Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - KI141502

KLASIFIKASI TINGKAT KEGANASAN KANKER PARU-PARU PADA CITRA COMPUTED TOMOGRAPHY (CT) SCAN MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK

ANDREAS GALANG ANUGERAH
NRP 05111440000153

Dosen Pembimbing I
Dr.Eng. Nanik Suciati, S.Kom, M.Kom

Dosen Pembimbing II
Dr.Eng. Chastine Fatichah, S.Kom, M.Kom

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

TUGAS AKHIR - KI141502

KLASIFIKASI TINGKAT KEGANASAN KANKER PARU-PARU PADA CITRA COMPUTED TOMOGRAPHY (CT) SCAN MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK

ANDREAS GALANG ANUGERAH
NRP 05111440000153

Dosen Pembimbing I
Dr.Eng. Nanik Suciati, S.Kom, M.Kom

Dosen Pembimbing II
Dr.Eng. Chastine Fatichah, S.Kom, M.Kom

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI141502

CLASSIFICATION LEVEL OF CANCER RATE RATIONS IN COMPUTED TOMOGRAPHY (CT) SCAN USING CONVOLUTIONAL NEURAL NETWORK METHOD

ANDREAS GALANG ANUGERAH
NRP 05111440000153

Supervisor I
Dr.Eng. Nanik Suciati, S.Kom, M.Kom

Supervisor II
Dr.Eng. Chastine Fatichah, S.Kom, M.Kom

DEPARTMENT OF INFORMATICS
Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

KLASIFIKASI TINGKAT KEGANASAN KANKER PARU-PARU PADA CITRA COMPUTED TOMOGRAPHY (CT) SCAN MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer pada
Bidang Studi Komputasi Cerdas dan Visualisasi
Program Studi S-1 Departemen Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
ANDREAS GALANG ANUGERAH
NRP : 05111440000153

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Dr.Eng. Nanik Suciati, S.Kom, M.Kom. ..... (pembimbing 1)
NIP: 19710428 199412 2 001

Dr.Eng. Chastine Fatichah, S.Kom, M.Kom. ..... (pembimbing 2)
NIP: 19751220 200112 2 002

SURABAYA
JUNI 2018

[Halaman ini sengaja dikosongkan]

KLASIFIKASI TINGKAT KEGANASAN KANKER PARU-PARU PADA CITRA COMPUTED TOMOGRAPHY (CT) SCAN MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK

Nama Mahasiswa	**: ANDREAS GALANG ANUGERAH**
NRP	**: 05111440000153**
Departemen	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Dr.Eng. Nanik Suciati, S.Kom, M.Kom.**
Dosen Pembimbing 2	**: Dr.Eng. Chastine Fatichah, S.Kom, M.Kom.**

ABSTRAK

Deteksi dini kanker akan memudahkan dalam proses pengobatan maupun tindak lanjut untuk proses penyembuhan sehingga dapat menyelamatkan jutaan nyawa di seluruh dunia setiap tahunnya. Diperlukan suatu metode yang dapat mengenali kanker paru-paru hanya dengan melalui citra Computed Tomography (CT) scan paru-paru seorang pasien. Salah satu metode pengenalan maupun klasifikasi citra yaitu metode Convolutional Neural Network (CNN).

Tugas akhir ini mengimplementasikan metode CNN pada klasifikasi tingkat keganasan kanker paru-paru melalui citra CT scan seorang pasien. Dalam penyusunan Tugas Akhir ini dilakukan percobaan terhadap beragam arsitektur CNN dan optimasi yang biasa digunakan. Tugas Akhir ini membandingkan 4 arsitektur CNN untuk klasifikasi tingkat keganasan paru-paru, yaitu CanNet, LeNet, VGG16, dan VGG19. Dataset yang digunakan yaitu dataset LIDC-IDRI. Model akan dilatih menggunakan berbagai macam metode optimasi CNN. Model menghasilkan keluaran berupa kelas malignant atau kelas benign.

Dari hasil uji coba, didapatkan nilai akurasi klasifikasi rata-rata terbesar pada data citra CT scan paru-paru yaitu 81.4% dengan menggunakan arsitektur LeNet. Arsitektur CanNet memiliki run time yang paling lama yaitu sekitar 34 jam. Penggunaan ukuran kernel terbukti berpengaruh terhadap run time sebuah arsitektur CNN. Metode optimasi Stochastic Gradient Descent (SGD) menjadi metode optimasi yang paling optimal dengan hasil akurasi 91.4%. Ketiga metode optimasi lainnya, yaitu Root Mean Square Propagation (RMSProp), Adaptive Gradient Algorithm (Adagrad), dan Adaptive Moment Estimation (Adam) tidak mengalami perubahan akurasi sejak epoch pertama.

Kata kunci: Convolutional Neural Network, Deep Learning, CT Scan Paru-paru, LIDC-IDRI

CLASSIFICATION LEVEL OF CANCER RATE RATIONS IN COMPUTED TOMOGRAPHY (CT) SCAN USING CONVOLUTIONAL NEURAL NETWORK METHOD

Student's Name : **ANDREAS GALANG ANUGERAH**
Student's ID : **05111440000153**
Department : **Teknik Informatika FTIF-ITS**
First Advisor : **Dr.Eng. Nanik Suciati, S.Kom, M.Kom.**
Second Advisor : **Dr.Eng. Chastine Fatichah, S.Kom, M.Kom.**

ABSTRACT

Early detection of cancer is preferred for the treatment process as well as follow-up to the healing process so that it can save millions of lives around the world every year. It requires a method that can recognize lung cancer only through the image of a patient's computed tomography (CT) scan. One of recognition and image classification methods is Convolutional Neural Network (CNN).

This final project implements the CNN method on the classification of lung cancer malignancy level using CT scan image. This Final Project, conducted experiments on various CNN architecture and optimization. This Final Project compares the 4 CNN architectures for the classification of lung malignancy rates, i.e. CanNet, LeNet, VGG16, and VGG19. The CT scan images used is from the LIDC-IDRI dataset. Models was trained using various methods of optimization for CNN. The model produces output in the form of malignant class or benign class.

The experimental results achiev the greatest average classification accuracy value on the image of CT scan lung image is 81.4% using the LeNet architecture. The CanNet architecture has the longest run time of about 34 hours. The use of kernel size has been shown to affect the run time of a CNN architecture. Stochastic Gradient Descent (SGD) optimization method became the optimum optimization method with 91.4% accuracy. The three

other optimization methods, Root Mean Square Propagation (RMSProp), Adaptive Gradient Algorithm (Adagrad), and Adaptive Moment Estimation (Adam), have not changed accurately since the first epoch

Keywords: Convolutional Neural Network, Deep Learning, Lung CT Scan, LIDC-IDRI

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan tugas akhir yang berjudul:

KLASIFIKASI TINGKAT KEGANASAN KANKER PARU-PARU PADA CITRA COMPUTED TOMOGRAPHY (CT) SCAN MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK

Melalui lembar ini, penulis ingin menyampaikan ucapan terima kasih dan penghormatan yang sebesar-besarnya kepada:

1. Kedua orang tua, Bapak Yohanes Supriyo serta Ibu Sriati, kedua saudara Ignatius Agung Pratama, serta Tria Agatha Estu Rahajeng dan keluarga besar yang selalu memberikan dukungan penuh untuk menyelesaikan tugas akhir ini.
2. Ibu Dr.Eng. Nanik Suciati, S.Kom, M.Kom. selaku dosen pembimbing tugas akhir pertama yang telah membimbing dan memberikan banyak nasihat dalam pengerjaan tugas akhir ini.
3. Ibu Dr.Eng. Chastine Fatichah, S.Kom, M.Kom. selaku dosen pembimbing tugas akhir kedua yang selalu memberikan koreksi serta banyak nasihat yang dapat penulis kembangkan dari tugas akhir ini.
4. Bapak dan Ibu dosen Departemen Informatika ITS yang telah banyak menyampaikan ilmu dan bimbingan yang tak ternilai harganya bagi penulis.
5. Habib, Luqman, Anwar, Upik, Dini, Mala yang selalu menemani, mengingatkan dan menyemangati dalam pengerjaan tugas akhir ini.
6. Rekan-rekan di laboratorium Komputasi Cerdas dan Visi, Afiif, TP, Datin, Lucha, Tion, Delia, Dzaky, Petrus dan teman-teman yang lain yang menemani penulis dalam suka maupun duka.

7. Teman-teman TC2014 yang telah memberikan banyak dukungan moral kepada penulis.
8. Administrator Laboratorium Komputasi Cerdas Visi yang memberikan ruang dan semangat untuk mengerjakan tugas akhir.
9. Teman-teman C1E yang telah menemani selama organisasi, pengaderan dan perkuliahan.
10. Serta pihak-pihak lain yang namanya tidak dapat penulis sebutkan satu per satu.

Bagaimanapun juga penulis telah berusaha sebaik-baiknya dalam menyusun tugas akhir ini, namun penulis mohon maaf apabila terdapat kekurangan yang penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, Juni 2018

Andreas Galang Anugerah

DAFTAR ISI

LEMBAR PENGESAHAN v
ABSTRAK vii
ABSTRACT ix
KATA PENGANTAR xi
DAFTAR ISI xiii
DAFTAR GAMBAR xvii
DAFTAR TABEL xix
DAFTAR KODE SUMBER xxi
BAB I PENDAHULUAN 1
1.1. Latar Belakang 1
1.2. Rumusan Permasalahan 2
1.3. Batasan Masalah 2
1.4. Tujuan 3
1.5. Manfaat 3
1.6. Metodologi 3
1.7. Sistematika Penulisan Laporan Tugas Akhir 5
BAB II DASAR TEORI 7
2.1. Kanker Paru-paru 7
2.2. Data CT Scan Paru-paru 8
2.3. Feature Learning 9
2.4. Convolutional Neural Network 9
2.4.1. Konsep CNN 10
2.4.2. Arsitektur Jaringan 11
2.4.3. Fungsi Aktivasi 12
2.4.4. Metode Optimasi 16
2.5. K-Fold Cross Validation 18
BAB III ANALISIS DAN PERANCANGAN SISTEM 19
3.1. Tahap Analisis 19
3.1.1. Deskripsi Umum 19
3.1.2. Arsitektur Perangkat Lunak 19
3.2. Perancangan Data 21
3.2.1. Data Masukan 22
3.2.2. Data Pembelajaran 23

3.2.3. Data Keluaran 23
3.3. Perancangan Proses 24
3.3.1. *Preprocessing* 25
3.3.2. Tahap Training Data dan Pembuatan Model 25
3.3.3. Tahap Evaluasi 30
BAB IV IMPLEMENTASI 33
4.1. Lingkungan implementasi 33
4.2. Implementasi Proses 33
4.2.1. Implementasi Pemuatan Dataset CT scan 33
4.2.2. Implementasi *Resize* dan Normalisasi Nilai Piksel Citra 35
4.2.3. Implementasi Pembagian Dataset dengan K-Fold 36
4.2.4. Implementasi Menyimpan Dataset dalam Format HDF5 36
4.2.5. Implementasi Training dan Klasifikasi dengan Model CanNet 37
4.2.6. Implementasi Training dan Klasifikasi dengan Model LeNet 38
4.2.7. Implementasi Training dan Klasifikasi dengan Model VGG16 39
4.2.8. Implementasi Training dan Klasifikasi dengan Model VGG19 41
BAB V PENGUJIAN DAN EVALUASI 43
5.1. Lingkungan Pengujian 43
5.2. Data Uji Coba 43
5.3. Uji Coba Performa Arsitektur 45
5.3.1. Arsitektur Jaringan 45
5.3.2. Running Time Program 58
5.3.3. Metode Optimasi 58
5.4. Analisis Hasil Uji Coba 63
BAB VI KESIMPULAN DAN SARAN 65
6.1. Kesimpulan 65
6.2. Saran 65
DAFTAR PUSTAKA 67

A. LAMPIRAN 71

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

Gambar 2.1 Arsitektur CNN 10
Gambar 2.2 Hasil ekstraksi fitur dari convolutional layer [5] 11
Gambar 2.3 Cara kerja *Max Pooling* [13] 12
Gambar 2.4 Distribusi Fungsi Sigmoid 13
Gambar 2.5 Distribusi Fungsi Tanh 14
Gambar 2.6 Distribusi Fungsi ReLu 15
Gambar 2.7 Ilustrasi cross validation menggunakan tiga fold 18
Gambar 3.1 Diagram Alir Desain Sistem Proses Training dan Testing 20
Gambar 3.2 Contoh Data Masukan Seorang Pasien Kanker Malignant 22
Gambar 3.3 Contoh Data Masukan Seorang Pasien Tumor Benign 23
Gambar 3.4 Diagram Alir Desain Sistem Praproses Data 24
Gambar 3.5 Arsitektur CanNet [3] 28
Gambar 3.6 Arsitektur LeNet [16] 29
Gambar 3.7 Arsitektur VGG16 [15] 30
Gambar 3.8 Arsitektur VGG19 [15] 31
Gambar 5.1 Contoh Citra Uji [3] 44
Gambar 5.2 Perbandingan Akurasi Train dan Test Model CanNet pada Fold 1 46
Gambar 5.3 Perbandingan Akurasi Train dan Test Model CanNet pada Fold 2 46
Gambar 5.4 Perbandingan Akurasi Train dan Test Model CanNet pada Fold 3 47
Gambar 5.5 Perbandingan Akurasi Train dan Test Model CanNet pada Fold 4 48
Gambar 5.6 Perbandingan Akurasi Train dan Test Model CanNet pada Fold 5 48
Gambar 5.7 Perbandingan Akurasi Train dan Test Model Lenet pada Fold 1 49

Gambar 5.8 Perbandingan Akurasi Train dan Test Model Lenet pada Fold 2 50
Gambar 5.9 Perbandingan Akurasi Train dan Test Model Lenet pada Fold 3 50
Gambar 5.10 Perbandingan Akurasi Train dan Test Model Lenet pada Fold 4 51
Gambar 5.11 Perbandingan Akurasi Train dan Test Model Lenet pada Fold 5 51
Gambar 5.12 Perbandingan Akurasi Train dan Test Model VGG16 pada Fold 1 52
Gambar 5.13 Perbandingan Akurasi Train dan Test Model VGG16 pada Fold 2 53
Gambar 5.14 Perbandingan Akurasi Train dan Test Model VGG16 pada Fold 3 53
Gambar 5.15 Perbandingan Akurasi Train dan Test Model VGG16 pada Fold 4 54
Gambar 5.16 Perbandingan Akurasi Train dan Test Model VGG16 pada Fold 5 54
Gambar 5.17 Perbandingan Akurasi Train dan Test Model VGG19 pada Fold 1 55
Gambar 5.18 Perbandingan Akurasi Train dan Test Model VGG19 pada Fold 2 56
Gambar 5.19 Perbandingan Akurasi Train dan Test Model VGG19 pada Fold 3 56
Gambar 5.20 Perbandingan Akurasi Train dan Test Model VGG19 pada Fold 4 57
Gambar 5.21 Perbandingan Akurasi Train dan Test Model VGG19 pada Fold 5 57
Gambar 5.22 Grafik Perbandingan Akurasi CanNet terhadap Optimasi 59
Gambar 5.23 Grafik Perbandingan Akurasi LeNet terhadap Optimasi 60
Gambar 5.24 Grafik Perbandingan Akurasi VGG16 terhadap Optimasi 61
Gambar 5.25 Grafik Perbandingan Akurasi VGG19 terhadap Optimasi 62

DAFTAR TABEL

Tabel 5.1 Parameter Pengujian *k-fold cross validation* 45
Tabel 5.2 Hasil Rata-rata *Run Time* Aristektur CNN 58
Tabel 5.3 Hasil Pengujian Metode Optimasi Arsitektur CanNet 59
Tabel 5.4 Hasil Pengujian Metode Optimasi Arsitektur LeNet .. 60
Tabel 5.5 Hasil Pengujian Metode Optimasi Arsitektur VGG16 61
Tabel 5.6 Hasil Pengujian Metode Optimasi Arsitektur VGG19 62
Tabel 5.7 Hasil Pengujian Akurasi Setiap Arsitektur 63
Tabel 5.8 Hasil Pengujian Akurasi Setiap Metode Optimasi 64

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

Kode Sumber 4.1 Kode Program Pemuatan Dataset 35
Kode Sumber 4.2 Kode Program unruk *Resize* dan Normalisasi Nilai Piksel 36
Kode Sumber 4.3 Kode Program Pembagian Dataset dengan K Fold 36
Kode Sumber 4.4 Kode Program untuk Menyimpan Dataset dalam Format HDF5 37
Kode Sumber 4.5 Kode Program Training dan Klasifikasi Model CanNet 38
Kode Sumber 4.6 Kode Program Training dan Klasifikasi Model Lennet 39
Kode Sumber 4.7 Kode Program Training dan Klasifikasi Model VGG16 40
Kode Sumber 4.8 Kode Program Training dan Klasifikasi Model VGG19 42

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini dibahas mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika laporan tugas akhir. Diharapkan dari penjelasan dalam bab ini gambaran tugas akhir secara umum dapat dipahami.

1.1. Latar Belakang

Diagnosa dan pengobatan kanker telah menjadi salah satu tantangan terbesar yang dihadapi umat manusia dalam beberapa dekade terakhir. Deteksi dini kanker akan memudahkan dalam menyelamatkan jutaan nyawa di seluruh dunia setiap tahunnya. Khususnya kanker paru-paru merupakan salah satu jenis kanker yang paling umum terjadi. Pemeriksaan kanker paru-paru dapat dilakukan dengan menggunakan CT scan, ronsen dada, dan sitologi air liur. Namun CT scan bisa mendeteksi tumor kanker paru-paru lebih dini dan pada tahap yang lebih mungkin dapat diobati, dibanding ronsen biasa [1]. Tidak seperti prosedur X-ray tradisional, CT scan menyediakan gambaran yang lebih rinci dan akurat yang menunjukkan hingga abnormalitas atau ketidakteraturan.

