# DETEKSI API PADA VIDEO MENGGUNAKAN METODE ARTIFICIAL NEURAL NETRWORK

Tugas akhir

untuk memenuhi sebagian persyaratan

mencapai derajat Sarjana S-1 Program Studi Teknik Informatika


**Oleh :**

**BUDIMAN RABBANI**
**F1D 016 018**



**PROGRAM STUDI TEKNIK INFORMATIKA**
**FAKULTAS TEKNIK**
**UNIVERSITAS MATARAM**

**2020**

# PRAKATA

Puji syukur penulisan panjatkan ke hadirat Allah SWT, Tuhan semesta alam, karena dengan berkat, rahmat dan limpahan karunianya sehingga penulis dapat menyelesaikan Laporan Tugas Akhir ini sebagaimana mestinya.

Adapun Tugas Akhir (TA) ini penulis membangun sebuah model *artificial neural network backpropagation* untuk mendeteksi api dengan judul " Deteksi Api Pada Video menggunakan Metode *Artificial Neural Network* ". Model ini nantinya akan dijadikan acuan untuk pembuatan sistem untuk mendeteksi api. Dalam pembuatan laporan ini penulis berpedoman pada bahan kuliah, petunjuk dari pembimbing utama, pembimbing pendamping, referensi dan literatur yang terkait dengan penulisan laporan.

Penulis menyadari bahwa "Tak ada Gading yang Tak Retak " begitu pula dengan laporan ini masih jauh dari kesempurnaan. Oleh karena itu, penulis mengharapkan kritik serta saran yang bersifat membangun agar dapat menghasilkan karya yang lebih baik dimasa mendatang. Semoga laporan ini dapat bermanfaat bagi penulis dan pembaca sekalian.

Mataram, 2020

Penulis,

# UCAPAN TERIMA KASIH

Pada kesempatan ini penulis tidak lupa mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Ibu, Bapak, dan Saudara-Saudara penulis yang telah memberikan dukungan baik material maupun doa kepada penulis.
2. Bapak Prof. Dr.Eng. I Gede Pasek Suta Wijaya, S.T.,M.T. Selaku Dosen Pembimbing 1.
3. Bapak Ramaditia Dwiyansaputra, ST., M.Eng. selaku Dosen Pembimbing 2.
4. Semua pihak yang telah membantu penulis yang tidak dapat penulis sebutkan satu persatu yang telah membantu selama pelaksanaan TA ini.

Semoga Tuhan Yang Maha Esa memberikan imbalan yang setimpal atas bantuan yang diberikan kepada penulis.

**TUGAS AKHIR**

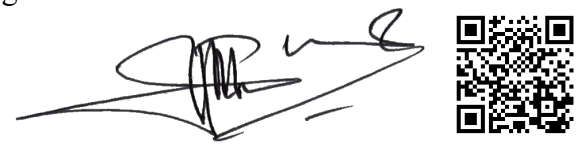
# DETEKSI API PADA VIDEO MENGGUNAKAN METODE ARTIFICIAL NEURAL NETRWORK

Oleh :

**BUDIMAN RABBANI**
F1D016018

Telah diperiksa oleh Tim Pembimbing :

1. Pembimbing Utama



Tanggal: 28/07/2020

**Prof. Dr. Eng, I Gede Pasek Suta Wijaya, ST., MT.**
**NIP. 197311302000031001**

2. Pembimbing Pendamping

Tanggal: 27/07/2020

**Ramaditia Dwiyansaputra, S.T., M.Eng.**
**NIP. -**

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Teknik
Universitas Mataram

**Prof. Dr. Eng. I Gede Pasek Suta Wijaya, ST., MT.**
**NIP. 197311302000031001**

**TUGAS AKHIR**

# DETEKSI API PADA VIDEO MENGGUNAKAN METODE ARTIFICIAL NEURAL NETRWORK

Oleh :

**BUDIMAN RABBANI**

F1D016018

Telah diujikan di depan penguji
Pada tanggal 22 Juli 2020
Dan dinyatakan telah memenuhi syarat mencapai derajat Sarjana S-1
Program Studi Teknik Informatika

Susunan Tim Penguji :

1. Penguji 1


Tanggal: 24/07/2020

**Arik Aranta, S.Kom., M.Kom.**
**NIP. 199402202019031004**

2. Penguji 2


Tanggal: 27/07/2020

**Fitri Bimantoro, ST.,M.Kom.**
**NIP. 198606222015041002**

3. Penguji 3


Tanggal: 26/07/2020

**Ahmad Zafrullah M., S.T., M.Eng.**
**NIP.**

Mataram, 28 Juli 2020
Dekan Fakultas Teknik
Universitas Mataram

**Akmaluddin, ST., M.Sc Eng., Ph.D.**
**NIP. 196812311994121001**

## DAFTAR ISI

**HALAMAN AWAL** ..... **i**

**PRAKATA** ..... **ii**

**UCAPAN TERIMA KASIH** ..... **iii**

**LEMBAR PENGESAHAN** ..... **iv**

**DAFTAR ISI** ..... **iv**

**DAFTAR TABEL** ..... **ix**

**DAFTAR GAMBAR** ..... **x**

**DAFTAR LAMPIRAN** ..... **xii**

**INTISARI** ..... **xiiii**

**ABSTRACT** ..... **xivi**

**BAB I PENDAHULUAN** ..... **1**

1.1 Latar Belakang ..... 1

1.2 Rumusan Masalah ..... 2

1.3 Batasan Masalah ..... 2

1.4 Tujuan ..... 3

1.5 Manfaat ..... 3

1.6 Sistematika Penulisan ..... 3

**BAB II TUNJAUAN PUSTAKA** ..... **5**

2.1 Tinjauan Pustaka ..... 5

2.2 Dasar Teori ..... 7

2.2.1 Sampel ..... 7

2.2.2 Api ..... 8

2.2.3 Kecerdasan Buatan ..... 8

2.2.4 *Gray-Level Co-Occurrence Matrix* ..... 9

2.2.5 *Local Binary Pattern* ..... 10

2.2.6 Konsep Dasar Jaringan Saraf Tiruan ..... 11
2.2.7 Arsitektur Jaringan Syaraf Tiruan ..... 12
2.2.8 Fungsi Aktivasi *Backpropagation* ..... 13
2.2.9 Pelatihan Standar *Backpropagation* ..... 14
**BAB III METODE PENELITIAN ..... 16**
3.1 Bahan dan Alat Penelitian ..... 16
3.2 Rencana Penelitian ..... 17
3.3 Rancangan Model Deteksi Apia ..... 18
3.3.1 Tahap *Prepocessing* ..... 18
3.3.2 Extraction *Feature* ..... 20
3.3.3 Klasifikasi ..... 24
3.4 Teknik Pengujian ..... 30
3.4.1 Skenario Pengujian ..... 30
3.5 Jadwal Kegiatan ..... 32
**BAB IV HASIL DAN PEMBAHASAN ..... 33**
4.1 Pengumpulan Data ..... 33
4.2 Pengujian ..... 33
4.2.1 Pengaruh Variasi Pembagian Dataset ..... 34
4.2.2 Pengaruh Batas Error Terhadap Unjuk Kerja ANN ..... 34
4.2.3 Pengaruh Batas *Epoch* Terhadap Unjuk Kerja ANN ..... 35
4.2.4 Pengaruh *Learning Rate* ..... 35
4.2.5 Pengaruh Variasi Jumlah *Hidden Layer* ..... 36
4.2.6 Pengujian Nilai Ambang (*Threshold*) ..... 37
4.2.7 Validasi ..... 38
**BAB V KESIMPULAN DAN SARAN ..... 40**
**5.1 Kesimpulan** ..... 40

**5.2 Saran**..... 40

**DAFTAR PUSTAKA.....41**

**LAMPIRAN .....43**

# DAFTAR TABEL

Tabel 2.1 Tabel fitur tekstur GLCM. ..... 6
Tabel 3. 1 Contoh data *frame* video ..... 14
Tabel 3. 2 Data *dummy* ..... 22
Tabel 3. 3 Bias dan bobot akhir dari *input layer* ke *hidden layer* pertama ..... 27
Tabel 3. 4 Bias dan bobot akhir dari *hidden layer* pertama ke *output layer* ..... 27
Tabel 3. 5 *Output* data latih ..... 28
Tabel 3. 6 Confusion matrix 3x3. ..... 29
Tabel 3. 7 Hasil deteksi ..... 30
Tabel 3. 8 Jadwal kegiatan perancangan sistem ..... 31
Tabel 4. 1 Akurasi terbaik variasi pembagaian dataset ..... 33
Tabel 4. 2 Akurasi terbaik variasi *error* ..... 33
Tabel 4. 3 Akurasi terbaik variasi batas *epoch* ..... 34
Tabel 4. 4 Akurasi terbaik variasi *learning rate* ..... 34
Tabel 4. 5 Hidden layer arsitektur pengujian ..... 35
Tabel 4. 6 Hasil pengujian arsitektur jaringan ..... 35
Tabel 4. 7 Hasil pengujian arsitektur jaringan ..... 36
Tabel 4. 8 Validasi ..... 37

# DAFTAR GAMBAR

Gambar 2. 1 Contoh gambar adanya objek api ..... 5
Gambar 2. 2 Penerapan Konsep Kecerdasan Buatan di Komputer ..... 6
Gambar 2. 3 Arsitektur Lapisan Tunggal ..... 10
Gambar 2. 4 Arsitektur Lapisan *Multilayer* ..... 11
Gambar 2. 5 Fungsi Aktivasi *Sigmoid Biner* ..... 11
Gambar 2. 6 Fungsi Aktivasi *Sigmoid Bipolar* ..... 12

## DAFTAR LAMPIRAN


Tabel 1 Pembagian dataset 70:30, toleransi0.001,epoch 100 ..... 43
Tabel 2 Pembagian dataset 70:30, toleransi 0.001,epoch 200 ..... 43
Tabel 3 Pembagian dataset 70:30, toleransi 0.001,epoch 300 ..... 43
Tabel 4 Pembagian dataset 70:30, toleransi 0.001,epoch 400 ..... 44
Tabel 5 Pembagian dataset 70:30, toleransi 0.001,epoch 500 ..... 44
Tabel 6 Pembagian dataset 70:30, toleransi 0.0001,epoch 100 ..... 44
Tabel 7 Pembagian dataset 70:30, toleransi 0.0001,epoch 200 ..... 44
Tabel 8 Pembagian dataset 70:30, toleransi 0.0001,epoch 300 ..... 45
Tabel 9 Pembagian dataset 70:30, toleransi 0.0001,epoch 400 ..... 45
Tabel 10 Pembagian dataset 70:30, toleransi 0.0001,epoch 500 ..... 45
Tabel 11 Pembagian dataset 70:30, toleransi 0.00001,epoch 100 ..... 45
Tabel 12 Pembagian dataset 70:30, toleransi 0.0001,epoch 400 ..... 46
Tabel 13 Pembagian dataset 70:30, toleransi 0.00001,epoch 300 ..... 46
Tabel 14 Pembagian dataset 70:30, toleransi 0.00001,epoch 400 ..... 46
Tabel 15 Pembagian dataset 70:30, toleransi 0.00001,epoch 500 ..... 46
Tabel 16 Pembagian dataset 80:20, toleransi 0.001,epoch 100 ..... 47
Tabel 17 Pembagian dataset 80:20, toleransi 0.001,epoch 200 ..... 47
Tabel 18 Pembagian dataset 80:20, toleransi 0.001,epoch 300 ..... 47
Tabel 19 Pembagian dataset 80:20, toleransi 0.001,epoch 400 ..... 47
Tabel 20 Pembagian dataset 80:20, toleransi 0.001,epoch 500 ..... 47
Tabel 21 Pembagian dataset 80:20, toleransi 0.0001,epoch 100 ..... 48
Tabel 22 Pembagian dataset 80:20, toleransi 0.0001,epoch 200 ..... 48
Tabel 23 Pembagian dataset 80:20, toleransi 0.0001,epoch 300 ..... 48
Tabel 24 Pembagian dataset 80:20, toleransi 0.0001,epoch 400 ..... 49
Tabel 25 Pembagian dataset 80:20, toleransi 0.0001,epoch 500 ..... 49
Tabel 26 Pembagian dataset 80:20, toleransi 0.00001,epoch 100 ..... 49
Tabel 27 Pembagian dataset 80:20, toleransi 0.00001,epoch 200 ..... 49
Tabel 28 Pembagian dataset 80:20, toleransi 0.00001,epoch 300 ..... 50
Tabel 29 Pembagian dataset 80:20, toleransi 0.00001,epoch 400 ..... 50
Tabel 30 Pembagian dataset 80:20, toleransi 0.00001,epoch 500 ..... 50
Tabel 31 Pembagian dataset 90:20, toleransi 0.001,epoch 100 ..... 50

Tabel 32 Pembagian dataset 90:20, toleransi 0.001,epoch 200 ..... 51
Tabel 33 Pembagian dataset 90:20, toleransi 0.001,epoch 300 ..... 51
Tabel 34 Pembagian dataset 90:20, toleransi 0.001,epoch 400 ..... 51
Tabel 35 Pembagian dataset 90:20, toleransi 0.001,epoch 500 ..... 51
Tabel 36 Pembagian dataset 90:20, toleransi 0.0001,epoch 100 ..... 52
Tabel 37 Pembagian dataset 90:20, toleransi 0.0001,epoch 200 ..... 52
Tabel 38 Pembagian dataset 90:20, toleransi 0.0001,epoch 300 ..... 52
Tabel 39 Pembagian dataset 90:20, toleransi 0.0001,epoch 400 ..... 52
Tabel 40 Pembagian dataset 90:20, toleransi 0.0001,epoch 500 ..... 52
Tabel 41 Pembagian dataset 90:20, toleransi 0.00001,epoch 100 ..... 53
Tabel 42 Pembagian dataset 90:20, toleransi 0.00001, epoch 200 ..... 53
Tabel 43 Pembagian dataset 90:20, toleransi 0.00001,epoch 300 ..... 53
Tabel 44 Pembagian dataset 90:20, toleransi 0.00001,epoch 400 ..... 53
Tabel 45 Pembagian dataset 90:20, toleransi 0.00001,epoch 100 ..... 54
Tabel 46 Bobot dan Bias dari input layer ke hidden layer 1..... 54
Tabel 47 Bobot dan Bias dari hidden layer 1 ke output layer..... 55
Tabel 48 Jumlah frame yang terdeteksi non-api pada video jenis non-api..... 55
Tabel 49 Jumlah *frame* yang terdeteksi api pada video jenis api ..... 55

# INTISARI

Kamera merupakan salah satu alat yang digunakan untuk menangkap gambar. Kamera sering kali dimanfaatkan sebagai alat keamanan seperti keamanan rumah, jalan raya dan lain-lain. Kemudian pada penelitian ini digunakan hasil tangkapan kamera untuk mendeteksi objek api karena api merupakan salah satu penyebab kebakaran apabila tak dapat dikendalikan. Oleh karena itu, dengan memanfaatkan tangkapan kamera akan dilihat model yang terbaik *artificial neural network backpropagation* dengan penggabungan metode ekstraksi fitur gabungan *local binary patern* (LBP) dan *Gray Level Co-occurrence Matrix* (GLCM) untuk mendeteksi api. Kemudian untuk mengevaluasi kinerja dari model yang dibuat digunakan tiga buah parameter yakni akurasi, *recall*, *precision*. Dataset dalam penelitian berupa video dengan variasi video api dan non-api. Berdasarkan hasil akhir penelitian, akurasi, *recall*, *precision* terbaik yang diperoleh berturut-turut adalah 96%, 97%, 97%. Kemudian dilakukan proses validasi menggunakan 30 video dengan variasi 15 video api dan 15 video non-api dan diperoleh hasil akurasi sebesar 86.6% dengan rata-rata waktu sebesar 6,029 menit.

**Kata kunci:** Deteksi Api, GLCM, LBP, ANN *Backpropagation*.

**ABSTRACT**

*The camera is one of the tools used to collect images. Cameras are often used for the safety of homes, highways and others. Then in this study camera captures are used to support fire objects because fire is one of the causes of safety that can be controlled. Therefore, by utilizing a capture camera will see the best model of backpropagation neural network by combining the local binary patern (LBP) feature extraction method and the Gray Level Co-occurrence Matrix (GLCM) to access the fire. Then to evaluate the performance of the model created using three parameters that contain accuracy, recall, precision. The data in this study consisted of videos with variations of fire and non-fire videos. Based on the final results of the study, accuracy, remember, the best precision obtained simultaneously 96%, 97%, 97%. Then the validation process was done using 30 videos with a variation of 15 fire videos and 15 non-fire videos and obtained an accuracy of 86.6% with an average time value of 6.029 minutes.*

**Keywords:** *Fire Detection*, GLCM, LBP, ANN *Backpropagation*.

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Api merupakan sebuah hasil dari reaksi kimia antara bahan bakar dan oksigen. Api sering kali dimanfaatkan oleh masyarakat untuk memasak, penerangan ruangan dan lain sebagainya. Namun api juga berpotensi menimbulkan bahaya jika api tidak dapat dikontrol dengan baik sehingga dapat menyebabkan kebakaran.

Kebakaran hutan merupakan hal yang selalu menjadi sorotan di Indonesia, pasalnya dari tahun ke tahun kebakaran selalu terjadi di beberapa wilayah Indonesia. Tercatat pada website resmi BNPB (Badan Nasional Penanggulangan Bencana) bahwa jumlah kasus kebakaran cukup banyak disetiap tahunnya. Selama 3 tahun ini tercatat kasus tertinggi terjadi pada tahun 2018. Pada tahun 2017 tercatat ada 96 kasus, 2018 tercatat ada 527 kasus dan 2019 tercatat ada 150 kasus kebakaran hutan dan lahan[1]. Kebakaran juga kerap terjadi di daerah-daerah padat penduduk yang umumnya terjadi akibat hubungan singkat arus listrik, kebocoran gas LPG atau kelalaian yang dilakukan manusia seperti lupa mematikan kompor, lupa mematikan lilin dan sebab lain yang dapat menimbulkan terjadinya kebakaran. Banyak kasus kebakaran yang terlambat dideteksi dari awal sehingga menimbulkan korban jiwa, kerugian materil, serta kematian flora dan fauna.

Kecerdasan buatan merupakan salah satu bentuk maju nya teknologi, dimana dengan kecerdasan buatan teknologi dapat yang meniru kecerdasan manusia yang dapat mengenali objek-objek seperti api, jenis kelamin dan lain sebagainya. Kecerdasan juga dapat meniru kecerdasan manusia untuk menyelesaikan masalah-masalah dengan cepat seperti contoh pendeteksian api yang dapat menyebabkan terjadinya kebakaran. Saat ini telah dikembangkan sensor-sensor untuk mendeteksi kebakaran seperti sensor yang bekerja untuk mendeteksi api. Dimana sensor bekerja untuk mendeteksi api bergantung pada karakteristik tertentu seperti suhu dan asap. Akan tetapi kemampuan sensor-sensor tersebut memiliki kelemahan seperti sensor api kurang cukup menjangkau area yang luas dan sensor asap yang bergantung pada arah mata angin. Hal tersebutlah yang menyebabkan sensor tidak ideal untuk mendeteksi kebakaran dini [2].

Kamera merupakan suatu alat yang memiliki banyak fungsi. Kamera saat ini banyak digunakan karena kemampuannya yang dapat menangkap gambar sebagaimana fungsi indra penglihatan. Dengan hasil tangkapan kamera ini pernah dilakukan penelitian untuk mendeteksi api. Dimana pada penelitian [3] dan [4] yang menggunakan ruang warna HSV dalam melakukan segmentasi warna api. Pada penelitian [3] kesalahan terutama disebabkan oleh pantulan cahaya kebakaran pada dinding atau permukaan yang mengkilap. Lalu pada penelitian [4], eksperimen menunjukan bahwa algoritma yang digunakan tidak dapat mendeteksi api dikarenakan selalu salah mengidentifikasi lampu sebagai api. Oleh karena itu proses yang hanya menggunakan proses segmentasi belum optimal untuk mendeteksi api.

Lalu selanjutnya penelitian yang tidak sampai hanya pada proses segmentasi warna saja, yakni melakukan proses ekstraksi fitur terhadap citra. Dimana pada penelitian [5] dilakukan proses ekstraksi fitur menggunakan metode *Gray Level Co-occurrence Matrix* (GLCM) dan klasifikasi menggunakan metode *Artificial Neural Network* (ANN) *backpropagation*. Pada penelitian ini didapatkan *error* yang cukup rendah yakni sebesar 7%. Namun peneliti menyarankan agar dilakukannya proses kombinasi metode ektraksi fitur dengan metode *Local Binary Patern* (LBP).

Oleh sebab itu, pada penelitian kali ini dengan memanfaatkan hasil tangkapan gambar dari kamera akan dibangun sebuah model arsitektur *Artificial Neural Network* (ANN) *backpropagation* dengan ekstraksi fitur gabungan antara *Local Binary Patern* (LBP) dan *Gray Level Co-occurrence Matrix* (GLCM) yang dapat mendeteksi api. Dari penelitian ini, diharapkan hasilnya dapat dimanfaatkan untuk pendeteksi api.

## 1.2 Rumusan Masalah

Berdasarkan pada latar belakang yang telah diuraikan, perumusan masalah yang didapat pada tugas akhir ini yaitu Bagaimana cara membangun model arsitektur ANN *backpropagation* terbaik dengan metode ektraksi fitur LBP dan GLCM yang dapat mendeteksi api?

## 1.3 Batasan Masalah

Dari permasalahan yang telah diuraikan di atas terdapat batasan-batasan masalah dalam pembangunan aplikasi ini yaitu:

1. Klasifikasi dibagi menjadi dua kelas yaitu api dan non-api
2. Objek yang dideteksi hanya api

3. Data video memiliki *channel* R,G,B dengan panjang antara 10-60 detik
4. Jarak titik api ke kamera pada video tidak ditentukan
5. Format video yang digunakan .mp4, .avi
6. Resolusi video maksimal 640x640

## 1.4 Tujuan

Adapun tujuan yang akan dicapai dari tugas akhir ini yaitu membangun model ANN *backpropagation* terbaik dengan metode ekstraksi fitur gabungan LBP dan GLCM yang dapat mendeteksi api.

