TUGAS AKHIR – EC184801

KOMPUTASI PARALEL UNTUK MENDETEKSI GELOMBANG QRS COMPLEX MENGGUNAKAN YOLO DEEP LEARNING

Muhammad Achsan Hujjatul Islam
NRP 0721 16 4000 0018

Dosen Pembimbing
Arief Kurniawan, S.T., M.T.
Reza Fuad Rachmadi, S.T., M.T., Ph. D.

DEPARTEMEN TEKNIK KOMPUTER
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2021

Halaman ini sengaja dikosongkan.

TUGAS AKHIR – EC184801

KOMPUTASI PARALEL UNTUK MENDETEKSI GELOMBANG QRS COMPLEX MENGGUNAKAN YOLO DEEP LEARNING

Muhammad Achsan Hujjatul Islam
NRP 0721 16 4000 0018

Dosen Pembimbing
Arief Kurniawan, S.T., M.T.
Reza Fuad Rachmadi, S.T., M.T., Ph. D.

DEPARTEMEN TEKNIK KOMPUTER
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2021

Halaman ini sengaja dikosongkan.

FINAL PROJECT – EC184801

PARALLEL COMPUTING FOR QRS COMPLEX DETECTION USING YOLO DEEP LEARNING

Muhammad Achsan Hujjatul Islam
NRP 0721 16 4000 0018

Advisors
Arief Kurniawan, S.T., M.T.
Reza Fuad Rachmadi, S.T., M.T., Ph. D.

Department of Computer Engineering
Faculty of Intelligent Electrical and Informatics Technology
Sepuluh Nopember Institute of Technology
Surabaya 2021

Halaman ini sengaja dikosongkan.

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "**Komputasi Paralel untuk Mendeteksi Gelombang *QRS Complex* Menggunakan YOLO *Deep Learning***" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Februari 2021

Muhammad Achsan Hujjatul Islam
NRP. 07211640000018

Halaman ini sengaja dikosongkan.

LEMBAR PENGESAHAN

Komputasi Paralel untuk Mendeteksi Gelombang *QRS Complex* Menggunakan YOLO *Deep Learning*

Tugas Akhir ini disusun untuk memenuhi salah satu syarat memperoleh gelar Sarjana Teknik di Institut Teknologi Sepuluh Nopember Surabaya

Oleh : Muhammad Achsan Hujjatul Islam (NRP: 07211640000018)

Tanggal Ujian : 08 Februari 2021 Periode Wisuda : April 2021

Disetujui oleh:

Arief Kurniawan, S.T., M.T. (Pembimbing I)
NIP. 19740907 200212 1 001

Reza Fuad Rachmadi, S.T., M.T., Ph.D. (Pembimbing II)
NIP. 19850403 201212 1 001

Ahmad Zaini, S.T., M.Sc. (Penguji I)
NIP. 19750419 200212 1 003

Dr. Diah Puspito Wulandari, S.T.,M.Sc. (Penguji II)
NIP. 19801219 200501 2 001

Susi Juniastuti, S.T., M.Eng. (Penguji III)
NIP. 19650618 199903 2 001

Mengetahui
Kepala Departemen Teknik Komputer

Dr. Supeno Mardi Susiki Nugroho, ST., MT.
NIP. 19700313 199512 1 001

Halaman ini sengaja dikosongkan.

ABSTRAK

Nama Mahasiswa	:	Muhammad Achsan Hujjatul Islam
Judul Tugas Akhir	:	Komputasi Paralel untuk Mendeteksi Gelombang *QRS Complex* Menggunakan YOLO *Deep Learning*
Dosen Pembimbing	:	1. Arief Kurniawan, S.T., M.T. 2. Reza Fuad Rachmadi, S.T., M.T., Ph.D.

Penyakit jantung merupakan salah satu penyebab kematian terbesar di dunia. Berdasarkan Riset Kesehatan Dasar (Riskesdas) Kementerian Kesehatan, pada tahun 2018 prevalansi penyakit jantung di Indonesia mencapai 1.5% dari jumlah seluruh penduduk. Salah satu penyebab munculnya penyakit jantung dapat diketahui melalui kondisi ritme jantung. Aritmia merupakan suatu kelainan ritme jantung yang tidak beraturan, terlalu cepat atau terlalu lambat. Kelainan aritmia dapat dideteksi menggunakan Elektrokardiogram (EKG). Deteksi aritmia dengan menggunakan teknologi pada bidang machine learning sangat diperlukan, sehingga dapat mendeteksi dan dilakukan terapi/pengobatan aritmia sedini mungkin untuk mengurangi resiko. Saat ini telah berkembang cara pembacaan dalam jangka panjang dikenal *Long Term ECG*. Pembacaan ini menghasilkan banyak data, yang tentunya membutuhkan waktu yang lama dalam mengolahnya. Berdasarkan hal tersebut, maka dimanfaatkannya teknologi Komputasi Paralel yaitu pemrosesan komputasi secara bersamaan dengan memaksimalkan sumber daya perangkat yang dimiliki sehingga dapat mempersingkat waktu yang di butuhkan.

Kata Kunci : Aritmia, EKG, *Parallel Computing*.

Halaman ini sengaja dikosongkan.

ABSTRACT

Student Name	:	Muhammad Achsan Hujjatul Islam
Final Project Title	:	*Parallel Computing for QRS Complex Detection Using YOLO Deep Learning*
Advisors	:	1. Arief Kurniawan, S.T., M.T. 2. Reza Fuad Rachmadi, S.T., M.T., Ph.D.

Heart disease is one of the biggest causes of death in the world. Based on Riset Kesehatan Dasar (Riskesdas) Kementerian Kesehatan, in 2018 the prevalence of heart disease in Indonesia reached 1.5% of the total population. One of the causes of heart disease can be identified through a heart rhythm condition. Arrhythmia is a heart rhythm disorder that is irregular, too fast or too slow. Arrhythmia Abnormalities can be detected using an electrocardiogram. Arrhythmia detection using technology in the field of machine learning is necessary, so that arrhythmias can be treated and treated as early as possible to reduce the risk. Currently, a long-term reading method known as the Long-Term ECG has been developed. Electrocardiogram produces a lot of data, which of course takes a long time to process. Based on this, Parallel Computing technology is utilized. Parallel Computing is computational processing simultaneously by maximizing device resources so that it can shorten the time needed.

Keywords : *Arrhythmia*, *ECG*, *Parallel Computing*.

Halaman ini sengaja dikosongkan.

KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa atas segala rahmat-Nya, penulis dapat menyelesaikan tugas akhir ini dengan judul **Komputasi Paralel untuk Mendeteksi Gelombang *QRS Complex* Menggunakan YOLO *Deep Learning***. Penelitian ini disusun dalam rangka memenuhi salah satu persyaratan untuk memperoleh gelar sarjana di Departemen Teknik Komputer FTEIC ITS. Keberhasilan penulis dalam pelaksanaan penelitian ini tidak lepas dari bantuan banyak pihak yang terlibat. Oleh karena itu, penulis mengucapkan terima kasih kepada :

1. Bapak Dr. Supeno Mardi Susiki Nugroho, S.T., M.T. selaku Kepala Departemen Teknik Komputer, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember.
2. Bapak Arief Kurniawan, S.T., M.T. dan bapak Reza Fuad Rachmadi, S.T., M.T., Ph.D. selaku dosen pembimbing, atas dukungan dan bimbingan selama pengerjaan penelitian ini.
3. Bapak Ahmad Zaini, S.T., M.Sc., ibu Dr. Diah Puspito Wulandari, S.T.,M.Sc. dan ibu Susi Juniastuti, S.T.,M.Eng. selaku dosen penguji yang telah memberikan pengarahan dalam penulisan naskah Tugas Akhir.
4. Bapak Dr. Adhi Dharma Wibawa, S.T.,M.T. selaku dosen wali yang telah memberikan semangat dan masukan selama perkuliahan.
5. Kedua orang tua, beserta keluarga, dan saudara-saudari serta Uzlifatil Jannah tercinta yang selalu memberikan dukungan baik secara moral maupun material selama pelaksanaan penelitian ini.
6. Bapak-ibu dosen pengajar serta staff Departemen Teknik Komputer, atas pengajaran, bimbingan, serta perhatian yang diberikan kepada penulis selama ini.
7. Seluruh teman-teman dari Teknik Komputer dan juga angkatan e56.
8. Pihak lain yang telah ikut mendukung mendoakan dan membantu terselesaikannya naskah Skripsi ini, yang tidak dapat disebutkan satu-persatu.

Penulis menyadari bahwa dalam penyusunan buku hasil penelitian ini jauh dari kata sempurna. Oleh karena itu, Penulis mengharapkan kritik dan saran yang membangun dari pembaca untuk meningkatkan kualitas dan perbaikan lebih lanjut. Semoga naskah ini dapat memberikan manfaat bagi penulis dan pembaca.

Surabaya, Februari 2021

Penulis

DAFTAR ISI

Halaman ini sengaja dikosongkan.

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan.

DAFTAR TABEL

Halaman ini sengaja dikosongkan.

NOMENKLATUR

T_s	: Waktu eksekusi secara sekuensial
T_p	: Waktu eksekusi secara paralel
S_p	: Nilai kecepatan pada P-*processor*
E_p	: Nilai efisiensi pada P-*processor*
TP	: True Positive
TN	: True Negative
FP	: False Positive
FN	: False Negative

Halaman ini sengaja dikosongkan.

BAB 1
PENDAHULUAN

Penelitian ini di latar belakangi oleh berbagai kondisi yang menjadi acuan. Selain itu juga terdapat beberapa permasalahan yang akan dijawab sebagai luaran dari penelitian.

1.1 Latar Belakang

Penyakit jantung masih menjadi ancaman di Indonesia bahkan di dunia, badan Kementerian Kesehatan Indonesia menyimpulkan bahwa ada dua kondisi aritmia yang dialami oleh pasien yaitu 87% dari data pasien penyakit jantung yang berujung pada meninggal dunia secara mendadak telah menderita aritmia, kesimpulan kedua adalah 1 dari 6 penderita stroke disebabkan oleh aritmia [11]. Salah satu badan organisasi kesehatan dunia yaitu Global Burden of Disease (GBD) juga menyampaikan bahwa 33,5 juta orang memiliki kelainan Atrial Fibrilasi (AF) pada tahun 2010 yang telah mewakili 0,5% dari populasi dunia [12]. Atrial Fibrilasi merupakan salah satu kelainan dari Aritmia dengan irama denyut jantung cepat. Salah satu penyakit jantung yang dialami oleh pasien adalah Aritmia yaitu suatu gejala/kelainan detak jantung yang tidak teratur, ditandai dengan aktivitas jantung berdetak lebih cepat atau lebih lambat.

Teknologi pada bidang kesehatan telah menyediakan alat yang digunakan untuk mengetahui aritmia yaitu dengan melakukan perekaman aktivitas jantung. Alat perekaman ini sudah berkembang luas dan bervariasi, salah satunya adalah alat rekam EKG rawat jalan (ambulatory EKG) yaitu Holter Monitoring dengan minimal waktu perekaman 24 jam. Hasil perekaman dibutuhkan dokter dalam mendiagnosa kondisi pasien dengan menganalisa keadaan gelombang EKG yang dihasilkan.

Perkembangan teknologi juga mengarah pada media pemrosesan data yang dikerjakan oleh *processor* komputer. Perusahaan penyedia *processor* telah memberikan inovasi-inovasi dalam mengembangkan *processor* yang tersedia saat ini. *Processor* menjadi semakin lebih cepat dengan ukurannya yang semakin kecil. Adanya keterbatasan dalam pengemasan ukuran membuat fokusan teknolo-

gi pengembangan berpindah kepada cara mengkomunikasikan beberapa prosesor yang digunakan sehingga bekerja menjadi lebih efektif yaitu *Parallel Computing Architecture*.

Perkembangan teknologi pada era masa kini sangatlah pesat apalagi dibidang komputer, hal ini ditandai dengan hampir semua pengelola data dan informasi telah dilakukan dengan komputer. Banyak pekerjaan manusia yang tergantikan dengan komputer dikarenakan sistem yang ada di komputer lebih optimal dan lebih efisien serta menghemat waktu oleh karena itu peran komputer didunia saat ini sangatlah penting.

1.2 Permasalahan

Hasil sinyal EKG dengan waktu perekaman lama mempunyai data yang besar hal ini sebanding dengan waktu proses yang dibutuhkan, sumber daya perangkat juga mempengaruhi waktu eksekusi yang dibutuhkan.

1.3 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah memberikan laporan waktu-waktu terjadinya kelainan aktivitas jantung dengan waktu eksekusi lebih singkat melalui pemrosesan secara parallel.

1.4 Batasan Masalah

Untuk memfokuskan permasalahan yang diangkat maka dilakukan pembatasan masalah. Batasan-batasan masalah tersebut diantaranya adalah :

1. Sinyal EKG yang diolah adalah sinyal yang telah tersedia pada database MIT-BIH (tidak *realtime*)
2. Tidak membahas tentang kenyamanan yang dirasakan oleh pasien ketika pengambilan data menggunakan perangkat *Holter Monitoring*.
3. Prosesor yang digunakan adalah jenis prosesor CPU.
4. Data output berupa laporan waktu pada titik puncak aktifitas jantung.

1.5 Sistematika Penulisan

Laporan penelitian Tugas akhir ini tersusun dalam sistematika dan terstruktur sehingga mudah dipahami dan dipelajari oleh pembaca maupun seseorang yang ingin melanjutkan penelitian ini. Alur sistematika penulisan laporan penelitian ini yaitu:

1. BAB 1 Pendahuluan
   Bab I berisi uraian tentang latar belakang, permasalahan, tujuan, batasan masalah, metodologi, dan sistematika penulisan dari penelitian tugas akhir ini.
2. BAB 2 Tinjauan Pustaka
   Bab II berisi tentang uraian secara sistematika teori-teori yang berhubungan dengan permasalahan pada penelitian ini. Teori-teori ini digunakan sebagai dasar dalam penelitian, yaitu teori mengenai sinyal EKG, komputasi paralel beserta jenis model paralelnya, deteksi objek, YOLOv4, dan teori penunjang lain.
3. BAB 3 Perancangan Sistem dan Implementasi
   Bab III berisi tentang rancangan pemecahanan masalah beserta implementasinya. Berisikan mengenai perencanaan rancangan, uraian rinci mengenai metodologi yang digunakan, dan pemaparan hasil implementasi sistem yang dilakukan.
4. BAB 4 Pengujian dan Analisis
   Bab IV berisi tentang pengujian eksperimen yang telah dilakukan terhadap sistem dan parameter yang terlibat. Selain itu, bab ini juga memuat hasil dan analisis dari uji coba yang dilakukan pada Tugas Akhir ini.
5. BAB 5 Penutup
   BAB V berisi tentang kesimpulan yang diambil berdasarkan penelitian dan pengujian yang telah dilakukan. Saran dan kritik yang membangun untuk pengembangan lebih lanjut juga dituliskan pada bab ini.

Halaman ini sengaja dikosongkan.

BAB 2
TINJAUAN PUSTAKA

Demi mendukung penelitian ini, dibutuhkan beberapa teori penunjang sebagai bahan acuan dan referensi. Pada bagian ini, teori penunjang tersebut dijabarkan secara ringkas untuk menjadi dasar dalam menyelesaikan penelitian yang lebih terarah.

2.1 Dasar Teori

Pada sub bab ini merupakan bagian pemaparan dasar teori atau referensi dalam penelitian yang berkaitan dengan 3 topik utama yaitu sinyal EKG, komputasi, dan deteksi objek.

2.1.1 Sinyal ECG

Pada sub bagian sinyal EKG ini menjelaskan terkait beberapa sub bahasan yang saling berhubungan yaitu: jantung, sinyal EKG, dan perangkat holter.

2.1.1.1 Jantung

Jantung adalah organ terpenting dalam tubuh manusia dengan bentuk rongga dan berotot yang terletak diantara kedua paru-paru kanan dan kiri pada rongga dada serta bertumpu pada permukaan superior dari diafragma. Jantung memiliki peran dalam sistem peredaran darah yang berfungsi untuk memompa dan menyebarkan darah dengan mengangkut oksigen ke seluruh tubuh oleh kontraksi berirama yang berulang. Jantung memompa darah melewati dua sistem sirkulasi, darah itu sendiri membawa okisgen dan nutrisi dan juga membantu menghilangkan sisa-sisa metabolisme.

