SKRIPSI

SISTEM WEARABLE PENDETEKSI JATUH DENGAN IMPLEMENTASI EDGE COMPUTING

Disusun dan diajukan oleh:

SAPHIRA NOER S.
D121 17 1520

PROGRAM STUDI SARJANA TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS HASANUDDIN
GOWA
2024

LEMBAR PENGESAHAN SKRIPSI

SISTEM WEARABLE PENDETEKSI JATUH DENGAN IMPLEMENTASI EDGE COMPUTING

Disusun dan diajukan oleh

SAPHIRA NOER S.
D121 17 1520

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian Studi Program Sarjana Program Studi Teknik Informatika
Fakultas Teknik Universitas Hasanuddin
Pada tanggal 01 Juli 2024
dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui,

Pembimbing Utama,	Pembimbing Pendamping,
Dr. Adnan, S.T., M.T.	Dr. Ir. Zahir Zainuddin, M.Sc.
NIP 197404262005011002	NIP 196404271989101002

Ketua Program Studi,

Prof. Dr. Ir. Indrabayu, ST., MT., M.Bus.Sys., IPM, ASEAN. Eng.
NIP 197507162002121004

PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini;
Nama : Saphira Noer S.
NIM : D121171520
Program Studi : Teknik Informatika
Jenjang : S1

Menyatakan dengan ini bahwa karya tulisan saya berjudul

Sistem Wearable Pendeteksi Jatuh Dengan Implementasi Edge Computing

Adalah karya tulisan saya sendiri dan bukan merupakan pengambilan alihan tulisan orang lain dan bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri.

Semua informasi yang ditulis dalam skripsi yang berasal dari penulis lain telah diberi penghargaan, yakni dengan mengutip sumber dan tahun penerbitannya. Oleh karena itu semua tulisan dalam skripsi ini sepenuhnya menjadi tanggung jawab penulis. Apabila ada pihak manapun yang merasa ada kesamaan judul dan atau hasil temuan dalam skripsi ini, maka penulis siap untuk diklarifikasi dan mempertanggungjawabkan segala resiko.

Segala data dan informasi yang diperoleh selama proses pembuatan skripsi, yang akan dipublikasi oleh Penulis di masa depan harus mendapat persetujuan dari Dosen Pembimbing.

Apabila dikemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan isi skripsi ini hasil karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Gowa, 10 Juli 2024

Yang Menyatakan

SAPHIRA NOER S.

ABSTRAK

SAPHIRA NOER S. Sistem Wearable Pendeteksi Jatuh Dengan Implementasi Edge Computing (dibimbing oleh Adnan dan Zahir Zainuddin)

Meningkatnya kebutuhan akan sistem deteksi jatuh yang efektif dan akurat pada dekade terakhir ini meningkat terutama bagi populasi lansia yang rentan terhadap cedera akibat jatuh. Teknologi sensor akselerometer telah berkembang pesat dan menawarkan solusi potensial untuk mendeteksi jatuh secara *real-time* sehingga sistem wearable banyak diimplementasikan untuk sistem deteksi jatuh. Namun, tantangan utama adalah *deployment* model *deep learning* yang dapat membedakan antara jatuh dan aktivitas sehari-hari dengan tingkat akurasi tinggi ke dalam perangkat bersumber-daya terbatas.

Penelitian ini bertujuan untuk mengembangkan sistem deteksi jatuh mengimplementasikan *edge computing* dengan dukungan Edge Impulse untuk membangun model deep learning dan Bluetooth Low Energy untuk memungkinkan komunikasi antar-perangkat berdaya rendah dengan konsumsi daya yang minim. Edge computing diharapkan dapat mengefisienkan pemrosesan dan pengiriman data dengan cara melakukan pemrosesan lebih dekat secara fisik dengan sumber data. Tujuan lain dari penelitian ini adalah untuk mengevaluasi kinerja model yang telah disematkan ke dalam perangkat edge bila dibandingkan dengan akurasi model secara teori.

Data akselerasi tri-aksial UniMiB SHAR yang mengandung data akselerasi jatuh dan kegiatan sehari-hari diproses menggunakan filter Butterworth orde keenam dengan frekuensi *cut-off* 3 Hz untuk menghilangkan *noise*. Selanjutnya, data yang telah difilter dihitung besaran magnitudonya dan digunakan sebagai input untuk model *deep learning*. Perangkat edge dan wearable yang masing-masing merupakan mikrokontroller saling berkomunikasi menggunakan Bluetooth Low Energy, dimana wearable akan mengirimkan data akselerasi kepada edge yang akan memroses data tersebut untuk menghasilkan keputusan.

Hasil penelitian menunjukkan bahwa model deteksi jatuh yang dikembangkan memiliki tingkat akurasi yang tinggi dalam membedakan jatuh dari kegiatan sehari-hari, yaitu sebesar 91%.

Kata Kunci: Edge Computing, Jatuh, Wearable

ABSTRACT

SAPHIRA NOER S. *Fall Detection Wearable System with Edge Computing* (supervised by Adnan and Zahir Zainuddin)

The increasing need for effective and accurate fall detection systems in the last decade has increased especially for the elderly population who are vulnerable to fall injuries. Accelerometer sensor technology has developed rapidly and offers a potential solution to detect falls in real-time, thus wearable systems are widely implemented for fall detection systems. However, a major challenge is the deployment of deep learning models that can distinguish between falls and daily activities with a high degree of accuracy into limited-resource devices.

This research aims to develop a fall detection system implementing edge computing with the support of Edge Impulse to build deep learning models and Bluetooth Low Energy to enable low-power inter-device communication with minimal power consumption. Edge computing is expected to streamline data processing and transmission by performing processing physically closer to the data source. Another objective of this research is to evaluate the performance of the deep learning model that have been embedded into the edge device when compared to the acquired theoretical accuracy of said model.

The UniMiB SHAR dataset that contains tri-axial acceleration data of falls and daily activities were processed using a sixth-order Butterworth filter with a cut-off frequency of 3 Hz to remove noise. Next, the filtered data is calculated for magnitude and used as input for the deep learning model. The edge device and wearable, each of which is a microcontroller, communicate with each other using Bluetooth Low Energy, where the wearable will send acceleration data to the edge which will process the data to make a decision.

The results show that the fall detection model developed has a high accuracy rate in distinguishing falls from daily activities, which is 91%.