## 1.5 Manfaat

Manfaat yang diperoleh dari tugas akhir ini adalah:

1. Bagi penyusun
   a. Menerapkan ilmu yang telah diperoleh selama proses perkuliahan di Teknik Informatika Universitas Mataram.
   b. Menambah wawasan dalam bidang pengolahan citra digital (pengenalan pola).
2. Bagi pembaca
   a. Menambah ilmu pengetahuan terutama mengenai teknologi pengenalan pola api dalam pengolahan citra digital.
   b. Menerapkan hasil dari skripsi ini pada permasalahan yang berhubungan dengan pengenalan pola api.
   c. Dapat Mengetahui cara mendeteksi kebakaran menggunakan metode ANN.

## 1.6 Sistematika Penulisan

Sistematika penulisan dalam penyusunan tugas akhir ini adalah sebagai berikut:

1. Bab I Pendahuluan

   Bab ini menjelaskan dasar-dasar dari penulisan laporan tugas akhir, yang terdiri dari latar belakang, rumusan masalah, batasan masalah, tujuan, serta sistematika penulisan laporan tugas akhir.
2. Bab II Tinjauan Pustaka dan Dasar Teori

   Bab ini membahas teori-teori yang berhubungan dengan topik penelitian, meliputi Pengenalan Pola, LBP, GLCM dan ANN.
3. Bab III Metodologi Penelitian

Bab ini membahas tentang metodologi yang digunakan dalam penelitian.

4. BAB IV: HASIL DAN PEMBAHASAN

   Pada bab ini akan disajikan hasil dari penelitian serta pembahasan dari hasil penelitian tersebut

5. BAB V : KESIMPULAN DAN SARAN

   Bab ini berisi kesimpulan dari model yang telah dibangun dan digunakan serta saran yang diberikan sebagai bahan acuan dan evaluasi agar model mampu dikembangkan lebih baik lagi.

# BAB II
# TINJAUAN PUSTAKA DAN DASAR TEORI

## 2.1 Tinjauan Pustaka

Penelitian tentang penggunaan *Local Binary Pattern* (LBP) sebagai ekstraksi fitur telah di lakukan oleh beberapa peneliti sebelumnya. Pertama ekstraksi ciri pada telapak tangan dimana prosesnya dimulai dengan proses *pre-processing* yaitu tahap persiapan citra berwarna yang akan dirubah menjadi citra keabuan kemudian dilanjutkan dengan proses *regioning* atau proses pembagian citra menjadi beberapa region,sehingga didapatkan akurasi sebesar 92,31% [6]. Kedua pengenalan pola huruf hiragana dan katakana pada smartphone dengan 460 data uji sehingga didapatkan tingkat akurasi sebesar 81,1% yang merupakan hasil yang cukup besar namun metode ini kurang mampu mengenali perbedaan pola huruf yang kurang signifikan [7]. Ketiga deteksi keaslian mata uang rupiah dengan metode k-NN sebagai metode klasifikasi dengan jumlah total 120 data uji dengan masing – masing 30 set data pecahan 50k asli, 30 set data pecahan 50k palsu dan 30 set data pecahan 100k asli dan 30 set data pecahan 100k palsu sehingga didapatkan tingkat akurasi sebesar 95%[8].

Penelitian tentang penggunaan metode *Gray Level Co-occurrence Matrix* (GLCM) sebagai metode ekstraksinya telah dilakukan oleh beberapa peneliti diantaranya klasifikasi mutu pepaya dengan metode ANN *backpropagation* sebagai metode klasifikasi dengan jumlah data yang digunakan sebanyak 192 data citra dengan rincian 156 data latih yang terdiri dari 52 citra pepaya kelas super, 52 citra pepaya kelas A, dan 52 citra pepaya kelas B, dan data pengujian sebanyak 36 data citra papaya dan didapatkan sebesar 86,11% [9], Klasifikasi jenis daging sapi, kambing dan babi dengan metode k-*Nearest Neighbour* (k-NN) sebagai metode klasifikasi dengan jumlah citra yang digunakan adalah 25 untuk data latih dan 5 citra data uji dari masing-masing jenis daging, sehingga total keseluruhan dataset adalah sebanyak 90 citra dengan tingkat akurasi sebesar 73,3% [10], klasifikasi mutu buah jeruk keprok dengan metode *support vector machine* (SVM) sebagai metode klasifikasi dengan jumlah data sebanyak 100, 60 sebagai data latih dan 40 sebagai data uji dan akurasi sebesar 82,5%[11] dan klasifikasi kayu jati dan mahoni dengan metode jarak Euclidean sebagai metode klasifikasi dengan tingkat akurasi sebesar 82,5% [12].

Penelitian tentang penggunaan penggabungan metode LBP dan GLCM pernah dilakukan oleh beberapa peneliti antara lain pengenalan ekspresi mulut pembelajar untuk mengidentifikasi status ekspresi wajah pembelajar saat menggunakan *e-learning* dengan pengolahan citra digital dengan metode *multiclass* SVM sehingga didapatkan akurasi sebesar 95% [13] dan pengenalan ekspresi wajah pengguna elearning dengan mertode *backpropagation neural network* sebagai metode klasifikasi nya sehingga didapatkan tingkat akurasi sebesar 88,89% [14].

Selanjutnya penelitian tentang penggunaan metode *Artificial Neural Network* (ANN) *backpropagation* sebagai metode pengklasifikasian pernah digunakan oleh beberapa peneliti antara lain klasifikasi kardiotokografi dengan jumlah data sebanyak 36 dan diklasifikasiakan menjadi 3 kelas yaitu kelas normal, kelas *suspect* dan kelas *pathologic* dengan record sebanyak 2126 sehingga didapatkan akurasi sebesar 99,15% [15], klasifikasi daun tanaman *theobroma cacao* dengan jumlah data sebanyak 90 citra dan didapatkan akurasi sebesar 83% daun rusak, 96% daun rusak sedang dan 86% daun sehat[16],

Penelitian mengenai deteksi api menggunakan ruang warna sebagai segmentasi untuk mendeteksi api telah banyak dilakukan sebelumnya. Penelitian yang dilakukan oleh [3], [4] dan [17] yang menggunakan ruang warna dalam melakukan segmentasi warna api. Penelitian [3] menggunakan *Adaptive Gaussian Mixture* Model untuk mendeteksi gerakan dengan statistik warna api pada ruang warna HSV untuk mendeteksi kebakaran. Akurasi sistem terhadap video dengan latar siang hari sebesar 97,6 persen sedangkan pada video dengan latar malam hari sebesar 98,65 persen. Kesalahan terutama disebabkan oleh pantulan cahaya kebakaran pada dinding atau permukaan yang mengkilap. Hasil penelitian menunjukkan bahwa biaya komputasi cukup rendah sehingga bisa dilakukan secara *real-time*. Di penelitian [4], segmentasi pada ruang warna HSV diterapkan pada robot pemadam api dengan hasil akurasi sebesar 80%. Namun eksperimen menunjukkan bahwa algoritma yang digunakan tidak dapat mendeteksi api dikarenakan selalu salah mengidentifikasi lampu sebagai api. Penelitian ini tidak menggunakan *Background Substraction* untuk deteksi gerakan. Selanjutnya di penelitian [17], dimana pada penelitian ini deteksi api dilakukan berbasis spatial temporal dengan metode ektraksi kontur dan *area movement analysis* menghasilkan akurasi tertinggi yaitu 88%. Namun pada penelitian ini dilihat

dari hasilnya, metode ini tidak dapat mendeteksi api jika api berada di area yang banyak cahaya. Oleh karena itu untuk mendeteksi api proses yang hanya menggunakan segmentasi saja tidaklah optimal.

Penelitian mengenai deteksi api yang prosesnya tidak sampai hanya menggunakan metode segmentasi telah banyak dilakukan. Penelitian yang dilakukan oleh [18] dan [5] menggunakan proses ekstraksi fitur dan klasifikasi setelah dilakukannya proses segmentasi. Kedua peneliti ini menggunakan klasifikasi yang sama yakni ANN *backpropagation* namun dengan metode ektraksi fitur yang berbeda. Pada peneliti [18] menggunakan metode ruang warna HSI sebagai metode ektraksi fitur yang menghasilkan akurasi sebesar 96,47%. Lalu oleh [5] menggunakan metode ekstraksi fitur GLCM menghasilkan akurasi sebesar 93%. Namun pada jurnal ini menyarankan untuk dilakukannya kombinasi pada proses ektraksi fitur menggunakan metode GLCM dan LBP.

Berdasarkan beberapa penelitian di atas, maka dapat dilakukan penelitian dengan penggabungan antara metode LBP dan GLCM sebagai ektraksi citra api dan metode ANN *backpropagation* sebagai metode pengklasifikasiannya.

## 2.2 Dasar Teori

### 2.2.1 Sampel

Sugiyono (2011 : 81) menyatakan bahwa sampel adalah bagian dari jumlah dan karakteristik yang dimiliki oleh populasi tersebut. Populasi yang dijadikan sampel pada penelitian yaitu konsumen yang melakukan pembelian dalam kurun waktu di tahun 2014. Karena populasi dari yang menggunakan sepatu *Customade* diketahui maka teknik sampel dalam penelitian ini adalah probability sampling dengan menggunakan sampling acak sederhana (Simple Random Sampling). Berikut pengambiolan sampel dalam pengambilan jumlah sampel penulis menggunakan Slovin [19]:

$$n = \frac{N}{1+Ne^2} \tag{2-1}$$

Keterangan :

n = ukuran sampel

N = ukuran populasi

e = persen kelonggaran ketidak telitian karena kesalahan pengambilan sampel yang masih dapat ditolerir

### 2.2.2 Api

Api dalam Kamus Besar Bahasa Indonesia (KBBI) berati panas dan cahaya yang berasal dari sesuatu yang terbakar. Api adalah sumber energi yang berpotensi dapat menyebabkan kerusakan baik ekonomis dan ekologis [5]. Kebakaran adalah kejadian yang menimbulkan terjadinya api yang tidak terkendali yang dapat membahayakan jiwa maupun harta benda[20]. Api memiliki aturan warna untuk api gelap dan api terang [17]. Nilai piksel api dapat dilihat pada persamaan 2-1 dan 2-2.

$$Piksel\ Api\ Gelap \begin{cases} 179 < R \\ 70 < G < 159 \\ B < 119 \end{cases} \quad (2\text{-}2)$$

$$Piksel\ Api\ Terang \begin{cases} 210 < R \\ 123 < G \\ B < 176 \end{cases} \quad (2\text{-}3)$$

Gambar 2.1 merupakan gambar yang terdapat objek api.

Gambar 2. 1 Contoh gambar adanya objek api

### 2.2.3 Kecerdasan Buatan

Kecerdasan Buatan (*Artificial Intelligence*) merupakan salah satu bagian dari ilmu komputer yang mempelajari bagaimana membuat mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia bahkan bisa lebih baik daripada yang dilakukan manusia[21].

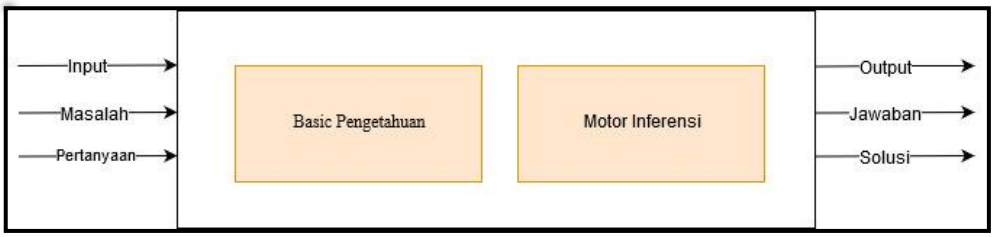
Manusia cerdas (pandai) dalam menyelesaikan permasalahan karena manusia mempunyai pengetahuan dan pengalaman. Pengetahuan diperoleh dari belajar. Semakin banyak bekal pengetahuan yang dimiliki tentu akan lebih mampu menyelesaikan permasalahan. Tapi bekal pengetahuan saja tidak cukup, manusia juga diberi akal untuk melakukan penalaran,mengambil kesimpulan berdasarkan pengetahuan dan pengalaman yang dimiliki. Tanpa memiliki kemampuan untuk menalar dengan baik, manusia dengan segudang pengalaman dan pengetahuan tidak

akan dapat menyelesaikan masalah dengan baik. Demikian juga dengan kemampuan menalar yang sangat baik,namun tanpa bekal pengetahuan dan pengalaman yang memadai, manusia juga tidak akan bisa menyelesaikan masalah dengan baik[21].

Demikin juga agar mesin bisa cerdas (bertindak seperti dan sebaik manusia) maka harus diberi bekal pengetahuan, sehingga mempunyai kemampuan untuk menalar. Untuk membuat aplikasi kecerdasan buatan ada 2 bagian utama yang sangat dibutuhkan :

1. Basis Pengetahuan (*Knowledge Base*), bersifat fakta-fakta, teori , pemikiran dan hubungan antar satu dengan yang lainnya.
2. Motor Inferensi (*Inference Engine*), kemampuan menarik kesimpulan berdasarkan pengetahuan dan pengalaman.

Penerapan Konsep Kecerdasan Buatan pada Komputer dapat dilihat pada Gambar 2.1 berikut :




Gambar 2. 2 Penerapan Konsep Kecerdasan Buatan di Komputer[21]

### 2.2.4 *Gray-Level Co-Occurrence Matrix*

*Gray Level Co-occurrence* Matrix (GLCM) adalah suatu metode yang digunakan untuk analisis tekstur/ekstraksi ciri. GLCM merupakan suatu matriks yang menggambarkan frekuensi munculnya pasangan dua piksel dengan intensitas tertentu dalam jarak dan arah tertentu dalam citra [11].

Koordinat pasangan piksel memiliki jarak d dan orientasi sudut ~~Θ~~. Jarak direpresentasikan dalam piksel dan sudut direpresentasikan dalam derajat. Orientasi sudut terbentuk berdasarkan empat arah sudut yaitu, 0°, 45°, 90° dan 135°, dan jarak antar piksel sebesar 1 piksel [11].

Penelitian yang dilakukan oleh Haralick yang berjudul "*Textural Features for Image Classification*" mengusulkan fitur tekstur yang mengandung informasi tentang karakteristik tekstur[22]. Pada Tabel 2.1 menjelaskan rumus perhitungan fitur tersebut.

Tabel 2.1 Tabel fitur tekstur GLCM.

| No. | Properti | Rumus | Persamaan |
|---|---|---|---|
| 1 | *Angular Second Moment : Energy/Uniformity* | $f_1 = \sum_i \sum_j \{p(i,j)\}^2$ | (2-4) |
| 2 | *Contrast* | $f_2 = \sum_{n=0}^{Ng-1} n^2 \left\{ \sum_{i=1}^{Ng} \sum_{\substack{j=1 \\ \lvert i-j \rvert = n}}^{Ng} p(i,j) \right\}$ | (2-5) |
| 3 | *Correlation* | $f_3 = \frac{\sum_i \sum_j (ij) p(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y}$<br>Dimana μ merupakan means dan $\sigma$ merupakan standar deviasi dari p. $N_g$ merupakan jumlah kolom/baris. n jumlah pixel. i merupakan baris ke-i, j merupakan kolom ke-j. | (2-6) |
| 4 | *Inverse Difference Moment* : *Homogeneity* | $f_5 = \sum_i \sum_j \frac{1}{1 + (i-j)^2} p(i,j)$ | (2-7) |
| 5 | *Entropy* | $f_9 = -\sum_i \sum_j p(i,j) \log\big(p(i,j)\big)$ | (2-8) |

Pada jurnal internasional dengan judul "*Efficient analysis of satellite image denoising and resolution enhancement for improving classification accuracy*", terdapat 5 fitur yang paling baik untuk digunakan yaitu *Energy, Contrast, Correlation, Homogeneity* dan *Entropy* [23].

### 2.2.5 *Local Binary Pattern*

Local Binary Pattern merupakan metode yang digunakan sebagai ukuran tekstur *grayscale* yang terbukti efektif dan *invariant* terhadap pencahayaan yang berbeda. Metode ini teruji ampuh untuk mendeskripsikan tekstur, karena memiliki daya pembeda yang akurat, serta mempunyai toleransi terhadap perubahan *grayscale*

yang *monotatic*. LBP dimanfaatkan untuk deskripsi tekstur dan didukung oleh komposisi pola mikro yang dapat dijelaskan oleh sebuah operator[6].

Operator tersebut bekerja dengan cara memberikan label pada piksel dengan melakukan *thresholding* pada setiap piksel tetangga sebagai nilai tengah dan mengubah hasilnya menjadi nilai 0 atau 1 (biner). Oleh sebab itu Ojala, et al menyebut bahwa fundamental *pattern* ini sebagai *uniform pattern* karena mengandung dua *bitwise transition* dari 0 ke 1 dan sebaliknya. Apabila piksel bernilai kurang dari nilai tengah (piksel yang diolah) maka akan diberi nilai 0, sedangkan piksel yang memiliki nilai lebih dari nilai tengah maka akan diberi nilai 1 [6].

Secara matematis *thresholding* ditunjukkan oleh Persamaan 3, dan Persamaan 3 untuk menghitung nilai biner dari hasil *thresholding* menjadi angka desimal [6].

$$s(x)\begin{cases}1, & x \geq 0\\ 0, & x<0\end{cases} \qquad (2-9)$$

$$LBPp,r(xc,y) = \sum_{p=0}^{p=1} S(gp-gc)2^p \qquad (2-10)$$

Dimana :

P : banyaknya piksel tetangga

R : nilai jarak/radius

gc : nilai dari piksel x dan y

gp : nilai piksel tetangga

xc , yc : koordinat pusat

Ekstraksi LBP dilakukan dengan cara membagi citra sama rata menjadi regionregion R0,R1,R2,……Rm. Pada setiap region dilakukan proses LBP yang kemudian menghasilkan histogram. Hasil histogram dari tiap *region* kemudian digabungkan menjadi histogram ciri [6].

Histogram ciri yang terbentuk menggambarkan tekstur lokal dan bentuk *global* dari telapak tangan. Beberapa parameter yang dioptimalkan untuk menghasilkan ekstraksi ciri yang lebih baik. Parameter tersebut adalah operator dari LBP dan jumlah pembagian citra (*region*)[6].

### 2.2.6 Konsep Dasar Jaringan Saraf Tiruan

Setiap pola-pola informasi *input* dan *output* yang diberikan ke dalam JST diproses dalam neuron. Neuron-neuron tersebut terkumpul di dalam lapisan-lapisan

yang disebut neuron *layers.* Lapisan-lapisan penyusun JST tersebut dapat dibagi menjadi 3[24], yaitu :

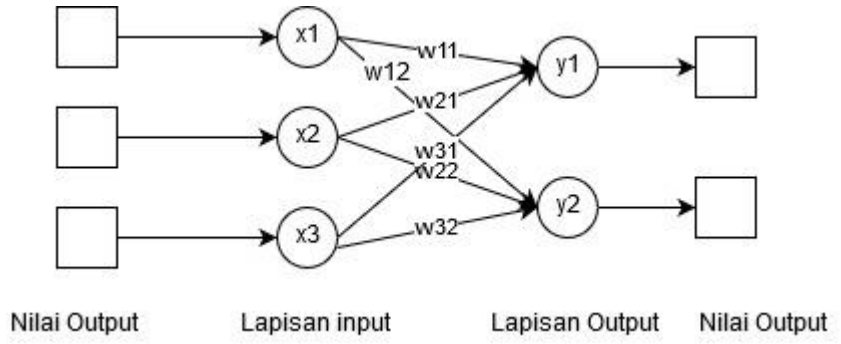
1. Lapisan *input*, unit-unit di dalam lapisan *input* disebut unit-unit *input*. Unit-unit *input* tersebut menerima pola data dari luar yang menggambarkan suatu permasalahan.
2. Lapisan tersembunyi, unit-unit di dalam lapisan tersembunyi disebut unit-unit tersembunyi. Di mana *output*-nya tidak dapat secara langsung diamati.
3. Lapisan *Output*, unit-unit di dalam lapisan *output* disebut unit-unit *output*. *Output* dari lapisan ini merupakan solusi JST terhadap suatu permasalahan.

### 2.2.7 Arsitektur Jaringan Syaraf Tiruan

JST memiliki beberapa arsitektur jaringan yang sering digunakan dalam berbagai aplikasi. Arsitektur JST tersebut, antara lain (Hermawan A., 2006)[24]:

1. Jaringan Lapisan Tunggal (*Single Layer Network*)

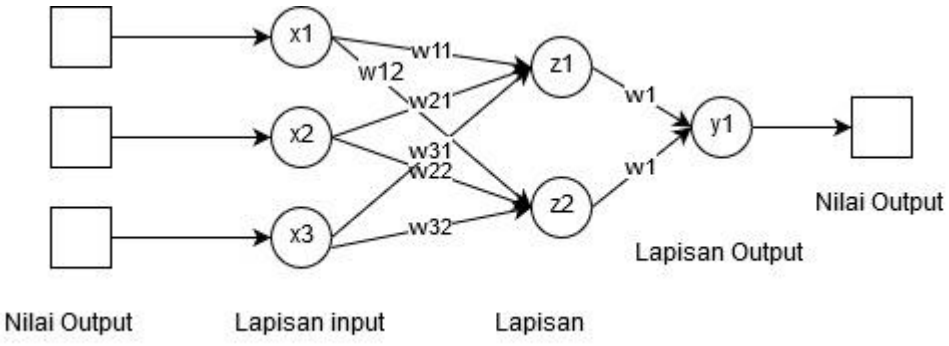
   Jaringan dengan lapisan tunggal terdiri dari 1 lapisan *input* dan 1 lapisan *output*. Setiap neuron yang terdapat di dalam lapisan *input* selalu terhubung dengan setiap neuron yang terdapat pada lapisan *output*. Jaringan ini hanya menerima input kemudian secara langsung akan mengolahnya menjadi *output* tanpa harus melalui lapisan tersembunyi.




