

DAFTAR RIWAYAT HIDUP

Nama : Annisa Arrumaisha Siregar
NIM : 061940351944
Tempat, Tanggal Lahir : Tanjung Pandan, 19 Mei 2002
Alamat : Jalan Sultan Mahmud Badaruddin 2 Perumahan
Villa Palem Cemerlang Blok H-7 RT 37 RW 11,
Kelurahan Karya Baru, Kecamatan Alang-Alang
Lebar, Kode Pos 30151, Palembang
No. Telepon : 087848058986

Riwayat Pendidikan Formal

Pendidikan	Nama Sekolah	Tamat Tahun
SD	SD Negeri 42 Palembang	2013
SMP	SMPIT Izzuddin Palembang	2016
SMA	SMA Negeri 3 Palembang	2019

Riwayat Pendidikan Non Formal

No	Jenis Pendidikan Non Formal	Tahun
1	Global <i>English Language Center</i>	2020
2	Bimbingan Belajar Nurul Fikri	2017

Pengalaman Penelitian

No	Nama Penelitian	Tahun
-	-	-

Penghargaan

No	Penghargaan	Tahun
-	-	-

Pengalaman Organisasi

No	Pengalaman Organisasi	Tahun
1	Anggota <i>Externship</i> Zarp Labs R&D Pvt. Ltd.	2021
2	Anggota Departemen PSDM BEM KM Politeknik Negeri Sriwijaya	2021
3	Anggota Divisi <i>Human Resource</i> ZeroOne.it	2021
4	Anggota Divisi KRPAI Tim Robotik Politeknik Negeri Sriwijaya	2019

Semua data yang saya isikan dan tercantum dalam daftar riwayat hidup ini adalah benar dan dapat dipertanggung jawabkan.

Palembang, Agustus 2023

Annisa Arrumaisha Siregar



KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA

Jalan Srijaya Negara, Palembang 30139 Telp. 0711-353414

Laman: <http://polsri.ac.id>, Pos El : info@polsri.ac.id



KESEPAKATAN BIMBINGAN TUGAS AKHIR (TA)

Kami yang bertanda tangan dibawah ini,

Pihak Pertama

Nama : Annisa Arrumaisha Siregar
NIM : 061940351944
Jurusan : Teknik Elektro
Program Studi : Sarjana Terapan Teknik Telekomunikasi

Pihak Kedua

Nama : Sopian Soim, S.T., M.T.
NIP : 197103142001121001
Jurusan : Teknik Elektro
Program Studi : Sarjana Terapan Teknik Telekomunikasi

Pada hari ini Rabu..... tanggal 22 Maret 2023 telah sepakat untuk melakukan konsultasi bimbingan Tugas Akhir (TA).

Konsultasi bimbingan sekurang- kurangnya 1 (satu) kali dalam satu minggu. Pelaksanaan bimbingan pada setiap hari Jumat..... pukul 13.00 tempat di Politeknik Negeri Sriwijaya.

Demikianlah kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Tugas Akhir.

Pihak Pertama

(Annisa Arrumaisha Siregar)
NIM. 061940351944

Palembang, 2023
Pihak Kedua

(Sopian Soim, S.T., M.T.)
NIP. 197103142001121001

Mengetahui
Ketua Jurusan Teknik Elektro

(Ir. Iskandar Lutfi, M.T.)
NIP. 196501291991031002



KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA

Jalan Srijaya Negara, Palembang 30139 Telp. 0711-353414
Laman: <http://polsri.ac.id>, Pos El : info@polsri.ac.id



KESEPAKATAN BIMBINGAN TUGAS AKHIR (TA)

Kami yang bertanda tangan dibawah ini,

Pihak Pertama

Nama : Annisa Arrumaisha Siregar
NIM : 061940351944
Jurusan : Teknik Elektro
Program Studi : Sarjana Terapan Teknik Telekomunikasi

Pihak Kedua

Nama : Mohammad Fadhli, S.Pd., M.T.
NIP : 199004032018031001
Jurusan : Teknik Elektro
Program Studi : Sarjana Terapan Teknik Telekomunikasi

Pada hari ini Rabu.. tanggal 22 Maret.. 2023 telah sepakat untuk melakukan konsultasi bimbingan Tugas Akhir (TA).

Konsultasi bimbingan sekurang- kurangnya 1 (satu) kali dalam satu minggu. Pelaksanaan bimbingan pada setiap hari Senin..... pukul 16.00 tempat di Politeknik Negeri Sriwijaya.