Ahli radiologi yang terlatih diperlukan untuk mengidentifikasi kanker paru-paru pada citra CT scan secara akurat. Biaya yang diperlukan relatif tinggi, hal ini menyebabkan masyarakat kelas bawah dan menengah tidak dapat menjangkau biaya yang diperlukan. Secara tidak langsung menyebabkan berkurangnya deteksi tanda-tanda awal kanker paru-paru dan dengan demikian membuat penyembuhan penyakit ini jauh lebih sulit. Oleh karena itu perlu dilakukan automasi dalam mendeteksi kanker paru-paru beserta stadium kanker dalam citra CT scan. Mengembangkan metode dengan bantuan komputer untuk deteksi keganasan kanker yang akurat, dapat mengurangi biaya, sehingga proses perawatan dan pemulihan lebih berhasil. Salah satu pendekatan yang berhasil digunakan adalah menggunakan Jaringan Saraf Tiruan (JST) yang terinspirasi dari jaringan saraf pada manusia.

Konsep tersebut kemudian dikembangkan lebih lanjut dalam Deep Neural Network (DNN).

Metode DNN yang hingga kini memiliki hasil paling signifikan dalam pengenalan citra adalah *Convolutional Neural Network* (CNN). CNN pertama kali diperkenalkan oleh Yann LeCun dan teman teman [2]. CNN memungkinkan mengekstraksi fitur secara hirarki dengan menggunakan banyak *convolution layer* dan *max-pooling*. Namun CNN, seperti metode Deep Learning lainnya, memiliki kelemahan yaitu proses pelatihan model yang lama. Dalam tugas akhir ini, akan diterapkan model CNN dalam mengklasifikasikan tingkat keganasan kanker pada citra CT scan paru-paru. Arsitektur CNN yang digunakan adalah CanNet. Diharapkan model yang dibangun dapat melakukan klasifikasi citra dengan akurasi yang tinggi, serta dengan proses pelatihan model yang cepat.

1.2. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana cara membangun model *Convolutional Neural Network* untuk mengklasifikasikan tingkat keganasan kanker pada citra CT scan paru-paru?
2. Bagaimana mengevaluasi kinerja model *Convolutional Neural Network* yang dibuat?

1.3. Batasan Masalah

Berikut beberapa hal yang menjadi batasan masalah dalam pengerjaan tugas akhir:

1. Data yang digunakan adalah dataset LIDC-IDRI yang tersedia secara terbuka di internet.
2. Model klasifikasi yang digunakan untuk mengklasifikasikan tingkat keganasan kanker pada citra CT scan paru-paru adalah *Convolutional Neural Network.*
3. Implementasi program dilakukan pada lingkungan komputer desktop.

4. Keluaran akhir dari program adalah satu label untuk tiap data citra masukan yang menunjukkan jenis kanker paru-paru apakah bersifat jinak atau ganas.

1.4. Tujuan

Tujuan dari tugas akhir ini adalah untuk membangun model *Convolutional Neural Network* yang dapat melakukan klasifikasi tingkat keganasan kanker pada citra CT scan paru-paru.

1.5. Manfaat

Tugas akhir ini diharapkan mampu membangun sebuah model *Convolutional Neural Network* yang dapat melakukan klasifikasi tingkat keganasan kanker pada citra CT scan paru-paru. Disamping itu, model tersebut diharapkan dapat diaplikasikan pada permasalahan pengenalan kanker pada citra CT scan bagian tubuh manusia yang lain.

1.6. Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.
   Proposal tugas akhir ini berisi tentang deskripsi pendahuluan dari tugas akhir yang akan dibuat. Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah untuk tugas akhir. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir. Subbab metodologi berisi penjelasan mengenai tahapan penyusunan tugas akhir mulai dari penyusunan proposal hingga penyusunan buku tugas akhir. Terdapat pula subbab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir.

2. Studi literatur
   Pada tahap ini dilakukan pencarian literatur berupa jurnal atau paper yang digunakan sebagai referensi untuk pengerjaan tugas akhir ini. Literatur yang dipelajari pada pengerjaan Tugas Akhir

ini berasal dari jurnal ilmiah yang diambil dari berbagai sumber di internet, beserta berbagai literatur online tambahan terkait *Convolutional Neural Network, GPU Computing,* dan Theano. Makalah yang digunakan sebagai acuan adalah "Convolutional Neural Networks for Lung Cancer Screening in Computed Tomography (CT) Scan" [3], "ImageNet Classification with Deep Convolutional Neural Networks" [12], dan "A Deep Learning Approach for Tumor Tissue Image Classification" [13].

3. Perancangan perangkat lunak
   Pada tahap ini akan dilakukan analisis dan desain perancangan model sesuai dengan tujuan yang dijabarkan. Selain itu, pada tahap ini akan dilakukan eksplorasi terkait arsitektur *Convolutional Neural Network*.

4. Implementasi perangkat lunak
   Pada tahap ini akan dilaksanakan implementasi metode dan algoritma yang telah direncanakan. Implementasi perangkat lunak dilakukan di dalam platform desktop dengan menggunakan bahasa pemrograman Python serta librari pendukung Theano dan Keras yang mendukung teknologi CUDA.

5. Pengujian dan evaluasi
   Pada tahap ini dilakukan uji coba dengan menggunakan dataset Lung Image Database Consortium (LIDC-IDRI) beserta evaluasi hasil pengenalan tingkat keganasan kanker pada citra menggunakan metodologi yang diajukan sebelumnya.

6. Penyusunan buku Tugas Akhir.
   Pada tahapan ini disusun buku yang memuat dokumentasi mengenai pembuatan serta hasil dari implementasi perangkat lunak yang telah dibuat.

1.7. Sistematika Penulisan Laporan Tugas Akhir

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini.

1. Bab I Pendahuluan
   Bab ini berisi penjelasan mengenai latar belakang masalah, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu rumusan permasalahan, batasan masalah, dan sistematika penulisan juga merupakan bagian dari bab ini.

2. Bab II Tinjauan Pustaka
   Bab ini berisi penjelasan tentang citra dan penjelasan tentang *Convolutional Neural Network.*

3. Bab III Perancangan Perangkat Lunak
   Bab ini berisi penjelasan mengenai desain, perancangan, bahan, dan pemodelan proses yang digunakan dalam Tugas Akhir ini yang direpresentasikan dengan *pseudocode*.

4. Bab IV Implementasi
   Bab ini merupakan pembangunan aplikasi dengan Python sesuai permasalahan dan batasan yang telah dijabarkan pada Bab I.

5. Bab V Hasil Uji Coba dan Evaluasi
   Bab ini berisi penjelasan mengenai data hasil percobaan, pengukuran, dan pembahasan mengenai hasil percobaan yang telah dilakukan.

6. Bab VI Kesimpulan dan Saran
   Bab ini berupa hasil penelitian yang menjawab permasalahan atau yang berupa konsep, program, dan karya rancangan. Selain itu, pada bab ini diberikan saran-saran yang berisi hal-hal yang masih dapat dikerjakan dengan lebih baik dan dapat dikembangkan lebih lanjut, atau berisi masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir.

[Halaman ini sengaja dikosongkan]

BAB II
DASAR TEORI

Bab ini berisi penjelasan teori-teori yang berkaitan dengan pembuatan aplikasi. Penjelasan ini bertujuan untuk memberikan dasar teori yang mendasari pengembangan perangkat lunak.

2.1. Kanker Paru-paru

Kanker paru-paru berasal dari jaringan tipis paru-paru, pada umumnya berupa lapisan sel yang terletak pada saluran udara. Kanker paru-paru merupakan kanker paling umum kedua yang diidap pria. Pria memiliki resiko kanker paru-paru 3 kali lebih tinggi dari wanita. Kanker paru-paru ini dapat mempengaruhi pasien pada usia berapa pun [22].

Terdapat dua jenis kanker paru-paru primer berdasarkan jenis selnya, yaitu kanker paru-paru sel kecil (*small-cell lung cancer/SCLC)* dan kanker paru-paru non sel kecil (*non—small-cell lung cancer/NSCLC)* [23]. Kanker paru-paru SCLC biasanya hanya menimpa para perokok berat dan penyebarannya lebih cepat dibandingkan dengan kanker paru-paru NSCLC. Lebih dari 80% kanker paru-paru merupakan tipe kanker paru-paru non-sel kecil. Jenis yang kurang umum lain dari kanker paru non-sel kecil yaitu pleomorfik, tumor karsinoid, karsinoma kelenjar ludah, dan karsinoma tak terklasifikasi[22].

Merokok bisa dikatakan sebagai penyebab utama kanker paru-paru. Orang yang paling berisiko terkena kanker paru-paru adalah perokok aktif. Sekitar 85 persen kanker paru-paru dikaitkan dengan kebiasaan merokok. Meski begitu, bukan berarti setiap perokok akan terkena kanker paru-paru[23]. Selain itu, orang yang tidak merokok juga berkemungkinan terserang kanker paru-paru, meski lebih rendah jumlahnya. Selain rokok, beberapa penyebab kanker paru-paru lain adalah menghirup arsenik, radiasi, dan polusi udara. Kanker paru-paru juga lebih umum terjadi pada orang yang sudah lanjut usia.

Untuk mengonfirmasi diagnosis, dokter akan meminta beberapa tes. Tes mungkin hanya berupa tes pencitraan (spiral CT scan, PET scan) untuk melihat paru-paru, atau tes laboratorium yang disebut sitologi dahak untuk mengidentifikasi tumor. Tes pencitraan akan menunjukkan dengan foto apakah terdapat tumor, sementara sitologi dahak akan memeriksa sampel lendir batuk dari paru-paru yang memiliki sel-sel kanker.

2.2. Data CT Scan Paru-paru

CT scan paru-paru merupakan salah satu metode pencitraan yang digunakan untuk mendiagnosis berbagai kelainan pada paru-paru [4]. CT scan atau pemindaian tomografi terkomputerisasi melibatkan berbagai gambar yang diambil dari sudut-sudut yang berbeda, yang kemudian akan dikombinasikan untuk menghasilkan gambaran melintang dan gambaran 3 dimensi dari struktur internal paru-paru. CT scan paru-paru biasanya tergolong kedalam kategori CT scan dada atau toraks.

Prosedur untuk melakukan CT scan paru-paru meliputi penghasilan berbagai gambaran X-ray, yang disebut dengan irisan yang dilakukan di dada atau abdomen bagian atas pasien [8]. Irisan-irisan tersebut kemudian dimasukkan kedalam komputer untuk melihat gambaran akhir yang dapat dilihat dari berbagai sudut, sisi, dan bidang. Tidak seperti prosedur X-ray tradisional, CT scan menyediakan gambaran yang lebih rinci dan akurat yang menunjukkan hingga abnormalitas atau ketidakteraturan yang bersifat minor. Data CT scan memiliki standar dalam format penyimpanannya, yaitu berformat file DICOM.

Digital Imaging and Communication in Medicine (DICOM) merupakan standar yang diciptakan oleh National Electrical Manufacturers Association/NEMA (Asosiasi Produsen Elektrik Nasional) untuk mendukung proses pendistribusian dan proses review gambar medis, seperti CT scan, MRI, dan USG [17]. Sebuah file DICOM berisi sebuah header yang menyimpan data nama pasien, jenis scan, dimensi gambar dan lainnya. Ukuran dari

header ini bervariasi tergantung pada seberapa banyak informasi yang disimpan.

2.3. Feature Learning

Feature Learning merupakan sebuah teknik yang memungkinkan sebuah sistem untuk secara otomatis menemukan representasi yang diperlukan untuk deteksi fitur atau klasifikasi dari data mentah [5]. *Feature learning* didasari konsep bahwa pembelajaran dari sebuah mesin seperti klasifikasi seringkali membutuhkan masukan yang mudah untuk diproses. Diperlukan suatu upaya untuk mengubah data mentah supaya dimengerti oleh mesin, seperti data citra, video, dan sensor. Namun pada proses tersebut memerlukan banyak waktu. Sehingga diperlukan *feature learning* yang mampu secara adaptif menyesuaikan terhadap data yang digunakan.

Model CNN merupakan model yang telah menerapkan *feature learning* dalam pengaplikasiannya. Model CNN terdapat *convolutional layer* yang secara otomatis mengekstrak informasi pada matrix dua dimensi. Oleh sebab itu, CNN tidak memerlukan proses ekstraksi fitur dan hanya memerlukan normalisasi data untuk data inputnya.

2.4. Convolutional Neural Network

Convolutional Neural Network (CNN) sangat mirip dengan *Multilayer Neural Network*, terdiri dari rangkaian neuron yang memiliki bobot dan bias. CNN termasuk dalam jenis *Deep Neural Network* dan banyak diaplikasikan dalam bidang *Deep Learning*, khususnya pada pengolahan data citra. CNN pertama kali diperkenalkan oleh Yann LeCun, yang terinspirasi dari model yang telah dibuat sebelumnya yaitu *neurocognitron*. Model CNN dengan nama LeNet berhasil diterapkan oleh LeCun pada kasus pengenalan angka tulisan tangan [6].

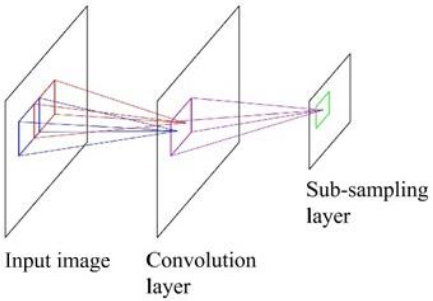
Pada tahun 2012 muncul terobosan yang baru lagi pada *deep learning*. CNN dengan arsitektur tertentu yang dipadukan dengan tambahan berbagai macam parameter seperti *drop out*

regularization, pemanfaatan *Rectified Linear Unit* (ReLU) sebagai fungsi aktivasi, dan data augmentasi mampu mencapai terobosan pada *large scale image classificatio*n (ImageNet) yang memiliki 1000 kategori objek dan kurang lebih 1 juta gambar, melebihi performa manusia. Model ini dikenal dengan nama AlexNet.

2.4.1. Konsep CNN

Konsep dasar CNN adalah membangun sifat invariant ke dalam jaringan syaraf tiruan dengan membuat model yang invariant dengan transformasi input tertentu. Konsep ini berawal dari masalah pada MLP yang sering terjadi. Semua lapisan MLP terhubung satu sama lain. Hal tersebut menghilangkan informasi spatial dari input yang dibutuhkan untuk perhitungan.

Berbeda dengan jaringan syaraf biasa, CNN memiliki arsitektur khusus. Arsitektur CNN biasanya terdiri dari lapisan konvolusi dan lapisan sub-sampling. Lapisan konvolusi menerapkan operasi konvolusi, dan operasi sub-sampling dilakukan di lapisan sub-sampling. Di sini, sub-sampling juga dikenal sebagai *pooling*. CNN dibangun berdasarkan tiga gagasan dasar, yaitu, bidang *receptive fields*, *weight sharing*, dan *pooling*.

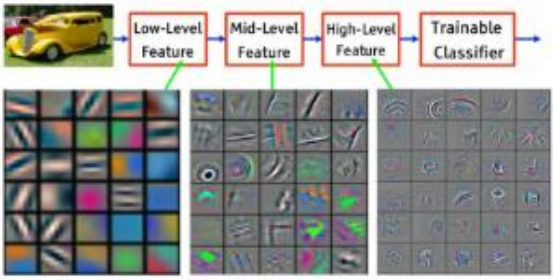


Gambar 2.1 Arsitektur CNN

2.4.2. Arsitektur Jaringan

a. *Convolutional Layer*

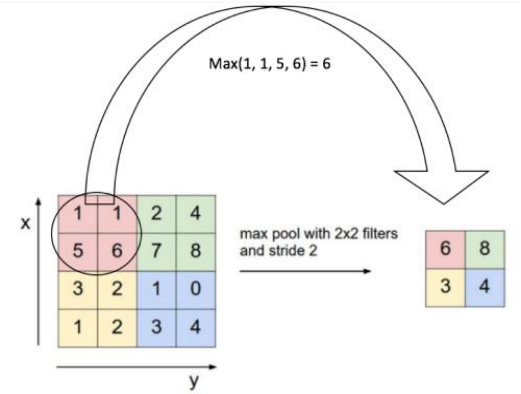
Fungsi utama dari *convolutional layer* adalah mengekstraksi fitur dari layer sebelumnya. *Convolution Layer* awal akan mengesktraksi fitur *low-level* dari citra, dan *Convolution layer* selanjutnya akan mengekstraksi fitur yang level yang lebih tinggi. Bobot pada layer tersebut menspesifikasikan filter konvolusi yang digunakan, sehingga filter yang digunakan dapat dilatih berdasarkan input pada CNN.



Gambar 2.2 Hasil ekstraksi fitur dari convolutional layer [5]

b. *Subsampling Layer*

Subsampling layer biasanya digunakan setelah *convolutional layer*. Fungsi dari *sampling layer* adalah mereduksi ukuran dari data. Terdapat beberapa tipe *subsampling layer* diantaranya yaitu *max, average, sum* dan lainnya. Metode sampling dalam CNN yang biasa digunakan adalah *Max-Pooling*. *Max Pooling* membagi output dari *convolution layer* menjadi beberapa matrix kecil lalu mengambil nilai maksimal dari tiap matrix untuk menyusun matriks citra yang telah direduksi. Proses tersebut memastikan fitur yang didapatkan akan sama meskipun obyek citra mengalami translasi. Cara kerja *Max Pooling* dapat ditunjukkan dengan gambar 2.3.



Gambar 2.3 Cara kerja *Max Pooling* [13]

c. *Fully Connected Layer*

Fully connected layer merupakan layer terakhir dari CNN dan juga merupakan tradisional *Multi Layer Perceptron* yang menggunakan fungsi aktivasi *softmax*. Layer tersebut adalah layer yang biasanya digunakan dalam penerapan jaringan syaraf tiruan pada umumnya. Layer tersebut menghubungkan setiap neuron lapisan sebelumnya ke setiap neuron di lapisan berikutnya. Pada layer ini menghubungkan keluaran dari *convolutional layer* dan *subsampling layer* dengan kelas dari citra masukan.

Arsitektur umum CNN terdiri dari *convolution layer* dan *subsampling layer* yang disusun secara bergantian dan *fully connected layer* pada layer terakhir.