Gambar 2. 3 Arsitektur Lapisan Tunggal

2. Jaringan Banyak Lapisan (*Multilayer net*)

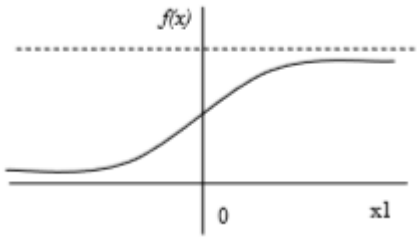
   Jaringan dengan lapisan jamak memiliki ciri khas tertentu yaitu memiliki 3 jenis lapisan yakni lapisan *input*, lapisan *output*, dan lapisan tersembunyi. Jaringan dengan banyak lapisan ini dapat menyelesaikan permasalahan yang lebih kompleks dibandingkan jaringan dengan lapisan tunggal. Namun, proses pelatihan sering membutuhkan waktu yang cenderung lama.




Gambar 2. 4 Arsitektur Lapisan *Multilayer*

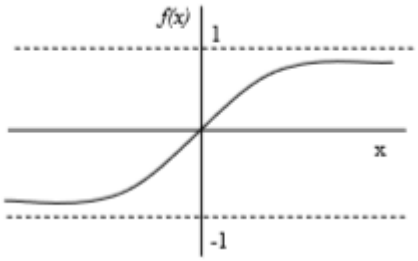
### 2.2.8 Fungsi Aktivasi *Backpropagation*

Pada *Artificial Neural Network backpropagation*, fungsi aktivasi yang dipakai harus memenuhi beberapa syarat yaitu : kontinu, terdiferensial dengan mudah dan merupakan fungsi yang tidak turun. Salah satu fungsi yang memenuhi ketiga syarat tersebut sehingga sering dipakai adalah fungsi *sigmoid* biner yang memiliki range (0,1). Diberikan $F(x) = \frac{1}{1+e^{-x}}$ dengan turunan $f'(x) = f(x)(1 - f(x))$. Grafik fungsinya tampak pada Gambar 5.




Gambar 2. 5 Fungsi Aktivasi *Sigmoid Biner*

Fungsi lain yang sering dipakai adalah fungsi *sigmoid bipolar* yang bentuk fungsinya mirip dengan fungsi *sigmoid* biner, tapi dengan *range* (-1,1). Diberikan $F(x) = \frac{2}{1+e^{-x}} - 1$ dengan turunan $f(x) = \frac{(1+f(x))(1-f(x))}{2}$. Grafik fungsinya tampak pada Gambar 2.7.




Gambar 2. 6 Fungsi Aktivasi *Sigmoid Bipolar*

Fungsi *sigmoid* memiliki nilai maksimum = 1. Maka untuk pola yang targetnya lebih dari 1, pola masukan dan keluaran harus terlebih dahulu ditransformasi sehingga semua polanya memiliki *range* yang sama seperti fungsi *sigmoid* yang dipakai. Alternatif lain adalah menggunakan fungsi aktivasi *sigmoid* hanya pada lapisan yang bukan lapisan keluaran. Pada lapisan keluaran, fungsi aktivasi yang dipakai adalah fungsi identitas[24] : $f(x) = x$

### 2.2.9 Pelatihan Standar *Backpropagation*

Pelatihan *backpropagation* meliputi 3 Tahap yaitu tahap maju, propagasi mundur, dan perubahan bobot. Algoritma pelatihan untuk jaringan dengan satu lapisan tersembunyi (dengan fungsi aktivasi *sigmoid* biner) adalah sebagai berikut :

Langkah 0 : Bobot dengan bilangan acak kecil antara 0 hingga 1 diinisialisasi

Langkah 1 : Jika kondisi penghentian belum terpenuhi, langkah 2 - 9 dilakukan

Langkah 2 : Langkah 3 - 8 berikut ini dilakukan untuk setiap data pelatihan.

**Tahap I : Propagasi Maju**

Langkah 3 : Sinyal input diterima unit *input* dan diteruskan ke *hidden neuron*.

Langkah 4 : *Output* di setiap *hidden neuron* $z_j(j = 1,2, \dots, p)$ dihitung.

$$z\_net_j = v_{j_0} \sum_{i=0}^{n} x_i v_{ji} \tag{2-11}$$

Dimana : bias = 1

Fungsi aktivasi *sigmoid biner*:

$$z_j = f\left(z\_net_j\right) = \frac{1}{1+e^{-z\_net_j}} \tag{2-12}$$

Fungsi aktivasi *sigmoid bipolar*:

$$z_j = f\left(z\_net_j\right) = \frac{2}{1+e^{z\_net_j}} - 1 \tag{2-13}$$

Langkah 5 : Output di setiap unit $y_k(k = 1,2, \dots, m)$ dihitung

$$y\_net_k = bias + \sum_{j=1}^{p} z_j w_{kj} \quad (2\text{-}14)$$

Dimana : bias =1.

Fungsi aktivasi *sigmoid biner*:

$$z_j = f\left(z\_net_j\right) = \frac{1}{1+e^{-z\_net_j}} \quad (2\text{-}15)$$

Fungsi aktivasi *sigmoid bipolar*:

$$z_j = f\left(z\_net_j\right) = \frac{2}{1+e^{z\_net_j}} - 1 \quad (2\text{-}16)$$

**Tahap II : Propagasi mundur**

Langkah 6 : Faktor $\delta$ unit *output* dihitung berdasarkan kesalahan di setiap unit *output* $y_k(k = 1,2,\dots,m)$

$$\delta_k = (t_k - y_k) f'(y_{net_k}) = (t_k - y_k) y_k (1 - y_k) \quad (2\text{-}17)$$

Suku perubahan bobot $w_{kj}$ dihitung dengan laju pembelajaran $\alpha$

.

$$\triangle w_{kj} = \alpha \delta_k z_j \quad (2\text{-}18)$$

dimana : k = 1, 2, ..., m dan j = 0, 1, ..., p

Langkah 7 : Faktor $\delta$ pada setiap *hidden neuron* dihitung berdasarkan kesalahan di setiap *hidden neuron* $z_j(j = 1,2,\dots,p)$

$$\delta\_net_j = \sum_{k=1}^{m} \delta_j w_{kj} \quad (2\text{-}19)$$

Faktor δ unit tersembunyi:

$$\delta_j = \delta_{net_j} f'\left(z_{net_j}\right) = \delta_{net_j} z_j (1 - z_j) \quad (2\text{-}20)$$

Suku perubahan bobot $v_{ji}$ dihitung.

$$\triangle v_{ji} = \alpha \delta_j x_i \quad (2\text{-}21)$$

dimana : j = 1, 2, ..., p dan i = 0, 1, ..., n

**Tahap III : Perubahan bobot**

Langkah 8 : Semua perubahan bobot dihitung. Perubahan bobot garis yang menuju ke unit *output*:

$$w_{kj}(baru) = w_{kj}(lama) + \triangle w_{kj} \quad (2\text{-}22)$$

Dimana : k = 1, 2, ..., m dan j = 0, 1, ..., p

Perubahan bobot garis yang menuju ke *hidden neuron* :

$$v_{ji}(baru) = v(lama) + \triangle v_{ji} \quad (2\text{-}23)$$

dimana : j = 1, 2, ..., p dan i = 0, 1, ..., n [26].

# BAB III
# METODE PENELITIAN

## 3.1 Bahan dan Alat Penelitian

Bahan-bahan yang digunakan pada penelitian ini adalah citra api dan non api yang disajikan dalam bentuk video sebanyak 100 video yang diambil menggunakan kamera. Video dapat diperoleh dari internet atau membuat video rekaman menggunakan kamera. Terdapat 2 pengkategorian video yaitu api dan non api. Tabel 3. 1 menunjukkan contoh *frame* dari video dengan kategori api dan non api.

Tabel 3. 1 Contoh data *frame* video

| Api | Non Api |
|---|---|
|  |   |

Alat yang digunakan dalam proses penelitian ini terbagi menjadi dua bagian yaitu:

1. Perangkat keras

   Perangkat keras yang digunakan dalam penelitian ini adalah laptop dengan spesifikasi sebagai berikut:

   a. Prosesor Intel® Core™ i3
   b. Memori RAM 4 GB

2. Perangkat lunak

   Perangkat lunak yang digunakan dalam penelitian ini, yaitu:

   a. Sistem operasi *Linux* dan *Windows*

      Sistem operasi *linux* digunakan untuk pembuatan sistem pendeteksi api sedangkan *windows* digunakan untuk pembuatan laporan dan pembuatan data.

   b. *Jupyterlab*

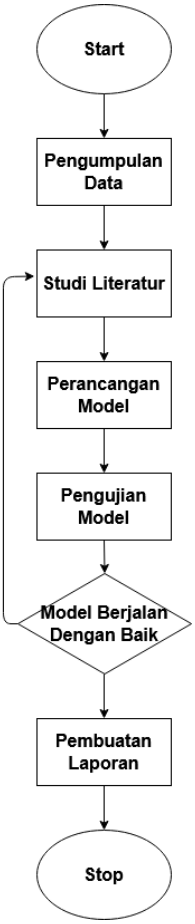
      *Jupyterlab* adalah sebuah *software* yang digunakan untuk membuat model pendeteksi api

c. Filmora

Filmora adalah salah satu *software* yang digunakan untuk proses pengeditan *video* yang melingkupi pemotongan *video* agar durasinya menjadi 10-60 detik

## 3.2 Rencana Penelitian

Pada tahap proses penelitian dilakukan beberapa tahap secara sistematis, dimulai dari tahap pengumpulan data hingga tahap penarikan kesimpulan dari model yang telah dibuat dan diuji. Tahap proses penelitian ini dapat dilihat pada diagram alir Gambar 3. 1



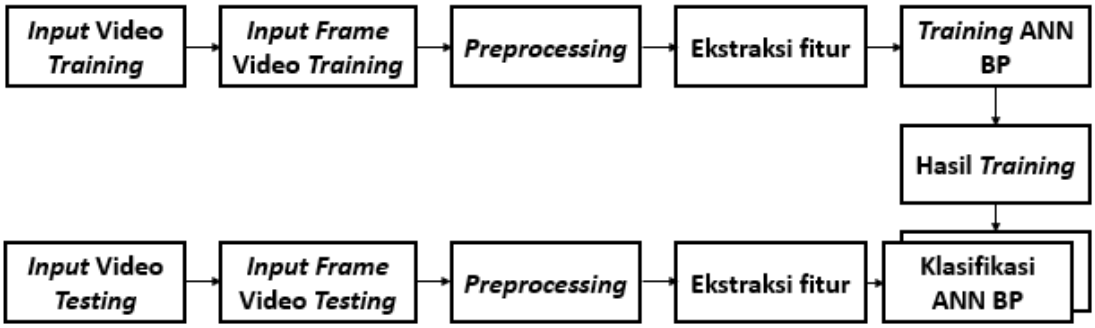

Gambar 3. 1 Diagram alir pembuatan model

Langkah pertama dalam pembuatan model ini adalah pengumpulan citra api dan non api dalam bentuk video. Citra tersebut di ambil dari berbagai sumber seperti internet ataupun citra yang dibuat sendiri. Langkah kedua yakni studi literatur dilakukan dengan cara mempelajari buku-buku, jurnal-jurnal penelitian serta sumber lain yang berhubungan dengan permasalahan yang sama. Adapun materi yang dipelajari dalam studi literatur yakni materi yang berkaitan dengan ekstraksi fitur menggunakan metode LBP dan GLCM, serta metode klasifikasi menggunakan metode

ANN *backpropagation* serta materi yang lain yang berkaitan dengan penelitian yang dilakukan. Selanjutnya adalah tahap pembangunan model sesuai yang telah direncanakan. Tahap pengujian merupakan tahap untuk menguji apakah model berjalan sesuai dengan tujuan, jika tidak sesuai maka kembali ke tahap studi literatur jika sudah sesuai maka akan dilanjutkan ke proses pembuatan laporan.

## 3.3 Rancangan Model Deteksi Apia

Model ini secara garis besar terdapat tiga proses utama yaitu *training* (pelatihan), *testing* (pengujian) dan klasifikasi, seperti yang ditunjukan pada Gambar 3.2.




Gambar 3. 2 Proses *training* (pelatihan), *testing* (pengujian) dan klasifikasi.

### 3.3.1 Tahap *Prepocessing*

Tahap *preprocessing* yang terdiri atas proses *segmentation* dan *cropping*. Setiap citra latih dan citra uji akan melewati tahapan awal tersebut untuk menghilangkan dan mengurangi *error* yang dapat berdampak pada akurasi akhir pada saat masuk ke tahap klasifikasi.

1. *Segmentation*

   Pada tahapan ini, akan dilakukan proses segmentasi terhadap citra dengan tujuan untuk mengetahui posisi dari api. Dimana untuk mengetahui posisi api digunakan aturan nilai piksel api, yaitu aturan untuk api gelap dan api terang [17]. Nilai piksel api dapat dilihat pada persamaan (2-3) dan (2-4).

   Pada Gambar 3.3 dan Gambar 3.4 merupakan tampilan dari tahap segmentasi

Gambar 3. 3 Citra *input*

Gambar 3. 4 Hasil segmentasi

2. *Cropping*

Pada tahap ini, dilakukan proses *cropping* terhadap citra setelah diketahui posisi api dengan tujuan untuk pengolahan citra hanya difokuskan terhadap api saja. Lalu untuk Penentuan bagian yang di *crop* digunakan metode *connected components* untuk mendeteksi objek-objek yang sudah di segemntasi dalam 1 frame kemudian dipilih bagian dengan lokasi api terbanyak. Pada Gambar 3. 5 merupakan tampilan dari tahap *cropping*.

Gambar 3. 5 Proses *Cropping*

### 3.3.2 Extraction *Feature*

Tahap selanjutnya yaitu *extraction feature* yang merupakan bagian dari teknik pengenalan pola (*pattern recognition*) yang bertujuan untuk mendapatkan ciri dari citra. Pada penelitian ini digunakan penggabungan Antara metode eksrtaksi *Local Binary Patern* (LBP) dan *Gray Level Co-occurrence Matrix* (GLCM). Proses ekstraksi dibagi manjadi 2 tahap Antara lain

1. *Local Binary Patern*

   *Local Binary Patern* adalah LBP adalah metode analisis tekstur yang menggunakan model statistika dan struktur. Operator LBP menggunakan perbandingan nilai keabuan dari piksel-piksel ketetanggaan. Operator dasar LBP berukuran 3 x 3 menggunakan 8 piksel ketetanggan dari sebuah piksel tengah[7]. Berikut pada Gambar 3.6 merupakan contoh komputasi LBP pada citra 4x4 pixel

| | | | |
|---|---|---|---|
| 5 | 2 | 4 | 2 |
| 2 | 4 | 6 | 10 |
| 1 | 7 | 8 | 11 |
| 2 | 3 | 6 | 5 |

Gambar 3. 6 Citra 4x4

Berikut tahap-tahap Komputasi LBP:

a. Lakukan *threshold* berpusat pada titik tengah. Pixel yang memiliki nilai sama atau lebih dibandingkan dengan titik tengah diberi nilai 1 selain itu diberi nilai 0. Sebagai contoh akan dilakukannya *threshold* dengan posisi titik pusat berada pada pixel (1,1). Kemudian titik pusat akan dibandingkan dengan pixel-pixel tetangga dari titik pusat. Gambar 3. 7 menyajikan hasil dari proses *threshold*

A =

| | | | |
|---|---|---|---|
| 5 | 2 | 4 | 2 |
| 2 | 4 | 6 | 10 |
| 1 | 7 | 8 | 11 |
| 2 | 3 | 6 | 5 |

(A)

X =

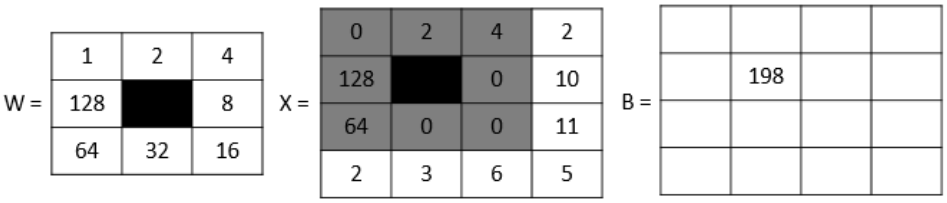
| | | | |
|---|---|---|---|
| 0 | 1 | 1 | 2 |
| 1 | | 0 | 10 |
| 1 | 0 | 0 | 11 |
| 2 | 3 | 6 | 5 |

(B)

Gambar 3. 7 (a) penentuan titik pusat index(1,1) (b) hasil *threshold*

b. Ganti nilai 1 dari hasil *threshold* yang ada pada setiap *index* sesuai dengan posisi *weights*. Lalu jumlahkan seluruh nilai pixel tetangga dari pixel pusat

kemudian hasilnya ditempatkan pada matriks B dengan posisi pixel sama dengan posisi pixel titik pusat Gambar 3.8 menunjukan proses penempatan hasil akhir dari proses komputasi pada posisi pixel (1,1)




Gambar 3. 8 (a) weights (b)hasil perubahan nilai sesuai weights(b) hasil akhir komputasi LBP pixel(1,1)

c. Lakukan proses tersebut disetiap pixel citra sampai semua pixel pada matriks B terpenuhi.

d. Sehingga hasil akhirnya dapat dilihat pada Gambar 3.9.

| | | | |
|---|---|---|---|
| 255 | 71 | 207 | 31 |
| 255 | 198 | 135 | 223 |
| 193 | 247 | 243 | 255 |
| 243 | 241 | 248 | 124 |

Gambar 3. 9 Hasil akhir perhitungan

2. *Gray Level Co-occurrence Matrix*

*Gray Level Co-occurence Matrix* adalah matriks yang menggambarkan frekuensi munculnya pasangan dua *pixel* dengan intensitas tertentu dalam jarak (d) dan orientasi arah dengan sudut (θ) tertentu dalam citra [25]. Proses ekstraksi fitur dengan metode GLCM dibagi menjadi dua tahap, yaitu pembentukan matriks GLCM dan perhitungan nilai fitur GLCM.

a. Pembentukan matriks GLCM

Misalkan terdapat citra *grayscale* dengan matriks seperti pada Gambar 3.10.

A=

| | | |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 0 | 1 |
| 2 | 2 | 2 |

Gambar 3. 10 Contoh matriks citra *grayscale*

Dari matriks di atas dibentuk matriks GLCM dengan orientasi sudut 0º, 45º, 90º, 135º. Adapun langkah-langkah pembentukan matriks GLCM sebagai berikut:

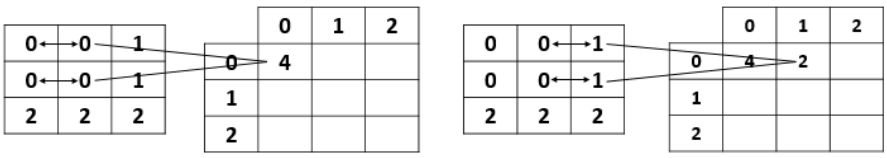
1) Langkah pertama yaitu membuat matriks dengan ordo n x n dimana n merupakan nilai elmen terbesar dari matriks *grayscale* A. pada matriks A nilai elemen terbesar adalah 2 sehingga ukuran matriks yang akan dibuat adalah 2x2 dengan nilai indeks dimulai dari 0 seperti yang terlihat pada Gambar 3.11 berikut.

B =

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | | | |
| 1 | | | |
| 2 | | | |

Gambar 3. 11 Matriks GLCM 3x3

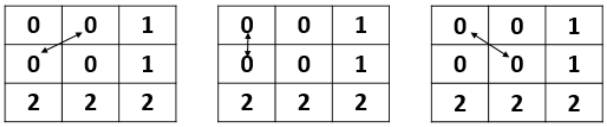
2) Penentuan nilai matriks B dimulai dari elemen (0,0). Dimana untuk mengisi setiap elemen dari matriks B, dilakukan iterasi terhadap 2 blok pada matriks A untuk mengecek nilai elemennya. Sebagai contoh, membuat matriks GLCM dengan sudut 0º, maka arah iterasi yang dilakukan yaitu horizontal.
3) Iterasi pertama dilakukan untuk mengecek jumlah matriks ketetanggaan yang bernilai (0,0). Iterasi dimulai dari elemen (0,0) dan (0,1) pada matriks A. Pengecekan berlaku dua arah, artinya elemen (0,0) dan (0,1) dicek sebanyak dua kali dengan arah berlawanan. Jumlah matriks ketetanggaan dengan nilai (0,0) ditulis pada matriks B elemen (0,0). Selanjutnya geser satu blok ke kanan untuk mengecek elemen matriks (0,1) dan (0,2). Lakukan pengecekan hingga elemen (0,2) dan (0,3) kemudian pindah ke baris selanjutnya. Gambar 3. 12 menunjukkan proses pembentukan matriks GLCM pada sudut 0º.