Demikianlah kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Tugas Akhir.

Pihak Pertama

(Annisa Arrumaisha Siregar)
NIM. 061940351944

Palembang, 2023

Pihak Kedua

(Mohammad Fadhli, S.Pd., M.T.)
NIP. 199004032018031001

Mengetahui
Ketua Jurusan Teknik Elektro

(Ir. Iskandar Lutfi, M.T.)
NIP. 196501291991031002



**KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI**

POLITEKNIK NEGERI SRIWIJAYA

Jalan Sriwijaya Negara, Palembang 30139 Telp. 0711-353414

Laman: <http://polsri.ac.id>, Pos El : info@polsri.ac.id

LEMBAR BIMBINGAN TUGAS AKHIR

Lembar : 1

Nama : Annisa Arrumaisha Siregar
 NIM : 061940351944
 Jurusan/Program Studi : Teknik Elektro/Sarjana Terapan Teknik Telekomunikasi
 Judul Tugas Akhir : Penerapan Metode *Back Propagation* Untuk Peningkatan Performa
 Algoritma *Neural Network* Pada Sistem Deteksi *Malware*
 Pembimbing I : Sopian Soim, S.T., M.T.

No.	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
1.	$\frac{03}{04}$ 2023	Konsultasi Perancangan Sistem	
2.	$\frac{19}{04}$ 2023	Revisi Latar Belakang	
3.	$\frac{23}{06}$ 2023	Revisi Rumusan Masalah	
4.	$\frac{26}{06}$ 2023	Bimbingan Bab I	
5.	$\frac{28}{06}$ 2023	Bimbingan Bab II	
6.	$\frac{04}{07}$ 2023	Konsultasi Hasil Eksperimen I Menggunakan Dataset I	
7.	$\frac{07}{07}$ 2023	Revisi Bab II dan Konsultasi Hasil Eksperimen II Menggunakan Dataset I	
8.	$\frac{09}{07}$ 2023	Konsultasi Hasil Eksperimen III dan IV Menggunakan Dataset II	

No.	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
9.	$\frac{16}{07}$ 2023	Bimbingan Bab III dan konsultasi Perancangan Website Deteksi Malware	
10.	$\frac{23}{07}$ 2023	Revisi Diagram Alir Metodologi Penelitian dan Proses pemodelan Back Propagation NN	
11.	$\frac{30}{07}$ 2023	Bimbingan Jurnal	
12.	$\frac{09}{08}$ 2023	Bimbingan Bab IV	
13.	$\frac{07}{08}$ 2023	Revisi Bab IV	
14.	$\frac{09}{08}$ 2023	Bimbingan Bab V dan Uji Website Deteksi Malware	
15.	$\frac{10}{08}$ 2023	Revisi Minor Bab I - V	
16.	$\frac{11}{08}$ 2023	AEC Final Ujian	

Palembang, Agustus 2023
KPS Sarjana Terapan Teknik Telekomunikasi


(Lindawati, S.T., M.T.I)
NIP. 197105282006042001

Catatan:

*) melingkari angka yang sesuai.

Ketua Jurusan/Ketua Program Studi harus memeriksa jumlah pelaksanaan bimbingan sesuai yang dipersyaratkan dalam Pedoman Tugas Akhir sebelum menandatangani lembar bimbingan ini.

Lembar pembimbingan Tugas Akhir ini harus dilampirkan dalam Tugas Akhir.



**KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI**

POLITEKNIK NEGERI SRIWIJAYA

Jalan Srijaya Negara, Palembang 30139 Telp. 0711-353414

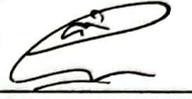
Laman: <http://polsri.ac.id>, Pos El : info@polsri.ac.id

LEMBAR BIMBINGAN TUGAS AKHIR

Lembar : 1

Nama : Annisa Arrumaisha Siregar
 NIM : 061940351944
 Jurusan/Program Studi : Teknik Elektro/Sarjana Terapan Teknik Telekomunikasi
 Judul Tugas Akhir : Penerapan Metode *Back Propagation* Untuk Peningkatan Performa
 Algoritma *Neural Network* Pada Sistem Deteksi *Malware*
 Pembimbing II : Mohammad Fadhli, S.Pd., M.T