Pada saat jantung berdenyut maka setiap ruang jantung mengendur dan darah masuk ke jantung yang biasa disebut diastol lalu jantung berkontraksi dan memompa darah keluar dari ruang jantung atau disebut sistol. Kedua serambi pada jantung berkontraksi dan berelaksasi secara bersamaan begitu pula kedua bilik jantung yang bergerak mengendur dan berkontraksi secara bersamaan. Organ ini mempunyai ukuran sebesar kapalan tangan. dengan ukuran panjang berkisar 12 cm dengan lebar 8-9 cm dan tebal 6 cm. Ketika

beristirahat, jantung berdetak kurang lebih 72 kali per menit dan darah akan mengalir satu arah ke seluruh tubuh melalui pembuluh darah.

2.1.1.2 Kelistrikan jantung

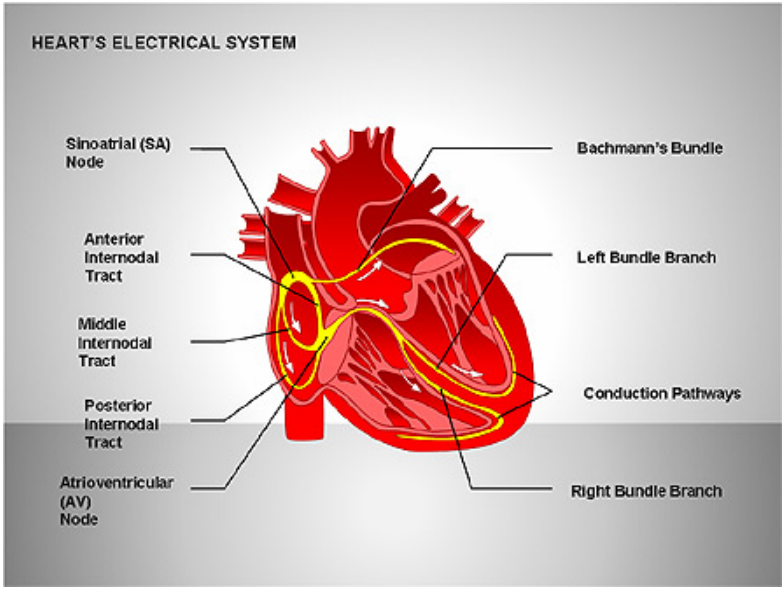
Jantung memiliki sistem kelistrikan yang membuatnya berdetak dengan ritme normal dan menyebabkab kontraksi sel otot jantung. Jantung akan berkontraksi secara ritmik, akibat adanya impuls listrik yang dibangkitkan oleh jantung sendiri. Setiap kali jantung berdenyut, sinyal listrik merambat dari jantung bagian atas ke bagian bawah dan pada saat sinyal ini berjalan, jantung berkontraksi dan memompa darah ke seluruh tubuh.

Hal ini terjadi oleh adanya potensial aksi yang dihantarkan sepanjang membran sel otot jantung. Potensial aksi pada membran saraf dan otot rangka dapat terjadi bila ada rangsangan dari luar sedangkan pada membran sel otot jantung potensial aksi dapat terjadi tanpa adanya rangsangan. Terdapat kumpulan sel-sel dalam jantung yang memiliki sifat menghasilkan impuls listrik secara spontan dan bertindak secara independen tanpa adanya koordinasi aktivitas listrik dan kontraksi otot. Sel-sel ini membuat dan mendistribusikan arus listrik yang mengarah ke kontraksi jantung yang terkendali dan juga efektif.

Gambar 2.1 menunjukkan sistem kelistrikan pada jantung beserta letak dari simpul SA yang berada pada serambi kanan. Sinyal listrik dimulai dari sel-sel sinoatrial (SA) node atau simpul SA yang ada di serambi kanan dan bisa dikatakan sebagai gardu pembangkit listrik. Pada kondisi normal, sinyal listrik dari simpul SA melambat begitu pula Ketika kita sedang beristirahat atau tidur, denyut jantungpun ikut melambat. Namun sebaliknya ketika kita sedang berolahraga, maka sinyal listrik akan meningkat dan jantung akan berdetak lebih cepat hal ini juga terjadi Ketika kita dalam situasi yang menegangkan.

2.1.1.3 Aritmia

Aritmia adalah kondisi tidak normalnya irama atau laju detak jantung yang berupa irama jantung berdetak terlalu cepat, terlalu lambat, atau ritmenya tidak teratur. Pemicu penyakit jantung ini



Gambar 2.1: Gambar sistem kelistrikan pada jantung [1]

adalah gangguan aliran impuls listrik ke sel-sel jantung miokardium, sinyal listrik yang bertugas mengatur koordinasi detak jantung tidak berfungsi dengan baik karena gangguan elektrik yang disebabkan oleh interupsi sinyal pada sistem kelistrikan jantung.

Aritmia sering kali tidak bergejala sehingga penderita aritmia tidak mengetahui bahwa sedang memiliki kelainan aritmia, sebenarnya seseorang dapat menyadarinya ketika mengalami sensasi hilangnya denyut jantung atau munculnya denyut jantung tambahan secara acak dalam waktu tertentu tanpa penyebab yang jelas.

Terdapat dua macam aritmia yang disebut *bradycardia* dan *tachycardia*. *Bradycardia* adalah keadaan denyut jantung pasien dalam kondisi sangat lambat yaitu ketika denyut jantung kurang dari 60 kali/menit sedangkan *tachycardia* adalah kondisi denyut jantung pasien yang berdetak sangat cepat lebih dari 100 kali/menit. Aritmia juga memiliki dua sifat yaitu ventrikular yang dimulai dari ventrikel jantung bawah dan *supraventricular* yang dimulai di luar

atau di atas ventrikel, biasanya di atrium, bilik jantung atas.

Dari faktor eksternal juga terdapat banyak penyebab timbulnya aritmia bisa dikarenakan stres, terkena infeksi hingga merasa demam dan juga dari pengobatan atau stimulan lainnya seperti obat-obatan terlarang maupun alkohol [13]. Aritmia juga dapat disebabkan oleh masalah struktural pada jantung atau dalam hal ini adalah cacat jantung bawaan dan juga bisa terjadi karena faktor genetik maupun usia.

Ketika impuls listrik yang berfungsi mengatur detak jantung tidak bekerja dengan baik maka pompa darah ke paru-paru, otak, dan organ lainnya menjadi tidak efektif yang akhirnya dapat mengakibatkan kerusakan pada organ. Kebanyakan jenis gangguan aritmia tidak berbahaya namun apabila gangguannya muncul akibat kondisi jantung yang lemah atau rusak maka bisa menyebabkan timbulnya gejala serius dan mengancam keselamatan serta komplikasi parah yang akan berujung kepada kematian.

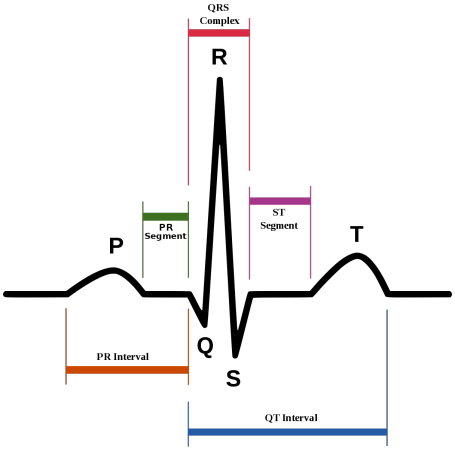
Gangguan irama jantung atau aritmia bisa terjadi pada siapa saja dan sebenarnya aritmia normal terjadi pada kondisi jantung yang sehat. Apabila aritmia terjadi terus menerus atau berulang secara frekuensi maka hal ini menandakan adanya masalah pada organ jantung. Jumlah kasus aritmia jantung di Indonesia mencapai jutaan dan jenis yang paling sering ditemukan adalah *fibrilasi atrium*.

2.1.1.4 Sinyal EKG

Jantung berdenyut dalam irama teratur dan dikendalikan oleh aktivitas listrik dalam tubuh. Node sinus dalam jantung memantau banyaknya darah yang dibutuhkan tubuh, dan mengirimkan impuls listrik yang menyebabkan bilik jantung berkontraksi dengan laju menyesuaikan dengan kondisi tubuh.

Jumlah detak jantung normal akan berbeda setiap orangnya, Asosiasi Jantung Amerika (AHA) menyebutkan bahwa detak jantung normal berada pada kisaran 60-100 bpm [14]. Seseorang akan kesulitan untuk menghitung detak jantungnya sendiri bahkan denyut nadi yang berada pada pergelangan tangan cenderung sulit untuk diraba. Namun gejala aritmia bisa di lihat melalui Elektrokardiogram atau bisa disebut dengan istilah EKG atau ECG dalam bahasa inggris. EKG dilakukan dengan merekam aktifitas listrik

yang berlangsung dalam jantung. Salah satu bagian alat/elektroda dari alat elektrokardiografi dipasang pada kulit manusia sebagai monitor adanya perubahan tegangan antara elektroda. Hasil keluaran EKG merupakan data sinyal rekaman aktifitas kelistrikan jantung berupa kurva tegangan fungsi waktu yang terdiri dari berbagai puncak. Gambar 2.2 menunjukkan data sinyal PQRST hasil rekaman EKG. Sinyal tersebut terdiri atas 1 gelombang P, 1 kompleks QRS dengan R sebagai titik puncak, dan 1 gelombang T.



Gambar 2.2: Sinyal PQRST pada sinyal EKG [2]

2.1.1.5 Perangkat Holter

Perekaman aktivitas jantung diberikan kepada pasien selama melakukan rutinitas setiap hari salah satunya dengan menggunakan perangkat holter. Perangkat Holter adalah salah satu sistem elektrokardiografi rawat jalan yang ditemukan oleh Dr. Norman J. Holter dan timnya pada tahun 1957. Dengan adanya pernagkat ini perkeman ritme jantung bisa dilakukan secara personal di luar rumah sakit, perangkat ini bekerja berdasarkan prinsip Galvanometer untuk merekam sinyal elektrokardiografik dari seseorang yang melakukan aktivitas sehari-hari, seperti elektrokardiografi rawat jalan berkelanjutan [15].

Sejak tahun 1961 berbagai modalitas dan perangkat canggih lainnya telah diproduksi untuk tujuan pengembangan ini sehingga bisa membuktikan kefektifan perangkat Holter dalam mendiagnosa dan memantau aritmia jantung. Teknologi terbaru bisa melakukan pemantauan irama jantung dalam jangka waktu lama dari beberapa hari hingga beberapa bulan, sehingga akan membantu untuk mendeteksi aritmia yang jarang terjadi pada pasien dari segala usia.

2.1.2 Komputasi Paralel

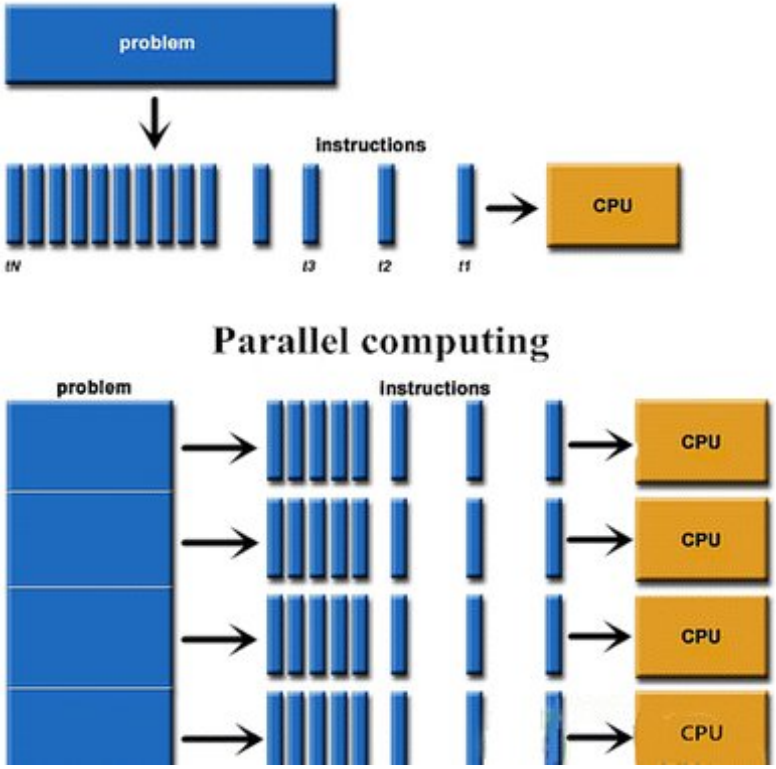
Pada sub bagian ini akan menjelaskan beberapa sub bahasan yang diawali dengan pengenalan tentang komputasi paralel kemudian membahas lebih dalam dari komputasi paralel beserta tipe komputasi paralel dan terakhir adalah evaluasi dari model paralel itu sendiri.

2.1.2.1 Komputasi Paralel

Komputasi paralel adalah metode komputasi untuk menyelesaikan permasalahan komputasi dengan membagi beban komputasi ke dalam beberapa bagian kecil sub proses komputasi. Sub komputasi tersebut dijalankan bersamaan pada prosesor yang berbeda secara independen dan saling berinteraksi satu sama lain [16]. Komputasi paralel menggunakan 2 atau lebih processor untuk meningkatkan performa komputasi dalam mengolah data besar/banyak [17].

Gambar 2.3 menunjukkan skema penyelesaian masalah menggunakan komputasi paralel dengan membagi data untuk dikerjakan secara bersamaan. Komputasi paralel dilakukan dengan membagi sebuah masalah besar ke dalam beberapa masalah kecil. Hal ini bertujuan untuk meningkatkan kinerja komputer dalam menyelesaikan berbagai masalah sehingga komputasi menjadi lebih maksimal dan mempersingkat waktu.

Implementasi komputasi paralel biasanya digunakan untuk menyelesaikan satu masalah seperti penyelesaian masalah komputasi, rendering dan sebagainya. Umumnya paralel diperlukan saat kapasitas komputasi yang diperlukan sangat besar, baik karena jumlah data yang besar maupun karena tuntutan proses komputasi yang banyak. Kasus lain juga diperlukan dalam komputasi numerik untuk



Gambar 2.3: Ilustrasi pembagian data untuk dikerjakan bersama secara paralel [3]

menyelesaikan persamaan matematis di bidang fisika (fisika komputasi), kimia (kimia komputasi), teknik sipil, dan bidang lainnya.

2.1.2.2 Model Paralel

Peningkatan kekuatan komputasi saat ini didapat dengan menggganti paradigma pemrograman menjadi pemrograman paralel dengan cara memecah proses menjadi sub-proses yang bisa memaksimalkan kapasitas core dan memory. Saat ini telah ada dua pen-

dekatan di bidang komputasi berkinerja tinggi (*High Performance Computing*) yaitu *supercomputer* dan *multicomputer* [18]. Era tahun 80-an sampai dengan pertengahan 90-an, komputasi tingkat lanjut sangat bergantung pada super komputer (supercomputer). Seiring dengan meningkatnya kebutuhan komputasi yang cepat dan naiknya harga peralatan super komputer, maka pengembangan aplikasi-aplikasi berbasis paralel termasuk komputasi paralel didalamnya semakin banyak dilakukan.