Gambar 3. 12 Pembentukan matriks GLCM sudut 0º

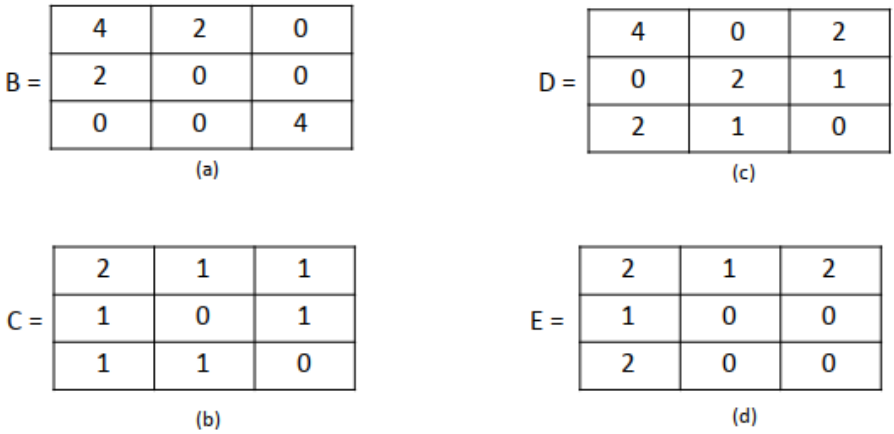
4) Iterasi selanjutnya adalah mengecek jumlah matriks ketetanggaan yang bernilai (0,1) untuk di isi di matriks B di elemen (0,1), begitu seterusnya sampai elemen terakhir pada matriks B.

5) Hal yang sama dilakukan untuk membuat matriks GLCM dengan sudut 45º, 90º dan 135º namun dengan arah yang berbeda. Matriks GLCM sudut 45º dibuat dengan arah diagonal-kanan (*right-diagonal*), sudut 90º dengan arah vertikal, sedangkan sudut 135º dibuat dengan arah diagonal-kiri (*left-diagonal*) seperti yang terlihat pada Gambar 3.13.



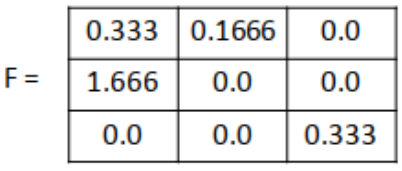

Gambar 3. 13 Arah matriks GLCM a. Sudut 45º. (b) Sudut 90º. (c) Sudut 135º

6) Didapatkan empat buah matriks GLCM seperti pada Gambar 3.14.




Gambar 3. 14 Hasil pembentukan matriks GLCM (a) $\theta = 0°$ $d = 1$. (b) $\theta = 45°$ $d = 1$. (c) $\theta = 90°$ $d = 1$. (d) $\theta = 135°$ $d = 1$

7) Matriks GLCM kemudian dinormalisasi agar jumlah seluruh elemennya sama dengan satu. Gambar 3.15 merupakan hasil normalisasi terhadap matriks B.




Gambar 3. 15 Matriks GLCM yang telah dinormalisasi

Proses normalisasi menghasilkan 4 matriks GLCM normal yang kemudian akan digunakan pada tahap selanjutnya.

b. Perhitungan nilai fitur GLCM

Matriks GLCM yang telah dihasilkan pada tahap sebelumnya digunakan untuk menghitung 5 fitur GLCM yang terdiri atas ASM (*Energy*), *Contrast*, IDM (*Homogeneity*), *Entropy* dan *Correlation*.

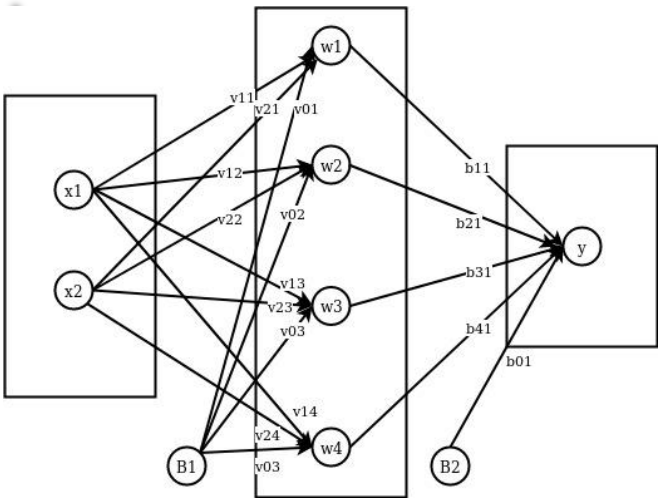
### 3.3.3 Klasifikasi

Proses klasifikasi yang dilakukan dalam penelitian ini adalah menggunakan metode ANN *backpropagation*. Berikut ini diberikan contoh *data training* menggunakan metode *backpropagation* pada pendeteksian api dengan data pelatihan *dummy* yang tertera pada Tabel 3.2. Data *dummy* tersebut merupakan data pelatihan yang terdiri dari 4 data yang terbagi ke dalam 2 kelas berdasarkan target ada atau tidak adanya api, yaitu 1 (ada) dan 0 (tidak ada).

Tabel 3. 2 Data *dummy*

| No. | Input 1 | Input 2 | Target |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 1 |
| 3 | 1 | 0 | 1 |
| 4 | 1 | 1 | 1 |

Dalam pelatihan data ini, arsitektur jaringan syaraf tiruan *backpropagation* yang digunakan dapat dilihat pada Gambar 3.16.




. Gambar 3. 16 Arsitektur *backpropagation*

Arsitektur *backpropagation* yang digunkan dalam contoh perhitungan ini adalah *single layer* yang terdiri dari 1 *input*, 1 *hidden layer* dan 1 *output layer*. Di

dalam *input layer* terdapat 2 buah *neuron*, 4 buah *neuron* dalam *hidden layer* dan 1 buah *neuron* pada layer *output*.

Proses pelatihan data dengan menggunakan metode *artificial neural network backpropagation* ini menggunakan beberapa ketentuan, berikut merupakan ketentuan-ketentuan untuk contoh pelatihan menggunakan metode ini.

1. Batas *error* = 0.0001
2. Batas *epoch* = 2000
3. *Learning rate*=0.01
4. Menggunakan fungsi aktivasi *sigmoid biner*

Alogoritma metode *backpropagation* terdiri dari 10 langkah termasuk dalam tahap inisisalisasi yang terbagi menjadi 3 fase.

Langkah 0 : Inisialisai bobot awal dari *input* awal ke *hidden layer* pertama yang disajikan dalam Tabel 3.3 berikut.

Tabel 3.3 Bias dan bobot awal dari *input* layer ke *hidden layer* pertama.

| Dari- / Ke- | Bias (B1) | X1 | X2 |
|---|---|---|---|
| W1 | 0.28183 | 0.1377 | 0.0045 |
| W2 | 0.7558 | 0.1031 | -0.0380 |
| W3 | 0.61836 | -0.0317 | 0.1135 |
| W4 | 0.2505 | -0.0964 | -0.0786 |

Bias dan bobot awal dari *input layer* ke *hidden layer* pertama dinyatakan dengan $V_{01}$ sampai $V_{24}$, dapat dilihat pada Tabel 3.3. Selanjutnya untuk bias dan bobot awal dari *hidden layer* pertama ke *output layer* kedua dinyatakan dengan $b_{01}$ sampai $b_{41}$ yang disajikan dalam Table 3.4:

Tabel 3.4 Bias dan bobot awal dari *hidden layer* pertama ke *output layer* kedua

| Dari- / Ke- | Bias (B2) | W1 | W2 | W3 | W4 |
|---|---|---|---|---|---|
| Y | 0.9097 | -0.0936 | 0.3335 | 1.6324 | 0.0187 |

Langkah 1: Jika kondisi penghentian belum terpenuhi, lakukan langkah 2 sampai 9. Kondisi penghentian terpenuhi jika *error* < 0.0001 atau *epoch* > 2000.

Langkah 2: Untuk setiap data pelatihan, lakukan langkah 3 sampai 8

**Fase I: *Feedforward***

Langkah 3: Tiap unit masukan ($x_i$, $i$ = 1, 2,…, $n$) menerima sinyal dan meneruskannya ke unit selanjutnya, yaitu lapisan tersembunyi. Yang digunakan pada contoh ini adalah data pertama dengan $X_1 = 1$, $X_2 = 0$.

Langkah 4 : Hitung semua keluaran pada lapisan tersembunyi ($Z_j$, $j$ = 1, 2,…, $p$) menggunakan Persamaan (2-11)

W_net$_1$ = 0.28183 + (1 * 0.1377) + (0 *0.0045)

= 0.41953

W_net$_2$ = 0.7558 + (1 * 0.1031) + (0 * -0.0380)

= 0.8589

W_net$_3$ = 0.61836 + (1 * -0.0317) + (0 * 0.1135)

= 0.58666

W_net$_4$ = 0.2505+ (1 * -0.0964) + (0 * -0.0786)

= 0.012542

Gunakan fungsi aktivasi untuk menghitung sinyal keluaran tiap *neuron hidden layer* pertama. Fungsi aktivasi yang digunakan adalah *sigmoid biner* dan menggunakan Persamaan (2-12).

$W_1 = \frac{1}{1+e^{-0.41953}} - 1 \quad = 0.60333$

$W_2 = \frac{1}{1+e^{-0.8589}} - 1 \quad = 0.7024$

$W_3 = \frac{1}{1+e^{-0.58666}} - 1 \quad = 0.6425$

$W_4 = \frac{1}{1+e^{-0.012542}} - 1 \quad = 0.5031$

Sinyal W1, W2, dan W3 akan diteruskan ke *output layer*.

Langkah 5 : Hitung semua keluaran jaringan di lapisan *output* ($Y_k$, $k$ = 1, 2,…, $m$) menggunakan Persamaan (2-17)

Y_net$_1$ = 0.9097 + (0.60333 * -0.0936) + (0.7024 * 0.3335) + (0.7024 * 1.6324) + (0.5031* 0.0187)

= 2.24348725

Gunakan fungsi aktivasi *sigmoid biner* untuk menghitung sinyal *output*-nya:

$Y_1 = \frac{1}{1+e^{-0.16276}} - 1 = 0.904087276$

**Fase II: *Backpropagation***

Langkah 6: Hitung faktor $\delta$ unit keluaran berdasarkan kesalahan di setiap unit keluaran ($y_k$, $k$ = 1, 2,…, $m$) menggunakan Persamaan (2-17).

$\delta_1 = (t_1 - y_1)\, f'(y\_net_1)$

= (1 – 0.904087276) $f'$ (2.24348725)

= 0.008316925

$\delta$ merupakan unit kesalahan yang akan digunakan dalam perubahan bobot *layer* di bawahnya (langkah 7). Kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki $W_{jk}$) dengan *learning rate* ($\alpha$), menggunakan Persamaan (2-18)

$\Delta b_{11}$ = 0.01 * 0.008316925* 0.60333 = 0.000050179

$\Delta b_{21}$ = 0.01 * 0.008316925* 0.7024= 0.000058418

$\Delta b_{31}$ = 0.01 * 0.008316925* 0.6425= 0.000053436

$\Delta b_{41}$ = 0.01 * 0.008316925* 0.5031= 0.000041842

Hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai $w_{0k}$) menggunakan Persamaan (2-18)

$\Delta b_{01}$ = 0.01 * 0.008316925= 0.00008316925

Langkah 7: Hitung faktor $\delta$ unit tersembunyi berdasarkan kesalahan di setiap unit tersembunyi ($Z_j$, $j$ = 1, 2,…, $p$) menggunakan Persamaan (2-19)

$\delta\_net_1$ = 0.008316925* -0.0936= -0.000778

$\delta\_net_2$ = 0.008316925* 0.3335= 0.00277369

$\delta\_net_4$ = 0.008316925* 1.6324= 0.01357654837

$\delta\_net_4$ = 0.008316925* 0.0187 = 0.000155526

Faktor $\delta$ unit tersembunyi *hidden layer* pertama dihitung menggunakan Persamaan (2-20):

$\delta_1 = \delta\_net_7\, f'(Z\_net_7)$

= -0.000778 * $f'$ (0.41953)

= -0.000186667

$\delta_2$ = 0.00277369 * $f'$ (0.8589)

= 0.00057976161

$\delta_3$ = 0.01357654837 * $f'$ (0.58666)

= 0.003118067325

$\delta_4$ = 0.000155526 * $f'$ (0.012542)

= 0.00003887997

Hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai $v_{ij}$) dari *input layer* ke *hidden layer* pertama menggunakan Persamaan (2-18) dengan $x_i = Z_i$

$\Delta V_{11}$ = 0.01 * -0.000186667* 1= -0.00000186667

$\Delta V_{12}$ = 0.01 * 0.00057976161* 0= 0

$\Delta V_{13}$ = 0.01 * 0.003118067325* 1= 0.00003118067325

$\Delta V_{14}$ = 0.01 * 0.00003887997 * 0= 0

$\Delta V_{21}$ = 0.01 * -0.000186667* 1= -0.00000186667

$\Delta V_{22}$ = 0.01 * 0.00057976161* 0 = 0

$\Delta V_{23}$ = 0.01 * 0.003118067325* 1 = 0.00003118067325

$\Delta V_{24}$ = 0.01 * 0.00003887997 * 0= 0

Hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai $V_{0j}$) menggunakan Persamaan (2-19)

$\Delta V_{01}$ = 0.01 * -0.000186667= -0.000001866

$\Delta V_{02}$ = 0.01 * 0.00057976161= 0.000005797

$\Delta V_{03}$ = 0.01 * 0.003118067325= 0.00003118

$\Delta V_{04}$ = 0.01 * 0.00003887997= 0.0000003887

**Fase III: Perubahan Bobot**

Langkah 8: Tiap-tiap unit *output* ($Y_k$ , $k$ = 1, 2,…, $m$) memperbaiki bias dan bobotnya ($j$ = 0, 1, 2,..., $p$) menggunakan Persamaan (2-20)

$W_{jk}$ ($baru$) = $W_{jk}$ ($lama$) + $\Delta W_{jk}$

$b_{01}$ ($baru$) = 0.9097+ 0.00008316925= 0.90978316925

$b_{11}$ ($baru$) = -0.0936 + 0.000050179= -0.093549821

$b_{21}$ ($baru$) = 0.3335+ 0.000058418= 0.33358418

$b_{31}$ ($baru$) = 1.6324+ 0.000053436=1.632453436

$b_{41}$ ($baru$) = 0.0187+ 0.000041842= 0.018741842

Tiap-tiap unit tersembunyi ($Z_j$, $j$ = 1, 2, 3,..., $p$) memperbaiki bias dan bobotnya ($j$ = 0, 1, 2, 3,..., $n$) menggunakan Persamaan (2-21)

$V_{ij}$ ($baru$) = $V_{ij}$ ($lama$) + $\Delta V_{ij}$

$V_{01}$ ($baru$) = 0.28183 + (-0.000001866) = 0.281828134

$V_{11}$ ($baru$) = 0.1377+ -0.000001866= 0.0040000207

$V_{12}$ ($baru$) = 0.0045+ 0 = 0.0045

Demikian juga untuk $V_{13}$ ($baru$) sampai $V_{24}$ ($baru$) dihitung dengan cara yang sama untuk memperoleh semua $V_{ij}$ ($baru$)

Langkah 9: Kondisi pelatihan berhenti, jika *error* <= 0.0001 atau jumlah *epoch* mencapai 100. *Error* dihitung dengan *Mean Square Error* menggunakan persamaan:

$$Error = 0.5 * \{(t_1 - y_1)^2 + (t_2 - y_2)^2 + \ldots + (t_m - y_m)^2\} \quad (3\text{-}1)$$

Dimana m adalah jumlah data yang dilatih.

*Software Jupyterlab* digunakan untuk pembuatan jaringan *backpropagation* yang sesuai dengan inisialisasi awal dan parameter dari pelatihan manual yang telah dilakukan. Sehingga diperoleh bias dan bobot akhir yang dapat dilihat pada Tabel 3.3, Tabel 3.5.

Tabel 3. 3 Bias dan bobot akhir dari *input layer* ke *hidden layer* pertama

| Dari- / Ke- | Bias (B1) | X1 | X2 |
|---|---|---|---|
| W1 | 0.41622322099 | -0.18717812 | -0.115188427727 |
| W2 | 2.37629377222 | -1.656265855 | -1.4001292658 |
| W3 | 0.346785132537 | 0.959095285 | -0.6163883398 |
| W4 | 0.412857015836 | -0.20146158 | -0.122667397 |

Tabel 3. 4 Bias dan bobot akhir dari *hidden layer* pertama ke *output layer*

| Dari- / Ke- | Bias (B2) | W1 | W2 | W3 | W4 |
|---|---|---|---|---|---|
| Y | 1.50274692507 | 0.557285153 | -2.184492 | 0.67748887558 | 0.54299331 |

Berdasarkan jaringan *backpropagation* yang telah dibuat, 9 data pelatihan *dummy* yang telah disiapkan kemudian dimasukkan ke dalam jaringan sehingga diperoleh hasil yang dapat dilihat pada Tabel 3.5. *Threshold* pada *output* data latih dilakukan dengan membulatkan *output* ke bilangan bulat (*integer*) terdekat, dengan kata lain menggunakan $\theta = 0.5$.

Tabel 3. 5 *Output* data latih

| **Data ke-** | ***Output*** | **Hasil *threshold*** | **Target** |
|---|---|---|---|
| 1 | 0.023894362639337377 | 0 | 0 |
| 2 | -0.000870907443 | 0 | 0 |
| 3 | 0.9814555682475549 | 1 | 1 |
| 4 | 0.013271482008336536 | 0 | 0 |

Untuk mendapatkan tingkat akurasi dari hasil pelatihan, jumlah hasil *threshold* yang sesuai target dibagi dengan jumlah data. Tingkat akurasi = (4/4) * 100 % = 100 %.

## 3.4 Teknik Pengujian

Tahapan pengujian merupakan tahapan yang ditujukan sebagai bahan evaluasi model arsitektur ANN untuk mengetahui model yang dibuat telah sesuai dengan perancangan ataupun model diharapkan. Dalam melakukan pengujian pada model ada beberapa metode pengujian model yang dapat digunakan.

### 3.4.1 Skenario Pengujian

Pada model deteksi api menggunakan metode *backpropagation* ini, dilakukan pengujian terhadap pengaruh jumlah sampling terhadap unjuk kerja dan waktu komputasinya. Dimana rasio data *traning* dan *testing* adalah 70:30, 80:20, dan 90:10. Setiap rasio data *training* memiiki ciri citra yang telah disimpan di dalam *database* masing-masing yang digunakan sebagai *training* untuk menedeteksi api dan non api. Sebagai tolak ukur penentuan target pada setiap rasio data video *testing*, dilakukan dengan mengambil ukuran sampel pada video dengan ukuran sampel ditentukan dengan cara perhitungan menggunakan metode sampling yakni metode solvin yang perhitungannya dapat dilihat pada persamaan (2-1).

Setelah didapatkan ukuran sampel, maka dilakukan proses klasifikasi terhadap semua sampel. Kemudian untuk menentukan hasil klasifikasi dari sampel tersebut akan dihitung tingkat pendeteksian api dengan rumus sebagai berikut

$$T = \frac{jumlah\ api\ yang\ terdeteksi}{jumlah\ frame\ yang\ di\ ambil} 100\% \qquad (3\text{-}1)$$

Pada setiap rasio, untuk menetukan nilai T, maka nilai T akan dibandingkan dengan nilai pembanding (25%, 50%, 75%). Jika nilai T yang didapatkan lebih besar sama dengan nilai pembanding maka video tersebut terindikasi api, jika T lebih kecil dari nilai pemabnding maka video tersebut terindikasi non api.

Pengujian dilakukan untuk menentukan model terbaik, dimana terdapat beberapa tahap yang akan dilakukan untuk medapatkan model yang terbaik, berikut merupakan tahap pengujian.

1. Penentuan rasio, pembanding, menentukan variabel *artificial neural network* terbaik yang terdiri dari variasi *learning rate* (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9), variasi *error* (0.001, 0.0001, 0.00001), serta variasi *epoch* dari rentang nilai 100-500 dengan kelipatan 100. Adapun untuk *input*-an digunakan 4 buah fitur dari GLCM dengan 4 buah sudut yaitu 0, 45, 90, 135 derajat, jumlah *hidden layer* 1 [5] dan bias dengan nilai acak dengan rentang 0-1.

2. Setelah itu, maka dilanjutkan dengan penentuan variasi jumlah *hidden layer* yang baik (1,2,3)

Sebegai tolak ukur penetuan model terbaik maka digunakan tiga buah parameter yaitu akurasi, *precision*, *recall*.

Lalu untuk menentukan ketiga parameter tersebut maka dilakukan perhitungan dari hasil pendeteksian api dan non api. Hal tersebut dilakukan agar terdapat kesesuaian antara target dibandingkan dengan hasil dari model yang dibuat. Dalam melakukan *confusion matrix* seperti yang tertera *Tabel* 3. 6.