No.	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
1.	3/4 2023	Konsultasi Perancangan Sistem	
2.	19/4 2023	Konsultasi Hasil Eksperimen I Menggunakan Dataset I	
3.	3/5 2023	Bimbingan Bab I	
4.	15/5 2023	Revisi Bab I	
5.	30/5 2023	Konsultasi Hasil Eksperimen II Menggunakan Dataset I	
6.	7/6 2023	Bimbingan Bab II	
7.	12/6 2023	Revisi Bab II	
8.	28/6 2023	Konsultasi Hasil Eksperimen III dan IV Menggunakan Dataset II	

No.	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
9.	9/7 2023	Bimbingan Bab III dan konsultasi Perancangan Website Deteksi Malware	
10.	16/7 2023	Revisi Bab III	
11.	28/7 2023	Bimbingan Jurnal	
12.	30/7 2023	Revisi Jurnal, Bimbingan Bab IV	
13.	4/8 2023	Bimbingan Bab V, Uji Website Deteksi Malware	
14.	7/8 2023	Revisi Minor Bab I - V	
15.	9/8 2023	ACC	
16.			

Palembang, Agustus 2023
KPS Sarjana Terapan Teknik Telekomunikasi


(Lindawati, S.T., M.T.I)
NIP. 197105282006042001

Catatan:

*) melingkari angka yang sesuai.

Ketua Jurusan/Ketua Program Studi harus memeriksa jumlah pelaksanaan bimbingan sesuai yang dipersyaratkan dalam Pedoman Tugas Akhir sebelum menandatangani lembar bimbingan ini.

Lembar pembimbingan Tugas Akhir ini harus dilampirkan dalam Tugas Akhir.



	<p style="text-align: center;">KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI POLITEKNIK NEGERI SRIWIJAYA Jalan Srijaya Negara Bukit Besar - Palembang 30139 Telepon (0711) 353414 Laman : http://polsri.ac.id, POS El: info@polsri.ac.id</p>	
REKOMENDASI UJIAN TUGAS AKHIR		

Pembimbing Tugas Akhir memberikan rekomendasi kepada,

Nama : Annisa Arrumaisha Siregar
NIM : 061940351944
Jurusan/Program Studi : Teknik Elektro / Sarjana Terapan Teknik Telekomunikasi
Judul Laporan Akhir : Penerapan Metode *Back Propagation* Untuk Peningkatan Performa Algoritma *Neural Network* Pada Sistem Deteksi *Malware*

Mahasiswa tersebut telah memenuhi persyaratan dan dapat mengikuti Ujian Tugas Akhir (TA) pada Tahun Akademik 2022/2023.

Pembimbing I



Sopian Spim, S.T., M.T
NIP.197103142001121001

Palembang, Agustus 2023
Pembimbing II



Mohammad Fadhli, S.Pd., M.T
NIP.199004032018031001



**KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI**

POLITEKNIK NEGERI SRIWIJAYA

Jalan Srijaya Negara, Palembang 30139 Telp. 0711-353414
Laman: <http://polsri.ac.id>, Pos El : info@polsri.ac.id

PELAKSANAAN REVISI TUGAS AKHIR

Mahasiswa berikut,

Nama : Annisa Arrumaisha Siregar
NIM : 061940351944
Jurusan/Program Studi : Teknik Elektro/Sarjana Terapan Teknik Telekomunikasi
Judul Tugas Akhir : Penerapan Metode *Back Propagation* Untuk Peningkatan Performa Algoritma *Neural Network* Pada Sistem Deteksi *Malware*

Telah melaksanakan revisi terhadap Tugas Akhir yang diujikan pada hari Rabu tanggal 16 bulan Agustus tahun 2023. Pelaksanaan revisi terhadap Tugas Akhir tersebut telah disetujui oleh Dosen Penguji yang memberikan revisi:

No.	Komentar	Nama Dosen Penguji*)	Tanggal	Tanda Tangan
1.	Sudah revisi	Ir. Jon Endri, M.T NIP 196201151993031001	21/08 2023	
2.	SUDAH DIREVISI	Hj. Adewasti, S.T., M.Kom NIP 197201142001122001	21/08 2023	
3.	Acc	Sopian Soim, S.T., M.T NIP 197103142001121001	28/08 2023	
4.	Acc	RA. Halimatussa'diyah, S.T., M.Kom NIP 197406022005012002	28/08 2023	

Palembang, Agustus 2023
Ketua Penguji **),

(Ir. Jon Endri, M.T)
NIP 196201151993031001

Catatan:

*) Dosen penguji yang memberikan revisi saat ujian Tugas Akhir.

