

**TUGAS AKHIR**

***PROTEIN FUNCTION PREDICTION USING DEEP LEARNING***

|  |  |
| --- | --- |
| **12S20030** | **LAMSIHAR SIAHAAN** |
| **12S20046** | **PATRICIA DIAN MARGARETHA S** |
| **12S20047** | **RUTH CHRISTINE MANURUNG** |

**PROGRAM STUDI SARJANA SISTEM INFORMASI**

**FAKULTAS INFORMATIKA DAN TEKNIK ELEKTRO**

**INSTITUT TEKNOLOGI DEL**

**2024**



****

**TUGAS AKHIR**

***PROTEIN FUNCTION PREDICTION USING DEEP LEARNING***

**Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer (S.Kom)**

|  |  |
| --- | --- |
| **12S20030** | **LAMSIHAR SIAHAAN** |
| **12S20046** | **PATRICIA DIAN MARGARETHA S** |
| **12S20047** | **RUTH CHRISTINE MANURUNG** |

**PROGRAM STUDI SARJANA SISTEM INFORMASI**

**FAKULTAS INFORMATIKA DAN TEKNIK ELEKTRO**

**INSTITUT TEKNOLOGI DEL**

**2024**

# HALAMAN PERNYATAAN ORISINALITAS

Tugas Akhir ini adalah hasil karya Penulis sendiri, dan semua sumber baik yang dikutip maupun dirujuk telah Penulis nyatakan dengan benar.

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | Nama | : | Lamsihar Siahaan |
|  | NIM | : | 12S20030 |
|  | Tanda Tangan | : |  |
|  | Tanggal | : | 27 Juni 2024 |

|  |  |  |  |
| --- | --- | --- | --- |
| 2. | Nama | : | Patricia Dian Margaretha S |
|  | NIM | : | 12S20046 |
|  | Tanda Tangan | : |  |
|  | Tanggal | : | 27 Juni 2024 |

|  |  |  |  |
| --- | --- | --- | --- |
| 3. | Nama | : | Ruth Christine Manurung |
|  | NIM | : | 12S20047 |
|  | Tanda Tangan | : |  |
|  | Tanggal | : | 27 Juni 2024 |

# HALAMAN PENGESAHAN

Tugas Akhir ini diajukan oleh:

1. Nama : Lamsihar Siahaan

NIM : 12S20047

1. Nama : Patricia Dian Margaretha S

NIM : 12S20046

1. Nama : Ruth Christine Manurung

NIM : 12S20047

Program Studi : Sarjana Sistem Informasi

Judul Tugas Akhir : *Protein Function Prediction using Deep Learning*

**Telah berhasil dipertanggungjawabkan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Komputer (S.Kom) pada program studi Sarjana Sistem Informasi, Fakultas Informatika dan Teknik Elektro, Institut Teknologi Del.**

**DEWAN PENGUJI**

Pembimbing 1: Humasak Tomy Argo Simanjuntak, S.T, M.ISD ( )

Pembimbing 2: Susi Eva Maria Purba, S.Kom, M.Cs ( )

Penguji 1: Rosni Lumbantoruan, Ph.D ( )

Penguji 2: Sarah Rosdiana Tambunan, S.Kom, M.Cs ( )

Ditetapkan di : Laguboti

Tanggal : 27 Juni 2024

# HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai civitas akademik Institut Teknologi Del, kami yang bertanda tangan dibawah ini:

1. Nama : Lamsihar Siahaan

NIM : 12S20047

1. Nama : Patricia Dian Margaretha S

NIM : 12S20046

1. Nama : Ruth Christine Manurung

NIM : 12S20047

Program Studi : Sarjana Sistem Informasi

Fakultas : Informatika dan Teknik Elektro

Jenis Karya : Tugas Akhir

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Institut Teknologi Del **Hak Bebas Royalti Non-eksklusif (*Non-eexclusive Royalty Free Right*)** atas karya ilmiah penulis yang berjudul:

***Protein Function Prediction using Deep Learning***

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Institut Teknologi Del berhak menyimpan, mengalih/media format dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir kami selama tetap mencantumkan nama kami sebagai penulis/pencipta dana sebagai pemiliki Hak Cipta.

Demikian pernyataan ini kami buat dengan sebenarnya.

Ditetapkan di : Laguboti

Tanggal : 27 Juni 2024

Yang menyatakan

Lamsihar Siahaan Patricia Dian Margaretha S Ruth Christine Manurung

# KATA PENGANTAR

Puji syukur kepada Tuhan yang Maha Esa atas berkat dan rahmat-Nya, penulis dapat menyelesaikan Tugas Akhir ini sebagai syarat mencapai gelar Sarjana Komputer pada Program Studi Sistejm Informasi, Fakultas Informatika dan Teknik Elektro di Institut Teknologi Del. Selama pengerjaan Tugas Akhir ini, penulis mendapat bantuan, bimbingan, dan dukungan dari berbagai pihak. Oleh sebab itu, penulis mengucapkan terima kasih kepada:

1. Orang tua, saudara, dan keluarga dari ketiga anggota kelompok memberikan didikan, motivasi, moral, materi, dan nasihat demi kelancaran Tugas Akhir;
2. Bapak Humasak Tommy Argo Simanjuntak, ST, M.ISD, selaku dosen pembimbing I dan Ibu Susi Eva Maria Purba, S.Kom., M.Cs, selaku dosen pembimbing II yang telah menyediakan waktu, tenaga, pikiran, dan motivasi untuk mengarahkan penulis dalam penyusunan dan pengerjaan Tugas Akhir;
3. Ibu Rosni Lumbantoruan, Ph.D, selaku dosen penguji I dan Ibu Sarah Rosdiana Tambunan, S.Kom, M.Cs selaku dosen penguji II yang telah mengarahkan penulis dalam penyelesaian Tugas Akhir ini;
4. Teman-teman Sistem Informasi Angkatan 2020, rekan sekerja kelompok TASI-2324-101, dan diri sendiri yang sudah berjuang sampai tahap ini.

Akhir kata, penulis berharap Tuhan yang Maha Esa berkenan membalas segala kebaikan kepada semua pihak yang telah membantu. Semoga Tugas Akhir ini membawa manfaat bagi pengembangan ilmu pengetahuan.

Laguboti, 27 Juni 2024

Penulis

# ABSTRAK

Nama : Lamsihar Siahaan

Patricia Dian Margaretha S

Ruth Christine Manurung

Program Studi : Sarjana Sistem Informasi

Judul : *Protein Function Prediction using Deep Learning*

Protein memainkan peran krusial dalam berbagai proses biologis, mempengaruhi struktur, fungsi, dan aktivitas seluler. Dengan ribuan jenis protein yang terdiri dari sekuens asam amino yang berbeda, memahami hubungan antara sekuens asam amino dan fungsi protein sangat penting dalam pengembangan obat dan pemahaman penyakit genetik. Dalam era genetika dan biologi molekuler yang terus berkembang, pemahaman ini mendorong penggunaan berbagai metode prediksi fungsi protein, seperti model LSTM menggunakan *embedding* *layer* dari model atau *Protein Language Model* seperti *T5-Embedding* dan ProtBERT. Penelitian ini memanfaatkan pendekatan *Deep Learning*, khususnya *Long Short-Term Memory* (LSTM), untuk memprediksi fungsi protein berdasarkan urutan asam amino. LSTM, yang dirancang untuk data sekuensial, mampu mengingat informasi dari *input* sebelumnya, sehingga cocok untuk memproses urutan asam amino dalam protein. Misalnya, dalam prediksi fungsi protein, jika sekuens asam amino dominan mengandung *Leusin* (L), protein tersebut dapat berperan penting dalam pembentukan protein dan metabolisme otot. Data dari CAFA5 (*Critical Assessment of Functional Annotation*) digunakan untuk meningkatkan pemahaman mengenai fungsi protein melalui analisis sekuens asam amino yang belum dianotasi. Penelitian sebelumnya menunjukkan bahwa penerapan LSTM dengan *embedding layer* masih kurang memuaskan. Oleh karena itu, penelitian ini melakukan eksperimen menggunakan LSTM dengan *embedding* dari *Protein Language Model* seperti *T5-Embedding* dan ProtBERT, yang menghasilkan evaluasi model lebih baik dan prediksi fungsi protein lebih akurat. Hasil dari model *Deep Learning* ini diharapkan dapat meningkatkan akurasi prediksi fungsi protein, mendukung pengembangan terapi dan obat baru, serta memberikan wawasan mendalam mengenai mekanisme kerja protein. Upaya berkelanjutan dalam bioinformatika akan memperdalam pemahaman kita tentang kehidupan dan proses biologis yang mendasarinya.

**Kata Kunci:** prediksi fungsi protein, *Deep Learning*, *Long Short-Term Memory* (LSTM), CAFA5, bioinformatika

# 

# ABSTRACT

Names : Lamsihar Siahaan

Patricia Dian Margaretha S

Ruth Christine Manurung

Study Program : Bachelor of Information Systems

Title : Protein Function Prediction using Deep Learning

Proteins play a crucial role in various biological processes, influencing cellular structure, function, and activity. With thousands of protein types consisting of diverse amino acid sequences, understanding the relationship between these sequences and protein functions is vital for drug development and understanding genetic diseases. In the evolving fields of genetics and molecular biology, this understanding drives the use of various protein function prediction methods, such as LSTM models with *embedding* layers or Protein Language Models like T5-*Embedding* and ProtBERT. This research utilizes a Deep Learning approach, specifically Long Short-Term Memory (LSTM), to predict protein functions based on amino acid sequences. LSTM, designed for sequential data, can retain information from previous inputs, making it suitable for processing amino acid sequences in proteins. For instance, in predicting protein functions, if a sequence dominantly contains Leucine (L), the protein may play a significant role in protein synthesis and muscle metabolism. Data from CAFA5 (Critical Assessment of Functional Annotation) are used to enhance understanding of protein functions through the analysis of unannotated amino acid sequences. Previous studies have shown that using LSTM with *embedding* layers yielded unsatisfactory results. Therefore, this study experiments with LSTM using *embedding*s from Protein Language Models such as T5-*Embedding* and ProtBERT, resulting in better model evaluation and more accurate protein function predictions. The results are expected to improve prediction accuracy, support the development of new therapies and drugs, and provide deeper insights into protein mechanisms. Ongoing bioinformatics efforts will deepen our understanding of life and its underlying biological processes.

**Keywords:** protein function prediction, Deep Learning, Long Short-Term Memory (LSTM), CAFA5, bioinformatics

**DAFTAR ISI**

[**HALAMAN PERNYATAAN ORISINALITAS ii**](#_Toc172661444)

[**HALAMAN PENGESAHAN iii**](#_Toc172661445)

[**HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS iv**](#_Toc172661446)

[**KATA PENGANTAR vi**](#_Toc172661447)

[**ABSTRAK vii**](#_Toc172661448)

[**ABSTRACT viii**](#_Toc172661449)

[**BAB I PENDAHULUAN 1**](#_Toc172661450)

[1.1 Latar Belakang 1](#_Toc172661451)

[1.2 Pertanyaan Penelitian 4](#_Toc172661452)

[1.3 Tujuan Penelitian 4](#_Toc172661453)

[1.4 Ruang Lingkup 4](#_Toc172661454)

[1.5 Sistematika Penyajian 4](#_Toc172661455)

[**BAB II LANDASAN TEORI 6**](#_Toc172661456)

[2.1 Protein 6](#_Toc172661457)

[2.1.1 Pengertian Protein 6](#_Toc172661458)

[2.1.2 Struktur Protein 8](#_Toc172661459)

[2.1.3 Fungsi Protein 10](#_Toc172661460)

[2.2 *Gene Ontology* (GO) 12](#_Toc172661461)

[2.3 *Taksonomi* 17](#_Toc172661462)

[2.4 *Data Mining* 19](#_Toc172661463)

[2.4.1 CRISP-DM *(Cross Industry Standard Process for Data Mining)* 21](#_Toc172661464)

[2.4.2 *Data* *Preprocessing* 23](#_Toc172661465)

[2.5 *Embedding* pada *Tensorflow.Keras* 26](#_Toc172661466)

[2.6 *Protein Language Model* (PLM) 27](#_Toc172661467)

[2.6.1 *ProtBERT* 28](#_Toc172661468)

[2.6.2 T5 (*Text-to-text Transfer Transformer*) 30](#_Toc172661469)

[2.7 *Deep Learning* 32](#_Toc172661470)

[2.7.1 *RNN (Recurrent Neural Network)* 32](#_Toc172661471)

[2.7.2 LSTM *(Long Short Term Memory)* 33](#_Toc172661472)

[2.8 Pencarian Sekuens Protein 35](#_Toc172661473)

[2.8.1 *Diamond Search* 35](#_Toc172661474)

[2.8.2 *BLAST* (*Basic Local Alignment Search Tool*) 37](#_Toc172661475)

[2.9 Matriks Evaluasi Performansi 37](#_Toc172661476)

[2.10 Penelitian Terdahulu dalam Prediksi Fungsi Protein 38](#_Toc172661477)

[**BAB III ANALISIS 48**](#_Toc172661478)

[3.1 Tahapan Penelitian 48](#_Toc172661479)

[3.2 Analisis Data 50](#_Toc172661480)

[3.2.1 Deskripsi *Dataset* 50](#_Toc172661481)

[3.2.2 Persiapan Data 58](#_Toc172661482)

[3.2.3 Analisis Statistik 59](#_Toc172661483)

[3.3 Analisis *Data Preprocessing* 70](#_Toc172661484)

[3.3.1 Penanganan *Missing Values* 70](#_Toc172661485)

[3.3.2 Tokenisasi Sekuens Protein 71](#_Toc172661486)

[3.3.3 *Padding* pada Sekuens Protein 72](#_Toc172661487)

[3.4 Analisis Model LSTM 74](#_Toc172661488)

[3.5 Analisis *Embedding* 75](#_Toc172661489)

[3.5.1 Analisis *Layer Embedding* pada *Tensorflow.Keras* 76](#_Toc172661490)

[3.5.2 Analisis *T5-Embedding* 76](#_Toc172661491)

[3.5.3 Analisis *ProtBERT* 77](#_Toc172661492)

[3.8 AnalisisPencarian Sekuens Protein 78](#_Toc172661493)

[3.9 Analisis Metriks Evaluasi 79](#_Toc172661494)

[3.10 Analisis Nilai *Hyperparameter* Model 81](#_Toc172661495)

[3.11 Analisis Eksperimen 83](#_Toc172661496)

[3.11.1 Prediksi Fungsi Protein menggunakan *Embedding* dari *TensorFlow.Keras* dan Model LSTM 83](#_Toc172661497)

[3.11.2 Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model* yaitu *T5 Embeds*) dan Model LSTM 84](#_Toc172661498)

[3.11.3 Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model* yaitu *ProtBERT*) dan Model LSTM 86](#_Toc172661499)

[3.11.4 Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model yaitu ProtBERT*) dan Model LSTM + *Diamond Search* 86](#_Toc172661500)

[**BAB IV DESAIN 89**](#_Toc172661501)

[4.1 Desain Penelitian 89](#_Toc172661502)

[4.2 Desain *Embedding* 91](#_Toc172661503)

[4.2.1 Desain Penggunaan *Layer Embedding* dari *Tensorflow.Keras* dalam Pengolahan Sekuens Protein 92](#_Toc172661504)

[4.2.2 Desain *Protein Language Model* (PLM) pada *T5-Embedding* 94](#_Toc172661505)

[4.2.3 Desain *Protein Language Model* (PLM) pada *ProtBERT* 96](#_Toc172661506)

[4.3 Desain Model LSTM 99](#_Toc172661507)

[4.4 Desain Eksperimen 102](#_Toc172661508)

[4.4.1 Desain Eksperimen Prediksi Fungsi Protein menggunakan *Embedding* dari *TensorFlow.Keras* dan Model LSTM 102](#_Toc172661509)

[4.4.2 Desain Eksperimen Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model* yaitu *T5 Embedding*) dan Model LSTM 104](#_Toc172661510)

[4.4.3 Desain Eksperimen Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model* yaitu *ProtBERT*) dan Model LSTM 106](#_Toc172661511)

[4.4.4 Desain Eksperimen Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model yaitu ProtBERT*) dan Model LSTM + *Diamond Search* 109](#_Toc172661512)

[**BAB V IMPLEMENTASI 111**](#_Toc172661513)

[5.1 Lingkungan Implementasi 111](#_Toc172661514)

[5.2 Batasan Implementasi 112](#_Toc172661515)

[5.3 Implementasi *Preprocessing* 112](#_Toc172661516)

[5.3.1. Memeriksa *Missing Value* 113](#_Toc172661517)

[5.4 Implementasi Eksperimen 114](#_Toc172661518)

[5.4.1 Implementasi Prediksi Fungsi Protein menggunakan *Embedding* dari *TensorFlow.Keras* dan Model LSTM 114](#_Toc172661519)

[5.4.2 Implementasi Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model* yaitu *T5 Embedding*) dan Model LSTM 119](#_Toc172661520)

[5.4.3 Implementasi Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model* yaitu *ProtBERT*) dan Model LSTM 123](#_Toc172661521)

[5.4.4 Implementasi Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model yaitu ProtBERT*) dan Model LSTM + *Diamond Search* 128](#_Toc172661522)

[**BAB VI HASIL DAN PEMBAHASAN 142**](#_Toc172661523)

[6.1 Hasil Eksperimen Penelitian 142](#_Toc172661524)

[6.1.1 Hasil Prediksi Fungsi Protein menggunakan *Embedding* dari *TensorFlow.Keras* dan Model LSTM 142](#_Toc172661525)

[6.1.2 Hasil Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model* yaitu *T5 Embedding*) dan Model LSTM 144](#_Toc172661526)

[6.1.3 Hasil Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model* yaitu *ProtBERT*) dan Model LSTM 146](#_Toc172661527)

[6.1.4 Hasil Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model* yaitu *ProtBERT*) dan Model LSTM + *Diamond Search* 148](#_Toc172661528)

[6.2 Pembahasan Eksperimen Penelitian 150](#_Toc172661529)

[6.2.1 Pembahasan berdasarkan Nilai Metriks Evaluasi 150](#_Toc172661530)

[6.2.2 Pembahasan berdasarkan Hasil Prediksi Fungsi Protein 153](#_Toc172661531)

[**BAB VII KESIMPULAN DAN SARAN 182**](#_Toc172661532)

[7.1 Kesimpulan 182](#_Toc172661533)

[7.2 Saran 184](#_Toc172661534)

[**DAFTAR PUSTAKA 186**](#_Toc172661535)

[**LAMPIRAN 192**](#_Toc172661536)

**DAFTAR GAMBAR**

[Gambar 2.1 Sintesis Protein [11] 7](#_Toc172627208)

[Gambar 2.2 Struktur Protein [16] 9](#_Toc172627209)

[Gambar 2.3 Relasi Struktur Gene Ontology [23] 14](#_Toc172627210)

[Gambar 2.4 Struktur Grafik GO:0106126 [23] 16](#_Toc172627211)

[Gambar 2.5 Proses Knowledge Discovery in Database (KDD)[10] 20](#_Toc172627212)

[Gambar 2.6 Tahapan CRISP-DM [29] 21](#_Toc172627213)

[Gambar 2.7 Data Preparation[32] 24](#_Toc172627214)

[Gambar 2.8 Cara Kerja ProtBERT 28](#_Toc172627215)

[Gambar 2.9 T5 (Text-to-text Transfer Transformer)[37] 30](#_Toc172627216)

[Gambar 2.10 Struktur RNN (Recurrent Neural Network) 33](#_Toc172627217)

[Gambar 2.11 Struktur LSTM (Long Short Term Memory) [41] 34](#_Toc172627218)

[Gambar 2.12 Diamond Search 36](#_Toc172627219)

[Gambar 3.1 Tahapan Penelitian 48](#_Toc172627220)

[Gambar 3.2 Struktur Grafik Istilah GO:0044249 56](#_Toc172627221)

[Gambar 3.3 Visualisasi Distribusi Panjang Sekuens 63](#_Toc172627222)

[Gambar 3.4 Visualisasi Distribusi Asam Amino pada Sekuens Protein 64](#_Toc172627223)

[Gambar 3.5 Frekuensi Relatif Kelas Fungsi 66](#_Toc172627224)

[Gambar 3.6 Analisis Frekuensi Kelas GO\_Term 67](#_Toc172627225)

[Gambar 3.7 Analisis Frekuensi Taksonomi 68](#_Toc172627226)

[Gambar 4.1 Desain Penelitian 89](#_Toc172627227)

[Gambar 4.2 Desain Layer *Embedding* dari Tensorflow.Keras dalam Pengolahan Sekuens Protein 92](#_Toc172627228)

[Gambar 4.3 Desain *Embedding* T5-*Embedding* 94](#_Toc172627229)

[Gambar 4.4 Desain *Embedding* ProtBERT 97](#_Toc172627230)

[Gambar 4.5 Desain Model LSTM 99](#_Toc172627231)

[Gambar 4.6 Desain Eksperimen menggunakan *Embedding* dari Tensorflow.Keras dan LSTM 102](#_Toc172627232)

[Gambar 4.7 Desain Eksperimen menggunakan T5-*Embedding* dan LSTM 104](#_Toc172627233)

[Gambar 4.8 Desain Eksperimen menggunakan ProtBERT dan LSTM 107](#_Toc172627234)

[Gambar 4.9 Desain Eksperimen menggunakan *Embedding* ProtBERT dan LSTM + Diamond Search 109](#_Toc172627235)

[Gambar 6.1 Hasil Metriks Evaluasi Eksperimen 1 144](#_Toc172627236)

[Gambar 6.2 Hasil Metriks Evaluasi Eksperimen 2 146](#_Toc172627237)

[Gambar 6.3 Hasil Metriks Evaluasi Eksperimen 3 148](#_Toc172627238)

[Gambar 6.4 Grafik Perbandingan Metriks Evaluasi Eksperimen 151](#_Toc172627239)

[Gambar 6.5 Analisis Perbandingan Skor Probabilitas pada Protein\_ID 'Q61824' 176](#_Toc172627240)

[Gambar 6.6 Analisis Perbandingan Skor Probabilitas pada Protein\_ID 'P29618' 177](#_Toc172627241)

[Gambar 6.7 Analisis Perbandingan Skor Probabilitas pada Protein\_ID 'A2RSY6' 178](#_Toc172627242)

[Gambar 6.8 Analisis Perbandingan Skor Probabilitas pada Protein\_ID 'Q9JIX8' 179](#_Toc172627243)

[Gambar 6.9 Analisis Perbandingan Skor Probabilitas pada Protein\_ID 'QFF3W6' 180](#_Toc172627244)

**DAFTAR TABEL**

[Tabel 2.1 Penelitian Terdahulu dalam Prediksi Fungsi Protein 39](#_Toc172627245)

[Tabel 3.1 Simbol Asam Amino 50](#_Toc172627246)

[Tabel 3.2 *File train\_sekuens.fasta* 51](#_Toc172627247)

[Tabel 3.3 *File test\_superset.fasta* 52](#_Toc172627248)

[Tabel 3.4 *File train\_terms.tsv* 53](#_Toc172627249)

[Tabel 3.5 Contoh Isi *File train\_terms.tsv* 54](#_Toc172627250)

[Tabel 3.6 Contoh Isi *File go.obo* 55](#_Toc172627251)

[Tabel 3.7 *File testsuperset-taxon-list.tsv* 57](#_Toc172627252)

[Tabel 3.8 *File train\_taxonomy.tsv* 58](#_Toc172627253)

[Tabel 3.9 Statistik Sekuens Protein 60](#_Toc172627254)

[Tabel 3.10 Distribusi Asam Amino pada Panjang Sekuens 63](#_Toc172627255)

[Tabel 3.11 Deskripsi GO\_Term 67](#_Toc172627256)

[Tabel 3.12 Analisis Frekuensi Taksonomi 69](#_Toc172627257)

[Tabel 3.13 Penanganan Missing Values 70](#_Toc172627258)

[Tabel 3.14 Tokenisasi berbasis Karakter 72](#_Toc172627259)

[Tabel 3.15 Hasil Tokenisasi pada Sekuens Protein 72](#_Toc172627260)

[Tabel 3.16 *Padding* pada Sekuens Protein 73](#_Toc172627261)

[Tabel 3.17 Analisis Hyperparameter 81](#_Toc172627262)

[Tabel 5.1 Spesifikasi Perangkat Keras (Hardware) 111](#_Toc172627263)

[Tabel 5.2 Spesifikasi Perangkat Lunak (Software) 111](#_Toc172627264)

[Tabel 6.1 Hasil Prediksi Eksperimen 1 143](#_Toc172627265)

[Tabel 6.2 Hasil Prediksi Eksperimen 2 145](#_Toc172627266)

[Tabel 6.3 Hasil Prediksi Eksperimen 3 146](#_Toc172627267)

[Tabel 6.4 Hasil Prediksi Eksperimen 4 149](#_Toc172627268)

[Tabel 6.5 Pembahasan berdasarkan Nilai Metriks Evaluasi 150](#_Toc172627269)

[Tabel 6.6 Hasil Prediksi Fungsi Protein Eksperimen 1 154](#_Toc172627270)

[Tabel 6.7 Hasil Prediksi Fungsi Protein dengan Skor Probabilitas pada Eksperimen 1 158](#_Toc172627271)

[Tabel 6.8 Hasil Prediksi Fungsi Protein Eksperimen 2 160](#_Toc172627272)

[Tabel 6.9 Hasil Prediksi Fungsi Protein dengan Skor Probabilitas pada Eksperimen 2 162](#_Toc172627273)

[Tabel 6.10 Hasil Prediksi Fungsi Protein Eksperimen 3 164](#_Toc172627274)

[Tabel 6.11 Hasil Prediksi Fungsi Protein dengan Skor Probabilitas pada Eksperimen 3 167](#_Toc172627275)

[Tabel 6.12 Hasil matches.tsv 170](#_Toc172627276)

[Tabel 6.13 Hasil Combine ProtBERT +LSTM + Diamond Search 172](#_Toc172627277)

[Tabel 6.14 Analisis Perbandingan Hasil Prediksi dari Ketiga Metode *Embedding* 175](#_Toc172627278)

**DAFTAR LAMPIRAN**

[Lampiran 1. Analisis Frekuensi GO\_Term 174](#_Toc172193818)

# BAB I PENDAHULUAN

Bab pendahuluan menjelaskan terkait dengan latar belakang penelitian, pertanyaan penelitian, tujuan penelitian, ruang lingkup penelitian, serta tahapan yang dilakukan selama penelitian dan sistematika penyajian laporan penelitian tugas akhir.

## 1.1 Latar Belakang

Dalam era genetika dan biologi molekuler yang terus berkembang, pemahaman tentang fungsi protein menjadi penting [1]. Hal ini tidak hanya membuka peluang untuk penelitian obat-obatan baru, tetapi juga membantu penanggulangan penyakit genetik yang masih menjadi misteri di dunia medis. Penelitian mengenai fungsi protein dapat membantu mengidentifikasi jalur molekuler yang terlibat dalam proses biologis, seperti pertumbuhan sel, respons imun, dan penyembuhan luka. Informasi ini sangat penting untuk pengembangan terapi yang lebih efektif dan untuk meningkatkan pemahaman kita tentang kehidupan [2].

Dalam upaya memahami fungsi protein, penelitian mendalam dilakukan dengan fokus pada sekuens asam amino pembentuk protein. Metode ini tidak melibatkan proses sekuensing protein ulang, melainkan memanfaatkan data protein yang telah ada dari penelitian-penelitian sebelumnya. Informasi mengenai sekuens asam amino ini diintegrasikan ke dalam basis data protein yang luas dan terstruktur, membantu peneliti menghindari duplikasi dan memanfaatkan informasi yang sudah terverifikasi dari penelitian yang sudah ada sebelumnya. Dalam melakukan prediksi, urutan asam amino yang membentuk protein tersebut menjadi penentu apa fungsi dari protein tersebut. Salah satu metode yang digunakan adalah analisis homologi, dimana urutan asam amino protein baru dibandingkan dengan protein yang sudah diketahui fungsinya. Jika terdapat kemiripan yang signifikan, kemungkinan besar protein baru tersebut memiliki fungsi yang serupa.

Penelitian di CAFA (*Critical Assessment of Functional Annotation*) telah melibatkan berbagai model untuk memprediksi fungsi protein dengan menggunakan pendekatan konvensional hingga *deep learning* [3]. Salah satu model yang menonjol dalam penelitian konvensional adalah INGA, yang mendapatkan prediksi GO dengan menggabungkan data dari sumber yang berbeda [4]. Sementara itu, salah satu model yang menonjol dalam penelitian *deep learning* adalah DeepGOPlus [3]. DeepGOPlus dapat memprediksi fungsi protein berdasarkan urutan asam amino dari protein tersebut, dan setelah model DeepGOPlus dilatih ulang dengan data dari CAFA3, model ini menghasilkan prediksi yang akurat [5]. Model *deep learning* memiliki kemampuan untuk secara otomatis mengekstrak fitur-fitur kompleks dari data mentah dan membuat prediksi secara menyeluruh dari awal hingga akhir.

Berdasarkan penelitian DeepGoPlus yang mengembangkan metode untuk memprediksi fungsi protein dari urutan protein menggunakan *Convolutional Neural Network* (CNN) dan *similarity-based predictions*, ditemukan bahwa model LSTM tidak memberikan hasil yang memuaskan dalam konteks ini karena perbedaan jenis data yang digunakan [5]. Meskipun demikian, penelitian tersebut memberikan wawasan penting mengenai pentingnya pemilihan model yang tepat dalam tugas prediksi fungsi protein. Oleh karena itu, penelitian ini bertujuan untuk mengeksplorasi potensi penggunaan model *Long Short-Term Memory* (LSTM) berdasarkan temuan dari DeepGoPlus, dengan fokus pada penggunaan model LSTM untuk prediksi fungsi protein dari urutan protein. Sebelumnya, penelitian lain juga berhasil menerapkan model RNN dan LSTM untuk prediksi fungsi protein [6]. Dengan mempertimbangkan keunggulan dan keterbatasan model LSTM serta hasil dari penelitian sebelumnya, penelitian ini bertujuan untuk mengevaluasi dan mengoptimalkan model LSTM dalam memprediksi fungsi protein, dengan harapan dapat memberikan kontribusi baru dalam pengembangan metode yang lebih efektif dan akurat dalam prediksi fungsi protein.

Seiring berjalannya waktu, sekuens protein mengalami perubahan. Hal ini tergambar dari hasil penelitian CAFA, mulai dari CAFA1 hingga CAFA5. CAFA1, yang dilakukan pada tahun 2010-2011, bertujuan mengevaluasi metode anotasi otomatis fungsi protein dan kemajuan dalam pengembangan metode tersebut. Dalam CAFA2 (2013-2014), penelitian diperluas ke ontologi baru dengan mempertimbangkan jumlah ontologi, jumlah protein target, dan benchmark yang diperluas. CAFA3 (2017-2019) melibatkan penyaringan eksperimental untuk mengidentifikasi *gen* yang terkait dengan fungsi tertentu, dan penelitian ini juga menekankan pendekatan evaluasi terhadap pertumbuhan anotasi antar waktu [7].

Penelitian CAFA memberikan pemahaman mendalam tentang dinamika perubahan sekuens protein. Oleh karena itu, pada penelitian ini, peneliti akan melakukan prediksi fungsi protein dari sekuens yang belum dianotasi fungsinya di CAFA5, menyoroti pentingnya memperbaharui pengetahuan tentang fungsi protein. Hal ini penting karena sekuens protein dapat memberikan informasi mendalam mengenai fungsi biologis dan mekanisme kerja protein, yang sering kali berubah atau berkembang seiring ditemukannya data baru. Meskipun urutan asam amino dalam protein tampak sederhana, menentukan fungsinya tetap menjadi tantangan dalam biologi molekuler komputasi. Pada CAFA 5, fungsi protein diprediksi berdasarkan urutan asam amino sebagai *input*. LSTM (*Long Short-Term Memory*) dirancang untuk menangani data sekuensial dan dapat mengingat informasi dari *input* sebelumnya dalam urutan yang panjang, sehingga sangat cocok untuk memproses urutan asam amino dalam protein, di mana setiap asam amino mempengaruhi yang berikutnya dalam rangkaian.

Dengan demikian, organisasi CAFA, melalui iterasi seperti CAFA5, terus berperan dalam mengevaluasi kemajuan dalam prediksi fungsi protein, meningkatkan anotasi komputasi dari fungsi protein, dan memperkenalkan metode prediksi fungsi protein yang lebih akurat. Dalam penelitian ini, peneliti mengadopsi metode *deep learning* seperti LSTM, yang memberikan kemungkinan untuk memperdalam pemahaman dan meningkatkan akurasi prediksi fungsi protein. Melalui upaya berkelanjutan CAFA, dengan peningkatan evaluasi dan pengenalan metode prediksi yang lebih akurat, peneliti dapat memahami fungsi protein dengan lebih mendalam, memberikan kontribusi signifikan terhadap kemajuan dalam bidang biologi molekuler dan komputasi, serta mendukung pengembangan terapi dan obat-obatan baru.

## 1.2 Pertanyaan Penelitian

Pertanyaan penelitian dalam tugas akhir ini adalah sebagai berikut.

1. Bagaimana melakukan prediksi fungsi protein menggunakan model *deep learning*?
2. Bagaimana performansi metode prediksi fungsi protein dengan menggunakan model *deep learning*?
3. Bagaimana proses anotasi fungsi protein terhadap sekuens asam amino yang terdapat pada CAFA5 diimplementasikan dalam penelitian ini?

## 1.3 Tujuan Penelitian

Tujuan penelitian dalam tugas akhir ini adalah sebagai berikut.

1. Untuk mengembangkan model *deep learning* yang dapat memprediksi fungsi protein dengan menggunakan data sekuens protein.
2. Untuk mengukur dan membandingkan performansi model *deep learning* dalam memprediksi fungsi protein, yaitu evaluasi akurasi, *precision, recall,* dan *F1-score* dari model tersebut.
3. Untuk mengetahui proses implementasi dalam menganotasi fungsi protein terhadap sekuens asam amino yang terdapat pada CAFA5.

## 1.4 Ruang Lingkup

Ruang lingkup penelitian dalam tugas akhir ini, kami menggunakan *dataset* dari website *Kaggle* pada kompetisi *CAFA5*.

## 1.5 Sistematika Penyajian

Dokumen tugas akhir ini terdiri dari tujuh bab. Adapun ketujuh bab tersebut antara lain sebagai berikut.

1. Pada Bab 1 Pendahuluan, berisi latar belakang, pertanyaan tujuan, ruang lingkup, dan sistematika penulisan dokumen tugas akhir.
2. Pada Bab 2 Landasan Teori, berisi teori yang mendukung pengerjaan penelitian dan rangkuman penelitian yang relevan dengan topik penelitian.
3. Pada Bab 3 Metode Penelitian, berisi mengenai metode penelitian serta jadwal penelitian.
4. Pada Bab 4 Desain, berisi rancangan yang akan dilakukan berdasarkan analisis yang telah dibuat sebelumnya.
5. Pada Bab 5 Implementasi, berisi mengenai pengimplementasian algoritma, percobaan penelitian dan hasil percobaan.
6. Pada Bab 6 Hasil dan Pembahasan, berisi hasil akhir dari implementasi dan percobaan yang telah dilakukan dan membahas hasil prediksi yang diperoleh.
7. Pada Bab 7 Kesimpulan dan Saran, berisi penjelasan terkait kesimpulan yang diperoleh dari pelaksanaan penelitian ini apakah hasil yang diperoleh dari penelitian ini sesuai dengan keputusan yang diharapkan, dan saran yang dapat diberikan untuk penelitian tugas akhir yang dikerjakan.

# BAB II TINJAUAN PUSTAKA

Bab ini menjelaskan berbagai teori, metode, teknik, ataupun prosedur yang terkait dengan topik penelitian. Selain itu, terdapat juga kajian yang membahas mengenai hasil- hasil penelitian sebelumnya yang terkait dengan topik penelitian.

## 2.1 Protein

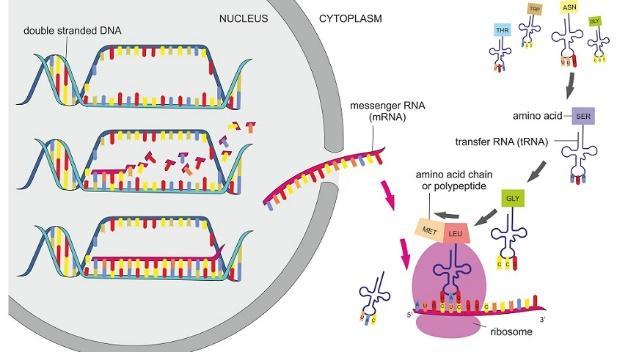
Pada sub bab ini akan dijelaskan pengertian dari protein, struktur dari protein, dan fungsi dari protein.

2.1.1 Pengertian Protein

Protein adalah kata yang berasal dari bahasa Yunani yaitu "*protos*" [8] yang berarti "yang paling utama”, merupakan senyawa organik kompleks yang memiliki molekul berat yang tinggi [9]. Protein terbentuk oleh *polimer* asam amino yang terhubung melalui ikatan *peptida*, dimana *polimer* adalah molekul besar yang terbentuk dari hasil penggabungan asam amino *(amino acids)* [9]. Protein merupakan makromolekul yang terdiri dari rantai asam amino, membentuk rantai *peptida* dengan panjang bervariasi, mulai dari *dipeptida* hingga *polipeptida* [10].

Untuk memperoleh protein di dalam tubuh, tubuh manusia melakukan serangkaian proses yang kompleks dan teratur yang dikenal sebagai sintesis protein. Dalam proses sintesisnya, DNA dan RNA dilibatkan dalam inti sel dan juga *ribosom*. DNA berfungsi sebagai penyimpan informasi genetik, sementara RNA berperan dalam mentransfer informasi tersebut untuk membentuk protein [11].

Proses sintesis ini dimulai dengan proses transkripsi, dimana informasi genetik dari DNA akan disalin menjadi RNA, sehingga namanya menjadi mRNA (*RNA messenger*). Oleh karena DNA tidak mampu bergerak sendiri ke *ribosom*, maka DNA tetap tinggal di dalam intisel dan mRNA akan bergerak mendatangi DNA kemudian menyalin kode genetik yang ada pada DNA dan membawa kode genetik tersebut menuju *ribosom*. Selanjutnya, kode genetik yang dibawa oleh mRNA tersebut dibaca oleh *ribosom*, dimana tahap ini disebut dengan tahapan translasi. Kode genetik tersebut kemudian dibaca atau ditranslasi menjadi urutan asam amino, dimana pada proses translasi ini melibatkan tRNA *(*RNA *transfer*). Urutan asam amino tersebut terdiri dari tiap *kodon* yang membentuknya, dimana dengan start *kodon* yaitu AUG (*adenin-urasil-guanin*) dan diakhiri dengan stop *kodon* yang terdiri dari tiga jenis, yaitu UAA *(urasil-adenin-adenin)*, UAG *(urasil-adenin-guanin)*, dan UGA (urasil-guanin-adenin). Ketika *ribosom* mencapai titik dari salah satu *kodon* ini, maka proses dari translasi sintesis protein akan otomatis berhenti dan asam amino akan dilepaskan sebagai protein yang sudah terbentuk [11]. Proses dari sintesis protein dapat dilihat pada Gambar 2.1 [11].



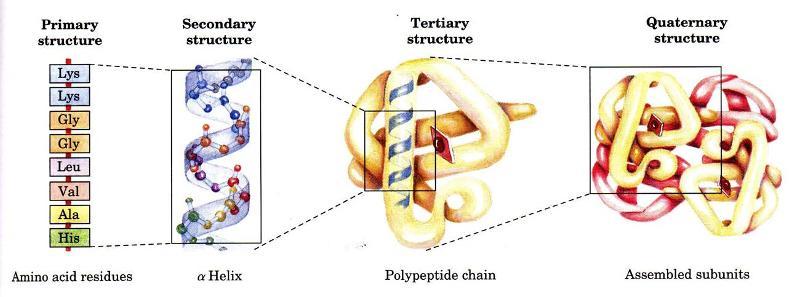
Gambar 2.1 Sintesis Protein [11]

Protein terdiri dari kombinasi 20 jenis asam amino, meskipun secara total terdapat lebih dari 300 variasi asam amino yang dapat ditemui di alam [12]. Jenis asam amino pembentuk protein tersebut adalah *alanin, arginin, asparagin, aspartat, glisin, glutamin, glutamat, histidin, isoleusin, leusin, lisin, metionin, fenilalanin, prolin, serin, sistein, treonin, tirosin, treonin, dan triptofan* [13]. Dari jenis-jenis tersebut, asam amino memiliki peran khusus dalam membentuk struktur protein dan berkontribusi pada fungsi biologis yang beragam dalam organisme. Oleh karena itu, sebagai *makromolekul*, protein berfungsi sebagai *katalisator*, molekul pengangkut, *reseptor* sinyal biologis, dan komponen struktural dalam sel hidup [14]. Semua organisme bergantung pada protein untuk menjalankan fungsi metabolik penting dalam kehidupan.

2.1.2 Struktur Protein

Struktur protein terbagi menjadi empat struktur utama diantaranya seperti, struktur *primer*, struktur *sekunder*, struktur *tersier*, dan struktur *kuartener*. Struktur protein dapat dilihat pada Gambar 2.2 beserta penjelasannya di bawah ini.

1. **Struktur *Primer***. Struktur ini adalah struktur yang dibentuk dari urutan asam amino dalam satu rantai protein tanpa memiliki ikatan khusus [12].
2. **Struktur *Sekunder*.** Struktur ini adalah struktur yang membentuk rantai protein dengan bentuk *heliks*, baik secara sejajar maupun berlawanan arah, dengan membentuk ikatan *hidrogen* antar asam amino. Jika ikatan terbentuk dari ujung *N-terminus* ke arah *C-terminus* akan disebut sebagai *paralel*, sedangkan sebaliknya sebagai *antiparalel* [12]. Dalam bidang informatika, identifikasi struktur *sekunder* pada protein menjadi point yang penting dalam melakukan prediksi fungsi protein [15].
3. **Struktur *Tersier***. Struktur *tersier* menggambarkan *polipeptida* yang telah melipat dengan sempurna dan padat. Beberapa *polipeptida* yang melipat tersebut terdiri dari beberapa protein *globular* yang berbeda, yang terhubung oleh residu asam amino. Stabilitas struktur *tersier* tergantung pada interaksi antara gugus R yang tidak berdekatan dalam rantai *polipeptida*. Dengan pembentukan struktur *tersier*, struktur *primer* dan *sekunder* menjadi lebih dekat satu sama lain [10].
4. **Struktur *Kuartener***. Struktur ini adalah struktur yang melibatkan dua atau lebih protein, dimana protein tersebut harus memiliki struktur *tersier* [10]. Dalam struktur *kuartener*, protein-protein tersebut bergabung untuk membentuk satu kesatuan fungsional yang lebih besar. Interaksi ini dapat melibatkan berbagai macam gaya, termasuk ikatan *hidrogen*, gaya elektrostatis, ikatan *kovalen*, dan interaksi *hidrofobik*.



Gambar 2.2 Struktur Protein [16]

Dalam *database* CAFA5, data sekuens asam amino yang ada merupakan representasi dari struktur *primer* pada protein. Struktur tersebut dikategorikan sebagai struktur *primer* karena terdiri dari urutan linear asam amino yang dihubungkan oleh ikatan *peptida*. Struktur *primer* protein tersusun dari 20 jenis asam amino yang berbeda dan membentuk suatu rantai. Ikatan *peptida* ini terbentuk selama proses sintesis protein di *ribosom*, di mana setiap asam amino ditambahkan secara berurutan sesuai dengan instruksi yang dikodekan dalam mRNA. Berbeda dengan struktur *sekunder* yang melibatkan pelipatan lokal dari rantai tersebut menjadi pola seperti *alpha* *helix* dan *beta* *sheet*, serta struktur *tersier* yang melibatkan susunan tiga dimensi penuh dari rantai *polipeptida* tunggal. Struktur *kuartener*, pada gilirannya, mencakup interaksi antara beberapa rantai *polipeptida* untuk membentuk kompleks protein fungsional.

Struktur *primer* di CAFA5 sangat penting karena menentukan bagaimana protein akan terlipat dan berinteraksi untuk membentuk struktur *sekunder*, *tersier*, dan *kuartener*. Keakuratan dalam menentukan struktur *primer* adalah dasar untuk memprediksi struktur yang lebih kompleks. Informasi ini sangat berharga dalam bioinformatika dan kimia, terutama untuk pengembangan obat dan *enzim* baru. Dengan kemajuan teknologi komputer, model prediktif yang berdasarkan struktur *primer* menjadi alat yang efektif untuk mempelajari dan memahami kompleksitas struktur protein, membawa inovasi dan kemajuan dalam pengetahuan kita tentang kehidupan dan kesehatan.

2.1.3 Fungsi Protein

Sub bab ini akan menjelaskan fungsi dari protein. Dengan mengetahui struktur asam amino yang membentuk protein, maka fungsi protein pun dapat diketahui.

2.1.3.1 Katalis Enzimatik

Hampir semua reaksi kimia dalam sistem biologis terjadi berkat *enzim*, yang kebanyakan adalah protein. Fungsi protein sebagai *katalis enzimatik* sangat berperan dalam pembuatan obat-obatan dengan metode yang efisien dan ramah lingkungan. Dalam terapi kanker, *enzim* seperti *matriks* *metalloproteinase* (MMP) digunakan untuk merancang sistem penghantaran obat nano yang tepat di jaringan tumor, memungkinkan terapi yang lebih presisi. Dengan integrasi biokatalis dalam pengembangan obat-obatan, dapat dicapai peningkatan dalam akurasi terapi, penetrasi obat yang lebih baik ke dalam jaringan tumor, dan pengurangan toksisitas lingkungan. Oleh karena itu, pemahaman dan pemanfaatan fungsi protein sebagai *katalis enzimatik* menjadi kunci dalam mengarahkan inovasi dalam pengembangan obat dan terapi yang lebih efektif dalam bidang kedokteran [17].

2.1.3.2 Transportasi dan Penyimpanan

Protein memiliki peran vital dalam transportasi, seperti *hemoglobin* membawa oksigen ke seluruh tubuh, dan penyimpanan, seperti *insulin* mengatur kadar *glukosa*. Fungsi ini penting untuk mencegah penyakit, seperti *hipoksia* tanpa transportasi oksigen atau diabetes akibat gangguan penyimpanan *glukosa*. Pemahaman mendalam terhadap peran protein dalam transportasi dan penyimpanan membuka peluang untuk terapi dan pencegahan penyakit yang lebih efektif dalam dunia kedokteran. Selain itu, fungsi protein sebagai transporter juga menjadi target obat dalam penyakit *neurologis* [18].

2.1.3.3 Penunjang Mekanis

Protein *fibrosa* seperti kolagen memberikan kekuatan pada kulit dan tulang. Kolagen membentuk jaringan ikat untuk kekencangan kulit dan membentuk matriks dengan mineral pada tulang, memberikan kekuatan struktural. Pentingnya kolagen terlihat dalam konteks pencegahan penyakit seperti *Osteogenesis Imperfecta* (OI), yang disebabkan oleh kelainan *kolagen*, menyebabkan kelemahan tulang dan risiko patah. Dalam mencegah atau mengobati penyakit terkait kekurangan *kolagen*, obat atau terapi yang mengandung protein atau merangsang produksi *kolagen* dapat efektif. Misalnya, obat yang memperbaiki struktur *kolagen* dapat meningkatkan kekuatan kulit dan tulang, mengurangi risiko penyakit. Memahami peran penting kolagen dalam menjaga integritas jaringan memungkinkan pengembangan obat dan terapi yang menargetkan *kolagen*, menjadi inovasi dalam pencegahan dan penanganan penyakit struktural. Penggunaan obat dengan kandungan protein ini berpotensi berkontribusi positif pada dunia kedokteran, mendukung kesehatan dan kualitas hidup pasien [19].

2.1.3.4 Proteksi Imun

Protein berperan sebagai antibodi, suatu bentuk pertahanan penting dalam sistem kekebalan tubuh, melindungi tubuh dari serangan *patogen* seperti virus dan bakteri. Antibodi, jenis protein khusus, bekerja dengan mengenali dan berinteraksi dengan *antigen* pada permukaan *patogen*, memicu respons kekebalan yang melibatkan penghancuran *patogen* oleh sel-sel kekebalan atau menonaktifkan kemampuan *patogen* merusak sel tubuh. Dalam dunia kedokteran, pengembangan obat dan terapi memanfaatkan antibodi sebagai strategi efektif. Vaksin merangsang produksi antibodi untuk melindungi dari infeksi, terapi antibodi *monoklonal* dapat menghancurkan sel kanker atau *patogen*, *imunomodulator* mengatur respon kekebalan, dan terapi antivirus mempercepat pemulihan dari infeksi virus. Pemanfaatan protein sebagai antibodi membuka peluang untuk pengembangan pendekatan inovatif dalam mencegah dan mengobati penyakit dengan obat yang memanfaatkan sifat protektif antibodi [20].

2.1.3.5 Membangkitkan dan Menghantarkan Impuls Saraf

Protein reseptor pada sel saraf, seperti *rodopsin* pada mata, berperan dalam merespons rangsangan, termasuk cahaya. Protein berperan kunci dalam sistem saraf dengan membantu pembangkitan dan penghantaran *impuls* saraf. Protein seperti *ion* *channel* dan *neurotransmiter* memainkan peran vital dalam proses ini. Pemahaman tentang fungsi protein dalam hal ini dapat membuka peluang pengembangan obat atau terapi yang memanfaatkan protein-protein tersebut. Contoh penggunaan obat berprotein untuk tujuan ini termasuk terapi *neurologis* yang menargetkan *neurotransmitter* untuk mengatur fungsi saraf, memberikan kontribusi pada pencegahan penyakit *neurologis* [18].

2.1.3.6 Pengaturan Pertumbuhan dan Diferensiasi

Protein seperti faktor pertumbuhan saraf mengontrol perkembangan jaringan saraf, dan banyak hormon juga merupakan protein. Dalam dunia kedokteran, peran protein, termasuk faktor pertumbuhan saraf dan *hormon* *insulin*, memiliki dampak besar dalam mengatur pertumbuhan jaringan saraf dan menjaga keseimbangan gula darah. Terapi berbasis protein, seperti penggunaan faktor pertumbuhan saraf untuk mendukung penyembuhan cedera saraf *perifer* dan terapi *insulin* untuk mengelola *diabetes* *mellitus*, membuka peluang inovatif dalam pencegahan dan penanganan penyakit. Pemahaman mendalam tentang fungsi protein menjadi landasan untuk pengembangan obat dan terapi yang dapat memberikan solusi yang lebih efektif dalam menjaga kesehatan manusia [18].

## 2.2 *Gene Ontology* (GO)

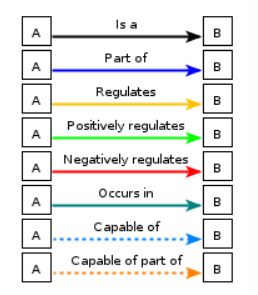
*Gene Ontology* (GO) adalah sumber utama informasi biologis yang penting karena menyediakan definisi yang spesifik mengenai fungsi protein. GO merupakan sistem klasifikasi yang membantu dalam menggambarkan dan mengelompokkan peran biologis dari gen dan protein, memberikan pemahaman yang mendalam terhadap fungsionalitas *molekuler* dalam konteks biologi [21]. *Gene* *Ontology* (GO) adalah cara untuk mengorganisir dan mengklasifikasikan informasi genetik, dan terbagi menjadi tiga jenis utama: ontologi proses biologis, ontologi fungsi molekuler, dan ontologi komponen seluler [22].

1. **Ontologi Proses Biologis (*Biological Process*)**. Proses ini biasanya disingkat dengan BP atau *Biological Process*. Ontologi gen ini merujuk pada tujuan biologis yang dicapai melalui kontribusi gen atau produk gen. Proses ini melibatkan satu atau lebih perakitan terurut dari fungsi molekuler dan sering melibatkan transformasi kimia atau fisika. Contoh tingkat tinggi dari ontologi ini termasuk 'pertumbuhan dan pemeliharaan sel' atau 'transduksi sinyal', sementara contoh tingkat rendahnya mencakup 'translasi', 'metabolisme *pirimidin'*, atau *'biosintesis* *cAMP'*.
2. **Ontologi Fungsi Molekuler (*Molecular Function*)**. Proses ini biasanya disingkat dengan MF atau *Molecular Function*. Ontologi gen ini mengacu pada aktivitas biokimia dari produk gen, termasuk ikatan spesifik dengan ligan atau struktur lainnya. Ini menjelaskan apa yang dilakukan oleh gen atau produknya tanpa menentukan di mana atau kapan peristiwa itu terjadi. Contoh tingkat tinggi meliputi *'enzim'*, 'transporter', atau *'ligand'*, sedangkan contoh tingkat rendahnya mencakup *'adenilat* *siklase'* atau *'ligand* *reseptor* *Toll'*.
3. **Ontologi Komponen Seluler (*Cellular Component*)**. Proses ini biasanya disingkat dengan CC atau *Cellular Component*. Ontologi gen ini menunjukkan lokasi di dalam sel di mana produk gen aktif. Istilah ini mencerminkan pemahaman kita tentang struktur sel *eukariotik*. Ini termasuk istilah seperti *'ribosom'* atau *'proteasom'*, yang menentukan di mana beberapa produk gen akan ditemukan, serta istilah seperti *'membran* *nukleus'* atau *'aparatus* *Golgi'*.

Pemetaan GO\_Term ke fungsi protein adalah proses menghubungkan sekuens protein dengan deskripsi fungsi spesifik menggunakan kode unik yang disebut GO\_Term. Gene Ontology (GO) menyediakan berbagai GO\_Term yang masing-masing mewakili fungsi molekuler (MF), proses biologis (BP), atau komponen seluler (CC). Misalnya, GO:0005575 adalah salah satu kode yang digunakan untuk menggambarkan "komponen seluler" secara umum dalam klasifikasi komponen seluler (CC). Melalui pemetaan ini, dapat diketahui peran atau lokasi protein dalam sel berdasarkan kecocokan sekuens protein dengan GO\_Term tertentu.

Misalkan terdapat salah satu sekuens protein yang belum diketahui fungsinya. Dengan menggunakan alat bioinformatika, dilakukan proses memetakan sekuens tersebut dan menemukan bahwa sekuens tersebut cocok dengan GO\_Term GO:0005575. Artinya, sekuens protein ini terkait dengan "komponen seluler," yang memberitahu bahwa protein ini berfungsi dalam atau berlokasi di suatu bagian dari sel. Sebagai contoh lain, jika sekuens protein dipetakan ke GO:0004674, ini berarti protein tersebut memiliki "aktivitas kinase," yaitu enzim yang menambahkan gugus fosfat ke molekul lain dalam sel. Dengan cara ini, pemetaan GO\_Term membantu dalam mengidentifikasi dan memahami fungsi spesifik dari protein berdasarkan sekuensnya, serta memberikan konteks biologis yang lebih dalam melalui anotasi ke Gene Ontology yaitu fungsi molekuler (MF), proses biologis (BP), atau komponen seluler (CC).

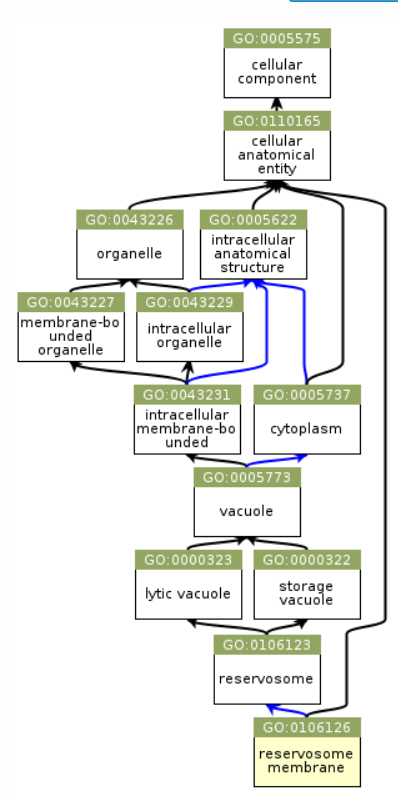
Dalam konteks menggambarkan dan mengklasifikasikan informasi genetik, *Gene Ontology* (GO) dapat direpresentasikan secara visual melalui struktur grafik. Dalam struktur ini, setiap suku GO menjadi simpul, dan hubungan antar suku diwakili oleh tepi antar simpul. GO, sebagai sistem klasifikasi hierarkis longgar, menciptakan gambaran yang dinamis tentang bagaimana fungsi-fungsi biologis saling terhubung. Setiap istilah GO diidentifikasi secara unik melalui penggunaan kode tujuh digit yang dimulai dengan "GO." Sebagai contoh, GO:0000118 merujuk pada "*histone deacetylase complex*." Pengidentifikasi ini memberikan representasi numerik eksklusif untuk setiap istilah GO, memudahkan penggunaan dan pemahaman dalam konteks analisis genetik. Dalam membuat struktur grafik tersebut, *Gene Ontology* menggunakan beberapa relasi untuk menghubungkan entitas dalam ontologi yang akan dijelaskan pada Gambar 2.3 [23].



