

TUGAS AKHIR

PEMBUATAN PROGRAM TINY MACHINE LEARNING PADA MIKROKONTROLER UNTUK IDENTIFIKASI RADIONUKLIDA PEMANCAR RADIASI GAMMA

**Diajukan sebagai salah satu syarat
untuk memperoleh gelar Sarjana Terapan Teknik**

**Oleh
DAVID IRFAN JASIR
NIM. 022100010**



**PROGRAM STUDI ELEKTRONIKA INSTRUMENTASI
POLITEKNIK TEKNOLOGI NUKLIR INDONESIA
BADAN RISET DAN INOVASI NASIONAL
2025**

TUGAS AKHIR

PEMBUATAN PROGRAM TINY MACHINE LEARNING PADA MIKROKONTROLER UNTUK IDENTIFIKASI RADIONUKLIDA PEMANCAR RADIASI GAMMA

**Diajukan sebagai salah satu syarat
untuk memperoleh gelar Sarjana Terapan Teknik**

**Oleh
DAVID IRFAN JASIR
NIM. 022100010**



**PROGRAM STUDI ELEKTRONIKA INSTRUMENTASI
POLITEKNIK TEKNOLOGI NUKLIR INDONESIA
BADAN RISET DAN INOVASI NASIONAL
2025**

FINAL PROJECT

**BUILDING OF THE TINY MACHINE LEARNING
PROGRAMS ON MICROCONTROLLER FOR THE
IDENTIFICATION OF GAMMA-RAY-EMITTING
RADIONUCLIDES**

*Proposed as one of the requirement
to obtain Bachelor of Applied Engineering degree*

By
DAVID IRFAN JASIR
Student ID. 022100010



**ELECTRONIC INSTRUMENTATION STUDY PROGRAM
POLYTECHNIC INSTITUTE OF NUCLEAR TECHNOLOGY
NATIONAL RESEARCH AND INNOVATION AGENCY
2025**

HALAMAN PENGESAHAN

PEMBUATAN PROGRAM TINY MACHINE LEARNING PADA MIKROKONTROLER UNTUK IDENTIFIKASI RADIONUKLIDA PEMANCAR RADIASI GAMMA

Disusun oleh

David Irfan Jasir
NIM. 022100010

Telah dipertahankan di depan Dewan Pengaji pada tanggal 2 Juli 2025 dan
dinyatakan telah memenuhi syarat.

Susunan Dewan Pengaji

Ketua Dewan Pengaji



Dr. Maria Christina Prihatiningsih, S.ST., M.Eng
NIP.19661005 198803 2 003

Anggota I



Risky Nurseila Karthika, S.ST., M.Sc
NIP. 19940913 202012 2 008

Anggota II



Toto Trikasjono, S.T, M.Kes
NIP. 19601211 198103 1 005

Mengetahui,
Direktur Politeknik Teknologi Nuklir Indonesia



Dr. Eng. Zainal Arief, S. T., M.T.
NIP. 19670128 19903 1 002

HALAMAN PERSETUJUAN

TUGAS AKHIR

PEMBUATAN PROGRAM TINY MACHINE LEARNING PADA MIKROKONTROLER UNTUK IDENTIFIKASI RADIONUKLIDA PEMANCAR RADIASI GAMMA

Telah diperiksa dan disetujui oleh:

Pembimbing I



Pembimbing II



Dr. Sutanto, M.Eng.
NIP. 19820218 200604 1 016

Toto Trikasjono, S.T, M.Kes
NIP. 19601211 198103 1 005

Mengetahui,
Ketua Program Studi Elektronika Instrumentasi



Dr. Sutanto, M.Eng.
NIP. 19820218 200604 1 016

HALAMAN PERNYATAAN

Yang bertanda tangan di bawah ini:

Nama : David Irfan Jasir
NIM : 022100010
Program Studi : Elektronika Instrumentasi
Judul Tugas Akhir : PEMBUATAN PROGRAM TINY MACHINE LEARNING PADA MIKROKONTROLER UNTUK IDENTIFIKASI RADIONUKLIDA PEMANCAR RADIASI GAMMA

Menyatakan bahwa tugas akhir ini adalah hasil karya penulis dan semua sumber baik yang dikutip maupun dirujuk oleh penulis adalah benar.

Yogyakarta, 1 Juli 2025

Penyusun,



David Irfan Jasir

PRAKATA

Segala puji dan syukur kepada Allah SWT, Tuhan Yang Maha Esa, atas segala rahmat, hidayah, dan petunjuk-Nya sehingga penulis dapat menyelesaikan Laporan Tugas Akhir dengan judul “Pembuatan Program Tiny Machine Learning Pada Mikrokontroler Untuk Identifikasi Radionuklida Pemancar Radiasi Gamma”. Laporan Tugas Akhir ini disusun untuk memenuhi salah satu persyaratan dalam kelulusan pendidikan Diploma Empat Program Studi Elektronika Instrumentasi, Politeknik Teknologi Nuklir Indonesia.

Tugas akhir ini mengambil topik tentang Tiny Machine Learning (selanjutnya disebut sebagai TinyML) merupakan keinginan penulis untuk berkontribusi pada bidang keilmuan ini. Berdasarkan penelusuran penulis, topik tentang TinyML pada bidang radiasi dan kenukliran, khususnya di Indonesia, belum banyak dibahas sehingga penulis berharap topik ini dapat dibahas, didiskusikan, dan dikembangkan lebih lanjut. Penulis juga ingin menghadirkan program TinyML yang dapat digunakan maupun dimodifikasi oleh siapapun untuk dapat dikembangkan dan diterapkan, khususnya pada bidang radiasi dan kenukliran.

Penulis mengucapkan terima kasih kepada berbagai pihak yang mendukung penelitian dan penulisan Laporan Tugas Akhir ini, diantaranya kepada:

1. Allah SWT. Tuhan Yang Maha Esa yang telah memberikan segala karunia, hidayah, dan petunjuk kepada hamba-Nya sehingga proses penelitian Tugas Akhir ini berjalan dengan baik.
2. Kedua orang tua penulis, Bapak Subagyo dan Ibu Sri Suhartatik, yang telah memberikan doa, dukungan, dan semangat kepada penulis. Semoga Allah SWT melimpahkan kenikmatan kepada Bapak dan Ibu. Aamiin.
3. Bapak Dr.Eng. Zainal Arief, S.T., M.T., selaku Direktur Politeknik Teknologi Nuklir Indonesia,

4. Bapak Dr. Eng. Sutanto, M. Eng selaku Ketua Program Studi Elektronika Instrumentasi, Politeknik Teknologi Nuklir Indonesia,
5. Bapak Dr. Eng. Sutanto, M. Eng. selaku dosen pembimbing pertama dan Bapak Toto Trikasjono, S.T., M.Kes, selaku dosen pembimbing kedua, yang telah memberikan arahan dalam pelaksanaan Tugas Akhir ini.
6. Ibu Dr. Maria Christina Prihatiningsih, S.ST., M.Eng dan Ibu Rizky Nurseila Karthika, S.ST., M.Sc. selaku dosen penguji yang telah memberikan penilaian dan saran untuk Tugas Akhir ini.
7. Dosen-dosen Politeknik Teknologi Nuklir Indonesia yang telah membimbing dan memberikan ilmu kepada penulis.
8. Teman-teman Elins 2021 dan angkatan 2021.
9. Berbagai pihak lain yang telah membantu, baik langsung ataupun tidak langsung.

Penulis menyadari terdapat kekurangan dalam Tugas Akhir ini sehingga penulis berharap saran dan kritik serta partisipasi pembaca untuk turut mengembangkan Tugas Akhir ini. Semoga Laporan Tugas Akhir ini memberikan manfaat, inspirasi, dan wawasan kepada penulis dan pembaca.

Yogyakarta, 1 Juli 2025

Penulis,

David Irfan Jasir

DAFTAR ISI

HALAMAN PENGESAHAN	iii
HALAMAN PERSETUJUAN	iv
HALAMAN PERNYATAAN	v
PRAKATA	vi
DAFTAR ISI	viii
DAFTAR GAMBAR	xi
DAFTAR TABEL	xiii
DAFTAR SINGKATAN	xvi
ABSTRAK	xviii
<i>ABSTRACT</i>	xix
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Keaslian Tugas Akhir	3
1.3 Rumusan Masalah	9
1.4 Batasan Masalah	9
1.5 Tujuan	9
1.6 Manfaat	10
BAB 2 TINJAUAN PUSTAKA	11
2.1 Tinjauan Pustaka	11
2.2 Landasan Teori	15
2.2.1 Machine Learning	15
2.2.2 Pengukuran Kinerja Machine Learning Model Klasifikasi	18

2.2.3 <i>Tiny Machine Learning</i>	22
2.2.4 <i>Artificial Neural Network</i>	25
2.2.5 TensorFlow.....	29
2.2.6 Keras	30
2.2.7 TensorFlow Lite	31
2.2.8 Mikrokontroler	32
2.2.9 Radiasi Gamma	35
2.2.10 Detektor Radiasi.....	37
2.2.11 Spektroskopi Gamma	41
2.2.12 Radionuklida	44
2.2.13 Universal Computer Spectrometer UCS30.	47
2.3 Hipotesis.....	49
BAB 3 METODOLOGI	50
3.1 Waktu & Tempat	50
3.2 Alat dan Bahan	50
3.3 Metodologi	58
BAB 4 HASIL DAN PEMBAHASAN.....	71
4.1 Kalibrasi Energi, Efisiensi, dan Resolusi Spektroskopi Gamma	71
4.2 Hasil Pengujian Tidak Langsung dan Langsung Model TinyML	74
4.3 Perbandingan Kinerja Model	118
BAB 5 KESIMPULAN DAN SARAN	124
5.1 Kesimpulan	124
5.2 Saran.....	125
DAFTAR PUSTAKA	126

LAMPIRAN 1 PROGRAM PENGOLAHAN DATASET	131
LAMPIRAN 2 PROGRAM PENYUSUNAN MODEL TINYML	148
LAMPIRAN 3 PROGRAM PENGUJIAN MODEL TINYML BERBASIS PYTHON	164
LAMPIRAN 4 PROGRAM PENGUJIAN MODEL TINYML BERBASIS ARDUINO	169
BIOGRAFI PENULIS	179

DAFTAR GAMBAR

Gambar 2.1 Lingkup AI, ML, dan DL	16
Gambar 2.2 Perbedaan pemrograman konvensional dan ML	16
Gambar 2.3 Ilustrasi jenis eror pada model klasifikasi ML	20
Gambar 2.4 Ilustasi perbedaan ML konvensional dan TinML pada perangkat komputasi terbatas.....	23
Gambar 2.5 Alur Penyusunan program ML dan TinyML	24
Gambar 2.6 Ilustrasi susunan lapisan <i>nodes</i> dalam model ANN	26
Gambar 2.7 Ilustrasi model matematis dalam setiap <i>nodes</i> ANN.....	27
Gambar 2.8 Grafik dari fungsi sigmoid	28
Gambar 2.9 Grafik fungsi ReLU.....	29
Gambar 2.10 Ilustrasi susunan komponen MCU	32
Gambar 2.11 Contoh produk MCU.....	34
Gambar 2.12 Ilustrasi efek fotolistrik	35
Gambar 2.13 Ilustrasi efek hamburan Compton	36
Gambar 2.14 Ilustrasi efek produksi pasangan	37
Gambar 2.15 Ilustrasi susunan detektor isian gas	38
Gambar 2.16 Ilustrasi susunan detektor sintilasi.....	39
Gambar 2.17 Ilustrasi susunan detektor semikonduktor.....	40
Gambar 2.18 skema peluruhan Cs-137	45
Gambar 2.19 Spektrum radiasi gamma Cs-137	45
Gambar 2.20 skema peluruhan Co-60.....	46
Gambar 2.21 Spektrum energi radiasi gamma Co-60.....	46
Gambar 2.22 Modul hardware UCS30	48
Gambar 2.23 Tampilan software UCS30	48
Gambar 3.1 a) Detektor sintilasi dan b) Ilustrasi susunan komponen detektor	53
Gambar 3.2 Perangkat komputer akuisisi data spektroskopi gamma beserta detektor NaI(Tl) dan UCS30	54
Gambar 3.3 ESP32	55

Gambar 3.4 Susunan perangkat keras	57
Gambar 3.5 Alur pelaksanaan tugas akhir.....	57
Gambar 3.6 Grafik salah satu dataset Co-60.....	63
Gambar 3.7 Grafik salah satu dataset Cs-137	64
Gambar 3.8 Grafik dataset pelatihan.....	66
Gambar 3.9 Grafik dataset validasi.....	66
Gambar 3.10 Grafik dataset pengujian.....	66
Gambar 3.11 Flowchart program a) Python dan b) Arduino untuk pengujian langsung model TinyML	69
Gambar 4.1 Kurva kalibrasi energi	72
Gambar 4.2 <i>Confusion matrix</i> pengujian tidak langsung model A.1	75
Gambar 4.3 <i>Confusion matrix</i> pengujian tidak langsung model A.2	79
Gambar 4.4 <i>Confusion matrix</i> pengujian tidak langsung model A.3	82
Gambar 4.5 <i>Confusion matrix</i> pengujian tidak langsung model A.4	86
Gambar 4.6 <i>Confusion matrix</i> pengujian tidak langsung model A.5	90
Gambar 4.7 <i>Confusion matrix</i> pengujian tidak langsung model A.6	94
Gambar 4.8 <i>Confusion matrix</i> pengujian tidak langsung model B.1	98
Gambar 4.9 <i>Confusion matrix</i> pengujian tidak langsung model B.2	102
Gambar 4.10 <i>Confusion matrix</i> pengujian tidak langsung model B.3	105
Gambar 4.11 <i>Confusion matrix</i> pengujian tidak langsung model B.4	109
Gambar 4.12 <i>Confusion matrix</i> pengujian tidak langsung model B.5	112
Gambar 4.13 <i>Confusion matrix</i> pengujian tidak langsung model B.6	116

DAFTAR TABEL

Tabel 1.1 Referensi Tugas Akhir	3
Tabel 2.1 Contoh Tabel <i>Confusion matrix</i>	19
Tabel 2.2 Perbandingan spesifikasi produk-produk MCU	34
Tabel 2.3 Tabel perbandingan kualitas kinerja ketiga tipe detektor radiasi	40
Tabel 3.1 Waktu pelaksanaan tugas akhir	50
Tabel 3.2 Perangkat yang digunakan	50
Tabel 3.3 Spesifikasi radionuklida yang digunakan.....	51
Tabel 3.4 Konfigurasi Perangkat Spektroskopi Gamma	60
Tabel 4.1 Titik kalibrasi energi.....	71
Tabel 4.2 Hasil perhitungan kalibrasi efisiensi	73
Tabel 4.3 Resolusi spektroskopi gamma	73
Tabel 4.4 Susunan model A.1.....	74
Tabel 4.5 <i>Confusion matrix</i> pengujian langsung model A.1	75
Tabel 4.6 Metrik kinerja pengujian tidak langsung model A.1	76
Tabel 4.7 Metrik kinerja pengujian langsung model A.1	77
Tabel 4.8 Susunan model A.2.....	78
Tabel 4.9 <i>Confusion matrix</i> pengujian langsung model A.2	79
Tabel 4.10 Metrik kinerja pengujian tidak langsung model A.2	80
Tabel 4.11 Metrik kinerja pengujian langsung model A.2	81
Tabel 4.12 Susunan model A.3.....	82
Tabel 4.13 <i>Confusion matrix</i> pengujian langsung model A.3	83
Tabel 4.14 Metrik kinerja pengujian tidak langsung model A.3	83
Tabel 4.15 Metrik kinerja pengujian langsung model A.3	84
Tabel 4.16 Susunan model A.4.....	85
Tabel 4.17 <i>Confusion matrix</i> pengujian langsung model A.4	86
Tabel 4.18 Metrik kinerja pengujian tidak langsung model A.4	87
Tabel 4.19 Metrik kinerja pengujian langsung model A.4	87
Tabel 4.20 Susunan model A.5.....	89

Tabel 4.21 <i>Confusion matrix</i> pengujian langsung model A.5	90
Tabel 4.22 Metrik kinerja pengujian tidak langsung model A.5	91
Tabel 4.23 Metrik kinerja pengujian langsung model A.5	91
Tabel 4.24 Susunan model A.6.....	94
Tabel 4.25 <i>Confusion matrix</i> pengujian langsung model A.6	95
Tabel 4.26 Metrik kinerja pengujian tidak langsung model A.6	95
Tabel 4.27 Metrik kinerja pengujian langsung model A.6	96
Tabel 4.28 Susunan model B.1	98
Tabel 4.29 <i>Confusion matrix</i> pengujian langsung model B.1	99
Tabel 4.30 Metrik kinerja pengujian tidak langsung model B.1	99
Tabel 4.31 Metrik kinerja pengujian langsung model B.1	100
Tabel 4.32 Susunan model B.2.....	101
Tabel 4.33 <i>Confusion matrix</i> pengujian langsung model B.2	102
Tabel 4.34 Metrik kinerja pengujian tidak langsung model B.2	103
Tabel 4.35 Metrik kinerja pengujian langsung model B.2	103
Tabel 4.36 Susunan model B.3.....	105
Tabel 4.37 <i>Confusion matrix</i> pengujian langsung model B.3	106
Tabel 4.38 Metrik kinerja pengujian tidak langsung model B.3	106
Tabel 4.39 Metrik kinerja pengujian langsung model B.3	107
Tabel 4.40 Susunan model B.4.....	108
Tabel 4.41 <i>Confusion matrix</i> pengujian langsung model B.4	109
Tabel 4.42 Metrik kinerja pengujian tidak langsung model B.4	110
Tabel 4.43 Metrik kinerja pengujian langsung model B.4	110
Tabel 4.44 Susunan model B.5.....	112
Tabel 4.45 <i>Confusion matrix</i> pengujian langsung model B.5	113
Tabel 4.46 Metrik kinerja pengujian tidak langsung model B.5	113
Tabel 4.47 Metrik kinerja pengujian langsung model B.5	114
Tabel 4.48 Susunan model B.6.....	115
Tabel 4.49 <i>Confusion matrix</i> pengujian langsung model B.6	116
Tabel 4.50 Metrik kinerja pengujian tidak langsung model B.6	117

Tabel 4.51 Metrik kineja pengujian langsung model B.6	117
Tabel 4.52 Tabel perbandingan kinerja semua model	120

DAFTAR SINGKATAN

Singkatan	Arti
A	Aktivitas radionuklida
AI	Artificial Intelligence
Am	Americium
ANN	Artificial Neural Network
API	Application Programming Interface
Ba	Barium
Ch	Channel
Ci	Curie
Co	Cobalt (Kobalt)
CPU	Central Processing Unit
Cps	cacah per sekon
Cs	Cesium
csv	comma separated value
d	jarak detektor dan radionuklida
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DDPG	Deep Deterministic Policy Gradient
DL	Deep Learning
E	Energi
Ef	Efisiensi
f	faktor geometri
FN	False Negative
FP	False Positive
FWHM	<i>Full Width Half Modulation</i>
GPU	Graphic Processing Unit
HPGe	High Purity Germanium
IDE	Integrated Development Environment

IEEE	Institute of Electrical and Electronics Engineers
In	Indium
K	Potassium
KeV	Kilo electron volt
KNN	K-Nearest Neighbors
LiPo	Lithium Polymer
MCA	Multi Channel Analyzer
MCTS	Monte Carlo Tree Research
MCU	Microcontroller Unit
MeV	Mega electron volt
ML	Machine Learning
MLR	Multiple Linier Regression
MSE	Mean Square Error
Na	Natrium
NaI(Tl)	thallium-doped sodium iodide
NLP	Natural Language Processing
PC	Personal Computer
PCB	Printed Circuit Board
PM	Photo Multiplier
Pu	Plutonium
r	jari-jari detektor
RAM	Natural Language Processing
RMSE	Root Mean Square Error
SiPM	Silicon Photo Multiplier
SVM	Support Vector Machine
TinyML	Tiny Machine Learning
TN	True Negative
TP	True Positive
XGboost	Extra Gradient Boost
Y	yield

ABSTRAK

Identifikasi radionuklida menggunakan spektroskopi gamma dapat dilakukan dengan *Machine Learning* (ML). Program ML sulit diterapkan pada perangkat *Microcontroller Unit* (MCU). Saat ini terdapat teknik pemrograman *Tiny Machine Learning* (TinyML) yang dapat diterapkan pada perangkat MCU. Tujuan penelitian tugas akhir ini adalah untuk membuat, menguji, serta menganalisis kinerja program TinyML untuk identifikasi radionuklida berbasis spektroskopi gamma. Tahapan penelitian ini dimulai dari perencanaan, pengambilan dataset spektroskopi gamma radionuklida Co-60 dan Cs-137 dengan menggunakan detektor sintilasi dan modul spektroskopi UCS30, pengolahan dataset, pembuatan berbagai program model dalam bentuk *Artificial Neural Network* (ANN) dengan memvariasikan jumlah *layers* dan *nodes*, pengujian tidak langsung pada *software* Visual Studio Code, pengujian langsung pada perangkat MCU ESP32, dan analisis hasil pengujian. Kinerja hasil pengujian setiap model bervariasi. Model dengan kinerja yang baik, efektif, dan optimal adalah model dengan nilai metrik kinerja yang tinggi dan seimbang. Model tersebut adalah model A.3 dengan struktur ANN berupa tiga layer serta jumlah *nodes* setiap *layers* sebanyak 30, 15, dan 1. Model A.3 mampu mengidentifikasi kedua radionuklida dengan nilai akurasi sebesar 0,73; nilai presisi sebesar 0,73 dan 0,74; nilai *recall* sebesar 0,73 dan 0,75; serta nilai *F1-Score* sebesar 0,74. Berbagai model lain sebagian besar memiliki nilai metrik kinerja yang cukup tinggi namun tidak seimbang. Kinerja dari setiap model tersebut dipengaruhi oleh susunan model berupa jumlah *layers*, *nodes*, dan fungsi aktivasi.

Kata kunci : Artificial Neural Network, Machine Learning, Microcontroller Unit, Radionuklida, Spektroskopi Gamma, Tiny Machine Learning.

ABSTRACT

Radionuclide identification using gamma spectroscopy can be done with Machine Learning (ML). However, ML programmes are difficult to implement on Microcontroller Unit (MCU) devices. Currently there is a Tiny Machine Learning (TinyML) programming technique that can be applied to MCU devices. The purpose of this final project research is to create, test, and analyse the performance of the TinyML program for gamma spectroscopy-based radionuclide identification. The stages of this research start from planning, getting gamma spectroscopy datasets of Co-60 and Cs-137 radionuclides using a scintillation detector and UCS30 spectroscopy module, dataset processing, making various model programs in the form of Artificial Neural Network (ANN) by varying the number of layers and nodes, indirect testing on Visual Studio Code software, direct testing on ESP32 MCU devices, and analysis of test results. The performance of the test results of each model varies. The model with good, effective, and optimal performance is model A.3 with an ANN structure of three layers and the number of nodes in each layer of 30, 15, and 1. Model A.3 is able to identify both radionuclides with an accuracy value of 0.73; precision values of 0.73 and 0.74; recall values of 0.73 and 0.75; and F1-Score value of 0.74. The other models mostly had fairly high but unbalanced performance metric values. The performance of each model is influenced by the arrangement of the model in the form of the number of layers, nodes, and activation functions.

Keywords: ***Artificial Neural Network, Gamma Spectroscopy, Machine Learning, Microcontroller Unit, Radionuclides, Tiny Machine Learning.***

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Terdapat berbagai jenis radionuklida. Setiap radionuklida memiliki karakteristik khusus yang membedakan dengan radionuklida lain. Salah satu teknik untuk mengidentifikasi radionuklida adalah spektroskopi gamma. Dengan teknik ini, radionuklida, khususnya yang memancarkan radiasi gamma, akan diukur pancaran energi dan cacah radiasinya dan diinterpretasikan sebagai spektrum radiasi dalam bentuk grafik (x,y). Setiap radionuklida memiliki bentuk grafik spektrum radiasi yang unik dan berbeda-beda. Hasil spektroskopi gamma dianalisis untuk mengidentifikasi nama radionuklida.

Dalam melakukan analisis spektroskopi gamma untuk mengidentifikasi radionuklida, terdapat beberapa tantangan. Dalam publikasi ilmiah oleh M. Rawool-Sullivan, dkk pada tahun 2010 menjelaskan proses identifikasi radionuklida menggunakan teknik spektroskopi yang dilakukan oleh ahli spektroskopi beserta sejumlah tantangannya. Analisis spektroskopi gamma untuk mengidentifikasi radionuklida memerlukan pengetahuan, keterampilan, dan pengalaman dalam keilmuan fisika nuklir, penggunaan perangkat spektroskopi gamma, dan teknik analisisnya. Dalam spektroskopi gamma dan analisisnya untuk mengidentifikasi radionuklida, sejumlah tantangan yang dapat terjadi ialah kualitas peralatan spektroskopi yang digunakan, proses kalibrasi perangkat spektroskopi, spektrum gamma yang kompleks dari material yang tersusun dari berbagai radionuklida, dan efek radiasi hamburan dari perangkat *shielding* serta radiasi latar belakang.

Untuk mengatasi sejumlah tantangan tersebut, proses analisis spektroskopi gamma untuk mengidentifikasi radionuklida dapat dilakukan dengan menggunakan teknologi *Machine Learning* (ML) pada perangkat komputasi tertentu. Program ML pada perangkat komputer dapat mempelajari pola spektrum radiasi hingga mengidentifikasi radionuklida berdasarkan pada hasil pengukuran dari modul detektor radiasi gamma. Akurasi dan keandalan dari proses spektroskopi gamma dapat meningkat dengan menggunakan ML. Algoritma-algoritma ML seperti *Support Vector*

Machines (SVM), *K-Nearest Neighbors* (KNN), dan *Artificial Neural Networks* (ANN) telah diteliti dan digunakan dalam peningkatan kemampuan prediksi dari sistem spektroskopi gamma. Metode ini mampu dalam mengolah data secara *real-time* dengan akurasi dan keandalan yang tinggi (Zehtabvar, dkk. 2024).

Sebagian besar program ML memerlukan perangkat komputasi dengan spesifikasi tertentu dan sulit untuk diterapkan pada perangkat dengan sumber daya komputasi terbatas seperti *microcontroller unit* (MCU). Saat ini telah berkembang teknik pemrograman *Tiny Machine Learning* (TinyML) untuk diterapkan pada MCU sehingga perangkat tersebut dapat menjalankan program ML. Fokus utama dalam penelitian pengembangan TinyML adalah bagaimana menjalankan program ML pada perangkat dengan sumber daya komputasi terbatas. TinyML memberikan kemampuan bagi perangkat MCU untuk melakukan akuisisi data sekaligus analisis data dengan algoritma ML secara langsung tanpa perlu mengirim data menuju komputer pusat terlebih dahulu.

TinyML telah diteliti dan diterapkan untuk melakukan berbagai tugas tertentu, seperti untuk identifikasi-klasifikasi objek secara visual (Ziliwu, dkk. 2024), prediksi kecepatan angin (Kiang Hong, dkk, 2023), dan prediksi polusi udara (Wardana, dkk. 2024). Penelitian dan penggunaan TinyML di bidang radiasi dan kenukliran belum banyak publikasi ilmiahnya. (Istofa, dkk. 2023). Hingga penulisan tugas akhir ini, terdapat satu publikasi terkait penggunaan TinyML di bidang radiasi dan kenukliran, yaitu untuk penelitian perancangan perangkat identifikasi radionuklida berdasarkan spektroskopi radiasi gamma yang memiliki program TinyML pada perangkat MCU-nya (Altayeb, dkk. 2023).

Penelitian tugas akhir ini akan membahas mengenai pembuatan dan penerapan program TinyML pada perangkat MCU, yaitu ESP32, untuk identifikasi radionuklida pemancar radiasi gamma beserta analisis kinerja program TinyML tersebut.

1.2 Keaslian Tugas Akhir

Pelaksanaan Tugas Akhir ini didasarkan pada berbagai penelitian-penelitian sebelumnya yang terkait dengan TinyML dan penggunaan program ML pada spektroskopi gamma diantaranya:

Tabel 1.1 Referensi Tugas Akhir

No	Nama	Judul Karya Ilmiah	Deskripsi
1	Istofa, Prawito Prajitno, I Putu Susila	<i>A Systematic Literature Review of TinyML for Environmental Radiation Monitoring System</i>	Publikasi penelitian mengenai penggunaan TinyML untuk pemantauan radiasi lingkungan belum banyak beredar. Publikasi penelitian ini berupa studi literatur mengenai berbagai penggunaan TinyML serta potensi pengembangan dan penelitian terkait dengan TinyML untuk pemantauan radiasi lingkungan. Studi literatur menggunakan metode <i>Systematic Literature Review</i> terhadap publikasi ilmiah berupa jurnal dan prosiding dari Scopus, ScienceDirect, IEEE Xplore, dan Web of Science. Terdapat 34 publikasi ilmiah yang dipilih dengan kriteria pemilihan kata kunci tertentu, inklusi-eksklusi, dan ekstraksi data dengan Mendeley dan Microsoft Excel. Berdasarkan pada sejumlah publikasi ilmiah tersebut diketahui bahwa teknologi TinyML

No	Nama	Judul Karya Ilmiah	Deskripsi
			<p>merupakan kombinasi perangkat keras dan perangkat lunak tertentu untuk dapat menjalankan program ML pada perangkat berdaya listrik terbatas, memori terbatas, dan kemampuan komputasi terbatas. TinyML telah diterapkan sebagai pemantauan lahan pertanian, identifikasi suara, deteksi gas berbahaya, deteksi aktivitas manusia, deteksi anomali peralatan, hingga prediksi cuaca. Berbagai contoh penerapan tersebut menggunakan perangkat MCU dari jenis dan spesifikasi yang beragam serta program TinyML yang beragam. Disimpulkan juga bahwa terdapat potensi bagi pengembangan TinyML untuk pemantauan radiasi lingkungan.</p>
2	Moez Altayeb, Marco Zennaro, Ermanno Pietrosemoli	<i>TinyML Gamma Radiation Classifier</i>	<p>Tim peneliti mendesain sistem deteksi dan identifikasi radiasi gamma otomatis dalam satu sistem tertanam (<i>embedded system</i>). Susunan rancangan sistem tersebut terdiri dari sensor radiasi <i>Silicon Multiplier</i> (SiPM), rangkaian komponen elektronik untuk</p>

No	Nama	Judul Karya Ilmiah	Deskripsi
			<p>pengolahan sinyal sensor, <i>pulse filtering and shaping</i>, dan mikrokontroler ATSAMD51P. Susunan sistem tersebut terpasang pada papan <i>Printed Circuit Board</i> (PCB) empat lapis. Pada mikrokontroler diprogram TinyML menggunakan <i>framework</i> Edge Impulse. Dataset untuk pelatihan dan pengujian TinyML berupa radionuklida pemancar radiasi gamma Co-60, Cs-137, Na-22, Am-241, dan Ba-133. Program TinyML yang disusun di Edge Impulse merupakan program dengan algoritma <i>Convolutional Neural Network</i>. Diperoleh hasil pengujian terhadap kelima radionuklida tersebut dengan nilai akurasi yang tinggi, berkisar antara 95,5% hingga 100% dengan nilai <i>F1-Score</i> 0,96 hingga 1,00.</p>
3	Bayu Rukmana Jati	Perbandingan Akurasi Prediksi dan Kecepatan Proses Antar <i>Classifier Machine Learning</i> Untuk Klasifikasi Spektrum Gamma Co-	Pada skripsi penelitian ini, penulis mencoba berbagai jenis <i>classifier machine learning</i> seperti <i>Decision Tree</i> , <i>Random Forest</i> , <i>AdaBoost</i> , <i>Naive Bayes</i> , <i>Gaussian Process</i> , dan <i>Support Vector</i> , untuk melakukan

No	Nama	Judul Karya Ilmiah	Deskripsi
		60, Na-22, Am-241, Cs-137, Sr-90	klasifikasi radiasi bahan-bahan radioaktif Co-60, Na-22, Am-241, Cs-137, Sr-90, serta mengukur kecepatan proses dan akurasi prediksinya. Pengambilan data spektrum radiasi bahan-bahan radioaktif tersebut menggunakan detektor sintilator NaI(Tl) dengan hasil berupa akurasi prediksi dan kecepatan proses berbeda-beda untuk tiap jenis <i>classifier</i> dengan metode <i>classifier</i> terunggul adalah <i>classifier Support Vector Machine</i> dengan akurasi 100% dan kecepatan proses 0,244 detik.
4	John-Ryan Romo, Kai Tyrus Nelson, Mateusz Monterial, Karl E. Nelson, Simon E. Labov, Adam Hecht	Classifier Comparison for Radionuclide Identification from Gamma-ray Spectra	Penelitian ini membahas bagaimana kinerja algoritma ML untuk identifikasi dan klasifikasi spektrum gamma dari berbagai radionuklida. Sebanyak 33 jenis radionuklida digunakan dalam penelitian ini, beberapa diantaranya Cs-137, Co-60, Na-22, In-111, K-40, dan lain sebagainya. Data spektrum gamma dari semua radionuklida tersebut diperoleh dengan menggunakan detektor Natrium Ioda (NaI). Algoritma ML yang digunakan yaitu

No	Nama	Judul Karya Ilmiah	Deskripsi
			<p><i>Multilayer Perceptron Artificial Neural Network (ANN), Extreme Gradient Boost Trees (XGBoost), dan Random Forest.</i> Kinerja tiap jenis algoritma untuk identifikasi dan klasifikasi setiap jenis radionuklida diukur dengan perhitungan nilai <i>F1-Score</i>. Hasil klasifikasi untuk tiap radionuklida dan tiap algoritma berbeda-beda. Hasil klasifikasi terbaik berupa klasifikasi radionuklida In-111 dengan algoritma <i>Multilayer Perceptron ANN</i> dengan nilai <i>F1-Score</i> 0,995. Hasil klasifikasi terburuk berupa klasifikasi Pu-239 dengan algoritma <i>Random Forest</i> dengan nilai <i>F1-Score</i> sebesar 0,337. Secara keseluruhan, jenis algoritma terbaik untuk klasifikasi radionuklida berdasarkan spektrum radiasi gamma adalah <i>XGBoost</i> dengan nilai rata-rata <i>F1-Score</i> sebesar 0,910. Nilai <i>F1-Score</i> <i>Multilayer Perceptron ANN</i> sebesar 0,888. Nilai <i>F1-Score</i> <i>Random Forest</i> sebesar 0.859.</p>

No	Nama	Judul Karya Ilmiah	Deskripsi
5	Chua Kiang Hong, Mohd Azlan Abu, Mohd Ibrahim Shapiai, Mohamad Fadzli Haniff, Rahdir Sham Mohamad, Aminudin Abu	<i>Analysis of Wind Speed Prediction using Artificial Neural Network and Multiple Linear Regression Model using Tinyml on Esp32</i>	Penelitian ini membandingkan dua jenis algoritma TinyML yaitu <i>Artificial Neural Network</i> dan <i>Multiple Linier Regresion</i> untuk memprediksi kecepatan angin dengan menggunakan perangkat mikrokontroler ESP32 dan sensor DHT11. Data yang dikumpulkan dari perangkat sensor adalah data pengukuran suhu dan kelembapan. Akuisisi data pengukuran dilakukan di daerah Setapak, Kuala Lumpur. Penyusunan program TinyML menggunakan <i>library</i> TensorFlow Lite Micro dan Eloquent TinyML. Hasil perkiraan dari dua jenis algoritma TinyML menunjukkan tingkat akurasi sebesar 71.029% untuk algoritma <i>Multiple Linier Regression</i> dan 75,866% untuk algoritma <i>Artificial Neural Network</i> .

Berdasarkan pada referensi penelitian-penelitian tersebut, maka kebaruan penelitian Tugas Akhir ini terdapat pada penggunaan jenis perangkat MCU yaitu ESP32 yang diprogram untuk mengidentifikasi radionuklida pemancar radiasi gamma berdasarkan pada pengukuran spektroskopi gamma.

1.3 Rumusan Masalah

Berdasarkan latar belakang tersebut, permasalahan dalam penelitian ini dirumuskan sebagai berikut.

1. Bagaimana proses pembuatan dan pengujian program TinyML pada perangkat MCU untuk identifikasi radionuklida pemancar radiasi gamma?
2. Bagaimana kinerja program TinyML yang dikembangkan pada perangkat MCU untuk identifikasi radionuklida pemancar radiasi gamma?
3. Apa yang mempengaruhi kinerja program TinyML pada perangkat MCU untuk identifikasi radionuklida pemancar radiasi gamma?
4. Bagaimana kinerja model TinyML terbaik dan optimal dalam penggunaan pada perangkat MCU untuk identifikasi radionuklida pemancar radiasi gamma?

1.4 Batasan Masalah

Rumusan masalah tersebut dibatasi pada lingkup sebagai berikut.

1. Menggunakan radionuklida pemancar radiasi gamma yang tersedia di Laboratorium Instrumentasi Nuklir yaitu Cobalt-60 (Co-60) dengan kode 10, 11 dan 12 serta Cesium-137 (Cs-137) dengan kode 16, 19 dan 21.
2. Berfokus pada penyusunan dan pengujian program TinyML menggunakan algoritma jenis klasifikasi Artificial Neural Network.
3. Memvariasikan jumlah *layers* sebanyak 3 atau 4 *layers*, dan jumlah *nodes* dari 1 hingga 30 *nodes*.
4. Menggunakan perangkat MCU dari jenis ESP32-WROOM-32.

1.5 Tujuan

Penelitian Tugas Akhir ini bertujuan untuk

1. Memahami proses pembuatan dan pengujian program TinyML pada perangkat MCU untuk identifikasi radionuklida pemancar radiasi gamma.

2. Mengevaluasi kinerja program TinyML pada perangkat MCU untuk identifikasi radionuklida pemancar radiasi gamma dengan target kinerja akurasi program sebesar 80%.
3. Mengidentifikasi faktor yang mempengaruhi kinerja penggunaan TinyML pada perangkat MCU untuk identifikasi radionuklida pemancar radiasi gamma.

1.6 Manfaat

Penelitian Tugas Akhir ini bermanfaat untuk

1. Bagi penulis, sebagai sarana penerapan ilmu pengetahuan mengenai *Machine Learning* dan Pengukuran Spektroskopi Gamma.
2. Menunjukkan proses pembuatan program TinyML pada perangkat MCU untuk identifikasi zat radioaktif pemancar radiasi gamma.
3. Menjadi referensi bagi penelitian selanjutnya terkait dengan penerapan program TinyML pada perangkat MCU untuk identifikasi radionuklida gamma maupun untuk pengukuran radiasi.

BAB 2 TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Publikasi penelitian tahun 2023 oleh Istofa, Prawito Prajitno, dan I Putu Susila dengan judul “A Systematic Literature Review of TinyML for Environmental Radiation Monitoring System” merupakan studi literatur mengenai penggunaan TinyML pada sistem pemantauan radiasi lingkungan. Studi literatur dengan teknik *Systematic Literature Review* untuk menentukan kebutuhan dan spesifikasi untuk pengembangan sistem pemantauan radiasi lingkungan di waktu mendatang. Peneliti mengumpulkan publikasi ilmiah sejumlah 335 judul dari sumber jurnal dan prosiding dari basis data ilmiah Scopus, ScienceDirect, IEEE Xplore, dan Web of Science. Pemilihan dan seleksi publikasi ilmiah memperhatikan judul, abstrak, dan kata kunci, menghapus duplikasi, serta menggunakan kriteria inklusi dan eksklusi tertentu dengan mengikuti aspek tren publikasi ilmiah, perangkat keras dan studi kasus yang digunakan. Proses seleksi tersebut menghasilkan sejumlah 34 karya terpilih untuk dianalisis lebih lanjut. Dari sejumlah karya ilmiah tersebut, diketahui bahwa TinyML merupakan kombinasi dari perangkat keras dan perangkat lunak untuk mengaplikasikan kemampuan *deep learning* pada berbagai jenis mikrokontroler yang memiliki kemampuan komputasi yang terbatas. Contoh mikrokontroler yang telah diteliti kemampuannya menggunakan TinyML adalah Arduino Nano 33, STM32F, ESP32, Raspberry Pi 4B, Arduino Portenta H7, Nordic Semi nRF52840 DK, dan ATSAMD51G19A. Beberapa contoh studi kasus penggunaan TinyML pada perangkat-perangkat tersebut diantaranya sebagai prediksi suhu rumah kaca, deteksi keamanan jalur suplai makanan, sistem kemudi otomatis kendaraan mini, prediksi lingkungan, klasifikasi gambar, dan pengenalan suara. Berdasarkan studi tersebut, terdapat potensi pengembangan dan penggunaan TinyML sebagai sistem deteksi radiasi lingkungan. Sistem ini dapat dikembangkan berkaitan dengan sistem caca radiasi, transmisi data, dan manajemen perangkat.

Publikasi penelitian oleh Moez Altayeb, Marco Zennaro, dan Ermanno Pietrosemoli pada tahun 2023 dengan judul “TinyML Gamma Radiation Classifier”

membahas mengenai perancangan sistem deteksi dan identifikasi radiasi gamma otomatis berbasis TinyML. Rancangan yang dikembangkan berupa perangkat sistem tertanam dari PCB 4 lapis yang terdiri dari komponen detektor *Silicon Photomultiplier* (SiPM), rangkaian komponen elektronik pengolah sinyal, dan mikrokontroler ATSAMD51P yang telah diprogram dengan model *Machine Learning*. Perangkat dilatih dan diuji dengan menggunakan sumber radiasi gamma Co-60, Cs-137, Na-22, Am-241, dan Ba-133. Sumber radiasi tersebut diakuisisi sebagai dataset spektroskopi dengan *silicon photomultiplier* dan ditingkatkan ukuran datanya dengan MATLAB. Data tersebut digunakan dalam tahapan pelatihan dan validasi model TinyML. Penyusunan model TinyML menggunakan *platform* Edge Impulse dengan algoritma jenis *Convolution Neural Network*. Desain perangkat yang dihasilkan berukuran kecil dengan konsumsi daya listrik yang rendah. Besar konsumsi listriknya yaitu arus listrik sebesar 100 mA dengan tegangan 4,5 V dengan daya listrik 0,45 W, dapat beroperasi dengan satu baterai LiPo 7,4 Wh selama 16,4 jam. Hasil pengujian model TinyML untuk mendeteksi dan mengidentifikasi kelima radionuklida tersebut menunjukkan kemampuan perangkat untuk memproses data secara real-time dengan akurasi sebesar 95,5% hingga 98,3% dengan nilai *F1-Score* sebesar 0,96 hingga 1,0.