2.4.3. Fungsi Aktivasi

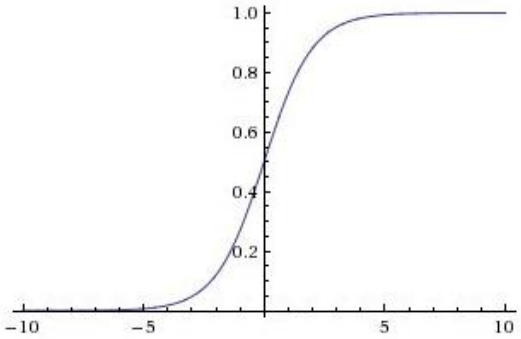
a. Fungsi Sigmoid

Fungsi sigmoid akan menerima angka tunggal dan mengubah nilai x menjadi sebuah nilai yang memiliki range mulai dari 0 sampai 1 [19]. Fungsi sigmoid menghasilkan kurva yang akan berada dalam bentuk S.

Fungsi sigmoid ditunjukkan dengan rumus seperti yang didefiniskan pada persamaan 2.1.

$$f(x) = \frac{1}{1 + e^{-x}} \quad \textbf{(2.1)}$$

Fungsi sigmoid pada saat ini sudah jarang digunakan. Hal tersebut terjadi karena jika aktivasi pada neuron mengeluarkan nilai hampir mendekati nilai 0 atau 1, maka neuron tersebut tidak akan mengalami update atau menjadi tidak aktif.

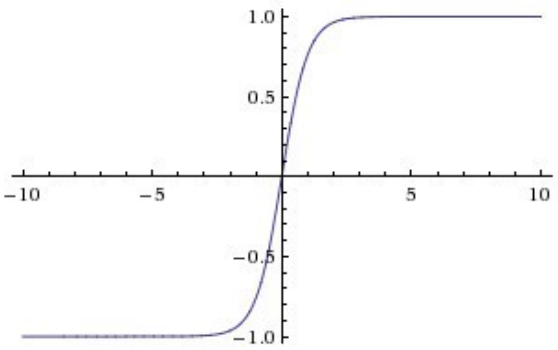


Gambar 2.4 Distribusi Fungsi Sigmoid

b. Fungsi Tangen Hiperbolik (tanh)

Fungsi tanh akan mengubah nilai input x nya menjadi sebuah nilai yang memiliki range mulai dari -1 sampai 1 [19]. Seperti halnya fungsi sigmoid, fungsi tanh memiliki kekurangan yaitu bisa menonaktifkan neuron, tetapi kelebihannya adalah output yang dimiliki fungsi tanh terpusat pada nol. Dalam praktiknya fungsi tanh lebih menjadi pilihan jika dibandingkan dengan fungsi sigmoid. Fungsi tanh ditunjukkan dengan rumus seperti yang didefiniskan pada persamaan 2.2.

$$f(x) = \tanh(x) \quad \textbf{(2.2)}$$



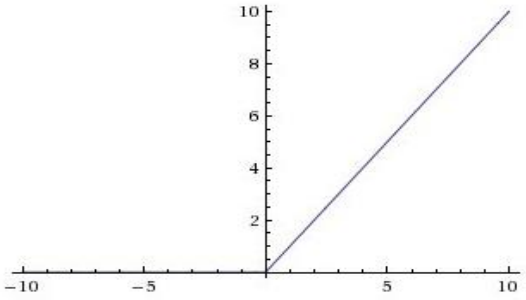
Gambar 2.5 Distribusi Fungsi Tanh

c. Fungsi Rectified Linear Unit (ReLU)

Fungsi ReLU merupakan fungsi yang saat ini paling populer digunakan. ReLU pada intinya hanya membuat pembatas pada bilangan nol, artinya apabila x merupakan bilangan negatif maka akan dirubah menjadi angka 0.

$$f(x) = \max(0, x) \qquad \textbf{(2.3)}$$

Untuk pertama kalinya pada tahun 2011, penggunaan ReLU sebagai non-linearitas telah menunjukkan kemungkinan pelatihan CNN tanpa memerlukan *unsupervised pre-training*. ReLU dibandingkan dengan fungsi sigmoid atau fungsi aktivasi serupa, memungkinkan pelatihan yang lebih cepat dan efektif untuk arsitektur syaraf pada kumpulan data yang besar dan kompleks. Sayangnya, unit ReLU bisa menjadi rapuh pada saat proses training dan bisa membuat unit tersebut tidak aktif. Namun *learning rate* diinisialisai secara tepat maka hal seperti ini jarang menjadi masalah.



Gambar 2.6 Distribusi Fungsi ReLu

d. Softmax

Fungsi *softmax* menghitung probabilitas distribusi pada n kejadian. Secara umum, fungsi ini akan menghitung probabilitas masing-masing kelas target di atas semua kelas sasaran yang mungkin ada [20]. Kemudian probabilitas yang dihitung akan sangat membantu untuk menentukan kelas target untuk input yang diberikan. Keuntungan utama menggunakan *softmax* adalah rentang probabilitas output. Rentang akan 0 sampai 1, dan jumlah semua probabilitas akan sama dengan satu. Jika fungsi *softmax* digunakan untuk model multi-klasifikasi, maka akan mengembalikan probabilitas masing-masing kelas dan kelas target akan memiliki probabilitas tinggi.

$$f(x) = \frac{e^{Xi}}{\sum_{j=0}^{k} e^{Xj}} \qquad \textbf{(2.4)}$$

Rumusnya menghitung eksponensial dari nilai masukan yang diberikan dan jumlah nilai eksponensial dari semua nilai pada input. Kemudian rasio eksponensial nilai input dan jumlah nilai eksponensial adalah output dari fungsi *softmax*. Fungsi s*oftmax* pada CNN biasanya digunakan pada *fully connected layer* untuk klasifikasi.

2.4.4. Metode Optimasi

a. *Stochastic Gradient Descent* (SGD)

Gradient descent merupakan salah satu algoritma optimasi yang popular untuk neural network. *Gradient descent* bertujuan untuk menemukan fungsi minimum lokal dengan menggunakan kemiringan *gradien negative* [21]. *Stochastic Gradient Descent* (SGD) sering disebut juga *incremental gradient descent* merupakan optimasi dari *gradient descent*. Meskipun SGD telah lama berada di dalam *machine learning* sejak lama, namun baru-baru ini telah mendapat banyak perhatian dalam konteks pembelajaran berskala besar.

$$Q(w) = \frac{1}{n}\sum_{i=1}^{n} Q_i(w) \qquad \textbf{(2.5)}$$

Keterangan:

Q(w): Fungsi objektif dengan parameter w

$Q_i(w)$: Nilai evaluasi pada fungsi objektif Q(w), iterasi ke-i

n : jumlah iterasi

b. Adagrad

AdaGrad memberikan kecepatan update yang berbeda pada tiap dimensi vektor parameter dan kemudian mampu beradaptasi berdasarkan indikator tertentu [21]. Salah satu indikator yang dapat dipakai adalah besarnya perubahan nilai vektor gradient pada dimensi tertentu.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t \qquad \textbf{(2.6)}$$

Keterangan:

η : merupakan *learning rate*

g_t : merupakan gradien

G_t : merupakan matrik diagonal dimana setiap elemennya merupakan jumlah dari g_t dikuadratkan

ϵ : merupakan penambahan untuk menghindari pembagian dengan nol dimana $\epsilon = 1e - 8$

c. RMSprop

Adagrad memiliki memiliki sebuah problem, yaitu pada waktu tertentu nilai matriks yang dihasilkan berpotensi sangat besar sehingga akan memperlambat proses optimasi [21]. Tieleman dan Hinton melakukan sedikit modifikasi terhadap Adagrad dengan menambahkan konstanta. Modifikasi ini menghasilkan nama algoritma baru yang disebut dengan RMSprop.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \odot g_t \quad \textbf{(2.7)}$$

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2 \quad \textbf{(2.8)}$$

Keterangan:

η : merupakan *learning rate*

g_t : merupakan gradien

E : rata-rata

ϵ : merupakan penambahan untuk menghindari pembagian dengan nol dimana $\epsilon = 1e - 8$

d. *Adaptive Moment Estimation* (Adam)

Adam adalah algoritma optimasi yang dapat digunakan sebagai pengganti *prosedur stochastic gradient descent* klasik untuk memperbarui bobot jaringan yang berbasis pada data pelatihan.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t \quad \textbf{(2.9)}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad \textbf{(2.10)}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad \textbf{(2.11)}$$

Keterangan:

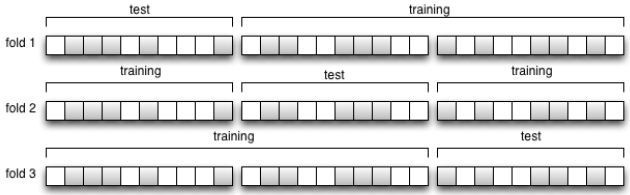
η : merupakan *learning rate*

m_t : nilai *decay* rata-rata dari iterasi sebelumnya

v_t : nilai *decay* rata-rata yang dikuadratkan dari iterasi sebelumnya
β : nilai *decay*
ϵ : merupakan penambahan untuk menghindari pembagian dengan nol dimana $\epsilon = 1e - 8$

2.5. K-Fold Cross Validation

Adalah salah satu teknik validasi model terhadap set data independen. Secara umum caranya dengan membagi sebuah *dataset* menjadi K *sub-dataset*. Dari semua sub akan diambil satu sub sebagai *testing data*, dan sisanya menjadi *training data*, begitu seterusnya sampai sub ke K menjadi *testing data*. Lalu performa dihitung dari rata-rata hasil *cross validation*.



Sumber: Wikipedia.org

Gambar 2.7 Ilustrasi cross validation menggunakan tiga fold

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini dijelaskan mengenai rancangan sistem perangkat lunak yang akan dibuat. Perancangan yang dijelaskan meliputi data dan proses. Data yang dimaksud adalah data yang akan diolah dalam perangkat lunak baik digunakan sebagai pembelajaran maupun pengujian sehingga tujuan tugas akhir ini bisa tercapai.

3.1. Tahap Analisis

Tahap analisis mendefinisikan kebutuhan yang akan dipenuhi dalam pembangunan aplikasi dalam mengklasifikasikan tingkat keganasan kanker.

3.1.1. Deskripsi Umum

Pada tugas akhir ini akan dibangun sebuah sistem untuk melakukan pendeteksian tingkat keganasan kanker paru-paru. Proses-proses yang terlibat di dalam implementasi sistem ini meliputi tahap praproses data lalu dilanjutkan dengan pembuatan model klasifikasi menggunakan *Convolutional Neural Network*.

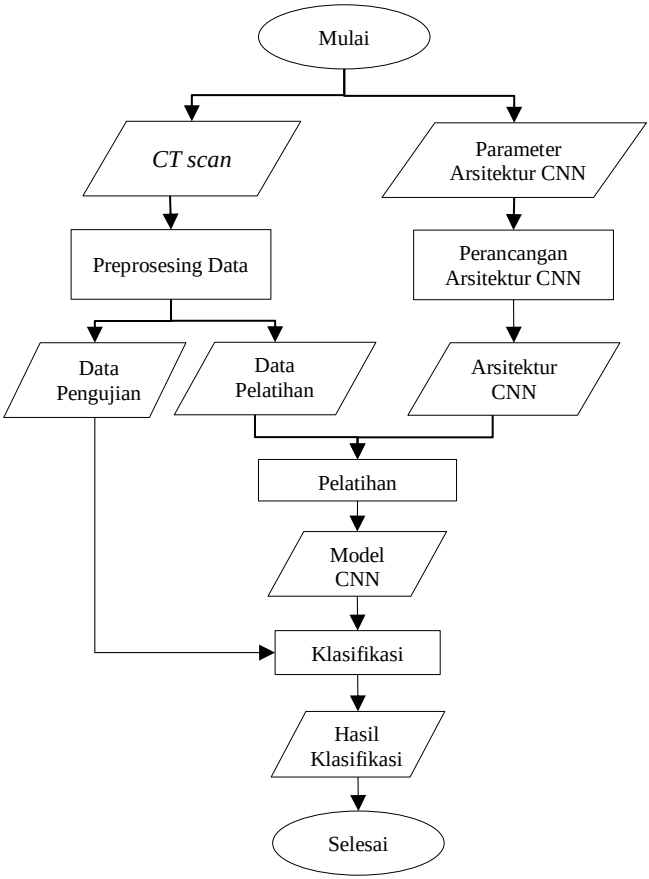
3.1.2. Arsitektur Perangkat Lunak

Seluruh proses yang terdapat pada klasifikasi tingkat keganasan kanker dijalankan pada komputer. Sistem menerima data masukan berupa citra *CT scan* paru-paru yang telah di normalisasi skala intensitas citra. Program akan dibangun menggunakan bahasa pemrograman python menggunakan *library* bernama Theano. Theano dikembangkan dengan berfokus pada *deep learning* sehingga memiliki banyak fungsi untuk menunjang pembuatan model CNN.

Dalam tugas akhir ini, akan dibuat dua program terpisah, yaitu program untuk praproses data dan program untuk klasifikasi. Program praproses akan menerima input data citra *CT scan*, lalu melakukan pengolahan data hingga siap untuk digunakan oleh

program klasifikasi. Program klasifikasi akan mengolah data untuk membangun sebuah model klasifikasi.

Program praproses akan menerima input data citra *CT scan*, lalu membaginya menjadi data pengujian dan pelatihan. Kemudian akan dilakukan transformasi citra agar citra lebih cocok untuk pemakaian dalam proses pelatihan. Transformasi citra dilakukan dengan cara me-*resize* citra dan normalisasi nilai pixel citra.



Gambar 3.1 Diagram Alir Desain Sistem Proses Training dan Testing

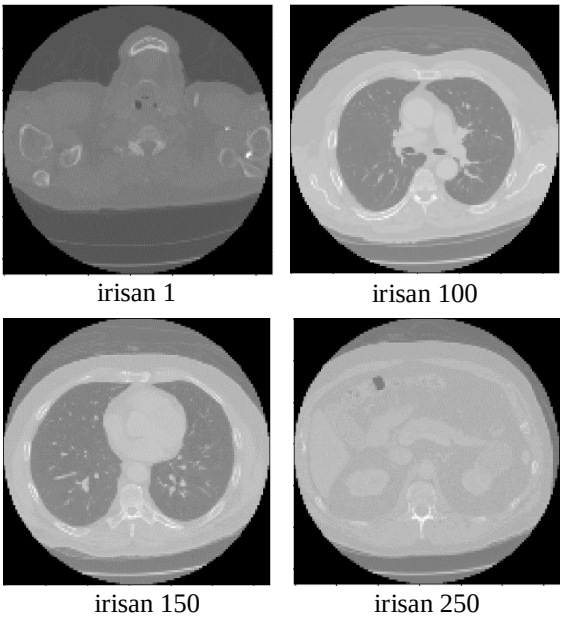
Program klasifikasi akan menyusun sebuah model untuk melakukan klasifikasi citra. Ada dua proses besar dalam program tersebut, yaitu pembangunan model CNN dan fitting model. Pada proses pembangunan model CNN, akan menerima sejumlah parameter yang mendefinisikan jaringan CNN, kemudain akan dibangun sebuah model CCN sesuai input data. Model awal akan diolah dalam proses *fitting* data dimana parameter pada jaringan (bobot dan bias) akan dikonfigurasi sesuai dataset melalui proses pelatihan. Proses pelatihan akan menghasilkan model CNN yang siap digunakan untuk proses klasifikasi. Diagram alir desain sistem untuk praproses data dapat ditunjukkan dengan Gambar 3.1.

3.2. Perancangan Data

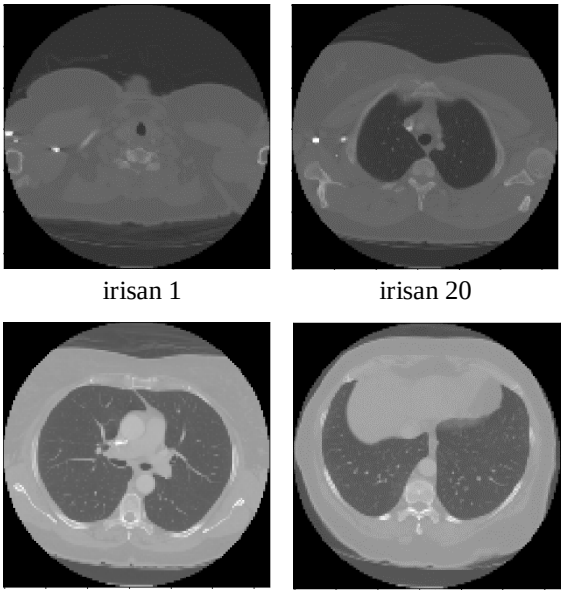
Perancangan data dilakukan untuk memastikan pengoperasian aplikasi berjalan dengan benar. Dataset yang digunakan pada tugas akhir ini didapatkan dari yaitu *Lung Image Database Consortium* (LIDC-IDRI). LIDC-IDRI adalah sebuah dataset citra CT scan yang dipublikasikan secara umum di internet. Diprakarsai oleh National Cancer Institute (NCI), yang kemudian dikembangkan oleh Foundation for National Institutes of Health (FNIH), dan didampingi oleh Food and Drug Administration (FDA). Dataset berisi hasil pemindaian dari 1018 pasien. Setiap citra CT scan pasien terdiri dari sekitar 150 sampai 550 citra yang berformat dicom yang disebut dengan irisan yang dilakukan di dada atau abdomen bagian atas pasien. Dataset menyediakan empat klasifikasi yaitu *Unknown, Benign* (jinak), *Malignant* (ganas), dan *Metastatic*. Pada tugas akhir ini hanya akan menggunakan kelas *benign* dan *malignant* sebagai label keganasan kanker. Setelah memisahkan kelas secara keseluruhan, didapatkan total 70 data pasien. Masing-masing kelas memiliki komposisi dengan jumlah 35 data pasien untuk kelas *benign* dan 35 data pasien untuk kelas *malignant*.