Tabel 3. 6 *Confusion matrix* 3x3.

| | ***Actual Class*** | |
|---|---|---|
| ***Predict Class*** | TP | FP |
| | FN | TN |

Berikut ini merupakan cara menghitungnilai akurasi, *recall*, persisi menggunakan *Confusion matrix* pada Tabel 3.6.

$$\text{Akurasi} = \frac{TP+TN}{(\text{TP+FP+FN+TN})} * 100\ \% \qquad (3-2)$$

$$Percision = \frac{TP}{(\text{TP+FP})} * 100\ \% \qquad (3-3)$$

$$Recall = \frac{TP}{(\text{TP}+\text{FN})} * 100\ \% \qquad (3-4)$$

Keterangan:

TP= *True Positif*

TN= *True Negatif*

FP= *False Positif*

FN= *False Negatif*

Misalnya, dari pengkajian 10 video terdapat 5 video api dan 5 video non api, Hasil pendeteksian disajikan dalam Tabel 3.7

Tabel 3. 7 Hasil deteksi

| NO | Status Sebenarnya | Predict |
|---|---|---|
| 1 | Non Api | Non Api |
| 2 | Non Api | Non Api |
| 3 | Non Api | Non Api |
| 4 | Non Api | Non Api |

| | | |
|---|---|---|
| 5 | Non Api | Api |
| 6 | Api | Api |
| 7 | Api | Api |
| 8 | Api | Api |
| 9 | Api | Api |
| 10 | Api | Api |

Berdasarkan kasus tersebut, dari 10 video yang dikaji didapatkan nilai akurasi, *percision*, *recall* dengan rumus pada persamaan (3-2), (3-3), dan (3-4) sebesar:

Akurasi $= \frac{5+4}{(5+1+0+4)} * 100\% = 90\%$

*Percision* $= \frac{5}{(5+1)} * 100\ \% = 83\ \%$

*Recall* $= \frac{5}{(5+0)} * 100\ \% = 100\ \%$

### 3.5 Jadwal Kegiatan

Waktu yang digunakan dalam proses pengembangan model pendeteksi api yaitu selama enam bulan. Jadwal kegiatan pengembangan model pendeteksi api seperti pada Tabel 3.8

Tabel 3. 8 Jadwal kegiatan perancangan

| No. | Kegiatan | Waktu (Bulan) | | | | | Keterangan |
|---|---|---|---|---|---|---|---|
| | | I | II | III | IV | V | |
| 1 | Analisa | | | | | | Analisa kebutuhan |
| 2 | Perancangan | | | | | | Perancangan model |
| 3 | *Coding* | | | | | | Pengkodean model |
| 4 | *Testing* | | | | | | Pengujian model |
| 5 | Dokumentasi | | | | | | Dokumentasi model |

# BAB IV
# HASIL DAN PEMBAHASAN

## 4.1 Pengumpulan Data

Pada penelitian ini digunakan *dataset* terdiri atas 100 video, dengan perbandingan 50 video jenis api dan 50 video jenis non-api. Durasi video 10 detik dan kualitas video 30 *frame per second* (*fps*). Dari 100 video akan diambil secara framenya *random* dalam bentuk file .jpg, dimana per-video akan diambil sampel dengan menggunakan metode sampling yakni *slovin* [19] dengan toleransi sebesar 8%. Cara perhitungan pengambilan jumlah sampling menggunakan persamaan (2-1) dengan metode *slovin* sebagai berikut.

$$jumlah\_sampel = \frac{300}{1 + 300 * 0.08^2} = 103$$

Dataset tersebut digunakan untuk melakukan proses *training* dan *testing* untuk mendapatkan model *artificial neural network* yang baik. Dari hasil model yang telah didapatkan akan dilanjutkan proses pengujian nilai ambang. Setelah itu, dilanjutkan dengan proses validasi dengan 30 data video yang berbeda dengan data video pada pengujian sebelumnya dengan variasi 15 video jenis api dan 15 video jenis non-api.

## 4.2 Pengujian

Pengujian akurasi dilakukan untuk mengetahui seberapa baik kinerja metode *backpropagation* dalam mengklasifikasikan objek api dan non-api. Pada tahap ini, dilakukan pengujian untuk mengetahui arsitektur ANN yang paling baik, untuk percobaan secara keseluruhan dapat dilihat pada lampiran 1. Parameter yang digunakan dalam pengujian sebagai berikut.

- *Input layer* : 20 *neuron*
- *Output layer* : 1 *neuron*
- Fungsi aktivasi : *Sigmoid biner*
- Batas *epoch* : 100, 200, 300, 400, 500
- Batas *error* : 0.001, 0.0001, 0.00001
- *Learning rate* : 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
- *Hidden layer* : 1
- *Neuron* : 7
- Pembagian dataset: 70:30, 80:20, 90:10

### 4.2.1 Pengaruh Variasi Pembagian Dataset

Pada penelitian ini dilakukan pengujian pengaruh pembagian dataset terhadap hasil akurasi yang didapatkan. Dimana dataset dibagi menjadi 70:30, 80:20, 90:10, dengan jumlah *input*-an yakni 20 dan 1 *hidden layer*. Berikut hasil akurasi terbaik dari pembagian dataset yang disajikan pada Tabel 4.1.

Tabel 4. 1 Akurasi terbaik variasi pembagaian dataset

| | | DataSet | | | |
|---|---|---|---|---|---|
| | | 60_40 | 70_30 | 80_20 | 90_10 |
| Test | Akurasi | 0.92 | **0.96** | 0.95 | 0.96 |
| | Percission | 0.93 | **0.97** | 0.95 | 0.96 |
| | Recall | 0.93 | **0.97** | 0.95 | 0.96 |
| Train | Akurasi | 0.93 | **0.97** | 0.96 | 0.96 |
| | Percission | 0.93 | **0.97** | 0.96 | 0.97 |
| | Recall | 0.93 | **0.97** | 0.96 | 0.97 |
| | error | 0.08 | **0.03** | 0.04 | 0.04 |

Dari Tabel 4.1 dapat disimpulkan bahwa hasil tertinggi yang didapatkan terdapat pada pembagian dataset 70:30 dengan akurasi sebesar 96%, *percission* 97%, dan *recall* 97%. Kemudian, pengaruh variasi pembagian dataset terhadap hasil terbaik yang didapatkan tidak memberikan dampak yang besar. Hal ini karena nilai akurasi yang didapatkan tidak mengalami perbedaan yang signifikan. Pada tahap berikutnya pembagian dataset 70:30 akan digunakan pada pengujian selanjutnya.

### 4.2.2 Pengaruh Batas Error Terhadap Unjuk Kerja ANN

Pada penelitian ini dilakukan pengujian pengaruh batas *error* terhadap hasil akurasi. Dimana variasi *error* yang digunakan pada penelitian ini adalah 0.001, 0.0001, 0.00001 dengan 20 jumlah *input*-an dan pembagian dataset 70:30. Berikut pada Tabel 4.2 disajikan hasil akurasi terbaik dari variasi *error* yang didapatkan.

Tabel 4. 2 Akurasi terbaik variasi *error*

| | | Batas Error | | |
|---|---|---|---|---|
| | | 0.001 | 0.0001 | 0.00001 |
| Test | Akurasi | **0.96** | 0.94 | 0.94 |
| | Percission | **0.97** | 0.95 | 0.95 |
| | Recall | **0.97** | 0.95 | 0.95 |
| Train | Akurasi | **0.97** | 0.94 | 0.95 |
| | Percission | **0.97** | 0.95 | 0.95 |
| | Recall | **0.97** | 0.95 | 0.95 |
| | error | **0.03** | 0.05 | 0.04 |

Dari tabel Tabel 4.2, dapat simpulkan bahwa pengaruh batas *error* yang telah ditentukan pada penelitian ini adalah batas *error* tidak memberikan dampak apapun terhadap tingkat akurasi yang didapatkan. Hal ini karena *error* yang didapatkan masih jauh dari dari batas *error* yang telah ditentukan.

### 4.2.3 Pengaruh Batas *Epoch* Terhadap Unjuk Kerja ANN

Pada penelitian ini dilakukan pengujian pengaruh batas *epoch* terhadap akurasi yang didapatkan. Dimana variasi *epoch* yang digunakan adalah 100, 200, 300, 400, 500 dengan 20 jumlah *input*-an dan pembagian dataset sebesar 70:30. Berikut pada Tabel 4.3 disajikan hasil akurasi yang didapatkan

Tabel 4. 3 Akurasi terbaik variasi batas *epoch*

| Epoch | Test | | | Train | | | |
|---|---|---|---|---|---|---|---|
| | Akurasi | Percission | Recall | Akurasi | Percission | Recall | error |
| 100 | 0.92 | 0.93 | 0.93 | 0.93 | 0.94 | 0.94 | 0.07 |
| 200 | 0.92 | 0.93 | 0.93 | 0.93 | 0.93 | 0.93 | 0.07 |
| 300 | 0.93 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.06 |
| 400 | 0.94 | 0.94 | 0.94 | 0.95 | 0.95 | 0.95 | 0.05 |
| 500 | **0.96** | **0.97** | **0.97** | **0.97** | **0.97** | **0.97** | **0.03** |
| 600 | 0.94 | 0.94 | 0.94 | 0.94 | 0.95 | 0.94 | 0.06 |
| 700 | 0.94 | 0.94 | 0.94 | 0.94 | 0.95 | 0.95 | 0.06 |
| 800 | 0.94 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.06 |
| 900 | 0.95 | 0.95 | 0.95 | 0.95 | 0.96 | 0.96 | 0.05 |
| 1000 | 0.95 | 0.95 | 0.95 | 0.96 | 0.96 | 0.96 | 0.05 |

Dari Tabel 4.3, dapat simpulkan bahwa hasil tertinggi yang didapatkan terdapat pada epoch 500 dengan tingkat akurasi sebesar 96%, *percission* 97%, dan *recall* 97%. Kemudian, pengaruh peningkatan batas *epoch* yang telah ditentukan pada penelitian adalah cukup memberikan pengaruh terhadap peningkatan tingkat akurasi yang didapatkan. Namun titik maksimal peningkatan akurasi yang didapatkan berada pada *epoch* 500. Dengan demikian 500 merupakan batas *epoch* yang akan digunakan untuk tahap selanjutnya.

### 4.2.4 Pengaruh *Learning Rate*

Pada penelitian ini dilakukan pengujian terhadap pengaruh variasi learning rate terhadap tingkat akurasi yang didapatkan. Dimana variasi *learning rate* yang digunakan adalah 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 dengan pembagian dataset 70:30 dan batas *epoch* 500. Berikut pada Tabel 4.4 disajikan hasil akurasi terbaik yang didapatkan.

Tabel 4. 4 Akurasi terbaik variasi *learning rate*

| *Learning rate* | | | | | | | |
|---|---|---|---|---|---|---|---|
| LR | Test | | | Train | | | |
| | Akurasi | Percission | Recall | Akurasi | Percission | Recall | error |
| 0.1 | 0.93 | 0.93 | 0.93 | 0.93 | 0.94 | 0.94 | 0.07 |
| 0.2 | 0.93 | 0.94 | 0.93 | 0.94 | 0.94 | 0.94 | 0.07 |
| 0.3 | 0.94 | 0.95 | 0.95 | 0.95 | 0.96 | 0.96 | 0.06 |
| 0.4 | **0.96** | **0.97** | **0.97** | **0.97** | **0.97** | **0.97** | **0.03** |
| 0.5 | 0.93 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.07 |
| 0.6 | 0.93 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.07 |
| 0.7 | 0.93 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.07 |
| 0.8 | 0.93 | 0.93 | 0.93 | 0.93 | 0.93 | 0.93 | 0.07 |
| 0.9 | 0.94 | 0.94 | 0.94 | 0.94 | 0.95 | 0.95 | 0.05 |

Dari Tabel 4.4, dapat simpulkan bahwa hasil tertinggi yang didapatkan terdapat pada learning rate 0.4 dengan tingkat akurasi sebesar 96%, *percission* 97%, dan *recall* 97%. Kemudian, pengaruh besar atau kecilnya nilai *learning rate* pada penelitian ini tidak memberikan pengaruh terhadap tingkat akurasi yang didapatkan.

### 4.2.5 Pengaruh Variasi Jumlah *Hidden Layer*

Pada penelitian selanjutnya yakni tentang pengaruh perbedaan variasi *hidden layer* dengan parameter-parameter yang terbaik yang telah didapatkan pada tahap-tahap sebelumnya. Berikut parameter yang digunakan dalam pengujian sebagai berikut:

- *Input layer* : 20 *neuron*
- *Output layer* : 1 *neuron*
- Fungsi aktivasi : *Sigmoid biner*
- Batas *epoch* : 500
- Batas *error* : 0.001
- *Learning rate* : 0.4
- Pembagian dataset : 70:30

Dengan rincian hidden layer untuk masing-masing arsitektur jaringan yang akan diuji dapat dilihat pada Tabel 4.5. Hasil dari pengujian masing-masing skenario dapat dilihat pada Tabel 4.6.

Tabel 4. 5 *Hidden layer* arsitektur pengujian

| Arsitektur ke- | Jumlah hidden layer | Jumlah neuron |
|---|---|---|
| 1 | 1 | 7 |
| 2 | 2 | 7, 4 |
| 3 | 3 | 7, 4, 2 |

Tabel 4. 6 Hasil pengujian arsitektur jaringan

| | | *Hidden Layer* | | |
|---|---|---|---|---|
| | | 1 | 2 | 3 |
| Test | Akurasi | **0.96** | 0.93 | 0.94 |
| | Percission | **0.97** | 0.94 | 0.94 |
| | Recall | **0.97** | 0.94 | 0.94 |
| Train | Akurasi | **0.97** | 0.94 | 0.94 |
| | Percission | **0.97** | 0.95 | 0.95 |
| | Recall | **0.97** | 0.95 | 0.95 |
| | error | **0.03** | 0.07 | 0.06 |

Dari Tabel 4.6, dapat simpulkan bahwa arsitektur dengan jumlah 1 hidden layer merupakan hasil dengan akurasi tertinggi. Dengan demikian, arsitektur tersebut akan digunakan pada tahap pengujian selanjutnya. Penggunaan arsitektur ini dilakukan dengan cara memindahkan bobot dan bias yang diperoleh dari Arsitektur yang didapatkan secara manual, dan melakukan fase *feedforward* dari algoritma *Backpropagation*. Bobot dan bias pelatihan dari arsitektur yang diperoleh dapat dilihat pada Lampiran 2.

### 4.2.6 Pengujian Nilai Ambang (*Threshold*)

Selanjutnya dilakukan proses pencarian nilai ambang (*threshold*) yang terbaik yang digunakan untuk menentukan video termasuk api atau non api dengan menggunakan arsitektur ANN yang telah didapatkan pada pengujian sebelumnya. Adapun video yang digunakan untuk menentukan pembanding terbaik adalah dataset video api dan non api yang digunakan pada proses *training* dan *testing* sebelumnya, dalam artian di pengujian kembali. Pada Tabel 4.7 disajikan hasil pengujian yang didapatkan dari 50 video api dan 50 video non api.

Tabel 4. 7 Hasil pengujian arsitektur jaringan

| | 25% | | 50% | | 75% | |
|---|---|---|---|---|---|---|
| | Api | Non Api | Api | Non Api | Api | Non Api |
| TRUE | 49 | 49 | 46 | 47 | 36 | 41 |
| FALSE | 1 | 1 | 4 | 3 | 14 | 9 |

Dari Tabel 4.7, dapat simpulkan bahwa nilai ambang (*threshold*) yang terbaik untuk menentukan api atau non apidari sebuah video adalah 25%.

### 4.2.7 Validasi

Pada tahap ini, dilakukan proses pengujian ulang terhadap arsitektur yang telah didapatkan dengan menggunakan video yang berbeda dari video yang digunakan pada tahap training dan testing. Dimana video berjumlah 30 buah yang terdiri dari 15 video api dan 15 video non api. Berikut parameter yang digunakan dalam validasi sebagai berikut:

- *Input layer* : 20 *neuron*
- *Output layer* : 1 *neuron*
- Fungsi aktivasi : *Sigmoid biner*
- Batas *epoch* : 500
- Batas *error* : 0.001
- *Learning rate* : 0.4
- Pembagian dataset : 70:30
- Ambang : 25%

Berikut pada tabel 4.8 merupakan tabel hasil pengujian ulang dengan 30 video tersebut.

Tabel 4. 8 Validasi

| Video | Status Sebenarnya | Prediksi | Waktu(menit) | Keterangan |
|---|---|---|---|---|
| 1 | api | api | 6,36 | valid |
| 2 | api | api | 5,44 | valid |
| 3 | api | api | 6,80 | valid |
| 4 | api | api | 7,80 | valid |
| 5 | api | api | 8,27 | valid |
| 6 | api | api | 8,12 | valid |
| 7 | api | api | 3,83 | valid |
| 8 | api | api | 7,68 | valid |
| 9 | api | api | 3,39 | valid |
| 10 | api | api | 2,90 | valid |
| 11 | api | api | 3,21 | valid |
| 12 | api | api | 6,14 | valid |
| 13 | api | api | 6,11 | tidak valid |
| 14 | api | api | 3,89 | valid |
| 15 | api | non api | 2,78 | tidak valid |
| 16 | non api | non api | 11,1 | valid |
| 17 | non api | non api | 4,08 | valid |
| 18 | non api | api | 2,90 | tidak valid |
| 19 | non api | non api | 8,03 | valid |
| 20 | non api | non api | 7,49 | valid |
| 21 | non api | non api | 3,19 | valid |
| 22 | non api | non api | 3,42 | valid |
| 23 | non api | non api | 9,46 | valid |
| 24 | non api | non api | 2,49 | valid |
| 25 | non api | api | 7,96 | tidak valid |
| 26 | non api | non api | 19,4 | valid |
| 27 | non api | non api | 7,57 | valid |
| 28 | non api | non api | 6,36 | valid |
| 29 | non api | non api | 2,06 | valid |
| 30 | non api | non api | 2,64 | valid |

Dari Tabel 4.8, maka tingkat akurasi dan rata-rata waktu yang dibutuhkan untuk mendeteksi api atau non api tersebut dihitung sebagai berikut:

Akurasi $= \frac{26}{30} x100 = 86.6\ \%$

Waktu rata-rata $= \frac{\sum waktu}{30} = 6.029$ menit

Berdasarkan proses validasi yang telah dilakukan diperoleh tingkat akurasi yang cukup baik, yaitu sebesar 86.6% dari 30 video yang diuji dengan waktu rata-rata yang dibutuhkan untuk mendeteksi api atau non api pada sebuah video sejumlah 6.029 menit. Dari hasil tersebut maka dapat disimpulkan bahwa model arsitekur ANN *Backpropagation* tersebut cukup baik untuk mendeteksi api. .Detail jumlah *frame* yang dideteksi api atau non-api dapat dilihat pada lampiran 3.

# BAB V
# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Berdasarkan pengujian dan analisis dari model arsitektur ANN yang telah dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Model arsitektur jaringan *backpropagation* terbaik untuk kasus pendeteksian api pada video ini adalah arsitektur jaringan dengan 1 *hidden layer*, *learning rate* sebesar 0.4, epoch 500, dan pembagian dataset sebesar 70:30, yang menghasilkan akurasi *training* sebesar 97% dan akurasi *testing* sebesar 96% dari 100 data video.
2. Tingkat akurasi model arsitektur ANN setelah di uji menggunakan ulang menggunakan 30 video yang berbeda diperoleh akurasi sebesar 86.6% dengan waktu rata-rata 6,029 menit.
3. Penggunaan hanya menggunakan pixel warna untuk posisi dari api berjalan dengan kurang sempurna. Hal ini karena terdapat beberapa kesalahan dalam penentuan posisi api.

## 5.2 Saran

Adapun saran yang dapat diberikan untuk penelitian lebih lanjut, yaitu:

1. Menggunakan metode segmentasi yang lain karena terdapat beberapa kesalahan dalam proses segmentasi dengan metode *Adaptive Gaussian Mixture* Model (GMM)
2. Menambahkan proses perbaikan kualitas citra pada tahap *pre-processing*
3. Tambahkan proses *resize* pada tahap *pre-processing* karena perbedaan waktu yang dibutuhkan untuk mendeteksi api atau non-api.

# DAFTAR PUSTAKA


[1] BNPB, "Data Informasi Bencana Indonesia." [Online]. Available: http://dibi.bnpb.go.id/. [Accessed: 04-Oct-2019].

[2] S. Dani, A. Mahendra, "Rancang Bangun Sistem Pendeteksi Kebakaran Berbasis Iot Dan Sms Gateway Menggunakan Arduino," Jurnal SIMETRIS, vol. 8, no. 2, November 2017.

[3] A. Prahara, "Deteksi Kebakaran pada Video Berbasis Pengolahan Citra dengan Dukungan GPU," *Semin. Nas. Apl. Teknol. Inf.*, ISSN: 1907 – 5022, 2015.

[4] R. R. Suryadi, I. Wijayanto, A. Rusdinar, "Perancangan Dan Implementasi Sistem Pendeteksi Api Pada Robot Pemadam Api Dengan Menggunakan Sensor Api Dan Design and Implementation System Fire Detection on Fire,", *e-Proceeding of Engineering* vol. 4 no. 3, Desember 2017.

[5] A. Kamarudin, V. Suhartono , R. A. Pramunendar, "Deteksi Api Menggunakan Background Substraction Dan", Jurnal Teknologi Informasi vol. 12 no.1, April 2016.

[6] D. Retnoningrum, A. W. Widodo, M. A. Rahman, "Ekstraksi Ciri Pada Telapak Tangan Dengan Metode Local Binary Pattern ( LBP ),", Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer vol. 3, no. 3, Maret 2019.

[7] R. Amat, J. Y. Sari, I. P. Ningrum, "Implementasi Metode Local Binary Patterns Untuk Pengenalan Pola Huruf Hiragana Dan Katakana Pada Smartphone,", *JUTI J. Ilm. Teknol. Inf.*, vol. 15 no. 2, Juli 2017.

[8] Miladiah, Rusydi. Umar, I. Riadi, "Implementasi Local Binary Pattern untuk Deteksi Keaslian Mata Uang Rupiah," Jepin, vol. 5 no. 2, Agustus 2019.

[9] F. Wibowo, A.Harjoko, "Klasifikasi Mutu Pepaya Berdasarkan Ciri Tekstur GLCM Menggunakan Jaringan Saraf Tiruan," Khazanah Informatika Vol. 3 No.2, Desember 2017.

[10] N. Lihayati, R. E. Pawening, M. Furqan, " Klasifikasi Jenis Daging Berdasarkan Tekstur Menggunakan Metode Gray Level Coocurent Matrix," Prosiding SENTIA Vol. 8- ISSN: 2085-2347, 2016.

[11] R. Widodo, A. W. Widodo, A. Supriyanto, "Pemanfaatan Ciri Gray Level Co-Occurrence Matrix ( GLCM ) Citra Buah Jeruk Keprok ( Citrus reticulata Blanco ) untuk Klasifikasi Mutu," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.* vol. 2 No. 11, November 2018.

[12] *S. Saifudin,* A. Fadlil, "Sistem Identifikasi Citra Kayu Berdasarkan Tekstur Menggunakan Gray Level Coocurrence Matrix ( Glcm ) Dengan Klasifikasi Jarak Euclidean," SINERGI Vol. 19, No. 3, Oktober 2015.