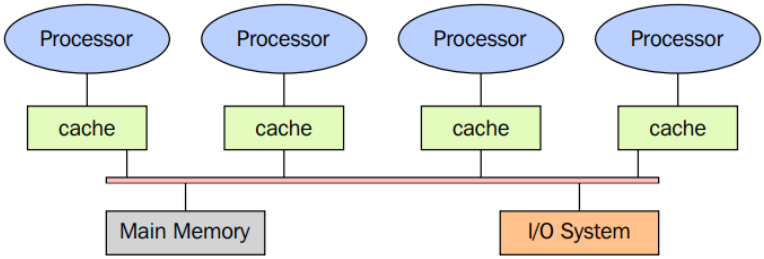
Dengan konsep komputasi paralel, biaya investasi untuk super komputer bisa ditekan dan konsumsi energi listrik, biaya instalasi serta pemeliharaan juga menjadi lebih rendah. Lebih penting lagi, sistem ini fleksibel terhadap perubahan teknologi komputer yang sangat cepat. Perkembangan teknologi komputasi paralel menjadi opsi dalam mengatasi permasalahan yang ada dan menjadi alternatif dibanding superkomputer yang dari segi harga tidak murah / terjangkau. Implementasi komputasi paralel mudah untuk dilakukan yang bisa dilakukan di semua mesin komputasi hanya dengan membangun infrastruktur yang dapat melakukan pemrosesan secara bersamaan di prosesor yang berbeda. Berdasarkan pembagian informasi -dalam hal ini adalah data yang digunakan untuk pemrosesan- model paralel dapat dibedakan menjadi 2 yaitu [4]:

1. *Shared Memory*

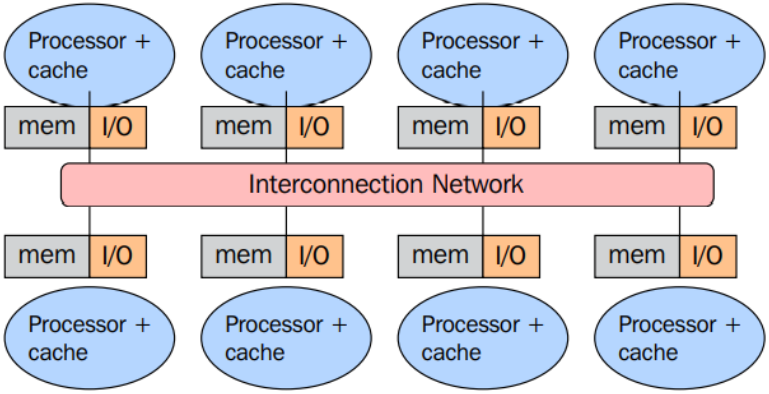
   Model paralel yang informasinya berada pada *memory* yang sama begitu pula ketika mengakses *memory* baik membaca maupun menulis data, hal tersebut menggunakan sumber daya dari perangkat yang sama seperti yang diilustrasikan oleh gambar 2.4. Model ini memberikan kemudahan bagi seorang programmer dengan tidak membuat lajur komunikasi antar tugas untuk semua prosesornya.

2. *Distributed Memory/Message Passing*

   Gambar 2.5 merupakan ilustrasi dari model *Distributed Memory* dengan menggunakan *memory* berbeda yang dimiliki oleh masing-masing prosesornya dan akan bertanggung jawab terhadap memory secara mandiri. Setiap prosesor hanya bisa mengalamatkan data pada *memory* yang dimilikinya karena *local memory* hanya bisa diakses oleh pemiliknya. Untuk melakukan pertukaran data, model menggunakan protocol mes-



Gambar 2.4: Skema arsitektur paralel dengan model *Shared Memory* [4]



Gambar 2.5: Skema arsitektur paralel dengan model *Distributed Memory* [4]

sage passing yang disalurkan melalui jaringan komunikasi dan antar mesin harus terhubung pada jaringan tersebut. Pembagian tugas bisa dibagikan ke prosesor baik yang masih berada dalam satu mesin yang sama maupun mesin berbeda yang masih saling terhubung.

Tidak ada model pemrograman paralel yang paling bagus diantara ke empat jenis tersebut, masing-masing model memiliki kelebihan tersendiri pada bidangnya. Penggunaan model yang sesuai

akan memberikan hasil baik dan efisien dalam menyelesaikan permasalahan komputasi yang digunakan beserta dengan sumber daya yang dimilikinya.

2.1.2.3 Evaluasi Model Paralel

Pengembangan *paralel programming* diciptakan berfokus untuk menyelesaikan permasalahan besar dengan waktu yang singkat. Menilai keberhasilan dari penerapan paralel dalam menyelesaikan permasalahan tersebut ditunjukkan dari perbandingan waktu eksekusi antara pemrosesan program secara sekuensial dengan secara paralel dan juga menghitung jumlah prosesor yang digunakan untuk menilai efisiensi dari penggunaan sumber daya perangkat secara paralel tersebut. Hal ini mengacu pada algoritma paralel yang diterapkan, model paralel yang dipilih maupun kemampuan perangkat yang digunakan serta permasalahan yang diangkat untuk diselesaikan. Namun, teknologi komputasi paralel juga masih memiliki keterbatasan dalam melakukan komputasi yang dijelaskan pada persamaan *Amdahl's Law* dan *Gustafson's Law* [4].

A *Speed Up*

Perbandingan antara waktu eksekusi algoritma sekuensial tercepat dengan waktu eksekusi algoritma paralel, atau waktu eksekusi algoritma paralel pada satu prosesor dengan waktu eksekusi algoritma tersebut pada sejumlah prosesor. Secara matematis speedup dinyatakan dengan Persamaan 2.1.

$$S_p = \frac{T_s}{T_p} \qquad ;1 \leq S_p \leq p \tag{2.1}$$

Secara ideal *speedup* meningkat sebanding dengan bertambahnya jumlah prosesor, jika digunakan **p** prosesor, speedup idealnya adalah p, namun bisa juga terjadi kasus dengan kondisi superlinear speedup seperti yang terdapat pada keterangan dalam persamaan 2.1. *Superlinear speedup* ini tidak menunjukkan peningkatan kinerja yang sebenarnya, karena peningkatan kecepatan yang terjadi diakibatkan oleh fitur lain dari arsitektur paralel, misalnya ukuran cache yang lebih besar pada lingkup pemrograman paralel dibandingkan dengan

ukuran cache pada lingkup pemrograman sekuensial [19, 20].

B *Efficiency*

Suatu ukuran kinerja yang memiliki hubungan erat dengan speed up, Efisiensi yaitu indikasi derajat sebenarnya dari performa kecepatan yang dicapai dibanding dengan nilai maksimum. Efisiensi rendah terjadi pada kasus ketika seluruh kode program yang dimasukkan dieksekusi secara berurutan pada satu prosesor sedangkan efisiensi maksimum dicapai Ketika semua n-prosesor secara penuh digunakan untuk periode eksekusi. Secara matematis efisiensi dapat dinyatakan dengan persamaan 2.2.

$$E = \frac{S_p}{p} \qquad ; \frac{1}{p} \leq E \leq 1 \tag{2.2}$$

C *Amdahl's Law*

Penurunan *speedup* disebabkan karena faktor serial dari program (seperti ketergantungan data, waktu pemuatan program dan penyumbatan / *bottlenecks* I/O), biaya tambahan (*overhead*) sinkronisasi dan komunikasi. Amdahl's (1967) mempertimbangkan faktor serial dari program dengan hukum Amdahl's sebagai berikut:

$$S = \frac{1}{1-p} \tag{2.3}$$

Nilai 1-p pada persamaan diatas adalah eksekusi serial pada program yang tidak bisa dikerjakan secara parallel.

D *Gustafson's Law*

Gustafson melihat adanya kekurangan dari Amdahl's Law sebelumnya dan menambahkan cara terbaik dalam menetapkan waktu dari permasalahan paralel juga bergantung pada kemampuan sumber daya komputasi dan jumlah informasi / datanya. *Speedup* dan efisiensi akan meningkat jika ukuran data juga ditingkatkan. Gustafson berpendapat berdasarkan dua pertimbangan berikut:

1. Ketika ada peningkatan dimensi dari ukuran permasalahan, maka algoritma program yang sekuensial juga akan

memiliki nilai secara konstan.

2. Ketika ada peningkatan jumlah prosesor, maka tugas yang dikerjakan oleh setiap prosesor masih tetap sama.

Dari penjelasan tersebut sehingga didapatkan hukum Gustafson sebagai berikut:

$$S = P - \alpha(P - 1) \tag{2.4}$$

2.1.3 Deteksi Objek

Pada sub bagian deteksi objek ini menjelaskan terkait beberapa sub bahasan yang saling berhubungan yaitu: deteksi objek secara umum, YOLOv4 sebagai salah satu metode deteksi objek dan evaluasi deteksi objek dari YOLOv4.

2.1.3.1 Deteksi Objek

Deteksi Objek adalah teknologi komputer yang masih memiliki keterkaitan dengan visi komputer dan pemrosesan gambar seperti mengidentifikasi dan menemukan objek dari kelas tertentu pada suatu gambar. Metode deteksi objek dapat dilakukan dengan berbagai cara, salah satunya dengan membuat kotak pembatas di sekitar objek atau dengan menandai setiap piksel pada gambar yang berisi objek tersebut atau biasa disebut dengan segmentasi.

Perkembangan deteksi objek beriringan dengan perkembangan *convolutional networks* yang telah dimulai pada tahun 1998 oleh LeCun et al. *Convolutional Neural Network* (CNN) bisa dikatakan sebagai pokok dasar dari visi komputer di era pembelajaran mesin saat ini. CNN sendiri merupakan salah satu jenis *neural network* yang digunakan pada data gambar untuk mendeteksi dan mengenali objek pada sebuah gambar.

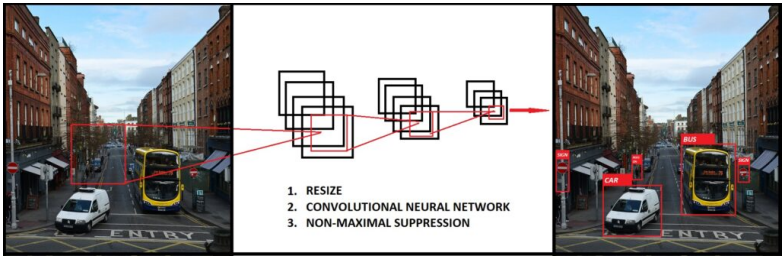
CNN memanfaatkan proses konvolusi dengan menggerakkan sebuah kernel konvolusi (filter) berukuran tertentu ke sebuah gambar sehingga komputer bisa mendapatkan informasi representatif baru dari hasil perkalian bagian gambar tersebut dengan filter yang digunakan. Konvolusi merupakan operasi matematika antara dua matriks untuk menghasilkan matriks ketiga. Matriks keluaran dapat mengenali pola spesifik yang ada pada gambar masukan bergantung pada angka dalam matriks filter. Angka-angka dalam filter

dipelajari oleh jaringan saraf dan polanya diturunkan dengan sendirinya.

Algoritma deteksi objek memanfaatkan pembelajaran mesin (*machine learning*) atau pembelajaran mendalam (*deep learning*) untuk melakukan pemrosesan gambar dalam mendeteksi objeknya. Saat kita melihat gambar atau video, kita dapat mengenali dan menemukan objek yang menarik dalam sekejap. Tujuan dari deteksi objek adalah untuk mereplikasi kecerdasan visual tersebut menggunakan komputer. Saat ini, sudah ada banyak model untuk deteksi objek (YOLO, RCNN, Fast RCNN, Mask RCNN, Multibox, dll.) bahkan setiap tahun selalu muncul algoritma / model baru yang terus dikembangkan dan diimprovisasi.

2.1.3.2 YOLOv4

Seiring perkembangan sistem pengenalan objek visual, terdapat beberapa metode yang menjadikan proses pendeteksian sebagai masalah regresi, dua metode yang paling populer adalah YOLO dan SSD [21]. *You Only Look Once* atau yang disingkat YOLO adalah suatu metode pendeteksi objek pada citra yang di dasari oleh CNN. Berbeda dengan sistem pendeteksi objek lainnya yang memiliki proses kompleks dalam melakukan pendeteksian, YOLO menggantikan keseluruhan proses tersebut dengan menggunakan *single neural network* sehingga dapat mendeteksi objek dengan sangat cepat dan akurat.



Gambar 2.6: Sistem YOLO [5]

Secara umum, proses pendeteksian yang dilakukan oleh YOLO terbagi ke dalam tiga langkah utama yang ditunjukkan oleh Gambar 2.6. YOLO mendeteksi objek dengan menggunakan *unified model*

yaitu *single convolutional network* yang memprediksi beberapa kotak pembatas (*bounding boxes*) serta probabilitas dari kotak-kotak tersebut secara bersamaan. Pertama-tama, YOLO membagi citra input ke dalam grid S x S. Jika pusat dari sebuah objek jatuh di dalam salah satu sel grid, maka sel grid itu bertanggung jawab untuk mendeteksi objek tersebut. Setiap sel grid memprediksi kotak pembatas dan *confidence score* dari tiap kotak pembatas tersebut [22].

Confidence score merefleksikan seberapa yakin dan akurat model bahwa terdapat sebuah objek di dalam kotak tersebut. Setiap kotak pembatas terdiri dari 5 prediksi: x, y, w , h, dan *confidence*. Koordinat (x, y) mewakili pusat dari kotak relatif ke batas sel grid. (w, h) atau lebar dan tinggi mewakili pusat dari kotak relatif ke gambar. Dan terakhir adalah *confidence* yang mewakili *Intersection over Union* (IoU) antara kotak prediksi dan kotak *ground-truth*. Probabilitas dikondisikan pada sel grid yang memuat objek dan hanya satu kelas probabilitas yang dideteksi per sel grid tanpa memperhitungkan jumlah kotak pembatas.

YOLO sudah memiliki beberapa versi dalam pengembangannya yaitu YOLOv1, YOLOv2, YOLOv3 dan yang digunakan dalam pengujian tugas akhir ini yaitu YOLOv4 yang dirilis pada tanggal 24 April 2020 oleh 3 orang antara lain: Alexey Bochkovskiy, Bersama Chien-Yao Wang, and Hong-Yuan Mark Liao yang membangun YOLO pada versi Windows nya. YOLOv4 merancang operasi pendeteksian objek yang cepat dan optimisasi untuk komputasi parallel, hal inilah yang membuat algoritma YOLO mengalami peningkatan dalam hal kecepatan dan performa [23].

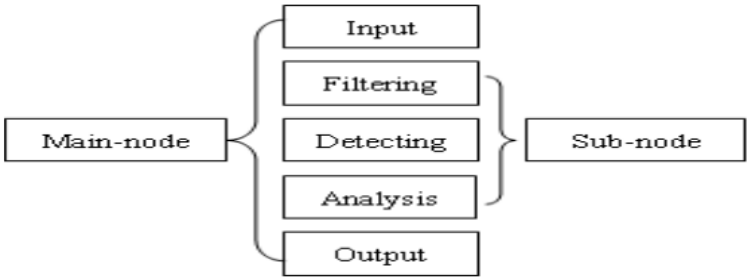
2.2 Artikel Terkait

Pada sub bab ini dijelaskan mengenai beberapa penelitian yang berelasi dengan tugas akhir ini diantaranya *Detection and Analysis of ECG Based on paralel Computing Technology*, *Paralel Processing Architecture for ECG Signal Analysis* dan *Automatic Detection of Arrhythmias Using a YOLO-Based Network with Long-Duration ECG Signals*.

2.2.1 *Detection and Analysis of ECG Based on paralel Computing Technology*

Penelitian ini menjelaskan model paralel untuk deteksi dan analisis elektrokardiogram (EKG) dengan menggunakan arsitektur Message Passing Interface (MPI) yaitu dari Cluster IBM dengan 15 node. Dalam memodelkan algoritma paralel, penulis menganalisa algoritma serial dari deteksi dan analisis EKG yang selanjutnya setiap algoritma serial tersebut diperlakukan secara paralel untuk digunakan pada arsitektur paralel menggunakan MPI [6].

Skema yang digunakan pada penelitian ini seperti ditunjukkan pada gambar 2.7, terdiri dari 5 modul yaitu Input Module, Filtering Module, Detecting Module, Analysis Module dan Output Module. Pada tiga bagian dari lima module tersebut dapat dikerjakan oleh sub node secara parallel. Sistem menggunakan teknik komputasi paralel untuk menyelesaikan pemrosesan dan analisis data yang besar.



Gambar 2.7: Skema model paralel yang digunakan [6]

Hasil percobaan yang dilakukan penulis berhasil memberikan efisiensi waktu dalam menganalisa EKG dengan nilai yang meningkat. Tabel 2.1 merupakan hasil percobaan yang dilakukan oleh penulis dan telah ditunjukkan bahwa semua parallel yang dilakukan membutuhkan waktu yang lebih singkat dibandingkan mengeksekusi secara sekuensial.

Tabel 2.1: Hasil percobaan komputasi [6]

Node	1	2			3		
	Ts	Tp	Sp	Ep	Tp	Sp	Ep
30 min	224	184.9	1.21	0.61	158.9	1.41	0.47
1 hour	445.3	369.8	1.20	0.60	320.3	1.39	0.46
6 hour	2714.8	2214.8	1.23	0.62	1623.7	1.67	0.56
24 hour	10854.2	8846.4	1.23	0.62	6410.2	1.69	0.56

2.2.2 *Paralel Processing Architecture for ECG Signal Analysis*

Pada makalah penelitian ini merancang salah satu platform yang fleksibel untuk analisa EKG yang fleksibel, platform ini dibangun untuk kebutuhan komputasi yang lebih cepat, kebutuhan sistem secara *real-time*, dan persyaratan lainnya. Perkembangan teknologi telah menciptakan berbagai teknologi yang memadai sebagai resolusi untuk kebutuhan *real-time* tersebut.

