

**KLASIFIKASI PENYAKIT BERDASARKAN DATASET CITRA  
THORAX MENGGUNAKAN ALGORITMA *SUPPORT VECTOR  
MACHINE* (SVM) DENGAN EKSTRAKSI FITUR  
*CONVOLUTIONAL NEURAL NETWORK* (CNN)**

Tugas Akhir

Untuk memenuhi sebagian persyaratan

Mencapai derajat Sarjana S-1 Jurusan Teknik Elektro



Oleh:

**Diaz Dwi Kurniawan**

**F1B 019 040**

**JURUSAN TEKNIK ELEKTRO**

**FAKULTAS TEKNIK**

**UNIVERSITAS MATARAM**

**2023**

## Tugas Akhir

# KLASIFIKASI PENYAKIT BERDASARKAN DATASET CITRA THORAX MENGGUNAKAN ALGORITMA *SUPPORT VECTOR* *MACHINE* (SVM) DENGAN EKSTRAKSI FITUR *CONVOLUTIONAL NEURAL NETWORK* (CNN)

Oleh:

**Diaz Dwi Kurniawan**  
**F1B 019 040**

Telah diperiksa dan disetujui oleh:

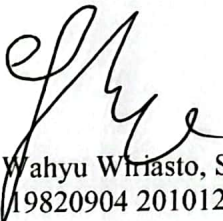
### 1. Pembimbing Utama



L. Ahmad S. Irfan Akbar, S.T., M.Eng.  
NIP: 19830310 200912 1 004

Tanggal: 13/11/2023

### 2. Pembimbing Pendamping



Giri Wahyu Winasto, S.T., M.T.  
NIP: 19820904 201012 1 001

Tanggal: 13/11/2023

Mengetahui  
Ketua Jurusan Teknik Elektro  
Fakultas Teknik  
Universitas Mataram



Sjamsijar Rachman, ST., MT  
NIP: 19711124 199903 1 004

**Tugas Akhir**

**KLASIFIKASI PENYAKIT BERDASARKAN DATASET CITRA  
THORAX MENGGUNAKAN ALGORITMA *SUPPORT VECTOR  
MACHINE* (SVM) DENGAN EKSTRAKSI FITUR  
*CONVOLUTIONAL NEURAL NETWORK* (CNN)**

Oleh:

**Diaz Dwi Kurniawan  
F1B 019 040**

Telah dipertahankan di depan Dewan Penguji

Pada Tanggal **7/11/2023**

dan dinyatakan telah memenuhi syarat mencapai derajat Sarjana S-1  
Jurusan Teknik Elektro

**Susunan Tim Penguji:**

1. Penguji I



Cipta Ramadhani, ST., M.Eng.  
NIP: 198506162019031008

Tanggal: **10 / 11 / 2023**

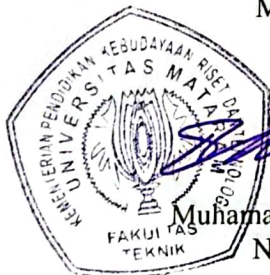
3. Penguji II




Dr. Ir. I Made Ginarsa, S.T., M.T., IPU.  
NIP: 197003251999031001

Tanggal: **9 / 11 / 2023**

Mataram,  
Dekan Fakultas Teknik  
Universitas Mataram



  
Muhammad Syamsu Iqbal, ST., MT., Ph.D.  
NIP: 197202221999031002

## SURAT PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Diaz Dwi Kurniawan  
NIM : F1B019040  
Program Studi : Teknik Elektro  
Perguruan Tinggi : Universitas Mataram

Dengan ini menyatakan bahwa tugas akhir yang telah saya buat dengan judul: “Klasifikasi Penyakit Berdasarkan Dataset Citra *Thorax* Menggunakan Algoritma *Support Vector Machine* (SVM) dengan Ekstraksi Fitur *Convolutional Neural Network* (CNN)” adalah asli (orisinal) atau tidak plagiat (menjiplak) dan belum pernah diterbitkan/dipublikasikan dimanapun dan dalam bentuk apapun.

Demikianlah surat pernyataan ini saya buat dengan sebenar-benarnya tanpa ada paksaan dari pihak manapun juga. Apabila dikemudian hari ternyata saya memberikan keterangan palsu dan atau ada pihak lain yang mengklaim bahwa tugas akhir yang telah saya buat adalah hasil karya milik seseorang atau badan tertentu, saya bersedia diproses baik secara pidana maupun perdata dan kelulusan saya dari Universitas Mataram dicabut/dibatalkan.

Dibuat di : Mataram  
Pada tanggal :  
Yang menyatakan



Diaz Dwi Kurniawan  
NIM. F1B019040

## PRAKATA

Dalam kesempatan ini, penulis ingin mengucapkan rasa syukur dan terima kasih atas dukungan dan bantuan dari berbagai pihak yang telah membantu dalam penyelesaian penelitian ini. Penelitian ini merupakan hasil dari dedikasi dan kerja keras penulis dalam mengeksplorasi dan memahami penelitian ini yang berjudul “Klasifikasi Penyakit Berdasarkan Dataset Citra *Thorax* Menggunakan Algoritma *Support Vector Machine* (SVM) dengan Ekstraksi Fitur *Convolutional Neural Network* (CNN)” dengan lebih mendalam.

Penelitian ini bertujuan untuk melihat tingkat akurasi yang dihasilkan dari hasil klasifikasi menggunakan algoritma CNN sebagai ekstraksi fitur dan SVM sebagai algoritma klasifikasi dalam membedakan citra *thorax* normal, positif tuberkulosis, dan positif pneumonia., serta membuat suatu *website* yang dapat digunakan untuk melakukan klasifikasi citra *thorax*. Penulis berharap hasil penelitian ini dapat memberikan kontribusi positif dalam pengembangan ilmu pengetahuan di bidang *machine learning*. Selama proses penelitian, penulis menghadapi berbagai tantangan dan kesulitan, namun berkat dukungan dan bimbingan dari berbagai pihak, penulis dapat mengatasi setiap hambatan tersebut.

Meskipun penelitian ini telah selesai, penulis menyadari bahwa penelitian ilmiah adalah sebuah proses berkesinambungan. Oleh karena itu, penulis terbuka untuk menerima masukan dan saran dari berbagai pihak guna meningkatkan kualitas penelitian ini di masa mendatang.

Akhir kata, penulis berharap hasil dari penelitian ini dapat memberikan manfaat bagi seluruh masyarakat. Semoga penelitian ini dapat menjadi pijakan untuk penelitian lebih lanjut di bidang yang sama, dan berkontribusi dalam upaya memajukan ilmu pengetahuan dan teknologi.

Mataram, 6 Oktober 2023



Penulis

## UCAPAN TERIMA KASIH

Tugas Akhir ini dapat diselesaikan berkat bimbingan, dukungan dan bantuan dari berbagai pihak, oleh karena itu pada kesempatan kali ini penulis ingin mengucapkan rasa terimakasih yang tulus kepada:

1. Tuhan Yang Maha Esa yang telah memberikan kesehatan, rahmat dan karunia-Nya sehingga penulis selalu tetap semangat dalam menjalani kehidupan ini.
2. Keluarga penulis, terutama kedua orang tua tercinta yaitu Bapak Sudiarto dan Nurhaidawati serta kedua saudara/i penulis yaitu Hafiz Satriawan dan Oktavianti Putri Cahyadi atas dukungan moril dan materil yang tak henti diberikan kepada penulis.
3. Bapak Muhamad Syamsu Iqbal, ST., MT., Ph.D., selaku Dekan Fakultas Teknik Universitas Mataram.
4. Bapak A. Sjamsjiar Rachman, S.T., M.T. selaku Ketua Jurusan Teknik Elektro Universitas Mataram.
5. Bapak L. Ahmad S. Irfan Akbar, S.T., M.Eng. dan Bapak Giri Wahyu Wiriasto, ST., M.T. selaku Dosen Pembimbing yang telah memberikan bimbingan dan arahan selama penyusunan Tugas Akhir ini dilakukan sehingga dapat diselesaikan.
6. Bapak Cipta Ramadhani, S.T., M.Eng., Bapak Djul Fikry Budiman, S.T., M.T. dan Bapak Dr. I Made Ginarsa, S.T., M.T., IPU. selaku Tim Dosen Penguji yang telah memberikan saran, kritik dan arahan untuk meningkatkan kualitas Tugas Akhir ini.
7. Jajaran Dosen, Staff dan Karyawan Teknik Elektro Universitas Mataram yang telah banyak memberikan ilmu yang bermanfaat selama masa perkuliahan.
8. Teman-teman seperjuangan Teknik Elektro 2019 yang telah banyak membantu dan memotivasi penulis dalam menyelesaikan Tugas Akhir ini.
9. Hasna Azkia, perempuan yang terus memberikan semangat dan dukungan. Tetap kebersamaan dan tidak tunduk kepada apa-apa.
10. Semua pihak lain yang tidak dapat disebutkan satu-persatu yang telah membantu baik dalam bentuk saran, kritik dan motivasi.

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

## DAFTAR ISI

|   |      |
|---|------|
| COVER.....  | i    |
| LEMBAR PENGESAHAN PEMBIMBING.....                     | ii   |
| LEMBAR PENGESAHAN PENGUJI.....                        | iii  |
| SURAT PERNYATAAN KEASLIAN TUGAS AKHIR .....           | iv   |
| PRAKATA.....  | v    |
| UCAPAN TERIMA KASIH .....                             | vi   |
| DAFTAR ISI.....                                       | vii  |
| DAFTAR TABEL.....                                     | ix   |
| DAFTAR GAMBAR .....                                   | x    |
| DAFTAR SCRIPT .....                                   | xii  |
| ABSTRAK.....  | xiii |
| ABSTRACT.....   | xiv  |
| BAB I PENDAHULUAN.....                                | 1    |
| 1.1 Latar Belakang .....                              | 1    |
| 1.2 Rumusan Masalah .....                             | 3    |
| 1.3 Batasan Masalah .....                             | 3    |
| 1.4 Tujuan Penelitian .....                           | 4    |
| 1.5 Manfaat Penelitian .....                          | 4    |
| 1.6 Sistematika Penulisan Laporan .....               | 4    |
| BAB II TINJAUAN PUSTAKA DAN DASAR TEORI .....         | 7    |
| 2.1 Tinjauan Pustaka .....                            | 7    |
| 2.2 Dasar Teori .....                                 | 9    |
| 2.2.1 Tuberkulosis (TB).....                          | 9    |
| 2.2.2 Pneumonia.....                                  | 10   |
| 2.2.3 Citra Digital.....                              | 11   |
| 2.2.4 Klasifikasi .....                               | 11   |
| 2.2.5 <i>Machine learning</i> .....                   | 12   |
| 2.2.6 <i>Deep Learning</i> .....                      | 12   |
| 2.2.7 <i>Artificial Neural Network (ANN)</i> .....    | 13   |
| 2.2.8 <i>Convolutional Neural Network (CNN)</i> ..... | 14   |
| 2.2.9 <i>DenseNet-169</i> .....                       | 18   |
| 2.2.10 <i>Support Vector Machine (SVM)</i> .....      | 20   |



|  |    |
|--|----|
| 2.2.11 <i>Performance Evaluation Measure (PEM)</i> .....       | 22 |
| 2.2.12 <i>Streamlit</i> .....                                  | 23 |
| <b>BAB III METODOLOGI PENELITIAN</b> .....                     | 24 |
| 3.1 Alur Pengerjaan Tugas Akhir .....                          | 24 |
| 3.2 Uraian Metodologi .....                                    | 25 |
| 3.2.1 Studi Literatur.....                                     | 25 |
| 3.2.2 Pengumpulan Data.....                                    | 25 |
| 3.2.3 <i>Labelling Dataset</i> .....                           | 26 |
| 3.2.4 <i>Splitting Dataset</i> .....                           | 27 |
| 3.2.5 <i>Preprocessing Data</i> .....                          | 27 |
| 3.2.6 Arsitektur Model .....                                   | 29 |
| 3.2.7 Evaluasi Model.....                                      | 34 |
| 3.2.8 Perancangan Sistem Klasifikasi Citra <i>Thorax</i> ..... | 35 |
| 3.2.9 Desain Aplikasi .....                                    | 37 |
| 3.3 Analisis Kebutuhan .....                                   | 38 |
| <b>BAB IV HASIL DAN PEMBAHASAN</b> .....                       | 39 |
| 4.1 Hasil Pembuatan Model .....                                | 39 |
| 4.1.1 Hasil Pengumpulan Data.....                              | 39 |
| 4.1.2 <i>Labelling Dataset</i> .....                           | 39 |
| 4.1.3 <i>Splitting Dataset</i> .....                           | 41 |
| 4.1.4 <i>Preprocessing</i> .....                               | 41 |
| 4.1.5 <i>Convolutional Neural Network (CNN)</i> .....          | 44 |
| 4.1.6 <i>Support Vector Machine (SVM)</i> .....                | 48 |
| 4.2 Evaluasi Model .....                                       | 50 |
| 4.2.1 <i>Convolutional Neural Network (CNN)</i> .....          | 50 |
| 4.2.2 CNN-SVM .....  | 56 |
| 4.3 Perbandingan Model CNN dan CNN-SVM .....                   | 60 |
| 4.4 Hasil Perancangan Aplikasi Berbasis <i>Web</i> .....       | 61 |
| <b>BAB V KESIMPULAN DAN SARAN</b> .....                        | 64 |
| 5.1 Kesimpulan .....   | 64 |
| 5.2 Saran.....   | 64 |
| <b>DAFTAR PUSTAKA</b> .....                                    | 66 |



## DAFTAR TABEL

|  |    |
|--|----|
| Tabel 2.1. <i>Confusion Matrix</i> .....                             | 22 |
| Tabel 2.2. <i>Confusion Matrix multiclass</i> .....                  | 22 |
| Tabel 3.1. Contoh proses pelabelan data .....                        | 26 |
| Tabel 3.2. Arsitektur model CNN .....                                | 31 |
| Tabel 3.3. <i>Confusion matrix multiclass</i> .....                  | 35 |
| Tabel 4.1. Jumlah dataset citra <i>thorax</i> .....                  | 39 |
| Tabel 4.2. Hasil <i>Splitting Dataset</i> .....                      | 41 |
| Tabel 4.3. <i>Hyperparameter training</i> .....                      | 45 |
| Tabel 4.4. <i>Hyperparameter</i> fungsi <i>EarlyStopping</i> .....   | 46 |
| Tabel 4.5. <i>Hyperparameter</i> fungsi <i>ModelCheckpoint</i> ..... | 46 |
| Tabel 4.6. <i>Training model</i> .....                               | 50 |
| Tabel 4.7. Performa model klasifikasi CNN .....                      | 54 |
| Tabel 4.8. Performa model klasifikasi CNN-SVM .....                  | 58 |
| Tabel 4.9. Perbandingan model CNN dan CNN-SVM .....                  | 60 |

## DAFTAR GAMBAR

|  |    |
|--|----|
| Gambar 2.1. (a) dan (b) Citra <i>thorax</i> pasien tuberkulosis .....                                  | 9  |
| Gambar 2.2. (a) Citra <i>thorax</i> sehat dan (b) Citra <i>thorax</i> pneumonia .....                  | 10 |
| Gambar 2.3. <i>Layer-layer</i> pada <i>Deep Learning</i> .....   | 13 |
| Gambar 2.4. Arsitektur <i>Convolutional Neural Network</i> .....                                       | 14 |
| Gambar 2.5. Proses <i>Convolution Layer</i> .....  | 15 |
| Gambar 2.6. Proses <i>Pooling Layer</i> metode <i>Max Pooling</i> .....                                | 16 |
| Gambar 2.7. Perbandingan dengan <i>Dropout</i> .....   | 17 |
| Gambar 2.8. <i>Layer-layer</i> pada <i>DenseNet-169</i> .....  | 19 |
| Gambar 2.9. Arsitektur <i>DenseNet-169</i> .....   | 20 |
| Gambar 2.10. Alternatif bidang pemisah terbaik.....  | 21 |
| Gambar 3.1. Alur penelitian tugas akhir.....   | 24 |
| Gambar 3.2. Dataset citra <i>thorax</i> normal. ....   | 25 |
| Gambar 3.3. Dataset citra <i>thorax</i> positif tuberkulosis. ....                                     | 25 |
| Gambar 3.4. Dataset citra <i>thorax</i> positif pneumonia. ....  | 26 |
| Gambar 3.5. <i>Flowchart</i> alur <i>preprocessing</i> . ....  | 28 |
| Gambar 3.6. Arsitektur model penelitian. ....  | 29 |
| Gambar 3.7. (a) <i>Dense block</i> dan (b) <i>Transition layer</i> . ....                              | 30 |
| Gambar 3.8. Proses konvolusi <i>layer</i> .....  | 32 |
| Gambar 3.9. Proses <i>pooling layer</i> . ....   | 33 |
| Gambar 3.10. Ilustrasi proses <i>flatten</i> .....   | 33 |
| Gambar 3.11. Alur klasifikasi model SVM. ....  | 34 |
| Gambar 3.12. Diagram alir sistem klasifikasi.....  | 36 |
| Gambar 3.13. <i>Activity diagram user</i> pada halaman aplikasi .....                                  | 36 |
| Gambar 3.14. Desain aplikasi klasifikasi. ....   | 37 |
| Gambar 3.15. Desain aplikasi setelah <i>upload</i> citra. ....   | 37 |
| Gambar 4.1. Hasil <i>labelling dataset</i> . ....  | 41 |
| Gambar 4.2. Perbandingan (a) Citra asli dan (b) Hasil proses <i>CLAHE</i> . ....                       | 42 |
| Gambar 4.3. Perbandingan (a) Citra asli dan (b) Citra hasil proses <i>resize</i> . ....                | 43 |
| Gambar 4.4. <i>Model summary</i> CNN.....  | 45 |
| Gambar 4.5. (a) Grafik <i>accuracy</i> dan (b) Grafik <i>loss</i> pada <i>training</i> model CNN. .... | 51 |
| Gambar 4.6. <i>Confusion Matrix</i> hasil klasifikasi model CNN.....                                   | 52 |

|   |    |
|---|----|
| Gambar 4.7. Grafik <i>performance evaluation measure</i> model CNN. ....                        | 54 |
| Gambar 4.8. <i>Confusion Matrix</i> hasil klasifikasi model CNN-SVM. ....                       | 56 |
| Gambar 4.9. Grafik <i>performance evaluation measure</i> model CNN-SVM.....                     | 58 |
| Gambar 4.10. Grafik perbandingan model CNN dan CNN-SVM. ....                                    | 60 |
| Gambar 4.11. Halaman awal aplikasi <i>web</i> klasifikasi. ....                                 | 61 |
| Gambar 4.12. Tampilan <i>web</i> saat <i>user</i> telah <i>upload</i> citra <i>thorax</i> ..... | 62 |
| Gambar 4.13. Tampilan <i>web</i> hasil klasifikasi citra <i>thorax</i> . ....                   | 62 |

## DAFTAR SCRIPT

|   |    |
|---|----|
| <i>Script 4.1. Labelling dataset.</i> .....                   | 40 |
| <i>Script 4.2. Fungsi CLAHE.</i> .....                        | 41 |
| <i>Script 4.3. Resizing dataset.</i> .....                    | 43 |
| <i>Script 4.4. Rescaling dataset.</i> .....                   | 44 |
| <i>Script 4.5. Augmentasi data.</i> .....                     | 44 |
| <i>Script 4.6. Training model CNN.</i> .....                  | 47 |
| <i>Script 4.7. Menyimpan model ekstraksi fitur CNN.</i> ..... | 48 |
| <i>Script 4.8. Klasifikasi SVM.</i> .....                     | 49 |

## ABSTRAK

Tuberkulosis merupakan suatu penyakit infeksi kronis atau menahun dan menular langsung yang disebabkan oleh bakteri *mycobacterium tuberculosis*. Sedangkan pneumonia merupakan peradangan pada paru-paru yang disebabkan oleh infeksi virus, bakteri, atau jamur. Menurut WHO pada tahun 2019, di Indonesia jumlah kasus penyakit tuberkulosis dengan jumlah sebanyak 842.000 atau 46% dari total kasus yang ada. Pada tahun yang sama, pneumonia merenggut nyawa 740.180 anak dibawah usia 5 tahun di seluruh dunia. Untuk mengurangi tingginya angka kematian tersebut, perlu dilakukan pengendalian dengan cara mendiagnosa penyakit tersebut kepada pasien dengan cepat. Tujuan dari penelitian ini adalah untuk memperoleh akurasi yang tinggi dengan *Convolutional Neural Network* (CNN) dengan arsitektur *Densenet-169* sebagai ekstraksi fitur dan *Support Vector Machine* (SVM) sebagai algoritma klasifikasi. Dataset yang digunakan diperoleh dari *Kaggle* dan berjumlah 7.200 citra yang terdiri dari 4.000 citra normal, 700 citra tuberkulosis, dan 2.500 citra pneumonia. Perancangan *website* dilakukan untuk mengembangkan *website* klasifikasi penyakit citra *thorax* dengan menggunakan model yang telah dibuat. Pada data uji sebanyak 720 citra, didapatkan nilai terbaik dengan model CNN-SVM dengan nilai *Precision* 98,25%, *Recall* 95,38%, dan *F1-score* 96,79% untuk kelas normal. *Precision* 85,71%, *Recall* 95,23%, dan *F1-score* 90,21% untuk kelas TBC. *Precision* 96%, *Recall* 98%, dan *F1-score* 97% pada kelas Pneumonia. Sedangkan nilai *Accuracy model* sebesar 96,25%.

**Kata Kunci:** Tuberkulosis, Pneumonia, *Support Vector Machine*, *Convolutional Neural Network*, *Densenet-169*.

## ABSTRACT

*Tuberculosis is a chronic and directly infectious disease caused by the bacteria mycobacterium tuberculosis. Meanwhile, pneumonia is an inflammation of the lungs caused by viral, bacterial, or fungal infections. According to WHO in 2019, Indonesia had 842,000 cases of tuberculosis or 46% of the total cases. In the same year, pneumonia claimed the lives of 740,180 children under the age of 5 worldwide. To reduce the high mortality rate, it is necessary to control it by diagnosing the disease to patients quickly. The purpose of this study is to obtain high accuracy with Convolutional Neural Network (CNN) with Densenet-169 architecture as feature extraction and Support Vector Machine (SVM) as classification algorithm. The dataset used was obtained from Kaggle and amounted to 7,200 images consisting of 4,000 normal images, 700 tuberculosis images, and 2,500 pneumonia images. Website design is carried out to develop a thorax image disease classification website using the model that has been made. In the test data of 720 images, the best value was obtained with the CNN-SVM model with Precision 98.25%, Recall 95.38%, and F1-score 96.79% for the normal class. Precision 85.71%, Recall 95.23%, and F1-score 90.21% for TB class. Precision 96%, Recall 98%, and F1-score 97% on the Pneumonia class. While the Accuracy value of the model is 96.25%.*

**Keywords:** *Tuberculosis, Pneumonia, Support Vector Machine, Convolutional Neural Network, Densenet-169.*

# **BAB I**

## **PENDAHULUAN**

### **1.1 Latar Belakang**

Penyakit *Tuberculosis* (TB) merupakan suatu penyakit infeksi kronis atau menahun dan menular langsung yang disebabkan oleh bakteri *mycobacterium tuberculosis*. Bakteri ini lebih sering menginfeksi organ paru-paru dibandingkan bagian lain dari tubuh manusia, sehingga selama ini kasus tuberkulosis yang sering terjadi di Indonesia adalah kasus tuberkulosis paru (Indriani et al., 2005).

Pneumonia merupakan peradangan pada paru-paru yang disebabkan oleh infeksi virus, bakteri, atau jamur. Seseorang yang mengalami pneumonia, kantong udara pada paru-parunya akan berisi cairan maupun pus (dahak purulen). Berdasarkan penyebarannya, pneumonia dibedakan menjadi tiga yaitu pneumonia komunitas yang penyebarannya terjadi di komunitas (lingkungan umum), pneumonia yang ditularkan di rumah sakit, dan pneumonia yang penyebarannya melalui alat bantu pernapasan atau ventilator (Makarim, 2022).

Dari data *World Health Organization* (2019), Indonesia adalah Negara yang menduduki peringkat ketiga dalam jumlah kasus penyakit tuberkulosis dengan jumlah sebanyak 842.000 atau 46% dari total kasus yang ada. Kemudian di tahun yang sama, penyakit pneumonia merenggut nyawa 740.180 anak dibawah usia 5 tahun di seluruh dunia, terhitung 14% kematian anak dibawah 5 tahun disebabkan oleh pneumonia (*World Health Organization*, 2019).

