeteksi Api ini akan dilakukan pengumpulan data yang dibutuhkan dan perencanaan pembangunan Sistem Aplikasi. Data berupa citra yang mengandung api dan yang tidak mengandung api diambil dari website Mivia, github, dan google. Metode yang digunakan untuk mendeteksi api dalam video adalah *Convolutional Neural Network* (CNN) yang merupakan salah satu metode dalam *Deep Learning*. Tahap awal dari perancangan Program Aplikasi ini adalah membuat tampilan *Graphical User Interface* (GUI) yang akan menerima input dari *user* berupa sebuah video. Selama video tersebut berputar aplikasi akan memberikan keluaran apakah ada api pada suatu *frame* video.

3.1.2 Tahap Analisis

Pada tahap analisis dilakukan analisis identifikasi kebutuhan yang digunakan untuk pembuatan Program Aplikasi. Metode pendeteksian api yang akan digunakan merupakan metode *deep learning* dengan model CNN. Citra akan dimasukkan ke dalam Program Aplikasi pada saat pelatihan. Citra tersebut akan dinormalisasi dengan cara diubah semua ukuran citra yang dimasukkan menjadi ukuran yang sama dan menghilangkan *noise* yang mungkin ada pada citra. Setelah citra dinormalisasi, maka akan dimasukkan ke dalam lapisan konvolusi, keluaran dari lapisan konvolusi akan dihitung dengan menggunakan fungsi aktivasi ReLU. Keluaran dari lapisan konvolusi akan masuk ke lapisan *pooling,* dan setelah diproses dalam lapisan *pooling* hasilnya akan di *flatten* dan dimasukkan ke dalam *fully-connected layer* yang terdiri dari *hidden layer* berupa *dropout layer* untuk mengurangi *overfitting*, *dense layer* pertama dengan fungsi aktivasi ReLU, dan *dense layer* yang kedua (*output layer*) dengan fungsi aktivasi *softmax* yang akan menentukan apakah citra tersebut mengandung api atau tidak. Hasil pelatihan berupa bobot keterhubungan akan disimpan dan digunakan sebagai model CNN yang akan digunakan pada proses pengujian untuk menentukan adanya api pada citra atau tidak dalam video.

Pada saat pengujian program aplikasi akan mengambil citra dari video yang telah dimasukkan setiap satu detik selama video tersebut berputar. Citra tersebut akan dinormalisasi seperti saat pelatihan dengan diubah ukurannya dan dihilangkan *noise* yang ada. Setelah citra dinormalisasi maka akan dimasukkan ke dalam model CNN yang telah dilatih dan model tersebut akan menentukan apakah ada api pada citra yang diambil dari video itu. Proses ini akan terus diulang sampai video berhenti berputar. Diagram alur ide rancangan program ap likasi dapat dilihat pada **gambar 6**.

Perangkat keras yang digunakan untuk membangun Program Aplikasi ini adalah:

1. Laptop dengan Prosesor Intel i7-7700
2. RAM 16GB
3. Nvidia GeForce GTX 1050
4. HDD dengan kapasitas 289 GB

Dan Perangkat lunak yang digunakan adalah:

1. Sistem Operasi Windows 10
2. Bahasa Pemrograman Python 3.7.4
3. Google Colab
4. Jupyter Notebook

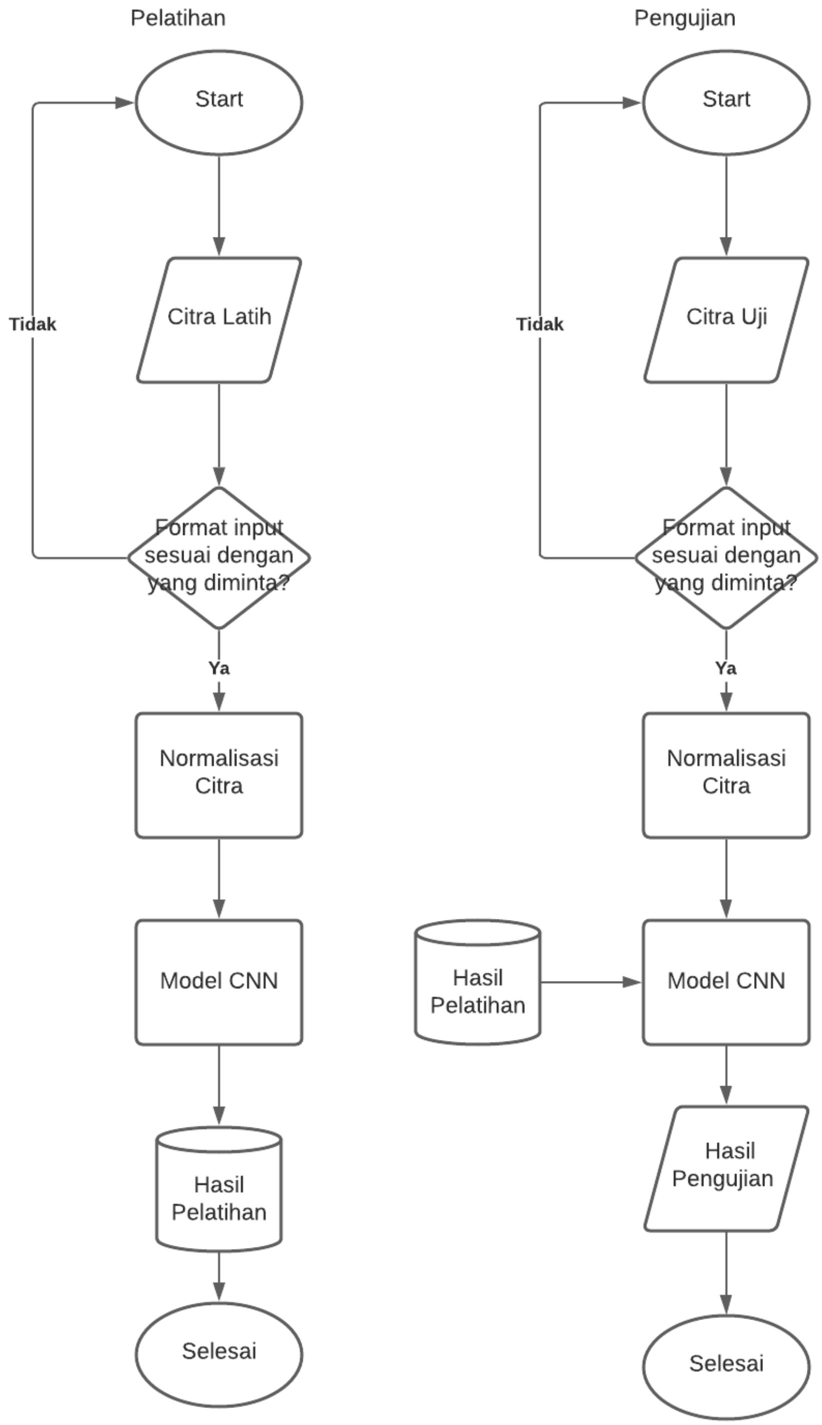
5. *Library* Tensorflow, Keras, numpy, dan cv2.