Skripsi penelitian berjudul “Perbandingan Akurasi Prediksi dan Kecepatan Proses Antar *Classifier Machine Learning* Untuk Klasifikasi Spektrum Gamma Co-60, Na-22, Am-241, Cs-137, Sr-90” yang ditulis oleh Bayu Rukmana Jati dan diterbitkan pada tahun 2020, membahas mengenai kemampuan dari tiap jenis algoritma klasifikasi ML dalam mengidentifikasi jenis-jenis bahan radioaktif. Kemampuan yang dianalisis yaitu kecepatan proses identifikasi dan akurasi prediksi spektrum gamma radionuklida. Data spektrum gamma diperoleh dari detektor sintilasi Natrium Ioda Talium atau NaI(Tl) dari perangkat Leybold Didactic dan aplikasi Cassy Lab. Bahan – bahan radioaktif yang digunakan terdiri dari Co-60, Na-22, Am-241, Cs-137, dan Sr-90. Data spektrum gamma dari bahan – bahan radioaktif tersebut diuji pada berbagai jenis model klasifikasi ML, diantaranya *Decision Tree*, *Random Forest*, *AdaBoost*, *Naive Bayes*, *Gaussian Process*, dan *Support Vector*. Setiap model ML *classifier* tersebut dilakukan

pelatihan dengan data pelatihan. Selanjutnya dilakukan proses validasi untuk mengetahui hasil awal akurasi prediksi data dengan menggunakan data validasi. Model yang tervalidasi akan diujicoba dengan data pengujian. Hasil akurasi pengujian klasifikasi model ML berupa nilai persentase yang ditampilkan dalam grafik untuk setiap variasi parameter klasifikasi model ML dan *Confusion matrix* untuk perbandingan ketepatan prediksi dan kondisi riil. Hasil pengujian juga ditampilkan dalam bentuk *Confusion matrix* yang membandingkan ketepatan jumlah hasil prediksi dengan kondisi riilnya. Setiap jenis model tersebut menghasilkan kinerja yang berbeda-beda dalam mengidentifikasi bahan-bahan radioaktif. Model *machine learning* terbaik yang dihasilkan dari penelitian ini adalah model *Support Vector Classifier* dengan parameter kernel linier dengan hasil akurasi sebesar 100% dengan lama proses pengujian 0,244 detik. Model klasifikasi ML lain, seperti model *Decision Tree* menghasilkan akurasi sebesar 80% dengan waktu pelatihan 6,48 detik, model *Random Forest* dengan akurasi 100% dan waktu pelatihan 4,2 detik, model *AdaBoost* dengan akurasi 100% dan waktu pelatihan 22 detik, model *Naive Bayes* dengan akurasi 22% dan waktu training 0,054 detik, model *Gaussian process* dengan akurasi 100% dan waktu training 390 detik.

Penelitian yang dilakukan oleh John-Ryan Romo, Kai Tyrus Nelson, Mateusz Monterial, Karl E. Nelson, Simon E. Labov, dan Adam Hecht terpublikasi pada tahun 2021 berjudul “Classifier Comparison for Radionuclide Identification from Gamma-ray Spectra”. Penelitian ini membahas kinerja algoritma ML untuk identifikasi dan klasifikasi spektrum gamma dari berbagai radionuklida. Radionuklida yang digunakan sebanyak 33 jenis, beberapa diantaranya Cs-137, Co-60, Na-22, In-111, K-40, Am-241, Ba-133, Eu-152, Cd-109, dan lain sebagainya. Detektot untuk mengakuisisi data spektrum gamma dari radionuklida-radionuklida tersebut adalah detektor Natrium Ioda (NaI). Sampel radionuklida dicacah dengan total terdistribusi secara eksponensial diantara 1.000 hingga 100.000 cacah dengan laju cacah seragam antara 50 hingga 200 cacah per detik. Metode klasifikasi radionuklida menggunakan metode *one vs all* atau “satu lawan semua” karena menggunakan berbagai jenis radionuklida sehingga data

radionuklida-radionuklida tersebut merupakan data multi-label dan multi-kelas. Metode ini akan mencocokkan satu data per tiap kelas radionuklida dengan *library* data-data berbagai radionuklida. Algoritma program klasifikasi ML yang digunakan dan diteliti adalah *Multilayer Perceptron Artificial Neural Network* (ANN), *Extreme Gradient Boost Trees* (XGBoost), dan *Random Forest*. Hasil prediksi klasifikasi berupa kecocokan antara hasil prediksi dengan data riilnya. Kinerja tiap jenis model algoritma klasifikasi terhadap setiap jenis radionuklida diukur dengan perhitungan nilai *F1-Score* dan ketepatan hasil prediksi klasifikasi dengan data aktual ditampilkan secara visual dalam bentuk *Confusion matrix*. Diperoleh hasil klasifikasi untuk tiap radionuklida dan tiap model algoritma berbeda-beda. Hasil klasifikasi terbaik berupa klasifikasi radionuklida In-111 dengan algoritma *Multilayer Perceptron ANN* dengan nilai *F1-Score* 0,995. Hasil klasifikasi terburuk berupa klasifikasi Pu-239 dengan algoritma *Random Forest* dengan nilai *F1-Score* sebesar 0,337. Secara keseluruhan, jenis algoritma terbaik untuk klasifikasi radionuklida berdasarkan spektrum radiasi gamma adalah *XGBoost* dengan nilai rata-rata *F1-Score* sebesar 0,910. Nilai *F1-Score Multilayer Perceptron ANN* sebesar 0,888. Nilai *F1-Score Random Forest* sebesar 0.859.

Penelitian yang dilakukan oleh Chua Kiang Hong, Mohd Azlan Abu, Mohd Ibrahim Shapiai, Mohamad Fadzli Haniff, Radhir Sham Mohamad, dan Aminudin Abu berjudul “Analysis of Wind Speed Prediction using Artificial Neural Network and Multiple Linear Regression Model using Tinym1 on Esp32” terpublikasi pada tahun 2023. Publikasi ilmiah ini membahas kinerja dua algoritma ML, yaitu *Artificial Neural Network* (ANN) dan *Multiple Linier Regression* (MLR) dalam bentuk TinyML untuk memprediksi kecepatan angin. Perangkat yang digunakan adalah ESP32 sebagai MCU, sensor DHT11 untuk pengukuran suhu dan kelembapan, dan LCD *Display* untuk menampilkan hasil prediksi. Data yang dikumpulkan dari perangkat sensor adalah data pengukuran suhu dan kelembapan. Akuisisi data pengukuran tersebut dilakukan di daerah Setapak, Kuala Lumpur. Data tersebut digunakan dalam pelatihan dan pengujian kedua model TinyML. Kedua model agoritma TinyML tersebut disusun

menggunakan *library* TensorFlow Lite Micro dan Eloquent TinyML. Hasil prediksi kecepatan angin dari kedua model algoritma TinyML tersebut dibandingkan dengan nilai kecepatan angin aktual yang diukur dengan alat anemometer. Kinerja kedua model algoritma TinyML berupa perbandingan nilai hasil prediksi dengan nilai sebenarnya dihitung dengan menggunakan persamaan R^2 , *Mean Square Error* (MSE), dan *Root Mean Square Error* (RMSE). Hasil pengujian prediksi kecepatan angin dari kedua model TinyML menunjukkan nilai yang beragam. Proses perhitungan nilai R^2 , MSE, dan RMSE dilakukan sebanyak dua kali perhitungan, yaitu perhitungan menyertakan data sewaktu hujan dan perhitungan mengecualikan data sewaktu hujan. Hasil perhitungan nilai R^2 , MSE, dan RMSE yang menyertakan data kondisi hujan berupa nilai R^2 model MLR sebesar 43,048%, nilai R^2 model ANN sebesar 43,025%; nilai MSE model MLR sebesar 2,200; nilai MSE model ANN sebesar 2,200; nilai RMSE model MLR sebesar 1,483; nilai RMSE model ANN sebesar 1,484. Nilai perhitungan yang mengecualikan data sewaktu hujan berupa nilai R^2 model MLR sebesar 71,029%, nilai R^2 model ANN sebesar 75,866%; nilai MSE model MLR sebesar 1,159; nilai MSE model ANN sebesar 0,965; nilai RMSE model MLR sebesar 1,076; nilai RMSE model ANN sebesar 0,982. Perbedaan kedua hasil perhitungan tersebut dikarenakan data yang digunakan pada saat pelatihan model TinyML mengecualikan data sewaktu hujan. Kedua model TinyML tersebut menunjukkan akurasi prediksi kecepatan angin yang tinggi pada kondisi tidak hujan. Model ANN lebih unggul dalam hal akurasi prediksi kecepatan angin berdasarkan hasil perhitungan R^2 yang tinggi, MSE dan RMSE yang rendah dibandingkan dengan model MLR.

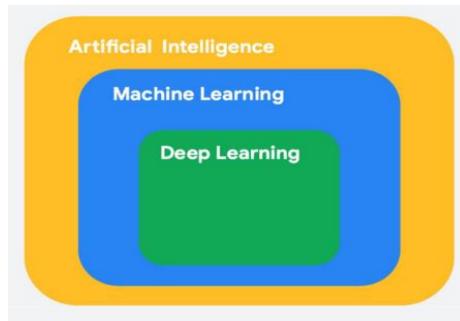
2.2 Landasan Teori

Teori dari penelitian tugas akhir ini terkait dengan Machine Learning, TinyML, spektroskopi gamma, serta perangkat yang digunakan untuk mendukung pelaksanaan penelitian Tugas Akhir ini.

2.2.1 Machine Learning

Machine Learning, selanjutnya disebut sebagai ML, diterjemahkan sebagai Pembelajaran Mesin, merupakan metode komputasi yang memberikan kemampuan

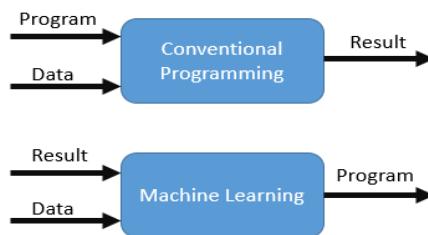
bagi perangkat untuk mempelajari suatu data hingga menemukan suatu pola pengetahuan tertentu serta dengan pola pengetahuan tersebut dapat memberikan hasil output prediksi tertentu setelah diberi input baru tanpa perlu diprogram secara eksplisit. Machine Learning merupakan bagian dari Artificial Intelligence (AI) sebagai jenis teknologi yang memberikan kemampuan kecerdasan pada perangkat untuk menyelesaikan masalah kompleks. Deep Learning (DL) sebagai bagian dari ML merupakan metode pembelajaran mesin dengan meniru cara kerja sistem syaraf otak manusia. (Sarno, R, dkk, 2023)



Gambar 2.1 Lingkup AI, ML, dan DL

Sumber gambar: <https://tinyml.seas.harvard.edu/SustainableDev-24/>

Berbeda dengan pemrograman konvensional, ML bekerja dengan membuat model matematis berdasarkan proses pelatihan menggunakan dataset pelatihan, mempelajari pola yang ada di dataset tersebut dengan metode matematis tertentu hingga menghasilkan suatu model program ML yang akan memprediksi hasil baru setelah diberi dataset terbaru berupa dataset pengujian. (Stefanus, R, 2019)



Gambar 2.2 Perbedaan pemrograman konvensional dan ML

Sumber gambar : <https://rstefanus16.medium.com/conventional-programming-vs-machine-learning-a3b7b3425531>

Proses pembelajaran ML terbagi menjadi tiga kategori, yaitu *Supervised Learning*, *Unsupervised Learning*, dan *Reinforcement Learning*. Perbedaan proses tersebut terkait dengan bagaimana mesin memperoleh pengetahuan dari dataset yang ada selama proses pelatihannya.

Supervised Learning, diterjemahkan sebagai Pembelajaran Terbimbing, merupakan jenis ML yang memberikan kemampuan pada perangkat untuk mempelajari data dengan dipandu atau dibimbing menggunakan data berlabel. Jenis algoritma ML yang termasuk jenis *Supervised Learning* diantaranya *Support Vector Machine* (SVM), *Linear Regression*, *Logistic Regression*, *Naive Bayes*, *Decision Tree*, dan *K-Nearest Neighbor*. Contoh penerapan metode *Supervised Learning* dalam pemrograman ML diantaranya untuk klasifikasi objek dan prediksi harga.

Unsupervised Learning, diterjemahkan sebagai Pembelajaran Tidak Terbimbing, merupakan jenis ML yang memberikan kemampuan pada perangkat untuk mempelajari pola data tanpa dipandu atau dibimbing. Algoritma ML yang menggunakan metode pembelajaran *Unsupervised Learning* diantaranya *K-Means*, *Fuzzy C-Means*, *Hierarchical Clustering*, dan *DBSCAN*. Contoh penerapan metode *Unsupervised Learning* dalam pemrograman ML diantaranya untuk deteksi anomali, sistem rekomendasi, dan pembagian kelompok pelanggan.

Reinforcement Learning, diterjemahkan sebagai Pembelajaran Penguatan, merupakan metode ML berupa memberikan kemampuan pada perangkat untuk belajar secara mandiri pada suatu lingkungan tertentu dengan memberikan semacam penghargaan apabila bertindak benar dan memberikan semacam hukuman apabila bertindak salah. Algoritma ML yang menggunakan metode ini diantaranya *Q-Learning*, *Deep Deterministic Policy Gradient* (DDPG), dan *Monte Carlo Tree Research* (MCTS). *Reinforcement Learning* diterapkan pada pemrograman ML di bidang robotika, game, kendali industri, dan sistem pelatihan personalisasi.

Berdasarkan pada tugas atau fungsinya, ML melakukan tiga jenis tugas, yaitu Klasifikasi, Regresi, dan Klasterisasi. Perbedaan ketiga jenis tersebut berupa perbedaan

pada proses pembelajarannya, data input yang digunakan, dan data output yang dihasilkan. Klasifikasi merupakan kemampuan mengenali suatu objek diantara objek-objek lainnya, berdasarkan pada persamaan dan perbedaan kriteria, ciri-ciri, atau deskripsi tertentu. Kemampuan melakukan klasifikasi merupakan salah satu tugas dari algoritma ML. Klasterisasi sama seperti dengan klasifikasi. Yang membedakan ialah, klasifikasi disertai dengan label nama suatu objek sebagai pembeda dengan objek lainnya, sedangkan klasterisasi tidak disertai dengan label nama pembeda tersebut.

2.2.2 Pengukuran Kinerja Machine Learning Model Klasifikasi

Untuk mengetahui kinerja suatu model ML dilakukan pengukuran atau perhitungan kinerja dengan menggunakan metrik tertentu. Metrik merupakan alat pengukuran kinerja atau kualitas suatu model ML dalam memberikan suatu hasil output tertentu berdasarkan pada input yang diberikan dan model pelatihan yang digunakan. Output yang dihasilkan diberi suatu skor dalam meninjau model terbaik yang akan digunakan. Nilai skor dari metrik yang digunakan menjadi bahan evaluasi dalam pengembangan model ML.

Metrik mengukur kemampuan model dalam mempelajari pola-pola dari data pelatihan dan menerapkan model pada data validasi dan pengujian. Metrik juga berguna dalam perbandingan kinerja beberapa jenis model ML yang digunakan. Metrik terbagi menjadi dua model yang umum digunakan dalam ML yaitu model Regresi dan model klasifikasi. Metrik kinerja yang digunakan pada model regresi berupa *Mean Absolute Error* (MAE), *Mean Squared Error* (MSE), *Root Mean Squared Error* (RMSE), dan R^2 . Metrik kinerja untuk model klasifikasi berupa *Confusion matrix*, Akurasi, Presisi, *Recall*, dan *F1-Score*. (Tharwat. A., 2021)

Terkhusus untuk metrik kinerja model ML berjenis klasifikasi, penggunaan dari metrik kinerja beserta proses perhitungannya adalah sebagai berikut.

1. *Confusion matrix*.

Tahapan pertama dalam evaluasi model klasifikasi ML adalah membuat *Confusion matrix*, dengan membandingkan kelas hasil prediksi dengan kelas

yang sebenarnya. Dari perbandingan tersebut dihasilkan empat kondisi ketepatan, yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) dan *False Negative* (FN). *True* atau *False* merupakan ketepatan hasil prediksi dengan nilai sebenarnya. *True* berarti hasil prediksi tepat atau sesuai dengan kondisi aktual. *False* berarti hasil prediksi tidak tepat dengan kondisi aktual. *Positive* atau *Negative* merupakan nilai kondisi atau kelas klasifikasi. *Positive* berarti jenis kelas klasifikasi yang diharapkan, dan *Negative* berarti jenis kelas klasifikasi yang tidak diharapkan. Secara umum, bentuk dari *Confusion matrix* terdapat pada Tabel 2.1

Tabel 2.1 Contoh Tabel *Confusion matrix*

		Aktual	Positif	Negatif
Prediksi	Aktual			
	Positif	TP	FP	
	Negatif	FN	TN	

Confusion matrix merupakan gambaran hasil prediksi dengan kondisi aktualnya. Berdasarkan tabel tersebut, deskripsi dari keempat kondisi ketepatan *Confusion matrix* adalah sebagai berikut.

- 1) *True Positive* (TP) berarti hasil prediksinya kelas positif dan kondisi aktualnya kelas positif.
- 2) *True Negative* (TN) berarti hasil prediksinya kelas negatif dan kondisi aktualnya kelas negatif.
- 3) *False Positive* (FP) berarti hasil prediksinya kelas positif dan kondisi aktualnya negatif.
- 4) *False Negative* (FN) berarti hasil prediksinya kelas negatif dan kondisi aktualnya positif.

Berdasarkan keempat nilai kondisi tersebut, terdapat dua jenis eror, yaitu FP dan FN seperti diilustrasikan pada Gambar 2.4.



Gambar 2.3 Ilustrasi jenis eror pada model klasifikasi ML

Sumber gambar: <https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/>

Berdasarkan pada Tabel *Confusion matrix* tersebut, metrik kinerja program ML dapat dihitung untuk mengetahui kualitas kinerja model ML yang disusun. Metrik kinerja tersebut terdiri dari akurasi, presisi, *recall*, dan *F1-Score*.

2. Akurasi.

Metrik ini membandingkan ketepatan model dalam mengklasifikasikan data dengan benar dibandingkan dengan total prediksi yang dilakukan. Metrik akurasi diukur atau dihitung dengan menggunakan persamaan (2.1).

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \quad 2.1$$

Rentang nilai akurasi antara 0 dan 1. Pada kondisi riilnya, secara teknis nilai akurasi berkisar antara 0,5 dan 1. Nilai akurasi terbaik sebesar 1 atau mendekati 1 karena model ML memiliki nilai dengan kondisi *True*, yaitu *True Positive* dan *True Negative* yang lebih banyak dan mendominasi dibandingkan dengan nilai berkondisi *False* (*False Positive* dan *False Negative*). Metrik akurasi dapat digunakan sebagai penilaian kinerja model jika hasil *Confusion matrix* pada nilai FN dan FP mendekati atau simetri.

3. Presisi.

Metrik ini mengukur jumlah hasil prediksi positif yang benar (TP) dibandingkan dengan keseluruhan nilai yang terprediksi positif, baik benar ataupun salah (TP dan FP). Metrik presisi dihitung dengan menggunakan persamaan (2.2).

$$Presisi = \frac{TP}{TP + FP} \quad 2.2$$

Metrik presisi bernilai diantara 0 dan 1. Nilai presisi terbaik yaitu mendekati angka 1 karena pada kondisi ini, nilai TP lebih banyak dibandingkan nilai FP.

4. *Recall*.

Recall, atau disebut juga sebagai Sensitivitas atau *True Positive Rate* (TPR). Metrik ini mengukur ketepatan model dalam mengidentifikasi suatu kelas dengan tepat dibandingkan dengan jumlah seluruh kelas tersebut, baik teridentifikasi dengan tepat ataupun tidak tepat. *Recall* sebagai gambaran kemampuan model dalam mendeteksi data positif secara keseluruhan. Metrik ini dihitung dengan menggunakan persamaan (2.3)

$$Recall = \frac{TP}{TP + FN} \quad 2.3$$

Metrik ini bernilai diantara 0 dan 1. Nilai *recall* terbaik yaitu mendekati 1 karena pada kondisi ini, mayoritas kelas positif teridentifikasi dengan tepat. Nilai TP lebih besar dibandingkan dengan nilai FN.

5. *F1-Score*.

Nilai *F1-Score* merupakan nilai rata-rata harmonis untuk nilai presisi dan *recall* sebagai satu nilai yang seimbang pada aspek kedua nilai tersebut. Metrik ini merupakan penggabungan ukuran dari presisi dan *recall* yang menunjukkan keseimbangan kedua metrik tersebut. Metrik ini dihitung dengan menggunakan persamaan (2.4)

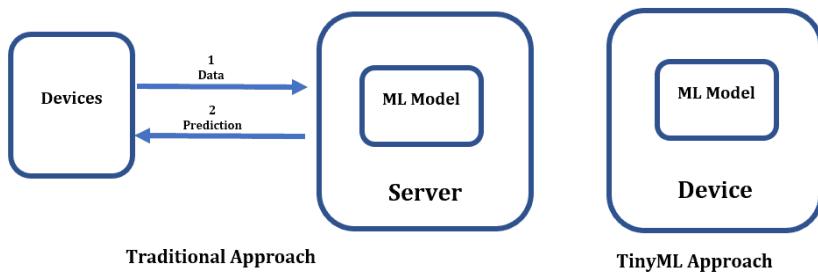
$$F1 Score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 * Precision * Recall}{Precision + Recall} \quad 2.4$$

Nilai *F1-Score* antara 0 dan 1. Semakin mendekati nilai 1 maka model semakin baik dalam performanya, yang menandakan keseimbangan yang bagus antara *precision* dan *recall*. Semakin rendah nilainya (mendekati 0), maka semakin buruk performa model tersebut dalam mengklasifikasikan data secara akurat.

Suatu model ML dikatakan baik apabila jumlah eror FP dan FN lebih sedikit dibandingkan dengan TP dan TN, jumlah error FP dan FN yang sama atau hampir sama dan seimbang, serta jumlah TP dan TN yang sama atau hampir sama dan seimbang.

2.2.3 *Tiny Machine Learning*

Tiny Machine Learning, selanjutnya akan disebut sebagai TinyML, merupakan pendekatan dalam pengaplikasian ML pada perangkat dengan sumber daya komputasi terbatas, baik itu kapasitas prosesor, kapasitas memori, dan juga sedikit energi listrik. Perangkat komputasi yang dapat diprogram dengan TinyML diantaranya *smartphone*, *embedded system*, dan juga mikrokontroler. TinyML memberikan kemampuan baru pada perangkat - perangkat tersebut dalam menggunakan algoritma ML. Sebelumnya, perangkat dengan kemampuan komputasi terbatas tersebut hanya digunakan untuk mengakuisisi data dan mengirimnya ke perangkat lain, seperti server yang dapat menjalankan program ML untuk menganalisis data dan memberikan hasil prediksi. Dengan TinyML, perangkat tidak lagi perlu mengirim data ke perangkat lain karena sudah dapat menjalankan program TinyML untuk menganalisis data dan memberikan hasil prediksi secara langsung sesudah menerima input data. Perbedaan penggunaan perangkat mikrokontroler dengan ML dan TinyML dapat diamati pada ilustrasi Gambar 2.5.



Gambar 2.4 Ilustrasi perbedaan ML konvensional dan TinML pada perangkat komputasi terbatas

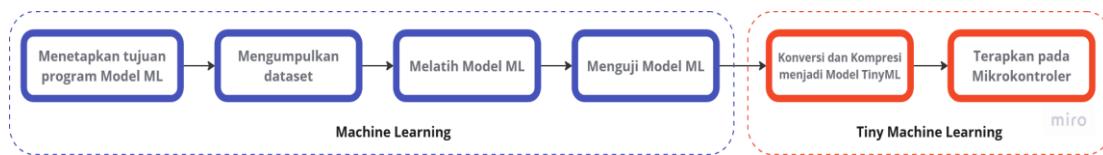
Sumber gambar: <https://www.baeldung.com/cs/tinyml>

TinyML merupakan kombinasi dari perangkat keras dan perangkat lunak tertentu untuk dapat melakukan program ML seringan mungkin. Diperlukan perangkat keras yang memadai dan teknik pemrograman tertentu untuk menyusun dan menjalankan program TinyML. Salah satu jenis perangkat keras yang dapat menjalankan program TinyML adalah mikrokontroler. Untuk dapat diprogram dan menjalankan TinyML, mikrokontroler yang dapat digunakan memiliki prosesor yang andal, jumlah memori yang cukup, dan frekuensi *clock* yang memadai, serta menggunakan sedikit energi listrik. Diperlukan mikrokontroler dengan komponen CPU (*Central Processing Unit*) dengan laju *clock* sekitar 16 MHz hingga 64 MHz. Kapasitas memori untuk menyimpan data dan program sekitar 64 hingga 256 KB untuk komponen memori RAM dan sekitar 2 MB untuk komponen memori *Flash*. Jumlah energi listrik yang diperlukan berkisar beberapa miliwatt hingga mikrowatt sehingga mikrokontroler dengan program TinyML dapat dioperasikan dengan sumber energi dari baterai. Contoh mikrokontroller yang dapat menggunakan TinyML adalah Arduino Nano 33, STM32F, ESP32, Raspberry Pi Pico, Arduino Portenta H7, Nordic Semi nRF52840 DK, dan ATSAMD51G19A. (Abadade, Y. dkk. 2023)

Penyusunan program model TinyML dibantu dengan berbagai *library* atau *platform* pemrograman tertentu yang mampu menyusun program ML dan mengonversinya menjadi model TinyML hingga dapat diprogramkan ke perangkat mikrokontroler. Berbagai *library* atau *platform* yang dapat digunakan dalam

penyusunan program model TinyML diantaranya TensorFlow Lite, Edge Impulse, Neutron, MicroML, Cube.AI, TinyNN dan TinyMLPerf. Program model TinyML yang dihasilkan berupa model ML yang hanya menggunakan sedikit memori, sekitar beberapa KB, dan dapat berjalan tanpa memerlukan Sistem Operasi.

Proses penyusunan program TinyML hampir sama seperti ML konvensional, terdiri dari menentukan tujuan penyusunan program TinyML, mengumpulkan dataset, mendesain arsitektur model ML, melatih model ML, menguji model ML, hingga menerapkan model ML pada perangkat keras. Pada TinyML, sebelum model ML diterapkan pada perangkat keras, seperti mikrokontroler, model ML yang telah dilatih dan diuji harus dikonversi dan dikompresi ukuran model ML konvensional menjadi model TinyML sehingga dapat diterapkan pada perangkat mikrokontroler. Alur penyusunan ML hingga menjadi TinyML dapat diamati pada Gambar 2.6.



Gambar 2.5 Alur Penyusunan program ML dan TinyML

Kinerja TinyML diukur dengan empat metrik, yaitu akurasi, konsumsi energi, latensi, dan memori. Akurasi menjadi salah satu metrik utama untuk pengukuran kinerja model ML, termasuk juga pada TinyML. Konsumsi energi dari model TinyML yang baik berupa dapat beroperasi dengan sumber energi baterai dalam jangka waktu yang lama dengan hanya menggunakan beberapa milliwatts. Konsumsi energi ini dipengaruhi oleh perangkat keras berupa spesifikasi dari mikrokontroler dan prosesor serta model TinyML yang digunakan. Dikarenakan pengoperasian model TinyML tidak membutuhkan koneksi internet atau terhubung dengan jaringan komunikasi untuk proses analisis datanya sehingga kecepatan proses atau latensinya lebih cepat. Latensi tergantung pada perangkat lunak dan perangkat keras yang digunakan. Memori juga merupakan salah satu aspek terpenting dalam kinerja TinyML karena model ML yang disusun akan dikompresi sehingga model program TinyML dapat disimpan dalam memori perangkat. Pengompresian memori dari model ML menjadi salah satu

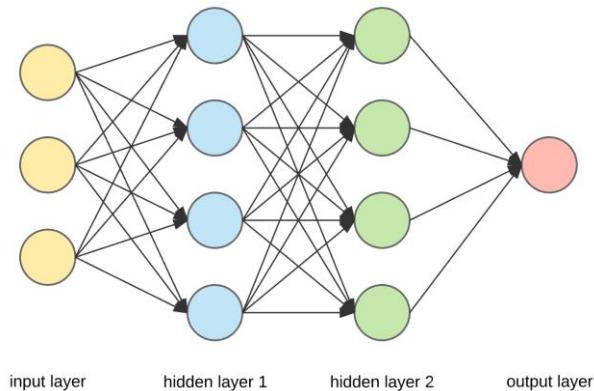
tantangan dalam pengembangan TinyML. Teknik yang dapat digunakan dalam kompresi memori model ML menjadi model TinyML yaitu teknik *pruning* dan kuantisasi. Penyusunan model hingga pengompresiannya tergantung pada susunan program perangkat lunaknya.

Teknologi program model TinyML telah diteliti dan diterapkan untuk berbagai keperluan, diantaranya untuk prediksi kerusakan dan pemeliharaan peralatan mesin industri, deteksi dan klasifikasi objek, prediksi kondisi lingkungan, memantau kondisi kesehatan tumbuhan pertanian, memantau kondisi kesehatan dan aktivitas manusia, dan juga pada perangkat teraktivasi suara.

2.2.4 *Artificial Neural Network*

Artificial Neural Network (ANN), atau diterjemahkan sebagai Jaringan Syaraf Tiruan, merupakan jenis algoritma termasuk dalam bagian dari ML yang meniru cara kerja saraf otak manusia dalam piranti komputer untuk melakukan suatu tugas dan pemecahan solusi dengan tingkat kecerdasan tertentu. Sesuai dengan definisi tersebut, ANN tersusun dari berbagai *nodes* atau titik syaraf yang terkumpul dalam satu kelompok lapisan tertentu.

Pada ANN, terdapat tiga jenis lapisan *nodes*, yaitu lapisan input (*input layer*), lapisan tersembunyi (*hidden layer*), dan lapisan output (*output layer*). Lapisan input sebagai lapisan pertama dalam program ANN yang menerima nilai-nilai input dari suatu dataset. Lapisan tersembunyi merupakan lapisan yang akan melakukan komputasi dan transformasi data dari input data. Lapisan tersembunyi dapat berupa satu lapisan (*single hidden layer*) ataupun banyak lapisan (*multi hidden layer*). Setiap *nodes* dan setiap lapisan pada jenis ini memiliki konfigurasi program atau fungsi aktivasi tertentu untuk mengolah data input. Lapisan output merupakan lapisan terakhir yang menghasilkan suatu nilai output tertentu setelah melalui pengolahan data di lapisan-lapisan sebelumnya. Setiap *nodes* dalam setiap *layers* saling terhubung satu sama lain.



Gambar 2.6 Ilustrasi susunan lapisan *nodes* dalam model ANN

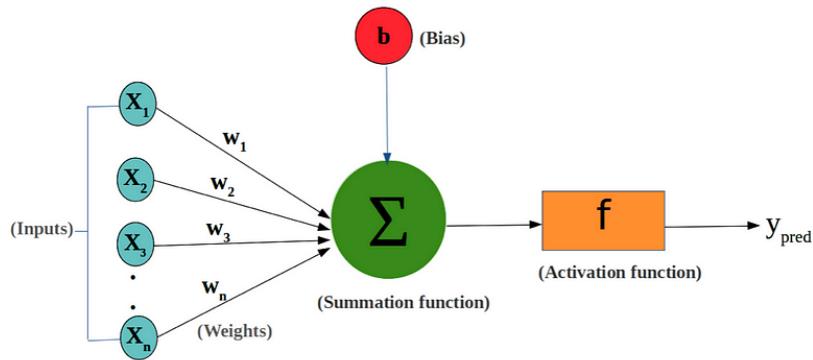
Sumber gambar: <https://brilliant.org/wiki/artificial-neural-network/>

Dalam setiap *nodes*, terdapat tiga parameter yang mempengaruhi cara kerja dan kualitas dari model ANN, yaitu *weights*, *bias*, dan fungsi aktivasi. Ketiga parameter tersebut mempengaruhi kemampuan ANN dalam proses pengolahan dan perhitungan data pada tiap *nodesnya* untuk mempelajari pola data dan memprediksi nilai output dengan akurat.

Weights merupakan parameter numerik yang mempengaruhi kekuatan koneksi antar neuron. Nilai *weights* mempengaruhi nilai input dari setiap *nodes*. Selama proses pelatihan model ANN, nilai *weights* sebagai koefisien akan berubah, bertambah ataupun berkurang, menyesuaikan secara berulang untuk meminimalkan perbedaan nilai antara nilai output model dengan nilai output riil. Nilai *weights* mempengaruhi cara ANN mempelajari data dengan mencari tahu hubungan antara nilai input dengan nilai output yang ditargetkan sehingga ANN dapat menggeneralisasi dan menentukan nilai output prediksi nilai data input yang baru.

Bias merupakan parameter krusial dalam model ANN yang berfungsi untuk memberikan fleksibilitas. Secara teknis, bias adalah nilai numerik yang ditambahkan pada hasil penjumlahan input yang telah dikalikan dengan *weights* (*weighted sum*), sebelum nilai tersebut dimasukkan ke dalam fungsi aktivasi.

Fungsi aktivasi (*activation function*) adalah fungsi matematika yang diterapkan pada setiap *nodes* neuron dalam setiap *layer* ANN. Fungsi ini akan mempelajari pola non-linear yang kompleks dari suatu data. Pola non-linear berarti hubungan antara data input dan data output tidak berupa garis lurus atau linear. Tanpa fungsi ini, suatu program dan model ANN hanya dapat mempelajari pola linear sehingga akan sama seperti model ML Regresi Linear. Fungsi aktivasi akan menentukan kondisi aktivasi suatu *nodes* neuron dengan menghitung jumlah *weight* dari input-input serta menambah nilai bias.



Gambar 2.7 Ilustrasi model matematis dalam setiap *nodes* ANN

Sumber gambar : <https://medium.itsayush.com/breaking-down-the-concept-of-activation-functions-in-deep-learning-9b22381186a>

Jenis - jenis fungsi aktivasi dalam ANN diantaranya Fungsi Sigmoid, Fungsi Tanh, Fungsi ReLU (Rectified Linear Unit), Fungsi Softmax, Fungsi SoftPlus. Setiap fungsi aktivasi memiliki persamaan matematis tersendiri yang menentukan cara model ANN mempelajari dan menginterpretasi data.

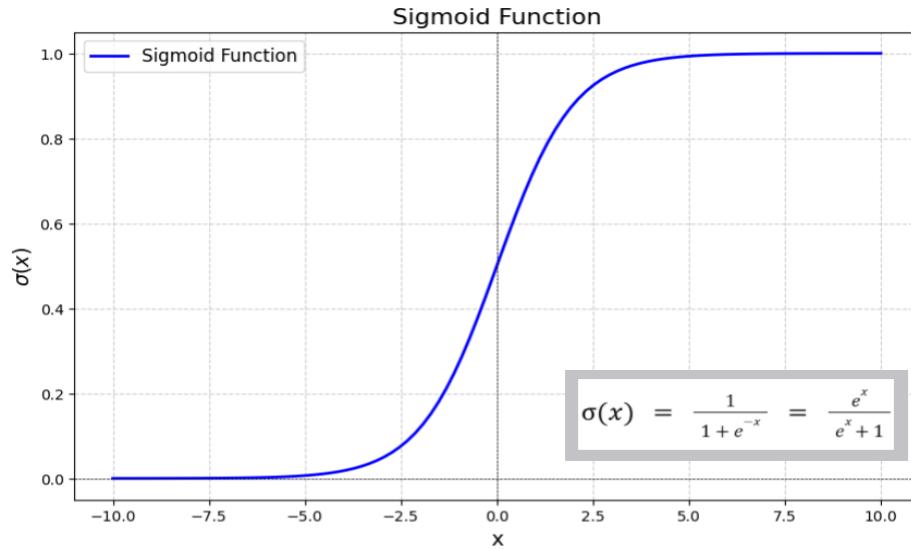
1. Fungsi sigmoid.

Fungsi sigmoid merupakan fungsi matematika yang mengelompokkan bilangan riil ke dalam nilai biner 0 ataupun 1. Karakteristik dari fungsi ini adalah menghasilkan kurva berbentuk huruf “S”

Fungsi Sigmoid memiliki persamaan matematis berikut

$$\sigma = \frac{1}{1 + e^{-x}} \quad 2.5$$

Bentuk grafik dari fungsi sigmoid adalah sebagai berikut



Gambar 2.8 Grafik dari fungsi sigmoid

Sumber gambar : <https://www.geeksforgeeks.org/derivative-of-the-sigmoid-function/>

Berdasarkan pada grafik tersebut, nilai pada sumbu x sebagai nilai input pada rentang antara $-\infty$ hingga $+\infty$. Nilai sumbu-y merupakan nilai output sebagai nilai probabilitas yang menghasilkan antara 0 dan 1.

Fungsi sigmoid sebagai fungsi non-linier mampu melakukan analisis data non-linier yang kompleks.

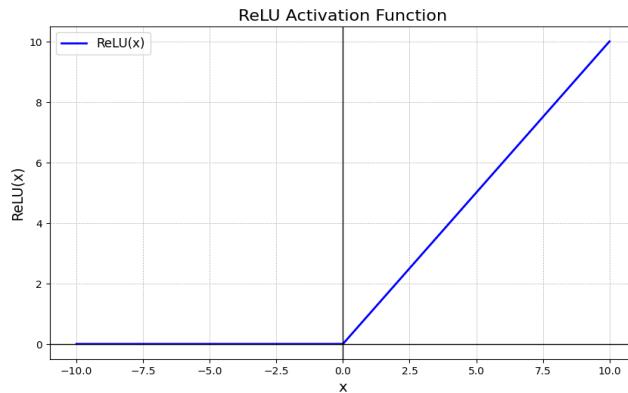
2. Fungsi ReLU

Fungsi ReLU (*Rectified Linear Unit*) memiliki persamaan matematis berikut.

$$f(x) = \max(0, x) \quad 2.6$$

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \quad 2.7$$

Fungsi ini akan menghasilkan nilai output x apabila nilai input x lebih besar dari nol. Apabila nilai input x lebih kecil atau sama dengan 0, maka fungsi ini akan menghasilkan nilai output 0. Fungsi ini akan mengizinkan nilai input positif untuk diproses tanpa mengubah nilainya sementara nilai input negatif akan dinolkan. Apabila digambar dalam grafik sebagai berikut.



Gambar 2.9 Grafik fungsi ReLU

Sumber gambar :

Fungsi ini cukup sederhana sehingga proses komputasi menjadi lebih efisien. Kesederhanaan ini membuat pelatihan model ANN menjadi lebih efektif dengan mempertahankan ketidaklinearan tanpa mengubah kerumitan.

Jenis algoritma ini dapat diterapkan sebagai metode rekomendasi dalam media sosial, pengenalan wajah, pengenalan pola perilaku pelanggan, pengenalan pola penyakit dan kesehatan manusia, dan lain sebagainya.

2.2.5 TensorFlow

TensorFlow merupakan *library* pemrograman *open-source* yang digunakan sebagai *framework* untuk pemrograman ML dan *Deep Learning*. TensorFlow dikembangkan pertama kali oleh Google dan dirilis ke publik pada tahun 2015. Library TensorFlow dapat digunakan dalam Sistem Operasi Linux, Windows, dan macOS baik sebagai komputer dekstop ataupun server. TensorFlow memiliki arsitektur yang fleksibel sehingga dapat digunakan pada perangkat dengan GPU ataupun CPU

sehingga dapat digunakan pada berbagai jenis perangkat, dari perangkat *mobile* hingga *data center* berskala besar. Dengan berbagai *tools* dan contoh yang telah banyak berkembang menjadikan pemrograman ini cukup populer digunakan.

TensorFlow beroperasi sebagai program komputasi grafik yang tersusun dari *nodes* dan *edges*. Pada *nodes* terdapat operasi matematika dan pada *edges* terdapat tensor. Tensor merupakan data dalam bentuk *array* multi-dimensi. Edges menghubungkan berbagai *nodes* sehingga data-data mengalir sepanjang edges diantara berbagai operasi-operasi matematis pada *nodes*. Dengan pola seperti itu, proses komputasi berlangsung lebih efisien dan aliran data dapat divisualisasikan.

TensorFlow mampu menyusun model yang kompleks sehingga dapat melakukan pengolahan data dalam jumlah yang masif dan melakukan perhitungan matematis yang kompleks sehingga cocok digunakan dalam aplikasi riset dan industri.

2.2.6 Keras

Keras adalah *framework* API ML yang digunakan dalam pembuatan model *Deep Learning*. Keras bersifat *open-source*, *high-level*, dan mudah digunakan dalam pengembangan model *Deep Learning* sehingga ML *developer* dapat menyusun, melatih, dan menggunakan model deep learning tanpa baris sintaks yang banyak.

Dalam penggunaannya, Keras bersifat fleksibel. Keras dapat digunakan dalam kombinasi dengan *framework* atau *backend engine* yang lain seperti TensorFlow, Theano, dan MXNet, serta dapat digunakan di berbagai platform yang beroperasi dengan GPU ataupun CPU. Fleksibilitas ini menjadikan framework Keras cukup populer digunakan dalam proyek-proyek ML dan DL.

Arsitektur Keras bersifat modular yang tersusun dari *frontend*, *backend*, dan *engine*. Komponen *frontend* menghadirkan API yang intuitif untuk pembangunan model ML dan DL. Komponen backend berinteraksi dengan mesin DL pada programnya. Komponen engine akan mengeksekusi grafik komputasinya. Komponen-komponen tersebut dihadirkan dengan fitur-fitur utama seperti API yang sederhana dan

intuitif, desain yang modular, pre-built *layers*, model terserial, dan dapat digunakan pada berbagai GPU atau komputasi terdistribusi.

Pemrograman penyusunan model Deep Learning dengan Keras dapat dilakukan dengan dua metode, yaitu *Sequential API* dan *Functional API*. *Sequential API* digunakan dalam penyusunan model linier *deep learning* yang sederhana. *Functional API* digunakan dalam penyusunan model *deep learning* yang lebih kompleks.

Keras dapat menyusun berbagai jenis algoritma *neural network*, seperti *convolutional neural network* dan *recurrent neural network*. *Framework* Keras telah digunakan untuk berbagai keperluan dan di berbagai bidang, seperti dalam pemrosesan gambar dan video, *Natural Language Processing* (NLP), prediksi berbasis waktu, sistem otomatis pada robot dan kendaraan, pengembangan gim, deteksi penyakit dan kesehatan, dan lain sebagainya.

2.2.7 TensorFlow Lite

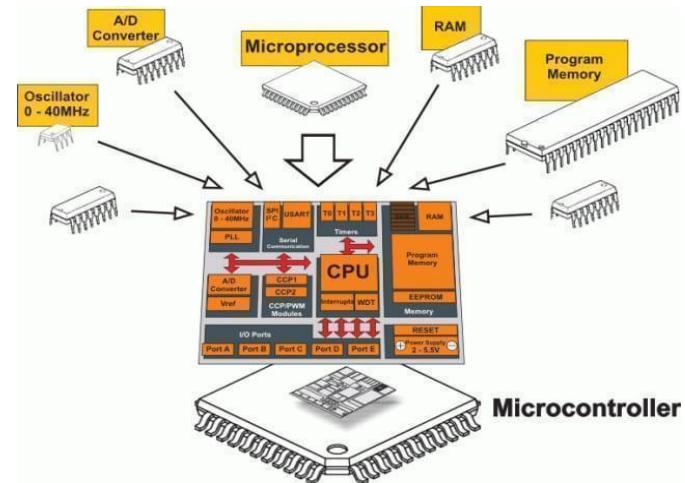
TensorFlow Lite merupakan *framework* bagian dari library TensorFlow yang digunakan dalam penyusunan model TinyML untuk perangkat komputasi terbatas seperti *smartphones* dan mikrokontroler. Diluncurkan oleh Google pada tahun 2017. Framework ini cukup populer digunakan dalam pengembangan ML, DL, hingga AI menjadi TinyML hingga EdgeAI yang dapat beroperasi pada perangkat bersumber daya komputasi terbatas.