Dalam mendiagnosis penyakit ini, umumnya teknik diagnosis yang dilakukan adalah melakukan rontgen dengan sinar-x pada bagian dada pasien. Pada hasil rontgen dapat dilihat gambaran permukaan dalam dada pasien yang digunakan sebagai bahan pertimbangan oleh seorang ahli untuk mengetahui apakah seorang pasien terjangkit penyakit tuberkulosis atau pneumonia.

Seiring pesatnya kemajuan teknologi setiap tahunnya, ilmu teknologi telah dimanfaatkan oleh manusia dalam segala bidang kehidupannya, termasuk dalam bidang kesehatan. Salah satu ilmu teknologi yang dimanfaatkan dalam bidang kesehatan adalah *Computer Vision*, dimana *Computer Vision* ini menggunakan teknik pembelajaran mesin atau *Machine Learning*. *Machine learning* merupakan cabang dari kecerdasan buatan / *Artificial intelligence* (AI) dan ilmu komputer yang berfokus



pada penggunaan data dan algoritma untuk meniru cara manusia belajar dan secara bertahap dapat meningkatkan akurasi (IBM, 2020).

Salah satu metode yang bisa diterapkan pada *deep learning* adalah metode klasifikasi dengan algoritma *Convolutional Neural Network* (CNN). CNN adalah salah satu jenis *neural network* yang biasa digunakan pada pengolahan data citra. CNN termasuk dalam jenis *deep learning* karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra. Pada penelitian terkait tentang implementasi *deep learning* menyatakan bahwa CNN memiliki beberapa lapisan (*layer*) yang mengekstrak informasi dari gambar dan menentukan klasifikasi dari gambar berupa skor klasifikasi. Adapun beberapa model arsitektur CNN yang sudah tersedia di Keras seperti Densenet169, VGG16, Inception, ResNetV2, MobileNet, NasNetMobile dan masih banyak lagi (Keras, 2022).

Arsitektur DenseNet-169 merupakan varian DenseNet yang memiliki kedalaman *layer* sebesar 169. DenseNet menghubungkan langsung satu *convolution layer* dengan semua *convolution layer* setelahnya, di mana semua *layer* mengambil *input* berupa *feature map* dari semua *layer* sebelumnya. Dengan semakin banyak hubungan langsung, arsitektur ini dapat menjadi lebih akurat dan lebih efisien karena koneksi antara *layer* dekat *input* dan *layer* dekat *output* semakin sedikit (Huang et al, 2017)

*Support Vector Machine* (SVM) adalah salah satu metode klasifikasi yang juga sering digunakan, terutama dalam konteks *machine learning* dan pengolahan data. SVM bekerja dengan cara mencari *hyperplane* terbaik yang dapat memisahkan dua kelas data dengan jarak maksimum. *Hyperplane* ini adalah batas keputusan yang memaksimalkan *margin* antara dua kelas (Aziz, 2017).

Berdasarkan uraian diatas, maka dalam penelitian ini penulis akan melakukan penelitian dengan judul **“Klasifikasi Penyakit Berdasarkan Dataset Citra Thorax Menggunakan Algoritma Support Vector Machine (SVM) dengan Ekstraksi Fitur Convolutional Neural Network (CNN)”**. Pada penelitian ini diharapkan mampu mendapatkan hasil klasifikasi yang baik dengan penggunaan algoritma CNN sebagai ekstraksi fitur karena CNN memiliki kemampuan untuk menangkap hierarki fitur mulai dari fitur sederhana seperti tepian hingga fitur kompleks seperti tekstur atau bentuk yang lebih abstrak dan algoritma SVM sebagai algoritma klasifikasi karena SVM merupakan salah satu algoritma klasifikasi yang kuat dan memiliki performa yang baik dalam berbagai jenis tugas klasifikasi. Selain itu, proses ekstraksi

fitur dengan CNN pada penelitian ini menggunakan arsitektur DenseNet169 karena memiliki performa yang baik dalam mengatasi masalah *overfitting* dan mampu mencapai akurasi yang tinggi pada tugas pengolahan citra yang kompleks. Hasil akhir pada penelitian ini juga akan dilakukan implementasi model *machine learning* kedalam bentuk sistem aplikasi klasifikasi berbasis *web* menggunakan *framework streamlit*, sehingga proses klasifikasi dapat dilakukan dalam suatu sistem aplikasi berbasis *web*.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan sebelumnya, maka permasalahan yang akan dikaji dalam penelitian ini adalah sebagai berikut:

- a. Bagaimana tingkat akurasi yang dihasilkan dari hasil klasifikasi menggunakan algoritma *Convolutional Neural Network* (CNN) sebagai ekstraksi fitur dan *Support Vector Machine* (SVM) sebagai algoritma klasifikasi berdasarkan dataset yang digunakan?
- b. Bagaimana pengimplementasian metode klasifikasi menggunakan algoritma *Convolutional Neural Network* (CNN) sebagai ekstraksi fitur dan *Support Vector Machine* (SVM) sebagai algoritma klasifikasi berdasarkan citra *thorax*?

## 1.3 Batasan Masalah

Untuk menjaga fokus dalam penelitian maka beberapa batasan yang diberikan dalam penelitian adalah sebagai berikut:

- a. Data yang digunakan pada penelitian ini merupakan data pada *Kaggle*.
- b. Penelitian ini hanya akan memfokuskan pada klasifikasi yang meliputi tiga kelas, yaitu citra *thorax* normal, positif tuberkulosis, dan positif pneumonia.
- c. Metode yang akan digunakan adalah metode algoritma CNN sebagai ekstraksi fitur dengan arsitektur *DenseNet169* dan menggunakan metode SVM sebagai algoritma pengklasifikasian.
- d. *Output* dari penelitian ini adalah memberikan hasil klasifikasi citra *thorax* normal, positif tuberkulosis, dan positif pneumonia dari metode algoritma CNN sebagai ekstraksi fitur dan SVM sebagai algoritma pengklasifikasian.

#### 1.4 Tujuan Penelitian

Adapun tujuan yang akan dicapai dari penelitian yang akan dilakukan adalah sebagai berikut:

- a. Untuk mengetahui algoritma CNN sebagai ekstraksi fitur dan SVM sebagai algoritma klasifikasi dalam melakukan klasifikasi citra *thorax* normal, positif tuberkulosis dan positif pneumonia.
- b. Untuk mengetahui tingkat akurasi yang dihasilkan dari hasil klasifikasi menggunakan algoritma CNN sebagai ekstraksi fitur dan SVM sebagai algoritma klasifikasi dalam membedakan citra *thorax* normal, positif tuberkulosis, dan positif pneumonia.
- c. Untuk mengimplementasikan pada *website* metode klasifikasi menggunakan algoritma CNN sebagai ekstraksi fitur dan SVM sebagai algoritma klasifikasi dalam mengklasifikasikan citra *thorax* normal, positif tuberkulosis, dan positif pneumonia.

#### 1.5 Manfaat Penelitian

Berdasarkan dengan permasalahan dan tujuan penelitian, maka penulis mengharapkan penelitian ini dapat memberikan beberapa manfaat antara lain:

- a. Memberikan kontribusi pada bidang medis dan masyarakat dengan mengembangkan model CNN sebagai ekstraksi fitur dan SVM sebagai algoritma klasifikasi yang dapat membantu diagnosis penyakit tuberkulosis dan pneumonia secara lebih efektif.
- b. Memberikan pengetahuan lebih kepada penulis dalam mengklasifikasikan citra *thorax* normal, positif tuberkulosis dan positif pneumonia menggunakan metode algoritma CNN sebagai ekstraksi fitur dan SVM sebagai algoritma klasifikasi.
- c. Memberikan hasil performa tingkat akurasi pengklasifikasian menggunakan metode algoritma CNN sebagai ekstraksi fitur dan SVM sebagai algoritma klasifikasi.

#### 1.6 Sistematika Penulisan Laporan

Adapun sistematika penulisan dari Laporan Tugas Akhir ini terbagi menjadi beberapa bab, sesuai dengan ketentuan yang ditetapkan di Fakultas Teknik, Universitas Mataram adalah sebagai berikut:

## **BAB I PENDAHULUAN**

Pada bab ini berisi gambaran umum dan penjelasan mengenai latar belakang pemilihan judul penelitian, perumusan masalah yang dimana merupakan tulisan singkat berisi pertanyaan tentang topik penelitian yang nantinya akan dijawab oleh penulis sehingga penelitian yang dilakukan memiliki suatu kesimpulan dari hasil analisis dan visualisasi data penelitian yang dilakukan, selain itu terdapat pula batasan masalah yang merupakan batas-batas dari topik penelitian yang sedang dikaji atau diteliti, serta terdapat juga tujuan penelitian dan manfaat penelitian yang merupakan keuntungan yang didapat atau diperoleh oleh berbagai pihak dari penelitian yang dilakukan.

## **BAB II TINJAUAN PUSTAKA DAN DASAR TEORI**

Pada bab ini berisi mengenai landasan teori dan tinjauan pustaka, yang dimana dalam tinjauan pustaka mengulas beberapa penelitian sebelumnya yang sejenis mengenai klasifikasi penyakit, sedangkan pada dasar teori membahas mengenai teori-teori yang berkaitan dengan klasifikasi penyakit yang akan dilakukan pada penelitian yang didapatkan melalui beberapa sumber-sumber seperti jurnal, buku, dsb.

## **BAB III METODE PENELITIAN**

Pada bab ini menjelaskan tentang langkah-langkah dalam pelaksanaan penelitian yang dimana terdiri dari alur penelitian yang membahas mengenai gambaran umum dalam bentuk diagram terkait alur penelitian yang dilakukan dan uraian metodologi yang membahas tahapan yang dilakukan pada alur penelitian diantaranya yaitu, pengambilan data, *preprocessing*, Ekstraksi Fitur menggunakan *Convolutional Neural Network* (CNN), Klasifikasi menggunakan *Support Vector Machine* (SVM), Uji Model, Evaluasi, Visualisasi Data dan Interpretasi.

## **BAB IV HASIL DAN PEMBAHASAN**

Pada bab ini membahas mengenai hasil dari penelitian yang telah dilakukan yang dimana terdiri dari hasil analisa, perancangan sistem, dan klasifikasi yang dilakukan menggunakan algoritma *Convolutional Neural Network* (CNN) sebagai ekstraksi fitur dan *Support Vector Machine* (SVM) sebagai algoritma klasifikasi.

## **BAB V KESIMPULAN DAN SARAN**

Pada bab ini berisi kesimpulan yang merupakan jawaban dari masalah yang dirumuskan dalam bentuk kalimat tanya di rumusan masalah dan berisi saran yang membangun untuk pengembangan yang lebih baik di masa depan.

## BAB II

### TINJAUAN PUSTAKA DAN DASAR TEORI

#### 2.1 Tinjauan Pustaka

Pada tinjauan pustaka ini akan membahas mengenai beberapa penelitian-penelitian yang telah dilakukan sebelumnya yang relevan dan menjadi acuan daripada penelitian yang akan dilakukan oleh penulis.

Hipzi et al., (2023) melakukan penelitian dengan judul “Klasifikasi Pneumonia Pada Augmentasi Citra X-Ray Paru-Paru Menggunakan Metode *Convolution Neural Network* (CNN)”. Tujuan dari penelitian ini untuk memperoleh model CNN dengan nilai akurasi yang tinggi sehingga hasil klasifikasi dapat dijadikan acuan dalam mendiagnosa penyakit pneumonia. Dataset yang digunakan secara keseluruhan berjumlah 7.442 citra dimana data dibagi menjadi dua kategori yaitu kategori pneumonia dan normal. Arsitektur Model CNN yang dibuat memiliki 5 blok konvolusi dengan jumlah channel/kernel tiap blok berturut-urut 16, 32, 64, 128, dan 128 serta 3 *Fully Connected Layer* pada *hidden layer*. Optimasi menggunakan adam *optimizer* dengan *learning rate* 0,001, ukuran *input*  $200 \times 200 \times 1$ , dan *max epoch* 50. Sedangkan fungsi aktivasi menggunakan fungsi *Rectifier Linear Unit* (ReLU) dan Sigmoid. Hasil pengujian model CNN mendapatkan nilai akurasi sebesar 95,36%. Berdasarkan hasil pengujian tersebut, model CNN dapat dikatakan mampu untuk mengklasifikasi suatu citra X-Ray paru-paru termasuk kategori pneumonia atau normal.

Soni et al., (2021) melakukan penelitian dengan judul “*Mycobacterium Tuberculosis Detection using Support Vector Machine Classification Approach*”. Pada penelitian ini penulis menyajikan pendekatan baru untuk deteksi otomatis dan klasifikasi *mycobacterium tuberculosis* gambar CT scan paru-paru. Perangkat lunak MATLAB digunakan untuk mengeksekusi ide yang diusulkan. Sistem ini terutama didasarkan pada AHE (*Adaptive Histogram Equalization*), *Embossing*, dan SVM untuk mengekstraksi objek yang diinginkan. Hasil eksperimen menunjukkan bahwa yang dikembangkan algoritma yang dikembangkan mahir dalam menghemat tenaga, kekayaan, dan waktu. Hasil yang disajikan dalam penelitian ini cukup menjanjikan dengan akurasi 96,50%. Dengan demikian, hasil eksperimen mengungkapkan bahwa hanya beberapa sampel yang salah diklasifikasikan tetapi sistem yang dikembangkan

dapat ditingkatkan di masa depan melalui teknik berbasis pembelajaran mesin lainnya berbasis mesin lainnya dengan basis data yang besar.

Rasyid & Heryawan, (2023) melakukan penelitian dengan judul “Klasifikasi Penyakit Tuberculosis (TB) Organ Paru Manusia Berdasarkan Citra *Rontgen Thorax* Menggunakan Metode *Convolutional Neural Network* (CNN)”. Pada penelitian ini dataset yang digunakan penulis berjumlah 2.564 (2.000 normal dan 564 tuberkulosis), selain itu penulis menggunakan arsitektur *MobileNet*. Hasil akhir dari penelitian ini menunjukkan *optimizer* terbaik yaitu Adam menggunakan *preprocessing* CLAHE pada *epoch* 50 batch size 32 dan menghasilkan nilai akurasi validasi sebesar *accuracy* sebesar 96.837%, *precision* 95%, *recall* 93%, *F-1 score* sebesar 93%, dan *loss* sebesar 0.210.

Maka dari semua pemaparan pada penelitian terkait dengan berbagai masalah dan solusi yang ditawarkan, maka peneliti berkeinginan untuk menulis penelitian dengan judul “Klasifikasi Penyakit Berdasarkan Dataset Citra *Thorax* Menggunakan Algoritma *Support Vector Machine* (SVM) dengan Ekstraksi Fitur *Convolutional Neural Network* (CNN)”. Adapun pembaharuan penelitian ini jika dibandingkan dengan penelitian sebelumnya adalah pada penelitian ini telah menggunakan dataset yang lebih banyak dari beberapa penelitian sebelumnya, penggunaan arsitektur *DenseNet169* pada CNN untuk ekstraksi fitur, serta pengimplementasian metode klasifikasi kedalam *web*.

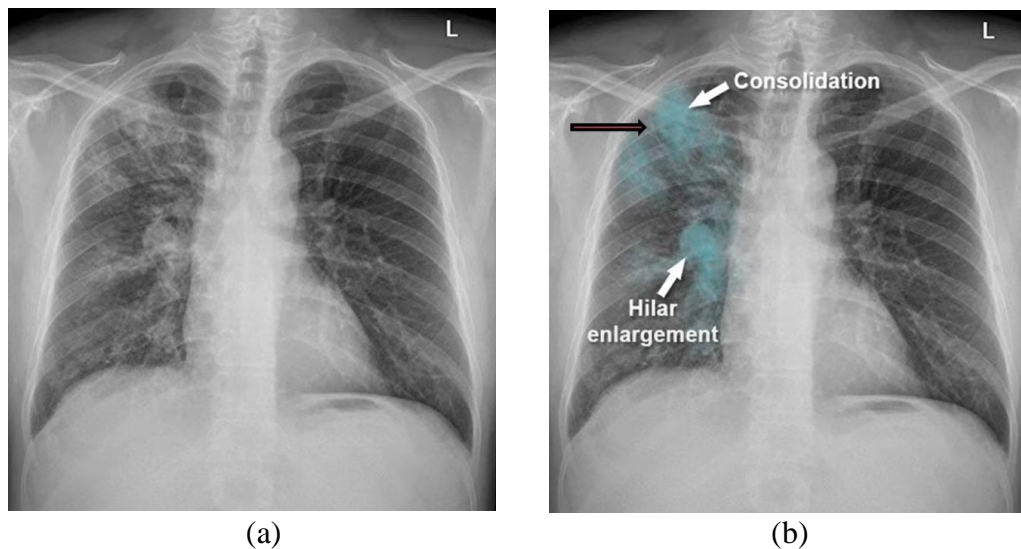
Selain itu, penelitian ini menggunakan dua metode algoritma, yaitu metode *Convolutional Neural Network* (CNN) sebagai ekstraksi fitur dan *Support Vector Machine* (SVM) sebagai algoritma klasifikasi. CNN dipilih sebagai metode untuk ekstraksi fitur karena mampu secara efektif mengekstraksi fitur dari gambar dan dengan menggunakan arsitektur *DenseNet169* karena memiliki performa yang lebih baik dalam mengatasi masalah *overfitting* dan mampu mencapai akurasi yang lebih tinggi. Sementara itu, SVM dipilih sebagai metode klasifikasi karena mampu menghasilkan keputusan yang akurat dengan cepat, terutama pada data yang memiliki dimensi tinggi seperti ekstraksi fitur yang dihasilkan dari CNN. SVM juga dapat bekerja dengan baik pada data yang memiliki pemisah yang jelas, sehingga sangat cocok untuk mengklasifikasikan data dalam kategori yang berbeda.



## 2.2 Dasar Teori

### 2.2.1 Tuberkulosis (TB)

Tuberkulosis (TB) adalah penyakit yang disebabkan oleh *Mycobacterium tuberculosis*. TB dapat menyerang bagian paru-paru dan bisa menyerang semua bagian tubuh. *Mycobacterium tuberculosis* merupakan basil tahan asam. Ditularkan melalui droplet udara dalam bentuk percikan dahak (droplet nuclei/percik renik) yang berasal dari penderita TB paru ataupun TB laring pada waktu batuk, bersin, berbicara, maupun menyanyi. Bakteri ini lebih sering menginfeksi organ paru-paru dibandingkan bagian lain dari tubuh manusia, sehingga selama ini kasus tuberkulosis yang sering terjadi di Indonesia adalah kasus tuberkulosis paru. Gejala yang paling utama pada pengidap TB adalah batuk selama 2 minggu ataupun lebih, gejala batuk ini biasanya juga diikuti dengan gejala lainnya seperti batuk berdarah dan berdahak, mengalami sesak nafas, badan akan menjadi lebih mudah lelah dan lemas, tiap malam hari badan akan mudah berkeringat, serta penderita akan mengalami penurunan nafsu makan. (Indriani et al., 2005).



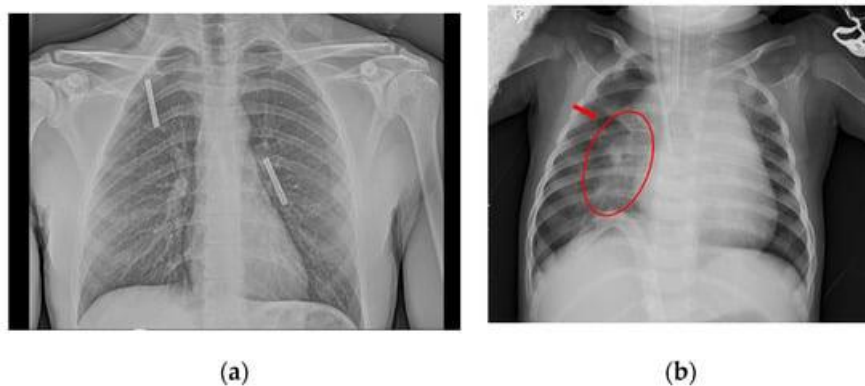
Gambar 2.1. (a) dan (b) citra *thorax* pasien tuberkulosis. (Utami, 2022)

Pada pasien TB paru, rontgen *thorax* dapat menunjukkan bercak atau nodul infiltrat, terutama di lobus atas paru-paru. Selain itu, rontgen toraks juga dapat menunjukkan pembentukan kavitas, nodul kalsifikasi seperti tuberkuloma, dan lesi nodular kecil banyak yang menunjukkan infeksi TB (Utami, 2022)

### 2.2.2 Pneumonia

Secara klinis, pneumonia didefinisikan sebagai suatu peradangan parenkim paru distal dari bronkiolus terminalis yang mencakup bronkiolus respiratorius dan alveoli serta menimbulkan konsolidasi jaringan paru dan gangguan pertukaran gas setempat (Sudoyo, 2005). Pneumonia dibedakan menjadi dua yaitu pneumonia komunitas dan pneumonia nosokomial. Pneumonia komunitas adalah pneumonia yang terjadi akibat infeksi di luar rumah sakit, sedangkan pneumonia nosokomial adalah pneumonia yang terjadi lebih dari 48 jam atau lebih setelah dirawat di rumah sakit.

Pneumonia dapat diklasifikasikan dalam berbagai cara, klasifikasi paling sering ialah menggunakan klasifikasi berdasarkan tempat didapatkannya pneumonia (pneumonia komunitas dan pneumonia nosokomial), tetapi pneumonia juga dapat diklasifikasikan berdasarkan area paru yang terinfeksi (lobar pneumonia, multilobar pneumonia, bronchial pneumonia, dan interstitial pneumonia) atau agen kausatif. Pneumonia juga sering diklasifikasikan berdasarkan kondisi yang mendasari pasien, seperti pneumonia rekuren (pneumonia yang terjadi berulang kali, berdasarkan penyakit paru kronik), pneumonia aspirasi (alkoholik, usia tua), dan pneumonia pada gangguan imun (pneumonia pada pasien transplantasi organ, onkologi, dan AIDS).



Gambar 2.2. (a) Citra *thorax* sehat dan (b) Citra *thorax* pneumonia, cairan dan konsolidasi ditemukan di paru-paru kanan. (Lujan-Garcia, 2020)

Pada hasil rontgen penderita pneumonia menunjukkan ciri-ciri radiologis, seperti konsolidasi oleh akumulasi cairan. Konsolidasi paru adalah suatu kondisi di mana udara yang biasanya mengisi saluran udara kecil di paru-paru digantikan dengan sesuatu yang lain, seperti cairan, benda padat, atau benda lainnya. Hal ini dapat dilihat sebagai gambaran bercak berawan pada lapang paru dalam hasil pemeriksaan rontgen dada (Mayasari, 2019).

### 2.2.3 Citra Digital

Citra (*image*) adalah kombinasi antara titik, garis, bidang dan warna untuk menciptakan suatu imitasi dari suatu objek, biasanya objek fisik atau manusia. Citra bisa berwujud gambar (*picture*) dua dimensi, seperti lukisan, foto dan berwujud tiga dimensi, seperti patung. Citra terbagi 2 yaitu ada citra yang bersifat analog dan ada citra yang bersifat digital. Citra analog tidak dapat direpresentasikan dalam komputer, sehingga tidak bisa diproses oleh komputer secara langsung. Citra analog harus dikonversi menjadi citra digital terlebih dahulu agar dapat diproses di komputer (Sutojo, 2017).

Sebuah citra dapat didefinisikan sebagai fungsi  $f(x, y)$  berukuran M baris dan N kolom, dengan  $x$  dan  $y$  adalah koordinat spasial, dan amplitudo  $f$  di titik koordinat  $(x, y)$  dinamakan intensitas atau tingkat keabuan dari citra pada titik tersebut. Apabila nilai  $x$ ,  $y$  dan nilai amplitudo  $f$  secara keseluruhan berhingga (*finite*) dan bernilai diskrit maka dapat dikatakan bahwa citra tersebut adalah citra digital (Putra, 2010).