Gambar 2.3 Relasi Struktur Gene Ontology [23]

1. *is a* (merupakan): Relasi ini menunjukkan bahwa node A adalah subtipe dari node B, membentuk struktur dasar GO.
2. *part of* (bagian dari): Relasi ini merepresentasikan hubungan bagian-keseluruhan, di mana B adalah bagian yang pasti dari A.
3. *regulates* (mengatur): Relasi ini menggambarkan kasus di mana satu proses secara langsung mempengaruhi manifestasi proses atau kualitas lainnya.
4. *positively regulates* (mengatur positif): Relasi ini menggambarkan regulasi positif, di mana satu proses meningkatkan aktivitas atau manifestasi proses atau kualitas lainnya.
5. *negatively regulates* (mengatur negatif): Relasi ini menggambarkan regulasi negatif, di mana satu proses mengurangi aktivitas atau manifestasi proses atau kualitas lainnya.
6. *occurs in* (terjadi di): Relasi ini menggambarkan tempat atau lokasi di mana suatu proses atau kualitas terjadi.
7. *capable of* (mampu): Relasi ini menunjukkan kemampuan suatu entitas atau proses dalam melakukan suatu aktivitas atau fungsi.
8. *capable of part of* (mampu bagian dari): Relasi ini menunjukkan bahwa suatu entitas atau proses mampu memiliki suatu bagian sebagai bagian dari kemampuannya.

Dengan berbagai relasi ini, *Gene Ontology* menyajikan kerangka kerja yang kompleks untuk memahami dan mengelompokkan informasi genetik dalam konteks fungsi biologis. Sebagai contoh, dapat diperhatikan pada istilah GO:0106126 yang mengacu pada fungsi protein yaitu "*reservosome membrane*." Dalam struktur grafiknya, terlihat bahwa "*reservosome membrane*" memiliki dua relasi utama, yaitu *is\_a* dengan GO:0110165 ("*cellular anatomical entity*") dan *part\_of* dengan GO:0106123 ("*reservosome*"). Struktur grafik tersebut dapat dilihat pada Gambar 2.4.



Gambar 2.4 Struktur Grafik GO:0106126 [23]

Dalam gambaran struktur grafik, "*reservosome* *membrane*" terhubung dengan "*cellular* *anatomical* *entity*" melalui relasi *is\_a*, menunjukkan bahwa *reservosome* *membrane* merupakan salah satu jenis dari *cellular* *anatomical* *entity*. Selain itu, "*reservosome* *membrane*" juga diidentifikasi sebagai bagian dari "*reservosome*," terbentuk melalui relasi *part\_of*. Untuk memberikan pemahaman tambahan, dalam representasi visual ini, kotak yang mewakili "*reservosome* *membrane*" diwarnai hijau. Warna hijau menandakan bahwa fungsi protein yang terkait diklasifikasikan ke dalam ontologi *Cellular Component* (CC), memberikan informasi tambahan terkait lokasi dan struktur di tingkat sel. Apabila kotak GO berwarna biru diklasifikasikan ke dalam ontologi *Biological Process* (BP) dan berwarna hitam diklasifikasikan ke dalam ontologi *Molecular Function* (MF)*.*

Semua ontologi ini membantu mengorganisir pengetahuan biologis, memberikan struktur yang terdefinisi dengan baik untuk mewakili informasi genetik, dan menjadi alat penting bagi peneliti untuk mengubah data menjadi pengetahuan. Sehingga dengan adanya ontologi gen ini, hasil dari prediksi fungsi protein tersebut nantinya akan digolongkan kedalam tiga jenis ontologi gen ini.

## 2.3 *Taksonomi*

*Taksonomi* merupakan cabang ilmu yang fokus pada analisis, pengenalan, penamaan, perbandingan, dan pengelompokan organisme dalam kehidupan [24]. *Taksonomi* adalah ilmu yang mempelajari cara mengelompokkan organisme berdasarkan karakteristik yang memiliki kesamaan. Dalam konteks sekuens asam amino, takson digunakan sebagai representasi umum untuk sejumlah besar organisme. Takson sendiri merujuk pada unit klasifikasi dalam sistem hierarkis, termasuk tingkatan tinggi seperti *domain, kingdom, phylum,* hingga tingkatan lebih rendah seperti *genus* dan *spesies*. Tingkatan ini mulai dari yang paling umum hingga yang paling spesifik [25].

1. ***Domain*:** Tingkatan tertinggi dalam taksonomi. Semua bentuk kehidupan dibagi menjadi tiga domain utama: *Bacteria, Archaea,* dan *Eukarya*.
2. ***Kingdom*:** Tingkatan di bawah domain, di mana organisme dikelompokkan berdasarkan karakteristik umum yang mereka miliki. Contoh *kingdom* pada domain *Eukarya* meliputi *Plantae*, *Animalia*, *Fungi*, dan lain-lain.
3. ***Phylum*:** Tingkatan setelah *kingdom*, yang lebih khusus mengelompokkan organisme berdasarkan karakteristik tertentu. Sebagai contoh, dalam *kingdom* *Animalia*, *Phylum* mencakup kelompok besar hewan yang memiliki ciri-ciri umum tertentu.
4. ***Genus* dan *Spesies*:** Tingkatan paling spesifik dalam taksonomi. *Genus* adalah kelompok yang lebih besar, sedangkan *species* adalah kelompok yang paling spesifik dan mencakup individu-individu yang sangat mirip secara genetik dan morfologis.

Dalam sekuens asam amino, pada data CAFA5, setiap protein ID dan sekuensnya mencerminkan suatu organisme tertentu. Meskipun setiap organisme diwakili secara unik, namun mereka dikelompokkan ke dalam taksonnya sendiri. Dalam data CAFA5, terdapat sekuens asam amino yang berasal dari organisme (*spesies*) yang berbeda-beda. Sekuens asam amino pada organisme yang berbeda tetapi masih dalam *genus* yang sama dapat memiliki urutan asam amino yang mirip karena mereka masih merupakan bagian dari satu keluarga. Kesamaan ini terjadi karena mereka memiliki leluhur evolusi yang sama, sehingga protein-protein mereka mungkin memiliki fungsi dan struktur yang serupa. Ini berarti bahwa meskipun ada variasi genetik antar *spesies*, urutan asam amino bisa tetap mirip dalam satu *genus* atau keluarga, mencerminkan fungsi biologis yang penting yang dipertahankan oleh seleksi alam.

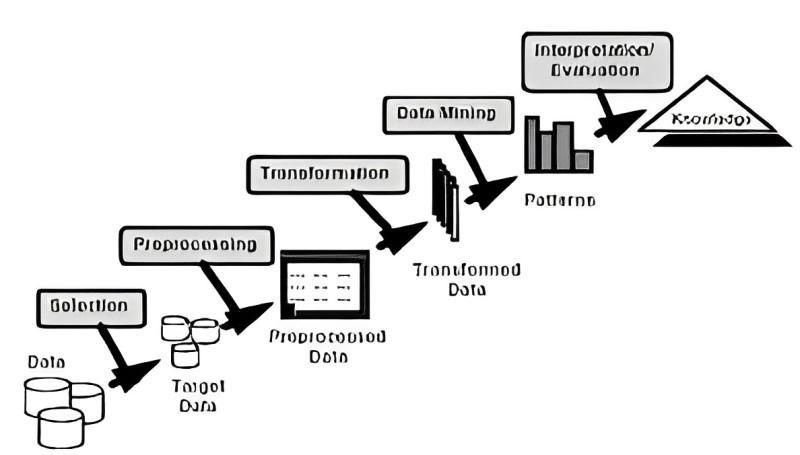
Berikut ini adalah beberapa nama organisme yang dapat menjadi contoh untuk menjelaskan terkait hubungan genus organisme dalam taksonomi.

1. ***Sus scrofa****:* Babi hutan, atau *Sus scrofa*, adalah *spesies* babi yang tersebar luas dari Eropa dan Asia, namun juga telah diperkenalkan di Amerika Utara. *Spesies* ini memiliki hubungan dengan *spesies* lain, seperti *Sus barbatus*, *Sus philippensis*, *Sus celebensis*, dan *Sus Cebifrons*. Hubungan tersebut bisa terjadi, dikarenakan *spesies-spesies* tersebut berada di satu *genus* yang sama, yaitu *Sus* (babi) dan *family* yang sama, yaitu *Suidae* (babi dan kerabatnya). Oleh karena kesamaan tersebut, *spesies* tersebut pun memiliki kemiripan dari sekuens asam amino hingga fungsi dari protein tersebut. Hal ini bisa dilihat dari protein ID **P24964** (*Sus scrofa*) memiliki kemiripan sekuens dengan protein ID **Q8M703** (*Sus barbatus*). Oleh karena sekuens asam amino yang mirip, kedua protein ID tersebut juga memiliki fungsi protein yang sama.
2. ***Escherichia marmotae***: *Escherichia marmotae*, atau *marmot's E. coli*, adalah *spesies* bakteri yang umumnya ditemukan pada marmut. *Spesies* ini memiliki hubungan dengan *spesies* lain dalam *genus* yang sama, seperti *Escherichia coli* dan *Escherichia fergusonii ATCC 35469*. Hubungan ini terjadi karena *spesies-spesies* tersebut berada dalam satu *genus* yang sama, yaitu *Escherichia*, dan keluarga yang sama, yaitu *Enterobacteriaceae*. Karena kesamaan tersebut, *spesies-spesies* ini memiliki kemiripan dari segi sekuens asam amino hingga fungsi dari protein mereka.
3. ***Xenopus laevis***: Katak cakar Afrika, atau *Xenopus laevis*, adalah *spesies* katak yang tersebar luas di Afrika sub-Sahara. *Spesies* ini memiliki hubungan dengan *spesies* lain, seperti *Xenopus tropicalis*. Hubungan tersebut terjadi karena kedua *spesies* tersebut berada dalam satu *genus* yang sama, yaitu *Xenopus* (katak cakar), dan *family* yang sama, yaitu *Pipidae* (katak tanpa lidah dan kerabatnya). Karena kesamaan tersebut, kedua *spesies* ini memiliki kemiripan dalam sekuens asam amino dan fungsi protein.

Kesamaan sekuens asam amino antar organisme dalam satu genus mencerminkan hubungan evolusi dan fungsional yang erat. Informasi taksonomi dalam CAFA5 membantu memahami bagaimana organisme yang terkait secara genetik dapat memiliki struktur dan fungsi protein yang serupa, yang penting untuk penelitian biologi, pengembangan obat, dan bioteknologi.

## 2.4 *Data Mining*

*Data mining* dalam prediksi fungsi protein merupakan suatu pendekatan yang memanfaatkan teknik-teknik analisis data untuk menggali informasi yang tersembunyi dan pola yang signifikan dari kumpulan data protein. Proses ini terintegrasi dalam kerangka *Knowledge Discovery in Database* [26], yang melibatkan langkah-langkah kritis untuk merumuskan prediksi yang akurat mengenai fungsi protein. Tahapan mengenai penggunaan *data mining* dalam konteks prediksi fungsi protein dapat dilihat pada Gambar 2.5 [10]:



Gambar 2.5 Proses Knowledge Discovery in Database (KDD)[10]

1. **Tahap *selection***, dimana bertujuan untuk memahami domain aplikasi, yaitu karakteristik dan fungsi protein yang ingin diprediksi. Pada tahap ini bertujuan untuk prediksi fungsi protein serta menentukan variabel atau fitur yang relevan untuk dianalisis.
2. **Tahap *preprocessing***, dimana bertujuan untuk menyusun data-set target dengan pemilihan data yang relevan terkait protein dan pembentukan subset data yang representatif.
3. **Tahap *transformation***, dimana bertujuan untuk melakukan pembersihan dan transformasi data, termasuk penghilangan derau, *outliers*, nilai yang hilang serta pemilihan fitur yang relevan untuk mendukung prediksi fungsi protein. Tahapan ini juga melibatkan reduksi dimensi agar model *machine learning* dapat bekerja dengan efisien.
4. **Tahapn i*nterpretation/evaluation***, dimana bertujuan untuk menginterpretasikan, mengevaluasi hasil prediksi fungsi protein. Pada tahap ini, dilakukan iterasi jika diperlukan untuk meningkatkan performa model dan memastikan keakuratan prediksi fungsi protein.

2.4.1 CRISP-DM *(Cross Industry Standard Process for Data Mining)*

Dalam era teknologi yang semakin berkembang, proses *data mining* atau penambangan data menjadi semakin penting dalam membantu penelitian untuk melakukan penggalian data dalam jumlah yang besar. Pada proses *data mining*, terdapat pendekatan atau metodologi yang telah diakui secara internasional, dimana pendekatan ini dikenal dengan CRISP-DM *(Cross Industry Standard Process for Data Mining)*. Menurut (Larose, 2006), CRISP-DM adalah suatu metodologi yang dikembangkan pada tahun 1996 dan didirikan oleh komisi Eropa [27] yang telah menyediakan cara untuk memecahkan masalah dalam *data mining*. Dengan adanya CRISP-DM, proses prediksi struktur fungsi protein menjadi lebih terstruktur dan sistematis, sehingga dapat meningkatkan akurasi dari hasil prediksi. CRISP-DM terdiri dari beberapa tahapan, yang dapat dijelaskan pada gambar Gambar 2.6 [28].



Gambar 2.6 Tahapan CRISP-DM [29]

1. **Tahapan *Business Understanding***

Pada tahap ini, tujuannya adalah untuk mengkomprehensikan tujuan dan kebutuhan dari perspektif bisnis dimana merupakan langkah awal yang penting, yang kemudian diartikan ke dalam penentuan permasalahan dalam ranah *data* *mining*. Proses berikutnya melibatkan perancangan rencana dan strategi untuk mencapai tujuan yang telah ditetapkan. Dengan melakukan tahap ini, kami dapat menjawab pertanyaan penelitian yang dibuat, sehingga dapat memahami tujuan dan kebutuhan dalam yang direncanakan sebelumnya untuk memprediksi fungsi dari protein tersebut.

1. **Tahapan *Data Understanding***

Pada tahap ini, tujuannya adalah untuk memahami data yang sudah dikumpulkan sebelumnya. Proses ini dimulai dengan mengumpulkan data, kemudian dilanjutkan dengan langkah-langkah untuk mencapai pemahaman mendalam tentang data. Hal ini mencakup identifikasi masalah kualitas data dan deteksi potensi area menarik dalam *dataset* yang dapat membentuk dasar hipotesis guna mengungkap informasi tersembunyi. Untuk memproses *dataset* tersebut dan membangunnya ke dalam model, baik itu data *training* maupun data *test*, diperlukan terlebih dahulu untuk memahami isi data, dengan tujuan untuk mendapatkan wawasan yang mendalam tentang *dataset*, dimana pada kasus ini, kami harus dapat memahami data terkait dengan *sequencing* protein untuk memprediksi fungsi dari protein tersebut.

1. **Tahapan *Data Preparation***

Pada tahap ini, tujuannya adalah untuk menyusun *dataset* akhir, yang merupakan data yang akan diproses pada tahap pemodelan. Proses ini dapat mengalami iterasi beberapa kali. Pada tahap ini, termasuk dalamnya pemilihan tabel, catatan, dan atribut-atribut data, serta melibatkan proses pembersihan dan transformasi data agar dapat digunakan sebagai *input* dalam tahap pemodelan.

1. **Tahapan *Modeling***

Pada tahap ini, tujuannya adalah untuk melakukan pemilihan dan implementasi beragam teknik pemodelan, dengan penyesuaian beberapa parameter untuk mencapai nilai optimal. Secara khusus, terdapat berbagai teknik yang dapat digunakan untuk menangani masalah *data mining* yang sama, sementara beberapa teknik pemodelan memerlukan format data khusus. Oleh karena itu, pada tahap ini, masih dimungkinkan untuk kembali ke tahap sebelumnya jika diperlukan.

1. **Tahapan *Evaluation***

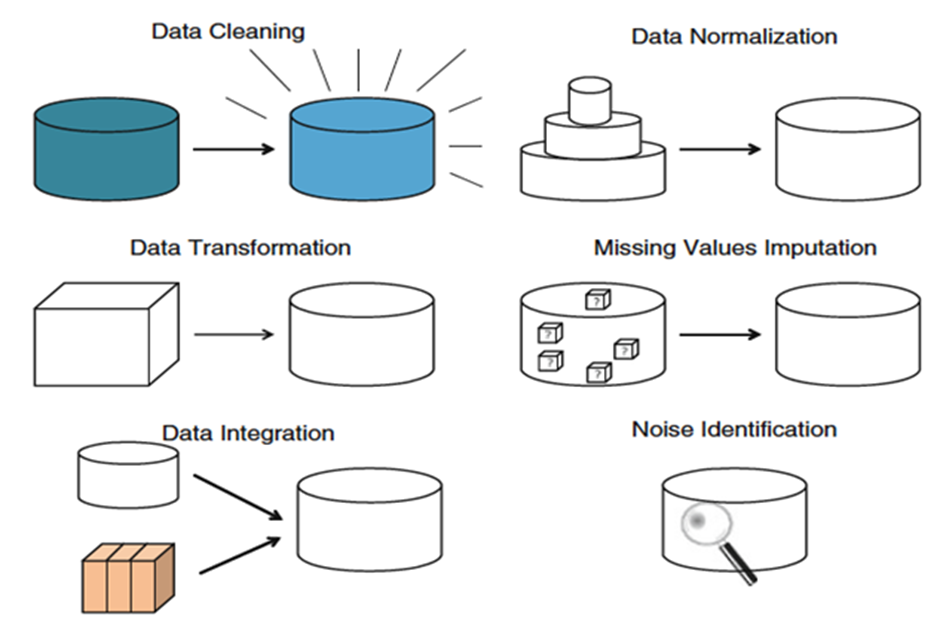
Pada tahap ini, model telah terbentuk dan diharapkan memenuhi standar kualitas yang baik berdasarkan analisis data. Tahap ini melibatkan evaluasi terhadap efektivitas dan kualitas model sebelum diterapkan, serta menentukan apakah model mampu mencapai tujuan yang telah ditetapkan pada fase awal, yaitu pemahaman bisnis.

2.4.2 *Data* *Preprocessing*

*Data* *preprocessing* merupakan proses mengubah data mentah menjadi data dengan format yang lebih mudah dan efektif untuk digunakan [30]. Memprediksi fungsi protein dengan menggunakan teknik *data preprocessing*, tahapan-tahapan seperti persiapan data, pembersihan data, integrasi data, transformasi data, dan reduksi data [31] memegang peran penting dalam memastikan kualitas data yang diperlukan untuk pengembangan model *machine learning*.

2.4.2.1 Data Preparation

*Data preparation* melibatkan kumpulan teknik untuk menginisialisasi data dengan benar sebelum diterapkan pada algoritma *Data Mining* (DM) [31]. Proses *data preprocessing* menjadi langkah yang tidak bisa diabaikan, mengubah data mentah yang mungkin tidak berguna menjadi bentuk baru yang sesuai dengan kebutuhan proses *Data Mining* (DM). Pentingnya *data preparation* dalam prediksi fungsi protein terlihat pada fakta bahwa tanpa tahapan ini, algoritma DM mungkin tidak dapat beroperasi secara optimal atau bahkan melaporkan kesalahan selama *runtime*. Oleh karena itu, teknik *data preparation* seperti pembersihan data *(data cleaning)*, transformasi data, integrasi data, normalisasi data, imputasi data yang hilang, dan identifikasi kebisingan (*noise*) menjadi esensial untuk memastikan bahwa data yang digunakan untuk prediksi fungsi protein telah diolah dengan benar, sehingga hasilnya dapat dianggap menjadi nilai akurat. Tahapan *data preparation* dapat dilihat pada Gambar 2.7.



Gambar 2.7 Data Preparation[32]

2.4.2.2 Data Cleaning

*Data cleaning* merupakan tahapan pertama dalam *data preprocessing* yang dilakukan untuk memastikan kebenaran, konsistensi, dan kegunaan data yang digunakan untuk prediksi fungsi protein. Sebagian besar *raw data* mentah seringkali memiliki kekurangan seperti inkonsistensi, redudansi, *noise* data, dan nilai yang hilang. Kekurangan ini dapat muncul karena berbagai alasan, termasuk kesalahan penulisan dan pengetikan, kerusakan data, duplikasi, dan sebagainya [33]. Penggunaan data dengan kekurangan tersebut dapat berdampak negatif pada proses penambangan data, menghasilkan keluaran yang tidak dapat diandalkan dan kualitas buruk. Dalam upaya memperbaiki data yang tidak bersih, seringkali diperlukan tindakan penghapusan baris, atau kolom dari *dataset*, atau alternatifnya, mengganti nilai yang tidak *valid* dengan nilai baru yang sesuai [33]. Proses ini dikenal sebagai *data cleaning*, juga dapat melibatkan pembagian data menjadi segmen-segmen dengan ukuran serupa, yang kemudian dihaluskan menggunakan teknik seperti *binning*.

2.4.2.3 Data Integration

*Data integration* merupakan tahapan yang dilakukan setelah data dibersihkan. Proses ini bertujuan untuk menyelesaikan konflik yang mungkin timbul dari representasi data yang berbeda dari berbagai sumber, menciptakan penyimpanan data yang konsisten, dan melakukan konversi atau konsolidasi untuk meningkatkan efisiensi proses penambangan data [33]. Dalam tahap integrasi data, beberapa masalah perlu diperhatikan, seperti integrasi skema, pencocokan objek, dan redudansi. Analisis korelasi menjadi alat yang berguna untuk mendeteksi potensi redudansi, dimana korelasi antara variabel-variabel dapat mengukur sejauh mana atribut yang seragam menjadi hal penting, sambil menghapus atribut yang tidak lagi diperlukan dari *dataset* atau sumber data yang diperlukan [34]. Selain itu, deteksi nilai data yang konflik juga menjadi fokus untuk memastikan konsistenan data yang diintegrasikan.

2.4.2.4 Data Transformation

Tahapan *data transformation* menjadi esensial untuk menghasilkan *input* yang sesuai untuk proses penambangan data. Proses ini dilakukan dengan mengubah jenis atau distribusi variabel dari data [34], menciutkan transformasi yang dapat meningkatkan efisiensi atau kemudahan pemahaman informasi. Data diubah ke dalam bentuk yang sesuai dengan kebutuhan metode penyelesaian masalah yang digunakan dalam prediksi fungsi protein. Terdapat beberapa teknik yang dapat dilakukan untuk mentransformasi data, yaitu :

1. ***Smoothing***

*Smoothing* dilakukan untuk menghilangkan noise dari data. *Smoothing* dapat dilakukan dengan menggunakan teknik seperti *clustering*, *regression* dan *binning*.

1. ***Aggregation***

*Aggregation* adalah proses penerapan metrik statistik seperti *mean*, *median* dan *variance* yang diperlukan untuk meringkas data, hasil dari agregasi ini akan digunakan pada algoritma *data mining*.

1. ***Generalization***

*Generalization* adalah mengganti data dengan *level low* menjadi tingkat yang lebih tinggi (*high*) menggunakan konsep hierarki.

1. ***Normalization***

*Normalization* adalah menyesuaikan nilai data ke dalam *range* tertentu. metode ini berguna untuk diterapkan dalam teknik penambangan seperti *classification*, *artificial neural network* (ANN) dan *clustering algorithms*.

2.4.2.5 Data Reduction

Pada tahap *data reduction* dilakukan untuk mengurangi jumlah data sehingga proses penyimpanan data dapat menjadi lebih efisien. Proses reduksi dapat melibatkan serangkaian teknik untuk memperoleh representasi data yang lebih ringkas dari data asli [32], terutama pada daya yang memiliki kapasitas berlebihan seperti data transaksi. Meskipun dianggap sebagai tahapan yang opsional, namun dalam *data mining,* dimana parameter yang digunakan untuk algoritma berbanding lurus dengan ukuran basis data masukan, reduksi data menjadi sama pentingnya dengan *data preparation* [33]. Dalam melakukan reduksi data ini akan diperlukan waktu yang cukup lama karena dibutuhkan waktu untuk mengurangi data dengan jumlah yang sangat besar.

1. ***Attribute Selection***, yaitu mengkombinasikan *tag* atau *feature* menjadi satu kesatuan untuk menyederhanakan data.
2. ***Numerosity Selection***, yaitu merepresentasikan data sebagai model atau persamaan seperti model regresi.
3. ***Dimensionality Reduction***, yaitu teknik pengurangan jumlah fitur yang berulang atau redundan pada *dataset*.

## 2.5 *Embedding* pada *Tensorflow.Keras*

*Embedding* adalah salah satu lapisan (*layer*) yang disediakan oleh *TensorFlow.Keras*. Lapisan *embedding* merupakan sebuah antarmuka tingkat tinggi yang dirancang untuk membangun dan melatih model pembelajaran mendalam (*deep learning*). Lapisan ini berfungsi untuk mengubah data *input*an berbentuk *integer* menjadi representasi vektor berdimensi tetap, yang lebih mudah diproses oleh model. Umumnya, lapisan *embedding* digunakan dalam pemrosesan bahasa alami (*Natural Language Processing)*, dimana kata-kata atau token yang awalnya direpresentasikan sebagai angka (*indeks*) akan diubah menjadi vektor yang merepresentasikan makna atau fitur dari kata tersebut dalam ruang vektor. Dalam penelitian ini,  lapisan ini juga dapat digunakan untuk sekuensprotein, dimana untuk mengubah data *input* yang sudahberbentuk *integer*, seperti *indeks* yang mewakili asam amino dalam urutan protein, menjadi representasi vektor berdimensi tetap. Proses ini memungkinkan model untuk mempelajari pola dan hubungan antara asam amino dalam sekuensprotein [35]. Dengan menggunakan lapisan *embedding*, sekuensprotein dapat diubah menjadi representasi vektor yang dapat diproses lebih lanjut oleh model *deep learning*. Representasi vektor ini mengkodekan fitur-fitur penting dari asam amino yang dapat dimanfaatkan oleh model pembelajaran mendalam untuk mengidentifikasi pola-pola kompleks dalam urutan protein. Dengan demikian, hal ini memungkinkan model untuk memprediksi fungsi protein atau melakukan tugas-tugas lainnya yang terkait dengan pemrosesan sekuensprotein.

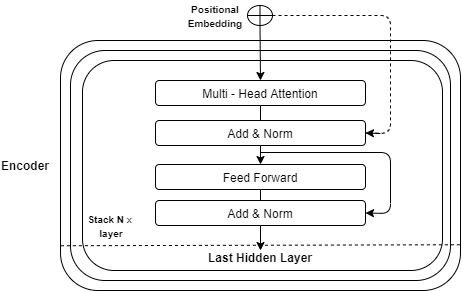
## 2.6 *Protein Language Model* (PLM)

*Protein Language Model* (PLM) adalah teknologi canggih dalam bioinformatika yang dirancang untuk mengolah dan memahami urutan protein dengan memanfaatkan konsep dari pemrosesan bahasa alami (*Natural Language Processing*) dan pembelajaran mendalam (*deep learning*). PLM tidak hanya sekadar mengenali urutan asam amino dalam protein, tetapi juga mampu memprediksi berbagai karakteristik biologis esensial, seperti struktur, fungsi, dan interaksi protein dengan zat lainnya. Dalam *Protein Language Model*, model-model seperti *ProtBERT* dan Prot-T5 telah menonjol sebagai *pionir* dalam menghasilkan representasi yang mendalam dan informatif dari urutan protein. Kemampuan PLM untuk meningkatkan akurasi dalam berbagai tugas penting seperti prediksi fungsi protein kompleks, analisis dampak mutasi, dan identifikasi motif penting dalam urutan protein, membuatnya menjadi sebuah alat yang sangat penting dalam bidang bioinformatika modern.

PLM juga menguasai kemampuan untuk menginterpretasikan konteks urutan asam amino secara mendalam, yang memungkinkannya untuk menemukan pola-pola unik yang menentukan fungsi protein secara lebih akurat. Teknologi ini memberikan kontribusi signifikan dalam penelitian biologi molekuler, mempercepat penemuan dan pengembangan obat baru, serta memfasilitasi desain protein yang lebih efektif. Secara keseluruhan, PLM bukan hanya alat penting dalam ilmu bioinformatika modern, tetapi juga menjadi pendorong utama dalam inovasi dan pemahaman yang lebih dalam terhadap biomolekul. Dengan menggabungkan kecerdasan buatan (AI) dan biologi, *Protein Language Model* membuka jalan bagi pemecahan masalah dari tantangan kompleks dalam studi biokimia dan biologi molekuler.

2.6.1 *ProtBERT*

*ProtBERT* adalah model *deep learning* yang dirancang khusus untuk memahami urutan protein dengan menggunakan arsitektur *transformer* yang terkenal dari BERT (*Bidirectional Encoder Representations from Transformers*). Pada dasarnya, *ProtBERT* merupakan hasil adaptasi dari BERT yang dilatih dengan *dataset* yang sangat besar yang terdiri dari urutan protein, dengan tujuan untuk menghasilkan representasi yang kaya dan bermakna dari sekuens asam amino [36]. ProtBERT dilatih secara pre-training pada kumpulan data urutan protein yang besar menggunakan metode masked language modeling (MLM). Pada tahap ini, beberapa asam amino dalam sekuens protein secara acak "ditutupi" dan model dilatih untuk memprediksi asam amino yang hilang berdasarkan konteksnya. Setelah pre-training, ProtBERT dapat di-fine-tune untuk berbagai tugas spesifik, seperti prediksi fungsi protein, prediksi struktur sekunder, dan lain-lain.



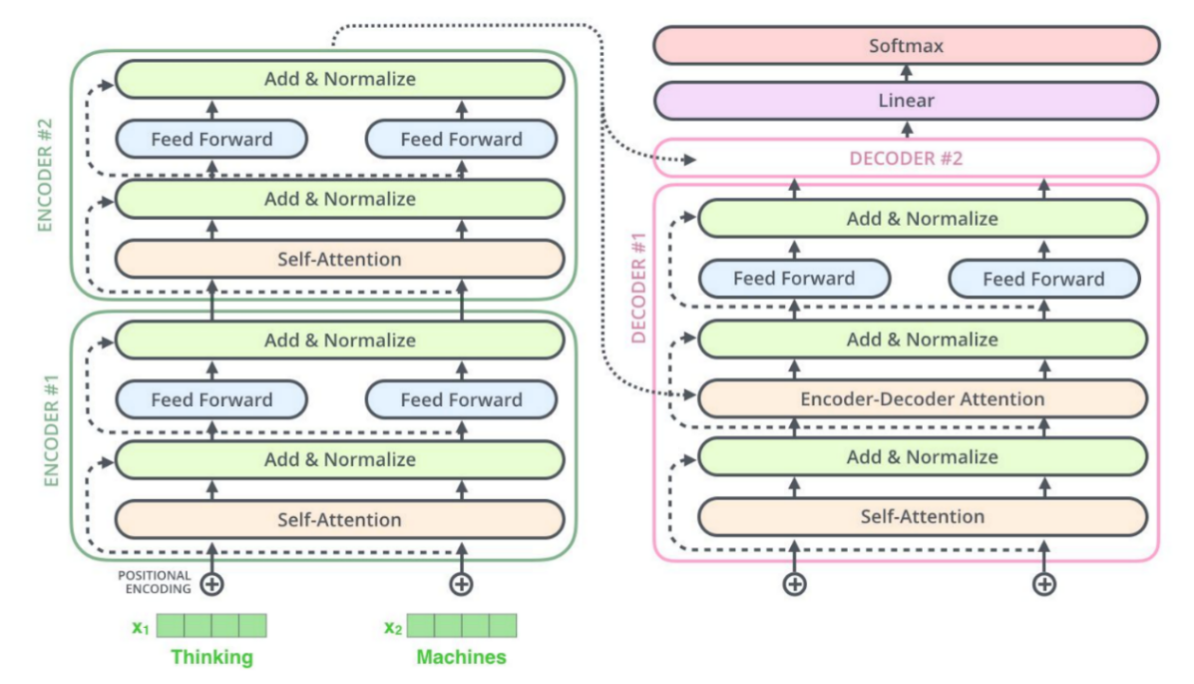
Gambar 2.8 Cara Kerja ProtBERT

Pada Gambar 2.8, menggambarkan bagaimana cara kerja dari arsitektur *embedding* ProtBERT secara keseluruhan, dimana pendekatan ini memungkinkan ProtBERT untuk menangkap berbagai fitur dan pola yang terdapat dalam data protein yang tidak bisa diidentifikasi oleh model konvensional. ProtBERT dilatih menggunakan teknik *Masked Language Modeling* (MLM), di mana arsitektur ini terdiri dari beberapa komponen utama yang bekerja bersama untuk menghasilkan representasi yang bermakna dari urutan protein. Dimulai dengan *positional embedding*, yang menambahkan informasi posisi ke setiap token asam amino dalam urutan, memungkinkan model untuk mempertahankan informasi urutan. Kemudian, *multi-head attention* digunakan untuk menangkap hubungan kontekstual antara token yang jauh terpisah dalam urutan, dengan setiap "*head*" memproses informasi dari perspektif yang berbeda dan menggabungkan hasilnya untuk membentuk representasi yang lebih kaya. Selanjutnya, *output* dari tahap ini ditambahkan kembali ke *input* asli dan dinormalisasi melalui proses *add & norm* yang membantu menstabilkan pelatihan. *Feed Forward Neural Network (FFN)* kemudian melakukan transformasi kompleks pada representasi token secara independen pada setiap posisi dalam urutan, diikuti oleh langkah *add & norm* lagi untuk menjaga stabilitas dan efisiensi pelatihan. Struktur *encoder* ini diulang beberapa kali (*Stack* N x *Layer*) untuk menangkap hubungan yang lebih dalam dan lebih kompleks dalam urutan protein. Akhirnya, *last hidden layer* menghasilkan representasi yang kaya dan bermakna dari setiap token, yang dapat digunakan untuk berbagai tugas *downstream* seperti prediksi fungsi protein dan klasifikasi struktur sekunder melalui proses *fine-tuning* [36].

Salah satu keunggulan utama dari ProtBERT adalah kemampuannya untuk menangkap representasi kontekstual yang mendalam dari urutan protein, yang memungkinkan pemahaman yang lebih baik tentang hubungan antara asam amino. Mekanisme *multi-head attention* memungkinkan model untuk memproses informasi dari berbagai perspektif secara simultan, meningkatkan akurasi dalam memahami struktur dan fungsi protein. Selain itu, penggunaan *positional embedding* memastikan bahwa urutan informasi tetap terjaga, yang sangat penting dalam analisis protein. Proses *add & norm* dan *feed forward neural network* yang diulang beberapa kali membantu model dalam menangkap kompleksitas urutan protein dengan lebih baik. Dengan struktur yang berlapis-lapis, ProtBERT mampu menghasilkan representasi yang sangat kaya, yang dapat dengan mudah diadaptasi untuk berbagai tugas bioinformatika melalui *fine-tuning*, menjadikannya alat yang sangat berguna dalam penelitian dan aplikasi bioinformatika. Dengan menggunakan ProtBERT, para peneliti dapat lebih mudah mengidentifikasi dan memprediksi fungsi protein, serta memahami efek dari mutasi yang terjadi pada urutan protein. Hal ini membuka peluang baru untuk penemuan obat, desain protein, dan penelitian terkait biologi molekuler lainnya. Sebagai contoh, dalam penemuan obat, kemampuan ProtBERT untuk menganalisis dan memahami urutan protein dapat digunakan untuk menemukan target protein baru atau mengidentifikasi potensi efek samping dari obat yang sedang dikembangkan.

2.6.2 T5 (*Text-to-text Transfer Transformer*)

T5 (*Text-To-Text Transfer Transformer*) adalah model bahasa yang dikembangkan oleh *Google Research*, yang dapat digunakan untuk berbagai tugas pemrosesan bahasa alami dengan mengonversi semua masalah menjadi tugas teks-ke-teks. Dalam konteks prediksi fungsi protein, *T5-Embedding* digunakan untuk mengubah urutan protein menjadi representasi numerik yang dapat diproses lebih lanjut oleh model [37].



Gambar 2.9 T5 (Text-to-text Transfer Transformer)[37]

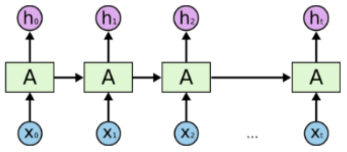
Pada Gambar 2.9, arsitektur T5 terdiri dari dua komponen utama yaitu *encoder* dan *decoder*. Pada bagian *encoder*, terdapat dua lapisan yaitu *Encoder* 1 dan *Encoder* 2, yang masing-masing terdiri dari beberapa sub-komponen: *self-attention*, *feed forward*, dan *add* *& normalize*. Setiap kata dalam urutan *input* diberi *positional encoding* untuk menyertakan informasi urutan. *Positional encoding* ini memungkinkan model untuk memahami posisi relatif dari setiap kata dalam urutan tanpa memprosesnya secara berurutan seperti pada RNN. *Self-attention* memungkinkan model untuk memperhatikan setiap kata dalam urutan *input* ketika memproses kata tertentu, sehingga model dapat memahami konteks secara lebih baik. Setelah proses *self-attention*, hasilnya diumpankan ke *feed forward layer* yang bertanggung jawab untuk menerapkan transformasi linear dan non-linear pada data. Proses *add & normalize* digunakan untuk menjaga stabilitas jaringan dengan menambahkan hasil dari setiap lapisan sebelumnya dan menerapkan normalisasi. Pada bagian *decoder*, proses yang serupa diterapkan dengan penambahan komponen *encoder-decoder attention*, yang memungkinkan *decoder* untuk memperhatikan seluruh urutan *input* dari *encoder* ketika menghasilkan keluaran. *Decoder* juga menggunakan lapisan *self-attention* dan *feed forward* yang sama seperti *encoder*. Hasil akhirnya adalah representasi yang dapat digunakan untuk berbagai tugas pemrosesan bahasa alami, termasuk prediksi fungsi protein. Dalam konteks prediksi fungsi protein, model *Protein Language Model* (PLM) menggunakan *T5-Embedding*, di mana tidak semua proses dari arsitektur T5 digunakan, namun hanya bagian *encoder*-nya saja. Pada proses ini, urutan protein atau sekuens protein diberikan sebagai *input* ke *encoder* T5. *Positional encoding* ditambahkan untuk menyertakan informasi urutan. Kemudian, melalui proses *self-attention*, model memperhatikan setiap asam amino dalam urutan protein untuk memahami konteks secara keseluruhan. Hasil dari *self-attention* diumpankan ke *feed forward layer* untuk transformasi lebih lanjut, dan *add & normalize* menjaga stabilitas dan efisiensi proses. Akhirnya, representasi numerik yang dihasilkan oleh *encoder* T5 digunakan untuk memprediksi fungsi protein dengan lebih akurat. Representasi ini kaya akan informasi kontekstual yang sangat penting untuk analisis fungsi protein secara menyeluruh [37].

## 2.7 *Deep Learning*

*Deep learning* merupakan sub-bidang dari *machine learning* yang menggunakan algoritma terinspirasi dari struktur otak manusia, yang dikenal sebagai *Artificial* *Neural Network*. *Deep learning* menjadi metode pilihan untuk banyak tugas pemodelan genom karena menghasilkan prediksi yang akurat dalam tugas-tugas tertentu salah satunya, analisis data sekuensing protein, dibandingkan dengan teknik canggih seperti metode *machine learning* dan bioinformatika [38]. Konsep *deep learning* didasarkan pada jaringan saraf tiruan *(neural network)* yang menggambarkan jaringan saraf sebagai kumpulan *node* yang terhubung. Dalam konteks prediksi fungsi protein, *deep learning* memainkan peran penting dengan menggabungkan berbagai jenis data, seperti sekuens protein, struktur 3D, ekspresi genom, dan interaksi protein-protein. Data sekuens protein memberikan informasi tentang urutan asam amino dan posisinya, sedangkan struktur 3D menunjukkan domain dan motif struktural. Data ekspresi genom dapat menangkap pola dalam ekspresi genom, dan anotasi GO yang sesuai [39]. Database interaksi protein-protein membantu dalam menemukan pola interaksi protein ke protein. Dengan mengintegrasikan data ini, *deep learning* dapat memberikan prediksi fungsi protein yang lebih akurat, membuka potensi besar untuk pemahaman yang lebih mendalam tentang hubungan kompleks antara struktur dan fungsi protein [39].

**2.7.1 *RNN (Recurrent Neural Network)***

*Recurrent Neural Network* (RNN) adalah salah satu arsitektur jaringan saraf tiruan tertua yang dikembangkan sejak tahun 1980-an. RNN dirancang untuk menangani data sekuensial dengan menyimpan informasi dari langkah-langkah sebelumnya dalam urutan, memungkinkan pemrosesan data yang memiliki keterkaitan temporal. Namun, RNN sering mengalami masalah "*vanishing gradient*". Untuk mengatasi masalah ini, beberapa unit RNN populer telah dikembangkan, seperti, *Long Short Term Memory* (LSTM), dan *Gated Recurrent* *Unit* (GRU). RNN terdiri dari satu atau lebih hidden *layer*, di mana setiap *layer* menerima nilai dari *layer* sebelumnya dan juga mengirim informasi kembali ke *layer* yang sama, menciptakan umpan balik. Koneksi umpan balik ini memungkinkan RNN untuk mempertahankan informasi dari *input* sebelumnya yang berguna untuk prediksi di langkah berikutnya. Model RNN dapat dilihat seperti pada Gambar 2.10.

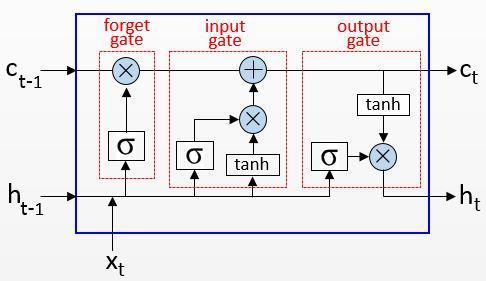


Gambar 2.10 Struktur RNN (Recurrent Neural Network)

Keunggulan utama RNN dibandingkan model *deep learning* lainnya adalah kemampuannya untuk menangkap dependensi temporal dalam data sekuensial. Namun, kelemahan RNN terletak pada masalah *"vanishing gradient"* yang menghambat kemampuannya untuk mengingat informasi jangka panjang. LSTM dan GRU, sebagai varian RNN, mengatasi masalah ini dengan mekanisme pintu (*gates*) yang mengatur aliran informasi, memungkinkan penyimpanan informasi lebih lama dan mengurangi hilangnya *gradien* selama pelatihan. Secara umum, meskipun RNN memiliki keunggulan dalam menangani data sekuensial, LSTM dan GRU lebih unggul dalam mengatasi masalah terkait dengan pelatihan jangka panjang dan efisiensi memori, menjadikannya pilihan yang lebih disukai dalam banyak aplikasi pemodelan urutan.

**2.7.2 LSTM *(Long Short Term Memory)***

LSTM *(Long Short-Term Memory)* merupakan turunan dari RNN *(Recurrent Neural Network* [40]. LSTM dan RNN dirancang khusus untuk memproses data yang berurutan. Namun, LSTM lebih dapat memproses data yang lebih kompleks dibandingkan dengan RNN. Sehingga, LSTM merupakan salah satu model yang paling efektif dalam memprediksi struktur fungsi protein. Hal ini karena LSTM memiliki kemampuan untuk mengingat dan memproses informasi jangka panjang, yang merupakan karakteristik penting dari struktur fungsi protein. LSTM terdiri dari tiga struktur, yang dapat dilihat pada Gambar 2.11[41]:



Gambar 2.11 Struktur LSTM (Long Short Term Memory) [41]

1. ***Forget gate***

Lapisan pertama dalam LSTM dikenal sebagai *forget gate*, yang memiliki fungsi sederhana yaitu menghapus informasi yang dianggap tidak relevan dan tidak lagi diperlukan oleh sistem. Dengan demikian, LSTM dapat menyajikan set lengkap informasi yang sesuai dengan kebutuhan, sambil tetap memastikan kebenarannya [42].

1. ***Input gate***

Gerbang berikutnya adalah *input gate*, yang berperan dalam menyertakan informasi yang mendukung keakuratan data. Fungsi dari *input gate* adalah menambahkan informasi yang sebelumnya telah dipilah melalui *forget gate*. RNN, yang hanya memungkinkan satu *input* data untuk satu output data, tidak memiliki fitur seperti ini. Dalam konteks *input gate*, ada yang disebut sebagai *input modulation gate*, suatu konsep yang sering diabaikan dalam beberapa tinjauan mengenai LSTM. Sesuai namanya, *input modulation gate* berfungsi untuk mengatur informasi yang ada, dengan demikian mengurangi kecepatan konvergensi dari data *zero-mean* [43].

1. ***Output gate***

Puncaknya adalah *output gate*, yang bertindak sebagai gerbang terakhir dalam menghasilkan informasi data yang lengkap dan terkini. Gerbang ini dapat menjadi langkah terakhir dalam suatu informasi atau hanya menjadi komponen tahap awal, sebelum informasi tersebut mengalami proses melalui *input gate* pada sel berikutnya [44].

Keunggulan utama LSTM dibandingkan model *deep learning* lainnya adalah kemampuannya untuk mengatasi masalah *"vanishing gradient"* yang sering terjadi pada RNN tradisional, sehingga memungkinkan penyimpanan informasi dalam jangka panjang. Dibandingkan dengan CNN, yang unggul dalam menangkap fitur spasial, LSTM lebih baik dalam menangani dependensi temporal dalam data sekuensial. Namun, kelemahan LSTM termasuk kompleksitas komputasi yang lebih tinggi dan waktu pelatihan yang lebih lama dibandingkan dengan model RNN sederhana atau CNN.

## 2.8 **Pencarian** Sekuens **Protein**

Pencarian sekuensdalam bioinformatika merujuk pada proses pencocokan atau perbandingan sekuensprotein untuk menemukan kemiripan antara sekuensyang dicari (*query*) dengan sekuens yang telah ada sebelumnya dalam basis data (*database*) [45]. Ketika terdapat sekuensprotein baru diidentifikasi, alat-alat bioinformatika dapat digunakan untuk membandingkan sekuenstersebut dengan sekuensprotein yang telah dikenal dalam basis data, sehingga proses memprediksi fungsi protein pun dapat lebih cepat diketahui. Jika terdapat kemiripan yang signifikan antara sekuensbaru dan sekuensyang sudah dikenal, informasi tentang fungsi protein yang telah diketahui dapat diterapkan pada sekuens baru tersebut. Seperti halnya, jika sekuens protein baru sangat mirip dengan sekuens protein tertentu, ada kemungkinan besar bahwa protein baru tersebut juga memiliki fungsi protein yang serupa. Berikut ini adalah *tools* bioinformatika yang dapat digunakan untuk proses pencarian sekuens.

2.8.1 *Diamond Search*

*Diamond search* atau pencarian berlian merupakan sebuah metode atau algoritma yang digunakan dalam bidang analisis bioinformatika, khususnya dalam prediksi fungsi protein menggunakan pendekatan *deep learning*. Algoritma ini termasuk program penyelarasan *throughput* tinggi yang membandingkan *file* sekuensing DNA yang dibaca dengan *file* sekuensreferensi protein, seperti *NCBI-nr19* atau KEGG, dengan tujuan mencari pola atau hubungan kompleks dalam data sekuensprotein, yang dapat digunakan untuk mengidentifikasi dan memahami fungsi biologis dari protein tersebut [46]. Pada penelitian terdahulu (DeepGoPlus), untuk menghitung skor prediksi akhir, digunakan rumus yang menggabungkan dua skor prediksi menggunakan model penjumlahan berbobot (Fishburn, 1967):

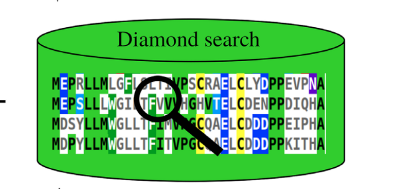
Dimana:

α = parameter bobot relatif dari kedua metode prediksi

S = skor akhir dari kombinasi prediksi

SDiamondscore = skor akhir dari hasil *similarity* menggunakan *diamond search*

SDeepGOCNN = skor yang diperoleh dari model DeepGoCNN



Gambar 2.12 Diamond Search

Pada Gambar 2.12, dapat dilihat bahwa algoritma tersebut dirancang untuk mencari pola atau hubungan kompleks dalam data sekuensprotein, yang kemudian dapat digunakan untuk mengidentifikasi dan memahami fungsi biologis dari protein tersebut. Adapun *diamond search* mengadopsi prinsip dasar dari algoritma pencarian titik puncak (*peak search*) di dalam distribusi data. Dengan menggunakan pendekatan ini, algoritma dapat secara efisien menemukan area-area signifikan dalam ruang fitur yang kompleks, memungkinkan identifikasi ciri-ciri kritis yang berkaitan dengan fungsi protein [47]. *Diamond search* adalah sebuah program yang digunakan untuk membandingkan sekuens DNA dengan sekuensprotein referensi. Program ini sangat cepat dan tetap akurat. Kecepatan dan akurasi *diamond search* dicapai dengan beberapa teknik, seperti penggunaan dua indeks, *alfabet* yang disederhanakan, dan *spaced seeds*. Pendekatan dua indeks ini membantu menghemat memori dan mempercepat akses data. Selain itu, *diamond search* menggunakan teknologi *SIMD-accelerated Smith-Waterman* untuk mempercepat proses pencarian. *Diamond search* juga mampu melakukan analisis sekuens dengan cara yang efisien dalam penggunaan memori. Teknik yang digunakan membagi ruang *seed* menjadi 1.024 bagian, sehingga hanya memproses sebagian data pada satu waktu, yang menghemat penggunaan memori.

2.8.2 *BLAST* (*Basic Local Alignment Search Tool*)

BLAST (*Basic Local Alignment Search Tool*) adalah sebuah algoritma yang digunakan dalam program bioinformatika yang sangat berguna untuk mencari kemiripan sekuens, baik DNA maupun protein. Algoritma BLAST memecah urutan menjadi kata-kata kecil dan membandingkannya dengan *database* [45]. Hal ini memfasilitasi identifikasi dan analisis kesamaan antara sekuensyang di-*input* dan yang ada dalam basis data, memungkinkan para peneliti mengidentifikasi *gen* atau protein dengan fungsi serupa. Salah satu fitur utama BLAST adalah kemampuannya untuk mempercepat pencarian sekuenspanjang, yang sangat bermanfaat dalam analisis data genomik dan proteomik yang kompleks. Program ini juga memberikan nilai "*expect*" yang memberikan informasi statistik tentang signifikansi setiap kesamaan sekuensyang ditemukan [48]. BLAST dapat memecah sekuenspanjang menjadi bagian-bagian kecil untuk meningkatkan kecepatan pencarian. Dengan mencocokkan sekuensprotein yang tidak diketahui dengan sekuensprotein dalam basis data yang sudah terverifikasi, BLAST memungkinkan para peneliti membuat prediksi tentang fungsi protein yang belum teridentifikasi. Kemiripan yang ditemukan oleh BLAST bisa disebabkan oleh hubungan evolusioner yang nyata antara dua sekuens, atau hanya kebetulan belaka.

## 2.9 Matriks Evaluasi Performansi

Dalam melatih model untuk memprediksi fungsi protein, evaluasi performansi adalah tahap krusial yang memerlukan perhatian serius. Oleh karena itu, berbagai metrik evaluasi performansi telah diimplementasikan untuk mengevaluasi kinerja model. Salah satu metrik yang umum digunakan adalah:

1. **Akurasi (*Accuracy*)**, yang memberikan gambaran umum tentang seberapa sering model memberikan prediksi yang benar.
2. **Presisi (*Precision*)** dan ***Recall*** juga sangat penting dalam konteks ini. Presisi mengukur seberapa sering model memberikan prediksi positif yang benar dari semua prediksi positif yang diberikan, sementara *recall* mengukur seberapa sering model dapat menemukan semua *instance* yang positif.
3. ***F1-score***, yang merupakan *harmonic* *mean* dari presisi dan *recall*, memberikan keseimbangan yang baik antara kedua matriks tersebut.
4. ***Specificity***, yang mengukur kemampuan model untuk mengidentifikasi semua *instance* negatif dengan benar.
5. **AUC-ROC,** juga menjadi penting untuk mengukur kemampuan model dalam membedakan antara kelas positif dan negatif. Semakin besar AUC, semakin kuat klasifikasi yang digunakan. AUC merupakan daerah berbentuk persegi yang nilainya selalu berada antara 0 dan 1. Nilai AUC sebesar 0,5 menunjukkan performa model yang acak. Jika AUC yang dihasilkan < 0.5 maka model tersebut memiliki performa yang sangat buruk sehingga tidak bagus untuk digunakan [49].
6. ***Loss function,*** digunakan untuk memberikan informasi tentang seberapa baik model menghasilkan prediksi yang benar. Dengan memperhatikan metrik-metrik evaluasi performansi ini, dapat dipastikan bahwa model yang dihasilkan memiliki kinerja yang dapat diandalkan dalam memprediksi fungsi protein.

## 2.10 Penelitian Terdahulu dalam Prediksi Fungsi Protein

Pada Tabel 2.1 disajikan beberapa penelitian yang berkaitan dengan yang telah diterapkan sebelumnya agar mendapatkan perbedaan dari penerapan metode yang telah dikembangkan.

Tabel 2.1 Penelitian Terdahulu dalam Prediksi Fungsi Protein

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Penulis (Tahun Terbit)** | **Objek / Tujuan Penelitian** | **Metode yang digunakan** | **Keterangan** |
| 1. | Fuhao Zhang, Hong Song, Min Zeng, Yaohang Li, Lukasz Kurgan, and Min Li (2019). | *DeepFunc: A Deep Learning Framework for Accurate Prediction of Protein Functions from Protein* Sekuens *and Interactions.* | *Deep Learning Function (DeepFunc)* | **Kelebihan:**  **-** Metode *DeepFunc* sangat komparatif saat dilakukan pengujian terhadap BLAST dan metode lainnya.  - Kinerja prediktif *DeepFunc* lebih baik dan konsisten dari model lainnya.  **Kekurangan:**  **-** Hanya menerapkan *input*an turunan dari urutan sekuens saja. |
| 2. | Rosalin Bonetta,Gianluca Valentino (2019) | *Machine Learning Techniques for Protein Function Prediction* | *Machine Learning* | **Kelebihan:**  **-** Metode pengoptimalan *hyperparameter* sangat membantu dalam peningkatan kinerja prediksi.  **Kekurangan:**  **-** Dalam masalah klasifikasi kelas dalam label fungsi pada Gen Ontology(GO) sangat kurang untuk bagian Proses Biologis.  - Tidak dapat menampung jumlah label output dari *input*an dalam jumlah besar.  - Memiliki ruang fitur sangat besar yang berakibat data yang available hanya sedikit yang berujung penurunan performa. |
| 3. | Naihui Zhou, Yuxiang Jiang, Timothy R Bergquist, Alexandra J Lee, dkk (2019) | *The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens.* | *CAFA 1:*  *Metode: Basic Local Alignment Search Tool (BLAST)*  *CAFA 2:*  *Metode:*  *Naive Bayes* dan *Metode: Basic Local Alignment Search Tool (BLAST).*  CAFA 3:  *Metode: Basic Local Alignment Search Tool (BLAST)* | **Kelebihan:** CAFA 1:  - Penerapan metode yang disajikan untuk menganalisis prediksi fungsi protein dapat meningkatkan akurasi.  CAFA 2:  - Penerapan metode yang disajikan dari CAFA1 mengalami peningkatan akurasi dengan adanya kemajuan metodologi yang digunakan.  CAFA 3:  - Menggunakan *Genom-wide screens* yang berbasis anotasi yang memberikan gambaran luas terkait prediksi protein.  **Kekurangan:**  CAFA 1:  - Fungsi protein sulit diprediksi karena fungsi dari protein dipelajari dari berbagai aspek dan dari berbagai tingkatan.  - Anotasi fungsional yang tersedia sangat rawan kesalahan terdapat masalah pada interpretasi pada eksperimen.  - Protein seringkali multifungsi.  CAFA 2:  - Dalam pembuatan anotasi dari fungsi makromolekul biologis sangat sulit.  - Hanya beberapa istilah onologi yang dapat diprediksi dengan tingkat akurasi yang tinggi.  CAFA 3:  - Metode yang digunakan tidak dapat digeneralisasikan dengan baik untuk beberapa istilah-istilah tertentu. |
| 4. | Divyanshu Aggarwal,  Yasha Hasija (2021) | *A Review of Deep Learning Techniques for Protein Function Prediction* | - Model DeepGO, menggunakan Metode CNN+DNN.  - Model DeepNF, menggunakan Metode *Multi-modal Deep Learning.*  - Model DeepPred, menggunakan Metode *Hierarchical DNN*.  - Model DeepGOPlus,  menggunakan Metode CNN+DNN.  -Model UDSMProt, menggunakan Metode LSTMs.  - Model SDN2GO, menggunakan Metode CNN+DNN.  - Model MultiPredGO, menggunakan Metode *Multi-modal Deep Learning*.  - Model TALE+, menggunakan Metode *Transformer*. | **Kelebihan:**  - Teknik yang dipakai pada model *Deep Learning* akan sangat baik digunakan pada data sekuens mentah.  - Akan menghasilkan tingkat akurasi yang baik dengan menggunakan 2 metode.  **Kekurangan:**  **-** Untuk hasil dari skor akurasi Fmax belum dapat mencapai hasil yang maksimal atau belum melewati 70%. |
| 5. | Kai Hakala , Suwisa Kaewphan , Jari Bjorne, Farrokh Mehryary, Hans Moen, Martti Tolvanen ,  Tapio Salakoski, and Filip Ginter (2022) | *Neural Network and Random Forest Models in Protein Function Prediction* | *Neural Network*  dan *Random Forest* | **Kelebihan:**  - Untuk Random Forest sangat baik dalam klasifikasi *multi-label* untuk ribuan label.  **Kekurangan:**  - Neural Network harus mengurangi jumlah fitur dengan cara menghapus fitur yang variasinya dibawah 0,0001 agar dapat menghasil tingkat akurasi yang akurat. |
| 6. | Maxat Kulmanov dan Robert Hoehndorf (2020) | *DeepGoPlus: improved protein function prediction from sequence* | CNN (*Convolutio-nal Neural Network)* | **Kelebihan:**  Kemampuannya dalam mendeteksi motif urutan asam amino, sehingga meningkatkan akurasi prediksi fungsi protein.  **Kekurangan:**  - Membutuhkan komputer yang kuat untuk melatih model.  - Memerlukan banyak data latihan untuk mencegah *overvitting.*  - Tidak berhasil menerapkan model RNN dan LSTM dalam prediksi fungsi protein. |
| 7. | Damlano Plovesan, Manuel Glollo, Emanuela Leonardi, Carlo Ferrari, Silvio C.E. Tosatto | *INGA: protein function prediction combining interaction networks, domain assigntments and sequence similarity* | Model INGA (model tradisional) | **Kelebihan:**  Kemampuannya menggabungkan informasi dari berbagai sumber untuk prediksi fungsi protein.  **Kekurangan:**  Ketergantungan pada data yang tersedia dan rentan terhadap kesalahan sistematis dalam basis data yang digunakan. |

Penelitian terdahulu dalam konteks CAFA1, CAFA2, dan CAFA3 menunjukkan evolusi dan perluasan pendekatan untuk prediksi fungsi protein. CAFA1 (2010-2011) mengevaluasi metode anotasi otomatis fungsi protein menggunakan evaluasi waktu tertunda (*time-delayed evaluation*) dengan Naive Bayes dan BLAST. Dalam CAFA1, banyak protein yang tidak memiliki anotasi atau informasi eksperimental tentang fungsinya. Peneliti harus membuat prediksi sebelum memperoleh informasi eksperimental terbaru, dan evaluasi dilakukan untuk menilai sejauh mana prediksi tersebut akurat berdasarkan data eksperimental yang muncul kemudian. CAFA2 (2013-2014) memperluas tantangan dalam memprediksi fungsi protein dengan mengembangkan metode prediksi, memperluas ke ontologi baru, meningkatkan jumlah protein target, dan memperbaiki matriks penilaian. CAFA3 (2017-2019) menekankan penyaringan eksperimental dan pembuatan *benchmark*, serta menerapkan pendekatan evaluasi waktu tertunda untuk pertumbuhan anotasi.