Keywords: Edge Computing, Fall, Wearable

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SINGKATAN DAN ARTI SIMBOL

Lambang/Singkatan	Arti dan Keterangan
IoT	Internet of Things
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
ANN	Artificial Neural Network
DNN	Deep Neural Network
MLOps	Machine Learning Operations
HAR	Human Activity Recognition
ADL	Activities Of Daily Living
BLE	Bluetooth Low Energy
RFID	Radio Frequency Identification
SBC	Single Board Computer
TFLM	Tensor Flow
UART	Universal Asynchronous Receiver Transmitter
UUID	Universally Unique Identifier
RAM	Random Access Memory
EON	Edge Optimized Neural
SMV	Sum Vector Magnitude / Signal Vector Magnitude
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
CSV	Comma Separated Values
DSP	Digital Signal Processing
FFT	Fast Fourier Transform

DAFTAR LAMPIRAN

KATA PENGANTAR

Bismillahirrahmanirrahim. Alhamdulillahi rabbil ‘alamin. Skripsi berjudul “Sistem Wearable Pendeteksi Jatuh Dengan Implementasi Edge Computing” ini mengambil permasalahan fatalitas jatuh yang dapat dimitigasi dengan cara mengimplementasikan *edge computing* dalam sebuah sistem deteksi jatuh. Skripsi yang menjadi syarat kelulusan jenjang Strata-1 di Universitas Hasanuddin ini tidak akan dapat diselesaikan jikalau bukan atas kehendak Allah Subhanahu wa Ta’ala dan bantuan setiap orang-orang yang telah Allah pertemukan atau sampaikan jasa-jasanya kepada penulis:

1. Keluarga penulis, Hafizhahumullah.
2. Para Pembimbing Skripsi penulis; Bapak Adnan, ST., MT., Ph.D. serta Bapak Dr. Ir. Zahir Zainuddin, M.Sc. yang telah membimbing penulis dalam pengerjaan skripsi dengan sabar.
3. Bapak Prof. Dr. Ir. Indrabayu, ST., MT, M.Bus.sys. selaku Pembimbing Akademik penulis dan Kepala Departemen Teknik Informatika yang telah memberi banyak motivasi dan inspirasi kepada penulis selama studi.
4. Segenap staf kampus, terkhusus para staf gedung elektro dan COT yang senantiasa berbagi dengan penulis baik berupa makanan maupun pengalaman tangan pertama yang begitu berharga.
5. Rekan-rekan Informatika terkhusus Kak Fadel, Amir, Fadhil, Ikbal, Devy, Irma, Nurina, Yudi, Alisyah, Herlina, Wahyu, dan Kak Andy yang telah banyak membantu dan menemani penulis selama mengerjakan skripsi.
6. Orang-orang yang belum pernah ditemui oleh penulis yang jerih payahnya tersampaikan dalam bentuk ide, fasilitas, maupun doa.
7. My past selves, for not giving up on me.

Semoga Allah membalas dengan sebaik-baiknya kebaikan. Terima kasih.

Gowa, Juli 2024
Penulis,

Saphira Noer S.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Secara global, jatuh merupakan penyebab utama kedua untuk kematian akibat cedera yang tidak disengaja. Setiap tahun diperkirakan sebanyak 684.000 individu meninggal karena jatuh. Definisi umum jatuh yaitu kejadian yang mengakibatkan seseorang berbaring secara tidak sengaja di permukaan yang lebih rendah. Sebanyak 37,3 juta kejadian jatuh yang cukup parah memerlukan perhatian medis setiap tahunnya. Secara global, kejadian jatuh bertanggung jawab untuk lebih dari 38 juta DALY (usia kematian di bawah angka harapan hidup dengan dampak dari cacat) yang bertambah tiap tahun (*Falls*, 2021).

Diantara upaya mitigasi jatuh yang dapat dilakukan adalah dengan mendetek si jatuh sesaat setelah jatuh itu terjadi dan memberitahukan pihak terkait. Beberapa jenis pendeteksi jatuh yaitu dengan konsep sistem *context-aware* dan sistem *wearable*. Deteksi jatuh yang menggunakan konsep *context-aware* dapat diterapkan dengan cara menggunakan kamera dan sensor-sensor *ambient* lainnya seperti kamera dalam ruangan, sensor tekanan pada lantai, atau mikrofon di sekitar pengguna yang ingin dipantau. Namun, menurut Gold dan Shaw, mobilitas pasien dapat dipantau secara lebih adaptif dan praktis menggunakan sensor-sensor ringan yang dapat dikenakan secara langsung oleh pasien karena meluasnya penggunaan perangkat elektronik menyebabkan penurunan biaya produksi sehingga sistem deteksi jatuh dengan penerapan sistem *wearable* semakin efisien (Gold & Shaw, 2022). Implementasi paling umum saat ini berupa sensor miniatur yang dapat dikenakan pada tubuh untuk mengamati perubahan gaya hidup dan perilaku pemakainya. Sebagai hasilnya, penelitian ekstensif telah dilakukan dalam pengembangan algoritma penalaran untuk menyimpulkan aktivitas dari data sensor-sensor *wearable*. Akselerometer merupakan sensor yang paling sering digunakan untuk tujuan pengenalan aktivitas dan dapat menyediakan akurasi klasifikasi tinggi (Janidarmian et al., 2017).

Proses deteksi jatuh dapat dilakukan secara langsung di perangkat *wearable*, atau di *cloud*. Namun, alasan perangkat *wearable* banyak digunakan oleh

masyarakat adalah kepraktisan dan kenyamanan karena ukuran perangkat yang kecil. Batasan ukuran ini menyebabkan terbatasnya kapasitas penyimpanan dan kemampuan komputasi perangkat-perangkat *wearable*. Pendeteksian di *cloud* juga dapat dilakukan. Akan tetapi, memanfaatkan layanan *cloud computing* seringkali bukanlah opsi yang ekonomis, dan layanan yang disediakan secara gratis seringkali memiliki batasan. Selain itu, meningkatnya pertumbuhan lalu lintas *Big Data* yang diakibatkan oleh pertumbuhan pesat IoT (Internet of Things) mengakibatkan koneksi data dan komunikasi data melalui jaringan menjadi lambat. Solusi yang efektif untuk permasalahan ini yaitu *Edge computing*. *Edge computing* merupakan komputasi yang terjadi di dalam perangkat *edge* yang peletakannya dekat dengan pusat data yang umunya berupa perangkat-perangkat yang berinteraksi langsung dengan pengguna akhir (*end user*). Oleh karena itu, *edge* menyumbang latensi yang lebih rendah bila dibandingkan dengan *cloud*. Data-data yang diperoleh dari *edge* akan diproses atau disimpan secara lokal, lalu semua data yang telah diproses dikirim ke *cloud* sehingga mengurangi beban komputasi dan jaringan, serta menghindari potensi *bottleneck* di layanan pusat. (*Edge Computing*, 2019).

Pengimplementasian *wearable* untuk memantau atau mengawasi kesehatan merupakan salah satu *use case* IoT yang paling umum. IoT merujuk pada objek-objek fisik yang disematkan sensor-sensor dan aktuator-aktuator yang saling berkomunikasi dengan sistem-sistem komputasi melalui jaringan kabel maupun nirkabel untuk memantau atau mengatur kehidupan sehari-hari (*What Is the Internet of Things?*, 2022). Transmisi nirkabel antara sensor-sensor ini dan *base station* berperan penting. Terdapat banyak solusi transmisi nirkabel, seperti teknologi IEEE 802.11 (WiFi), ZigBee, Bluetooth *Classic*, dan Bluetooth Low Energy (BLE). Meskipun teknologi WiFi dapat mentransmisi data dengan kecepatan tinggi, WiFi membutuhkan konsumsi daya yang besar untuk memfasilitasi lalu lintas *throughput* yang tinggi. Selain itu, WiFi tidak praktis dari perspektif koneksi antar perangkat. BLE yang menggabungkan konsumsi daya rendah dan penggunaan meluas merupakan protokol komunikasi radio jarak dekat yang memiliki kemampuan v untuk mengurangi konsumsi daya dan biaya perangkat pada perangkat-perangkat berdaya rendah (Liu et al., 2021).