### 2.2.4 Klasifikasi

Klasifikasi adalah proses untuk menemukan model atau fungsi yang menggambarkan dan membedakan kelas data atau konsep dengan tujuan memprediksikan kelas untuk data yang tidak diketahui kelasnya. Di dalam klasifikasi, terdapat target variabel kategori. Sebagai contoh, penggolongan pendapatan dapat dipisahkan dalam tiga kategori, yaitu pendapatan tinggi, pendapatan sedang, dan pendapatan rendah. Dalam melakukan klasifikasi data terdapat dua proses yang dilakukan yaitu (Putra, 2018):

#### 1) Proses *Training*

Pada proses *training* digunakan training set yang telah diketahui label-labelnya untuk membangun model atau fungsi.

#### 2) Proses *Testing*

Untuk mengetahui keakuratan model atau fungsi yang akan dibangun pada proses *training*, maka digunakan data yang disebut dengan *testing set* untuk memprediksi label-labelnya.

### 2.2.5 *Machine learning*

*Machine learning* merupakan suatu ilmu yang membuat sistem dapat secara otomatis belajar sendiri tanpa harus berulang kali diprogram oleh manusia. *Machine Learning* sendiri merupakan salah satu disiplin ilmu dalam kecerdasan buatan atau yang sering biasa dikenal dengan *Artificial Intelligent* (AI). *Machine Learning* juga sering disebut dengan *Artificial Intelligent* (AI) konvensional karena merupakan kumpulan metode-metode yang digunakan dalam penerapannya. *Machine Learning* berfokus pada pengembangan program komputer yang dapat mengakses data dan menggunakannya untuk belajar sendiri. Sebelum *Machine Learning* bisa bekerja, ia membutuhkan data untuk training kemudian hasil dari *training* tersebut akan diuji atau dites dengan data yang sama atau bertolak belakang (Imron, 2019). Selain itu, metode dalam *Machine Learning* ini juga dapat menghitung akurasi. *Machine Learning* ini dikembangkan agar dapat mendeteksi pola, mengklasifikasikan pola, menghitung akurasi, serta membuat keputusan. Terdapat tiga metode *Machine Learning* yang sering digunakan, antara lain *supervised learning*, *unsupervised learning*, dan *reinforcement learning* (Khasanah, 2022).

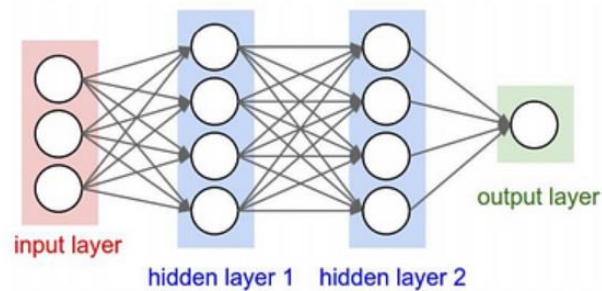
Teknik yang digunakan oleh *Supervised Learning* adalah metode klasifikasi di mana kumpulan data sepenuhnya diberikan label untuk mengklasifikasikan kelas yang tidak dikenal. Sedangkan teknik *Unsupervised Learning* sering disebut *cluster* dikarenakan tidak ada kebutuhan untuk pemberian label dalam kumpulan data dan hasilnya tidak mengidentifikasi contoh di kelas yang telah ditentukan. Sedangkan *Reinforcement Learning* biasanya berada antara *Supervised Learning* dan *Unsupervised Learning*, teknik ini bekerja dalam lingkungan yang dinamis di mana konsepnya harus menyelesaikan tujuan tanpa adanya pemberitahuan dari komputer secara eksplisit jika tujuan tersebut telah tercapai.

### 2.2.6 *Deep Learning*

*Deep Learning* merupakan cabang ilmu dari *Machine Learning* berbasis jaringan saraf tiruan yang mengajarkan komputer untuk melakukan suatu tindakan yang dianggap alami oleh manusia. Dalam *Deep Learning*, sebuah komputer belajar mengklasifikasi secara langsung dari gambar, teks atau suara.

*Deep Learning* adalah teknik dalam *neural network* yang menggunakan teknik tertentu seperti *Restricted Boltzmann Machine* (RBM) untuk mempercepat proses pembelajaran dalam *Neural Network* yang menggunakan lapis yang banyak.

Lapisan pada *Deep Learning* terdiri atas tiga bagian yaitu *input layer*, *hidden layer* dan *output layer*. *Input layer* berisi *node-node* yang masing-masing menyimpan sebuah nilai masukan yang tidak berubah pada fase latih dan hanya bisa berubah jika diberikan nilai masukan baru. Pada *hidden layer* dapat dibuat berlapis-lapis untuk menemukan komposisi algoritma yang tepat agar meminimalisir *error* pada *output*. Kemudian *output layer* berfungsi untuk menampilkan hasil perhitungan sistem oleh fungsi aktivasi pada lapisan *hidden layer* berdasarkan input yang diterima.



Gambar 2.3. *Layer-layer* pada *Deep Learning*.  
(stats.Stackexchange.com, 2017)

Arsitektur pada Gambar 2.3 diatas biasa disebut sebagai *Multi Layer Perceptron* (MLP). Arsitektur pertama mempunyai 3 buah neuron pada input layer dan 2 buah *node output layer*. Diantara *input* dan *output*, terdapat 2 *hidden layer* dengan 4 buah neuron. Neuron-neuron tersebut akan terhubung langsung dengan neuron lain pada *layer* selanjutnya.

*Deep Learning* memungkinkan model komputasi yang terdiri dari beberapa *processing layer* untuk mempelajari representasi data dengan berbagai tingkat abstraksi (Peryanto et al., 2019).

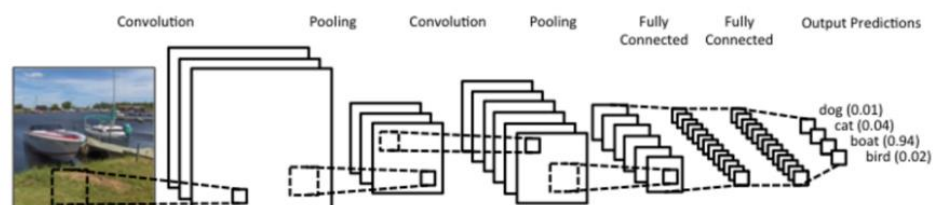
### 2.2.7 *Artificial Neural Network* (ANN)

*Artificial Neural Network* (ANN) adalah salah satu algoritma *supervised learning* yang populer dan bisa juga digunakan untuk *semi-supervised* atau *unsupervised learning*. Walaupun tujuan awalnya adalah untuk mensimulasikan jaringan saraf biologis, jaringan tiruan ini sebenarnya simulasi yang terlalu disederhanakan, artinya simulasi yang dilakukan tidak mampu menggambarkan kompleksitas jaringan biologis manusia. (Putra, 2020)

*Artificial Neural Network* (ANN) menghasilkan model yang sulit dibaca dan dimengerti oleh manusia karena memiliki banyak *layer* (kecuali *single perceptron*) dan sifat non-linier (merujuk pada fungsi aktivasi). Konsep matematis ANN cukup solid, tetapi interpretability model rendah menyebabkan kita tidak dapat menganalisa proses inferensi yang terjadi pada model ANN. Secara matematis, ANN ibarat sebuah graf. ANN memiliki *neuron/node* (vertex), dan sinapsis (*edge*). Karena memiliki struktur seperti graf, operasi pada ANN mudah dijelaskan dalam notasi aljabar linear.

### 2.2.8 *Convolutional Neural Network* (CNN)

*Convolutional Neural Network* (CNN) adalah salah satu algoritma dari *Deep Learning* yang merupakan pengembangan dari *Multi Layer Perceptron* (MLP) yang dirancang untuk mengolah data dalam bentuk *grid*, salah satunya citra dua dimensi, misalnya gambar atau suara. *Convolutional Neural Network* digunakan untuk mengklasifikasikan data yang terlabel dengan menggunakan metode *supervised learning*, yang mana cara kerja dari *supervised learning* adalah terdapat data yang dilatih dan terdapat variabel yang ditargetkan sehingga tujuan dari metode ini adalah mengelompokkan suatu data ke data yang sudah ada.



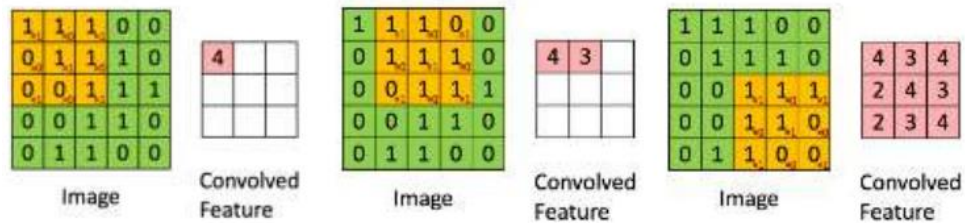
Gambar 2.4. Arsitektur *Convolutional Neural Network*. (Wildml.com, 2015)

*Convolutional Neural Network* menggabungkan tiga pokok arsitektur, yaitu *local receptive fields*, *Shared weight* yang berupa *filter*, dan *spatial subsampling* yang berupa *Pooling*. Konvolusi atau yang biasa disebut dengan *Convolution* merupakan matriks yang berfungsi untuk melakukan *filter*. Arsitektur yang dimiliki oleh *Convolutional Neural Network* sebagai berikut.

#### 1. *Convolution Layer*

*Convolution layer* melakukan operasi konvolusi pada output dari lapisan sebelumnya. *Layer* tersebut adalah proses utama yang mendasari sebuah CNN. *Convolution layer* merupakan lapisan utama yang paling penting untuk

digunakan. Konvolusi merupakan suatu istilah matematis yang dalam pengolahan citra berarti mengaplikasikan sebuah kernel (kotak kuning) pada citra disemua offset yang memungkinkan seperti yang ditunjukkan pada Gambar 2.3, sedangkan kotak berwarna hijau secara keseluruhan merupakan citra yang akan dikonvolusi. Kernel (kotak kuning) bergerak dari sudut kiri atas ke kanan bawah. Sehingga hasil konvolusi dari citra tersebut dapat dilihat pada gambar disebelah kanannya.



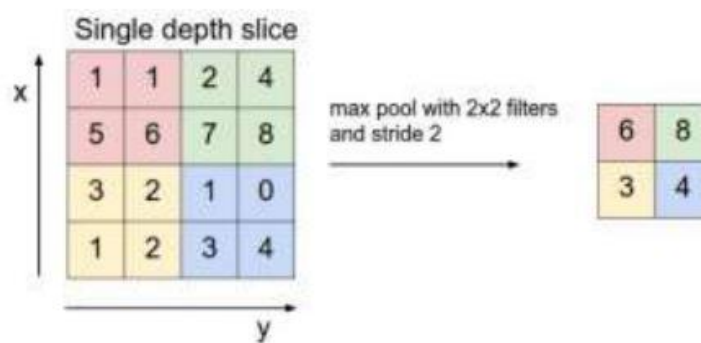
Gambar 2.5. Proses *Convolution Layer*. (Wildml.com, 2015)

Tujuan konvolusi pada data citra adalah untuk mengekstraksi fitur dari citra input. Konvolusi akan menghasilkan transformasi linear dari data input sesuai informasi spasial pada data. Bobot pada layer tersebut menspesifikasikan kernel konvolusi yang digunakan, sehingga kernel konvolusi dapat dilatih berdasarkan input pada CNN. (Stathakis, 2008).

## 2. *Pooling Layer*

*Pooling Layer* merupakan lapisan yang menggunakan fungsi dengan *feature map* sebagai masukan dan mengolahnya dengan berbagai macam operasi statistik berdasarkan nilai piksel terdekat. *Pooling layer* pada model CNN biasanya disisipkan secara teratur setelah beberapa *convolution layer*. *Pooling layer* yang dimasukkan di antara lapisan konvolusi secara berturut-turut dalam arsitektur model *Convolutional Neural Network* dapat secara progresif mengurangi ukuran volume *output* pada *feature map*, sehingga jumlah parameter dan perhitungan di jaringan berkurang, serta untuk mengendalikan *overfitting*. *Pooling layer* digunakan untuk mengambil nilai maksimal (*max-pooling*) atau nilai rata-rata (*average pooling*) dari bagian-bagian piksel pada citra. Metode pooling yang sering digunakan dalam CNN adalah metode *max-pooling*. *Max-pooling* membagi output dari *convolution layer* menjadi beberapa *grid* kecil lalu mengambil nilai maksimal dari setiap *grid* untuk menyusun matriks citra yang telah direduksi seperti yang ditunjukkan pada Gambar 2.4.





Gambar 2.6. Proses *Pooling Layer* metode *Max Pooling*. (wildml.com, 2015)

Kotak yang berwarna merah, hijau, kuning dan biru pada sisi kiri merupakan kelompok kotak yang akan dipilih nilai maksimumnya. Sehingga hasil dari proses tersebut dapat dilihat pada kumpulan kotak disebelah kanannya. Proses tersebut memastikan fitur yang didapatkan akan sama meskipun objek citra mengalami translasi (pergeseran). Penggunaan pooling layer pada CNN hanya bertujuan untuk mereduksi ukuran citra sehingga dapat dengan mudah digantikan dengan sebuah *convolution layer* dengan *stride* yang sama dengan *pooling layer* yang bersangkutan. *Stride* merupakan parameter yang menentukan berapa jumlah pergeseran filter. Jika nilai *stride* adalah satu, maka filter akan bergeser sebanyak satu piksel secara horizontal lalu vertikal. Semakin kecil *stride* yang digunakan, maka semakin detail informasi yang didapatkan dari sebuah input, namun membutuhkan komputasi lebih jika dibandingkan dengan *stride* yang besar.

### 3. *Normalization Layer*

*Normalization layer* (lapisan normalisasi) digunakan untuk menangani perbedaan perbandingan nilai yang sangat berbedapada citra masukan. Para ahli telah mengajukan variasi jenis lapis normalisasi (*normalization layers*). Namun, sampai saat ini lapisan normalisasi tidak banyak digunakan secara praktis karena akibatnya yang tidak terlalu berdampak, atau bahkan tidak ada sama sekali.

### 4. *Activation Function / ReLU Layer*

*Activation function* adalah sebuah node yang dimasukan di akhir keluaran dari setiap jaringan syaraf. Pada arsitektur CNN, fungsi aktivasi dapat digunakan pada perhitungan akhir keluaran *feature map* atau sehabis proses konvolusi maupun *pooling layer*. Terdapat banyak fungsi aktivasi yang banyak digunakan dalam *neural network* yaitu sigmoid, tanh, ReLu (*Rectified Linear*), parameter ReLu, dan leaky ReLu. ReLU layer atau lapisan *Rectified Linear Units* ini menerapkan

fungsi aktivasi  $f(x) = \max(0, x)$ . Ini meningkatkan sifat nonlinearitas fungsi keputusan dan jaringan secara keseluruhan tanpa mempengaruhi bidang-bidang reseptif pada *convolution layer*.

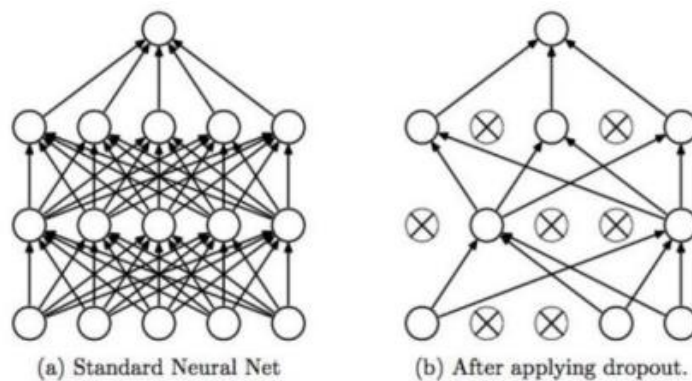
$$\sigma(x) = \max(0, x)$$

#### 5. *Fully Connected Layer*

Asdsad *Fully connected layer* merupakan lapisan dimana semua neuron aktivasi dari lapisan sebelumnya terhubung semua dengan neuron di lapisan selanjutnya. *Fully connected layer* biasanya digunakan dalam penerapan *Multi Layer Perceptron* (MLP) dan bertujuan untuk melakukan transformasi pada dimensi data agar dapat diklasifikasikan secara linear. Perbedaan antara *fully connected layer* dan *convolution layer* biasanya adalah neuron di *convolution layer* terhubung hanya ke daerah tertentu pada input, sedangkan *fully connected layer* mempunyai neuron yang secara keseluruhan terhubung. Namun, kedua lapisan tersebut masih mengoprasikan produk *dot*, sehingga fungsinya tidak begitu berbeda. (Kusumaningrum, 2018)

#### 6. *Dropout Regulation*

*Regularization* merupakan teknik yang dilakukan untuk mengurangi *overfitting* atau *noise* adalah keadaan dimana sistem mampu belajar dengan baik dengan data pelatihan, namun tidak dapat menggeneralisasi dengan data uji. *Dropout* adalah teknik yang dapat digunakan untuk mencegah *overfitting* serta mempercepat proses pelatihan. *Dropout* menghapuskan neuron yang berupa *hidden* maupun *layer* yang tampak pada jaringan. Neuron yang hilang akan dipilih secara random oleh sistem dengan probabilitas dari nol hingga satu. Gambar 2.5 menunjukkan apa yang terjadi ketika adanya *dropout*.



Gambar 2.7. Perbandingan dengan *Dropout*. (Cendekia, 2022)

#### 7. *Cross Entropy Loss Function*

*Loss Function* atau *Cost Function* adalah fungsi yang menjelaskan kerugian yang terkait dengan semua kemungkinan yang dihasilkan oleh model. *Loss Function* bekerja Ketika model pembelajaran memberikan kesalahan yang harus diberi perhatian lebih. *Loss Function* yang baik adalah fungsi yang menghasilkan error terkecil yang diharapkan.

#### 8. Proses *Forward Propagation* pada CNN

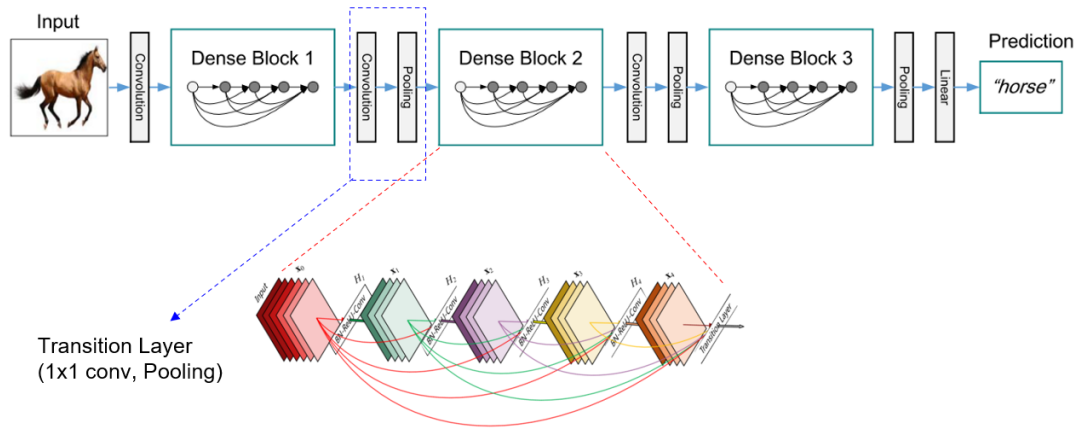
Proses *forward propagation* pada jaringan CNN dilakukan untuk melanjutkan nilai pada lapisan masukan hingga pada lapisan keluaran. Nilai ini dilanjutkan melalui lapisan konvolusi, *subsampling* dan lapisan *layer* di *fully connected* sesuai dengan deretan lapisan tersebut diletakkan pada jaringan yang digunakan.

#### 9. Proses *Back Propagation* pada CNN / *Loss Layer*

*Loss layer*, yang merupakan lapisan terakhir dalam CNN, menentukan bagaimana pelatihan memberikan sanksi karena perbedaan antara prediksi dan label. Terdapat sejumlah variasi *loss function*, termasuk *softmax loss* yang digunakan untuk memprediksi dari jumlah kelas yang saling eksklusif; *sigmoid cross-entropy loss* yang digunakan untuk memprediksi sekumpulan nilai probabilitas untuk interval  $[0,1]$ ; dan *Euclidean loss* yang digunakan untuk regresi nilai kontinu. Propagasi balik merupakan proses untuk memperbaharui nilai filter dan bobot pada jaringan. Perhitungan perubahan nilai bobot dilakukan perhitungan dimulai dari lapisan *fully connected*. Pada lapisan ini perubahan bobot dicari dengan mencari derivatif *loss function* terhadap bobot.

#### 2.2.9 *DenseNet-169*

*Densely Connected Convolutional Networks* (DenseNet) merupakan salah satu arsitektur *Convolutional Neural Network* (CNN) yang dirancang oleh Gao Huang, Zhuang Liu, Laurens van der Maaten, dan Kilian Q. Weinberger pada tahun 2016. Pada *DenseNet*, setiap *layer* memperoleh *input* tambahan dari semua *layer* sebelumnya dan meneruskan *feature-map* ke semua *layer* berikutnya. Oleh karena itu, setiap *layer* menerima “pengetahuan kolektif” dari semua *layer* sebelumnya. Karena setiap *layer* menerima *feature-map* dari semua *layer* sebelumnya, jaringan menjadi lebih tipis dan padat sehingga memiliki efisiensi komputasi dan efisiensi memori yang lebih tinggi.



Gambar 2.8. Layer-layer pada DenseNet-169. (Github.com, 2020)

DenseNet-169 terdiri dari lapisan konvolusi awal dan lapisan *pooling*. Terdiri dari empat *dense block*, sehingga ada lapisan transisi setelah setiap *dense block* dan ada lapisan klasifikasi dengan fungsi aktivasi *softmax* di akhir jaringan. Setiap *layer convolutional* terdiri dari *Batch Normalisation* (BN), *Rectified Linear Unit* (ReLU), dan *Convolution*. Setiap *dense block* terdiri dari konvolusi 1x1 yang diikuti oleh konvolusi 3x3. DenseNet-169 memiliki 6, 12, 32, dan 32 set konvolusi 1x1 dan 3x3. *Dense block* pertama, kedua, dan ketiga diikuti oleh *transition layer* yang berisi konvolusi 1x1 dan 2x2 *average pool* dengan *stride 2*. *Dense block* keempat terhubung ke *classification layer*. *Classification layer* ini menggunakan fungsi aktivasi *softmax* untuk mengklasifikasikan gambar. Oleh karena itu, ada  $6 + 12 + 32 + 32 = 82$  set *convolutional layer* 1x1 dan 3x3. Ini memberikan total  $82 \times 2 = 164$  *convolutional layer*. Ini juga berisi tiga *transition layer*, satu *convolutional layer*, dan satu *classification layer*. Oleh karena itu, kita mendapatkan total  $164 + 3 + 1 + 1 = 169$  lapisan (Dalvi et al., 2023).

| Layers               | Output Size    | DenseNet 169   |
|----------------------|----------------|--|
| Convolution          | 112×112        | 7×7 conv, stride 2   |
| Pooling              | 56×56          | 3×3 max pool, stride 2   |
| Dense Block (1)      | 56×56          | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$  |
| Transition Layer (1) | 56×56<br>28×28 | 1×1 conv<br>2×2 average pool, stride 2   |
| Dense Block (2)      | 28×28          | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$ |
| Transition Layer (2) | 28×28<br>14×14 | 1×1 conv<br>2×2 average pool, stride 2   |
| Dense Block (3)      | 14×14          | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$ |
| Transition Layer (3) | 14×14<br>7×7   | 1×1 conv<br>2×2 average pool, stride 2   |
| Dense Block (4)      | 7×7            | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$ |
| Classification Layer | 1×1<br>1000    | 7×7 global average pool<br>1000D fully-connected, softmax                                    |

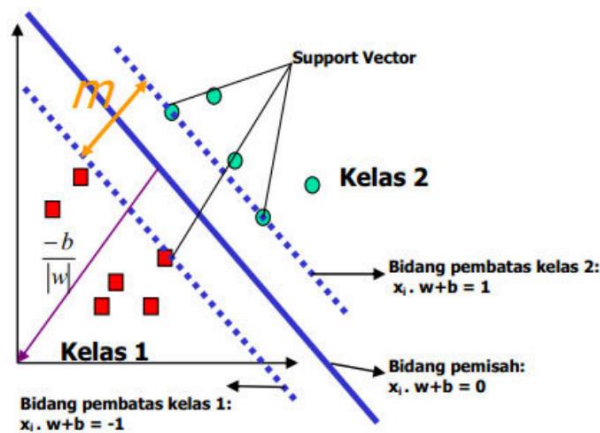
Gambar 2.9. Arsitektur *DenseNet-169*. (Dalvi et al., 2023)

#### 2.2.10 Support Vector Machine (SVM)

*Support Vector Machine* (SVM) adalah algoritma pembelajaran mesin yang umumnya digunakan untuk tugas klasifikasi pada data yang diawasi. SVM mencari *hyperplane* (bidang pemisah) optimal yang dapat memisahkan data dalam kelas yang berbeda dengan margin yang maksimal. *Hyperplane* yang memiliki margin terbesar dapat menghasilkan model SVM yang lebih umum dan memiliki kemampuan generalisasi yang lebih baik pada data yang belum pernah dilihat sebelumnya.