[13] A. Azwar, "Integrasi Ekstraksi Fitur Local Binary Pattern Dan Gray-Level Cooccurence Metrix Untuk Pengenalan Ekspresi Mulut Pembelajar," Ilkom Jurnal Ilmiah, vol. 9, no. 1, 2017.

[14] H. Husdi, "Pengenalan Ekspresi Wajah Pengguna Elearning Menggunakan Artificial Neural Network Dengan Fitur Ekstraksi Local Binary Pattern Dan Gray Level Co-Occurrence Matrix," *Ilkom Jurnal Ilmiah* vol. 8 no. 3, 2016.

[15] Y. Ramdhani, S. Susanti, M. F. Adiwisastra, S. Topiq, "Penerapan Algoritma Neural Network Untuk Klasifikasi Kardiotokografi," Jurnal Informatika vol. 5 no. 1, April 2018.

[16] E. Permata, A. Suherman, A. Maulana, "Klasifikasi Daun Tanaman Theobroma Cacao Menggunakan Metode Neural Network," Seminar. Nasional Teknologi Informasi dan Komunikasi ISSN: 2089-9813, 2014.

[17] B. A. Prabowo, T. A. Budi, F. Sthevanie, "Sistem Deteksi Api Berbasis Spatial Temporal dengan Metode Ekstraksi Kontur dan Area Movement Analysis," Telkom University,.

[18] W. B. Horng, J. W. Peng, "Image-based fire detection using neural networks," Department of Computer Science and Information Engineering, Oktober 2006.

[19] M. Pradana, A. Reventiary, "Pengaruh Atribut Produk Terhadap Keputusan Pembelian Sepatu Merek Customade (Studi Di Merek Dagang Customade Indonesia)," Jurnal.Manajemen vol. 6 no. 1, Juni 2016.

[20] M. Nasir, M. Arhami, "Sistem Pendeteksi Dini Kebakaran Menggunakan Colour Image Processing dan Raspberry Pi," Proceeding Seminar Nasional Politeknik Negeri Lhokseumawe vol. 2 no. 1, September 2018.

[21] M. Dahria, "Kecerdasan buatan ( Artificial Intelligence )," *Artificial Intell.*, vol. 1, no. 2, pp. 1–10, 2014.

[22] R. M. Haralick, K. Shanmugam, "Textural Features for Image Classification," *IEEE Trans. Syst. Man. Cybern.*, vol. SMC-3 , 1973.

[23] T. Sree Sharmila, "Efficient analysis of satellite image denoising and resolution enhancement for improving classification accuracy," *Faculty Of Information Andcommunication Engineeringanna Universitychennai* pp. 1–7, 2014.

[24] Y. A. Lesnussa, S. Latuconsina, E. R. Persulessy, "Aplikasi Jaringan Saraf Tiruan Backpropagation untuk Memprediksi Prestasi Siswa SMA (Studi kasus: Prediksi Prestasi Siswa SMAN 4 Ambon)," jurnal matematika Integratif vol. 11 no. 2, Oktober 2015.

[25] Noordama, "Identifikasi varietas durio zibethinus berdasarkan sebaran trikoma daun menggunakan glcm dan knn noordama," Insitut Pertanian Bogor, 2014.

[26] I. N. T. Adnyana, I G. P. S. Wijaya, M. A. Albar, "Jaringan Syaraf Tiruan *Backpropagation* Untuk Peramalan Suhu Minimum dan Maksimum, Kelembaban, Tekanan Udara, Jumlah Hari Hujan, dan Curah Hujan Bulanan Di Kota Mataram," J-Cosine vol. 3, Desember 2019

# LAMPIRAN

## Lampiran 1. Hasil Keseluruhan Percobaan

a. Pembagian dataset

1) 70_30

a) Toleransi 0.001

i. Epoch 100

Tabel 1 Pembagian dataset 70:30, toleransi0.001,epoch 100

| | | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Test | akurasi | 0.8902 | 0.9032 | 0.913 | 0.90867 | 0.9227 | 0.91668 | 0.9214 | 0.896559 | **0.92988** |
| | percission | 0.89 | 0.90 | 0.91 | 0.91 | 0.92 | 0.92 | 0.92 | 0.90 | 0.93 |
| | recall | 0.89 | 0.90 | 0.91 | 0.91 | 0.92 | 0.92 | 0.92 | 0.90 | 0.93 |
| Train | akurasi | 0.8942 | 0.90732 | 0.9180 | 0.916 | 0.92977 | 0.924 | 0.92708 | 0.90352 | 0.936085 |
| | percission | 0.90 | 0.91 | 0.92 | 0.92 | 0.92 | 0.92 | 0.93 | 0.91 | 0.94 |
| | recall | 0.89 | 0.91 | 0.92 | 0.92 | 0.92 | 0.92 | 0.93 | 0.90 | 0.94 |
| | error | 0.0792 | 0.0719 | 0.06545 | 0.06727 | 0.063197 | 0.0648795 | 0.07098 | 0.076548 | 0.0670 |

ii. Epoch 200

Tabel 2 Pembagian dataset 70:30, toleransi 0.001,epoch 200

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.91040 | 0.91928 | 0.92122 | **0.92837** | 0.9160 | 0.9164 | 0.920796 | 0.9262 | 0.8980 |
| | 0.91 | 0.92 | 0.92 | 0.93 | 0.92 | 0.92 | 0.92 | 0.93 | 0.90 |
| | 0.91 | 0.92 | 0.92 | 0.93 | 0.92 | 0.92 | 0.92 | 0.93 | 0.90 |
| Train | 0.9110 | 0.92365 | 0.925 | 0.9378478 | 0.91957 | 0.92161 | 0.923376 | 0.933951 | 0.9061 |
| | 0.91 | 0.92 | 0.93 | 0.93 | 0.92 | 0.92 | 0.92 | 0.93 | 0.91 |
| | 0.91 | 0.92 | 0.93 | 0.93 | 0.92 | 0.92 | 0.92 | 0.93 | 0.91 |
| | 0.06399 | 0.0561 | 0.052664 | 0.05026 | 0.060421 | 0.06121 | 0.0614866 | 0.0596 | 0.0663 |

iii. Epoch 300

Tabel 3 Pembagian dataset 70:30, toleransi 0.001,epoch 300

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.9251 | 0.9197 | 0.927 | 0.93 | **0.9372** | 0.93334 | 0.92187 | 0.9249 | 0.911 |
| | 0.93 | 0.92 | 0.93 | 0.93 | 0.94 | 0.93 | 0.92 | 0.93 | 0.91 |
| | 0.93 | 0.92 | 0.93 | 0.93 | 0.94 | 0.93 | 0.92 | 0.92 | 0.91 |
| Train | 0.9297 | 0.92161 | 0.933951 | 0.93487 | 0.9410 | 0.936 | 0.9229 | 0.92838 | 0.91762 |
| | 0.93 | 0.92 | 0.94 | 0.93 | 0.94 | 0.94 | 0.92 | 0.93 | 0.92 |
| | 0.93 | 0.92 | 0.94 | 0.93 | 0.94 | 0.94 | 0.92 | 0.93 | 0.92 |
| | 0.0540 | 0.0630 | 0.0483 | 0.05869 | 0.05549 | 0.05922 | 0.05497 | 0.05956 | 0.06191 |

iv. Epoch 400

Tabel 4 Pembagian dataset 70:30, toleransi 0.001,epoch 400

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.910 | 0.92469 | 0.9275 | 0.9288 | 0.93442 | **0.94460** | 0.9320 | 0.92534 | 0.9277 |
| | 0.91 | 0.92 | 0.93 | 0.93 | 0.93 | 0.94 | 0.93 | 0.93 | 0.93 |
| | 0.91 | 0.92 | 0.93 | 0.93 | 0.93 | 0.94 | 0.93 | 0.93 | 0.93 |
| Train | 0.9189 | 0.93423 | 0.9356 | 0.933 | 0.940 | 0.9508 | 0.9381 | 0.9274 | 0.932282 |
| | 0.92 | 0.93 | 0.94 | 0.93 | 0.94 | 0.95 | 0.94 | 0.93 | 0.93 |
| | 0.92 | 0.93 | 0.94 | 0.93 | 0.94 | 0.95 | 0.94 | 0.93 | 0.93 |
| | 0.0624 | 0.053191 | 0.0443 | 0.053 | 0.054 | 0.04687 | 0.04999 | 0.0531 | 0.0522 |

v. Epoch 500

Tabel 5 Pembagian dataset 70:30, toleransi 0.001,epoch 500

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.93421 | 0.93486 | 0.94784 | **0.967755** | 0.93551 | 0.939407 | 0.937892 | 0.930318 | 0.9428695 |
| | 0.93 | 0.94 | 0.95 | 0.97 | 0.94 | 0.94 | 0.94 | 0.93 | 0.94 |
| | 0.93 | 0.93 | 0.95 | 0.97 | 0.94 | 0.94 | 0.94 | 0.93 | 0.94 |
| Train | 0.9381 | 0.9419 | 0.95565 | 0.9702 | 0.94397 | 0.94192 | 0.942393 | 0.9333 | 0.9474 |
| | 0.94 | 0.94 | 0.96 | 0.97 | 0.94 | 0.94 | 0.94 | 0.93 | 0.95 |
| | 0.94 | 0.94 | 0.96 | 0.97 | 0.94 | 0.94 | 0.94 | 0.93 | 0.95 |
| | 0.05376 | 0.04893 | 0.04371 | 0.0310003 | 0.05109 | 0.05328 | 0.05462 | 0.048839 | 0.04545562 |

b) Toleransi 0.0001

i. Epoch 100

Tabel 6 Pembagian dataset 70:30, toleransi 0.0001,epoch 100

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.87967 | 0.88638 | 0.90153 | 0.9039 | 0.9190 | **0.9197** | 0.9169 | 0.89374 | 0.8714563 |
| | 0.89 | 0.89 | 0.90 | 0.90 | 0.92 | 0.92 | 0.92 | 0.90 | 0.88 |
| | 0.89 | 0.89 | 0.90 | 0.90 | 0.92 | 0.92 | 0.92 | 0.89 | 0.97 |
| Train | 0.8862 | 0.8911 | 0.905287 | 0.90816 | 0.92755 | 0.92189 | 0.92588 | 0.9037105 | 0.8813 |
| | 0.89 | 0.89 | 0.91 | 0.91 | 0.93 | 0.92 | 0.93 | 0.91 | 0.89 |
| | 0.89 | 0.89 | 0.91 | 0.91 | 0.93 | 0.92 | 0.93 | 0.90 | 0.88 |
| | 0.0774 | 0.0717 | 0.06698 | 0.066201 | 0.0708 | 0.07118 | 0.070288 | 0.0726 | 0.073676 |

ii. Epoch 200

Tabel 7 Pembagian dataset 70:30, toleransi 0.0001,epoch 200

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.91538 | 0.92079 | 0.92144 | 0.927288 | **0.93702** | 0.9344 | 0.91322 | 0.92057 | 0.91906 |
| | 0.92 | 0.92 | 0.92 | 0.93 | 0.94 | 0.93 | 0.91 | 0.92 | 0.92 |
| | 0.92 | 0.92 | 0.92 | 0.93 | 0.94 | 0.93 | 0.92 | 0.92 | 0.92 |
| Train | 0.9178 | 0.92393 | 0.92476 | 0.9342 | 0.9412 | 0.93914 | 0.9188 | 0.92551 | 0.91456 |
| | 0.92 | 0.92 | 0.93 | 0.94 | 0.94 | 0.94 | 0.92 | 0.93 | 0.92 |
| | 0.92 | 0.92 | 0.92 | 0.93 | 0.94 | 0.94 | 0.92 | 0.93 | 0.92 |
| | 0.06822 | 0.055874 | 0.059093 | 0.0580 | 0.0649 | 0.0602 | 0.06171 | 0.05845 | 0.06218 |

iii. Epoch 300

Tabel 8 Pembagian dataset 70:30, toleransi 0.0001,epoch 300

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.915 | 0.9236 | 0.918 | 0.92555 | 0.93551 | 0.92144 | 0.933 | **0.93832** | 0.929 |
| | 0.92 | 0.92 | 0.92 | 0.93 | 0.94 | 0.92 | 0.93 | 0.94 | 0.93 |
| | 0.92 | 0.92 | 0.92 | 0.93 | 0.94 | 0.92 | 0.93 | 0.94 | 0.93 |
| Train | 0.92124 | 0.92884 | 0.920871 | 0.93005 | 0.9388682 | 0.92875 | 0.93961 | 0.942486 | 0.9351 |
| | 0.92 | 0.93 | 0.92 | 0.93 | 0.94 | 0.93 | 0.94 | 0.94 | 0.94 |
| | 0.92 | 0.93 | 0.92 | 0.93 | 0.94 | 0.93 | 0.94 | 0.94 | 0.94 |
| | 0.057929 | 0.06063 | 0.06337 | 0.05629 | 0.05595 | 0.057201 | 0.057007 | 0.05604 | 0.0581 |

iv. Epoch 400

Tabel 9 Pembagian dataset 70:30, toleransi 0.0001,epoch 400

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.9266 | 0.9398 | 0.9307 | **0.94135** | 0.93356 | 0.9242 | 0.9333 | 0.93096 | 0.9318 |
| | 0.93 | 0.94 | 0.93 | 0.94 | 0.93 | 0.92 | 0.93 | 0.93 | 0.93 |
| | 0.93 | 0.94 | 0.93 | 0.94 | 0.93 | 0.92 | 0.93 | 0.93 | 0.93 |
| Train | 0.93200 | 0.948330 | 0.93395 | 0.94239 | 0.938126 | 0.92949 | 0.9345 | 0.9331 | 0.93664 |
| | 0.93 | 0.95 | 0.93 | 0.94 | 0.94 | 0.93 | 0.93 | 0.93 | 0.94 |
| | 0.93 | 0.95 | 0.93 | 0.94 | 0.94 | 0.93 | 0.93 | 0.93 | 0.94 |
| | 0.0519 | 0.0425 | 0.05266 | 0.060398 | 0.060578 | 0.05965 | 0.059829 | 0.05236 | 0.06094 |

v. Epoch 500

Tabel 10 Pembagian dataset 70:30, toleransi 0.0001,epoch 500

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.920796 | 0.93789 | 0.94200 | 0.93637 | 0.94330 | 0.937459 | 0.9171 | 0.93810 | **0.9450** |
| | 0.92 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.92 | 0.94 | 0.95 |
| | 0.92 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.92 | 0.94 | 0.95 |
| Train | 0.92866 | 0.94369 | 0.9487 | 0.94656 | 0.95343 | 0.939424 | 0.9315 | 0.946289 | 0.948794 |
| | 0.93 | 0.94 | 0.95 | 0.94 | 0.94 | 0.94 | 0.93 | 0.95 | 0.95 |
| | 0.93 | 0.94 | 0.95 | 0.94 | 0.94 | 0.94 | 0.93 | 0.95 | 0.95 |
| | 0.05035 | 0.04765 | 0.04028404 | 0.042376 | 0.03959 | 0.055860 | 0.04327 | 0.04799 | 0.04953 |

c) Toleransi 0.00001

i. Epoch 100

Tabel 11 Pembagian dataset 70:30, toleransi 0.00001,epoch 100

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.88725 | 0.88249 | 0.8974247 | 0.9091105 | **0.925990** | 0.911274 | 0.91668 | 0.89482 | 0.89764120 |
| | 0.89 | 0.89 | 0.90 | 0.91 | 0.93 | 0.91 | 0.92 | 0.90 | 0.90 |
| | 0.89 | 0.88 | 0.90 | 0.91 | 0.93 | 0.91 | 0.92 | 0.90 | 0.90 |
| Train | 0.8938775 | 0.891836 | 0.9072356 | 0.912708 | 0.9324 | 0.9141 | 0.92217 | 0.90482 | 0.903896 |
| | 0.90 | 0.90 | 0.90 | 0.91 | 0.93 | 0.92 | 0.92 | 0.91 | 0.90 |
| | 0.89 | 0.89 | 0.90 | 0.91 | 0.93 | 0.91 | 0.92 | 0.90 | 0.90 |
| | 0.07916524 | 0.07155 | 0.0671804 | 0.06669441 | 0.06631 | 0.069606 | 0.0698055 | 072672172 | 0.0701014 |

ii. Epoch 200

Tabel 12 Pembagian dataset 70:30, toleransi 0.0001,epoch 400

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.90391 | 0.9158 | 0.91993 | 0.9186323 | **0.934429** | 0.914087 | 0.92793 | 0.922960 | 0.91257 |
| | 0.91 | 0.92 | 0.92 | 0.92 | 0.93 | 0.92 | 0.93 | 0.92 | 0.91 |
| | 0.90 | 0.92 | 0.92 | 0.92 | 0.93 | 0.91 | 0.93 | 0.92 | 0.91 |
| Train | 0.9067717 | 0.9206 | 0.924953 | 0.92560 | 0.94100185 | 0.92068 | 0.937291 | 0.93016 | 0.917996 |
| | 0.91 | 0.92 | 0.93 | 0.93 | 0.94 | 0.92 | 0.94 | 0.93 | 0.92 |
| | 0.91 | 0.92 | 0.92 | 0.93 | 0.94 | 0.92 | 0.94 | 0.93 | 0.92 |
| | 0.0660304 | 0.0592481 | 0.059565 | 0.0566427 | 0.0489256 | 0.061598 | 0.058187 | 0.058361 | 0.0623 |

iii. Epoch 300

Tabel 13 Pembagian dataset 70:30, toleransi 0.00001,epoch 300

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.91127 | 0.932482 | 0.92577 | 0.91603 | 0.93204 | 0.92085 | 0.9284155 | **0.941354** | 0.9337805 |
| | 0.91 | 0.93 | 0.93 | 0.92 | 0.93 | 0.93 | 0.93 | 0.94 | 0.93 |
| | 0.91 | 0.93 | 0.93 | 0.92 | 0.93 | 0.93 | 0.93 | 0.94 | 0.93 |
| Train | 0.91410 | 0.93896 | 0.93497 | 0.914100 | 0.9347866 | 0.938033 | 0.934415 | 0.950 | 0.9389610 |
| | 0.91 | 0.94 | 0.94 | 0.93 | 0.93 | 0.94 | 0.94 | 0.95 | 0.94 |
| | 0.91 | 0.94 | 0.94 | 0.93 | 0.93 | 0.94 | 0.94 | 0.95 | 0.94 |
| | 0.072288 | 0.04717 | 0.059614 | 0.06961 | 0.055385 | 0.052322 | 0.0589968 | 0.055036 | 0.052140 |

iv. Epoch 400

Tabel 14 Pembagian dataset 70:30, toleransi 0.00001,epoch 400

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.931832 | 0.9350789 | 0.936593 | 0.9381086 | **0.945466** | 0.928586 | 0.94222 | 0.920579 | 0.926422 |
| | 0.93 | 0.94 | 0.94 | 0.94 | 0.95 | 0.93 | 0.94 | 0.92 | 0.93 |
| | 0.93 | 0.94 | 0.94 | 0.94 | 0.95 | 0.93 | 0.94 | 0.92 | 0.93 |
| Train | 0.93599 | 0.94174 | 0.941187 | 0.94517 | 0.9538 | 0.93599 | 0.94304 | 0.92820 | 0.93274 |
| | 0.94 | 0.94 | 0.94 | 0.95 | 0.95 | 0.93 | 0.94 | 0.93 | 0.93 |
| | 0.94 | 0.94 | 0.94 | 0.95 | 0.95 | 0.93 | 0.94 | 0.93 | 0.93 |
| | 0.05080697 | 0.049578 | 0.054338 | 0.04371775 | 0.04877 | 0.05546 | 0.049691 | 0.0599 | 0.06316 |

v. Epoch 500

Tabel 15 Pembagian dataset 70:30, toleransi 0.00001,epoch 500

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.9145206 | 0.9335 | 0.93183 | 0.94286 | 0.932698 | 0.9285 | **0.9471975** | 0.935295 | 0.94005 |
| | 0.91 | 0.93 | 0.93 | 0.94 | 0.93 | 0.93 | 0.95 | 0.94 | 0.94 |
| | 0.91 | 0.93 | 0.93 | 0.94 | 0.93 | 0.93 | 0.95 | 0.94 | 0.94 |
| Train | 0.9228 | 0.9418 | 0.9391 | 0.9500927 | 0.94063 | 0.92949 | 0.9542 | 0.94369 | 0.943413 |
| | 0.92 | 0.94 | 0.94 | 0.95 | 0.94 | 0.93 | 0.95 | 0.94 | 0.94 |
| | 0.92 | 0.94 | 0.94 | 0.95 | 0.94 | 0.93 | 0.95 | 0.94 | 0.94 |
| | 0.06028 | 0.041823 | 0.0434788 | 0.042658 | 0.05305 | 0.055465 | 0.043089 | 0.0480523 | 0.05176 |