3.1.3 Tahap Perancangan

Pada tahap ini bertujuan untuk mendapatkan gambaran bentuk rancangan Program Aplikasi Pendeteksi Api. Tahap perancangan ini menghasilkan beberapa tipe rancangan, yaitu flowchart program**,** diagram alir data, dan tampilan antarmuka.

3.1.3.1 Rancangan Flowchart

Flowchart merupakan diagram alir yang terdiri dari simbol-simbol tertentu yang digunakan untuk menggambarkan bagaimana proses dari sistem yang dirancang berjalan dari awal sampai selesai. Flowchart dapat dilihat pada **gambar 7** di bawah ini**.**



Gambar 7. Flowchart Program Aplikasi Pendeteksi Api

3.1.3.2 Rancangan Tampilan Antarmuka

Rancangan tampilan antar muka adalah rancangan modul yang memberikan gambaran mengenai tampilan dari aplikasi yang dibangun dimana pengguna dapat berinteraksi dengan sistem aplikasi. Berikut adalah rancangan tampilan pada aplikasi **:**

1. Modul Home

   Modul Home adalah tampilan utama yang keluar saat aplikasi dijalankan. Modul ini merupakan tempat navigasi utama yang memiliki tombol menuju semua modul lain. Pada modul ini terdapat tombol navigasi menuju modul *detect*, modul *help* dan modul *about*. Modul *home* dapat dilihat pada **gambar 8.**

2. Modul Detect

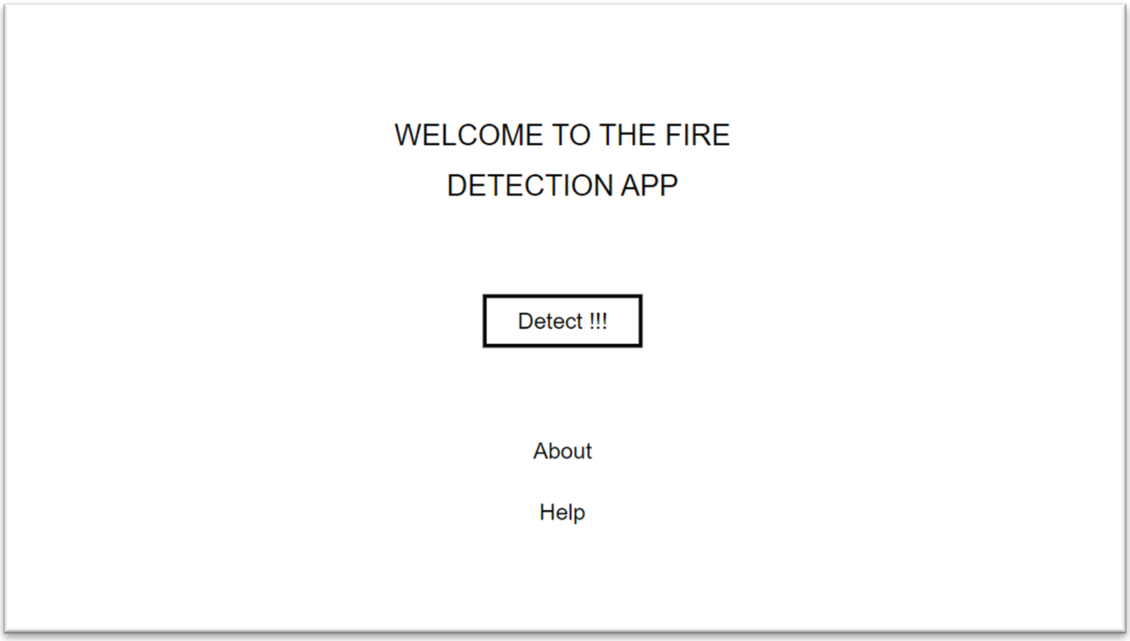
   Modul *Detect* adalah tempat dimana pengguna memasukkan video yang ingin dideteksi. Pada modul ini jika video telah dimasukkan dan dijalankan, maka aplikasi akan mulai mendeteksi apakah adanya api pada video tersebut atau tidak dan terdapat hasil klasifikasi di bawah video. Pada modul ini terdapat tombol mulai video, pause video, masukkan video, navigasi ke modul home dan tombol navigasi ke modul help. Modul *detect* dapat dilihat pada **gambar 9.**