Fitur utama dalam *framework* ini ialah mampu melakukan inferensi secara langsung di perangkat lokal, dapat digunakan di berbagai jenis dan tipe perangkat mobile, dan memiliki berbagai teknik untuk optimisasi model. Kemampuan inferensi secara langsung berarti menggunakan model TinyML secara langsung pada perangkat untuk menghasilkan suatu nilai output setelah menerima nilai input tanpa memerlukan jaringan atau koneksi dengan perangkat lain. Kemampuan ini dapat mengurangi latensi dan meningkatkan aspek keamanan sistem. Model TinyML dari TensorFlow Lite dapat digunakan pada berbagai jenis dan tipe perangkat, seperti perangkat mobile smartphone Android dan iOS, embedded Linux system, hingga mikrokontroler. Teknik optimisasi

model untuk menyimpan program dalam perangkat mobile, seperti kuantisasi untuk mengurangi nilai presisi angka yang digunakan dalam proses komputasi, pruning untuk menghapus bagian - bagian tertentu dalam model ML, dan format Flatbuffer untuk efisiensi penyimpanan dan eksekusi program. Teknik-teknik tersebut berguna dalam mengurangi ukuran model dan meningkatkan kecepatan inferensi tanpa mengurangi akurasi.

2.2.8 Mikrokontroler

Microcontroller unit (selanjutnya disebut sebagai MCU) dapat diartikan sebagai suatu sistem komputer kecil dalam bentuk satu papan sirkuit terintegrasi yang tersusun dari perangkat prosesor CPU (*Central Processing Unit*), memori, perangkat antarmuka input-output, dan perangkat elektronik komputasi pendukung lainnya. MCU umumnya digunakan hanya untuk satu tugas khusus tertentu. Perangkat akan memperoleh input data, memproses data tersebut, dan menghasilkan suatu nilai output tertentu atau mengirim hasil proses data tersebut ke sistem perangkat lainnya. MCU cenderung beroperasi pada kecepatan yang lebih rendah, sekitar 1 MHz hingga 200 MHz, serta beroperasi menggunakan daya energi yang rendah dibandingkan perangkat komputer umumnya.



Gambar 2.10 Ilustrasi susunan komponen MCU

Sumber gambar : <https://labprojectsbd.com/2023/04/25/learning-pic-microcontrollers-programming-in-c-chapter-1/>

Susunan komponen perangkat MCU diilustrasikan pada Gambar 2.7. MCU tersusun dari berbagai komponen penting, diantaranya *Central Processing Unit* (CPU), *Random Access Memory* (RAM), *Flash Memory*, *Serial Bus Interface*, *Input-Output port*, dan komponen pendukung lainnya.

CPU, disebut juga sebagai Prosesor, merupakan komponen yang mengendalikan semua instruksi program dan aliran data. Tugasnya berupa mengolah data input yang diterima, melakukan instruksi yang diprogramkan, dan mengirim data hasil pemrosesan ke komponen lain.

RAM merupakan komponen memori untuk menyimpan sementara data-data untuk dapat digunakan pada proses selanjutnya. Data akan disimpan hanya dalam waktu yang singkat. Data pada memori ini akan terhapus apabila perangkat MCU dimatikan.

Flash memory merupakan komponen memori yang dapat menyimpan data dalam jangka waktu yang lama, bahkan saat perangkat dimatikan. Komponen memori ini untuk menyimpan program yang diunggah ke MCU.

Serial Bus Interface merupakan komponen yang menjalankan protokol komunikasi serial antara MCU dengan perangkat lain. Contohnya untuk protokol komunikasi SPI dan I2C.

Input-Output Ports merupakan pin penghubung antara MCU dengan perangkat lain seperti sensor dan aktuator sebagai semacam jalur aliran data yang masuk dan data yang keluar. Dengan komponen ini, MCU dapat digunakan pada aplikasi di dunia nyata, seperti memperoleh data suhu dari sensor suhu sebagai input data, diproses di komponen CPU sesuai yang diprogramkan dan dapat memberikan respon atau output data berupa lampu indikator atau menjalankan motor.

Untuk menggunakan MCU, diperlukan suatu teknik pemrograman tertentu. Penggunaan dan cara pemrogramannya berbeda-beda pada setiap produk MCU. Umumnya, MCU diprogram dengan menggunakan bahasa pemrograman C, C++ dan

Assembly Language. Program tersebut disusun dalam suatu *code editor* tertentu dan diunggah ke perangkat MCU.

MCU dapat digunakan untuk melakukan berbagai tugas dan fungsi, seperti untuk sistem pemantauan, sistem kendali, robotik, dan lain sebagainya. Contoh penerapannya seperti pada pemantauan kondisi lingkungan air, udara, dan tanah, penyiraman lahan pertanian otomatis, pemantauan kondisi kesehatan pasien, sistem *Smart Home* otomatis, dan lain sebagainya. Beberapa contoh produk dari MCU diantaranya, Arduino Nano, Espressif ESP32, Raspberry Pi Pico, dan lain sebagainya.



Gambar 2.11 Contoh produk MCU

Sumber gambar : Arduino, Espressif, Raspberry Pi

Setiap produk dan variasi dari MCU memiliki spesifikasi tertentu. Perbandingan spesifikasi setiap produk MCU terdapat pada tabel 2.2

Tabel 2.2 Perbandingan spesifikasi produk-poduk MCU

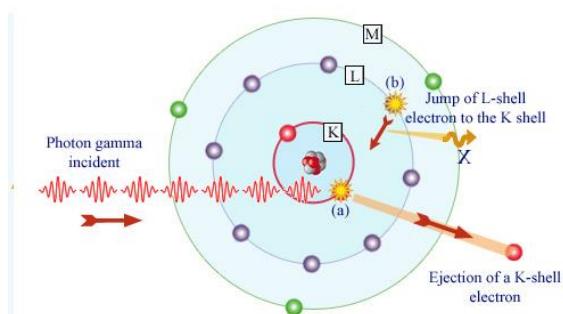
Spesifikasi	Arduino Uno R3	ESP32-WROOM-32	Raspberry Pi Pico
Jenis CPU	8-bit single-core ATmega328P RISC	Dual-core 32-bit Tensilica LX6	RP2040 (Dual-core ARM Cortex-M0+)
Clock Speed	Up to 16 MHz	Up to 240 MHz	Up to 133 MHz
Memory	2 KB RAM, 32 KB Flash, 1 KB EEPROM	520 KB SRAM, 4 MB Flash	264 KB SRAM, 2 MB Flash

2.2.9 Radiasi Gamma

Radiasi gamma merupakan jenis radiasi berupa gelombang elektromagnetik yang dipancarkan dari peluruhan atom-atom tertentu yang tidak menyebabkan perubahan nomor atom dan nomor massa pada atom tersebut. Sifat dari radiasi gamma yaitu daya ionisasi terhadap suatu medium relatif kecil sehingga daya tembusnya sangat besar dibandingkan radiasi alpha dan beta, tidak memiliki muatan sehingga tidak dapat dibelokkan oleh medan listrik ataupun magnet, serta panjang gelombang yang terpendek dalam spektrum gelombang elektromagnetik.

Radiasi gamma berasal dari peluruhan suatu atom dari kondisi tidak stabil atau tereksitas (*excited state*) menuju kondisi stabil (*ground state*). Peluruhan untuk menghasilkan radiasi gamma biasanya diikuti dengan menghasilkan radiasi alpha ataupun beta. Pada proses peluruhan tersebut, atom akan melepaskan sejumlah energi.

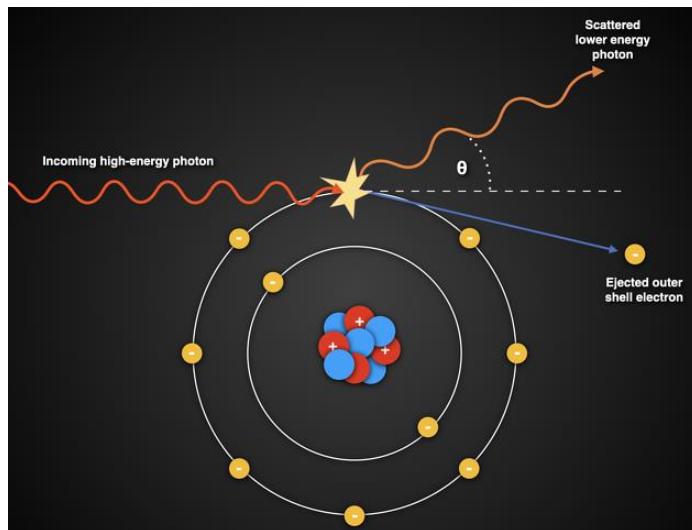
Apabila radiasi gamma berinteraksi dengan suatu materi, maka akan menimbulkan reaksi, yaitu efek fotolistrik, hamburan Compton, dan produksi pasangan. Efek fotolistrik merupakan efek pelepasan elektron dari atom setelah berinteraksi dengan radiasi gamma. Efek ini terjadi pada material bermomor atom tinggi, seperti timbal dan tungsten, pada elektron di kulit atom K, dan pada radiasi gamma berenergi rendah, sekitar 50 – 500 keV. Efek fotolistrik terjadi ketika foton radiasi gamma, yang memiliki energi yang lebih besar daripada energi ikat elektron, memberikan seluruh energinya ke elektron suatu atom pada materi yang mengenainya sehingga elektron tersebut keluar dari materi sebagai fotoelektron.



Gambar 2.12 Ilustrasi efek fotolistrik

Sumber gambar : https://radioactivity.eu.com/articles/phenomenon/photoelectric_effect

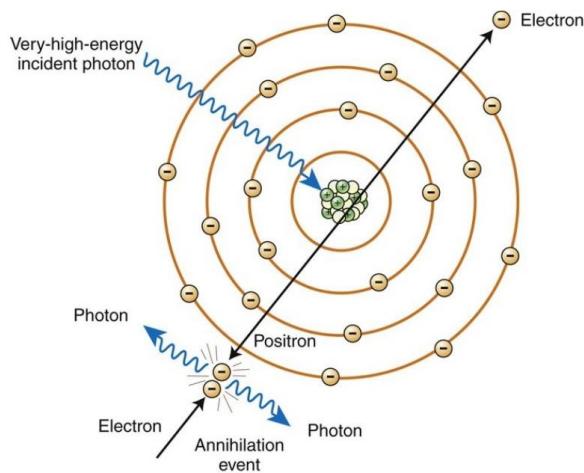
Hamburan Compton terjadi ketika elektron yang bebas atau hampir terlepas dari kulit terluar suatu atom materi terkena foton radiasi gamma berenergi sekitar 0,5 – 5 MeV. Sebagian energi dan momentum dari foton gamma diserap oleh elektron sehingga elektron akan memantul dan foton gamma tersebut masih memiliki sisa energi akan berbelok menyebar dari jalur awalnya dengan sudut tertentu. Sebelum dan sesudah interaksi tersebut, total energi dan momentum tetap sama sesuai hukum konservasi energi-momentum. Sudut belok dari foton tersebut berbanding terbalik dengan sisa energi yang dimiliki foton. Sedikit sisa energi berarti sudut belok foton yang besar. (Qiao, C.-K., dkk, 2021)



Gambar 2.13 Ilustrasi efek hamburan Compton

Sumber gambar: <https://radiopaedia.org/articles/compton-effect>

Produksi pasangan merupakan reaksi yang terjadi dari interaksi radiasi gamma berenergi tinggi mengenai inti atom sehingga menghasilkan pasangan materi dan antimateri, umumnya merupakan elektron-positron. Untuk menghasilkan reaksi ini, energi gamma yang diperlukan sebesar dua kali atau lebih dari energi elektron yaitu dua kali 0,511 MeV sama dengan 1,02 MeV atau lebih. Pasangan elektron-positron tersebut akan berinteraksi sehingga saling menghilangkan satu sama lain (annihilasi) (Buchtela, K. 2014)



Gambar 2.14 Ilustrasi efek produksi pasangan

Sumber gambar: Ismail, 2017

Dari pengukuran radiasi gamma suatu radionuklida, dapat diketahui berbagai hal mengenai radionuklida, seperti aktivitas radiasi, identifikasi radionuklida, dan pemetaan radiasi lingkungan. Terkhusus untuk identifikasi radionuklida, spektroskopi gamma akan menampilkan bentuk

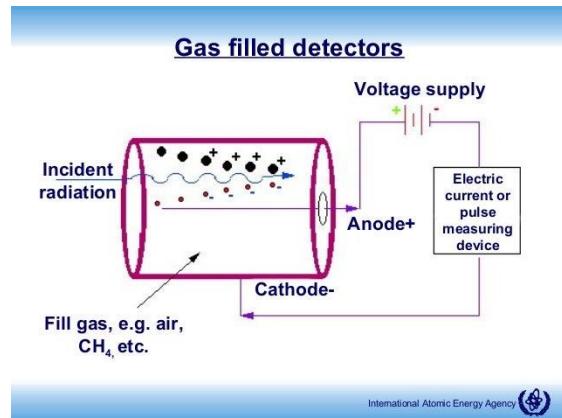
2.2.10 Detektor Radiasi

Detektor radiasi merupakan peralatan yang memiliki permukaan dari bahan yang peka terhadap penceran radiasi dan menghasilkan respon berupa energi listrik ataupun percikan cahaya. Prinsip kerja detektor radiasi secara umum adalah permukaan detektor yang menghadap ke arah sumber radiasi merupakan medium dari jenis bahan yang sensitif terhadap perubahan energi radiasi. Efek yang terjadi saat radiasi berinteraksi dengan materi permukaan tersebut akan diukur dan dikonversi menjadi bentuk energi lain melalui berbagai komponen - komponen detektornya. Hasil pengukuran tersebut berupa sinyal listrik yang selanjutnya akan diproses lagi melalui modul-modul pendukung lainnya.

Ada tiga jenis detektor radiasi berdasarkan susunan komponennya, yaitu detektor isian gas, detektor sintilasi, dan detektor semikonduktor. Ketiga jenis detektor tersebut memiliki susunan komponen yang beragam dengan kelebihan dan kekurangan masing-

masing. Setiap jenis detektor juga memiliki kemampuan pengukuran radiasi pada jenis dan rentang pengukuran yang berbeda-beda. Penting untuk memahami spesifikasi dan kemampuan tiap detektor sebelum menggunakan untuk pengukuran dan spektroskopi radiasi.

Detektor isian gas (*gas-filled detector*) merupakan detektor yang tersusun dari tabung dengan elektroda positif dan elektroda negatif serta berisi gas tertentu di antara kedua elektroda tersebut. Jenis gas yang digunakan yaitu Argon, Helium, Kryton, dan Xenon. Kedua elektroda dihubungkan dengan sumber daya listrik. Elektroda positif, atau anoda, dihubungkan dengan kutub positif sumber listrik. Elektroda negatif, atau katoda dihubungkan dengan kutub negatif sumber listrik. Susunan detektor isian gas adalah sebagai berikut.

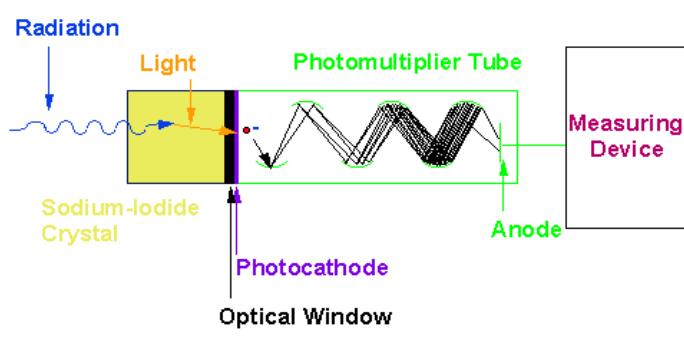


Gambar 2.15 Ilustrasi susunan detektor isian gas

Sumber gambar : IAEA. (2013). Neutron Dosimetry and Monitoring : Active Methods of Neutron Detection.

Prinsip kerja dari detektor isian gas adalah radiasi yang mengenai permukaan detektor akan mengionisasi gas yang terdapat di dalam tabung detektor menghasilkan sejumlah ion. Ion tersebut akan ditangkap oleh kedua elektroda yang telah diberi beda potensial listrik dari sumber listrik. Jumlah ion pada elektroda merupakan jumlah muatan listrik yang terdapat pada detektor yang membentuk pulsa listrik dengan nilai tertentu. Besar atau tinggi nya nilai pulsa listrik tersebut tergantung pada energi radiasi dan tegangan kerja dari detektor.

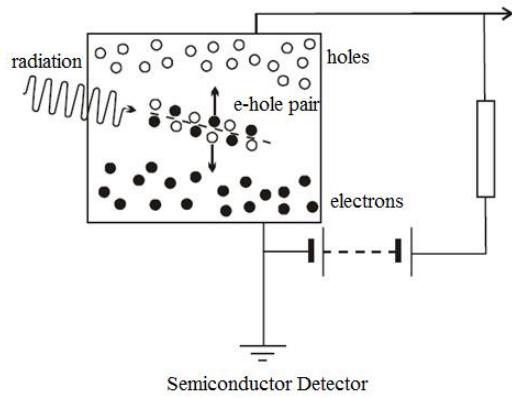
Detektor sintilasi merupakan detektor radiasi dari bahan sintilator sebagai medium peka radiasi dan memiliki komponen *photomultiplier*. Bahan sintilator merupakan jenis bahan yang peka dan bereaksi terhadap radiasi dan menghasilkan berkas cahaya. Bahan sintilasi yang digunakan seperti Sodium Iodide ($\text{NaI}(\text{Tl})$), Lanthanum Bromide ($\text{LaBr}_3(\text{Ce})$), Cerium Bromide (CeBr_3), dan Strontium Iodide (SrI_2). Selanjutnya, berkas cahaya tersebut akan melewati *photomultiplier tube* (PMT) untuk meningkatkan intensitas berkas cahaya tersebut dan dikonversi menjadi sinyal listrik. Sinyal listrik dari PMT akan diolah komponen - komponen elektronik dan modul pendukung lain untuk pencacahan dan diskriminasi energi.



Gambar 2.16 Ilustrasi susunan detektor sintilasi

Sumber gambar : https://link.springer.com/chapter/10.1007/978-3-030-80116-8_5

Detektor semikonduktor merupakan jenis detektor radiasi yang menggunakan bahan semikonduktor sebagai bahan yang peka terhadap pancaran radiasi. Jenis bahan semikonduktor yang digunakan adalah silikon ataupun germanium. Apabila bahan semikonduktor tersebut berinteraksi dengan radiasi pengion, maka bahan tersebut akan menghasilkan pasangan-pasangan *electron-hole*. Semakin banyak radiasi yang mengenainya, maka semakin banyak jumlah pasangan yang dihasilkan. Saat detektor terhubung dengan sumber energi listrik dan dihasilkan medan listrik pada detektor, elektron akan bergerak menuju ke salah satu elektrode dan hole akan bergerak ke elektrode yang lain. Pergerakan ini menghasilkan sinyal listrik. Sinyal tersebut akan diolah dan dianalisis pada komponen elektronik lain.



Gambar 2.17 Ilustrasi susunan detektor semikonduktor

Sumber gambar : Hasan, H, A, 2014

Ketiga jenis detektor tersebut memiliki kualitas dan karakteristik yang berbeda-beda. Kualitas pada detektor radiasi meliputi resolusi energi dan efisiensi deteksi. Karakteristik berupa prinsip kerja dan konstruksi di dalam detektor tersebut. Berikut merupakan tabel perbandingan kualitas kinerja ketiga tipe detektor radiasi.

Tabel 2.3 Tabel perbandingan kualitas kinerja ketiga tipe detektor radiasi

Kriteria	Detektor Isian Gas	Detektor Sintilasi	Detektor Semikonduktor
Prinsip kerja	Ionisasi gas oleh radiasi.	Cahaya foton dari bahan sintilasi akibat radiasi.	Radiasi mengenai bahan semikonduktor menghasilkan pasangan <i>electron-hole</i> .
Resolusi	Rendah	Sedang	Tinggi
Efisiensi	Rendah pada radiasi gamma dan x-ray	Tinggi pada radiasi gamma dan x-ray	Sedang, pada radiasi gamma dan x-ray
Sensitivitas	Terbatas pada radiasi gamma dan x-ray.	Baik pada radiasi gamma dan x-ray	Baik pada radiasi gamma dan x-ray.

2.2.11 Spektroskopi Gamma

Spektroskopi, dalam penelitian tugas akhir ini merupakan spektroskopi nuklir, merupakan metode pengukuran dan analisis radiasi berupa pengukuran distribusi energi yang dihasilkan dari sumber radioaktif dalam bentuk fungsi spektrum energi.

Spektrum energi radiasi bersifat unik. Setiap radionuklida memiliki ciri khas spektrum energi radiasi tertentu dan hanya dimiliki oleh radionuklida tersebut. Dengan metode ini, jenis radionuklida pada suatu bahan dapat ditentukan berdasarkan spektrum energi yang terukur dan puncak-puncak energi yang terdapat pada spektrum energi radiasi tersebut. Setiap jenis radiasi, seperti radiasi alpha, beta, dan gamma, juga memiliki bentuk spektrum energi radiasi yang berbeda-beda berdasarkan pada jenis radionuklida, jenis peluruhan dari radionuklida tersebut, dan jenis perangkat detektor yang digunakan untuk spektroskopi.

Spektroskopi gamma merupakan teknik analisis spektrum energi radiasi gamma dari suatu radionuklida. Radiasi sinar gamma menjadi salah satu ciri khas pada beberapa radionuklida. Dengan menganalisis distribusi energi radiasi gamma, jenis radionuklida dapat diidentifikasi dan dihitung konsentrasiannya.

Setiap radionuklida menghasilkan energi radiasi gamma yang berbeda-beda. Energi radiasi tersebut diukur dengan perangkat detektor, seperti HPGe (*High Purity Germanium*) ataupun NaI (*Sodium Iodide*), disertai dengan modul-modul elektronik. Saat radiasi gamma mengenai detektor, dihasilkan pulsa muatan listrik. Besarnya muatan sebanding dengan energi radiasi gamma. Pulsa muatan listrik tersebut diolah dengan memperbesar dan membentuk menjadi sinyal listrik terdigitalisasi. Sinyal tersebut dipisah dan disimpan di MCA (*Multi Channel Analyzer*) untuk membentuk spektrum gamma. Spektrum ini menampilkan hasil pencacahan sebagai intensitas radiasi gamma yang terdeteksi pada setiap level energi. Dengan menganalisis puncak spektrumnya, maka dapat ditentukan jenis radionuklida.

Proses spektroskopi gamma dimulai dengan kalibrasi. Kalibrasi spektroskopi gamma untuk memastikan keakuratan hasil spektroskopi gamma. Proses kalibrasi akan

menentukan hubungan antara energi gamma dan nomor *channel*, serta mengetahui efisiensi dari pengukuran spektroskopi gamma. Ada dua jenis kalibrasi yang perlu dilakukan, yaitu kalibrasi energi dan kalibrasi efisiensi. (Malta, A., R., 2025)

Kalibrasi energi merupakan analisis kualitatif dalam spektroskopi gamma untuk menentukan hubungan antara energi gamma suatu radionuklida dengan nomor *channel* pada perangkat spektroskopi. Kalibasi ini dilakukan dengan sumber radioaktif standar yang sudah diketahui energinya dengan tepat kemudian dicacah secara langsung dengan perangkat spektroskopi gamma. Energi dan nomor *channel* memiliki hubungan berbanding lurus, sehingga kurva yang didapatkan berupa garis lurus yang linear dengan persamaan matematis linearnya adalah sebagai berikut.

$$E = A \cdot Ch + B \quad 2.8$$

Ch merupakan nomor *channel*, E adalah nilai energi gamma dari puncak spektrum, serta A dan B merupakan konstanta kalibrasi.

Kalibrasi efisien merupakan analisis kuantitatif dalam spektroskopi gamma untuk mengetahui persentase jumlah energi gamma aktual yang terukur oleh detektor. Radionuklida memancarkan energi radiasi gamma ke segala arah, namun detektor hanya mengukur energi radiasi tersebut pada satu arah. Perhitungan nilai kalibrasi efisiensi menggunakan persamaan berikut.

$$Ef = \frac{Cps}{A \cdot Y \cdot f} \times 100\% \quad 2.9$$

Ef merupakan nilai efisiensi mutlak detektor.

Cps adalah cacah per sekon terukur pada saat pengukuran.

A adalah aktivitas radionuklida terhitung pada saat pengukuran.

Y adalah *yield* atau probabilitas jumlah dari foton energi gamma radionuklida pada tiap aktivitasnya.

f adalah faktor geometri, yang ditentukan dengan persamaan berikut.

$$f = 2\pi \left(1 - \frac{d}{\sqrt{d^2 + r^2}} \right) \quad 2.10$$

d merupakan jarak detektor dengan sumber (cm)

r merupakan jari-jari jendela detektor (cm)

Nilai Cps ditentukan dengan persamaan berikut.

$$Cps = \frac{\text{total cacah puncak energi nett}}{\text{lama waktu pencacahan (sekon)}} \quad 2.11$$

Kualitas dari proses spektroskopi juga ditentukan dari resolusi spektroskopi. Resolusi merupakan kemampuan perangkat modul spektroskopi gamma dalam membedakan berbagai energi yang dihasilkan dari sumber radiasi. Resolusi yang baik dapat membedakan energi – energi radiasi dengan lebih detail dan teliti. Penentuan resolusi dilakukan dengan menetapkan ROI (*Region of Interest*). ROI merupakan daerah terpilih yang terdapat puncak spektrum energi gamma. Dari ROI tersebut dapat ditentukan ukuran lebar spektrum puncak tersebut sehingga nilai resolusi dapat ditentukan. Penentuan resolusi menggunakan persamaan berikut

$$\text{Resolusi} = \frac{FWHM}{P} \times 100\% \quad 2.12$$

FWHM, atau *Full Width Half Modulation*, merupakan ukuran dari ROI yang berupa puncak spektrum dengan ukuran lebar setengah dari cacah energi puncaknya.

P merupakan nilai puncak energi spektrum pada ROI.

Teknik spektroskopi gamma menghadirkan berbagai fungsi diantaranya untuk mengidentifikasi jenis radionuklida, mengukur konsentrasi radionuklida, mengetahui sifat dan komposisi radionuklida, dan memantau kontaminasi lingkungan akibat radionuklida. Spektroskopi gamma untuk mengidentifikasi jenis radionuklida berupa menganalisis nilai energi radiasi gamma terukur serta intensitas atau cacah dari energi tersebut. Setiap jenis radionuklida memiliki ciri khas yang berbeda satu sama yang lain berupa besar intensitas atau cacah dari energi radiasi gamma yang dipancarkannya.

2.2.12 Radionuklida

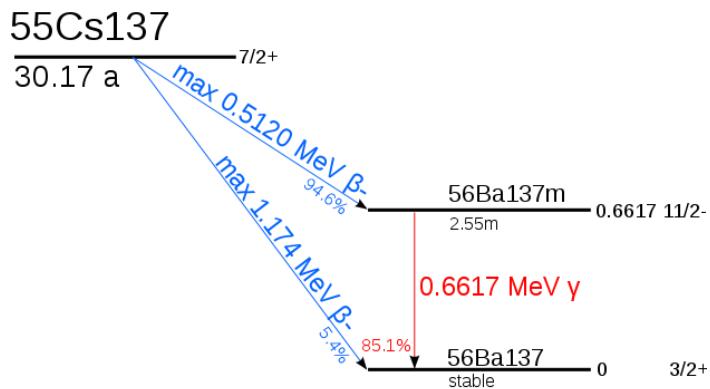
Radionuklida, atau isotop radioaktif, merupakan zat yang memancarkan radiasi elektromagnetik pengion berupa radiasi alpha, beta, dan gamma dikarenakan sifat dari inti atom zat tersebut yang tidak stabil. Radionuklida dihasilkan dari berbagai proses, baik dari proses alami dan juga proses buatan manusia. Radionuklida yang berasal dari proses alam umumnya dapat ditemukan pada tanah dan batuan. Contoh radionuklida dari proses alam adalah Uranium-238, Thorium-232, dan Potassium-40. Radionuklida yang berasal dari proses buatan manusia diperoleh setelah proses reaksi nuklir seperti reaksi di reaktor nuklir ataupun ledakan senjata nuklir. Contoh radionuklida buatan manusia adalah Cesium-137 dan Strontium-90.

Terdapat berbagai radionuklida pemancar radiasi gamma diantaranya

1. Cesium-137.

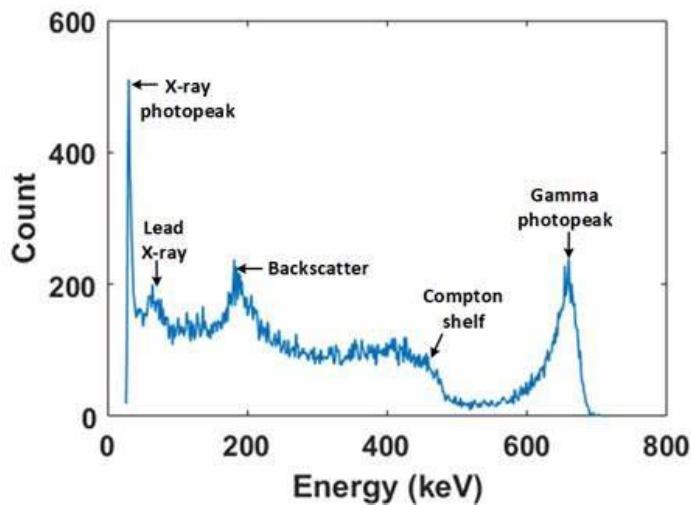
Cesium-137 (Cs-137) merupakan isotop radioaktif dari unsur Cesium dengan nomor massa 137 dan nomor atom 55. Umur paruh radionuklida ini selama 30,17 tahun. Isotop ini meluruh menjadi Ba-137 dan menghasilkan radiasi beta dan gamma. Skema peluruhan Cs-137 terdapat pada Gambar 2.18. Energi gamma yang dihasilkan dari peluruhan sebesar 661,65 keV. Bentuk spektrum radiasi gammanya terdapat pada Gambar 2.19. Berdasarkan pada gambar tersebut, spektrum radiasi gamma Cs-137 terdiri dari berbagai jenis interaksi radiasi, diantaranya X-ray, *backscatter*, *compton shelf*, dan *gamma photopeak*.

Radionuklida ini digunakan untuk berbagai keperluan, seperti di bidang kesehatan untuk radioterapi pengobatan kanker, di bidang industri untuk pengukuran ketebalan dan densitas material, sterilisasi peralatan medis dan produk makanan, dan juga cukup sering digunakan dalam kalibrasi peralatan deteksi radiasi.



Gambar 2.18 skema peluruhan Cs-137

Sumber gambar : <https://maximus.energy/index.php/2020/10/24/the-rich-physics-of-cs-137-gamma-spectrum/>



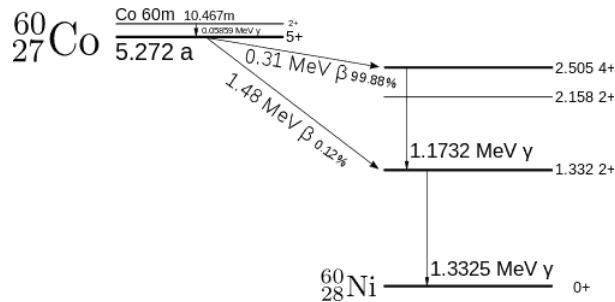
Gambar 2.19 Spektrum radiasi gamma Cs-137

Sumber gambar : Ukaegbu, I. K., & Gamage, K. A. A. (2018)

2. Cobalt-60.

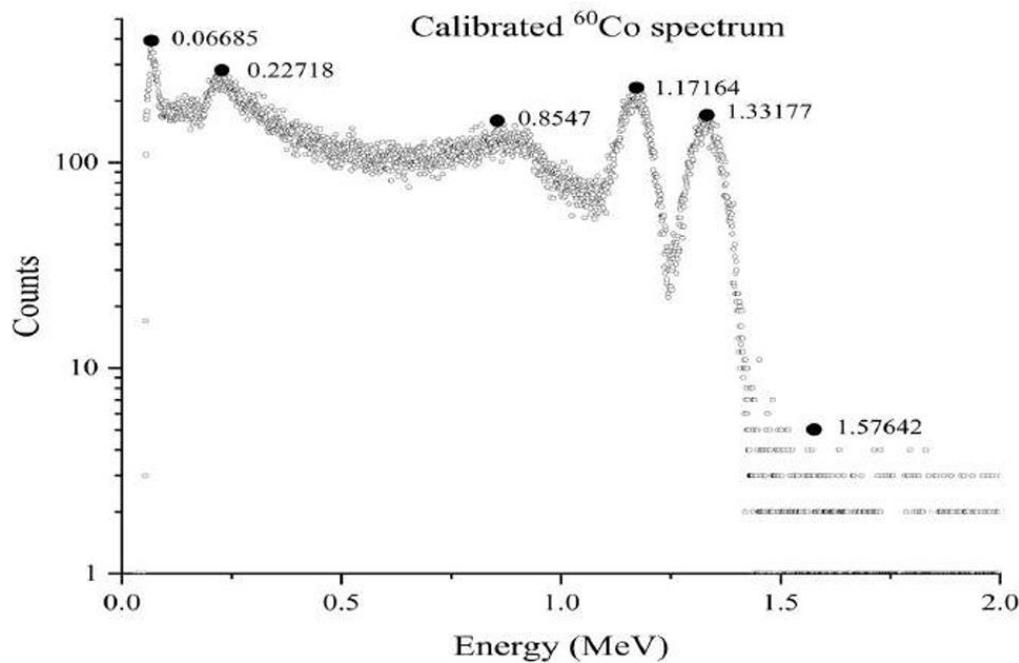
Cobalt-60 (Co-60) merupakan isotop dari unsur Cobalt dengan nomor massa 60 dan nomor atom 27. Umur paruhnya selama 5,27 tahun. Cobalt-60 meluruh dan menghasilkan radiasi gamma dengan energi sebesar 1,17 MeV dan 1,33 MeV. Radionuklida Co-60 dihasilkan secara sintesis dari proses penembakan neutron ke Cobalt-59 pada reaktor nuklir. Skema peluruhan Co-60 terdapat

pada Gambar 2.20. Sebagai atom yang tidak stabil, Co-60 meluruh menjadi Nickel-60 dan menghasilkan radiasi beta dan gamma.



Gambar 2.20 skema peluruhan Co-60

Sumber gambar : <https://www.nuclear-power.com/nuclear-engineering/radiation-detection/gamma-spectroscopy/cobalt-60-spectrum/>



Gambar 2.21 Spektrum energi radiasi gamma Co-60

Sumber gambar : Rashid. N. 2018

Bentuk spektrum energi radiasi gamma dari Co-60 terdapat pada Gambar 2.21. Berdasarkan pada gambar tersebut, spektrum Co-60 memiliki dua puncak energi pada energi 1,17 MeV dan 1,33 MeV. Energi puncak *Compton scatter* pada energi 0,85 MeV. Energi puncak *backscatter* pada energi 0,23 MeV.

Co-60 digunakan untuk berbagai keperluan, seperti pada terapi pengobatan kanker, sterilisasi peralatan medis, inspeksi komponen pada radiografi, dan juga pada proses iradiasi produk makanan.

2.2.13 Universal Computer Spectrometer UCS30.

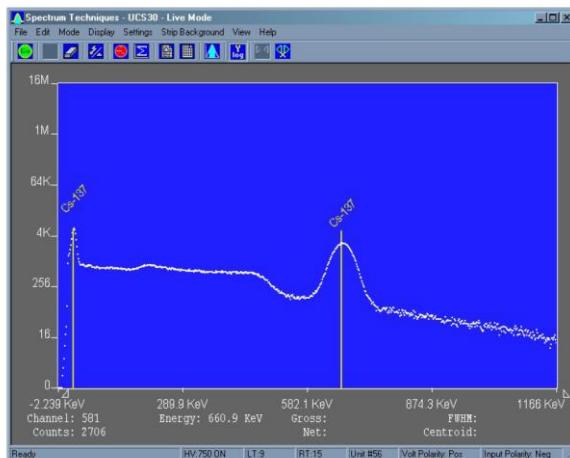
UCS30 merupakan modul instrumentasi kenukliran untuk pengukuran dan akuisisi data spektroskopi nuklir yang dioperasikan dengan menggunakan perangkat PC. Paket modul instrumentasi ini terdiri dari perangkat keras (*hardware*) dan perangkat lunak (*software*). Pada modul *Hardware*, komponen ADC (*Analog-to-Digital Converter*) 4K dengan memori 8K dan *multi-channel scaling* untuk memproses sinyal listrik dari detektor sintilasi menjadi data spektrometri. Modul *hardware* terhubung dengan komputer melalui metode antarmuka *Universal Serial Bus* (USB). Modul *hardware* dioperasikan dengan menggunakan sumber daya listrik AC 100 – 250 V. Terdapat komponen mikrokontroler dalam modul *hardware* sebagai *master controller* dan menyimpan data serta penghubung komunikasi dengan antarmuka USB. Komponen *multi-channel analyzer* memiliki berbagai fitur seperti mengubah pengaturan *amplifier* dan *high voltage* untuk tabung PM, *upper* dan *lower-level discriminator*, dan memori data pada instrumen. Komponen *amplifier* dan *high voltage* yang terintegrasi dan kompatibel dengan detektor sintilasi sehingga tidak memerlukan tambahan modul lainnya. Pengaturan kalibrasi, *coarse gain*, *fine gain*, *high voltage*, dan *lower-upper-level discriminator* dapat dilakukan melalui perangkat lunak yang terdapat pada perangkat PC. Komponen ADC dapat diatur untuk *conversion gain* berupa 4096, 2048, 1024, 512, hingga 256 *channel*.



Gambar 2.22 Modul hardware UCS30

Sumber gambar : Spectrum Techniques, LLC

Modul *software* UCS-30 bernama Spectrum Techniques USX dapat digunakan pada perangkat komputer dengan sistem operasi versi Windows 2000 hingga yang terbaru. Perangkat lunak ini memberikan pengaturan penuh terhadap fitur-fitur dalam melakukan pengukuran spektroskopi seperti *preset live/real time, region of interest, centroid, gross* dan *net area calculations*. Perangkat lunak ini juga dapat mengatur komponen modul *hardware*, seperti *amplifier, high voltage, ADC*, dan diskriminator. *Software* UCS-30 menghasilkan tampilan data spektrum secara langsung dan berwarna serta beresolusi tinggi.



Gambar 2.23 Tampilan software UCS30

Sumber gambar : Spectrum Techniques, LLC

2.3 Hipotesis

Berdasarkan pada tinjauan pustaka dan landasan teori tersebut, susunan hipotesis penelitian tugas akhir ini adalah sebagai berikut

1. Program model TinyML dapat digunakan pada perangkat MCU ESP32-WROOM-32 untuk identifikasi dan klasifikasi radionuklida pemancar radiasi gamma, khususnya pada jenis radionuklida Co-60 dan Cs-137.
2. Kinerja model TinyML berupa jumlah *type 1 error (false positive)* dan *type 2 error (false negative)* yang minimum, dengan target nilai akurasi sebesar 80%

BAB 3 METODOLOGI

3.1 Waktu & Tempat

Pelaksanaan penelitian tugas akhir ini dilaksanakan di Laboratorium Instrumentasi Nuklir, Politeknik Teknologi Nuklir Indonesia pada bulan Januari hingga Juni 2025. Pembagian waktu beserta tahapan kegiatan penelitian terdapat pada tabel 3.1

Tabel 3.1 Waktu pelaksanaan tugas akhir

No	Kegiatan	2025					
		Jan	Feb	Mar	Apr	Mei	Jun
1	Studi Pustaka & Dokumentasi	X	X	X	X	X	X
2	Pengumpulan dan pengolahan dataset	X	X	X			
3	Pelatihan model TinyML		X	X	X	X	
4	Pengujian model TinyML			X	X	X	
5	Analisis data			X	X	X	X
6	Penyusunan Laporan Tugas Akhir				X	X	

3.2 Alat dan Bahan

Pelaksanaan Tugas Akhir ini menggunakan perangkat dan bahan berupa perangkat keras dan perangkat lunak sebagai berikut

Tabel 3.2 Perangkat yang digunakan

Perangkat Keras (<i>Hardware</i>)	Perangkat Lunak (<i>Software</i>)
<ol style="list-style-type: none">1. Radionuklida Pemancar Radiasi Gamma.2. Perangkat komputer.3. <i>Microcontroller Unit</i> ESP32.4. Detektor Sintilasi NaI(Tl).	<ol style="list-style-type: none">1. <i>Software UCS30</i>.2. <i>Code Editor</i> (Visual Studio Code dan Arduino IDE).3. <i>Language program</i> (Python dan C++).

Perangkat Keras (<i>Hardware</i>)	Perangkat Lunak (<i>Software</i>)
5. Modul Universal Computer Spectrometer UCS30.	4. <i>Library</i> pemrograman

Keterangan tambahan dan spesifikasi mengenai peralatan dan bahan yang digunakan tersebut adalah sebagai berikut.