Selama dekade terakhir ini, kita telah melihat pengembangan algoritma *Machine Learning* (ML) bersamaan dengan pengembangan *Internet of Things* di bidang mikroelektronik, komunikasi, dan teknologi informasi. Miliaran perangkat IoT terhubung ke internet; dengan jumlah perangkat IoT yang luas datang pula produksi massal platform IoT. Peran dari perangkat-perangkat ini adalah untuk mendeteksi fitur fisik dari lingkungan penempatannya—dua puluh empat jam sehari, tujuh hari seminggu. Hal ini menyebabkan peningkatan volume data yang dihasilkan yang sebaliknya juga memerlukan kinerja komputasi yang tinggi dan ruang penyimpanan yang besar. Dewasa ini, algoritma *Machine Learning* diintegrasikan dengan perangkat IoT agar mampu memproses jumlah data yang besar dan memungkinkan perangkat-perangkat tersebut untuk dapat mengambil keputusan. *Deep Neural Network* (DNN) atau *Deep Learning* (DL) yang kerap kali dimanfaatkan untuk mengolah data pada perangkat-perangkat *edge* sering dihadapkan dengan tantangan-tantangan berupa: terbatasnya kapabilitas perangkat-perangkat *edge* bila dibandingkan dengan besarnya model DL sehingga membebani biaya komputasi, sumber daya, daya, memori, dan waktu; serta tantangan lama waktu yang dibutuhkan bagi model untuk membuat keputusan. TinyML merupakan pendekatan yang dikembangkan untuk menghubungkan ML dengan perangkat *edge*. TinyML memungkinkan *deployment* model DL berukuran kecil ke dalam perangkat *edge* yang memiliki keterbatasan sumber daya yang signifikan. Dengan TinyML, analisis dan interpretasi data dapat dilakukan secara lokal pada perangkat sehingga pengambilan keputusan atau tindakan dapat dilakukan secara *real time* (Alajlan & Ibrahim, 2022).

Salah satu platform *Machine Learning Operations* (MLOps) yang memungkinkan pengguna individu maupun industri untuk merambah ke ranah TinyML adalah Edge Impulse, sebuah layanan berbasis *cloud* untuk pengembangan sistem-sistem *edge* dan tersemat. Edge Impulse menyederhanakan proses pengumpulan data, pelatihan model deep learning beserta evaluasinya, dan *deployment* model ke dalam perangkat-perangkat *edge computing*. Para pengguna dipermudah untuk berinteraksi dengan sistem dalam proses pelatihan model dan *deployment* oleh antarmuka grafis (GUI) and API. Selain itu, Edge Impulse menyediakan *library* C/C++ yang portable dan ekstensif guna meng-enkapsulasi

kode *preprocessing* dan model untuk menyederhanakan proses inferensi pada berbagai perangkat sekaligus mengoptimalisasi waktu inferensi dan konsumsi memori model (Hymel et al., 2022).

Berdasarkan paparan di atas, penelitian ini bermaksud untuk menghasilkan sebuah sistem deteksi jatuh memanfaatkan sensor akselerometer yang terintegrasi dalam perangkat *wearable* untuk memantau aktivitas pengguna. Sebagian besar pemrosesan data dijalankan pada perangkat *edge* yang telah disematkan sebuah model *machine learning*, sedangkan *cloud* hanya digunakan untuk mengirim notifikasi ke kontak darurat yang telah ditentukan. Guna mengoptimalkan penggunaan daya pada *wearable*, protokol komunikasi yang digunakan adalah BLE.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang di atas dapat disimpulkan rumusan masalah:

1. Bagaimana mengimplementasikan *edge computing* dalam membangun sistem deteksi jatuh?
2. Bagaimana performa model dalam mendeteksi jatuh?

1.3 Tujuan Penelitian

Tujuan dari penelitian yang dilakukan yaitu:

1. Membangun sistem deteksi jatuh mengimplementasikan *edge computing*.
2. Mengukur performa model dalam mendeteksi jatuh.

1.4 Manfaat Penelitian

Penelitian ini diharapkan dapat menambah pengetahuan pembaca dan menjadi referensi yang baik bagi penelitian-penelitian lain yang mengangkat tema *edge computing* dan IoT.

1.5 Ruang Lingkup

Ruang lingkup dalam membangun sistem ini adalah:

1. Penelitian ini hanya menggunakan dataset UniMIB SHAR sebagai data pelatihan.
2. Pelatihan model dilakukan di Edge Impulse.
3. Kategori pergerakan yang dideteksi dalam penelitian ini dikerucutkan menjadi dua kelas, yaitu ADL dan Fall.
4. Penelitian ini hanya mendeteksi jatuh berdasarkan sinyal akselerometer.
5. Perangkat *wearable* pada penelitian ini dikenakan di area pinggang atau paha (saku celana) pengguna berdasarkan konfigurasi yang dijabarkan pada bab-bab berikutnya.
6. Penelitian ini hanya menggunakan satu buah perangkat *wearable*.

BAB II
TINJAUAN PUSTAKA

2.1 Jatuh

Menurut WHO, jatuh yang tidak disengaja merupakan penyebab kematian akibat cedera paling banyak kedua setelah kecelakaan transportasi. Setiap tahun, diperkirakan 684 ribu kasus jatuh mengakibatkan kematian di seluruh dunia, dan 37,3 juta kasus jatuh diperlukan perhatian medis darurat. Meskipun hampir 40% dari total disabilitas akibat jatuh di seluruh dunia terjadi pada anak-anak, pengukuran ini mungkin tidak secara akurat mencerminkan dampak disabilitas akibat jatuh pada lansia. Selain itu, secara statistik diestimasikan terjadi kematian setiap 19 menit disebabkan cedera akibat jatuh oleh lansia (*Falls*, 2021; Pugh et al., 2019). Jatuh terjadi sepanjang hidup manusia, tetapi menjadi lebih berisiko pada orang yang berusia lebih dari 50 tahun. Beberapa penelitian menunjukkan bahwa sekitar 30% lansia berusia lebih dari 65 tahun mengalami jatuh setiap tahunnya. Persentase ini meningkat menjadi 40% untuk orang yang berusia lebih dari 80 tahun (Lahouar et al., 2024).