3. Modul Help

   Modul Help adalah tampilan yang menunjukkan panduan untuk menggunakan aplikasi. Pada modul ini terdapat tombol navigasi menuju modul home. Modul *help* dapat dilihat pada **gambar 10.**

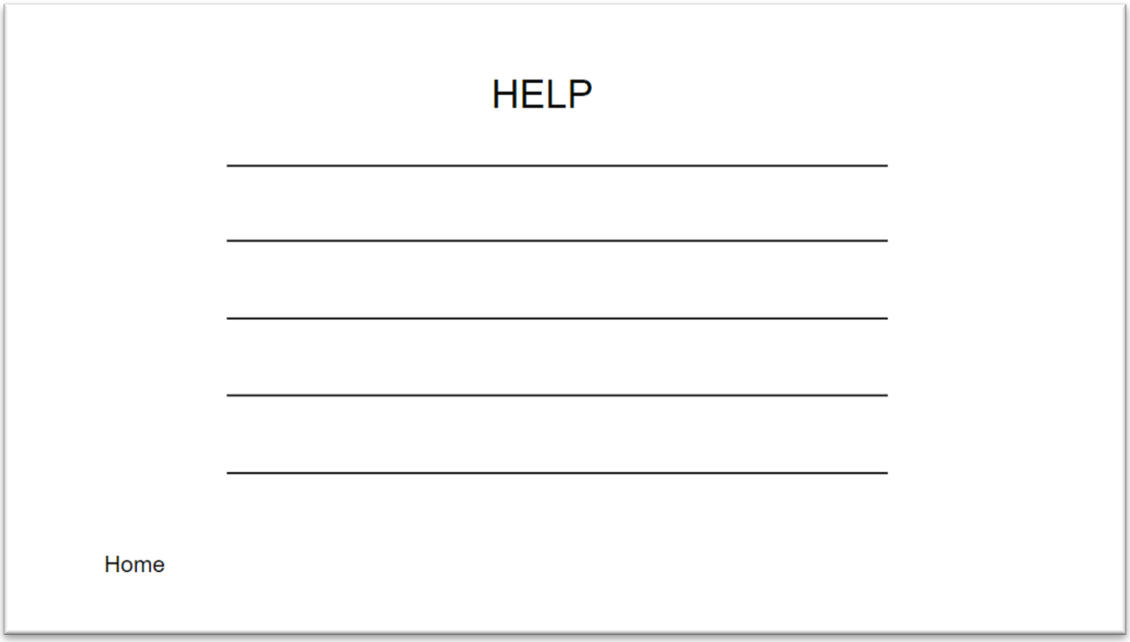
4. Modul About

   Modul About adalah tampilan yang menunjukkan informasi tentang aplikasi deteksi adanya api dengan menggunakan Convolutional Neural Network pada video dan tentang pembuat aplikasi. Pada modul ini terdapat tombol navigasi. menuju modul home. Modul *about* dapat dilihat pada **gambar 11.**