Secara sederhana konsep SVM adalah usaha mencari *hyperplane* “terbaik” yang berperan penting sebagai garis batas dua buah *class*. SVM mencari *hyperplane* ini berdasarkan *support vectors* dan *margin*. *Support vectors* adalah seluruh vektor data yang berjarak paling mendekati *hyperplane*, sedangkan *margin* menyatakan lebar dari *separating hyperplane*.

*Linearly separable* data merupakan data yang dapat dipisahkan secara linier. Misalkan  $\{x_1, \dots, x_n\}$  adalah dataset dan  $y_i \in \{+1, -1\}$  adalah label kelas dari data  $x_i$ , label +1 menandakan bahwa data tersebut diklasifikasikan sebagai kelas +1 dan label -1 menandakan sebaliknya. Tujuan dari SVM adalah menghasilkan sebuah model klasifikasi berupa fungsi  $\text{sign}(x)$ ,  $f(x) = y$ , agar dapat mengklasifikasikan data pada proses testing.



Gambar 2.10. Alternatif bidang pemisah terbaik. (Jumeilah, 2017)

Pada Gambar 2.10 tersebut, dua kelas dapat dipisahkan oleh sepasang bidang pembatas (*hyperplane*) yang sejajar. Bidang pembatas pertama menjadi batas kelas pertama sedangkan bidang pembatas kedua adalah batas dari kelas kedua, sehingga diperoleh Persamaan 1.

$$\begin{aligned} x_i \cdot w + b &\geq +1 \text{ for } y_i = +1 \\ x_i \cdot w + b &\leq -1 \text{ for } y_i = -1 \end{aligned} \quad (2-1)$$

Keterangan:

$w$  = Normal bidang

$b$  = Posisi bidang relatif terhadap pusat koordinat

Secara umum, cara kerja dari SVM adalah menemukan jarak terjauh dari *hyperplane* dengan kedua kelas. Proses penentuan jarak terjauh dilakukan berulang kali hingga menemukan *hyperplane* terbaik. Untuk itulah diperlukan optimasi pada SVM untuk menemukan jarak maksimum *hyperplane* dengan kedua kelas tersebut. Dalam pembangunan SVM, terdapat dua bentuk optimasi yang digunakan untuk menemukan *hyperplane*. Bentuk optimasi pertama yaitu *Primal Form SVM* dan yang kedua adalah *Dual Form SVM*. *Primal Form* tidak dapat digunakan dalam penelitian ini karena tidak akan pernah memenuhi konstrain.

Dalam klasifikasi, SVM memproyeksikan setiap sampel data ke dalam ruang fitur yang lebih tinggi, dan mencari *hyperplane* yang optimal untuk memisahkan antara kelas. Misalnya, pada tugas klasifikasi biner, SVM mencari *hyperplane* yang dapat memisahkan data ke dalam dua kelas yang berbeda secara linear, yaitu dengan garis lurus. Jika data tidak dapat dipisahkan secara linear, SVM dapat menggunakan

kernel untuk mengubah data ke dalam ruang fitur yang lebih tinggi dan kemudian mencari *hyperplane* yang optimal pada ruang fitur yang baru (Aziz, 2017).

### 2.2.11 Performance Evaluation Measure (PEM)

*Performance Evaluation Measure (PEM)* atau dalam Bahasa Indonesia bisa disebut pengukuran evaluasi performa adalah satu bundel tahapan yang digunakan untuk mengukur performa suatu sistem. PEM dalam banyak kasus digunakan dalam *training* data, tujuannya untuk mengevaluasi model yang sudah dibuat.

*Performance Evaluation Measure (PEM)* biasanya digambarkan dalam *confusion matrix*, yaitu berupa tabel yang berisi hasil pengujian model yang telah dibandingkan dengan dataset, terdiri dari kelas *true* dan *false*. (Lufiana, 2021)

Tabel 2.1. *Confusion Matrix*

|                     | <i>Predicted Class</i> |                 |
|---------------------|------------------------|-----------------|
| <i>Actual Class</i> | <i>Positive</i>        | <i>Negative</i> |
| <i>Positive</i>     | TP                     | FP              |
| <i>Negative</i>     | FN                     | TN              |

Tabel 2.2. *Confusion Matrix multiclass*

|                     |         | <i>Predicted Class</i> |         |         |
|---------------------|---------|------------------------|---------|---------|
|                     |         | Kelas 1                | Kelas 2 | Kelas 3 |
| <i>Actual Class</i> | Kelas 1 | TP                     | FN      | FN      |
|                     | Kelas 2 | FN                     | TP      | FN      |
|                     | Kelas 3 | FN                     | FN      | TP      |

Keterangan:

TP (*true positive*): contoh data bernilai positif yang diprediksi benar sebagai positif

TN (*true negative*): contoh data bernilai negatif yang diprediksi benar sebagai negatif

FP (*false positive*): contoh data bernilai negatif yang diprediksi salah sebagai positif

FN (*false negative*): contoh data bernilai positif yang diprediksi salah sebagai negatif

Ada banyak perhitungan untuk mendapatkan nilai PEM, biasanya diterapkan sebagai kombinasi atau juga secara parsial. Beberapa perhitungan dalam PEM antara lain:

#### a. *Precision*

*Precision* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif. Dari semua kelas positif yang telah di prediksi dengan benar, berapa banyak data yang benar-benar positif.



Nilai *precision* dapat menjawab pertanyaan “Berapa persen citra *thorax* yang benar normal dari keseluruhan citra *thorax* yang diprediksi normal?”

Rumus *precision* (pre):

$$pre = \frac{TP}{FP+TP} \quad (2-2)$$

b. *Accuration*

*Accuration* merupakan rasio prediksi benar (positif dan negatif) dengan keseluruhan data. Dengan kata lain, *accuracy* merupakan tingkat kedekatan nilai prediksi dengan nilai aktual (sebenarnya).

Rumus *accuration* (acc):

$$acc = \frac{TN+TP}{FN+FP+TN+TP} \quad (2-3)$$

c. *Recall*

*Recall* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. Nilai *recall* dapat menjawab pertanyaan “Berapa persen citra *thorax* yang diprediksi normal dibandingkan keseluruhan citra *thorax* yang sebenarnya normal”.

Rumus *recall* (rec):

$$rec = \frac{TP}{FN+TP} \quad (2-4)$$

d. *F1-score*

*F-1 Score* adalah ukuran perbandingan rata-rata *precision* dan *recall* yang dibobotkan.

$$f1 - score = 2 \frac{precision \times recall}{precision+recall} \quad (2-5)$$

### 2.2.12 *Streamlit*

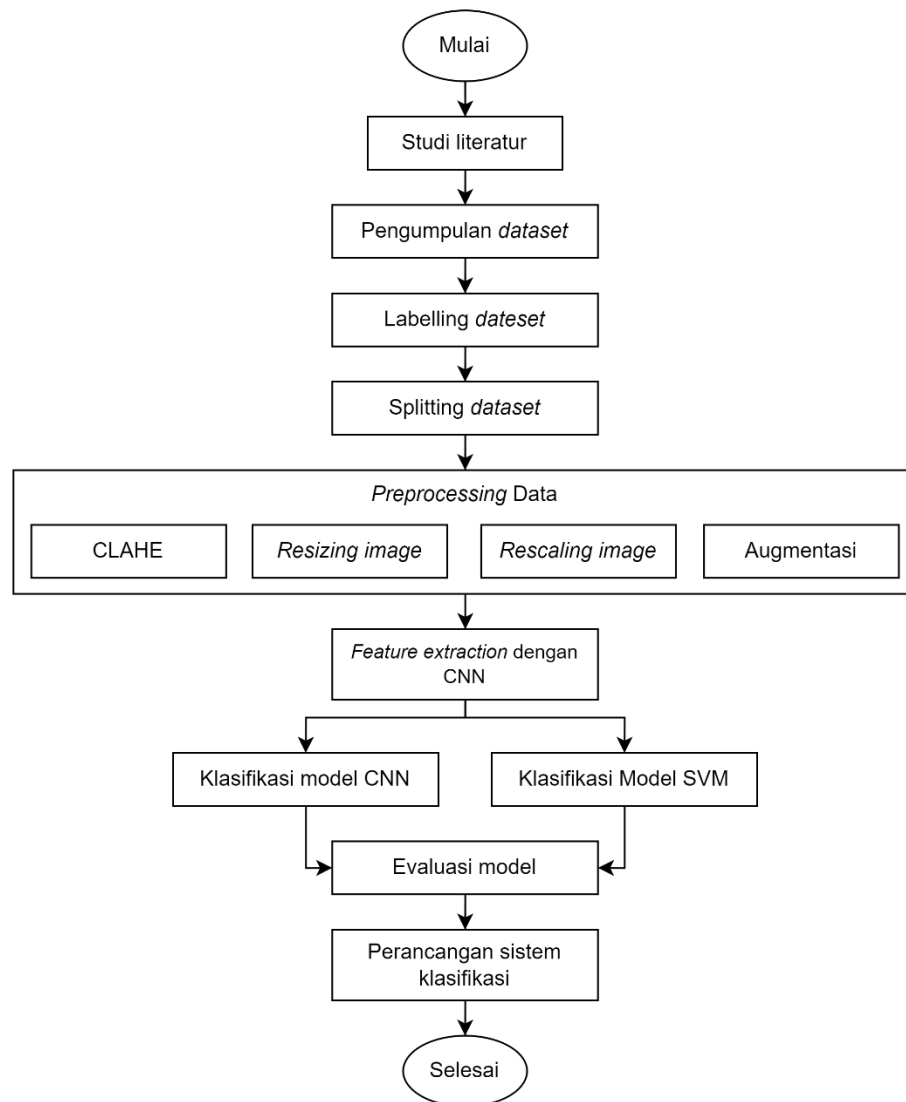
*Streamlit* adalah sebuah *framework* berbasis *Python* dan bersifat *open-source* yang dibuat untuk memudahkan dalam membangun aplikasi *web* di bidang sains data dan *machine learning* yang interaktif. *Streamlit* juga dapat didefinisikan sebagai kerangka kerja *web* yang ditujukan untuk menyebarkan model dan visualisasi dengan mudah menggunakan bahasa *Python*, yang cepat dan minimalis tetapi juga memiliki tampilan yang cukup baik serta ramah pengguna (Maulid, 2023).

## BAB III

### METODOLOGI PENELITIAN

#### 3.1 Alur Pengerjaan Tugas Akhir

Perancangan alur dari pengerjaan tugas akhir ini merupakan gambaran umum terkait alur penelitian yang akan dilakukan dalam pengerjaan tugas akhir mulai dari awal hingga akhir. Alur kerja dari pengerjaan tugas akhir ini dapat dilihat pada Gambar 3.1. berikut:



Gambar 3.1. Alur penelitian tugas akhir

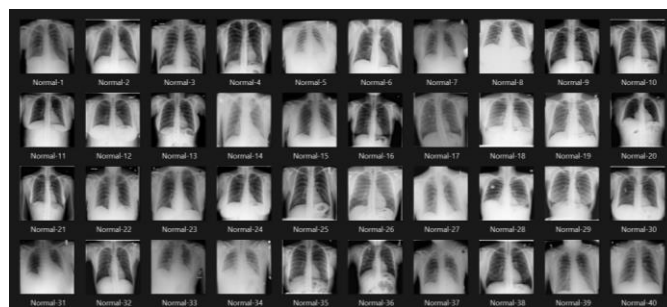
## 3.2 Uraian Metodologi

### 3.2.1 Studi Literatur

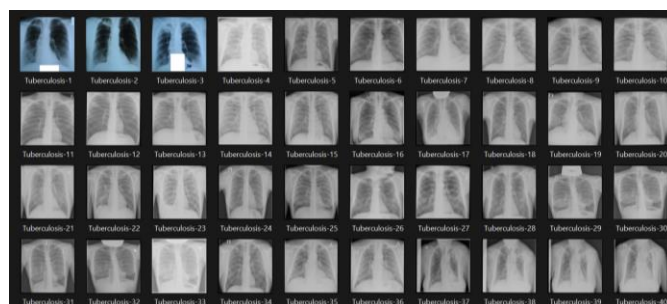
Pada penelitian ini dilakukan studi literatur sebagai tahapan awal untuk melakukan penelitian, tujuan dari studi literatur ini sendiri adalah untuk memahami konsep dan dasar teori yang akan menjadi pembanding dan pendukung untuk penelitian ini yang berjudul “Klasifikasi Penyakit Berdasarkan Dataset Citra *Thorax* Menggunakan Algoritma *Support Vector Machine* (SVM) dengan Ekstraksi Fitur *Convolutional Neural Network* (CNN)”.

### 3.2.2 Pengumpulan Data

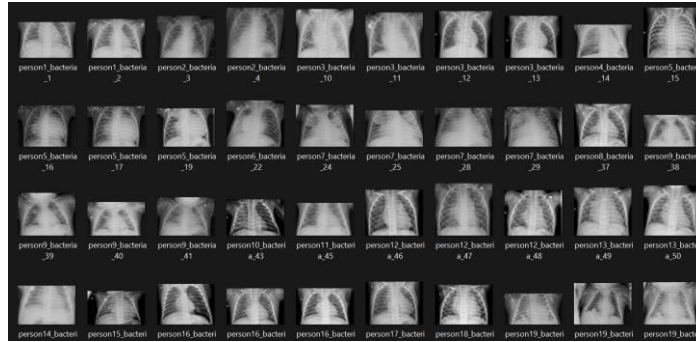
Pada penelitian ini data yang dikumpulkan terdiri atas tiga kategori yaitu citra *thorax* normal, citra *thorax* positif tuberkulosis, dan citra *thorax* positif pneumonia. Data citra yang digunakan berasal dari *kaggle* diperoleh sebanyak 7.200 citra *thorax*, dimana terdapat 4.000 citra *thorax* normal, 700 citra *thorax* positif tuberkulosis, dan 2.500 citra *thorax* positif pneumonia. Ukuran bentuk gambar pada dataset bervariasi mulai dari 512×512 piksel sampai ukuran 1328×1160 piksel. Dataset bisa diakses melalui link: (<https://www.kaggle.com/tawsifurrahman/tuberculosis-tb-chest-xray-dataset>) dan (<https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia>). Dataset yang telah diunduh terbagi menjadi tiga folder, yaitu Normal, Tuberkulosis, dan Pneumonia sebagai berikut:



Gambar 3.2. Dataset citra *thorax* normal.



Gambar 3.3. Dataset citra *thorax* positif tuberkulosis.

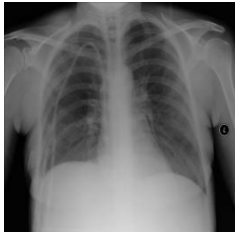




Gambar 3.4. Dataset citra *thorax* positif pneumonia.

### 3.2.3 Labelling Dataset

Pada bagian proses *labelling dataset* akan dilakukan pembagian kelas klasifikasi pada data citra *thorax* yang telah didapatkan menjadi kelas 0, 1, dan 2, dimana kelas 0 merupakan citra *thorax* normal, kelas 1 merupakan citra *thorax* positif tuberkulosis, dan kelas 2 merupakan citra *thorax* positif pneumonia. Contoh pelabelan citra rontgen dapat dilihat pada Tabel 3.1.

Tabel 3.1. Contoh proses pelabelan data

| Citra Thorax  | Label |
|---|-------|
| <br>Normal-1.png       | 0     |
| <br>Tuberkulosis-1.png | 1     |
| <br>Pneumonia-1.png    | 2     |

### 3.2.4 *Splitting Dataset*

Pada tahapan ini, dilakukan pembagian dataset dengan total data 7.200 menjadi 3 subset, yaitu *Training*, *Validation*, dan *Testing*. Pembagian *dataset* dilakukan dengan perbandingan 80% data *training*, 10% data *validation*, dan 10% data *testing*. Pembagian dataset dengan skala 80:10:10 ini memberikan keseimbangan antara melatih model dengan jumlah data yang cukup besar, memvalidasi model untuk mengoptimalkan kinerjanya, dan menguji model pada data yang belum pernah dilihat sebelumnya. Dimana semakin banyak dan bervariasi data yang dilatih maka kemungkinan model mendapatkan akurasi terbaik akan semakin meningkat.

#### 1) *Data Training*

Data *training* akan digunakan untuk melatih model atau algoritma pembelajaran mesin. Model akan belajar pola dan fitur dari citra dalam data *training* ini dan data *training* biasanya merupakan subset terbesar dalam pembagian dataset.

#### 2) *Data Validation*

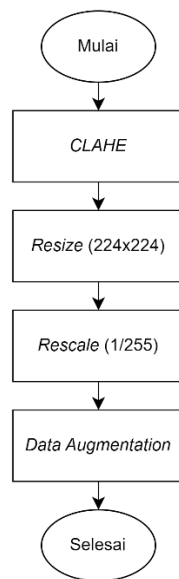
Data *validation* digunakan untuk mengukur dan memvalidasi kinerja model selama proses pelatihan. Setelah model dilatih menggunakan data *train*, data *validation* digunakan untuk memonitor kemajuan model, menyesuaikan parameter, dan mencegah *overfitting*.

#### 3) *Data Testing*

Data *testing* digunakan untuk menguji performa model atau algoritma setelah pelatihan selesai. Data *test* merupakan data yang belum pernah dilihat oleh model selama pelatihan. Pengujian pada data *test* ini memberikan indikasi kinerja model secara objektif.

### 3.2.5 *Preprocessing Data*

Tujuan dari tahap ini adalah untuk mempersiapkan data sebelum melakukan *training* pada model untuk mengklasifikasikan gambar *thorax* normal, positif tuberkulosis, atau positif pneumonia. Adapun alur dari tahapan *preprocessing* yang dilakukan pada Gambar 3.5 berikut:



Gambar 3.5. *Flowchart* alur *preprocessing*.

a) *Contrast Limited Adaptive Histogram Equalization (CLAHE)*

Tahap ini merupakan proses untuk meningkatkan kontras pada gambar dengan memperkuat perbedaan intensitas piksel yang terdapat pada citra.

b) *Resizing Image*

Mengubah dimensi gambar dengan mengubah panjang dan lebar gambar. Hal ini dapat dilakukan dengan cara menambah atau mengurangi piksel gambar. Pada tahap *resizing* ini, gambar akan di *resize* dengan ukuran  $224 \times 224$  piksel. Hal ini dilakukan karena pada data citra dari dataset citra *thorax* memiliki beragam orientasi, sudut pandang, dan ukuran yang terlalu besar atau kecil sehingga diperlukan praproses pada *dataset* yang ada dengan tujuan agar data memiliki ukuran yang sama sehingga mempercepat proses pelatihan.

c) *Rescaling Image*

Semua data citra digital direpresentasikan dalam bentuk matriks. Setiap elemen piksel pada matriks memiliki nilai yang merepresentasikan warna. Nilainya berkisar antara 0-255. Setelah itu, citra masuk ke proses *rescale*, di mana nilai piksel yang awalnya berskala 0-255 diubah menjadi skala antara 0-1. Caranya yaitu dengan membagi setiap nilai piksel dengan angka 255. Proses ini disebut juga dengan normalisasi.

#### d) Augmentasi Data

Pada tahap ini, augmentasi dilakukan untuk meningkatkan jumlah data dengan memodifikasi data sehingga program mengenali data tersebut adalah data yang berbeda. Proses ini dapat membantu penelitian dengan data berjumlah sedikit untuk mencegah *overfitting* maupun *underfitting*. Selain itu, proses augmentasi dapat meningkatkan kinerja model dalam proses klasifikasi gambar. Pada tahap ini dilakukan tiga proses transformasi data citra sebagai berikut:

##### 1. *Horizontal Flip*

*Horizontal Flip* merupakan proses transformasi data citra dengan membalik ke arah *horizontal* data citra secara acak.

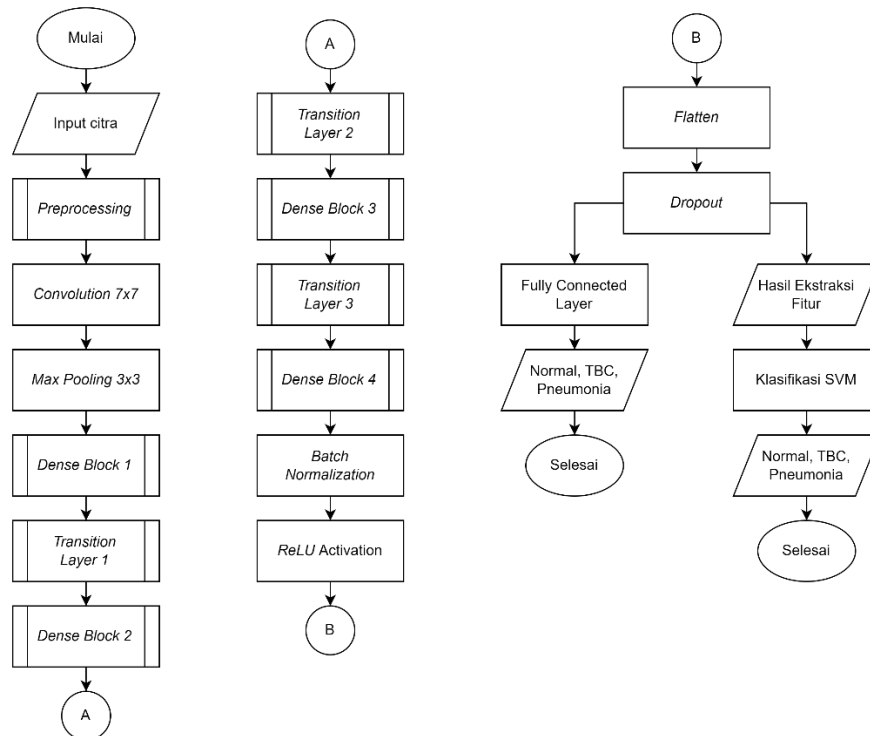
##### 2. *Rotation Range*

*Rotation range* merupakan proses pemutaran data citra secara acak. Pada penelitian ini *range* perputaran citra adalah  $20^\circ$ .

##### 3. *Zoom Range*

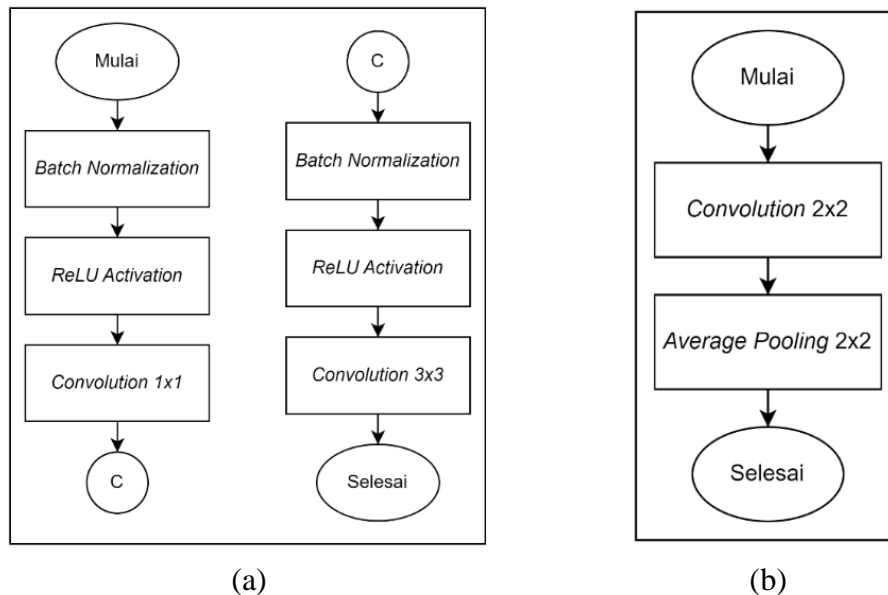
*Zoom range* merupakan transformasi gambar citra dengan memperbesar bentuk ukuran gambar secara acak. Besar *zoom range* yang dilakukan pada penelitian ini adalah 10% dari data citra asli.