2.2 Internet of Things

Internet of Things (IoT) merujuk pada objek-objek fisik yang disematkan sensor-sensor (untuk memantau perubahan lingkungan) dan aktuator-aktuator (yang menerima sinyal dari sensor dan melakukan sesuatu sebagai respon dari perubahan lingkungan yang terjadi) yang saling berkomunikasi dengan sistem-sistem komputasi melalui jaringan kabel maupun nirkabel untuk memantau atau mengendalikan dunia nyata secara digital. Objek-objek fisik tersebut tidak mesti benda buatan, melainkan dapat berupa objek-objek di alam seperti manusia dan hewan (*What Is the Internet of Things?*, 2022).

Konsep IoT diciptakan oleh anggota komunitas pengembangan *Radio Frequency Identification* (RFID) pada tahun 1999. Saat ini, perkembangan IoT sangat meningkat didorong oleh pertumbuhan perangkat seluler, komunikasi yang semakin mudah dilakukan, *cloud computing*, dan *data analytics*. Tujuan IoT adalah untuk memungkinkan segala sesuatunya selalu terhubung kapan saja dengan apa

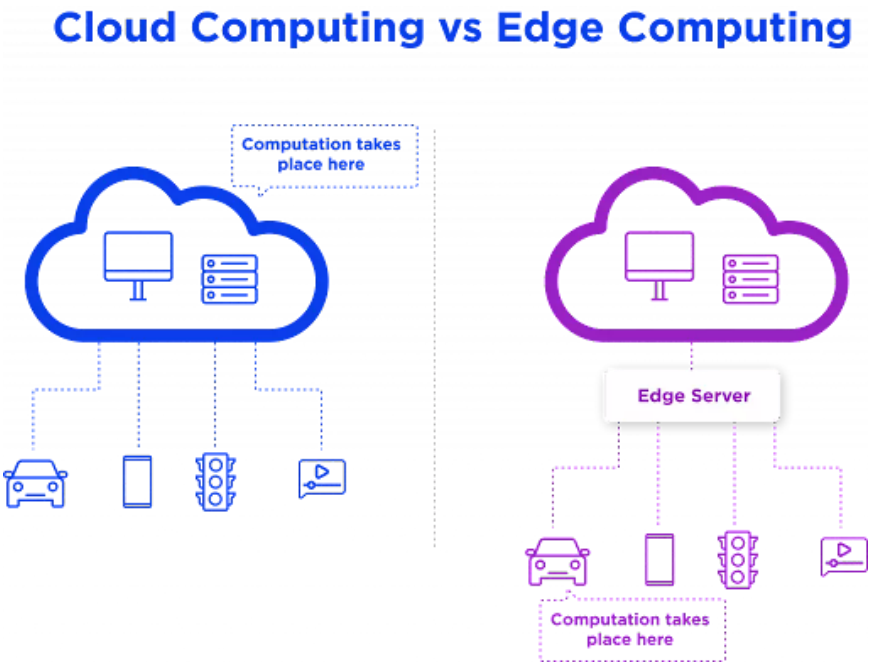
saja dan siapa saja yang ideal menggunakan jaringan dan layanan apapun. IoT mengacu pada penggunaan perangkat dan sistem yang terhubung untuk memanfaatkan data yang diperoleh dari sensor, aktuator tersemat dalam mesin, atau objek fisik lainnya (Thamrin et al., 2020).

2.3 Edge Computing

Cloud computing adalah komputasi, penyimpanan, pelayanan, dan sumber daya aplikasi yang terpusat dan tervirtualisasi melalui Internet. *Cloud computing* memisahkan layanan dari infrastruktur asalnya sehingga mengeliminasi biaya dan kompleksitas mengelola infrastruktur teknologi informasi secara independen. Namun, *cloud computing* tidak mampu melayani aplikasi IoT secara *real-time* karena infrastruktur *cloud* biasanya terletak di beberapa tempat yang secara fisik jauh dari pengguna. Selain itu, kebutuhan aplikasi IoT diantara lain meliputi latensi rendah, *jitter* rendah, *bandwidth* tinggi, dan moblitas tinggi (Mansouri & Babar, 2021).

Untuk memenuhi kebutuhan aplikasi IoT, dibutuhkan layanan berbasis *cloud* yang dekat dengan sumber data untuk memproses, menganalisis, dan memfilter data untuk mengambil keputusan yang tepat. Ekstensi layanan komputasi dari paradigma *cloud* ke tepi jaringan adalah *edge computing*. *Edge computing* dapat meningkatkan efisiensi keseluruhan infrastruktur dengan cara mengurangi latensi, mengurangi beban *backhaul*, mendukung layanan mobilitas, dan meningkatkan ketahanan layanan (Mansouri & Babar, 2021).

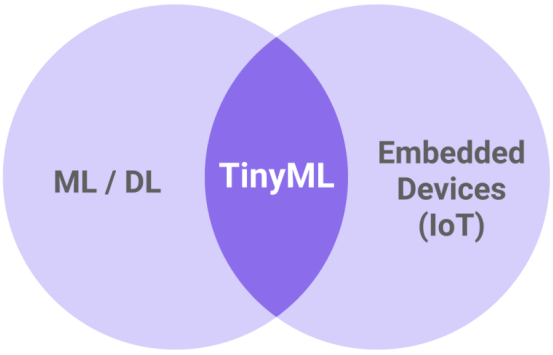
Paradigma *edge computing* terdiri atas perhubungan antara perangkat-perangkat yang terbatas sumber daya (*smartphone*, *wearable*, SBC), perangkat jaringan (switch, router), dan perangkat-perangkat dengan sumber daya moderat (*base station*), LAN, Cloudlets, dll. Perangkat-perangkat yang terhubung menghasilkan dan mengumpulkan sejumlah besar data yang awalnya diproses dan dianalisis di tepi jaringan. Data-data ini kemudian dapat ditransfer ke *cloud* untuk ekstraksi menggunakan berbagai pendekatan *Machine Learning* dan Optimasi (konveks atau Bayesian). Dengan demikian, *edge computing* melengkapi komputasi *cloud* (Mansouri & Babar, 2021). Gambar 1 mengilustrasikan penjelasan tersebut.



Gambar 1 Ilustrasi perbedaan *cloud computing* dengan *edge computing* (*Edge Computing vs. Cloud Computing: What's the Difference?*, 2021)

Seperti halnya *cloud computing*, teknologi fundamental dari paradigma *edge computing* adalah virtualisasi sumber daya yang memisahkan sumber daya perangkat keras dari perangkat lunak untuk menjalankan proses-proses pada beberapa pengguna menggunakan perangkat keras yang sama. Namun, *edge computing* memiliki perbedaan dengan komputasi *cloud* dalam hal lokalitas, layanan berbasis mobilitas, perangkat yang terbatas sumber daya dan heterogen, serta distribusi perangkat yang luas. (Mansouri & Babar, 2021).

2.4 TinyML



Gambar 2 Diagram venn korelasi ML, IoT, dan TinyML (Tomas, 2021).