Studi lebih lanjut lagi tentang cara menganalisa EKG beserta ruang lingkup didalamnya yang digabungkan dengan menciptakan platform yang fleksibel dan dapat diperluas/diskalakan di kemudian hari. Platform dirancang dengan arsitektur *pipeline* dengan adanya prosesor tambahan yang bertugas melakukan *compute the thresholds, selection of the thresholds, tuning parameters, decision algorithms* dll yang dibutuhkan oleh blok fungsional dari arsitektur *pipeline* tersebut [24].

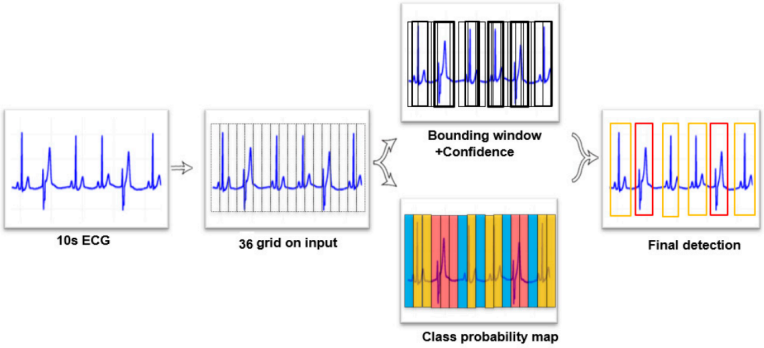
2.2.3 *Automatic Detection of Arrhythmias Using a YOLO-Based Network with Long-Duration ECG Signals*

Penelitian ini berkaitan dengan salah satu metode *machine learning* dalam mendeteksi objek yaitu YOLO, hasil model klasifikasi aritmia berbasis YOLO dapat mendeteksi masing-masing detak jantung dan mengklasifikasikan lima jenis aritmia dalam segmen EKG 10-detik tanpa langkah *beat extraction* dengan menggabungk-

an langkah *beat extraction*, *feature extraction*, dan klasifikasi menjadi satu module.

Secara umum, proses deteksi aritmia dilakukan dalam tiga langkah yang terdiri dari *beat segmentation*, *feature extraction*, dan klasifikasi. *Beat segmentation* adalah proses deteksi puncak-R dalam sinyal EKG, *Feature extraction* adalah langkah dalam beat segmentation dengan menggunakan *machine learning*. Metode berbasis deep learning mengatasi masalah dengan pembelajaran end-to-end yang *Feature extraction* dan klasifikasinya digabungkan menjadi satu. Namun, metode berbasis *deep learning* ini membutuhkan proses segmentasi yang cepat.

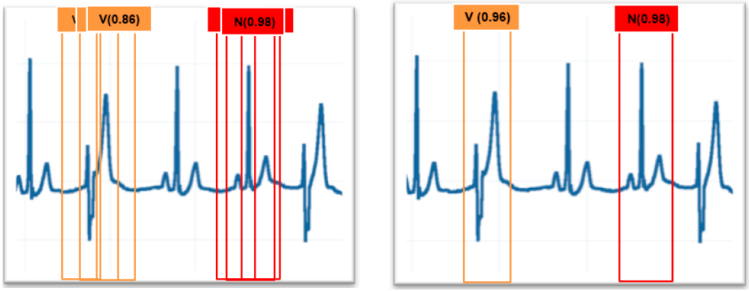
Penulis mengusulkan model YOLO 1D yang dapat mendeteksi aritmia seperti yang ditunjukkan oleh gambar 2.8, model tersebut dapat mendeteksi lebih dari satu jenis aritmia dalam urutan yang panjang. Model YOLO dimodifikasi menjadi model YOLO 1D dan mengganti kotak pembatas dengan jendela pembatas dan pada akhirnya model hanya memprediksi satu koordinat untuk membuat jendela pembatas.



Gambar 2.8: Bentuk YOLO1D menggunakan bounding window [7]

Hasil pengujian menunjukkan bahwa algoritma ini memiliki kinerja bagus yaitu mempunyai kecepatan tinggi dan nilai presisi rata-rata bagus dalam mendeteksi aritmia. Selain itu, jendela pembatas dapat memprediksi panjang jendela yang berbeda pada berbagai jenis aritmia hal ini dikarenakan model ini dapat memilih

panjang jendela pembatas yang optimal, pemilihan jendela pembatas ini dijelaskan pada gambar 2.9.



(**a**) (**b**)

Gambar 2.9: Pemilihan bounding window optimal [7]

BAB 3
DESAIN DAN IMPLEMENTASI SISTEM

Penelitian ini dilaksanakan sesuai dengan desain sistem serta implementasinya. Desain sistem merupakan konsep dari pembuatan dan perancangan infrastruktur kemudian diwujudkan dalam bentuk blok-blok alur yang harus dikerjakan. Implementasi merupakan pelaksanaan teknis untuk setiap blok pada desain sistem.

3.1 Gambaran Umum

Penelitian ini bertujuan untuk mendapatkan titik puncak-R pada data rekaman jantung dari seorang pasien yang dilaporkan berupa laporan aktifitas jantung dengan memberikan waktu-waktu kejadian titik puncak-R tersebut. Sistem ini dimulai dari input berupa data ECG, sebagai percobaan menggunakan data dari *MIT-BIH Arrhytmia Database*. Kemudian dari data tersebut dilakukan preprocessing data meliputi *split data*, mengubah data menjadi bentuk *spectogram*, selanjutnya dataset di-*training* untuk didapatkan *weight file*nya.

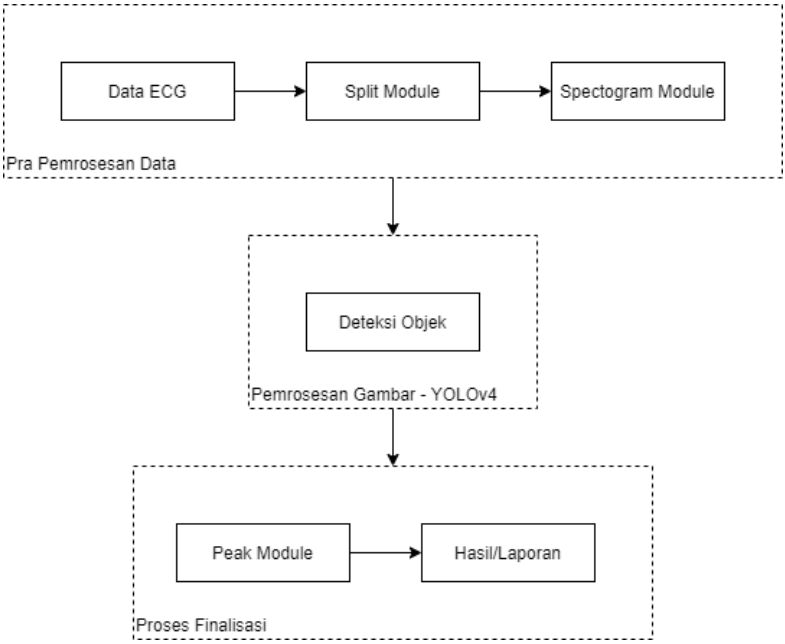
Setelah mendapatkan hasil model *training -weight file-*, model tersebut digunakan untuk mendeteksi objek PQRST sinyal ECG. Dan pada proses akhir menjalankan proses pencarian puncak R pada grafik PQRST dan memberikan hasil laporan waktu kejadian.

3.2 Desain Sistem

Pada sistem ini pengguna -dalam hal ini adalah pasien/perawat- dapat mengaplikasikan program dengan memberikan *input* data ECG yang nantinya akan diproses dan dikomputasi oleh sistem untuk didapatkan laporan waktu-waktu kejadian aktivitas jantung yang nantinya bisa dijadikan acuan dalam pengklasifikasian aritmia dari data ECG tersebut maupun analisa lainnya. Data ECG harus sesuai dengan format yang disediakan yang didalamnya memiliki data waktu dan data aktifitas jantung.

Alur kerja dari sistem ini dijelaskan melalui blok diagram pada gambar 3.1, terdapat 3 proses utama yang membangun tujuan

dari penelitian ini, yaitu *preprocessing data* mengenai pengelompokan kondisi mata, kemudian deteksi objek dengan menggunakan YOLOv4, tahap terakhir adalah pencarian nilai puncak dari hasil prediksi deteksi objek dan menyusunnya dalam bentuk laporan waktu.



Gambar 3.1: Blok Diagram utama sistem

3.3 *Preprocessing Dataset*

Pada tahapan ini, diperlukan *dataset* ECG untuk melakukan deteksi objek pada proses selanjutnya. Dataset merupakan kumpulan data sesuai dengan isi database tunggal atau matriks data statistik tunggal. Setiap kolom dan baris pada tabel menunjukkan parameter tertentu, dimana kolom tabel mewakili variabel dan baris tabel menunjukkan anggota tertentu dari data. Kumpulan data tersebut mencantumkan nilai untuk masing-masing variabel, seper-

ti halnya pada data ECG ini yang memiliki waktu dan nilai sinyal serta data lainnya, untuk setiap baris data. Setiap nilai dikenal sebagai datum. Kumpulan data dapat terdiri dari data untuk satu atau lebih anggota, sesuai dengan jumlah baris.

Pada tugas akhir ini, *dataset* yang digunakan berupa data aktivitas sinyal ECG yang didapat dari *MIT-BIH Arrhytmia Database* (https://physionet.org). Pada database tersebut terdapat 48 rekaman sinyal ECG yang diambil dari 48 pasien. Setiap rekaman memiliki frekuensi 360 Hz dengan durasi rekaman rata-rata selama 30 menit (648000 data).

Agar data dapat diolah menggunakan metode YOLOv4 untuk mendeteksi objek *peak*-nya, maka dibutuhkan data berupa gambar. Data gambar yang digunakan adalah gambar spektogram yang dihasilkan dari proses *Spectogram Module*. Spektogram merepresentasikan visual dari spektrum frekuensi sinyal terhadap waktu dan diketahui bahwa kurang lebih terdapat 1 puncak QRS yang terjadi setiap 1 detiknya.

Sehingga, diperlukan pemotongan sinyal terlebih dahulu yaitu pemotongan data rekaman raw ECG menjadi beberapa bagian data berukuran kecil dengan hasil akhir setiap bagian datanya membawa rekaman aktivitas jantung selama 1 detik (360 data rekaman). Pemotongan 360 data rekaman ini dilakukan dengan menggunakan acuan detik dari datum waktu yang tersedia pada dataset.

Setiap *Machine Learning* diperlukan pembagian dataset terlebih dahulu dan pada tugas akhir ini dataset dibagi menjadi 2 bagian, yaitu data untuk *training model* sebanyak 80%, data *validation/testing model* sebanyak 20%. Kemudian dataset tersebut di *training* menggunakan YOLOv4 dan didapatkan keluaran berupa *weight* yang digunakan untuk proses *testing* dalam mendeteksi objek puncak -QRS pada grafik Sinyal ECG-.

3.3.1 *Split Module*

Berdasarkan data rekaman yang diambil data raw EKG yang dihasilkan sangat bervariasi bahkan terdapat rekaman dengan waktu rekaman selama 24 jam atau lebih. Data rekaman yang terlalu besar ini tentunya akan memberatkan komputasi bergitu pula memory yang ada pada perangkat apabila dilakukan proses secara langsung dan bersamaan. Sehingga dibutuhkan proses pemecahan

data yang berfungsi untuk membagi-bagi data raw tersebut menjadi sub bagian-sub bagian lebih kecil. Selain itu pemecahan data ini juga bertujuan untuk menyesuaikan data masukan pada proses deteksi objek menggunakan YOLO di proses selanjutnya.

Proses split dataset menjadi beberapa bagian data dengan ukuran yang lebih kecil ditunjukkan oleh flowchart pada lampiran 1.1 yang setiap data keluarannya akan berisikan rekaman selama 1 detik.

Selama proses *Split Module* ini, data yang digunakan tidak dirubah baik menambahkan/mengurangi nilai pada data rekaman aktifitas jantung yang ada, namun hanya saja membagi-bagi datanya menjadi data yang lebih kecil dan menyimpannya menjadi file baru. Pemecahan data dilakukan berdasarkan waktu dengan rentang 1 detik sehingga setiap data nya membawa 360 data aktifitas jantung.

Jenis pemotongan data selama 1 detik ini terbagi menjadi 2 yaitu, jenis pertama adalah data hasil pemotongan dari pemotongan murni detik datum waktu dan jenis kedua diperoleh dari penggabungan setengah terakhir data detik sebelumnya ditambahkan dengan setengah pertama dari data detik setelahnya sehingga didapatkan data dengan panjang 1 detik pula.

Sebagai contoh terdapat data dengan label ecg-100, memiliki data rekaman aktifitas jantung selama 30 menit selanjutnya dilakukan pemecahan data dan akan terbentuk file baru ecg-100 (menit ke 0, detik ke 0-1), ecg-100 (menit ke 0, detik ke 1-2), dan seterusnya sampai menit detik terakhir dari sample ecg-100. Sedangkan untuk file pemotongan jenis kedua terbentuk dari file ECG-100 (menit ke 0, detik ke 0.5-1.5), ecg-100 (menit ke 0, detik ke 1.5-2.5), dan seterusnya sampai menit detik terakhir dari sample ecg-100.

Setiap masukan file data raw EKG akan dibaca setiap barisnya mulai dari baris pertama sampai baris terakhir. Pada setiap baris tersebut membawa data waktu dan data waktu itulah yang dijadikan acuan dalam penamaan file dan tentunya untuk proses *Split Module* ini. Apabila ketika membaca baris baru dan ditemukan perbedaan data waktu -nilai detiknya- maka ada akan membuat file baru dengan nama file baru.

Untuk penamaan file baru, format yang digunakan adalah -menit-detik.extension- sebagai contoh 00-00.csv merupakan pena-

maan file ECG pada menit pertama detik ke-0. Penamaan ini bertujuan untuk mempermudah proses *tracking* apabila terjadi *trouble/error* selama eksekusi proses, maka bisa mengambil tindakan langsung terhadap filenya dan tidak perlu melakukan proses split ulang dari awal.

3.3.2 *Spectogram Module*

Hasil rekaman aktifitas jantung terdapat kemungkinan *noise* pada sinyal EKG yang didapatkan sehingga diperlukan perlakuan khusus dalam melakukan pengolahan sinyal EKG tersebut salah satunya yaitu dengan pendekatan STFT (*Short-Time Fourier Transform*) yang masih merupakan pengembangan dari FFT (*Fast Fourier Transform*) yaitu salah satu algoritma yang digunakan untuk pengolahan sinyal. Pendekatan melalui STFT ini diharapkan bisa mengoptimalkan nilai akurasi yang dihasilkan oleh sistem namun ternyata juga akan memberikan efek samping yaitu kebutuhan konsumsi waktu yang besar.

Metode yang terjadi pada STFT adalah membagi sinyal waktu yang lebih lama menjadi segmen yang lebih pendek dengan panjang yang sama dan kemudian menghitung transformasi Fourier secara terpisah pada setiap segmen yang lebih pendek [25]. Hal ini diterapkan pada penelitian ini dengan memecah sinyal EKG dalam bentuk satuan detik yang telah dihasilkan oleh proses *Split Module* dan selanjutnya spektrum tersebut diplot ke dalam bentuk spektogram.

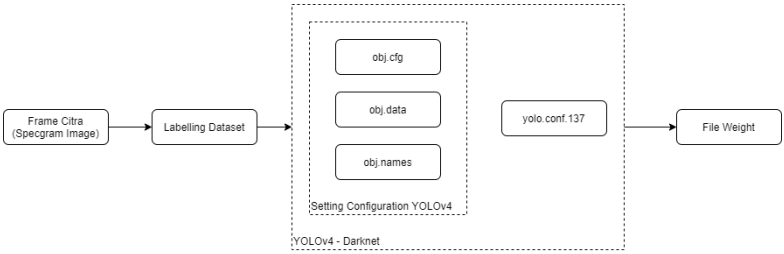
Gambar 3.2: Blok Diagram *Spectogram Module*

Spektogram sendiri merupakan representasi visual dari spektrum frekuensi sinyal terhadap waktu yang direpresentasikan dalam bentuk berupa gambar grafik. Proses pembuatan spektogram ditunjukkan oleh gambar 3.2 yang diawali dengan membaca data masukan

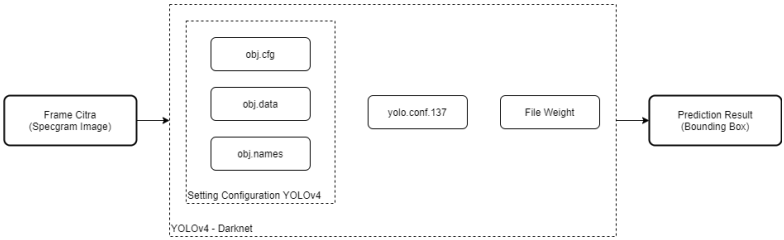
yaitu file baru yang tercipta dari proses *Split Module* dan keluarannya akan di simpan menjadi file gambar berupa spektogram.