1. Radionuklida Pemancar Radiasi Gamma.

Penelitian tugas akhir ini menggunakan radionuklida pemancar radiasi gamma yang tersedia di Laboratorium Instrumentasi Nuklir, yaitu Co-60 dan Cs-137. Kondisi atau spesifikasi dari radionuklida tersebut terdapat pada tabel 3.3

Tabel 3.3 Spesifikasi radionuklida yang digunakan

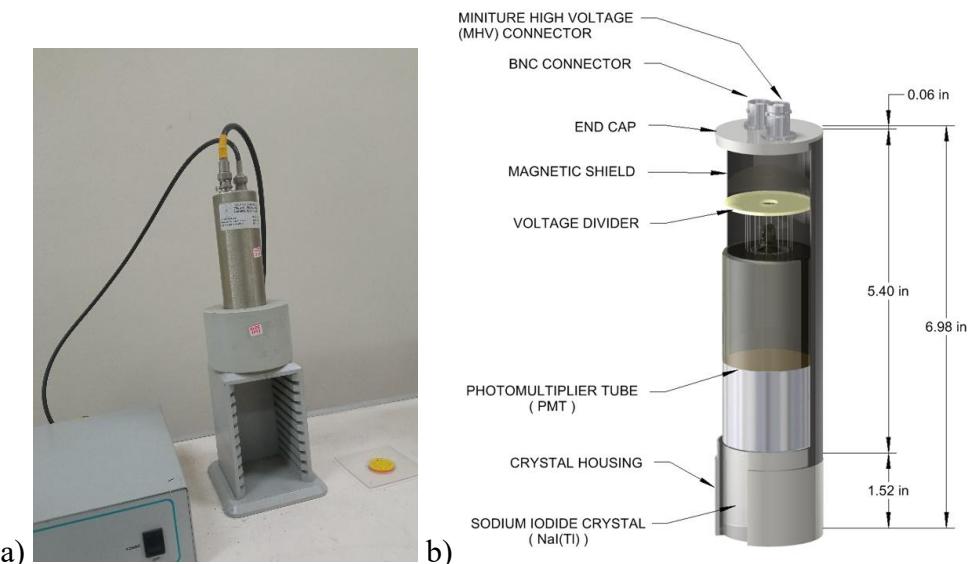
No	Radionuklida	Spesifikasi	Nilai
1	Co-60 (kode 10)	Aktivitas awal	1 mikroCi
		Tanggal awal	Nov 2011
		Waktu paruh	5,27 tahun
		Tanggal terbaru	5 Mei 2025
		Aktivitas terbaru	0,169 mikroCi
2	Co-60 (kode 11)	Aktivitas awal	1 mikroCi
		Tanggal awal	September 2017
		Waktu paruh	5,27 tahun
		Tanggal terbaru	21 Januari 2025
		Aktivitas terbaru	0,358 mikroCi
3	Co-60 (kode 12)	Aktivitas awal	1 mikroCi
		Tanggal awal	September 2017
		Waktu paruh	5,27 tahun
		Tanggal terbaru	21 Januari 2025
		Aktivitas terbaru	0,358 mikroCi

No	Radionuklida	Spesifikasi	Nilai
4		Aktivitas awal	0,25 mikroCi
		Tanggal awal	September 2017
		Waktu paruh	30,07 tahun
		Tanggal terbaru	21 Januari 2025
		Aktivitas terbaru	0,21 mikroCi
5		Aktivitas awal	1 mikroCi
		Tanggal awal	November 2011
		Waktu paruh	30,07 tahun
		Tanggal terbaru	21 Januari 2025
		Aktivitas terbaru	0,736 mikroCi
6		Aktivitas awal	5 mikroCi
		Tanggal awal	September 2011
		Waktu paruh	30,07 tahun
		Tanggal pengukuran	5 Mei 2025
		Aktivitas pengukuran	3,65 mikroCi

2. Detektor Sintilasi NaI(Tl) dan Modul UCS30.

Detektor sintilasi NaI(Tl) digunakan untuk mendeteksi dan mengukur pancaran radiasi gamma. Produk detektor NaI(Tl) yang digunakan adalah Radiation Sensors, LLC NaI(Tl) detector model 6s6p1.5vdc2. Detektor ini beroperasi dengan menggunakan tegangan tinggi positif yang berasal dari modul spektroskopi gamma dan terhubung melalui *port connector* MHV (*Miniture High Voltage*) yang terdapat di atas detektor. Nilai tegangan input sebesar 625 V hingga 1000 V. Detektor sintilasi 6S6P1.5VD memiliki tabung *Photomultiplier* (PMT) berukuran 1,5 inchi dalam 10 *stage* yang terhubung langsung dengan silinder kristal sintilasi NaI(Tl) berukuran 1,5 inchi x 1,5 inchi. Di atas PMT terdapat rangkaian *voltage divider*. Di bagian bawah detektor terdapat jendela masuk untuk radiasi dari logam aluminium tipis dengan

ketebalan 0,01 inchi. Detektor ini terlindungi dengan *casing* protektif untuk melindungi dari medan elektromagnetik luar. Sinyal output dari detektor melalui konektor BNC terhubung dengan komponen preamplifier modul spektroskopi gamma. Detektor ini dapat mendeteksi pancaran radiasi X-Ray dan gamma berenergi 30 KeV hingga 1 MeV. Detektor ini memiliki kinerja resolusi energi sebesar 8.5% dengan menggunakan radionuklida Cs-137 berenergi 662 keV. (6S6P1.5VD User's Manual)



Gambar 3.1 a) Detektor sintilasi dan b) Ilustrasi susunan komponen detektor

Sumber gambar : Radiation Sensors 6S6P1.5VD Scintillation Detector User's Manual

Perangkat detektor tersebut terhubung dengan modul Spectrum Technique Universal Computer Spectrometer UCS30. Modul UCS30 digunakan untuk memproses sinyal dari detektor menjadi spektrum radiasi gamma. Pancaran radiasi gamma dari radionuklida akan diterima dan diubah menjadi sinyal listrik analog oleh detektor. Sinyal tersebut oleh modul UCS30. Di dalam modul UCS30, terdapat sejumlah komponen elektronik untuk memproses sinyal tersebut, diantaranya amplifier, *discriminator*, dan ADC. Pemprosesan sinyal dari detektor berupa meningkatkan kekuatan sinyal (amplifikasi), optimasi karakteristik sinyal, dan konversi sinyal analog menjadi sinyal digital. Data sinyal digital tersebut tersimpan sementara dalam komponen memori untuk selanjutnya dikirim ke perangkat komputer.

3. Perangkat komputer.

Perangkat komputer yang digunakan terdiri dari dua jenis, yaitu perangkat komputer untuk akuisisi dataset spektroskopi gamma dan perangkat komputer untuk penyusunan, pelatihan, dan pengujian model TinyML

Komputer untuk akuisisi dataset spektroskopi gamma berupa komputer *deskstop* Hewlett-Packard HP Compaq 6000 Pro SFF PC dengan prosesor Intel Core2 Duo CPU E7500 2,93 GHz dan RAM 972 MB. Sistem operasi pada komputer tersebut adalah Windows XP. Komputer ini terhubung dengan modul UCS30 dan terdapat *software* untuk menggunakan modul UCS30.



Gambar 3.2 Perangkat komputer akuisisi data spektroskopi gamma beserta detektor NaI(Tl) dan UCS30

Komputer untuk penyusunan program dan pengujian model TinyML berupa komputer portabel Lenovo IdeaPad 3 14ALC6 dengan prosesor AMD Ryzen 5 5500U dengan Radeon Graphics 2.10 GHz dan RAM 8 GB. Sistem operasi komputer ini adalah Windows 10. Perangkat komputer ini memiliki *software* dan *library* untuk penyusunan model TinyML.

4. Software UCS30.

Hardware UCS30 disertai dengan *software* UCS30. *Software* ini digunakan untuk mengatur kondisi dari perangkat *hardware* UCS30 dalam proses akuisisi data spektroskopi gamma, menampilkan data hasil spektroskopi serta menyimpan datanya.

Di *software* ini tersedia berbagai konfigurasi pengaturan perangkat UCS30. Terdapat pengaturan HV, Amplifier, ADC, pengaturan Live Time, pengaturan kalibrasi energi, dan pengaturan penentuan ROI (Region of Interest).

5. ESP32.



Gambar 3.3 ESP32

Sumber gambar: Espressif system

Pada penelitian tugas akhir ini akan menggunakan perangkat MCU dari produk Espressif ESP-32-WROOM-32. Perangkat MCU ini memiliki spesifikasi sebagai berikut.

- Processor : Dual-core Tensilica LX6
- Frekuensi clock 240 MHz.
- RAM : 520 KB
- Memori Flash : 4 MB
- Jumlah Pin I/O : 38 pin
- Tegangan sumber daya : 3,0 V – 3,6 V

ESP32-WROOM-32 dipilih berdasarkan pertimbangan perbandingan spesifikasi terhadap beberapa produk MCU lain yang ditunjukkan pada tabel 2.2. ESP32-WROOM-32 memilih keunggulan spesifikasi pada jenis prosesor, ukuran memori, dan frekuensi clock yang digunakan. ESP32-WROOM-32 dapat diprogram dari Arduino IDE ataupun Espressif IDF. Dengan spesifikasi tersebut, ESP32 dapat menjalankan program TinyML.

6. *Code Editor*

Code Editor merupakan software untuk menyusun dan mengubah baris program hingga menguji program. *Software code editor* yang digunakan dari produk Microsoft Visual Studio Code versi 1.92.2 dan juga Arduino IDE dari versi 2.3.4. Visual studio code digunakan untuk menyusun program TinyML untuk melatih dan menguji program model TinyML. Arduino IDE digunakan untuk menyusun program deploy model TinyML, dan mengunggah nya ke ESP32.

7. Bahasa pemrograman Python dan C++

Penyusunan program TinyML menggunakan bahasa pemrograman Python dan C++. Python digunakan dalam proses pengolahan dataset, penyusunan dan pelatihan model TinyML, pengujian model TinyML, hingga analisis data. Bahasa pemrograman C++ digunakan dalam penyusunan program untuk mengunggah model TinyML ke perangkat MCU ESP32-WROOM-32 dan menguji model TinyML pada perangkat MCU tersebut.

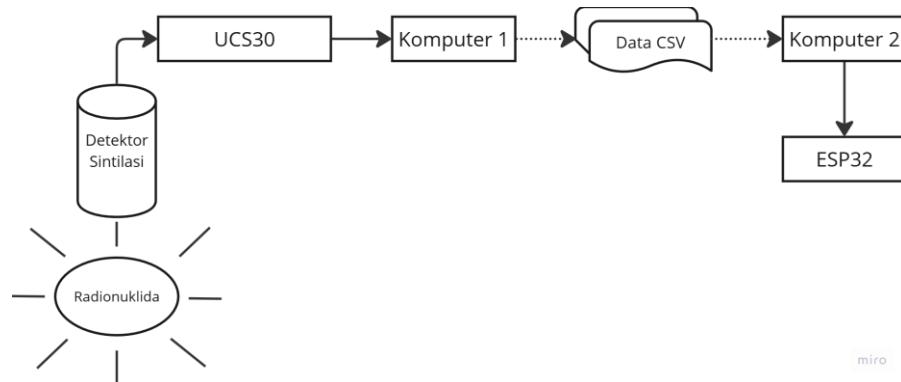
8. Library pemrograman.

Dalam penyusunan program model TinyML, diperlukan beberapa *library* pemrograman. *Library* pemrograman merupakan set program pendukung yang telah disusun oleh *developer* untuk menyederhanakan pembuatan dan pengembangan model ML dan TinyML. Pada *library*, terdapat abstraksi *high-level* untuk operasi matematika kompleks, susunan lapisan neuron, fungsi aktivasi, *optimizers*, dan *loss function*.

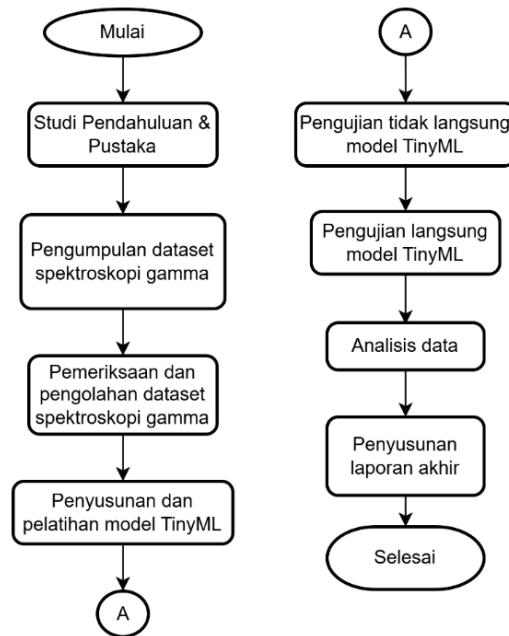
Pada proses penyusunan model TinyML, *Library* pemrograman digunakan dalam pengolahan dataset, penyusunan dan pelatihan model TinyML, pengujian model TinyML, dan analisis data. *Library* pemrograman yang digunakan dalam proses pembuatan dan pelatihan model ML dan TinyML diantaranya TensorFlow, Keras, TensorFlow Lite, Pandas, Numpy, dan Matplotlib.

Untuk menguji program TinyML secara langsung pada perangkat MCU, diperlukan *library* pemrograman khusus pada program Arduino. Library yang digunakan adalah ArduTFLite.

Susunan perangkat keras yang digunakan adalah sebagai berikut.



Gambar 3.4 Susunan perangkat keras



Gambar 3.5 Alur pelaksanaan tugas akhir

3.3 Metodologi

Pelaksanaan tugas akhir ini dilaksanakan dalam delapan tahapan berurutan sebagaimana terdapat pada *flowchart* pada gambar 3.5.

- 1. Studi pendahuluan & pustaka.**

Studi pendahuluan berupa penentuan rumusan dan batasan masalah dengan melakukan studi pustaka terhadap publikasi ilmiah yang terkait dengan TinyML dan program machine learning untuk identifikasi radionuklida berdasarkan spektroskopi gamma. Sumber referensi dan publikasi ilmiah yang digunakan berupa karya ilmiah dalam bentuk jurnal, prosiding konferensi ilmiah, artikel situs web, dan dokumentasi pemrograman. Pada tahapan ini, berdasarkan perumusan masalah dan studi pustaka yang dilakukan, ditentukan dan dipersiapkan juga peralatan serta bahan penelitian, teknik akuisisi hingga analisis data, serta jenis algoritma TinyML yang digunakan. Masalah dalam penelitian tugas akhir ini adalah bagaimana proses pembuatan program TinyML untuk identifikasi radionuklida dan bagaimana kinerja program TinyML yang disusun. Permasalahan tersebut dibatasi pada penggunaan hanya dua jenis radionuklida, yaitu Co-60 dan Cs-137, menggunakan jenis algoritma ML Artificial Neural Network serta perangkat MCU dari produk Espressif ESP32.

- 2. Pengumpulan dataset spektroskopi gamma.**

Tahapan pengumpulan dataset spektroskopi gamma berupa melakukan pengukuran spektroskopi gamma dari radionuklida yang terdapat pada tabel 3.3 dengan menggunakan perangkat detektor NaI(Tl), modul spektroskopi UCS30, dan komputer akuisisi data spektroskopi yang terdapat di Laboratorium Instrumentasi Nuklir, Politeknik Teknologi Nuklir Indonesia. Setiap radionuklida yang diukur juga akan dicatat nilai spesifikasinya berupa waktu paruh, tanggal awal, aktivitas awal, tanggal akuisisi data, dan aktivitas tertanggal terakhir sebagai data tambahan pendukung dataset spektroskopi gamma.

Sebelum dilakukan pengumpulan dataset spektroskopi gamma pada seluruh radionuklida tersebut, perangkat spektroskopi gamma beserta *software-nya* dikalibrasi

terlebih dahulu. Kalibrasi yang dilakukan berupa kalibrasi energi, kalibrasi efisiensi, dan resolusi dari perangkat modul spektroskopi gamma. Kalibrasi energi untuk menyesuaikan antara nilai *channel* dengan nilai energi radiasi gamma. Kalibrasi efisiensi untuk menentukan jumlah intensitas cacah yang diterima perangkat detektor dibandingkan dengan total intensitas cacah dari sumber radiasi. Penentuan resolusi untuk mengetahui kemampuan perangkat dalam membedakan berbagai energi radiasi yang dihasilkan dari sumber radiasi.

Kalibrasi perangkat detektor dan modul dilakukan dengan menggunakan sumber Co-60 kode 11. Tahapan dalam proses kalibrasi terdiri dari:

- 1) Menghapus data spektroskopi gamma yang masih tersimpan di *software*.
- 2) Ubah konfigurasi kalibrasi *software* menjadi *uncalibrated*
- 3) mengatur konfigurasi perangkat UCS30 yang terdiri dari *high voltage*, *conversion gain*, *coarse gain*, *fine gain*, LLD, dan ULD,
- 4) mengatur waktu *live time* akuisisi data spektroskopi,
- 5) memulai proses spektroskopi terhadap sumber standar hingga proses berhenti sesuai waktu *live time* yang telah ditetapkan,
- 6) memilih pengaturan kalibrasi mode dua *peak*,
- 7) menentukan dua puncak gelombang spektrum energi-cacah yang terukur,
- 8) memberi input nilai energi berdasarkan standar dan referensi mengenai Co-60 pada kedua peak.
- 9) Menentukan ROI (Region of Interest) dengan klik ikon ROI dan klik "Set ROI".
- 10) Pilih ROI berupa spektrum FWHM (*Full Width Half Modulation*) dari puncak spektrum yang ada.

Sebelum terkalibrasi, data hasil spektroskopi hanya berupa data *channel* dan *counts*. Setelah terkalibrasi, data hasil spektroskopi terdiri dari data *channel*, data energi, data *counts*. Apabila posisi *marker* spektrum pada *software* diarahkan pada ROI, akan ditampilkan data *Gross*, *Net*, *FWHM*, dan *Centroid*. Data kalibrasi tersebut akan dianalisis dengan menggunakan persamaan 2.8 dan 2.9.

Konfigurasi perangkat spektroskopi gamma UCS30 diatur dari *software* nya. Pengaturan tersebut terkait dengan pengaturan kondisi perangkat untuk proses pengolahan sinyal listrik oleh detektor, amplifier, dan ADC yang terdapat pada perangkat modul spektroskopi gamma. Pengaturan konfigurasi yang digunakan adalah sebagai berikut.

Tabel 3.4 Konfigurasi Perangkat Spektroskopi Gamma

Kriteria Konfigurasi	Nilai Konfigurasi
<i>Acquisition Mode</i>	PHA
<i>High voltage</i>	550
<i>Conversion gain</i>	1024
<i>Coarse gain</i>	4
<i>Fine gain</i>	1,78
<i>LLD</i>	2
<i>ULD</i>	102,3
<i>Live time</i>	100 s

Mode akuisisi yang digunakan adalah mode PHA (*Pulse Height Analysis*). Perangkat UCS30 dalam mode ini akan menganalisis amplitudo atau tinggi pulsa sinyal listrik maksimum dari detektor. Tinggi pulsa tersebut berbanding lurus dengan energi yang dihasilkan dari radionuklida yang dideteksi oleh detektor. Setiap tinggi pulsa tersebut akan diurutkan pada salah satu *channel*. Selama proses pencacahan, berbagai sinyal tersebut akan membentuk histogram atau spektrum energi.

Pengaturan *High Voltage* merupakan pengaturan nilai tegangan listrik sebagai sumber daya PMT (*photomultiplier tube*). Modul UCS30 dapat mengatur nilai suplai tegangan pada detektor dengan rentang nilai 0 - 2048 Volt dengan arus maksimum 1 mA. Dalam pengaturan nilai High Voltage, perlu memperhatikan karakteristik sumber daya dari detektor dan PMT. Tegangan listrik dengan nilai tersebut akan menghasilkan medan listrik pada PMT yang diperlukan untuk multiplikasi elektron pada saat proses atau reaksi sintilasi pada tabung detektor. Dengan nilai 550 dan polaritas positif, maka

pengaturan ini menghasilkan tegangan listrik positif sebesar 550 V sesuai dengan spesifikasi dari detektor yang digunakan.

Pengaturan coarse gain dan fine gain merupakan pengaturan perangkat Amplifier pada modul UCS30. Kedua nilai gain tersebut mengatur ukuran amplifikasi atau pembesaran sinyal dari detektor sebelum dikonversi oleh ADC. Nilai gain akan mempengaruhi bentuk dari spektrum energi dan kalibrasi energi dari modul UCS30. Nilai coarse gain sebesar 4 berarti faktor pengali dari amplifikasi sebesar 4. Fine gain sebesar 1,78 memberikan tambahan faktor pengali amplifikasi yang lebih kontinu.

Pengaturan conversion gain mengatur jumlah *channel* yang digunakan untuk konversi sinyal analog menjadi sinyal digital dan menampilkan spektrum energi. Nilai conversion gain sebesar 1024 berarti sinyal analog akan dikonversi menjadi sinyal digital pada *channel* diskrit sebanyak 1024 *channel*. Setiap sinyal merepresentasikan rentang energi tertentu.

Pengaturan nilai LLD (Low Level Discriminator) dan ULD (Upper Level Discriminator) merupakan pengaturan jangkauan atau batas nilai sinyal pulsa yang diterima dan direkam sebagai spektrum pada energi tertentu. LLD bernilai 2 berarti nilai minimum sinyal pulsa yang diterima. Apabila sinyal nilai pulsa bernilai dibawah 2, maka sinyal tersebut akan ditolak. ULD bernilai 102,3 berarti nilai maksimum tinggi pulsa yang diterima. Apabila sinyal pulsa bernilai lebih dari 102,3 berarti sinyal pulsa tersebut tidak diterima. Pengaturan nilai LLD dan ULD untuk memastikan tidak ada sinyal *noise* diluar kedua batas tersebut yang diterima dan diproses lebih lanjut.

Perangkat spektroskopi yang telah terkalibrasi energi dapat digunakan untuk akuisisi data spektroskopi gamma pada radionuklida lainnya. Setiap radionuklida yang akan dicacah dan diukur spektroskopinya diletakkan dalam wadah khusus dan berjarak sekitar 1 cm dari permukaan sensitif radiasi dari detektor. Sebelum pengukuran dijalankan, konfigurasi dari perangkat spektroskopi gamma juga diperiksa dahulu. Konfigurasi yang digunakan sama dengan konfigurasi pada tahapan kalibrasi, kecuali konfigurasi *live time* yang divariasikan. Pada setiap pengukuran spektroskopi gamma

dan pada setiap radionuklida, nilai *live time* divariasikan untuk mendapatkan berbagai variasi data spektroskopi gamma. Variasi nilai *live time*, yaitu 60 detik, 70 detik, 80 detik, 90 detik, dan 100 detik. Berbagai variasi tersebut berguna untuk proses pelatihan dan pengujian model TinyML. Berikut adalah variasi radionuklida dan nilai *live time* yang digunakan selama proses akuisisi data spektroskopi gamma.

Setiap hasil spektroskopi gamma disimpan dalam format CSV (*Comma Separated Value*). Di dalam file CSV tersebut terdapat berbagai data terkait dengan hasil spektroskopi gamma. Berikut contoh data spektroskopi gamma yang tersimpan dalam format CSV.

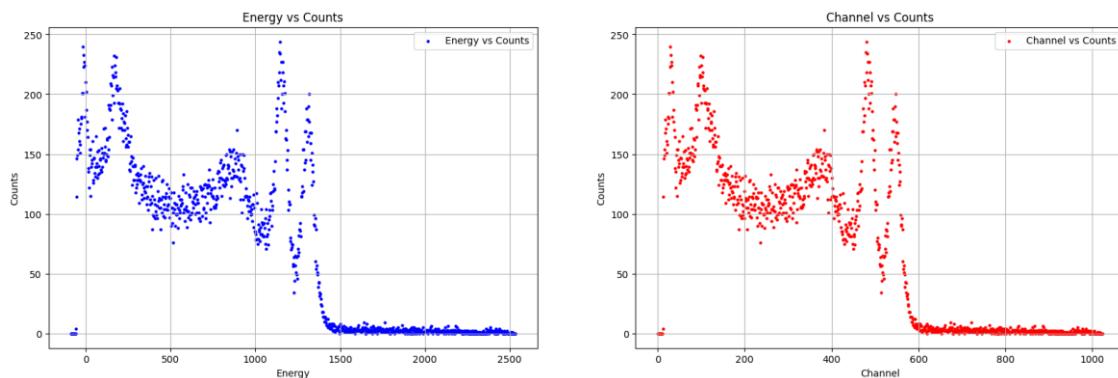
```
1 Spectrum Name:,  
2 Description:,  
3 Student ID:,  
4 Detector Used:,  
5 Comments:,  
6 Acquisition Mode:,PHA  
7 High Voltage:,550  
8 Conversion Gain:,1024  
9 Coarse Gain:,4  
10 Fine Gain:,1.78  
11 ULD:,102.300000%  
12 LLD:,2.000000%  
13 Calibration Coefficients:,-88.74,2.565  
14 Channels used to calibrate:,492,554  
15 Energies used to calibrate:,1173,1332  
16 Real Time Elapsed:,60.81  
17 Live Time Elapsed:,60.00  
18 Start Time:,Tuesday, January 21, 2025, 14:47:39  
19 End Time:,Tuesday, January 21, 2025, 14:48:40  
20  
21 Channel Data:  
22 Chan,Energy,Counts  
23 0, -88.74,0  
24 1, -86.18,0  
25 2, -83.61,0
```

Data yang tersimpan di komputer akuisisi data spektroskopi gamma dipindahkan ke komputer untuk penyusunan dan pengujian model TinyML

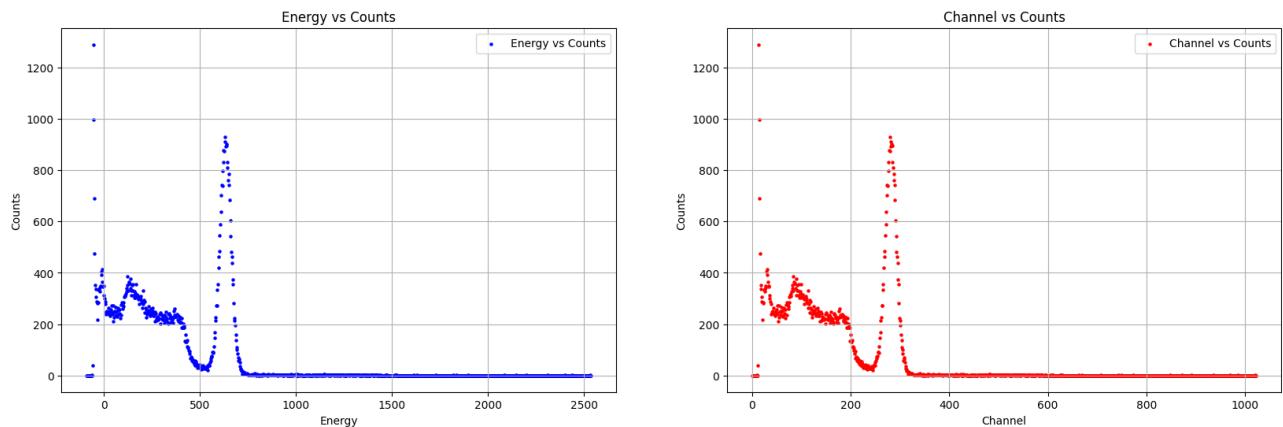
3. Pemeriksaan dan pengolahan dataset spektroskopi gamma.

Dari tahapan pengumpulan data spektroskopi gamma telah diperoleh total sebanyak 20 file dataset spektroskopi gamma. Seluruh data tersebut tidak dapat langsung digunakan untuk pelatihan dan pengujian model TinyML. Dataset tersebut harus diperiksa dan diolah terlebih dahulu. Pemeriksaan dan pengolahan dataset bertujuan untuk mempersiapkan dataset agar dapat digunakan dalam proses pelatihan dan pengujian model sesuai dengan kriteria model TinyML. Pemeriksaan dan pengolahan dataset dilakukan secara pemrograman menggunakan bahasa Python pada *code* editor Visual Studio Code. Program pemeriksaan dan pengolahan dataset beserta penjelasannya terdapat pada Lampiran 1.

Pemeriksaan data berupa pemeriksaan kelengkapan, tipe, dan ukuran dataset. Setiap dataset CSV spektroskopi gamma terdiri dari 1024 titik cacah dan tiga kolom, yaitu kolom *chan*, *energi*, dan *counts*. Sejumlah 1024 titik cacah tersebut merupakan jumlah *channel* pengukuran spektroskopi yang telah diatur sebelumnya dalam konfigurasi sistem perangkat spektroskopi. Tipe data pada tiap titik cacah merupakan tipe numerik yang berbeda-beda. Data pada kolom *Channel* dan *Counts* merupakan data dengan tipe *integer* atau data berbilangan cacah. Data pada kolom *Energi* merupakan data dengan tipe *float* atau data dengan bilangan desimal. Pemeriksaan data juga termasuk mengetahui kesesuaian grafik dataset dengan grafik spektroskopi secara teori.



Gambar 3.6 Grafik salah satu dataset Co-60



Gambar 3.7 Grafik salah satu dataset Cs-137

Pada gambar 3.6 dan gambar 3.7, terdapat dua jenis grafik yaitu grafik *Energi-Counts* dengan warna biru dan grafik *Channel-Counts* dengan warna merah. Kedua grafik tersebut identik dan sesuai dengan grafik spektrum gamma berdasarkan teori yang ada.

Pengolahan data berupa penghapusan data, pemberian label kelas, pengelompokan dan penggabungan dataset. Pengolahan dataset dilakukan secara langsung dan juga melalui program di *code editor*.

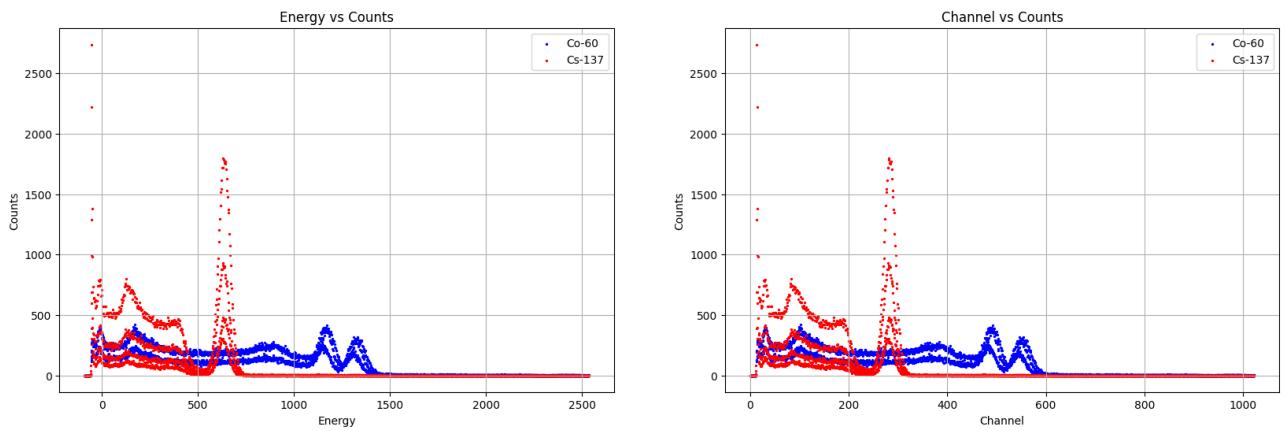
Penghapusan data berupa menghapus beberapa baris data yang tidak terpakai dan tidak dapat digunakan di proses pelatihan dan pengujian model TinyML. Data yang dihapus merupakan data yang tidak berbentuk baris dan kolom. Data tersebut merupakan data mengenai konfigurasi sistem spektroskopi gamma. Data yang tidak dihapus dan dipertahankan adalah baris-baris data yang terdiri dari tiga kolom yang terpisah koma. Penghapusan dilakukan secara langsung di file CSV. Penghapusan juga dilakukan pada kolom energi untuk menghasilkan data tidak terkalibrasi yang hanya terdiri dari 2 kolom, yaitu kolom *Chan* dan kolom *Counts*.

Setiap dataset radionuklida akan diberi label kelas. Terdapat dua label kelas, yaitu label 0 untuk Co-60 dan label 1 untuk Cs-137. Pemberian label kelas diperlukan untuk mempermudah model TinyML untuk mempelajari pola pada dataset.

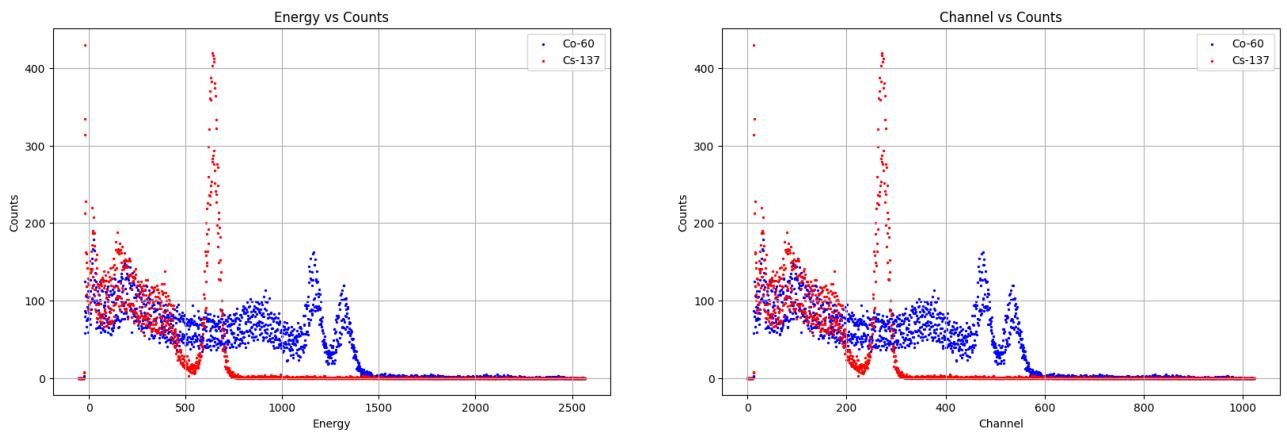
Pengelompokkan dan penggabungan dataset berupa mengelompokkan dataset menjadi tiga jenis berdasarkan kegunaannya di proses penyusunan model TinyML, yaitu dataset pelatihan, dataset validasi, dan dataset pengujian. Pengelompokkan dan penggabungan dataset bertujuan untuk mempermudah dalam pelatihan dan pengujian model TinyML. Dataset pelatihan digunakan pada tahapan pelatihan model TinyML. Dataset validasi digunakan untuk menyesuaikan konfigurasi model dengan data yang lain selama proses pelatihan. Dataset pengujian digunakan menguji kualitas kinerja model dengan data yang lain sesudah proses pelatihan selesai. Dataset yang telah dikelompokkan dapat digabung, sehingga dari total 20 file dataset menjadi tiga file dataset.

Dataset pelatihan berasal dari penggabungan empat dataset Co-60 dan empat dataset Cs-137. Total pada dataset pelatihan terdapat 8192 titik cacah. Dataset validasi berasal dari penggabungan dua dataset Co-60 dan dua dataset Cs-137. Total pada dataset validasi terdapat 4096 titik cacah. Dataset pengujian berasal dari penggabungan empat dataset Co-60 dan empat dataset Cs-137. Total pada dataset pengujian terdapat 8192 titik cacah

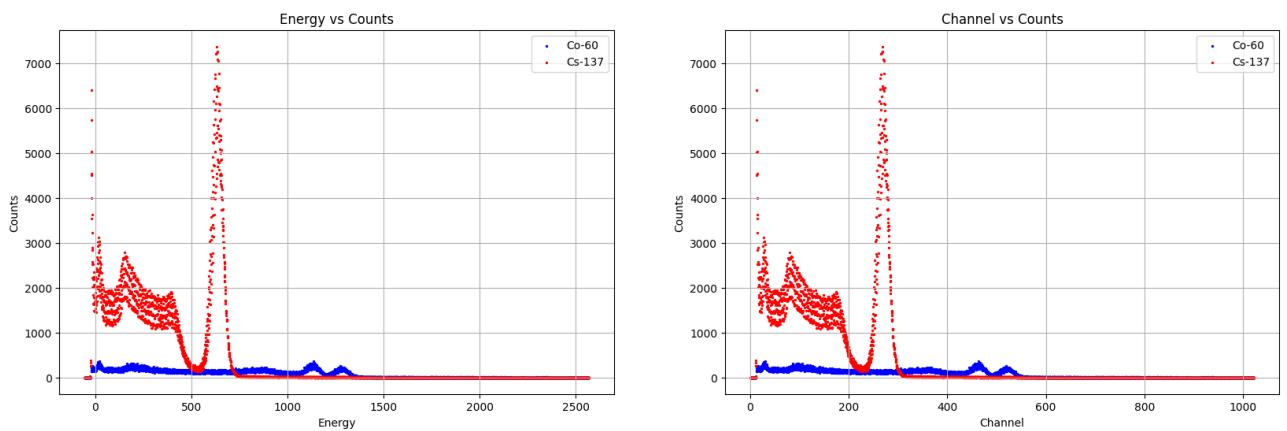
Terkhusus untuk dataset pengujian terdiri dari menjadi dua jenis dataset sesuai penggunaanya di tahapan pengujian, yaitu dataset pengujian tidak langsung dan dataset pengujian langsung. Pengujian tidak langsung menggunakan dataset yang telah digabungkan sedangkan pengujian langsung menggunakan dataset yang masih terpisah atau masih sebagai dataset spektroskopi gamma radionuklida.



Gambar 3.8 Grafik dataset pelatihan



Gambar 3.9 Grafik dataset validasi



Gambar 3.10 Grafik dataset pengujian

Dataset juga dibagi menjadi dua jenis, yaitu dataset terkalibrasi dan dataset tidak terkalibrasi. Dataset terkalibrasi memiliki tiga kolom data, yaitu kolom Chan, Energi, dan *Counts*. Dataset tidak terkalibrasi hanya memiliki dua kolom data, yaitu kolom Chan dan *Counts*.

4. Penyusunan dan pelatihan model TinyML.

Pada tahapan ini, program model TinyML disusun dan dilatih dengan menggunakan bahasa Python pada *software* Visual Studio Code dalam format file Jupyter Notebook. Penyusunan model berupa menyusun *layer*, *nodes*, dan fungsi aktivasi sebagai model ML dengan algoritma Artificial Neural Network. Ketiga kriteria tersebut divariasikan untuk mempeoleh kinerja model ML yang berbeda-beda sehingga dapat menentukan kriteria model ML dengan kualitas kinerja yang terbaik. Terdapat dua jenis model yang disusun sesuai dengan jenis data pelatihan yang digunakan, yaitu model untuk data terkalibrasi dan model untuk data tidak terkalibrasi. Program penyusunan model ML dengan algoritma Artificial Neural Network terdapat pada Lampiran 3.

Seluruh model tersebut dilatih menggunakan dataset pelatihan yang sama yang telah disusun di tahapan sebelumnya. Pelatihan model menggunakan program yang terdapat di Lampiran 3. Sesudah proses pelatihan, setiap model ML diperiksa kualitas kinerjanya sebelum dikonversi menjadi model TinyML. Pemeriksaan kualitas kinerja model ML dilakukan pada tahapan pengujian tidak langsung model TinyML.

Model ML disimpan dalam format KERAS, lalu dikonversi menjadi model TensorFlow Lite dengan format .tflite. Model dikonversi lebih lanjut menjadi model bertipe *header file* dengan format .h. Model bertipe *header file* merupakan model TinyML yang siap diuji dan dipakai di perangkat MCU.

5. Pengujian tidak langsung model TinyML

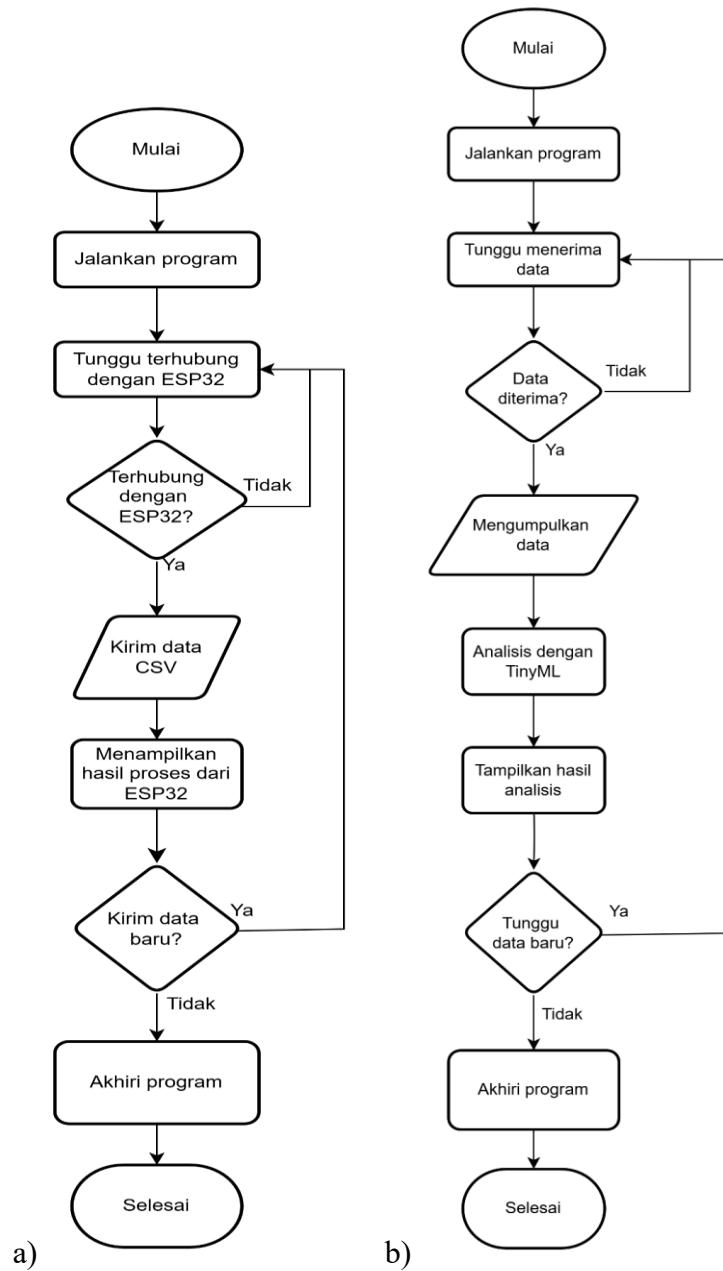
Pengujian tidak langsung merupakan pengujian model ML sebelum dikonversi menjadi TinyML. Pengujian tidak langsung dilakukan dengan menggunakan program

Python yang terdapat pada Lampiran 2 yang dijalankan pada *software* Visual Studio Code. Hasil yang diperoleh dari program untuk pengujian tidak langsung berupa *Confussion Matrix* dan laporan metrik kinerja program ML. Pengujian tidak langsung dilakukan untuk mengetahui kualitas kinerja awal dari model ML sebelum dikonversi menjadi model TinyML.

6. Pengujian langsung model TinyML

Pengujian langsung model TinyML dilakukan dengan menggunakan perangkat MCU ESP32 dan *software* Arduino IDE serta Visual Studio Code. Pengujian langsung menggunakan dua jenis program, yaitu program Arduino dan program Python. Program Arduino merupakan program untuk digunakan perangkat MCU dalam menerima data dan menganalisis data tersebut dengan menggunakan model TinyML yang telah disusun. Program Arduino disusun, dikompilasi, dan diunggah dengan menggunakan *software* Arduino IDE. Program lengkap Arduino terdapat pada Lampiran 4. Program Python merupakan program untuk mengirim dataset pengujian ke perangkat MCU dan juga menerima data hasil pengujian dari perangkat MCU. Program Python disusun dan dijalankan di *software* Visual Studio Code. Program lengkap Python terdapat di Lampiran 3.

Pelaksanaan pengujian langsung dimulai dengan mengompilasi dan mengunggah program Arduino beserta model TinyML ke perangkat MCU. Sesudah proses kompilasi dan pengunggahan selesai, program Python dapat dijalankan untuk mengirim dataset pengujian dan menerima hasil pengujian. Setiap dataset pengujian dikirim satu per satu. Dalam setiap dataset, terdapat 1024 titik cacah. Program Python akan mengirim dan menerima satu per satu data titik cacah tersebut. Alur pelaksanaan pengujian langsung terdapat pada gambar 3.11.



Gambar 3.11 Flowchart program a) Python dan b) Arduino untuk pengujian langsung model TinyML

7. Analisis data

Data dari hasil pengujian tidak langsung dan langsung berupa perbandingan antara jumlah titik cacah yang diprediksi dengan jumlah titik cacah yang sebenarnya. Perbandingan tersebut dibentuk sebagai *Confusion matrix*. Dari *Confusion matrix*

tersebut akan dianalisis kualitas kinerja model TinyML menggunakan metrik kinerja model TinyML yang terdapat pada bab 2, subbab 2.2.2. Metrik kinerja terdiri dari Akurasi, Presisi, *Recall*, dan *F1-Score*. Hasil metrik dari setiap model dibandingkan untuk menentukan pengaruh variasi kriteria model TinyML dengan kualitas kinerja dalam memprediksi jenis radionuklida dengan tepat.

8. Penyusunan laporan tugas akhir.

Seluruh data dan hasil yang diperoleh dari tahapan awal tugas akhir hingga hasil pengujian program model TinyML didokumentasikan, dianalisis, dijelaskan, dan ditulis dalam laporan Tugas Akhir.