TinyML merupakan pendekatan yang dikembangkan untuk menghubungkan ML dengan perangkat *edge.* TinyML memungkinkan pengimplementasian model DL berukuran kecil ke dalam perangkat *edge* yang memiliki keterbatasan sumber daya yang signifikan, seperti kemampuan komputasi terbatas, kapasitas penyimpanan rendah, atau perangkat berdaya rendah. Tujuan utama TinyML adalah untuk meningkatkan efektivitas sistem *Deep Learning* melalui pengurangan kebutuhan komputasi dan data untuk memfasilitasi kebutuhan *edge* AI dan IoT sebagaimana diilustrasikan pada Gambar 2.

Deep Neural Network (DNN) atau *Deep Learning* (DL) yang merupakan bagian dari *Machine Learning* (ML) yang data kerap kali dimanfaatkan untuk mengolah data pada perangkat-perangkat *edge* sering dihadapkan dengan tantangan berupa terbatasnya kapabilitas perangkat-perangkat edge bila dibandingkan dengan besarnya model DL, dan tantangan lamanya waktu yang dibutuhkan bagi model untuk membuat keputusan. TinyML memungkinkan *deployment* model DL berukuran kecil ke dalam perangkat *edge* yang memiliki keterbatasan sumber daya yang signifikan (Alajlan & Ibrahim, 2022).

Dengan TinyML, analisis dan interpretasi data dapat dilakukan secara lokal pada perangkat sehingga pengambilan keputusan atau tindakan dapat dilakukan secara *real time*. Model DL yang telah dilatih sebelumnya dapat di-*deploy* ke dalam perangkat *edge* setelah melakukan kompresi model DL dan mengoptimalkan proses inferensi. Misalnya, menggunakan teknik kuantisasi yang merupakan teknik konversi yang mengubah angka *float-point* menjadi angka presisi minimal untuk mengurangi ukuran model DL dengan degradasi akurasi yang minimal. Teknik *pruning* memungkinkan penghapusan struktur dan parameter jaringan yang berlebihan. TinyML memungkinkan berbagai perangkat IoT untuk mengolah data secara lokal tanpa perlu terhubung ke *cloud* untuk memproses data. TinyML menawarkan berbagai keuntungan, termasuk penghematan biaya, pengurangan konsumsi energi, dan peningkatan privasi (Alajlan & Ibrahim, 2022).

2.5 Edge Impulse

Edge Impulse merupakan platform operasi pembelajaran mesin (MLOps) berbasis *cloud* yang dikembangkan untuk sistem pembelajaran mesin terintegrasi

dan *edge* ML (TinyML), yang dapat diterapkan pada berbagai target perangkat keras. Edge Impulse diperkenalkan sebagai platform MLOps praktis untuk pengembangan sistem TinyML dalam skala besar dan menyederhanakan siklus desain TinyML dengan mendukung berbagai optimasi perangkat lunak dan perangkat keras untuk menciptakan tumpukan perangkat lunak yang ekstensif dan portabel untuk beragam sistem terintegrasi (Hymel et al., 2022).

Edge Impulse mengintegrasikan ekstraksi fitur *preprocessing* dengan DL. Selain itu, Edge Impulse menciptakan *Edge Optimized Neural* (EON) *Tuner* yang secara otomatis mengeksplorasi ruang pencarian yang ditentukan pengguna, mencakup *preprocessor* dan model ML. Selain itu, dengan kompiler EON, Edge Impulse menyediakan *library* inferensi yang ekstensibel dan portabel, yang dapat diterapkan di berbagai sistem *edge* dan terintegrasi, mengoptimalkan penggunaan sumber daya dengan mengurangi ruang RAM dan flash yang biasanya diperlukan oleh interpreter TensorFlow Lite Micro (TFLM). Compiler ini tidak hanya mengompilasi *neural network* TFLM menjadi *source code* C++ tetapi juga mengeliminasi kebutuhan akan interpreter TFLM dengan menghasilkan kode yang langsung memanggil kernel yang mendasarinya sehingga memungkinkan penghapusan instruksi yang tidak digunakan dan secara signifikan menurunkan penggunaan RAM dan ROM. Edge Impulse mendukung optimasi spesifik perangkat untuk kebutuhan perangkat keras tertentu dengan ekstensibilitas platform memungkinkan penambahan blok pemrosesan, pembelajaran, dan *deployment* kustom untuk mengakomodasi target dan optimasi tambahan sehingga menyesuaikan solusi untuk kemampuan perangkat keras spesifik dan meningkatkan efisiensi keseluruhan implementasi *neural network* pada perangkat *edge* (Hymel et al., 2022).

2.6 Bluetooth Low Energy

Bluetooth Low Energy (BLE) merupakan protokol komunikasi radio jarak dekat yang pertama kali digunakan dalam Bluetooth 4.0. BLE dan memiliki kemampuan untuk meminimalkan konsumsi daya dan biaya perangkat pada perangkat-perangkat berdaya rendah bila dibandingkan dengan teknologi Bluetooth sebelumnya. Fitur paling signifikan adalah konsumsi daya yang lebih rendah karena

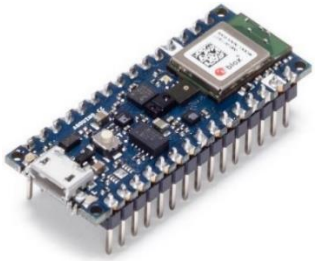
BLE berada dalam mode tidur pada sebagian besar waktu dan hanya membangunkan perangkat ketika aktivitas terjadi. BLE dirancang dengan keadaan *deep sleep* (Duty-Cycle) untuk menggantikan waktu *idle* dari Bluetooth classic. Selain itu, modifikasi BLE terhadap Bluetooth tradisional terletak pada mode *broadcast* dan mode *scanning*. Dalam mode *broadcast*, BLE hanya menggunakan 3 pita frekuensi, sedangkan Bluetooth classic menggunakan 16 hingga 32 pita frekuensi. Perubahan tersebut sangat mengurangi konsumsi daya. (Liu et al., 2021).

Tidak seperti komunikasi Bluetooth classic yang pada dasarnya berbasis koneksi serial asinkron (UART), perangkat-perangkat yang terhubung dengan BLE memiliki salah satu dari dua macam peran, yaitu sebagai perangkat *peripheral* dan perangkat *central*. Perangkat *peripheral* adalah perangkat yang menyalurkan informasi dengan cara mengiklankan (*advertising*) data tersebut kepada radio BLE di sekitarnya. Informasi tersebut akan diakses oleh perangkat *central* bila data yang disajikan merupakan informasi yang dicari. Komunikasi antara perangkat *peripheral* dan central melibatkan struktur yang terdiri dari *services* dan *characteristics*, serta menggunakan mekanisme seperti *notify*, *indicate*, *read*, dan *write*. Perangkat *peripheral* bertugas mengumpulkan data dan mengirimkannya ke perangkat *central*. Data ini diatur dalam *services*, yang merupakan kumpulan dari *characteristic*. Setiap *characteristic* memiliki UUID berisi nilai data serta properti yang menentukan bagaimana nilai tersebut dapat diakses, termasuk apakah dapat dibaca (*readable*) atau ditulis (*writable*) (*ArduinoBLE*, n.d.).