Penelitian tentang prediksi fungsi protein menggunakan *machine learning* menunjukkan evolusi teknik dalam melatih model berbasis data. Meskipun penggunaan teknik *machine learning* masih lebih rendah dibandingkan *deep learning* dalam hal mengekstraksi fitur dan pengembangan prediktor berperforma tinggi, *machine learning* tetap mampu mengungguli beberapa pendekatan *deep learning*. Teknik *machine learning* sering menggunakan fitur numerik dan NLP (*Natural Language Processing*) untuk merepresentasikan rangkaian asam amino sebagai teks.

Pada penelitian yang menggunakan *framework deep learning*, seperti dalam pengembangan model DeepFunc, meskipun hasil akurasinya terbukti meningkatkan kinerja prediktif, pendekatan ini memerlukan perancangan model yang kompleks. DeepFunc menggunakan pendekatan berbasis urutan dengan pembuatan vektor informasi berdimensi tinggi (sekitar 35.000 dimensi) untuk mendeskripsikan domain, kelompok, dan motif yang diekstraksi oleh InterPro. Data ini harus direduksi sebelum digabungkan dengan data berdimensi rendah yang diekstraksi dari jaringan PPI dengan menggabungkan hubungan fungsional dari EggNOG dan interaksi dari STRING. Penelitian ini juga menggunakan algoritma DeepWalk untuk mengekstrak fitur topologi komprehensif dari jaringan PPI.

DeepFunc menggunakan jaringan dalam dua tujuan berbeda, yaitu mengubah pendekatan berbasis urutan berdimensi tinggi menjadi format berdimensi rendah yang kaya informasi dan menggabungkan data tersebut dengan informasi topologi dari jaringan PPI. Hasil penelitian menunjukkan bahwa DeepFunc dapat mengungguli prediktor lain seperti DeepGO, FFPred3, dan GOPDR, dengan nilai Fmax = 0,54 dan AUC = 0,94, dibandingkan dengan FFPred3 (Fmax = 0,38 dan AUC = 0,86), GoFDR (Fmax = 0,52 dan AUC = 0,84), dan DeepGO (Fmax = 0,47 dan AUC = 0,90).

Penelitian tentang DeepGOPlus menunjukkan bahwa model CNN (*Convolutional Neural Network*) memiliki beberapa keunggulan dalam memprediksi fungsi protein. CNN dapat mendeteksi pola dalam urutan asam amino dengan sangat baik, sehingga meningkatkan akurasi prediksi. Namun, CNN membutuhkan banyak data untuk dilatih agar tidak *overfitting* dan memerlukan sumber daya komputasi yang besar. Selain itu, terdapat satu model tradisional yaitu INGA, yang menunjukkan bahwa metode ini unggul dalam menggabungkan berbagai jenis informasi, seperti interaksi antara protein, kesamaan urutan, dan penugasan domain, untuk menghasilkan prediksi yang lebih akurat. Namun, INGA sangat bergantung pada ketersediaan data jaringan interaksi protein. Jika data ini tidak lengkap atau salah, prediksinya bisa kurang akurat.

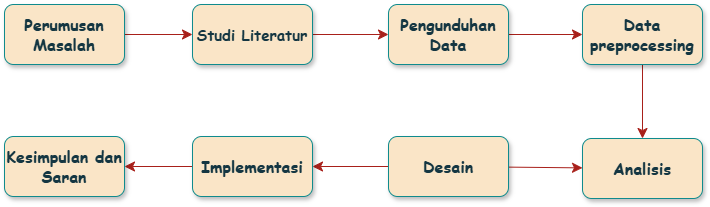
Penelitian terdahulu menunjukkan variasi dalam metode dan pendekatan yang digunakan untuk prediksi fungsi protein. Meskipun beberapa metode menunjukkan kelebihan, seperti tingkat akurasi yang baik atau peningkatan kinerja dengan pengoptimalan *hyperparameter*, setiap metode juga memiliki keterbatasan yang perlu diperhatikan, seperti kesulitan dalam memprediksi fungsi protein multifungsi atau kendala dalam penggunaan metode tertentu untuk beberapa istilah tertentu. Selain itu, masih ada beberapa hal yang perlu dilakukan dalam mencapai tingkat akurasi maksimal dalam prediksi fungsi protein menggunakan pendekatan *deep learning*.

# BAB III ANALISIS

Bab ini menjelaskan analisis terhadap tahapan yang dilakukan pada penelitian. Analisis yang dilakukan akan menjadi acuan pada tahap implementasi. Adapun analisis yang dilakukan yaitu analisis metodologi penelitian.

## 3.1 **Tahapan Penelitian**

Tahapan penelitian yang dilakukan pada Tugas Akhir ini dapat dilihat pada Gambar 3.1.



Gambar 3.1 Tahapan Penelitian

Berikut adalah penjelasan dari setiap metode penelitian.

1. **Perumusan Masalah Penelitian**

Pada tahap ini, dilakukan pengenalan terhadap latar belakang dari penelitian yang sedang dikembangkan yang terkait dengan permasalahan yang akan diatasi. Proses ini bertujuan untuk mengidentifikasi masalah yang relevan dalam prediksi fungsi protein. Rumusan masalah dalam penelitian ini didasarkan pada studi literatur yang mencakup berbagai sumber seperti, *e-book* dan jurnal penelitian terkait.

1. **Studi Literatur**

Pada tahap ini, dilakukan pencarian landasan teori dari berbagai sumber penelitian, seperti buku dan juga jurnal yang merupakan sumber-sumber informasi mengenai keakuratan jawaban yang diberikan oleh metode *Deep Learning* dengan model LSTM. Tujuan dari tahapan ini adalah untuk mengeksplorasi teori dan metodologi yang akan digunakan untuk menjawab permasalahan penelitian dan untuk membentuk dasar teoritis.

1. **Pengumpulan Data**

Pada tahap ini, peneliti mengumpulkan data yang akan digunakan dalam penelitian sebagai materi dasar dalam melakukan identifikasi terhadap permasalahan yang terkait dalam melakukan prediksi fungsi protein berdasarkan *dataset*.

1. ***Data Preprocessing***

Pada tahap ini, yaitu melakukan *data cleaning*. Pembersihan data dilakukan untuk mengantisipasi atau mengatasi adanya *missing value* atau *noise* pada *dataset* yang digunakan.

1. **Analisis**

Pada tahap ini, dilakukan analisis terhadap data untuk mengetahui karakteristik data yang dimiliki dan analisis dari model yang digunakan yaitu LSTM.

1. **Desain**

Pada tahap ini, peneliti merancang tahapan yang menggambarkan bagaimana proses-proses pelaksanaan dari hasil analisis yang telah diperoleh dari bab sebelumnya. Desain yang telah dibuat akan menjadi pdoman utama dalam proses implementasi yang akan dilakukan.

1. **Implementasi**

Pada tahap ini, dilakukan implementasi dengan menerapkan desain yang telah dirancang. Implementasi ini mencakup pembentukan *dataset* dan metode yang dipilih berdasarkan hasil rancangan yang telah dibuat sebelumnya. Dari hasil implementasi tersebut, didapatkan metrik evaluasi dari model yang digunakan pada penelitian ini.

1. **Kesimpulan dan Saran**

Pada tahap ini, akan dihasilkan kesimpulan dari hasil pembahasan yang sudah dilakukan sebelumnya dan juga memberikan saran yang bertujuan untuk membantu peneliti pada penelitian selanjutnya.

## 3.2 Analisis Data

Pada analis data memiliki tujuan untuk menganalisis dan menginterpretasi data yang telah dikumpulkan selama penelitian. Proses analisis data ini melibatkan beberapa langkah, termasuk deskripsi *dataset*, identifikasi atribut yang digunakan, persiapan data, *pre-processing* data, analisis statistik, dan analisis model.

3.2.1 Deskripsi *Dataset*

Pada sub bab ini, akan menjelaskan tentang sekuens asam amino dalam *file* .fasta, fungsi protein dalam *file* .tsv, dan taksonomi protein dalam *file* .tsv.

3.2.1.1 Sekuens Asam Amino

*File* dengan format .fasta adalah format berbasis teks untuk mewakili urutan *nukleotida* atau sekuens asam amino (protein), dimana *nukleotida* atau asam amino diwakili menggunakan kode huruf tunggal. Dalam format .fasta, terdapat dua bagian utama yaitu *header* dan sekuens (baik DNA, RNA, maupun protein). *Header* dimulai dengan simbol **“>”** yang diikuti oleh deskripsi atau informasi tambahan terkait sekuens. Sekuens kemudian mengikuti bagian *header* yang terdiri dari baris-baris huruf yang mewakili *nukleotida*. Dalam sekuens protein dalam format .fasta, setiap huruf atau simbol mewakili satu asam amino dalam rantai *polipeptida*. Simbol ataupun huruf dari sekuens protein tersebut dapat dilihat pada Tabel 3.1.

Tabel 3.1 Simbol Asam Amino

|  |  |  |  |
| --- | --- | --- | --- |
| **Huruf Asam Amino** | **Arti** | **Huruf Asam Amino** | **Arti** |
| A | *Alanine* | M | *Methionine* |
| C | *Cysteine* | N | *Asparagine* |
| D | *Aspartic Acid* | P | *Proline* |
| E | *Glutamic Acid* | Q | *Glutamine* |
| F | *Phenylalanine* | R | *Arginine* |
| G | *Glycine* | S | *Serine* |
| H | *Histidine* | T | *Threonine* |
| I | *Isoleucine* | V | *Valine* |
| K | *Lysine* | W | *Tryptophan* |
| L | *Leucine* | Y | *Tyrosine* |

Dalam konteks ini, *file* .fasta digunakan untuk menyimpan sekuens asam amino dari protein dalam *dataset*, dimana pada data CAFA5, di *train\_*sekuens*.fasta* terdapat 142.246 sekuens asam amino dan pada *test\_superset.fasta* terdapat 141.865 sekuens asam amino yang belum dianotasi fungsi proteinnya. Berikut ini adalah deskripsi dari isi format file train\_sekuens.fasta yang dapat dilihat pada Tabel 3.2.

Tabel 3.2 *File train\_sekuens.fasta*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Nama Atribut** | **Tipe Atribut** | **Keterangan** | **Contoh** |
| 1. | UniProt | *object* | Id protein yang mewakili sekuensatau urutan dari asam amino *(amino acids)* | >Q9CQV8 10090 |
| 2. | SekuensProtein | *object* | Urutan asam amino *(amino acids)* | MTMDKSELVQKAKLAEQAERYDDMAAAMKAVTEQGHELSNEERNLLSVAYKNVVGARRSS  ….  AGEGEN |

Berikut ini adalah deskripsi dari isi format file test\_superset.fasta yang dapat dilihat pada Tabel 3.3.

Tabel 3.3 *File test\_superset.fasta*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Nama Atribut** | **Tipe Atribut** | **Keterangan** | **Contoh** |
| 1. | UniProt | *object* | Id protein yang mewakili sekuensatau urutan dari asam amino *(amino acids)* | >P20536 sp|P20536|UNG\_VACCC Uracil-DNA glycosylase OS=Vaccinia virus (strain Copenhagen) OX=10249 GN=UNG PE=1 SV=1 |
| 2. | SekuensProtein | *object* | Urutan asam amino *(amino acids)* | MNSVTVSHAPYTITYHDDWEPVMSQLVEFYNEVASWLLRDETSPIPDKFFIQLKQPLRNK  ….  NWAQGFIY |

3.2.1.2 Fungsi Protein

*File* dengan format .tsv atau nilai yang dipisahkan oleh tab adalah format berkas sederhana berbasis teks untuk menyimpan data tabular, dimana data tabular adalah tata letak data yang diatur dalam bentuk tabel dengan informasi yang disusun dalam baris dan kolom. Rekaman dipisahkan oleh baris baru, dan nilai dalam satu rekaman dipisahkan oleh karakter tab. Format .tsv ini merupakan format nilai yang dipisahkan oleh delimiter, mirip dengan nilai yang dipisahkan oleh koma. Dalam *file* ini berisi daftar istilah beranotasi yang terdiri dari tiga kolom, yaitu pada kolom pertama terdapat UniProt, kolom kedua adalah ID istilah GO, dan kolom ketiga adalah kolom yang menunjukkan aspek GO (di ontologi yang mana istilah GO tersebut muncul). *File* .tsv menyajikan fungsi dari protein. Dalam data CAFA5 terdapat *train\_terms.tsv* yang berisikan 31.466 istilah GO yang menganotasikan fungsi protein dari 142.246 sekuens asam amino yang berada pada *file* *train\_*sekuens*.fasta*. Berikut ini adalah deskripsi dari isi format *file* .tsv yang dapat dilihat pada Tabel 3.4.

Tabel 3.4 *File train\_terms.tsv*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Nama Atribut** | **Tipe Atribut** | **Keterangan** | **Contoh** |
| 1. | UniProt | *object* | Id protein yang mewakili sekuensatau urutan dari asam amino *(amino acids)* | >P20536 |
| 2. | ID Istilah GO | *object* | Istilah *Gen Ontology* (GO) yang menggambarkan fungsi protein | GO:0008152 |
| 3. | *Ontology* | *object* | Aspek *Gen Ontology* (GO) yang menunjukkan di aspek mana istilah GO tersebut muncul (cth: BPO, CCO, MFO) | BPO |

Atribut-atribut diatas adalah atribut yang digunakan dalam menunjukkan fungsi dari protein. Untuk dapat lebih memahami atribut diatas, berikut ini akan ditampilkan beberapa contoh yang akan menampilkan fungsi dari protein pada Tabel 3.5.

Tabel 3.5 Contoh Isi *File train\_terms.tsv*

|  |  |  |
| --- | --- | --- |
| **UniProt** | **ID Istilah GO** | **Ontology** |
| P20536 | GO:0044249 | BPO |
| P20491 | GO:1903305 | BPO |
| A0A009IHW8 | GO:0003674 | MFO |
| X6RM59 | GO:0005634 | CCO |
| V9HWA2 | GO:0005737 | CCO |

Dari ***Tabel 3.5***, terdapat hubungan antara *file* .tsv dengan .fasta, dimana atribut UniProt yang terdapat pada *file* .tsv akan terdapat juga pada *file* .fasta. Sebagai contoh, dari tabel diatas, di *file* .txt terdapat **“UniProt:P20536”** yang diikuti dengan fungsi **“GO:0044249:cellular biosynthetic process”** daripada UniProt tersebut, sedangkan pada *file* .fasta terdapat juga **“UniProt:P20536”** tetapi menampilkan sekuens protein dari UniProt tersebut.

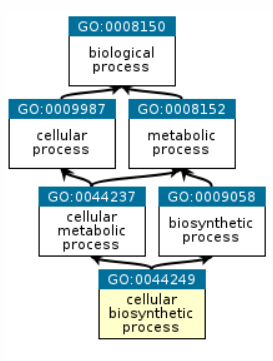
Selain *file* diatas, terdapat juga *file* *go.obo* yang menampilkan *graph* dari fungsi protein (ID istilah GO) tersebut. Format *file* OBO merupakan bahasa yang difokuskan pada bidang biologi untuk membangun ontologi. Format ini didasarkan pada prinsip-prinsip *Web Ontology Language* (OWL), sebuah keluarga bahasa representasi pengetahuan yang digunakan untuk pembuatan ontologi. Dalam konteks pemahaman aspek fungsi protein, ontologi bertindak sebagai representasi formal yang menggambarkan taksonomi dan jaringan klasifikasi untuk memahami fungsi-fungsi protein. Ontologi menggunakan kata benda sebagai representasi kelas objek dan kata kerja sebagai representasi hubungan antar objek, memberikan struktur pengetahuan yang mendalam dalam memahami kompleksitas fungsi protein dalam berbagai domain biologis. Sebagai contoh dari istilah GO di atas yaitu **“GO:0044249:*cellular biosynthetic process*”**, dari *file* .obo terdapat deskripsi dari istilah GO tersebut yang menggambarkan *graph* dari istilah tersebut. Deskripsi dari istilah GO tersebut dapat dilihat pada Tabel 3.6.

Tabel 3.6 Contoh Isi *File go.obo*

|  |
| --- |
| **[*Term*]** |
| id: GO:0044249  *name: cellular biosynthetic process*  *namespace: biological\_process*  *def: "The chemical reactions and pathways resulting in the formation of substances, carried out by individual cells."* [GOC:jl]  *synonym: "cellular anabolism”* EXACT []  *synonym: "cellular biosynthesis"* EXACT []  *synonym: "cellular formation"* EXACT []  *synonym: "cellular synthesis"* EXACT []  is\_a: GO:0009058 ! *biosynthetic process*  is\_a: GO:0044237 ! *cellular metabolic process* |

Dari deskripsi tersebut, didapat struktur grafik yang dapat digambarkan dengan tujuan untuk memudahkan penggunaan dan pemahaman dalam konteks analisis genetik. Dalam struktur grafik tersebut, *Gene Ontology* menggunakan beberapa relasi untuk menghubungkan entitas dalam ontologi, dimana dalam istilah GO tersebut terdapat relasi *is\_a* dengan istilah GO **“GO:0009058: *biosynthetic process*”** dan *is\_a* dengan istilah GO **“GO:0044237: *cellular metabolic process*”**. Kotak istilah GO juga memiliki warna biru dikarenakan aspek dari istilah GO ini adalah *biological process*.

Sebagai tambahan dalam Tugas Akhir ini nantinya, kami melakukan analisis terhadap data protein yang terdapat dalam *file* .fasta dan fungsi-fungsinya yang dianotasi dalam *file* .tsv. Kami menggunakan ontologi *Gene Ontology* yang direpresentasikan dalam *file* .obo untuk memahami hubungan antara berbagai fungsi protein. Hasil dari Tugas Akhir kami adalah penelusuran dan analisis mendalam terhadap istilah GO tertentu, seperti GO:XXXXXX (contoh: GO:0044249: *cellular biosynthetic process*), yang memberikan informasi tambahan dalam pemahaman tentang proses biologis tertentu. Struktur grafik dari istilah GO tersebut dapat dilihat pada Gambar 3.2.



Gambar 3.2 Struktur Grafik Istilah GO:0044249

3.2.1.3 Taksonomi Protein

Dalam CAFA5, *file* untuk menampilkan taksonomi juga disimpan dalam format TSV *(Tab-Separated Values)*, yang merupakan format *file* sederhana berbasis teks untuk menyimpan data tabular. Setiap catatan *(record)* dalam *file* dipisahkan oleh baris baru, dan nilai-nilai di dalam catatan dipisahkan oleh karakter tab. Format TSV ini membuat penyimpanan dan pertukaran data taksonomi menjadi lebih mudah, serupa dengan format CSV *(Comma-Separated Values)*, dimana struktur data tabular dapat diorganisir dengan jelas dan dapat diakses dengan mudah. Dalam konteks ini, *file* .tsv digunakan untuk menyimpan taksonomi dari tiap sekuens asam amino, dimana pada data CAFA5, di *testsuperset-taxon-list.tsv* terdapat 90 taksonomi dan pada *train\_taxonomy.tsv* terdapat 3156 taksonomi dari sekuens asam amino tersebut. Berikut adalah deskripsi dari isi format *file* *testsuperset-taxon-list.tsv* yang dapat dilihat pada Tabel 3.7.

Tabel 3.7 *File testsuperset-taxon-list.tsv*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Nama Atribut** | **Tipe Atribut** | **Keterangan** | **Contoh** |
| 1. | ID | *object* | Id *taxonomy* yang mewakili takson dari sekuens asam amino tersebut | 9606 |
| 2. | *Spesies* | *object* | Nama spesies dari takson sekuens asam amino tersebut | *homo sapiens*[All Names] |

Berikut adalah deskripsi dari isi format *file* *train\_taxonomy.tsv* yang dapat dilihat pada Tabel 3.8.

Tabel 3.8 *File train\_taxonomy.tsv*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Nama Atribut** | **Tipe Atribut** | **Keterangan** | **Contoh** |
| 1. | *EntryID* | *object* | Id protein yang mewakili sekuensatau urutan dari asam amino *(amino acids)* | Q8IXT2 |
| 2. | *TaxonomyID* | *object* | Id *taxonomy* yang mewakili takson dari sekuens asam amino tersebut | 9606 |

3.2.2 Persiapan Data

Pada sub bab ini, akan dijelaskan data yang digunakan dalam penelitian ini, yang mencakup tiga jenis *file* utama, yaitu .fasta, .tsv, dan .obo, untuk melakukan analisis dan prediksi pada urutan asam amino protein. Dalam *file* .fasta, terdapat dua bagian penting, yaitu:

1. ***File* .fasta**: *File* ini berisi urutan asam amino untuk protein. Ada dua *file* utama dalam format .fasta yang digunakan:

* ***train\_*sekuens*s.fasta***: *File* ini berisi urutan asam amino untuk protein dalam set pelatihan. Formatnya mencakup informasi seperti ID protein, *spesies*, dan urutan asam amino. Contoh isi *file* ini adalah:

>P20536 sp|P20536|UNG\_VACCC Uracil-DNA glycosylase OS=Vaccinia *virus* (*strain Copenhagen*) OX=10249 GN=UNG PE=1 SV=1 MNSVTVSHAPYTITYHDDWEPVMSQLVEFYNEVASWLLRDETSPIPDKFFIQLKQPLRNK…

* ***testsuperset.fasta***: *File* ini berisi urutan asam amino untuk protein di mana prediksi harus dilakukan.

1. ***File* .tsv**: *File* ini memiliki format *tab-separated values* dan menyajikan informasi tambahan tentang protein.

* ***train\_terms.tsv***: *File* ini menyajikan hubungan antara protein dalam set pelatihan dan istilah *Gene Ontology* (GO) yang beranotasi. Contohnya, protein dengan EntryID “A0A009IHW8” memiliki term “GO:000815” dengan aspek “BPO”.
* ***train\_taxonomy.tsv***: *File* ini menyajikan ID taksonomi untuk protein dalam set pelatihan, sehingga setiap protein dapat diidentifikasi pada tingkat taksonomi tertentu.
* ***testsuperset-taxon-list.tsv***: *File* ini berisi ID taksonomi untuk protein dalam superset uji, yang memungkinkan identifikasi dan analisis pada tingkat taksonomi.

1. ***File* .obo**: *File* ini berisi struktur grafik ontologi untuk istilah GO. Sebagai contoh, *term* GO:0000001 memiliki nama "*mitochondrion inheritance*" dan termasuk dalam *namespace biological\_process*. Definisinya adalah "Distribusi *mitokondria*, termasuk *genom* *mitokondria*, ke dalam sel anak setelah *mitosis* atau *meiosis*, yang dimediasi oleh interaksi antara *mitokondria* dan *sitoskeleton*."

*File* tersebut saling terkait dan membentuk kerangka kerja yang holistik untuk mendukung penelitian dan prediksi dalam domain biologi molekuler. Data CAFA5 ini, yang merupakan tantangan untuk memprediksi fungsi protein berdasarkan urutan asam amino dan informasi terkait lainnya.

3.2.3 Analisis Statistik

Pada sub bab ini, akan dilakukan analisis statistik terhadap dua aspek penting dari data yang digunakan dalam penelitian, yaitu panjang sekuens asam amino dari protein dan jumlah frekuensi dari kelas fungsi protein. Analisis ini bertujuan untuk memberikan pemahaman yang lebih mendalam tentang kompleksitas struktur data dan fungsi biologis protein tersebut. Analisis statistik pada panjang sekuens asam amino dapat mencakup beberapa parameter statistik deskriptif, seperti rata-rata, median, kuartil, dan rentang.

3.2.3.1 Statistik Sekuens Protein

Dalam analisis statistik panjang sekuens, langkah ini melibatkan pemahaman tentang distribusi panjang sekuens protein untuk memahami variasi yang ada. Perhitungan rata-rata dan median panjang sekuens dilakukan untuk mendapatkan gambaran pusat distribusi. Visualisasi distribusi menggunakan histogram membantu mengidentifikasi pola dan kecenderungan dalam sekuens, memberikan wawasan yang penting dalam pemahaman struktur dan fungsi protein. Hasil dari nilai rata - rata dan median yang didapat dari hasil kalkulasi dapat dilihat pada Tabel 3.9.

Tabel 3.9 Statistik Sekuens Protein

|  |  |
| --- | --- |
| **Jenis Statistik** | ***Sequence\_len*** |
| *count* | 142246.000000 |
| *mean* | 553.636679 |
| std | 641.728770 |
| min | 3.000000 |
| 25% | 248.000000 |
| 50% | 411.000000 |
| 75% | 654.000000 |
| max | 35375.000000 |

Dari Tabel 3.9, dapat dijelaskan sebagai berikut.

1. ***Count* (Jumlah)**

*Count* dapat diindikasikan sebagai jumlah total observasi atau entitas dalam *dataset*. Dalam *dataset* *training* CAFA5, terdapat 142.246 data sekuens protein.

1. ***Mean* (Rata-Rata)**

Rata-rata panjang sekuens dihitung dengan menjumlahkan panjang semua sekuens lalu membaginya dengan jumlah sekuens protein yang ada pada *dataset*. Jumlah panjang keseluruhan sekuens protein adalah 78.752.603, sedangkan jumlah sekuens protein yang ada pada *dataset* CAFA5 adalah 142.246. Berikut rumus yang digunakan untuk mencari rata-rata dari panjang sekuens protein:

Dengan demikian, rata-rata panjang sekuens yang didapat adalah sekitar 553,636 asam amino. Perhitungan rata-rata tersebut dapat memberikan gambaran pusat dari distribusi panjang sekuens, menunjukkan nilai tengah yang dapat mewakili panjang sekuens secara keseluruhan.

1. **Std (Deviasi Standar)**

*Standard deviation* atau deviasi standar mengukur seberapa jauh data tersebar dari rata-rata. Dalam kasus panjang sekuens protein, nilai deviasi standar sekitar 641,728 menunjukkan bahwa panjang sekuens cenderung bervariasi dalam rentang yang cukup besar dari rata-rata.

1. **Min (Minimum)**

Nilai minimum adalah nilai terkecil dalam *dataset*. Dalam hal panjang sekuens protein, nilai minimum adalah 3 yang menunjukkan panjang sekuens terpendek yang ada dalam *dataset*.

1. **25% (Kuartil Pertama - Q1)**

Kuartil pertama (Q1) adalah nilai yang membagi 25% data terkecil dalam *dataset*. Dalam konteks panjang sekuens protein, nilai kuartil pertama sebesar 248 menunjukkan bahwa 25% sekuens memiliki panjang kurang dari 248 asam amino.

1. **50% (Median)**

Median adalah nilai tengah dalam *dataset* yang membagi data menjadi dua bagian setara. Dalam kasus ini, median panjang sekuens protein sebesar 411, yang berarti setengah dari total sekuens memiliki panjang kurang dari 411 asam amino.

1. **75% (Kuartil Ketiga - Q3)**

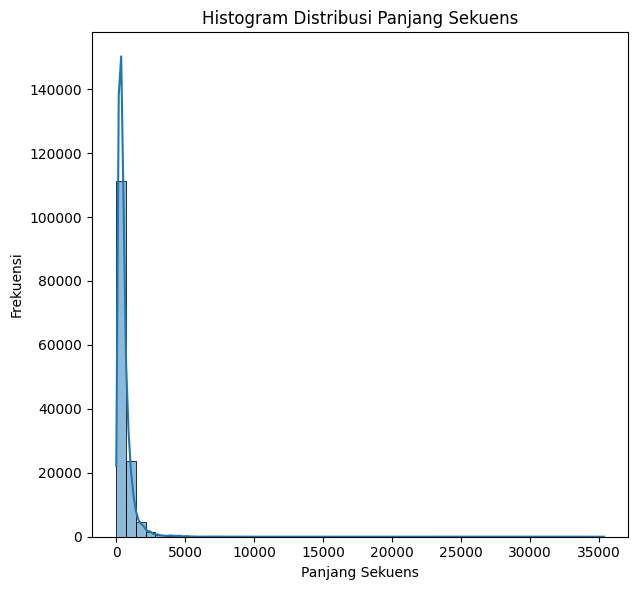
Kuartil ketiga (Q3) adalah nilai yang membagi 75% data terkecil dalam *dataset*. Dalam konteks panjang sekuens protein, nilai kuartil ketiga sebesar 654 menunjukkan bahwa 75% sekuens memiliki panjang kurang dari 654 asam amino.

1. **Max (Maksimum)**

Nilai maksimum adalah nilai terbesar dalam *dataset*. Dalam hal panjang sekuens protein, nilai maksimum sebesar 35.375 menunjukkan panjang sekuens terpanjang yang ada dalam *dataset*.

1. **Visualisasi Distribusi Panjang Sekuens**

Histogram digunakan sebagai alat visual untuk memahami distribusi panjang sekuens. Histogram membagi rentang panjang sekuens ke dalam *interval* (*bins*) dan menunjukkan frekuensi kemunculan setiap interval, dimana dalam pembuatan histogram kami membuat ke dalam interval (*bins*) sejumlah 50. Hal ini memberikan gambaran visual tentang sebaran panjang sekuens yang dapat dilihat pada Gambar 3.3.



Gambar 3.3 Visualisasi Distribusi Panjang Sekuens

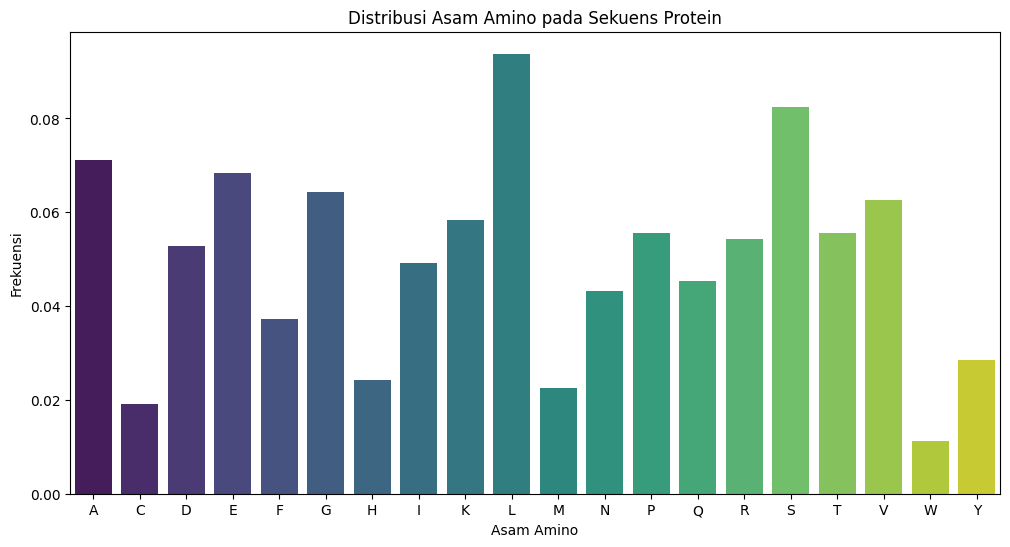
3.2.3.2 Analisis Distribusi Asam Amino pada Sekuens Protein

Pada tahap ini, dilakukan analisis untuk mengetahui sebaran frekuensi asam amino pada sekuens protein dalam *dataset*. Analisis ini penting untuk memahami distribusi relatif dari setiap asam amino yang digunakan dalam sekuens protein. Dengan dilakukannya analisis ini, maka dapat diidentifikasi asam amino yang paling umum atau jarang ditemukan dalam *dataset* serta pola distribusinya. Distribusi asam amino pada sekuens protein dalam *dataset* dapat dilihat pada Tabel 3.10.

Tabel 3.10 Distribusi Asam Amino pada Panjang Sekuens

|  |  |  |
| --- | --- | --- |
| **Huruf Asam Amino** | **Jumlah Huruf** | **Frekuensi Huruf** |
| A | 5.601.393 | 0,071129 |
| C | 1.504.314 | 0,019102 |
| D | 4.151.113 | 0,052713 |
| E | 5.388.873 | 0,068430 |
| F | 2.930.638 | 0,037215 |
| G | 5.061.691 | 0,064276 |
| H | 1.914.456 | 0,024311 |
| I | 3.879.567 | 0,049265 |
| K | 4.594.229 | 0,058340 |
| L | 7.386.109 | 0,093792 |
| M | 1.776.672 | 0,022561 |
| N | 3.400.751 | 0,043184 |
| P | 4.382.337 | 0,055649 |
| Q | 3.563.728 | 0,045254 |
| R | 4.283.020 | 0,054388 |
| S | 6.493.284 | 0,082455 |
| T | 4.378.528 | 0,055601 |
| V | 4.931.367 | 0,062621 |
| W | 888.449 | 0,011282 |
| Y | 2.239.153 | 0,028434 |

Berdasarkan Tabel 3.10, dapat ditampilkan visualisasi distribusi asam amino pada sekuens protein yang dapat dilihat pada Gambar 3.4.



Gambar 3.4 Visualisasi Distribusi Asam Amino pada Sekuens Protein

Pada Gambar 3.4 menunjukkan distribusi frekuensi berbagai jenis asam amino dalam sekuens protein. Dari grafik ini, asam amino L (Leusin) memiliki frekuensi tertinggi dalam sekuens protein, diikuti oleh asam amino A (Alanin), E (Glutamat), dan S (Serin) yang juga memiliki frekuensi yang relatif tinggi. Sebaliknya, asam amino C (Sistein) dan W (Triptofan) memiliki frekuensi yang sangat rendah, sementara asam amino lainnya seperti G (Glycin), H (Histidin), K (Lisin), T (Treonin), dan Y (Tirosin) berada di tengah-tengah. Selain itu, dapat disimpulkan bahwa asam amino yang memiliki frekuensi tinggi, seperti Leusin, dapat memainkan peran penting dalam struktur dan fungsi protein, dimana Leusin berperan penting dalam proses pembentukan protein dan metabolisme otot. Tingginya frekuensi Leusin mungkin menunjukkan peran protein ini dalam aktivitas metabolik tinggi atau fungsi struktural tertentu. Sebaliknya, asam amino dengan frekuensi rendah, seperti Sistein dan Triptofan, mungkin kurang terlibat dalam fungsi spesifik protein ini atau perannya lebih terkait dengan kondisi tertentu yang jarang terjadi. Dengan adanya analisis distribusi asam amino ini, maka dapat memberikan wawasan penting mengenai karakteristik dari sekuens protein yang dianalisis, sehingga dapat membantu dalam memprediksi fungsi dari protein tersebut.

3.2.3.3 Analisis Frekuensi Kelas Fungsi

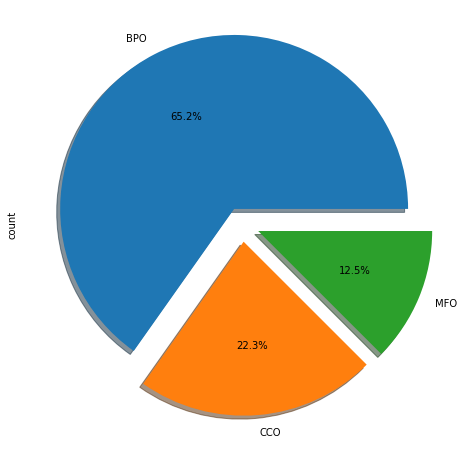
Pada tahap ini, dilakukan analisis untuk mengetahui sebaran frekuensi ID protein dalam setiap kelas fungsi (BPO, CCO, MFO). Informasi ini dapat memberikan wawasan tentang seimbangnya *dataset* dalam hal distribusi kelas fungsi. Langkah-langkah analisis melibatkan:

1. **Jumlah Unik Kelas Fungsi**

Menghitung jumlah kelas fungsi unik (BPO, CCO, MFO) dalam *dataset* memberikan pemahaman tentang variasi kelas fungsi yang tercakup. Dalam kasus ini, terdapat 3 kelas fungsi unik.

1. **Frekuensi Relatif Kelas Fungsi**

Menghitung frekuensi relatif masing-masing kelas fungsi dibandingkan dengan total jumlah protein membantu memahami proporsi relatif dari setiap kelas fungsi dalam *dataset*. Hasil dari frekuensi tersebut dapat dilihat pada Gambar 3.5.



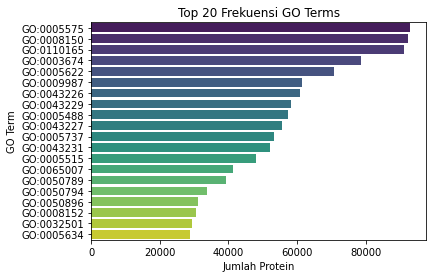
Gambar 3.5 Frekuensi Relatif Kelas Fungsi

* Kelas Fungsi BPO *(Biological Process)* memiliki frekuensi relatif sekitar 65.2%, menunjukkan bahwa lebih dari setengah protein terkait dengan proses biologis.
* Kelas Fungsi CCO *(Cellular Component)* memiliki frekuensi relatif sekitar 22.3%, menunjukkan proporsi protein yang terkait dengan komponen sel.
* Kelas Fungsi MFO *(Molecular Function)* memiliki frekuensi relatif sekitar 12.5%, menunjukkan proporsi protein yang terkait dengan fungsi molekuler.

Berdasarkan informasi yang telah disampaikan diatas, dapat memberikan gambaran tentang sebaran proporsi kelas fungsi dalam *dataset* dan membantu peneliti memahami distribusi fungsi biologis yang tercakup dalam data protein. Proporsi yang berbeda dapat mengindikasikan variasi peran biologis dalam *dataset* tersebut.

3.2.3.4 Analisis Frekuensi Kelas Term GO

Analisis frekuensi kelas GO\_Term adalah teknik yang digunakan untuk mengetahui GO\_Term yang paling sering muncul dalam *dataset*. GO\_Term adalah istilah yang digunakan untuk menggambarkan fungsi protein. Analisis ini dapat digunakan untuk memahami fungsi protein pada *dataset* tersebut yang dapat dilihat pada Gambar 3.6.



Gambar 3.6 Analisis Frekuensi Kelas GO\_Term

Pada Gambar 3.6 menunjukkan hasil “Analisis Frekuensi Kelas GO\_Term” pada *dataset* fungsi protein, di mana untuk melihat deskripsi dari 5 GO\_Term yang paling sering muncul tersebut dapat dilihat pada Tabel 3.11 dan 20 GO\_Term lengkapnya dapat dilihat pada Lampiran 1.

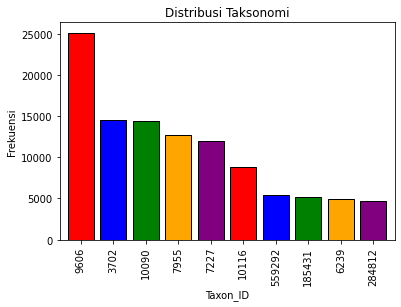
Tabel 3.11 Deskripsi GO\_Term

|  |  |
| --- | --- |
| **GO term** | **Definisi** |
| GO:0005575 - *protein kinase activity* | Protein kinase adalah enzim yang mengkatalisis fosforilasi protein. Fosforilasi adalah proses penambahan gugus fosfat ke protein. Gugus fosfat ini dapat mengubah struktur dan fungsi protein. |
| GO:0008150 - *transferase activity* | Transferase adalah enzim yang mengkatalisis transfer gugus dari satu molekul ke molekul lain. Gugus yang ditransfer dapat berupa gugus fosfat, gula, atau amino. |
| GO:0110165 - *cytoplasmic localization* | Protein yang memiliki GO\_Term ini ditemukan di sitoplasma, yaitu bagian sel yang terletak di luar nukleus. |
| GO:0003674 - *catalytic activity* | Protein yang memiliki GO\_Term ini memiliki aktivitas katalitik, yaitu kemampuan untuk mempercepat reaksi kimia. |
| GO:0005622 - *protein binding* | Protein yang memiliki GO\_Term ini dapat berikatan dengan protein lain. Ikatan ini dapat bersifat sementara atau permanen. |

GO\_Term ini menunjukkan bahwa protein dalam *dataset* fungsi protein terutama terlibat dalam metabolisme, transportasi, dan ikatan. GO\_Term ini dapat digunakan untuk mengembangkan model yang dapat memprediksi struktur fungsi protein.

3.2.3.5 Analisis Frekuensi Taksonomi

Analisis frekuensi taksonomi dilakukan untuk memahami distribusi protein berdasarkan kelompok taksonominya. Analisis ini dapat memberikan informasi tentang kelompok taksonomi yang paling banyak diwakili dalam data, serta protein-protein yang paling banyak dibagikan di berbagai kelompok taksonomi. Analisis ini dapat digunakan untuk mempelajari berbagai hal, seperti keanekaragaman hayati, pola penyebaran, dan hubungan antara spesies, yang dapat dilihat pada Gambar 3.7.



Gambar 3.7 Analisis Frekuensi Taksonomi

Pada Gambar 3.7 menunjukkan histogram distribusi ID taksonomi dalam data taksonomi. Histogram ini memiliki sumbu *horizontal* yang menunjukkan ID taksonomi dan sumbu vertikal yang menunjukkan jumlah protein yang memiliki ID taksonomi tersebut. Pada sumbu *horizontal*, dapat dilihat bahwa ID taksonomi berkisar dari 1 hingga 3156. ID taksonomi yang lebih rendah menunjukkan kelompok taksonomi yang lebih primitif, sedangkan ID taksonomi yang lebih tinggi menunjukkan kelompok taksonomi yang lebih kompleks. Pada sumbu vertikal, dapat dilihat bahwa jumlah protein yang memiliki ID taksonomi tertentu bervariasi secara signifikan. Kelompok taksonomi yang paling banyak diwakili dalam data adalah 9606, yang merupakan ID taksonomi untuk Homo sapiens, diikuti dengan ID taksonomi 3702 (*Arabidopsis thaliana*), 10090 (*mus musculus*), 7955 (*Danio rerio*), 7227 *(Mus musculus)*, dan seterusnya. Sepuluh Takson\_ID yang memiliki jumlah frekuensi terbanyak dapat dilihat pada Tabel 3.12.

Tabel 3.12 Analisis Frekuensi Taksonomi

|  |  |
| --- | --- |
| **Taxon ID** | **Jumlah Protein\_ID** |
| 9606 | 25125 |
| 3702 | 14461 |
| 10090 | 14384 |
| 7955 | 12671 |
| 7227 | 12020 |
| 10116 | 8817 |
| 559292 | 5469 |
| 185431 | 5209 |
| 6239 | 4915 |
| 284812 | 4634 |

## 3.3 Analisis *Data Preprocessing*

*Preprocessing data* merupakan tahapan yang digunakan untuk membersihkan dan mengubah data mentah menjadi bentuk yang lebih mudah dipahami dan dapat digunakan untuk dianalisis.

3.3.1 Penanganan *Missing Values*

Penanganan nilai yang hilang merupakan aspek kritis dalam analisis data, terutama ketika bekerja dengan *dataset* bioinformatika. Langkah-langkah yang efektif dalam penanganan nilai yang hilang dapat memastikan integritas dan keandalan hasil analisis. Identifikasi atribut dengan nilai yang hilang adalah langkah awal yang penting. Dengan menggunakan metode seperti ‘***isnull()***’, kita dapat mengidentifikasi dimana nilai-nilai yang hilang terdapat dalam *dataset*. Hasil dari identifikasi dalam penanganan *missing value* yang dilakukan dapat dilihat pada Tabel 3.13.

Tabel 3.13 Penanganan Missing Values

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Protein\_ID** | **GO\_Term\_ID** | **Ontology** | **Taxon\_ID** | **Sequence** |
| FALSE | FALSE | FALSE | FALSE | FALSE |
| FALSE | FALSE | FALSE | FALSE | FALSE |
| FALSE | FALSE | FALSE | FALSE | FALSE |
| FALSE | FALSE | FALSE | FALSE | FALSE |
| **....** | **....** | **....** | **....** | **....** |
| FALSE | FALSE | FALSE | FALSE | FALSE |
| FALSE | FALSE | FALSE | FALSE | FALSE |
| FALSE | FALSE | FALSE | FALSE | FALSE |
| FALSE | FALSE | FALSE | FALSE | FALSE |
| FALSE | FALSE | FALSE | FALSE | FALSE |

Dalam Tabel 3.13, hasil dari *isnull*() menunjukkan bahwa tidak ada nilai yang hilang dalam kolom-kolom seperti 'Protein\_ID', 'GO\_Term\_ID', 'Ontology', 'Taxon\_ID', ‘Sequence’, ‘Weight’, dan 'Sequence\_len' pada keseluruhan *dataset* dengan 536.3863 baris. Kondisi ini memberikan kepastian bahwa *dataset* ini memiliki integritas data yang baik dan siap untuk dilanjutkan ke tahap analisis berikutnya. Penerapan strategi penggantian nilai sering melibatkan penggunaan statistik deskriptif, seperti rata-rata atau median, untuk mengisi nilai yang hilang dan menjaga konsistensi *dataset*. Penggunaan imputasi bergantung pada jenis data dan konteks biologis yang mendasarinya. Selain itu, jika nilai yang hilang tidak dapat digantikan dengan cara yang akurat, atau jika jumlahnya signifikan, pertimbangkan penghapusan baris atau kolom yang mengandung nilai yang hilang untuk memastikan kualitas analisis. Dengan pendekatan yang cermat terhadap nilai yang hilang, kita dapat memperoleh hasil analisis yang lebih akurat dan dapat diandalkan dari *dataset* biologis yang kompleks.

3.3.2 Tokenisasi Sekuens Protein

Tokenisasi adalah salah satu tahapan penting yang dilakukan pada *data preprocessing*, dimana dalam hal ini dilakukan proses membagi ataupun memecah suatu kalimat, frasa, paragraf, atau beberapa dokumen teks menjadi unit yang lebih kecil [50]. Token bisa berupa kata, subkata, atau bahkan karakter, tergantung pada algoritma yang digunakan dalam proses tokenisasi. Tujuan dari tokenisasi adalah untuk menguraikan data yang tidak terstruktur menjadi potongan-potongan informasi numerik yang dapat digunakan dalam model *deep learning*. Dalam konteks sekuens asam amino untuk memprediksi fungsi protein, tokenisasi yang digunakan adalah algoritma berbasis karakter, dimana dilakukan metode tokenisasi dengan memecah teks menjadi karakter individu. Dalam konteks sekuens asam amino, untuk memprediksi fungsi protein, setiap asam amino dalam sekuens akan dianggap sebagai satu unit atau token yang berdiri sendiri. Misalnya, apabila terdapat sekuens asam amino "ACDE", algoritma berbasis karakter akan memecahnya menjadi "A", "C", "D", dan "E" sebagai unit-unit terpisah. Dengan memecah sekuens menjadi karakter individu, model *deep learning* dapat memproses dan memahami pola-pola di antara asam amino tersebut dengan lebih baik [51]. Berikut ini adalah contoh tokenisasi berbasis karakter terhadap sekuens protein yang dapat dilihat pada Tabel 3.14.

Tabel 3.14 Tokenisasi berbasis Karakter

|  |  |
| --- | --- |
| **Sebelem Tokenisasi** | **Tokenisasi** |
| ‘A’, C’, ‘D’, ‘E’, ‘F’, ‘G’, ‘H’, ‘I’, ‘K’, ‘L’, ‘M’, ‘N’, ‘P’, ‘Q’, ‘R’, ‘S’, ‘T’, ‘V’, ‘W’, ‘Y’ | ‘A : 1’, C : 2’, ‘D : 3’, ‘E : 4’, ‘F : 5’,  ‘G : 6’, ‘H : 7’, ‘I : 8’, ‘K : 9’, ‘L : 10’, ‘M : 11’, ‘N : 12’, ‘P : 13’, ‘Q : 14’,  ‘R : 15’, ‘S : 16’, ‘T : 17’, ‘V : 18’,  ‘W : 19’, ‘Y : 20’ |

Berdasarkan Tabel 3.14 diatas, berikut ini adalah contoh daripada hasil tokenisasi pada sekuens protein yang dapat dilihat pada Tabel 3.15.

Tabel 3.15 Hasil Tokenisasi pada Sekuens Protein

|  |  |
| --- | --- |
| **Sekuens Protein** | **Hasil Tokenisasi pada Sekuens Protein** |
| MTMDKSELVQKAKLAEQAERYDDMA | [18, 8, 18, 12, 7, 2, 4, 1, 6, 13, 7, 3, 7, 1, 3, 4, 13, 3, 4, 10, 16, 12, 12, 18, 3] |
| MDDREDLVYQAKLAEQAERYDEMVE | [18, 12, 12, 10, 4, 12, 1, 6, 16, 13, 3, 7, 1, 3, 4, 13, 3, 4, 10, 16, 12, 4, 18, 6, 4] |

Dengan demikian, penggunaan tokenisasi berbasis karakter terhadap sekuens protein menjadi karakter-karakter individu memainkan peran penting dalam mengembangkan teknologi baru untuk memahami lebih dalam tentang protein.

3.3.3 *Padding* pada Sekuens Protein

*Padding* adalah tahap *preprocessing* yang dilakukan untuk membuat setiap sekuens protein memiliki panjang yang seragam sebelum dimasukkan ke dalam model. Dalam konteks *deep learning* dan pengolahan data urutan seperti teks atau urutan asam amino pada protein, panjang urutan seringkali bervariasi. Oleh karena itu, *padding* diperlukan karena model *deep learning* harus memiliki *input*an (sekuens protein) dengan ukuran yang seragam. *Padding* dilakukan dengan menambahkan nilai 0 di awal atau akhir sekuens yang lebih pendek hingga mencapai panjang maksimum yang ditentukan. *Padding* juga dapat memotong sekuens yang lebih panjang agar sesuai dengan panjang maksimum. Hasil akhirnya adalah semua sekuens memiliki panjang yang seragam. *Padding* biasanya dilakukan dengan menambahkan nilai 0 di awal sekuens yang lebih pendek, yang disebut *pre-padding*, atau di akhir sekuens yang lebih pendek, yang disebut *post-padding*. Berikut ini adalah contoh penggunaan *padding* dalam sekuens protein yang dapat dilihat pada Tabel 3.16, dimana panjang maksimal sekuens protein pada contoh tersebut adalah 5.

Tabel 3.16 Padding pada Sekuens Protein

|  |  |  |
| --- | --- | --- |
| ***Sekuens Protein*** | ***Pre-padding*** | ***Post-padding*** |
| [18, 8, 18, 12, 7, 6] | [18, 8, 18, 12, 7] | [18, 8, 18, 12, 7] |
| [12, 10, 4] | [0, 0, 12, 10, 4] | [12, 10, 4, 0, 0] |

Saat melakukan *padding* pada sekuens protein, jika panjang sekuens melebihi panjang maksimum yang diinginkan atau panjang maksimum yang ditetapkan untuk di*input* ke model, maka sekuens protein tersebut akan dipotong menjadi panjang yang diinginkan. Proses pemotongan ini biasanya dilakukan dengan menghapus bagian akhir dari sekuens protein yang melebihi panjang maksimum yang ditetapkan. Meskipun informasi tentang bagian yang dipotong dari sekuens protein tersebut hilang, namun informasi yang terkandung dalam bagian yang dipertahankan masih dapat memberikan cukup konteks dan fitur untuk model *deep learning* dalam melakukan prediksi.

Namun, jika menggunakan panjang sekuens terpanjang sebagai panjang maksimum untuk *padding*, hal ini dapat memperlambat proses komputasi. Hal ini disebabkan oleh fakta bahwa semakin panjang sekuens yang diproses oleh model, semakin banyak memori dan waktu komputasi yang diperlukan untuk melatih dan menjalankan model. Sebagai contoh, jika menentukan panjang maksimal *padding* berdasarkan sekuens terpanjang dalam *dataset*, maka setiap sekuens yang lebih pendek harus dipenuhi dengan nilai nol hingga mencapai panjang tersebut, yang dapat meningkatkan kompleksitas komputasi secara signifikan. Oleh karena itu, penting untuk memilih panjang maksimal *padding* yang seimbang, yang cukup besar untuk mempertahankan informasi yang penting tetapi tidak terlalu panjang sehingga menghambat efisiensi komputasi.

Berdasarkan penelitian sebelumnya, ProtBERT dan ProtT5 menggunakan *padding* pada sekuens protein untuk menghasilkan vektor *embedding* dengan panjang 1024 [52]. Panjang vektor *embedding* 1024 ini telah terbukti efektif dalam representasi sekuens protein. Dengan menggunakan vektor *embedding* yang konsisten sepanjang 1024, model dapat memahami dan memproses informasi dari sekuens protein dengan cara yang optimal. Kemudian, pada penelitian ProtXLNet [36], ukuran vektor *embedding* 1024 digunakan dengan ProtBERT sebagai model bahasa protein karena penggunaan *batch size* yang kecil dalam penelitian ProtXLNet. Hal ini memungkinkan untuk menyertakan urutan yang lebih pendek dan tetap mempertahankan panjang *embedding* yang diinginkan. *Protein Language Model* (PLM) memerlukan setiap *minibatch* memiliki panjang urutan sekuens yang seragam dan penggunaan *padding* menjadi kunci dalam mengolah sekuens protein yang lebih panjang. Dengan demikian, penelitian ini menetapkan ukuran *padding* pada sekuens protein untuk memastikan bahwa setiap urutan memiliki panjang yang seragam sebelum diproses menjadi vektor *embedding* dengan panjang 1024. Diharapkan model yang dikembangkan dapat bekerja dengan baik dalam mengolah sekuens protein yang memiliki variasi panjang yang signifikan, namun tetap efisien dalam penggunaan memori dan waktu komputasi.

## 3.4 Analisis Model LSTM

Penggunaan model *Long Short-Term Memory* (LSTM) dalam prediksi fungsi protein sangat efektif karena kemampuannya mengatasi masalah *vanishing* *gradient* yang sering muncul pada *Recurrent Neural Networks* (RNN). *Vanishing gradient* merupakan kendala signifikan dalam melatih RNN, terutama ketika berhadapan dengan urutan data yang panjang, seperti urutan asam amino dalam protein. *Recurrent Neural Networks* (RNN) dirancang untuk memproses data sekuensial dengan mempertahankan informasi dari langkah-langkah sebelumnya dalam memori internalnya. Namun, ketika urutan data menjadi panjang, *gradient* yang dihitung selama pelatihan dapat mengecil secara eksponensial saat diteruskan kembali melalui banyak lapisan. Hal ini menyebabkan penurunan efektivitas *gradient* dalam memperbarui bobot jaringan untuk langkah-langkah awal dalam urutan.

Sebagai contoh, sebuah urutan asam amino dalam protein sebagai berikut: “MTVPKSRETQAILD”. Dalam pengolahan urutan asam amino tersebut, RNN memiliki kelemahan dalam menjaga informasi jangka panjang. Saat RNN memproses asam amino 'M' pada langkah pertama, informasi ini cenderung hilang atau melemah ketika mencapai asam amino 'D' di akhir urutan. Sebaliknya, LSTM mengatasi masalah ini dengan memperkenalkan sel memori yang dapat menyimpan informasi penting lebih lama. LSTM memiliki tiga gerbang utama: *input*, *output*, dan *forget* yang mengatur informasi yang masuk, keluar, dan dilupakan dari sel memori. Misalnya, saat LSTM memproses 'M', informasi penting tentang 'M' dapat disimpan dalam sel memori dan digunakan saat memproses 'D', sehingga hubungan jangka panjang tetap terjaga dan fungsi protein dapat dipahami dengan lebih baik.

Adapun alasan penggunaan dua *layer* dalam arsitektur model LSTM adalah untuk meningkatkan kapasitas model dalam memahami representasi yang lebih kompleks dari data sekuensial. Setiap tambahan *layer* memungkinkan model untuk mengekstraksi fitur-fitur yang lebih abstrak dari urutan asam amino. *Layer* pertama LSTM bertujuan untuk mengidentifikasi pola dasar dan fitur-fitur kritis dari urutan, sementara *layer* kedua memanfaatkan *output* dari *layer* sebelumnya untuk membangun representasi yang lebih tingkat dari informasi hierarkis yang tersedia. Dengan demikian, model tidak hanya mampu memahami korelasi jangka panjang antar asam amino secara lebih efektif, tetapi juga mengatasi tantangan *vanishing gradient* dengan lebih baik melalui pengelolaan informasi yang lebih kompleks dan terstruktur di berbagai tingkat representasi.