3.2.1. Data Masukan

Data masukan merupakan data yang akan diolah menggunakan aplikasi klasifikasi tingkat keganasan kanker paru-paru. Data yang digunakan sebagai masukan adalah citra *screening thoracic computed tomography (CT) scan* dari pasien. Dari citra *CT scan* diambil diagnosa pasien yang memiliki penyakit kanker paru-paru *benign* dan *malignant*. Selanjutnya masing-masing citra *grayscale* digabungkan secara linier untuk membuat *array 3D* dan mengubah format citra menjadi HDF5. *Array 3D* ini kemudian ditambahkan *zero padded* untuk menyamakan ukuran *array 3D*. Satu data pasien yang berformat HDF5 akan akan dirubah lagi menjadi *array* berukuran 1 x 550 x 128 x 128. HDF5 berisi dua masukan, yaitu data dan label kelas. Contoh data masukan seorang pasien dapat ditunjukkan dengan Gambar 3.2 dan Gambar 3.3.



Gambar 3.2 Contoh Data Masukan Seorang Pasien Kanker Malignant



irisan 1 irisan 20

irisan 50 irisan 90

Gambar 3.3 Contoh Data Masukan Seorang Pasien Tumor Benign

3.2.2. Data Pembelajaran

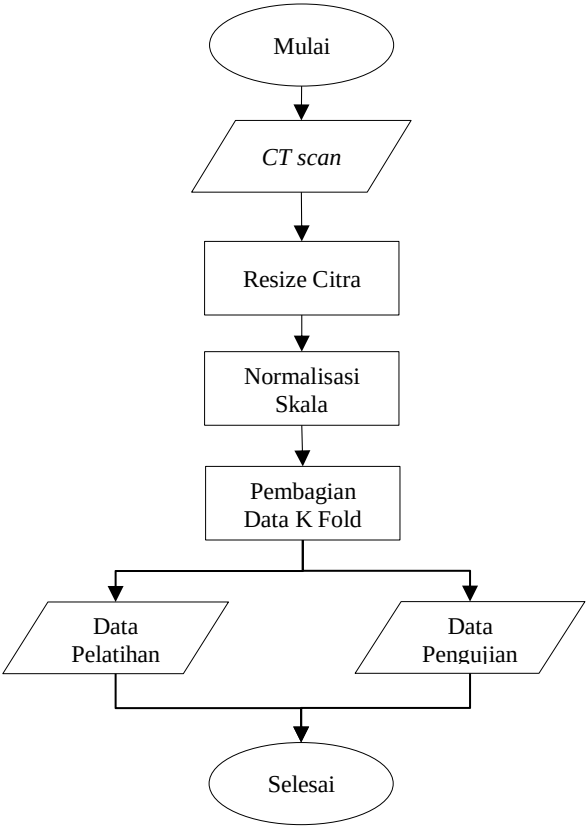
Data pembelajaran digunakan pada proses *metric learning*. Data yang digunakan adalah citra CT scan pasien yang dipilih secara acak dari kelas *benign* dan kelas *malignant*. Proporsi jumlah data diatur 80% untuk pembelajaran, dan 20% sisanya untuk data test atau dengan kata lain data berbanding 4:1.

3.2.3. Data Keluaran

Data keluaran dari sistem ini berupa hasil klasifikasi yang menentukan apakah CT scan dari pasien tersebut menderita penyakit kanker paru dengan nilai 0 sebagai tumor *benign* dan nilai 1 sebagai kanker *malignant*.

3.3. Perancangan Proses

Perancangan proses dilakukan untuk memberikan gambaran mengenai setiap proses yang terdapat pada aplikasi klasifikasi tingkat keganasan kanker paru-paru.



Gambar 3.4 Diagram Alir Desain Sistem Praproses Data

3.3.1. *Preprocessing*

Dalam mengawali proses pengerjaan program, maka harus disiapkan terlebih dahulu data yang akan digunakan dalam proses pengerjaan. *Preprocessing* adalah tahapan yang penting untuk dilakukan. Pada tugas akhir ini tahapan *preprosessing* yang dilakukan adalah *resize* citra dan normalisasi skala. Data mentah *CT scan* memiliki dimensi sebesar 512x512, kemudian akan diresize menjadi ukuran 128x128. Hal ini bertujuan supaya memori yang digunakan komputer saat proses training data tidak terlalu besar. Nilai piksel *CT scan* memiliki nilai diantara -2000 sampai dengan 2000, kemudian akan dinormalisasi menjadi 0 sampai 1. Tahap akhir dari praproses data yaitu menyimpan hasil praproses dalam format HDF5. Diagram alir desain sistem untuk praproses data dapat ditunjukkan dengan Gambar 3.4.

3.3.2. Tahap Training Data dan Pembuatan Model

Pada CNN, terdapat dua bagian utama dari arsitektur model, yaitu bagian ekstraksi fitur dan bagian klasifikasi. Bagian ekstraksi fitur berfungsi untuk mentransformasi data input sehingga dapat diklasifikasikan dengan baik. Bagian ekstraksi fitur berfungsi mereduksi data sehingga dapat dimasukkan ke dalam *softmax classifier* untuk dilakukan klasifikasi. Setelah proses pelatihan selesai, akan dihasilkan model klasifikasi yang siap digunakan dalam proses pengujian. Arsitektur CanNet akan dibandingkan performanya dengan arsitektur lainnya, yaitu LeNet, VGG16, dan VGG19.

3.3.2.1. Network Layer

Pada bagian ini akan dijelaskan secara mendalam terkait proses yang dilakukan pada masing-masing layer.

a. *Convolutional Layer*

   Convolutional Layer mengolah data dua dimensi, input dari layer tersebut berukuran 4 dimensi, yaitu [jumlah_data, jumlah_kernel_output, tinggi_data, lebar_data]. Layer tersebut

memiliki parameter bobot dan bias. Hyperparameter yang digunakan dalam konvolusi adalah ukuran *padding*.

b. *Max Pooling Layer*
   Citra akan dibagi menjadi beberapa *pool*, lalu akan dikalkulasi satu nilai dari tiap *pool* untuk menghasilkan data output. *Max pooling layer* mengambil nilai terbesar dari setiap *pool*. Hyperparameter pada *max pooling layer* yaitu ukuran *pool* (pool_size), serta jarak antar pool (pool_stride). Pooling dilakukan pada setiap *feature maps* pada data.
c. *Fully Connected Layer*
   Fully Connected Layer melakukan operasi perkalian matrix untuk mentransformasi data secara linier. Hyperparameter pada layer tersebut yaitu jumlah kernel output dan jumlah kernel intput.
d. *Dropout Layer*
   Dropout layer merupakan layer yang berfungsi melakukan regularisasi pada CNN dengan mematikan neuron secara acak dengan kemungkinan *drop rate* pada setiap iterasi pelatihan.

3.3.2.2. Perancangan Model CanNet

Pada tugas akhir ini arsitektur CNN yang akan digunakan adalah arsitektur CanNet. CanNet terdiri dari dua *convolution layer*, sebuah *pooling layer*, sebuah *dropout layer* dan sebuah *fully connected layer*. Setiap *convolutional layer* diikuti *Rectified linier output layer* (ReLu) dan *pooling layer* diikuti sebuah *dropout layer*. Arsitektur CanNet dapat ditunjukkan dengan Gambar 3.5.

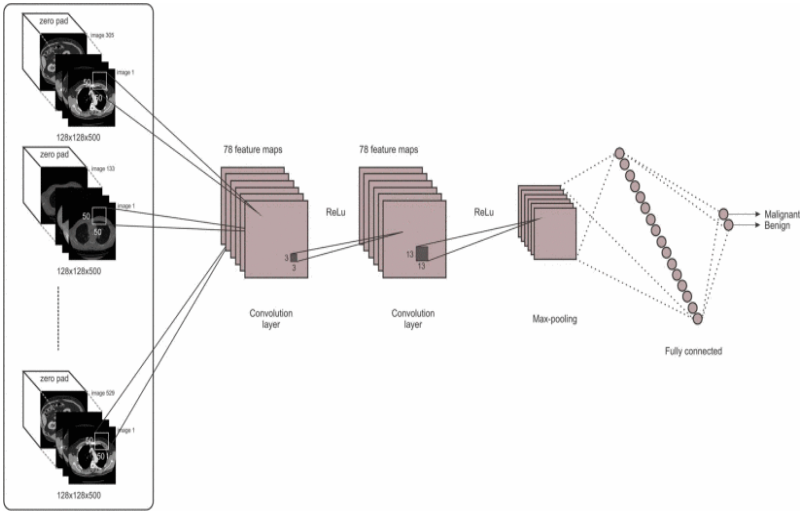
a. *Convolutional layer 1*
   Lapisan ini memiliki ukuran kernel 50x50. Output dari lapisan ini menghasilkan 78 fitur. *Weight filler* diatur ke perubahan distribusi *Gaussian* sebesar 0,01 dan bias ditetapkan pada nilai 0 konstan. Hasil keluaran ini kemudian diumpankan ke lapisan *Rectified Linear* (ReLu) untuk membawa semua aktivasi negatif ke nilai nol. Fungsi lapisan ini adalah untuk mendeteksi fitur tingkat terendah.

b. *Convolutional layer 2*
Hasil dari *concolutional layer 1* dimasukkan ke dalam *convolutional layer 2* yang memiliki ukuran kernel 3x3. Pengaturan *weight filler* sama dengan *convolutional layer 1* dan bias ditetapkan pada nilai 1 konstan. Lapisan ini juga diikuti oleh lapisan ReLu. Fungsi lapisan ini adalah mendeteksi pola seperti *popcorn, diffuse* dan lainnya.

c. *Max-pooling layer*
Ukuran kernel yang digunakan pada lapisan ini adalah 13x13. Fungsi lapisan ini untuk mengurangi jumlah komputasi, karena setiap hasil citra CT *scan* terdiri dari hingga 550 citra sehingga jumlah perhitungan yang dibutuhkan sangat besar. Lapisan ini bermanfaat untuk mengurangi jumlah memori yang digunakan.

d. *Dropout layer*
Lapisan ini digunakan untuk mencegah terjadi *over-fitting*, dengan cara mematikan neuron acak pada jaringan. Arsitektur CanNet meggunakan *dropout layer* dengan rasio drop sebesar 0,5. Fungsi dari lapisan ini untuk meningkatkan kualitas klasifikasi pada data uji.

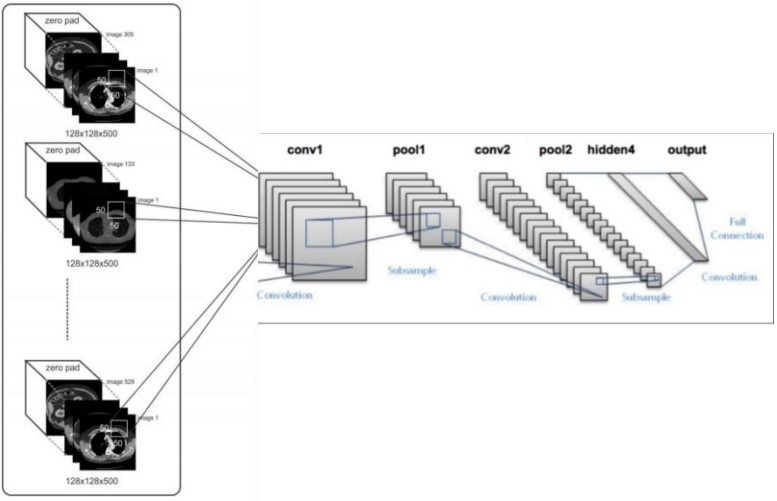
e. *Fully connected layer*
Pada lapisan ini menghasilkan hasil dari klasifikasi. *Fully connected layer* menggunakan *gaussian weight filler* sebesar 0,5 dan bias ditetapkan pada nilai 0 konstan. Keluaran lapisan ini menghasilkan dua *neuron* yang memberikan klasifikasi *benign* atau *malignant*.



Gambar 3.5 Arsitektur CanNet [3]

3.3.2.3. Perancangan Model LeNet

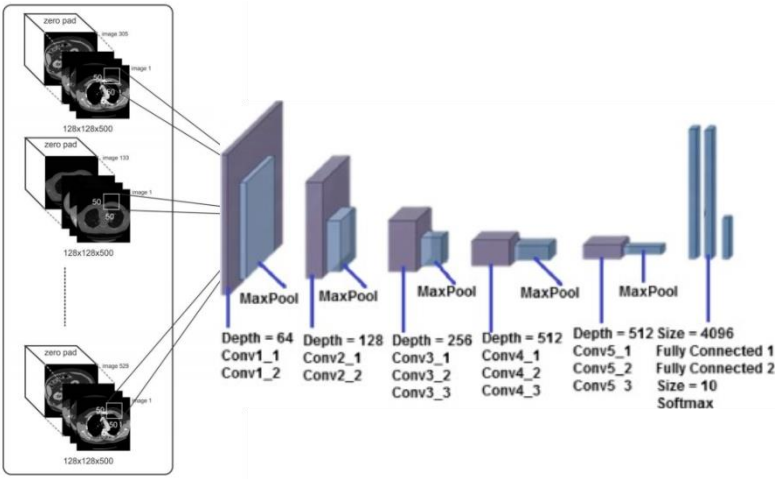
Arsitektur LeNet pertama kali diperkenalkan oleh LeCun pada tahun 1998. Awal mula LeNet digunakan untuk memecahkan masalah pengenalan tulisan tangan. Arsitektur ini memiliki ini terdiri dari dua *convolution layer*, dua *max pooling layer*, dan dua *fully connected layer*. Setiap *convolution layer* diikuti oleh *max pooling layer*. Setiap *convolution layer* memiliki kernel 5x5. *Convolution layer* pertama menghasilkan 20 fitur sedangkan *convolution layer* kedua menghasilkan 50 fitur. Ukuran kernel yang digunakan pada *max pooling layer* adalah 2x2. Arsitektur LeNet dapat ditunjukkan dengan Gambar 3.6.



Gambar 3.6 Arsitektur LeNet [16]

3.3.2.4. Perancangan Model VGG16

Arsitektur VGG16 merupakan singakatan dari Visual Geometry Group of Oxford University in the United Kingdom. Arsitektur VGG merupakan usulan dari Simonyan dan Zisserman dan memperoleh 92.7% akurasi peringkat 5 teratas dalam kompetisi *ImageNet*. VGG16 memiliki total 16 layer terdiri dari 13 *convolution layer* dan 3 *fully connected layer*. Semua *convolution layer* memiliki ukuran kernel 3x3 dan menggunakan fungsi aktivasi ReLu. Pada bagian pertama dan kedua terdapat dua *convolution layer,* masing-masing menghasilkan 64 dan 128 fitur map. Bagian ketiga, keempat dan kelima terdapat tiga *convolution layer*, masing-masing menghasilkan 256, 512, dan 512 fitur map. Pada bagian terakhir terdapat tiga *fully connected layer*. Ukuran kernel yang digunakan pada *max pooling layer* adalah 2x2. Arsitektur LeNet dapat ditunjukkan dengan Gambar 3.7.



Gambar 3.7 Arsitektur VGG16 [15]

3.3.2.5. Perancangan Model VGG19

Arsitektur VGG19 merupakan modifikasi dari arsitektur VGG16. Arsitektur ini memiliki total 19 layer terdiri dari 16 *convolution layer* dan 3 *fully connected layer*. Perbedaan arsitektur ini dengan VGG16 hanyalah terletak pada bagian ketiga, keempat dan kelima. Masing-masing bagian tersebut hanya ditambahkan dengan sebuah *convolution layer* dengan konfigrasi yang sama pada setiap bagian. Arsitektur LeNet dapat ditunjukkan dengan Gambar 3.8.

3.3.3. Tahap Evaluasi

Pada tahap evaluasi, masing-masing arsitektur akan diuji menggunakan *cross validation* dengan k bernilai 5. Lalu performa dihitung dari rata-rata hasil *cross validation*. Pada proses pengujiam akan dilakukan pengecekan keputusan algoritma terhadap pasangan gambar, kemudian hasilnya akan dicatat pada laporan evaluasi. Perhitungan hasil hanya dilakukan pada akurasi. Metode optimasi yang akan digunakan yaitu *SGD, AdaGrad,*

RMSprop, dan *Adam.* Masing-masing metode optimasi juga akan dihitung performanya dengan akurasi. Akurasi memilki persamaan.

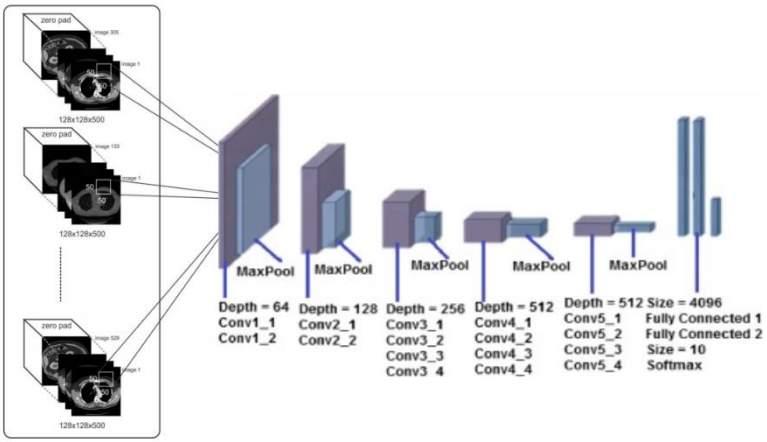
$$Akurasi = \frac{tp + tn}{N} \qquad \textbf{(3.1)}$$

Keterangan:

$\boldsymbol{tp}$: true positive, label yang benar dan dideteksi benar

$\boldsymbol{tn}$: true negative, label yang salah namun dideteksi benar

$\boldsymbol{N}$: Jumlah semua data



Gambar 3.8 Arsitektur VGG19 [15]

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Bab ini membahas implementasi dari perancangan sistem sesuai dengan perancangan yang telah dibuat. Bahasa pemrograman yang digunakan untuk implementasi sistem adalah bahasa pemrograman Python.

4.1. Lingkungan implementasi

Lingkungan implementasi sistem yang digunakan untuk mengembangkan tugas akhir memiliki spesifikasi perangkat keras dan perangkat lunak seperti ditampilkan pada tabel.