BAB 4 HASIL DAN PEMBAHASAN

4.1 Kalibrasi Energi, Efisiensi, dan Resolusi Spektroskopi Gamma

Kalibrasi energi menggunakan sumber radiasi dengan dua puncak energi, yaitu Co-60 kode 11 yang dicacah selama 60 detik. Nomor *channel* dan energi yang diatur dan diterapkan pada kalibrasi energi terdapat pada tabel 4.1

Tabel 4.1 Titik kalibrasi energi

Nomor <i>channel</i>	Energi
492	1173
554	1332

Dari kedua nomor *channel*-energi tersebut dilakukan perhitungan kofisien kalibrasi dengan menggunakan persamaan 2.8 sebagai berikut.

$$E = A \cdot Ch + B$$

$$1173 = A (492) + B$$

$$1332 = A (554) + B$$

$$A = 2,565$$

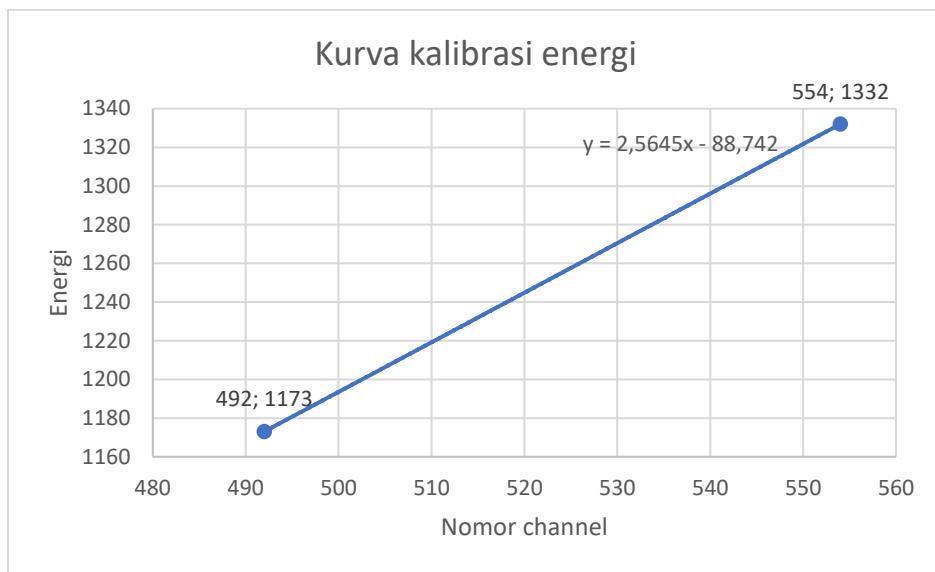
$$B = -88,74$$

Diperoleh persamaan kalibrasi energi sebagai berikut

$$E = 2,565 \cdot (Ch) - 88,74$$

Hasil kalibrasi energi dalam bentuk kurva kalibrasi terdapat pada gambar 4.1. Hasil dari kalibrasi energi dapat diketahui dari dataset csv yang tersimpan dari *software* UCS30 berikut.

```
Calibration Coefficients:,-88.74,2.565
Channels used to calibrate:,492,554
Energies used to calibrate:,1173,1332
```



Gambar 4.1 Kurva kalibrasi energi

Kalibrasi efisiensi menggunakan semua radionuklida pada tabel 3.3. Dengan perhitungan menggunakan persamaan 2.9, 2.10, dan 2.11, hasil dari perhitungan kalibrasi efisiensi terdapat pada tabel 4.2.

Hasil dari perhitungan efisiensi menunjukkan bahwa tingkat efisiensi dari sistem spektroskopi sebesar 0,03% hingga 0,14%. Berarti hanya sebagian kecil dari radiasi gamma radionuklida yang terdeteksi dan terukur oleh sistem spektroskopi gamma. Efisiensi tertinggi saat melakukan pengukuran radionuklida Cs-137 yang berenergi rendah dan efisiensi terendah saat mengukur spektroskopi gamma dari radionuklida Co-60 berenergi tinggi.

Penentuan resolusi spektroskopi gamma menggunakan semua radionuklida pada tabel 3.3. Dengan perhitungan menggunakan persamaan 2.12, hasil resolusi dari spektroskopi gamma terdapat pada tabel 4.3.

Hasil perhitungan resolusi spektroskopi gamma menunjukkan nilai resolusi yang kecil, berkisar sekitar 3% hingga 6%. Nilai resolusi tersebut menunjukkan bahwa sistem spektroskopi gamma mampu membedakan energi gamma cukup baik. Nilai resolusi yang terendah saat mengukur spektroskopi gamma dari radionuklida Co-60

dikarenakan tingginya energi gamma puncak dibandingkan dengan radionuklida Cs-137.

Tabel 4.2 Hasil perhitungan kalibrasi efisiensi

Jari-jari detektor (cm)			2,25	Waktu cacah (s)	60
Jarak sumber-detektor (cm)			1		
Faktor geometri			3,73		
Radionuklida	Aktivitas terakhir terhitung (dps)	Energi puncak	Net area	Fraksi pencacahan	Efisiensi
Co-60 kode 10	6189,61	1178	859	0,9985	0,0621%
	6189,61	1321,9	492	0,9998	0,0355%
Co-60 kode 11	13340,08	1173	1530	0,9985	0,0513%
	13340,08	1332	1073	0,9998	0,0360%
Co-60 kode 12	13263,39	1173	1271	0,9985	0,0429%
	13263,39	1332	1088	0,9998	0,0367%
Cs-137 kode 16	6761,57	675,8	1360	0,85	0,1056%
Cs-137 kode 19	27046,27	675,8	7071	0,85	0,1373%
Cs-137 kode 21	134575,32	670,8	25005	0,85	0,0976%

Tabel 4.3 Resolusi spektroskopi gamma

Radionuklida	energi	FWHM	resolusi
Co-60 kode 10	1178	48	4,07 %
	1321,9	45,4	3,43 %
Co-60 kode 11	1173	40,4	3,44 %
	1332	48	3,60 %
Co-60 kode 12	1173	35,3	3,00 %
	1332	48	3,60 %

Radionuklida	energi	FWHM	resolusi
Cs-137 kode 16	675,8	40,4	5,97 %
Cs-137 kode 19	675,8	40,4	5,97 %
Cs-137 kode 21	670,8	37,9	5,65 %

4.2 Hasil Pengujian Tidak Langsung dan Langsung Model TinyML

Hasil pengujian tidak langsung ditampilkan dengan menjalankan program Python di Visual Studio Code. Program tersebut terdapat pada Lampiran 2. Hasil pengujian langsung berupa jumlah kelas 0 dan kelas 1 yang teridentifikasi oleh program Arduino. *Confusion matrix* dan metrik kinerja dikumpulkan dan dihitung dengan sarana *software* Microsoft Excel dengan menggunakan persamaan 2.1, 2.2, 2.3, dan 2.4. Kedua pengujian tersebut menggunakan dataset pelatihan yang terdiri dari 8192 titik cacah dan terbagi rata di kedua kelas. Pengujian tidak langsung untuk mengetahui kinerja model ML dalam mengidentifikasi kelas radionuklida dari dataset pengujian sebelum dikonversi menjadi model TinyML dan dijalankan di perangkat MCU. Pengujian langsung untuk mengetahui kinerja model TinyML yang dijalankan di perangkat MCU.

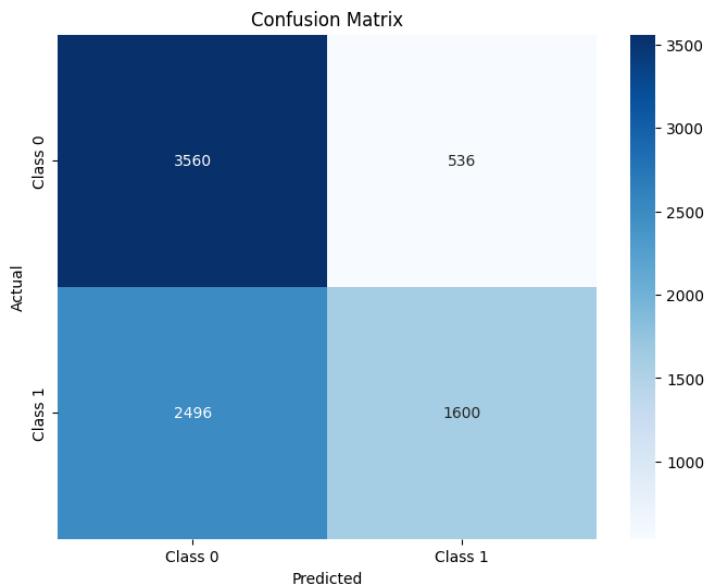
1. Model A.1

Model A.1 merupakan model untuk tipe data terkalibrasi, dengan data input berupa data *channel*, energi, dan *counts*. Susunan *layers*, *nodes*, dan fungsi aktivasi terdapat pada tabel 4.4.

Tabel 4.4 Susunan model A.1

Layer	Jumlah nodes	Fungsi aktivasi
1	10	relu
2	5	relu
3	1	sigmoid

Hasil pengujian tidak langsung model A.1 dalam bentuk *Confusion matrix* terdapat pada gambar 4.2. Hasil pengujian langsung model A.1 dalam bentuk *Confusion matrix* terdapat pada tabel 4.5.



Gambar 4.2 *Confusion matrix* pengujian tidak langsung model A.1

Tabel 4.5 *Confusion matrix* pengujian langsung model A.1

Dataset	Aktual \ Prediksi	Co-60	Cs-137
Dataset 60 detik	Co60	943	82
	Cs137	583	442
Dataset 70 detik	Co60	914	111
	Cs137	615	410
Dataset 80 detik	Co60	874	151
	Cs137	640	385
Dataset 90 detik	Co60	833	192
	Cs137	662	363
Total	Co60	3564	536
	Cs137	2500	1600

Berdasarkan hasil kedua *Confusion matrix* tersebut, jumlah salah satu kondisi yang eror lebih banyak daripada jumlah salah satu kondisi yang tepat. Model ini cenderung mengidentifikasi sebagian besar data yang diinput sebagai kelas 0 yang dapat dilihat dari jumlah nilai yang terprediksi sebagai kelas 0 jauh lebih banyak dibandingkan jumlah yang terprediksi sebagai kelas 1. Model ini juga memiliki selisih dua kondisi ketepatan yang cukup tinggi. Model ini tidak akurat dan tidak seimbang dalam mengidentifikasi kedua kelas radionuklida.

Dari *Confusion matrix* pengujian tidak langsung, metrik kinerja model pada pengujian tidak langsung pada tabel 4.6. Dari *Confusion matrix* pengujian langsung, metrik kinerja model pada pengujian langsung terdapat pada tabel 4.7.

Tabel 4.6 Metrik kinerja pengujian tidak langsung model A.1

Metriks kinerja model	Nilai	
Akurasi	0,63	
Presisi	Kelas 0	0,59
	Kelas 1	0,75
<i>Recall</i>	Kelas 0	0,87
	Kelas 1	0,39
<i>F1-Score</i>	Kelas 0	0,70
	Kelas 1	0,51

Berdasarkan pada metrik kinerja pengujian tidak langsung pada tabel 4.6 menunjukkan bahwa model A.1 tidak efektif dalam memprediksi data cacah kelas 1 dan memiliki bias yang kuat terhadap data cacah kelas 0. Dalam memprediksi kelas 0, model ini memiliki nilai *recall* yang tinggi, sebesar 0,87 namun dengan nilai presisi yang rendah, sebesar 0,59. Berarti masih terdapat jumlah eror False Positive yang besar. Kedua nilai pada kelas 0 tersebut menghasilkan nilai F1-Score sebesar 0,70. Dalam memprediksi kelas 1, model ini memiliki nilai *recall* yang sangat rendah, sebesar 0,39, dengan nilai presisi yang tinggi, sebesar 0,75. Berarti terdapat jumlah eror False

Negative yang besar. Dari nilai *recall* dan presisi tersebut menghasilkan nilai *F1-Score* yang rendah, sebesar 0,51.

Nilai *recall* pada kelas 0 cenderung lebih tinggi dibandingkan nilai *recall* pada kelas 1. Nilai presisi pada kelas 0 cenderung lebih rendah dibandingkan nilai presisi pada kelas 1. Nilai *F1-Score* pada kelas 0 cenderung lebih tinggi dibandingkan dengan nilai *F1-Score* pada kelas 1. Kecenderungan tersebut menunjukkan bahwa model A.1 tidak seimbang sehingga tidak efektif untuk digunakan untuk identifikasi radionuklida pemancar radiasi gamma.

Tabel 4.7 Metrik kinerja pengujian langsung model A.1

Kelompok Dataset Waktu Cacah	Kelas Positif	Akurasi	Recall	Presisi	F1-Score
60 detik	0	0,68	0,92	0,62	0,74
	1		0,43	0,84	0,57
70 detik	0	0,65	0,89	0,60	0,72
	1		0,40	0,79	0,53
80 detik	0	0,61	0,85	0,58	0,69
	1		0,38	0,72	0,49
90 detik	0	0,58	0,81	0,56	0,66
	1		0,35	0,65	0,46
Total	0	0,63	0,87	0,59	0,70
	1		0,39	0,75	0,51

Berdasarkan data pada tabel, hasil evaluasi model klasifikasi menunjukkan kinerja yang bervariasi antar kelas dan kelompok dataset. Secara keseluruhan, model mencapai akurasi total sebesar 0,63. Terdapat perbedaan performa yang signifikan antara Kelas 0 dan Kelas 1. Untuk Kelas 0, model menunjukkan nilai *recall* yang tinggi (0,87) dengan presisi yang lebih rendah (0,59), menghasilkan *F1-Score* sebesar 0,70. Hal ini mengindikasikan bahwa model mampu mengidentifikasi sebagian besar data

cacah yang tergolong kelas 0 dengan benar, namun rentan terhadap *false positive*. Sebaliknya, Kelas 1 menunjukkan *recall* yang sangat rendah (0,39) namun dengan presisi yang lebih tinggi (0,75), dan *F1-Score* yang rendah (0,51). Kondisi ini menandakan bahwa model kesulitan mengidentifikasi data cacah yang tergolong Kelas 1.

Analisis pada setiap kelompok dataset menunjukkan tren penurunan performa seiring dengan bertambahnya durasi waktu cacah pada dataset spektroskopi gamma dari 60 detik menjadi 90 detik. Akurasi tertinggi (0,68) dan *F1-Score* terbaik untuk kedua kelas (0,74 untuk Kelas 0 dan 0,57 untuk Kelas 1) dicapai pada kelompok dataset 60 detik. Seiring bertambahnya nilai durasi waktu cacah, semua metrik evaluasi—akurasi, *recall*, presisi, dan *F1-Score*—secara konsisten menurun, dengan performa terendah tercatat pada kelompok dataset 90 detik yang memiliki akurasi 0,58. Tren ini menyiratkan bahwa segmen data dengan durasi yang lebih pendek memberikan fitur yang lebih relevan dan diskriminatif bagi model klasifikasi.

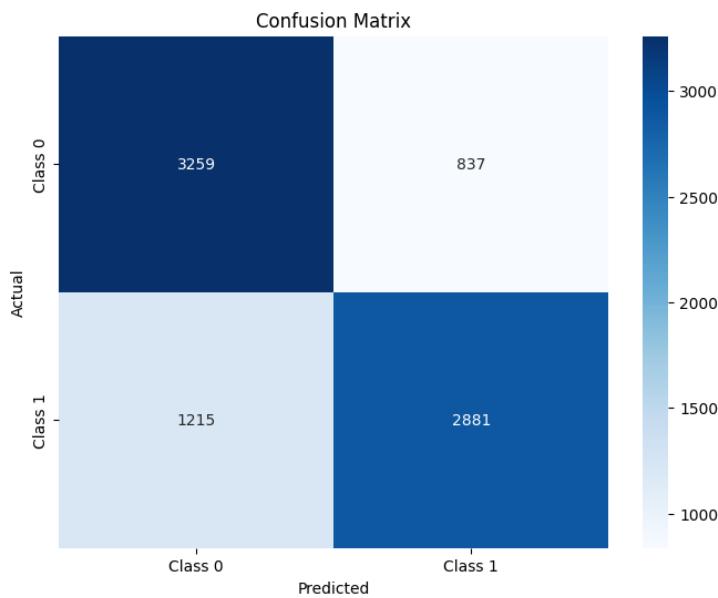
2. Model A.2

Model A.2 merupakan model untuk tipe data terkalibrasi, dengan data input berupa data *channel*, energi, dan *counts*. Susunan *layers*, *nodes*, dan fungsi aktivasi model A.2 terdapat pada tabel 4.8.

Tabel 4.8 Susunan model A.2

Layer	Jumlah nodes	Fungsi aktivasi
1	20	relu
2	10	relu
3	1	sigmoid

Hasil pengujian tidak langsung model A.2 dalam bentuk *Confusion matrix* terdapat pada gambar 4.3. Hasil pengujian langsung model A.2 dalam bentuk *Confusion matrix* terdapat pada tabel 4.9.



Gambar 4.3 *Confusion matrix* pengujian tidak langsung model A.2

Tabel 4.9 *Confusion matrix* pengujian langsung model A.2

Kelompok Dataset Waktu Cacah	Prediksi Aktual	Co-60	Cs-137
Dataset 60 detik	Co60	847	178
	Cs137	310	715
Dataset 70 detik	Co60	830	195
	Cs137	311	714
Dataset 80 detik	Co60	789	236
	Cs137	299	726
Dataset 90 detik	Co60	793	232
	Cs137	295	730
Total	Co60	3259	841
	Cs137	1215	2885

Dari kedua *Confusion matrix* tersebut, model ini dapat mengidentifikasi kedua kelas dengan tepat dilihat dari lebih banyaknya kondisi yang tepat dibandingkan

dengan kondisi yang eror. Selisih antara kedua kondisi yang tepat cukup tinggi sehingga model tidak seimbang

Dari *Confusion matrix* pengujian tidak langsung, metrik kinerja model pada pengujian tidak langsung terdapat pada tabel 4.10. Dari *Confusion matrix* pengujian langsung, metrik kinerja model pada pengujian langsung terdapat pada tabel 4.11..

Tabel 4.10 Metrik kinerja pengujian tidak langsung model A.2

Metriks kinerja model	Nilai	
Akurasi	0,75	
Presisi	Kelas 0	0,73
	Kelas 1	0,77
<i>Recall</i>	Kelas 0	0,80
	Kelas 1	0,70
<i>F1-Score</i>	Kelas 0	0,76
	Kelas 1	0,74

Hasil evaluasi model berdasarkan pada tabel 4.10 menunjukkan kinerja yang baik dan cukup seimbang dengan akurasi keseluruhan mencapai 0,75. Pada Kelas 0, model mencapai *F1-Score* sebesar 0,76, yang didukung oleh nilai presisi 0,73 dan *recall* 0,80. Nilai *recall* yang relatif tinggi (0,80) menandakan bahwa model sangat andal dalam mengidentifikasi sebagian besar data cacah aktual dari Kelas 0. Pada Kelas 1, performanya juga sangat kompetitif dengan *F1-Score* sebesar 0,74. Kelas ini mencatatkan nilai presisi yang sedikit lebih tinggi (0,77), yang berarti prediksi positif untuk Kelas 1 memiliki tingkat kebenaran yang sangat baik, meskipun dengan nilai *recall* yang lebih rendah (0,70). Secara keseluruhan, nilai *F1-Score* yang hampir setara antara Kelas 0 (0,76) dan Kelas 1 (0,74) mengonfirmasi bahwa model ini memiliki performa yang baik dan seimbang.

Tabel 4.11 Metrik kinerja pengujian langsung model A.2

Kelompok Dataset Waktu Cacah	Kelas Positif	Akurasi	Recall	Presisi	F1-Score
60 detik	0	0,76	0,83	0,73	0,78
	1		0,70	0,80	0,75
70 detik	0	0,75	0,81	0,73	0,77
	1		0,70	0,79	0,74
80 detik	0	0,74	0,77	0,73	0,75
	1		0,71	0,75	0,73
90 detik	0	0,74	0,77	0,73	0,75
	1		0,71	0,76	0,73
Total	0	0,75	0,79	0,73	0,76
	1		0,70	0,77	0,74

Berdasarkan tabel 4.11, model klasifikasi menunjukkan performa yang baik dan seimbang di seluruh kelompok dataset. Secara total, model mencapai akurasi keseluruhan sebesar 0,75, dengan *F1-Score* yang baik untuk Kelas 0 (0,76) dan Kelas 1 (0,74). Keseimbangan ini menunjukkan bahwa model tidak memiliki bias yang signifikan terhadap salah satu kelas. Secara spesifik, model menunjukkan *recall* yang sedikit lebih tinggi untuk Kelas 0 (0,79) dan presisi yang sedikit lebih tinggi untuk Kelas 1 (0,77).

Analisis terhadap setiap kelompok waktu cacah dataset menunjukkan bahwa performa optimal dicapai pada dataset dengan durasi 60 detik. Pada segmen ini, model mencatatkan akurasi tertinggi (0,76) dan *F1-Score* puncak untuk Kelas 0 (0,78) dan Kelas 1 (0,75). Seiring dengan bertambahnya durasi waktu dari 60 detik menjadi 90 detik, teramatid adanya tren penurunan kinerja yang bersifat minor namun konsisten. Penurunan ini terutama terlihat pada *recall* Kelas 0 yang menurun dari 0,83 menjadi

0,77, yang kemudian berdampak pada penurunan *F1-Score*-nya dari 0,78 menjadi 0,75. Sementara itu, kinerja untuk Kelas 1 cenderung lebih stabil di semua kelompok dataset.

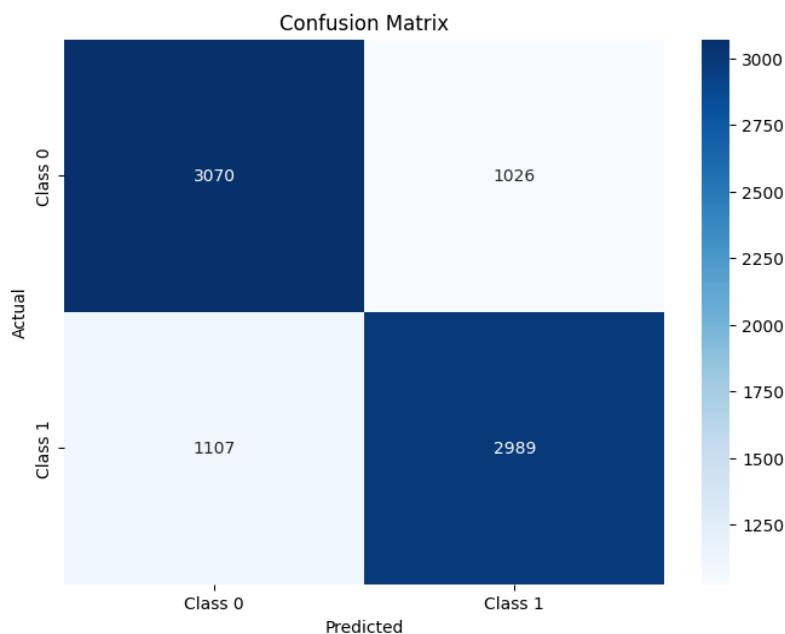
3. Model A.3

Model A.3 merupakan model untuk tipe data terkalibrasi, dengan data input berupa data *channel*, energi, dan *counts*. Susunan *layers*, *nodes*, dan fungsi aktivasi terdapat pada tabel 4.12.

Tabel 4.12 Susunan model A.3

<i>Layer</i>	Jumlah <i>nodes</i>	Fungsi aktivasi
1	30	relu
2	15	relu
3	1	sigmoid

Hasil pengujian tidak langsung model A.3 dalam bentuk *Confusion matrix* terdapat pada gambar 4.4. Hasil pengujian langsung model A.3 dalam bentuk *Confusion matrix* terdapat pada tabel 4.12.



Gambar 4.4 *Confusion matrix* pengujian tidak langsung model A.3

Tabel 4.13 *Confusion matrix* pengujian langsung model A.3

Kelompok Dataset Waktu Cacah	Aktual	Prediksi	Co-60	Cs-137
Dataset 60 detik	Co60	773	252	
	Cs137	261	764	
Dataset 70 detik	Co60	780	245	
	Cs137	273	752	
Dataset 80 detik	Co60	757	268	
	Cs137	283	742	
Dataset 90 detik	Co60	760	265	
	Cs137	290	735	
Total	Co60	3070	1030	
	Cs137	1107	2993	

Dari *Confusion matrix* pengujian tidak langsung, metrik kinerja model pada pengujian tidak langsung terdapat pada tabel 4.14. Dari *Confusion matrix* pengujian langsung, metrik kinerja model pada pengujian langsung terdapat pada tabel 4.15.

Tabel 4.14 Metrik kinerja pengujian tidak langsung model A.3

Metriks kinerja model	Nilai	
Akurasi	0,74	
Presisi	Kelas 0	0,73
	Kelas 1	0,74
<i>Recall</i>	Kelas 0	0,75
	Kelas 1	0,73
<i>F1-Score</i>	Kelas 0	0,74
	Kelas 1	0,74

Dari kedua *Confusion matrix* tersebut, model ini dapat mengidentifikasi kedua kelas dengan tepat dilihat dari lebih banyaknya nilai kondisi ketepatan TP dan TN dibandingkan dengan nilai kondisi eror FP dan FN.

Metrik kinerja pada tabel 4.14 menunjukkan kinerja yang baik dan seimbang, dengan akurasi keseluruhan mencapai 0,74. Keunggulan utama dari model ini adalah kemampuannya untuk melakukan klasifikasi secara merata tanpa menunjukkan bias terhadap kelas manapun, yang dibuktikan dengan metrik performa yang hampir identik untuk kedua kelas. Secara spesifik, model ini memiliki nilai *F1-Score* yang sama persis, yaitu 0,74, untuk Kelas 0 dan Kelas 1. Nilai *F1-Score* yang identik ini menandakan adanya keseimbangan sempurna antara presisi dan *recall* untuk kedua kelas, yang merupakan indikator dari model yang baik dan efektif. Untuk Kelas 0, model mencatatkan presisi 0,73 dan *recall* 0,75. Sementara itu, untuk Kelas 1, nilainya sangat berdekatan, dengan presisi 0,74 dan *recall* 0,73. Angka-angka yang sangat konsisten ini menegaskan bahwa model tidak hanya akurat secara umum, tetapi juga dapat diandalkan untuk mengidentifikasi dan memprediksi setiap kelas.

Tabel 4.15 Metrik kinerja pengujian langsung model A.3

Kelompok Dataset Waktu Cacah	Kelas Positif	Akurasi	Recall	Presisi	F1-Score
60 detik	0	0,75	0,75	0,75	0,75
	1		0,75	0,75	0,75
70 detik	0	0,75	0,76	0,74	0,75
	1		0,73	0,75	0,74
80 detik	0	0,73	0,74	0,73	0,73
	1		0,72	0,73	0,73
90 detik	0	0,73	0,74	0,72	0,73
	1		0,72	0,74	0,73
Total	0	0,74	0,75	0,73	0,74
	1		0,73	0,74	0,74

Berdasarkan tabel 4.15, model menunjukkan kinerja yang sangat baik dan seimbang, yang dibuktikan oleh nilai *F1-Score* total yang identik untuk Kelas 0 dan Kelas 1, yaitu 0,74. Analisis lebih lanjut pada setiap kelompok dataset mengungkap adanya tren di mana kinerja model menurun seiring bertambahnya durasi segmen data. Puncak performa terjadi pada kelompok 60 detik, di mana model mencapai kondisi keseimbangan statistik yang sempurna dengan semua metrik—akurasi, *recall*, presisi, dan *F1-Score*—secara seragam bernilai 0,75 untuk kedua kelas. Seiring bertambahnya durasi ke 70 detik, kinerja tetap tinggi namun mulai menunjukkan sedikit penurunan, yang kemudian menjadi lebih jelas pada durasi 80 dan 90 detik di mana akurasi dan *F1-Score* stabil di level yang lebih rendah, yaitu 0,73. Tren ini secara kuat mengindikasikan bahwa segmen data dengan durasi yang lebih pendek menyediakan fitur yang paling relevan dan efektif untuk klasifikasi oleh model ini.

4. Model A.4

Model A.4 merupakan model untuk tipe data terkalibrasi, dengan data input berupa data *channel*, energi, dan *counts*. Susunan *layers*, *nodes*, dan fungsi aktivasi terdapat pada tabel 4.16.

Tabel 4.16 Susunan model A.4

<i>Layer</i>	Jumlah <i>nodes</i>	Fungsi aktivasi
1	10	relu
2	5	relu
3	5	relu
4	1	sigmoid

Hasil pengujian tidak langsung model A.4 dalam bentuk *Confusion matrix* terdapat pada gambar 4.5. Hasil pengujian langsung model A.4 dalam bentuk *Confusion matrix* terdapat pada tabel 4.17



Gambar 4.5 *Confusion matrix* pengujian tidak langsung model A.4

Tabel 4.17 *Confusion matrix* pengujian langsung model A.4

Kelompok Dataset Waktu Cacah	Prediksi	Co-60	Cs-137
	Aktual		
Dataset 60 detik	Co60	747	278
	Cs137	243	782
Dataset 70 detik	Co60	752	273
	Cs137	237	788
Dataset 80 detik	Co60	747	278
	Cs137	225	800
Dataset 90 detik	Co60	752	273
	Cs137	221	804
Total	Co60	2998	1102
	Cs137	926	3174

Dari *Confusion matrix* pengujian tidak langsung, metrik kinerja model pada pengujian tidak langsung terdapat pada tabel 4.18. Dari *Confusion matrix* pengujian langsung, metrik kinerja model pada pengujian langsung terdapat pada tabel 4.19.

Tabel 4.18 Metrik kinerja pengujian tidak langsung model A.4

Metriks kinerja model	Nilai	
Akurasi	0,75	
Presisi	Kelas 0	0,76
	Kelas 1	0,74
<i>Recall</i>	Kelas 0	0,73
	Kelas 1	0,77
<i>F1-Score</i>	Kelas 0	0,75
	Kelas 1	0,76

Tabel 4.19 Metrik kinerja pengujian langsung model A.4

Kelompok Dataset	Kelas	Akurasi	Recall	Presisi	F1-Score
Waktu Cacah	Positif				
60 detik	0	0.75	0.73	0.75	0.74
	1		0.76	0.74	0.75
70 detik	0	0.75	0.73	0.76	0.75
	1		0.77	0.74	0.76
80 detik	0	0.75	0.73	0.77	0.75
	1		0.78	0.74	0.76
90 detik	0	0.76	0.73	0.77	0.75
	1		0.78	0.75	0.76
Total	0	0.75	0.73	0.76	0.75
	1		0.77	0.74	0.76

Dari kedua *Confusion matrix* tersebut, model A.4 dapat mengidentifikasi kedua kelas radionuklida dengan tepat kedua kelas dan jumlah eror FP dan FN yang lebih

sedikit. Model ini kurang seimbang karena terdapat ketidaksamaan jumlah kedua kelas yang tepat teridentifikasi cukup tinggi dengan selisih bernilai 172.

Berdasarkan tabel 4.18, model klasifikasi menunjukkan kinerja yang efektif dan seimbang dengan akurasi keseluruhan sebesar 0,75. Analisis yang lebih terperinci pada setiap kelas mengonfirmasi bahwa model ini bekerja secara konsisten tanpa bias yang signifikan, seperti yang ditunjukkan oleh nilai *F1-Score* yang hampir identik untuk Kelas 0 (0,75) dan Kelas 1 (0,76).

Untuk Kelas 0, model mencapai presisi yang lebih tinggi (0,76) namun dengan *recall* yang sedikit lebih rendah (0,73). Hal ini mengindikasikan bahwa ketika model memprediksi Kelas 0, prediksinya cenderung sangat andal, meskipun ada beberapa data cacah aktual yang tidak teridentifikasi. Sebaliknya, untuk Kelas 1, model menunjukkan *recall* yang lebih tinggi (0,77) dengan presisi yang sedikit lebih rendah (0,74). Ini berarti model sangat cakap dalam mengidentifikasi sebagian besar instans aktual dari Kelas 1, menjadikannya sangat sensitif terhadap kelas ini. Secara keseluruhan, keseimbangan antara metrik-metrik ini menghasilkan *F1-Score* yang kuat dan sebanding untuk kedua kelas.

Berdasarkan tabel 4.19, model klasifikasi secara keseluruhan menunjukkan kinerja yang sangat baik dan seimbang, di seluruh kelompok dataset. Analisis pada baris "Total" menyimpulkan performa model yang menghasilkan *F1-Score* 0,75 untuk Kelas 0 dan 0,76 untuk Kelas 1. Nilai yang sangat berdekatan ini menegaskan bahwa model tidak memiliki bias signifikan dan mampu menangani kedua kelas dengan efektivitas yang hampir setara. Terdapat pertukaran (trade-off) yang jelas antara kedua kelas: Kelas 0 unggul dalam presisi (0,76), menunjukkan keandalan prediksi yang tinggi, sementara Kelas 1 unggul dalam *recall* (0,77), yang menandakan kemampuannya untuk mengidentifikasi sebagian besar instans positif dengan benar.

Analisis lebih lanjut pada setiap Kelompok Dataset menunjukkan tren yang menarik terkait pengaruh durasi data terhadap kinerja model. Berbeda dengan ekspektasi umum di mana penambahan data dapat menurunkan performa, model ini

menunjukkan stabilitas dan bahkan sedikit peningkatan kinerja seiring bertambahnya durasi dari 60 menjadi 90 detik. Akurasi model sangat konsisten, berada di angka 0,75 untuk tiga kelompok pertama dan sedikit meningkat menjadi 0,76 pada kelompok 90 detik.

Secara spesifik, metrik untuk kedua kelas cenderung membaik atau tetap stabil. Untuk Kelas 1, nilai *recall* secara bertahap meningkat dari 0,76 menjadi 0,78. Sementara itu, untuk Kelas 0, nilai presisi juga meningkat dari 0,75 menjadi 0,77. Hal ini mengindikasikan bahwa model mampu memanfaatkan informasi tambahan dari segmen data yang lebih panjang untuk menyempurnakan prediksinya. Kinerja *F1-Score* untuk kedua kelas mencapai puncaknya (0,75 untuk Kelas 0 dan 0,76 untuk Kelas 1) pada durasi 70 detik dan mampu mempertahankan level tersebut hingga durasi 90 detik. Kesimpulannya, model ini tidak hanya robust tetapi juga mampu beradaptasi dengan baik terhadap variasi panjang data, dengan kinerja optimal yang konsisten pada durasi 70 detik ke atas.

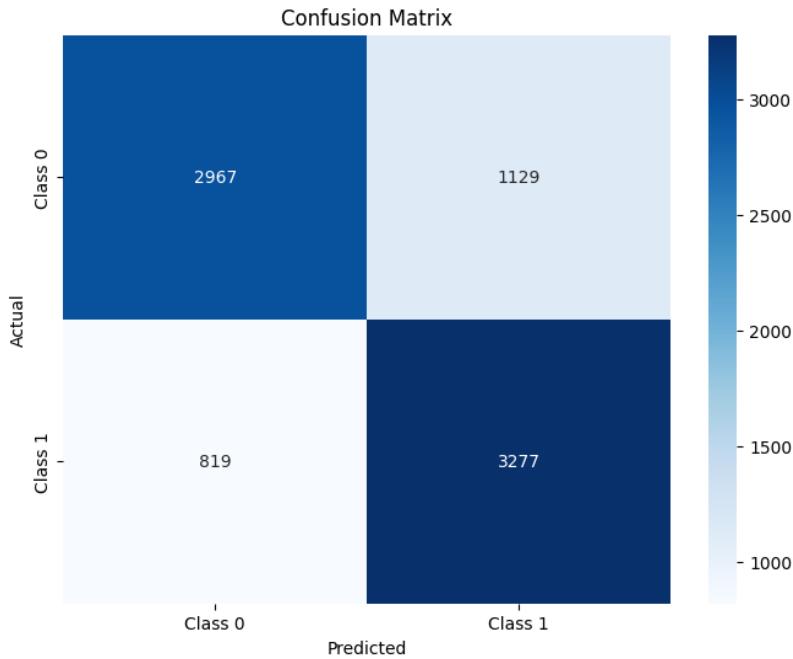
5. Model A.5

Model A.5 merupakan model untuk tipe data terkalibrasi, dengan data input berupa data *channel*, energi, dan *counts*. susunan *layers*, *nodes*, dan fungsi aktivasi terdapat pada tabel 4.20.

Tabel 4.20 Susunan model A.5

<i>Layer</i>	<i>Jumlah nodes</i>	<i>Fungsi aktivasi</i>
1	20	relu
2	10	relu
3	5	relu
4	1	sigmoid

Hasil pengujian tidak langsung model A.5 dalam bentuk *Confusion matrix* terdapat pada gambar 4.6. Hasil pengujian langsung model A.5 dalam bentuk *Confusion matrix* terdapat pada tabel 4.21.



Gambar 4.6 *Confusion matrix* pengujian tidak langsung model A.5

Tabel 4.21 *Confusion matrix* pengujian langsung model A.5

Kelompok Dataset Waktu Cacah	Prediksi Aktual	Co-60	Cs-137
Dataset 60 detik	Co60	766	259
	Cs137	195	830
Dataset 70 detik	Co60	759	266
	Cs137	196	829
Dataset 80 detik	Co60	727	298
	Cs137	209	816
Dataset 90 detik	Co60	715	310
	Cs137	219	806
Total	Co60	2967	1133
	Cs137	819	3281

Dari *Confusion matrix* pengujian tidak langsung, metrik kinerja model pada pengujian tidak langsung terdapat pada tabel 4.22. Dari *Confusion matrix* pengujian langsung, metrik kinerja model pada pengujian langsung terdapat pada tabel 4.23.

Tabel 4.22 Metrik kinerja pengujian tidak langsung model A.5

Metriks kinerja model	Nilai	
Akurasi	0,75	
Presisi	Kelas 0	0,78
	Kelas 1	0,74
<i>Recall</i>	Kelas 0	0,72
	Kelas 1	0,80
<i>F1-Score</i>	Kelas 0	0,75
	Kelas 1	0,77

Tabel 4.23 Metrik kinerja pengujian langsung model A.5

Kelompok Dataset Waktu Cacah	Kelas Positif	Akurasi	Recall	Presisi	F1-Score
60 detik	0	0.78	0.75	0.80	0.77
	1		0.81	0.76	0.79
70 detik	0	0.77	0.74	0.79	0.77
	1		0.81	0.76	0.78
80 detik	0	0.75	0.71	0.78	0.74
	1		0.80	0.73	0.76
90 detik	0	0.74	0.70	0.77	0.73
	1		0.79	0.72	0.75
Total	0	0.76	0.72	0.78	0.75
	1		0.80	0.74	0.77

Berdasarkan pada *Confusion matrix* pengujian tidak langsung dan pengujian langsung, model A.5 dapat mengidentifikasi sebagian besar titik cacah pada kedua kelas dengan baik sebagai TP dan TN. Jumlah eror FP dan FN lebih sedikit dibandingkan jumlah TP dan TN. Perbandingan jumlah kelas TP dan TN berbeda dengan total selisih sekitar 314. Berarti model ini kurang seimbang dalam mengidentifikasi kedua kelas.

Berdasarkan tabel 4.22, model klasifikasi menunjukkan kinerja yang sangat efektif dengan akurasi keseluruhan sebesar 0,75. Analisis pada metrik per kelas menunjukkan bahwa model ini memiliki performa yang cukup seimbang, yang divalidasi oleh nilai *F1-Score* yang tinggi dan hampir setara untuk Kelas 0 (0,75) dan Kelas 1 (0,77). Keseimbangan ini menandakan bahwa model tidak memiliki bias yang signifikan dan mampu menangani kedua kelas secara kompeten.

Model ini menampilkan karakteristik pertukaran (trade-off) yang jelas antara presisi dan *recall*. Untuk Kelas 0, model mencapai presisi yang sangat tinggi (0,78) dengan *recall* yang lebih rendah (0,72). Hal ini mengindikasikan bahwa ketika model membuat prediksi untuk Kelas 0, prediksinya memiliki tingkat keandalan yang tinggi, meskipun model ini cenderung melewatkannya beberapa data cacah aktual dari kelas tersebut. Pada Kelas 1, model menunjukkan kekuatan dalam hal *recall* yang mencapai 0,80, yang berarti model sangat andal dalam mengidentifikasi sebagian besar instans aktual dari Kelas 1. Performa *recall* yang tinggi ini dicapai dengan presisi yang sedikit lebih rendah (0,74).

Secara keseluruhan, meskipun kedua kelas menunjukkan profil kinerja yang berbeda, keseimbangan yang dicapai antara presisi dan *recall* menghasilkan *F1-Score* yang kuat untuk keduanya.

Berdasarkan hasil tabel 4.23, model klasifikasi secara keseluruhan menunjukkan kinerja yang kuat dan seimbang dengan akurasi total mencapai 0,76. Analisis pada baris "Total" menunjukkan bahwa model ini memiliki *F1-Score* yang tinggi dan kompetitif untuk kedua kelas, yaitu 0,75 untuk Kelas 0 dan 0,77 untuk Kelas 1. Keseimbangan ini

menandakan bahwa model mampu menangani kedua kelas secara efektif. Terdapat pertukaran (trade-off) yang jelas antara kedua kelas: Kelas 0 unggul dalam presisi (0,78), yang berarti prediksinya sangat andal, sedangkan Kelas 1 unggul dalam *recall* (0,80), menunjukkan kemampuannya yang superior dalam mendeteksi sebagian besar instans positif.

Analisis lebih lanjut pada setiap Kelompok Dataset mengungkap kinerja model secara konsisten menurun seiring dengan bertambahnya durasi segmen data. Performa puncak secara absolut dicapai pada kelompok dataset 60 detik. Pada durasi ini, model mencatatkan akurasi tertinggi (0,78) serta *F1-Score* terbaik untuk Kelas 0 (0,77) dan Kelas 1 (0,79). Seiring bertambahnya durasi ke 70, 80, dan 90 detik, terjadi degradasi performa yang stabil di semua metrik. Akurasi menurun secara bertahap dari 0,78 menjadi 0,74. Begitu pula, *F1-Score* untuk kedua kelas juga terus menurun; misalnya, *F1-Score* untuk Kelas 1 turun dari puncaknya 0,79 pada 60 detik menjadi 0,75 pada 90 detik. Tren penurunan ini secara kuat mengindikasikan bahwa segmen data dengan durasi yang lebih pendek (60 detik) mengandung fitur-fitur yang paling informatif dan diskriminatif untuk model ini. Penambahan durasi data kemungkinan memasukkan noise atau informasi yang kurang relevan, sehingga mengurangi efektivitas klasifikasi secara keseluruhan.

Berdasarkan pada hasil *Confusion matrix* dan metrik kinerja pengujian tidak langsung dan pengujian langsung, tidak terdapat perbedaan nilai yang signifikan berarti model A.5 memiliki kinerja yang konsisten sebelum dan sesudah dikonversi menjadi model TinyML.

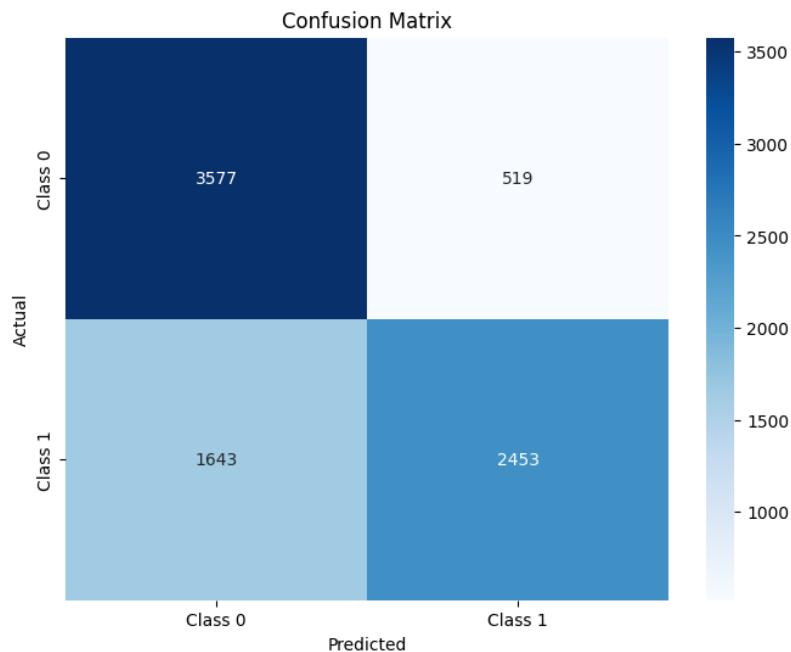
6. Model A.6

Model A.6 merupakan model untuk tipe data terkalibrasi, dengan data input berupa data *channel*, energi, dan *counts*. Susunan *layers*, *nodes*, dan fungsi aktivasi terdapat pada tabel 4.24.