## 3.5 Analisis *Embedding*

Pada sub bab ini akan dijelaskan analisis terkait dengan metode *embedding* yang akan digunakan pada eksperimen penelitian ini. Metode *embedding* yang akan digunakan yaitu, *embedding layer* dari *Tensorflow. Keras*, *T5-Embedding*, dan ProtBERT.

3.5.1 Analisis *Layer Embedding* pada *Tensorflow.Keras*

Dalam penelitian ini, kami menggunakan lapisan *embedding* yang disediakan oleh *TensorFlow.Keras* untuk eksperimen pertama dalam penelitian. Lapisan *embedding* adalah salah satu komponen penting dalam pengembangan dan pelatihan model pembelajaran mendalam, yang dirancang untuk mengubah data *input* berbentuk *integer* menjadi representasi vektor berdimensi tetap. Penggunaan lapisan ini sangat umum dalam pemrosesan bahasa alami (*Natural Language Processing*), dan dalam penelitian ini, lapisan *embedding* digunakan untuk melakukan *embedding* pada sekuens protein. Urutan protein yang direpresentasikan sebagai indeks-indeks yang mewakili asam amino diubah menjadi vektor-vektor berdimensi tetap. Representasi ini memungkinkan model pembelajaran mendalam untuk mempelajari dan mengidentifikasi pola serta hubungan antara asam amino dalam sekuens protein. Vektor-vektor ini mengkodekan fitur-fitur penting dari asam amino, sehingga memudahkan model untuk melakukan berbagai tugas analisis, seperti melakukan prediksi fungsi protein atau tugas lainnya yang berhubungan dengan pemrosesan sekuens protein. Dengan demikian, lapisan *embedding* memainkan peran kunci dalam meningkatkan kemampuan model untuk menangkap informasi kompleks dari sekuens protein, yang sangat berguna dalam penelitian ini.

3.5.2 Analisis *T5-Embedding*

*T5-Embedding* adalah model bahasa yang dikembangkan berdasarkan arsitektur T5 (*Text-To-Text Transfer Transformer*). Meskipun dirancang untuk memahami dan menghasilkan teks berbahasa alami, model ini juga dapat diterapkan dalam konteks prediksi fungsi protein dengan menggunakan representasi *embedding*-nya. Proses kerja *T5-Embedding* dimulai dengan tokenisasi sekuens protein, dimana setiap asam amino dipisahkan menjadi token-token individual. Setelah tokenisasi, setiap token akan diolah pada lapisan *encoder* untuk menghasilkan vektor berdimensi tinggi. Vektor ini merepresentasikan sekuens protein dalam sebuah matriks berukuran n x d, dimana n adalah panjang total sekuens dan d adalah dimensi panjang vektor dengan panjang vektor adalah 1024. Setelah melewati tahap *encoder*, representasi vektor ini kemudian disimpan sebagai data latih dalam format *.npy* dengan nama *file train\_ids.npy*, *train\_embeds.npy*, serta sebagai data uji dengan format *test\_ids.npy* dan *test\_embeds.npy*. *File*-*file* ini nantinya akan digunakan sebagai *input* untuk model *Long Short-Term Memory* (LSTM). LSTM, yang terkenal dengan kemampuannya dalam menangkap pola temporal dan ketergantungan jangka panjang dalam data sekuensial, digunakan untuk memproses data ini. Dengan menggabungkan representasi awal dari *T5-Embedding* dengan model LSTM dalam analisis sekuensial, maka model dapat menghasilkan prediksi yang akurat terkait dengan fungsi protein berdasarkan informasi sekuens yang diberikan.

3.5.3 Analisis *ProtBERT*

Dalam subbab analisis *Protein Language Model* (PLM), kami melakukan eksplorasi terhadap potensi dan kinerja model PLM, khususnya ProtBERT, dalam menangani kompleksitas urutan protein. ProtBERT dipilih sebagai model dasar karena kemampuannya yang mendalam dalam analisis urutan asam amino, menggunakan arsitektur *transformer* adaptif, dan dilatih dengan *dataset* yang luas. Sebagai contoh implementasi, kami menerapkan ProtBERT untuk memprediksi fungsi protein dengan mengolah sekuens protein yang diberikan. Sekuens protein diubah menjadi representasi numerik oleh ProtBERT, menghasilkan *embedding* yang mencerminkan fitur-fitur penting dari protein tersebut. Keunggulan utama ProtBERT terletak pada kemampuannya dalam memahami konteks urutan asam amino, yang memungkinkannya untuk mengidentifikasi pola-pola penting yang menentukan fungsi protein secara akurat.

Selanjutnya, kami menjelajahi integrasi ProtBERT dengan teknik lain seperti LSTM (*Long Short-Term Memory*), dengan harapan menghasilkan model yang lebih kuat dalam memahami pola-pola kompleks dalam urutan protein. Gabungan dari kedua pendekatan ini diharapkan memberikan pemahaman yang lebih mendalam tentang sifat dan fungsi protein, serta meningkatkan kinerja model dalam berbagai tugas bioinformatika seperti prediksi fungsi protein atau analisis efek mutasi. Analisis eksperimen ini bertujuan untuk mengevaluasi apakah integrasi antara ProtBERT dan LSTM dapat memberikan peningkatan kinerja yang signifikan dalam masalah-masalah bioinformatika. Dengan pendekatan ini, kami berharap mengungkap cara yang lebih efektif untuk memahami dan memanfaatkan informasi dari urutan protein, membuka peluang untuk penemuan baru dalam bidang biologi molekuler dan bioinformatika.

## 3.8 AnalisisPencarian Sekuens Protein

Dalam penelitian ini, kami memilih untuk menggunakan *diamond search* sebagai alat pencarian sekuens protein untuk mendukung proses prediksi fungsi protein. *Diamond search* merupakan program penyelarasan *throughput* tinggi yang dirancang untuk membandingkan *file* sekuensing DNA dengan *file* sekuens referensi protein dengan kecepatan dan akurasi tinggi. *Diamond search* adalah alat pencarian sekuens protein yang memanfaatkan pendekatan berbasis pengoptimalan *database* untuk meningkatkan kecepatan dan sensitivitas dalam pencarian. Metode ini didasarkan pada konsep pengindeksan sekuens protein yang diubah menjadi format yang dioptimalkan, memungkinkan program untuk melakukan penyejajaran sekuens dengan cepat dan akurat. Hal ini didasarkan pada berbagai keunggulan yang dimiliki *diamond search* dibandingkan dengan alat pencarian sekuens lainnya, seperti BLAST.

*Diamond search* menggunakan beberapa teknik canggih, seperti penggunaan dua indeks, *alfabet* yang disederhanakan, dan *spaced seeds*, yang memungkinkan pemrosesan data dengan efisien dan hemat memori. Pendekatan dua indeks membantu menghemat memori dan mempercepat akses data, sementara teknologi *SIMD-accelerated Smith-Waterman* mempercepat proses pencarian dengan mempertahankan akurasi yang tinggi. Selain itu, *diamond search* membagi ruang *seed* menjadi 1.024 bagian [46], yang memungkinkan pemrosesan sebagian data pada satu waktu, sehingga lebih efisien dalam penggunaan memori.

*Diamond search* lebih cepat daripada BLAST karena menggunakan dua indeks untuk mencocokkan data dengan lebih efisien. Dua indeks yang digunakan dalam *diamond search* adalah indeks untuk *query* dan referensi. *Diamond search* menggunakan pendekatan *double-indexing* di mana kedua *query* dan referensi diindeks untuk mencocokkan data dengan lebih efisien. Teknik ini membantu program menemukan pola dan lokasi data dengan lebih cepat serta mengurangi penggunaan memori utama. Selain itu, *diamond search* menggunakan pola awal yang dipilih dengan hati-hati untuk meningkatkan kecepatan dan ketepatan pencarian. Pola awal penggunaan urutan pendek spesifik yang dipilih sebagai titik awal untuk proses pencocokan sekuens [46].

Dalam konteks *diamond search*, pola ini digunakan untuk memfokuskan pencarian pada bagian-bagian sekuens yang lebih mungkin memiliki kesamaan dengan sekuens dalam *database*. Dengan memilih pola awal yang spesifik, *diamond search* dapat mempercepat proses pencocokan dengan menghindari pencarian yang tidak relevan dan fokus pada bagian sekuens yang lebih mungkin cocok. Selain itu, *diamond search* mengelola indeksnya satu per satu, yang berarti setelah selesai menggunakan satu indeks, program membebaskan memori yang digunakan sebelum beralih ke indeks berikutnya. Pendekatan ini tidak hanya mempercepat pencarian tetapi juga memastikan penggunaan memori yang lebih efisien, sehingga memungkinkan pemrosesan data dalam jumlah besar tanpa mengorbankan performa. Dengan kombinasi teknik-teknik ini, *diamond search* dapat memberikan hasil pencarian sekuens protein yang cepat dan akurat, menjadikannya alat yang sangat efektif dalam analisis bioinformatika.

## 3.9 Analisis Metriks Evaluasi

Metriks evaluasi adalah teknik yang penting dalam mengevaluasi kinerja model prediksi struktur fungsi protein. Teknik ini membagi sampel data protein menjadi dua set data, yaitu data *training* dan *targets*. Data *training* digunakan untuk melatih model, sementara *targets* digunakan untuk mengevaluasi kinerja model. Dalam konteks prediksi struktur fungsi protein, data *training* berisi data protein dengan struktur fungsi yang diketahui, yang digunakan untuk melatih model prediksi. Setelah model dilatih, kinerja model dievaluasi menggunakan *dataset* *targets* [53]. Kinerja model dapat dinilai menggunakan berbagai matrik evaluasi, termasuk *accuracy*, *precision, recall,* dan *F1-Score*.

1. ***Binary accuracy*** adalah rasio jumlah protein yang diprediksi dengan benar terhadap jumlah total protein. Dalam penelitian ini, relevansi *binary accuracy* sangat tinggi karena setiap protein dapat memiliki beberapa fungsi yang berbeda, sehingga masalah ini dikategorikan sebagai *multi-label classification*. Hal ini terjadi karena setiap protein dapat memiliki beberapa fungsi yang mungkin, sehingga pendekatan yang tepat adalah dengan menggunakan *binary accuracy*. Berikut rumus *binary accuracy* sebagai berikut:
2. ***Precision*** adalah rasio jumlah protein yang diprediksi dengan benar memiliki struktur fungsi yang benar terhadap jumlah protein yang diprediksi memiliki struktur fungsi tersebut. Rumus *precision* adalah sebagai berikut:
3. ***Recall*** adalah rasio jumlah protein yang diprediksi dengan benar memiliki struktur fungsi yang benar terhadap jumlah protein yang sebenarnya memiliki struktur fungsi tersebut. Rumus *recall* adalah sebagai berikut:
4. ***F1-score*** adalah kombinasi dari *recall* dan *precision*. *F1-score* dapat digunakan untuk menyeimbangkan antara *recall* dan *precision*. Rumus *F1-score* adalah sebagai berikut:
5. **AUC (*Area Under Curve*)** adalah metrik evaluasi yang digunakan untuk mengukur seberapa baik suatu model dapat membedakan antara dua atau lebih kategori yang berbeda. Dalam hal ini, AUC digunakan untuk mengevaluasi seberapa baik model melakukan prediksi fungsi protein dalam membedakan antara protein yang memiliki fungsi tertentu dengan yang tidak memiliki fungsi tersebut. Dengan AUC, peneliti dapat mengukur kemampuan model dalam mengklasifikasikan protein ke dalam kategori fungsi yang benar.

Nilai metrik-metrik tersebut dapat digunakan untuk menilai kinerja model prediksi struktur fungsi protein. Nilai metrik yang tinggi menunjukkan bahwa model tersebut dapat memprediksi struktur fungsi protein dengan baik [54]. Dengan memperhatikan metrik-metrik evaluasi ini, diharapkan bahwa model prediksi struktur fungsi protein yang dikembangkan dapat memberikan prediksi yang akurat dan dapat diandalkan, sehingga dapat memberikan kontribusi yang signifikan dalam penelitian di bidang biologi molekuler.

## 3.10 Analisis Nilai *Hyperparameter* Model

Pada bagian ini dijelaskan terkait nilai *hyperparameter* yang akan digunakan dalam melakukan analisis *hyperparameter* model. *Hyperparameter* adalah parameter yang ditentukan sebelum proses pelatihan model *deep learning* dimulai. Nilai *hyperparameter* dapat mempengaruhi cara model mempelajari data dan bagaimana model dalam mengambil keputusan. Oleh karena itu, penting untuk memilih nilai *hyperparameter* yang tepat untuk meningkatkan performa model. Terdapat beberapa *hyperparameter* yang akan digunakan dapat dilihat pada Tabel 3.17.

Tabel 3.17 Analisis Hyperparameter

|  |  |  |
| --- | --- | --- |
| **No.** | ***Hyperparameter*** | **Nilai yang diuji** |
| 1. | *Optimizer* | Adam |
| 2. | *Batch size* | 32 |
| 3. | *Learning rate* | 0,001 |
| 4. | *Epochs* | 50 |
| 5. | Jumlah *layer* | 7 |

1. ***Optimizer***

Adam merupakan *optimizer* yang cukup sering digunakan dalam pelatihan jaringan saraf. Adam memiliki keunggulan dalam menangani masalah pelatihan yang kompleks dengan kecepatan konvergensi yang cepat. Selain itu juga dapat memanfaatkan perhitungan yang efisien dan skala parameter yang stabil untuk mengoptimalkan parameter.

1. ***Batch size***

Penggunaan *batch size* 32 dalam pelatihan model jaringan saraf untuk prediksi fungsi protein dipilih karena memberikan keseimbangan yang baik antara kecepatan pelatihan, kestabilan perhitungan, dan efisiensi penggunaan komputasi. *Batch size* 32 memberikan variasi data yang cukup di setiap iterasi, membantu model untuk belajar representasi yang lebih baik dan mencapai konvergensi lebih cepat tanpa memerlukan terlalu banyak iterasi. Selain itu, *batch size* ini memanfaatkan kemampuan paralelisme GPU secara efektif, memungkinkan *update* parameter yang lebih cepat dibandingkan *batch size* yang lebih kecil, dan tetap efisien dalam penggunaan memori dibandingkan *batch size* yang lebih besar.

1. ***Learning rate***

*Learning rate* yang digunakan dalam penelitian ini adalah 0,001. Nilai 0,001 menunjukkan perubahan yang kecil pada setiap iterasi. Nilai ini biasanya digunakan karena memberikan konvergensi yang baik tanpa menyebabkan divergensi pada banyak kasus. Pemilihan nilai *learning rate* penting dan juga memerlukan eksperimen untuk menemukan nilai optimal untuk model dan *dataset* tertentu.

1. ***Epochs***

*Epoch* adalah satu kali iterasi melalui seluruh *dataset* pelatihan yang bertujuan meningkatkan akurasi dan kinerja model dan juga model mungkin akan *overfitting* pada data latih. Jika jumlah *epoch* terlalu rendah, model mungkin tidak memiliki waktu yang cukup untuk mengekstrak pola yang lebih kompleks pada data latih. Maka analisis *epoch* saat dinaikkan dan diturunkan, pengaruh jumlah *epoch* optimal harus ditentukan dengan melakukan eksperimen dan validasi model pada data uji yang belum pernah dilihat sebelumnya. Pada penelitian ini *hyperparameter* *epoch* yang digunakan adalah 50.

1. **Jumlah *layer***

*Layer* dalam jaringan saraf tiruan adalah susunan *neuron* yang memproses *input* dan mentransmisikan hasilnya ke *layer* berikutnya, dengan masing-masing *layer* memiliki fungsi spesifik dalam pemrosesan data. Dalam penelitian ini, model yang dilatih menggunakan tujuh *layer* yang terdiri dari dua *layer* LSTM, tiga *layer dropout*, dan dua *layer dense*. *Layer* LSTM (*Long Short-Term Memory*) digunakan untuk menangani data urutan dengan kemampuan mengingat informasi jangka panjang. Pada model ini, dua *layer* LSTM memiliki ukuran *unit* 128 dan 64, masing-masing menangkap dan menyempurnakan informasi fitur secara bertahap. Tiga *layer dropout* digunakan dengan tingkat *dropout* 0.3 untuk mencegah *overfitting* dengan mengabaikan sejumlah *unit* secara acak selama *training*, membuat jaringan saraf bekerja lebih baik dan tidak terlalu bergantung pada beberapa *neuron* tertentu, sehingga hasil prediksi menjadi lebih akurat. Dua *layer dense* terakhir adalah *layer fully connected* yang melakukan klasifikasi atau regresi berdasarkan fitur yang telah diekstraksi. *Layer dense* pertama memiliki 64 *unit* yang bertugas mengumpulkan dan menyatukan informasi dari *layer* sebelumnya, dan *layer dense* kedua memiliki 1500 *unit* untuk prediksi fungsi protein sesuai jumlah label yang digunakan dalam penelitian.

## 3.11 Analisis Eksperimen

Pada bagian ini akan menganalisis hasil dari serangkaian eksperimen yang dilakukan untuk mengevaluasi performa berbagai metode prediksi fungsi protein, termasuk penggunaan metode *embedding* yang berbeda-beda namun dengan jumlah data yang sama, dengan berfokus pada akurasi prediksi dan efektivitas model dalam menafsirkan sekuens protein yang kompleks.

3.11.1 Prediksi Fungsi Protein menggunakan *Embedding* dari *TensorFlow.Keras* dan Model LSTM

Pada eksperimen melakukan prediksi fungsi protein menggunakan *embedding layer* dari *TensorFlow.Keras* menggunakan model LSTM, bertujuan untuk meningkatkan representasi fitur dari urutan protein serta meningkatkan akurasi klasifikasi hierarkis fungsi protein. Penggunaan lapisan *embedding* memungkinkan model untuk mempelajari representasi vektor yang lebih informatif dari urutan asam amino, sehingga dapat lebih baik dalam memahami struktur urutan protein dan meningkatkan kemampuan dalam memprediksi fungsi-fungsi kompleks yang terkait. Selain itu, eksperimen ini memanfaatkan informasi dari *embedding* untuk memperhitungkan dependensi kuat antar kelas dalam *Gene Ontology* (GO), yang menggambarkan hierarki fungsi-fungsi protein. Prediksi yang tepat untuk satu fungsi harus mempertimbangkan kemungkinan adanya fungsi-fungsi yang lebih spesifik atau lebih umum dalam hierarki GO. Dengan menggunakan informasi ini, model dapat mengoptimalkan prediksi hierarkisnya dengan lebih baik, meningkatkan akurasi prediksi untuk kelas-kelas yang terkait secara struktural dalam GO.

Penambahan *dropout layers* dapat membantu mengurangi risiko *overfitting* dalam model, yang penting mengingat kompleksitas dan keragaman dalam data urutan protein. Dengan menerapkan *dropout*, model LSTM dapat belajar pola-pola yang lebih umum dari data latih tanpa terjebak dalam detail-detail yang spesifik yang mungkin hanya berlaku untuk sampel-sampel tertentu. Secara keseluruhan, eksperimen ini diharapkan dapat menghasilkan model prediksi fungsi protein yang lebih akurat, dengan kemampuan untuk menginterpretasikan dan memanfaatkan informasi dari urutan protein secara lebih efektif melalui penggunaan *embedding* *layer* dan teknik regularisasi *dropout*. Dengan demikian, eksperimen ini tidak hanya bertujuan untuk meningkatkan akurasi prediksi fungsi protein, tetapi juga untuk memperdalam pemahaman kita tentang bagaimana model dapat belajar dan menerapkan informasi dari data urutan protein dengan cara yang lebih kontekstual dan informatif.

3.11.2 Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model* yaitu *T5 Embeds*) dan Model LSTM

Pada eksperimen ini, digunakan *Protein Language Model* (PLM) yang terdiri dari T5 *Embedding* dan *Long Short-Term Memory* (LSTM) untuk melakukan prediksi fungsi protein berdasarkan sekuens asam amino. T5-*Embedding* berfungsi sebagai *transformer* teks yang mengubah sekuens asam amino menjadi representasi vektor. Model T5-*Embedding* telah dilatih pada *dataset* besar, memungkinkan model untuk menangkap makna, struktur urutan asam amino, serta hubungan dan pola struktural yang relevan di antara asam amino. Tujuan dari penggunaan T5-*Embedding* dalam eksperimen ini adalah untuk memanfaatkan kemampuan *transformer* dalam menghasilkan representasi numerik dari sekuens asam amino sebelum diolah oleh LSTM. *Embedding* ini harus dilakukan sebelum diolah ke LSTM karena LSTM memerlukan *input* dalam bentuk vektor atau *numerik* untuk dapat memproses data secara efektif.

Selanjutnya, representasi vektor yang dihasilkan oleh T5-*Embedding* diintegrasikan ke dalam model LSTM. LSTM bertanggung jawab untuk mengelola dependensi urutan panjang dalam data dengan memanfaatkan mekanisme gerbang, termasuk *input gate, forget gate,* dan *output gate*. Dengan cara ini, LSTM dapat mempertahankan konteks yang relevan dari sekuens asam amino, yang penting untuk membuat prediksi akurat tentang fungsi protein. Setelah LSTM melakukan pemrosesan terhadap vektor yang diberikan, langkah selanjutnya adalah melakukan evaluasi menggunakan matriks evaluasi. Matriks evaluasi digunakan untuk mengukur kinerja model dalam memprediksi fungsi protein. Matriks evaluasi seperti *binary accuracy, precision, recall,* dan *F1-score* digunakan untuk mengevaluasi seberapa baik model dapat memprediksi fungsi protein berdasarkan sekuens asam amino yang diberikan. Berdasarkan hasil evaluasi, model dapat menghasilkan prediksi fungsi protein yang akurat. Pendekatan ini memanfaatkan kekuatan PLM, khususnya T5-*Embedding* dan LSTM, untuk meningkatkan akurasi dan interpretasi dalam prediksi fungsi protein. Eksperimen ini dilakukan dengan beberapa tujuan utama. Pertama, untuk mengembangkan *model deep learning* yang mampu memprediksi fungsi protein berdasarkan data sekuens protein. Kedua, untuk mengukur dan membandingkan performa model *deep learning* dalam memprediksi fungsi protein melalui metrik seperti akurasi, *precision, recall,* dan *F1-score.* Terakhir, untuk memahami proses implementasi dalam menganotasi fungsi protein terhadap sekuens asam amino yang terdapat pada CAFA5.

Dengan kata lain, eksperimen ini bertujuan untuk mengeksplorasi dan memanfaatkan potensi dari PLM, khususnya T5-*Embedding*s dan LSTM, dalam meningkatkan keakuratan dan efektivitas prediksi fungsi protein berdasarkan sekuens asam amino. Selain itu, eksperimen ini juga bertujuan untuk memberikan pemahaman yang lebih mendalam tentang proses menganotasi fungsi protein dari sekuens asam amino yang terdapat pada CAFA5.

3.11.3 Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model* yaitu *ProtBERT*) dan Model LSTM

Pada eksperimen ini menggabungkan Protein Language Model (PLM) khusus untuk protein, yaitu ProtBERT, dengan model Long Short-Term Memory (LSTM) untuk melakukan prediksi fungsi protein berdasarkan sekuens asam amino. Dimana, Protein merupakan molekul kompleks yang memainkan peran sentral dalam berbagai proses biologis, sehingga prediksi fungsi protein dari urutan asam amino menjadi esensial dalam bioinformatika dan biologi komputasional. Pada eksperimen ini, dilakukan pendekatan yang memanfaatkan *transformer*s teks yaitu, *embedding* ProtBERT yang bertujuan untuk menangkap representasi sekuens protein secara kontekstual, memungkinkan pemahaman yang lebih dalam terhadap struktur dan fungsi molekuler. Sementara itu, untuk model LSTM dipilih karena kemampuannya dalam mengelola dependensi jangka panjang dalam data sekuensial, yang penting dalam analisis protein untuk memahami konteks sekuensial yang relevan dalam menentukan fungsi protein.

Arsitektur model diperkuat dengan lapisan *dense* untuk transformasi output LSTM ke format prediksi kelas, serta lapisan *dropout* untuk mengurangi risiko overfitting dengan menonaktifkan unit secara acak selama pelatihan. Proses eksperimen melibatkan langkah-langkah penting seperti pengolahan data sekuens protein dan anotasi GO\_Term, transformasi sekuens protein menjadi representasi *embedding* yang dapat digunakan untuk pelatihan model, serta evaluasi menggunakan matriks *binary accuracy, AUC, precision, recall,* dan *F-1 Score* untuk mengukur kinerja prediksi model.

**3.11.4 Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model yaitu ProtBERT*) dan Model LSTM + *Diamond Search***

Pada eksperimen ini, akan membahas eksperimen yang dilakukan untuk prediksi fungsi protein menggunakan kombinasi *Protein Language Model* (PLM) ProtBERT, model *Long Short-Term Memory* (LSTM), dan metode pencarian *diamond search*. Adapun tujuan dari eksperimen ini adalah untuk mengeksplorasi bagaimana kombinasi dari teknologi mutakhir ini dapat meningkatkan akurasi dan efektivitas dalam prediksi fungsi protein berdasarkan sekuens asam amino.

ProtBERT adalah model *transformer* yang telah dilatih secara ekstensif pada *dataset* sekuens protein. Kemampuan ProtBERT dalam menangkap representasi kontekstual dari sekuens protein memungkinkan pemahaman yang lebih dalam terhadap struktur dan fungsi molekuler. Dengan menggunakan *embedding* ProtBERT, kita dapat mengonversi sekuens asam amino menjadi vektor representasi yang kaya akan informasi, yang kemudian dapat diolah oleh model *deep learning* seperti LSTM, CNN, RNN, dan lain-lainnya. Model LSTM dipilih karena kemampuannya dalam menangani data sekuensial dan mempertahankan dependensi jangka panjang dalam data. Dalam konteks prediksi fungsi protein, memahami konteks sekuensial adalah kunci untuk mengidentifikasi fungsi protein yang akurat. Model LSTM menerima representasi *embedding* dari ProtBERT dan memprosesnya untuk menghasilkan prediksi fungsi protein. Kombinasi ini memungkinkan pemanfaatan kekuatan dari kedua model untuk mencapai hasil yang lebih baik.

*Diamond search* adalah alat pencarian berkecepatan tinggi untuk membandingkan sekuens protein terhadap *database* besar dari sekuens protein lainnya. Metode ini sangat efisien dan akurat dalam menentukan sekuens protein yang homolog, yang dapat memberikan petunjuk penting tentang fungsi protein target. Dengan mengintegrasikan *diamond search* ke dalam *pipeline*, kita dapat meningkatkan kemampuan model untuk memanfaatkan informasi homolog dari *database* sekuens protein yang luas, yang pada gilirannya dapat meningkatkan akurasi prediksi fungsi protein. Eksperimen ini melibatkan beberapa langkah penting yaitu *preprocessing* data sekuens protein dan anotasi GO\_Term untuk menghasilkan data yang siap digunakan oleh model, *embedding* dengan ProtBERT untuk mengubah sekuens protein menjadi representasi vektor, pemodelan dengan LSTM untuk memproses dan memahami konteks sekuensial, pencarian dengan diamond search untuk menemukan sekuens protein homolog, dan evaluasi model menggunakan metrik *binary accuracy*, AUC, *precision*, *recall*, dan *F1-score* untuk mengukur akurasi dan efektivitas prediksi fungsi protein.

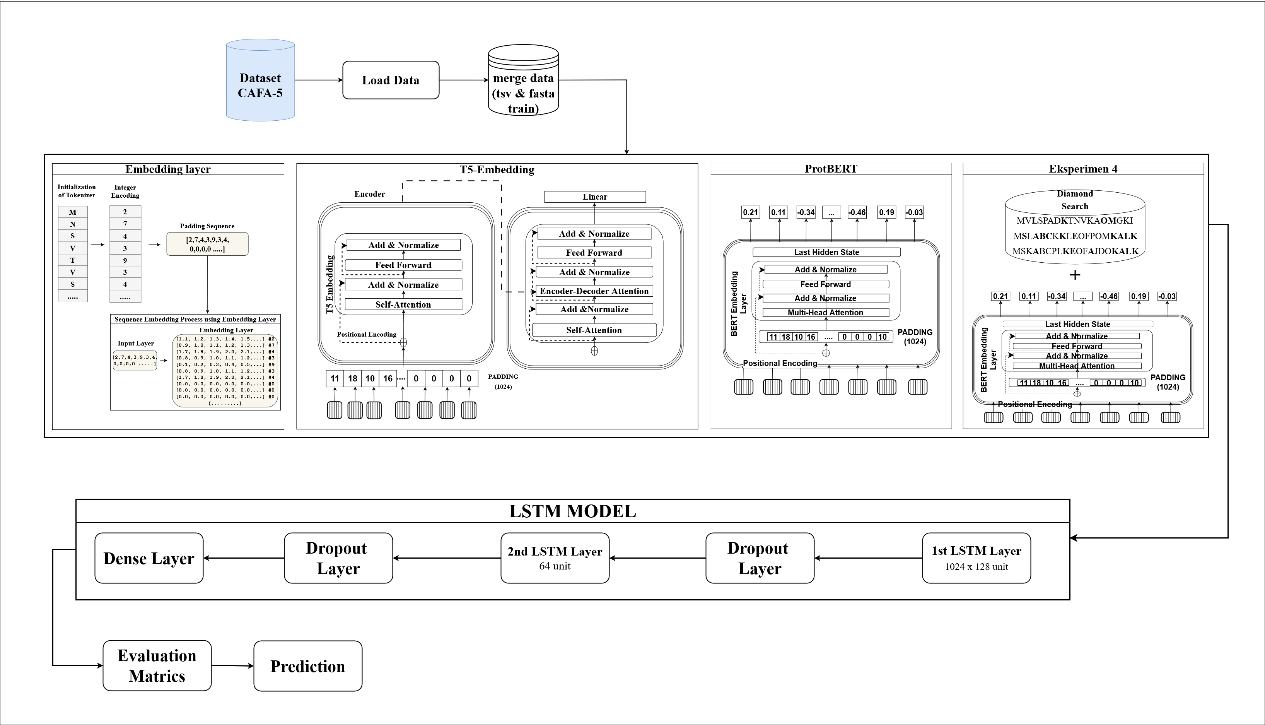
Eksperimen ini bertujuan untuk mengembangkan model prediksi fungsi protein yang lebih akurat dan efisien dengan memanfaatkan *embedding* PLM (ProtBERT), LSTM, dan *diamond search*. Dengan menggabungkan pendekatan ini, diharapkan model dapat lebih baik dalam memahami konteks sekuens protein dan memanfaatkan informasi homolog untuk meningkatkan akurasi prediksi. Selain itu, eksperimen ini juga bertujuan untuk memberikan wawasan yang lebih mendalam tentang bagaimana kombinasi dari berbagai pendekatan dapat digunakan secara efektif dalam bioinformatika dan biologi komputasional.

# BAB IV DESAIN

Pada bab ini dijelaskan mengenai desain dari sistem yang diimplementasikan oleh peneliti. Desain yang dimaksud terdiri dari desain tahapan penelitian, desain arsitektur sistem yang dibangun, dan desain eksperimen yang dilakukan pada penelitian ini.

## 4.1 Desain Penelitian

Pada desain penelitian, model *deep learning* yang dibangun akan dapat melakukan penilaian terhadap prediksi fungsi protein sehingga didapat *output* berupa fungsi protein, yang dapat dilihat pada Gambar 4.1.



Gambar 4.1 Desain Penelitian

Berdasarkan Gambar 4.1, akan dijelaskan tahapan-tahapannya secara rinci pada penjelasan di bawah ini.

1. ***CAFA5 Dataset***

*Dataset* yang dipakai dalam penelitian ini adalah *dataset* *CAFA5 Protein Function Prediction* yang diunduh dari situs *Kaggle* dengan jumlah *file* sebanyak 6 *file*. *Dataset* *CAFA5 Protein Function Prediction* ini terdapat 5  atribut.

1. ***Load Data***

Pada tahapan ini, *load* data dilakukan untuk membaca urutan protein dari *file* *.fasta* dan *.tsv* menggunakan *Bio.SeqIO.parse* dan menyimpan urutan tersebut dalam *DataFrame* dengan kolom *Protein\_ID, Sequence, GO\_Term* dan *Ontology*.

1. ***Merge Data***

Pada tahapan ini, *merge* data berguna untuk menggabungkan informasi dari *file* *.fasta* dan *.tsv* menjadi satu set data. Proses ini menyederhanakan pengolahan data dengan mengurangi kebutuhan untuk mengakses beberapa set data secara terpisah sehingga model akan lebih mudah dalam menganalisis hubungan antara *Protein\_ID, Sequence, GO\_Term*, dan *Ontology* nya.

1. ***Tensorflow.Keras***

Pada bagian ini, digunakan *embedding* *layer* untuk melakukan *embedding* pada sekuens protein. Akan tetapi, sebelum masuk ke *embedding layer*, terlebih dahulu sekuens protein akan melalui tahapan tokenisasi. Setelah sekuens tersebut di tokenisasi, sekuens akan melalui tahapan *padding* yang bertujuan untuk menyamakan semua panjang sekuens protein.

1. ***Pre-train T5-Embedding dan ProtBERT***

Menggunakan data *pre-train* *T5-Embedding* dan *ProtBERT* membantu model memahami konteks protein dengan lebih baik. Ini meningkatkan akurasi prediksi dan mengurangi risiko *overfitting*. Dengan memanfaatkan pengetahuan dari *dataset* yang lebih besar, model bisa fokus pada pola-pola penting yang spesifik untuk tugas prediksi fungsi protein. Secara keseluruhan, *pre-trained embedding* membuat representasi data protein lebih berkualitas dan meningkatkan performa model.

1. ***LSTM Model***

Model LSTM (*Long Short-Term Memory*) digunakan untuk memprediksi fungsi protein dari urutan (sekuens) asam amino. Model ini melibatkan lapisan LSTM yang memungkinkan pengelompokan informasi sepanjang urutan yang panjang dan kompleks, serta lapisan *dense* yang bertanggung jawab untuk menghasilkan output akhir.

* ***LSTM Layer (Long Short-Term Memory)***

Mengolah data urutan menggunakan LSTM. LSTM yang memiliki kemampuan untuk memahami pola temporal dalam data urutan dan mempertahankan informasi kontekstual dari urutan tersebut.

* ***Dropout Layer***

Lapisan ini membantu mengurangi risiko *overfitting* dalam model dengan secara acak mengabaikan sebagian *neuron* selama proses pelatihan.

* ***Dense Layer***

Lapisan ini membantu mengurangi risiko *overfitting* dalam model dengan secara acak mengabaikan sebagian *neuron* selama proses pelatihan.

1. ***Evaluation Model***

Proses evaluasi model dilakukan untuk mengukur kinerja model dalam memprediksi fungsi protein. Matrik evaluasi yang umum digunakan termasuk *accuracy*, *precision*, *recall*, dan *F1-score.*

1. ***Prediction Result***

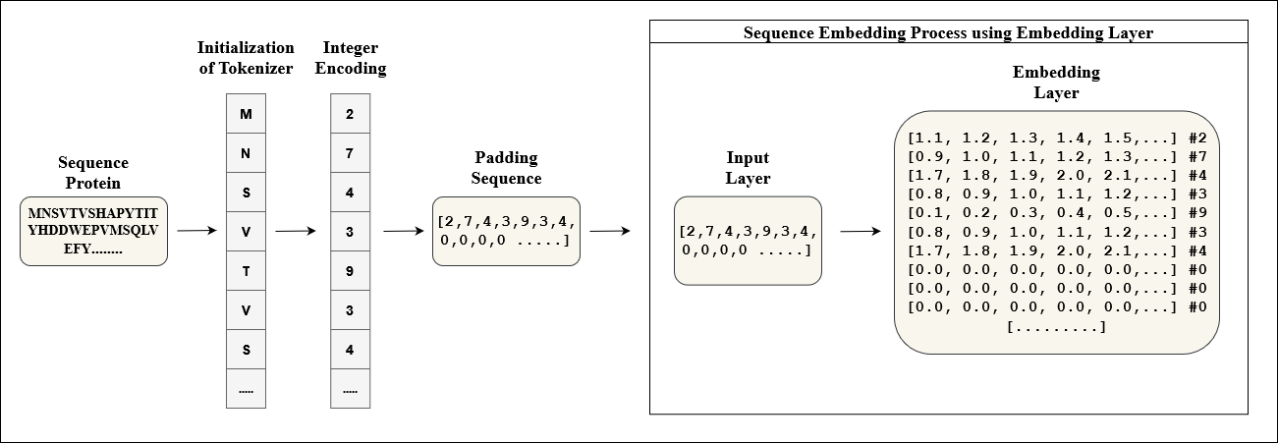
Menyajikan hasil prediksi akhir dari model dalam bentuk GO\_Term dan sub-ontologinya.

## 4.2 Desain *Embedding*

*Data embedding design* adalah proses perancangan yang memiliki tujuan untuk menyematkan atau merepresentasikan data ke dalam dimensi yang lebih rendah atau format yang lebih efisien. Teknik *embedding* sering diterapkan dalam konteks *deep learning*, sebuah metode yang akan kami gunakan untuk memprediksi fungsi protein. Dalam hal ini, teknik *embedding* berperan penting dalam mengubah data berbasis urutan, seperti urutan huruf abjad asam amino pada protein, menjadi representasi numerik yang dapat dimengerti oleh model *deep learning*. Tujuan utama dari *embedding* adalah mempertahankan informasi penting dari data semula sambil mengurangi dimensi atau kompleksitas.

4.2.1 Desain Penggunaan *Layer Embedding* dari *Tensorflow.Keras* dalam Pengolahan Sekuens Protein

Pada bab ini dijelaskan mengenai desain dalam proses penerapan *embedding* pada *embedding layer* dari *Tensorflow.Keras* dalam penelitian ini yang dapat dilihat pada Gambar 4.2.



Gambar 4.2 Desain Layer Embedding dari Tensorflow.Keras dalam Pengolahan Sekuens Protein

Berdasarkan Gambar 4.2, berikut penjelasan tahapan mengenai desain dalam melakukan *embedding* *layer* dari *Tensorflow.Keras*.

1. ***Input*an Sekuens Protein**

Urutan protein yang akan diolah merupakan serangkaian asam amino yang direpresentasikan sebagai *string*, dimana setiap huruf dalam *string* tersebut melambangkan satu asam amino. Misalnya, urutan protein dapat dituliskan sebagai “MNSVTVSHAPYTITYHDDWEPVMSQLVEFY...”. Urutan awal inilah yang akan diolah lebih lanjut agar dapat digunakan untuk melatih model.

1. **Tokenisasi**

Tokenisasi adalah proses memecah urutan protein menjadi unit-unit lebih kecil yang disebut token. Dalam konteks ini, setiap karakter (asam amino) adalah sebuah token. Proses tokenisasi mengidentifikasi setiap asam amino dalam urutan protein dan mempersiapkannya untuk langkah selanjutnya. Sebagai contoh, urutan protein dapat dipecah sebagai berikut: [“M”, “N”, “S”, “V”, “T”, “V”, “S”, ...]. Setiap elemen dalam proses tokenisasi ini adalah token yang mewakili satu asam amino. Pada tahap ini, token-token tersebut masih dalam bentuk karakter.

1. ***Integer Encoding***

Setelah tokenisasi, langkah berikutnya adalah mengonversi token-token tersebut menjadi bilangan bulat melalui proses yang disebut *integer encoding*. *Integer encoding* mengubah setiap token (karakter asam amino) menjadi bilangan bulat unik. Ini diperlukan karena model pembelajaran mesin lebih efektif memproses data dalam format numerik. Sebagai contoh, urutan asam amino akan diberikan indeks numerik sebagai berikut: [M:2, N:7, S:4, V:3, T:9, ...], dimana dengan *integer encoding*, urutan tersebut akan menjadi: [2, 7, 4, ...].

1. ***Padding***

*Padding* adalah sebuah proses yang diterapkan pada urutan data, seperti urutan protein dalam konteks ini, untuk memastikan bahwa setiap urutan memiliki panjang yang seragam. Hal ini dilakukan dengan menambahkan token khusus yang tidak memiliki makna, umumnya berupa token dengan nilai nol, ke urutan yang lebih pendek. Tujuannya adalah untuk memungkinkan penggunaan *batch processing* pada model *deep learning*, di mana setiap *batch* atau kelompok data yang diproses harus memiliki ukuran yang konsisten. Misalnya, jika panjang maksimum yang diizinkan untuk urutan protein adalah 100, dan sebuah urutan tertentu hanya memiliki panjang 80, maka akan ditambahkan 20 token *padding* dengan nilai nol di bagian akhir urutan tersebut.

1. ***Input Layer***

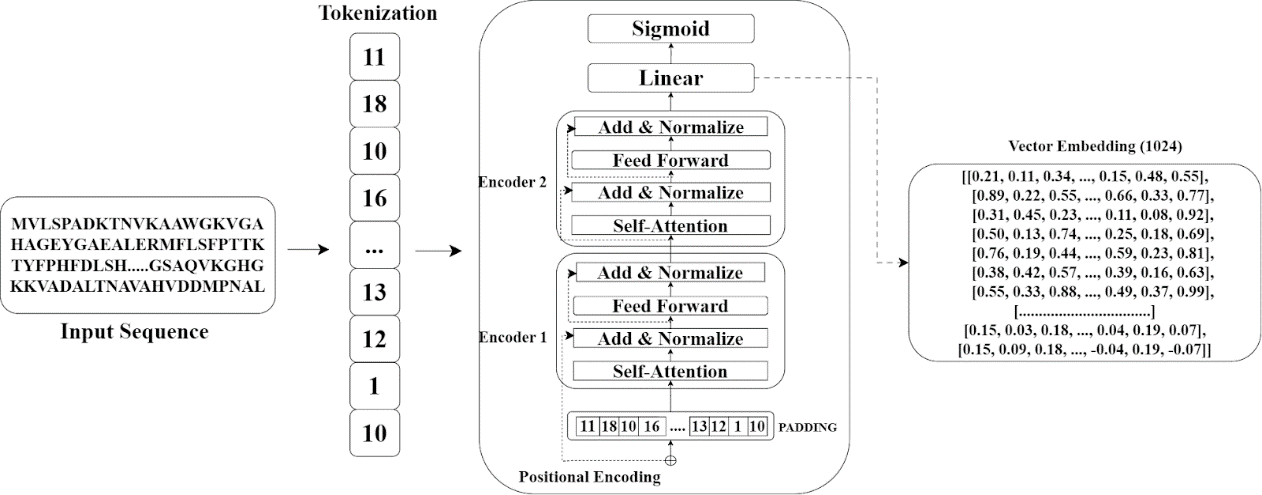
*Input layer* berperan sebagai titik awal dimana urutan protein, yang awalnya direpresentasikan sebagai *string* dari asam amino, diubah menjadi representasi numerik melalui proses tokenisasi, *integer encoding*, dan *padding* seperti yang telah dijelaskan sebelumnya. *Input layer* menerima urutan bilangan bulat yang telah diproses ini dan meneruskannya ke *layer embedding*. Proses ini penting karena memungkinkan model untuk memahami struktur dan konteks dari urutan protein yang akan dipelajari. Dengan mempersiapkan data dalam format yang sesuai, model *deep learning* menggunakan *input layer* sebagai titik masuk untuk memproses informasi sebelum memasuki tahap representasi vektor dalam *embedding layer*.

1. ***Embedding Layer***

Setelah urutan protein diubah menjadi urutan bilangan bulat dan dilakukan *padding*, langkah berikutnya adalah menggunakan *layer embedding*. *Layer embedding* mengonversi urutan bilangan bulat ini menjadi vektor densitas rendah yang memiliki representasi numerik dari setiap asam amino. *Layer embedding* belajar dari data pelatihan untuk menemukan representasi numerik yang optimal sehingga model dapat memahami hubungan kontekstual antara asam amino dalam urutan protein.

4.2.2 Desain *Protein Language Model* (PLM) pada *T5-Embedding*

Pada bab ini dijelaskan mengenai desain dalam proses penerapan *T5-Embedding* dalam penelitian ini yang dapat dilihat pada Gambar 4.3.



Gambar 4.3 Desain Embedding T5-Embedding

Berdasarkan Gambar 4.3 tersebut dijelaskan tiap tahapannya dibawah ini.

1. ***Input Sequence* dan Tokenisasi**

Tahap Awal (Tokenisasi), dimana urutan *input* (misalnya, urutan asam amino pada protein) diubah menjadi bentuk token. Tokenisasi dalam konteks ini masih dibutuhkan untuk mengkonversi karakter dari urutan *input* menjadi token yang dapat diproses oleh model. Misalnya, urutan asam amino "MKT" akan diubah menjadi token berdasarkan posisi urutan asam amino menjadi [11, 9, 17], namun untuk dapat menentukan tokenisasi  pada urutan sekuens tergantung pada kamus token yang akan digunakan.

1. ***Positional Encoding***

Setelah tokenisasi, setiap token diberi *encoding* posisi untuk mempertahankan informasi urutan asli. *Positional encoding* menambahkan informasi posisi token dalam urutan *input* yang penting bagi model untuk memahami konteks sekuensial.

1. **Proses Model T5-*Embedding***

Arsitektur T5 memiliki beberapa lapisan *encoder*. Setiap lapisan *encoder* terdiri dari beberapa komponen utama:

* *Self-Attention*: *Self-attention* memungkinkan model untuk mempertimbangkan setiap token dalam urutan dengan token lainnya, memberikan perhatian pada token yang relevan untuk setiap token dalam urutan *input*.
* *Add & Normalize*: Setelah *self-attention*, hasilnya dijumlahkan dengan *input* awal (*residual connection*) dan dinormalisasi.
* *Feed Forward*: Lapisan *feed-forward* menerapkan transformasi linear pada hasil normalisasi.
* *Add & Normalize*: Hasil dari *feed-forward* dijumlahkan lagi dengan *input* awal *feed-forward* dan dinormalisasi kembali.

Berdasarkan tahapan T5, *Encoder* 1 dan *Encoder* 2 adalah contoh dari dua lapisan *encoder* dalam model T5. Setiap *encoder* mengandung komponen *self-attention*, *add & normalize*, dan *feed-forward* seperti yang dijelaskan sebelumnya.

1. ***Padding***

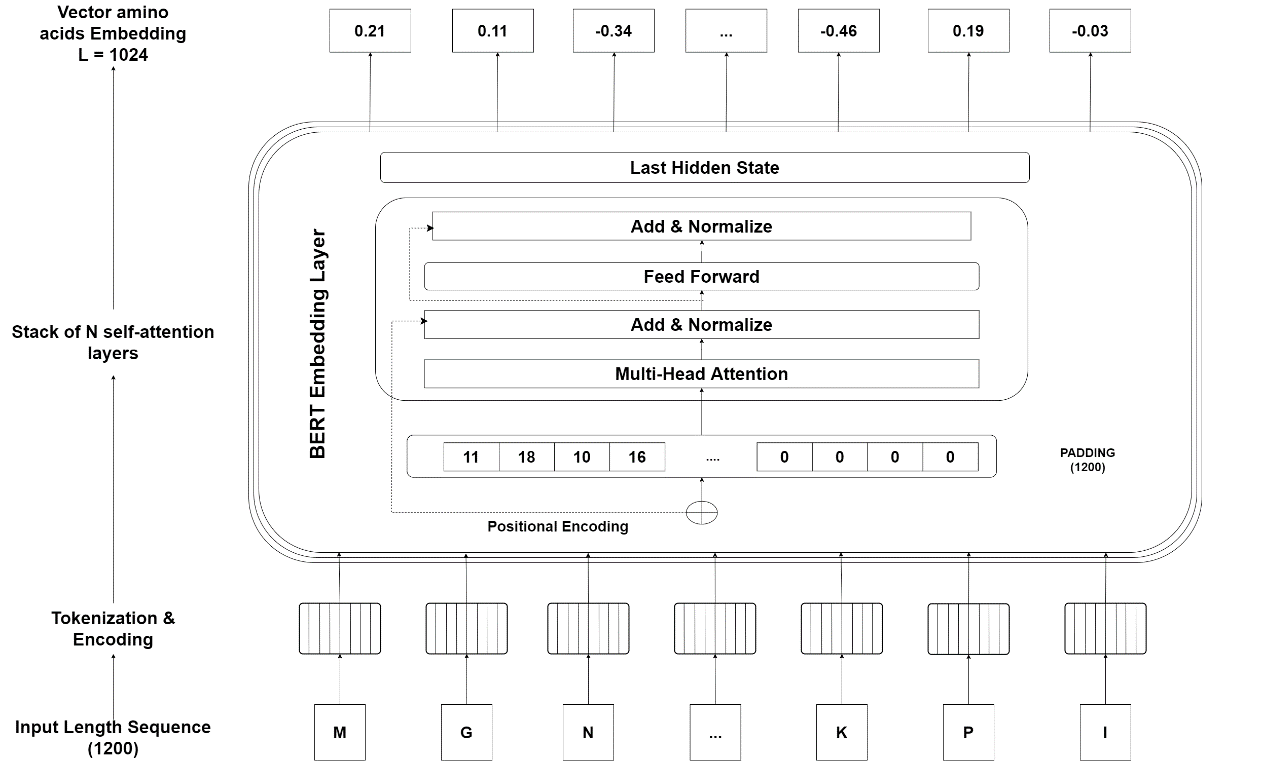
*Padding* diperlukan untuk memastikan bahwa semua urutan *input* memiliki panjang yang sama, terutama saat memproses *batch* data. *Padding* ditambahkan pada urutan *input* sebelum masuk ke model. *Padding* biasanya terdiri dari token khusus (misalnya, token 'PAD') yang diabaikan oleh model selama proses *self-attention* dan *feed-forward*. *Padding* juga biasanya ditambahkan pada akhir urutan *input* untuk membuat panjang semua urutan sama. Misalnya, jika urutan asli adalah [11, 9, 17] dan panjang maksimum yang diinginkan adalah 5, maka urutan di-*padding* menjadi [11, 9, 17, 0, 0] (dengan 0 sebagai token *padding*).

1. ***Linear* dan *Sigmoid***

Setelah melewati beberapa lapisan *encoder*, hasilnya diteruskan ke lapisan linear yang mengaplikasikan transformasi linear pada vektor keluaran dari *encoder* yang dibentuk menjadi matriks n, d, dimana n merupakan panjang sekuens keseluruhan dan d merupakan dimensi panjang sekuens. Sedangkan *sigmoid* adalah hasil dari lapisan linear kemudian diteruskan ke fungsi aktivasi *sigmoid* yang mengubah nilai ke rentang antara 0 dan 1, yang bisa diinterpretasikan sebagai probabilitas.

4.2.3 Desain *Protein Language Model* (PLM) pada *ProtBERT*

Pada bab ini dijelaskan mengenai desain dalam proses penerapan *embedding* dari *Protein Language Model* menggunakan metode BERT atau yang disebut *ProtBERT* dalam penelitian ini. Pada penerapan *embedding* *ProtBERT* dilakukan beberapa langkah untuk memastikan bahwa representasi yang dihasilkan benar-benar mencerminkan fitur-fitur penting dari urutan protein. Langkah-langkah ini meliputi pengumpulan data, *preprocessing*, ekstraksi *embedding*, dan penggunaan *embedding* dalam tugas lanjutan. Dimana, setiap urutan protein yang telah diproses dimasukkan ke dalam model *ProtBERT*, yang menghasilkan *embedding* atau vektor fitur yang mencerminkan informasi penting dari urutan protein dan pada penelitian ini label GO\_Term\_ID dilakukan berdasarkan istilah GO yang paling sering muncul. Berikut desain *embedding* *ProtBERT* yang dapat dilihat pada Gambar 4.4.



Gambar 4.4 Desain Embedding ProtBERT

Berdasarkan Gambar 4.4, berikut penjelasan tahapan mengenai desain dalam melakukan *embedding* *ProtBERT*.

1. ***Input* Sekuens dan Tokenisasi**

Pada tahap pertama, sekuens protein seperti "MKGNKPI" dijadikan *input* untuk proses tokenisasi. Proses ini mengubah setiap asam amino dalam sekuens menjadi token unik, misalnya "M" untuk *metionin*, "K" untuk *lisin*, dan seterusnya, sehingga sekuens "MKGNKPI" diubah menjadi urutan token ["M", "K", "G", "N", "K", "P", "I"]. Tokenisasi ini merupakan langkah kunci untuk mengurai sekuens protein menjadi unit-unit terkecil yang dapat diproses oleh model.

1. ***Padding***

Setelah tokenisasi, langkah berikutnya adalah *padding*. *Padding* dilakukan untuk memastikan bahwa semua urutan *input* memiliki panjang yang konsisten, yang penting untuk konsistensi dalam pengolahan data oleh model. Dalam kasus ini, jika panjang asli sekuens adalah 7 token ("MKGNKPI"), *padding* akan menambahkan token [PAD], yang biasanya merupakan token kosong, di akhir sekuens sehingga panjang total menjadi 1200 token. Hal ini memungkinkan model untuk mengelola berbagai panjang sekuens tanpa mempengaruhi hasil akhir secara negatif.

1. ***Positional Encoding***

Setelah tokenisasi dan *padding* selesai, token-token tersebut diberi *positional* *encoding*. *Positional* *encoding* memberikan informasi mengenai posisi setiap token dalam urutan, yang membantu model memahami hubungan sekuensial antar token. Sebagai contoh, token "M" berada pada posisi 1, "K" pada posisi 2, dan seterusnya hingga token [PAD] pada posisi 8 hingga 1200.

1. **Proses *Embedding* ProtBERT**

Selanjutnya, urutan token yang telah di-tokenisasi, di-*padding*, dan diberi *positional* *encoding* dimasukkan ke dalam lapisan *embedding* BERT. Di dalam lapisan ini, token-token tersebut diproses melalui beberapa lapisan *self-attention* yang terdiri dari *multi-head attention*, *feed forward*, dan proses *add & normalize*. Proses *self-attention* memungkinkan model untuk fokus pada berbagai bagian dari urutan secara bersamaan dan menghasilkan representasi yang lebih kaya. Sebagai contoh, token "M" pada posisi 1 mungkin mendapat perhatian lebih dari token "P" pada posisi 6 dalam urutan yang sama, tergantung pada konteksnya.

1. ***Last Hidden State***

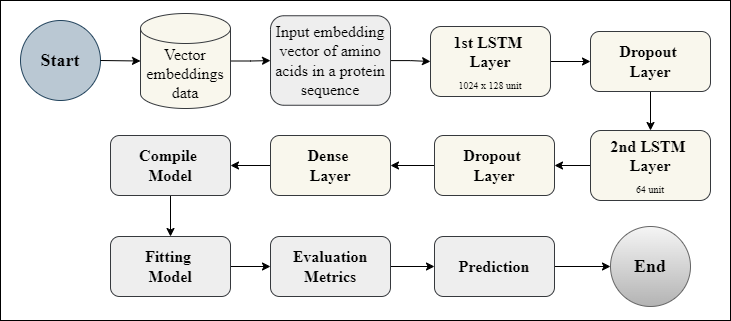
Hasil akhir dari proses *embedding* adalah *last hidden state*, yang merupakan representasi vektor dari setiap token dalam urutan. Setiap vektor memiliki dimensi 1024, misalnya, yang mencerminkan kompleksitas dan informasi yang terkandung dalam token tersebut. Contohnya, token "M" mungkin direpresentasikan oleh vektor *embedding* [0.21, 0.11, -0.34, ..., 0.19, -0.03], sedangkan token [PAD] akan memiliki representasi vektor yang sesuai dengan nilai-nilai default yang ditetapkan selama proses *embedding*.

1. **Ekstraksi dan Penyimpanan Hasil Vektor *Embedding***

Setelah proses *embedding* selesai, hasil ekstraksi dari urutan *input* disimpan dalam format tensor *PyTorch* (*return\_tensors='pt'*). Tensor ini kemudian diekstraksi dan disimpan dalam *file numpy* untuk keperluan data latih dan uji dalam *pre-trained* model. Sebagai contoh, hasil ekstraksi dari tokenisasi sekuens "MKGNKPI" disimpan sebagai tensor, misalnya tensor([0.21, 0.11, -0.34, ..., 0.19, -0.03]) untuk masing-masing token, dan disimpan dalam file numpy seperti *train.npy* dan *test.npy*. Dengan proses ini, ProtBERT dapat mengubah urutan asam amino menjadi representasi vektor yang kaya informasi, memungkinkan analisis yang lebih mendalam tentang struktur dan fungsi protein.

## 4.3 Desain Model LSTM

Setelah melalui tahapan *embedding*, sekuens protein tersebut masuk ke dalam model LSTM yang dapat dilihat pada Gambar 4.5.



Gambar 4.5 Desain Model LSTM

Berdasarkan Gambar 4.5 tersebut, akan dijelaskan tiap tahapannya dibawah ini.

1. ***Vector embeddings data* (Data vektor *embedding*)**

*Embedding* adalah representasi numerik dari urutan asam amino yang digunakan sebagai *input* ke model. Representasi ini dibuat menggunakan metode *embedding* yang memungkinkan urutan asam amino diubah menjadi format yang dapat diproses oleh model. Pada tahapan ini, data *vector embedding* dari hasil *embedding* urutan asam amino telah dipersiapkan.

1. ***Input embedding vector of amino acids in a protein sequence* (*Input* vektor *embedding* dari urutan asam amino dalam protein)**

Vektor *embedding* dari urutan asam amino ini dimasukkan ke dalam model untuk diproses lebih lanjut, dimana proses ini merupakan langkah pertama dalam memanfaatkan data *embedding* untuk melakukan pelatihan model.

1. **1st LSTM *Layer* (Lapisan LSTM pertama)**

Lapisan LSTM (*Long Short-Term Memory*) pertama dengan 128 unit digunakan untuk memproses urutan *embedding*. *Embedding* vektor yang masuk ke lapisan ini memiliki panjang 1024. LSTM adalah jenis jaringan saraf yang dirancang untuk menangani data urutan dan mampu mempertahankan informasi penting dari langkah-langkah sebelumnya dalam urutan tersebut, sehingga sangat cocok untuk analisis data sekuensial seperti urutan asam amino.