Perangkat	Spesifikasi
Perangkat keras	Prosesor: Intel® Core™ i7-7700 CPU @ 3.60GHz (8 CPUs) , ~3.6GHz Memori: 16384MB
Perangkat lunak	Sistem Operasi: Microsoft Windows 10 64-bit Perangkat Pengembang: Python Perangkat Pembantu: Spyder, Microsoft Word, Microsoft Excel, Microsoft PowerPoint, Snipping Tool

4.2. Implementasi Proses

Implementasi proses dilakukan berdasarkan perancangan proses yang dijelaskan pada bab analisis dan perancangan.

4.2.1. Implementasi Pemuatan Dataset CT scan

Implementasi pemuatan dataset dimulai dengan memuat seluruh dataset yang memiliki kategori CT scan. Selanjutnya dataset akan dipilih berdasarkan kelas yang hanya memiliki kelas *benign* dan

malignant. Implementasi fungsi ini dapat ditunjukkan pada Kode Sumber 4.1.

```
1.  def path_():
2.      data_ =
                pd.read_excel("C:/Users/incr/Documents/c
                ode/data/tcia-diagnosis-data-2012-04-
                20.xls")
3.
4.      data_copy = data_
5.      data_copy.columns =
                ['1','2','3','4','5','6','7','8','9','10
                ','11','12','13','14']
6.
7.      list_drop=[]
8.      for i in range(len(data_)):
9.          if data_.iloc[i][1] == 3 or
                 data_.iloc[i][1] == 0:
10.                list_drop.append(data_.index[i])
11.         elif data_.iloc[i][4] == 3 or
                 data_.iloc[i][4] == 0:
12.                list_drop.append(data_.index[i])
13.         elif data_.iloc[i][6] == 3 or
                 data_.iloc[i][6] == 0:
14.                list_drop.append(data_.index[i])
15.
16.     data_=data_.drop(list_drop,axis=0)
17.
18.     for i in data_.index:
19.         data_.loc[i,'2']=data_.loc[i,'2']-1
20.
21.     path_data = data_['1']
22.     class_ = data_['2']
23.     return path_data, class_
24.
25. path_data, class_ = path_()
26. path_data= path_data.as_matrix()
27. class_ = class_.as_matrix()
28.
29. def path_full(path):
30.     count =0
```

```
31.     patient_filename ={}
32.     for i in path:
33.         PathDicom = "D:/dataset/DOI/"+i+"/"
34.         lstFilesDCM = []
35.         for dirName, subdirList, fileList in
                os.walk(PathDicom):
36.             for filename in fileList:
37.                 if ".dcm" in filename.lower():  #
                        check whether the file's DICOM
38.
                    lstFilesDCM.append(os.path.join
                    (dirName,filename))
39.                  count +=1
40.         patient_filename[i] = lstFilesDCM
41.     del fileList, filename, lstFilesDCM, dirName,
         PathDicom
42.     return patient_filename
```

Kode Sumber 4.1 Kode Program Pemuatan Dataset

4.2.2. Implementasi *Resize* dan Normalisasi Nilai Piksel Citra

Implementasi *resize* dan normalisasi nilai piksel terlebih dulu dilakukan sebelum memasuki proses training data. Fungsi ini bertujuan bertujuan supaya memori yang digunakan komputer saat proses training data tidak terlalu besar. Citra akan *diresize* menjadi ukuran 128x18. Selanjutnya nilai pikesl akan dinormalisasi menjadi skala 0 sampai 1. *Implementasi* fungsi ini dapat ditunjukkan pada Kode Sumber 4.2.

```
1.  def norm_piksel(dictionary, path):
2.      x_arr = np.zeros([len(dictionary), 545, 128,
          128],dtype="float32")
3.      for i in range(len(path)):
4.          count = 0
5.          for j in dictionary[path[i]]:
6.              RefDs = dicom.read_file(j)
7.              if RefDs.Modality=="CT":
8.                  x_arr[i,RefDs.InstanceNumber,:,:]
                     =resize(RefDs.pixel_array,(128,
```

```
                        128),order=1,
                        preserve_range=False)
                    count +=1
    return x_arr

full_path=path_full(path_data)
class_cat = keras.utils.to_categorical(class_, 2)
image_data=load_image(full_path, path_data)
```

Kode Sumber 4.2 Kode Program unruk *Resize* dan Normalisasi Nilai Piksel

4.2.3. Implementasi Pembagian Dataset dengan K-Fold

Implementasi pembagian dataset dengan k-fold merupakan scenario uji coba pada tugas akhir ini. Implementasi fungsi ini dapat ditunjukkan pada Kode Sumber 4.3.

```
from sklearn.model_selection import
StratifiedKFold

skf = StratifiedKFold(n_splits=5)
skf.get_n_splits(path_data, class_)
image_data = (image_data-
image_data.min())/float(image_data.max()-
image_data.min())
```

Kode Sumber 4.3 Kode Program Pembagian Dataset dengan K Fold

4.2.4. Implementasi Menyimpan Dataset dalam Format HDF5

Implementasi menyimpan dataset dalam format HDF5 dilakukan dengan tujuan untuk menyimpan dataset yang telah di preproses. Implementasi fungsi ini dapat ditunjukkan pada Kode Sumber 4.4.

```
count = 0

for train_index, test_index in
skf.split(path_data, class_):
    count += 1
    print count
```

```
6.
7.      x_train, x_test = image_data[train_index],
            image_data[test_index]
8.      y_train, y_test = class_[train_index],
            class_[test_index]
9.      y_train = keras.utils.to_categorical(y_train,
            2)
10.     y_test = keras.utils.to_categorical(y_test, 2)
11.
12.     h5f =
             h5py.File('C:/Users/incr/Documents/code/d
             ata/new_dataset'+str(count)+'.h5','w')
13.   h5f.create_dataset('x_train',data=x_train)
14.   h5f.create_dataset('x_test',data=x_test)
15.   h5f.create_dataset('y_train',data=y_train)
16.   h5f.create_dataset('y_test',data=y_test)
17.
18.     h5f.close()
```

Kode Sumber 4.4 Kode Program untuk Menyimpan Dataset dalam Format HDF5

4.2.5. Implementasi Training dan Klasifikasi dengan Model CanNet

Pada proses ini dilakukan training dan klasifikasi dengan model CanNet. Implementasi fungsi ini dapat ditunjukkan pada Kode Sumber 4.5.

```
1.  model = Sequential()
2.
3.  model.add(ZeroPadding2D((1,1),input_shape=(545,128
    ,128)))
4.  model.add(Conv2D(78, (50,
    50),strides=(6,6),padding='same',
5.                   data_format="channels_first",
    use_bias=True,
6.
    bias_initializer=keras.initializers.Constant(value
    =0),
7.
    kernel_initializer='glorot_normal'))
8.  model.add(Activation('relu'))
```

```
9.  model.add(Conv2D(78, (3, 3),
    padding='same',use_bias=True,
10.
    bias_initializer=keras.initializers.Constant(value
    =1),
11.
    kernel_initializer='glorot_normal', strides=1))
12. model.add(Activation('relu'))
13. model.add(MaxPooling2D(pool_size=(13,
    13),strides=(13,13)))
14. model.add(Dropout(0.5))
15. model.add(Flatten())
16. model.add(Dense(2, use_bias=True,
    bias_initializer=keras.initializers.Constant(value
    =0)))
17. model.add(Activation('softmax'))
18.
19. model.compile(loss='categorical_crossentropy',
    optimizer=opt, metrics=['accuracy'])
20.
21. history= model.fit(x_train, y_train,
    batch_size=20, epochs=1000, verbose=1,
    validation_data=(x_test, y_test))
```

Kode Sumber 4.5 Kode Program Training dan Klasifikasi Model CanNet

4.2.6. Implementasi Training dan Klasifikasi dengan Model LeNet

Pada proses ini dilakukan training dan klasifikasi dengan model LeNet. Implementasi fungsi ini dapat ditunjukkan pada Kode Sumber 4.6.

```
1.  model = Sequential()
2.
3.  model.add(Conv2D(20, (5, 5), padding="same",
    data_format="channels_first",
4.      input_shape=(545,128,128)))
5.  model.add(Activation("relu"))
6.  model.add(MaxPooling2D(pool_size=(2, 2),
    strides=(2, 2)))
7.
```

```
8.  model.add(Conv2D(50, (5, 5), padding="same"))
9.  model.add(Activation("relu"))
10. model.add(MaxPooling2D(pool_size=(2, 2),
    strides=(2, 2)))
11.
12. model.add(Flatten())
13. model.add(Dense(500))
14. model.add(Activation("relu"))
15.
16. model.add(Dense(2))
17. model.add(Activation("softmax"))
18.
19. model.compile(loss='categorical_crossentropy',
    optimizer=opt, metrics=['accuracy'])
20.
21. history= model.fit(x_train, y_train, batch_size=20,
    epochs=1000, verbose=1, validation_data=(x_test,
    y_test))
```

Kode Sumber 4.6 Kode Program Training dan Klasifikasi Model Lennet

4.2.7. Implementasi Training dan Klasifikasi dengan Model VGG16

Pada proses ini dilakukan training dan klasifikasi dengan model VGG16. Implementasi fungsi ini dapat ditunjukkan pada Kode Sumber 4.7.

```
1.  model = Sequential()
2.
3.  model.add(ZeroPadding2D((1,1),input_shape=(545,128
    ,128)))
4.  model.add(Conv2D(64, (3, 3), activation='relu',
    data_format="channels_first"))
5.  model.add(ZeroPadding2D((1,1)))
6.  model.add(Conv2D(64, (3, 3), activation='relu'))
7.  model.add(MaxPooling2D((2,2), strides=(2,2)))
8.
9.  model.add(ZeroPadding2D((1,1)))
10. model.add(Conv2D(128, (3, 3), activation='relu'))
11. model.add(ZeroPadding2D((1,1)))
12. model.add(Conv2D(128, (3, 3), activation='relu'))
13. model.add(MaxPooling2D((2,2), strides=(2,2)))
```

```
14.
15. model.add(ZeroPadding2D((1,1)))
16. model.add(Conv2D(256, (3, 3), activation='relu'))
17. model.add(ZeroPadding2D((1,1)))
18. model.add(Conv2D(256, (3, 3), activation='relu'))
19. model.add(ZeroPadding2D((1,1)))
20. model.add(Conv2D(256, (3, 3), activation='relu'))
21. model.add(MaxPooling2D((2,2), strides=(2,2)))
22.
23. model.add(ZeroPadding2D((1,1)))
24. model.add(Conv2D(512, (3, 3), activation='relu'))
25. model.add(ZeroPadding2D((1,1)))
26. model.add(Conv2D(512, (3, 3), activation='relu'))
27. model.add(ZeroPadding2D((1,1)))
28. model.add(Conv2D(512, (3, 3), activation='relu'))
29. model.add(MaxPooling2D((2,2), strides=(2,2)))
30.
31. model.add(ZeroPadding2D((1,1)))
32. model.add(Conv2D(512, (3, 3), activation='relu'))
33. model.add(ZeroPadding2D((1,1)))
34. model.add(Conv2D(512, (3, 3), activation='relu'))
35. model.add(ZeroPadding2D((1,1)))
36. model.add(Conv2D(512, (3, 3), activation='relu'))
37. model.add(MaxPooling2D((2,2), strides=(2,2)))
38.
39. model.add(Flatten())
40. model.add(Dense(4096, activation='relu'))
41. model.add(Dropout(0.5))
42. model.add(Dense(4096, activation='relu'))
43. model.add(Dropout(0.5))
44. model.add(Dense(2, activation='softmax'))
45.
46. model.compile(loss='categorical_crossentropy',
    optimizer=opt, metrics=['accuracy'])
47.
48. history= model.fit(x_train, y_train,
    batch_size=20, epochs=1000, verbose=1,
    validation_data=(x_test, y_test))
```

Kode Sumber 4.7 Kode Program Training dan Klasifikasi Model VGG16

4.2.8. Implementasi Training dan Klasifikasi dengan Model VGG19

Pada proses ini dilakukan training dan klasifikasi dengan model VGG19. Implementasi fungsi ini dapat ditunjukkan pada Kode Sumber 4.8.

```
1.  model = Sequential()
2.
3.  model.add(ZeroPadding2D((1,1),input_shape=(545,128
    ,128)))
4.  model.add(Conv2D(64, (3, 3), activation='relu',
    data_format="channels_first"))
5.  model.add(ZeroPadding2D((1,1)))
6.  model.add(Conv2D(64, (3, 3), activation='relu'))
7.  model.add(MaxPooling2D((2,2), strides=(2,2)))
8.
9.  model.add(ZeroPadding2D((1,1)))
10. model.add(Conv2D(128, (3, 3), activation='relu'))
11. model.add(ZeroPadding2D((1,1)))
12. model.add(Conv2D(128, (3, 3), activation='relu'))
13. model.add(MaxPooling2D((2,2), strides=(2,2)))
14.
15. model.add(ZeroPadding2D((1,1)))
16. model.add(Conv2D(256, (3, 3), activation='relu'))
17. model.add(ZeroPadding2D((1,1)))
18. model.add(Conv2D(256, (3, 3), activation='relu'))
19. model.add(ZeroPadding2D((1,1)))
20. model.add(Conv2D(256, (3, 3), activation='relu'))
21. model.add(ZeroPadding2D((1,1)))
22. model.add(Conv2D(256, (3, 3), activation='relu'))
23. model.add(MaxPooling2D((2,2), strides=(2,2)))
24.
25. model.add(ZeroPadding2D((1,1)))
26. model.add(Conv2D(512, (3, 3), activation='relu'))
27. model.add(ZeroPadding2D((1,1)))
28. model.add(Conv2D(512, (3, 3), activation='relu'))
29. model.add(ZeroPadding2D((1,1)))
30. model.add(Conv2D(512, (3, 3), activation='relu'))
31. model.add(ZeroPadding2D((1,1)))
32. model.add(Conv2D(512, (3, 3), activation='relu'))
```

```
33. model.add(MaxPooling2D((2,2), strides=(2,2)))
34.
35. model.add(ZeroPadding2D((1,1)))
36. model.add(Conv2D(512, (3, 3), activation='relu'))
37. model.add(ZeroPadding2D((1,1)))
38. model.add(Conv2D(512, (3, 3), activation='relu'))
39. model.add(ZeroPadding2D((1,1)))
40. model.add(Conv2D(512, (3, 3), activation='relu'))
41. model.add(ZeroPadding2D((1,1)))
42. model.add(Conv2D(512, (3, 3), activation='relu'))
43. model.add(MaxPooling2D((2,2), strides=(2,2)))
44.
45. model.add(Flatten())
46. model.add(Dense(4096, activation='relu'))
47. model.add(Dropout(0.5))
48. model.add(Dense(4096, activation='relu'))
49. model.add(Dropout(0.5))
50. model.add(Dense(2, activation='softmax'))
51.
52. model.compile(loss='categorical_crossentropy',
    optimizer=opt, metrics=['accuracy'])
53.
54. history= model.fit(x_train, y_train,
    batch_size=20, epochs=1000, verbose=1,
    validation_data=(x_test, y_test))
```

Kode Sumber 4.8 Kode Program Training dan Klasifikasi Model VGG19

BAB V
PENGUJIAN DAN EVALUASI

Dalam bab ini dibahas mengenai hasil uji coba sistem yang telah dirancang dan dibuat. Uji coba dilakukan untuk mengetahui kinerja sistem dengan lingkungan uji coba yang telah ditentukan serta dilakukan sesuai dengan skenario uji coba.

5.1. Lingkungan Pengujian

Lingkungan pengujian sistem pada pengerjaan tugas akhir ini dilakukan pada lingkungan dan alat kakas sebagai berikut:

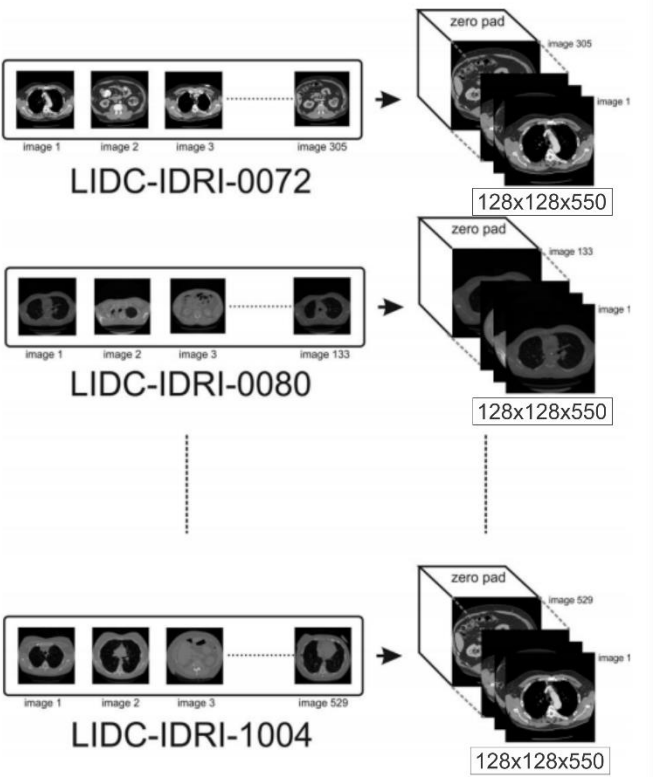
Prosesor	: Prosesor: Intel® Core™ i7-7700 CPU @ 3.60GHz (8 CPUs) ~3.60GHz
RAM	: 16384 MB
Jenis *Device*	: Personal Komputer
Sistem Operasi	: Microsoft Windows 10

5.2. Data Uji Coba

Dataset yang akan digunakan pada tugas akhir ini adalah dataset LIDC-IDRI berisikan hasil pemindaian 1018 total pasien. Setiap citra CT scan pasien terdiri dari sekitar 150 sampai 550 citra yang berformat dicom. Dataset menyajikan empat kelas yaitu *Unknown, Benign* (jinak), *Malignant* (ganas), dan *Metastatic*. Pada tugas akhir ini hanya akan menggunakan kelas *benign* dan *malignant* sebagai label tingkat keganasan kanker. Hal ini disebabkan karena mengklasifikasikan kelas *Unknown* merupakan hal yang sia-sia. Kategori *Metastatic* memerlukan citra CT scan bagian tubuh manusia yang lain yang tidak disediakan, sehingga menjadi hal yang sulit untuk dilakukan klasifikasi.