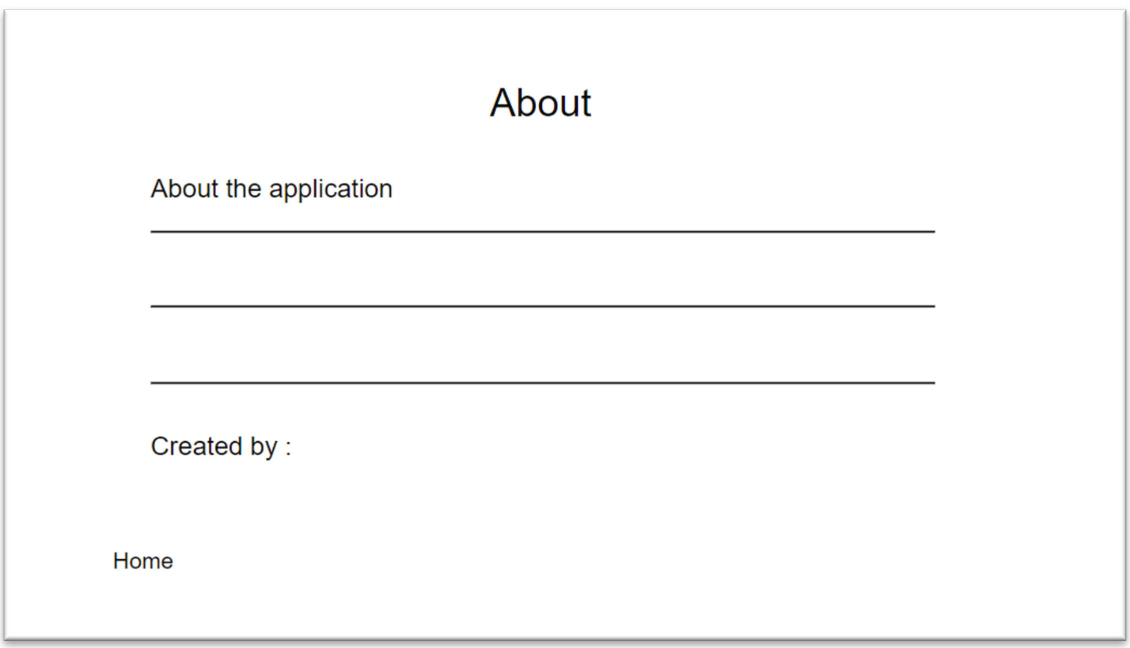


Gambar 8. Rancangan Modul Home

Gambar 9. Rancangan Modul Detect



Gambar 10. Rancangan Modul Help



Gambar 11. Rancangan Modul About

3.2 Manfaat Penelitian

Beberapa manfaat yang diharapkan dari aplikasi pada penelitian ini adalah:

- Program aplikasi yang dihasilkan dari penelitian ini dapat dimanfaatkan untuk mendeteksi citra api pada video yang dikirimkan melalui CCTV atau Webcam dari suatu lokasi sehingga informasi ini dapat diketahui oleh user. Dengan adanya informasi ini kebakaran dapat diketahui lebih cepat.
- Hasil penelitian ini dapat dipublikasikan pada Seminar atau Jurnal Nasional/ Internasional.

BAB 4
HASIL DAN PEMBAHASAN

Dalam Penelitian "Pendeteksian Citra Dataset yang digunakan untuk pelatihan, validasi, dan pengujian pada model CNN merupakan citra yang mengandung api dan tidak mengandung api yang berasal dari github, Mivia, dan google yang dikumpulkan dan dibagi menjadi 3 kategori tersebut. Setelah itu, data uji yang digunakan untuk melakukan pengujian secara langsung pada aplikasi menggunakan data streaming dari video *live* oleh kamera *webcam* laptop dan CCTV. Contoh data pengujian dapat dilihat pada **lampiran 1 tabel 1 sampai 3.**

4.1 Metode Pengujian

Data set yang sudah terkumpul akan dinormalisasi dengan menggunakan fungsi *pre-processing* yang ada di *library* tensorflow keras. Setelah itu akan dilakukan pembuatan model CNN dengan arsitektur mobilenet. Setelah model CNN di download dari web maka akan dilakukan proses pelatihan menggunakan data latih dan data validasi, dan pengujian dengan data uji. Model dibuat dengan memanggil fungsi yang ada pada *library* tensorflow keras, dengan begitu model memiliki arsitektur mobilenet yang telah dilatih oleh google. Hasil pelatihan dan pengujian yang memuaskan akan disimpan untuk digunakan pada program **aplikasi** pendeteksi api. Setelah bobot hasil akhir telah didapat, akan dibuat tampilan antarmuka dengan menggunakan Python untuk menjalankan program aplikasi.**Pada Model** Arsitektur yang diganti adalah pada bagian *fully connected layer* dan *output layer* agar dapat menghasilkan bobot untuk dua kelas yang dibutuhkan (secara default terdapat seribu kelas). Citra yang diperoleh akan menjadi input kedalam model ini. Bentuk 18 layer terakhir model dapat dilihat pada gambar 12.

conv_pw_11_bn (BatchNormaliz	(None, 14, 14, 512)	2048
conv_pw_11_relu (ReLU)	(None, 14, 14, 512)	0
conv_pad_12 (ZeroPadding2D)	(None, 15, 15, 512)	0
conv_dw_12 (DepthwiseConv2D)	(None, 7, 7, 512)	4608
conv_dw_12_bn (BatchNormaliz	(None, 7, 7, 512)	2048
conv_dw_12_relu (ReLU)	(None, 7, 7, 512)	0
conv_pw_12 (Conv2D)	(None, 7, 7, 1024)	524288
conv_pw_12_bn (BatchNormaliz	(None, 7, 7, 1024)	4096
conv_pw_12_relu (ReLU)	(None, 7, 7, 1024)	0
conv_dw_13 (DepthwiseConv2D)	(None, 7, 7, 1024)	9216
conv_dw_13_bn (BatchNormaliz	(None, 7, 7, 1024)	4096
conv_dw_13_relu (ReLU)	(None, 7, 7, 1024)	0
conv_pw_13 (Conv2D)	(None, 7, 7, 1024)	1048576
conv_pw_13_bn (BatchNormaliz	(None, 7, 7, 1024)	4096
conv_pw_13_relu (ReLU)	(None, 7, 7, 1024)	0
global_average_pooling2d (Gl	(None, 1024)	0
dropout (Dropout)	(None, 1024)	0
dense (Dense)	(None, 2)	2050

Gambar 12 Bentuk 18 Layer Model Akhir

4.2 Pengujian

Pengujian aplikasi dibagi menjadi dua bagian, yaitu pengujian pada bagian *frontend* dan pengujian pada bagian *backend*. Pengujian pada bagian *frontend* akan berfokus pada fungsionalitas dari tampilan antar muka yang telah dibuat. Pada bagian *backend* akan berfokus pada hasil pendeteksian api dengan menggunakan CNN yang telah dibangun.