3.4 Deteksi Objek

Proses deteksi objek terbagi menjadi 2 bagian proses yaitu *Training YOLOv4* yang bertujuan untuk mendapatkan *file weight/model Machine Learning* dan *Testing YOLOv4* yang digunakan untuk menghasilkan hasil prediksi deteksi objeknya. Dua proses dari YOLOv4 ini seperti digambarkan pada gambar 3.3.



(a) Blok Diagram Training YOLOv4



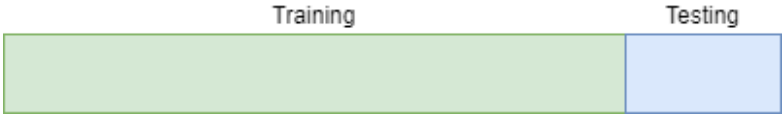
(b) Blok Diagram Testing YOLOv4

Gambar 3.3: Blok Diagram YOLOv4

Terlihat pada gambar 3.3a, keluaran yang dihasilkan berupa file weight yang diperlukan ketika menjalankan proses *Testing YOLOv4* yang ditunjukkan juga pada gambar 3.3b bahwa dalam lingkup YOLOv4-nya terdapat parameter file weight.

Sebelum menjalankan YOLOv4 terdapat tahapan preprocessing data yang diawali dengan pembagian dataset terlebih dahulu yang ditunjukkan pada gambar 3.4. Pada kasus algoritma YOLOv4

pembagian dataset hanya menjadi 2 *subset* yaitu *training set* dan *validation set* atau *testing set*, pembagian dataset ini berbeda dengan *Machine learning* pada umumnya, terdapat pemisahan *validation set* dengan *testing set*.



(a) Pembagian dataset pada algoritma YOLOv4

(b) Bentuk lain pembagian dataset pada algoritma *machine learning*

Gambar 3.4: Pembagian dataset pada algoritma *Machine Learning*

1. Training Set

   Subset dari dataset yang digunakan untuk melatih model. Pada umumnya, diberikan 80% bagian dari jumlah keseluruhan dataset sebagai *training set*.

2. Validation Set

   Subset yang terpisah dari *training set* yang digunakan dalam melakukan validasi untuk mengevaluasi hasil dari *training set*. Dengan kata lain *validation set* bisa dikatakan sebagai pengujian tahap awal dari model yang dihasilkan proses training. Hal ini dilakukan dengan cara menjadikan *validation set* sebagai *testing set* untuk memeriksa kembali evaluasi tersebut setelah model telah lulus atau melalui tahapan validasi. Pada umumnya, *validation set* diberikan 10% bagian dari jumlah keseluruhan dataset.

3. Testing Set

   Subset dari dataset yang digunakan untuk menguji sebuah model setelah model tersebut melalui tahapan validasi oleh *validation set*.

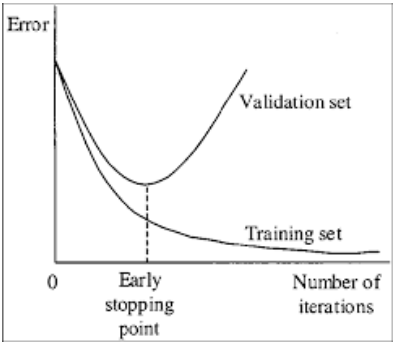
Preprocessing data pada YOLOv4 ini merupakan tahapan opsional yang dilakukan selama pengujian untuk mendapatkan file weightnya, apabila sudah mempunyai file weight maka tidak diperlukan pembagian dataset dan proses deteksi objek bisa langsung ke tahapan *Testing YOLOv4*.

3.4.1 Training YOLOv4

Setelah dataset terbagi selanjutnya dataset tersebut akan melalui tahapan *labelling data* yaitu pemberian label objek peak pada spektogram. Setiap spektogram pada dataset akan ditandai dengan kotak pembatas yang dinamakan *bounding box* untuk dijadikan sebagai pembelajaran model ketika proses *training YOLOv4* dan akan menghasilkan keluaran model training berupa *weight-file*.

Sesuai dengan dokumentasi yang dituliskan oleh Alexey AB, jumlah iterasi selama training adalah kelipatan 2000 sesuai dengan jumlah kelas yang digunakan namun tidak boleh kurang dari jumlah dataset training dan minimal sebanyak 6000 iterasi [8]. Sistem dalam tugas akhir ini kelas yang ingin dicari hanyalah 1 kelas yaitu objek peak sehingga konfigurasi untuk jumlah iterasi pada training YOLO sebesar 6000 iterasi.

Dalam dokumentasi juga dijelaskan bahwa adanya proses pemberhentian secara paksa. Pemberhentian iterasi dilakukan untuk mencegah *overfitting* terhadap model yang telah dijelaskan pada dokumentasi dan ditunjukkan oleh gambar 3.5.



Gambar 3.5: Grafik *Early Stop* [8]

Tahapan training tidak harus dilakukan setiap kali menjalankan sistem, namun sebagai gantinya harus mempunyai file weight yang diperoleh dari training sebelumnya. Setelah file weight didapatkan maka setiap kali deteksi objek bisa langsung melakukan tahapan *Testing YOLOv4* dengan file weight tersebut. File weight juga perlu di update sesuai kebutuhan dengan melihat nilai evaluasi dari model *Machine Learning* atau dalam hal ini adalah file weight yang dihasilkan tersebut.

3.4.2 Testing YOLOv4

Selanjutnya adalah tahapan testing dengan menggunakan hasil pembelajaran dari keluaran oleh tahapan training yang berupa file weight. File weight ini digunakan sebagai acuan saat melakukan deteksi objek, file weight ini juga menunjukkan kualitas dari algoritma YOLOv4 dengan model training yang telah dibuat sebelumnya.

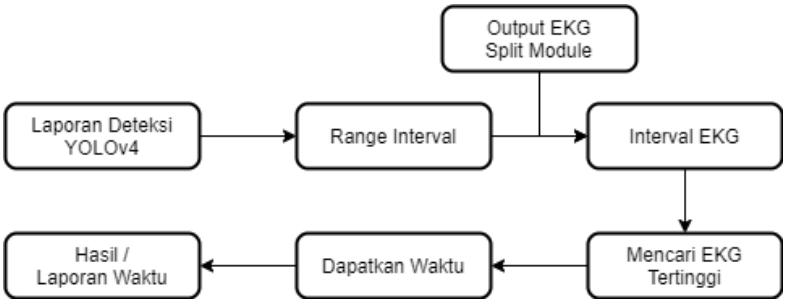
3.5 Laporan data

Setelah proses deteksi objek selesai dijalankan dan menghasilkan keluaran deteksi objek yang disimpan dalam bentuk file text. File keluaran dari proses YOLOv4 ini tidak hanya berisikan hasil deteksi objek saja melainkan juga memberikan keterangan lainnya termasuk metadata dari setiap masukan gambar, sehingga diperlukan filter untuk mendapatkan data yang diinginkan.

Laporan data ini merupakan hasil akhir dari sistem yang berisikan waktu-waktu yang menjadi titik puncak dari aktifitas jantung. Untuk bisa mendapatkan laporan waktu yang sesuai maka diperlukan proses lagi terhadap hasil keluaran deteksi YOLO.

3.5.1 *Peak Module*

Proses ini bertujuan untuk mendapatkan nilai peak sebenarnya dan membuat laporan dengan menyimpan data dalam bentuk file sesuai dengan EKG nya. Gambar 3.6 merupakan alur proses yang terjadi pada *Peak Module*, jadi hasil keluaran deteksi objek menghasilkan nilai prediksi deteksi objek yang ditandai dengan titik *bounding box*, selanjutnya titik-titik ini digunakan sebagai batasan dalam penentuan peak-R. Jadi hasil keluaran ini akan dikombinasik-



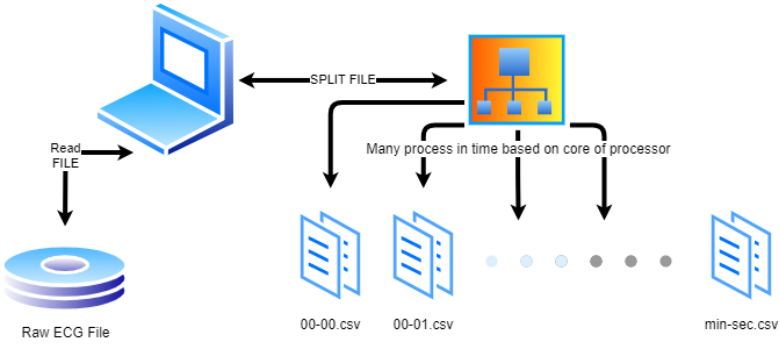
Gambar 3.6: Blok diagram *Peak Module*

an dengan data keluaran dari proses *Split Module* sebagai data yang membawa rekaman aktivitas jantung selanjutnya dilakukan pencarian nilai tertinggi dengan batasan dari *bounding box* tadi sehingga bisa didapatkan nilai peak sebenarnya dari data rekaman aktivitas jantung tersebut.

3.6 *Paralel Model*

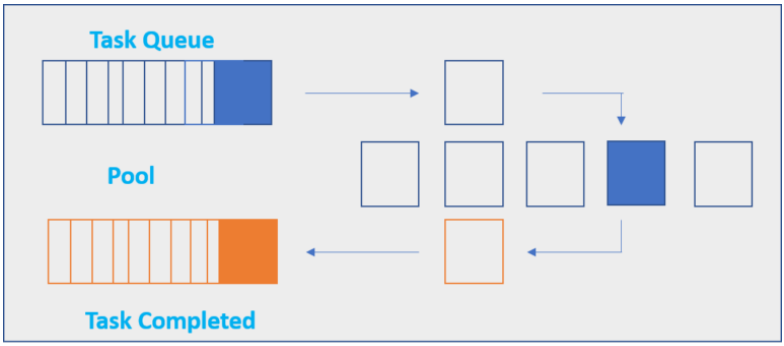
Untuk mempercepat waktu eksekusinya terutama karena menggunakan metode spektogram yang memberikan efek samping yaitu waktu eksekusi komputasi akan menjadi lebih lama dalam menjalankan proses perubahan data menjadi spektogram maka sistem pada tugas akhir ini menerapkan komputasi secara paralel. Komputasi paralel yang diterapkan pada penelitian ini seperti diilustrasikan pada gambar 3.7. Terlihat pada gambar tersebut bahwa banyak data yang diproses secara bersamaan bergantung pada banyaknya prosesor yang tersedia.

Semua algoritma pada rangkaian sistem ini bisa dijalankan bersamaan secara paralel dengan model paralel yang diterapkan menggunakan sistem *Pool*. *Pool* membutuhkan *manager* untuk mengatur data dan *worker* selama eksekusi berlangsung, hal ini dilakukan dengan cara mengontrol data dan membagikan data tersebut kepada *Pool Worker* untuk diproses oleh *Pool Worker* masing-masing. Secara otomatis yang bertugas sebagai *Pool Manager* adalah core utama yaitu prosesor yang digunakan saat program dieksekusi sedangkan *Pool Worker*-nya adalah sisa prosessor lainnya dan selama



Gambar 3.7: Ilustrasi *Paralel Model*

menjalankan program *Pool Manager* juga ikut serta dalam mengeksekusi prosesnya bergantung dengan pembagian yang diterapkan.

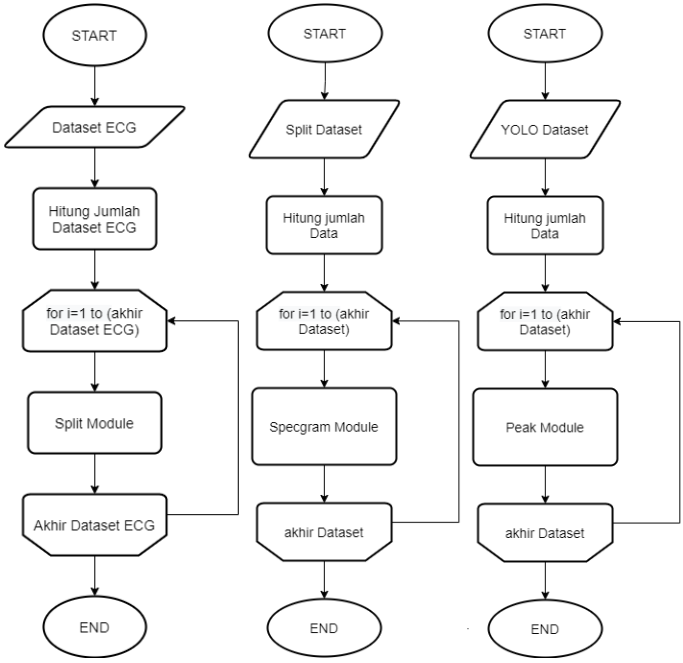


Gambar 3.8: Ilustrasi *Pool Management* [9]

Gambar 3.8 menunjukkan bahwa data yang diproses pada pool tersusun pada sebuah wadah yang dinamakan *task queue*. Sistem *pool* membutuhkan parameter data masukan berupa daftar yang bersifat bisa diiterasikan ketika hanya terdapat 1 data maka data tersebut harus diubah ke bentuk daftar yang bisa dilakukan dengan cara menambahkan data ke dalam daftar. *Pool* menerapkan penjadwalan FIFO (*First In First Out*) sehingga hanya akan membebankan memory dengan memberikan data yang dieksekusi pada

saat itu saja.

Pool Manager mengontrol data dengan menyalurkannya kepada *Pool Worker* dan akan mengumpulkan hasil komputasi dari para *Pool Worker*-nya. Data yang dikelola merupakan daftar pekerjaan yang harus diselesaikan, pool juga menggabungkan hasil komputasi yang diterima dari worker untuk disatukan kembali. Selama proses berlangsung eksekusi utama akan diblok sampai semua komputasi telah selesai dieksekusi dan data telah dikumpulkan kembali oleh manajer.



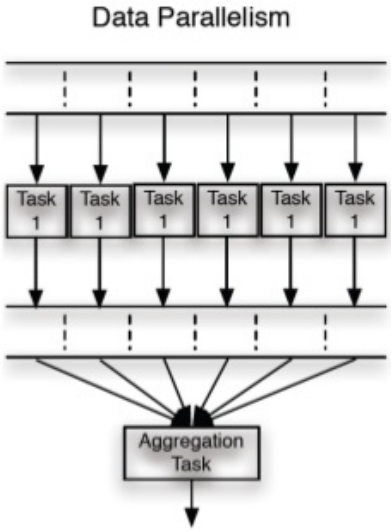
(a) *Flowchart Split Module* **(b)** *Flowchart Spectogram Module* **(c)** *Flowchart Peak Module*

Gambar 3.9: *Flowchart* dari 3 sub bagian utama pada sistem

Pada sub bagian utama *Split Module* dan *Spec Module* begitu pula *Peak Module* yang ditunjukkan oleh *flowchart* pada gambar

3.9. Dari semua flowchart tersebut memiliki kesamaan yaitu terdapat tahapan pengumpulan data dengan menghitung jumlah data masukan sebelum di eksekusi. Dalam hal ini bertujuan untuk memberikan daftar iterasi kepada pool untuk dilakukan proses secara paralel.

Task Queue yang didapatkan oleh *Pool* merupakan daftar data yang akan di proses yaitu banyaknya data masukan di setiap proses eksekusinya dan setiap prosesornya akan menjalankan tugas eksekusi program yang sama, sehingga bisa dikatakan bahwa metode paralel yang diterapkan menggunakan *data parallelism* yang diilustrasikan pada gambar 3.10.



Gambar 3.10: Ilustrasi *Data Parallelism* [10]

Halaman ini sengaja dikosongkan.

BAB 4
PENGUJIAN DAN ANALISIS

Pada bab ini menguraikan hasil pengujian serta analisa dari pengujian terhadap sistem yang telah dirancang pada Tugas Akhir ini. Pengujian ini dilakukan untuk menguji kemampuan sistem yang telah dibuat dalam menjawab permasalahan yang diangkat. Pengujian yang dilakukan adalah mengaplikasikan semua proses dan mendapatkan hasil laporan waktu kejadian. Pengujian dilakukan untuk mendapatkan analisa yang terbagi menjadi 2 bagian utama antara lain :

1. Analisa efektifitas dan efisiensi sistem
2. Analisa hasil laporan waktu titik puncak-R

Selama pengujian dalam menjalankan program, spesifikasi sumber daya perangkat yang digunakan seperti ditunjukkan pada tabel 4.1.