Mekanisme *notify* memungkinkan *peripheral* untuk mengirim pembaruan nilai characteristic secara otomatis ke perangkat *central* ketika terjadi perubahan data, tanpa memerlukan perangkat central untuk mengirim permintaan *read*. Di sisi lain, *indicate* berfungsi serupa dengan *notify*, tetapi dengan tambahan pengakuan (*acknowledgement*) dari perangkat *central* setelah menerima pembaruan untuk menjamin bahwa data telah diterima dengan sukses. Untuk mekanisme *read*, perangkat *central* mengirim permintaan ke *peripheral* untuk membaca nilai *characteristic* saat itu juga. Mekanisme ini sering digunakan untuk data yang cenderung statis. Sedangkan *write* memungkinkan perangkat *central* untuk mengirim data ke *peripheral* guna mengubah nilai dari *characteristic*, mekanisme

ini sering kali digunakan untuk mengontrol perangkat atau mengubah pengaturannya. (*ArduinoBLE*, n.d.).

2.7 Arduino Nano 33 BLE Sense



Gambar 3 Arduino Nano 33 BLE Sense (Machine Learning on Arduino Nano 33 BLE Sense, 2021)

Arduino Nano 33 BLE Sense dibangun menggunakan pondasi mikrokontroler nRF52840 dan mengoperasikan sistem operasi Arm Mbed. Arduino Nano 33 BLE Sense memiliki kapasitas penyimpanan sebesar 1 MB dan RAM 256 KB. Modul NINA B306 yang tersemat memungkinkan pengguna untuk dapat saling terhubung menggunakan teknologi Bluetooth Low Energy sejauh kurang lebih 50 meter dengan kapasitas transmisi 2 Mbps. Selain itu, Arduino Nano 33 BLE Sense juga dilengkapi dengan berbagai macam sensor untuk mendeteksi warna, kedekatan dan gestur (APDS9960), gerakan (LSM9DS1), suhu dan kelembaban (HTS221), audio (MP34DT05), dan tekanan barometrik (LPS22HB). Papan ini beroperasi pada daya 3.3 V dan 5 V yang optimal untuk penggunaan berdaya rendah jangka panjang (*Arduino Nano 33 BLE Sense*, n.d.).

2.8 Signal Vector Magnitude / Sum Vector Magnitude (SMV)

Akselerometer 3 sumbu adalah sebuah sensor yang menghasilkan estimasi percepatan bernilai nyata di sepanjang sumbu x, y, dan z. Sensor ini mengukur percepatan dan menghasilkan proyeksi vektor percepatan yang direpresentasikan dalam sistem koordinat 3D (Gjoreski & Gams, 2011). Akselerasi atau percepatan adalah sebuah istilah yang diberikan pada suatu proses dimana terdapat perubahan kecepatan (*velocity*). Kecepatan merupakan kelajuan (*speed*) dan arah (*direction*), sehingga terdapat tiga cara untuk berakselerasi, yaitu dengan mengubah kelajuan, mengubah arah, atau mengubah keduanya. Secara spesifik, percepatan

didefinisikan sebagai derajat perubahan pada kecepatan, persamaan 1 menjelaskan hal ini.

$$a = \frac{dv}{dt} = \frac{v_b - v_a}{dt} \quad (1)$$

Dimana akselerasi (a) sama dengan selisih antara kecepatan awal (v_a) dan kecepatan akhir (v_b) dibagi dengan waktu (dt) yang dibutuhkan untuk mengubah kecepatan dari a ke b (*What Is Acceleration?*, 2015).

Satuan percepatan adalah meter per detik kuadrat (m/s^2). Namun, sensor akselerometer biasanya menyatakan pengukuran dalam "g" atau gravitasi. Satu "g" (1g) sama dengan kurang lebih 9.8 m/s^2. Karena gravitasi bumi, semua benda mengalami tarikan gravitasi ke arah pusat bumi. Ketika akselerometer dalam keadaan diam, satu-satunya gaya yang mempengaruhi sensor adalah gravitasi bumi. Akibatnya, semua benda mengalami percepatan 1 g (Gjoreski & Gams, 2011). Menurut Su dan Twu dalam penelitian mereka, penggunaan akselerometer untuk sistem deteksi berbasis akselerometer disarankan untuk diletakkan di saku pinggang alih-alih di tangan atau kaki karena pusat gravitasi manusia berada sekitar tulang panggul (Su & Twu, 2020). Akselerasi tiga dimensi direpresentasikan oleh persamaan 2:

$$\vec{A} = \frac{\mathrm{d}\vec{v}}{\mathrm{dt}} = (\vec{g} + \vec{l}), \begin{pmatrix} A_x \\ A_y \\ A_z \end{pmatrix} = \begin{pmatrix} g_x + l_x \\ g_y + l_y \\ g_z + l_z \end{pmatrix} \quad (2)$$

Dimana akselerasi $(\vec{A})$, akselerasi akibat gravitasi $(\vec{g})$, dan akselerasi linier resultan $(\vec{l})$ diukur dalam m/s^2 (Janidarmian et al., 2017).

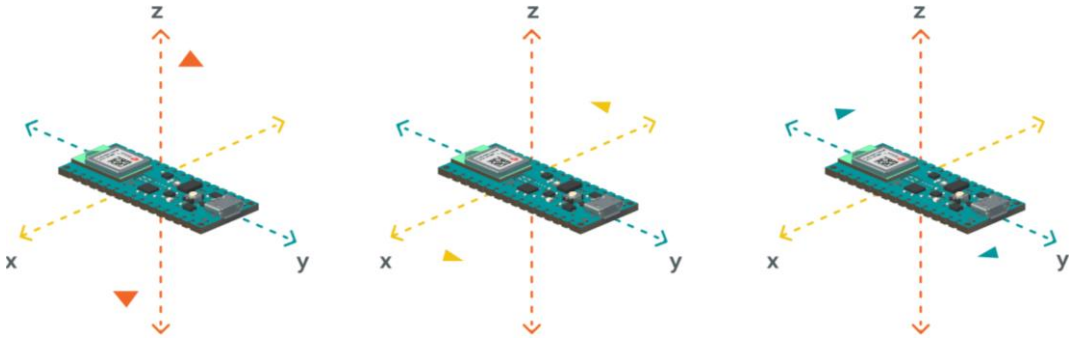
Magnitudo merujuk pada ukuran atau panjang dari sebuah vektor terlepas dari arah vektor tersebut. Rumus magnitudo dijabarkan pada persamaan 3 di bawah:

$$\vec{m} = \sqrt{x^2 + y^2} \quad (3)$$

Dimana x dan y merupakan komponen-komponen yang merepresentasikan jarak atau perpindahan dari vektor tersebut pada masing-masing sumbu dari asalnya ke titik akhirnya. Perpindahan x dari asalnya ke titik akhirnya akan terjadi sepanjang sumbu x saja, begitu pula dengan sumbu y, baik ke arah positif maupun ke arah negatif. Bila terdapat tiga sumbu, maka persamaan magnitudo dijabarkan sebagai berikut:

$$\vec{m} = \sqrt{x^2 + y^2 + z^2} \quad (4)$$

Dimana x, y, dan z merepresentasikan tiap sumbu tiga dimensi sebagaimana yang ditampilkan pada Gambar 4.