4.2.1 Pengujian Backend

Pengujian *backend* merupakan pengujian pada model CNN yang telah dibuat dalam 3 tahap, yaitu pengujian hyperparameter model pada saat pelatihan, pengujian hasil pelatihan dengan data yang telah dikumpulkan dari internet, dan pengujian model melalui tampilan antarmuka yang telah dibuat. Pengujian dijalankan untuk model yang telah dilatih dengan data citra yang mengandung api dan tidak mengandung api, masing-masing berjumlah 615 citra latih dan 260 citra validasi.

Pengujian pertama dilakukan dengan mengubah hyperparameter model yaitu epoch, *learning rate,* batch, dan *optimizer* pada saat pelatihan sampai menemukan yang terbaik. Pengujian kedua dengan membuat model untuk mendeteksi 200 citra yang telah disiapkan dari internet dan memperlihatkan hasilnya dalam *confution matrix*. Pengujian ketiga dilakukan dengan menggunakan program aplikasi dari tampilan antarmuka yang telah dibuat dan meletakkan api di depan kamera secara langsung untuk melihat apakah aplikasi dapat mendeteksi adanya api di depan kamera. Api akan dikatakan terdeteksi jika aplikasi dapat mendeteksinya dari video selama 5 detik tanpa berhenti.

4.2.2 Pengujian Frontend

Pengujian *frontend* yang akan dilakukan adalah melakukan pengujian terhadap tampilan antarmuka yang telah dibuat. Pengujian ini akan dijalankan dengan metode *Black Box Testing* yang memastikan bahwa setiap fungsi berjalan dengan baik, pengguna mendapatkan response yang semestinya, dan aplikasi berjalan sesuai yang diinginkan.

4.3 Hasil Pengujian

4.3.1 Hasil Pengujian Backend

Pengujian *hyperparameter* dilakukan untuk menemukan model CNN dengan akurasi tertinggi pada saat pelatihan. Hasil pengujian *hyperparameter* dapat dilihat pada **Lampiran 2 Tabel 4.** *Confution matrix* hasil pengujian dapat dilihat pada **Lampiran 3 Tabel 5.** Berdasarkan hasil *confution matrix* tersebut, maka tingkat akurasi model adalah sebesar 97%.

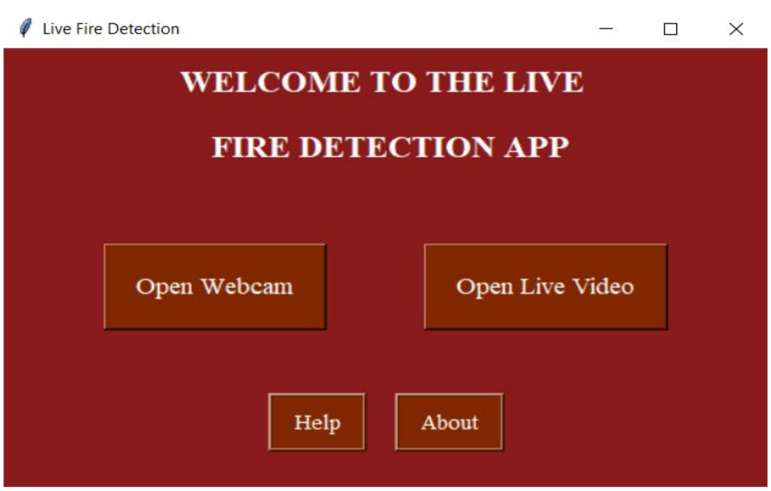
Setelah itu digunakan *optimizer* SGD, 0.1 *learning rate*, 32 *batch*, dan 100 epoch pada pengujian berikutnya. Pengujian kedua dilakukan untuk melihat tingkat akurasi model terhadap citra diluar dataset pelatihan yang hasilnya dapat dilihat pada **Lampiran 4 Tabel 6**. Hasil pengujian kedua dimasukkan kedalam *confution matrix* yang dapat dilihat pada **Lampiran 5 Tabel 7.** Berdasarkan hasil *confution matrix* tersebut, maka tingkat akurasi model adalah sebesar 84%.

4.3.2 Hasil Pengujian Frontend

Pengujian aplikasi dilakukan dengan menggunakan Black Box Testing pada modul aplikasi yang telah dibuat. Tujuan black box testing adalah untuk melihat jika terjadi suatu kesalahan pada aplikasi yang telah dibangun. Black box testing membantu pembuat aplikasi untuk memastikan bahwa semua output telah berjalan dengan semestinya. Hasil pengujian *frontend* aplikasi sebagai berikut :

1. Modul Home

   Pada modul ini terdapat tombol *Open Webcam* untuk secara langsung membuka kamera webcam pada laptop, tombol *Open Live Video* untuk navigasi menuju modul URL, tombol *help* untuk navigasi menuju modul *help,* dan tombol *about* untuk navigasi menuju modul about. Hasil tampilan akhir modul home dapat dilihat pada **gambar 13.**



Gambar 13 Tampilan Modul *Home*

2. Modul Detect

Modul *Detect* adalah tempat dimana video dari kamera yang dipilih diputar dan dideteksi oleh CNN. Pengguna yang memiliki *webcam* dan menekan tombol *Open Webcam* atau pengguna yang menekan tombol *Open Live Video* dan memasukkan URL dengan benar akan membuka modul ini. Hasil deteksi, lama deteksi per frame, dan *frame per second* jalannya video juga ditampilkan pada modul ini, dan jika terdeteksi api selama 5 detik maka aplikasi akan mengeluarkan notofikasi berupa suara alarm. Hasil akhir modul *detect* dapat dilihat pada **Gambar 14**.