Tabel 4.24 Susunan model A.6

<i>Layer</i>	<i>Jumlah nodes</i>	<i>Fungsi aktivasi</i>
1	30	relu
2	15	relu
3	5	relu
4	1	sigmoid

Hasil pengujian tidak langsung model A.6 dalam bentuk *Confusion matrix* terdapat pada gambar 4.7. Hasil pengujian langsung model A.6 dalam bentuk *Confusion matrix* terdapat pada tabel 4.25.



Gambar 4.7 *Confusion matrix* pengujian tidak langsung model A.6

Tabel 4.25 *Confusion matrix* pengujian langsung model A.6

Kelompok Dataset Waktu Cacah	Prediksi Aktual	Co-60	Cs-137
		Co-60	Cs-137
Dataset 60 detik	Co60	915	110
	Cs137	410	615
Dataset 70 detik	Co60	905	120
	Cs137	410	615
Dataset 80 detik	Co60	892	133
	Cs137	406	619
Dataset 90 detik	Co60	865	160
	Cs137	417	608
Total	Co60	3577	523
	Cs137	1643	2457

Dari *Confusion matrix* pengujian tidak langsung, metrik kinerja model pada pengujian tidak langsung terdapat pada tabel 4.26. Dari *Confusion matrix* pengujian langsung, metrik kinerja model pada pengujian langsung terdapat pada tabel 4.27

Tabel 4.26 Metrik kinerja pengujian tidak langsung model A.6

Metriks kinerja model	Nilai	
Akurasi	0,74	
Presisi	Kelas 0	0,69
	Kelas 1	0,83
Recall	Kelas 0	0,87
	Kelas 1	0,60
F1-Score	Kelas 0	0,77
	Kelas 1	0,69

Tabel 4.27 Metrik kinerja pengujian langsung model A.6

Kelompok Dataset Waktu Cacah	Kelas Positif	Akurasi	Recall	Presisi	F1-Score
60 detik	0	0.75	0.89	0.69	0.78
	1		0.60	0.85	0.70
70 detik	0	0.74	0.88	0.69	0.77
	1		0.60	0.84	0.70
80 detik	0	0.74	0.87	0.69	0.77
	1		0.60	0.82	0.70
90 detik	0	0.72	0.84	0.67	0.75
	1		0.59	0.79	0.68
Total	0	0.74	0.87	0.69	0.77
	1		0.60	0.82	0.69

Berdasarkan pada *Confusion matrix* pengujian tidak langsung dan pengujian langsung, model A.5 dapat mengidentifikasi sebagian besar titik cacah pada kedua kelas dengan baik sebagai TP dan TN. Jumlah eror FP dan FN lebih sedikit dibandingkan jumlah TP dan TN. Jumlah salah satu jenis eror cukup tinggi dibanding jumlah jenis eror lain. Selisih keduanya sebesar 1124. Begitu juga selisih antara jumlah TP dan TN. Berarti model ini tidak seimbang dalam mengidentifikasi kedua kelas.

Berdasarkan tabel 4.27, model klasifikasi mencapai akurasi keseluruhan sebesar 0,74, yang mengindikasikan tingkat performa yang baik secara umum. Analisis yang lebih mendalam pada metrik per kelas menunjukkan adanya ketidakseimbangan performa dan bias yang signifikan pada model.

Untuk Kelas 0, model menunjukkan nilai *F1-Score* yang tinggi sebesar 0,77. Hal ini didorong oleh nilai *recall* yang sangat tinggi (0,87), menandakan bahwa model mampu mengidentifikasi sebagian besar data cacah aktual dari Kelas 0. Akan tetapi,

kemampuan deteksi yang tinggi ini dicapai dengan mengorbankan presisi, yang tercatat lebih rendah (0,69), sehingga model rentan menghasilkan prediksi *false positive*. Sebaliknya, kinerja pada Kelas 1 lebih rendah dengan *F1-Score* hanya 0,69. Model ini memiliki karakteristik yang berlawanan, di mana ia menunjukkan presisi yang sangat tinggi (0,83), namun dengan *recall* yang rendah (0,60). Ini berarti, meskipun prediksi yang dibuat untuk Kelas 1 sangat dapat diandalkan, model gagal mengidentifikasi 40% dari total sampel aktual Kelas 1. Perbedaan yang jelas dalam profil kinerja ini—di mana model sangat sensitif terhadap Kelas 0 namun kurang mampu mendeteksi Kelas 1—menunjukkan adanya bias dalam identifikasi kedua kelas pada model A.6.

Berdasarkan tabel 4.27, model klasifikasi secara keseluruhan mencapai akurasi total sebesar 0,74. Meskipun angka ini menunjukkan performa yang cukup baik, analisis metrik yang lebih rinci pada baris "Total" mengungkap adanya ketidakseimbangan kinerja (performance imbalance) yang signifikan antara kedua kelas.

Model menunjukkan bias yang jelas terhadap Kelas 0, yang berhasil mencapai *F1-Score* tinggi sebesar 0,77. Kinerja ini didorong oleh nilai *recall* yang sangat tinggi (0,87), menandakan kemampuan model untuk mengidentifikasi hampir semua instans aktual dari Kelas 0. Sensitivitas yang tinggi ini dicapai dengan mengorbankan presisi, yang tercatat lebih rendah (0,69). Pada Kelas 1 menunjukkan *F1-Score* yang lebih rendah (0,69). Kelas ini memiliki karakteristik performa yang berlawanan: presisi yang sangat tinggi (0,82) namun dengan *recall* yang rendah (0,60), yang berarti model gagal mendeteksi 40% dari instans aktual Kelas 1.

Analisis lebih lanjut pada setiap Kelompok Dataset menunjukkan tren yang konsisten: kinerja model menurun seiring dengan bertambahnya durasi segmen data. Performa optimal tercatat pada kelompok dataset 60 detik, yang menghasilkan akurasi tertinggi (0,75) serta *F1-Score* terbaik untuk Kelas 0 (0,78) dan Kelas 1 (0,70). Seiring bertambahnya durasi menjadi 70, 80, dan 90 detik, terjadi degradasi performa yang sistematis. Akurasi menurun secara bertahap hingga mencapai titik terendah 0,72 pada

90 detik, di mana *F1-Score* untuk kedua kelas juga mengalami penurunan paling signifikan.

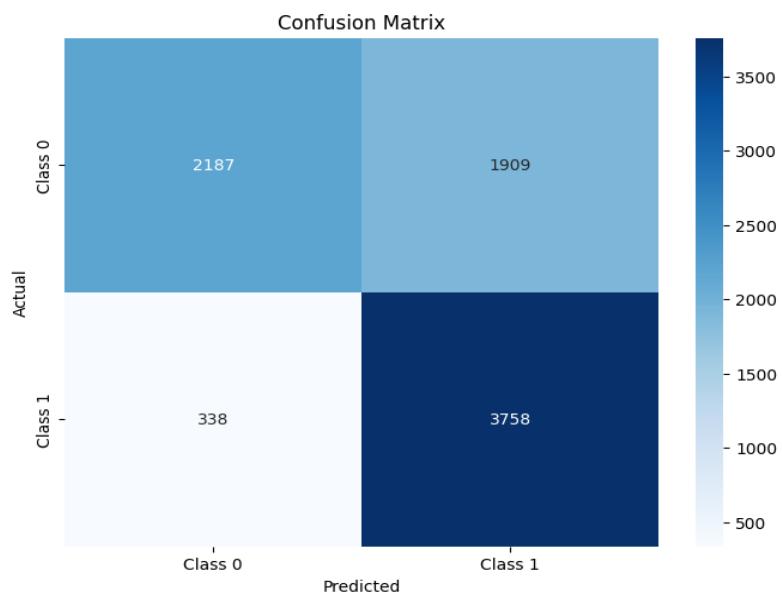
7. Model B.1

Model B.1 merupakan model untuk tipe data tidak terkalibrasi, dengan data input berupa data *channel* dan *counts*. Susunan *layers*, *nodes*, dan fungsi aktivasi terdapat pada tabel 4.28.

Tabel 4.28 Susunan model B.1

<i>Layer</i>	Jumlah <i>nodes</i>	Fungsi aktivasi
1	10	relu
2	5	relu
3	1	sigmoid

Hasil pengujian tidak langsung model B.1 dalam bentuk *Confusion matrix* terdapat pada gambar 4.8. Hasil pengujian langsung model B.1 dalam bentuk *Confusion matrix* terdapat pada tabel 4.29.



Gambar 4.8 *Confusion matrix* pengujian tidak langsung model B.1

Tabel 4.29 *Confusion matrix* pengujian langsung model B.1

Kelompok Dataset Waktu Cacah	Prediksi	Co-60	Cs-137
	Aktual		
60 detik	Co60	518	507
	Cs137	75	950
70 detik	Co60	539	486
	Cs137	79	946
80 detik	Co60	556	469
	Cs137	90	935
90 detik	Co60	574	451
	Cs137	94	931
Total	Co60	2187	1913
	Cs137	338	3762

Dari *Confusion matrix* pengujian tidak langsung, metrik kinerja model pada pengujian tidak langsung terdapat pada tabel 4.30. Dari *Confusion matrix* pengujian langsung, metrik kinerja model pada pengujian langsung terdapat pada tabel 4.31.

Tabel 4.30 Metrik kinerja pengujian tidak langsung model B.1

Metriks kinerja model	Nilai	
Akurasi	0,73	
Presisi	Kelas 0	0,87
	Kelas 1	0,66
Recall	Kelas 0	0,53
	Kelas 1	0,92
<i>F1-Score</i>	Kelas 0	0,66
	Kelas 1	0,77

Tabel 4.31 Metrik kinerja pengujian langsung model B.1

Kelompok Dataset	Kelas	Akurasi	Recall	Presisi	F1-Score
Waktu Cacah	Positif				
60 detik	0	0,72	0,51	0,87	0,64
	1		0,93	0,65	0,77
70 detik	0	0,72	0,53	0,87	0,66
	1		0,92	0,66	0,77
80 detik	0	0,73	0,54	0,86	0,67
	1		0,91	0,67	0,77
90 detik	0	0,73	0,56	0,86	0,68
	1		0,91	0,67	0,77
Total	0	0,73	0,53	0,87	0,66
	1		0,92	0,66	0,77

Berdasarkan pada *Confusion matrix* pengujian tidak langsung dan pengujian langsung, model B.1 dapat mengidentifikasi sebagian besar titik cacah pada kedua kelas dengan baik sebagai TP dan TN. Jumlah eror FP dan FN lebih sedikit dibandingkan jumlah TP dan TN. Jumlah salah satu jenis eror cukup tinggi dibanding jumlah jenis eror lain. Selisih keduanya sebesar 1571. Begitu juga selisih antara jumlah TP dan TN. Sehingga model ini tidak seimbang dalam mengidentifikasi kedua kelas.

Dari metrik kinerja pengujian tidak langsung, model mencapai akurasi keseluruhan sebesar 0,73, namun terdapat ketidakseimbangan kinerja yang signifikan dan bias yang kuat terhadap Kelas 1. Model ini sangat efektif dalam mendeteksi Kelas 1, yang ditunjukkan oleh nilai *recall* yang sangat tinggi (0,92), meskipun dengan presisi yang lebih rendah (0,66) sehingga menghasilkan *F1-Score* 0,77. Sebaliknya, model sangat kesulitan mengidentifikasi Kelas 0, yang terlihat dari nilai *recall* yang sangat rendah yaitu 0,53, membuatnya melewatkannya hampir separuh dari sampel aktual kelas ini. Meskipun prediksi untuk Kelas 0 sangat andal ketika dibuat (presisi 0,87),

kegalannya dalam mendeteksi secara komprehensif menghasilkan *F1-Score* yang jauh lebih rendah, yaitu 0,66. Kesenjangan performa yang dikonfirmasi oleh perbedaan *F1-Score* ini menunjukkan bahwa model sangat sensitif terhadap Kelas 1, namun tidak untuk Kelas 0.

Berdasarkan metrik kinerja pengujian langsung, model menunjukkan akurasi keseluruhan yang cukup baik sebesar 0,73, namun terdapat ketidakseimbangan kinerja yang signifikan dan bias yang kuat terhadap Kelas 1. Analisis total menunjukkan bahwa model sangat efektif dalam mendeteksi Kelas 1 (*recall* 0,92; *F1-Score* 0,77), namun hal ini dicapai dengan nilai presisi yang rendah (0,66). Sebaliknya, model sangat kesulitan mengidentifikasi Kelas 0 (*recall* 0,53), meskipun nilai presisi tinggi (presisi 0,87), yang menghasilkan *F1-Score* yang jauh lebih rendah (0,66). Analisis tren pada setiap kelompok dataset mengungkap bahwa penambahan durasi dari 60 menjadi 90 detik secara konsisten meningkatkan kinerja pada Kelas 0 yang lebih lemah—terlihat dari *F1-Score* yang naik dari 0,64 menjadi 0,68—sementara performa untuk Kelas 1 tetap sangat stabil (*F1-Score* 0,77). Hal ini mengindikasikan bahwa meskipun bias dasar tetap ada, penggunaan segmen data yang lebih panjang membantu mengurangi kesenjangan performa dan menghasilkan kinerja keseluruhan yang sedikit lebih baik dan seimbang pada durasi 90 detik.

8. Model B.2

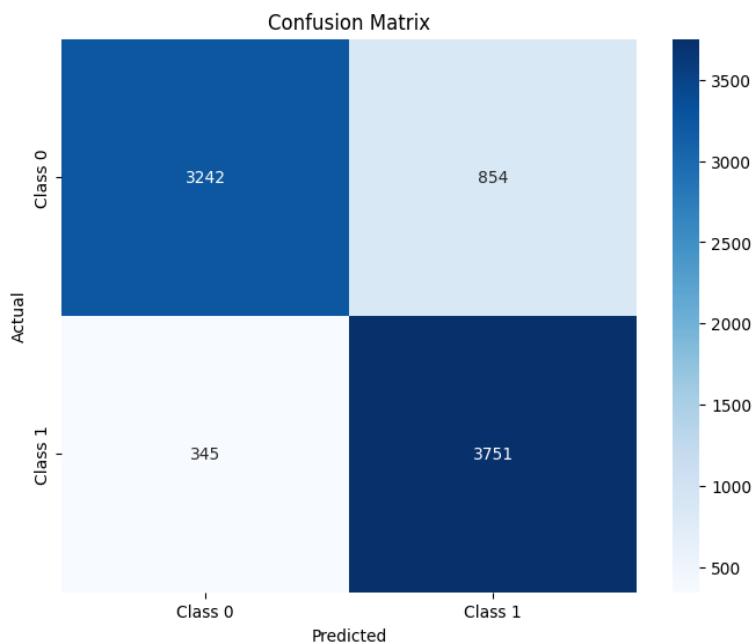
Model B.2 merupakan model untuk tipe data tidak terkalibrasi, dengan data input berupa data *channel* dan *counts*. susunan *layers*, *nodes*, dan fungsi aktivasi terdapat pada tabel 4.32.

Tabel 4.32 Susunan model B.2

<i>Layer</i>	<i>Jumlah</i>	<i>Fungsi</i>
--------------	---------------	---------------

	<i>nodes</i>	<i>aktivasi</i>
1	20	relu
2	10	relu
3	1	sigmoid

Hasil pengujian tidak langsung model B.2 dalam bentuk *Confusion matrix* terdapat pada gambar 4.9. Hasil pengujian langsung model B.2 dalam bentuk *Confusion matrix* terdapat pada tabel 4.33.



Gambar 4.9 *Confusion matrix* pengujian tidak langsung model B.2

Tabel 4.33 *Confusion matrix* pengujian langsung model B.2

Kelompok Dataset Waktu Cacah	Prediksi Aktual	Co-60	Cs-137

60 detik	Co60	746	279
	Cs137	94	931
70 detik	Co60	809	216
	Cs137	89	936
80 detik	Co60	825	200
	Cs137	91	934
90 detik	Co60	866	159
	Cs137	75	950
Total	Co60	3246	854
	Cs137	349	3751

Dari *Confusion matrix* pengujian tidak langsung, metrik kinerja model pada pengujian tidak langsung terdapat pada tabel 4.34. Dari *Confusion matrix* pengujian langsung, metrik kinerja model pada pengujian langsung terdapat pada tabel 4.35.

Tabel 4.34 Metrik kinerja pengujian tidak langsung model B.2

Metriks kinerja model	Nilai	
Akurasi	0,85	
Presisi	Kelas 0	0,90
	Kelas 1	0,81
<i>Recall</i>	Kelas 0	0,79
	Kelas 1	0,92
<i>F1-Score</i>	Kelas 0	0,84
	Kelas 1	0,86

Tabel 4.35 Metrik kinerja pengujian langsung model B.2

Kelompok Dataset Waktu Cacah	Kelas Positif	Akurasi	Recall	Presisi	<i>F1-Score</i>
60 detik	0	0,82	0,73	0,89	0,80
	1		0,91	0,77	0,83

70 detik	0	0,85	0,79	0,90	0,84
	1		0,91	0,81	0,86
80 detik	0	0,86	0,80	0,90	0,85
	1		0,91	0,82	0,87
90 detik	0	0,89	0,84	0,92	0,88
	1		0,93	0,86	0,89
Total	0	0,85	0,79	0,90	0,84
	1		0,91	0,81	0,86

Berdasarkan pada *Confusion matrix* pengujian tidak langsung dan pengujian langsung, model B.2 dapat mengidentifikasi sebagian besar titik cacah pada kedua kelas dengan baik sebagai TP dan TN. Jumlah eror FP dan FN lebih sedikit dibandingkan jumlah TP dan TN. jumlah salah satu jenis eror cukup tinggi dibanding jumlah jenis eror lain. Selisih keduanya sebesar 509. Begitu juga selisih antara jumlah TP dan TN. Berarti model ini tidak seimbang dalam mengidentifikasi kedua kelas.

Dari metrik kinerja pengujian tidak langsung pada tabel 4.34, model B.2 menunjukkan kinerja yang baik dengan akurasi keseluruhan mencapai 0,85. Performa yang kuat ini divalidasi lebih lanjut oleh nilai *F1-Score* yang tinggi dan hampir setara untuk kedua kelas, yaitu 0,84 untuk Kelas 0 dan 0,86 untuk Kelas 1, yang mengonfirmasi bahwa model tidak memiliki bias yang signifikan. Secara spesifik, model menunjukkan karakteristik yang saling melengkapi: Kelas 0 unggul dalam presisi (0,90), yang menandakan keandalan prediksi yang sangat tinggi, sementara Kelas 1 unggul dalam *recall* (0,92)

Berdasarkan tabel 4.35, model menunjukkan kinerja keseluruhan yang baik, dengan akurasi total mencapai 0,85 serta *F1-Score* yang tinggi untuk Kelas 0 (0,84) dan Kelas 1 (0,86). Kinerja model ini juga menunjukkan tren peningkatan yang jelas dan konsisten seiring dengan bertambahnya durasi segmen data dari 60 menjadi 90 detik. Peningkatan ini terlihat pada semua metrik utama, di mana akurasi naik secara signifikan dari 0,82 menjadi 0,89. Secara spesifik, *F1-Score* untuk Kelas 0 meningkat dari 0,80 menjadi 0,88, dan untuk Kelas 1 dari 0,83 menjadi 0,89. Performa puncak

dicapai pada kelompok dataset 90 detik, yang menghasilkan kinerja tertinggi dan paling seimbang.

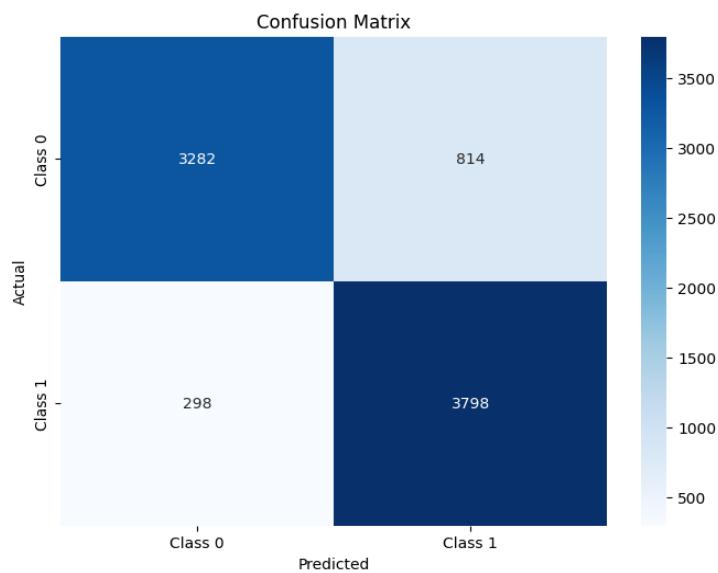
9. Model B.3

Model B.3 merupakan model untuk tipe data tidak terkalibrasi, dengan data input berupa data *channel* dan *counts*. Susunan *layers*, *nodes*, dan fungsi aktivasi terdapat pada tabel 4.36.

Tabel 4.36 Susunan model B.3

<i>Layer</i>	Jumlah <i>nodes</i>	Fungsi aktivasi
1	30	relu
2	15	relu
3	1	sigmoid

Hasil pengujian tidak langsung model B.3 dalam bentuk *Confusion matrix* terdapat pada gambar 4.10. Hasil pengujian langsung model B.3 dalam bentuk *Confusion matrix* terdapat pada tabel 4.37.



Gambar 4.10 *Confusion matrix* pengujian tidak langsung model B.3

Tabel 4.37 *Confusion matrix* pengujian langsung model B.3

Kelompok Dataset	Aktual	Prediksi	Co-60	Cs-137
60 detik	Co60	786	239	
	Cs137	86	939	
70 detik	Co60	826	199	
	Cs137	75	950	
80 detik	Co60	826	199	
	Cs137	76	949	
90 detik	Co60	848	177	
	Cs137	65	960	
Total	Co60	3286	814	
	Cs137	302	3798	

Dari *Confusion matrix* pengujian tidak langsung, metrik kinerja model pada pengujian tidak langsung terdapat pada tabel 4.38.

Tabel 4.38 Metrik kinerja pengujian tidak langsung model B.3

Metriks kinerja model	Nilai	
Akurasi	0,86	
Presisi	Kelas 0	0,92
	Kelas 1	0,82
Recall	Kelas 0	0,80
	Kelas 1	0,93
F1-Score	Kelas 0	0,86
	Kelas 1	0,87

Dari *Confusion matrix* pengujian langsung, metrik kinerja model pada pengujian langsung terdapat pada tabel 4.39.

Tabel 4.39 Metrik kinerja pengujian langsung model B.3

Kelompok Dataset Waktu Cacah	Kelas Positif	Akurasi	Recall	Presisi	F1-Score
60 detik	0	0,84	0,77	0,90	0,83
	1		0,92	0,80	0,85
70 detik	0	0,87	0,81	0,92	0,86
	1		0,93	0,83	0,87
80 detik	0	0,87	0,81	0,92	0,86
	1		0,93	0,83	0,87
90 detik	0	0,88	0,83	0,93	0,88
	1		0,94	0,84	0,89
Total	0	0,86	0,80	0,92	0,85
	1		0,93	0,82	0,87

Berdasarkan pada *Confusion matrix* pengujian tidak langsung dan pengujian langsung, model B.3 dapat mengidentifikasi sebagian besar titik cacah pada kedua kelas dengan baik sebagai TP dan TN. Jumlah eror FP dan FN lebih sedikit dibandingkan jumlah TP dan TN. Jumlah salah satu jenis eror cukup tinggi dibanding jumlah jenis eror lain. Selisih keduanya sebesar 516. Begitu juga selisih antara jumlah TP dan TN. Berarti model ini tidak seimbang dalam mengidentifikasi kedua kelas.

Berdasarkan tabel 4.38, model B.3 pada pengujian tidak langsung memiliki kinerja yang baik dengan akurasi keseluruhan sebesar 0,86 disertai *F1-Score* yang baik dan hampir identik untuk kedua kelas, yaitu 0,86 untuk Kelas 0 dan 0,87 untuk Kelas 1, yang menunjukkan bahwa model ini tidak memiliki bias yang signifikan. Pada kelas 0, nilai presisi yang tinggi, sebesar 0,92, menunjukkan model dapat memprediksi data cacah sebagai kelas 0 dengan baik dengan tingkat eror False Positive yang rendah. Begitu juga dengan kinerja nilai *recall* sebesar 0,80, model ini dapat meminimalkan tingkat eror false negative. Begitu juga pada kelas 1 dengan nilai presisi sebesar 0,82 dan nilai *recall* sebesar 0,93 yang menunjukkan kinerja model dalam meminimalkan tingkat kedua jenis eror.

Berdasarkan tabel 4.39, model ini menunjukkan kinerja keseluruhan yang baik dan seimbang, dengan nilai kinerja yang hampir sama dengan tabel pengujian sebelumnya. Nilai akurasi total mencapai 0,86 serta *F1-Score* yang tinggi untuk Kelas 0 (0,85) dan Kelas 1 (0,87). Berdasarkan tabel tersebut juga mengungkap adanya tren peningkatan performa yang jelas dan konsisten seiring dengan bertambahnya durasi segmen data dari 60 menjadi 90 detik. Peningkatan ini teramat pada semua metrik utama, di mana akurasi naik dari 0,84 menjadi 0,88, dan *F1-Score* untuk kedua kelas juga terus membaik. Kinerja puncak dicapai pada kelompok dataset 90 detik, yang menghasilkan akurasi tertinggi (0,88) dan *F1-Score* yang paling seimbang dan tinggi (0,88 untuk Kelas 0 dan 0,89 untuk Kelas 1). Hal ini secara kuat mengindikasikan bahwa model mampu secara efektif memanfaatkan informasi dari segmen data berdurasi lebih panjang untuk meningkatkan baik kemampuan deteksi maupun keandalan prediksinya pada kedua kelas.

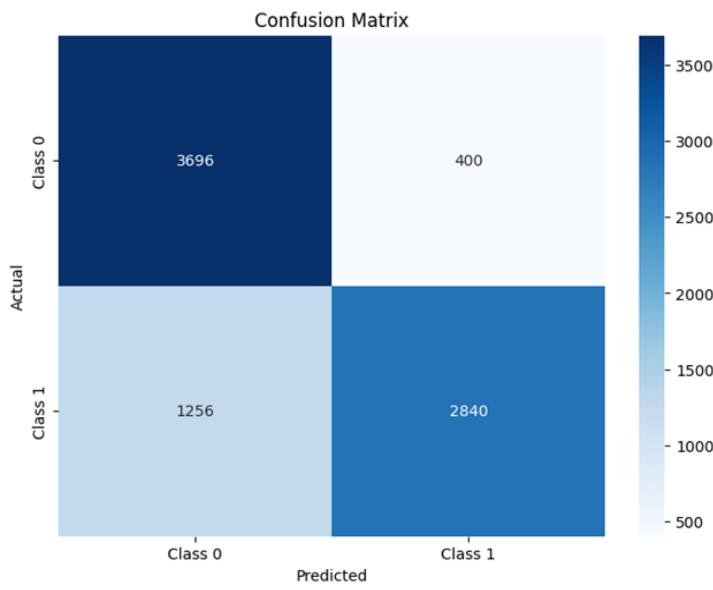
10. Model B.4

Model B.4 merupakan model untuk tipe data tidak terkalibrasi, dengan data input berupa data *channel* dan *counts*. susunan *layers*, *nodes*, dan fungsi aktivasi terdapat pada tabel 4.40

Tabel 4.40 Susunan model B.4

<i>Layer</i>	Jumlah <i>nodes</i>	Fungsi aktivasi
1	10	relu
2	5	relu
3	5	relu
4	1	sigmoid

Hasil pengujian tidak langsung model B.4 dalam bentuk *Confusion matrix* terdapat pada gambar 4.11. Hasil pengujian langsung model B.4 dalam bentuk *Confusion matrix* terdapat pada tabel 4.41.



Gambar 4.11 *Confusion matrix* pengujian tidak langsung model B.4

Tabel 4.41 *Confusion matrix* pengujian langsung model B.4

Kelompok Dataset	Prediksi	Co-60	Cs-137
Waktu Cacah	Aktual		
60 detik	Co60	899	126
	Cs137	297	728
70 detik	Co60	923	102
	Cs137	308	717
80 detik	Co60	927	98
	Cs137	324	701
90 detik	Co60	951	74
	Cs137	331	694
Total	Co60	3700	400
	Cs137	1260	2840

Dari *Confusion matrix* pengujian tidak langsung, metrik kinerja model pada pengujian tidak langsung terdapat pada tabel 4.42. Dari *Confusion matrix* pengujian langsung, metrik kinerja model pada pengujian langsung terdapat pada tabel 4.43.

Tabel 4.42 Metrik kinerja pengujian tidak langsung model B.4

Metriks kinerja model	Nilai	
Akurasi	0,80	
Presisi	Kelas 0	0,75
	Kelas 1	0,88
<i>Recall</i>	Kelas 0	0,90
	Kelas 1	0,69
<i>F1-Score</i>	Kelas 0	0,82
	Kelas 1	0,77

Tabel 4.43 Metrik kinerja pengujian langsung model B.4

Kelompok Dataset Waktu Cacah	Kelas Positif	Akurasi	Recall	Presisi	F1-Score
60 detik	0	0.79	0.88	0.75	0.81
	1		0.71	0.85	0.77
70 detik	0	0.80	0.90	0.75	0.82
	1		0.70	0.88	0.78
80 detik	0	0.79	0.90	0.74	0.81
	1		0.68	0.88	0.77
90 detik	0	0.80	0.93	0.74	0.82
	1		0.68	0.90	0.77
Total	0	0.80	0.90	0.75	0.82
	1		0.69	0.88	0.77

Berdasarkan pada *confusion matrix* pengujian tidak langsung dan pengujian langsung, model B.4 dapat mengidentifikasi sebagian besar titik cacah pada kedua kelas dengan baik sebagai TP dan TN. Jumlah eror FP dan FN lebih sedikit dibandingkan jumlah TP dan TN. Jumlah salah satu jenis eror cukup tinggi dibanding

jumlah jenis eror lain. Selisih keduanya sebesar 856. Begitu juga selisih antara jumlah TP dan TN. Berarti model ini tidak seimbang dalam mengidentifikasi kedua kelas.

Berdasarkan tabel 4.42, model menunjukkan kinerja yang baik dengan akurasi keseluruhan sebesar 0,80, namun terdapat ketidakseimbangan performa dan bias yang signifikan terhadap Kelas 0. Model ini sangat unggul dalam mendeteksi Kelas 0, yang dibuktikan dengan nilai *recall* yang sangat tinggi (0,90), artinya model berhasil mengidentifikasi 90% dari semua sampel aktual Kelas 0. Kemampuan deteksi yang ini menghasilkan *F1-Score* yang kuat sebesar 0,82, meskipun dengan nilai presisi yang lebih rendah (0,75). Sebaliknya, model menunjukkan karakteristik yang berlawanan untuk Kelas 1, di mana ia sangat andal dalam prediksinya dengan presisi yang sangat tinggi (0,88), namun performa ini dibayar dengan *recall* yang rendah (0,69), yang berarti model gagal mendeteksi 31% dari sampel aktual Kelas 1. Kesenjangan performa yang jelas ini, yang dikonfirmasi oleh perbedaan nilai *F1-Score* (0,82 untuk Kelas 0 dibanding 0,77 untuk Kelas 1), menunjukkan bahwa model beroperasi dengan bias yang kuat dalam hal mendeteksi sebagian besar data cacah sebagai kelas 0.

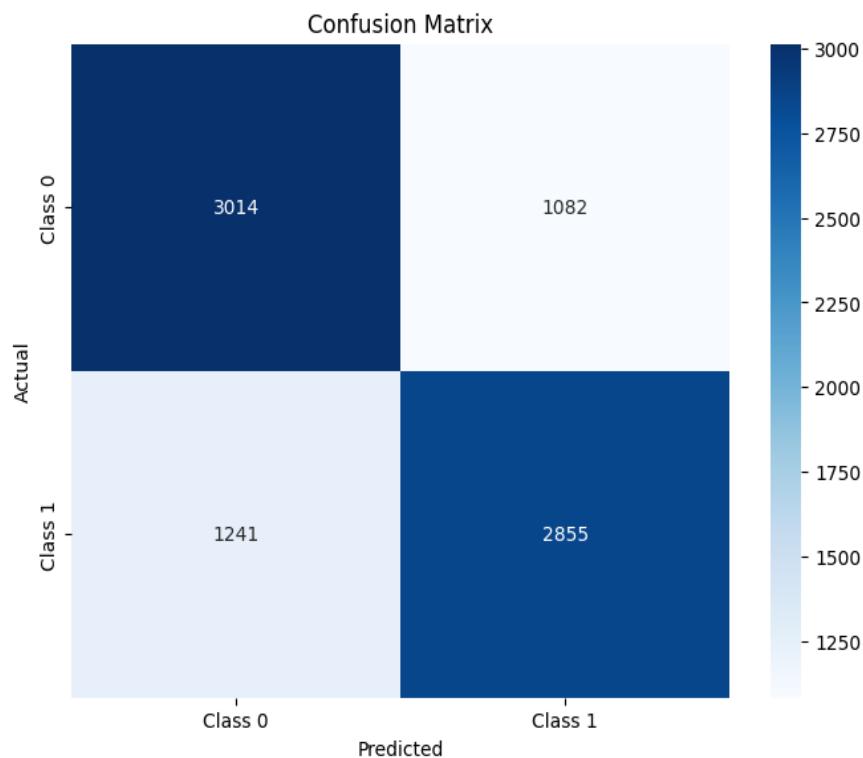
Berdasarkan tabel 4.43, model menunjukkan kinerja keseluruhan yang baik dengan akurasi total 0,80, namun data ini mengungkap adanya bias performa yang signifikan terhadap Kelas 0. Secara agregat, model ini unggul dalam mendeteksi Kelas 0 dengan *recall* sangat tinggi (0,90) dan *F1-Score* 0,82, meskipun dengan presisi yang lebih rendah (0,75). Sebaliknya, untuk Kelas 1, model menunjukkan keandalan prediksi yang tinggi dengan presisi sangat tinggi (0,88), namun dengan rendahnya nilai *recall* (*recall* 0,69) yang menghasilkan *F1-Score* lebih rendah yaitu 0,77. Analisis tren pada setiap kelompok dataset menunjukkan bahwa kinerja model secara keseluruhan sangat stabil dan tidak banyak berubah seiring penambahan durasi dari 60 hingga 90 detik. Durasi yang lebih panjang cenderung memperkuat karakteristik awal model: *recall* untuk Kelas 0 yang sudah tinggi semakin meningkat, dan presisi untuk Kelas 1 yang juga sudah tinggi turut meningkat, sementara nilai *F1-Score* untuk kedua kelas tetap konsisten.

11. Model B.5

Model B.5 merupakan model untuk tipe data tidak terkalibrasi, dengan data input berupa data *channel* dan *counts*. Susunan *layers*, *nodes*, dan fungsi aktivasi terdapat pada tabel 4.44. Hasil pengujian tidak langsung model B.5 dalam bentuk *Confusion matrix* terdapat pada gambar 4.12. Hasil pengujian langsung model B.5 dalam bentuk *Confusion matrix* terdapat pada tabel 4.45

Tabel 4.44 Susunan model B.5

Layer	Jumlah nodes	Fungsi aktivasi
1	20	relu
2	10	relu
3	5	relu
4	1	sigmoid



Gambar 4.12 *Confusion matrix* pengujian tidak langsung model B.5

Tabel 4.45 *Confusion matrix* pengujian langsung model B.5

Kelompok Dataset Waktu Cacah	Prediksi Aktual		
		Co-60	Cs-137
60 detik	Co60	679	346
	Cs137	310	715
70 detik	Co60	754	271
	Cs137	310	715
80 detik	Co60	770	255
	Cs137	313	712
90 detik	Co60	815	210
	Cs137	312	713
Total	Co60	3018	1082
	Cs137	1245	2855

Tabel 4.46 Metrik kinerja pengujian tidak langsung model B.5

Metriks kinerja model	Nilai	
Akurasi	0,72	
Presisi	Kelas 0	0,71
	Kelas 1	0,73
<i>Recall</i>	Kelas 0	0,74
	Kelas 1	0,70
<i>F1-Score</i>	Kelas 0	0,72
	Kelas 1	0,71

Tabel 4.47 Metrik kinerja pengujian langsung model B.5

Kelompok Dataset Waktu Cacah	Kelas Positif	Akurasi	Recall	Presisi	F1-Score
60 detik	0	0.68	0.66	0.69	0.67
	1		0.70	0.67	0.69
70 detik	0	0.72	0.74	0.71	0.72
	1		0.70	0.73	0.71
80 detik	0	0.72	0.75	0.71	0.73
	1		0.69	0.74	0.71
90 detik	0	0.75	0.80	0.72	0.76
	1		0.70	0.77	0.73
Total	0	0.72	0.74	0.71	0.72
	1		0.70	0.73	0.71

Dari *Confusion matrix* pengujian tidak langsung, metrik kinerja model pada pengujian tidak langsung terdapat pada tabel 4.46. Dari *Confusion matrix* pengujian langsung, metrik kinerja model pada pengujian langsung terdapat pada tabel 4.47.

Berdasarkan pada *confusion matrix* pengujian tidak langsung dan pengujian langsung, model B.5 dapat mengidentifikasi sebagian besar titik cacah pada kedua kelas dengan baik sebagai TP dan TN. Jumlah eror FP dan FN lebih sedikit dibandingkan jumlah TP dan TN. Jumlah salah satu jenis eror cukup tinggi dibanding jumlah jenis eror lain. Selisih keduanya sebesar 159. Begitu juga selisih antara jumlah TP dan TN. Berarti model ini tidak seimbang dalam mengidentifikasi kedua kelas.

Berdasarkan tabel 4.46, model menunjukkan kinerja yang baik dan seimbang dengan akurasi keseluruhan sebesar 0,72. Keseimbangan ini dilihat nilai *F1-Score* yang hampir identik untuk kedua kelas, yaitu 0,72 untuk Kelas 0 dan 0,71 untuk Kelas 1, yang menunjukkan tidak adanya bias signifikan dalam performa model. Kinerja model dalam mengidentifikasi kelas 0 menunjukkan nilai presisi sebesar 0,71 dan nilai *recall*

sebesar 0,74. Kedua nilai tersebut menunjukkan kemampuan model untuk mengidentifikasi data cacah sebagai kelas 0 dengan tepat dan meminimalkan eror. Begitu juga dengan kemampuan dalam mengidentifikasi kelas 1 dengan nilai presisi sebesar 0,73 dan nilai *recall* sebesar 0,70. Model B.5 dapat mengidentifikasi data cacah sebagai kedua kelas tersebut dengan tingkat ketepatan yang cukup tinggi dan tingkat eror yang cukup rendah.

Berdasarkan tabel 4.47, model menunjukkan kinerja yang baik dan sangat seimbang dengan akurasi total 0,72 dan nilai *F1-Score* yang hampir identik untuk Kelas 0 (0,72) dan Kelas 1 (0,71), yang mengonfirmasi tidak adanya bias yang signifikan. Analisis tren pada setiap kelompok dataset mengungkap adanya peningkatan performa yang konsisten dan positif seiring bertambahnya durasi segmen data dari 60 menjadi 90 detik. Pada Kelas 0, terdapat peningkatan nilai *recall* dari 0,66 menjadi 0,80, sementara untuk Kelas 1, nilai presisi bertambah dari 0,67 menjadi 0,77). Kinerja puncak dicapai pada kelompok 90 detik dengan akurasi tertinggi (0,75) dan *F1-Score* terbaik (0,76 untuk Kelas 0 dan 0,73 untuk Kelas 1).

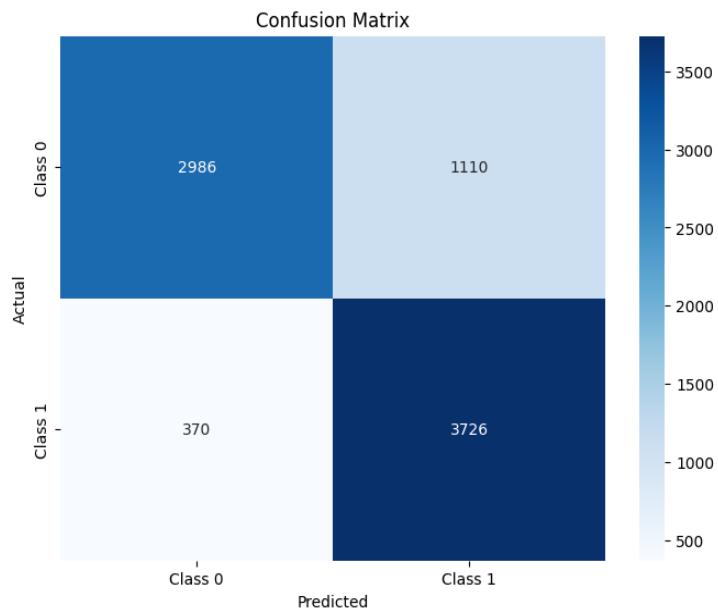
12. Model B.6

Model B.6 merupakan model untuk tipe data tidak terkalibrasi, dengan data input berupa data *channel* dan *counts*. Susunan *layers*, *nodes*, dan fungsi aktivasi terdapat pada tabel 4.48.

Tabel 4.48 Susunan model B.6

<i>Layer</i>	Jumlah <i>nodes</i>	Fungsi aktivasi
1	30	relu
2	15	relu
3	5	relu
4	1	sigmoid

Hasil pengujian tidak langsung model B.6 dalam bentuk *Confusion matrix* terdapat pada gambar 4.13. Hasil pengujian langsung model B.6 dalam bentuk *Confusion matrix* terdapat pada tabel 4.49.



Gambar 4.13 *Confusion matrix* pengujian tidak langsung model B.6

Tabel 4.49 *Confusion matrix* pengujian langsung model B.6

Kelompok Dataset Waktu Cacah	Prediksi Aktual	Co-	Cs-
		60	137
60 detik	Co60	704	321
	Cs137	96	929
70 detik	Co60	756	269
	Cs137	93	930
80 detik	Co60	758	267
	Cs137	99	926
90 detik	Co60	772	253
	Cs137	84	941
Total	Co60	2990	1110
	Cs137	372	3726

Dari *Confusion matrix* pengujian tidak langsung, metrik kinerja model pada pengujian tidak langsung adalah sebagai berikut.

Tabel 4.50 Metrik kinerja pengujian tidak langsung model B.6

Metriks kinerja model	Nilai	
Akurasi	0,82	
Presisi	Kelas 0	0,89
	Kelas 1	0,77
<i>Recall</i>	Kelas 0	0,73
	Kelas 1	0,91
<i>F1-Score</i>	Kelas 0	0,80
	Kelas 1	0,83

Dari *Confusion matrix* pengujian langsung, metrik kinerja model pada pengujian langsung adalah sebagai berikut.