2) 80_20

a) Toleransi 0.001

i. Epoch 100

Tabel 16 Pembagian dataset 80:20, toleransi 0.001,epoch 100

| | | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Test | akurasi | 0.896137 | 0.89419 | 0.9156 | **0.92502** | 0.91593 | 0.88997 | 0.89256 | 0.91528 | 0.9165 |
| | percission | 0.90 | 0.90 | 0.92 | 0.93 | 0.92 | 0.89 | 0.90 | 0.92 | 0.92 |
| | recall | 0.90 | 0.89 | 0.92 | 0.93 | 0.92 | 0.89 | 0.89 | 0.92 | 0.92 |
| Train | akurasi | 0.895129 | 0.90732 | 0.915259 | 0.92573 | 0.92977 | 0.8968 | 0.89553 | 0.91899 | 0.91834 |
| | percission | 0.90 | 0.90 | 0.92 | 0.93 | 0.92 | 0.90 | 0.90 | 0.92 | 0.92 |
| | recall | 0.90 | 0.90 | 0.92 | 0.93 | 0.92 | 0.90 | 0.90 | 0.92 | 0.92 |
| | error | 0.074525 | 0.070284 | 0.06413 | 0.06464 | 0.06531 | 0.06410 | 0.06151941 | 0.068280 | 0.075904 |

ii. Epoch 200

Tabel 17 Pembagian dataset 80:20, toleransi 0.001,epoch 200

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.90434 | 0.91322 | 0.918632 | 0.931191 | 0.9250243 | 0.9230769 | **0.93313859** | 0.933138 | 0.91755 |
| | 0.90 | 0.91 | 0.92 | 0.93 | 0.93 | 0.92 | 0.93 | 0.93 | 0.92 |
| | 0.90 | 0.91 | 0.92 | 0.93 | 0.93 | 0.92 | 0.93 | 0.93 | 0.92 |
| Train | 0.90760 | 0.91948 | 0.92374 | 0.93449675 | 0.93262 | 0.896834 | 0.938798 | 0.93181 | 0.91801 |
| | 0.91 | 0.92 | 0.92 | 0.94 | 0.93 | 0.93 | 0.94 | 0.93 | 0.92 |
| | 0.91 | 0.92 | 0.92 | 0.93 | 0.93 | 0.93 | 0.94 | 0.93 | 0.92 |
| | 0.06416 | 0.05644 | 0.052549 | 0.058789 | 0.05868 | 0.0607999 | 0.05907 | 0.0603733 | 0.062705 |

iii. Epoch 300

Tabel 18 Pembagian dataset 80:20, toleransi 0.001,epoch 300

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.9263226 | 0.9337877 | 0.93184 | **0.942226** | 0.92989 | 0.940603 | 0.936384 | 0.938331 | 0.934436 |
| | 0.93 | 0.93 | 0.93 | 0.94 | 0.93 | 0.94 | 0.94 | 0.94 | 0.94 |
| | 0.93 | 0.93 | 0.93 | 0.94 | 0.93 | 0.94 | 0.94 | 0.94 | 0.93 |
| Train | 0.929220 | 0.9385551 | 0.939448 | 0.92573 | 0.93246 | 0.94042 | 0.9362 | 0.93587 | 0.91762 |
| | 0.93 | 0.94 | 0.94 | 0.95 | 0.93 | 0.94 | 0.94 | 0.94 | 0.94 |
| | 0.93 | 0.94 | 0.94 | 0.95 | 0.93 | 0.94 | 0.94 | 0.94 | 0.94 |
| | 0.0546896 | 0.0519357 | 0.054828 | 0.04746 | 0.056956 | 0.056544 | 0.062275 | 0.0556113 | 0.05533 |

iv. Epoch 400

Tabel 19 Pembagian dataset 80:20, toleransi 0.001,epoch 400

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.949042 | **0.954235** | 0.943524 | 0.947744 | 0.92956 | 0.93995 | 0.935086 | 0.93346 | 0.94352 |
| | 0.95 | 0.95 | 0.94 | 0.93 | 0.93 | 0.94 | 0.94 | 0.93 | 0.94 |
| | 0.95 | 0.95 | 0.94 | 0.93 | 0.93 | 0.94 | 0.94 | 0.93 | 0.94 |
| Train | 0.95357 | 0.961282 | 0.94472 | 0.952191 | 0.93092 | 0.94188 | 0.932548 | 0.93303 | 0.91834 |
| | 0.95 | 0.96 | 0.94 | 0.95 | 0.93 | 0.94 | 0.93 | 0.93 | 0.95 |
| | 0.95 | 0.96 | 0.94 | 0.95 | 0.93 | 0.94 | 0.93 | 0.93 | 0.95 |
| | 0.04099 | 0.0346553 | 0.051244 | 0.0430737 | 0.053584 | 0.0528 | 0.0590165 | 0.058415 | 0.04645 |

v. Epoch 500

Tabel 20 Pembagian dataset 80:20, toleransi 0.001,epoch 500

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.927296 | **0.953586** | 0.944498 | 0.9522 | 0.9367 | 0.945472 | 0.936384 | 0.9396299 | 0.943524 |
| | 0.93 | 0.95 | 0.94 | 0.95 | 0.94 | 0.95 | 0.94 | 0.94 | 0.94 |
| | 0.93 | 0.95 | 0.94 | 0.95 | 0.94 | 0.95 | 0.94 | 0.94 | 0.94 |
| Train | 0.927922 | 0.95876 | 0.946266 | 0.9567 | 0.93766 | 0.943425 | 0.93547077 | 0.93206 | 0.94821 |
| | 0.93 | 0.96 | 0.95 | 0.96 | 0.94 | 0.94 | 0.94 | 0.93 | 0.95 |
| | 0.93 | 0.96 | 0.95 | 0.96 | 0.94 | 0.94 | 0.94 | 0.93 | 0.95 |
| | 0.060766 | 0.037366 | 0.0465203 | 0.0415 | 0.047308 | 0.05204 | 0.053668 | 0.0564023 | 0.04645 |

b) Toleransi 0.0001

i. Epoch 100

Tabel 21 Pembagian dataset 80:20, toleransi 0.0001,epoch 100

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.89581 | 0.89678 | 0.89905 | **0.92632** | 0.902304 | 0.906199 | 0.91496 | 0.90976 | 0.91853 |
| | 0.90 | 0.90 | 0.90 | 0.93 | 0.90 | 0.91 | 0.92 | 0.91 | 0.92 |
| | 0.90 | 0.90 | 0.90 | 0.93 | 0.90 | 0.91 | 0.91 | 0.91 | 0.92 |
| Train | 0.89423 | 0.89797 | 0.90 | 0.9270 | 0.9056006 | 0.911201 | 0.9056 | 0.90762 | 0.91899 |
| | 0.90 | 0.90 | 0.90 | 0.93 | 0.91 | 0.91 | 0.92 | 0.91 | 0.92 |
| | 0.89 | 0.90 | 0.90 | 0.93 | 0.91 | 0.91 | 0.92 | 0.91 | 0.92 |
| | 0.07486 | 0.07164299 | 0.066377 | 0.065122 | 0.068322 | 0.065349 | 0.06303 | 0.071127 | 0.06916945 |

ii. Epoch 200

Tabel 22 Pembagian dataset 80:20, toleransi 0.0001,epoch 200

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.905874 | 0.92437 | 0.936384 | **0.938007** | 0.92827 | 0.92145 | 0.9302174 | 0.9243752 | 0.93605 |
| | 0.91 | 0.92 | 0.94 | 0.94 | 0.93 | 0.92 | 0.93 | 0.92 | 0.94 |
| | 0.91 | 0.92 | 0.94 | 0.94 | 0.93 | 0.92 | 0.93 | 0.92 | 0.94 |
| Train | 0.9098 | 0.92881 | 0.9418019 | 0.93701 | 0.9340097 | 0.926298 | 0.93717 | 0.92516 | 0.93344 |
| | 0.91 | 0.93 | 0.94 | 0.94 | 0.93 | 0.93 | 0.94 | 0.93 | 0.93 |
| | 0.91 | 0.93 | 0.94 | 0.94 | 0.93 | 0.93 | 0.94 | 0.93 | 0.93 |
| | 0.06435 | 0.06113683 | 0.0449303 | 0.05743500 | 0.06085932 | 0.065480 | 0.056599 | 0.062969 | 0.054264 |

iii. Epoch 300

Tabel 23 Pembagian dataset 80:20, toleransi 0.0001,epoch 300

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.92762 | 0.9357 | **0.9393** | 0.93768 | 0.9292437 | 0.9266 | 0.9354105 | 0.92599 | 0.931515 |
| | 0.93 | 0.94 | 0.94 | 0.94 | 0.93 | 0.93 | 0.94 | 0.93 | 0.93 |
| | 0.93 | 0.94 | 0.94 | 0.94 | 0.93 | 0.93 | 0.94 | 0.93 | 0.93 |
| Train | 0.9310876 | 0.9392 | 0.939448 | 0.94017 | 0.937 | 0.937012 | 0.936363 | 0.933116 | 0.93612 |
| | 0.93 | 0.94 | 0.94 | 0.94 | 0.94 | 0.93 | 0.94 | 0.93 | 0.94 |
| | 0.93 | 0.94 | 0.94 | 0.94 | 0.94 | 0.93 | 0.94 | 0.93 | 0.94 |
| | 0.0538878 | 0.0472225 | 0.05569 | 0.049239 | 0.05316 | 0.0624186 | 0.05558052 | 0.053239 | 0.05264 |

iv. Epoch 400

Tabel 24 Pembagian dataset 80:20, toleransi 0.0001,epoch 400

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.93346 | 0.92534 | 0.9432 | 0.924699 | **0.945472** | 0.9337 | 0.93054 | 0.924375 | 0.9526 |
| | 0.93 | 0.93 | 0.94 | 0.93 | 0.95 | 0.93 | 0.93 | 0.93 | 0.95 |
| | 0.93 | 0.93 | 0.94 | 0.92 | 0.95 | 0.93 | 0.93 | 0.92 | 0.95 |
| Train | 0.93200 | 0.93287 | 0.950162 | 0.924756 | 0.946590 | 0.93336 | 0.933847 | 0.92524 | 0.9545454 |
| | 0.94 | 0.93 | 0.95 | 0.93 | 0.95 | 0.93 | 0.93 | 0.93 | 0.95 |
| | 0.94 | 0.93 | 0.95 | 0.92 | 0.95 | 0.93 | 0.93 | 0.93 | 0.95 |
| | 0.0493521 | 0.055015 | 0.04656 | 0.053687 | 0.05084157 | 0.05096 | 0.059843 | 0.05778 | 0.047605 |

v. Epoch 500

Tabel 25 Pembagian dataset 80:20, toleransi 0.0001,epoch 500

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.934112 | 0.934436 | 0.9380 | **0.94774** | 0.9411 | 0.9419 | 0.946445 | 0.91755 | 0.93768 |
| | 0.93 | 0.93 | 0.94 | 0.95 | 0.95 | 0.94 | 0.95 | 0.92 | 0.94 |
| | 0.93 | 0.93 | 0.94 | 0.95 | 0.95 | 0.94 | 0.95 | 0.92 | 0.94 |
| Train | 0.934902 | 0.930519 | 0.9399350 | 0.9494318 | 0.949431 | 0.94375 | 0.94918 | 0.91387 | 0.935795 |
| | 0.94 | 0.93 | 0.94 | 0.95 | 0.95 | 0.94 | 0.95 | 0.92 | 0.94 |
| | 0.93 | 0.93 | 0.94 | 0.95 | 0.95 | 0.94 | 0.95 | 0.91 | 0.94 |
| | 0.0588178 | 0.0599047 | 0.051462 | 0.045165 | 0.05259 | 0.053976 | 0.05118674 | 0.05688 | 0.05708 |

c) Toleransi 0.00001

i. Epoch 100

Tabel 26 Pembagian dataset 80:20, toleransi 0.00001,epoch 100

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.8967 | 0.88997 | 0.91139 | 0.91074 | **0.913988** | 0.91074 | 0.91041 | 0.906848 | 0.9117 |
| | 0.90 | 0.89 | 0.91 | 0.91 | 0.92 | 0.91 | 0.91 | 0.91 | 0.91 |
| | 0.90 | 0.89 | 0.91 | 0.91 | 0.91 | 0.91 | 0.91 | 0.91 | 0.91 |
| Train | 0.895779 | 0.89529 | 0.914123 | 0.9103084 | 0.915178 | 0.9181 | 0.90974 | 0.90868 | 0.91501 |
| | 0.90 | 0.90 | 0.91 | 0.91 | 0.92 | 0.92 | 0.91 | 0.91 | 0.92 |
| | 0.90 | 0.90 | 0.91 | 0.91 | 0.92 | 0.92 | 0.91 | 0.91 | 0.92 |
| | 0.075540 | 0.070979 | 0.0678738 | 0.06789 | 0.069749 | 0.06878781 | 0.070801 | 0.07465 | 0.067639 |

ii. Epoch 200

Tabel 27 Pembagian dataset 80:20, toleransi 0.00001,epoch 200

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.91366 | 0.9276 | 0.9302 | 0.9102 | 0.90003 | 0.93216 | **0.93638** | 0.92697 | 0.93508 |
| | 0.91 | 0.93 | 0.93 | 0.91 | 0.91 | 0.93 | 0.94 | 0.93 | 0.94 |
| | 0.91 | 0.93 | 0.93 | 0.91 | 0.90 | 0.93 | 0.94 | 0.93 | 0.94 |
| Train | 0.9148 | 0.9283 | 0.934577 | 0.9104 | 0.89529 | 0.938961 | 0.936038 | 0.931006 | 0.93457 |
| | 0.91 | 0.93 | 0.93 | 0.91 | 0.90 | 0.94 | 0.94 | 0.93 | 0.93 |
| | 0.91 | 0.93 | 0.93 | 0.91 | 0.90 | 0.94 | 0.94 | 0.93 | 0.93 |
| | 0.0618205 | 0.0558 | 0.05108 | 0.06108 | 0.065687 | 0.0582 | 0.06422 | 0.06114 | 0.066596 |

iii. Epoch 300

Tabel 28 Pembagian dataset 80:20, toleransi 0.00001,epoch 300

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.9406 | 0.928594 | **0.94320** | 0.92534 | 0.92469 | 0.93703 | 0.93541 | 0.935410 | 0.928919 |
| | 0.94 | 0.93 | 0.94 | 0.93 | 0.92 | 0.94 | 0.94 | 0.94 | 0.93 |
| | 0.94 | 0.93 | 0.94 | 0.93 | 0.92 | 0.94 | 0.94 | 0.94 | 0.93 |
| Train | 0.9429 | 0.9313311 | 0.943506 | 0.931087 | 0.926055 | 0.9386363 | 0.941964 | 0.93392857 | 0.9318181 |
| | 0.94 | 0.93 | 0.94 | 0.93 | 0.93 | 0.94 | 0.94 | 0.93 | 0.93 |
| | 0.94 | 0.93 | 0.94 | 0.93 | 0.93 | 0.94 | 0.94 | 0.93 | 0.93 |
| | 0.05008 | 0.05243 | 0.0534794 | 0.048431 | 0.062048 | 0.055509 | 0.052378 | 0.05713146 | 0.0626875 |

iv. Epoch 400

Tabel 29 Pembagian dataset 80:20, toleransi 0.00001,epoch 400

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.9360597 | 0.92762 | 0.937358 | **0.9477** | 0.9419 | 0.938656 | 0.93898 | 0.9334631 | 0.933138 |
| | 0.94 | 0.93 | 0.94 | 0.95 | 0.94 | 0.94 | 0.93 | 0.94 | 0.93 |
| | 0.94 | 0.93 | 0.94 | 0.95 | 0.94 | 0.94 | 0.93 | 0.94 | 0.93 |
| Train | 0.94009 | 0.9238636 | 0.9487 | 0.949594 | 0.94862 | 0.94245 | 0.929788 | 0.941477 | 0.93287337 |
| | 0.94 | 0.92 | 0.95 | 0.95 | 0.95 | 0.94 | 0.93 | 0.94 | 0.93 |
| | 0.94 | 0.92 | 0.95 | 0.95 | 0.95 | 0.94 | 0.93 | 0.94 | 0.93 |
| | 0.0438542 | 0.0592053 | 0.0406160 | 0.048404 | 0.04323 | 0.0477987 | 0.0573752 | 0.04631933 | 0.0579104 |

v. Epoch 500

Tabel 30 Pembagian dataset 80:20, toleransi 0.00001,epoch 500

| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Test | 0.938007 | 0.945472 | **0.9516390** | 0.94060 | 0.93476 | 0.93898 | 0.944498 | 0.937358 | 0.926971 |
| | 0.94 | 0.95 | 0.95 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.93 |
| | 0.94 | 0.95 | 0.95 | 0.94 | 0.93 | 0.94 | 0.94 | 0.94 | 0.93 |
| Train | 0.945616 | 0.945616 | 0.95649 | 0.940503 | 0.93449 | 0.939853 | 0.9441558 | 0.9386363 | 0.928977 |
| | 0.95 | 0.95 | 0.96 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.93 |
| | 0.95 | 0.95 | 0.96 | 0.94 | 0.93 | 0.94 | 0.94 | 0.94 | 0.93 |
| | 0.04682 | 0.049826 | 0.041586 | 0.0483319 | 0.0544938 | 0.048194 | 0.0522074 | 0.05527 | 0.057793 |

3) 90_10

a) Toleransi 0.001

i. Epoch 100

Tabel 31 Pembagian dataset 90:20, toleransi 0.001,epoch 100

| | | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Test | akurasi | 0.887086 | 0.894224 | 0.894873 | 0.909798 | 0.923426 | 0.902011 | **0.929915** | 0.899415 | 0.907203 |
| | percission | 0.89 | 0.90 | 0.90 | 0.91 | 0.92 | 0.91 | 0.93 | 0.90 | 0.91 |
| | recall | 0.89 | 0.89 | 0.89 | 0.91 | 0.92 | 0.90 | 0.93 | 0.90 | 0.91 |
| Train | akurasi | 0.887590 | 0.89430 | 0.893939 | 0.919841 | 0.925468 | 0.904689 | 0.934271 | 0.903535 | 0.911399 |
| | percission | 0.89 | 0.90 | 0.90 | 0.92 | 0.93 | 0.91 | 0.93 | 0.91 | 0.91 |
| | recall | 0.89 | 0.89 | 0.89 | 0.92 | 0.93 | 0.90 | 0.93 | 0.90 | 0.91 |
| | error | 0.075624 | 0.074050 | 0.0722856 | 0.067550 | 0.064934 | 0.063741 | 0.069296 | 0.070435 | 0.069703 |

ii. Epoch 200

Tabel 32 Pembagian dataset 90:20, toleransi 0.001,epoch 200

| | | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Test | akurasi | 0.904607 | 0.927968 | 0.921479 | **0.935756** | 0.926670 | 0.923426 | 0.927319 | 0.915639 | 0.916937 |
| | percission | 0.90 | 0.93 | 0.92 | 0.94 | 0.93 | 0.92 | 0.93 | 0.92 | 0.92 |
| | recall | 0.90 | 0.93 | 0.92 | 0.94 | 0.93 | 0.92 | 0.93 | 0.92 | 0.92 |
| Train | akurasi | 0.917027 | 0.931240 | 0.929653 | 0.937445 | 0.937518 | 0.933044 | 0.938311 | 0.923376 | 0.920995 |
| | percission | 0.90 | 0.93 | 0.93 | 0.94 | 0.94 | 0.93 | 0.94 | 0.92 | 0.92 |
| | recall | 0.90 | 0.93 | 0.93 | 0.94 | 0.94 | 0.93 | 0.94 | 0.92 | 0.92 |
| | error | 0.060589 | 0.053014 | .061071 | 0.057711 | 0.057528 | 0.063504 | 0.053988 | 0.061547 | 0.060570 |

iii. Epoch 300

Tabel 33 Pembagian dataset 90:20, toleransi 0.001,epoch 300

| | | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Test | akurasi | 0.925373 | 0.926670 | **0.937702** | 0.928617 | 0.914990 | 0.929266 | 0.923426 | 0.883841 | 0.933160 |
| | percission | 0.93 | 0.93 | 0.94 | 0.93 | 0.92 | 0.93 | 0.92 | 0.89 | 0.93 |
| | recall | 0.93 | 0.93 | 0.94 | 0.93 | 0.92 | 0.93 | 0.92 | 0.89 | 0.93 |
| Train | akurasi | 0.933838 | 0.939393 | 0.951948 | 0.933549 | 0.937012 | 0.936507 | 0.926911 | 0.885858 | 0.942640 |
| | percission | 0.93 | 0.94 | 0.95 | 0.93 | 0.94 | 0.94 | 0.93 | 0.90 | 0.94 |
| | recall | 0.93 | 0.94 | 0.95 | 0.93 | 0.94 | 0.94 | 0.93 | 0.90 | 0.94 |
| | error | 0.048441 | 0.049440 | 0.047809 | .056001 | 0.047380 | 0.051445 | 0.052934 | 0.062593 | 0.054443 |

iv. Epoch 400

Tabel 34 Pembagian dataset 90:20, toleransi 0.001,epoch 400

| | | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Test | akurasi | 0.931213 | 0.934458 | 0.942245 | 0.929266 | 0.937702 | 0.922777 | **0.940947** | 0.928617 | 0.918883 |
| | percission | 0.93 | 0.93 | 0.94 | 0.93 | 0.94 | 0.92 | 0.94 | 0.93 | 0.92 |
| | recall | 0.93 | 0.93 | 0.94 | 0.93 | 0.94 | 0.92 | 0.94 | 0.93 | 0.92 |
| Train | akurasi | 0.941269 | 0.941847 | 0.946248 | 0.940331 | 0.943650 | 0.935569 | 0.943217 | 0.940548 | 0.931601 |
| | percission | 0.94 | 0.94 | 0.95 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.93 |
| | recall | 0.94 | 0.94 | 0.95 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.93 |
| | error | 0.048703 | 0.053293 | 0.047039 | 0.043159 | 0.053931 | 0.044103 | 0.051611 | 0.047998 | 0.047069 |

v. Epoch 500

Tabel 35 Pembagian dataset 90:20, toleransi 0.001,epoch 500

| | | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Test | akurasi | 0.907203 | 0.929266 | 0.936404 | **0.955223** | 0.933160 | 0.940947 | 0.947436 | 0.945489 | 0.936404 |
| | percission | 0.91 | 0.93 | 0.94 | 0.96 | 0.93 | 0.94 | 0.95 | 0.95 | 0.94 |
| | recall | 0.91 | 0.93 | 0.94 | 0.96 | 0.93 | 0.94 | 0.95 | 0.95 | 0.94 |
| Train | akurasi | 0.927128 | 0.943217 | 0.944372 | 0.964213 | 0.942712 | 0.951154 | 0.951370 | 0.955627 | 0.946608 |
| | percission | 0.93 | 0.94 | 0.94 | 0.96 | 0.94 | 0.95 | 0.95 | 0.96 | 0.95 |
| | recall | 0.93 | 0.94 | 0.94 | 0.96 | 0.94 | 0.95 | 0.95 | 0.96 | 0.95 |
| | error | 0.058862 | 0.049938 | 0.051838 | 0.031853 | 0.053574 | 0.042009 | 0.047996 | 0.036371 | 0.049390 |