### 3.2.6 Arsitektur Model



Gambar 3.6. Arsitektur model penelitian.

Pada Gambar 3.6, sistem dimulai dengan *input* citra kemudian melakukan *preprocessing*, setelah melalui tahap *preprocessing*, dilakukannya operasi *convolution 7x7* dengan *strides 2*, operasi dilanjutkan dengan *max pooling 3x3* dan akan mendapatkan suatu nilai matriks yang diproses ke operasi *dense block 1*, *transition layer 1*, *dense block 2*, *transition layer 2*, *dense block 3*, *transition layer 3*, dan *dense block 4*. Setelah itu dilakukan *batch normalization*, aktivasi *relu*, *flatten*, dan *dropout*. Kemudian terdapat 2 tahap, yang pertama *output* terakhir yang telah diproses akan masuk ke dalam *fully connected layer* dan melakukan klasifikasi dengan model CNN. Tahap kedua *output* terakhir yang telah diproses sebelumnya yaitu berupa ekstraksi fitur diproses kembali pada tahapan klasifikasi menggunakan algoritma SVM.



Gambar 3.7. (a) *Dense block* dan (b) *Transition layer*.

Lapisan yang berada pada *dense block* berisikan *batch normalization*, *ReLU activation* dan *convolution* dengan filter 3×3. Lapisan antara 2 blok yang berdekatan disebut sebagai lapisan transisi dan perubahan ukuran fitur melalui *convolution* dan *pooling*. Selanjutnya *output* dari *layer* terakhir yang merupakan hasil dari ekstraksi fitur model CNN akan diklasifikasikan menggunakan algoritma SVM.



**a) Convolutional Neural Network (CNN)**

Pada tahap ini, ekstraksi fitur bertujuan untuk mengambil ciri-ciri penting dari gambar, seperti tepi, bentuk, tekstur, dan warna, menggunakan algoritma *Convolutional Neural Network (CNN)* dengan arsitektur *DenseNet-169*.

Tabel 3.2. Arsitektur model CNN.

| No. | Layer              | Output Shape   | Strides | Params   |
|-----|--------------------|----------------|---------|----------|
| 1.  | Input (image)      | (224, 224, 3)  | -       | -        |
| 2.  | Conv2D             | (112, 112, 64) | (2, 2)  | 9408     |
| 3.  | MaxPooling2D       | (56, 56, 64)   | (2, 2)  | -        |
| 4.  | Dense Block 1      | (56, 56, 256)  | -       | -        |
| 5.  | Transition Layer 1 | (28, 28, 128)  | (2, 2)  | 33024    |
| 6.  | Dense Block 2      | (28, 28, 512)  | -       | -        |
| 7.  | Transition Layer 2 | (14, 14, 256)  | (2, 2)  | 132096   |
| 8.  | Dense Block 3      | (14, 14, 1024) | -       | -        |
| 9.  | Transition Layer 3 | (7, 7, 512)    | (2, 2)  | 525312   |
| 10. | Dense Block 4      | (7, 7, 1664)   | -       | -        |
| 11. | Batch Norm         | (7, 7, 1664)   | -       | 6656     |
| 12. | ReLU               | (7, 7, 1664)   | -       | -        |
| 13. | Flatten            | (81536,)       | -       | -        |
| 14. | Dropout            | (81536,)       | -       | -        |
| 15. | Dense              | (256,)         | -       | 20873472 |

Pada Tabel 3.2, pertama, melalui lapisan *Conv2D* awal, citra input dengan resolusi 224×224 piksel dan 3 saluran warna (RGB) direduksi, menghasilkan 64 saluran dengan parameter 9408. Langkah selanjutnya adalah *MaxPooling2D* yang memangkas resolusi setengahnya. Selanjutnya, melalui empat *Dense block*, jaringan memperdalam representasi gambar dengan masing-masing blok menghasilkan keluaran dengan dimensi yang semakin kompleks. *Transition layer* diantara blok-blok ini membantu dalam mengurangi jumlah saluran dan resolusi gambar, memungkinkan jaringan untuk mempertahankan kompleksitas fitur yang tinggi. Terakhir, tahap *Flatten* dan *Dropout*, data diolah dan disiapkan untuk lapisan *Dense* terakhir, yang menghasilkan output akhir dengan dimensi 256.

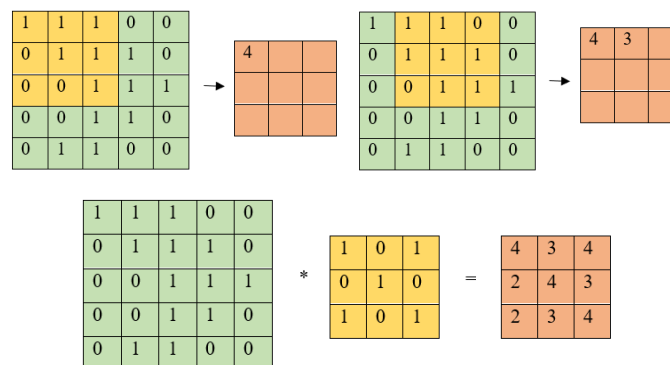
1. *Convolutional Layer*. Proses pada *convolutional layer* akan melakukan operasi konvolusi antara *feature map* masukan dengan matriks kernel. Matriks kernel ini akan digeser ke seluruh bagian *feature map* masukan sehingga akan menghasilkan keluaran berupa *feature map* baru. Sebagai contoh pada Gambar 3.10, *feature map* masukan akan dikonvolusi dengan kernel  $3 \times 3$  dan *stride* 1 langkah. Bagian dari *feature map* masukan akan diambil sebagian matriks dengan ukuran sama dengan ukuran kernel. Pada tahap pertama bagian matriks yang diambil terdiri dari 1, 1, 1, 0, 1, 1, 0, 0, dan 1 sedangkan matriks kernel terdiri dari 1, 0, 1, 0, 1, 0, 1, 0, dan 1. Operasi konvolusi dari kedua matriks tersebut dapat dilihat sebagai berikut:

$$y = \sum x[m, n].k[m, n]$$

$$y = 1x1 + 1x0 + 1x1 + 0x0 + 1x1 + 1x0 + 0x1 + 0x0 + 1x1$$

$$y = 1 + 0 + 1 + 0 + 1 + 0 + 0 + 0 + 1$$

$$y = 4$$



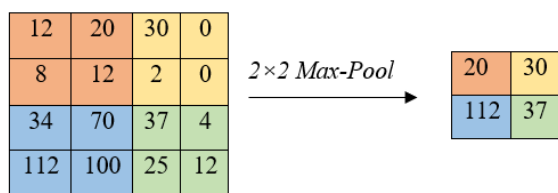
Gambar 3.8. Proses konvolusi *layer*.

Setelah dilakukan proses konvolusi, *feature map* yang dihasilkan akan diaktivasi menggunakan fungsi aktivasi *Rectified Linear Unit* (ReLU). Setiap piksel pada *feature map* akan dimasukkan ke dalam fungsi ReLU, dimana setiap nilai negatif akan diubah nilainya menjadi 0 dengan persamaan sebagai berikut:

$$\sigma(x) = \max(0, x)$$

2. *Pooling layer*. Digunakan untuk mengurangi ukuran spasial dari *feature map*. Jenis *pooling* yang digunakan adalah *max pooling*, dimana nilai maksimum pada suatu *layer* tertentu dipilih. Prosesnya mirip dengan *convolution layer* yaitu menggeser *pooling layer* ke seluruh permukaan *feature map*, tetapi disini

*pooling layer* digunakan sebagai acuan untuk memilih nilai maksimum pada suatu area tertentu dalam rentang jendela *pooling layer*. Proses ini menghasilkan sebuah matriks *feature map* yang berisi nilai-nilai maksimum yang dipilih. Sebagai contoh pada Gambar 3.11, menggunakan *max pooling* dengan ukuran *pooling*  $2 \times 2$  dan *stride* 2 langkah. Jendela *pooling layer* pada *feature map* dibedakan dengan blok warna, dimana tiap bagian blok warna akan diambil nilai tertinggi. Pada blok warna merah muda terdapat nilai matriks 12, 20, 8, dan 12. Keempat nilai tersebut akan dijadikan sebagai acuan untuk dipilih dan dijadikan nilai matriks pada *feature map* hasil *pooling layer*. Nilai yang diambil adalah nilai tertinggi dari keempat nilai tersebut yaitu angka 20. Nilai 20 inilah yang selanjutnya akan menjadi salah satu bagian dari *feature map* baru. Kemudian dilanjutkan untuk blok warna yang lainnya sampai semua bagian pada *feature map* masukan habis diproses.



Gambar 3.9. Proses *pooling layer*.

3. *Flatten Layer*. *Flatten* merupakan tahapan mengubah matriks kebentuk *vector/matrix* 1 dimensi. Dari *output dropout layer* sebesar  $2 \times 2 \times 32$  akan didapatkan bentuk *vector* baru, gambar ilustrasi dapat dilihat pada gambar 3.8 dibawah ini.

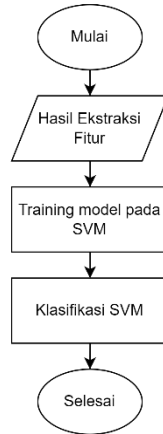


Gambar 3.10. Ilustrasi proses *flatten*.

4. *Dropout Layer*

*Dropout* berfungsi untuk mengurangi kompleksitas model yang telah dibangun dengan membuang *neural network* yang tidak terpakai. Ukuran model akan tetap  $2 \times 2 \times 32$ , tidak ada perubahan.

### b) *Support Vector Machine (SVM)*



Gambar 3.11. Alur klasifikasi model SVM.

Setelah melewati tahap ekstraksi fitur dengan algoritma CNN, selanjutnya hasil dari ekstraksi fitur tersebut digunakan untuk melatih model SVM. Selama pelatihan, SVM akan mempelajari pengetahuan yang terkandung dalam data *train* untuk membedakan citra *thorax* normal, positif tuberkulosis, dan positif pneumonia.

#### 3.2.7 Evaluasi Model

Evaluasi model dilakukan untuk mengetahui kinerja model. Evaluasi model dilakukan dengan cara melihat tingkat akurasi metode melalui *confusion matrix* dan tabel akurasi serta presisi untuk tiap model. Setelah data *test* diujikan terhadap data *training*, maka akan menghasilkan daftar kelas-kelas dari data *test* atau prediksi kelas. Kemudian prediksi kelas dibandingkan dengan kelas yang sebenarnya dari data *test* yang disembunyikan sebelumnya. Sehingga dapat dilihat dan dihitung nilai *accuracy*, *precision*, *recall*, dan *f1-score* dengan perhitungan dilakukan menggunakan persamaan-persamaan sebagai berikut:

$$accuracy = \frac{TP}{Total\ jumlah\ data} \quad (3.1)$$

$$precision = \frac{TP}{FP+TP} \quad (3.2)$$

$$recall = \frac{TP}{FN+TP} \quad (3.3)$$

$$F1 - score = 2 \frac{precision \times recall}{precision+recall} \quad (3.4)$$

$$Average\ Macro = \frac{precision\ kelas\ 1 + precision\ kelas\ 2 + precision\ kelas\ 3}{Jumlah\ kelas} \quad (3.5)$$

Tabel 3.3. *Confusion matrix multiclass.*

|                     |                  | <i>Predicted Class</i> |            |                  |
|---------------------|------------------|------------------------|------------|------------------|
|                     |                  | <b>Normal</b>          | <b>TBC</b> | <b>Pneumonia</b> |
| <i>Actual Class</i> | <b>Normal</b>    | TP                     | FN         | FN               |
|                     | <b>TBC</b>       | FN                     | TP         | FN               |
|                     | <b>Pneumonia</b> | FN                     | FN         | TP               |

Berdasarkan Tabel 3.3, *confusion matrix* terdiri dari tiga kelas, yaitu Normal, TBC (Tuberkulosis), dan Pneumonia. Adapun penjelasannya sebagai berikut:

1. TP (*True Positive*)

Merupakan jumlah data yang diklasifikasikan dengan benar sebagai kelas yang sebenarnya. Dalam kasus ini, ada satu data yang diklasifikasikan dengan benar sebagai Normal, satu data diklasifikasikan dengan benar sebagai TBC, dan satu data diklasifikasikan dengan benar sebagai Pneumonia.

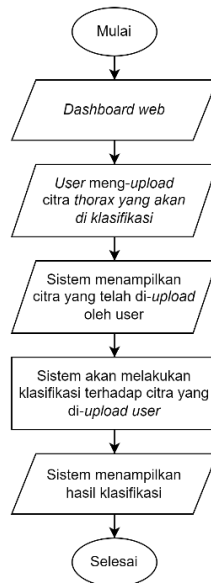
2. FN (*False Negative*)

Merupakan jumlah data yang seharusnya diklasifikasikan sebagai suatu kelas, tetapi salah diklasifikasikan sebagai kelas lain. Dalam kasus ini, ada dua data yang seharusnya diklasifikasikan sebagai Normal, tetapi salah diklasifikasikan sebagai TBC dan Pneumonia. Selain itu, ada dua data yang seharusnya diklasifikasikan sebagai TBC, tetapi salah diklasifikasikan sebagai Normal dan Pneumonia. Terakhir, ada dua data yang seharusnya diklasifikasikan sebagai Pneumonia, tetapi salah diklasifikasikan sebagai Normal dan TBC.

### 3.2.8 Perancangan Sistem Klasifikasi Citra *Thorax*

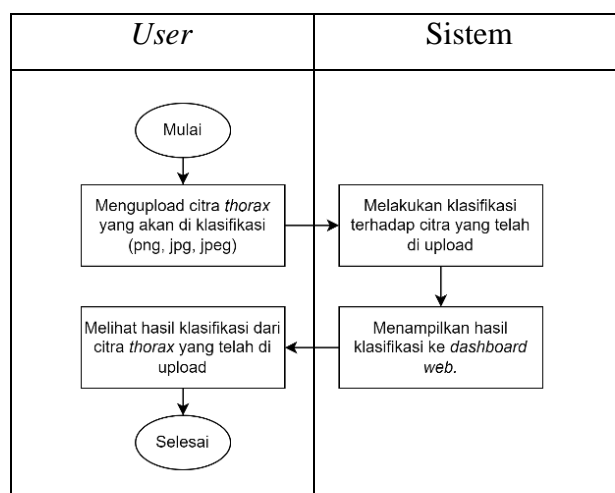
Dalam membangun sistem aplikasi yang dapat digunakan untuk klasifikasi citra dengan mengimplementasikan metode CNN sebagai ekstraksi fitur dan SVM sebagai algoritma klasifikasi, dibuatlah sebuah diagram alir untuk membantu dalam proses memahami alur dari proses sistem berlangsung. Diagram alir tersebut menggambarkan tahapan-tahapan dalam menjalankan sistem, dimulai dari tahap *user* mengakses *dashboard web*, lalu mengupload citra *thorax* yang akan di klasifikasi, setelah itu sistem akan menampilkan citra *thorax* yang telah di *upload* beserta citra setelah melewati tahap *preprocessing*, kemudian sistem menampilkan hasil klasifikasi pada citra *thorax* tersebut apakah diklasifikasi sebagai normal, tbc, atau pneumonia

beserta nilai *confidence* atau seberapa yakin model terhadap klasifikasi yang dihasilkan.



Gambar 3.12. Diagram alir sistem klasifikasi.

Pada perancangan sistem aplikasi klasifikasi dalam penelitian ini, akan menerapkan implementasi dari model klasifikasi CNN sebagai ekstraksi fitur dan SVM sebagai algoritma klasifikasi yang telah dibuat sebelumnya, sehingga sistem aplikasi klasifikasi ini dapat memudahkan *user* dalam melakukan klasifikasi penyakit tuberkulosis dan pneumonia secara lebih efektif. Berikut adalah *activity* diagram dari sistem aplikasi klasifikasi:

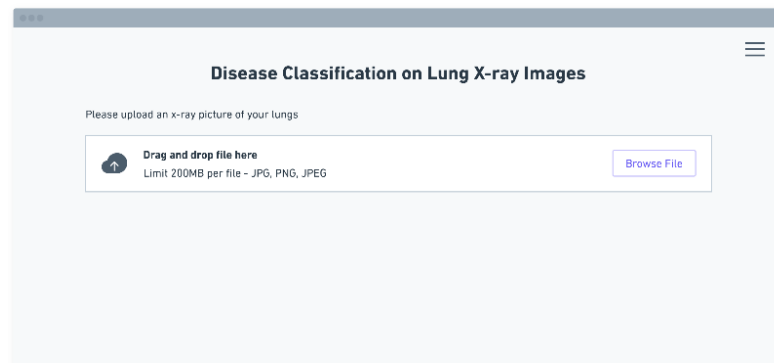


Gambar 3.13. *Activity diagram* user pada halaman aplikasi

Pada Gambar 3.13. merupakan *use case* diagram pada halaman aplikasi, yang dimana halaman ini nantinya akan digunakan *user* untuk dapat melakukan klasifikasi terhadap citra *thorax*.

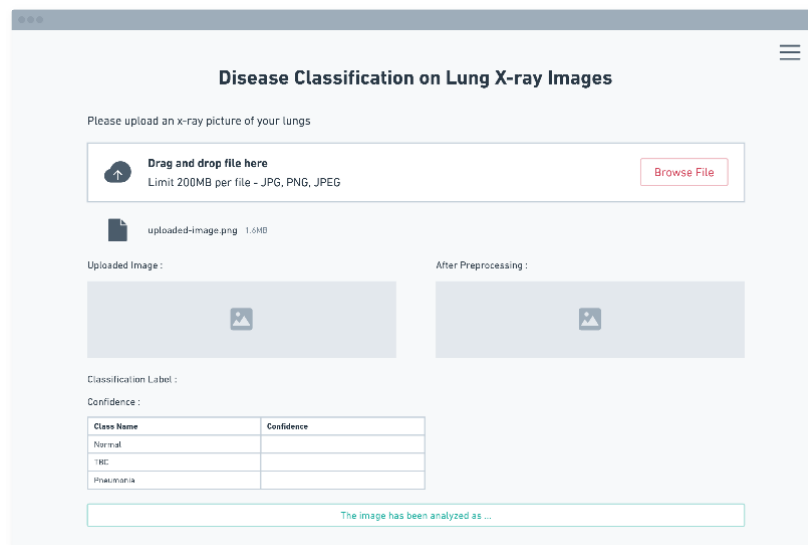
### 3.2.9 Desain Aplikasi

Pada tahap desain aplikasi ini tujuannya adalah merencanakan dan merancang aplikasi dengan tujuan menciptakan pengalaman *user* yang baik dalam melakukan klasifikasi terhadap citra *thorax*. Pada penelitian ini, penulis menggunakan *framework streamlit* dalam pembuatan aplikasi web dan desain aplikasi untuk klasifikasi tuberkulosis dapat dilihat pada Gambar 3.14 sebagai berikut:



Gambar 3.14. Desain aplikasi klasifikasi.

Pada Gambar 3.14 diatas, merupakan desain dalam pembuatan aplikasi klasifikasi, dimana *user* bisa mengakses halaman tersebut lalu meng-*upload* citra *thorax* yang akan di klasifikasi, setelah mengupload citra *thorax*, maka tampilan aplikasi akan seperti Gambar 3.15 berikut:



Gambar 3.15. Desain aplikasi setelah *upload* citra.

Pada Gambar 3.14 diatas sebagai contoh, *user* telah mengupload citra *thorax* dengan file yang bernama 'uploaded-image.png'. Setelah ter-*upload*, citra tersebut secara otomatis akan melewati tahap klasifikasi berdasarkan model yang telah dibuat. Setelah melewati tahap klasifikasi, akan ditampilkan gambar citra yang telah di-*upload* dan gambar setelah melewati *preprocessing*, selanjutnya ditampilkan nilai *confidence* dari model beserta keterangan apakah citra tersebut normal, positif tuberkulosis, atau positif pneumonia.

### 3.3 Analisis Kebutuhan

#### 1. Perangkat Keras (*Hardware*)

Laptop Asus tipe A416 dengan spesifikasi *processor* Intel Core i3-1005G1, RAM 12 GB, dan *Hardisk* penyimpanan SSD 250 GB.

#### 2. Perangkat Lunak (*Software*)

Pada penelitian ini, *software* dan *library python* yang digunakan dalam melakukan proses pengujian sebagai berikut:

##### a. *Software*

- Aplikasi *Draw.io* untuk membuat diagram.
- Aplikasi *Whimsical* untuk membuat desain *user interface*.
- *Google Collab* sebagai kode editor dalam melakukan perancangan model dan *training* model.
- *Visual Studio Code* sebagai kode editor dalam perancangan sistem aplikasi *web* menggunakan *streamlit*.
- *Python* versi 3.10.6 sebagai bahasa pemrograman dalam *machine learning*.

##### b. *Library Python*

- *Tensorflow* untuk membangun, melatih dan mengevaluasi model jaringan saraf tiruan.
- *Pandas* untuk memuat, menyimpan, mengubah dan menggabungkan data.
- *OpenCV* untuk memproses citra sebelum memasukkan ke dalam model.
- *Numpy* untuk melakukan operasi matriks, vektor dan aljabar.
- *Matplotlib* membuat visualisasi data dalam bentuk *plot*.
- *Scikit-learn* untuk melakukan pembagian *dataset*, membuat *confusion matrix*
- *Pickle* untuk menyimpan dan membaca model dalam bentuk file.
- *Streamlit* untuk pengembangan aplikasi *website machine learning*.



## BAB IV

### HASIL DAN PEMBAHASAN

#### 4.1 Hasil Pembuatan Model

##### 4.1.1 Hasil Pengumpulan Data

Data diperoleh dari *Kaggle* dan berasal dari beberapa institut seperti, *National Library of Medicine* (NLM), *National Institute of Allergy & Infectious Diseases*, *National Institute of Allergy and Infectious Diseases Republic of Belarus*, *Radiological Society of North America* (RSNA) dan *Guangzhou Women and Children's Medical Center*. Data yang dikumpulkan berjumlah total 7.200 data citra, yaitu 4.000 citra *thorax* normal, 700 citra *thorax* positif tuberkulosis, dan 2.500 citra *thorax* positif pneumonia.