Tabel 4.51 Metrik kinerja pengujian langsung model B.6

Kelompok Dataset Waktu Cacah	Kelas Positif	Akurasi	Recall	Presisi	F1-Score
60 detik	0	0.80	0.69	0.88	0.77
	1		0.91	0.74	0.82
70 detik	0	0.82	0.74	0.89	0.81
	1		0.91	0.78	0.84
80 detik	0	0.82	0.74	0.88	0.81
	1		0.90	0.78	0.83
90 detik	0	0.84	0.75	0.90	0.82
	1		0.92	0.79	0.85
Total	0	0.82	0.73	0.89	0.80
	1		0.91	0.77	0.83

Berdasarkan pada *confusion matrix* pengujian tidak langsung dan pengujian langsung, model B.6 dapat mengidentifikasi sebagian besar titik cacah pada kedua kelas dengan baik sebagai TP dan TN. Jumlah eror FP dan FN lebih sedikit dibandingkan jumlah TP dan TN. Namun jumlah salah satu jenis eror cukup tinggi dibanding jumlah jenis eror lain. Selisih keduanya sebesar 740. Begitu juga selisih antara jumlah TP dan TN. Berarti model ini tidak cukup seimbang dalam mengidentifikasi kedua kelas.

Berdasarkan tabel 4.50, model ini menunjukkan kinerja yang baik dengan akurasi keseluruhan 0,82 serta nilai *F1-Score* yang tinggi dan berdekatan untuk kedua kelas, yaitu 0,80 untuk Kelas 0 dan 0,83 untuk Kelas 1. Pada kelas 0, terdapat perbedaan nilai yang cukup tinggi antara nilai presisi (0,89) dan nilai *recall* (0,73). Begitu juga pada kelas 1 juga terdapat selisih pada nilai presisi dan *recall* dengan nilai presisi sebesar 0,77 dan nilai *recall* sebesar 0,91. Model ini menunjukkan tingkat keseimbangan yang rendah dikarenakan perbedaan nilai presisi dan *recall* meskipun memiliki nilai *F1-Score* yang tinggi

Berdasarkan tabel 4.51, model ini menunjukkan kinerja keseluruhan yang sangat baik dengan akurasi total 0,82 dan *F1-Score* yang tinggi untuk Kelas 0 (0,80) dan Kelas 1 (0,83).. Secara rinci, model ini menunjukkan karakteristik komplementer, di mana Kelas 0 unggul dalam presisi (0,89) sedangkan Kelas 1 unggul dalam *recall* (0,91). Analisis lebih lanjut mengungkap adanya tren peningkatan performa yang konsisten seiring bertambahnya durasi dataset dari 60 menjadi 90 detik, dengan akurasi yang naik dari 0,80 menjadi 0,84. Peningkatan ini didorong oleh naiknya *recall* pada Kelas 0 (dari 0,69 menjadi 0,75) dan naiknya presisi pada Kelas 1 (dari 0,74 menjadi 0,79). Kinerja puncak tercapai pada kelompok 90 detik, yang menghasilkan *F1-Score* tertinggi untuk kedua kelas (0,82 dan 0,85)

4.3 Perbandingan Kinerja Model

Perbandingan kinerja semua model terdapat pada tabel 4.52. Berdasarkan tabel tersebut, diketahui bahwa setiap model memiliki kinerja yang berbeda-beda. Perbedaan tersebut dipengaruhi oleh perbedaan spesifikasi model, yaitu jumlah *layers*, dan *nodes*.

Perbedaan kinerja juga dipengaruhi oleh data pengujian yang digunakan. Berdasarkan pada tabel tersebut, beberapa model dengan jumlah *layers* dan *nodes* yang lebih banyak memiliki kinerja yang lebih baik dibandingkan dengan model lainnya. Namun, ada beberapa model yang memiliki dengan jumlah *layers* dan *nodes* yang banyak memiliki kinerja yang serupa atau lebih rendah dengan model dengan jumlah *layers* dan *nodes* yang sedikit.

Model yang efektif adalah model dengan nilai metrik kinerja yang tinggi dan seimbang. Nilai metrik yang tinggi berarti jumlah nilai yang terprediksi benar sebagai TP dan TN lebih banyak dibandingkan nilai yang tidak terprediksi dengan tepat atau sebagai nilai FP dan FN. Model yang seimbang berarti kualitas kinerja dalam mengidentifikasi kedua kelas bernilai sama. Dan sebagai model TinyML, ukuran memori yang kecil menunjukkan bahwa model TinyML yang efektif dan efisien.

Berdasarkan pada tabel 4.52, model yang efektif dan efisien adalah model A.3 dikarenakan nilai akurasi yang cukup tinggi sebesar 0,74 disertai dengan tingkat keseimbangan kinerja tertinggi dalam identifikasi kedua kelas dan penggunaan memori yang cukup kecil dibandingkan dengan model-model lainnya.

Model A.2, A.4, A.5, A.6, B.1, B.2, B.3, B.4, B.5, dan B.6 memiliki kinerja dengan akurasi yang tinggi dan juga seimbang dalam mengidentifikasi kedua kelas, namun tingkat keseimbangan kinerja lebih rendah dibandingkan model A.3. Model A.1 memiliki kinerja yang rendah dan tidak seimbang dalam mengidentifikasi kedua kelas.

Tabel 4.52 Tabel perbandingan kinerja semua model

Model	Spesifikasi model			Ukuran memori di MCU (KB)		Jenis dataset pengujian	Kinerja Pengujian Langsung				Kinerja Pengujian Langsung			
	Layers	Nodes	Fungsi Aktivasi	Flash	RAM		Akurasi	Presisi	Recall	F1-Score	Akurasi	Presisi	Recall	F1-Score
A.1	1	10	relu	598,7	54,6	Co-60	0,63	0,59	0,87	0,7	0,63	0,59	0,87	0,70
	2	5	relu			Cs-137		0,75	0,39	0,51		0,75	0,39	0,51
	3	1	Sigmoid											
A.2	1	20	relu	599,5	55,4	Co-60	0,75	0,73	0,80	0,76	0,75	0,73	0,79	0,76
	2	10	relu			Cs-137		0,77	0,70	0,74		0,77	0,70	0,74
	3	1	Sigmoid											
A.3	1	30	relu	600	56,6	Co-60	0,74	0,73	0,75	0,74	0,74	0,73	0,75	0,74
	2	15	relu			Cs-137		0,74	0,73	0,74		0,74	0,73	0,74
	3	1	Sigmoid											
A.4	1	10	relu	599,2	55,1	Co-60	0,75	0,76	0,73	0,75	0,75	0,76	0,73	0,75
	2	5	relu			Cs-137		0,74	0,77	0,76		0,74	0,77	0,76
	3	5	relu											
	4	1	Sigmoid											

Model	Spesifikasi model			Ukuran memori di MCU (KB)		Jenis dataset pengujian	Kinerja Pengujian Langsung				Tidak				Kinerja Pengujian Langsung								
	Layers	Nodes	Fungsi Aktivasi	Flash	RAM		Akurasi	Presisi	Recall	F1-Score	Akurasi	Presisi	Recall	F1-Score	Akurasi	Presisi	Recall	F1-Score					
A.5	1	20	relu	600	56,0	Co-60	0,76	0,78	0,72	0,75	0,76	0,78	0,72	0,75	0,74	0,8	0,77	0,74	0,80	0,77			
	2	10	relu			Cs-137		0,74	0,8	0,77													
	3	5	relu																				
	4	1	Sigmoid																				
A.6	1	30	relu	601,4	57,3	Co-60	0,74	0,69	0,87	0,77	0,74	0,69	0,87	0,77	Cs-137	0,83	0,60	0,69	0,82	0,60	0,69		
	2	15	relu			Cs-137																	
	3	5	relu																				
	4	1	Sigmoid																				
B.1	1	10	relu	598,5	50,5	Co-60	0,73	0,87	0,53	0,66	0,73	0,87	0,53	0,66	Cs-137	0,66	0,92	0,77	0,66	0,92	0,77		
	2	5	relu			Cs-137																	
	3	1	Sigmoid																				

Model	Spesifikasi model			Ukuran memori di MCU (KB)		Jenis dataset pengujian	Kinerja Pengujian Langsung				Kinerja Pengujian Langsung			
	Layers	Nodes	Fungsi Aktivasi	Flash	RAM		Akurasi	Presisi	Recall	F1-Score	Akurasi	Presisi	Recall	F1-Score
B.2	1	20	relu	599,3	51,2	Co-60	0,85	0,90	0,79	0,84	0,85	0,90	0,79	0,84
	2	10	relu			Cs-137		0,81	0,92	0,86		0,81	0,91	0,86
	3	1	Sigmoid											
B.3	1	30	relu	600	52,4	Co-60	0,86	0,92	0,80	0,86	0,86	0,92	0,80	0,85
	2	15	relu			Cs-137		0,82	0,93	0,87		0,82	0,93	0,87
	3	1	Sigmoid											
B.4	1	10	relu	599	50,9	Co-60	0,8	0,75	0,9	0,82	0,80	0,75	0,90	0,82
	2	5	relu			Cs-137		0,88	0,69	0,77		0,88	0,69	0,77
	3	5	relu											
	4	1	Sigmoid											
B.5	1	20	relu	599,9	51,8	Co-60	0,72	0,71	0,74	0,72	0,72	0,71	0,74	0,72
	2	10	relu			Cs-137		0,73	0,70	0,71		0,73	0,70	0,71
	3	5	relu											
	4	1	Sigmoid											

Model	Spesifikasi model			Ukuran memori di MCU (KB)		Jenis dataset pengujian	Kinerja Pengujian Langsung				Tidak				Kinerja Pengujian Langsung			
	Layers	Nodes	Fungsi Aktivasi	Flash	RAM		Akurasi	Presisi	Recall	F1-Score	Akurasi	Presisi	Recall	F1-Score				
B.6	1	30	relu	601,1	53	Co-60	0,82	0,89	0,73	0,80	0,82	0,89	0,73	0,80				
	2	15	relu			Cs-137		0,77	0,91	0,83		0,77	0,91	0,83				
	3	5	relu															
	4	1	Sigmoid															

BAB 5 KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan pada metodologi yang dilaksanakan dan hasil yang diperoleh, disimpulkan bahwa

1. Tahapan atau proses pembuatan dan pengujian model TinyML terdiri dari menetapkan tujuan program model ML, mengumpulkan dan mengolah dataset, membuat model ML, menguji model ML, mengonversi model ML menjadi model TinyML, mengunggah model TinyML ke MCU, menguji model TinyML, dan mengevaluasi hasil - hasil pengujian model ML dan TinyML.
2. Kinerja program TinyML ditampilkan dalam bentuk *Confusion matrix* dan metrik kinerja model TinyML. Nilai metrik kinerja pengujian berbagai model TinyML bervariasi. Nilai akurasi sebesar 0,63 – 0,86. Nilai presisi sebesar 0,59 – 0,92. Nilai *recall* sebesar 0,39 – 0,93. Nilai *F1-Score* sebesar 0,51 – 0,87.
3. Model dengan nilai metrik kinerja tertinggi adalah model B.3. Model ini untuk dataset tidak terkalibrasi dengan jumlah *layers* sebanyak 3 dan jumlah *nodes* sebanyak 30, 15, dan 1. Nilai akurasi sebesar 0,86. Nilai presisi untuk kelas 0 sebesar 0,92 dan untuk kelas 1 sebesar 0,82. Nilai *recall* untuk kelas 0 sebesar 0,80 dan untuk kelas 1 sebesar 0,93. Nilai *F1-Score* untuk kelas 0 sebesar 0,86 dan untuk kelas 1 sebesar 0,87.
4. Model dengan nilai metrik kinerja terendah adalah model A.1. Model ini untuk dataset terkalibasi dengan jumlah *layers* sebanyak 3 dan jumlah *nodes* sebanyak 10, 5, dan 1. Nilai akurasi sebesar 0,63. Nilai presisi untuk kelas 0 sebesar 0,59 dan untuk kelas 1 sebesar 0,75. Nilai *recall* untuk kelas 0 sebesar 0,87 dan untuk kelas 1 sebesar 0,39. Nilai *F1-Score* untuk kelas 0 sebesar 0,7 dan untuk kelas 1 sebesar 0,51. Berbagai model lainnya memiliki kualitas kinerja yang baik dengan nilai metrik kinerja yang tinggi, namun tidak seimbang.

5. Kinerja model TinyML dipengaruhi oleh susunan model, terkhusus pada model Artificial Neural Network, berupa jumlah *layers*, *nodes* dan fungsi aktivasi yang digunakan. Dataset yang digunakan juga mempengaruhi kualitas kinerja model TinyML.
6. Model TinyML dengan kinerja yang baik, efektif, dan optimal untuk identifikasi radionuklida pemancar radiasi gamma berupa model dengan nilai kinerja yang tinggi dan seimbang. Model dengan kualitas kinerja yang baik, efektif, dan optimal adalah model A.3 dengan 3 *layers* serta *nodes* sebanyak 30, 15, dan 1. Nilai metrik kinerja model terdiri dari nilai akurasi keseluruhan sebesar 0,74, nilai presisi kelas 0 sebesar 0,73 dan kelas 1 sebesar 0,74, nilai *recall* kelas 0 sebesar 0,75 dan kelas 1 sebesar 0,73, dan nilai *F1-Score* kelas 0 dan kelas 1 sebesar 0,74.

5.2 Saran

Penelitian Tugas Akhir ini dapat dikembangkan sebagai berikut.

1. Menggunakan jenis algoritma ML yang lain, seperti *Random Forest*, *Naive-Bayes*, *Decision Tree*, dan *XGBoost*.
2. Menggunakan perangkat MCU yang lain seperti Arduino atau Raspberry Pi.
3. Menggunakan dan menambah jenis radionuklida yang lain, seperti Na-22, Am-241, Eu-152, dan Ba-133.
4. Mengintegrasikan MCU dengan perangkat detektor dan modul spektroskopi gamma secara langsung.

DAFTAR PUSTAKA

- Abadade, Y., Temouden, A., Bamoumen, H., Benamar, N., Chtouki, Y., & Hafid, A. S. (2023). A Comprehensive Survey on TinyML. *IEEE Access*, 11, 96892–96922. <https://doi.org/10.1109/ACCESS.2023.3294111>
- Altayeb, M., Zennaro, M., & Pietrosemoli, E. (2023). TinyML Gamma Radiation Classifier. *Nuclear Engineering and Technology*, 55(2), 443–451. <https://doi.org/10.1016/J.NET.2022.09.032>
- Arduino. (2025). *Programming - Arduino Documentation*. <https://docs.arduino.cc/programming/> Diakses pada 27 April 2025.
- A. D. Rasamoelina, F. Adjailia and P. Sinčák, (2020). A Review of Activation Function for Artificial Neural Network. *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, Herlany, Slovakia, 2020, pp. 281-286, doi: 10.1109/SAMI48414.2020.9108717.
- Banerjee, R. (2023). *Hands-on TinyML*. BPB Online, London.
- Buchtela, K. (2014). RADIOCHEMICAL METHODS | Gamma-Ray Spectrometry. Reference Module in Chemistry, Molecular Sciences and Chemical Engineering. doi:10.1016/b978-0-12-409547-2.11333-2
- Chicho, B. T., & Bibo Sallow, A. . (2021). A Comprehensive Survey of Deep Learning Models Based on Keras Framework. *Journal of Soft Computing and Data Mining*, 2(2), 49-62. <https://publisher.uthm.edu.my/ojs/index.php/jscdm/article/view/8732>
- Dudek, G. (2018). Generating Random Weights and Biases in Feedforward Neural Networks with Random Hidden Nodes. *Information Sciences*. doi:10.1016/j.ins.2018.12.063
- Espressif Systems. (2016). *ESP32-WROOM-32 Datasheet*

Gammaspectacular. *Gama* *Spectrum* *Database.*

<https://www.gammaspectacular.com/blue/gamma-spectra>

Hasan, H. A. (2014). Studying the risks of natural radioactivity in Tobacco and Cigarettes Smoke. *Thesis for Master of Science.* <http://dx.doi.org/10.13140/RG.2.2.18522.95680>

Hong, C. K., Abu, M. A., Shapiai, M. I., Haniff, M. F., Mohamad, R. S., & Abu, A. (2023). Analysis of Wind Speed Prediction using Artificial Neural Network and Multiple Linear Regression Model using Tinyml on Esp32. *Journal of Advanced Research in Fluid Mechanics and Thermal Sciences*, 107(1), 29–44. <https://doi.org/10.37934/ARFMTS.107.1.2944>

IAEA. (2013). Neutron Dosimetry and Monitoring : Active Methods of Neutron Detection. [PowerPoint Slides]. <https://www.slideshare.net/slideshow/active-methods-of-neutron-detection/16950943>

Ismail, M. A. (2017). Nuclear gauge application in road industry. *IOP Conference Series: Materials Science and Engineering*, 271, 012087. <https://doi.org/10.1088/1757-899X/271/1/012087>

Istofa, I., Prajitno, P., & Putu Susila, I. (2023). A Systematic Literature Review of TinyML for Environmental Radiation Monitoring System. *The 6th Mechanical Engineering, Science and Technology International Conference (MEST 2022)*, 461–473. https://doi.org/10.2991/978-94-6463-134-0_44

Jati, Bayu Rukmana. (2020). Perbandingan akurasi prediksi dan kecepatan proses antar classifier machine learning untuk klasifikasi spektrum gamma ^{60}Co , ^{22}Na , ^{241}Am , ^{137}Cs , ^{90}Sr (Skripsi, Fakultas Sains dan Teknologi Universitas Islam Negeri Syarif Hidayatullah Jakarta).

L'annunziata, M. F. (2003). Nuclear Radiation, Its Interaction With Matter And Radioisotope Decay. *Handbook of Radioactivity Analysis*, 1–121. doi:10.1016/b978-012436603-9/50006-5

Python Software Foundation. (2025). *Python 3.13.5 Documentation*.

<https://docs.python.org/3/>

Qamar, Roheen & Zardari, Baqar. (2023). Artificial Neural Networks: An Overview.

Mesopotamian *Journal of Computer Science*. 2023. 130-139.
10.58496/MJCSC/2023/015.

Qiao, C.-K., Wei, J.-W., & Chen, L. (2021). An Overview of the Compton Scattering Calculation. *Crystals*, 11(5), 525. <https://doi.org/10.3390/crust11050525>

RadiaCode. *Spectrum Isotopes Library*. <https://www.radiacode.com/spectrum-isotopes-library>. Diakses pada 21 Januari 2025.

Radiation Sensors. (2010). 6S6P1.5VD SCINTILLATION DETECTOR USER'S MANUAL

Rashid, N. (2018). Gamma Ray Spectroscopy of Co-60 Radioactive Source.

Rawool-Sullivan, M., Bounds, J., Brumby, S., Prasad, L., & Sullivan, J. (2010). Steps toward automated gamma ray spectroscopy. *Los Alamos National Laboratory*.

Ray, P. P. (2022). A review on TinyML: State-of-the-art and prospects. In *Journal of King Saud University - Computer and Information Sciences* (Vol. 34, Issue 4, pp. 1595–1623). King Saud bin Abdulaziz University. <https://doi.org/10.1016/j.jksuci.2021.11.019>

Romo, J. R., Nelson, K. T., Monterial, M., Nelson, K. E., Labov, S. E., & Hecht, A. (2021). Classifier Comparison for Radionuclide Identification from Gamma-ray Spectra (No. LLNL-PROC-825096). Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States).

Rovai, M. J. (2024). *Embedded ML (TinyML) Intro & Applications* [PowerPoint slides]. Workshop on TinyML for Sustainable Development. https://tinyml.seas.harvard.edu/SustainableDev-24/assets/slides/03-rovai-TinyML_Introduction.pdf

Salsabila. Shafa, (2021), Uji Kualitas Sistem Spektrometer Gamma UCS30 di Laboratorium Instrumentasi Nuklir STTN-BATAN Yogyakarta, Tugas Akhir Sarjana Terapan Teknik, Prodi Elins, Jurusan Teknophysika Nuklir, STTN-BATAN, Yogyakarta

Sarno, R., Sabilla, S. I., Malikahah, Purbawa, D. P., Ardani, M. S. H. (2023). *Machine Learning dan Deep Learning - Konsep dan Pemrograman Python*. Penerbit Andi, Yogyakarta.

Shultis, J.K., Faw, R.E., (2017), Fundamentals of Nuclear Science and Engineering, CBC Press, Boca Raton

Stefanus, R. (2019) Conventional Programming VS Machine Learning.
<https://rstefanus16.medium.com/conventional-programming-vs-machine-learning-a3b7b3425531>

Spectrum Techniques LLC. *Spectrum Techniques UCS-30 Manual*.

Tharwat, A. (2021), "Classification assessment methods", Applied Computing and Informatics, Vol. 17 No. 1, pp. 168-192. <https://doi.org/10.1016/j.aci.2018.08.003>

Tsoukas, V., Gkogkidis, A., Boumpa, E., & Kakarountas, A. (2024). A Review on the emerging technology of TinyML. *ACM Computing Surveys*, 56(10).
<https://doi.org/10.1145/3661820>

Tsoulfanidis, N., Landsberger, S., (2015), Measurement & Detection of Radiation, CBC Press, Boca Raton

Ukaegbu, I. K., & Gamage, K. A. A. (2018). A Model for Remote Depth Estimation of Buried Radioactive Wastes Using CdZnTe Detector. *Sensors*, 18(5), 1612. <https://doi.org/10.3390/s18051612s>

Wardana, I. N. K., Fahmy, S. A., & Gardner, J. W. (2024). TinyML with Meta-Learning on Microcontrollers for Air Pollution Prediction. *Eurosensors 2023*, 163. <https://doi.org/10.3390/proceedings2024097163>

Warden, P., Situnayake, D. (2020). TinyML-Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers. O'Reilly Media, Inc., Sebastopol.

Zehtabvar, M., Taghandiki, K., Madani, N., Sardari, D., & Bashiri, B. (2024). A Review on the Application of Machine Learning in Gamma Spectroscopy: Challenges and Opportunities. *Spectroscopy Journal*, 2(3), 123-144.
<https://doi.org/10.3390/spectroscj2030008>

Ziliwu, J. R., Setyawan, G. C., & Budiati, H. (2024). Penerapan ESP32-CAM dan TinyML dalam Klasifikasi Gambar Buah dan Sayuran. *Jutisi : Jurnal Ilmiah Teknik Informatika Dan Sistem Informasi*, 13(1), 584. <https://doi.org/10.35889/jutisi.v13i1.1869>

Matplotlib Development Team. (2025). Matplotlib 3.10.3 Documentation.
<https://matplotlib.org/stable/index.html>. Diakses pada 21 April 2025

NumFOCUS, Inc. (2025). pandas documentation. <https://pandas.pydata.org/docs/>

NumPy Developers. (2025). NumPy Documentation. <https://numpy.org/doc/stable/>

TensorFlow Team. (2025). API Documentation | TensorFlow v2.16.1.
https://www.tensorflow.org/api_docs

Keras Team. (2025). Keras 3 API documentation. <https://keras.io/api/>

scikit-learn. (2025). scikit-learn 1.7.0 documentation. <https://scikit-learn.org/stable/index.html>

LAMPIRAN 1 PROGRAM PENGOLAHAN DATASET

Pengolahan dataset dilakukan dengan menggunakan bahasa Python, dijalankan dengan Visual Studio Code dalam format Jupyter Notebook. Setiap blok program dijalankan satu per satu dan berurutan.

```
# Import library untuk pengolahan data
import matplotlib.pyplot as plt
import pandas as pd
import numpy as
```

Pada sel kode di atas, dilakukan import tiga library utama yang sering digunakan dalam pengolahan data dan visualisasi di Python, yaitu:

- `matplotlib.pyplot` (disingkat `plt`): Digunakan untuk membuat berbagai jenis grafik dan visualisasi data.
- `pandas` (disingkat `pd`): Library yang sangat populer untuk manipulasi dan analisis data berbasis tabel (`DataFrame`).
- `numpy` (disingkat `np`): Digunakan untuk komputasi numerik, terutama operasi pada array dan fungsi matematika.

Ketiga library ini akan digunakan secara berulang pada notebook ini untuk proses analisis dan visualisasi data.

```
# import csv data directly from same-directory of this notebook
data1 = 'co60_code11_t60.csv'
df1 = pd.read_csv(data1)

df1.info()
df1.head() # Print the first few rows of the DataFrame
```

Hasil

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1024 entries, 0 to 1023
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   Chan     1024 non-null   int64  
 1   Energy   1024 non-null   float64 
 2   Counts   1024 non-null   int64  
dtypes: float64(1), int64(2)
memory usage: 24.1 KB

      # Chan          # Energy        # Counts
0           0            -88.74          0
1           1            -86.18          0
2           2            -83.61          0
3           3            -81.05          0
```

Pada sel kode di atas, dilakukan langkah-langkah berikut:

- Mengimpor file CSV co60_code11_t60.csv ke dalam DataFrame pandas dengan nama df1.
- Menampilkan informasi struktur DataFrame menggunakan info(), seperti jumlah baris, kolom, dan tipe data setiap kolom.
- Menampilkan beberapa baris pertama dari DataFrame menggunakan head(100) untuk melihat data secara sekilas.

Langkah ini penting untuk memastikan data berhasil dimuat dan memahami struktur serta isi data sebelum dilakukan analisis lebih lanjut.

Program yang sama juga digunakan untuk data-data yang lain.

```
# import csv data directly from same-directory of this notebook
data2 = 'co60_code11_t100.csv'
df2 = pd.read_csv(data2)

df2.info()
df2.head() # Print the first few rows of the DataFrame
```

Hasil

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1024 entries, 0 to 1023
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   Chan     1024 non-null   int64  
 1   Energy   1024 non-null   float64 
 2   Counts   1024 non-null   int64  
dtypes: float64(1), int64(2)
memory usage: 24.1 KB

      # Chan          # Energy        # Counts
0           0            -88.74         0
1           1            -86.18         0
```

```
# import csv data directly from same-directory of this notebook
data3 = 'co60_code12_t60.csv'
df3 = pd.read_csv(data3)

df3.info()
df3.head(100) # Print the first few rows of the DataFrame
```

Hasil

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1024 entries, 0 to 1023
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   Chan     1024 non-null   int64  
 1   Energy   1024 non-null   float64 
 2   Counts   1024 non-null   int64  
dtypes: float64(1), int64(2)
memory usage: 24.1 KB

      # Chan          # Energy        # Counts
0           0            -88.74         0
1           1            -86.18         0
2           2            -83.61         0
3           3            -81.05         0
```

```
# import csv data directly from same-directory of this notebook
data4 = 'co60_code12_t100.csv'
df4 = pd.read_csv(data4)
df4.info()
df4.head(100) # Print the first few rows of the DataFrame
```

Hasil

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1024 entries, 0 to 1023
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   Chan     1024 non-null   int64  
 1   Energy   1024 non-null   float64 
 2   Counts   1024 non-null   int64  
dtypes: float64(1), int64(2)
memory usage: 24.1 KB

      # Chan          # Energy        # Counts
0           0            -88.74          0
1           1            -86.18          0
2           2            -83.61          0
```

```
# Combine dataframes df1, df2, df3, and df4 into a single DataFrame
co_df = pd.concat([df1, df2, df3, df4], ignore_index=True)

# Display info and first few rows of the combined DataFrame
co_df.info()
co_df.head(100)
```

Hasil

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4096 entries, 0 to 4095
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   Chan     4096 non-null   int64  
 1   Energy   4096 non-null   float64 
 2   Counts   4096 non-null   int64  
dtypes: float64(1), int64(2)
memory usage: 96.1 KB

      # Chan          # Energy        # Counts
0           0            -88.74          0
1           1            -86.18          0
2           2            -83.61          0
3           3            -81.05          0
```

Penggabungan DataFrame Co-60

Pada sel kode di atas, dilakukan penggabungan empat DataFrame (df1, df2, df3, dan df4) yang berisi data Co-60 menjadi satu DataFrame baru bernama co_df menggunakan

fungsi pd.concat(). Penggabungan ini dilakukan secara vertikal (baris demi baris) dengan opsi ignore_index=True agar indeks pada DataFrame hasil penggabungan di-reset.

Setelah penggabungan, informasi struktur DataFrame (co_df.info()) dan 100 baris pertama data (co_df.head(100)) ditampilkan untuk memastikan data sudah tergabung dengan benar dan untuk melihat sekilas isi data yang akan digunakan pada analisis selanjutnya.

```
# add one column as label "Co-60" or '0'

co_df['Label'] = '0'
co_df['Label'] = co_df['Label'].astype('int64')           # Convert the
'Label' column to integer type

co_df.info()
co_df.head(100)
```

Hasil

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4096 entries, 0 to 4095
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   Chan     4096 non-null   int64  
 1   Energy   4096 non-null   float64 
 2   Counts   4096 non-null   int64  
 3   Label    4096 non-null   int64  
dtypes: float64(1), int64(3)
memory usage: 128.1 KB
```

	# Chan	# Energy	# Counts	# Label
0	0	-88.74	0	0
1	1	-86.18	0	0
2	2	-83.61	0	0
3	3	-81.05	0	0
4	4	-78.48	0	0
5	5	-75.92	0	0
6	6	-73.35	0	0
7	7	-70.79	0	0
8	8	-68.23	0	0
9	9	-65.66	0	0

Menambahkan Kolom Label pada DataFrame Co-60

Pada sel kode di atas, dilakukan penambahan kolom baru bernama Label pada DataFrame co_df untuk memberikan label khusus pada data Co-60, yaitu 0. Langkah-langkah yang dilakukan adalah:

- Menambahkan kolom Label dengan nilai awal '0' untuk seluruh baris.
- Mengubah tipe data kolom Label menjadi int64 agar sesuai dengan kebutuhan analisis data numerik.
- Menampilkan struktur DataFrame menggunakan info() untuk memastikan kolom baru sudah ditambahkan dan tipe datanya benar.
- Menampilkan 100 baris pertama data menggunakan head(100) untuk melihat hasil penambahan kolom label.

Penambahan label ini penting agar data Co-60 dapat dibedakan dengan data lain pada proses analisis atau pelatihan model selanjutnya.

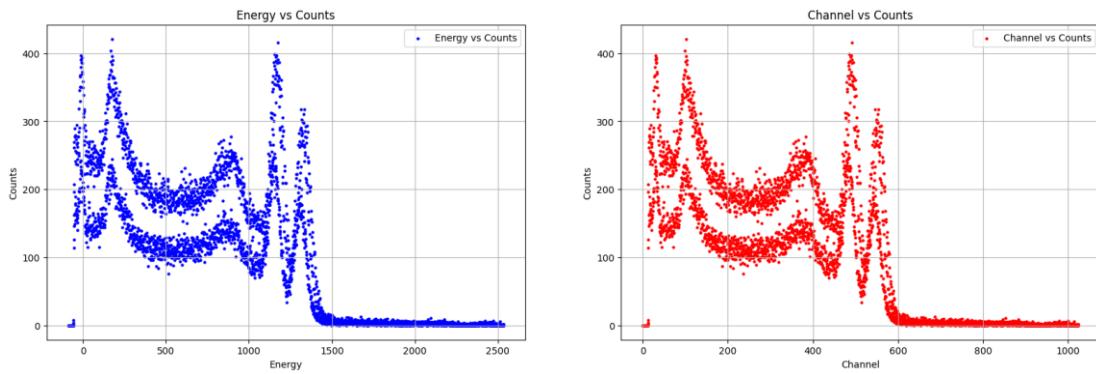
```
# Scatter plot for Energy vs Counts
plt.figure(figsize=(20, 6))

# Scatter plot for Energy vs Counts
plt.subplot(1, 2, 1) # 1 row, 2 columns, 1st subplot
plt.scatter(co_df['Energy'], co_df['Counts'], s=5, c='blue',
label='Energy vs Counts')
plt.xlabel('Energy')
plt.ylabel('Counts')
plt.title('Energy vs Counts')
plt.grid(True)
plt.legend()

# Scatter plot for Channel vs Counts
plt.subplot(1, 2, 2) # 1 row, 2 columns, 2nd subplot
plt.scatter(co_df['Chan'], co_df['Counts'], s=5, c='red',
label='Channel vs Counts')
plt.xlabel('Channel')
plt.ylabel('Counts')
plt.title('Channel vs Counts')
plt.grid(True)
plt.legend()

plt.show()
```

Hasil



Visualisasi Data Co-60

Pada sel kode di atas, dilakukan visualisasi data Co-60 yang telah digabungkan dalam DataFrame `co_df`. Dua scatter plot dibuat untuk memperlihatkan hubungan antara variabel-variabel penting:

1. Energy vs *Counts*:

- Sumbu x merepresentasikan nilai Energy, sedangkan sumbu y menunjukkan *Counts*.
- Setiap titik pada plot menggambarkan jumlah hitungan (*Counts*) pada energi tertentu.
- Plot ini membantu mengidentifikasi puncak-puncak energi pada spektrum Co-60.

2. Channel vs *Counts*:

- Sumbu x merepresentasikan *Channel*, sedangkan sumbu y menunjukkan *Counts*.
- Plot ini memperlihatkan distribusi hitungan berdasarkan *channel* detektor.
- Berguna untuk melihat pola distribusi *counts* pada rentang *channel* tertentu.

Penggunaan dua subplot dalam satu figure memudahkan perbandingan visual antara hubungan Energy-*Counts* dan Channel-*Counts*. Fitur grid, label sumbu, judul, dan legenda ditambahkan untuk memperjelas informasi pada masing-masing plot.

```

data5 = 'cs137_code16_t60.csv'
df5 = pd.read_csv(data5)

df5.info()
df5.head(100) # Print the first few rows of the DataFrame

```

Hasil

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1024 entries, 0 to 1023
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   Chan     1024 non-null   int64  
 1   Energy   1024 non-null   float64 
 2   Counts   1024 non-null   int64  
dtypes: float64(1), int64(2)
memory usage: 24.1 KB

      # Chan          # Energy        # Counts
0           0            -88.74         0
1           1            -86.18         0
2           2            -83.61         0
3           3            -81.05         0
4           4            -78.48         0

```

Import dan Pratinjau Data Cs-137 Code 16 T60

Pada sel kode di atas, dilakukan langkah-langkah berikut:

- Mendefinisikan nama file CSV cs137_code16_t60.csv ke dalam variabel data5.
- Membaca file CSV tersebut ke dalam DataFrame pandas dengan nama df5 menggunakan pd.read_csv().
- Menampilkan informasi struktur DataFrame menggunakan info(), seperti jumlah baris, kolom, dan tipe data setiap kolom.
- Menampilkan 100 baris pertama dari DataFrame menggunakan head(100) untuk melihat data secara sekilas.

Langkah ini penting untuk memastikan data berhasil dimuat dan memahami struktur serta isi data sebelum dilakukan analisis lebih lanjut.

```

data6 = 'cs137_code16_t100.csv'
df6 = pd.read_csv(data6)

df6.info()
df6.head(100) # Print the first few rows of the DataFrame

```

Hasil

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1024 entries, 0 to 1023
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   Chan     1024 non-null   int64  
 1   Energy   1024 non-null   float64 
 2   Counts   1024 non-null   int64  
dtypes: float64(1), int64(2)
memory usage: 24.1 KB

      # Chan          # Energy        # Counts
0           0            -88.74          0
1           1            -86.18          0
2           2            -83.61          0

```

```

data7 = 'cs137_code19_t60.csv'
df7 = pd.read_csv(data7)

df7.info()
df7.head(100) # Print the first few rows of the DataFrame

```

Hasil

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1024 entries, 0 to 1023
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   Chan     1024 non-null   int64  
 1   Energy   1024 non-null   float64 
 2   Counts   1024 non-null   int64  
dtypes: float64(1), int64(2)
memory usage: 24.1 KB

      # Chan          # Energy        # Counts
0           0            -88.74          0
1           1            -86.18          0
2           2            -83.61          0
3           3            -81.05          0

```

```
data8 = 'cs137_code19_t100.csv'
df8 = pd.read_csv(data8)

df8.info()
df8.head(100) # Print the first few rows of the DataFrame
```

Hasil

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1024 entries, 0 to 1023
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 #   Column  Non-Null Count  Dtype  
--- 
 0   Chan    1024 non-null   int64  
 1   Energy  1024 non-null   float64 
 2   Counts  1024 non-null   int64  
dtypes: float64(1), int64(2)
memory usage: 24.1 KB

      # Chan          # Energy        # Counts
0           0            -88.74         0
1           1            -86.18         0
2           2            -83.61         0
3           3            -81.05         0
4           4            -78.48         0
```

```
# Combine dataframes df1, df2, df3, and df4 into a single DataFrame
cs_df = pd.concat([df5, df6, df7, df8], ignore_index=True)

# Display info and first few rows of the combined DataFrame
cs_df.info()
cs_df.head(100)
```

Hasil

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4096 entries, 0 to 4095
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   Chan     4096 non-null   int64  
 1   Energy   4096 non-null   float64 
 2   Counts   4096 non-null   int64  
dtypes: float64(1), int64(2)
memory usage: 96.1 KB

      # Chan          # Energy        # Counts
0           0            -88.74          0
1           1            -86.18          0
2           2            -83.61          0
3           3            -81.05          0

```

Penggabungan DataFrame Cs-137

Pada sel kode di atas, dilakukan penggabungan empat DataFrame ('df5', 'df6', 'df7', dan 'df8') yang berisi data Cs-137 menjadi satu DataFrame baru bernama 'cs_df' menggunakan fungsi 'pd.concat()'. Penggabungan ini dilakukan secara vertikal (baris demi baris) dengan opsi 'ignore_index=True' agar indeks pada DataFrame hasil penggabungan di-reset.

Setelah penggabungan, informasi struktur DataFrame ('cs_df.info()') dan 100 baris pertama data ('cs_df.head(100)') ditampilkan untuk memastikan data sudah tergabung dengan benar dan untuk melihat sekilas isi data yang akan digunakan pada analisis selanjutnya.

```

# add one column as label "Cs-137"

cs_df['Label'] = '1'
cs_df['Label'] = cs_df['Label'].astype(np.int64) # Convert the 'Label'
column to int64 type

cs_df.info()
cs_df.head(100)

```

Hasil

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4096 entries, 0 to 4095
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   Chan    4096 non-null   int64  
 1   Energy   4096 non-null   float64 
 2   Counts   4096 non-null   int64  
 3   Label    4096 non-null   int64  
dtypes: float64(1), int64(3)
memory usage: 128.1 KB

      # Chan          # Energy        # Counts       # Label
0      0             -88.74         0              1
1      1             -86.18         0              1
2      2             -83.61         0              1
3      3             -81.05         0              1
4      4             -78.48         0              1
5      5             -75.92         0              1
```

Menambahkan Kolom Label pada DataFrame Cs-137

Pada sel kode di atas, dilakukan penambahan kolom baru bernama Label pada DataFrame cs_df untuk memberikan label khusus pada data Cs-137, yaitu 1. Langkah-langkah yang dilakukan adalah:

- Menambahkan kolom Label dengan nilai awal '1' untuk seluruh baris.
- Mengubah tipe data kolom Label menjadi int64 menggunakan fungsi astype(np.int64)
- agar sesuai dengan kebutuhan analisis data numerik.
- Menampilkan struktur DataFrame menggunakan info() untuk memastikan kolom baru sudah ditambahkan dan tipe datanya benar.
- Menampilkan 100 baris pertama data menggunakan head(100) untuk melihat hasil penambahan kolom label.

Penambahan label ini penting agar data Cs-137 dapat dibedakan dengan data lain pada proses analisis atau pelatihan model selanjutnya.

```
# Scatter plot for Energy vs Counts
plt.figure(figsize=(20, 6))

# Scatter plot for Energy vs Counts
```

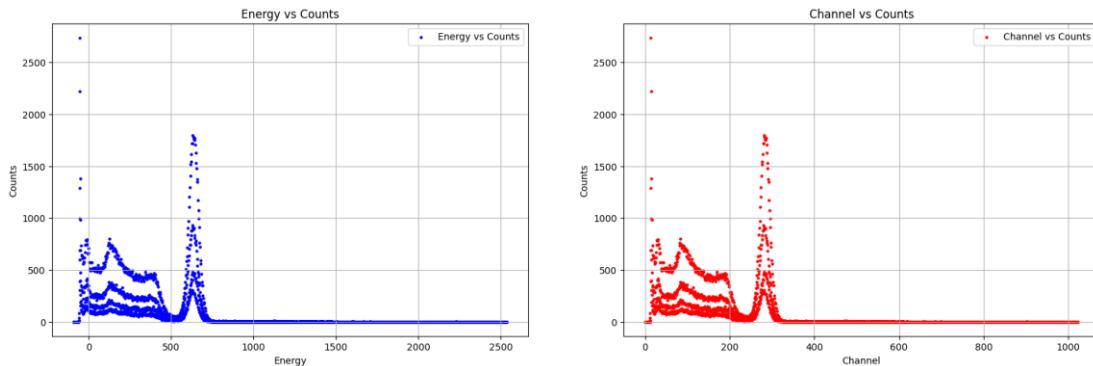
```

plt.subplot(1, 2, 1) # 1 row, 2 columns, 1st subplot
plt.scatter(cs_df['Energy'], cs_df['Counts'], s=5, c='blue',
label='Energy vs Counts')
plt.xlabel('Energy')
plt.ylabel('Counts')
plt.title('Energy vs Counts')
plt.grid(True)
plt.legend()

# Scatter plot for Channel vs Counts
plt.subplot(1, 2, 2) # 1 row, 2 columns, 2nd subplot
plt.scatter(cs_df['Chan'], cs_df['Counts'], s=5, c='red',
label='Channel vs Counts')
plt.xlabel('Channel')
plt.ylabel('Counts')
plt.title('Channel vs Counts')
plt.grid(True)
plt.legend()
plt.show()

```

Hasil



Visualisasi Data Cs-137

Pada sel kode di atas, dilakukan visualisasi data Cs-137 yang telah digabungkan dalam DataFrame `cs_df`. Dua scatter plot dibuat untuk memperlihatkan hubungan antara variabel-variabel penting:

1. Energy vs *Counts*:

- Sumbu x merepresentasikan nilai Energy, sedangkan sumbu y menunjukkan *Counts*.

- Setiap titik pada plot menggambarkan jumlah hitungan (*Counts*) pada energi tertentu.
 - Plot ini membantu mengidentifikasi puncak-puncak energi pada spektrum Cs-137.
2. *Channel vs Counts:*
- Sumbu x merepresentasikan *Channel*, sedangkan sumbu y menunjukkan *Counts*.
 - Plot ini memperlihatkan distribusi hitungan berdasarkan *channel* detektor.
 - Berguna untuk melihat pola distribusi *counts* pada rentang *channel* tertentu.