1. ***Dropout* *Layer* (Lapisan *Dropout*)**

Lapisan *dropout* diterapkan setelah lapisan LSTM pertama untuk mencegah *overfitting*. *Dropout* secara acak mengabaikan (*drop*) beberapa unit selama pelatihan untuk membuat model lebih umum dan tidak terlalu *overfitting* pada data pelatihan. *Overfitting* terjadi ketika model terlalu sesuai dengan data pelatihan dan tidak dapat melakukan generalisasi dengan baik pada data baru. Hal ini membuat model tidak terlalu bergantung pada pola-pola tertentu dalam data pelatihan dan membantu model untuk menjadi lebih fleksibel serta dapat bekerja lebih baik pada data baru yang belum pernah dilihat sebelumnya.

1. **2nd LSTM *Layer* (Lapisan LSTM kedua)**

Lapisan LSTM kedua dengan 64 unit digunakan untuk memproses *output* dari lapisan LSTM pertama yang sudah melalui lapisan *dropout*. Ini membantu model untuk menangkap informasi urutan yang lebih dalam dan lebih kompleks, memperkuat pemahaman model tentang data sekuensial.

1. ***Dropout* *Layer* (Lapisan *Dropout*)**

Lapisan *dropout* lain diterapkan setelah lapisan LSTM kedua untuk alasan yang sama seperti sebelumnya, yaitu mencegah *overfitting* dan memastikan model tidak terlalu bergantung pada data pelatihan tertentu.

1. ***Dense* *Layer* (Lapisan *Dense*)**

Lapisan *dense* (juga dikenal sebagai lapisan *fully connected*) digunakan setelah LSTM dan *dropout*. Lapisan ini menggabungkan semua informasi dari lapisan sebelumnya dan mengubahnya menjadi bentuk yang sesuai untuk keluaran akhir. Dalam lapisan *dense*, setiap unit terhubung ke semua unit di lapisan sebelumnya, memungkinkan integrasi dan pemrosesan informasi yang komprehensif.

1. ***Compile* Model (Kompilasi Model)**

Pada tahap ini, model akan dikompilasi, dimana model disiapkan untuk pelatihan dengan menentukan fungsi *loss*, *optimizer*, dan metrik evaluasi. Fungsi *loss* digunakan untuk mengukur seberapa baik model melakukan prediksi, *optimizer* digunakan untuk memperbarui bobot model untuk meminimalkan fungsi *loss*, dan metrik evaluasi digunakan untuk mengukur kinerja model selama pelatihan dan evaluasi.

1. ***Fitting* Model (Pelatihan Model)**

Pada tahapan ini, model dilatih menggunakan data pelatihan. Tahap ini adalah proses di mana model belajar dari data dengan menyesuaikan bobotnya untuk meminimalkan fungsi *loss*. Selama pelatihan, model mengevaluasi dan memperbarui bobotnya berdasarkan umpan balik dari fungsi *loss* dan *optimizer*, dengan tujuan mencapai kinerja yang optimal.

1. ***Evaluation* Metrics (Metode Evaluasi)**

Model dievaluasi menggunakan metrik evaluasi yang dipilih (seperti *binary accuracy*, *precision*, *recall*, dan lain-lain) untuk menilai kinerjanya pada data uji. Evaluasi ini penting untuk memahami seberapa baik model melakukan prediksi dan untuk mengidentifikasi area yang memerlukan perbaikan.

1. ***Prediction* (Prediksi)**

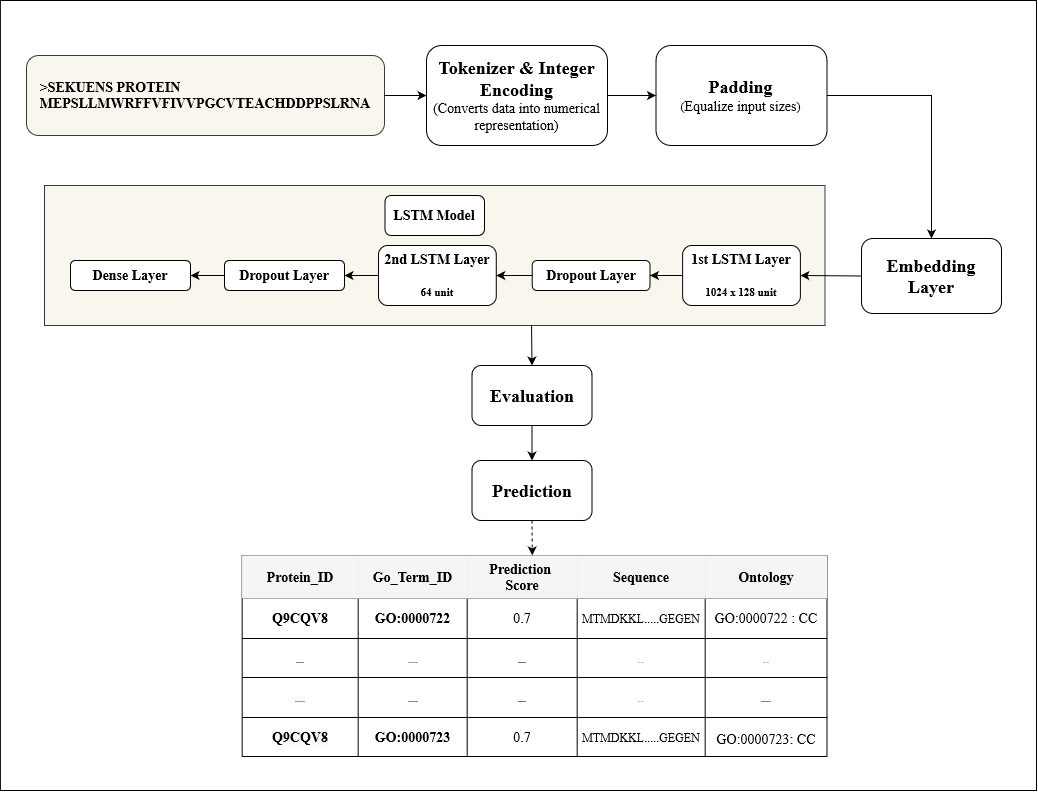
Setelah pelatihan dan evaluasi selesai, model digunakan untuk membuat prediksi pada data baru atau data uji. Pada tahap ini, model diterapkan pada data yang belum pernah dilihat sebelumnya untuk menguji kemampuannya dalam memprediksi fungsi protein dari urutan asam amino.

## 4.4 Desain Eksperimen

Bab ini bertujuan untuk menampilkan desain dari keempat eksperimen yang dilakukan dalam penelitian ini.

4.4.1 Desain Eksperimen Prediksi Fungsi Protein menggunakan *Embedding* dari *TensorFlow.Keras* dan Model LSTM

Pada subbab ini dijelaskan mengenai desain dari eksperimen prediksi fungsi protein dengan menggunakan *Layer Embedding* dari *Tensor flow Keras* yang telah dilakukan pada penelitian ini dapat dilihat pada Gambar 4.6.



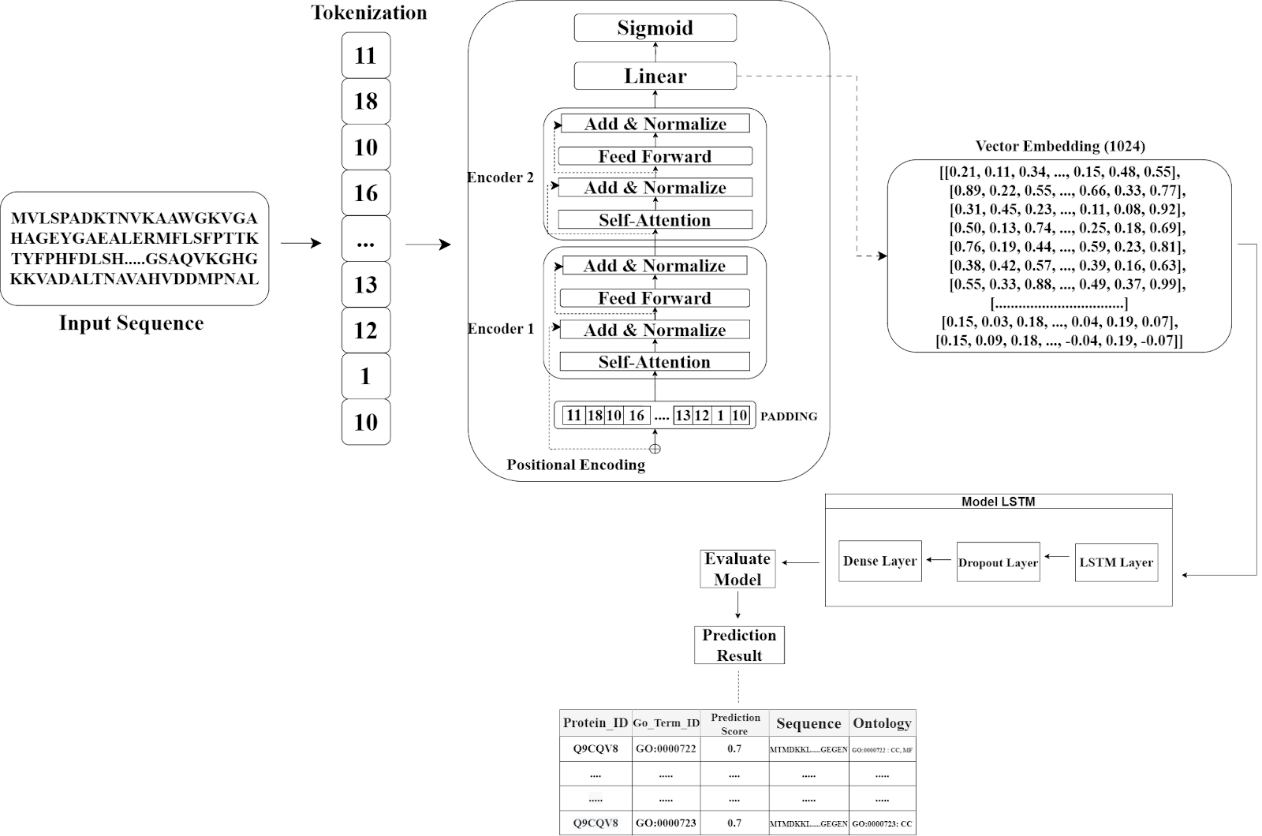
Gambar 4.6 Desain Eksperimen menggunakan Embedding dari Tensorflow.Keras dan LSTM

Berikut adalah penjelasan dari penggunaan *embedding* *layer* dari *TensorFlow Keras*:

1. Data dari CAFA5 *dataset* akan digabungkan antara *file train\_sekuens.fasta* dan *file* *train\_terms.tsv*. Proses penggabungan ini bertujuan untuk menghasilkan satu set data yang mencakup informasi tentang sekuens protein beserta ontologinya.
2. Setelah penggabungan data, langkah selanjutnya adalah melakukan proses *tokenizer* dan *integer encoding*, dimana proses ini mengubah data sekuens protein dari format *string* menjadi representasi numerik. *Tokenizer* akan mengubah setiap asam amino menjadi angka yang sesuai, menghasilkan vektor numerik yang merepresentasikan sekuens protein.
3. Setelah sekuens protein dikonversi menjadi representasi numerik, mereka diperlakukan dengan *padding* untuk menyamakan ukuran *input*. Hal ini diperlukan karena panjang sekuens protein bisa bervariasi, dan *padding* membantu menyamakan panjang sekuens untuk dimasukkan ke dalam model.
4. Selanjutnya masuk ke LSTM model, dimana digunakan 4 *layer*, yaitu *embedding layer* (digunakan untuk memetakan vektor numerik dari sekuens protein ke dalam ruang dimensi yang lebih rendah), LSTM *layer* (digunakan untuk mengingat informasi penting dalam sekuens protein dan mengabaikan informasi yang tidak relevan), *dropout layer* (digunakan untuk mencegah *overfitting* dengan secara acak mengabaikan beberapa *neuron* selama pelatihan), dan *dense layer* (digunakan untuk memproses *output* dari LSTM dan menghasilkan prediksi akhir).
5. Selanjutnya evaluasi model, dimana model yang telah dilatih dievaluasi untuk menilai kinerjanya dalam memprediksi fungsi protein. Evaluasi ini bisa mencakup metrik seperti akurasi, presisi, *recall*, dan *F1-score*.
6. Setelah model dievaluasi, digunakan untuk memprediksi fungsi protein dari sekuens yang belum pernah dilihat sebelumnya.
7. *Output* dari model adalah hasil prediksi ditampilkan dalam bentuk tabel yang berisi ID protein, ID term GO (*Gene Ontology*), nilai prediksi gabungan, sekuens protein, dan ontologi. Contoh *output* menunjukkan prediksi fungsi protein dengan nilai *probability* tertentu.

4.4.2 Desain Eksperimen Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model* yaitu *T5 Embedding*) dan Model LSTM

Pada subbab ini dijelaskan mengenai desain dari eksperimen prediksi fungsi protein dengan menggunakan proses *embedding* *Protein-Language-Model* menggunakan *T5-Embedding* yang telah dilakukan pada penelitian ini dapat dilihat pada Gambar 4.7.



Gambar 4.7 Desain Eksperimen menggunakan T5-Embedding dan LSTM

Berdasarkan Gambar 4.7, berikut merupakan tahapan pengimplementasian *T5-Embedding* dengan algoritma LSTM dapat dijelaskan sebagai berikut:

1. ***Input Sequence* dan Tokenisasi**

Tahap Awal (Tokenisasi), dimana urutan *input* (misalnya, urutan asam amino pada protein) diubah menjadi bentuk token. Tokenisasi dalam konteks ini masih dibutuhkan untuk mengkonversi karakter dari urutan *input* menjadi token yang dapat diproses oleh model. Misalnya, urutan asam amino "MKT" akan diubah menjadi token berdasarkan posisi urutan asam amino menjadi [11, 9, 17], namun untuk dapat menentukan tokenisasi  pada urutan sekuens tergantung pada kamus token yang akan digunakan.

1. ***Positional Encoding***

Setelah tokenisasi, setiap token diberi *encoding* posisi untuk mempertahankan informasi urutan asli. *Positional encoding* menambahkan informasi posisi token dalam urutan *input* yang penting bagi model untuk memahami konteks sekuensial.

1. **Proses Model T5-*Embedding***

Arsitektur T5 memiliki beberapa lapisan *encoder*. Setiap lapisan *encoder* terdiri dari beberapa komponen utama:

* *Self-Attention*: *Self-attention* memungkinkan model untuk mempertimbangkan setiap token dalam urutan dengan token lainnya, memberikan perhatian pada token yang relevan untuk setiap token dalam urutan *input*.
* *Add & Normalize*: Setelah *self-attention*, hasilnya dijumlahkan dengan *input* awal (*residual connection*) dan dinormalisasi.
* *Feed Forward*: Lapisan *feed-forward* menerapkan transformasi linear pada hasil normalisasi.
* *Add & Normalize*: Hasil dari *feed-forward* dijumlahkan lagi dengan *input* awal *feed-forward* dan dinormalisasi kembali.

Berdasarkan tahapan T5, *Encoder* 1 dan *Encoder* 2 adalah contoh dari dua lapisan *encoder* dalam model T5. Setiap *encoder* mengandung komponen *self-attention*, *add & normalize*, dan *feed-forward* seperti yang dijelaskan sebelumnya.

1. ***Padding***

*Padding* diperlukan untuk memastikan bahwa semua urutan *input* memiliki panjang yang sama, terutama saat memproses *batch* data. *Padding* ditambahkan pada urutan *input* sebelum masuk ke model. *Padding* biasanya terdiri dari token khusus (misalnya, token 'PAD') yang diabaikan oleh model selama proses *self-attention* dan *feed-forward*. *Padding* juga biasanya ditambahkan pada akhir urutan *input* untuk membuat panjang semua urutan sama. Misalnya, jika urutan asli adalah [11, 9, 17] dan panjang maksimum yang diinginkan adalah 5, maka urutan di-*padding* menjadi [11, 9, 17, 0, 0] (dengan 0 sebagai token *padding*).

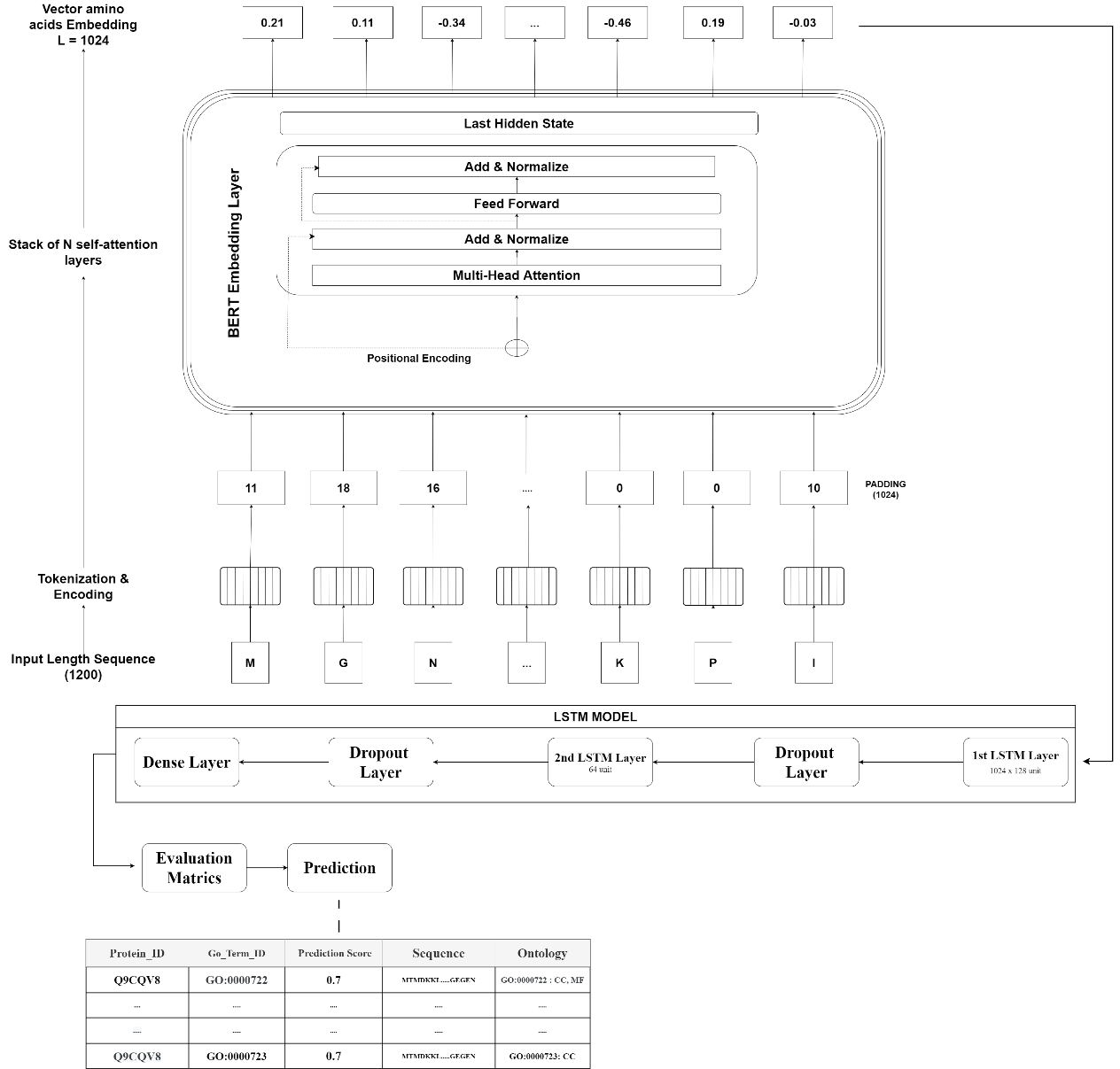
1. ***Linear* dan *Sigmoid***

Setelah melewati beberapa lapisan *encoder*, hasilnya diteruskan ke lapisan linear yang mengaplikasikan transformasi linear pada vektor keluaran dari *encoder* yang dibentuk menjadi matriks n, d, dimana n merupakan panjang sekuens keseluruhan dan d merupakan dimensi panjang sekuens. Sedangkan *sigmoid* adalah hasil dari lapisan linear kemudian diteruskan ke fungsi aktivasi *sigmoid* yang mengubah nilai ke rentang antara 0 dan 1, yang bisa diinterpretasikan sebagai probabilitas.

1. Kemudian vektor *embedding* yang dihasilkan oleh *encoder* digunakan sebagai *input* untuk model LSTM yang terdiri dari beberapa *layer*, yaitu LSTM *layer*, *dropout layer*, dan *dense layer.*
2. Setelah vektor *embedding* diolah pada model LSTM, kinerja model dievaluasi menggunakan metriks *binary accuracy*, *precision*, *recall*, dan *F1-score* untuk memastikan hasil prediksi yang akurat.
3. Tahapan akhir dari eksperimen ini adalah melakukan prediksi yang mencakup Protein\_ID, GO\_Term, probabilitas hasil prediksi, sekuens, dan ontologinya.

4.4.3 Desain Eksperimen Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model* yaitu *ProtBERT*) dan Model LSTM

Pada subbab ini dijelaskan mengenai desain dari eksperimen prediksi fungsi protein dengan menggunakan proses *embedding* *Protein-Language-Model* menggunakan *ProtBERT* yang telah dilakukan pada penelitian ini, yang dapat dilihat pada Gambar 4.8.



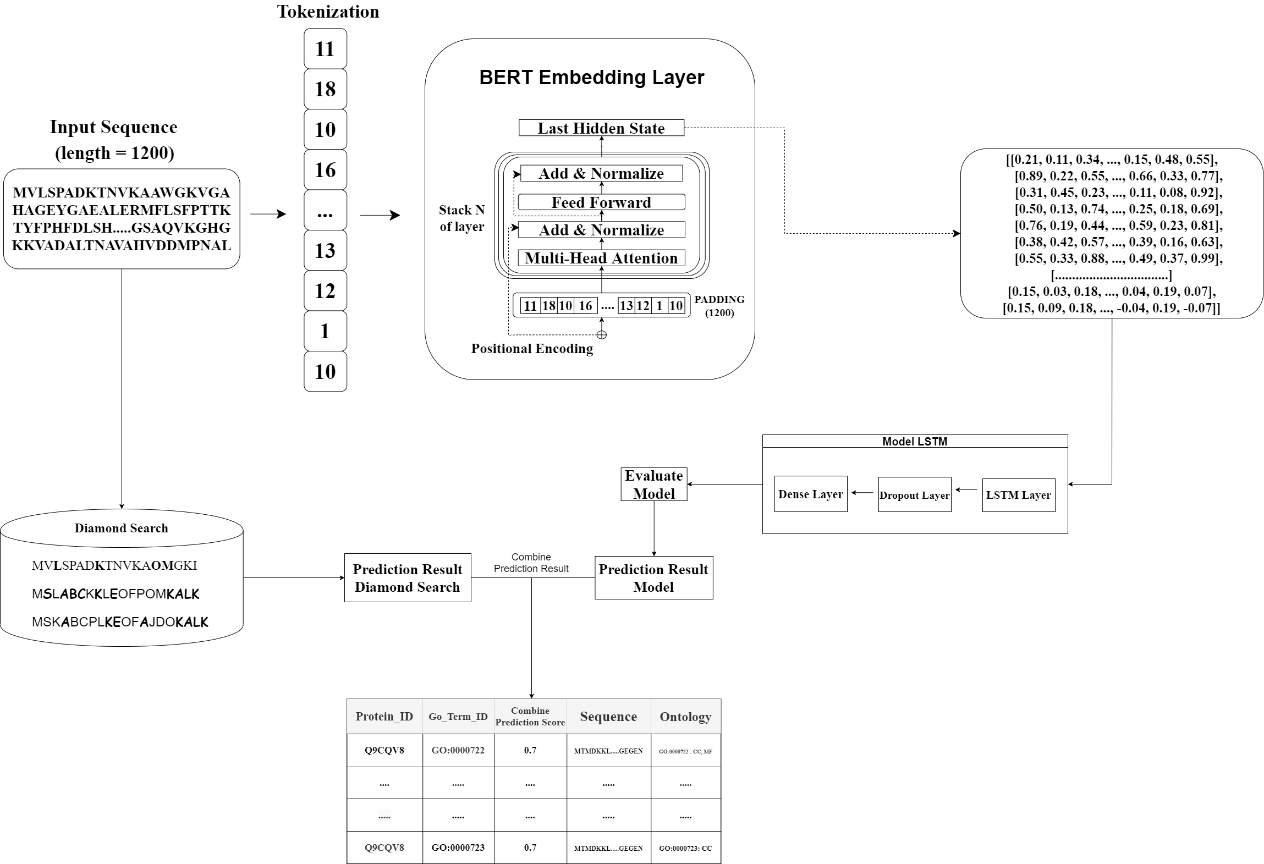
Gambar 4.8 Desain Eksperimen menggunakan ProtBERT dan LSTM

Berdasarkan Gambar 4.8, berikut langkah-langkah pengimplementasian algoritma LSTM dan *ProtBERT* yang dijelaskan sebagai berikut:

1. Pada *dataset* terdapat kolom sekuens protein yang berasal dari *file train\_sequences.fasta*. Sekuens ini nantinya diolah oleh model ProtBERT sebagai *input*. Sekuens protein di-tokenisasi untuk diubah menjadi urutan token yang dapat diproses oleh ProtBERT. Tokenisasi ini dilakukan dengan cara mengkonversi setiap asam amino menjadi representasi numeriknya. Misalnya, huruf 'M' dalam sekuens diubah menjadi angka 11, 'Y' menjadi angka 18, dan seterusnya.
2. Sekuens protein yang telah di-tokenisasi kemudian diolah oleh ProtBERT untuk menghasilkan keluaran berupa vektor *embedding* dengan ukuran 1024 yang merupakan representasi dari asam amino dalam bentuk vektor *embedding*. ProtBERT memproses token-token ini melalui beberapa lapisan transformasi, termasuk *Add & Normalize*, *Feed Forward*, dan *Multi-Head Attention*. Lapisan-lapisan ini menambahkan informasi posisi (*positional encoding*) dan melakukan normalisasi untuk menghasilkan representasi akhir dari asam amino dalam bentuk vektor *embedding*.
3. Setelah sekuens protein diubah menjadi vektor *embedding*, data *embedding* tersebut disimpan dalam *file numpy* untuk digunakan pada tahap pelatihan model LSTM. Penyimpanan ini memastikan bahwa *embedding* dapat diakses dengan mudah selama proses pelatihan dan evaluasi.
4. Data *embedding* yang dihasilkan oleh ProtBERT dilatih menggunakan model LSTM. Model LSTM ini terdiri dari beberapa *layer*, termasuk LSTM *layer*, *dense layer*, *dropout layer*, dan *output layer*. Pelatihan dilakukan dengan menggunakan sekuens protein sebagai *input* dan label fungsi protein sebagai target. LSTM *layer* memproses urutan *embedding* untuk menangkap dependensi jangka panjang dalam data, *dense layer* mengaplikasikan transformasi linear pada *output* dari LSTM untuk mengurangi dimensi, *dropout layer* menambahkan regularisasi untuk mencegah *overfitting*, dan *output layer* menghasilkan prediksi akhir untuk fungsi protein.
5. Setelah model dilatih, kinerjanya dievaluasi menggunakan urutan protein yang belum pernah dilihat sebelumnya (data uji). Evaluasi dilakukan dengan menggunakan metrik kinerja seperti *binary accuracy*, *precision*, *recall*, *F1-Score* dan AUC (*Area Under Curve*) untuk mengukur seberapa baik model dapat memprediksi fungsi protein.

4.4.4 Desain Eksperimen **Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model yaitu ProtBERT*) dan Model LSTM + *Diamond Search***

Pada subbab ini dijelaskan mengenai desain dari eksperimen prediksi fungsi protein dengan menggunakan proses *embedding* P*rotein-Language-Model* menggunakan *ProtBERT* dan model LSTM serta *diamond search* yang dilakukan pada penelitian ini, yang dapat dilihat pada Gambar 4.9.



Gambar 4.9 Desain Eksperimen menggunakan Embedding ProtBERT dan LSTM + Diamond Search

Berdasarkan gambar yang berikan, berikut langkah-langkah pengimplementasian algoritma LSTM dan *ProtBERT* dengan *Diamond Search* dapat dijelaskan sebagai berikut:

1. Desain eksperimen dimulai dengan persiapan sekuens protein sebagai *input* dengan panjang 1200. Sekuens ini, seperti contoh MVLSPADKTNVKAAW..., akan melalui proses tokenisasi menjadi token dengan panjang tertentu, misalnya 11, 18, 10, 16, .... Setelah melalui proses tokenisasi, token-token ini dimasukkan ke dalam lapisan *embedding* ProtBERT. Pada lapisan *embedding* ProtBERT ini, terdapat beberapa komponen utama yang terlibat, yaitu *add & normalize*, *feed forward*, dan *multi-head attention*. Komponen-komponen ini bekerja secara bersama-sama untuk menangkap hubungan antar token dan menghasilkan representasi terakhir dari *hidden state* yang berbentuk vektor.
2. Kemudian setelah itu, vektor hasil *embedding* dari ProtBERT kemudian dimasukkan ke dalam model LSTM.
3. Selanjutnya pada model LSTM ini terdiri dari beberapa lapisan, yaitu *dense layer* yang berfungsi untuk mengurangi dimensionalitas, *dropout layer* yang berfungsi untuk menghindari *overfitting*, dan LSTM *layer* yang berfungsi untuk mempelajari urutan dan konteks dari token-token tersebut.
4. Setelah data diolah dan dilatih pada model LSTM, selanjutnya hasil dari model LSTM ini dievaluasi melalui komponen ‘*evaluate model*’ untuk menghasilkan prediksi model.
5. Kemudian selain menggunakan model LSTM, sekuens protein yang telah di-*input* juga diproses melalui algoritma *diamond search* untuk melakukan pencarian cepat terhadap database protein.
6. Selanjutnya, hasil pencarian dari *diamond search* ini memberikan prediksi fungsi protein berdasarkan kecocokan dengan *database*.
7. Berikutnya, hasil prediksi dari model LSTM dan hasil dari *diamond search* kemudian digabungkan untuk menghasilkan prediksi akhir yang lebih akurat dengan memanfaatkan kelebihan dari kedua metode.
8. Setelah model dilatih, hasil akhir prediksi akan mencakup Protein\_ID, GO\_Term ID, *Sequence*, dan *Ontology* yang sesuai dengan *input* sekuens protein. Sebagai contoh, hasil prediksi menunjukkan bahwa untuk Protein\_ID ‘Q9VQJ9’ dengan GO\_Term ID “GO:0007276”, nilai prediksi gabungan adalah 0.7 dengan sekuens protein ‘MVLSPADKTN...’ yang terkait dengan ontologi yang benar.

# BAB V IMPLEMENTASI

Pada bab ini akan membahas mengenai tahapan implementasi untuk menghasilkan rekomendasi berdasarkan hasil analisis dan perancangan desain yang telah dibahas pada bab sebelumnya. Selain itu, pada bab ini juga, akan dibahas mengenai berbagai eksperimen yang telah dilakukan dalam rangka untuk memprediksi fungsi protein.

## 5.1 Lingkungan Implementasi

Lingkungan implementasi mencakup perangkat (*hardware*) dan perangkat lunak (*software*) yang digunakan pada proses implementasi. Spesifikasi perangkat keras dan perangkat lunak yang digunakan pada proses implementasi dijelaskan pada Tabel 5.1 dan Tabel 5.2.

Tabel 5.1 Spesifikasi Perangkat Keras (Hardware)

|  |  |  |
| --- | --- | --- |
| **No.** | **Perangkat Keras** | **Spesifikasi** |
| 1. | *Processor* | Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz |
| 2. | RAM | 8 GB |
| 3. | Laptop | *Asus VivoBook* |

Tabel 5.2 Spesifikasi Perangkat Lunak (Software)

|  |  |  |
| --- | --- | --- |
| **No.** | **Perangkat Lunak** | **Spesifikasi** |
| 1. | Sistem Operasi | *Windows 11* dan *HPC* |
| 2. | Bahasa Pemrograman | *Python 3.9.12* |
| 3. | *Text Editor* | *Visual Studio Code, Google Colab* |

## 5.2 Batasan Implementasi

Adapun batasan dalam melakukan implementasi pada penelitian *Protein Function Prediction using Deep Learning* adalah sebagai berikut:

1. Data CAFA5: *Protein Function Prediction* diperoleh dari *kaggle*, dimana data tersebut memiliki format yang berbeda-beda (.fasta, .tsv, .obo).
2. Data dengan format .fasta yang berisi Protein\_ID beserta sekuens protein dan *file* dengan format .tsv yang berisi Protein\_ID, GO\_Term, dan *Sub Ontology*, akan digabungkan ke dalam satu *frame* data dan disimpan, sehingga akan model akan lebih mudah dalam menganalisis hubungan antara Protein\_ID, sekuens protein, GO\_Term, dan *Sub Ontology*.
3. Label GO\_Term (fungsi protein) yang digunakan hanya sebanyak 1500 label. Pembatasan ini dilakukan untuk menjaga penggunaan memori dan komputasi agar tetap efisien, mengingat kapasitas memori dan kemampuan komputasi yang terbatas.
4. Panjang maksimal sekuens protein yang ditetapkan untuk *padding* adalah 1200. Sekuens protein yang lebih pendek dan lebih panjang akan di*padding* agar sesuai dengan panjang ini. Selain itu, proses *embedding* akan menghasilkan vektor dengan panjang 1024. Dari 142.246 Protein\_ID yang ada, terdapat 131.620 Protein\_ID yang memiliki panjang sekuens kurang lebih sama dengan 1200. Pembatasan ini membantu dalam standarisasi panjang sekuens, sehingga memudahkan pemrosesan data oleh model dan mengoptimalkan performa komputasi.

## 5.3 Implementasi *Preprocessing*

Proses pra-pemrosesan data adalah langkah pertama yang dilakukan pada set data sebelum mengimplementasikan algoritma. Tujuannya adalah memastikan bahwa data yang digunakan berkualitas tinggi, relevan, dan sesuai dengan kebutuhan analisis atau pemodelan yang dilakukan. Salah satu tahapan yang umum dilakukan dalam pra-pemrosesan data adalah memeriksa keberadaan nilai yang hilang.

5.3.1. Memeriksa *Missing Value*

Memeriksa nilai yang hilang dalam *dataset* bertujuan untuk mengevaluasi sejauh mana data yang tersedia lengkap dan *valid*, serta membantu dalam pengambilan keputusan tentang bagaimana mengelola nilai-nilai yang hilang. Hal ini dapat meliputi penghapusan baris atau kolom yang mengandung nilai hilang, atau penerapan metode pengisian nilai yang sesuai. Berikut adalah kode dan *output* untuk implementasi memeriksa nilai hilang untuk data *train* yang terdiri dari *train\_sekuens.fasta*, *train\_terms.tsv*, dan *train\_taxonomy.tsv*.

# Melihat jumlah *missing value*

fasta\_data.isnull().sum()

terms\_data.isnull().sum()

taxonomy\_data.isnull().sum()

fasta\_target\_path.isnull().sum()

output:

trains\_sequences.fasta  
Protein\_ID      0

Sequence        0

Sequence\_len    0

dtype: int64

trains\_terms.tsv

Protein\_ID    0

GO\_Term\_ID    0

Ontology      0

dtype: int64

taxonomy\_data

Protein\_ID    0

Taxon\_ID      0

dtype: int64

testsuperset.fasta

Protein\_ID    0

Sequence      0

dtype: int64

Berikut adalah kode dan *output* untuk implementasi memeriksa nilai hilang untuk data yang telah digabungkan yang terdiri dari semua data *train* yang telah digabungkan dan data *train* dan target yang digabungkan.

# Melihat jumlah *missing value*

merged\_data\_train.isnull().sum()

terms\_data.isnull().sum()

output:

merged\_data\_train  
Protein\_ID    0

GO\_Term\_ID    0

Ontology      0

Taxon\_ID      0

Sequence      0

dtype: int64

merged\_data\_traintarget

Protein\_ID    0

GO\_Term\_ID    0

Ontology      0

Taxon\_ID      0

dtype: int64

## 5.4 Implementasi Eksperimen

Pada subbab ini dijelaskan mengenai implementasi dari setiap eksperimen yang dilakukan pada penelitian ini.

5.4.1 Implementasi Prediksi Fungsi Protein menggunakan *Embedding* dari *TensorFlow.Keras* dan Model LSTM

Pada subbab ini dijelaskan mengenai implementasi dalam memprediksi fungsi protein menggunakan model LSTM. Pada implementasi ini, proses *embedding* yang dilakukan untuk sekuens protein adalah dengan menggunakan *layer embedding* dari *Tensorflow*.*Keras*, dimana *layer* ini mengubah token dari sekuens protein yang mewakili tiap asam amino menjadi representasi vektor. Berikut ini adalah langkah-langkah yang dilakukan pada implementasi ini.

(a) Langkah pertama adalah melakukan *import library* yang diperlukan pada implementasi ini.

import ast

import numpy as np

import obonet

import pandas as pd

from sklearn.preprocessing import LabelEncoder

from sklearn.preprocessing import MultiLabelBinarizer

import tensorflow as tf

from Bio import SeqIO

from tensorflow.keras.models import Sequential

from tensorflow.keras.*layer*s import LSTM, Dense, Dropout, *Embedding*, *Input*

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing import sequence

from tensorflow.keras.callbacks import EarlyStopping

from tensorflow.keras.metrics import *Precision*, *Recall*

from tensorflow.keras import backend as K

from tensorflow.keras import Sequential

import matplotlib.pyplot as plt

(b) Langkah kedua adalah membaca data dari *file*, di mana bagian ini digunakan untuk membaca data dari beberapa *file*, yaitu *file* FASTA untuk urutan protein, *file* TSV untuk data GO\_Term, dan *file* TSV untuk data taksonomi.

|  |
| --- |
| fasta\_path = "C:/Users/ruthc/OneDrive/Desktop/SEMESTER 8/TA II/Eksperimen TAII/Train/train\_sequences.fasta"  terms\_path = "C:/Users/ruthc/OneDrive/Desktop/SEMESTER 8/TA II/Eksperimen TAII/Train/train\_terms.tsv"  taxonomy\_path = "C:/Users/ruthc/OneDrive/Desktop/SEMESTER 8/TA II/Eksperimen TAII/Train/train\_taxonomy.tsv"  fasta\_data = pd.DataFrame([(record.id, str(record.seq)) for record in SeqIO.parse(fasta\_path, "fasta")], columns=['Protein\_ID', 'Sequence'])  # Read terms data, skipping the first row  terms\_data = pd.read\_csv(terms\_path, sep='\t', header=None, names=['Protein\_ID', 'GO\_Term\_ID', 'Ontology'], skiprows=1)  # Read taxonomy data, skipping the first row  taxonomy\_data = pd.read\_csv(taxonomy\_path, sep='\t', header=None, names=['Protein\_ID', 'Taxon\_ID'], skiprows=1) |

(c) Langkah ketiga adalah menggabungkan data urutan protein dengan GO\_Term, dimana bagian ini dilakukan dengan menggabungkan data urutan protein dengan data GO\_Term berdasarkan Protein\_ID dan kemudian memilih 1500 label GO\_Term yang paling sering muncul dan memfilter data berdasarkan label tersebut.

|  |
| --- |
| df\_protein = pd.merge(fasta\_data, terms\_data, on='Protein\_ID')  num\_of\_labels = 1500  labels = df\_protein['GO\_Term\_ID'].value\_counts().index[:num\_of\_labels].to\_list()  df\_protein = df\_protein.loc[df\_protein['GO\_Term\_ID'].isin(labels)]  display(df\_protein) |

(d) Langkah keempat adalah mengelompokkan data berdasarkan Protein\_ID dan mengubah format GO\_Term, di mana pada tahap ini data dikelompokkan berdasarkan Protein\_ID, Sequence, dan Ontology, lalu setiap GO\_Term yang berkaitan dengan grup tersebut digabungkan dalam bentuk *list*.

|  |
| --- |
| df\_protein = df\_protein.groupby(['Protein\_ID',                                   'Sequence','Ontology'])['GO\_Term\_ID'].apply(list).reset\_index(name='GO\_Term\_ID') |

(e) Langkah kelima adalah melakukan *encoding* GO\_Term menjadi label numerik, di mana pada bagian ini, fungsi *encoding* digunakan untuk mengubah GO\_Term dari bentuk *string* menjadi label numerik menggunakan LabelEncoder.

|  |
| --- |
| def encoding(df):      data = df['GO\_Term\_ID'].values      # get labels programatically from your data      labels = []      for nested\_list in data:          for label in nested\_list:              if label not in labels:                  labels.append(label)      # get labels programatically from your data      labels = []      for nested\_list in data:          for label in nested\_list:              if label not in labels:                  labels.append(label)      label\_encoder = LabelEncoder()      encoded\_labels = label\_encoder.fit\_transform(labels)      labels\_dict = {}      for i in range(len(labels)):          labels\_dict[labels[i]] = encoded\_labels[i]      encoded\_data = []      for labels\_list in data:          # for each label in a nested list replace it with the encoded value from dict          encoded\_data\_list = [l.replace(l, str(labels\_dict[l])) for l in labels\_list]          encoded\_data.append(encoded\_data\_list)      df['GO\_Term\_ID'] = encoded\_data      return df |

(f) Langkah keenam adalah melakukan binarisasi label, di mana menggunakan ‘*MultiLabelBinarizer*’ untuk mengubah list label numerik menjadi bentuk vektor biner.

|  |
| --- |
| mlb = MultiLabelBinarizer()  df\_protein = df\_protein.join(pd.DataFrame(mlb.fit\_transform(df\_protein.pop('GO\_Term\_ID')), columns=mlb.classes\_, index=df\_protein.index))  display(df\_protein.head()) |

(g) Langkah ketujuh adalah melakukan persiapan data untuk pemodelan, di mana pada tahap ini dilakukan persiapan data urutan protein (X) dan label biner (Y) untuk model. Urutan protein diubah menjadi bentuk numerik menggunakan *Tokenizer* dan kemudian di-*padding* agar memiliki panjang yang sama.

|  |
| --- |
| X = df\_protein['Sequence'].values  Y = df\_protein[df\_protein.columns[~df\_protein.columns.isin(['Protein\_ID', 'Sequence', 'Ontology'])]].values  max\_chars = len(set([\*''.join(X)]))  max\_len = 1024  tok = Tokenizer(num\_words=max\_chars, char\_level=True)  tok.fit\_on\_texts(X)  sequences = tok.texts\_to\_sequences(X)  sequences\_matrix = sequence.pad\_sequences(sequences, maxlen=max\_len) |

(h) Langkah kedelapan adalah membangun model LSTM menggunakan *Sequential* dari *Tensorflow.Keras*. Model ini terdiri dari *layer embedding*, LSTM, dan beberapa *layer dense* dengan *dropout* untuk mengurangi *overfitting*. Model dikompilasi dengan *loss function* *binary\_crossentropy* dan *optimizer* adam.

*embedding*\_vector\_features = 64

model=Sequential()

model.add(*Input*(shape=(max\_len,)))

model.add(*Embedding*(max\_chars, *embedding*\_vector\_features))

model.add(LSTM(units=128, return\_sequences=True))

model.add(Dropout(rate=0.3))

model.add(LSTM(units=64))

model.add(Dropout(rate=0.3))

model.add(Dense(units=64, activation='relu'))

model.add(Dropout(rate=0.3))

model.add(Dense((num\_of\_labels), activation='sigmoid'))

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['binary\_accuracy', tf.keras.metrics.AUC(name='auc'), *precision*, *recall*])

model.summary()

(i) Langkah kesembilan adalah melatih model dengan menggunakan data yang telah dipersiapkan dengan *batch size* 32, selama 50 *epoch*, dan menggunakan ‘*validation\_split*=0.1’, dimana 10% dari data pelatihan akan digunakan untuk validasi, memungkinkan pemantauan kinerja model pada data yang tidak dilihat selama pelatihan, yang berguna untuk mencegah *overfitting*. Hasil pelatihan ini disimpan dalam variabel *history*, yang mencakup informasi mengenai metrik pelatihan dan validasi seperti akurasi dan nilai kerugian pada setiap *epoch*.

history = model.fit(sequences\_matrix, Y, batch\_size=32, epochs=50, validation\_split=0.1)

(j) Langkah kesepuluh adalah membuat visualisasi dari hasil pelatihan model, termasuk *loss*, *binary\_accuracy*, AUC, *precision*, dan *recall*, baik untuk data pelatihan maupun validasi, untuk setiap *epoch*.

# Plot metrics

fig, axes = plt.subplots(nrows=1, ncols=5, figsize=(15, 3))

axes[0].plot(history.history['loss'])

axes[0].plot(history.history['val\_loss'])

axes[0].title.set\_text('model loss')

axes[0].set(xlabel='epoch', ylabel='model loss')

axes[0].legend(['train loss', 'valid loss'], loc='upper right')

axes[1].plot(history.history['accuracy'])

axes[1].plot(history.history['val\_accuracy'])

axes[1].title.set\_text('model accuracy')

axes[1].set(xlabel='epoch', ylabel='model accuracy')

axes[1].legend(['train acc', 'valid acc'], loc='upper right')

axes[2].plot(history.history['auc'])

axes[2].plot(history.history['val\_auc'])

axes[2].title.set\_text('model auc')

axes[2].set(xlabel='epoch', ylabel='model auc')

axes[2].legend(['train auc', 'valid auc'], loc='upper right')

axes[3].plot(history.history['*precision*'])

axes[3].plot(history.history['val\_*precision*'])

axes[3].title.set\_text('model *precision*')

axes[3].set(xlabel='epoch', ylabel='model *precision*')

axes[3].legend(['train prec', 'valid prec'], loc='upper right')

axes[4].plot(history.history['*recall*'])

axes[4].plot(history.history['val\_*recall*'])

axes[4].title.set\_text('model *recall*')

axes[4].set(xlabel='epoch', ylabel='model *recall*')

axes[4].legend(['train rec', 'valid rec'], loc='upper right')

fig.tight\_layout()

5.4.2 Implementasi Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model* yaitu *T5 Embedding*) dan Model LSTM

Pada subbab ini dijelaskan mengenai implementasi dalam memprediksi fungsi protein menggunakan *embedding* dengan pendekatan *Protein-Language-Model* yaitu *T5-Embedding* dengan model LSTM. Pada implementasi ini, proses *embedding* yang dilakukan untuk sekuens protein adalah dengan menggunakan tahapan *embedding* dari *Protein-Language-Model* yaitu *T5-Embedding* dan disimpan dalam bentuk *file* numpy yang digunakan sebagai *file* *Pre-Trainee*, dimana pada proses *embedding* ini mengubah sekuens asam amino representasi vektor. Berikut ini adalah tahapan yang dilakukan pada implementasi ini.

(a) Langkah pertama adalah melakukan *import* *library* yang diperlukan pada implementasi ini.

import numpy as np

import obonet

import pandas as pd

from sklearn.model\_selection import train\_test\_split

import tensorflow as tf

from tensorflow import keras

from Bio import SeqIO

import keras\_tuner as kt

from tensorflow.keras import *layer*s

from tensorflow.keras.utils import to\_categorical

from tqdm import tqdm

import warnings

import keras\_tuner as kt

from keras import backend as K

(b) Langkah kedua adalah melakukan *load* dan *read* data yang digunakan pada implementasi ini, dimana terdapat *file* yang berisi data protein,  termasuk urutan asam amino (dalam format *.fasta*) dan istilah GO (dalam format .*tsv*).

fasta\_path = '/content/drive/MyDrive/Eksperimen\_CAFA5/Train/train\_sequences.fasta'

terms\_path = '/content/drive/MyDrive/Eksperimen\_CAFA5/Train/train\_terms.tsv'

fasta\_data = pd.DataFrame([(record.id, str(record.seq)) for record in SeqIO.parse(fasta\_path, "fasta")], columns=['Protein\_ID', 'Sequence'])

terms\_data = pd.read\_csv(terms\_path, sep='\t', header=None, names=['Protein\_ID', 'GO\_Term\_ID', 'Ontology'])

print(fasta\_data.shape)

print(terms\_data.shape)

(c) Langkah ketiga melakukan *load* data *file* *pre-trained* *ProtBERT* untuk memuat tahapan *T5-Embedding* yang telah dilatih sebelumnya dan akan digunakan dalam pelatihan model prediksi fungsi protein menggunakan LSTM.

train\_protein\_ids = np.load('/content/drive/MyDrive/Eksperimen\_CAFA5/*embedding*-T5(PLM)/train\_ids.npy')

train\_protein\_ids.shape

train\_protein\_ids[:10]

train\_*embedding* = np.load('/content/drive/MyDrive/Eksperimen\_CAFA5/*embedding*-T5(PLM)/train\_embeds.npy')

# Now lets convert *embedding* numpy array(train\_*embedding*) into pandas dataframe.

column\_num = train\_*embedding*.shape[1]

trainT5\_df = pd.DataFrame(train\_*embedding*, columns = ["Column\_" + str(i) for i in range(1, column\_num+1)])

print(trainT5\_df.shape)

trainT5\_df.head()

(d) Langkah keempat adalah menetapkan batas jumlah label dengan menentukan *num\_of\_labels* sebesar 1500. Kemudian, akan mengambil nilai frekuensi kemunculan dari kolom GO\_Term dalam data *merged\_data* secara menurun, dan mengambil 1500 GO\_Term teratas sebagai label.

# Set the limit for label

num\_of\_labels = 1500

# Take value counts in descending order and fetch first 1500 `GO term ID` as labels

labels = merged\_data['GO\_Term\_ID'].value\_counts().index[:num\_of\_labels].tolist()

# Fetch the train\_terms data for the relevant labels only

merged\_data\_updated = merged\_data.loc[merged\_data['GO\_Term\_ID'].isin(labels)]

merged\_data\_updated

(e) Langkah kelima adalah dilakukan tahapan LSTM, dimana pada model ini menggunakan lapisan *input*, *reshaping*, dua lapisan LSTM dengan aktivasi ReLU dan *return* sekuens*s*, serta lapisan *dropout* untuk mencegah *overfitting*. Kemudian, ditambahkan lapisan *dense* dengan aktivasi ReLU dan *dropout*, diakhiri dengan lapisan *dense* dengan aktivasi *sigmoid* untuk menghasilkan prediksi. Model dikompilasi menggunakan *loss function binary cross-entropy*, *optimizer Adam*, dan metrik evaluasi seperti *binary accuracy*, AUC, *precision*, dan *recall*, sebelum diringkas untuk menampilkan struktur dan parameter model.

# Build the model

def build\_model():

model = models.Sequential()

model.add(*layer*s.LSTM(units=128, return\_sequences=True, *input*\_shape=*INPUT*\_SHAPE))

model.add(*layer*s.Dropout(rate=0.3))

model.add(*layer*s.LSTM(units=64))

model.add(*layer*s.Dropout(rate=0.3))

model.add(*layer*s.Dense(units=256, activation='relu'))

model.add(*layer*s.Dropout(rate=0.3))

model.add(*layer*s.Dense(num\_of\_labels, activation='sigmoid'))

model.compile(

optimizer='adam',

loss='binary\_crossentropy',

metrics=['binary\_accuracy', tf.keras.metrics.AUC(name='auc'), *precision*, *recall*]

)

return model

(f) Langkah keenam adalah membagi data menjadi set pelatihan dan validasi dengan menggunakan fungsi *train\_test\_split*, di mana 20% dari data digunakan untuk validasi (*test\_size*=0.2). Data fitur (*trainT5\_df.values*) dan label (*labels\_df.values*) dipecah menjadi X\_train, X\_val, y\_train, dan y\_val. Setelah itu, model diringkas menggunakan *model.summary*() untuk menampilkan arsitekturnya. Selanjutnya, model dilatih (*fit*) dengan data pelatihan selama 50 *epoch* dengan ukuran batch 32, serta divalidasi menggunakan set validasi.

# Split data into training and validation sets

X\_train, X\_val, y\_train, y\_val = train\_test\_split(train\_data\_reshaped, labels\_data\_reshaped, test\_size=0.2)

model = build\_model()

model.summary()

history = model.fit(

    X\_train, y\_train,

    validation\_data=(X\_val, y\_val),

    epochs=50,

    batch\_size=32)

(g) Langkah ketujuh adalah memvisualisasikan riwayat pelatihan model dengan membuat plot untuk `*loss*`, `*binary\_accuracy*`, `AUC`, `*precision*`, dan `*recall*` pada set pelatihan dan validasi, menggunakan *DataFrame `history\_df*`. Dimana, metrik yang dihasilkan pada eksperimen ini, masih kurang bagus. Seperti pada akurasi yang sangat tinggi, namun *precision* dan *recall* yang rendah menunjukkan bahwa model mungkin mengalami *overfitting* atau bias, dimana model dapat memprediksi mayoritas kelas dengan baik tetapi gagal pada kelas minoritas. Hal ini mengindikasikan perlunya penyesuaian lebih lanjut seperti mengubah arsitektur model, mengoptimalkan *hyperparameter*, atau menggunakan teknik regularisasi tambahan untuk meningkatkan keseimbangan performa model pada metrik yang berbeda.

# Visualize training history

history\_df = pd.DataFrame(history.history)

history\_df.loc[:, ['loss', 'val\_loss']].plot(title="Loss")

history\_df.loc[:, ['binary\_accuracy', 'val\_binary\_accuracy']].plot(title="*Binary accuracy*")

history\_df.loc[:, ['auc', 'val\_auc']].plot(title="AUC")

history\_df.loc[:, ['*precision*', 'val\_*precision*']].plot(title="*Precision*")

history\_df.loc[:, ['*recall*', 'val\_*recall*']].plot(title="*Recall*")

5.4.3 Implementasi Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model* yaitu *ProtBERT*) dan Model LSTM

Dalam melakukan implementasi prediksi fungsi protein menggunakan algoritma LSTM dengan pendekatan *Protein-Language.* Dapat dilihat dari sekuens protein yang memiliki rentang yang sangat panjang. Hal tersebut membuat proses *embedding* yang dilakukan untuk sekuens protein adalah dengan menggunakan tahapan tokenisasidan *Encoding* dari *Protein-Language-Model* yaitu, *ProtBERT* dari RostLabyang telah di *pre-train* dan disimpan dalam bentuk *file* *numpy* yang terpisah yaitu, untuk *train* dan *test*, dimana setelah sekuens ditokenisasi maka selanjutnya akan menghasilkan *embedding*. *Embedding* ini adalah representasi numerik dari sekuens protein yang menggambarkan fitur-fitur penting dari setiap residu dalam sekuensnya. Berikut ini adalah langkah-langkah yang dilakukan pada implementasi ini.

(a) Langkah pertama adalah melakukan *import library* yang diperlukan pada implementasi ini.