b) Toleransi 0.0001

i. Epoch 100

Tabel 36 Pembagian dataset 90:20, toleransi 0.0001,epoch 100

| | | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Test | akurasi | 0.895522 | 0.90395 | 0.919532 | 0.913043 | **0.922128** | 0.922128 | 0.909798 | 0.920181 | 0.920181 |
| | percission | 0.90 | 0.90 | 0.92 | 0.91 | 0.92 | 0.92 | 0.91 | 0.92 | 0.92 |
| | recall | 0.90 | 0.90 | 0.92 | 0.91 | 0.92 | 0.92 | 0.91 | 0.92 | 0.92 |
| Train | akurasi | 0.894372 | 0.908946 | 0.922438 | 0.920923 | 0.936435 | 0.926984 | 0.914502 | 0.920418 | 0.918975 |
| | percission | 0.90 | 0.91 | 0.92 | 0.92 | 0.94 | 0.93 | 0.92 | 0.92 | 0.92 |
| | recall | 0.90 | 0.91 | 0.92 | 0.92 | 0.94 | 0.93 | 0.91 | 0.92 | 0.92 |
| | error | 0.0858196 | 0.064285 | 0.063092 | 0.063014 | 0.058315 | 0.070367 | 0.070975 | 0.071091 | 0.066349 |

ii. Epoch 200

Tabel 37 Pembagian dataset 90:20, toleransi 0.0001,epoch 200

| | | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Test | akurasi | 0.897469 | 0.922777 | 0.935756 | 0.924724 | 0.906554 | 0.911096 | 0.920830 | **0.941596** | 0.918883 |
| | percission | 0.90 | 0.92 | 0.94 | 0.92 | 0.91 | 0.91 | 0.92 | 0.94 | 0.92 |
| | recall | 0.90 | 0.92 | 0.94 | 0.92 | 0.91 | 0.91 | 0.92 | 0.94 | 0.92 |
| Train | akurasi | 0.908297 | 0.931457 | 0.946320 | 0.935209 | 0.916305 | 0.914069 | 0.926334 | 0.941486 | 0.926695 |
| | percission | 0.91 | 0.93 | 0.95 | 0.94 | 0.92 | 0.92 | 0.93 | 0.94 | 0.93 |
| | recall | 0.91 | 0.93 | 0.95 | 0.94 | 0.92 | 0.91 | 0.93 | 0.94 | 0.93 |
| | error | 0.068132 | 0.050493 | 0.052463 | 0.060506 | 0.061912 | 0.061206 | 0.063607 | 0.058470 | 0.054150 |

iii. Epoch 300

Tabel 38 Pembagian dataset 90:20, toleransi 0.0001,epoch 300

| | | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Test | akurasi | 0.926670 | 0.929915 | 0.937702 | 0.93316 | **0.939649** | 0.930564 | 0.923426 | 0.936404 | 0.918883 |
| | percission | 0.93 | 0.93 | 0.94 | 0.93 | 0.94 | 0.93 | 0.92 | 0.94 | 0.92 |
| | recall | 0.93 | 0.93 | 0.94 | 0.93 | 0.94 | 0.93 | 0.92 | 0.94 | 0.92 |
| Train | akurasi | 0.933549 | 0.943578 | 0.949855 | 0.937012 | 0.942424 | 0.940909 | 0.932395 | 0.941774 | 0.926695 |
| | percission | 0.93 | 0.95 | 0.95 | 0.94 | 0.94 | 0.94 | 0.93 | 0.94 | 0.93 |
| | recall | 0.93 | 0.94 | 0.95 | 0.94 | 0.94 | 0.94 | 0.93 | 0.94 | 0.93 |
| | error | 0.052677 | 0.050959 | 0.045287 | 0.052387 | 0.0520319 | 0.054013 | 0.058196 | 0.056269 | 0.054150 |

iv. Epoch 400

Tabel 39 Pembagian dataset 90:20, toleransi 0.0001,epoch 400

| | | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Test | akurasi | 0.903309 | **0.961713** | 0.937053 | 0.930564 | 0.941596 | 0.937053 | 0.927319 | 0.902011 | 0.924724 |
| | percission | 0.90 | 0.96 | 0.94 | 0.93 | 0.94 | 0.94 | 0.93 | 0.90 | 0.93 |
| | recall | 0.90 | 0.96 | 0.94 | 0.93 | 0.94 | 0.94 | 0.93 | 0.90 | 0.93 |
| Train | akurasi | 0.920851 | 0.963997 | 0.940259 | 0.936652 | 0.946031 | 0.945093 | 0.936940 | 0.899494 | 0.935714 |
| | percission | 0.92 | 0.96 | 0.94 | 0.94 | 0.95 | 0.95 | 0.94 | 0.91 | 0.94 |
| | recall | 0.92 | 0.96 | 0.94 | 0.94 | 0.95 | 0.95 | 0.94 | 0.90 | 0.94 |
| | error | 0.062121 | 0.035619 | 0.052923 | 0.055557 | 0.054697 | 0.052367 | 0.051103 | 0.048734 | 0.055685 |

v. Epoch 500

Tabel 40 Pembagian dataset 90:20, toleransi 0.0001,epoch 500

| | | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Test | akurasi | 0.934458 | 0.964308 | 0.923426 | **0.963659** | 0.942245 | 0.935107 | 0.944841 | 0.928617 | 0.924724 |
| | percission | 0.94 | 0.96 | 0.92 | 0.96 | 0.94 | 0.94 | 0.95 | 0.93 | 0.93 |
| | recall | 0.93 | 0.96 | 0.93 | 0.96 | 0.94 | 0.94 | 0.94 | 0.93 | 0.92 |
| Train | akurasi | 0.945382 | 0.968975 | 0.926839 | 0.968326 | 0.944805 | 0.939826 | 0.949783 | 0.931313 | 0.934559 |
| | percission | 0.95 | 0.97 | 0.93 | 0.97 | 0.95 | 0.94 | 0.95 | 0.93 | 0.94 |
| | recall | 0.95 | 0.97 | 0.93 | 0.97 | 0.94 | 0.94 | 0.95 | 0.93 | 0.93 |
| | error | 0.045050 | 0.029909 | 0.059640 | 0.031534 | 0.049552 | 0.055032 | 0.044038 | 0.051511 | 0.053788 |

c) Toleransi 0.00001

i. Epoch 100

Tabel 41 Pembagian dataset 90:20, toleransi 0.00001,epoch 100

| | | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Test | akurasi | 0.892277 | 0.907203 | 0.913043 | 0.902660 | 0.912394 | **0.926022** | 0.908500 | 0.911096 | 0.885139 |
| | percission | 0.89 | 0.91 | 0.91 | 0.91 | 0.91 | 0.93 | 0.91 | 0.91 | 0.89 |
| | recall | 0.89 | 0.91 | 0.91 | 0.90 | 0.91 | 0.93 | 0.91 | 0.91 | 0.89 |
| Train | akurasi | 0.894660 | 0.910606 | 0.914718 | 0.906782 | 0.918326 | 0.932683 | 0.912626 | 0.912481 | 0.886868 |
| | percission | 0.90 | 0.91 | 0.92 | 0.91 | 0.92 | 0.93 | 0.92 | 0.92 | 0.90 |
| | recall | 0.89 | 0.91 | 0.91 | 0.91 | 0.92 | 0.93 | 0.92 | 0.92 | 0.89 |
| | error | 0.074464 | 0.067216 | 0.062946 | 0.062245 | 0.065906 | 0.067069 | 0.068702 | 0.069418 | 0.073856 |

ii. Epoch 200

Tabel 42 Pembagian dataset 90:20, toleransi 0.00001, epoch 200

| | | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Test | akurasi | 0.899415 | 0.929915 | **0.931862** | 0.926670 | 0.913692 | 0.903958 | 0.922128 | 0.917585 | 0.915639 |
| | percission | 0.90 | 0.93 | 0.93 | 0.93 | 0.91 | 0.91 | 0.92 | 0.92 | 0.92 |
| | recall | 0.90 | 0.93 | 0.93 | 0.93 | 0.91 | 0.90 | 0.92 | 0.92 | 0.92 |
| Train | akurasi | 0.907503 | 0.939177 | 0.942784 | 0.936002 | 0.924242 | 0.906421 | 0.927128 | 0.924963 | 0.923737 |
| | percission | 0.91 | 0.94 | 0.94 | 0.94 | 0.92 | 0.91 | 0.93 | 0.93 | 0.92 |
| | recall | 0.91 | 0.94 | 0.94 | 0.94 | 0.92 | 0.91 | 0.93 | 0.92 | 0.92 |
| | error | 0.068900 | 0.055137 | 0.053171 | 0.050336 | 0.061227 | 0.065741 | 0.066063 | 0.059674 | 0.063258 |

iii. Epoch 300

Tabel 43 Pembagian dataset 90:20, toleransi 0.00001,epoch 300

| | | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Test | akurasi | 0.927968 | 0.922777 | **0.941596** | 0.926022 | 0.927319 | 0.938351 | 0.938351 | 0.921479 | 0.917585 |
| | percission | 0.93 | 0.92 | 0.94 | 0.93 | 0.93 | 0.94 | 0.94 | 0.92 | 0.92 |
| | recall | 0.93 | 0.92 | 0.94 | 0.93 | 0.93 | 0.94 | 0.94 | 0.92 | 0.92 |
| Train | akurasi | 0.933982 | 0.928715 | 0.947835 | 0.929870 | 0.933694 | 0.939610 | 0.942135 | 0.932972 | 0.930086 |
| | percission | 0.93 | 0.93 | 0.95 | 0.93 | 0.94 | 0.94 | 0.94 | 0.93 | 0.93 |
| | recall | 0.93 | 0.93 | 0.95 | 0.93 | 0.93 | 0.94 | 0.94 | 0.93 | 0.93 |
| | error | 0.052012 | 0.049653 | 0.049867 | 0.060922 | 0.055402 | 0.052429 | 0.055138 | 0.053401 | 0.060009 |

iv. Epoch 400

Tabel 44 Pembagian dataset 90:20, toleransi 0.00001,epoch 400

| | | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Test | akurasi | 0.9253731 | 0.937053 | **0.937702** | 0.929266 | 0.945489 | 0.931213 | 0.926670 | 0.925373 | 0.934458 |
| | percission | 0.93 | 0.94 | 0.94 | 0.93 | 0.95 | 0.93 | 0.93 | 0.93 | 0.94 |
| | recall | 0.93 | 0.94 | 0.94 | 0.93 | 0.95 | 0.93 | 0.93 | 0.93 | 0.93 |
| Train | akurasi | 0.931529 | 0.940043 | 0.9518037 | 0.934343 | 0.952020 | 0.938455 | 0.940404 | 0.931529 | 0.944877 |
| | percission | 0.93 | 0.94 | 0.95 | 0.93 | 0.95 | 0.94 | 0.94 | 0.93 | 0.95 |
| | recall | 0.93 | 0.94 | 0.95 | 0.93 | 0.95 | 0.94 | 0.94 | 0.93 | 0.94 |
| | error | 0.057504 | 0.054271 | 0.044661 | 0.051373 | 0.042593 | 0.053512 | 0.051348 | 0.054104 | 0.047878 |

v. Epoch 500

Tabel 45 Pembagian dataset 90:20, toleransi 0.00001,epoch 100

| | | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Test | akurasi | 0.935756 | 0.942894 | 0.932511 | 0.933160 | **0.945489** | 0.931213 | 0.933809 | 0.939649 | 0.934458 |
| | percission | 0.94 | 0.94 | 0.93 | 0.93 | 0.95 | 0.93 | 0.93 | 0.94 | 0.94 |
| | recall | 0.94 | 0.94 | 0.93 | 0.93 | 0.95 | 0.93 | 0.93 | 0.94 | 0.93 |
| Train | akurasi | 0.943939 | 0.957864 | 0.942135 | 0.936724 | 0.949206 | 0.938455 | 0.939249 | 0.958008 | 0.944877 |
| | percission | 0.94 | 0.96 | 0.94 | 0.94 | 0.95 | 0.94 | 0.94 | 0.96 | 0.95 |
| | recall | 0.94 | 0.96 | 0.94 | 0.94 | 0.95 | 0.94 | 0.94 | 0.96 | 0.94 |
| | error | 0.042094 | 0.039614 | 0.051871 | 0.048479 | 0.045161 | 0.053512 | 0.051192 | 0.038674 | 0.047878 |

## Lampiran 2. Bobot dan bias pelatihan hasil pelatihan

Tabel 46 Bobot dan Bias dari input layer ke hidden layer 1

| Dari/ke | HN 1.1 | HN 1.2 | HN 1.3 | HN 1.4 | HN 1.5 | HN 1.6 | HN 1.7 |
|---|---|---|---|---|---|---|---|
| IN 1 | -11,91013716 | -28,257 | 20,3237637 | -2,9628 | -26,1 | 2,86709 | 20,0246 |
| IN 2 | -21,45480648 | -2,6345 | 2,60938597 | -18,846 | -3,1542 | -17,854 | 28,8272 |
| IN 3 | -0,318719505 | -25,077 | -0,3898957 | -12,053 | -24,809 | 40,6769 | 14,2571 |
| IN 4 | 6,539911054 | 12,1664 | 9,25669772 | 9,02291 | -6,9526 | -20,002 | -13,942 |
| IN 5 | -11,95452812 | 20,7197 | -14,873022 | 27,5474 | 21,5366 | -4,0685 | 54,965 |
| IN 6 | -21,82458312 | 12,2267 | -16,397038 | 7,19277 | 8,7169 | 23,8237 | 36,4572 |
| IN 7 | -20,6740468 | -23,685 | 28,4525776 | 11,0163 | 2,66483 | 17,896 | 16,5273 |
| IN 8 | -8,545075221 | -21,866 | 26,0809597 | -7,6585 | -18,921 | 40,6941 | 20,6412 |
| IN 9 | 24,03752118 | 11,6321 | -19,911305 | 18,1371 | -2,5585 | -5,6603 | 26,8052 |
| IN 10 | 22,6905245 | 3,39908 | 0,45591259 | 3,91247 | -15,689 | -10,896 | -24,162 |
| IN 11 | 37,24763801 | -7,3645 | 3,04357251 | 9,46892 | -19,523 | -1,4911 | -5,804 |
| IN 12 | 29,53046412 | -8,7039 | 17,507098 | -1,001 | -28,847 | -1,4712 | -33,859 |
| IN 13 | 14,216216631048855, | 17,2622 | 9,58927511 | -15,946 | 19,4069 | -13,661 | -21,969 |
| IN 14 | 18,34177233 | -13,611 | 19,6456113 | 11,0069 | -9,0422 | 26,6642 | -19,546 |
| IN 15 | -8,79119762337928, | 28,2466 | 3,76563098 | 13,3133 | 28,7519 | -61,736 | -10,385 |
| IN 16 | -13,25473675 | -33,926 | 5,57637317 | -19,572 | -0,8112 | 31,4373 | 30,1886 |
| IN 17 | 5,948858346 | 8,90289 | 0,4838078 | 3,74912 | -4,587 | -8,7027 | -1,3267 |
| IN 18 | 3,458127443 | 4,34684 | 1,49751512 | 0,98052 | -6,0076 | -8,4296 | -10,238 |
| IN 19 | 7,247311324 | 5,89488 | 2,02346035 | 1,5292 | -7,0731 | -9,5349 | -7,6423 |
| IN 20 | 5,747686174 | 2,0716 | 4,33840925 | 0,61881 | -9,1536 | -3,8525 | -11,63 |
| bias | 12,01629184 | 15,5276 | -37,748983 | 13,7934 | 7,14914 | -9,5773 | -40,882 |

Keterangan: IN = *Hidden Neuron*, HN = *Hidden Neuron*

Tabel 47 Bobot dan Bias dari *hidden* layer 1 ke *output* layer

| Dari/Ke | O 1 |
|---|---|
| HIN 1 | -11,2633 |
| HIN 2 | 15,26242 |
| HIN 3 | -17,9419 |
| HIN 4 | 23,25298 |
| HIN 5 | 10,13388 |
| HIN 6 | -18,9936 |
| HIN 7 | 16,36816 |
| Bias | -12,3157 |

Keterangan: HIN = *Hidden Neuron*, O 1= *Output*

**Lampiran 3**. Jumlah *Frame* yang dideteksi api atau non-api

Tabel 48 Jumlah frame yang terdeteksi non-api pada video jenis non-api

| NO Video | Non api yang dideteksi | Waktu(menit) | Keterangan |
|---|---|---|---|
| 1 | 92,23300971 | 11,1 | Valid |
| 2 | 70,87378641 | 4,08 | Valid |
| 3 | 0 | 2,90 | Non Valid |
| 4 | 29,12621359 | 8,03 | Valid |
| 5 | 97,08737864 | 7,49 | Valid |
| 6 | 44,66019417 | 3,19 | Valid |
| 7 | 94,17475728 | 3,42 | Valid |
| 8 | 67,96116505 | 9,46 | Valid |
| 9 | 52,42718447 | 2,49 | Valid |
| 10 | 9,708737864 | 7,96 | Non Valid |
| 11 | 62,13592233 | 19,4 | Valid |
| 12 | 74,75728155 | 7,57 | Valid |
| 13 | 95,14563107 | 6364 | Valid |
| 14 | 50,48543689 | 2,06 | Valid |
| 15 | 34,95145631 | 2,64 | Valid |

Tabel 49 Jumlah *frame* yang terdeteksi api pada video jenis api

| NO Video | Api yang dideteksi | Waktu(menit) | Keterangan |
|---|---|---|---|
| 1 | 64,0776699 | 6,36 | Valid |
| 2 | 89,32038835 | 5,44 | Valid |
| 3 | 97,08737864 | 6,80 | Valid |
| 4 | 97,08737864 | 7,80 | Valid |
| 5 | 96,11650485 | 8,27 | Valid |
| 6 | 68,93203883 | 8,12 | Valid |
| 7 | 45,63106796 | 3,83 | Valid |
| 8 | 96,11650485 | 7,68 | Valid |
| 9 | 60,19417476 | 3,39 | Valid |
| 10 | 97,08737864 | 2,90 | Valid |
| 11 | 35,9223301 | 3,21 | Valid |
| 12 | 97,08737864 | 6,14 | Valid |
| 13 | 23,30097087 | 6,11 | Non Valid |
| 14 | 58,25242718 | 3,89 | Valid |
| 15 | 17,47572816 | 2,78 | Non Valid |

**Lampiran 4** Potongan *Code*

```
from math import exp
from random import seed
from random import random

# Initialize a network
def initialize_network(n_inputs, n_hidden, n_outputs):
    network = list()
    hidden_layer = [{'weights': [random() for i in range(n_inputs +
1)]}
                    for i in range(n_hidden)]
    network.append(hidden_layer)
    output_layer = [{'weights': [random() for i in range(n_hidden +
1)]}
                    for i in range(n_outputs)]
    network.append(output_layer)
    return network

# Calculate neuron activation for an input
def activate(weights, inputs):
    activation = weights[-1]
    for i in range(len(weights)-1):
        activation += weights[i] * inputs[i]
    return activation

# Transfer neuron activation, ngitung gradien
def transfer(activation):
    return 1.0 / (1.0 + exp(-activation))


# Forward propagate input to a network output
def forward_propagate(network, row):
    inputs = row
    for layer in network:
        new_inputs = []
        for neuron in layer:
            activation = activate(neuron['weights'], inputs)
            neuron['output'] = transfer(activation)
            new_inputs.append(neuron['output'])
        inputs = new_inputs
    return inputs

# Calculate the derivative of an neuron output
def transfer_derivative(output):
    return output * (1.0 - output)


# Backpropagate error and store in neurons
def backward_propagate_error(network, expected):
    for i in reversed(range(len(network))):
        layer = network[i]
        errors = list()
        if i != len(network)-1:
            for j in range(len(layer)):
                error = 0.0
                for neuron in network[i + 1]:
                    error += (neuron['weights'][j] *
neuron['delta'])
```

```
                errors.append(error)
        else:
            for j in range(len(layer)):
                neuron = layer[j]
                errors.append(expected[j] - neuron['output'])
        for j in range(len(layer)):
            neuron = layer[j]
            neuron['delta'] = errors[j] *
transfer_derivative(neuron['output'])



# Update network weights with error
def update_weights(network, row, l_rate):
    for i in range(len(network)):
        inputs = row
        if i != 0:
            inputs = [neuron['output'] for neuron in network[i - 1]]
        for neuron in network[i]:
            for j in range(len(inputs)):
                neuron['weights'][j] += l_rate * neuron['delta'] *
inputs[j]
            neuron['weights'][-1] += l_rate * neuron['delta']


# Train a network for a fixed number of epochs
def train_network(network, train, targets, l_rate, n_epoch, n_error,
n_outputs):
    error = 0
    for epoch in range(n_epoch):
        sum_error = 0
        for i in range(len(train)):
            # for row, target in train, targets:
            outputs = forward_propagate(network, train[i])
            expected = targets[i]
            sum_error += l_rate*sum([(expected[i]-outputs[i]) **
                                2 for i in range(len(expected))])
            backward_propagate_error(network, expected)
            update_weights(network, train[i], l_rate)
#         sum_error = sum_error/len(train)
        error = sum_error
        if (error<n_error):
            break
    return error
```

**Lampiran 5**. Contoh 3 Frame Dataset

Gambar 1. Contoh 1 Frame dataset api

Gambar 2. Contoh 2 Frame dataset api

Gambar 3. Contoh 3 Frame dataset api

Gambar 4. Contoh 1 Frame dataset non api

Gambar 5. Contoh 2 Frame dataset non api

Gambar 6. Contoh 3 Frame dataset non api