Setelah memisahkan kelas secara keseluruhan, didapatkan total 70 data pasien. Pada awalnya semua citra berformat dicom, *grayscale* dan memiliki ukuran 512x512 piksel. Seluruh citra akan diperkecil menjadi berukuran 128x128 piksel. Setiap citra kemudian akan dinormalisasi, hal ini dilakukan supaya perhitungan menjadi lebih sederhana dan mengurangi kesalahan dalam

perhitungan. Selanjutnya, masing-masing citra *grayscale* digabungkan secara linier untuk membuat array 3D dan mengubah format citra menjadi HDF5. Array 3D ini kemudian ditambahkan *zero padded* untuk menyamakan ukuran array 3D. Satu data pasien yang berformat HDF5 akan akan dirubah lagi menjadi array berukuran 1 x 550 x 128 x 128. HDF5 berisi dua masukan, yaitu data dan label kelas. Pada tugas akhir ini nilai 0 menunjukkan kelas *benign* dan nilai 1 menunjukkan kelas *malignant***.**



Gambar 5.1 Contoh Citra Uji [3]

5.3. Uji Coba Performa Arsitektur

Pada pengujian ini akan dilakukan pengujian terhadap model CNN dengan tujuan melakukan perbandingan untuk mendapatkan model yang terbaik. Perbandingan arsitektur akan dilakukan melaui dua tahap, yaitu pengujian arsitektur jaringan menggunakan *cross validation* dan pengujian metode optimasi melalui hasil akurasi dan nilai *loss* pada setiap arsitektur.

5.3.1. Arsitektur Jaringan

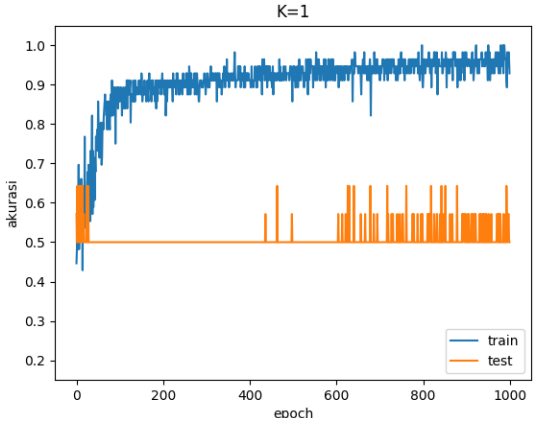
Pengujian akan dilakukan menggunakan data lengkap pada keempat model CNN dengan tujuan melakukam komparasi arsitektur jaringan untuk mendapatkan model terbaik. Pada pengujian ini, akan digunakan model CNN CanNet, LeNet, VGG16, dan VGG19. Masing-masing model akan diuji menggunakan *k-fold cross validation*. Parameter pengujian dapat ditunjukkan dengan Tabel 5.1.

Tabel 5.1 Parameter Pengujian *k-fold cross validation*

Parameter	Nilai
Optimasi	SGD learning rate 0.001
Bacth size	20
Epoch	1000
Jumlah K	5

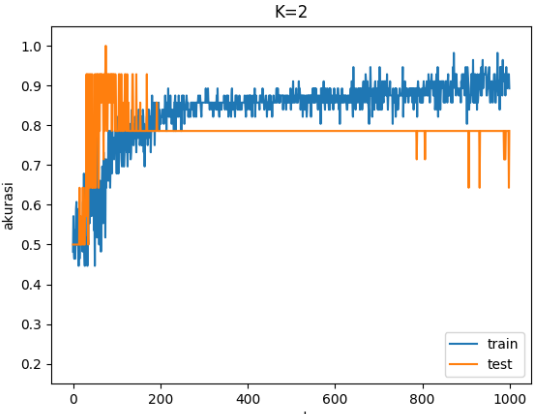
5.3.1.1. Performa Model CanNet

Model CanNet mendapatkan akurasi tertinggi pada fold 1 untuk akurasi data *training* sebesar 100%. Namun hasil akurasi data *testing* terbaik pada fold 1 hanya sebesar 64%. Fold 3 dan fold 4 menunjukkan akurasi hingga 100% pada data *testing*. Hingga epoch ke 200 model CanNet mengalami perubahan akurasi yang cukup tinggi. Pada fold 3 dan fold 4 grafik perubahan akurasi data *training* dan data *testing* menunjukkan perubahan secara fluktuasi.



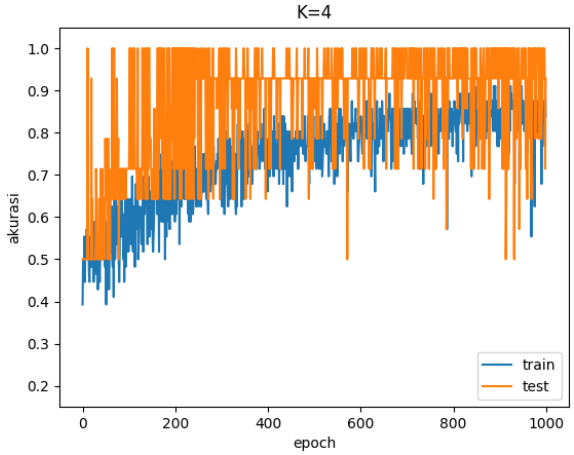
Gambar 5.2 Perbandingan Akurasi Train dan Test Model CanNet pada Fold 1

Pada fold pertama perubahan akurasi untuk data *training* menunjukkan perubahan secara signifikan hingga epoch 1000. Namun untuk data *testing*, perubahan akurasi cenderung konstan pada nilai 0.5 hingga 0.6. Grafik perbandingan akurasi pada fold pertama dapat ditunjukkan dengan Gambar 5.2.



Gambar 5.3 Perbandingan Akurasi Train dan Test Model CanNet pada Fold 2

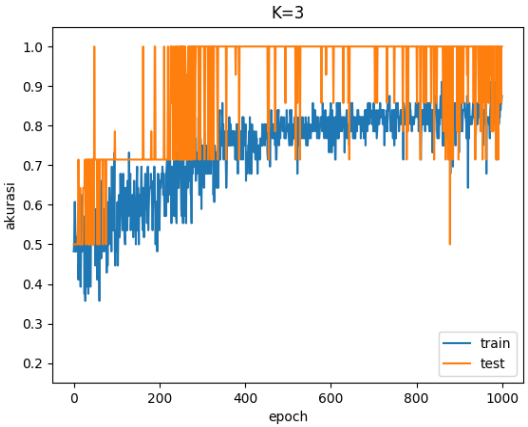
Pada fold 2 perubahan akurasi untuk data *training* menunjukkan perubahan secara signifikan seperti pada fold 1. Akurasi data testing menunjukkan hasil konstan mulai dari epoch 200 hingga epoch 1000. Gafik perbandingan akurasi pada fold 2 dapat ditunjukkan dengan Gambar 5.3.



Gambar 5.4 Perbandingan Akurasi Train dan Test Model CanNet pada Fold 3

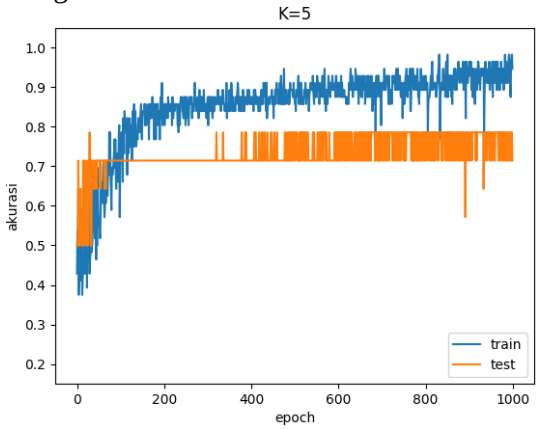
Fold 3 menunjukkan perubahan akurasi untuk data *training* perubahan secara fluktuatif. Perubahan fluktuatif ini juga terjadi untuk data *testing.* Grafik perbandingan akurasi pada fold 3 dapat ditunjukkan dengan Gambar 5.4.

Pada fold 4 perubahan akurasi untuk data *training* menunjukkan grafik naik hingga epoch 1000. Namun perubahan fluktuatif terjadi untuk data *testing*. Grafik perbandingan akurasi pada fold 4 dapat ditunjukkan dengan Gambar 5.5.



Gambar 5.5 Perbandingan Akurasi Train dan Test Model CanNet pada Fold 4

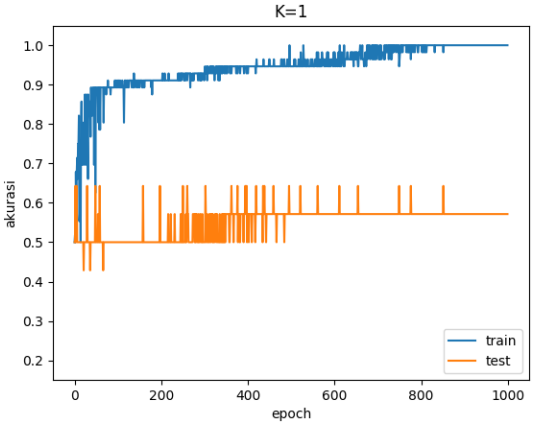
Seperti halnya pada fold 1, pada fold 5 menunjukkan perubahan akurasi secara signifikan untuk data *testing*. Akurasi data testing menunjukkan hasil konstan mulai dari epoch 50 hingga epoch 1000. Grafik perbandingan akurasi pada fold 5 dapat ditunjukkan dengan Gambar 5.6



Gambar 5.6 Perbandingan Akurasi Train dan Test Model CanNet pada Fold 5

5.3.1.2. Performa Model LeNet

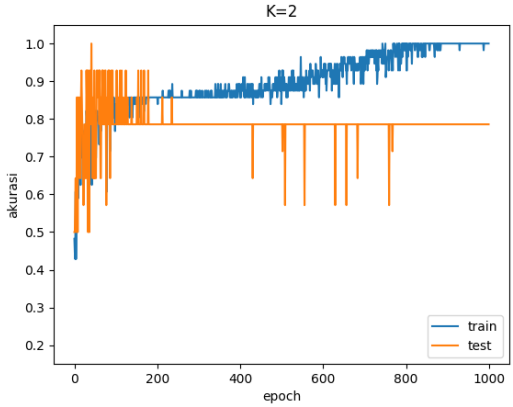
Model LeNet mendapatkan akurasi 100% untuk data training pada semua fold. Hasil akurasi data *testing* terendah pada fold 1 sebesar 64%. Model ini menjadi satu-satunya yang menunjukkan hasil 100% pada semua fold untuk data *training*. Mulai dari epoch 1 sampai epoch ke 50 terjadi perubahan akurasi yang sangat pesat. Pada epoch ke 700 akurasi sudah mencapai 100%.



Gambar 5.7 Perbandingan Akurasi Train dan Test Model Lenet pada Fold 1

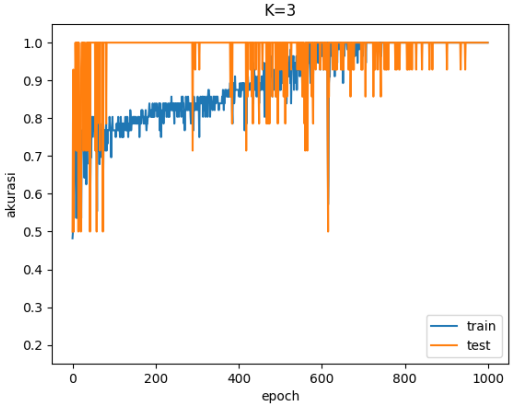
Pada fold pertama perubahan akurasi untuk data *training* menunjukkan perubahan secara signifikan hingga epoch 1000. Namun untuk data *testing*, perubahan akurasi cenderung konstan pada nilai 0.5 hingga 0.6. Grafik perbandingan akurasi pada fold pertama dapat ditunjukkan dengan Gambar 5.7.

Perubahan akurasi untuk data *training* pada fold 2 menunjukkan perubahan secara signifikan seperti pada fold 1. Akurasi data testing menunjukkan hasil konstan mulai dari epoch 200 hingga epoch 1000. Grafik perbandingan akurasi pada fold 2 dapat ditunjukkan dengan Gambar 5.8.

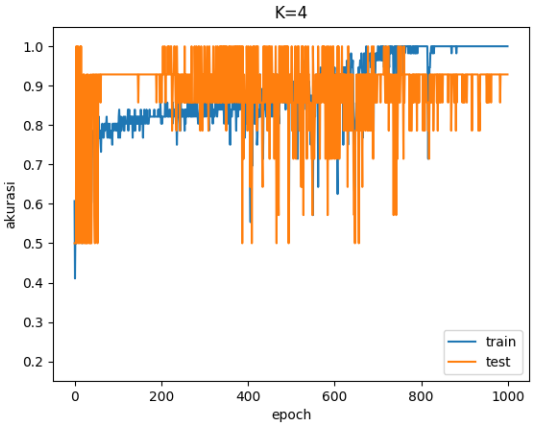


Gambar 5.8 Perbandingan Akurasi Train dan Test Model Lenet pada Fold 2

Pada fold 3 perubahan akurasi untuk data *training* menunjukkan perubahan secara signifikan seperti pada fold 1. Akurasi data testing menunjukkan hasil maksimal mulai epoch awal hingga akhir. Fold 3 mendapatkan hasil paling bagus untuk arsitektur Lenet dengan mendapatkan akurasi hingga 1.0. Grafik perbandingan akurasi pada fold 3 dapat ditunjukkan dengan Gambar 5.9.

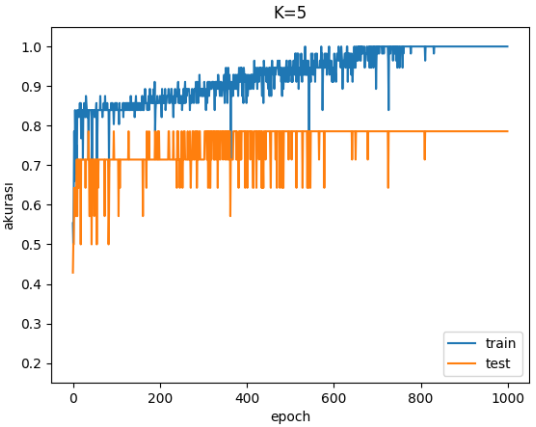


Gambar 5.9 Perbandingan Akurasi Train dan Test Model Lenet pada Fold 3



Gambar 5.10 Perbandingan Akurasi Train dan Test Model Lenet pada Fold 4

Pada fold 4 perubahan akurasi untuk data *training* menunjukkan grafik naik hingga epoch 1000. Pada epoch 700 akurasi sudah mencapai nilai maksimal. Namun perubahan fluktuatif terjadi untuk data *testing*. Grafik perbandingan akurasi pada fold 4 dapat ditunjukkan dengan Gambar 5.10.

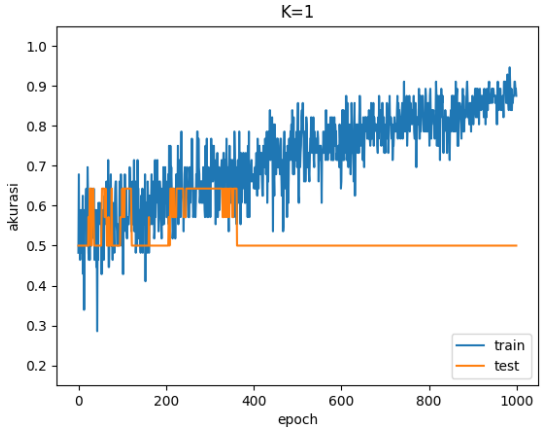


Gambar 5.11 Perbandingan Akurasi Train dan Test Model Lenet pada Fold 5

Pada fold 5 perubahan akurasi untuk data *training* menunjukkan perubahan secara signifikan. Akurasi data testing menunjukkan hasil konstan mulai dari epoch 50 hingga epoch 1000. Grafik perbandingan akurasi pada fold 5 dapat ditunjukkan dengan Gambar 5.11.

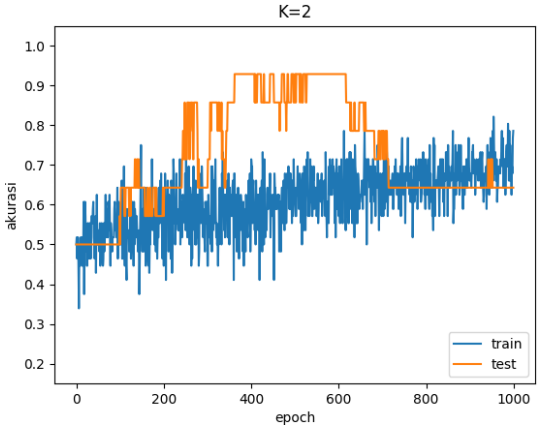
5.3.1.3. Performa Model VGG16

Akurasi terbaik model VGG16 terletak pada fold 1 sebesar 94%. Pada fold 3 data *testing* menunjukkan akurasi tertinggi sebesar 100%. Pada fold ke 2 arsitektur VGG16 mendapatkan akurasi terendah dari semua fold untuk data *training*. Perubahan akurasi pada model ini cukup fluktuatif hingga epoch ke 1000. Grafik perbandingan akurasi pada fold 5 dapat ditunjukkan dengan Gambar 5.12.



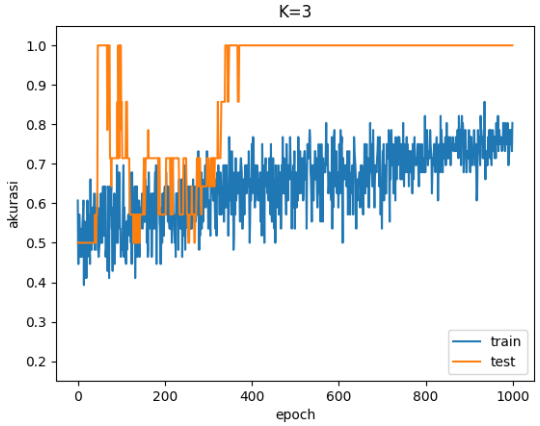
Gambar 5.12 Perbandingan Akurasi Train dan Test Model VGG16 pada Fold 1

Akurasi data *testing* terbaik pada fold 2 yaitu bernilai 0.928. Perubahan akurasi untuk data *training* menunjukkan perubahan secara signifikan seperti pada fold 1. Gafik perbandingan akurasi pada fold 2 dapat ditunjukkan dengan Gambar 5.13.