Tabel 4.1. Jumlah dataset citra *thorax*.

| Kelas citra <i>thorax</i> | Jumlah citra |
|---------------------------|--------------|
| Normal                    | 4.000        |
| Tuberkulosis              | 700          |
| Pneumonia                 | 2.500        |
| <b>Total</b>              | <b>7.200</b> |

##### 4.1.2 Labelling Dataset

Pada proses ini, akan dilakukan pembagian kelas klasifikasi pada data citra *thorax* yang telah didapatkan menjadi kelas 0, 1, dan 2, dimana kelas 0 merupakan citra *thorax* normal, kelas 1 merupakan citra *thorax* positif tuberkulosis, dan kelas 2 merupakan citra *thorax* positif pneumonia.

```
import os
import cv2

data = []
labels = []

# Labelling citra thorax normal menjadi '0'
normal = os.listdir("/content/drive/MyDrive/Dataset Tugas Akhir/Thorax_dataset/Normal/")
for i in normal:
    image = cv2.imread("/content/drive/MyDrive/Dataset Tugas Akhir/Thorax_dataset/Normal/" + i)
    resized_image = cv2.resize(image, (224, 224))
    data.append(resized_image)
    labels.append(0)
```

```

# Labelling citra thorax tuberkulosis menjadi '1'
tbc      =      os.listdir("/content/drive/MyDrive/Dataset      Tugas
Akhir/Thorax_dataset/Tuberculosis/")
for i in tbc:
    image  =  cv2.imread("/content/drive/MyDrive/Dataset      Tugas
Akhir/Thorax_dataset/Tuberculosis/" + i)
    resized_image = cv2.resize(image, (224, 224))
    data.append(resized_image)
    labels.append(1)

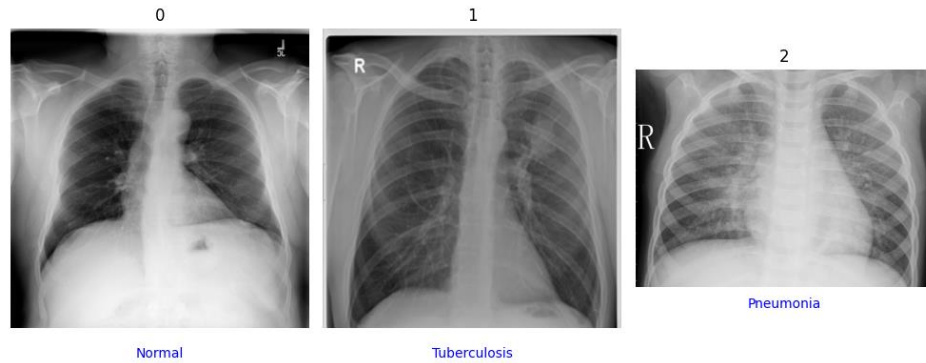
# Labelling citra thorax pneumonia menjadi '2'
pneumonia  =  os.listdir("/content/drive/MyDrive/Dataset      Tugas
Akhir/Thorax_dataset/Pneumonia/")
for i in pneumonia:
    image  =  cv2.imread("/content/drive/MyDrive/Dataset      Tugas
Akhir/Thorax_dataset/Pneumonia/" + i)
    resized_image = cv2.resize(image, (224, 224))
    data.append(resized_image)
    labels.append(2)

data = np.array(data)
labels = np.array(labels)

```

*Script 4.1. Labelling dataset.*

Kode pada *Script 4.1* diatas merupakan implementasi untuk melakukan *labelling dataset* citra *thorax*. Pertama, fungsi `data = []` dan `labels = []` diinisialisasi untuk menyimpan data citra dan label. Untuk setiap kategori, citra-citra diambil dari direktori yang sesuai, kemudian dibaca menggunakan *library OpenCV* (`cv2`) dan di-*resize* menjadi ukuran 224×224 piksel agar memiliki ukuran citra yang konsisten serta mengurangi kompleksitas data citra. Setelah *resizing*, citra-citra ini dimasukkan ke dalam variabel `data`, sementara label yang sesuai untuk setiap citra ditambahkan ke variabel `labels`. Setelah proses pemrosesan selesai, variabel `data` dan `labels` diubah menjadi *array Numpy* menggunakan pustaka *NumPy* (`np.array`), yang merupakan format yang umum digunakan untuk manipulasi data numerik dalam konteks pemrosesan data menggunakan *Python*. Dengan kedua *array* ini, data citra yang telah di-*resize* dan label yang sesuai siap untuk digunakan ke tahap selanjutnya. berikut merupakan gambaran hasil pada proses *labelling dataset* seperti pada gambar 4.1.



Gambar 4.1. Hasil *labelling dataset*.

#### 4.1.3 *Splitting Dataset*

Pada tahapan ini, dilakukan pembagian dataset dengan total data 7.200 menjadi 3 subset, yaitu *Training*, *Validation*, dan *Testing*. Pembagian dilakukan dengan perbandingan 80% untuk data *training*, 10% untuk data *validation*, dan 10% untuk data *testing*. Berikut merupakan hasil dari *splitting dataset*.

Tabel 4.2. Hasil *Splitting Dataset*

| <b>Citra Thorax</b> | <b><i>Train</i></b> | <b><i>Validation</i></b> | <b><i>Test</i></b> |
|---------------------|---------------------|--------------------------|--------------------|
| Normal              | 3.200               | 400                      | 400                |
| Tuberkulosis        | 560                 | 70                       | 70                 |
| Pneumonia           | 2.000               | 250                      | 250                |
| <b>Total</b>        | <b>5.760</b>        | <b>720</b>               | <b>720</b>         |

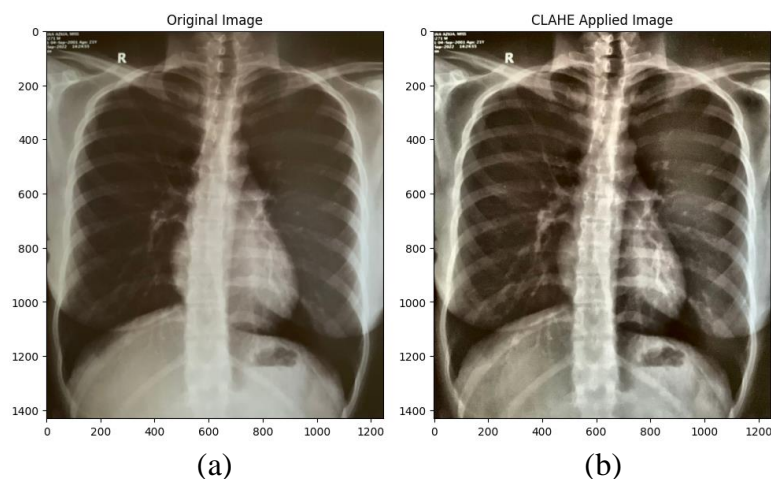
#### 4.1.4 *Preprocessing*

Tahap ini untuk mempersiapkan data sebelum melakukan *training* pada model untuk mengklasifikasikan gambar *thorax* normal, positif tuberkulosis, atau positif pneumonia. Berikut merupakan *source code* tahapan *preprocessing* yang dilakukan:

```
# fungsi CLAHE
def apply_clahe(image):
    lab = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)
    l, a, b = cv2.split(lab)
    clahe = cv2.createCLAHE(clipLimit = 2.0, tileGridSize = (8,
8))
    l = clahe.apply(l)
    lab = cv2.merge((l, a, b))
    clahe_image = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
    return clahe_image
```

Script 4.2. Fungsi CLAHE.

Kode pada *Script 4.2* diatas merupakan tahap mendefinisikan fungsi `apply_clahe` yang digunakan untuk menerapkan teknik CLAHE (*Contrast Limited Adaptive Histogram Equalization*) pada gambar. Pertama, gambar diubah dari ruang warna BGR (*blue, green, red*) ke LAB (*lightness, saluran warna A, saluran warna B*) menggunakan fungsi `cv2.cvtColor`. Kemudian, saluran kecerahan (L), saluran warna A, dan saluran warna B dipisahkan menggunakan fungsi `cv2.split`. Selanjutnya, objek CLAHE dibuat dengan parameter `clipLimit` sebesar 2.0 dan `tileGridSize` sebesar (8, 8) menggunakan fungsi `cv2.createCLAHE`. Teknik CLAHE kemudian diterapkan pada saluran kecerahan (L) menggunakan metode `apply` dari objek CLAHE. Setelah itu, saluran kecerahan (L), saluran warna A, dan saluran warna B digabungkan kembali menggunakan fungsi `cv2.merge`. Terakhir, gambar diubah kembali ke ruang warna BGR menggunakan fungsi `cv2.cvtColor` dengan parameter `cv2.COLOR_LAB2BGR`. Adapun hasil dari proses CLAHE seperti pada Gambar 4.2 berikut.



Gambar 4.2. Perbandingan (a) Citra asli dan (b) Hasil proses *CLAHE*.

```
# Preprocess data
train_data = []
train_labels = []
val_data = []
val_labels = []
test_data = []
test_labels = []

train_normal = os.listdir(train_folder + "/Normal/")
for i in train_normal:
    image = cv2.imread(train_folder + "/Normal/" + i)
    clahe_image = apply_clahe(image)
    resized_image = cv2.resize(clahe_image, (224, 224))
    train_data.append(resized_image)
    train_labels.append(0)
```

```

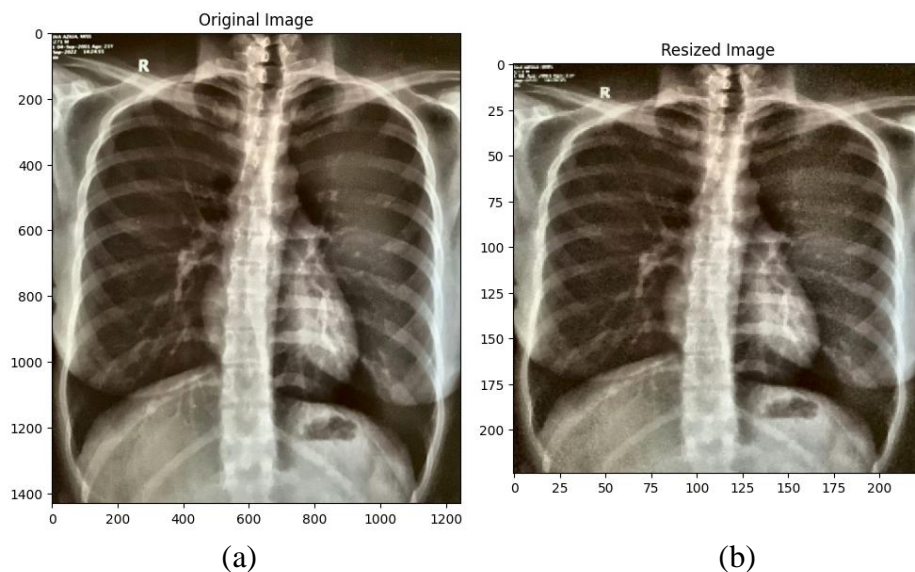
val_normal = os.listdir(val_folder + "/Normal/")
for i in val_normal:
    image = cv2.imread(val_folder + "/Normal/" + i)
    clahe_image = apply_clahe(image)
    resized_image = cv2.resize(clahe_image, (224, 224))
    val_data.append(resized_image)
    val_labels.append(0)

test_normal = os.listdir(test_folder + "/Normal/")
for i in test_normal:
    image = cv2.imread(test_folder + "/Normal/" + i)
    clahe_image = apply_clahe(image)
    resized_image = cv2.resize(clahe_image, (224, 224))
    test_data.append(resized_image)
    test_labels.append(0)

```

*Script 4.3. Resizing dataset.*

Kode pada *Script 3* diatas merupakan tahap *preprocessing* data *train*, *val*, dan *test* pada kelas normal, begitu juga dengan kelas yang lain, Langkah awal menginisialisasi variabel untuk menyimpan data hasil *preprocessing*. Selanjutnya menggunakan *cv2* untuk membaca gambar dan menyimpannya di dalam variabel *image*, lalu melakukan teknik CLAHE pada gambar dengan memanggil fungsi *apply\_clahe(image)* yang bertujuan untuk meningkatkan kontras gambar. Setelah menerapkan CLAHE, ukuran gambar diubah menjadi 224×224 piksel menggunakan fungsi *cv2.resize()* dan disimpan dalam variabel *resized\_image*. Berikut merupakan data citra asli dengan data citra hasil proses *resizing* dapat dilihat pada Gambar 4.3.



Gambar 4.3. Perbandingan (a) Citra asli dan (b) Citra hasil proses *resize*.

```
# Rescale Data
train_data = np.array(train_data) / 255
val_data = np.array(val_data) / 255
test_data = np.array(test_data) / 255
```

*Script 4.4. Rescaling dataset.*

Kode pada *Script 4* diatas merupakan tahap *rescaling*. Data pelatihan, validasi, dan pengujian dinormalisasi dengan membagi setiap elemen dengan 255. Normalisasi ini penting karena mengubah rentang nilai piksel dari 0 hingga 255 menjadi 0 hingga 1, memudahkan proses pelatihan model dan dapat meningkatkan kinerjanya. Dengan demikian, memastikan bahwa data-data yang digunakan memiliki skala yang seragam dan terkondisi dapat mempercepat proses konvergensi dan mengoptimalkan hasil dari model pembelajaran mesin atau pengolahan citra yang akan dikembangkan.

```
# Augmentasi Data
from tensorflow.keras.preprocessing.image
import ImageDataGenerator

data_augmentation = ImageDataGenerator(
    rotation_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True
)
```

*Script 4.5. Augmentasi data.*

Kode pada *Script 4.5* diatas merupakan implementasi dari teknik augmentasi data pada model menggunakan menggunakan `ImageDataGenerator` dengan beberapa parameter, seperti `rotation_range`, `zoom_range`, dan `horizontal_flip`. `rotation_range` menentukan rentang rotasi gambar dalam derajat, `zoom_range` mengontrol tingkat zoom, dan `horizontal_flip` digunakan untuk membalikkan gambar secara horizontal pada gambar. Tujuan dari augmentasi data ini yaitu memperbanyak variasi data latih, membantu model untuk memahami pola-pola yang lebih umum, dan meningkatkan kinerja keseluruhan dari model yang dilatih.

#### **4.1.5 Convolutional Neural Network (CNN)**

Pada proses ekstraksi fitur ini dilakukan menggunakan algoritma *Convolutional Neural Network* (CNN) dengan menggunakan arsitektur *DenseNet169*. *Flowchart* dan *model summary* dapat dilihat pada Gambar 4.4.

Model: "model"

| Layer (type)             | Output Shape          | Param #  |
|--------------------------|-----------------------|----------|
| input_2 (InputLayer)     | [(None, 224, 224, 3)] | 0        |
| sequential (Sequential)  | (None, 224, 224, 3)   | 0        |
| densenet169 (Functional) | (None, 7, 7, 1664)    | 12642880 |
| flatten (Flatten)        | (None, 81536)         | 0        |
| dropout (Dropout)        | (None, 81536)         | 0        |
| dense (Dense)            | (None, 256)           | 20873472 |

=====  
 Total params: 33,516,352  
 Trainable params: 20,873,472  
 Non-trainable params: 12,642,880  
 =====

Gambar 4.4. *Model summary* CNN.

Pada *model summary* diatas, terdapat beberapa lapisan dengan fungsi yang berbeda-beda. Pertama, terdapat lapisan *input* (`input_2`) yang menetapkan bentuk data masukan. Kemudian, ada lapisan `sequential` yang mungkin mengacu pada serangkaian operasi atau lapisan yang diterapkan secara berurutan. Selanjutnya, `densenet169` adalah lapisan inti dari model ini, dengan 12,642,880 parameter yang dapat di-*train*. Lapisan selanjutnya adalah `flatten`, yang mengubah keluaran dari lapisan sebelumnya menjadi *array* satu dimensi. Lapisan `dropout` berperan dalam pencegahan *overfitting* dengan menonaktifkan sebagian unit secara acak. Akhirnya, lapisan `dense` memiliki 20,873,472 parameter yang dapat di-*train* dan menghasilkan keluaran dengan dimensi 256.

Model CNN diatas dibuat dengan *hyperparameter* pada Tabel 4.3 sebagai berikut:

Tabel 4.3. *Hyperparameter training*.

| <i>Hyperparameter</i> |       |
|-----------------------|-------|
| <i>Batch size</i>     | 64    |
| <i>Epoch</i>          | 25    |
| <i>Optimizer</i>      | Adam  |
| <i>Learning Rate</i>  | 0.001 |

Selain parameter diatas, diterapkan beberapa metode pada *library Tensorflow* untuk membantu dalam proses *training*. Metode yang digunakan diantaranya sebagai berikut:

a. Pemberhentian proses *training*

Fungsi untuk menghentikan proses *training* bernama *EarlyStopping*. Menghentikan proses *training* dapat diatur dengan menentukan beberapa parameter seperti *monitor* untuk menentukan nilai yang dijadikan sebagai acuan pemberhentian, *patience* untuk menentukan jumlah rentang *epoch*, *verbose* mengaktifkan pemberitahuan tindakan pemberhentian *training*, dan *restore best weights* untuk mengembalikan nilai bobot terbaik berdasarkan perolehan nilai acuan terbaik. Penentuan parameter yang diatur dalam proses *training* dapat dilihat pada Tabel 4.4 berikut.

Tabel 4.4. *Hyperparameter* fungsi *EarlyStopping*.

|                 |                 |
|-----------------|-----------------|
| <i>Monitor</i>  | <i>val_loss</i> |
| <i>Mode</i>     | <i>min</i>      |
| <i>Patience</i> | 15              |
| <i>Verbose</i>  | 1               |

b. Penyimpanan model

Nama fungsi untuk menyimpan model adalah *ModelCheckpoint*. Fungsi ini memiliki parameter untuk mengatur bagaimana model yang sudah di *training* akan disimpan. Penentuan parameter untuk fungsi ini dapat dilihat pada Tabel 4.5.

Tabel 4.5. *Hyperparameter* fungsi *ModelCheckpoint*.

|                       |                 |
|-----------------------|-----------------|
| <i>Name</i>           | best_model.h5   |
| <i>Monitor</i>        | <i>val_loss</i> |
| <i>Mode</i>           | <i>Min</i>      |
| <i>Save best only</i> | <i>True</i>     |

Berdasarkan parameter tersebut, model yang sudah di *training* akan disimpan dengan nama '*best\_model*' berformat *Hierarchical Data Format* (HDF) atau h5. Parameter *mode* '*min*' ini digunakan jika parameter '*save best only*' diaktifkan (*True*) dan akan menyimpan model *training* dengan perolehan nilai acuan (*val\_loss*) paling kecil.

Berikut merupakan implementasi kode pada proses *training* model dengan data input hasil dari proses *augmentation data*. Sebelum *training* model dimulai, dilakukan



kompilasi model atau mengatur parameter seperti *optimizer*, fungsi *loss*, dan metode-metode yang digunakan untuk membantu dalam proses *training* model.

```
input_shape = (224, 224, 3)
num_label = 3
dense_net = DenseNet169(input_shape=input_shape, classes =
num_label)
for layer in dense_net.layers:
    layer.trainable = False
model = tf.keras.Sequential([
    tf.keras.Input(shape=(224,224,3)),
    dense_net,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(num_label),
])
model.compile(
    optimizer = tf.keras.optimizers.Adam(learning_rate=0.001),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits
=True),
    metrics=['acc']
)
early_stopping = EarlyStopping(monitor='val_loss', mode='min',
patience=15, verbose=1)
mc = ModelCheckpoint('best_model.h5', monitor='val_loss',
mode='min', save_best_only=True)

history = model.fit(
    data_augmentation.flow(train_data, train_labels, batch_size=64),
    validation_data = (val_data, val_labels),
    epochs = 25,
    callbacks = [mc, early_stopping]
)
```

Script 4.6. Training model CNN.

Kode pada Script 4.6 diatas merupakan implementasi dari model *Convolutional Neural Network* (CNN) menggunakan arsitektur *DenseNet169* untuk tugas ekstraksi fitur. Setelah mendefinisikan model, atribut *trainable* dari setiap *layer* dalam model *DenseNet169* diatur menjadi `False`, sehingga *layer-layer* tersebut tidak akan diubah selama proses pelatihan. Selanjutnya, model `Sequential` dibuat dengan menambahkan *layer-layer* seperti *dense\_net* (model *DenseNet169*), *Flatten layer* untuk meratakan *output* dari *dense\_net*, *Dropout layer* untuk menghindari *overfitting*, *Dense layer* dengan 256-unit dan fungsi aktivasi ReLU, dan *Dense layer* terakhir dengan jumlah unit yang sama dengan *num\_label*.

Model tersebut kemudian dikompilasi dengan *optimizer* "adam", *loss* function `SparseCategoricalCrossentropy`, dan metrik akurasi. Selama proses pelatihan, digunakan `callback = EarlyStopping` dan `ModelCheckpoint` untuk

menghentikan pelatihan jika `val_loss` tidak mengalami perbaikan selama 15 *epoch* berturut-turut dan menyimpan model dengan `val_loss` terbaik ke dalam file `'best_model.h5'`. Model tersebut dilatih dengan menggunakan data pelatihan dan validasi, dengan *batch size* 64 dan 25 *epoch* dan menggunakan *callback* `early_stopping` dan `mc` yang telah dibuat sebelumnya.

```
# menyimpan hasil ekstraksi fitur CNN
feature_extraction_model = keras.Model(inputs = model.input,
outputs = model.layers[-2].output)
feature_extraction_model.save('ekstraksi_fitur.h5')
```

Script 4.7. Menyimpan model ekstraksi fitur CNN.

Kode pada Script 4.7 diatas digunakan untuk membuat sebuah model baru menggunakan *library* Keras untuk ekstraksi fitur dari model *neural network* yang sudah ada. Model baru ini mengambil *input* dari model awal dan menghasilkan *output* dari *layer* kedua terakhir karena *layer* sebelum *layer output* cenderung memiliki representasi fitur yang lebih abstrak dan umum dari data. Ini karena selama pelatihan, lapisan-lapisan ini mempelajari representasi fitur yang berguna untuk membedakan berbagai objek atau pola di dalam data. Selanjutnya, model tersebut disimpan dalam format H5 dengan nama file `ekstraksi_fitur.h5`.

#### 4.1.6 Support Vector Machine (SVM)

Setelah melewati proses ekstraksi fitur menggunakan algoritma *Convolutional Neural Network* (CNN) dengan menggunakan arsitektur *DenseNet169*. Selanjutnya hasil ekstraksi fitur tersebut akan di klasifikasi menggunakan algoritma *Support Vector Machine* (SVM). Selama proses pelatihan, SVM akan mempelajari pola yang terkandung dalam data *train* guna membedakan antara citra *thorax* normal, positif tuberkulosis, dan positif pneumonia. Di bawah ini adalah potongan *source code* untuk klasifikasi menggunakan SVM.

```
# Load model ekstraksi fitur CNN
loaded_model = keras.models.load_model("ekstraksi_fitur.h5")

train_features = loaded_model.predict(train_data, batch_size=64)
val_features = loaded_model.predict(val_data, batch_size=64)
test_features = loaded_model.predict(test_data, batch_size=64)

def train_svm(features, labels, learning_rate=0.001,
num_epochs=25):
    num_samples, num_features = features.shape
    num_classes = len(np.unique(labels))
```

```

weights = np.zeros((num_classes, num_features))
biases = np.zeros(num_classes)

for epoch in range(num_epochs):
    features, labels = shuffle(features, labels)
    for i in range(num_samples):
        x = features[i]
        y = labels[i]
        scores = np.dot(weights, x) + biases
        correct_class_score = scores[y]
        for j in range(num_classes):
            if j == y:
                continue
            margin = scores[j] - correct_class_score + 1
            if margin > 0:
                weights[y] += learning_rate * x
                weights[j] -= learning_rate * x
                biases[y] += learning_rate
                biases[j] -= learning_rate
    return weights, biases

# fungsi prediksi SVM
def predict_svm(features, weights, biases):
    scores = np.dot(features, weights.T) + biases
    return np.argmax(scores, axis=1)

# Train model SVM pada data training
learning_rate = 0.001
num_epochs = 25
svm_weights, svm_biases = train_svm (train_features, train_labels,
learning_rate, num_epochs)

```

*Script 4.8. Klasifikasi SVM.*

Kode pada *Script 4.8* diatas merupakan implementasi pengklasifikasian menggunakan algoritma SVM. Pertama, model CNN dimuat dari file yang telah dilatih sebelumnya. Kemudian, fitur-fitur diekstraksi dari data latih, validasi, dan uji menggunakan model CNN tersebut. Selanjutnya, terdapat fungsi untuk melatih SVM dengan menggunakan fitur-fitur yang telah diekstraksi dan label yang sesuai. Fungsi ini digunakan untuk memperbarui bobot dan bias SVM selama pelatihan. Terakhir, terdapat fungsi untuk melakukan prediksi menggunakan SVM dengan menghitung skor untuk setiap kelas menggunakan bobot dan bias yang telah dilatih sebelumnya, dan mengembalikan indeks kelas dengan skor tertinggi sebagai prediksi. Dengan menggunakan kode ini, kita dapat melatih SVM menggunakan fitur-fitur yang diekstraksi dari model CNN dan melakukan prediksi menggunakan model SVM tersebut.

## 4.2 Evaluasi Model

Pada tahap ini, evaluasi model dilakukan untuk menilai kinerja model *machine learning* yang telah dibuat dalam mengklasifikasi penyakit pada citra *thorax*, mulai dari tahap klasifikasi menggunakan CNN hingga melakukan klasifikasi hasil ekstraksi fitur CNN dengan algoritma SVM. Tujuannya adalah untuk memahami seberapa baik model ini bekerja. Dalam melakukan evaluasi model, nilai-nilai seperti akurasi, presisi, *recall*, dan *F1-score* akan diamati.