Gambar 14 Tampilan Modul *Detect*

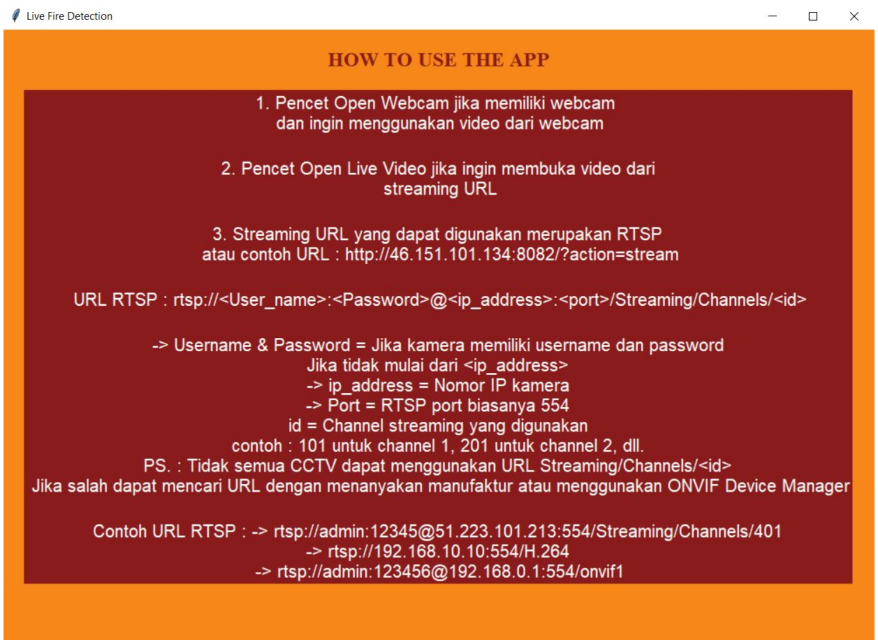
3. Modul URL

Modul URL merupakan tempat pengguna dapat memasukkan URL kamera yang ingin diakses dan dideteksi. Pengguna dapat membuka modul ini dengan menekan tombol *Open Live Video* pada modul *home*. Pengguna yang memasukkan URL kamera dengan benar pada modul ini dan menekan tombol *detect fire* akan membuka modul *Detect.* Tampilan modul URL dapat dilihat pada **gambar 15**.

Gambar 15 Tampilan Modul URL

4. Modul Help

   Modul *Help* merupakan tempat ditaruhnya instruksi untuk menggunakan aplikasi. Tampilan akhir modul *help* dapat dilihat pada **gambar 16**.



Gambar 16 Tampilan Modul *Help*

4.4 Pembahasan

1. Akurasi model CNN yang digunakan pada Program Aplikasi Pendrteksi Api ini dengan menggunakan data asli yang dibuat sendiri sebesar 84% seperti pada **Lampiran 4 Tabel 6** yang lebih rendah dibandingkan saat menggunakan data yang berasal dari internet yaitu sebesar 97% yang dapat dilihat pada **Lampiran 3 Tabel 5**. Hal ini disebabkan karena perbedaan noise dari citra yang didapat dari internet dan citra asli yang berasal dari kamera CCTV dan webcam. Pada pengujian *backend* menggunakan CCTV, jika terlalu banyak pergerakan atau noise pada video dapat dikira ada api oleh aplikasi. Selain itu, Api yang berasal dari lilin seperti yang apinya statis dan terletak sedikit jauh sulit untuk dideteksi sedangkan api lilin yang lebih dekat seperti lebih mudah untuk dideteksi.

2. Adam *optimizer* menghasilkan akurasi yang tidak baik dan tidak stabil disebabkan karena menggunakan *learning rate* yang dari awal besar dan dengan ditambahkan momentum dari adam sendiri, maka *learning rate* yang digunakan menjadi terlalu besar dan menyebabkan adam sulit untuk mendapat parameter pelatihan yang optimal. Dapat dilihat pada **Lampiran 2 Tabel 4** walaupun memiliki tingkat akurasi yang besar, memiliki juga loss yang sangat tinggi tidak seperti semestinya dimana tingkat akurasi tinggi dan loss yang kecil.

BAB 5
KESIMPULAN

Dari hasil pengujian dan pembahasan dapt disimpulkan Sistem Aplikasi Pendeteksi Api yang telah dirancang dapat berjalan dengan baik, karena sudah dapat menerima video *streaming* dari *webcam* ataupun CCTV yang dapat mendeteksi adanya api pada video tersebut dengan tingkat akurasi sebesar 97%. Tetapi, Program Aplikasi Pendeteksi Api belum berhasil dengan baik jika terdapat terlalu banyak noise pada video, letak api yang terlalu jauh dan ukuran api kecil, terdapat objek besar yang bercahaya terang di depan cctv, dan jika latar belakang api yang terlalu terang sehingga api sulit terlihat.

DAFTAR PUSTAKA

[1]. Nicholas True, Computer Vision Based Fire Detection , https://cseweb.ucsd.edu/classes/wi09/ cse190-a/reports/ntrue.pdf, 11 Februari 2021.

[2]. Byongjun Kim dan Joonwhoan Lee, A Vision-Based Fire Detection Using Deep Learning Models, https://www.researchgate.net/publication/334570785_A_Video-Based_Fire_Detection_Using_ Deep_Learning_Models , 11 Februari 2021.