Tabel 4.1: Spesifikasi *Personal Computer (PC)*

HARDWARE SPECIFICATIONS	
GPU	NVIDIA® GeForce® 940MX (2 GB GDDR5 VRAM)
CPU	Intel® CoreTM i3-6006U processor (3 MB L3 cache, 2.00 GHz)
Core Processor	2 core with HyperThread (Total 4 core)
RAM	12 GB (4GB + 8 GB)
OS	Linux - Ubuntu 20.04 LTS (Focal Fossa)
Storage	WD Blue™ SATA SSD 2.5" – 500 GB

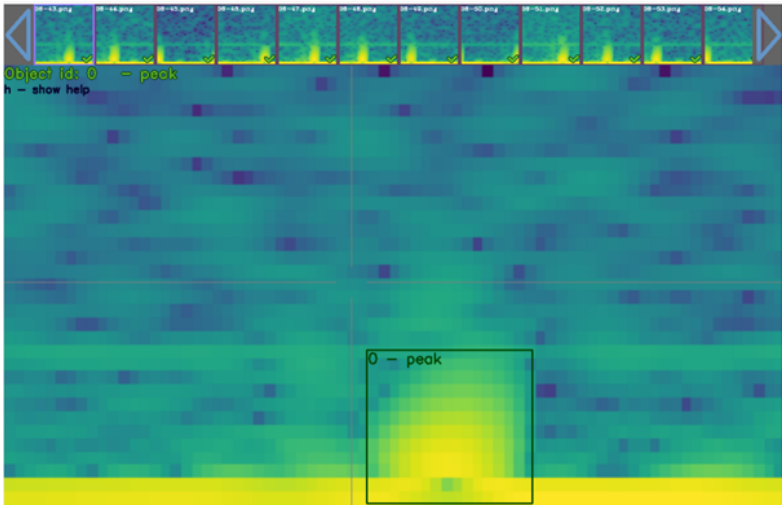
4.1 Pengujian

Sebelum dilakukan pengujian, dilakukan proses training YOLOv4 terlebih dahulu untuk mendapatkan model.

4.1.1 Konfigurasi *Weight-File* YOLO

Sebelum pengujian berlangsung, dilakukan proses training terhadap YOLO Module untuk mendapatkan *weight-file* nya. Pada tugas akhir ini menggunakan rekaman jantung 30 detik pertama dari 48 pasien sehingga jumlah dataset ± 30(*2(*48)) atau setara dengan ± 3000 dataset. Kemudian dataset tersebut dibagi menjadi 2 bagian yaitu *training set* dan *testing set* dan untuk komposisi *training dataset* adalah 80% atau sebanyak ± 2.400 data dan sisanya menjadi bagian dari *testing dataset* yaitu 20% atau sebanyak ± 600 data.

Selanjutnya dilakukan proses *data labelling* menggunakan aplikasi YOLO_mark [26], *data labelling* adalah proses pemberian label -dalam hal ini adalah kelas objek peak- kotak pembatas kepada dataset gambar spektogram seperti ditunjukkan pada gambar 4.1.



Gambar 4.1: Contoh proses *data labelling* menggunakan YOLO_mark

Pada bagian akhir dilakukan tahapan evaluasi terhadap *weight-file* hasil training model tersebut dan didapatkan laporan evaluasi model yang ditunjukkan pada tabel 4.2.

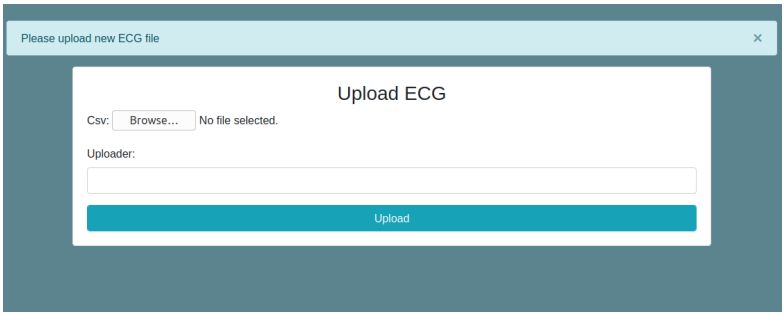
Tabel 4.2: Evaluasi YOLO terhadap *training model*

	Prescision	Recall	F1 Score	mAP (%)
yolov4-obj_1000.weights	0.63	0.91	0.74	83.57
yolov4-obj_2000.weights	0.82	0.95	0.88	95.01
yolov4-obj_3000.weights	0.84	0.95	0.89	93.51
yolov4-obj_4000.weights	0.81	0.92	0.86	88.49
yolov4-obj_last.weights	0.81	0.9	0.85	88.75
yolov4-obj_best.weights	0.83	0.95	0.89	93.8

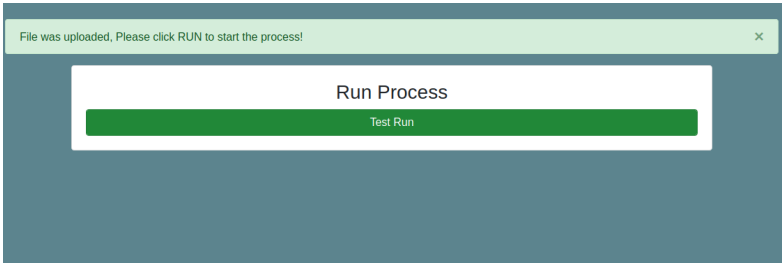
4.1.2 *Input Data*

Pengujian dilakukan dengan memberikan *input data* rekaman ECG ke dalam sistem untuk diproses. Pada gambar 4.2a terdapat tombol *upload* untuk mengunggah file ke dalam sistem dan gambar 4.2b *Run* untuk melakukan eksekusi program. Jadi data EKG yang akan dieksekusi haruslah diunggah ke dalam sistem melalui *web interface* dan data EKG ini harus berformat csv, setelah proses mengunggah file selesai diperlukan untuk menekan tombol eksekusi program sebagai tanda dalam menjalankan sistem.

Data masukan berupa file ECG yang di unggah ke sistem melalui web tersebut akan dieksekusi secara *default* proses eksekusi dijalankan secara paralel sesuai dengan jumlah inti prosesor yang dimiliki perangkat dan akan dieksekusi ketika tombol eksekusi ditekan. Berdasarkan spesifikasi yang digunakan dalam pengerjaan tugas akhir ini yaitu dengan menggunakan prosesor Intel Core i3 dengan jumlah inti 2 *core* dan mempunyai kemampuan hyper-thread maka eksekusi program akan dijalankan secara paralel dengan 4 inti



(a) Tampilan halaman unggah file pada Web



(b) Tampilan halaman eksekusi program pada Web

Gambar 4.2: Tampilan halaman pada Web

prosesor.

Namun selama pengujian tugas akhir ini, agar bisa menganalisa efektifitas dan efisiensi sistem maka eksekusi program diatur secara manual. Konfigurasi program di atur terlebih dahulu dengan mengubah pengaturan konfiguraasi eksekusi ke mode sekuensial / mode paralel, dalam konfigurasi mode paralel juga membutuhkan parameter lagi yaitu inti prosesor yang mau digunakan semisal 2 inti, 3 inti atau 4 inti. Memasukkan parameter jumlah inti prosesor yang berbeda dilakukan untuk menilai pengaruh dari komputasi paralel dalam menjalankan sistem.

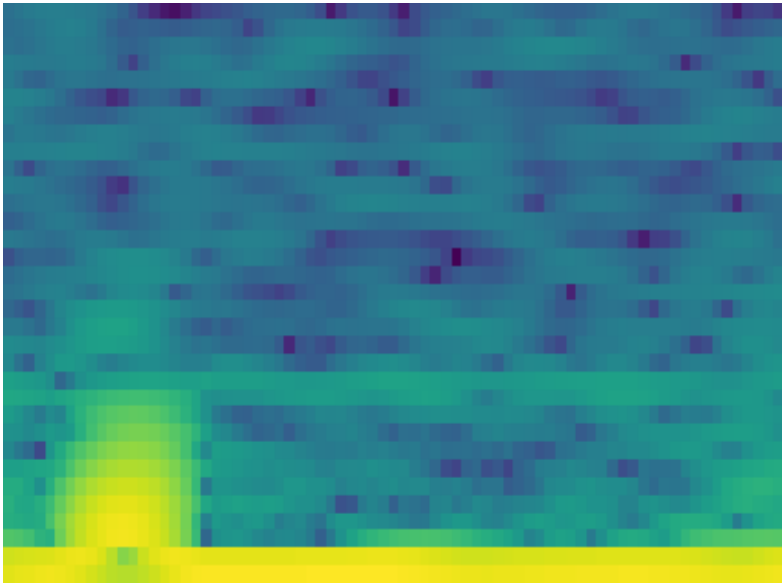
Selama program di eksekusi yang dimulai dengan menekan tombol eksekusi pada web, eksekusi program akan berjalan di background process sehingga pengguna bisa mengeluarkan aplikasi browser maupun bisa meninggalkan perangkat komputer. Meskipun ek-

sekusi program berjalan di background process, perangkat komputer harus masih dalam keadaan hidup dan tidak boleh ada gangguan sinyal PID dari eksekusi program selama program masih berjalan.

4.1.3 *Split Module*

Setelah file berhasil di unggah, pertama-tama file akan di salin ke folder proses sebagai tanda bahwa sedang dilakukan proses terhadap data tersebut. Selanjutnya, file akan melalui sub proses pertama yaitu *Split Module* dan akan menghasilkan file-file berukuran kecil yang ditunjukkan oleh tangkapan layar pada lampiran 1.2.

4.1.4 *Spectogram Module*

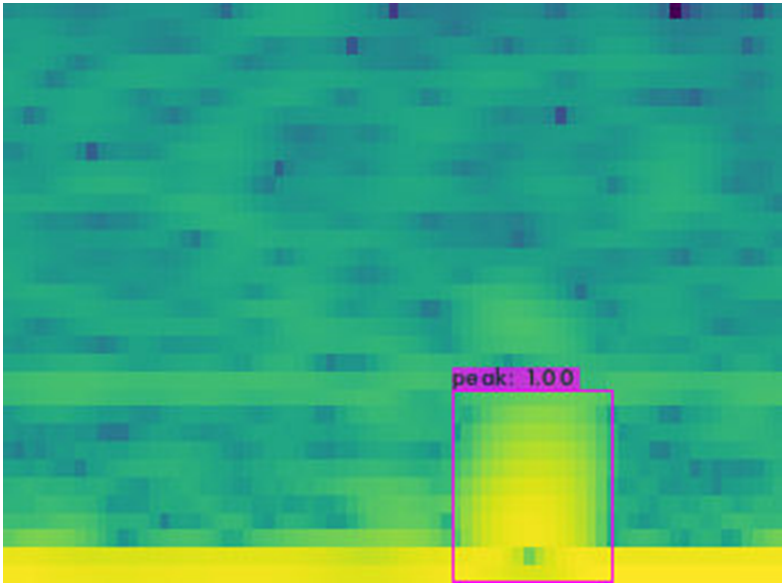


Gambar 4.3: Bentuk Spektogram dari data ECG-100 detik ke 0 sampai detik ke 1

Spectogram Module merupakan proses pengubahan data rekaman jantung berupa teks menjadi file gambar berupa spektogram.

Spectogram Module akan berjalan setelah proses *Split Module* selesai dilaksanakan karena pengubahan ke bentuk spektogram memerlukan data masukan yang bersumber dari hasil *Split Module*. Setiap data teks hasil *split module* akan diubah ke bentuk spektogram seperti yang ditunjukkan pada gambar 4.3 yaitu bentuk spektogram hasil dari pengubahan data file *Split Module* menjadi data gambar spektogram.

4.1.5 *YOLO Module*

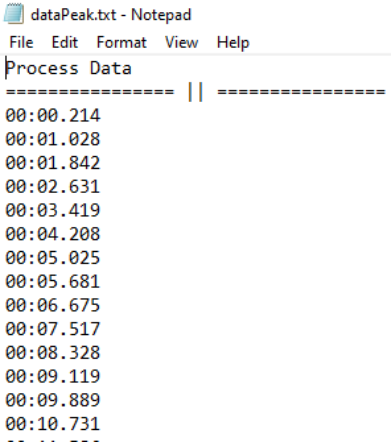


Gambar 4.4: Hasil deteksi objek peak menggunakan YOLOv4

Proses YOLO akan berjalan setelah proses pada spectogram module selesai dijalankan dan data telah diubah ke dalam bentuk gambar, karena proses pada YOLO Module hanya bisa dilakukan dengan masukan data berupa gambar. Weight-file hasil training akan dimuat lalu algoritma YOLO akan mendeteksi objek dengan memberikan bounding box dan menuliskan confidence score dari bounding box tersebut. Gambar 4.4 merupakan salah satu contoh

hasil deteksi objek menggunakan YOLOv4 yang telah berhasil melakukan deteksi objek ditandai dengan adanya *bounding box* dan juga nilai *confidence score* sebesar 1.00.

4.1.6 *Peak Module*



Gambar 4.5: Hasil laporan waktu setelah melakukan proses *Peak Module*

Hasil deteksi YOLO juga memberikan keluaran berupa file teks yang berisikan laporan hasil deteksi, lalu dilakukan proses filter untuk diambil nilai kotak pembatas khususnya nilai X dan Width pada laporan tersebut dan nantinya akan di kombinasikan dengan data hasil Split Module. Dan hasil akhirnya akan didapatkan hasil laporan waktu kejadian saat mencapai peak-R pada sinyal EKG seperti yang ditunjukkan pada gambar 4.5.

4.2 Hasil Pengujian

Pengujian dilakukan beberapa kali dengan input file ecg yang bervariasi baik dari sisi jumlah isi data dalam sebuah filenya maupun dari jumlah file yang dijalankan. Pengujian yang dilakukan dibagi menjadi beberapa bagian yaitu sebagai berikut:

1. Eksekusi program dengan masukan file ecg yang berisikan data

rekaman selama 10 menit.
2. Eksekusi program dengan masukan 2 file ecg yang berisikan data rekaman selama 10 menit.
3. Eksekusi program dengan masukan 3 file ecg yang berisikan data rekaman selama 10 menit.
4. Eksekusi program dengan masukan 4 file ecg yang berisikan data rekaman selama 10 menit.
5. Eksekusi program dengan masukan file ecg yang berisikan data rekaman selama 30 menit.

4.2.1 Pengujian dengan Input File ECG 10 Menit

Pengujian pertama dilakukan dengan sebagian data rekaman jantung yaitu data rekaman pada 10 menit pertama. Tabel 4.3 merupakan hasil perolehan waktu eksekusi oleh program baik secara sekuensial maupun paralel.

Tabel 4.3: Hasil Pengujian pertama menggunakan file ECG 10 menit

Sekuensial	Waktu yang dibutuhkan setiap *module*				
Paralel	Split	Spectogram	YOLO	Peak	Total
Sekuensial	00m 01d	01m 35d	09m 46d	00m 01d	11m 23d
2 Core	00m 02d	01m 09d	08m 55d	00m 01d	10m 07d
3 Core	00m 01d	00m 44d	09m 53d	00m 01d	10m 39d
4 Core	00m 01d	00m 42d	09m 16d	00m 00d	09m 59d

Pada table 4.3 menunjukkan bahwa proses pada *Split Module* dan *Peak Module* hanya membutuhkan waktu yang relatif singkat yaitu kurang dari 10 detik. Terdapat perbedaan waktu eksekusi saat memproses *Spectogram Module* yaitu eksekusi secara sekuensial membutuhkan waktu eksekusi selama 1 menit 35 detik, pada eksekusi secara paralel dengan 2 inti prosesor membutuhkan waktu eksekusi selama 1 menit 9 detik, pada eksekusi secara paralel dengan 3 inti prosesor membutuhkan waktu eksekusi selama 44 detik,

pada eksekusi secara paralel dengan 4 inti prosesor membutuhkan waktu eksekusi selama 42 detik.

Table 4.3 juga menunjukkan bahwa eksekusi program secara sekuensial membutuhkan waktu total selama 11 menit 23 detik, pada eksekusi program secara paralel dengan 2 inti membutuhkan waktu total selama 10 menit 7 detik sedangkan eksekusi program secara paralel dengan 3 inti membutuhkan waktu total selama 10 menit 39 detik dan eksekusi program secara paralel dengan 4 inti membutuhkan waktu total selama 9 menit 59 detik. Dari hasil pengujian pertama dapat diketahui bahwa waktu eksekusi program secara paralel lebih cepat dibandingkan dengan eksekusi secara sekuensial dan semakin banyak jumlah inti prosesor yang digunakan maka semakin cepat waktu eksekusi yang dibutuhkan.