Penggunaan dua subplot dalam satu figure memudahkan perbandingan visual antara hubungan Energy-*Counts* dan *Channel-Counts*. Fitur grid, label sumbu, judul, dan legenda ditambahkan untuk memperjelas informasi pada masing-masing plot.

```
all_df = pd.concat([co_df, cs_df], ignore_index=True)

# Display the combined DataFrame
all_df.info()
all_df.head(100)
```

Hasil

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8192 entries, 0 to 8191
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   Chan    8192 non-null   int64  
 1   Energy   8192 non-null   float64 
 2   Counts   8192 non-null   int64  
 3   Label    8192 non-null   int64  
dtypes: float64(1), int64(3)
memory usage: 256.1 KB
```

# Chan	# Energy	# Counts	# Label
0	0	-88.74	0
1	1	-86.18	0
2	2	-83.61	0
3	3	-81.05	0
4	4	-78.48	0

Penggabungan DataFrame Co-60 dan Cs-137

Pada sel kode di atas, dilakukan penggabungan dua DataFrame, yaitu co_df (berisi data Co-60) dan cs_df (berisi data Cs-137), menjadi satu DataFrame baru bernama all_df menggunakan fungsi pd.concat(). Penggabungan ini dilakukan secara vertikal (baris demi baris) dengan opsi ignore_index=True agar indeks pada DataFrame hasil penggabungan di-reset.

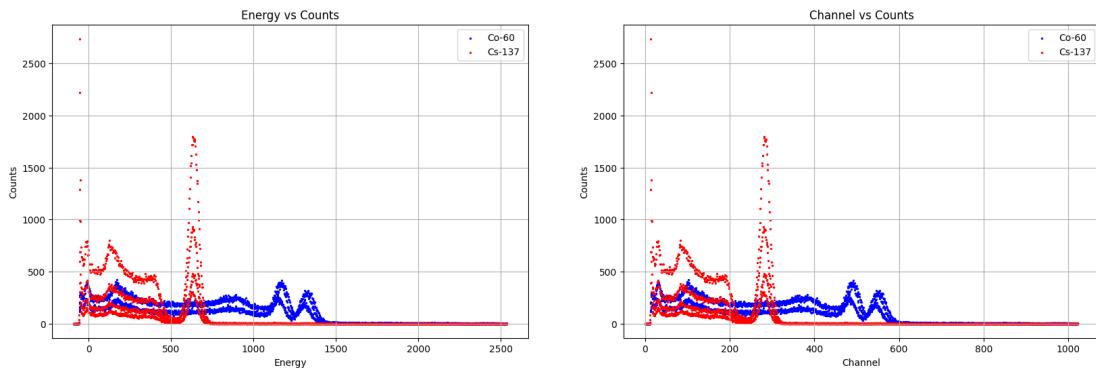
Setelah penggabungan, informasi struktur DataFrame (all_df.info()) dan 100 baris pertama data (all_df.head(100)) ditampilkan untuk memastikan data sudah tergabung dengan benar dan untuk melihat sekilas isi data yang akan digunakan pada analisis atau pelatihan model selanjutnya.

```
# Scatter plot for Energy vs Counts
plt.figure(figsize=(20, 6))

# Scatter plot for Energy vs Counts
plt.subplot(1, 2, 1) # 1 row, 2 columns, 1st subplot
plt.scatter(co_df['Energy'], co_df['Counts'], label='Co-60', s=2,
c='blue')
plt.scatter(cs_df['Energy'], cs_df['Counts'], label='Cs-137', s=2,
c='red')
plt.xlabel('Energy')
plt.ylabel('Counts')
plt.title('Energy vs Counts')
plt.grid(True)
plt.legend()

# Scatter plot for Channel vs Counts
plt.subplot(1, 2, 2) # 1 row, 2 columns, 2nd subplot
plt.scatter(co_df['Chan'], co_df['Counts'], label='Co-60', s=2,
c='blue')
plt.scatter(cs_df['Chan'], cs_df['Counts'], label='Cs-137', s=2,
c='red')
plt.xlabel('Channel')
plt.ylabel('Counts')
plt.title('Channel vs Counts')
plt.grid(True)
plt.legend()
plt.show()
```

Hasil



Visualisasi Gabungan Data Co-60 dan Cs-137

Pada sel kode di atas, dilakukan visualisasi gabungan antara data Co-60 (co_df) dan Cs-137 (cs_df) dalam satu figure menggunakan dua scatter plot:

1. Energy vs *Counts*:

- Sumbu x menunjukkan nilai Energy, sedangkan sumbu y menunjukkan *Counts*.
- Data Co-60 ditampilkan dengan warna biru, sedangkan Cs-137 dengan warna merah.
- Plot ini memudahkan perbandingan distribusi *counts* pada rentang energi antara kedua jenis data.

2. Channel vs *Counts*:

- Sumbu x menunjukkan Channel, sedangkan sumbu y menunjukkan *Counts*.
- Data Co-60 dan Cs-137 juga dibedakan berdasarkan warna.
- Plot ini memperlihatkan pola distribusi *counts* pada setiap *channel* untuk kedua jenis data.

Penggunaan dua subplot dalam satu figure memudahkan analisis visual perbedaan karakteristik antara data Co-60 dan Cs-137. Penambahan grid, label sumbu, judul, dan legenda bertujuan memperjelas informasi pada masing-masing plot.

```
# Save the DataFrame to a CSV file  
all_df.to_csv('data_training.csv', index=False)  
Menyimpan DataFrame Gabungan ke File CSV
```

Pada sel kode di atas, DataFrame gabungan all_df yang berisi data Co-60 dan Cs-137 disimpan ke dalam file CSV dengan nama data_training.csv menggunakan fungsi to_csv() dari pandas.

Opsi index=False digunakan agar indeks DataFrame tidak ikut disimpan ke dalam file. Penyimpanan ini bertujuan agar data hasil penggabungan dapat digunakan kembali untuk analisis atau pelatihan model di luar notebook ini tanpa perlu melakukan proses penggabungan ulang.

LAMPIRAN 2 PROGRAM PENYUSUNAN MODEL TINYML

Model TinyML disusun dengan menggunakan bahasa Python dan dijalankan dengan Visual Studio Code dalam format Jupyter Notebook. Setiap blok program dijalankan satu per satu dan berurutan.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import confusion_matrix
import seaborn as sns
import os
```

Penjelasan Kode

Pada sel kode di atas, dilakukan proses import berbagai library yang diperlukan untuk analisis data dan pembuatan model machine learning. Berikut penjelasan singkat masing-masing library:

1. pandas (pd): Digunakan untuk memproses dan menganalisis data dalam bentuk tabel (DataFrame).
2. numpy (np): Digunakan untuk operasi numerik dan manipulasi array.
3. matplotlib.pyplot (plt): Digunakan untuk membuat visualisasi data seperti grafik dan plot.
4. tensorflow (tf) dan keras: Digunakan untuk membangun dan melatih model deep learning.
5. sklearn.metrics.classification_report, accuracy_score, confusion_matrix: Untuk evaluasi performa model.
6. seaborn (sns): Untuk visualisasi data yang lebih interaktif dan informatif.
7. os: Untuk operasi terkait sistem file.

Import library ini merupakan langkah awal yang penting sebelum melakukan pemrosesan data, visualisasi, dan pembuatan model machine learning.

```
# Load dataset
data_train = pd.read_csv('data_training.csv')

data_train.info()
data_train.head()
```

Hasil

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8192 entries, 0 to 8191
Data columns (total 4 columns):
 #   Column Non-Null Count Dtype  
--- 
 0   Chan    8192 non-null   int64  
 1   Energy  8192 non-null   float64 
 2   Counts  8192 non-null   int64  
 3   Label   8192 non-null   int64  
dtypes: float64(1), int64(3)
memory usage: 256.1 KB

      # Chan          # Energy        # Counts        # Label      
0       0            -88.74         0             0
1       1            -86.18         0             0
2       2            -83.61         0             0
3       3            -81.05         0             0
4       4            -78.48         0             0
```

Penjelasan Kode

Pada sel kode di atas, dilakukan proses pemuatian (load) dataset pelatihan dari file data_training.csv menggunakan fungsi pd.read_csv(). Berikut penjelasan langkah-langkahnya:

1. Membaca Dataset:

data_train = pd.read_csv('data_training.csv') digunakan untuk membaca file CSV dan menyimpannya ke dalam sebuah DataFrame bernama data_train.

2. Menampilkan Informasi Dataset:

data_train.info() digunakan untuk menampilkan informasi ringkas mengenai dataset, seperti jumlah baris, kolom, tipe data setiap kolom, dan jumlah data yang tidak kosong.

3. Menampilkan Data Awal:

`data_train.head()` digunakan untuk menampilkan 5 baris pertama dari dataset, sehingga kita dapat melihat gambaran awal data yang dimiliki.

Langkah ini penting untuk memastikan data telah berhasil dimuat dengan benar serta memahami struktur dan isi data sebelum dilakukan proses analisis atau pemodelan lebih lanjut. Program yang sama juga digunakan untuk mengambil data csv lain.

Program yang sama juga digunakan untuk memuat dataset pengujian dan dataset validasi berikut.

```
data_test = pd.read_csv('data_testing.csv')

data_test.info()
data_test.head()
```

Hasil

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8192 entries, 0 to 8191
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   Chan     8192 non-null   int64  
 1   Energy   8192 non-null   float64 
 2   Counts   8192 non-null   int64  
 3   Label    8192 non-null   int64  
dtypes: float64(1), int64(3)
memory usage: 256.1 KB

      # Chan          # Energy         # Counts        # Label  
0       0             -55.4           0               0
1       1             -52.84          0               0
2       2             -50.27          0               0
3       3             -47.71          0               0
4       4             -45.15          0               0

data_val = pd.read_csv('data_validation.csv')

data_val.info()
data_val.head()
```

Hasil

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4096 entries, 0 to 4095
Data columns (total 4 columns):
 #   Column Non-Null Count Dtype  
--- 
 0   Chan    4096 non-null   int64  
 1   Energy  4096 non-null   float64 
 2   Counts  4096 non-null   int64  
 3   Label   4096 non-null   int64  
dtypes: float64(1), int64(3)
memory usage: 128.1 KB

```

	# Chan	# Energy	# Counts	# Label
0	0	-55.4	0	0
1	1	-52.84	0	0
2	2	-50.27	0	0
3	3	-47.71	0	0
4	4	-45.15	0	0

```

X_train = data_train.iloc[:, :-1].values
y_train = data_train.iloc[:, -1].values

X_test = data_test.iloc[:, :-1].values
y_test = data_test.iloc[:, -1].values

X_val = data_val.iloc[:, :-1].values
y_val = data_val.iloc[:, -1].values

```

Penjelasan Kode

Pada tiga sel kode di atas dilakukan proses pemisahan fitur (X) dan label (y) untuk tiga dataset berbeda, yaitu data pelatihan (data_train), data pengujian (data_test), dan data validasi (data_val). Berikut penjelasannya:

1. Pemisahan Fitur dan Label:

- X_train, X_test, dan X_val berisi data fitur, yaitu seluruh kolom kecuali kolom terakhir pada masing-masing dataset.
- y_train, y_test, dan y_val berisi label atau target, yaitu kolom terakhir pada masing-masing dataset. Label ini diasumsikan berupa nilai biner (0 atau 1).

2. Tujuan Pemisahan:

- Pemisahan ini penting agar model machine learning dapat dilatih menggunakan data fitur dan memprediksi label.
- Data pelatihan digunakan untuk melatih model, data validasi untuk mengevaluasi performa selama pelatihan, dan data pengujian untuk mengukur performa akhir model.

Langkah ini merupakan tahap awal yang krusial dalam proses pembuatan model machine learning, agar data terstruktur dengan baik sebelum masuk ke tahap pelatihan dan evaluasi.

```
print(X_train.shape, y_train.shape) # Check the shape of the data  
print(X_test.shape, y_test.shape) # Check the shape of the data  
print(X_val.shape, y_val.shape) # Check the shape of the data
```

```
(8192, 3) (8192,)
```

```
(8192, 3) (8192,)
```

```
(4096, 3) (4096,)
```

Penjelasan Kode

Pada tiga sel kode di atas, dilakukan pemeriksaan bentuk (shape) dari data fitur dan label untuk dataset pelatihan (X_{train} , y_{train}), pengujian (X_{test} , y_{test}), dan validasi (X_{val} , y_{val}). Berikut penjelasannya:

1. Pemeriksaan Shape Data:

- Fungsi `print(X_train.shape, y_train.shape)`, `print(X_test.shape, y_test.shape)`, dan `print(X_val.shape, y_val.shape)` digunakan untuk menampilkan jumlah baris (sample) dan kolom (fitur) pada masing-masing dataset.

- Hal ini penting untuk memastikan bahwa jumlah data fitur dan label sudah sesuai dan tidak terjadi ketidaksesuaian jumlah sample antara fitur dan label.
2. Tujuan:
- Memastikan data sudah dipisahkan dengan benar antara fitur dan label sebelum proses pelatihan model.
 - Memastikan tidak ada error terkait jumlah data saat proses training, validasi, maupun pengujian model.

Langkah ini merupakan bagian penting dari proses persiapan data sebelum membangun dan melatih model machine learning.

```
# Define the model
model = keras.Sequential([
    keras.layers.Dense(30, activation='relu',
input_shape=(X_train.shape[1],)),
    keras.layers.Dense(15, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])
```

Penjelasan Kode

Pada sel kode di atas, dilakukan proses pembuatan arsitektur model neural network menggunakan Keras dari TensorFlow. Berikut penjelasan langkah-langkahnya:

1. Inisialisasi Model:
 - `keras.Sequential()` digunakan untuk membuat model berurutan, di mana setiap *layer* ditambahkan secara berurutan.
2. *Layer* Input dan Hidden:
 - *Layer* pertama adalah `Dense` dengan 30 neuron dan fungsi aktivasi ReLU, serta menerima input sesuai jumlah fitur pada data pelatihan (`input_shape=(X_train.shape[1],)`).

- *Layer* kedua adalah Dense dengan 15 neuron dan fungsi aktivasi ReLU.
3. *Layer* Output:
- *Layer* terakhir adalah Dense dengan 1 neuron dan fungsi aktivasi sigmoid, yang cocok untuk tugas klasifikasi biner karena menghasilkan output berupa probabilitas antara 0 dan 1.

Arsitektur model ini dirancang untuk memproses data fitur dan melakukan klasifikasi ke dalam dua kelas (biner).

```
# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
```

Penjelasan Kode

Pada sel kode di atas, dilakukan proses kompilasi model neural network menggunakan fungsi

compile dari Keras. Berikut penjelasan langkah-langkahnya:

1. Optimizer:
 - optimizer='adam' berarti model akan menggunakan algoritma optimasi Adam, yang merupakan salah satu optimizer paling populer dan efisien untuk pelatihan neural net-work.
2. Loss Function:
 - loss='binary_crossentropy' menunjukkan bahwa fungsi loss yang digunakan adalah binary cross-entropy, yang cocok untuk tugas klasifikasi biner (dua kelas).
3. Metrics:

- metrics=['accuracy'] berarti selama proses pelatihan dan evaluasi, metrik yang akan dipantau adalah akurasi, yaitu persentase prediksi yang benar.

Proses komplilasi ini penting untuk mendefinisikan bagaimana model akan belajar dari data dan bagaimana performanya akan dievaluasi selama pelatihan.

```
# Train the model
history = model.fit(X_train, y_train, epochs=100, batch_size=32,
validation_data=(X_val, y_val))
```

Hasil

```
Epoch 1/100
256/256 - 4s - accuracy: 0.5584 - loss: 21.7736 - val_accuracy: 0.6201 - val_loss: 0.6972
Epoch 2/100
256/256 - 1s - 3ms/step - accuracy: 0.6051 - loss: 0.6867 - val_accuracy: 0.6511 - val_loss: 0.6718
Epoch 3/100
256/256 - 1s - 3ms/step - accuracy: 0.6589 - loss: 0.6469 - val_accuracy: 0.4976 - val_loss: 1.1108
Epoch 4/100
256/256 - 1s - 3ms/step - accuracy: 0.7000 - loss: 0.6317 - val_accuracy: 0.6392 - val_loss: 0.7164
Epoch 5/100
256/256 - 1s - 3ms/step - accuracy: 0.7017 - loss: 0.5845 - val_accuracy: 0.6704 - val_loss: 0.6265
Epoch 6/100
256/256 - 1s - 3ms/step - accuracy: 0.7027 - loss: 0.6160 - val_accuracy: 0.6475 - val_loss: 0.6326
```

Penjelasan Kode

Pada sel kode di atas, dilakukan proses pelatihan (training) model neural network menggunakan data pelatihan (`X_train`, `y_train`) dan data validasi (`X_val`, `y_val`). Berikut penjelasan langkah-langkahnya:

1. Pelatihan Model:

- `model.fit()` digunakan untuk melatih model dengan data yang telah dipersiapkan.
- Parameter `epochs=100` menunjukkan jumlah iterasi pelatihan penuh terhadap seluruh dataset.
- `batch_size=32` berarti data akan dibagi menjadi batch berukuran 32 sampel untuk setiap update bobot.

- `validation_data=(X_val, y_val)` digunakan untuk memantau performa model pada data validasi di setiap epoch, sehingga dapat menghindari overfitting.
2. Output History:

- Hasil pelatihan disimpan dalam variabel `history`, yang berisi informasi tentang perkembangan nilai loss dan akurasi selama proses pelatihan dan validasi.

Langkah ini sangat penting untuk membangun model yang mampu mengenali pola pada data pelatihan dan tetap memiliki performa baik pada data yang belum pernah dilihat (data validasi).

```
# Evaluate the model
test_loss, test_acc = model.evaluate(X_test, y_test)
print('Test accuracy:', test_acc)
```

Hasil

```
Test accuracy: 0.7396240234375
```

Penjelasan Kode

Pada sel kode di atas, dilakukan evaluasi model menggunakan data uji (`X_test` dan `y_test`). Berikut penjelasan langkah-langkahnya:

1. Evaluasi Model:

- `test_loss, test_acc = model.evaluate(X_test, y_test)` digunakan untuk mengevaluasi performa model yang telah dilatih pada data uji. Fungsi ini mengembalikan dua nilai, yaitu nilai loss (kerugian) dan akurasi pada data uji.

2. Menampilkan Akurasi:

- `print('Test accuracy:', test_acc)` digunakan untuk menampilkan nilai akurasi model pada data uji. Nilai ini menunjukkan seberapa baik model dalam mengklasifikasi data yang belum pernah dilihat sebelumnya.

Langkah ini penting untuk mengetahui kemampuan generalisasi model terhadap data baru dan memastikan bahwa model tidak hanya bekerja baik pada data pelatihan saja.

```
# Make predictions
y_pred = model.predict(X_test)
y_pred = (y_pred > 0.5).astype(int) # Convert probabilities to binary
predictions
```

Penjelasan Kode

Pada sel kode di atas, dilakukan proses prediksi menggunakan model yang telah dilatih terhadap data uji (`X_test`). Berikut penjelasan langkah-langkahnya:

1. Melakukan Prediksi:

- `y_pred = model.predict(X_test)` menghasilkan prediksi berupa probabilitas untuk setiap sampel pada data uji. Nilai probabilitas ini menunjukkan kemungkinan setiap sampel termasuk ke kelas 1.

2. Konversi Probabilitas ke Kelas Biner:

- `y_pred = (y_pred > 0.5).astype(int)` mengubah nilai probabilitas menjadi label kelas biner (0 atau 1). Jika probabilitas lebih dari 0.5, maka dikategorikan sebagai kelas 1, jika tidak maka kelas 0.

Langkah ini penting untuk mengetahui kemampuan generalisasi model terhadap data baru dan memastikan bahwa model tidak hanya bekerja baik pada data pelatihan saja.

```
# Print classification report
print(classification_report(y_test, y_pred))
```

Hasil

	precision	recall	f1-score	support
0	0.73	0.75	0.74	4096
1	0.74	0.73	0.74	4096
accuracy			0.74	8192
macro avg	0.74	0.74	0.74	8192
weighted avg	0.74	0.74	0.74	8192

Penjelasan Kode

Pada sel kode di atas, dilakukan evaluasi performa model menggunakan metrik classification report dari scikit-learn. Berikut penjelasan langkah-langkahnya:

1. Classification Report:

- Fungsi `classification_report(y_test, y_pred)` menghasilkan laporan yang berisi metrik evaluasi seperti *precision*, *recall*, *F1-Score*, dan *support* untuk masing-masing kelas (0 dan 1).
- Precision menunjukkan proporsi prediksi positif yang benar.
- *Recall* menunjukkan proporsi data positif yang berhasil teridentifikasi dengan benar.
- *F1-Score* merupakan rata-rata harmonis dari *precision* dan *recall*.
- Support menunjukkan jumlah sampel aktual untuk setiap kelas pada data uji.

2. Tujuan:

- Laporan ini memberikan gambaran menyeluruh tentang performa model pada masing- masing kelas, sehingga dapat diketahui apakah model bekerja baik secara seimbang atau terdapat bias pada salah satu kelas.

Langkah ini penting untuk memastikan model tidak hanya memiliki akurasi tinggi, tetapi juga performa yang baik pada setiap kelas, terutama jika data tidak seimbang.

```
# Generate the Confusion matrix
cm = confusion_matrix(y_test, y_pred)
```

```

# Plot the Confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Class 0', 'Class 1'], yticklabels=['Class 0', 'Class 1'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion matrix')
plt.show()

```

Hasil



Penjelasan Kode

Pada sel kode di atas, dilakukan pembuatan dan visualisasi *Confusion matrix* untuk mengevaluasi performa model pada data uji. Berikut penjelasan langkah-langkahnya.

1. Membuat *Confusion matrix*:

- Fungsi `confusion_matrix(y_test, y_pred)` digunakan untuk menghitung *Confusion matrix* berdasarkan label sebenarnya (`y_test`) dan hasil prediksi

model (`y_pred`). Confusion matrix menunjukkan jumlah prediksi benar dan salah untuk masing-masing kelas.

2. Visualisasi *Confusion matrix*:

- Menggunakan `sns.heatmap()`, *Confusion matrix* divisualisasikan dalam bentuk heatmap agar lebih mudah dianalisis. Nilai pada setiap sel ditampilkan secara jelas dengan parameter `annot=True`.
- Label sumbu x dan y diatur agar sesuai dengan kelas yang diprediksi dan aktual, serta diberikan judul untuk memperjelas isi plot.

3. Tujuan:

- *Confusion matrix* membantu dalam mengidentifikasi jenis kesalahan yang dilakukan model, seperti false positive dan false negative, sehingga dapat digunakan untuk analisis lebih lanjut dan perbaikan model.

Langkah ini penting untuk memahami detail performa model, tidak hanya dari segi akurasi, tetapi juga distribusi prediksi pada masing-masing kelas.

```
# Save the model as a Keras file
model.save('model.keras')
```

Penjelasan Kode

Pada sel kode di atas, dilakukan proses penyimpanan model Keras yang telah dilatih ke dalam file dengan format .keras. Berikut penjelasan langkah-langkahnya:

1. Menyimpan Model:

- `model.save('model.keras')` digunakan untuk menyimpan seluruh arsitektur, bobot, dan konfigurasi pelatihan model ke dalam satu file dengan ekstensi .keras.
- File ini dapat digunakan untuk memuat kembali model di masa mendatang tanpa perlu melatih ulang dari awal.

2. Tujuan:

- Penyimpanan model sangat penting agar hasil pelatihan tidak hilang dan dapat digunakan kembali untuk prediksi atau deployment di lingkungan lain.

Langkah ini merupakan bagian penting dalam workflow machine learning, terutama untuk keperluan deployment, sharing, atau backup model

```
# Convert the model to TensorFlow Lite format
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the TensorFlow Lite model to a file
with open('model.tflite', 'wb') as f:
    f.write(tflite_model)

print("Model has been converted to TensorFlow Lite format and saved as
'model.tflite'.")
```

Penjelasan Kode

Pada sel kode di atas, dilakukan proses konversi model Keras yang telah dilatih ke dalam format TensorFlow Lite dan penyimpanan hasil konversi tersebut ke dalam file. Berikut penjelasan langkah-langkahnya:

1. Konversi Model ke TensorFlow Lite:

- converter = tf.lite.TFLiteConverter.from_keras_model(model) digunakan untuk membuat objek konverter TensorFlow Lite dari model Keras yang sudah dilatih.
- tflite_model = converter.convert() melakukan proses konversi model Keras menjadi format TensorFlow Lite yang lebih ringan dan efisien untuk digunakan pada perangkat mobile atau embedded.

2. Menyimpan Model TensorFlow Lite:

- with open('model.tflite', 'wb') as f: f.write(tflite_model) menyimpan model hasil konversi ke dalam file bernama model.tflite dalam format biner.
3. Informasi Keberhasilan:
- print("Model has been converted to TensorFlow Lite format and saved as 'model.tflite'.") memberikan informasi bahwa proses konversi dan penyimpanan model telah berhasil dilakukan.

Langkah ini penting untuk mempersiapkan model agar dapat digunakan pada perangkat dengan sumber daya terbatas, seperti smartphone, IoT, atau mikrokontroler.

```
xxd -i model.tflite > model.h      # konvert program to .h header  
file
```

Penjelasan Kode

Pada sel kode di atas, dilakukan proses konversi file model TensorFlow Lite (model.tflite) menjadi file sumber kode header (model.h) menggunakan perintah xxd -i. Berikut penjelasan langkah-langkahnya:

1. Konversi ke File Header (.h):
 - Perintah xxd -i model.tflite > model.h juga mengubah file yang sama menjadi file header C, yang biasanya digunakan untuk deklarasi array model pada proyek embedded.
2. Penggunaan di Terminal:
 - Perintah ini dijalankan pada terminal Git Bash di Visual Studio Code, sehingga file hasil konversi dapat langsung digunakan dalam pengembangan aplikasi berbasis C/C++ atau pada perangkat embedded.

Langkah ini penting untuk integrasi model machine learning ke dalam aplikasi embedded, seperti pada microcontroller atau perangkat IoT, yang membutuhkan model dalam format array C.

LAMPIRAN 3 PROGRAM PENGUJIAN MODEL TINYML BERBASIS PYTHON

Program Python untuk pengujian model TinyML ini digunakan untuk mengirim dataset ke perangkat MCU dan menerima data hasil pengolahan model TinyML dari MCU.

Program Python

```
1 import serial
2 import time
3 import csv
4
5 # Adjust this to match your ESP32's COM port
6 ser = serial.Serial('COM3', 115200) # Open
7 serial port at 115200 baud rate
8
9 # Open and read the CSV file
10 with open('co60_11_t150.csv', 'r') as file: # Open the CSV file
11     in read mode
12     reader = csv.reader(file) # Create
13     a CSV reader object
14     for row in reader: # Iterate
15         through each row in the CSV file
16         if len(row) != 3: # Check
17             if the row has exactly 3 columns
18                 print(f"Skipping invalid row: {row}") # Skip
19             rows that don't have 3 columns
20
21         continue # Skip
22 rows that don't have 3 columns
```

```

16     # Construct the line to send to ESP32
17     line = f"{row[0]},{row[1]},{row[2]}\n"                      # Create
18     a string from the row data
19     ser.write(line.encode())                                     # Send
20     the line to ESP32
21     time.sleep(0.05)                                         # small
22     delay to let ESP32 handle input
23
24     # Read and print ESP32 response
25
26     time.sleep(0.5)                                         # Wait
27     for ESP32 to process the command
28     while ser.in_waiting:                                    # Check
29     if there is data in the serial buffer
30     print(ser.readline().decode().strip())                   # Read
31     and print the response from ESP32
32
33     time.sleep(1)
34
35     # Tell ESP32 to show stored data
36     ser.write(b'RUN\n')                                      # Send
37     command to ESP32 to show stored data
38
39     # Read and print ESP32 response
40
41     time.sleep(0.5)                                         # Wait
42     for ESP32 to process the command
43     while ser.in_waiting:                                    # Check
44     if there is data in the serial buffer
45     print(ser.readline().decode().strip())

```

Penjelasan Program Python

- import serial: Mengimpor pustaka **serial**. Pustaka ini menyediakan alat untuk membuat, mengelola, dan berkomunikasi melalui koneksi port serial (misalnya, USB-to-serial yang digunakan oleh ESP32).
- import time: Mengimpor pustaka **time**. Pustaka ini digunakan untuk fungsi-fungsi terkait waktu, terutama untuk memberikan jeda (delay) dalam eksekusi program.
- import csv: Mengimpor pustaka **csv**. Pustaka ini sangat berguna untuk menyederhanakan proses membaca dan menulis file dengan format *Comma-Separated Values* (CSV).
- ser = serial.Serial('COM3', 115200): Baris ini adalah inti dari koneksi.
 - **'COM3'**: Ini adalah nama port di mana ESP32 Anda terdeteksi oleh sistem operasi Windows. Nama ini mungkin berbeda di komputer Anda (misalnya COM4, COM5, dll.) dan harus disesuaikan.
 - **115200**: Ini adalah *baud rate*, yaitu kecepatan transfer data. Nilai ini harus sama persis dengan *baud rate* yang diatur dalam kode program di ESP32 Anda agar komunikasi berjalan lancar.
 - Hasil koneksi yang berhasil disimpan dalam variabel **ser**.
- time.sleep(2): Program akan berhenti selama **2 detik**. Jeda ini penting untuk memberikan waktu pada ESP32 untuk melakukan reset atau stabilisasi setelah koneksi serial baru terjalin.
- with open('co60_13_90s.csv', 'r') as file:: Membuka file co60_13_90s.csv dalam mode baca ('r'). Penggunaan with memastikan file akan ditutup secara otomatis setelah selesai digunakan.
- reader = csv.reader(file): Membuat sebuah objek "pembaca" yang akan memproses file CSV, memisahkan setiap baris menjadi daftar (list) kolom.
- for row in reader:: Memulai sebuah perulangan (*loop*) yang akan berjalan untuk setiap row (baris) yang ada di dalam file CSV.

- if len(row) != 3:: Melakukan validasi. Kode ini memeriksa apakah jumlah kolom di baris saat ini **tidak sama dengan 3**. Ini adalah langkah pengamanan untuk menghindari data yang formatnya salah.
- print(...) dan continue: Jika baris tidak valid, sebuah pesan akan dicetak di layar dan perintah continue akan membuat perulangan langsung melompat ke baris data berikutnya.
- line = f'{row[0]},{row[1]},{row[2]}\n': Jika baris valid, kode ini menggabungkan kembali data dari tiga kolom (row[0], row[1], row[2]) menjadi satu teks (string), dipisahkan oleh koma. Karakter \n (baris baru) ditambahkan di akhir, yang berfungsi sebagai sinyal "akhir transmisi" untuk baris ini bagi ESP32.
- ser.write(line.encode()): Mengirim data ke ESP32.
 - line.encode(): Mengubah format data dari teks (string) menjadi *bytes*, karena komunikasi serial bekerja dengan *bytes*.
 - ser.write(): Menuliskan *bytes* tersebut ke port serial untuk dikirim.
- time.sleep(0.05): Memberikan jeda singkat 0.05 detik antara pengiriman setiap baris agar ESP32 tidak kewalahan menerima data.
- time.sleep(0.5): Memberikan jeda 0.5 detik untuk memastikan ESP32 punya cukup waktu untuk memproses perintah dan menyiapkan respons.
- while ser.in_waiting:: Sebuah perulangan yang akan terus berjalan **selama** ada data yang menunggu untuk dibaca di buffer serial (ser.in_waiting berisi jumlah byte yang tersedia).
- print(ser.readline().decode().strip()): Baris ini melakukan tiga hal secara berurutan:
 1. ser.readline(): Membaca satu baris data (hingga karakter \n) dari ESP32 dalam format *bytes*.
 2. .decode(): Mengubah *bytes* tersebut kembali menjadi teks (string) yang bisa dibaca.

3. `.strip()`: Menghapus spasi atau karakter tak terlihat di awal dan akhir teks untuk merapikan hasil.
4. `print()`: Mencetak teks yang sudah bersih ke layar komputer.

LAMPIRAN 4 PROGRAM PENGUJIAN MODEL TINYML BERBASIS ARDUINO

```
1 #include <ArduTFLite.h>
2 #include "model.h" // replace with the actual model header name
3
4 // Adjust Tensor Arena size as needed based on your model
5 constexpr int kTensorArenaSize = 8000;
6 alignas(16) uint8_t tensor_arena[kTensorArenaSize];
7
8 #define MAX_ROWS 1025
9 #define NUM_COLS_IN 3
10 #define NUM_COLS_OUT 1
11
12 float inputData[MAX_ROWS][NUM_COLS_IN];
13 float outputData[MAX_ROWS][NUM_COLS_OUT];
14
15 int rowIndex = 0;
16 String inputLine = "";
17
18 void setup() {
19     Serial.begin(115200);
20     Serial.println("ESP32 is ready to receive float arrays...");
21
22     if (!modelInit(model, tensor_arena, kTensorArenaSize)) {
23         Serial.println("Model failed to initialize!");
24         while (true);
25     }
}
```

```

26 }
27
28 void loop() {
29     while (Serial.available()) {
30         char c = Serial.read();
31         if (c == '\n') {
32             inputLine.trim();
33             if (inputLine.length() > 0) {
34                 if (inputLine.equalsIgnoreCase("RUN")) {
35                     tinyMLinference();
36                     showOutput();
37                 }
38             } else {
39                 parseAndStoreRow(inputLine);
40             }
41         }
42         inputLine = "";
43     }
44     else {
45         inputLine += c;
46     }
47 }
48 }
49
50 // Parses a CSV row in the format "float1,float2,float3" and stores it
51 // in inputData
52 void parseAndStoreRow(String line) {

```

```

53  if (rowIndex >= MAX_ROWS) {
54      Serial.println("Array is full.");
55      return;
56  }
57
58  int firstComma = line.indexOf(',');
59  int secondComma = line.indexOf(',', firstComma + 1);
60
61  if (firstComma == -1 || secondComma == -1) {
62      Serial.println("Invalid input format. Expected:  
float1,float2,float3");
63      return;
64  }
65
66  float val1 = line.substring(0, firstComma).toFloat();
67  float val2 = line.substring(firstComma + 1, secondComma).toFloat();
68  float val3 = line.substring(secondComma + 1).toFloat();
69
70  inputData[rowIndex][0] = val1;
71  inputData[rowIndex][1] = val2;
72  inputData[rowIndex][2] = val3;
73
74  Serial.printf("Stored row %d: %.2f, %.2f, %.2f\n", rowIndex, val1,
75  val2, val3);
76  rowIndex++;
77
78 // Runs TinyML inference on each stored row and stores outputs in output
    arrays

```

```

79 void tinyMLinference() {
80     // int rowIndex = 0;
81
82     if (rowIndex == 0) {
83         Serial.println("No data to classify.");
84         return;
85     }
86
87     Serial.println("Running inference on stored data...");
88     // int countA = 0;
89     // int countB = 0;
90
91     // Process each row with the model.
92     for (int i = 0; i < rowIndex; i++) {
93         float output[2]; // Assuming the model outputs 2 scores for binary
94         classification
95         // modelRun() is provided by ArduTFLite; it copies inputData[i] into
96         // the model,
97         // runs inference, and populates the output array.
98
99         modelSetInput(inputData[i][0], 0); // First feature
100
101        modelSetInput(inputData[i][1], 1); // Second feature
102
103        modelSetInput(inputData[i][2], 2); // third feature
104
105        if (!modelRunInference()) {
106            Serial.printf("Inference failed at row %d\n", i);
107            return;
108        }
109    }
110
111    // Print the results
112    Serial.println("Classification results:");
113    for (int i = 0; i < countA; i++) {
114        Serial.print("A: ");
115        Serial.println(output[i]);
116    }
117    for (int i = 0; i < countB; i++) {
118        Serial.print("B: ");
119        Serial.println(output[i]);
120    }
121
122    // Clean up
123    modelDelete();
124
125    // Return to main menu
126    menu();
127}

```

```

106     float result = modelGetOutput(0);
107     int classification = result >= 0.5 ? 1 : 0; // Assuming threshold
108     0.5
109     outputData[i][0] = classification;
110 }
111
112 }
113
114 void showOutput() {
115
116     int count0 = 0;
117     int count1 = 0;
118
119     Serial.println("Input and Output each row :");
120     for (int i = 0; i < rowIndex; i++) {
121         Serial.printf("[%d] %.2f, %.2f, %.2f, %.2f \n", i, inputData[i][0],
122                     inputData[i][1], inputData[i][2], outputData[i][0]);
123
124         // Counts which is most frequent
125         if (outputData[i][0] == 0){
126             count0++;
127         } else if (outputData[i][0] == 1){
128             count1++;
129         }
130     }
131

```

```

132 Serial.println("The final result is ");
133 // Compare and display the most frequent value
134 if (count0 > count1) {
135     Serial.println("0 appears more frequently.");
136 }
137 else if (count1 > count0) {
138     Serial.println("1 appears more frequently.");
139 }
140
141 Serial.printf("Class 0 count: %d\n", count0);
142 Serial.printf("Class 1 count: %d\n", count1);
143 Serial.print("Most frequent class: ");
144 Serial.println((count0 > count1) ? "0" : "1");
145
146 }
```

Penjelasan Program Arduino

1. Inisialisasi dan Pustaka (Library)

Kode dimulai dengan menyertakan pustaka yang diperlukan dan mendefinisikan variabel global.

- `#include <ArduTFLite.h>`: Pustaka ini menyediakan fungsi-fungsi untuk menjalankan model TensorFlow Lite pada platform Arduino.
- `#include "model.h"`: Ini adalah file header yang berisi model machine learning yang telah dikonversi ke dalam format yang dapat dibaca oleh C/C++.
- `constexpr int kTensorArenaSize = 8000;;` Mendefinisikan ukuran memori (8000 byte) yang dialokasikan untuk operasi TensorFlow Lite. Ukuran ini harus cukup besar untuk menampung model dan data operasinya.

2. Variabel Global dan Buffer

Variabel ini digunakan di seluruh program untuk menyimpan data dan status.

- tensor_arena: Sebuah array byte yang merupakan area kerja atau "arena" memori untuk model TensorFlow Lite.
- MAX_ROWS: Menentukan jumlah maksimum baris data yang bisa ditampung, yaitu 1025 baris.
- NUM_COLS_IN: Menentukan bahwa setiap baris data masukan memiliki 3 kolom (3 nilai float).
- NUM_COLS_OUT: Menentukan bahwa setiap baris data keluaran memiliki 1 kolom.
- inputData & outputData: Array dua dimensi untuk menyimpan data masukan yang diterima dan data keluaran (hasil klasifikasi) dari model.
- rowIndex: Variabel untuk melacak jumlah baris data yang telah disimpan.

3. Fungsi setup()

Fungsi ini dieksekusi sekali saat ESP32 pertama kali dinyalakan atau di-reset.

- Serial.begin(115200): Memulai komunikasi serial pada kecepatan 115200 bps untuk mengirim dan menerima data (misalnya, melalui Serial Monitor di Arduino IDE).
- modelInit(...): Menginisialisasi model TensorFlow Lite. Jika inisialisasi gagal, program akan mencetak pesan kesalahan dan berhenti dalam satu loop tak terbatas (while (true)).

4. Fungsi loop()

Fungsi ini berjalan terus-menerus setelah setup() selesai. Ini adalah inti dari program yang menangani penerimaan data dan memicu proses inferensi.

- Fungsi ini terus memeriksa apakah ada data yang masuk melalui serial (Serial.available()).

- Setiap karakter yang diterima akan digabungkan ke dalam sebuah String bernama inputLine.
- Ketika karakter newline (\n) diterima, ini menandakan akhir dari satu baris masukan.
- Jika baris yang diterima adalah "RUN" (tidak peka huruf besar/kecil), maka program akan memanggil fungsi tinyMLinference() untuk menjalankan model dan showOutput() untuk menampilkan hasilnya.
- Jika baris tersebut bukan "RUN", program menganggapnya sebagai data dan memanggil parseAndStoreRow() untuk memproses dan menyimpannya.

5. Fungsi parseAndStoreRow()

Fungsi ini bertugas untuk mengurai baris data (berformat CSV) dan menyimpannya ke dalam array inputData.

- Fungsi ini pertama-tama memeriksa apakah array inputData sudah penuh.
- Ia mencari posisi koma untuk memisahkan tiga nilai float. Jika formatnya tidak valid, ia akan mencetak pesan kesalahan.
- Setelah berhasil diurai, ketiga nilai float tersebut disimpan ke dalam baris baru di array inputData.
- rowIndex kemudian dinaikkan untuk menunjuk ke baris kosong berikutnya.

6. Fungsi tinyMLinference()

Fungsi ini melakukan proses inferensi (prediksi) menggunakan model TinyML pada semua data yang telah tersimpan.

- Pertama, fungsi ini memastikan ada data yang bisa diproses.
- Kemudian, ia melakukan looping untuk setiap baris data yang ada di inputData.
- Untuk setiap baris, tiga nilai float ditetapkan sebagai masukan untuk model menggunakan modelSetInput().

- `modelRunInference()` dipanggil untuk menjalankan model pada data masukan tersebut. Jika gagal, pesan kesalahan akan ditampilkan.
- `modelGetOutput(0)` mengambil hasil prediksi dari model.
- Hasil ini kemudian dikonversi menjadi klasifikasi biner (0 atau 1) berdasarkan ambang batas 0.5, lalu disimpan ke dalam `outputData`.

7. Fungsi `showOutput()`

Fungsi ini menampilkan hasil akhir setelah semua proses inferensi selesai.

- Fungsi ini menginisialisasi penghitung untuk setiap kelas (kelas 0 dan kelas 1).
- Ia melakukan looping melalui semua hasil, mencetak data masukan asli beserta hasil klasifikasinya untuk setiap baris.
- Selama looping, ia juga menghitung jumlah kemunculan untuk setiap kelas (0 dan 1).
- Terakhir, ia membandingkan jumlah hitungan dan mencetak kelas mana yang paling sering muncul sebagai hasil akhir, beserta jumlah total untuk setiap kelas.

Alur Kerja Program

1. Mulai: ESP32 menyala, menginisialisasi komunikasi serial dan model TinyML.
2. Input Data: Pengguna mengirimkan beberapa baris data melalui Serial Monitor. Setiap baris berisi tiga angka float yang dipisahkan koma (contoh: 1.23,4.56,7.89) dan diakhiri dengan menekan Enter.
3. Penyimpanan Data: Program menerima setiap baris, mengurainya, dan menyimpannya dalam array `inputData`.
4. Menjalankan Inferensi: Setelah semua data dikirim, pengguna mengirimkan perintah RUN.
5. Proses Model: Program akan memproses setiap baris data yang tersimpan dengan model machine learning.

6. Output: Program menampilkan hasil klasifikasi (0 atau 1) untuk setiap baris data, dan diakhiri dengan ringkasan kelas mana yang paling dominan dari keseluruhan data.

BIOGRAFI PENULIS

David Irfan Jasir adalah mahasiswa Politeknik Teknologi Nuklir Indonesia angkatan 2021 dari Program Studi Elektronika Instrumentasi. Lahir di Kediri, 1 November 2002. Selama sebagai mahasiswa, cukup aktif dalam kegiatan akademik berupa mengikuti berbagai perlombaan karya tulis ilmiah, yaitu Lomba Karya Tulis Ilmiah Pekan Orientasi Ilmiah Fisika Prodi Pendidikan Fisika Universitas Bengkulu, Lomba Karya Tulis Ilmiah New and Renewable Energy Festival UKM Society of Renewable Energy Universitas Sumatera Utara, Lomba Esai Tingkat Mahasiswa Dies Natalis Fakultas Keguruan dan Ilmu Pendidikan Universitas Sanata Dharma, Lomba Esai Nasional Himpunan Mahasiswa Jurusan Matematika Fakultas Matematika dan Ilmu Pengatahan Alam Universitas Negeri Padang, dan Teknokrat Muda Indonesia Essay Competition. Beberapa proyek atau penelitian yang pernah diselesaikan diantaranya Perancangan Detektor Polusi Udara Portabel dan Pembuatan Program TinyML Identifikasi Radionuklida Pemancar Radiasi Gamma. Penulis dapat dihubungi melalui akun media sosial LinkedIn dengan nama yang sama.