[3]. Li-Wei Kang et al. Image-based Real-Time Fire Detection Using Deep Learning with Data Augmentation for Vision-based Surveillance Applications, https://ieeexplore.ieee.org/ document/8909899, 11 Februari 2021.

[4]. Computer Hope, Video, https://www.computerhope.com/jargon/v/video.htm, 19 Februari 2021.

[5]. TechTerms, FPS, https://techterms.com/definition/fps, 16 Februari 2021.

[6]. Julia Kuzmenko McKim, Understanding Image Noise, https://retouchingacademy.com/qualities-of-digital-images-understanding-image-noise/, 25 Februari 2021.

[7]. Samuel Sena, Pengenalan Deep Learning Part 1 : Neural Network , https://medium.com/@samuelsena/pengenalan-deep-learning-8fbb7d8028ac , 11 Februari 2021

[8]. Google, Machine Learning Glossary , https://developers.google.com/machine-learning/glossary?hl=id#s , 24 Februari 2021

[9]. Devin Soni , Supervised vs. Unsupervised Learning, https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d, 11 Februari 2021.

[10]. Samuel Sena, Pengenalan Deep Learning Part 7 : Convolutional Neural Network (CNN) , https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94, 11 Februari 2021.

[11]. Cs231n , Convolutional Neural Networks for Visual Recognition, https://cs231n.github.io/convolutional-networks/, 11 Februari 2021.

[12]. Samuel Sena, Pengenalan Deep Learning Part 7 : Convolutional Neural Network (CNN) , https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94, 12 Februari 2021.

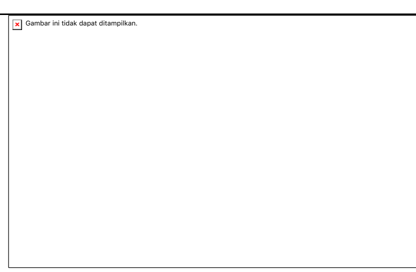
Lampiran 1

Contoh data set

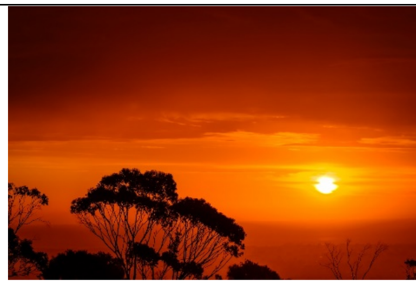
Tabel 1 Contoh Data Latih dari Internet

No.	Citra	Status
1.		Ada api
2.		Ada api
3.		Ada api
4.		Ada api

Tabel 1 (lanjutan)

No.	Citra	Status
5.		Ada api
6.		Ada api
7.		Ada api
8.		Tidak ada api
9.		Tidak ada api

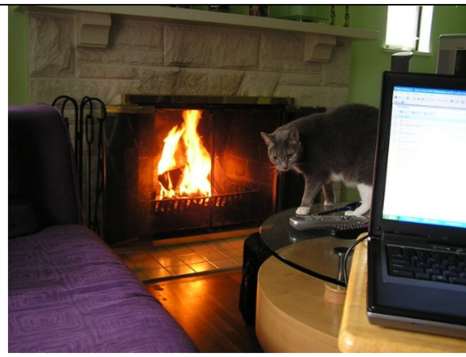
Tabel 1 (lanjutan)

No.	Citra	Status
10.		Tidak ada api

Tabel 2 Contoh Data Validasi dari Internet

No.	Citra	Status
1.		Ada api
2.		Ada api
3.		Ada api

Tabel 2 (lanjutan)

No.	Citra	Status
4.		Ada api
5.		Ada api
6.		Ada api
7.		Ada api

Tabel 2 (lanjutan)

No.	Citra	Status
8.		Tidak ada api
9.		Tidak ada api
10.		Tidak ada api

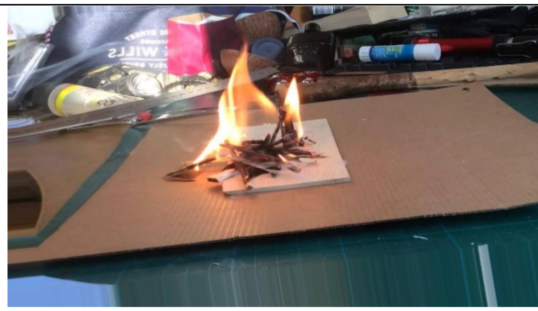
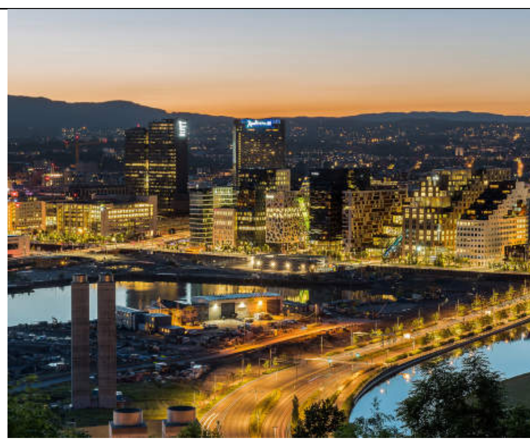
Tabel 3 Contoh Data Uji dari Internet

No.	Citra	Status
1.		Ada api

Tabel 3 (lanjutan)