### 4.2.1 Convolutional Neural Network (CNN)

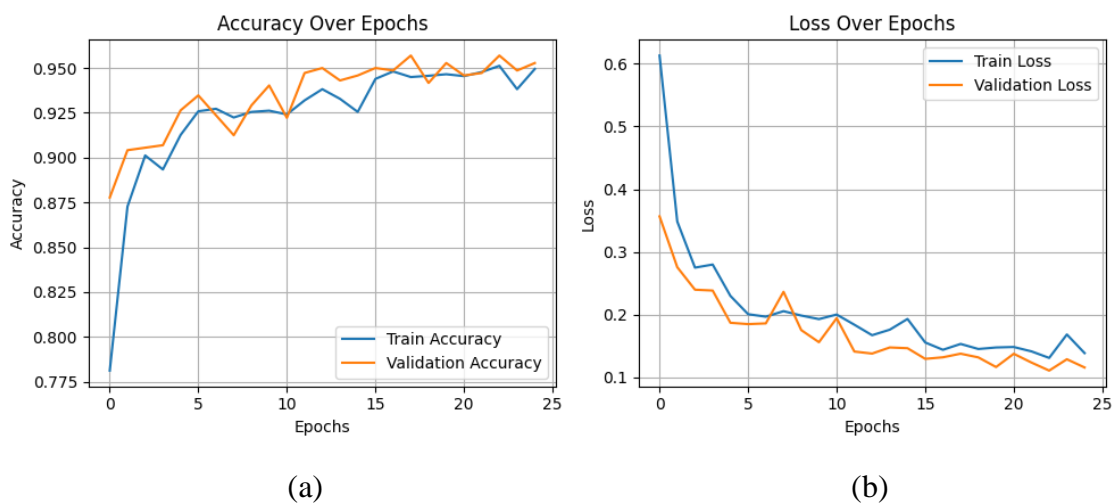
Setelah semua arsitektur model siap maka tahapan selanjutnya adalah melakukan proses *training* data. Pada proses ini dilakukan pembentukan model atau model *fitting* yang akan digunakan untuk proses pengujian. *Hyperparameter* yang sudah ditentukan pada Tahapan 4.1.5. Berikut merupakan hasil proses *training*:

Tabel 4.6. *Training model.*

| No.                             | Epoch    | Loss   |          | Accuracy |              |
|---------------------------------|----------|--------|----------|----------|--------------|
|                                 |          | Loss   | Val_loss | Accuracy | Val_accuracy |
| 1.                              | Epoch 1  | 0.6133 | 0.3566   | 0.7812   | 0.8778       |
| 2.                              | Epoch 2  | 0.3483 | 0.2755   | 0.8727   | 0.9042       |
| 3.                              | Epoch 3  | 0.2748 | 0.2395   | 0.9012   | 0.9056       |
| 4.                              | Epoch 4  | 0.2797 | 0.2382   | 0.8934   | 0.9069       |
| 5.                              | Epoch 5  | 0.2297 | 0.1867   | 0.9127   | 0.9264       |
| ...                             | ...      | ...    | ...      | ...      | ...          |
| 21.                             | Epoch 21 | 0.1481 | 0.1372   | 0.9455   | 0.9458       |
| 22.                             | Epoch 22 | 0.1411 | 0.1232   | 0.9477   | 0.9472       |
| 23.                             | Epoch 23 | 0.1306 | 0.1104   | 0.9512   | 0.9569       |
| 24.                             | Epoch 24 | 0.1680 | 0.1285   | 0.9382   | 0.9486       |
| 25.                             | Epoch 25 | 0.1383 | 0.1154   | 0.9495   | 0.9528       |
| Total Training akurasi = 96,82% |          |        |          |          |              |

Pada Tabel 4.6 *training model* diatas, hasil *training* model *Convolutional Neural Network* (CNN) yang dilakukan sebanyak 25 *epoch* menunjukkan adanya peningkatan performa seiring dengan berjalannya *epoch*. Pada awalnya, model memiliki *loss* sebesar 0.6133 dan akurasi sebesar 0.7812 pada data *training*, serta *loss*

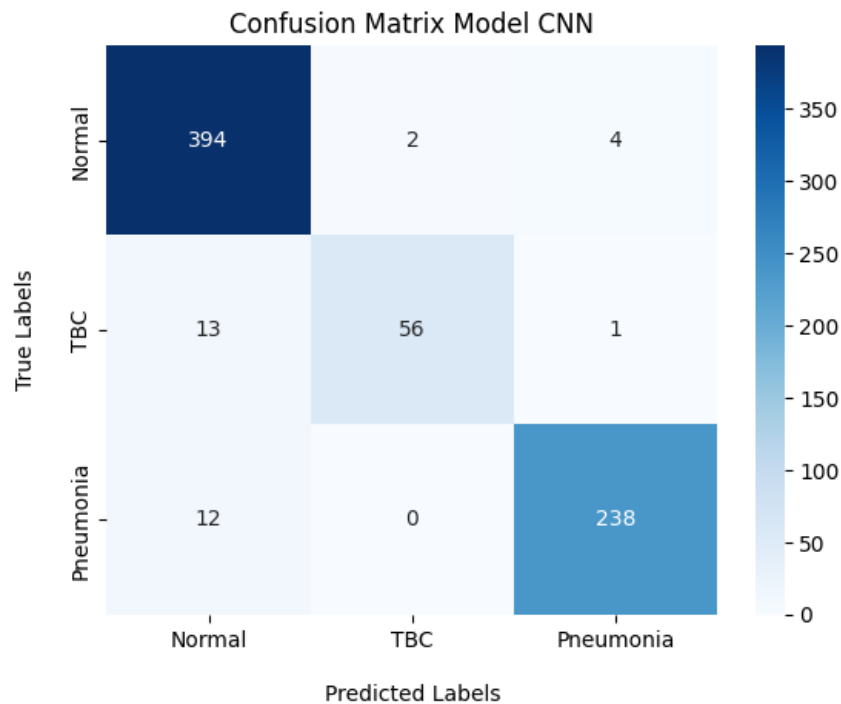
sebesar 0.3566 dan akurasi sebesar 0.8778 pada data validasi. Namun, seiring dengan berjalannya *epoch*, terjadi penurunan *loss* dan peningkatan akurasi pada kedua jenis data. Pada *epoch* terakhir, model mencapai *loss* sebesar 0.1383 dan akurasi sebesar 0.9495 pada data *training*, serta *loss* sebesar 0.1154 dan akurasi sebesar 0.9528 pada data validasi. Hasil ini menunjukkan bahwa model CNN yang dilatih mampu melakukan klasifikasi dengan baik, dengan akurasi yang tinggi dan *loss* yang rendah pada data *training* dan data validasi. Selain itu, model juga menghasilkan akurasi sebesar 96.82 % pada data *training*, yang menunjukkan kemampuan yang sangat baik dalam mengklasifikasikan data. Nilai pada Tabel 4.6 tersebut menghasilkan grafik seperti pada gambar berikut:



Gambar 4.5. (a) Grafik *accuracy* dan (b) Grafik *loss* pada *training* model CNN.

Gambar 4.5 menunjukan grafik akurasi pelatihan secara konsisten meningkat seiring dengan peningkatan jumlah iterasi. Hal ini menunjukkan bahwa model semakin baik dalam mempelajari pola-pola yang ada dalam data pelatihan. Selain itu, grafik kerugian pelatihan menunjukkan bahwa kerugian model secara signifikan menurun seiring dengan peningkatan jumlah iterasi. Penurunan kerugian ini menunjukkan bahwa model semakin efektif dalam mengurangi kesalahan prediksi. Dengan demikian, grafik tersebut menggambarkan peningkatan yang konsisten dalam akurasi pelatihan dan penurunan yang signifikan dalam kerugian pelatihan seiring dengan meningkatnya jumlah iterasi.

Setelah melakukan proses *training* dengan 25 *epoch*, berikutnya melakukan klasifikasi pada data uji dengan jumlah 720 data (400 normal, 70 tbc, dan 250 pneumonia). Berikut merupakan hasil pada proses klasifikasi menggunakan model CNN:



Gambar 4.6. *Confusion Matrix* hasil klasifikasi model CNN.

Berdasarkan *confusion matrix* pada Gambar 4.6 diatas, dapat dianalisa bahwa pada kelas normal, model CNN berhasil mengklasifikasikan 394 sampel dengan benar sebagai normal. Namun, terdapat 2 sampel yang salah diklasifikasikan sebagai tbc dan 4 sampel sebagai pneumonia. Untuk kelas TBC, model CNN berhasil mengklasifikasikan 56 sampel dengan benar sebagai TBC dan terdapat 13 sampel yang salah diklasifikasikan sebagai normal dan 1 sampel sebagai pneumonia. Untuk kelas pneumonia, model CNN berhasil mengklasifikasikan 238 sampel dengan benar sebagai pneumonia. Namun, terdapat 12 sampel yang salah diklasifikasikan sebagai normal dan tidak ada sampel yang diprediksi sebagai tbc. Berdasarkan data tersebut, dapat dikatakan model telah melakukan klasifikasi dengan baik dengan mengklasifikasikan 688 data citra secara benar dan 32 data citra yang tidak sesuai dengan kelas aktualnya.

Selanjutnya, dengan mengacu pada data *confusion matrix* yang telah dipaparkan sebelumnya, maka dapat dilakukan perhitungan akurasi, presisi, *recall*, dan *f1-score*, dengan langkah perhitungan sebagai berikut:

- Kelas Normal

- Presisi

$$pre = \frac{TP}{TP+FP} = \frac{394}{394+6} = 98.05\%$$

- Recall

$$rec = \frac{TP}{TP+FN} = \frac{394}{394+25} = 94.03\%$$

- F1-Score

$$f1 - score = 2 \frac{Precision \times Recall}{Precision+Recall} = 2 \frac{98.05 \times 94.03}{98.05 + 94.03} = 95.99\%$$

- Kelas TBC

- Presisi

$$pre = \frac{TP}{TP+FP} = \frac{56}{56+14} = 80\%$$

- Recall

$$rec = \frac{TP}{TP+FN} = \frac{56}{56+2} = 96.55\%$$

- F1-Score

$$f1 - score = 2 \frac{Precision \times Recall}{Precision+Recall} = 2 \frac{80 \times 96.55}{80 + 96.55} = 87.49\%$$

- Kelas Pneumonia

- Presisi

$$pre = \frac{TP}{TP+FP} = \frac{238}{238+12} = 95.02\%$$

- Recall

$$rec = \frac{TP}{TP+FN} = \frac{238}{238+5} = 97.94\%$$

- F1-Score

$$f1 - score = 2 \frac{Precision \times Recall}{Precision+Recall} = 2 \frac{95.02 \times 97.94}{95.02 + 97.94} = 96.45\%$$

- Akurasi

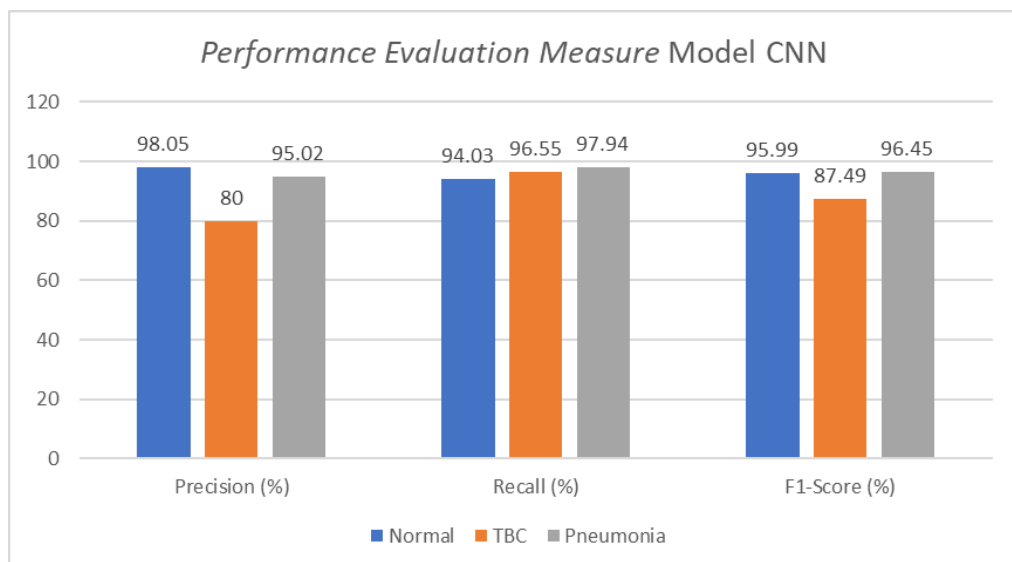
$$acc = \frac{TP}{Total \text{ jumlah data}} = \frac{394 + 56 + 238}{394 + 2 + 4 + 13 + 56 + 1 + 12 + 238} = 95,55 \%$$

Berdasarkan hasil perhitungan evaluasi model diatas, didapatkan nilai *precision*, *recall*, dan *f1-score* yang berbeda pada masing-masing kelas, hal tersebut dikarenakan setiap kelas memiliki proporsi yang berbeda dalam jumlah data citra. Pada nilai *precision* dipengaruhi oleh jumlah data *false positives*, semakin rendah jumlah data *false positives*, semakin tinggi nilai *precision* yang didapatkan. Pada nilai *recall* dipengaruhi oleh jumlah data *false negatives*. Semakin rendah jumlah data *false negatives*, semakin tinggi nilai *recall*. Sedangkan pada *f1-score* merupakan rata-rata harmonik dari nilai *precision* dan *recall* untuk melihat keseimbangan diantara kedua nilai tersebut. Performa model klasifikasi SVM akan ditampilkan Tabel 4.7 berikut.

Tabel 4.7. Performa model klasifikasi CNN.

| Kategori         | Hasil nilai          |                   |                     |                 |
|------------------|----------------------|-------------------|---------------------|-----------------|
|                  | <i>Precision (%)</i> | <i>Recall (%)</i> | <i>F1-Score (%)</i> | <i>Accuracy</i> |
| <b>Normal</b>    | 98,05                | 94,03             | 95,99               | 95,55%          |
| <b>TBC</b>       | 80                   | 96,55             | 87,49               |                 |
| <b>Pneumonia</b> | 95,02                | 97,94             | 96,45               |                 |

Berdasarkan Tabel 4.7, didapatkan grafik *performance evaluation measure* pada Gambar 4.7 berikut ini.



Gambar 4.7. Grafik *performance evaluation measure* model CNN.



Hasil pengujian menggunakan data *testing* sebanyak 720 citra (400 normal, 70 tbc, dan 250 pneumonia) menggunakan model CNN, didapatkan nilai *Precision* 98,05% untuk kelas normal, untuk kelas tbc memperoleh nilai *Precision* 80%, sedangkan untuk kelas pneumonia memperoleh nilai *Precision* 95,02%. Nilai *Precision* ini menentukan sebagaimana tingkat ketepatan atau ketelitian model dalam memprediksi suatu citra. Misalnya, pada kelas normal, presisinya sebesar 98,05% yang berarti 98,05% dari prediksi normal sistem pada kelas tersebut adalah benar. Pada kelas tbc presisinya sebesar 80%, yang berarti 80% dari prediksi tbc sistem pada kelas tersebut adalah benar. Sedangkan pada kelas pneumonia, presisinya sebesar 95,02% yang berarti 96% dari prediksi pneumonia sistem pada kelas tersebut adalah benar.

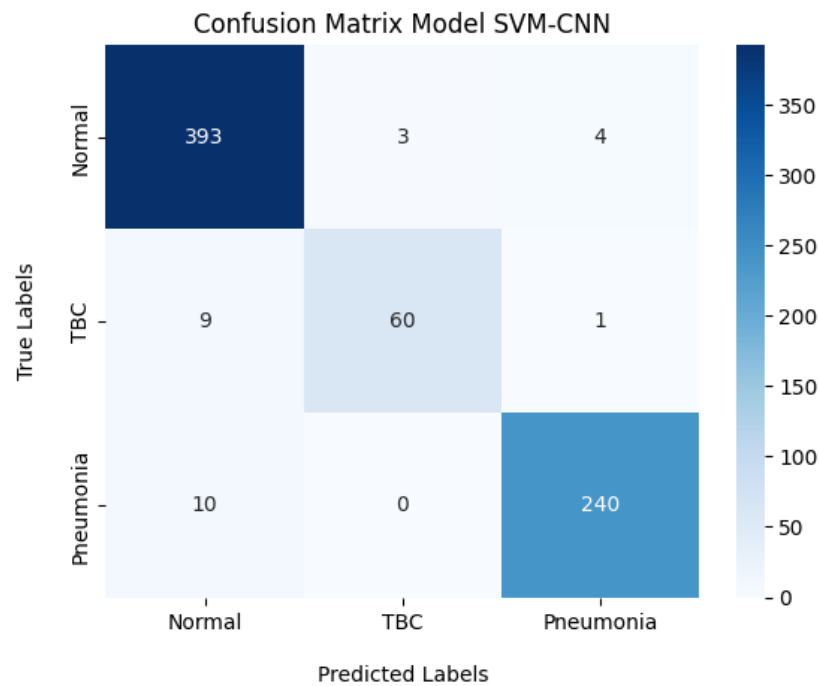
Nilai *recall* didapat 94,03% untuk kelas normal, untuk kelas tbc memperoleh nilai *recall* 96,55%, sedangkan untuk kelas pneumonia memperoleh nilai *recall* 97,94%. Nilai *recall* ini menandakan sebagaimana tingkat keberhasilan model dalam mengenali sebuah citra sesuai dengan kategorinya. Misalkan, pada kelas normal, *recall* mencapai 94,03%, yang berarti bahwa 94,03% dari keseluruhan citra normal berhasil diidentifikasi dengan benar oleh sistem. Hal serupa terjadi pada kelas tbc dan pneumonia dengan tingkat *recall* masing-masing 96,55% dan 97,94%.

Nilai *F1-score* untuk kelas normal adalah 95,99%, untuk kelas tbc memperoleh nilai *F1-score* 87,49%, sedangkan kelas pneumonia mendapat nilai *F1-score* 96,45%. *F1-score* ini merupakan rata-rata harmonik dari nilai *precision* dan *recall* untuk melihat keseimbangan diantara kedua nilai tersebut.

Nilai akurasi model yang diperoleh pada pengujian model sebesar 95,55%, nilai ini untuk mengevaluasi sebagaimana keakuratan hasil prediksi dengan hasil yang sebenarnya. Semua nilai yang diperoleh dari hasil pengujian menandakan kinerja dari model dalam klasifikasi citra normal, tbc atau pneumonia sangat baik.

#### 4.2.2 CNN-SVM

Selanjutnya adalah melakukan klasifikasi pada data uji dengan jumlah 720 data (400 normal, 70 tbc, dan 250 pneumonia) dengan menggunakan algoritma CNN sebagai ekstraksi fitur dan SVM sebagai algoritma klasifikasi. Berikut merupakan hasil pada proses klasifikasi dengan model CNN-SVM:



Gambar 4.8. *Confusion Matrix* hasil klasifikasi model CNN-SVM.

Berdasarkan *confusion matrix* pada Gambar 4.9 diatas, dapat dianalisa bahwa pada kelas normal, model SVM berhasil mengklasifikasikan 393 sampel dengan benar sebagai normal. Namun, terdapat 3 sampel yang salah diklasifikasikan sebagai tbc dan 4 sampel sebagai pneumonia. Untuk kelas TBC, model SVM berhasil mengklasifikasikan 60 sampel dengan benar sebagai TBC dan terdapat 9 sampel yang salah diklasifikasikan sebagai normal dan 1 sampel sebagai pneumonia. Untuk kelas pneumonia, model SVM berhasil mengklasifikasikan 240 sampel dengan benar sebagai pneumonia. Namun, terdapat 10 sampel yang salah diklasifikasikan sebagai normal dan tidak ada sampel yang diprediksi sebagai tbc. Berdasarkan data tersebut, dapat dikatakan model telah melakukan klasifikasi dengan baik dengan mengklasifikasikan 693 data citra secara benar dan 27 data citra yang tidak sesuai dengan kelas aktualnya.

Selanjutnya, dengan mengacu pada data *confusion matrix* yang telah dipaparkan sebelumnya, maka dapat dilakukan perhitungan akurasi, presisi, *recall*, dan *f1-score*, dengan langkah perhitungan sebagai berikut:

- Kelas Normal

- Presisi

$$pre = \frac{TP}{TP+FP} = \frac{393}{393+7} = 98.25\%$$

- Recall

$$rec = \frac{TP}{TP+FN} = \frac{393}{393+19} = 95.38\%$$

- F1-Score

$$f1 - score = 2 \frac{Precision \times Recall}{Precision+Recall} = 2 \frac{98.25 \times 95.38}{98.25 + 95.38} = 96.79\%$$

- Kelas TBC

- Presisi

$$pre = \frac{TP}{TP+FP} = \frac{60}{60+10} = 85.71\%$$

- Recall

$$rec = \frac{TP}{TP+FN} = \frac{60}{60+3} = 95.23\%$$

- F1-Score

$$f1 - score = 2 \frac{Precision \times Recall}{Precision+Recall} = 2 \frac{85.71 \times 95.23}{85.71 + 95.23} = 90.21\%$$

- Kelas Pneumonia

- Presisi

$$pre = \frac{TP}{TP+FP} = \frac{240}{240+10} = 96\%$$

- Recall

$$rec = \frac{TP}{TP+FN} = \frac{240}{240+5} = 98\%$$

- F1-Score

$$f1 - score = 2 \frac{Precision \times Recall}{Precision+Recall} = 2 \frac{96 \times 98}{96 + 98} = 97\%$$

- Akurasi

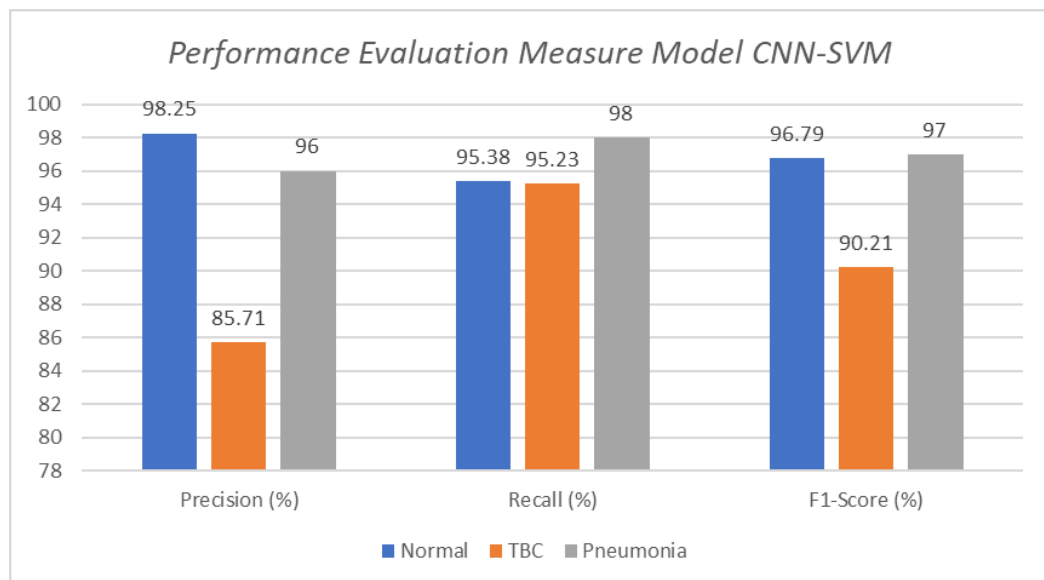
$$acc = \frac{TP}{Total \text{ jumlah data}} = \frac{393+60+240}{397+3+4+9+60+1+10+240} = 96,25 \%$$

Berdasarkan hasil perhitungan evaluasi model diatas, didapatkan nilai *precision*, *recall*, dan *f1-score* yang berbeda pada masing-masing kelas, hal tersebut dikarenakan setiap kelas memiliki proporsi yang berbeda dalam jumlah data citra. Pada nilai *precision* dipengaruhi oleh jumlah data *false positives*, semakin rendah jumlah data *false positives*, semakin tinggi nilai *precision* yang didapatkan. Pada nilai *recall* dipengaruhi oleh jumlah data *false negatives*. Semakin rendah jumlah data *false negatives*, semakin tinggi nilai *recall*. Sedangkan pada *f1-score* merupakan rata-rata harmonik dari nilai *precision* dan *recall* untuk melihat keseimbangan diantara kedua nilai tersebut. Performa model klasifikasi SVM akan ditampilkan Tabel 4.8 berikut.

Tabel 4.8. Performa model klasifikasi CNN-SVM.

| Kategori         | Hasil nilai          |                   |                     |                 |
|------------------|----------------------|-------------------|---------------------|-----------------|
|                  | <i>Precision</i> (%) | <i>Recall</i> (%) | <i>F1-Score</i> (%) | <i>Accuracy</i> |
| <b>Normal</b>    | 98,25                | 95,38             | 96,79               | 96,25%          |
| <b>TBC</b>       | 85,71                | 95,23             | 90,21               |                 |
| <b>Pneumonia</b> | 96                   | 98                | 97                  |                 |

Berdasarkan Tabel 4.8, didapatkan grafik *performance evaluation measure* pada Gambar 4.9 berikut ini.