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

import tensorflow as tf

from Bio import SeqIO

from tensorflow.keras import *layer*s, models

from keras import backend as K

(b) Langkah kedua adalah melakukan *load* dan *read* data yang digunakan pada implementasi ini, dimana terdapat *file* yang berisi data protein, termasuk urutan asam amino (dalam format *.fasta*) dan istilah GO (dalam format .*tsv*).

|  |
| --- |
| fasta\_path = '/content/drive/MyDrive/Eksperimen\_CAFA5/Train/train\_sequences.fasta'  terms\_path = '/content/drive/MyDrive/Eksperimen\_CAFA5/Train/train\_terms.tsv'    fasta\_data = pd.DataFrame([(record.id, str(record.seq)) for record in SeqIO.parse(fasta\_path, "fasta")], columns=['Protein\_ID', 'Sequence'])  terms\_data = pd.read\_csv(terms\_path, sep='\t', header=None, names=['Protein\_ID', 'GO\_Term\_ID', 'Ontology'])    print(fasta\_data.shape)  print(terms\_data.shape) |

(c) Langkah ketiga adalah melakukan *load* data *file* *Pre-trained* *ProtBERT* untuk memuat tahapan *embedding* *ProtBERT* yang telah dilatih sebelumnya dan akan digunakan dalam pelatihan model prediksi fungsi protein menggunakan LSTM.

|  |
| --- |
| train\_protein\_ids = np.load('/content/drive/MyDrive/Eksperimen\_CAFA5/*embedding*-*ProtBERT*2(PLM)/train\_ids.npy')  train\_protein\_ids.shape  protein\_df = pd.DataFrame(train\_protein\_ids, columns=[‘Protein ID’]    column\_num = train\_*embedding*.shape[1]train*ProtBERT*\_df = pd.DataFrame(train\_*embedding*, columns = ["Column\_" + str(i) for i in range(1, column\_num+1)])print(train*ProtBERT*\_df.shape)  train*ProtBERT*\_df |

(d) Langkah keempat adalah melakukan *data preprocessing*, kami menggabungkan *dataframe* yang berasal dari '*terms\_data*' dan '*fasta\_data*' berdasarkan kolom 'Protein\_ID', kemudian menyaring data berdasarkan daftar Protein\_ID yang terdapat dalam '*train\_protein\_ids*'. Selanjutnya, kami menentukan jumlah label GO\_Term sebanyak 1500, yang kemudian digunakan untuk membuat matriks label biner.

|  |
| --- |
| merged\_data = pd.merge(terms\_data, fasta\_data[['Protein\_ID', 'Sequence']], on='Protein\_ID')  merged\_data = merged\_data[merged\_data.Protein\_ID.isin(train\_protein\_ids)]    num\_of\_labels = 1500  labels = merged\_data['GO\_Term\_ID'].value\_counts().index[:num\_of\_labels].tolist()    merged\_data\_updated = merged\_data.loc[merged\_data['GO\_Term\_ID'].isin(labels)]  merged\_data\_updated    train\_size = train\_protein\_ids.shape[0]  train\_labels = np.zeros((train\_size ,num\_of\_labels))  series\_train\_protein\_ids = pd.Series(train\_protein\_ids)    for i in range(10):  n\_train\_terms = merged\_data\_updated[merged\_data\_updated['GO\_Term\_ID'] == labels[i]]  label\_related\_proteins = n\_train\_terms['Protein\_ID'].unique()  train\_labels[:,i] = series\_train\_protein\_ids.isin(label\_related\_proteins).astype(float)  labels\_df = pd.DataFrame(data = train\_labels, columns = labels)  print(labels\_df.shape)    *ProtBERT*\_ids = np.load('/content/drive/MyDrive/TA/*embedding*-*ProtBERT*2(PLM)/train\_ids.npy')*ProtBERT*\_embed = np.load('/content/drive/MyDrive/TA/*embedding*-*ProtBERT*2(PLM)/train\_*embedding*.npy')  train\_df = pd.DataFrame(*ProtBERT*\_embed)  train\_df |

(e) Langkah kelima adalah melibatkan pembangunan model LSTM yang kompleks dengan *embedding layer*, dua *layers* LSTM, *dropout layers*, dan *dense layers*. Model ini dirancang untuk memprediksi probabilitas *output* dari sejumlah besar label yang telah ditentukan sebelumnya. Proses *training* menggunakan data yang telah di-*reshape* sesuai dengan *input* model LSTM, dengan *batch size* 32 dan selama 50 *epoch*. Ini merupakan inti dari proses pembangunan dan pelatihan model LSTM untuk tugas klasifikasi *multi-label*.

|  |
| --- |
| def *precision*(y\_true, y\_pred): y\_pred\_binary = K.cast(K.greater(y\_pred, 0.5), dtype='float32') true\_positives = K.sum(K.cast(K.equal(y\_true \* y\_pred\_binary, 1), dtype='float32')) predicted\_positives = K.sum(y\_pred\_binary) *precision* = true\_positives / (predicted\_positives + K.epsilon()) return *precision*  def *recall*(y\_true, y\_pred): y\_pred\_binary = K.cast(K.greater(y\_pred, 0.5), dtype='float32') true\_positives = K.sum(K.cast(K.equal(y\_true \* y\_pred\_binary, 1), dtype='float32')) actual\_positives = K.sum(y\_true) *recall* = true\_positives / (actual\_positives + K.epsilon()) return *recall*    TIMESTEPS = 16*INPUT*\_SHAPE = (TIMESTEPS, train*ProtBERT*\_df.shape[1] // TIMESTEPS)  def build\_model(): model = models.Sequential() model.add(*layer*s.LSTM(units=128, return\_sequences=True, *input*\_shape=*INPUT*\_SHAPE)) model.add(*layer*s.Dropout(rate=0.3)) model.add(*layer*s.LSTM(units=64)) model.add(*layer*s.Dropout(rate=0.3)) model.add(*layer*s.Dense(units=64, activation='relu')) model.add(*layer*s.Dropout(rate=0.3)) model.add(*layer*s.Dense(num\_of\_labels, activation='sigmoid')) model.compile( optimizer='adam', loss='binary\_crossentropy', metrics=['binary\_accuracy', tf.keras.metrics.AUC(name='auc'), *precision*, *recall*] ) return model  train\_data\_reshaped = train*ProtBERT*\_df.values.reshape((-1, TIMESTEPS, *INPUT*\_SHAPE[1]))labels\_data\_reshaped = labels\_df.values  model = build\_model()  model.summary()  history = model.fit(X\_train,y\_train,validation\_data=(X\_val,y\_val),epochs=50,batch\_size=32) |

(f) Langkah keenam adalah melakukan menyimpan model yang telah dilatih ke dalam *file* .h5 dan menyimpan riwayat pelatihan ke dalam *file* CSV.

|  |
| --- |
| # Muat model dengan custom\_objects model.save('/content/drive/MyDrive/TA/Result/Eksperimen3-2.h5')    # Save training history to CSV  history\_df= pd.DataFrame(history.history)history\_df.to\_csv('/content/drive/MyDrive/TA/Result/training\_eksperimen3-2.csv', index=False) |

(g) Langkah ketujuh adalah melakukan visualisasi metrik performa model seperti *loss*, *binary accuracy*, AUC, *precision*, dan *recall* selama proses pelatihan. Visualisasi ini menggunakan grafik *line* *plot* untuk memperlihatkan perubahan performa model pada set pelatihan dan validasi sepanjang *epoch* pelatihan.

|  |
| --- |
| # Visualisasi metrik loss  plt.figure(figsize=(10, 6))  plt.plot(history.history['loss'], label='Training Loss')  plt.plot(history.history['val\_loss'], label='Validation Loss')  plt.title('Training and Validation Loss')  plt.xlabel('Epochs')  plt.ylabel('Loss')  plt.legend()  plt.show()  # Visualisasi metrik *binary accuracy*  plt.figure(figsize=(10, 6))  plt.plot(history.history['binary\_accuracy'], label='Training *Binary accuracy*')  plt.plot(history.history['val\_binary\_accuracy'], label='Validation *Binary accuracy*')  plt.title('Training and Validation *Binary accuracy*')  plt.xlabel('Epochs')  plt.ylabel('*Binary accuracy*')  plt.legend()  plt.show()  # Visualisasi metrik AUC  plt.figure(figsize=(10, 6))  plt.plot(history.history['auc'], label='Training AUC')  plt.plot(history.history['val\_auc'], label='Validation AUC')  plt.title('Training and Validation AUC')  plt.xlabel('Epochs')  plt.ylabel('AUC')  plt.legend()plt.show()  # Visualisasi metrik *Precision*  plt.figure(figsize=(10, 6))  plt.plot(history.history['*precision*'], label='Training *Precision*')  plt.plot(history.history['val\_*precision*'], label='Validation *Precision*')  plt.title('Training and Validation *Precision*')  plt.xlabel('Epochs')plt.ylabel('*Precision*')  plt.legend()  plt.show()  # Visualisasi metrik *Recall*plt.figure(figsize=(10, 6))  plt.plot(history.history['*recall*'], label='Training *Recall*')  plt.plot(history.history['val\_*recall*'], label='Validation *Recall*')  plt.title('Training and Validation *Recall*')  plt.xlabel('Epochs')plt.ylabel('*Recall*')  plt.legend()  plt.show() |

5.4.4 Implementasi **Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model yaitu ProtBERT*) dan Model LSTM + *Diamond Search***

Dalam melakukan implementasi prediksi fungsi protein menggunakan LSTM dan *ProtBERT* dengan tambahan *diamond search* untuk meningkatkan akurasi prediksi. Langkah-langkahnya meliputi tokenisasi dan *encoding* sekuens protein menggunakan *ProtBERT*, yang telah di-*pre-train* dan disimpan dalam *file numpy* terpisah untuk data *train* dan *test*. Proses tokenisasi menghasilkan *embedding*, yaitu representasi numerik dari sekuens protein yang mencerminkan fitur-fitur penting dari setiap residu dalam sekuens tersebut. *Diamond search* digunakan untuk memperbaiki hasil prediksi dari model LSTM-*ProtBERT* dengan teknik pencarian yang canggih dan mendalam, meningkatkan akurasi dan relevansi prediksi. Sebelum melakukan implementasi pada eksperimen ini, terlebih dahulu dilakukan implemntasi terhadap *diamond search.*

5.4.4.1 Implementasi Diamond Search

Sebelum dilakukan implementasi pada eksperimen prediksi fungsi protein menggunakan algoritma LSTM dan *embedding* data latih *ProtBERT* dengan *diamond search*, terlebih dahulu data diolah dengan menggunakan *diamond search* dengan melibatkan beberapa langkah. Berikut adalah penjelasan detail mengenai langkah-langkah implementasi *diamond search* dilakukan.

(a) Langkah pertama adalah melakukan import *library* yang diperlukan pada implementasi ini.

import os

import sys

from temp*file* import NamedTemporary*File*

from urllib.request import urlopen

from urllib.parse import unquote, urlparse

from urllib.error import HTTPError

from zip*file* import Zip*File*

import tar*file*

import shutil

import time

import numpy as np

import pandas as pd

from subprocess import Popen, PIPE

(b) Langkah kedua adalah mempersiapkan lingkungan dan direktori yang diperlukan untuk proses *diamond search*. Pertama, beberapa konstanta ditetapkan, seperti ukuran *chunk* untuk pengunduhan data (CHUNK\_SIZE) dan jalur direktori *input* serta *working* (KAGGLE\_*INPUT*\_PATH dan KAGGLE\_WORKING\_PATH). Direktori *input* kemudian dibersihkan dengan menghapus isinya dan membuat ulang direktori tersebut dengan izin akses penuh. Selain itu, symlink dibuat untuk menghubungkan direktori *input* dan *working* ke lokasi yang lebih mudah diakses. Hal ini memastikan semua direktori dan *symlink* yang diperlukan sudah siap sebelum melanjutkan ke langkah berikutnya dalam proses implementasi *diamond search****.***

CHUNK\_SIZE = 40960

DATA\_SOURCE\_MAPPINg='cafa-5-protein-function-prediction:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-competitions-data%..'KAGGLE\_*INPUT*\_PATH = '/kaggle/*input*'

KAGGLE\_WORKING\_PATH = '/kaggle/working'

KAGGLE\_SYMLINK = 'kaggle'

!umount /kaggle/*input*/ 2> /dev/null

shutil.rmtree('/kaggle/*input*', ignore\_errors=True)

os.makedirs(KAGGLE\_*INPUT*\_PATH, 0o777, exist\_ok=True)

os.makedirs(KAGGLE\_WORKING\_PATH, 0o777, exist\_ok=True)

try:

    os.symlink(KAGGLE\_*INPUT*\_PATH, os.path.join("..", '*input*'), target\_is\_directory=True)

except *File*ExistsError:

    pass

try:

    os.symlink(KAGGLE\_WORKING\_PATH, os.path.join("..", 'working'), target\_is\_directory=True)

except *File*ExistsError:

    pass

(c) Langkah ketiga adalah mengunduh dan mengekstrak data dari setiap sumber yang terdaftar dalam DATA\_SOURCE\_MAPPING ke direktori KAGGLE\_INPUT\_PATH. Proses ini melibatkan dekode URL unduhan, pembacaan data dalam potongan (chunks), penulisan ke file sementara, dan ekstraksi file berdasarkan formatnya (.zip atau format lain). Pesan kesalahan akan dicetak jika terjadi masalah dalam proses ini, dan setelah selesai, akan ditampilkan pesan 'Data source import complete.'

for data\_source\_mapping in DATA\_SOURCE\_MAPPING.split(','):

    directory, download\_url\_encoded = data\_source\_mapping.split(':')

    download\_url = unquote(download\_url\_encoded)

*file*name = urlparse(download\_url).path

    destination\_path = os.path.join(KAGGLE\_*INPUT*\_PATH, directory)

    try:

        with urlopen(download\_url) as *file*res, NamedTemporary*File*() as t*file*:

            total\_length = *file*res.headers['content-length']

            print(f'Downloading {directory}, {total\_length} bytes compressed')

            dl = 0

            data = *file*res.read(CHUNK\_SIZE)

            while len(data) > 0:

                dl += len(data)

                t*file*.write(data)

                done = int(50 \* dl / int(total\_length))

                sys.stdout.write(f"\r[{'=' \* done}{' ' \* (50-done)}] {dl} bytes downloaded")

                sys.stdout.flush()

                data = *file*res.read(CHUNK\_SIZE)

            if *file*name.endswith('.zip'):

                with Zip*File*(t*file*) as z*file*:

                    z*file*.extractall(destination\_path)

            else:

                with tar*file*.open(t*file*.name) as tar*file*:

                    tar*file*.extractall(destination\_path)

            print(f'\nDownloaded and uncompressed: {directory}')

    except (HTTPError, OSError) as e:

        print(f'Failed to load {download\_url} to path {destination\_path}')

        continue

print('Data source import complete.')

(d) Langkah keempat adalah mengunduh dan mengekstrak file diamond untuk digunakan dalam pencarian sekuens.

!wget http://github.com/bbuchfink/diamond/releases/download/v2.1.6/diamond-linux64.tar.gz

!tar xzf diamond-linux64.tar.gz

!rm diamond-linux64.tar.gz

(e) Langkah kelima adalah membuat basis data (database) menggunakan diamond dari file train\_sequences.fasta yang terletak di folder ‘/Train’, dengan nama basis data train\_db. Proses ini dilakukan menggunakan perintah ‘/diamond makedb’.

db\_name = 'train\_db'

p = Popen(['./diamond', 'makedb', '--in', '/kaggle/*input*/cafa-5-protein-function-prediction/Train/train\_sequences.fasta', '-d', db\_name], stdin=PIPE, stdout=PIPE)

stdout, stderr = p.communicate()

(f) Langkah ketujuh adalah menjalankan pencarian menggunakan diamond pada file testsuperset.fasta di dalam direktori ‘/Test (Targets)’ dengan menggunakan basis data train\_db. Hasil pencarian disimpan dalam file matches.tsv dan waktu eksekusi dicatat serta ditampilkan.

out*file*\_name = 'matches.tsv'

k = 16

time0 = time.time()

p = Popen(['./diamond', 'blastp', '-d', db\_name, '-q', '/kaggle/*input*/cafa-5-protein-function-prediction/Test (Targets)/testsuperset.fasta', '-o', out*file*\_name, '--max-target-seqs', str(k), '--quiet'], stdin=PIPE, stdout=PIPE)

stdout, stderr = p.communicate()

print(f'Execution time: {time.time()-time0}s')

(g) Langkah kedelapan adalah memproses dataframe matches yang berisi hasil pencarian. Data dari matches.tsv dibaca menggunakan Pandas dengan pemisah tab dan tanpa baris judul. Kolom-kolom utama seperti 'qseqid', 'sseqid', 'pident', 'length', 'mismatch', 'gapopen', 'qstart', 'qend', 'sstart', 'send', 'evalue', dan 'bitscore' ditetapkan. Kemudian, kolom 'qseqid' diperbarui dengan membagi nilai yang dianggap sebagai ID. Selanjutnya, informasi GO\_Term dari train\_terms.tsv ditambahkan ke dataframe menggunakan 'sseqid'

matches = pd.read\_csv(out*file*\_name, sep='\t', header=None, names=['qseqid', 'sseqid', 'pident', 'length', 'mismatch', 'gapopen', 'qstart', 'qend', 'sstart', 'send', 'evalue', 'bitscore'])

matches['qseqid'] = matches['qseqid'].apply(lambda x: x.split('\\t')[0])

matches.head(10)

train\_terms = pd.read\_csv('/kaggle/*input*/cafa-5-protein-function-prediction/Train/train\_terms.tsv', sep='\t')

arrayofgos = train\_terms.groupby('EntryID').term.apply(lambda x: np.array(x))

matches['terms'] = arrayofgos[matches.sseqid.values].values

matches['ntargets'] = matches.groupby(['qseqid']).qseqid.count()[matches['qseqid'].values].values

(h) Langkah kesembilan adalah menghasilkan dataframe test\_df dari dataframe matches yang telah diproses sebelumnya. Dalam langkah ini, test\_df dibuat dengan memilih kolom-kolom 'qseqid', 'terms', dan 'ntargets' dari matches, kemudian di-"explode" berdasarkan kolom 'terms' untuk memisahkan nilai-nilai yang ada dalam kolom tersebut. Setelah itu, nilai 'ntargets' dibalikkan (dihitung sebagai 1 dibagi dengan nilai 'ntargets' semula). Selanjutnya, dataframe di-"group by" berdasarkan kolom 'qseqid' dan 'terms', menjumlahkan nilai 'ntargets' untuk setiap kombinasi 'qseqid' dan 'terms', kemudian dibulatkan ke tiga desimal.

test\_df = matches[['qseqid', 'terms', 'ntargets']].explode('terms').reset\_index()

test\_df['ntargets'] = 1 / test\_df['ntargets']

test\_df = test\_df.groupby(['qseqid', 'terms']).sum().round(3).reset\_index()

test\_df.head()

(i) Langkah kesepuluh adalah menyimpan hasil akhir dari analisis dalam dataframe test\_df ke dalam file submission.tsv dalam format TSV (tab-separated values). Ini dilakukan dengan mengekspor kolom-kolom 'qseqid', 'terms', dan 'ntargets' ke dalam file tersebut, di mana setiap baris mewakili satu entri dengan nilai-nilai terpisah oleh tab.

test\_df[['qseqid', 'terms', 'ntargets']].to\_csv("submission.tsv", header=False, index=False, sep="\t")

5.4.4.2 Implementasi **menggunakan PLM (Protein Language Model yaitu ProtBERT) dan Model LSTM + Diamond Search**

Setelah dilakukan implementasi pada diamond search, berikut dilakukan implementasi menggunakan LSTM dan ProtBERT dengan tambahan diamond search.

(a) Langkah pertama adalah melakukan *import library* yang diperlukan pada implementasi ini.

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

import tensorflow as tf

from Bio import SeqIO

from tensorflow.keras import *layer*s, models

from keras import backend as K

(b) Langkah kedua adalah melakukan *load* dan *read* data yang digunakan pada implementasi ini, dimana terdapat *file* yang berisi data protein, termasuk urutan asam amino (dalam format *.fasta*) dan istilah GO (dalam format .*tsv*).

|  |
| --- |
| fasta\_path = '/content/drive/MyDrive/Eksperimen\_CAFA5/Train/train\_sequences.fasta'  terms\_path = '/content/drive/MyDrive/Eksperimen\_CAFA5/Train/train\_terms.tsv'    fasta\_data = pd.DataFrame([(record.id, str(record.seq)) for record in SeqIO.parse(fasta\_path, "fasta")], columns=['Protein\_ID', 'Sequence'])  terms\_data = pd.read\_csv(terms\_path, sep='\t', header=None, names=['Protein\_ID', 'GO\_Term\_ID', 'Ontology'])    print(fasta\_data.shape)  print(terms\_data.shape) |

(c) Langkah ketiga adalah melakukan *load* data *file* *pre-trained* *ProtBERT* untuk memuat tahapan *embedding* *ProtBERT* yang telah dilatih sebelumnya dan akan digunakan dalam pelatihan model prediksi fungsi protein menggunakan LSTM.

|  |
| --- |
| train\_protein\_ids = np.load('/content/drive/MyDrive/Eksperimen\_CAFA5/*embedding*-*ProtBERT*2(PLM)/train\_ids.npy')  train\_protein\_ids.shape  protein\_df = pd.DataFrame(train\_protein\_ids, columns=[‘Protein ID’]    column\_num = train\_*embedding*.shape[1]train*ProtBERT*\_df = pd.DataFrame(train\_*embedding*, columns = ["Column\_" + str(i) for i in range(1, column\_num+1)])print(train*ProtBERT*\_df.shape)  train*ProtBERT*\_df |

(d) Langkah keempat adalah melakukan *data preprocessing*, kami menggabungkan *dataframe* yang berasal dari '*terms\_data*' dan '*fasta\_data*' berdasarkan kolom 'Protein\_ID', kemudian menyaring data berdasarkan daftar Protein\_ID yang terdapat dalam '*train\_protein\_ids*'. Selanjutnya, kami menentukan jumlah label GO\_Term sebanyak 1500, yang kemudian digunakan untuk membuat matriks label biner.

|  |
| --- |
| merged\_data = pd.merge(terms\_data, fasta\_data[['Protein\_ID', 'Sequence']], on='Protein\_ID')  merged\_data = merged\_data[merged\_data.Protein\_ID.isin(train\_protein\_ids)]    num\_of\_labels = 1500  labels = merged\_data['GO\_Term\_ID'].value\_counts().index[:num\_of\_labels].tolist()  merged\_data\_updated = merged\_data.loc[merged\_data['GO\_Term\_ID'].isin(labels)]  merged\_data\_updated    train\_size = train\_protein\_ids.shape[0]  train\_labels = np.zeros((train\_size ,num\_of\_labels))  series\_train\_protein\_ids = pd.Series(train\_protein\_ids)    for i in range(10):  n\_train\_terms = merged\_data\_updated[merged\_data\_updated['GO\_Term\_ID'] == labels[i]]  label\_related\_proteins = n\_train\_terms['Protein\_ID'].unique()  train\_labels[:,i] = series\_train\_protein\_ids.isin(label\_related\_proteins).astype(float)    labels\_df = pd.DataFrame(data = train\_labels, columns = labels)  print(labels\_df.shape)    *ProtBERT*\_ids = np.load('/content/drive/MyDrive/TA/*embedding*-*ProtBERT*2(PLM)/train\_ids.npy')*ProtBERT*\_embed = np.load('/content/drive/MyDrive/TA/*embedding*-*ProtBERT*2(PLM)/train\_*embedding*.npy')  train\_df = pd.DataFrame(*ProtBERT*\_embed)  train\_df |

(e) Langkah kelima adalah melibatkan pembangunan model LSTM yang kompleks dengan *embedding layer*, dua *layers* LSTM, *dropout layers*, dan *dense layers*. Model ini dirancang untuk memprediksi probabilitas *output* dari sejumlah besar label yang telah ditentukan sebelumnya. Proses *training* menggunakan data yang telah di-*reshape* sesuai dengan *input* model LSTM, dengan *batch size* 32 dan selama 50 *epoch*. Ini merupakan inti dari proses pembangunan dan pelatihan model LSTM untuk tugas klasifikasi *multi-label*.

|  |
| --- |
| def *precision*(y\_true, y\_pred): y\_pred\_binary = K.cast(K.greater(y\_pred, 0.5), dtype='float32') true\_positives = K.sum(K.cast(K.equal(y\_true \* y\_pred\_binary, 1), dtype='float32')) predicted\_positives = K.sum(y\_pred\_binary) *precision* = true\_positives / (predicted\_positives + K.epsilon()) return *precision*  def *recall*(y\_true, y\_pred): y\_pred\_binary = K.cast(K.greater(y\_pred, 0.5), dtype='float32') true\_positives = K.sum(K.cast(K.equal(y\_true \* y\_pred\_binary, 1), dtype='float32')) actual\_positives = K.sum(y\_true) *recall* = true\_positives / (actual\_positives + K.epsilon()) return *recall*    TIMESTEPS = 16*INPUT*\_SHAPE = (TIMESTEPS, train*ProtBERT*\_df.shape[1] // TIMESTEPS)  def build\_model(): model = models.Sequential() model.add(*layer*s.LSTM(units=128, return\_sequences=True, *input*\_shape=*INPUT*\_SHAPE)) model.add(*layer*s.Dropout(rate=0.3)) model.add(*layer*s.LSTM(units=64)) model.add(*layer*s.Dropout(rate=0.3)) model.add(*layer*s.Dense(units=64, activation='relu')) model.add(*layer*s.Dropout(rate=0.3)) model.add(*layer*s.Dense(num\_of\_labels, activation='sigmoid')) model.compile( optimizer='adam', loss='binary\_crossentropy', metrics=['binary\_accuracy', tf.keras.metrics.AUC(name='auc'), *precision*, *recall*] ) return model  train\_data\_reshaped = train*ProtBERT*\_df.values.reshape((-1, TIMESTEPS, *INPUT*\_SHAPE[1]))labels\_data\_reshaped = labels\_df.values  model = build\_model()  model.summary()  history= model.fit(X\_train,y\_train,validation\_data=(X\_val,y\_val),epochs=50,batch\_size=32) |

(f) Langkah keenam adalah melakukan menyimpan model yang telah dilatih ke dalam *file* .h5 dan menyimpan riwayat pelatihan ke dalam *file* CSV.

|  |
| --- |
| # Muat model dengan custom\_objects model.save('/content/drive/MyDrive/TA/Result/Eksperimen3-2.h5')    # Save training history to CSV  history\_df= pd.DataFrame(history.history)history\_df.to\_csv('/content/drive/MyDrive/TA/Result/training\_eksperimen3-2.csv', index=False) |

(g) Langkah ketujuh adalah melakukan visualisasi metrik performa model seperti *loss*, *binary accuracy*, AUC, *precision*, dan *recall* selama proses pelatihan. Visualisasi ini menggunakan grafik *line* *plot* untuk memperlihatkan perubahan performa model pada set pelatihan dan validasi sepanjang *epoch* pelatihan.

|  |
| --- |
| # Visualisasi metrik loss  plt.figure(figsize=(10, 6))  plt.plot(history.history['loss'], label='Training Loss')  plt.plot(history.history['val\_loss'], label='Validation Loss')  plt.title('Training and Validation Loss')  plt.xlabel('Epochs')  plt.ylabel('Loss')  plt.legend()  plt.show()  # Visualisasi metrik *binary accuracy*  plt.figure(figsize=(10, 6))  plt.plot(history.history['binary\_accuracy'], label='Training *Binary accuracy*')  plt.plot(history.history['val\_binary\_accuracy'], label='Validation *Binary accuracy*')  plt.title('Training and Validation *Binary accuracy*')  plt.xlabel('Epochs')  plt.ylabel('*Binary accuracy*')  plt.legend()  plt.show()  # Visualisasi metrik AUC  plt.figure(figsize=(10, 6))  plt.plot(history.history['auc'], label='Training AUC')  plt.plot(history.history['val\_auc'], label='Validation AUC')  plt.title('Training and Validation AUC')  plt.xlabel('Epochs')  plt.ylabel('AUC')  plt.legend()plt.show()  # Visualisasi metrik *Precision*  plt.figure(figsize=(10, 6))  plt.plot(history.history['*precision*'], label='Training *Precision*')  plt.plot(history.history['val\_*precision*'], label='Validation *Precision*')  plt.title('Training and Validation *Precision*')  plt.xlabel('Epochs')plt.ylabel('*Precision*')  plt.legend()  plt.show()  # Visualisasi metrik *Recall*plt.figure(figsize=(10, 6))  plt.plot(history.history['*recall*'], label='Training *Recall*')  plt.plot(history.history['val\_*recall*'], label='Validation *Recall*')  plt.title('Training and Validation *Recall*')  plt.xlabel('Epochs')plt.ylabel('*Recall*')  plt.legend()  plt.show() |

(h) Langkah kedelapan adalah melakukan *embedding* data uji dimuat dari *file* *test\_embedding.npy*, dikonversi menjadi *DataFrame Pandas*, dan diubah sesuai format *input* model menggunakan TIMESTEPS dan *INPUT*\_SHAPE. Model kemudian melakukan prediksi, yang disimpan dalam variabel *predictions*. Sebuah *DataFrame* baru disiapkan untuk menyimpan hasil prediksi, dengan Protein\_ID dari *file test\_ids.npy*. Protein\_ID diulang sesuai jumlah prediksi dan kolom *DataFrame* diisi dengan Protein\_ID, GO\_Term, serta hasil prediksi yang didapatkan kemudian hasil prediksi diekspor ke *file* .tsv untuk disimpan.

|  |
| --- |
| # Memuat *embedding* data test  test\_*embedding* = np.load('/content/drive/MyDrive/TA/*embedding*-*ProtBERT*2(PLM)/test\_*embedding*.npy')  test\_df = pd.DataFrame(test\_*embedding*)  test\_data\_reshaped = test\_df.values.reshape((-1, TIMESTEPS, *INPUT*\_SHAPE[1]))  # Lakukan prediksi predictions = model.predict(test\_data\_reshaped)  # Siapkan DataFrame untuk menyimpan hasil prediksidf\_predik3 = pd.DataFrame(columns=['Protein Id', 'GO Term Id', 'Prediction'])test\_protein\_ids = np.load('/content/drive/MyDrive/TA/*embedding*-*ProtBERT*2(PLM)/test\_ids.npy')  l = []  for k in list(test\_protein\_ids):  l += [k] \* predictions.shape[1]  df\_predik3['Protein Id'] = l  df\_predik3['GO Term Id'] = labels \* len(test\_protein\_ids)  df\_predik3['Prediction'] = predictions.ravel()  # Simpan hasil prediksi ke *file*  df\_predik3.to\_csv("/content/drive/MyDrive/TA/Result/PredikEksperimen3-2.tsv", header=False, index=False, sep="\t")  df\_predik3 |

(i) Langkah kesembilan adalah menggabungkan hasil prediksi dari model LSTM+*ProtBERT* dan *diamond search* berdasarkan kolom 'Protein\_ID' dan 'GO\_Term'. Setelah kedua prediksi digabungkan, dilakukan perhitungan untuk mendapatkan prediksi gabungan dengan menghitung rata-rata dari prediksi LSTM+*ProtBERT* dan *diamond search*. Selanjutnya, data hasil prediksi yang telah digabungkan dengan prediksi gabungan tersebut digabungkan kembali dengan data sekuens (*sequence\_data*) dan data ontologi (*terms\_data*) untuk melengkapi informasi prediksi. Data yang relevan dipilih untuk *output* akhir yang kemudian disimpan dalam *file* .tsv dengan nama “*Combine\_Prediction.tsv*”.

|  |
| --- |
| # Load hasil prediksi dari LSTM+*ProtBERT*  df\_lstm\_*ProtBERT* = pd.read\_csv('/content/drive/MyDrive/TA/Result/PredikEksperimen3-2.tsv', sep='\t', header=None, names=['Protein\_ID', 'GO\_Term\_ID', 'Prediction'])  import pandas as pd    # Load hasil prediksi dari Diamond Search  df\_diamond = pd.read\_csv('/content/drive/MyDrive/TA/Result/HasilPrediksi\_Diamond.tsv', sep='\t', header=None, names=['Protein\_ID', 'GO\_Term\_ID', 'Prediction'])    # Filter predictions with probability > 0.50  df\_lstm\_*ProtBERT*\_filtered = df\_lstm\_*ProtBERT*[df\_lstm\_*ProtBERT*['Prediction'] > 0.50]  df\_diamond\_filtered = df\_diamond[df\_diamond['Prediction'] > 0.50]    # Combine predictions based on 'Protein\_ID' and 'GO\_Term\_ID'  df\_combined = pd.merge(df\_lstm\_*ProtBERT*\_filtered, df\_diamond\_filtered, on=['Protein\_ID', 'GO\_Term\_ID'], suffixes=('\_lstm', '\_diamond'))    # Calculate the average prediction for combined prediction  df\_combined['Combined\_Prediction'] = (df\_combined['Prediction\_lstm'] + df\_combined['Prediction\_diamond']) / 2    # Merge with sequence data to include the sequencesdf\_combined\_with\_seq = pd.merge(df\_combined, fasta\_data, on='Protein\_ID')    # Merge with terms data to include the ontologydf\_combined\_with\_seq\_and\_ontology = pd.merge(df\_combined\_with\_seq, terms\_data, on=['Protein\_ID', 'GO\_Term\_ID'])  # Select relevant columns for final outputdf\_final = df\_combined\_with\_seq\_and\_ontology[['Protein\_ID', 'GO\_Term\_ID', 'Combined\_Prediction', 'Sequence', 'Ontology']]    # Save combined predictions to *file*  df\_final.to\_csv("/content/drive/MyDrive/TA/Result/Combined\_Predictions\_With\_Sequence\_And\_OntologyEksperimen4.tsv", header=False, index=False, sep="\t") |

# BAB VI HASIL DAN PEMBAHASAN

Pada bab ini membahas hasil dan analisis dari eksperimen menggunakan model LSTM, yang dirancang dengan LSTM *layer*, *dropout layer*, dan *dense layer*. Tujuan dari bab ini adalah untuk menampilkan hasil dari eksperimen serta melakukan analisis mendalam terhadap hasil-hasil yang diperoleh. *Output* dari model ini berupa prediksi fungsi protein dalam format GO\_Term, seperti contoh GO:0005575, yang merupakan kode spesifik dalam *Gene Ontology* (GO) yang merujuk pada komponen selular. GO\_Term ini selanjutnya dianotasikan ke tiga aspek utama dalam *Gene Ontology*, yaitu *Biological Process* (BP), *Cellular Component* (CC), atau *Molecular Function* (MF). Anotasi ini bertujuan untuk memahami peran dan fungsi spesifik dari protein yang diprediksi oleh model dalam konteks biologis. Informasi yang dihasilkan tidak hanya mencakup GO\_Term (fungsi protein), tetapi juga aspek-aspek utama dari *Gene Ontology* (BP, CC, MF), yang sangat penting untuk interpretasi dan penerapan hasil prediksi dalam studi biologis dan bidang terkait.

## 6.1 Hasil Eksperimen Penelitian

Pada subbab ini dibahas mengenai hasil yang didapat peneliti ketika melakukan eksperimen dengan *embedding* menggunakan *embedding layer* dari *Tensorflow.Keras*, *T5-Embedding*, dan *ProtBERT.* Proses *embedding* tersebut diimplementasikan dengan model LSTM serta *diamond search.* Pada penelitian ini *hyperparameter* yang digunakan dalam eksperimen dapat dilihat pada **Tabel 3.13**.

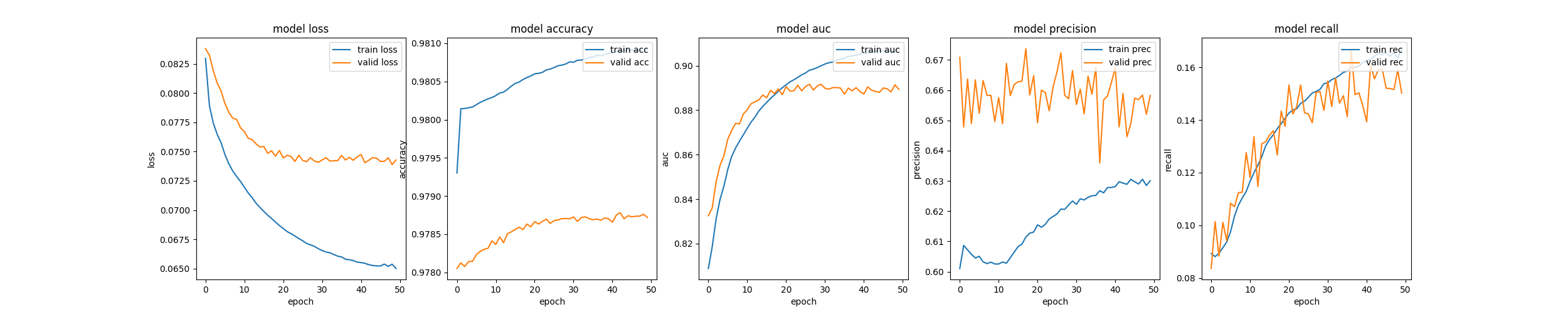
6.1.1 Hasil Prediksi Fungsi Protein menggunakan *Embedding* dari *TensorFlow.Keras* dan Model LSTM

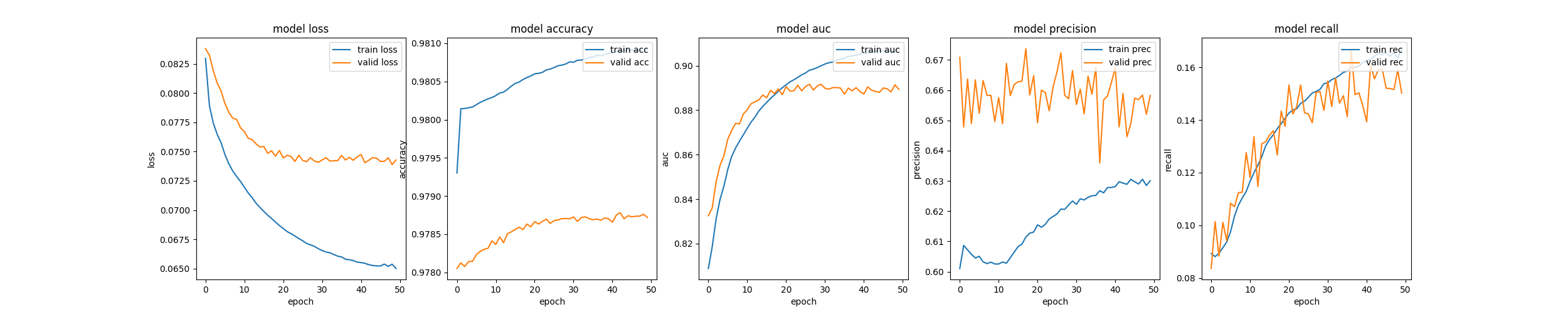
Pada subbab ini dijelaskan mengenai hasil yang didapat dari eksperimen pertama dari penelitian ini, dimana eksperimen ini menerapkan *hyperparameter* yang ada pada Tabel 3.17. Panjang sekuens protein yang digunakan adalah 1024 karakter, yang dicapai dengan *padding* urutan protein agar memiliki panjang yang sama dengan label GO\_Term yang digunakan sebanyak 1500. Setelah menerapkan *hyperparemeter* tersebut untuk melatih model, evaluasi kemudian dilakukan dengan mempertimbangkan metrik seperti *binary accuracy*, AUC, *precision*, *recall*, dan *f1-score* untuk setiap *epoch*. Hasil pelatihan divisualisasikan dalam bentuk grafik untuk menunjukkan perubahan *loss*, *binary accuracy*, AUC, *precision*, dan *recall* selama pelatihan. Berikut adalah evaluasi dari eksperimen ini yang dapat dilihat pada Tabel 6.1.

Tabel 6.1 Hasil Prediksi Eksperimen 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Epoch*** | ***Loss*** | ***Binary\_***  ***Accuracy*** | ***AUC*** | ***Precision*** | ***Recall*** | ***F1-Score*** |
| 1 | 0,08296 | 0,97930 | 0,80884 | 0,60107 | 0,08930 | 0,15550 |
| 2 | 0,07893 | 0,98014 | 0,81869 | 0,60870 | 0,08811 | 0,15394 |
| 3 | 0,07743 | 0,98014 | 0,83122 | 0,60720 | 0,08940 | 0,15586 |
| 4 | 0,07647 | 0,98015 | 0,83987 | 0,60576 | 0,09150 | 0,15899 |
| 5 | 0,07575 | 0,98016 | 0,84586 | 0,60453 | 0,09368 | 0,16222 |
| **...** | **...** | **...** | **...** | **...** | **...** | **...** |
| 46 | 0,06522 | 0,98091 | 0,90658 | 0,62971 | 0,16558 | 0,26222 |
| 47 | 0,06538 | 0,98089 | 0,90607 | 0,62899 | 0,16416 | 0,26037 |
| 48 | 0,06519 | 0,98092 | 0,90675 | 0,63056 | 0,16553 | 0,26223 |
| 49 | 0,06537 | 0,98088 | 0,90616 | 0,62849 | 0,16352 | 0,25952 |
| 50 | 0,06501 | 0,98092 | 0,90764 | 0,63005 | 0,16708 | 0,26412 |

Dari hasil *training* di atas, ditunjukkan bahwa hasil *binary accuracy,* AUC, *precision, recall,* dan *f1-score* secara berurut adalah **0,980**; **0,907; 0,630**; **0,167**; dan **0,264**. Hasil *training* divisualisasikan dalam bentuk grafik untuk menunjukkan perubahan *loss*, *binary accuracy*, AUC, *precision*, dan *recall* selama pelatihan. Berikut ini adalah hasil dari eksperimen ini yang dapat dilihat pada Gambar 6.1.





Gambar 6.1 Hasil Metriks Evaluasi Eksperimen 1

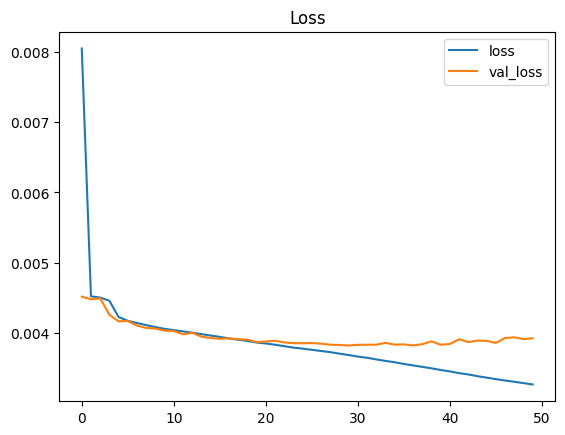
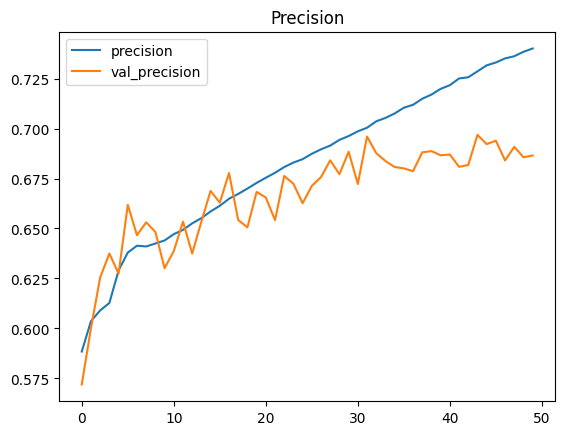
6.1.2 Hasil Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model* yaitu *T5 Embedding*) dan Model LSTM

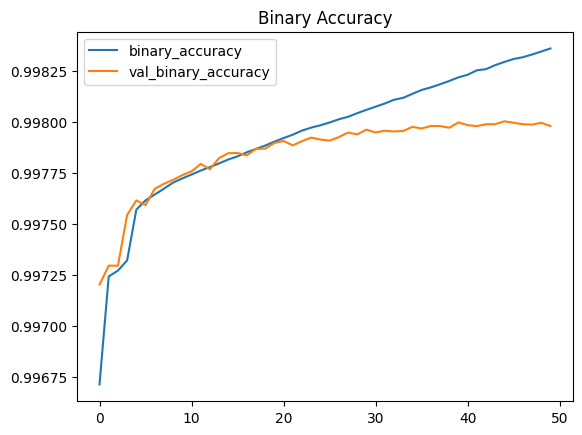
Pada subbab ini dijelaskan mengenai hasil yang didapat dari eksperimen kedua dari penelitian ini, dimana eksperimen ini menerapkan *hyperparameter* yang ada pada Tabel 3.17. Setelah menerapkan *hyperparameter* tersebut untuk melatih model, evaluasi kemudian dilakukan dengan mempertimbangkan metrik seperti *binary accuracy*, AUC, *precision*, *recall*, dan *f1-score* untuk setiap *epoch*. Hasil pelatihan divisualisasikan dalam bentuk grafik untuk menunjukkan perubahan *loss*, *binary accuracy*, AUC, *precision*, dan *recall* selama pelatihan. Berikut adalah evaluasi dari eksperimen ini yang dapat dilihat pada Tabel 6.2.

Tabel 6.2 Hasil Prediksi Eksperimen 2

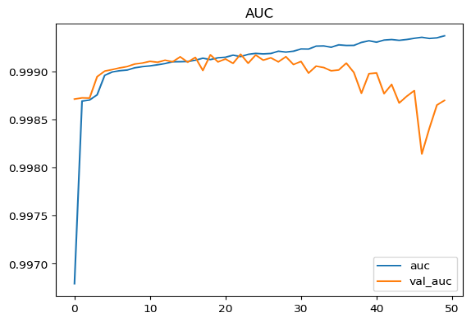
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Epoch*** | ***Loss*** | ***Binary\_***  ***Accuracy*** | ***AUC*** | ***Precision*** | ***Recall*** | ***F1-Score*** |
| 1 | 0.00876 | 0.99643 | 0.99658 | 0.59597 | 0.55178 | 0,57302 |
| 2 | 0.00451 | 0.99725 | 0.99869 | 0.60616 | 0.54580 | 0,57439 |
| 3 | 0.00449 | 0.99727 | 0.99870 | 0.60988 | 0.54086 | 0,57330 |
| 4 | 0.00449 | 0.99728 | 0.99870 | 0.61104 | 0.54027 | 0,57347 |
| 5 | 0.00448 | 0.99728 | 0.99871 | 0.61122 | 0.54047 | 0,57367 |
| **...** | **...** | **...** | **...** | **...** | **...** | **...** |
| 46 | 0.00334 | 0.99828 | 0.99930 | 0.72915 | 0.78416 | 0,75565 |
| 47 | 0.00332 | 0.99830 | 0.99927 | 0.73116 | 0.78551 | 0,75736 |
| 48 | 0.00330 | 0.99831 | 0.99931 | 0.73289 | 0.78769 | 0,75930 |
| 49 | 0.00328 | 0.99832 | 0.99928 | 0.73373 | 0.78958 | 0,76063 |
| 50 | 0.00327 | 0.99833 | 0.99931 | 0.73551 | 0.79108 | 0,76228 |

Pada hasil training di atas, ditunjukkan bahwa hasil *binary accuracy*, AUC, *precision*, *recall*, dan *f1-score* secara berurut adalah **0,998**; **0,999**; **0,735**; **0,791**; dan **0,762**. Hasil *training* divisualisasikan dalam bentuk grafik untuk menunjukkan perubahan *loss*, *binary accuracy*, AUC, *precision*, dan *recall* selama pelatihan. Berikut ini adalah hasil dari eksperimen ini yang dapat dilihat pada Gambar 6.2.









Gambar 6.2 Hasil Metriks Evaluasi Eksperimen 2

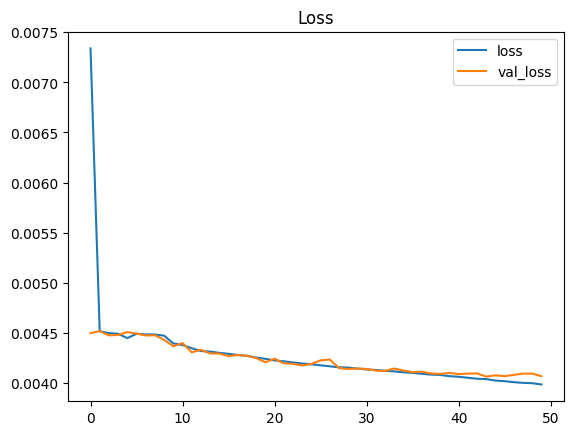
6.1.3 Hasil Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model* yaitu *ProtBERT*) dan Model LSTM

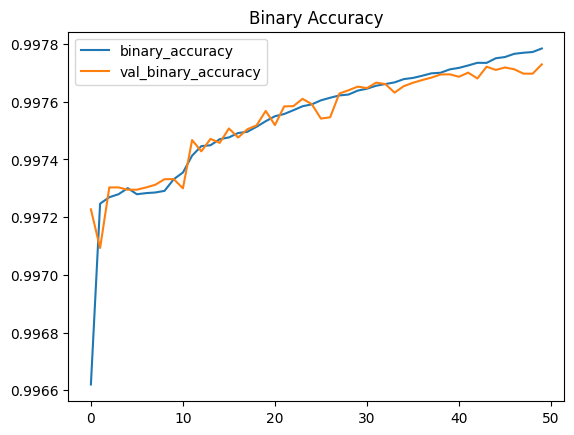
Pada subbab ini dijelaskan mengenai hasil yang didapat dari eksperimen ketiga dari penelitian ini, dimana eksperimen ini menerapkan *hyperparameter* yang ada pada Tabel 3.17. Setelah menerapkan *hyperparemeter* tersebut untuk melatih model, evaluasi kemudian dilakukan dengan mempertimbangkan metrik seperti *binary accuracy*, AUC, *precision*, dan *recall* untuk setiap *epoch*. Hasil pelatihan divisualisasikan dalam bentuk grafik untuk menunjukkan perubahan *loss*, *binary accuracy*, AUC, *precision*, dan *recall* selama pelatihan. Berikut adalah evaluasi dari eksperimen ini yang dapat dilihat pada Tabel 6.3.

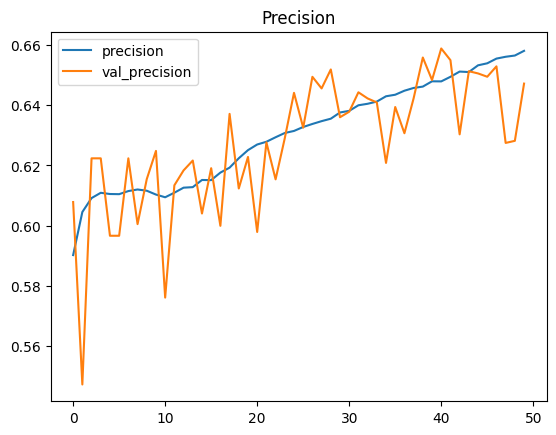
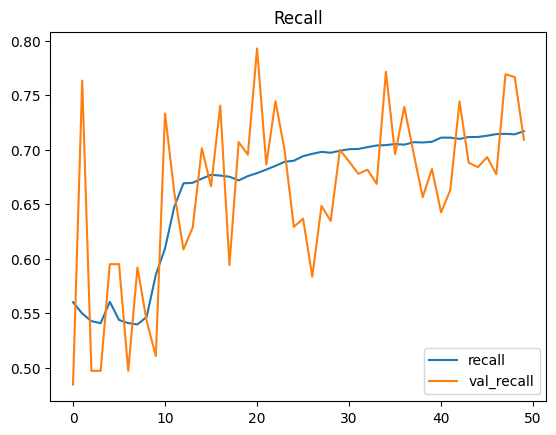
Tabel 6.3 Hasil Prediksi Eksperimen 3

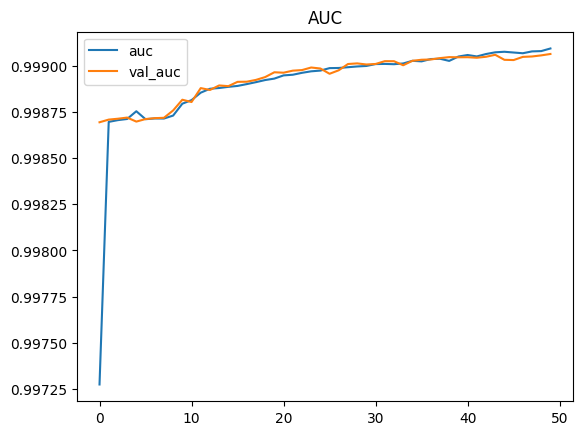
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Epoch*** | ***Loss*** | ***Binary\_***  ***Accuracy*** | ***AUC*** | ***Precision*** | ***Recall*** | ***F1-Score*** |
| 1 | 0,00859 | 0,99626 | 0,99671 | 0,59615 | 0,54977 | 0,57202 |
| 2 | 0,00450 | 0,99726 | 0,99869 | 0,60657 | 0,54293 | 0,57299 |
| 3 | 0,00449 | 0,99727 | 0,99870 | 0,60996 | 0,53887 | 0,57221 |
| 4 | 0,00449 | 0,99728 | 0,99870 | 0,61138 | 0,53805 | 0,57238 |
| 5 | 0,00448 | 0,99728 | 0,99870 | 0,61106 | 0,53992 | 0,57329 |
| **...** | **...** | **...** | **...** | **...** | **...** | **...** |
| 46 | 0,00389 | 0,99784 | 0,99911 | 0,66623 | 0,72489 | 0,69432 |
| 47 | 0,00388 | 0,99786 | 0,99912 | 0,66724 | 0,72855 | 0,69655 |
| 48 | 0,00387 | 0,99786 | 0,99913 | 0,66842 | 0,72828 | 0,69707 |
| 49 | 0,00386 | 0,99787 | 0,99912 | 0,66925 | 0,73140 | 0,69895 |
| 50 | 0,00385 | 0,99788 | 0,99914 | 0,67005 | 0,73348 | 0,70033 |

Dari hasil *training* di atas, ditunjukkan bahwa hasil *binary accuracy* sebesar **0,997***,* AUC sebesar **0,999**, *precision* sebesar **0,670***, recall* sebesar **0,733***, f1-score* sebesar **0,700** dan AUC sebesar **0,999**. Hasil ini cukup dibilang baik dengan nilai dari akurasi *binary accuracy* yang tinggi, *precision* dan *recall* yang moderat, *f-1 score* yang baik, dan AUC yang sangat tinggi menunjukkan bahwa model yang dihasilkan sangat baik dengan menggunakan model LSTM dan *embedding pre-train ProtBERT*. Hasil *training* divisualisasikan dalam bentuk grafik untuk menunjukkan perubahan *loss*, *binary accuracy*, AUC, *precision*, dan *recall* selama pelatihan yang dapat dilihat pada Gambar 6.3.







Gambar 6.3 Hasil Metriks Evaluasi Eksperimen 3

6.1.4 Hasil **Prediksi Fungsi Protein menggunakan PLM (*Protein Language Model* yaitu *ProtBERT*) dan Model LSTM + *Diamond Search***

Dalam eksperimen yang dilakukan merupakan lanjutan dari eksperimen sebelumnya, model *Long Short-Term Memory* (LSTM) menggunakan *embedding* dari *pre-trained* *ProtBERT* untuk memprediksi fungsi protein berdasarkan sekuensnya. Model *LSTM-ProtBERT* ini dilatih dengan data sekuens protein dan anotasi GO\_Term, dimana setelah mendapatkan hasil prediksi, model digabungkan untuk meningkatkan *probability* hasil prediksi. Hasil prediksi dari model ini dikombinasikan dengan metode *diamond search*, yang mencari kesamaan sekuens protein dengan basis data yang ada untuk memberikan skor yang lebih akurat. Prediksi dari kedua metode di *filter* untuk menyertakan hasil dengan *probability* diatas 0,50 dan kemudian digabungkan. Nilai prediksi akhir dihitung dengan menggunakan rumus berikut.

Berikut ini adalah nilai prediksi akhir yang dihitung dengan mengambil *alpha* sebesar 0.50 dari nilai prediksi yang dihasilkan oleh model LSTM-*ProtBERT* dan *diamond search* dengan dua Protein\_ID yang digunakan sebagai perbandingan dari *diamond* yaitu, **‘P35213’** dan **‘P68254’**.

Tabel 6.4 Hasil Prediksi Eksperimen 4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***Protein***  ***LSTM***  ***ProtBERT*** | ***GO\_Term***  ***LSTM-ProtBERT*** | ***Prob. Score\_***  ***LSTM-ProtBERT*** | ***Protein Diamond*** | ***GO\_Term***  ***Diamond*** | ***Prob. Score\_***  ***Diamond*** | ***Combine\_ Prediction*** | ***Ontology*** |
| Q9CQV8 | GO:0005575 | **0,6967** | **P35213** | GO:0005575 | 0,938 | **0,8173** | CCO |
| Q9CQV8 | GO:0005575 | **0,6967** | **P68254** | GO:0005575 | 0,938 | **0,8173** | CCO |

Berdasarkan Tabel 6.4, hasil akhir dari kombinasi ini menunjukkan bahwa penggunaan model gabungan antara LSTM-*ProtBERT* dan metode *diamond search* dapat secara signifikan meningkatkan akurasi prediksi fungsi protein. Model LSTM-*ProtBERT* memprediksi bahwa protein dengan ID ‘Q9CQV8’ memiliki GO\_Term ‘GO:0005575’ dengan *probability score* sebesar 0.6967. Sementara itu, metode *diamond search* mengidentifikasi dua protein (P35213 dan P68254) yang juga memiliki GO\_Term ‘GO:0005575’ dengan *probability score* yang lebih tinggi, yaitu 0.938. Dengan menggabungkan prediksi dari kedua metode menggunakan rumus yang telah disampaikan sebelumnya, hasil akhir menunjukkan *probability score* sebesar 0.8173 untuk GO\_Term ‘GO:0005575’. Prediksi ini dikategorikan sebagai "*cellular component*", yang mengindikasikan relevansi biologis yang tinggi. Dapat dilihat hasil akhir menunjukkan *probability score* sebesar 0.8173 untuk GO\_Term ‘GO:0005575’. Prediksi ini dikategorikan sebagai "*cellular component*", yang mengindikasikan relevansi biologis yang tinggi. Kombinasi kedua metode ini membuktikan bahwa integrasi antara pendekatan *deep learning* dan pencarian berbasis kesamaan sekuens dapat memberikan hasil yang lebih akurat dalam memprediksi fungsi protein. Pendekatan gabungan ini memberikan gambar yang efektif dan komprehensif untuk analisis fungsi protein dalam penelitian biologi molekuler dan bioinformatika.

## 6.2 Pembahasan Eksperimen Penelitian

Pada subbab ini dijelaskan mengenai pembahasan dari hasil eksperimen yang dilakukan pada penelitian ini, dimana eksperimen tersebut dibahas berdasarkan nilai metriks evaluasi dan hasil prediksi dari fungsi protein.

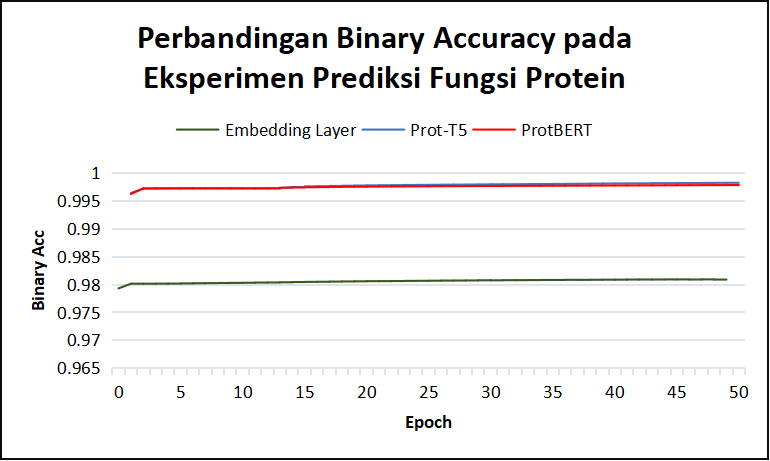
6.2.1 Pembahasan berdasarkan Nilai Metriks Evaluasi

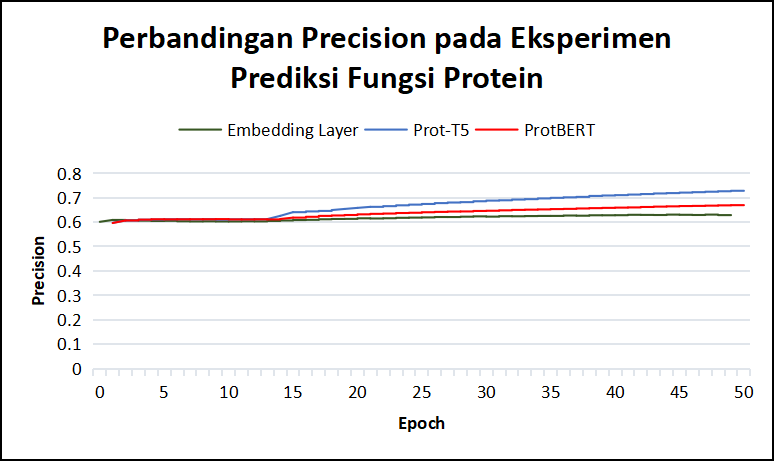
Pada subbab ini dijelaskan mengenai pembahasan terhadap eksperimen yang dilakukan pada penelitian ini berdasarkan nilai metriks evaluasinya, dimana metriks evaluasi yang digunakan adalah *binary accuracy*, *precision*, *recall*, dan *F1-score*. Berikut ini adalah pembahasan berdasarkan nilai metriks evaluasi yang dapat dilihat pada Tabel 6.5.