No.	Citra	Status
2.		Ada api
3.		Ada api
4.		Ada api
5.		Ada api

Tabel 3 (lanjutan)

No.	Citra	Status
6.		Ada api
7.		Ada api
8.		Tidak ada api
9.		Tidak ada api

Tabel 3 (lanjutan)

No.	Citra	Status
10.		Tidak ada api

Lampiran 2
Hasil pengujian Hyperparameter

Tabel 4 Hasil Pengujian Hyperparameter

No.	Optimizer	Epoch	Learning Rate	Batch	Train Acc.	Train Loss	Val Acc.	Val Loss
1.	Adam	100	0.1	32	0.937	0.171	0.75	1.329
2.	Adam	100	0.4	32	0.502	0.71	0.492	0.694
3.	Adam	100	0.5	32	0.963	25.322	0.948	43.55
4.	Adam	100	0.5	64	0.49	0.73	0.492	0.694
5.	Adam	100	0.5	128	0.529	5.148	0.5	2.903
6.	Adam	100	0.6	32	0.5	15.499	0.492	0.7
7.	Adam	150	0.5	32	0.516	2.31	0.492	1.999
8.	Adam	200	0.5	32	0.482	1.992	0.492	1.123
9.	SGD	100	0.1	32	0.999	1.17e-04	0.978	0.13
10.	SGD	100	0.1	64	0.998	0.0031	0.973	0.14
11.	SGD	100	0.1	128	0.998	0.0039	0.959	0.217
12.	SGD	100	0.3	32	0.999	2.0e-04	0.965	0.296
13.	SGD	100	0.5	32	0.999	3.78e-04	0.969	0.479
14.	SGD	150	0.1	32	0.995	0.0143	0.975	0.154
15.	SGD	200	0.1	32	1	1.33e-04	0.967	0.283

Lampiran 3

Confution Matrix hasil pengujian Backend

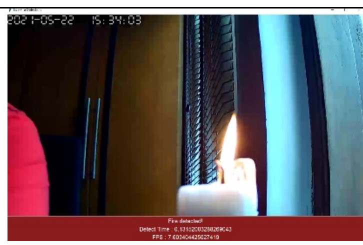
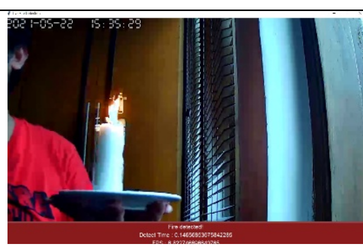
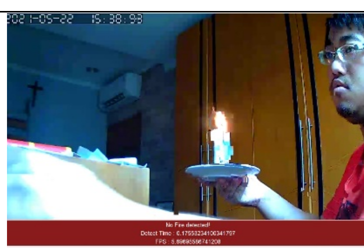
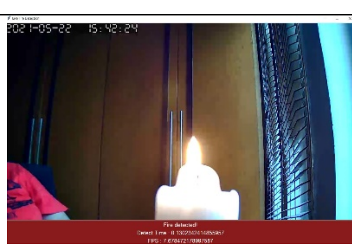
Tabel 5 *Confution Matrix* hasil pengujian Backend

Akurasi = 97%	**Hasil Deteksi**	
Citra Sebenarnya	Ada Api	Tidak ada Api
Ada Api	97	3
Tidak ada Api	3	97

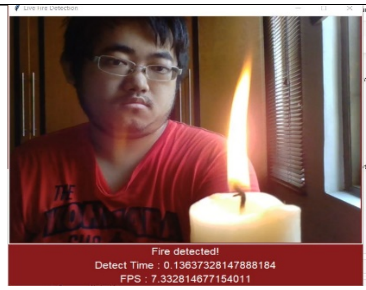
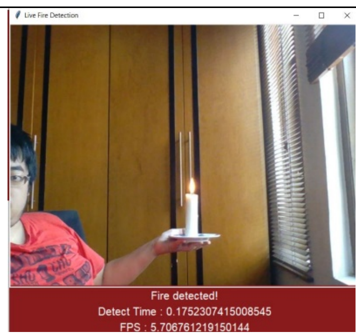
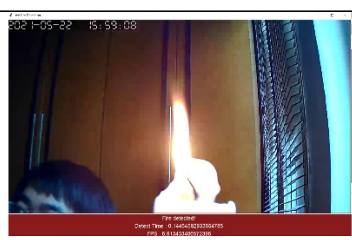
Lampiran 4

Hasil Pengujian ProgramAplikasi

Tabel 6 Contoh Hasil Pengujian Deteksi pada Program Aplikasi

No	Citra Uji	Hasil yang Diharapkan	Hasil Pengujian
1		Ada api	Tidak ada api
2		Ada api	Ada api
3		Ada api	Ada api
4		Ada api	Ada api
5		Ada api	Tidak ada api
6		Ada api	Ada api

Tabel 6 (lanjutan)

No	Citra Uji	Hasil yang Diharapkan	Hasil Pengujian
7		Ada api	Ada api
8		Ada api	Tidak ada api
9		Ada api	Ada api
10		Ada api	Ada api

Lampiran 5

Confution Matrix Hasil pengujian pada Program Aplikasi

Tabel 7 Confution Matrix Hasil pengujian pada Program Aplikasi

Akurasi = 84%	**Hasil Deteksi**	
Citra Sebenarnya	Ada api	Tidak ada api
Ada api	39	11
Tidak ada api	5	45