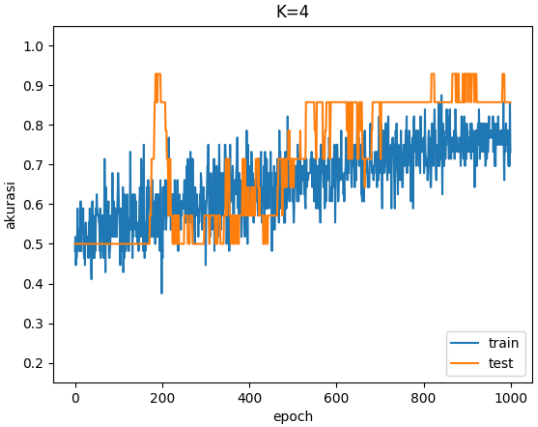


Gambar 5.13 Perbandingan Akurasi Train dan Test Model VGG16 pada Fold 2

Pada fold 3 merupakan akurasi terbaik dari semua fold yang ada. Akurasi bernilai 1.0 sejak epoch 350 hingga epoch ke 1000. Grafik perbandingan akurasi pada fold 3 dapat ditunjukkan dengan Gambar 5.14. Grafik perbandingan akurasi pada fold 3 dapat ditunjukkan dengan Gambar 5.15.

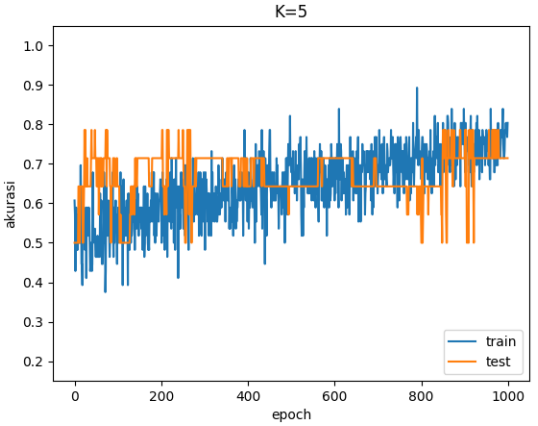


Gambar 5.14 Perbandingan Akurasi Train dan Test Model VGG16 pada Fold 3



Gambar 5.15 Perbandingan Akurasi Train dan Test Model VGG16 pada Fold 4

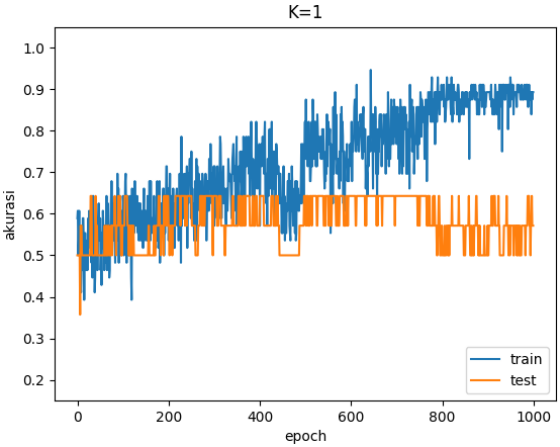
Nilai akurasi tertinggi pada fold 5 yaitu 0.89 untuk data *training* dan 0.785 untuk data *testing*. Grafik perbandingan akurasi pada fold 5 dapat ditunjukkan dengan Gambar 5.16.



Gambar 5.16 Perbandingan Akurasi Train dan Test Model VGG16 pada Fold 5

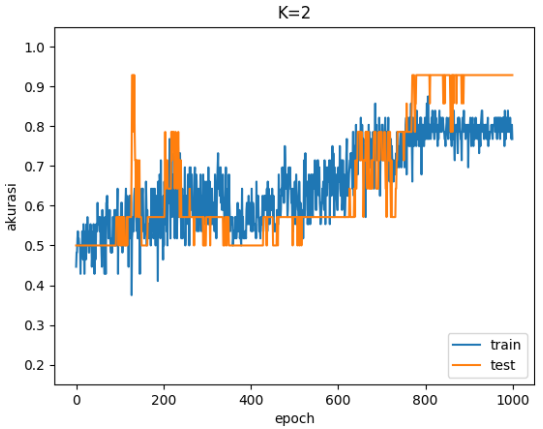
5.3.1.4. Performa Model VGG19

Akurasi terbaik model VGG16 terletak pada fold 1 sebesar 94%. Akurasi data *testing* terbaik terletak pad fold 3 dan fold 4. Hasil akurasi dari model VGG19 tidak beda jauh dari model VGG16. Hal ini menunjukkan bahwa penambahan tiga layer konvolusi pada VGG19 tidak berpengaruh besar pada proses klasifikasi. Grafik perbandingan akurasi pada fold 5 dapat ditunjukkan dengan Gambar 5.17.



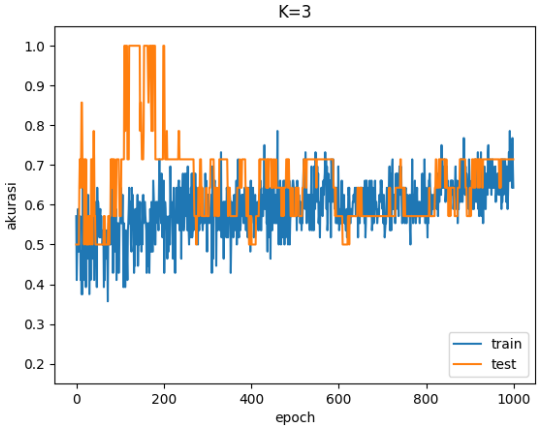
Gambar 5.17 Perbandingan Akurasi Train dan Test Model VGG19 pada Fold 1

Nilai akurasi tertinggi pada fold 2 dengan model VGG19 yaitu 0.928 untuk data *testing* dan 0.875 untuk data *training*. Pada epoch ke 128 akurasi mencapai nilai tertinggi untuk data *testing*, namun akurasi turun lagi hingga epoch ke 600. Grafik perbandingan akurasi pada fold 2 dapat ditunjukkan dengan Gambar 5.18.

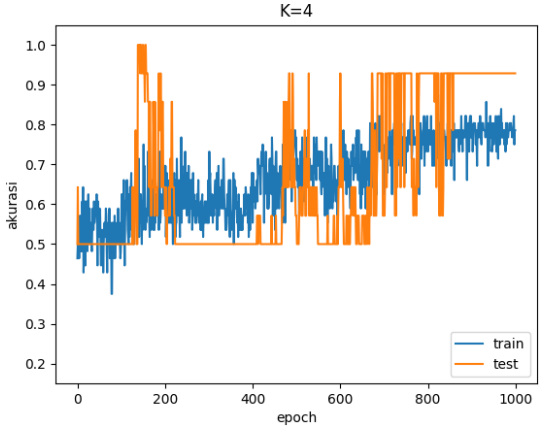


Gambar 5.18 Perbandingan Akurasi Train dan Test Model VGG19 pada Fold 2

Pada fold 3 akurasi bernilai 1.0 di epoch 110. Namun akurasi turun pada epoch 180 hingga epoch ke 1000. Grafik perbandingan akurasi pada fold 3 dapat ditunjukkan dengan Gambar 5.19.

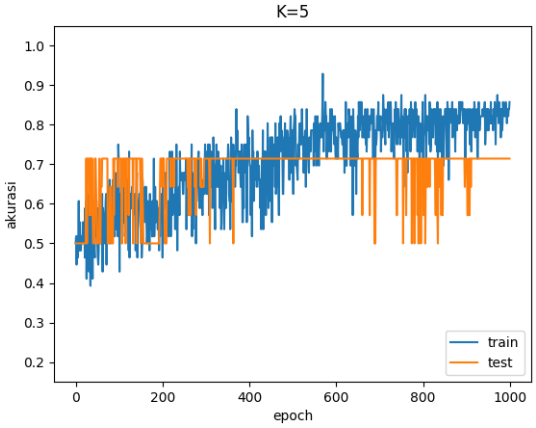


Gambar 5.19 Perbandingan Akurasi Train dan Test Model VGG19 pada Fold 3



Gambar 5.20 Perbandingan Akurasi Train dan Test Model VGG19 pada Fold 4

Akurasi pada fold 4 mencapai nilai 1.0 di epoch 140. Grafik perbandingan akurasi pada fold 3 dapat ditunjukkan dengan Gambar 5.20. Sedangkan pada fold 5, akurasi cenderung konstan untuk data *testing*. Grafik perbandingan akurasi pada fold 3 dapat ditunjukkan dengan Gambar 5.21.



Gambar 5.21 Perbandingan Akurasi Train dan Test Model VGG19 pada Fold 5

5.3.2. Running Time Program

Pada pengujian ini akan dihitung waktu *run time* setiap arsitektur jaringan. Pengujian akan dilakukan dengan parameter epoch sebesar 1000. Waktu *run time* dihitung mulai dari awal pelatihan hingga perhitungan klasifikasi. Hasil *run time* arsitektur CNN dapat ditunjukkan dengan Tabel 5.2.

Tabel 5.2 Hasil Rata-rata *Run Time* Aristektur CNN

Arsitektur	Waktu Pelatihan Model
CanNet	121981 detik (33.88 jam)
LeNet	38903 detik (10.8 jam)
VGG16	**33474 detik (9.89 jam)**
VGG19	36028 detik (10 jam)

Dari hasil pengujian, didapatkan bahwa arsitektur yang memakan waktu terlama adalah arsitektur CanNet yaitu 121981 detik atau sekitar 34 jam. Arsitektur VGG16 memakan waktu paling sedikit yaitu 33474 detik atau sekitar 10 jam. Hal tersebut dikarenakan arsitektur CanNet memiliki ukuran kernel yang besar pada matrik konvolusi yang pertama, yaitu sebesar 50x50. Dapat disimpulkan bahwa ukuran kernel yang besar mempengaruhi waktu pelatihan sebuah arsitektur CNN.

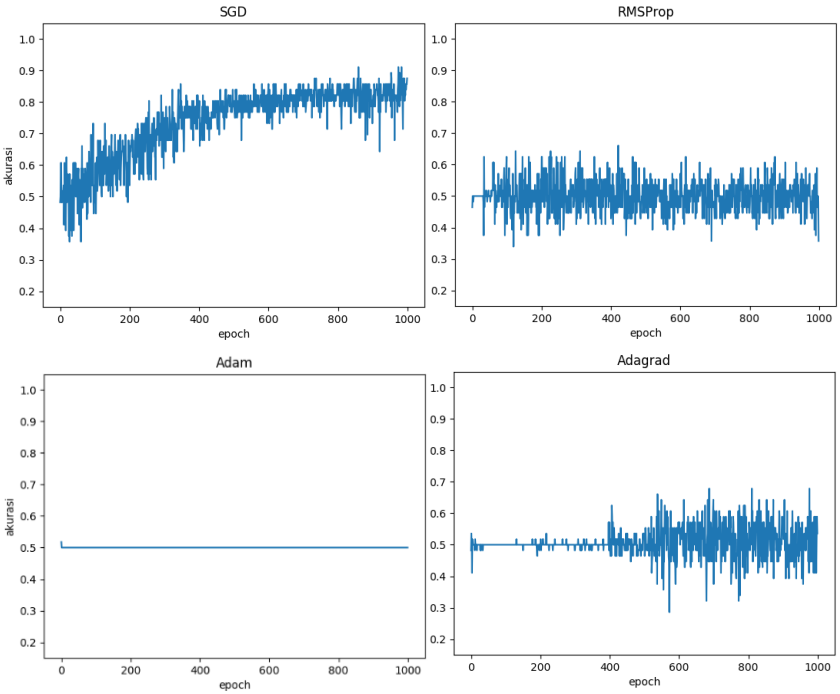
5.3.3. Metode Optimasi

Pengujian dilakukan dengan mengubah parameter optimasi pada setiap arsitektur. Pengujian akan dilakukan dengan parameter epoch sebesar 1000 dan menggunakan arsitektur CanNet, LeNet, VGG16, dan VGG19. Pengujian dilakukan menggunakan data training.

Dari hasil pengujian dengan arsitektur CanNet, didapatkan bahwa akurasi tertinggi didapatkan dengan metode optimasi SGD dengan nilai 91%. Ketiga optimasi lainnya, yaitu RMSProp, Adagrad, dan Adam tidak mengalami perubahan akurasi secara signifikan sejak epoch yang pertama. Hasil pengujian metode optimasi arsitektur CanNet dapat ditunjukkan dengan Tabel 5.3.

Tabel 5.3 Hasil Pengujian Metode Optimasi Arsitektur CanNet

Optimasi	Akurasi Tertinggi (Training)
SGD	**0.91**
RMSprop	0.66
Adam	0.517
Adagrad	0.678

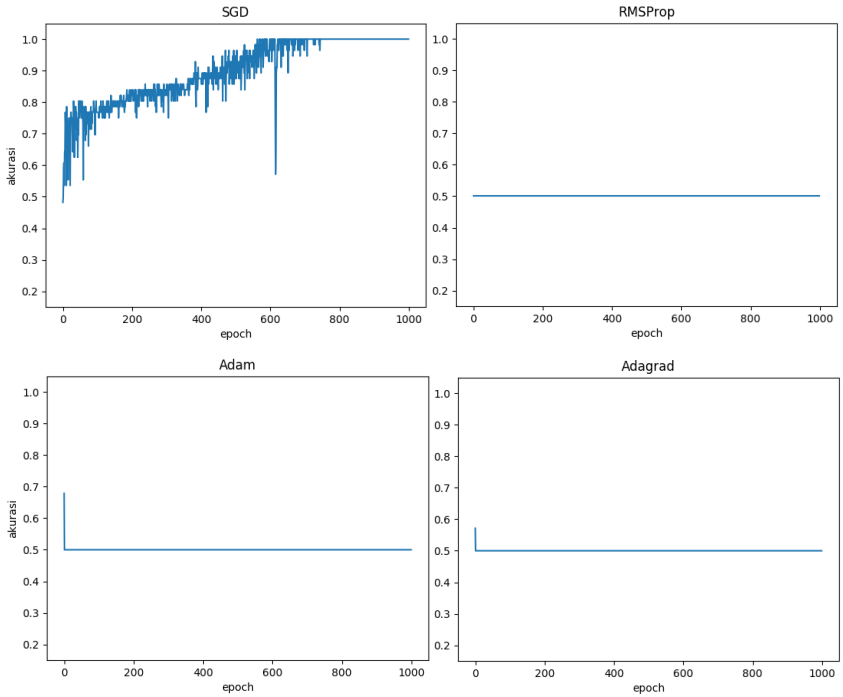


Gambar 5.22 Grafik Perbandingan Akurasi CanNet terhadap Optimasi

Dari hasil pengujian dengan arsitektur CanNet, didapatkan bahwa akurasi tertinggi didapatkan dengan metode optimasi SGD dengan nilai 100%. Seperti halnya arstitektur CanNet, hasil pegujian ketiga metode optimasi lainnya menunjukkan tidak adanya perubahan akurasi. Hasil pengujian metode optimasi arsitektur LeNet dapat ditunjukkan dengan Tabel 5.4.

Tabel 5.4 Hasil Pengujian Metode Optimasi Arsitektur LeNet

Optimasi	Akurasi Tertinggi (Training)
SGD	**1**
RMSprop	0.5
Adam	0.678
Adagrad	0.571

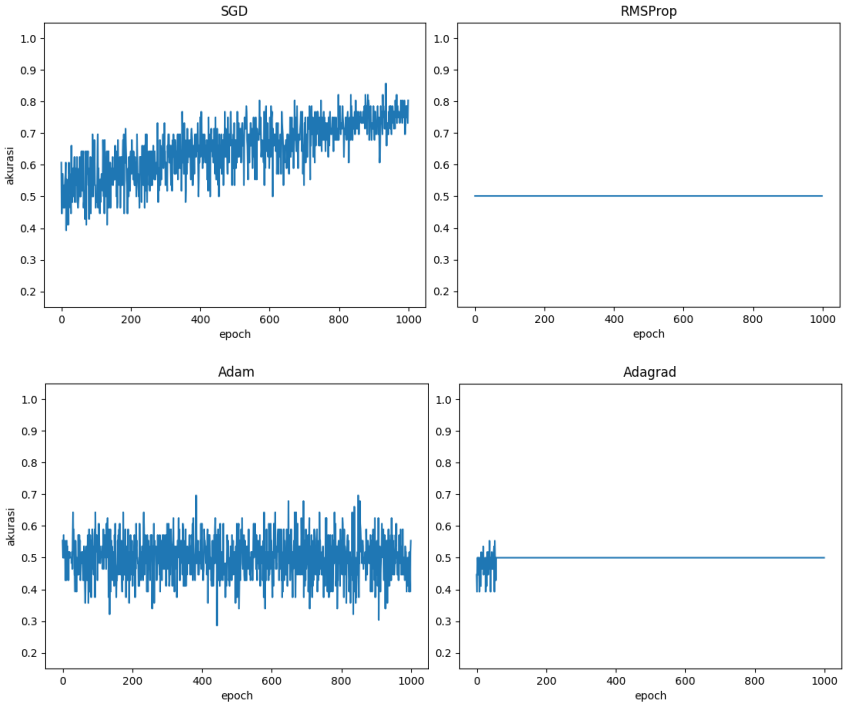


Gambar 5.23 Grafik Perbandingan Akurasi LeNet terhadap Optimasi

Selanjutnya, dari hasil pengujian dengan arsitektur VGG16, didapatkan bahwa akurasi tertinggi didapatkan dengan metode optimasi SGD dengan nilai 85.7%. Hasil pengujian metode optimasi arsitektur VGG16 dapat ditunjukkan dengan Tabel 5.5.