Gambar 4.9. Grafik *performance evaluation measure* model CNN-SVM.

Hasil pengujian menggunakan data *testing* sebanyak 720 citra (400 normal, 70 tbc, dan 250 pneumonia) menggunakan model CNN-SVM, didapatkan nilai *Precision* 98,25% untuk kelas normal, untuk kelas tbc memperoleh nilai *Precision* 85,71%, sedangkan untuk kelas pneumonia memperoleh nilai *Precision* 96%. Nilai *Precision* ini menentukan sebagaimana tingkat ketepatan atau ketelitian model dalam memprediksi suatu citra. Misalnya, pada kelas normal, presisinya sebesar 98.25, yang berarti 98.25% dari prediksi normal sistem pada kelas tersebut adalah benar. Pada kelas tbc presisinya sebesar 85.71, yang berarti 85.71% dari prediksi tbc sistem pada kelas tersebut adalah benar. Sedangkan pada kelas pneumonia, presisinya sebesar 96.00, yang berarti 96% dari prediksi pneumonia sistem pada kelas tersebut adalah benar.

Nilai *recall* didapat 95,38% untuk kelas normal, untuk kelas tbc memperoleh nilai *recall* 95,23%, sedangkan untuk kelas pneumonia memperoleh nilai *recall* 98%. Nilai *recall* ini menandakan sebagaimana tingkat keberhasilan model dalam mengenali sebuah citra sesuai dengan kategorinya. Misalkan, pada kelas normal, *recall* mencapai 95,38%, yang berarti bahwa 95,38% dari keseluruhan citra normal berhasil diidentifikasi dengan benar oleh sistem. Hal serupa terjadi pada kelas tbc dan pneumonia dengan tingkat Recall masing-masing 95,23% dan 98%.

Nilai *F1-score* untuk kelas normal adalah 96,79%, untuk kelas tbc memperoleh nilai *F1-score* 90,21%, sedangkan kelas pneumonia mendapat nilai *F1-score* 97%. *F1-score* ini merupakan rata-rata harmonik dari nilai *precision* dan *recall* untuk melihat keseimbangan diantara kedua nilai tersebut.

Nilai akurasi model yang diperoleh pada pengujian model sebesar 96,25%, nilai ini untuk mengevaluasi sebagaimana keakuratan hasil prediksi dengan hasil yang sebenarnya. Semua nilai yang diperoleh dari hasil pengujian menandakan kinerja dari model dalam klasifikasi citra normal, tbc atau pneumonia sangat baik. Dengan perolehan yang sudah dipaparkan tersebut, model dapat dijadikan sebagai kontribusi dalam pengembangan sistem *machine learning* untuk klasifikasi penyakit pada citra *thorax* menggunakan metode CNN sebagai ekstraksi fitur dan SVM sebagai algoritma klasifikasi.

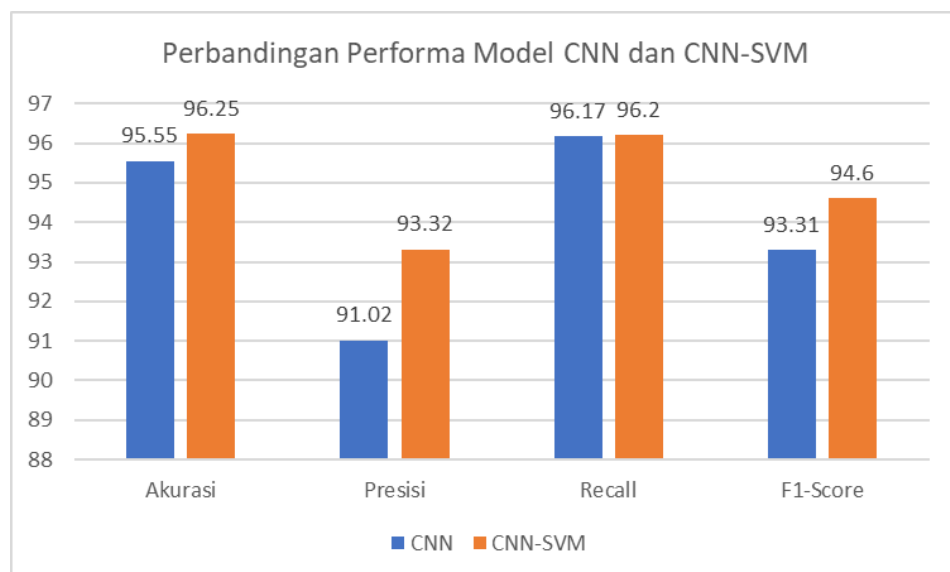
### 4.3 Perbandingan Model CNN dan CNN-SVM

Pada tahap ini, kedua model yaitu model CNN sebagai ekstraksi fitur sekaligus klasifikasi dan model CNN-SVM yang menggunakan CNN untuk ekstraksi fitur dan SVM sebagai algoritma klasifikasi. Tabel 4.9 menyajikan kinerja kedua metode untuk empat metrik yang diukur. Tabel tersebut menunjukkan bahwa klasifikasi yang dilakukan dengan menggunakan model CNN berada di angka 95,55% untuk nilai akurasi. Sementara itu, performa klasifikasi menggunakan metode CNN-SVM memberikan nilai yang lebih tinggi dengan akurasi sebesar 96,25%. Nilai *precision* didapatkan dengan mencari nilai *average macro* atau rata-rata dari total nilai *precision* tiap kelas, begitu juga nilai *recall* dan *f1-score*.

Tabel 4.9. Perbandingan model CNN dan CNN-SVM.

| Metrik   | Model  |         |
|----------|--------|---------|
|          | CNN    | CNN-SVM |
| Akurasi  | 95,55% | 96,25%  |
| Presisi  | 91,02% | 93,32%  |
| Recall   | 96,17% | 96,20%  |
| F1-Score | 93,31% | 94,6%   |

Berdasarkan Tabel 4.9, didapatkan grafik perbandingan model pada Gambar 4.10 berikut ini.

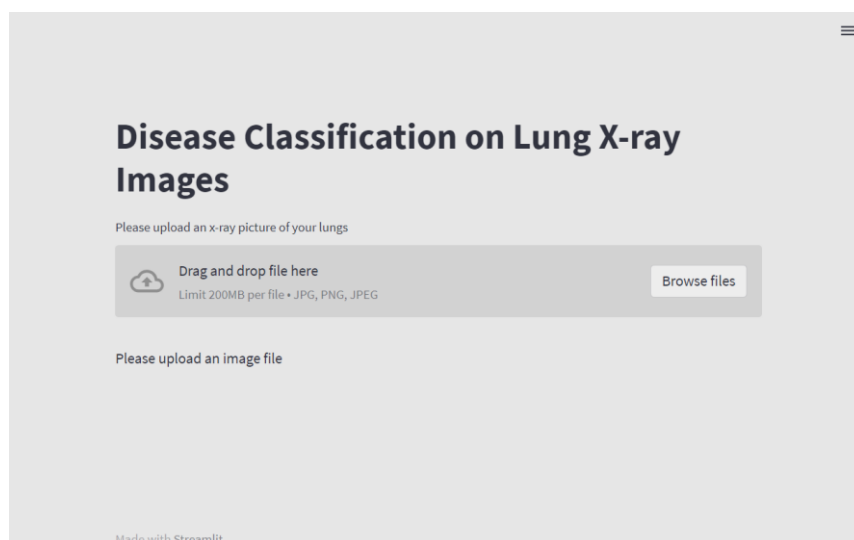


Gambar 4.10. Grafik perbandingan model CNN dan CNN-SVM.

Berdasarkan grafik diatas dapat dilihat perbandingan performa model CNN dan CNN-SVM. Didapatkan metode CNN dengan akurasi 95,55%, presisi 91,02%, *recall* 96,17%, dan *F1-Score* 93,31%. Sedangkan metode CNN-SVM mencapai performa terbaik dengan akurasi 96,25%, presisi 93,32%, *recall* 96,20%, dan *F1-Score* 94,6%. Berdasarkan hasil tersebut, secara keseluruhan dapat dilihat bahwa klasifikasi menggunakan SVM dengan ekstraksi fitur CNN memiliki kinerja yang lebih unggul daripada model klasifikasi yang hanya menggunakan algoritma CNN saja. Model CNN-SVM menunjukkan akurasi yang lebih tinggi dan memiliki nilai presisi serta *recall* yang lebih baik untuk ketiga kelas yaitu normal, tbc, dan pneumonia. Nilai *F1-Score* juga lebih tinggi pada model CNN-SVM yang menunjukkan kemampuannya dalam menjaga keseimbangan antara presisi dan *recall*.

#### 4.4 Hasil Perancangan Aplikasi Berbasis Web

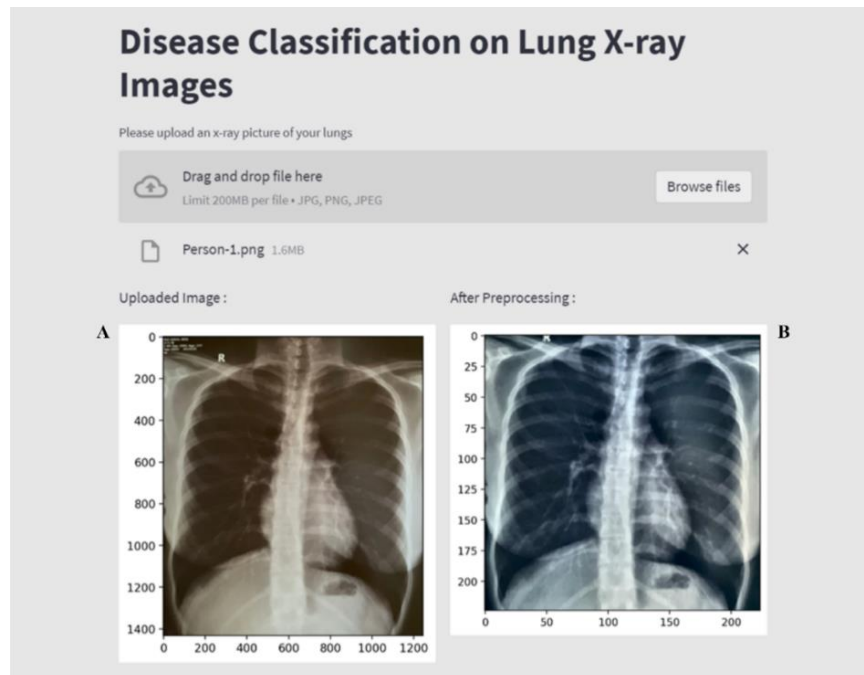
Setelah melalui tahapan perancangan model klasifikasi penyakit berdasarkan citra *thorax*, selanjutnya dengan menggunakan algoritma *Convolutional Neural Network* (CNN) sebagai ekstraksi fitur dan *Support Vector Machine* (SVM) sebagai algoritma klasifikasi akan dilakukan implementasi model klasifikasi *machine learning* dari model yang telah dibuat kedalam sebuah sistem berbasis *website*. *Website* ini dapat melakukan klasifikasi penyakit pada citra *thorax* yang telah di-*upload* oleh *user*. Berikut merupakan tampilan *website* klasifikasi.



Gambar 4.11. Halaman awal aplikasi *web* klasifikasi.

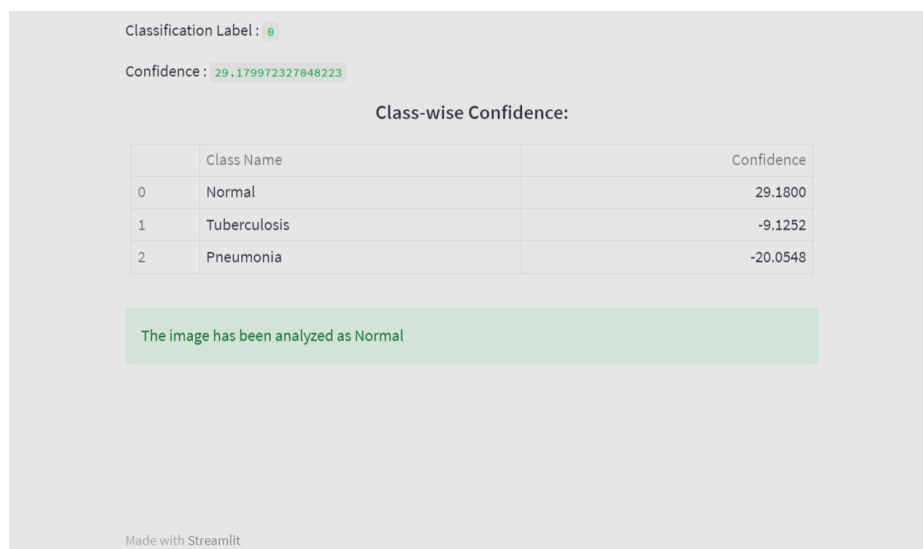
Gambar 4.11 diatas merupakan halaman awal aplikasi web klasifikasi, Halaman ini digunakan *user* untuk meng-*upload* citra *thorax* yang akan diklasifikasi

menggunakan model yang telah dilatih. Adapun tampilan saat *user* telah meng-*upload* citra *thorax* seperti Gambar 4.12 berikut ini.



Gambar 4.12. Tampilan *web* saat *user* telah *upload* citra *thorax*. (a) Citra *thorax* asli  
(b) Citra *thorax* telah melalui tahap *preprocessing*.

Pada Gambar 4.12 diatas, *user* telah meng-*upload* citra *thorax* dengan file yang bernama ‘Person-1.png’. Setelah ter-*upload*, sistem akan menampilkan citra *thorax* yang telah di-*upload* oleh *user* beserta gambar hasil setelah melewati tahap *preprocessing*.



Gambar 4.13. Tampilan *web* hasil klasifikasi citra *thorax*.



Pada Gambar 4.13, sistem akan menampilkan label klasifikasi (0, 1, atau 2), menampilkan nilai *confidence* atau seberapa yakin model terhadap klasifikasi yang dihasilkan. Setelah itu sistem akan menampilkan keterangan tambahan apakah citra *thorax* yang telah di-*upload* termasuk kelas normal, tuberkulosis, atau pneumonia. Pada gambar tersebut, dapat dilihat bahwa hasil klasifikasi citra yang telah di-*upload* termasuk ke kelas 0 (Normal) dengan nilai *confidence* sebesar 29,18 dan sistem memberi keterangan '*The image has been analyzed as Normal*'.

## BAB V

### KESIMPULAN DAN SARAN

#### 5.1 Kesimpulan

Berdasarkan penelitian yang sudah dilakukan, diperoleh beberapa kesimpulan yaitu sebagai berikut:

1. Klasifikasi citra *thorax* dapat dilakukan dengan menggunakan algoritma *Convolutional Neural Network* (CNN) sebagai ekstraksi fitur dan *Support Vector Machine* (SVM) sebagai algoritma klasifikasi.
2. Pada data uji sebanyak 720 citra, didapatkan nilai terbaik menggunakan model CNN-SVM dengan nilai *Precision* 98,25%, *Recall* 95,38%, dan *F1-score* 96,79% untuk kelas normal. *Precision* 85,71%, *Recall* 95,23%, dan *F1-score* 90,21% untuk kelas TBC. *Precision* 96%, *Recall* 98%, dan *F1-score* 97% pada kelas Pneumonia. Sedangkan nilai *Accuracy* model sebesar 96,25%. Perbedaan nilai *precision*, *recall*, dan *f1-score* pada setiap kelas dikarenakan proporsi jumlah data citra yang berbeda. *Precision* dipengaruhi oleh jumlah data *false positives*. *Recall* dipengaruhi oleh jumlah data *false negatives*, dan *F1-score* sebagai rata-rata harmonik dari *precision* dan *recall*.
3. Model klasifikasi penyakit berdasarkan citra *thorax* menggunakan *Convolutional Neural Network* (CNN) dan *Support Vector Machine* (SVM) dapat diimplementasikan kedalam aplikasi berbasis *web* tanpa melalui proses *training* pada aplikasi tersebut.

#### 5.2 Saran

Ada beberapa saran yang dapat penulis berikan apabila penelitian ini akan dikembangkan kembali antara lain sebagai berikut.

1. Untuk penelitian berikutnya dapat menerapkan arsitektur CNN yang berbeda dan algoritma klasifikasi yang berbeda.
2. Melakukan pengimplementasian pada android.
3. Dataset yang digunakan untuk selanjutnya lebih di perbanyak dan seimbang antara tiap kelas sehingga tidak diperlukan lagi *augmentation* data sebagai alternatif untuk memperbanyak data citra.

4. Spesifikasi komputer yang digunakan lebih tinggi, yaitu dengan menggunakan komputer dengan *Random Access Memory* (RAM) yang tinggi dan *Graphics Processing Unit* (GPU).

## DAFTAR PUSTAKA

- Alawi, A. E., Al-basser, A., Sallam, A., Al-sabaei, A., & Al-khateeb, H. (2021). Convolutional Neural Networks Model for Screening Tuberculosis Disease. *International Conference of Technology, Science and Administration (ICTSA)*. doi:10.1109/ICTSA52017.2021.9406520
- Aziz, F. A., & Setiyawan, I. (2017). Klasifikasi Data EEG Epilepsi Menggunakan Support Vector Machine dengan Kernel Radial Basis Function. *Jurnal Sistem Komputer dan Informatika (J-SAKTI)*.
- Dalvi, P. P., Edla, D. R., & Purushothama, B. R. (2023). Diagnosis of Coronavirus Disease From Chest X-Ray Images Using DenseNet-169 Architecture. *SN Computer Science*. doi:<https://doi.org/10.1007/s42979-022-01627-7>
- Darsyah, M. Y. (2014). Klasifikasi Tuberkulosis Dengan Pendekatan Metode Supports Vector Machine (SVM). *Statistika*, 2, No. 2.
- Hamid, A. (2019). Klasifikasi Penyakit Tuberkulosis dan Pneumonia Pada Paru-Paru Manusia Berdasarkan Citra Chest X-Ray Menggunakan Convolutional Neural Network. *Jakarta: UIN Syarif Hidayatullah*.
- Hipzi, A. A., Wiriasto, G. W., & Suksmadana, I. M. (2023). Klasifikasi Pneumonia Pada Augmentasi Citra X-Ray Paru-Paru Menggunakan Metode Convolution Neural Network (CNN). *Jurusan Teknik Elektro Universitas Mataram*.
- Huang, G., Liu, Z., Maaten, L. v., & Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. *IEEE*.
- Indriani, D., Adiningsih, S., Mahmudiono, T. (2005). Faktor resiko yang mem-pengaruhi kejadian TB paru pada anak jalanan dengan studi kasus di Yayasan Insani Surabaya. *Jurnal FKM UA. Surabaya*
- Kavitha, S., Poornima, S., Sitara, N. S., & Devi, A. S. (2020). Classification of Lung Tuberculosis using Non Parametric and Deep Neural Network Techniques. *4th International Conference on Computer, Communication and Signal Processing (ICCCSP)*. doi:10.1109/ICCCSP49186.2020.9315211
- Keras. (2022). Keras Applications. Keras.Io. <https://keras.io/api/applications/>
- Kermany, D., Zhang, K., & Goldbaum, M. (2018). Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification. *Mendeley Data*, V2. doi:10.17632/rschjbr9sj.2
- Khasanah, L. U. (2022). *Tiga Metode Machine Learning yang Wajib Diketahui*. Retrieved from DQLab: <https://dqlab.id/tiga-metode-machine-learning-yang-wajib-diketahui>

- Kusumaningrum, T. F. (2018). Implementasi Convolution Neural Network (Cnn) Untuk Klasifikasi Jamur Konsumsi Di Indonesia Menggunakan Keras. *Universitas Islam Indonesia*.
- Loey, M., Manogaran, G., & Khalifa, N. E. M. (2020). A deep transfer learning model with classical data augmentation and CGAN to detect Covid-19 from chest CT radiography digital images. *Neural Computing and Applications*, 0123456789
- Makarim, F. R. (2022, Juny 20). *halodoc*. Retrieved October 24, 2023, from <https://www.halodoc.com/kesehatan/pneumonia>
- Mayasari, K. (2019, May 28). *honestdocs*. Retrieved from <https://www.honestdocs.id/lung-consolidation>
- Mizan, M. B., Hasan, D. M., & Hassan, R. S. (2020). A Comparative Study of Tuberculosis Detection Using Deep Convolutional Neural Network. *2nd International Conference on Advanced Information and Communication Technology (ICAICT)*. doi:10.1109/ICAICT51780.2020.9333464
- Mu, N., & Qiao, D. (2019). Image Classification Based on Convolutional Neural Network and Support Vector Machine. *International Conference on Information, Cybernetics, and Computational Social Systems (ICCSS)*.
- Nazalia, C. L. (2022). Implementasi Algoritma Convolutional Neural Network Untuk Deteksi Hama Di Tanaman Sawi Hijau. *Jakarta: Institut Teknologi-PLN*.
- Peryanto, A., Yudhana, A., & Umar, R. (2019). Rancang Bangun Klasifikasi Citra Dengan Teknologi Deep Learning Berbasis Metode Convolutional Neural Network. *8, No. 2*.
- Rahman, T., Khandakar, A., Kadir, M. A., Islam, K. R., Islam, K. F., Mahbub, Z. B., Chowdhury, M. E. (2020). Reliable Tuberculosis Detection using Chest X-ray with Deep Learning, Segmentation and Visualization. *IEEE Access*, 8, pp 191586 - 191601. doi:10.1109/ACCESS.2020.3031384.
- Putra, Darma. 2010. Pengolahan Citra Digital (Edisi 1). Yogyakarta: Andi. Kadir, Abdul. & Susanto, Adhi. 2013. Teori dan Aplikasi Pengolahan Citra. Ed. I.
- Rasyid, A., & Heryawan, L. (2023). Klasifikasi Penyakit Tuberculosis (TB) Organ Paru Manusia Berdasarkan Citra Rontgen Thorax Menggunakan Metode Convolutional Neural Network (CNN). *Jurnal Manajemen Informasi Kesehatan Indonesia*, 11 No. 1. doi:10.33560/jmiki.v11i1.484
- Saputra, A. D., Hindarto, D., & Santoso, H. (2023). Klasifikasi Penyakit pada Daun Padi menggunakan Densenet121, Densenet169, Densenet201. *Universitas Pradita*.
- Scholastica, F. A. P. (2018). Asuhan Keperawatan Pada Pasien Dengan Gangguan Sistem Pernapasan. *Yogyakarta: Pustaka Baru Press*, pp. 14–99.
- Soni, A., Rai, A., & Ahirwar, S. K. (2021). Mycobacterium Tuberculosis Detection using Support Vector Machine Classification Approach. *10th IEEE International Conference on Communication Systems and Network Technologies*.

- Sutojo, Andono, P. Nurtantio, T., & Muljono. (2017). Pengolahan Citra Digital (A.Pramesta, ed). *Yogyakarta: Penerbit Andi*.
- Tampubolon, E. R., & Sari, A. P. (2018). Klasifikasi Jenis Buah-buahan pada Citra dengan Menggunakan *Support Vector Machine*. *Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI)*, 7(1), 36-43.
- Utami, d. P. (n.d.). Retrieved Oktober 2023, from Alomedika: <https://www.alomedika.com/penyakit/pulmonologi/tuberkulosis-paru/diagnosis>
- World Health Organization (2019). Tuberculosis Control in The South East Asia Region and Pneumonia. *New Delhi, India: Region Office for South-East As*.

#### Citation

(Hipzi, Wiriasto, & Suksmadana, 2023)  
(Nazalia, 2022)  
(Hamid, 2019)  
(Dalvi, Edla, & Purushothama, 2023)  
(Huang, Liu, Maaten, & Weinberger, 2017)  
(Kusumaningrum, 2018)  
(Saputra, Hindarto, & Santoso, 2023)  
(Rasyid & Heryawan, 2023)  
(Darsyah, 2014)  
(Peryanto, Yudhana, & Umar, 2019)  
(Mizan, Hasan, & Hassan, 2020)  
(Kavitha, Poornima, Sitara, & Devi, 2020)  
(Alawi, Al-basser, Sallam, Al-sabaei, & Al-khateeb, 2021)  
(Mu & Qiao, 2019)  
(Soni, Rai, & Ahirwar, 2021)  
(Khasanah, 2022)  
(Rahman, et al., 2020)  
(Kermany, Zhang, & Goldbaum, 2018)  
(Mayasari, 2019)