4.2.2 Pengujian dengan Input 2 File ECG 10 Menit

Pengujian kedua dilakukan dengan menggunakan data input yang sama seperti percobaan pertama, namun jumlah file tersebut sebanyak dua file dalam sekali program dieksekusi.

Tabel 4.4: Hasil Pengujian kedua menggunakan 2 file ECG 10 menit

Sequence paralel	Waktu yang dibutuhkan setiap *module*				
	Split	Spectogram	YOLO	Peak	Total
Sekuensial	00m 02d	03m 09d	19m 35d	00m 01d	22m 47d
2 Core	00m 01d	02m 14d	19m 09d	00m 02d	21m 26d
3 Core	00m 01d	01m 27d	19m 53d	00m 01d	21m 22d
4 Core	00m 01d	01m 16d	19m 57d	00m 01d	21m 15d

Pada table 4.4 menunjukkan bahwa proses pada *Split Module* dan *Peak Module* hanya membutuhkan waktu yang relatif singkat yaitu kurang dari 10 detik. Terdapat perbedaan waktu eksekusi saat memproses *Spectogram Module* yaitu eksekusi secara sekuensial

membutuhkan waktu eksekusi selama 3 menit 9 detik, pada eksekusi secara paralel dengan 2 inti prosesor membutuhkan waktu eksekusi selama 2 menit 14 detik, pada eksekusi secara paralel dengan 3 inti prosesor membutuhkan waktu eksekusi selama 1 menit 27 detik, pada eksekusi secara paralel dengan 4 inti prosesor membutuhkan waktu eksekusi selama 1 menit 16 detik.

Table 4.4 juga menunjukkan bahwa eksekusi program secara sekuensial membutuhkan waktu total selama 22 menit 47 detik, pada eksekusi program secara paralel dengan 2 inti membutuhkan waktu total selama 21 menit 26 detik sedangkan eksekusi program secara paralel dengan 3 inti membutuhkan waktu total selama 21 menit 22 detik dan eksekusi program secara paralel dengan 4 inti membutuhkan waktu total selama 21 menit 15 detik. Dari hasil pengujian pertama dapat diketahui bahwa waktu eksekusi program secara paralel lebih cepat dibandingkan dengan eksekusi secara sekuensial dan semakin banyak jumlah inti prosesor yang digunakan maka semakin cepat waktu eksekusi yang dibutuhkan.

4.2.3 Pengujian dengan Input 3 File ECG 10 Menit

Pengujian ketiga dilakukan dengan menggunakan data input yang dipotong-potong menjadi 3 bagian. Setiap bagian berisikan data selama 10 menit, file pertama berisikan 10 menit pertama file kedua berisikan data menit ke 10 sampai dengan menit ke 20 dan file ketiga berisikan sisa data terakhir.

Pada table 4.5 menunjukkan bahwa proses pada *Split Module* dan *Peak Module* hanya membutuhkan waktu yang relatif singkat yaitu kurang dari 10 detik. Terdapat perbedaan waktu eksekusi saat memproses *Spectogram Module* yaitu eksekusi secara sekuensial membutuhkan waktu eksekusi selama 4 menit 50 detik, pada eksekusi secara paralel dengan 2 inti prosesor membutuhkan waktu eksekusi selama 3 menit 25 detik, pada eksekusi secara paralel dengan 3 inti prosesor membutuhkan waktu eksekusi selama 2 menit 14 detik, pada eksekusi secara paralel dengan 4 inti prosesor membutuhkan waktu eksekusi selama 1 menit 53 detik.

Table 4.5 juga menunjukkan bahwa eksekusi program secara sekuensial membutuhkan waktu total selama 35 menit 16 detik,

Tabel 4.5: Hasil Pengujian ketiga menggunakan 3 file ECG 10 menit

Sequence paralel	Waktu yang dibutuhkan setiap *module*				
	Split	Spectogram	YOLO	Peak	Total
Sekuensial	00m 03d	04m 50d	30m 21d	00m 02d	35m 16d
2 Core	00m 03d	03m 25d	29m 31d	00m 03d	33m 02d
3 Core	00m 02d	02m 14d	28m 49d	00m 01d	31m 06d
4 Core	00m 02d	01m 53d	30m 01d	00m 02d	31m 58d

pada eksekusi program secara paralel dengan 2 inti membutuhkan waktu total selama 33 menit 02 detik sedangkan eksekusi program secara paralel dengan 3 inti membutuhkan waktu total selama 31 menit 06 detik dan eksekusi program secara paralel dengan 4 inti membutuhkan waktu total selama 31 menit 58 detik. Dari hasil pengujian pertama dapat diketahui bahwa waktu eksekusi program secara paralel lebih cepat dibandingkan dengan eksekusi secara sekuensial dan semakin banyak jumlah inti prosesor yang digunakan maka semakin cepat waktu eksekusi yang dibutuhkan.

4.2.4 Pengujian dengan Input 4 File ECG 10 Menit

Pengujian keempat dilakukan dengan menggunakan data input yang sama seperti percobaan ketiga, namun ditambahkan pula data ECG selama 10 menit sehingga total jumlah file yang dieksekusi sebanyak 4 file.

Pada table 4.6 menunjukkan bahwa proses pada *Split Module* dan *Peak Module* hanya membutuhkan waktu yang relatif singkat yaitu kurang dari 10 detik. Terdapat perbedaan waktu eksekusi saat memproses *Spectogram Module* yaitu eksekusi secara sekuensial membutuhkan waktu eksekusi selama 6 menit 8 detik, pada eksekusi secara paralel dengan 2 inti prosesor membutuhkan waktu eksekusi selama 3 menit 7 detik, pada eksekusi secara paralel dengan 3 inti

Tabel 4.6: Hasil Pengujian keempat menggunakan 4 file ECG 10 menit

Sequence paralel	Waktu yang dibutuhkan setiap *module*				
	Split	Spectogram	YOLO	Peak	Total
Sekuensial	00m 05d	06m 08d	41m 42d	00m 04d	47m 59d
2 Core	00m 03d	03m 07d	39m 54d	00m 02d	43m 06d
3 Core	00m 02d	03m 02d	40m 58d	00m 02d	44m 04d
4 Core	00m 02d	02m 29d	40m 41d	00m 02d	43m 14d

prosesor membutuhkan waktu eksekusi selama 3 menit 2 detik, pada eksekusi secara paralel dengan 4 inti prosesor membutuhkan waktu eksekusi selama 2 menit 29 detik.

Table 4.6 juga menunjukkan bahwa eksekusi program secara sekuensial membutuhkan waktu total selama 47 menit 59 detik, pada eksekusi program secara paralel dengan 2 inti membutuhkan waktu total selama 43 menit 6 detik sedangkan eksekusi program secara paralel dengan 3 inti membutuhkan waktu total selama 44 menit 4 detik dan eksekusi program secara paralel dengan 4 inti membutuhkan waktu total selama 43 menit 14 detik. Dari hasil pengujian pertama dapat diketahui bahwa waktu eksekusi program secara paralel lebih cepat dibandingkan dengan eksekusi secara sekuensial dan semakin banyak jumlah inti prosesor yang digunakan maka semakin cepat waktu eksekusi yang dibutuhkan.

4.2.5 Pengujian dengan Input File ECG 30 Menit

Pengujian kelima dilakukan dengan menggunakan data ECG secara utuh yaitu selama 30 menit.

Pada table 4.7 menunjukkan bahwa proses pada *Split Module* dan *Peak Module* hanya membutuhkan waktu yang relatif singkat yaitu kurang dari 10 detik. Terdapat perbedaan waktu eksekusi saat memproses *Spectogram Module* yaitu eksekusi secara sekuensial membutuhkan waktu eksekusi selama 4 menit 57 detik, pada

Tabel 4.7: Hasil Pengujian kelima menggunakan file ECG 30 menit

Sequence paralel	Waktu yang dibutuhkan setiap *module*				
	Split	Spectogram	YOLO	Peak	Total
Sekuensial	00m 03d	04m 57d	30m 43d	00m 03d	35m 46d
2 Core	00m 03d	03m 38d	29m 06d	00m 03d	32m 50d
3 Core	00m 03d	02m 08d	30m 02d	00m 01d	32m 14d
4 Core	00m 03d	01m 53d	30m 26d	00m 02d	32m 24d

eksekusi secara paralel dengan 2 inti prosesor membutuhkan waktu eksekusi selama 3 menit 38 detik, pada eksekusi secara paralel dengan 3 inti prosesor membutuhkan waktu eksekusi selama 2 menit 8 detik, pada eksekusi secara paralel dengan 4 inti prosesor membutuhkan waktu eksekusi selama 1 menit 53 detik.

Table 4.7 juga menunjukkan bahwa eksekusi program secara sekuensial membutuhkan waktu total selama 35 menit 46 detik, pada eksekusi program secara paralel dengan 2 inti membutuhkan waktu total selama 32 menit 50 detik sedangkan eksekusi program secara paralel dengan 3 inti membutuhkan waktu total selama 32 menit 14 detik dan eksekusi program secara paralel dengan 4 inti membutuhkan waktu total selama 32 menit 24 detik. Dari hasil pengujian pertama dapat diketahui bahwa waktu eksekusi program secara paralel lebih cepat dibandingkan dengan eksekusi secara sekuensial dan semakin banyak jumlah inti prosesor yang digunakan maka semakin cepat waktu eksekusi yang dibutuhkan.

4.3 Analisa Pengujian

1. Hasil semua pengujian yang dilakukan mulai dari pengujian pertama hingga pengujian kelima dengan masukan data rekaman kurang dari 30 menit menunjukkan bahwa *Split Module* dan *Peak Module* membutuhkan waktu eksekusi yang rendah yaitu waktu eksekusi kurang dari 10 detik.

2. *YOLO Module* secara otomatis berjalan secara paralel yang telah ditangani oleh *GPU* sehingga tidak bisa dijadikan acuan analisa perbandingan pada sistem karena semua eksekusi pada *YOLO Module* dikerjakan dengan ekosistem yang sama.

4.3.1 Evaluasi Paralel Model pada *Spectogram Module*

Evaluasi model paralel dari hasil pengujian diambil pada tahapan *Spectogram Module*, evaluasi ini meliputi perbandingan eksekusi secara sekuensial dengan eksekusi secara paralel, nilai *speed up* dan *efficiency* dari model paralel tersebut.

4.3.1.1 Perbandingan Eksekusi Sekuensial dengan Eksekusi Paralel Menggunakan 2 Inti Prosesor

Tabel 4.8 menunjukkan data hasil perbandingan antara waktu eksekusi komputasi secara sekuensial dengan secara paralel menggunakan 2 inti prosesor pada proses *Spectogram Module*.

Tabel 4.8: Evaluasi paralel model pada paralel yang menggunakan 2 inti prosesor

Node	Sekuensial	2 Core		
	Ts	Tp	Sp	Ep
File 10 Menit	95	69	1,38	0,69
2 File 10 Menit	189	134	1,41	0,71
3 File 10 Menit	290	205	1,41	0,71
4 File 10 Menit	368	187	1,97	0,98
File 30 Menit	297	218	1,36	0,68

Tabel 4.8 juga memberikan informasi bahwa pengolahan data 4 File ECG dengan rekaman selama 10 menit menghasilkan nilai

speed up paling besar yaitu 1,97 dengan nilai efisiensi sebesar 0,98.

4.3.1.2 Perbandingan Eksekusi Sekuensial dengan Eksekusi Paralel Menggunakan 3 Inti Prosesor

Tabel 4.9 menunjukkan data hasil perbandingan antara waktu eksekusi komputasi secara sekuensial dengan secara paralel menggunakan 3 inti prosesor pada proses *Spectogram Module*.

Tabel 4.9: Evaluasi paralel model pada paralel yang menggunakan 3 inti prosesor

Node	Sekuensial	3 Core		
	Ts	Tp	Sp	Ep
File 10 Menit	95	44	2,16	0,72
2 File 10 Menit	189	87	2,17	0,72
3 File 10 Menit	290	134	2,16	0,72
4 File 10 Menit	368	182	2,02	0,67
File 30 Menit	297	128	2,32	0,77

Tabel 4.9 juga memberikan informasi bahwa pengolahan data 1 File ECG dengan rekaman selama 30 menit speed up paling besar yaitu 2,32 dengan nilai efisiensi sebesar 0,77.

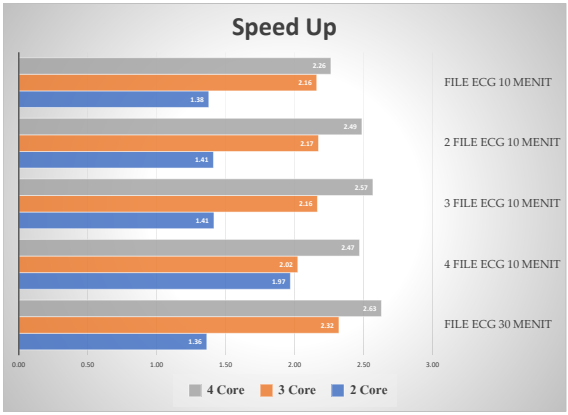
4.3.1.3 Perbandingan Eksekusi Sekuensial dengan Eksekusi Paralel Menggunakan 4 Inti Prosesor

Tabel 4.10 menunjukkan data hasil perbandingan antara waktu eksekusi komputasi secara sekuensial dengan secara paralel menggunakan 4 inti prosesor pada proses *Spectogram Module*.

Tabel 4.10: Evaluasi paralel model pada paralel yang menggunakan 4 inti prosesor

Node	Sekuensial	4 Core		
	Ts	Tp	Sp	Ep
File 10 Menit	95	42	2,26	0,57
2 File 10 Menit	189	76	2,49	0,62
3 File 10 Menit	290	113	2,57	0,64
4 File 10 Menit	368	149	2,47	0,62
File 30 Menit	297	113	2,63	0,66

Tabel 4.10 juga memberikan informasi bahwa pengolahan data 1 File ECG dengan rekaman selama 30 menit speed up paling besar yaitu 2,63 dengan nilai efisiensi sebesar 0,66.

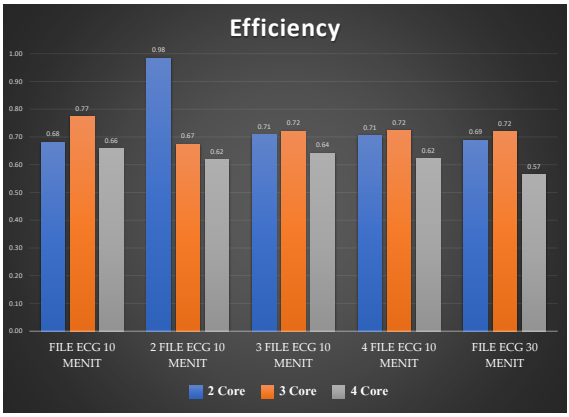


Gambar 4.6: Grafik *Speed Up* pada model paralel

4.3.1.4 Grafik *Speed Up*

Grafik pada gambar 4.6 merupakan grafik nilai *speed up* evaluasi dari paralel model yang menggunakan 2 inti, 3 inti, dan 4 inti prosesor. Terlihat pada grafik tersebut bahwa eksekusi paralel menggunakan 4 inti prosesor selalu mendapatkan nilai *speed up* yang paling tinggi, hal ini membuktikan bahwa peningkatan jumlah prosesor pada paralel model berbanding terbalik dengan waktu eksekusi yang dibutuhkan. Jadi, semakin banyak inti prosesor yang bekerja dalam mengeksekusi sistem maka semakin cepat waktu eksekusi yang dibutuhkan.

4.3.1.5 Grafik *Efficiency*



Gambar 4.7: Grafik *Efficiency* pada model paralel

Grafik pada gambar 4.7 merupakan grafik nilai *efficiency* evaluasi dari paralel model yang menggunakan 2 inti, 3 inti, dan 4 inti prosesor. Terlihat pada grafik tersebut bahwa nilai *efficiency* paralel model tertinggi sebesar 0,98 yang terjadi saat melakukan eksekusi program paralel menggunakan 2 inti prosesor dengan masukan 2 file EKG selama 10 menit.