Gambar 4 Ilustrasi akselerasi tiga dimensi (Alushi, 2022)

Signal Vector Magnitude, yang seringkali disingkat sebagai SMV untuk menghindari kekeliruan dengan *Support Vector Machine* (SVM), adalah fitur domain waktu yang menghitung total intensitas percepatan (magnitudo) yang dihasilkan oleh data percepatan triaksial. Hasilnya dapat menunjukkan seberapa signifikan perubahan percepatan dalam suatu momen waktu terlepas dari arahnya atau pada sumbu mana perubahan tersebut terjadi. SMV merupakan pengaplikasian spesifik magnitudo dalam konteks pemrosesan sinyal, yaitu dengan cara menggabungkan sinyal-sinyal dari ketiga sumbu akselerometer untuk menghasilkan sebuah nilai skalar yang merepresentasikan derajat intensitas suatu pergerakan pada suatu waktu. Rumus SMV dijabarkan pada persamaan x di bawah:

$$SMV[i] = \sqrt{|A_{xi}|^2 + |A_{yi}|^2 + |A_{zi}|^2} \quad (5)$$

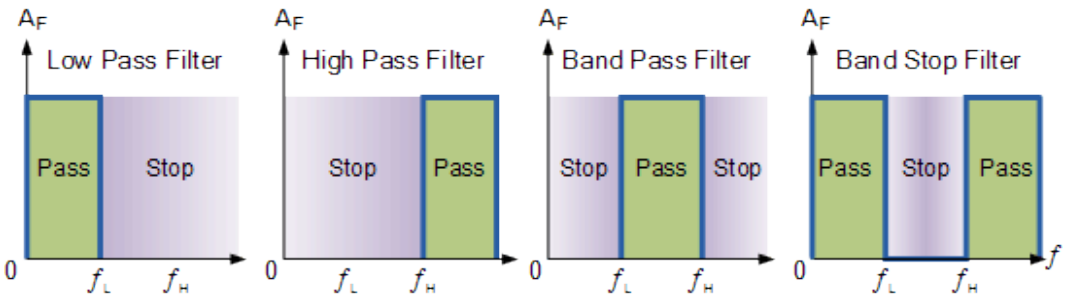
Dimana $SMV[i]$ adalah magnitudo vektor akselerasi ke-i sedangkan A_{xi}, A_{yi}, dan A_{zi} mendefinisikan komponen-komponen dari vektor akselerasi untuk sampel ke-i pada masing-masing vektor x, y, dan z (Al-Kababji et al., 2021; Casilari et al., 2020).

2.9 Filter

Filter digunakan untuk menyaring masukan menurut rentang frekuensi sinyal yang dibolehkan (*pass*) sembari memblokir (*stop*) sisanya. Desain filter yang paling umum digunakan adalah:

1. Filter *Low Pass*: filter *low pass* hanya membolehkan sinyal frekuensi rendah (dari 0 Hz hingga frekuensi *cut-off)* untuk melewatinya sambil memblokir sinyal yang lebih tinggi.
2. Filter *High Pass*: filter *high pass* hanya membolehkan sinyal frekuensi tinggi (dari frekuensi *cut-off* dan selebihnya) untuk melewatinya sambil memblokir sinyal yang lebih rendah.
3. Filter *Band Pass:* filter *band pass* membolehkan sinyal yang berada dalam pengaturan *band* frekuensi tertentu sambil memblokir frekuensi yang lebih rendah dan lebih tinggi daripada kedua sisi *band* frekuensi ini.
4. Filter *Band Stop:* filter ini membolehkan sinyal yang lebih rendah dan lebih tinggi daripada rentang band frequensi tertentu sambal memblokir sinyal pada rentang tengah yang ditentukan.

Gambar 5 mengilustrasikan jenis-jenis filter yang telah disebutkan.



Gambar 5 Jenis-jenis filter (Strorr, 2013)

Gjoreski & Gams dalam penelitian mereka menggunakan filter low-pass guna memperhalus data akselerasi untuk pengenalan aktivitas agar bagian data akselerasi yang disebabkan oleh gravitasi tersaring dari data pergerakan akselerometer itu sendiri (Gjoreski & Gams, 2011). Rumus filter *low-pass* yang digunakan oleh Gjoreski dan Gams adalah sebagai berikut.

$$alpha = 0{,}8$$

$$Low\_X = alpha * Previous\_X + (1 - alpha) * Current\_X \quad (6)$$

Dimana Low_X merupakan nilai yang telah melewati filter, Previous_X merupakan nilai x sebelumnya (x-1), dan Current_X merupakan nilai x saat ini (Gjoreski & Gams, 2011).

2.10 UniMiB SHAR

UniMiB SHAR merupakan sebuah dataset sampel akselerasi yang diperoleh menggunakan *smartphone* Android yang dirancang untuk *human activity recognition* (*HAR*) atau pengenalan aktivitas manusia, dan deteksi jatuh. Dataset ini mencakup 11.771 sampel dari *Activities of Daily Living* (ADL) dan *Fall* (jatuh). Sampel dibagi dalam 17 kelas terperinci yang dikelompokkan dalam dua kelas kasar: 9 jenis aktivitas sehari-hari (ADL) dan 8 jenis jatuh. Perangkat yang digunakan untuk akuisisi data merupakan Samsung Galaxy Nexus I9250 yang memiliki akselerometer Bosch BMA220, merekam data akselerasi dengan frekuensi sampel 50 Hz untuk setiap aktivitas. Tiap vektor data akselerometer terdiri atas 3 vektor dengan 151 nilai (vektor berukuran 1 × 453), satu untuk setiap arah percepatan. Dataset ini terdiri dari 11,771 sampel yang mendeskripsikan ADL (7759) dan jatuh (4192).

Terdapat 30 subjek yang melakukan simulasi ADL dan jatuh yang dengan rentang usia dari 18 hingga 60 tahun dan tinggi badan 160 cm hingga 190 cm. Protokol akuisisi data yang dilakukan yaitu dengan mengenakan *smartphone* yang akan merekam akselerasi dan suara di kantong celana subjek. Subjek menepuk tangan sebelum dan sesudah melakukan tiap aktivitas yang ditentukan. Rekaman audio membantu mengidentifikasi waktu mulai dan berhenti untuk tiap aktivitas. Window sinyal sepanjang 3 detik diekstraksi dari tiap data berpusat pada sebuah lonjakan dimana magnitudo sinyal mt pada waktu t > 1.5 g dan mt-1 pada waktu t-1 < 1.5 g. Segmentasi tersebut digunakan alih-alih menggunakan *overlapped sliding windows* karena dataset ini berfokus pada pengenalan kegiatan dan jatuh berbasis gerakan. Windows berdurasi 3 detik dipilih karena rerata manusia berjalan dengan 90 hingga 130 langkah per menit dan setidaknya satu siklus jalan (dua langkah) terjadi (Micucci et al., 2017).