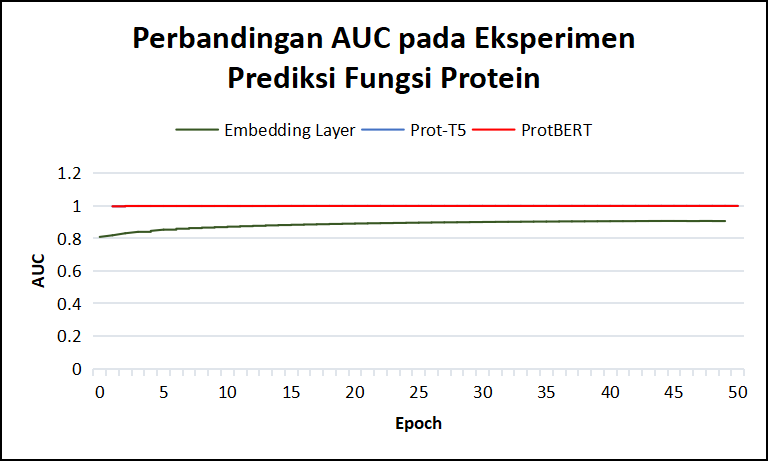
Tabel 6.5 Pembahasan berdasarkan Nilai Metriks Evaluasi

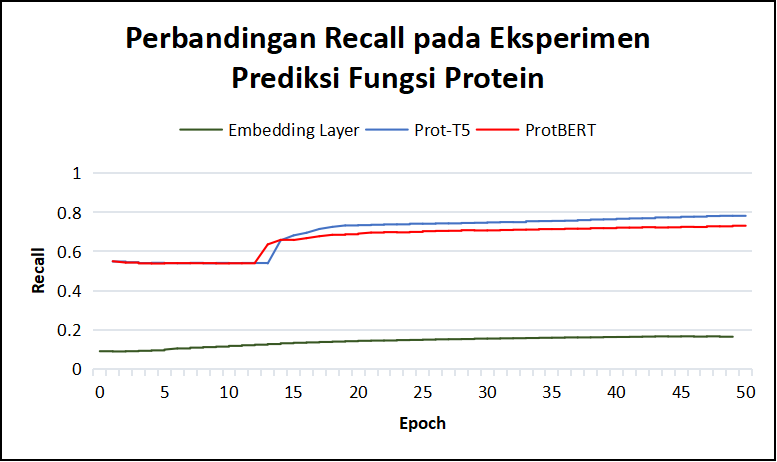
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Eksperimen** |  | **Metriks Evaluasi** | | | |
| ***Binary accuracy*** | **AUC** | ***Precision*** | ***Recall*** | ***F1-Score*** |
| Eksperimen 1 | 0,98092 | 0,90764 | 0,63005 | 0,16708 | 0,26412 |
| Eksperimen 2 | 0,99833 | 0.99931 | 0,73551 | 0,79108 | 0,76228 |
| Eksperimen 3 | 0,99788 | 0,99914 | 0,67005 | 0,73348 | 0,70033 |

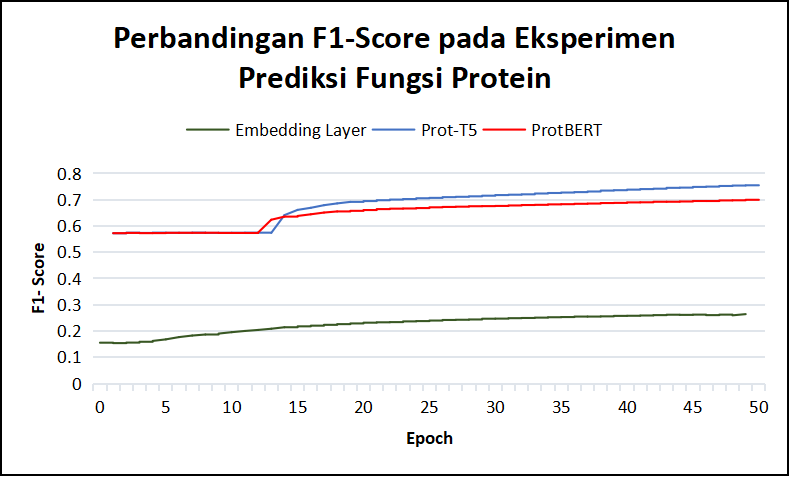
Berdasarkan Tabel 6.5, ditampilkan perbandingan nilai metriks evaluasi dari eksperimen yang dilakukan untuk memprediksi fungsi protein. Grafik ini memberikan gambaran visual mengenai performa masing-masing metode dalam memprediksi fungsi protein.











Gambar 6.4 Grafik Perbandingan Metriks Evaluasi Eksperimen

Pada Tabel 6.5 dan Gambar 6.4 diatas, ditampilkan hasil dari tiga eksperimen yang diukur berdasarkan matriks evaluasi *binary accuracy*, AUC, *precision*, *recall*, dan *F1-score*. Berikut ini adalah penjelasan perbandingan metriks evaluasi pada eksperimen tersebut.

1. **Eksperimen 1**

Pada eksperimen pertama, menghasilkan nilai *binary\_accuracy* sebesar 0,98092. Nilai ini menunjukkan bahwa 98,092% dari seluruh prediksi yang dilakukan model adalah benar. Nilai AUC, sebesar 0,90764 menunjukkan bahwa model memiliki kemampuan yang baik dalam membedakan antara kelas positif dan negatif. Nilai AUC yang tinggi (mendekati 1) mengindikasikan performa prediksi yang bagus secara keseluruhan. Meskipun demikian, nilai *precision* sebesar 0,63005 dan *recall* sebesar 0,16708 menunjukkan adanya ketidakseimbangan. *Precision* yang relatif tinggi menunjukkan bahwa ketika model memprediksi kelas positif, 63,005% dari prediksi tersebut benar. Namun, *recall* yang sangat rendah menunjukkan bahwa model hanya mampu mendeteksi 16,708% dari semua kasus positif yang sebenarnya, sehingga banyak kasus positif yang terlewat. Kombinasi *precision* dan *recall* yang rendah ini menghasilkan nilai *F1-score* sebesar 0,26412, yang merupakan rata-rata dari *precision* dan *recall*. Nilai *F1-score* yang rendah ini menunjukkan bahwa model memiliki kinerja yang buruk dalam menangkap dan memprediksi fungsi protein dengan tepat, terutama dalam mendeteksi semua kasus positif.

1. **Eksperimen 2**

Pada eksperimen kedua ini, menunjukkan nilai *binary\_accuracy* sebesar 0,99833, berarti model ini dapat mengklasifikasikan dengan benar hampir semua data. Nilai AUC yang sangat tinggi, yaitu 0,99931, menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam membedakan antara kelas positif dan negatif, yang artinya prediksinya sangat akurat. Nilai *precision* sebesar 0,73551 menunjukkan bahwa dari semua prediksi positif yang dibuat oleh model, sekitar 73,55% adalah benar-benar positif. Nilai *recall* sebesar 0,79108 yang mengindikasikan bahwa model mampu mengidentifikasi sekitar 79,11% dari semua contoh positif yang ada. *Precision* dan *recall* yang tinggi menunjukkan bahwa model ini tidak hanya akurat, tetapi juga berhasil menemukan sebagian besar data positif. Nilai *F1-score* sebesar 0,76228 menunjukkan bahwa model ini bagus dalam dua hal, yaitu dalam memprediksi data dengan benar *(Precision)*, tetapi juga mampu menangkap sebagian besar data yang ada *(Recall)*. Dalam hal ini, eksperimen dua menggunakan file *pre-trained* T5-*embedding* dan model LSTM dapat diandalkan dan cocok untuk digunakan karena memberikan prediksi yang akurat dengan sedikit kesalahan.

1. **Eksperimen 3**

Pada eksperimen ketiga ini, menunjukkan peningkatan yang signifikan dengan nilai *binary\_accuracy* sebesar 0,99788, yang berarti 99,788% dari prediksi yang dilakukan adalah benar. Nilai AUC yang sangat tinggi yaitu 0,99914 menunjukkan bahwa model hampir sempurna dalam membedakan antara kelas positif dan negatif, yang menandakan prediksi yang sangat baik. Nilai *precision* sebesar 0,67005 menunjukkan bahwa 67,005% dari prediksi positif adalah benar, sedangkan nilai *recall* sebesar 0,73348 menunjukkan bahwa model berhasil mendeteksi 73,348% dari semua kasus positif yang sebenarnya. Kedua metrik ini menunjukkan keseimbangan yang lebih baik antara *precision* dan *recall* dibandingkan dengan eksperimen pertama. Akibatnya, *F1-score* meningkat menjadi 0,70033, yang merupakan indikasi bahwa model ini memiliki kinerja yang lebih baik dalam mendeteksi dan memprediksi fungsi protein. Nilai *F1-score* yang lebih tinggi menunjukkan bahwa model tidak hanya baik dalam memprediksi kelas positif tetapi juga lebih andal dalam mendeteksi semua kasus positif, menghasilkan keseimbangan yang lebih baik antara *precision* dan *recall*.

Berdasarkan hasil metriks evaluasi dari ketiga eksperimen, ditemukan bahwa penggunaan *file* *pre-trained* dalam eksperimen kedua dan ketiga menghasilkan peningkatan signifikan dalam kinerja model dibandingkan dengan penggunaan *embedding layer* dari *tensorflow.keras* pada eksperimen pertama. Eksperimen kedua dan ketiga menunjukkan nilai *binary\_accuracy* yang hampir sempurna dan AUC yang sangat tinggi. *Precision* dan *recall* yang lebih seimbang juga menghasilkan *F1-score* yang lebih tinggi, menandakan model yang lebih andal dalam mendeteksi fungsi protein.

6.2.2 Pembahasan berdasarkan Hasil Prediksi Fungsi Protein

Pada sub bab ini, dijelaskan pembahasan terhadap eksperimen yang dilakukan dalam penelitian ini berdasarkan hasil prediksi fungsi protein. Selain membahas metriks evaluasi, penelitian ini juga membahas hasil prediksi yang diperoleh dari seluruh eksperimen yang dilakukan. Dalam hal prediksi fungsi protein, peneliti mengambil nilai probabilitas hasil prediksi yang berada di ambang batas di atas 0.5. Hal ini dikarenakan prediksi dengan probabilitas di atas 0.5 dianggap sebagai prediksi yang signifikan dan relevan untuk digunakan dalam hasil prediksi fungsi GO dari protein tersebut [55]. Ambang batas ini dipilih karena nilai tersebut dianggap memadai untuk memastikan keakuratan dan relevansi prediksi fungsi protein. Berikut ini akan dibahas mengenai hasil prediksi fungsi protein dari seluruh eksperimen.

6.2.2.1 Hasil Prediksi Fungsi Protein Eksperimen 1

Pada eksperimen ini, prediksi fungsi protein dilakukan dengan menggunakan *embedding layer* dari *Tensorflow.Keras* dengan menggunakan model LSTM. Hasil prediksi fungsi protein dari eksperimen ini dianalisis berdasarkan metriks evaluasi yang telah diperoleh pada Tabel 6.1, dimana hasil metriks evaluasi menunjukkan bahwa model berhasil mencapai *binary accuracy* sebesar 0,98092; AUC 0,90764; *precision* 0,63005; *recall* 0,16708; dan *F1-Score* 0,26412. Meskipun *binary accuracy* pada eksperimen ini tinggi, nilai *recall* yang rendah menunjukkan bahwa model kesulitan dalam mengidentifikasi semua fungsi protein yang benar-benar ada. Berikut ini adalah Tabel 6.6 yang berisi hasil prediksi fungsi protein dari implementasi yang telah dilakukan.

Tabel 6.6 Hasil Prediksi Fungsi Protein Eksperimen 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***No.*** | ***Protein\_ID*** | ***Actual GO\_Term & Ontology*** | ***Predict GO\_Term & Ontology*** | ***Overlap GO\_Term & Ontology*** |
| 1. | Q61824 | GO:0071944 (CCO)  GO:0005575 (CCO)  GO:0110165 (CCO)  GO:0016020 (CCO)  GO:0005886 (CCO)  GO:0005515 (MFO)  GO:0005488 (MFO)  GO:0003674 (MFO) | GO:0005575 (CCO)  GO:0008150 (BPO)  GO:0110165 (CCO)  GO:0003674 (MFO)  GO:0005488 (MFO)  GO:0005515 (MFO)  GO:0065007 (BPO)  GO:0050789 (BPO)  GO:0050794 (BPO)  GO:0050896 (BPO) | GO:0005575 (CCO)  GO:0110165 (CCO)  GO:0003674 (MFO)  GO:0005488 (MFO)  GO:0005515 (MFO) |
| 2. | P29618 | GO:0005515 (MFO)  GO:0005488 (MFO)  GO:0003674 (MFO) | GO:0005575 (CCO)  GO:0008150 (BPO)  GO:0110165 (CCO)  GO:0003674 (MFO)  GO:0005622 (CCO)  GO:0009987 (BPO)  GO:0005488 (MFO)  GO:0004672 (MFO) | GO:0003674 (MFO)  GO:0005488 (MFO) |
| 3. | A2RSY6 | GO:0008344 (BPO)  GO:0032501 (BPO)  GO:0030534 (BPO)  GO:0007610 (BPO)  GO:0008150 (BPO)  GO:0007626 (BPO) | GO:0005575 (CCO)  GO:0008150 (BPO)  GO:0110165 (CCO)  GO:0003674 (MFO)  GO:0005622 (CCO)  GO:0043226 (CCO)  GO:0043229 (CCO)  GO:0005488 (MFO) | GO:0008150 (BPO) |
| 4. | Q9JIX8 | GO:0005622 (CCO)  GO:0043229 (CCO)  GO:0031981 (CCO)  GO:0043226 (CCO)  GO:0110165 (CCO)  GO:0070013 (CCO)  GO:0005730 (CCO)  GO:0043233 (CCO)  GO:0043232 (CCO)  GO:0005575 (CCO)  GO:0043227 (CCO)  GO:0031974 (CCO)  GO:0005634 (CCO)  GO:0043231 (CCO)  GO:0043228 (CCO) | GO:0043228 (CCO)  GO:0110165 (CCO)  GO:0005622 (CCO)  GO:0043226 (CCO)  GO:0043229 (CCO)  GO:0043227 (CCO)  GO:0043231 (CCO)  GO:0005634 (CCO) | GO:0043228 (CCO)  GO:0110165 (CCO)  GO:0005622 (CCO)  GO:0043226 (CCO)  GO:0043229 (CCO)  GO:0043227 (CCO)  GO:0043231 (CCO)  GO:0005634 (CCO) |
| 5. | Q5F3W6 | GO:0005515 (MFO)  GO:0005488 (MFO)  GO:0003674 (MFO) | GO:0005575 (CCO)  GO:0008150 (BPO)  GO:0110165 (CCO)  GO:0003674 (MFO)  GO:0005622 (CCO)  GO:0043226 (CCO)  GO:0043229 (CCO)  GO:0005488 (MFO) | GO:0005488 (MFO)  GO:0003674 (MFO) |
| 6. | Q06807 | GO:0005737 (CCO)  GO:0005856 (CCO)  GO:0005576 (CCO)  GO:0005524 (MFO)  GO:0004714 (MFO)  GO:0030154 (BPO)  GO:0007169 (BPO)  GO:0007507 (BPO) | GO:0005575 (CCO)  GO:0008150 (BPO)  GO:0003674 (MFO)  GO:0030154 (BPO)  GO:0005488 (MFO)  GO:0071840 (BPO) | GO:0030154 (BPO) |
| 7. | Q3T0G1 | GO:0005634 (CCO)  GO:0016605 (CCO)  GO:0003700 (MFO)  GO:0042803 (MFO)  GO:0000978 (MFO)  GO:0043565 (MFO)  GO:0008270 (MFO)  GO:0006351 (BPO)  GO:0001935 (BPO)  GO:0006355 (BPO)  GO:0007346 (BPO)  GO:0006357 (BPO) | GO:0050789 (BPO)  GO:0050794 (BPO)  GO:0005634 (CCO)  GO:0019222 (BPO)  GO:0060255 (BPO)  GO:0003674 (MFO)  GO:0005488 (MFO)  GO:0010468 (BPO) | GO:0005634 (CCO) |

Berdasarkan Tabel 6.6, terdapat hasil prediksi fungsi protein untuk lima Protein\_ID yang dipilih. Setiap baris tabel memuat empat informasi utama, yaitu:

1. **Protein\_ID**, merupakan identifikasi unik untuk setiap protein yang ada dalam *dataset*. Dalam kolom Protein\_ID tersebut, terdapat dua Protein\_ID yang terbaru atau yang dibaharui yang tidak terdapat pada *train\_sequence.fasta* dalam *dataset* CAFA5 yaitu ‘Q06807’ dan ‘Q3T0G1’.
2. ***Actual* GO\_Term & *Ontology***, merupakan istilah GO dan ontologi yang benar-benar terkait dengan setiap protein berdasarkan data ground truth, dimana data *actual* tersebut diambil dari data *train\_terms.tsv*. Istilah ontologi pada data ini dibagi menjadi tiga kategori, yaitu:

* CCO (*Cellular Component Ontology*) adalah ontologi yang menjelaskan lokasi dalam sel tempat protein berada atau aktif.
* MFO (*Molecular Function Ontology*) adalah ontologi yang menjelaskan aktivitas dasar protein di tingkat molekul, seperti katalisis atau pengikatan.
* BPO (*Biological Process Ontology*) adalah ontologi yang menjelaskan proses biologis yang lebih besar yang melibatkan protein, seperti replikasi DNA atau metabolisme.

Untuk Protein\_ID 'Q06807' dan 'Q3T0G1', data aktual untuk istilah GO (GO\_Term) diambil dari *website* *UniProt* yang dapat diakses pada halaman https://www.uniprot.org/, dikarenakan Protein\_ID ini tidak terdapat pada *train\_terms.tsv* karena kedua Protein\_ID tersebut merupakan protein yang terbaru yang ditambahkan dalam *dataset* CAFA5, sehingga untuk data aktual istilah GO (GO\_Term) nya dapat dibandingkan ke data *UniProt*.

1. ***Predict* GO\_Term & Ontology**, merupakan istilah GO dan ontologi yang diprediksi oleh model LSTM berdasarkan *embedding layer* dari *TensorFlow.Keras*, dimana hal ini mencakup prediksi untuk kategori CCO, MFO, dan BPO.
2. ***Overlap* GO\_Term & Ontology**, merupakan istilah GO dan ontologi yang sesuai antara prediksi model dan data *ground truth*. Kolom ini menunjukkan istilah GO yang berhasil diprediksi dengan benar oleh model, dan mencerminkan kemampuan model untuk mengidentifikasi fungsi protein yang benar-benar ada.

Dari Tabel 6.6 ditunjukkan bahwa meskipun model mampu memprediksi beberapa fungsi protein dengan akurat, masih terdapat beberapa fungsi yang tidak terdeteksi oleh model. Hal ini terlihat dari jumlah istilah GO yang sesuai (*Overlap* GO\_Term & *Ontology*) yang bervariasi untuk setiap protein. Sebagai contoh, untuk Protein\_ID ‘Q61824’, terdapat lima istilah GO yang sesuai dari total delapan istilah GO yang sebenarnya, sedangkan untuk Protein\_ID ‘Q9JIX8’, terdapat delapan istilah GO yang sesuai dari total lima belas istilah GO yang sebenarnya. Hasil ini menunjukkan bahwa model LSTM dengan *embedding layer* dari *TensorFlow.Keras* memiliki potensi dalam memprediksi fungsi protein, namun masih perlu ditingkatkan terutama dalam hal *recall*, agar dapat mengidentifikasi semua fungsi protein yang benar-benar adaa tau benar-benar merupakan fungsi dari protein tersebut. Evaluasi dengan menggunakan metrik seperti *precision* dan *recall* penting untuk mendapatkan gambaran yang lebih lengkap tentang performa model dalam tugas prediksi fungsi protein ini. Untuk menambah informasi terkait hasil prediksi fungsi protein dari *Overlap* GO\_Term & *Ontology* pada eksperimen ini, ditampilkan tabel yang memuat istilah GO yang berhasil diprediksi dengan skor probabilitas yang dihasilkan oleh model serta deskripsi singkat dari istilah GO tersebut, yang dapat dilihat pada Tabel 6.7.

Tabel 6.7 Hasil Prediksi Fungsi Protein dengan Skor Probabilitas pada Eksperimen 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | ***Protein\_ID*** | ***Overlap GO\_Term Eksperimen 1*** | ***Probability Score*** | ***Ontology*** | ***Deskripsi GO\_Term*** |
| 1. | Q61824 | GO:0005575 | 0,6554 | CCO | *cellular component* |
| GO:0110165 | 0,6442 | CCO | *cellular anatomical entity* |
| GO:0003674 | 0,6409 | MFO | *molecular function* |
| GO:0005488 | 0,5808 | MFO | *binding* |
| GO:0005515 | 0,5217 | MFO | *protein binding* |
| 2. | P29618 | GO:0003674 | 0,6683 | MFO | *molecular function* |
| GO:0005488 | 0,5161 | MFO | *binding* |
| 3. | A2RSY6 | GO:0008150 | 0.7189 | BPO | *biological process* |
| 4. | Q9JIX8 | GO:0043228 | 0,7309 | CCO | *non-membrane-bounded organelle* |
| GO:0110165 | 0,7190 | CCO | *cellular anatomical entity* |
| GO:0005622 | 0,7262 | CCO | *intracellular anatomical structure* |
| GO:0043226 | 0,6534 | CCO | *organelle* |
| GO:0043229 | 0,6441 | CCO | *intracellular organelle* |
| GO:0043227 | 0,5939 | CCO | *membrane-bounded organelle* |
| GO:0043231 | 0,5683 | CCO | *intracellular membrane-bounded organelle* |
| GO:0005634 | 0,5188 | CCO | *nucleus* |
| 5. | Q5F3W6 | GO:0005488 | 0,5166 | MFO | *binding* |
| GO:0003674 | 0,5610 | MFO | *molecular function* |
| 6. | Q06807 | GO:0030154 | 0,6647 | BPO | *cell differentiation* |
| 7. | Q3T0G1 | GO:0005634 | 0,5279 | CCO | *nucleus* |

Berdasarkan Tabel 6.7 terlihat bahwa model LSTM dengan *embedding layer* dari *TensorFlow.Keras* berhasil memprediksi beberapa istilah GO dengan skor probabilitas yang cukup tinggi, menunjukkan tingkat kepercayaan model terhadap prediksi tersebut. Pada Protein\_ID ‘Q61824’, model berhasil memprediksi lima istilah GO dengan skor probabilitas yang bervariasi dari 0,5217 hingga 0,6554, dimana hal ini menunjukkan model yang cukup andal dalam memprediksi istilah-istilah yang terkait dengan komponen seluler dan fungsi molekuler protein ini. Untuk Protein\_ID ‘P29618’, hanya terdapat dua istilah GO yang sesuai dengan prediksi model, yang keduanya termasuk dalam kategori *Molecular Function Ontology* (MFO) dengan skor probabilitas 0,5161 dan 0,6683. Hal ini mengindikasikan bahwa model memiliki kemampuan terbatas dalam mendeteksi fungsi protein ini. Selanjutnya, pada Protein\_ID ‘A2RSY6’ menunjukkan hasil prediksi yang terbatas, dengan hanya satu istilah GO yang berhasil diprediksi oleh model sesuai dengan data aktual. Istilah GO yang sesuai adalah ‘GO:0008150’, yang berhasil diprediksi dengan skor probabilitas 0,7189. Pada Protein\_ID ‘Q9JIX8’, model berhasil memprediksi delapan istilah GO yang terkait dengan komponen seluler, dengan skor probabilitas tertinggi mencapai 0,7309 untuk istilah "*non-membrane-bounded organelle*". Kemudian, untuk Protein\_ID terakhir yaitu ‘Q5F3W6’, model hanya berhasil memprediksi dua istilah GO dalam kategori *Molecular Function Ontology* (MFO) dengan skor probabilitas 0,5166 dan 0,5610.

Selanjutnya, untuk Protein\_ID ‘Q06807’, model berhasil memprediksi istilah GO ‘GO:0030154’ yang termasuk dalam kategori *Biological Process Ontology* (BPO) dengan skor probabilitas 0,6647. Istilah ini menunjukkan bahwa model dapat mengenali proses biologis yang terkait dengan diferensiasi sel pada protein ini. Terakhir, pada Protein\_ID ‘Q3T0G1’, model berhasil memprediksi istilah GO ‘GO:0005634’ dengan skor probabilitas 0,5279, yang menunjukkan komponen seluler "*nucleus*". Hasil ini menunjukkan bahwa meskipun model mampu memprediksi beberapa istilah GO dengan cukup akurat, masih terdapat beberapa keterbatasan dalam mendeteksi fungsi-fungsi protein yang lebih kompleks. Secara keseluruhan, model menunjukkan kinerja yang baik dalam beberapa kasus, namun perlu peningkatan lebih lanjut terutama dalam mengidentifikasi fungsi protein yang lebih kompleks dan langka.

Dalam eksperimen ini, hasil *probability score* yang tidak cukup tinggi pada *embedding layer* dari *TensorFlow.Keras* disebabkan oleh kompleksitas yang lebih rendah dalam representasi fitur yang dihasilkan, dimana *embedding layer* mungkin tidak mampu menangkap informasi mendalam dan kompleks tentang urutan asam amino dalam hubungannya dengan fungsi protein yang diprediksi. Hal ini dapat mengakibatkan keterbatasan dalam kemampuan model untuk mengenali pola dan hubungan yang lebih rumit dalam data protein. Oleh karena itu, penggunaan metode *embedding* menggunakan *Protein Language Model* (PLM) lebih baik digunakan dikarenakan PLM dirancang khusus untuk memahami struktur dan fungsi protein dengan menggunakan data *training* yang lebih kaya, yang mencakup berbagai urutan protein dari banyak spesies dan fungsi.

6.2.2.2 Hasil Prediksi Fungsi Protein Eksperimen 2

Pada eksperimen ini, prediksi fungsi protein dilakukan dengan menggunakan PLM (*Protein Language Model* yaitu T5-*Embedding*) dan model LSTM. Hasil prediksi fungsi protein dari eksperimen ini dianalisis berdasarkan metriks evaluasi yang telah diperoleh pada Tabel 6.1, dimana hasil metriks evaluasi menunjukkan bahwa model berhasil mencapai *binary accuracy* sebesar 0,99833; AUC 0,99931; *precision* 0,73551; *recall* 0,79108; dan *F1-Score* 0,76228. Hasil ini menunjukkan bahwa penggunaan T5-*Embedding* memberikan kontribusi positif terhadap kinerja model, dengan metrik evaluasi yang sangat baik. Berikut ini adalah Tabel 6.8 yang berisi hasil prediksi fungsi protein dari implementasi yang telah dilakukan.

Tabel 6.8 Hasil Prediksi Fungsi Protein Eksperimen 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***No.*** | ***Protein\_ID*** | ***Actual GO\_Term & Ontology*** | ***Predict GO\_Term & Ontology*** | ***Overlap GO\_Term & Ontology*** |
| 1. | Q61824 | GO:0071944 (CCO)  GO:0005575 (CCO)  GO:0110165 (CCO)  GO:0016020 (CCO)  GO:0005886 (CCO)  GO:0005515 (MFO)  GO:0005488 (MFO)  GO:0003674 (MFO) | GO:0005575 (CCO)  GO:0008150 (BPO)  GO:0110165 (CCO)  GO:0003674 (MFO)  GO:0005488 (MFO)  GO:0005515 (MFO)  GO:0065007 (BPO)  GO:0050789 (BPO)  GO:0050794 (BPO)  GO:0050896 (BPO) | GO:0005575 (CCO)  GO:0110165 (CCO)  GO:0003674 (MFO)  GO:0005488 (MFO)  GO:0005515 (MFO) |
| 2. | P29618 | GO:0005515 (MFO)  GO:0005488 (MFO)  GO:0003674 (MFO) | GO:0005575 (CCO)  GO:0008150 (BPO)  GO:0110165 (CCO)  GO:0003674 (MFO)  GO:0005622 (CCO)  GO:0009987 (BPO)  GO:0005488 (MFO)  GO:0004672 (MFO) | GO:0003674 (MFO)  GO:0005488 (MFO) |
| 3. | A2RSY6 | GO:0008344 (BPO)  GO:0032501 (BPO)  GO:0030534 (BPO)  GO:0007610 (BPO)  GO:0008150 (BPO)  GO:0007626 (BPO) | GO:0005575 (CCO)  GO:0110165 (CCO)  GO:0008150 (BPO)  GO:0003674 (MFO)  GO:0005622 (CCO)  GO:0009987 (BPO)  GO:0043226 (CCO)  GO:0043227 (CCO) | GO:0008150 (BPO) |
| 4. | Q9JIX8 | GO:0005622 (CCO)  GO:0043229 (CCO)  GO:0031981 (CCO)  GO:0043226 (CCO)  GO:0110165 (CCO)  GO:0070013 (CCO)  GO:0005730 (CCO)  GO:0043233 (CCO)  GO:0043232 (CCO)  GO:0005575 (CCO)  GO:0043227 (CCO)  GO:0031974 (CCO)  GO:0005634 (CCO)  GO:0043231 (CCO)  GO:0043228 (CCO) | GO:0043228 (CCO)  GO:0110165 (CCO)  GO:0005622 (CCO)  GO:0043226 (CCO)  GO:0043229 (CCO)  GO:0043227 (CCO)  GO:0043231 (CCO)  GO:0005634 (CCO) | GO:0005622 (CCO)  GO:0043226 (CCO) |
| 5. | Q5F3W6 | GO:0005515 (MFO)  GO:0005488 (MFO)  GO:0003674 (MFO) | GO:0005575 (CCO)  GO:0008150 (BPO)  GO:0110165 (CCO)  GO:0003674 (MFO)  GO:0005622 (CCO)  GO:0043226 (CCO)  GO:0043229 (CCO)  GO:0005488 (MFO) | GO:0005488 (MFO)  GO:0003674 (MFO) |
| 6. | Q06807 | GO:0005737 (CCO)  GO:0005856 (CCO)  GO:0005576 (CCO)  GO:0005524 (MFO)  GO:0004714 (MFO)  GO:0030154 (BPO)  GO:0007169 (BPO)  GO:0007507 (BPO) | GO:0005575 (CCO)  GO:0110165 (CCO)  GO:0008150 (BPO)  GO:0030154 (BPO)  GO:0003674 (MFO)  GO:0005488 (MFO)  GO:0043229 (CCO) | GO:0030154 (BPO) |
| 7. | Q3T0G1 | GO:0005634 (CCO)  GO:0016605 (CCO)  GO:0003700 (MFO)  GO:0042803 (MFO)  GO:0000978 (MFO)  GO:0043565 (MFO)  GO:0008270 (MFO)  GO:0006351 (BPO)  GO:0001935 (BPO)  GO:0006355 (BPO)  GO:0007346 (BPO)  GO:0006357 (BPO) | GO:0005575 (CCO)  GO:0110165 (CCO)  GO:0008150 (BPO)  GO:0005634 (CCO) | GO:0005634 (CCO) |

Berdasarkan Tabel 6.8, terdapat dua Protein\_ID terbaru yang tidak terdapat pada *train\_sequence.fasta* dalam *dataset* CAFA5, yaitu ‘Q06807’ dan ‘Q3T0G1’. Data aktual untuk istilah GO (GO\_Term) bagi Protein\_ID 'Q06807' dan 'Q3T0G1' diambil dari situs *UniProt* ([https://www.uniprot.org/](https://www.uniprot.org/" \t "_new)), akan tetapi untuk data aktual istilah GO Protein\_ID lainnya diambil dari *train\_terms.*tsv, karena kedua Protein\_ID ini tidak terdapat pada *train\_terms.tsv*. Kedua Protein\_ID tersebut merupakan protein terbaru yang ditambahkan dalam *dataset* CAFA5, sehingga data aktual istilah GO (GO\_Term) nya dibandingkan dengan data dari *UniProt*. Dari Tabel 6.8, yang ditunjukkan bahwa meskipun model mampu memprediksi beberapa fungsi protein dengan akurat, masih terdapat beberapa fungsi yang tidak terdeteksi oleh model. Hal ini terlihat dari jumlah istilah GO yang sesuai (*Overlap* GO\_Term & *Ontology*) yang bervariasi untuk setiap protein. Sebagai contoh, untuk Protein\_ID ‘Q61824’, terdapat lima istilah GO yang sesuai dari total delapan istilah GO yang sebenarnya, sedangkan untuk Protein\_ID ‘P29618’, terdapat dua istilah GO yang sesuai dari total tiga istilah GO yang sebenarnya. Pada Protein\_ID 'Q06807' berhasil memprediksi satu istilah GO yang sesuai dengan data aktual, yaitu ‘GO:0030154’. Begitu juga dengan Protein\_ID 'Q3T0G1', berhasil memprediksi satu istilah GO yang sesuai dengan data aktual, yaitu 'GO:0005634’.

Hasil ini menunjukkan bahwa model LSTM dengan *T5-Embedding* dari memiliki potensi dalam memprediksi fungsi protein. Sebagai contoh, untuk Protein\_ID 'Q61824', model memprediksi istilah GO dengan probabilitas tinggi untuk kategori CCO dan MFO. Dari total tujuh istilah GO yang sebenarnya, model berhasil memprediksi lima istilah GO yang sesuai. Sedangkan untuk Protein\_ID 'P29618', model menunjukkan performa yang baik dengan skor probabilitas pada beberapa istilah GO. Dari total tiga istilah GO yang sebenarnya, model berhasil memprediksi dua istilah dengan skor probabilitas yang tinggi.

Tabel 6.9 Hasil Prediksi Fungsi Protein dengan Skor Probabilitas pada Eksperimen 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | ***Protein\_ID*** | ***Overlap GO\_Term Eksperimen 2*** | ***Probability Score*** | ***Ontology*** | ***Deskripsi GO\_Term*** |
| 1. | Q61824 | GO:0005575 | 0,8220 | CCO | *cellular component* |
| GO:0110165 | 0,8031 | CCO | *cellular anatomical entity* |
| GO:0003674 | 0,7838 | MFO | *molecular function* |
| GO:0005488 | 0,6294 | MFO | *binding* |
| GO:0005515 | 0,5258 | MFO | *protein binding* |
| 2. | P29618 | GO:0003674 | 0,7247 | MFO | *molecular function* |
| GO:0005488 | 0,6392 | MFO | *binding* |
| 3. | A2RSY6 | GO:0008150 | 0.8446 | BPO | *biological process* |
| 4. | Q9JIX8 | GO:0005622 | 0,6991 | CCO | *intracellular anatomical structure* |
| GO:0043226 | 0,6208 | CCO | *organelle* |
| 5. | Q5F3W6 | GO:0003674 | 0,7443 | MFO | *molecular function* |
| GO:0005488 | 0,5589 | MFO | *binding* |
| 6. | Q06807 | GO:0030154 | 0,5717 | BPO | *cell differentiation* |
| 7. | Q3T0G1 | GO:0005634 | 0,6564 | CCO | *nucleus* |

Berdasarkan Tabel 6.9, terlihat bahwa model LSTM dengan *embedding layer* dari *TensorFlow.Keras* pada eksperimen 1 memiliki performa yang cukup baik namun belum optimal. Namun, pada eksperimen 2 yang menggunakan *T5-Embedding*, skor probabilitas yang dihasilkan lebih tinggi, menunjukkan keakuratan yang lebih baik dalam memprediksi fungsi protein. Contohnya, untuk Protein\_ID ‘Q61824’, model berhasil memprediksi istilah GO dengan skor probabilitas setinggi 0,8220 dan 0,8031, yang menunjukkan peningkatan signifikan dibandingkan eksperimen sebelumnya. Hal ini menunjukkan bahwa penggunaan *T5-Embedding* lebih efektif dalam menangkap informasi kompleks tentang urutan asam amino dan hubungannya dengan fungsi protein yang diprediksi. Dengan demikian, model LSTM dengan *T5-Embedding* menunjukkan potensi yang lebih baik untuk meningkatkan akurasi prediksi fungsi protein.

6.2.2.3 Hasil Prediksi Fungsi Protein Eksperimen 3

Pada eksperimen ini, prediksi fungsi protein dilakukan dengan menggunakan PLM (*Protein Language Model* yaitu ProtBERT) dan model LSTM. Hasil prediksi fungsi protein dari eksperimen ini dianalisis berdasarkan metriks evaluasi yang telah diperoleh pada Tabel 6.1, dimana hasil metriks evaluasi menunjukkan bahwa model berhasil mencapai *binary accuracy* sebesar 0,99788; AUC 0,99914; *precision*  0,67005; *recall* 0,73348; dan *F1-Score* 0,70033. Hasil ini menunjukkan bahwa penggunaan ProtBERT memberikan kontribusi positif terhadap kinerja model, dengan metrik evaluasi yang sangat baik. Berikut ini adalah Tabel 6.10 yang berisi hasil prediksi fungsi protein dari implementasi yang telah dilakukan.

Tabel 6.10 Hasil Prediksi Fungsi Protein Eksperimen 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***No.*** | ***Protein\_ID*** | ***Actual GO\_Term & Ontology*** | ***Predict GO\_Term & Ontology*** | ***Overlap GO\_Term & Ontology*** |
| 1. | Q61824 | GO:0071944 (CCO)  GO:0005575 (CCO)  GO:0110165 (CCO)  GO:0016020 (CCO)  GO:0005886 (CCO)  GO:0005515 (MFO)  GO:0005488 (MFO)  GO:0003674 (MFO) | GO:0005575 (CCO)  GO:0008150 (BPO)  GO:0110165 (CCO)  GO:0003674 (MFO)  GO:0005488 (MFO)  GO:0005515 (MFO)  GO:0065007 (BPO)  GO:0050789 (BPO)  GO:0050794 (BPO)  GO:0050896 (BPO) | GO:0005575 (CCO)  GO:0110165 (CCO)  GO:0003674 (MFO) |
| 2. | P29618 | GO:0005515 (MFO)  GO:0005488 (MFO)  GO:0003674 (MFO) | GO:0005575 (CCO)  GO:0008150 (BPO)  GO:0110165 (CCO)  GO:0003674 (MFO)  GO:0005622 (CCO)  GO:0009987 (BPO)  GO:0043226 (CCO)  GO:0043229 (CCO)  GO:0005488 (MFO)  GO:0043227 (CCO) | GO:0003674 (MFO) |
| 3. | A2RSY6 | GO:0008344 (BPO)  GO:0032501 (BPO)  GO:0030534 (BPO)  GO:0007610 (BPO)  GO:0008150 (BPO)  GO:0007626 (BPO) | GO:0005575 (CCO)  GO:0008150 (BPO)  GO:0110165 (CCO)  GO:0003674 (MFO)  GO:0005622 (CCO)  GO:0043226 (CCO)  GO:0043229 (CCO)  GO:0043227 (CCO)  GO:0005737 (CCO)  GO:0043231 (CCO) | GO:0008150 (BPO) |
| 4. | Q9JIX8 | GO:0005622 (CCO)  GO:0043229 (CCO)  GO:0031981 (CCO)  GO:0043226 (CCO)  GO:0110165 (CCO)  GO:0070013 (CCO)  GO:0005730 (CCO)  GO:0043233 (CCO)  GO:0043232 (CCO)  GO:0005575 (CCO)  GO:0043227 (CCO)  GO:0031974 (CCO)  GO:0005634 (CCO)  GO:0043231 (CCO)  GO:0043228 (CCO) | GO:0043228 (CCO)  GO:0110165 (CCO)  GO:0005622 (CCO)  GO:0043226 (CCO)  GO:0043229 (CCO)  GO:0043227 (CCO)  GO:0043231 (CCO)  GO:0005634 (CCO) | GO:0110165 (CCO)  GO:0005622 (CCO)  GO:0043226 (CCO)  GO:0043229 (CCO)  GO:0043227 (CCO)  GO:0043231 (CCO)  GO:0005634 (CCO) |
| 5. | Q5F3W6 | GO:0005515 (MFO)  GO:0005488 (MFO)  GO:0003674 (MFO) | GO:0005575 (CCO)  GO:0008150 (BPO)  GO:0110165 (CCO)  GO:0003674 (MFO)  GO:0005622 (CCO)  GO:0043226 (CCO)  GO:0043229 (CCO)  GO:0005488 (MFO) | GO:0005488 (MFO)  GO:0003674 (MFO) |
| 6. | Q06807 | GO:0005737 (CCO)  GO:0005856 (CCO)  GO:0005576 (CCO)  GO:0005524 (MFO)  GO:0004714 (MFO)  GO:0030154 (BPO)  GO:0007169 (BPO)  GO:0007507 (BPO) | GO:0005575 (CCO)  GO:0110165 (CCO)  GO:0008150 (BPO)  GO:0030154 (BPO)  GO:0003674 (MFO)  GO:0005488 (MFO)  GO:0043229 (CCO)  GO:0009987 (BPO)  GO:0005737 (CCO) | GO:0030154 (BPO) |
| 7. | Q3T0G1 | GO:0005634 (CCO)  GO:0016605 (CCO)  GO:0003700 (MFO)  GO:0042803 (MFO)  GO:0000978 (MFO)  GO:0043565 (MFO)  GO:0008270 (MFO)  GO:0006351 (BPO)  GO:0001935 (BPO)  GO:0006355 (BPO)  GO:0007346 (BPO)  GO:0006357 (BPO) | GO:0005575 (CCO)  GO:0110165 (CCO)  GO:0008150 (BPO)  GO:0003674 (MFO)  GO:0005634 (CCO)  GO:0005622 (CCO)  GO:0043226 (CCO)  GO:0032502 (BPO) | GO:0005634 (CCO) |

Berdasarkan Tabel 6.10, terdapat dua Protein\_ID terbaru yang tidak terdapat pada *train\_sequence.fasta* dalam *dataset* CAFA5, yaitu ‘Q06807’ dan ‘Q3T0G1’. Data aktual untuk istilah GO (GO\_Term) bagi Protein\_ID 'Q06807' dan 'Q3T0G1' diambil dari situs *UniProt* ([https://www.uniprot.org/](https://www.uniprot.org/" \t "_new)), akan tetapi untuk data aktual istilah GO Protein\_ID lainnya diambil dari *train\_terms.*tsv, karena kedua Protein\_ID ini tidak terdapat pada *train\_terms.tsv*. Kedua Protein\_ID tersebut merupakan protein terbaru yang ditambahkan dalam *dataset* CAFA5, sehingga data aktual istilah GO (GO\_Term) nya dibandingkan dengan data dari *UniProt*.

Dari Tabel 6.10 ditunjukkan bahwa meskipun model mampu memprediksi beberapa fungsi protein dengan akurat, masih terdapat beberapa fungsi yang tidak terdeteksi oleh model. Hal ini terlihat dari jumlah istilah GO yang sesuai (*Overlap* GO\_Term & *Ontology*) yang bervariasi untuk setiap protein. Sebagai contoh, untuk Protein\_ID ‘Q61824’, terdapat tiga istilah GO yang sesuai dari total delapan istilah GO yang sebenarnya, sedangkan untuk Protein\_ID ‘Q9JIX8’, terdapat delapan istilah GO yang sesuai dari total lima belas istilah GO yang sebenarnya. Hasil ini menunjukkan bahwa model LSTM dengan *embedding ProtBERT* memiliki potensi dalam memprediksi fungsi protein, namun masih perlu ditingkatkan terutama dalam hal *recall*, agar dapat mengidentifikasi semua fungsi protein yang benar-benar merupakan fungsi dari protein tersebut. Evaluasi dengan menggunakan metrik seperti *precision* dan *recall* penting untuk mendapatkan gambaran yang lebih lengkap tentang performa model dalam tugas prediksi fungsi protein ini. Untuk menambah informasi terkait hasil prediksi fungsi protein dari *Overlap* GO\_Term & *Ontology* pada eksperimen ini, ditampilkan tabel yang memuat istilah GO yang berhasil diprediksi dengan skor probabilitas yang dihasilkan oleh model serta deskripsi singkat dari istilah GO tersebut, yang dapat dilihat pada Tabel 6.11.

Tabel 6.11 Hasil Prediksi Fungsi Protein dengan Skor Probabilitas pada Eksperimen 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | ***Protein\_ID*** | ***Overlap GO\_Term Eksperimen 3*** | ***Probability Score*** | ***Ontology*** | ***Deskripsi GO\_Term*** |
| 1. | Q61824 | GO:0005575 | 0,7272 | CCO | *cellular component* |
| GO:0110165 | 0,7209 | CCO | *cellular anatomical entity* |
| GO:0003674 | 0,6105 | MFO | *molecular function* |
| 2. | P29618 | GO:0003674 | 0,6097 | MFO | *molecular function* |
| 3. | A2RSY6 | GO:0008150 | 0,6297 | BPO | *biological process* |
| 4. | Q9JIX8 | GO:0110165 | 0,6342 | CCO | *cellular anatomical entity* |
| GO:0005622 | 0,7209 | CCO | *intracellular anatomical structure* |
| GO:0043226 | 0,6105 | CCO | *organelle* |
| GO:0043229 | 0,6617 | CCO | *intracellular organelle* |
| GO:0043227 | 0,5810 | CCO | *membrane-bounded organelle* |
| GO:0043231 | 0,5650 | CCO | *intracellular membrane-bounded organelle* |
| GO:0005634 | 0,5154 | CCO | *nucleus* |
| 5. | Q5F3W6 | GO:0003674 | 0,6387 | MFO | *molecular function* |
| GO:0005488 | 0,5329 | MFO | *binding* |
| 6. | Q06807 | GO:0030154 | 0,8563 | BPO | *cell differentiation* |
| 7. | Q3T0G1 | GO:0005634 | 0,6435 | CCO | *nucleus* |

Berdasarkan Tabel 6.11terlihat bahwa model LSTM dengan *embedding* ProtBERT berhasil memprediksi beberapa istilah GO dengan skor probabilitas yang cukup tinggi, menunjukkan tingkat kepercayaan model terhadap prediksi tersebut. Pada Protein\_ID ‘Q61824’, model berhasil memprediksi tiga istilah GO dengan skor probabilitas yang bervariasi dari 0,6105 hingga 0,7272, menunjukkan bahwa model cukup andal dalam memprediksi istilah-istilah yang terkait dengan komponen seluler dan fungsi molekuler protein ini.

Untuk Protein\_ID ‘P29618’, model berhasil memprediksi satu istilah GO yang sesuai dengan skor probabilitas 0,6097, yang termasuk dalam kategori *Molecular Function Ontology* (MFO). Pada Protein\_ID ‘A2RSY6’, model berhasil memprediksi satu istilah GO dengan skor probabilitas 0,6297. Pada Protein\_ID ‘Q9JIX8’, model berhasil memprediksi tujuh istilah GO yang terkait dengan komponen seluler, dengan skor probabilitas tertinggi mencapai 0,7209 untuk istilah "*intracellular anatomical structure*". Kemudian, untuk Protein\_ID terakhir yaitu ‘Q5F3W6’, model berhasil memprediksi dua istilah GO dalam kategori *Molecular Function Ontology* (MFO) dengan skor probabilitas bervariasi dari 0,5329 hingga 0,6387. Untuk Protein\_ID ‘Q06807’, model berhasil memprediksi satu istilah GO dengan skor probabilitas sangat tinggi sebesar 0,8563. Selanjutnya, untuk Protein\_ID ‘Q3T0G1’, model memprediksi satu istilah GO yang sesuai dengan skor probabilitas 0,6435.

Hasil *probability score* yang cukup tinggi pada *embedding* ProtBERT menunjukkan bahwa model ini mampu menangkap informasi mendalam dan kompleks tentang urutan asam amino dalam hubungannya dengan fungsi protein yang diprediksi. Hal ini menunjukkan bahwa penggunaan metode *embedding* menggunakan *Protein Language Model* (PLM) lebih baik dibandingkan *embedding* *layer* dari TensorFlow.Keras, dikarenakan PLM dirancang khusus untuk memahami struktur dan fungsi protein dengan menggunakan data training yang lebih lengkap, yang mencakup berbagai urutan protein dari banyak spesies dan fungsi.

Hasil prediksi fungsi protein dapat bervariasi tergantung pada model yang digunakan. Pada kasus ini, terdapat tiga model yang dibandingkan yaitu *embedding layer*, ProtBERT, dan *T5-Embedding*. *Embedding layer* standar menggunakan representasi vektor dari *TensorFlow.Keras* untuk mendapatkan representasi dasar dari urutan protein. Meskipun sederhana dan mudah diimplementasikan, *embedding layer* memiliki keterbatasan dalam menangkap hubungan kompleks antar asam amino dalam urutan protein. Model ini hanya mengandalkan representasi vektor statis tanpa memperhatikan konteks yang lebih luas dalam urutan, sehingga cenderung menghasilkan *probability score* yang lebih rendah.

ProtBERT, di sisi lain, menggunakan arsitektur *bidirectional transformer* yang memungkinkan pemahaman urutan asam amino dari dua arah (maju dan mundur). Dengan 110 juta parameter dan 12 lapisan *encoder* yang masing-masing memiliki 12 *attention heads*, ProtBERT mampu menangkap berbagai jenis hubungan dan interaksi antar asam amino. Model ini dilatih khusus pada data urutan protein yang besar, sehingga mampu mengenali pola dan motif yang ada dalam urutan asam amino. Mekanisme *self-attention* pada ProtBERT memungkinkan model untuk mempertimbangkan setiap asam amino relatif terhadap semua asam amino lainnya, memberikan representasi yang lebih kaya dan akurat, dan menghasilkan *probability score* yang lebih tinggi dibandingkan *embedding layer*.

T5 (*Text-to-Text Transfer Transformer*) menggunakan arsitektur *transformer* yang sangat dalam dan kompleks dengan 24 *encoder* dan 24 *decoder layers*. Setiap *layer* dilengkapi dengan *self-attention* dan *feed-forward* *neural network*, dengan *decoder* juga memanfaatkan *cross-attention* untuk interaksi yang lebih mendalam antara *encoder* dan *decoder*. Dengan 770 juta parameter, *hidden size* sebesar 1024, dan 16 *attention heads*, model T5 mampu menangkap berbagai aspek penting dari urutan protein secara rinci. Arsitektur yang sangat dalam dan kompleks memungkinkan T5 menangkap lebih banyak informasi dan hubungan kompleks dalam urutan protein, menghasilkan representasi yang sangat rinci dan akurat, serta *probability score* tertinggi di antara ketiga model. Meskipun T5 memerlukan sumber daya komputasi yang sangat besar dan waktu pelatihan yang lebih lama, kemampuannya untuk fokus pada berbagai aspek dari urutan protein memberikan hasil prediksi yang paling akurat.

6.2.2.4 Hasil Prediksi Fungsi Protein Eksperimen 4

Prediksi fungsi protein selanjutnya dilakukan pada eksperimen keempat, dimana dilakukan kombinasi nilai probabilitas yang didapat dari model LSTM dan *embedding* data latih *ProtBERT* dengan probabilitas dari *diamond* *search*. Hasil eksperimen ini dilakukan dengan menggabungkan prediksi dari model *long-short-time-memory* (LSTM) dengan prediksi berdasarkan kesamaan urutan menggunakan *diamond*. Pembahasan ini bertujuan untuk mengetahui apakah dengan melakukan kombinasi prediksi dari model dan *diamond* itu dapat menaikkan hasil skor probabilitas dari sebuah prediksi fungsi protein. Pertama, dilakukan *diamond search* untuk menemukan urutan yang serupa atau yang memiliki kemiripan. Dalam hal ini nilai dari *e-value* yang digunakan sebesar 0,001 atau menggunakan nilai dari *e-value* default *diamond*. Dengan menggunakan data *train\_seqeuence.fasta* sebagai *database* *diamond* dan dengan menggunakan data *testsuperset.fasta* sebagai pencocokan urutan sekuens asam amino dengan *database* di dapatkanlah hasil pencocokan dari sekuens asam amino yang dilakukan menggunakan *diamond* yang disimpan dalam sebuah *file* yang bernama *matches.tsv*. Berikut adalah hasil dari pencocokan sekuens asam amino yang telah dilakukan menggunakan *diamond* dapat dilihat pada Tabel 6.12**.**

Tabel 6.12 Hasil matches.tsv

|  |  |  |  |
| --- | --- | --- | --- |
| ***qseqid*** | ***sseqid*** | ***pident*** | ***length*** |
| Q61824 | Q61824 | 100.0 | 246 |
| Q61824 | O43184 | 81.0 | 246 |
| P29618 | P29618 | 100.0 | 244 |
| P29618 | P24100 | 83.8 | 242 |
| A2RSY6 | A2RSY6 | 100.0 | 728 |
| A2RSY6 | Q7Z2T5 | 85.7 | 733 |
| Q9JIX8 | Q9JIX8 | 100.0 | 1246 |
| Q9JIX8 | Q9UKV3 | 80.7 | 1251 |
| Q5F3W6 | Q5F3W6 | 100.0 | 247 |
| Q5F3W6 | P61982 | 99.6 | 247 |
| Q06807 | Q02763 | 94.7 | 1125 |
| Q06807 | Q02858 | 92.2 | 1125 |
| Q3T0G1 | Q9NVV9 | 94.4 | 213 |
| Q3T0G1 | Q8WY91 | 49.4 | 89 |

Berdasarkan Tabel 6.12 hasil pencocokan sekuens asam amino menggunakan diamond menunjukkan beberapa pasangan sekuens yang memiliki tingkat kemiripan yang tinggi. Sebagai contoh, sekuens dengan ID Q61824 memiliki tingkat kemiripan 100% dengan dirinya sendiri, dan 81% dengan sekuens O43184 sepanjang 246 asam amino. Hal serupa terlihat pada sekuens P29618 yang menunjukkan kemiripan 100% dengan dirinya sendiri dan 83.8% dengan sekuens P24100 sepanjang 242 asam amino. Selain itu, sekuens A2RSY6 memiliki kemiripan 100% dengan dirinya sendiri dan 85.7% dengan sekuens Q7Z2T5 sepanjang 733 asam amino. Kemiripan tinggi lainnya dapat dilihat pada sekuens Q9JIX8 dengan dirinya sendiri (100%) dan dengan sekuens Q9UKV3 (80.7%) sepanjang 1251 asam amino. Pada sekuens Q5F3W6, terdapat kemiripan 100% dengan dirinya sendiri dan 99.6% dengan sekuens P61982 sepanjang 247 asam amino. Terakhir, sekuens Q06807 menunjukkan kemiripan 94.7% dengan sekuens Q02763 dan 92.2% dengan sekuens Q02858 sepanjang 1125 asam amino, serta sekuens Q3T0G1 memiliki kemiripan 94.4% dengan sekuens Q9NVV9 dan 49.4% dengan sekuens Q8WY91 sepanjang 213 asam amino. Hasil ini menunjukkan bahwa terdapat sejumlah besar kesamaan sekuens yang dapat ditemukan menggunakan metode diamond, yang pada akhirnya dapat digunakan untuk meningkatkan akurasi prediksi fungsi protein dengan menggabungkan informasi ini dengan prediksi dari model LSTM dengan embedding *ProtBERT* dan *Diamond Search*. Dengan menggabungkan hasil ini dengan prediksi dari model LSTM, diharapkan dapat meningkatkan akurasi prediksi fungsi protein, dimana penggabungan hasil dari *diamond search* dengan prediksi dari model LSTM dilakukan dengan jumlah dari model parameter pembobotan:

Pada rumus tersebut,  S merupakan skor akhir dari kombinas prediksi,  SDiamond merupakan skor yang diperoleh dari hasil pencocokan menggunakan *diamond search*, SLSTM-*ProtBERT* merupakan skor yang diperoleh dari model LSTM yang memanfaatkan *embedding* data latih dari *ProtBERT*, dan α merupakan parameter dari pembobotan yang bernilai 0 ≤ α ≤ 1. Dalam eksperimen ini, peran dari α adalah mengatur seberapa besar kontribusi dari masing-masing pendekatan terhadap skor akhir, dimana jika nilai α mendekati 1, maka skor akhir akan lebih banyak dipengaruhi oleh hasil model *diamond*. Sebaliknya, jika α mendekati 0, maka skor akhir lebih banyak dipengaruhi oleh hasil prediksi dari  *LSTM-ProtBERT*. Berikut adalah hasil prediksi fungsi protein pada eksperimen keempat dengan melakukan kombinasi menggunakan Protein\_ID yang memiliki kemiripan yang dapat dilihat pada tabel berikut.