Tabel 5.5 Hasil Pengujian Metode Optimasi Arsitektur VGG16

Optimasi	Akurasi Tertinggi (Training)
SGD	**0.857**
RMSprop	0.5
Adam	0.696
Adagrad	0.553

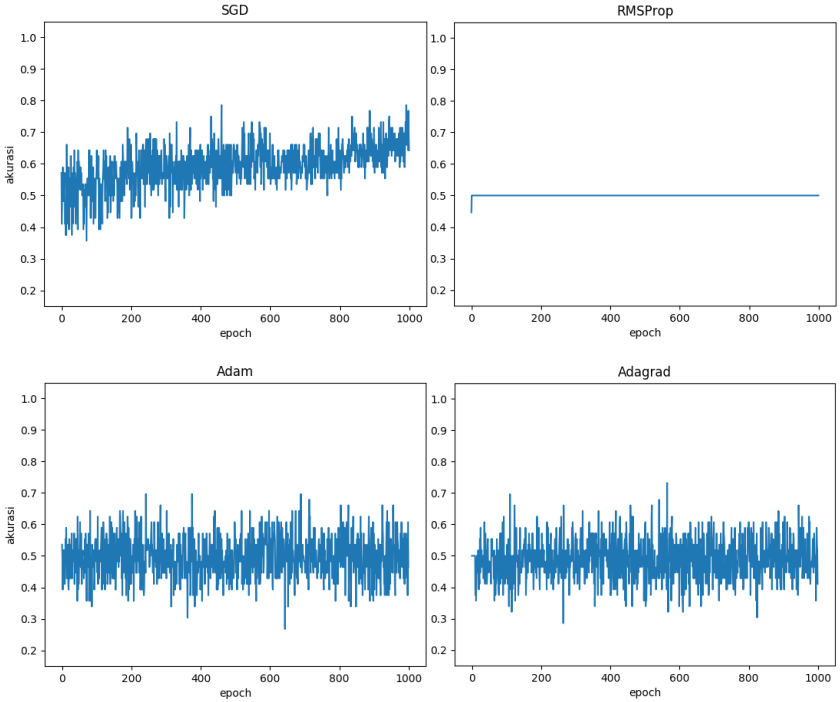


Gambar 5.24 Grafik Perbandingan Akurasi VGG16 terhadap Optimasi

Dari hasil pengujian dengan arsitektur VGG19, didapatkan bahwa akurasi tertinggi didapatkan dengan metode optimasi SGD dengan nilai 78.5%. Hasil pengujian metode optimasi arsitektur VGG19 dapat ditunjukkan dengan Tabel 5.6.

Tabel 5.6 Hasil Pengujian Metode Optimasi Arsitektur VGG19

Optimasi	Akurasi Tertinggi (Training)
SGD	**0.785**
RMSprop	0.5
Adam	0.696
Adagrad	0.732



Gambar 5.25 Grafik Perbandingan Akurasi VGG19 terhadap Optimasi

5.4. Analisis Hasil Uji Coba

Untuk menentukan arsitektur pada CNN, tidak ada rumus maupun konfigurasi parameter standar dikarenakan hasil sangat bergantung kepada data yang digunakan. Diperlukan serangkaian pengujian untuk mendapatkan model akhir dengan hasil yang optimal.

Model LeNet mencapai akurasi rata-rata terbesar dengan nilai 0.814. Hasil tersebut cukup signifikan dibandingkan model lainnya. Pada k pertama menghasilkan akurasi terendah, sedangkan pada k ketiga menghasilkan akurasi tertinggi dalam pengujian. Arsitektur yang memakan waktu terlama adalah arsitektur CanNet. Ketiga arsitektur lainnya, yaitu LeNet, VGG16, dan VGG19 memiliki waktu *run time* hampir sama. Hal tersebut dikarenakan arsitektur CanNet memiliki ukuran kernel yang besar pada matrik konvolusi yang pertama, yaitu sebesar 50x50. Dapat disimpulkan bahwa ukuran kernel yang besar mempengaruhi waktu pelatihan sebuah arsitektur CNN. Hasil pengujian akurasi setiap arsitektur dengan *K Fold Cross Validation* dapat ditunjukkan dengan Tabel 5.7.

Tabel 5.7 Hasil Pengujian Akurasi Setiap Arsitektur

K	CanNet		LeNet		VGG16		VGG19	
	Train	***Test***	***Train***	***Test***	***Train***	***Test***	***Train***	***Test***
1	0.928	0.5	1.0	0.571	0.875	0.5	0.892	0.571
2	0.893	0.786	1.0	0.786	0.785	0.642	0.767	0.929
3	0.875	1.0	1.0	1.0	0.803	1.0	0.642	0.714
4	0.839	0.929	1.0	0.929	0.732	0.857	0.785	0.929
5	0.946	0.714	1.0	0.786	0.803	0.714	0.857	0.714
Rata-rata	0.8962	0.786	1.0	**0.814**	0.799	0.743	0.788	0.771

Arsitektur yang memakan waktu terlama adalah arsitektur CanNet yaitu 121981 detik atau sekitar 34 jam. Arsitektur VGG16 memakan waktu paling sedikit yaitu 33474 detik atau sekitar 10 jam. Hal tersebut dikarenakan arsitektur CanNet memiliki ukuran kernel yang besar pada matrik konvolusi yang pertama, yaitu sebesar 50x50.

Metode optimasi SGD menunjukkan hasil yang bagus untuk semua arsitektur CNN yang telah diuji. Ketiga metode optimasi lainnya, yaitu RMSProp, Adagrad, dan Adam memiliki hasil relatif yang sama. Ketiga optimasi tersebut tidak mengalami perubahan akurasi secara signifikan sejak epoch yang pertama. Hasil pengujian metode optimasi dapat ditunjukkan dengan Tabel 5.8.

Tabel 5.8 Hasil Pengujian Akurasi Setiap Metode Optimasi

	CanNet	LeNet	VGG16	VGG19
SGD	**0.91**	**1**	**0.857**	**0.785**
RMSProp	0.66	0.5	0.5	0.5
Adam	0.517	0.678	0.696	0.696
Adagrad	0.678	0.571	0.553	0.732

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan dijelaskan mengenai kesimpulan dari proses dan uji coba dari program dan saran untuk pengembangan dari program itu sendiri.

6.1. Kesimpulan

Dari hasil uji coba yang telah dilakukan, dapat diambil kesimpulan sebagai berikut :

1. Nilai akurasi tertinggi diperoleh arsitektur LeNet dengan nilai rata-rata 81.4% yang menggunakan arsitektur sederhana dibandingkan dengan arsitektur yang lain. Arsitektur LeNet hanya memiliki dua buah matrik konvolusi dengan ukuran kernel 5x5 yang diikuti sebuah *max pooling layer*.
2. Metode optimasi SGD terbukti berpengaruh dalam proses pelatihan dataset. SGD memberikan perubahan akurasi yang signifikan dibandingkan dengan metode optimasi yang lainnya, yaitu RMSProp, Adam, dan Adagrad.
3. Ukuran kernel pada matrik konvolusi mempengaruhi *run time training* model CNN. Arsitektur CanNet yang memiliki ukuran kernel 50x50 pada matrik konvolusi pertama memakan waktu paling lama sekitar 34 jam. Sedangkan arsitektur VGG16 memakan waktu paling cepat yaitu sekitar 10 jam, walaupun VGG16 memiliki lebih banyak matrik konvolusi namun ukuran kernelnya hanya 3x3.

6.2. Saran

Saran yang diberikan untuk pengembangan perangkat lunak ini adalah:

1. Menggunakan GPU dalam membangun model CNN supaya dapat membuat model CNN dengan lebih cepat.

2. Mencoba berbagai macam arsitektur yang lain. Pada saat ini banyak pengembang yang telah menguji cobakan arsitektur CNN.
3. Melakukan eksplorasi tentang inisialisasi *weight* dan bias pada jaringan. Pada tugas akhir ini, belum dilakukan percobaan terkait beragam metode inisialisasi yang tersedia sehingga masih banyak hyperparameter yang bisa diuji.

DAFTAR PUSTAKA

[1] J. Berman, "CT Scan Lebih Efektif Deteksi Kanker Paru-Paru bagi Perokok dan Bekas Perokok," 12 Juli 2011. [Online]. Available: https://www.voaindonesia.com/a/ct-scan-lebih-efektif-deteksi-kanker-paru-paru-125438693/98447.html. [Diakses 17 Juni 2017].

[2] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE, vol. 86, no. 11, pp. 22782324,* 1998.

[3] P. Rao, N. A. Pereira and R. Srinivasan, "Convolutional neural networks for lung cancer screening in computed tomography (CT) scans," *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), Noida, pp. 489-493,* 2016.

[4] "Convolutional Neural Networks (CNNs / ConvNets)," [Online]. Available: http://cs231n.github.io/convolutional-networks/. [Diakses 14 June 2017].

[5] X. Yang S. Yeo J. Hong S. Wong W. Tang Z. Wu G. Lee S. Chen V. Ding B. Pang A. Choo Y. Su, "A Deep Learning Approach for Tumor Tissue Image Classification," *IASTED Biomedical Engineering,* 2016.

[6] Adnan Qayyum, Syed Muhammad Anwar, Muhammad Awais , Muhammad Majid, "Medical image retrieval using deep convolutional neural network," 2017.

[7] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," 2012.

[8] "What Is GPU Accelerated Computing?,"," NVIDIA, [Online]. Available: http://www.nvidia.com/object/what-is-gpu-computing.html. [Diakses 15 June 2017].

[9] A. Deshpande, “A Beginner's Guide To Understanding Convolutional Neural Networks,” 20 July 2016. [Online]. Available: https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/. [Diakses 14 June 2017].

[10] F. K., “Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position,” *Biological Cybernetics,* 1980.

[11] Y. LeCun, Handwritten Digit Recognition with a Back-Propagation Network, 1990.

[12] o. Z. J. S. G. Z. W. L. Yuanyuan Zhang, “Adaptive Convolutional Neural Network and Its,” *Neural Process Lett* , p. 43:389–399, 2016.

[13] K. Simonyan dan A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *arXiv:1409.1556.*

[14] J. Yang, “ReLU and Softmax Activation Functions,” 11 February 2017. [Online]. Available: https://github.com/Kulbear/deep-learning-nano-foundation/wiki/ReLU-and-Softmax-Activation-Functions. [Diakses 11 April 2018].

[15] S. Sena, “Pengenalan Deep Learning Part 1 : Neural Network,” 28 October 2017. [Online]. Available: https://medium.com/@samuelsena/pengenalan-deep-learning-8fbb7d8028ac. [Diakses 8 April 2018].

[16] A. Rosebrock, “LeNet – Convolutional Neural Network in Python,” 1 Agustus 2016. [Online]. Available: https://www.pyimagesearch.com/2016/08/01/lenet-convolutional-neural-network-in-python/. [Diakses 10 Mei 2018].

[17] "DICOM/DCM File - Sebuah Standar File Tunggal untuk Penyimpanan Citra Medis," 5 Agustus 2011. [Online]. Available: http://medixsoft.co.id/NewsArticles/newsArticles.aspx?id=A080411001. [Diakses 11 Mei 2018].

[18] S. Pöcheim, "Convolutional Neural Networks," 31 Januari 2017. [Online]. Available: https://wiki.tum.de/display/lfdv/Convolutional+Neural+Networks. [Diakses 15 Juni 2017].

[19] "Apa itu CT Scan Paru-paru: Gambaran Umum, Manfaat, dan Hasil yang Diharapkan," 2016. [Online]. Available: https://www.docdoc.com/id/info/procedure/pemindaian-ct-paru-paru/. [Diakses 11 Mei 2018].

[20] ujjwalkarn, "An Intuitive Explanation of Convolutional Neural Networks," 11 Agustus 2016. [Online]. Available: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets. [Diakses 14 Juni 2017].

[21] S. Ruder, "An Overview of Gradient Descent Optimization Algorithms," 19 Januari 2016. [Online]. Available: http://ruder.io/optimizing-gradient-descent/. [Diakses 10 April 2018].

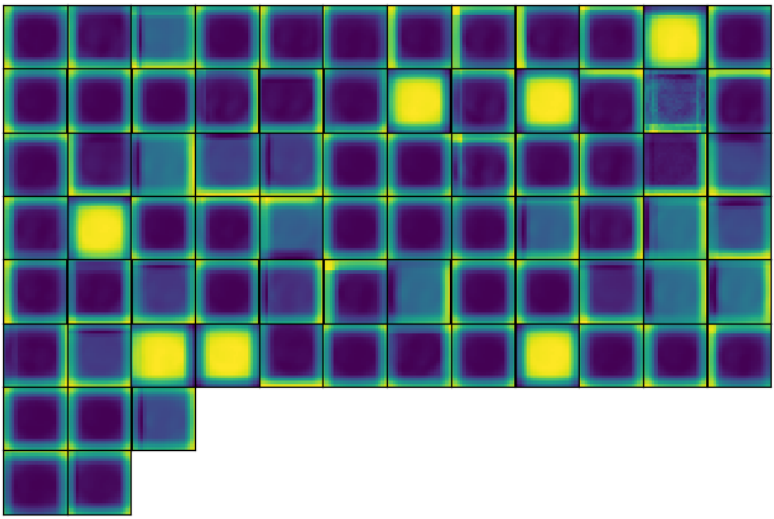
[22] "Apakah itu Kanker Paru-paru," [Online]. Available: https://id.parkwaycancercentre.com/informasi-kanker/jenis-kanker/apakah-itu-kanker-paru-paru/. [Diakses 9 Juli 2018].

[23] Marianti, "Pengertian Kanker Paru-Paru," [Online]. Available: https://www.alodokter.com/kanker-paru-paru. [Diakses 09 07 2018].

[Halaman ini sengaja dikosongkan]

LAMPIRAN

Lampiran 1 Contoh Output Layer Convolusi Pertama CanNet



Lampiran 2 Confusion Matrix CanNet (K=1)

		Kelas Target	
		Benign	***Malignant***
Prediksi	***Benign***	5	5
	Malignant	2	2

Lampiran 3 Confusion Matrix CanNet (K=2)

		Kelas Target	
		Benign	***Malignant***
Prediksi	***Benign***	7	3
	Malignant	0	4

Lampiran 4 Confusion Matrix CanNet (K=3)

		Kelas Target	
		Benign	***Malignant***
Prediksi	***Benign***	7	0
	Malignant	0	7

Lampiran 5 Confusion Matrix CanNet (K=4)

		Kelas Target	
		Benign	***Malignant***
Prediksi	***Benign***	7	1
	Malignant	0	6

Lampiran 6 Confusion Matrix CanNet (K=5)

		Kelas Target	
		Benign	***Malignant***
Prediksi	***Benign***	3	0
	Malignant	4	7

Lampiran 7 Confusion Matrix Lenet (K=1)

		Kelas Target	
		Benign	***Malignant***
Prediksi	***Benign***	6	5
	Malignant	1	2

Lampiran 8 Confusion Matrix Lenet (K=2)

		Kelas Target	
		Benign	***Malignant***
Prediksi	***Benign***	7	0
	Malignant	0	7

Lampiran 9 Confusion Matrix Lenet (K=3)

		Kelas Target	
		Benign	***Malignant***
Prediksi	***Benign***	7	0
	Malignant	0	7

Lampiran 10 Confusion Matrix Lenet (K=4)

		Kelas Target	
		Benign	***Malignant***
Prediksi	***Benign***	6	4
	Malignant	1	3

Lampiran 11 Confusion Matrix Lenet (K=5)

		Kelas Target	
		Benign	***Malignant***
Prediksi	***Benign***	4	0
	Malignant	3	7

Lampiran 12 Confusion Matrix VGG16 (K=1)

		Kelas Target	
		Benign	***Malignant***
Prediksi	***Benign***	5	5
	Malignant	2	2

Lampiran 13 Confusion Matrix VGG16 (K=2)

		Kelas Target	
		Benign	***Malignant***
Prediksi	***Benign***	7	5
	Malignant	0	2

Lampiran 14 Confusion Matrix VGG16 (K=3)

		Kelas Target	
		Benign	***Malignant***
Prediksi	***Benign***	7	0
	Malignant	0	7

Lampiran 15 Confusion Matrix VGG16 (K=4)

		Kelas Target	
		Benign	***Malignant***
Prediksi	***Benign***	7	2
	Malignant	0	5

Lampiran 16 Confusion Matrix VGG16 (K=5)

		Kelas Target	
		Benign	***Malignant***
Prediksi	***Benign***	3	0
	Malignant	4	7

Lampiran 17 Confusion Matrix VGG19 (K=1)

		Kelas Target	
		Benign	***Malignant***
Prediksi	***Benign***	6	5
	Malignant	1	2

Lampiran 18 Confusion Matrix VGG19 (K=2)

		Kelas Target	
		Benign	***Malignant***
Prediksi	***Benign***	7	1
	Malignant	0	6

Lampiran 19 Confusion Matrix VGG19 (K=3)

		Kelas Target	
		Benign	***Malignant***
Prediksi	***Benign***	7	4
	Malignant	0	3

Lampiran 20 Confusion Matrix VGG19 (K=4)

		Kelas Target	
		Benign	***Malignant***
Prediksi	***Benign***	7	1
	Malignant	0	6

Lampiran 21 Confusion Matrix VGG19 (K=5)

		Kelas Target	
		Benign	***Malignant***
Prediksi	***Benign***	3	0
	Malignant	4	7

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Andreas Galang Anugerah, lahir di Blitar pada tanggal 13 November 1995. Pendidikan dasar hingga menengah diselesaikan di Blitar, kemudian pada tahun 2014 hingga saat ini melanjutkan pendidikan tinggi di Departemen Teknik Informatika ITS.

Penulis memiliki beberapa pengalaman organisasi dan kepanitiaan semasa kuliah diantaranya Staf Departemen Minat Bakat HMTC ITS (2015-2016), anggota NST Schematics ITS 2015 dan 2016, anggota Web Apps ITS Expo 2016. Penulis juga aktif sebagai administrator Laboratorium Komputasi Cerdas dan Visi Teknik Informatika ITS periode 2016 hingga 2017. Selain itu, penulis pernah meraih Medali Emas Pengembangan Aplikasi Permainan Gemastik 10 2017 di Universitas Indonesia. Penulis dapat dihubungi melalui andreas.galang1@gmail.com.s