2.11 Deep Neural Network (DNN)

Jaringan Saraf Tiruan atau *Artificial Neural Network* (ANN) adalah mesin yang dirancang untuk melakukan tugas-tugas tertentu dengan meniru cara kerja otak manusia, dan membangun jaringan saraf yang terdiri dari ratusan atau bahkan ribuan neuron buatan atau unit pemrosesan. Implementasi praktis jaringan saraf

dimungkinkan karena *neural network* merupakan sistem komputasi paralel masif yang terdiri dari sejumlah besar unit pemrosesan dasar (neuron) dan bobot sinapsis. Tugas algoritma pembelajaran tersebut terdiri dari pengubahan bobot sinapsis jaringan secara berurutan untuk mencapai suatu tujuan tertentu (Montesinos-López et al., 2022).

Montesinos-López et al. mendefinisikan *deep learning* sebagai generalisasi dari ANN di mana lebih dari satu *hidden layer* digunakan yang menyiratkan bahwa lebih banyak neuron digunakan untuk mengimplementasikan model. Oleh karena itu, ANN dengan banyak *hidden layer* disebut dengan *Deep Neural Network* (DNN) dan praktik pelatihan jenis jaringan ini disebut *deep learning* (DL), yang merupakan cabang dari *machine learning* statistik di mana topologi *multilayer* digunakan untuk memetakan hubungan antara variabel input (variabel independen) dan variabel respons (hasil). (Montesinos-López et al., 2022).

2.11.1 Batch Size

Batch size merupakan sebuah *hyperparameter* yang mendefinisikan jumlah sampel yang akan diproses sebelum pembaruan parameter model internal. Umpamakan sebuah *batch* sebagai *for-loop* yang mengulang satu atau lebih sampel dan membuat prediksi. Setelah itu, prediksi yang telah dihasilkan kemudian dibandingkan dengan variabel keluaran yang diharapkan, lalu sebuah *error* dihitung. Dari *error* ini, algoritma pembaruan digunakan untuk meningkatkan model. Sebuah set data pelatihan dapat dibagi menjadi satu atau lebih batch (Brownlee, 2022).

2.11.2 Epoch

Epoch merupakan sebuah *hyperparameter* yang mendefinisikan jumlah perulangan yang akan dilalui oleh algoritma pembelajaran untuk memproses seluruh dataset pelatihan. Satu *epoch* berarti bahwa setiap sampel dalam dataset pelatihan memiliki satu kali kesempatan untuk memperbarui parameter model internal. Sebuah *epoch* terdiri dari satu atau beberapa batch (Brownlee, 2022).

Umpamakan sebuah *for-loop* pada jumlah *epoch* di mana setiap perulangan dilakukan pada kumpulan data pelatihan. Di dalam *for-loop* ini terdapat *nested for-loop* lainnya yang mengulang setiap *batch* sampel, di mana satu *batch* memiliki

jumlah sampel “batch size” yang ditentukan. Jumlah *epoch* biasanya besar, sering kali berjumlah ratusan atau ribuan agar algoritma pembelajaran berjalan hingga kesalahan dari model dapat diminimalkan (Brownlee, 2022).

2.11.3 Learning Rate

Learning Rate merupakan sebuah *hyperparameter* yang mengatur seberapa banyak kita menyesuaikan *weight* dari *network* dengan mengacu pada *loss gradient*. Semakin rendah nilainya, semakin lambat pergerakan di sepanjang *slope* penurunan. Meskipun menggunakan *learning rate* yang rendah memungkinkan untuk tidak melewatkan *local minima*, waktu yang dibutuhkan untuk konvergensi akan lebih lama, terutama jika terjebak di daerah *plateau* (Zulkifli, 2018).

Learning rate mengontrol laju atau kecepatan model belajar. Secara spesifik, ia mengontrol jumlah *error* yang diproporsikan dengan bobot model yang diperbarui setiap kali bobot tersebut diperbarui. Dengan konfigurasi *learning rate* yang sempurna, model akan belajar untuk membuat prakiraan terbaik dari fungsi dengan sumber daya yang tersedia (jumlah *layer* dan jumlah node per *layer*) dalam jumlah *epoch* tertentu. Secara umum, *learning rate* yang besar memungkinkan model untuk belajar lebih cepat meski set bobot akhir akan kurang optimal. *Learning rate* yang kecil memungkinkan model untuk mempelajari set bobot yang lebih optimal atau bahkan optimal secara global walau membutuhkan waktu pelatihan yang lebih lama (Brownlee, 2019).

2.11.4 Confusion Matrix

Confusion Matrix adalah tabel silang (*cross table*) yang menunjukkan jumlah kemunculan antara dua nilai kelas yaitu klasifikasi yang sebenarnya atau bernilai aktual dan klasifikasi prediksi, sebagaimana yang ditunjukkan pada Tabel 1, kolom-kolom pada tabel tersebut menunjukkan prediksi model sedangkan baris-baris menunjukkan klasifikasi yang sebenarnya (Grandini et al., 2020).

Tabel 1 *Confusion matrix*

		Kelas Prediksi	
		Positif	Negatif
Kelas Aktual	Positif	TP	FN
	Negatif	FP	TN

Dimana:

TP (*True Positive*): nilai aktual positif dan model memprediksi positif.

TN (*True Negative*): nilai aktual negatif dan model memprediksi negatif.

FP (*False Positive*): nilai aktual negatif, namun model memprediksi positif.

FN (*False Negative*): nilai aktual positif, namun model memprediksi negatif.

Confusion matrix membantu dalam memahami sejauh mana model dapat membedakan antara kelas positif dan negatif serta seberapa sering model menghasilkan prediksi yang salah. Informasi dari *confusion matrix* juga dapat digunakan untuk menghitung metrik evaluasi seperti akurasi, presisi, *recall*, dan *F1-score*. Dalam penelitian ini, digunakan perhitungan akurasi. Akurasi adalah perhitungan jumlah prediksi benar yang dihasilkan oleh model untuk seluruh dataset uji. Akurasi dapat diperoleh dengan menggunakan persamaan x di bawah:

$$Akurasi = \frac{TP + TN}{TP + FP + TN + FN} \times 100\% \quad (7)$$

Dimana TP dan TN pada pembilang adalah elemen-elemen yang diklasifikasikan dengan benar oleh model dan elemen-elemen tersebut berada pada diagonal utama *confusion matrix*, sementara penyebut juga memperhitungkan semua elemen di luar diagonal utama yang telah diklasifikasikan secara keliru oleh model (Grandini et al., 2020).