4.3.2 Analisa Efektifitas dan Efisiensi Sistem

Komputasi secara paralel berhasil memberikan efisiensi waktu dengan waktu eksekusi lebih cepat dibandingkan dengan komputasi secara sekuensial. Penggunaan sumber daya perangkat juga menunjang efektifitas sistem, semakin banyak jumlah inti prosesor yang bekerja maka akan mempercepat waktu eksekusi dan juga faktor banyaknya data yang diproses akan memberikan nilai efisien paralel model yang bagus pada sistem.

4.3.3 Analisa Hasil Laporan Waktu Deteksi Titik Puncak-R

Selama pengujian juga dilakukan evaluasi terhadap keluaran dari sistem yang berupa hasil laporan waktu kejadian titik puncak pada data rekaman jantung. Nilai evaluasi didapatkan dengan membandingkan hasil laporan dengan anotasi yang telah disediakan oleh physionet dan ditunjukkan pada tabel 4.11.

Tabel 4.11: Hasil analisa laporan waktu deteksi titik puncak-R

No	Records	Beats	TP	FP	FN
1	ECG 100	2.275	2.270	1	4
2	ECG 101	1.882	1.859	8	15
3	ECG 102	2.291	2.050	99	142
4	ECG 103	2.092	2.084	1	7
5	ECG 104	2.415	2.022	104	289
6	ECG 105	2.703	2.180	12	511
7	ECG 106	2.152	1.734	54	364
8	ECG 107	2.222	1.861	82	279
9	ECG 108	2.210	752	386	1.072
10	ECG 109	2.574	234	29	2.311
11	ECG 111	2.221	2.039	88	94
12	ECG 112	2.557	2.523	7	27
13	ECG 113	1.800	1.795	4	1
14	ECG 114	2.318	1.690	428	200
15	ECG 115	1.962	1.953	0	9
16	ECG 116	2.428	2.342	7	79
17	ECG 117	1.955	1.123	416	416

18	ECG 118	2.308	2.058	7	243
19	ECG 119	2.096	1.838	2	256
20	ECG 121	1.877	470	1	1.406
21	ECG 122	2.481	2.465	2	14
22	ECG 123	1.519	1.517	0	2
23	ECG 124	1.646	1.388	12	246
24	ECG 200	3.569	1.734	777	1.058
25	ECG 201	2.153	1.857	114	182
26	ECG 202	2.159	2.113	13	33
27	ECG 203	3.337	1.429	229	1.679
28	ECG 205	2.732	2.584	60	88
29	ECG 207	4.066	82	1.681	2.303
30	ECG 208	3.111	2.424	71	616
31	ECG 209	3.078	2.984	26	68
32	ECG 210	2.860	2.390	175	295
33	ECG 212	2.791	2.692	28	71
34	ECG 213	3.332	3.056	38	238
35	ECG 214	2.311	2.085	14	212
36	ECG 215	3.407	3.276	7	124
37	ECG 217	2.783	1.741	503	539
38	ECG 219	2.335	2.126	23	186
39	ECG 220	2.069	2.045	0	24
40	ECG 221	2.500	2.177	38	285
41	ECG 222	2.686	2.466	52	168
42	ECG 223	2.697	2.369	54	274
43	ECG 228	2.164	1.580	23	561
44	ECG 230	2.476	2.248	10	218
45	ECG 231	2.033	1.571	22	440
46	ECG 232	1.872	1.747	56	69
47	ECG 233	3.721	2.381	569	771
48	ECG 234	2.777	2.750	13	14
Total			94.154	6.346	18.503

Halaman ini sengaja dikosongkan.

BAB 5
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil beberapa pengujian yang sudah dilakukan dapat disimpulkan bahwa sistem telah bekerja dengan cukup baik sesuai dengan rancangan dan mengimplementasikan komputasi parallel dalam pengolahan sinyal ECG untuk mendeteksi QRS kompleks pada sinyal EKG. Kemudian untuk lebih detailnya dapat ditarik beberapa kesimpulan sebagai berikut:

1. Sistem berjalan dengan baik dan memberikan laporan waktu kejadian aktifitas jantung namun titik puncak masih buruk pada jenis tertentu.
2. Komputasi secara paralel telah berhasil memberikan efektifitas dalam penggunaan sumber daya perangkat dan efisiensi waktu lebih cepat dalam eksekusi program.
3. Hasil evaluasi dari paralel model memberikan nilai *speed up* tertinggi sebesar 2.63 yang terjadi pada paralel dengan menggunakan 4 inti prosesor dan nilai. *efficiency* tertinggi sebesar 0.98 yang terjadi ketika menggunakan paralel dengan 2 inti prosesor.
4. Waktu eksekusi yang dibutuhkan dalam menjalankan *Split Module* dan *Peak Module* sangat rendah dengan waktu kurang dari 10 detik.

5.2 Saran

Untuk pengembangan penelitian selanjutnya terdapat beberapa saran sebagai berikut :

1. Memperbarui model yang digunakan oleh YOLO dengan melakukan training ulang sehingga hasil laporan yang dihasilkan bisa menjadi lebih bagus.
2. Pengembangan dari tugas akhir ini dapat dikembangkan secara massive dengan menciptakan server pusat sebagai data center beserta arsitektur pembentuknya.
3. Penerapan komputasi parallel yang digunakan dilakukan pada ruang lingkup GPU (*GPU Programming*) atau secara *distri-*

buted system seperti *message passing*

4. Tambahan fitur pada web sistem informasi seperti konfigurasi program eksekusi dengan menggunakan user interface, kontrol data EKG baik file masukan maupun hasil laporan.

DAFTAR PUSTAKA

[1] PoweredTemplate, "Sistem kelistrikan bebas jantung," 2011. https://poweredtemplate.com/id/sistem-kelistrikan-bebas-jantung-26096/, Last accessed on 2021-02-19. (Dikutip pada halaman xxi, 7).

[2] Wikipedia, "Elektrokardiogram." https://id.wikipedia.org/wiki/Elektrokardiogram, Last accessed on 2021-02-19. (Dikutip pada halaman xxi, 9).

[3] Y. Ji, X. Zhang, G. Zhang, and H. Jingjing, "Research on fine grained software radio communication algorithm based on gpu parallel processing technology," Cluster Computing, vol. 22, pp. 1–10, 03 2019. (Dikutip pada halaman xxi, 11).

[4] G. Zaccone, Python parallel programming cookbook. Packt Publishing Ltd, 2015. (Dikutip pada halaman xxi, 12, 13, 14).

[5] datahacker.rs, "Tf yolo v3 object detection in tensorflow 2.0," 2019. http://datahacker.rs/tensorflow2-0-yolov3/, Last accessed on 2020-02-06. (Dikutip pada halaman xxi, 17).

[6] S. Qi and W. Yishan, "Detection and analysis of ecg based on parallel computing technology," in 2009 International Forum on Computer Science-Technology and Applications, vol. 1, pp. 106–109, 2009. (Dikutip pada halaman xxi, xxiii, 19, 20).

[7] W. H. Hwang, C. H. Jeong, D. H. Hwang, and Y. C. Jo, "Automatic detection of arrhythmias using a yolo-based network with long-duration ecg signals," Engineering Proceedings, vol. 2, no. 1, 2020. (Dikutip pada halaman xxi, 21, 22).

[8] Alexey AB, "darknet," 2020. https://github.com/AlexeyAB/darknet, Last accessed on 2020-02-06. (Dikutip pada halaman xxi, 30).

[9] P. Mane, "Introduction to python multiprocessing," 2017. https://www.ellicium.com/

python-multiprocessing-pool-process/, Last accessed on 2021-02-19. (Dikutip pada halaman xxi, 33).

[10] P. Tröger, "Openhpi - parallel programming concepts - week 6," 06 2014. (Dikutip pada halaman xxi, 35).

[11] Kementerian Kesehatan Republik Indonesia, "Penyuluhan promosi kesehatan rsmh gangguan irama jantung aritmia," 2017. http://yankes.kemkes.go.id/read-penyuluhan-promkes-rsmh-gangguan-irama-jantung-aritm html, Last accessed on 2019-11-30. (Dikutip pada halaman 1).

[12] Oxford Medicine, "Esc cardiomed," 2018. https://oxfordmedicine.com/view/10.1093/med/9780198784906.001.0001/med-9780198784906-chapter64#med-9780198784906-chapter-64-bibItem-130, Last accessed on 2019-11-30. (Dikutip pada halaman 1).

[13] Humas Sardjito, "Jangan anggap remeh gangguan irama jantung," 2019. https://sardjito.co.id/2019/08/28/jangan-anggap-remeh-gangguan-irama-jantung/, Last accessed on 2021-02-19. (Dikutip pada halaman 8).

[14] American Heart Association, "All about heart rate (pulse)," 2015. https://www.heart.org/en/health-topics/high-blood-pressure/the-facts-about-high-blood-pressure/all-about-heart-rate-pulse, Last accessed on 2020-02-06. (Dikutip pada halaman 8).

[15] A. Mubarik and A. M. Iqbal, "Holter monitor," StatPearls [Internet], 2019. (Dikutip pada halaman 9).

[16] I. Wibawa, I. Giriantari, and M. Sudarma, "Komputasi paralel menggunakan model message passing pada sim rs (sistem informasi manajemen rumah sakit)," Majalah Ilmiah Teknologi Elektro, vol. 17, p. 439, 12 2018. (Dikutip pada halaman 10).

[17] M. Susmikanti and W. Dewayatna, "Komputasi paralel eigenvalue dalam penyelesaian difusi multigroup menggunakan metoda householder deflasi dan divide conquer," Lokakarya Komputasi dalam Sains dan Teknologi Nuklir, pp. 341–352, 2012. (Dikutip pada halaman 10).

[18] D. A. Reed and R. M. Fujimoto, Multicomputer networks: Message-based parallel processing. MIT press, 1988. (Dikutip pada halaman 12).

[19] T. L. Freeman and C. Phillips, Parallel numerical algorithms. Prentice-Hall, 1992. (Dikutip pada halaman 15).

[20] I. T. Foster, Designing and building parallel programs: concepts and tools for parallel software engineering. Addison-Wesley, 1995. (Dikutip pada halaman 15).

[21] K. Sinhal and A. Sachan, "Zero to hero: Guide to object detection using deep learning: Faster r-cnn, yolo, ssd," 2017. https://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/, Last accessed on 2020-02-06. (Dikutip pada halaman 17).

[22] Haiqal Muhamad Alfarisi, "You only look once (yolo) algoritma deep learning object detection terbaik," 2020. https://haiqalmuhamadalfarisi.medium.com/you-only-look-once-yolo-algoritma-deep-learning-object- Last accessed on 2021-02-19. (Dikutip pada halaman 18).

[23] A. Bochkovskiy, C.-Y. Wang, and H.-y. Liao, "Yolov4: Optimal speed and accuracy of object detection," 04 2020. (Dikutip pada halaman 18).

[24] P. N, V. Hegde, and A. Thakur, "Parallel processing architecture for ecg signal analysis," International Journal of Machine Learning and Computing, pp. 291–293, 01 2013. (Dikutip pada halaman 20).

[25] Wikipedia, "Short-time fourier transform." https://en.wikipedia.org/wiki/Short-time_Fourier_

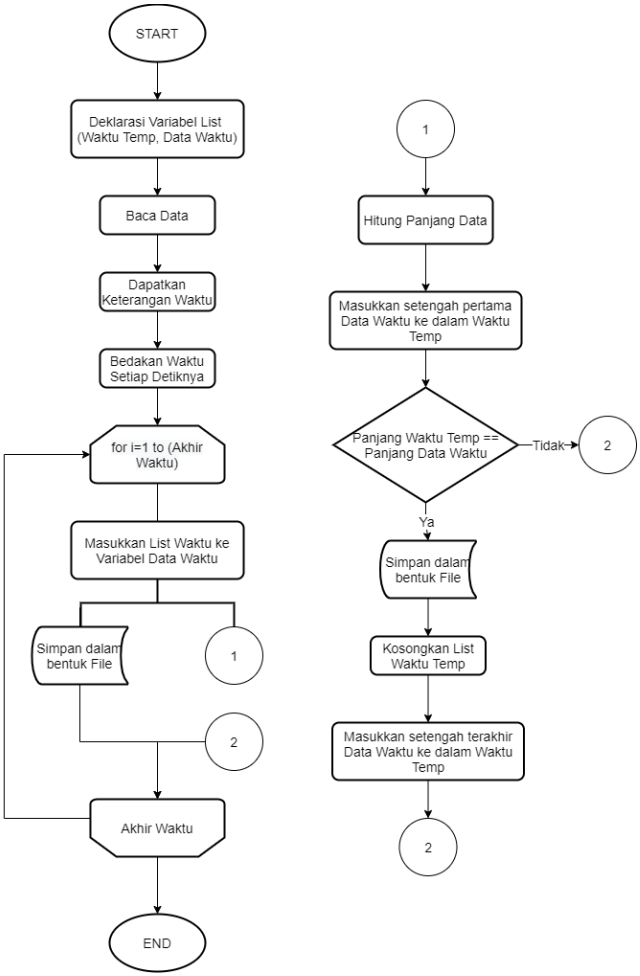
transform, Last accessed on 2021-02-20. (Dikutip pada halaman 27).

[26] Alexey AB, "yolo_mark," 2020. https://github.com/AlexeyAB/Yolo_mark, Last accessed on 2020-02-06. (Dikutip pada halaman 38).

LAMPIRAN

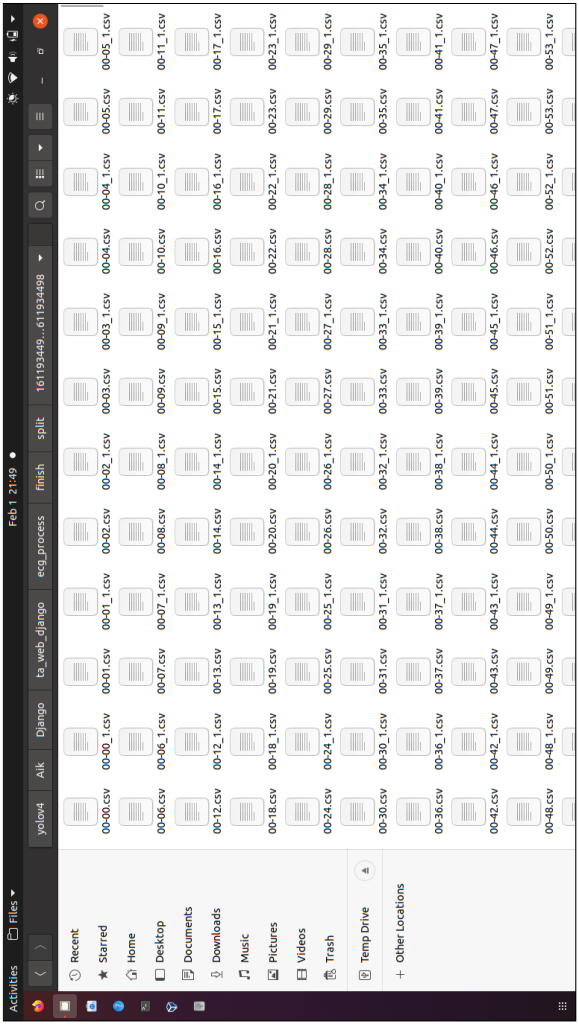
1 *Split Module*

1.1 *Flowchart Split Module*



Gambar 1.1: *Flowchart* pada *Split Module*

1.2 *Output Split Module*



Gambar 1.2: *Screenshot* hasil proses *Split Module*

BIOGRAFI PENULIS

Muhammad Achsan Hujjatul Islam, lahir di Madiun Jawa Timur pada tanggal 19 April 1998. Anak kedua dari delapan bersaudara ini telah menyelesaikan pendidikan di SMA Negeri Sooko Mojokerto tahun 2016 dan melanjutkan ke jenjang strata satu di Departemen Teknik Komputer Fakultas Teknologi Elektro dan Informatika Cerdas ITS. Selama ini, penulis tertarik dengan bidang *ethical hacking* terutama pada teknologi *cyber security* dan menyukai basis data khususnya *big data*, dalam menempuh perjalanan itu penulis sudah mengikuti serangkaian pelatihan *cyber security* yang diadakan oleh BSSN, dan juga telah memperdalam bahasa pemrograman Python melalui program *Digital Talent Scholarship* oleh KEMKOMINFO. Penulis pernah aktif dalam bidang keorganisasian pada sektor keagamaan JMMI dan juga berpartisipasi menjadi panitia pada event MAGE di jurusan. Bagi pembaca yang memiliki kritik, saran, atau pertanyaan mengenai tugas akhir ini dapat menghubungi penulis melalui email achsanh@gmail.com.

Halaman ini sengaja dikosongkan.