**Tabel …** Hasil Prediksi Fungsi Protein Eksperimen 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***No.*** | ***Protein\_ID*** | ***Actual GO\_Term & Ontology*** | ***Predict GO\_Term & Ontology*** | ***Overlap GO\_Term & Ontology*** |
| 1. | Q61824 | GO:0071944 (CCO)  GO:0005575 (CCO)  GO:0110165 (CCO)  GO:0016020 (CCO)  GO:0005886 (CCO)  GO:0005515 (MFO)  GO:0005488 (MFO)  GO:0003674 (MFO) | GO:0005575 (CCO)  GO:0008150 (BPO)  GO:0110165 (CCO)  GO:0003674 (MFO)  GO:0005488 (MFO) | GO:0005575 (CCO)  GO:0110165 (CCO)  GO:0003674 (MFO)  GO:0005488 (MFO) |
| 2. | P29618 | GO:0005515 (MFO)  GO:0005488 (MFO)  GO:0003674 (MFO) | GO:0005575 (CCO)  GO:0008150 (BPO)  GO:0110165 (CCO)  GO:0003674 (MFO)  GO:0005622 (CCO)  GO:0009987 (BPO)  GO:0043226 (MFO)  GO:0043229 (MFO)  GO:0005488 (MFO) | GO:0005488 (MFO) |
| 3. | A2RSY6 | GO:0008344 (BPO)  GO:0032501 (BPO)  GO:0030534 (BPO)  GO:0007610 (BPO)  GO:0008150 (BPO)  GO:0007626 (BPO) | GO:0008150 (BPO)  GO:0003674 (MFO)  GO:0009987 (BPO)  GO:0005488 (MFO) | GO:0008150 (BPO) |
| 4. | Q9JIX8 | GO:0005622 (CCO)  GO:0043229 (CCO)  GO:0031981 (CCO)  GO:0043226 (CCO)  GO:0110165 (CCO)  GO:0070013 (CCO)  GO:0005730 (CCO)  GO:0043233 (CCO)  GO:0043232 (CCO)  GO:0005575 (CCO)  GO:0043227 (CCO)  GO:0031974 (CCO)  GO:0005634 (CCO)  GO:0043231 (CCO)  GO:0043228 (CCO) | GO:0005575 (CCO)  GO:0008150 (BPO)  GO:0110165 (CCO)  GO:0005622 (CCO)  GO:0043226 (CCO)  GO:0043229 (CCO)  GO:0043227 (CCO) | GO:0005575 (CCO)  GO:0110165 (CCO)  GO:0005622 (CCO)  GO:0043226 (CCO)  GO:0043229 (CCO)  GO:0043227 (CCO) |
| 5. | Q5F3W6 | GO:0005515 (MFO)  GO:0005488 (MFO)  GO:0003674 (MFO) | GO:0005575 (CCO)  GO:0008150 (BPO)  GO:0110165 (CCO)  GO:0003674 (MFO)  GO:0005622 (CCO)  GO:0005488 (MFO) | GO:0005488 (MFO)  GO:0003674 (MFO) |
| 6. | Q06807 | GO:0005737 (CCO)  GO:0005856 (CCO)  GO:0005576 (CCO)  GO:0005524 (MFO)  GO:0004714 (MFO)  GO:0030154 (BPO)  GO:0007169 (BPO)  GO:0007507 (BPO) | GO:0005575 (CCO)  GO:0110165 (CCO)  GO:0008150 (BPO)  GO:0003674 (MFO)  GO:0009987 (MFO)  GO:0005488 (MFO)  GO:0030154 (BPO) | GO:0030154 (BPO) |
| 7. | Q3T0G1 | GO:0005634 (CCO)  GO:0016605 (CCO)  GO:0003700 (MFO)  GO:0042803 (MFO)  GO:0000978 (MFO)  GO:0043565 (MFO)  GO:0008270 (MFO)  GO:0006351 (BPO)  GO:0001935 (BPO)  GO:0006355 (BPO)  GO:0007346 (BPO)  GO:0006357 (BPO) | GO:0005575 (CCO)  GO:0008150 (BPO)  GO:0110165 (CCO)  GO:0003674 (MFO)  GO:0005622 (CCO)  GO:0009987 (BPO)  GO:0043226 (CCO)  GO:0043229 (CCO)  GO:0005488 (MFO)  GO:0043227 (CCO)  GO:0005515 (MFO)  GO:0005634 (CCO) | GO:0005634 (CCO) |

Berdasarkan **tabel …**, hasil prediksi fungsi protein pada eksperimen keempat menunjukkan variasi dalam kesamaan antara prediksi dan fungsi aktual protein. Misalnya, untuk Protein\_ID Q61824, prediksi berhasil mengidentifikasi beberapa fungsi dengan overlap seperti GO:0005575 (CCO), GO:0110165 (CCO), GO:0003674 (MFO), dan GO:0005488 (MFO). Sementara itu, untuk Protein\_ID P29618, prediksi hanya menunjukkan satu overlap pada GO:0005488 (MFO), meskipun terdapat beberapa GO\_Term yang diprediksi. Untuk Protein\_ID A2RSY6, terdapat satu overlap pada GO:0008150 (BPO), meskipun ada beberapa prediksi lainnya yang tidak sesuai dengan fungsi aktualnya. Protein\_ID Q9JIX8 menunjukkan lebih banyak *overlap* dengan empat GO\_Term yang sesuai antara prediksi dan fungsi aktual, termasuk GO:0005575 (CCO), GO:0110165 (CCO), GO:0005622 (CCO), dan GO:0043226 (CCO). Protein\_ID Q5F3W6 juga menunjukkan dua overlap pada GO:0005488 (MFO) dan GO:0003674 (MFO). Protein\_ID Q06807 memiliki satu *overlap* pada GO:0030154 (BPO), dan Protein\_ID Q3T0G1 menunjukkan kesamaan pada GO:0005634 (CCO).

Dari hasil ini, terlihat bahwa kombinasi prediksi dari model LSTM-ProtBERT dan *diamond* *search* dapat meningkatkan akurasi prediksi fungsi protein dengan adanya beberapa *overlap* antara prediksi dan fungsi aktual protein. Kontribusi dari masing-masing pendekatan, yang diatur oleh parameter *α*, memainkan peran penting dalam menentukan seberapa besar pengaruh dari hasil prediksi model atau hasil pencocokan sekuens pada skor akhir. Dengan demikian, eksperimen ini menunjukkan potensi pendekatan kombinasi untuk meningkatkan prediksi fungsi protein dibandingkan dengan penggunaan salah satu metode saja.

**Tabel ...** Hasil Prediksi Fungsi Protein dengan Skor Probabilitas pada Eksperimen 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | ***Protein\_ID*** | ***Overlap GO\_Term Eksperimen 4*** | ***Probability Score*** | ***Ontology*** | ***Deskripsi GO\_Term*** |
| 1. | Q61824 | GO:0005575 | 0,7019 | CCO | *cellular component* |
| GO:0110165 | 0,6987 | CCO | *cellular anatomical entity* |
| GO:0003674 | 0,6910 | MFO | *molecular function* |
| GO:0005488 | 0,5804 | MFO | *molecular function* |
| 2. | P29618 | GO:0005488 | 0,5420 | MFO | *binding* |
| 3. | A2RSY6 | GO:0008150 | 0.6483 | BPO | *biological process* |
| 4. | Q9JIX8 | GO:0005575 | 0,8638 | CCO | *Cellular\_component* |
| GO:0110165 | 0,8604 | CCO | *cellular anatomical entity* |
| GO:0005622 | 0,7683 | CCO | *intracellular anatomical structure* |
| GO:0043226 | 0.7280 | CCO | *organelle* |
| GO:0043229 | 0.7200 | CCO | *intracellular organelle* |
| GO:0043227 | 0.6962 | CCO | *membrane-bounded organelle* |
| 5. | Q5F3W6 | GO:0003674 | 0,6633 | MFO | *molecular function* |
| GO:0005488 | 0,6104 | MFO | *binding* |
| 6. | Q06807 | GO:0030154 | 0,8690 | BPO | *cell differentiation* |
| 7. | Q3T0G1 | GO:0005634 | 0,7403 | CCO | *nucleus* |

Berdasarkan **tabel …**, hasil prediksi fungsi protein dengan skor probabilitas pada eksperimen keempat memberikan informasi lebih lanjut mengenai prediksi fungsi protein dan skor keakuratan dari masing-masing prediksi. Untuk Protein\_ID Q61824, terdapat empat GO\_Term dengan skor probabilitas yang cukup tinggi, yaitu GO:0005575 (cellular component) dengan skor 0.7019, GO:0110165 (cellular anatomical entity) dengan skor 0.6987, GO:0003674 (molecular function) dengan skor 0.6910, dan GO:0005488 (molecular function) dengan skor 0.5804. Protein\_ID P29618 memiliki satu GO\_Term, GO:0005488 (binding), dengan skor probabilitas 0.5420. Protein\_ID A2RSY6 menunjukkan satu GO\_Term, GO:0008150 (biological process), dengan skor probabilitas 0.6483.

Untuk Protein\_ID Q9JIX8, terdapat enam GO\_Term dengan skor probabilitas tinggi, yaitu GO:0005575 (cellular component) dengan skor 0.8638, GO:0110165 (cellular anatomical entity) dengan skor 0.8604, GO:0005622 (intracellular anatomical structure) dengan skor 0.7683, GO:0043226 (organelle) dengan skor 0.7280, GO:0043229 (intracellular organelle) dengan skor 0.7200, dan GO:0043227 (membrane-bounded organelle) dengan skor 0.6962. Protein\_ID Q5F3W6 memiliki dua GO\_Term, yaitu GO:0003674 (molecular function) dengan skor 0.6633 dan GO:0005488 (binding) dengan skor 0.6104. Protein\_ID Q06807 menunjukkan satu GO\_Term, GO:0030154 (cell differentiation), dengan skor probabilitas tertinggi dalam tabel, yaitu 0.8690. Terakhir, Protein\_ID Q3T0G1 memiliki satu GO\_Term, GO:0005634 (nucleus), dengan skor probabilitas 0.7403.

Hasil ini menunjukkan bahwa kombinasi prediksi dari model LSTM-ProtBERT dan diamond search memberikan skor probabilitas yang bervariasi, mencerminkan keakuratan dari setiap prediksi fungsi protein. Skor probabilitas yang lebih tinggi menunjukkan tingkat keakuratan yang lebih besar dalam prediksi fungsi protein tersebut, dan ini menunjukkan potensi pendekatan kombinasi untuk memberikan prediksi yang lebih akurat dan andal dalam menentukan fungsi protein berdasarkan data yang ada.

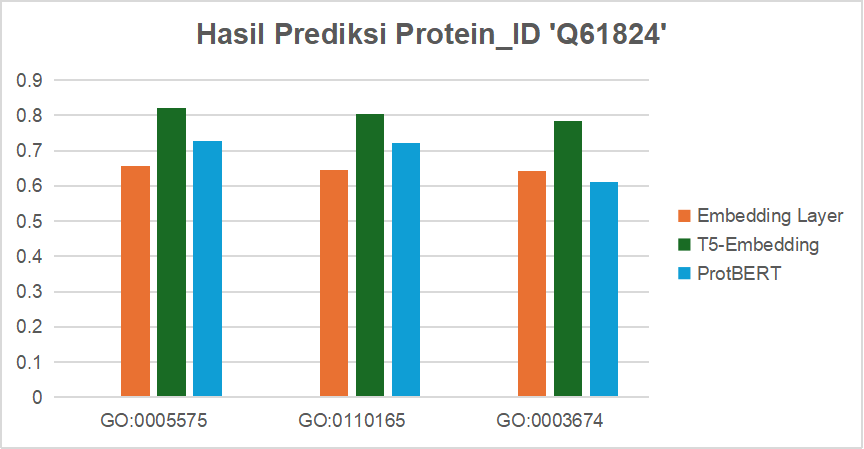
6.2.2.5 Perbandingan Hasil Prediksi Fungsi Protein terhadap Tiga Metode Embedding

Pada sub bab ini, akan dijelaskan mengenai analisis hasil prediksi fungsi protein terhadap eksperimen dengan menggunakan tiga metode *embedding* yang berbeda, yaitu *embedding layer*, *T5-embedding*, dan ProtBERT. Pada hasil prediksi fungsi protein terdapat skor probabilitas yang dihasilkan oleh masing-masing metode untuk Protein ID, yaitu 'Q61824', 'P29618', 'Q9JIX8', 'Q5F3W6', dan 'A2RSY6'. Analisis ini akan membahas perbandingan skor probabilitas untuk setiap istilah GO yang seragam dari hasil prediksi ketiga eksperimen dan mengidentifikasi kelebihan dan kekurangan dari setiap metode. Untuk melihat analisis tersebut, berikut adalah tabel yang berisi perbandingan skor probabilitas dari ketiga eksperimen yang dapat dilihat pada Tabel 6.14.

Tabel 6.14 Analisis Perbandingan Hasil Prediksi dari Ketiga Metode Embedding

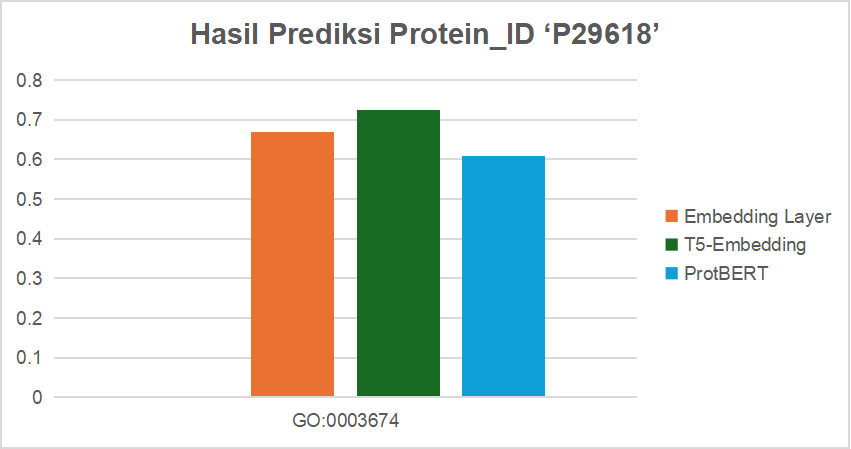
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Protein\_ID*** | ***GO\_Term*** | ***Embedding Layer*** | ***T5-Embedding*** | ***ProteBERT*** |
| Q61824 | GO:0005575 | 0,6554 | 0,8220 | 0,7272 |
| GO:0110165 | 0,6442 | 0,8031 | 0,7209 |
| GO:0003674 | 0,6409 | 0,7838 | 0,6105 |
| P29618 | GO:0003674 | 0,6683 | 0,7247 | 0,6097 |
| A2RSY6 | GO:0008150 | 0,7189 | 0,8446 | 0,6297 |
| Q9JIX8 | GO:0005622 | 0,6143 | 0,6991 | 0,7209 |
| GO:0043226 | 0,5979 | 0,6208 | 0,6105 |
| Q5F3W6 | GO:0005488 | 0,5166 | 0,5589 | 0,5329 |
| GO:0003674 | 0,5610 | 0,7443 | 0,6387 |
| Q06807 | GO:0030154 | 0,6647 | 0,5717 | 0,8563 |
| Q3T0G1 | GO:0005634 | 0,5279 | 0,6564 | 0,6435 |

Berdasarkan Tabel 6.14, dapat ditampilkan grafik perbandingan skor probabilitas dari ketiga metode *embedding* dalam memprediksi fungsi protein. Dalam grafik tersebut, terdapat Protein\_ID beserta istilah GO dan perbandingan skor probabilitas dari tiap metode *embedding*. Berdasarkan Tabel 6.14, Protein\_ID yang pertama adalah ‘Q61824’. Analisis perbandingan skor probabilitas terhadap hasil prediksi fungsi protein pada Protein\_ID 'Q61824' menunjukkan perbedaan yang signifikan antara metode *embedding layer*, *T5-Embedding*, dan ProtBERT. Pada istilah ‘GO:0005575’, *T5-Embedding* menghasilkan skor probabilitas tertinggi sebesar 0,822, diikuti oleh ProtBERT dengan skor 0,7272, dan *embedding layer* dengan skor 0,6554. Hal ini menunjukkan bahwa *T5-Embedding* memiliki kemampuan yang lebih baik dalam memprediksi fungsi protein untuk istilah GO ini. Selanjutnya, pada istilah ‘GO:0110165’, *T5-Embedding* kembali menunjukkan performa terbaik dengan skor 0,8031, diikuti oleh ProtBERT dengan skor 0,7209, dan *embedding layer* dengan skor 0,6442. Kembali, *T5-Embedding* unggul dalam menangkap informasi penting yang berkaitan dengan fungsi protein dibandingkan dengan kedua metode lainnya. Terakhir, pada istilah ‘GO:0003674, *T5-Embedding* juga menunjukkan skor probabilitas tertinggi sebesar 0,7838, diikuti oleh *embedding layer* dengan skor 0,6409, dan ProtBERT dengan skor 0,6105. Meskipun *embedding layer* menunjukkan skor yang lebih tinggi dibandingkan ProtBERT pada istilah GO ini, *T5-Embedding* tetap unggul dalam hal akurasi prediksi. Analisis perbandingan untuk Protein\_ID ini dapat dilihat pada Gambar 6.5.



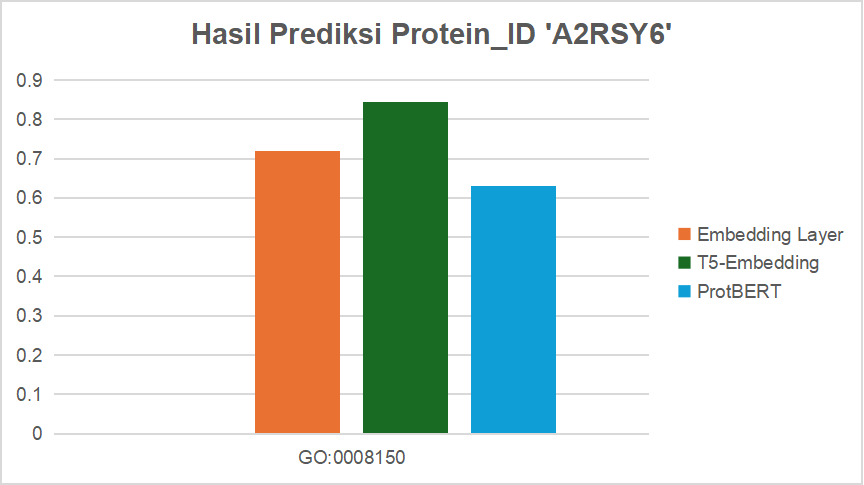
Gambar 6.5 Analisis Perbandingan Skor Probabilitas pada Protein\_ID 'Q61824'

Protein\_ID selanjutnya yang dibahas adalah ‘P29618’. Analisis perbandingan skor probabilitas terhadap hasil prediksi fungsi protein pada Protein\_ID 'P29618' menunjukkan perbedaan kinerja antara metode *embedding layer*, *T5-Embedding*, dan ProtBERT untuk istilah ‘GO:0003674’. *T5-Embedding* menghasilkan skor probabilitas tertinggi sebesar 0,7247, menunjukkan bahwa metode ini paling akurat dalam memprediksi fungsi protein untuk istilah GO ini. *Embedding layer* dari *TensorFlow.Keras* menghasilkan skor probabilitas sebesar 0,6683, yang menunjukkan bahwa metode ini masih dapat menangkap beberapa informasi penting terkait fungsi protein, tetapi tidak seakurat *T5-Embedding*. Sementara itu, ProtBERT menghasilkan skor probabilitas terendah sebesar 0,6097, menunjukkan bahwa dalam konteks ini, ProtBERT kurang efektif dibandingkan *T5-Embedding* dan *embedding layer*. Analisis perbandingan untuk Protein\_ID ini dapat dilihat pada Gambar 6.6.



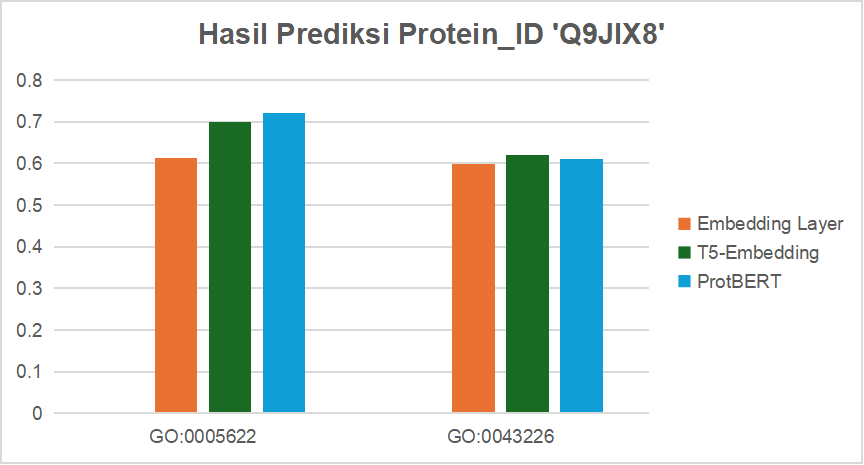
Gambar 6.6 Analisis Perbandingan Skor Probabilitas pada Protein\_ID 'P29618'

Selanjutnya, Protein\_ID yang akan dianalisis adalah ‘A2RSY6’. Analisis perbandingan skor probabilitas terhadap hasil prediksi fungsi protein pada Protein\_ID 'A2RSY6' untuk istilah ‘GO:0008150’ menunjukkan variasi kinerja antara metode *embedding layer*, *T5-Embedding*, dan ProtBERT. *T5-Embedding* menunjukkan kinerja terbaik dengan skor probabilitas tertinggi sebesar 0,8446, mengindikasikan kemampuannya yang lebih baik dalam menangkap informasi penting terkait fungsi protein. *Embedding layer* dari *TensorFlow.Keras* menghasilkan skor 0,7189, menunjukkan bahwa meskipun dapat menangkap informasi yang relevan, representasinya masih kurang mendalam dibandingkan dengan *T5-Embedding*. ProtBERT, dengan skor terendah 0,6297, kurang efektif dalam memprediksi fungsi protein untuk istilah ‘GO:0008150’ dibandingkan dengan dua metode lainnya. Analisis perbandingan untuk Protein\_ID ini dapat dilihat pada Gambar 6.7.



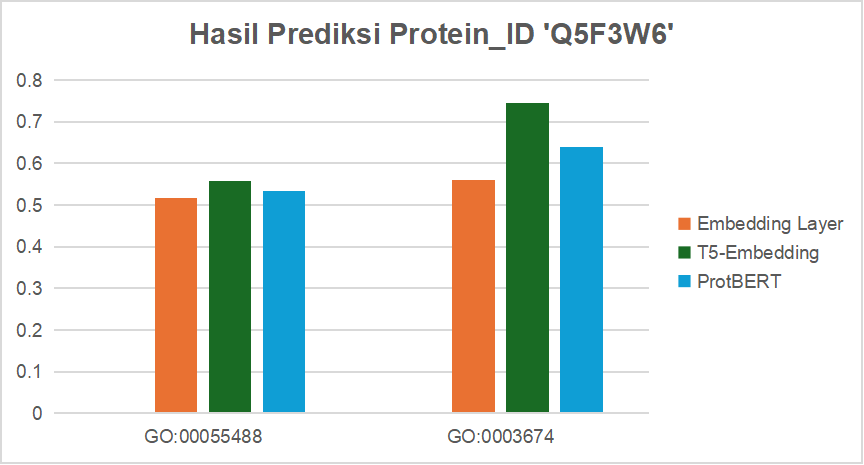
Gambar 6.7 Analisis Perbandingan Skor Probabilitas pada Protein\_ID 'A2RSY6'

Protein\_ID keempat yang akan dianalisis pada prediksi fungsi protein ini adalah ‘Q9JIX8’. Analisis perbandingan skor probabilitas terhadap hasil prediksi fungsi protein pada Protein\_ID 'Q9JIX8' menunjukkan variasi kinerja antara metode *embedding layer*, *T5-Embedding*, dan ProtBERT untuk dua istilah GO ‘GO:0005622’ dan ‘GO:0043226’. Untuk istilah ‘GO:0005622’, ProtBERT menunjukkan kinerja terbaik dengan skor probabilitas 0,7209, diikuti oleh *T5-Embedding* dengan skor 0,6991, dan *embedding layer* dari *TensorFlow.Keras* dengan skor 0,6143. Ini mengindikasikan bahwa ProtBERT lebih efektif dalam menangkap informasi terkait fungsi protein ini. Untuk istilah ‘GO:0043226’, *T5-Embedding* menunjukkan kinerja terbaik dengan skor probabilitas 0,6208, diikuti oleh ProtBERT dengan skor 0,6105, dan *embedding layer* dengan skor 0,5979. Ini menunjukkan bahwa *T5-Embedding* lebih baik dalam memprediksi fungsi protein untuk istilah ini. Analisis perbandingan untuk Protein\_ID ini dapat dilihat pada Gambar 6.8.



Gambar 6.8 Analisis Perbandingan Skor Probabilitas pada Protein\_ID 'Q9JIX8'

Untuk Protein\_ID 'Q5F3W6', analisis perbandingan skor probabilitas menunjukkan perbedaan yang signifikan dalam hasil prediksi fungsi protein di antara berbagai metode *embedding*. Pada istilah GO ‘GO:00055488’, skor probabilitas tertinggi diperoleh menggunakan metode *T5-Embedding* dengan nilai 0,5589, diikuti oleh ProtBERT dengan nilai 0,5329, dan *embedding layer* dengan nilai 0,5166. Hal ini menunjukkan bahwa metode *T5-Embedding* mungkin lebih baik dalam memprediksi fungsi protein untuk istilah GO ini dibandingkan dengan dua metode lainnya. Sementara itu, pada istilah GO ‘GO:0003674’, skor probabilitas tertinggi juga diperoleh dari *T5-Embedding* dengan nilai 0,7443, diikuti oleh ProtBERT dengan nilai 0,6387, dan *embedding layer* dengan nilai 0,561. Perbedaan skor ini mengindikasikan bahwa *T5-Embedding* cenderung memberikan hasil prediksi yang lebih akurat dibandingkan dengan *embedding layer* dan ProtBERT untuk kedua istilah GO tersebut. Secara keseluruhan, metode *T5-Embedding* menunjukkan kinerja yang lebih unggul dalam memprediksi fungsi protein pada Protein\_ID 'Q5F3W6', yang mungkin berimplikasi pada keefektifan metode ini dalam konteks analisis fungsi protein. Analisis perbandingan untuk Protein\_ID ini dapat dilihat pada Gambar 6.9.



Gambar 6.9 Analisis Perbandingan Skor Probabilitas pada Protein\_ID 'QFF3W6'

Selanjutnya, pada Protein\_ID 'Q06807', analisis skor probabilitas menunjukkan bahwa metode ProtBERT memberikan hasil yang paling tinggi dibandingkan dengan *embedding* layer dan T5-*Embedding*. Pada GO Term ID GO:0030154, ProtBERT menghasilkan skor probabilitas sebesar 0.8563, yang jauh lebih tinggi dibandingkan dengan skor yang diperoleh dari *Embedding* Layer (0.6647) dan T5-*Embedding* (0.5717). Hal ini menunjukkan bahwa ProtBERT, dengan kemampuannya untuk memahami konteks dan representasi mendalam dari data protein, lebih efektif dalam memprediksi fungsi protein pada GO Term ini. Skor yang lebih tinggi dari ProtBERT mengindikasikan kemampuannya untuk menangkap hubungan yang lebih kompleks dan relevansi yang lebih kuat dalam sekuens protein, menghasilkan prediksi yang lebih akurat. Sebaliknya, *Embedding* Layer dan T5-*Embedding* memberikan skor yang lebih rendah, mengindikasikan bahwa keduanya kurang efektif dalam memodelkan fitur-fitur kompleks dari protein untuk GO Term ini. Secara keseluruhan, hasil ini menegaskan keunggulan ProtBERT dalam konteks prediksi fungsi protein pada Protein\_ID 'Q06807'.

Metode *T5-Embedding* umumnya unggul karena kemampuannya untuk menangkap konteks yang lebih luas dan representasi yang lebih mendalam dari sekuens protein dibandingkan dengan metode lainnya. *T5-Embedding* merupakan model berbasis *transformer* yang secara signifikan unggul dalam memprediksi fungsi protein dibandingkan dengan *embedding layer* dan ProtBERT. Model ini menggunakan arsitektur *transformer* yang mampu menangkap konteks yang lebih luas dari urutan asam amino dengan memanfaatkan mekanisme perhatian (*attention mechanism*). Mekanisme ini memungkinkan model untuk mempertimbangkan hubungan antar elemen dalam urutan secara lebih efektif dan dalam skala yang lebih besar, sehingga meningkatkan kemampuan model untuk memahami dan memprediksi fungsi protein dengan lebih akurat. *T5-Embedding* dilatih pada berbagai jenis data dan tugas, memungkinkan model ini untuk memanfaatkan representasi yang lebih mendalam dan generalisasi yang lebih baik dalam konteks data protein, menghasilkan skor probabilitas yang lebih tinggi.

*Embedding layer*, di sisi lain, adalah metode yang lebih sederhana yang memetakan setiap asam amino ke dalam vektor berdimensi tetap sebelum diproses oleh model. Meskipun metode ini efektif dalam beberapa kasus, keterbatasannya terletak pada kemampuannya untuk menangkap hubungan kompleks antara asam amino. *Embedding layer* menghasilkan representasi yang tidak sekompleks dan sedalam model berbasis *transformer* seperti *T5-Embedding*. Akibatnya, metode ini seringkali memberikan skor probabilitas yang lebih rendah karena tidak dapat menangkap konteks dan interaksi antar fitur dengan efisien.

ProtBERT, yang juga merupakan model berbasis *transformer*, menunjukkan kinerja yang lebih baik dibandingkan dengan *embedding layer*, tetapi tidak sebaik *T5-Embedding*. ProtBERT dirancang khusus untuk data protein dan menggunakan pendekatan berbasis *transformer* untuk menghasilkan representasi yang mendalam dari sekuens protein. Meskipun ProtBERT memanfaatkan arsitektur *transformer*, kinerjanya mungkin terbatas oleh cara model ini dilatih dan dikonfigurasi. ProtBERT mungkin tidak sefleksibel atau seefisien *T5-Embedding* dalam menangkap konteks global dari data protein, sehingga memberikan hasil yang kurang optimal dibandingkan dengan *T5-Embedding*.

Dengan mempertimbangkan kemampuan setiap metode dalam memproses dan memahami data protein, dapat disimpulkan bahwa *T5-Embedding* adalah pilihan yang paling efektif untuk memprediksi fungsi protein, berkat kemampuannya dalam menangkap konteks dan hubungan yang kompleks dengan akurasi yang lebih tinggi. Meskipun *embedding layer* dan ProtBERT masing-masing memiliki kelebihan, keduanya tidak mampu menyamai kedalaman dan efektivitas *T5-Embedding* dalam memanfaatkan representasi yang mendalam dari sekuens protein. Oleh karena itu, dalam konteks analisis prediksi fungsi protein, *T5-Embedding* merupakan metode yang paling unggul dan dapat diandalkan.

# BAB VII KESIMPULAN DAN SARAN

Pada bab ini, akan menjelaskan kesimpulan yang telah diperoleh selama melakukan penelitian ini serta memberikan saran berdasarkan permasalahan ataupun temuan yang diperoleh selama melakukan pengerjaan Tugas Akhir yang berjudul *Protein Function Prediction using Deep Learning.*

## 7.1 Kesimpulan

Berdasarkan hasil yang diperoleh selama proses pengerjaan, terdapat beberapa kesimpulan yang dapat diambil selama proses tersebut, antara lain:

1. Pada penelitian ini menunjukkan bahwa penggunaan *file* *pre-trained* dalam eksperimen kedua dan ketiga secara signifikan meningkatkan kinerja model dibandingkan dengan penggunaan *embedding layer* dari *Tensorflow.Keras* pada eksperimen pertama. Hal ini tercermin dari nilai *binary\_accuracy* (0,99833) dan AUC (0,99931) yang hampir sempurna serta *F1-score* (0,76228) yang lebih tinggi, menandakan model yang lebih andal dalam mendeteksi fungsi protein. Dengan mengkombinasikan probabilitas kemiripan fungsi protein dari eksperimen ketiga dengan *diamond search* yang dicantumkan pada eksperimen keempat menunjukkan bahwa metode ini efektif dalam menemukan kemiripan fungsi protein dari Protein\_ID yang berbeda. Penelitian ini juga tidak hanya berhasil dalam memprediksi fungsi protein, tetapi hingga memprediksi fungsi protein berdasarkan aspeknya, yaitu *Biological Process* (BP), *Cellular Component* (CC), dan *Molecular Function* (MF), yang merupakan dasar yang kuat untuk prediksi fungsi protein yang lebih akurat.
2. Pada penelitian ini, performansi metode prediksi fungsi protein menggunakan *deep learning* dievaluasi dengan metrik *binary accuracy*, *precision*, *recall* dan *F1-score*. Pada metode *embedding layer* dari *TensorFlow* dan LSTM, meskipun memiliki *binary accuracy* dan *precision* yang relatif tinggi, *F1-score* yang rendah (0,26412) disebabkan oleh *recall* yang sangat rendah (0,16708). Hal ini menunjukkan bahwa meskipun model sering kali memprediksi dengan benar (*precision* tinggi) dan sering gagal mendeteksi sebagian besar fungsi protein yang sebenarnya ada (*recall* rendah), sehingga tidak seimbang dalam mengukur kualitas keseluruhan prediksinya. Sementara itu, pada metode *T5-Embedding*, *F1-score* mencapai 0,76228, yang menunjukkan kualitas prediksi yang sangat baik. Hal ini didukung oleh *binary accuracy* tinggi (0,99833), *precision* (0,73551), dan *recall* (0,79108) yang juga tinggi. Model ini tidak hanya akurat dalam membuat prediksi, tetapi juga konsisten dalam mendeteksi sebagian besar fungsi protein yang relevan, menciptakan keseimbangan yang baik antara *precision* dan *recall*. Metode *ProtBERT* *embedding* juga menunjukkan performa yang baik dengan *F1-score* sebesar 0,70033. Meskipun sedikit di bawah *T5-Embedding* dalam hal *F1-score*, ProtBERT *embedding* memberikan *binary accuracy* yang tinggi (0,99788), *precision* (0,67005), dan *recall* (0,73348), menunjukkan kemampuannya dalam mengenali dan memprediksi fungsi protein secara efektif. Berdasarkan hasil evaluasi ini, metode *T5-Embedding* adalah pilihan terbaik karena memberikan kualitas prediksi yang lebih baik secara keseluruhan, dengan keseimbangan yang optimal antara *precision* dan *recall*. Meskipun *ProtBERT* *embedding* juga efektif, *T5-Embedding* tetap unggul dalam konsistensi dan ketepatan deteksi fungsi protein yang benar.
3. Penelitian ini berhasil mengimplementasikan pendekatan *deep learning* untuk melakukan anotasi fungsi protein dari sekuens asam amino dalam *dataset* CAFA5. Melalui langkah-langkah yang terstruktur, mulai dari persiapan data hingga evaluasi model, penelitian ini menunjukkan bahwa model *deep learning* seperti LSTM atau *transformer* dapat efektif dalam memprediksi fungsi biologis (BP), komponen sel (CC), dan fungsi molekuler (MF) dari protein. Untuk menghasilkan anotasi fungsi protein yang akurat, prediksi fungsi dari model akan disesuaikan dengan data aktual yang terdapat dalam *file* pelatihan (*train\_terms.tsv*). Hal ini memastikan bahwa prediksi yang dihasilkan sesuai dengan anotasi GO\_Term yang telah ditetapkan sebelumnya, meningkatkan keakuratan dan kemampuan kita untuk memahami hasil prediksi dalam konteks biologi yang lebih luas.

## 7.2 Saran

Berdasarkan hasil yang diperoleh, saran yang dapat dieksplorasi dalam penelitian selanjutnya adalah:

1. Dalam penelitian ini, tidak dapat menggunakan seluruh label atau GO\_Term yang ada, dimana jumlah keseluruhan label pada *dataset* adalah 31.466. Karena keterbatasan dalam komputasi, penelitian ini hanya menggunakan 1.500 label. Untuk penelitian selanjutnya, disarankan untuk mengeksplorasi penggunaan lebih banyak label dengan meningkatkan kapasitas komputasi. Dengan demikian, model dapat mencakup lebih banyak fungsi protein yang mungkin, sehingga meningkatkan keakuratan dan kegunaan hasil prediksi dalam aplikasi nyata.
2. Dalam penelitian ini, tidak dapat menggunakan ukuran panjang maksimal sekuens protein sebagai panjang *padding*, dimana panjang maksimal sekuens protein adalah 35.213, sedangkan panjang *padding* yang digunakan adalah 1.024. Hal ini dilakukan karena keterbatasan komputasi. Untuk penelitian selanjutnya, disarankan untuk mengeksplorasi penggunaan panjang *padding* yang lebih panjang atau teknik pemangkasan dan *padding* yang lebih adaptif, dengan meningkatkan kapasitas komputasi. Hal ini akan memungkinkan model untuk memproses informasi dari sekuens protein yang lebih panjang dan kompleks, sehingga meningkatkan akurasi prediksi fungsi protein.
3. Untuk meningkatkan *recall* dan *F1-Score* pada eksperimen menggunakan *embedding layer* dari *Tensorflow.Keras*, disarankan untuk mengeksplorasi *dataset* yang digunakan. Salah satu faktor yang mempengaruhi kinerja model adalah keterbatasan variasi dan jumlah data dalam pelatihan. Dengan memperluas *dataset*, baik dalam jumlah maupun keragaman sekuens protein, model dapat belajar dari lebih banyak contoh yang representatif. *Dataset* yang lebih besar dan beragam membantu model menangkap pola yang lebih kompleks, sehingga meningkatkan kemampuan dalam mengenali dan memprediksi fungsi protein dengan lebih akurat. Penelitian ini tidak dapat menggunakan keseluruhan data karena keterbatasan komputasi, sehingga eksplorasi *dataset* yang lebih besar di masa depan akan sangat bermanfaat untuk meningkatkan kinerja model.
4. Disarankan untuk menggunakan *embedding* dari *Protein Language Model* yang sudah terlatih sebelumnya daripada menggunakan *embedding* *layer* bawaan dari *TensorFlow.Keras*. Hal ini dapat secara signifikan meningkatkan kinerja model dalam memprediksi fungsi protein. *Embedding* dari *Protein Language Model* memiliki kemampuan untuk menangkap informasi yang lebih kaya dan spesifik terkait dengan sekuens protein, karena telah dilatih pada data sekuens protein yang besar dan bervariasi. Akibatnya, model dapat memahami dan menerjemahkan sekuens protein dengan lebih akurat, yang dapat menghasilkan prediksi fungsi protein yang lebih tepat dan handal.

# DAFTAR PUSTAKA

[1] B. CHEN, “Chapter 2 Protein Sequence Motif Information Discovery.” https://www.oreilly.com/library/view/algorithmic-and-artificial/9781118567814/9781118567814c02.xhtml#c2\_level1\_1.

[2] D. A. R. Muhammad Rifqi Hariri, Miftahul Jannah, “METODE BIOLOGI MOLEKULER,” *August 2021*, 2021. https://www.researchgate.net/publication/354172865\_METODE\_BIOLOGI\_MOLEKULER.

[3] T. T. Duong Vu and J. Jung, “Protein function prediction with gene ontology: From traditional to deep learning models,” *PeerJ*, vol. 9, 2021, doi: 10.7717/peerj.12019.

[4] D. Piovesan, M. Giollo, E. Leonardi, C. Ferrari, and S. C. E. Tosatto, “INGA: Protein function prediction combining interaction networks, domain assignments and sequence similarity,” *Nucleic Acids Res.*, vol. 43, no. W1, pp. W134–W140, 2015, doi: 10.1093/nar/gkv523.

[5] M. Kulmanov and R. Hoehndorf, “DeepGOPlus: Improved protein function prediction from sequence,” *Bioinformatics*, vol. 36, no. 2, pp. 422–429, 2020, doi: 10.1093/bioinformatics/btz595.

[6] X. L. Liu, “Deep Recurrent Neural Network for Protein Function Prediction from Sequence.”

[7] N. Zhou *et al.*, “The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens,” *Genome Biol.*, vol. 20, no. 1, pp. 1–23, 2019, doi: 10.1186/s13059-019-1835-8.

[8] B. P. Yolanda Smith, “Amino Acids and Protein Sequences.” https://www.news-medical.net/life-sciences/Amino-Acids-and-Protein-Sequences.aspx#:~:text=There are two main methods,protein needs to be characterized.

[9] O. S. Saleh, “Bahan Ajar Bahan Ajar Bahan Ajar,” no. Mkb 7056, pp. 1–101, 2016.

[10] A. Jeklin *et al.*, “済無No Title No Title No Title,” *Corresp. Análisis*, no. 15018, pp. 1–23, 2016.

[11] A. Rahmah, “Mengenal Tahapan Sintesis Protein, Tujuan dan Prosesnya,” *Rabu, 30 Mar 2022*, 2022. https://www.detik.com/edu/detikpedia/d-6008247/mengenal-tahapan-sintesis-protein-tujuan-dan-prosesnya.

[12] A. Amino and P. Protein, “2. Asam Amino, Peptida dan Protein 1.,” pp. 15–35.

[13] F. Soleymani, E. Paquet, H. Viktor, W. Michalowski, and D. Spinello, “Protein–protein interaction prediction with deep learning: A comprehensive review,” *Comput. Struct. Biotechnol. J.*, vol. 20, pp. 5316–5341, 2022, doi: 10.1016/j.csbj.2022.08.070.

[14] Ф. Котлер, “No TitleМаркетинг по Котлеру,” p. 282, 2008.

[15] A. Tasari, D. Dinata Tarigan, E. Nia, D. B. Purba, K. Saputra, and I. Artikel, “Perbandingan Algoritma Support Vector Machine dan KNN dalam Memprediksi Struktur Sekunder Protein,” *J. Inform.*, vol. 9, no. 2, pp. 172–179, 2022, [Online]. Available: https://raw.githubusercontent.com/JiayingLi/Pr.

[16] “Levels of Protein Organization,” *2014*. https://comis.med.uvm.edu/VIC/coursefiles/MD540/MD540-Protein\_Organization\_10400\_574581210/Protein-org/Protein\_Organization\_print.html.

[17] S. Sengupta *et al.*, “Role and Application of Biocatalysts in Cancer Drug Discovery,” *Catalysts*, vol. 13, no. 2, 2023, doi: 10.3390/catal13020250.

[18] H. Mohimani *et al.*, “乳鼠心肌提取 HHS Public Access,” *Physiol. Behav.*, vol. 176, no. 3, pp. 139–148, 2017, doi: 10.1002/cpt.435.Transporters.

[19] V. R. Rahman, M. A. Bratadiredja, and N. M. Saptarini, M.Si, Apt., “Artikel Review: Potensi Kolagen sebagai Bahan Aktif Sediaan Farmasi,” *Maj. Farmasetika*, vol. 6, no. 3, p. 253, 2021, doi: 10.24198/mfarmasetika.v6i3.33621.

[20] R. Article, C. Floden, A, Combs, and R. Article, “基因的改变NIH Public Access,” *Bone*, vol. 23, no. 1, pp. 1–7, 2008, [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3624763/pdf/nihms412728.pdf.

[21] M. Milano, “Gene Prioritization Tools,” *Encycl. Bioinforma. Comput. Biol. ABC Bioinforma.*, vol. 1–3, pp. 907–914, Jan. 2019, doi: 10.1016/B978-0-12-809633-8.20406-8.

[22] T. Gene Ontology Consortium *et al.*, “Gene Ontology: tool for the unification of biology NIH Public Access Author Manuscript,” *Nat Genet*, vol. 25, no. 1, pp. 25–29, 2000, doi: 10.1038/75556.

[23] “Gene Ontology and GO Annotations.” https://www.ebi.ac.uk/QuickGO/.

[24] M. R. N. Alamsyah, S. J. Pamungkas, A. R. Meganingrum, and L. S. Nur’afifah, “Studi Anthophyta di Kota Magelang sebagai Sumber Pembelajaran Saintifik pada Perkuliahan Biosistematika Tumbuhan,” *Bioedusiana J. Pendidik. Biol.*, vol. 5, no. 2, pp. 160–175, 2020, doi: 10.37058/bioed.v5i2.2337.

[25] A. Aqilla, “Tingkatan Takson.” https://www.academia.edu/17610351/Tingkatan\_takson.

[26] V. F. Dr. Vladimir, “Hubungan Bidang Ilmu Data Mining, Man,” *Gastron. ecuatoriana y Tur. local.*, vol. 1, no. 69, pp. 5–24, 1967.

[27] S. E. Damayanti, “Analisis Dan Implementasi Framework Crisp-Dm (Cross Industry Standard Process for Data Mining) Untuk Clustering Perguruan Tinggi Swasta,” *J. STT Bandung*, 2006.

[28] Kotler, “Bab I ْ ُ با حض خ ِ ي,” *Galang Tanjung*, no. 2504, pp. 1–9, 2018.

[29] “Cross-industry standard process for data mining.” https://en.wikipedia.org/wiki/Cross-industry\_standard\_process\_for\_data\_mining.

[30] M. N. A. Ahmad, T, “Data preprocessing and feature selection for machine learning intrusion detection systems,” *January 2019*, 2019. https://www.researchgate.net/publication/331730342\_Data\_preprocessing\_and\_feature\_selection\_for\_machine\_learning\_intrusion\_detection\_systems.

[31] C. Li, “Preprocessing Methods and Pipelines of Data Mining: An Overview,” no. June, pp. 1–7, 2019, [Online]. Available: http://arxiv.org/abs/1906.08510.

[32] R. Agusalim, “Data Mining: Concepts and Techniques (3rd ed.),” *2018*. https://www.google.com/url?sa=i&url=https%3A%2F%2Fslidep*layer*.com%2Fslide%2F14608588%2F&psig=AOvVaw2gU22LO-EVsfNDn28SimX4&ust=1697687291390000&source=images&cd=vfe&opi=89978449&ved=0CBEQjRxqFwoTCMCMrZHY\_oEDFQAAAAAdAAAAABAJ.

[33] R. N. Ndung’u, “Data Preparation For Machine Learning Modelling,” *June 2022*, 2022. https://www.researchgate.net/publication/361095158\_Data\_Preparation\_For\_Machine\_Learning\_Modelling.

[34] S. Alasadi, “Review of Data Preprocessing Techniques in Data Mining,” *September 2017*, 2017. https://www.researchgate.net/publication/320161439\_Review\_of\_Data\_Preprocessing\_Techniques\_in\_Data\_Mining.

[35] H. Elabd, Y. Bromberg, A. Hoarfrost, T. Lenz, A. Franke, and M. Wendorff, “Amino acid encoding for deep learning applications,” *BMC Bioinformatics*, vol. 21, no. 1, pp. 1–14, 2020, doi: 10.1186/s12859-020-03546-x.

[36] A. Elnaggar *et al.*, “CodeTrans: Towards Cracking the Language of Silicone’s Code Through Self-Supervised Deep Learning and High Performance Computing,” vol. 14, no. 8, pp. 1–17, 2021, [Online]. Available: http://arxiv.org/abs/2104.02443.

[37] C. Raffel *et al.*, “Exploring the limits of transfer learning with a unified text-to-text *transformer*,” *J. Mach. Learn. Res.*, vol. 21, pp. 1–67, 2020.

[38] U. K. Devisetty, “Deep Learning for Genomics.” .

[39] D. Aggarwal and Y. Hasija, “A Review of Deep Learning Techniques for Protein Function Prediction,” no. section 2, pp. 1–6, 2022, [Online]. Available: http://arxiv.org/abs/2211.09705.

[40] B. D. Laraswati, “Perbedaan LSTM Network dengan Recurrent Neural Network,” 2022. https://blog.algorit.ma/lstm-network/.

[41] “5. Bleeding-Edge Techniques.” https://www.oreilly.com/library/view/hands-on-time-series/9781484259924/html/492113\_1\_En\_5\_Chapter.xhtml.

[42] “Forget gate.” https://www.oreilly.com/library/view/neural-network-projects/9781789138900/2941249a-fe96-475d-8109-79f7545a7805.xhtml.

[43] “*Input* gate.” .

[44] “Output gate.” https://www.oreilly.com/library/view/neural-network-projects/9781789138900/b45373a7-3bf3-4404-8c3f-3b9c26225196.xhtml.

[45] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, “Basic local alignment search tool,” *J. Mol. Biol.*, vol. 215, no. 3, pp. 403–410, 1990, doi: 10.1016/S0022-2836(05)80360-2.

[46] B. Buchfink, C. Xie, and D. H. Huson, “Fast and sensitive protein alignment using DIAMOND,” *Nat. Methods*, vol. 12, no. 1, pp. 59–60, 2014, doi: 10.1038/nmeth.3176.

[47] M. Kulmanov, F. Zhapa-Camacho, and R. Hoehndorf, “DeepGOWeb: Fast and accurate protein function prediction on the (Semantic) Web,” *Nucleic Acids Res.*, vol. 49, no. W1, pp. W140–W146, 2021, doi: 10.1093/nar/gkab373.

[48] C. Camacho *et al.*, “BLAST+: Architecture and applications,” *BMC Bioinformatics*, vol. 10, pp. 1–9, 2009, doi: 10.1186/1471-2105-10-421.

[49] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, 2006, doi: 10.1016/j.patrec.2005.10.010.

[50] V. S and J. R, “Text Mining: open Source Tokenization Tools – An Analysis,” *Adv. Comput. Intell. An Int. J.*, vol. 3, no. 1, pp. 37–47, 2016, doi: 10.5121/acii.2016.3104.

[51] DINA PUTRI AULIA, “KLASIFIKASI GEN ESENSIAL PADA DROSOPHILA MELANOGASTER BERDASARKAN PROTEIN SEQUENCE MENGGUNAKAN METODE LONG SHORT-TERM MEMORY (LSTM),” no. 0, pp. 1–23, 2016.

[52] Z. M. Chua, A. Rajesh, S. Sinha, and P. D. Adams, “PROTGOAT : Improved automated protein function predictions using Protein Language Models,” no. Cafa 5, pp. 1–15, 2024.

[53] G. A. Sandag and S. W. Taju, “Bioinformatics Tools for Data Processing and Prediction of Protein Function,” *CogITo Smart J.*, vol. 4, no. 2, pp. 305–315, 2019, doi: 10.31154/cogito.v4i2.137.305-315.

[54] R. Bonetta and G. Valentino, “Machine learning techniques for protein function prediction,” *Proteins Struct. Funct. Bioinforma.*, vol. 88, no. 3, pp. 397–413, 2020, doi: 10.1002/prot.25832.

[55] V. Gligorijević *et al.*, “Structure-based protein function prediction using graph convolutional networks,” *Nat. Commun.*, vol. 12, no. 1, 2021, doi: 10.1038/s41467-021-23303-9.

# LAMPIRAN

**Lampiran 1: Pembagian Tugas TASI-2324-101**

Dalam pelaksanaan penelitian Tugas Akhir kelompok TASI-2324-101 dengan judul penelitian "*Protein Function Prediction using Deep Learning*" kelompok peneliti terdiri dari 3 orang mahasiswa tingkat akhir yaitu:

1. 12S20030 - Lamsihar Siahaan
2. 12S20046 - Patricia Dian Margaretha S
3. 12S20047 - Ruth Christine Manurung

Untuk menyelesaikan penelitian ini, kelompok TASI-2324-101 melakukan tahapan-tahapan yaitu, pendahuluan, landasan teori, metode penelitian, desain, implementasi, dan hasil dan pembahasan. Sebelum dilakukan pembagian tugas, kelompok TASI-2324-101 mendiskusikan terlebih dahulu terkait pemilihan topik bersama pembimbing dan mempelajari studi literatur mengenai topik yang telah ditetapkan. Tahapan tersebut dilaksanakan dengan dilakukannya pembagian kedalam beberapa tugas dengan tujuan pengerjaan penelitian dapat dilakukan secara kolaboratif dengan kerjasama antar anggota kelompok. Berikut pembagian kerja kelompok TASI-2324-101.

**Lamsihar Siahaan** bertanggungjawab melakukan eksplorasi terhadap *paper* atau jurnal mengenai prediksi fungsi protein. Pada bab 1, Lamsihar menjabarkan latar belakang pada paragraf 1 serta mencari pengertian dan fungsi dari protein, merumuskan pertanyaan penelitian, melakukan eksplorasi terhadap penelitian terdahulu mengenai prediksi fungsi protein dan menuliskannya pada landasan teori dan jurnal terkait pada masing-masing sub-bab. Pada bab 2, Lamsihar menjelaskan tentang *Protein Language Model* dan penelitian terdahulu. Pada bab 3, Lamsihar menjabarkan tentang analisis statistik, analisis *data preprocessing,* analisis ProtBERT, analisis nilai *hyperparameter,* dan analisis eksperimen 3 dan 4. Pada bab 4, Lamsihar menggambarkan dan menjelaskan secara detail tentang desain dari *embedding* ProtBERTdan desain dari Eksperimen 3 dan 4. Pada bab 5, Lamsihar mengerjakan implementasi *preprocessing* dan implementasi terhadap eksperimen 3 dan 4. Pada bab 6, Lamsihar mengerjakan hasil prediksi terhadap eksperimen 3 dan 4.

**Patricia Dian Margaretha S** bertanggungjawab melakukan eksplorasi terhadap *paper* atau jurnal mengenai prediksi fungsi protein. Pada Bab 1, Patricia Silaban menjabarkan latar belakang pada paragraf 2 dan 3 untuk mencari metode konvensional yang digunakan untuk memprediksi sekuensing protein, menuliskan ruang lingkup mengenai *dataset* yang digunakan pada penelitian. Pada bab 2, Patricia menjelaskan tentang *data mining* baik itu CRISP-DM dan *data preprocessing.* Selain itu, pada bab 2, Patricia menjelaskan terkait *deep learning,* baik itu RNN dan LSTM. Pada bab 3, Patricia menjabarkan terkait dengan analisis persiapan data, analisis *T5-Embedding,* analisis model, analisis metriks evaluasi, dan analisis eksperimen 2. Pada bab 4, Patricia menggambarkan dan menjelaskan secara detail terkait dengan desain penelitian, desain dari model LSTM, desain dari *T5-Embedding*, dan desain dari eksperimen 2. Pada bab 5, Patricia mengerjakan eksperimen 2. Pada bab 6, Patricia mengerjakan hasil prediksi terhadap eksperimen 2.

**Ruth Christine Manurung** bertanggung jawab dalam melakukan eksplorasi terhadap *paper* atau jurnal mengenai prediksi fungsi protein. Pada proposal, Ruth Christine Manurung menjabarkan abstrak dan kata kunci untuk proposal prediksi fungsi protein. Pada bab 1 bertanggung jawab dalam menuliskan penelitian yang dilakukan dalam organisasi CAFA, menuliskan tujuan penelitian, dan pertanyaan penelitian. Pada bab 2, Ruth menjabarkan pengertian protein, struktur protein, fungsi protein, *Gene Ontology,* taksonomi, *embedding* dari *Tensorflow.Keras*, dan pencarian sekuens. Pada bab 3, Ruth menjabarkan tentang analisis deskripsi *dataset* yang meliputi sekuensprotein, fungsi protein, taksonomi dari sekuens*,* analisis *embedding* dari *Tensorflow.Keras*, analisis pencarian sekuens (*diamond search*), dan analisis eksperimen 1*.* Pada bab 4, Ruth menggambarkan dan menjelaskan secara detail tentang desain dari *embedding* dari *Tensorflow.Keras* dan desain dari eksperimen 1*.* Pada bab 5, Ruth mengerjakan eksperimen 1. Pada bab 6, Ruth mengerjakan hasil prediksi dari eksperimen.

**Lampiran 2: Analisis Frekuensi GO\_Term**

Pada lampiran ini berisikan analisis frekuensi pada kelas *Terms* GO, dimana terdapat 20 *Terms* GO yang paling sering digunakan pada data CAFA5.

**Lampiran 1. Analisis Frekuensi GO\_Term**

|  |  |
| --- | --- |
| **GO term** | **Definisi** |
| GO:0005575 - *protein kinase activity* | Protein kinase adalah enzim yang mengkatalisis fosforilasi protein. Fosforilasi adalah proses penambahan gugus fosfat ke protein. Gugus fosfat ini dapat mengubah struktur dan fungsi protein. |
| GO:0008150 - *transferase activity* | Transferase adalah enzim yang mengkatalisis transfer gugus dari satu molekul ke molekul lain. Gugus yang ditransfer dapat berupa gugus fosfat, gula, atau amino. |
| GO:0110165 - *cytoplasmic localization* | Protein yang memiliki GO term ini ditemukan di sitoplasma, yaitu bagian sel yang terletak di luar nukleus. |
| GO:0003674 - *catalytic activity* | Protein yang memiliki GO term ini memiliki aktivitas katalitik, yaitu kemampuan untuk mempercepat reaksi kimia. |
| GO:0005622 - *protein binding* | Protein yang memiliki GO term ini dapat berikatan dengan protein lain. Ikatan ini dapat bersifat sementara atau permanen. |
| GO:0009987 - *cellular localization* | Protein yang memiliki GO term ini ditemukan di bagian tertentu dari sel. Bagian sel ini dapat berupa sitoplasma, nukleus, membran sel, atau organel lainnya. |
| GO:0043226 - *receptor activity* | Protein yang memiliki GO term ini berfungsi sebagai reseptor, yaitu molekul yang dapat mengikat molekul lain dan memulai respons biologis. |
| GO:0043229 - *ion binding* | Protein yang memiliki GO term ini dapat berikatan dengan ion, seperti ion kalsium, ion magnesium, atau ion natrium. |
| GO:0005488 - *metabolic process* | Protein yang memiliki GO term ini terlibat dalam proses metabolisme, yaitu proses pemecahan molekul makanan untuk menghasilkan energi dan molekul lain yang dibutuhkan oleh sel. |
| GO:0043227 - ligand binding | Protein yang memiliki GO term ini dapat berikatan dengan ligan, yaitu molekul yang dapat mengikat protein dan memulai respons biologis. |
| GO:0005737 - *signal transduction* | Protein yang memiliki GO term ini terlibat dalam transduksi sinyal, yaitu proses pengiriman sinyal dari satu sel ke sel lain. |
| GO:0043231 - *nuclear localization* | Protein yang memiliki GO term ini ditemukan di nukleus, yaitu bagian sel yang mengandung DNA. |
| GO:0005515 - *structural molecule activity* | Protein yang memiliki GO term ini berfungsi sebagai struktur sel, seperti protein yang membentuk membran sel atau protein yang membentuk rangka sel. |