**) Dosen penguji yang ditugaskan sebagai Ketua Penguji saat ujian Tugas Akhir.
Lembaran pelaksanaan revisi ini harus dilampirkan dalam Tugas Akhir.



LETTER OF ACCEPTANCE

Nomor: RO-FST/U.201005/IJAIMD-PRDT/016/2023

Editor Kepala *Indonesian Journal of Artificial Intelligence and Data Mining* (IJAIMD) *Puzzle Research Data Technology* (Predatech) Fakultas Sains dan Teknologi (FST) UIN Sultan Syarif Kasim Riau, menerangkan bahwa paper dengan:

Judul : Optimizing Malware Detection using Back Propagation
Neural Network and Hyperparameter Tuning
Penulis : Annisa Arrumaisha Siregar, Sopian Soim,
Mohammad Fadhli
Afiliasi : Politeknik Negeri Sriwijaya
Korespondensi : Annisa Arrumaisha Siregar

Telah Diterima pada *Indonesian Journal of Artificial Intelligence and Data Mining* (IJAIMD) Predatech UIN Sultan Syarif Kasim Riau, dan akan diterbitkan pada Volume 6 Nomor 2 Tahun 2023.

Demikian surat keterangan ini dibuat dengan sebenarnya untuk dapat dipergunakan sebagaimana mestinya.

Pekanbaru, 28 Juli 2023

Editor Kepala

Mustakim, S.T., M.Kom.

NIDN. 2002068801

Optimizing Malware Detection using Back Propagation Neural Network and Hyperparameter Tuning

¹ Annisa Arrumaisha Siregar, ² Sopian Soim, S.T., M.T., ³ Mohammad Fadhli, S.Pd., M.T

^{1,2,3}Departement of Electrical Engineering, Telecommunication Engineering Study Program,
State Polytechnic of Sriwijaya

Email: ¹ annisaarrumaisha19@gmail.com, ²sopiansoim@gmail.com, ³ mohammad.fadhli@polsri.ac.id

Article Info

Article history:

Received Jul 17th, 2023

Revised Jul 25th, 2023

Accepted Aug 5th, 2023

Keyword:

Malware Detection

Back Propagation Neural Network

Hyperparameter Tuning

Dropout Regularization

GridSearchCV

Cyber threats

ABSTRACT

The escalating growth of the internet has led to an increase in cyber threats, particularly malware, posing significant risks to computer systems and networks. This research addresses the challenge of developing sophisticated malware detection systems by optimizing the Back Propagation Neural Network (BPNN) with hyperparameter tuning. The specific focus is on fine-tuning essential hyperparameters, including dropout rate, number of neurons in hidden layers, and number of hidden layers, to enhance the accuracy of malware detection. A Back Propagation Neural Network (BPNN) with dropout regularization is trained on an extensive dataset as part of the research design. Hyperparameter optimization is conducted using GridSearchCV, with experiments varying learning rates and epochs. The best configuration achieves outstanding results, with 98% accuracy, precision, recall, and F1-score. The proposed approach presents an efficient and reliable solution to bolster cybersecurity systems against malware threats.

Copyright © 2023 Puzzle Research Data Technology

Corresponding Author:

Annisa Arrumaisha Siregar,

Departement of Electrical Engineering, Telecommunication Engineering Study Program,

State Polytechnic of Sriwijaya

Jl. Srijaya Negara, Bukit Besar, Kecamatan Ilir Barat I, Kota Palembang, Sumatera Selatan.

Email: annisaarrumaisha19@gmail.com

DOI: <http://dx.doi.org/10.24014/ijaidm.v6i2.24864>

1. INTRODUCTION

The rapid expansion of the internet has become an essential aspect of daily life worldwide. As of April 2023, there were over five billion internet users, constituting approximately 64.6% of the global population [1]. This interconnected web of computers, networks, and interconnected devices has been made possible through continuous technological advancements in computer systems, networks, and mobile devices, leading to a significant surge in internet usage. However, this exponential growth in internet users has also escalated the vulnerability of computer systems and networks to cyberattacks [2]. Among the various cyber threats, one of the most significant risks to the security of computer systems and networks is malware, which stands for malicious software. Malicious programs are specifically designed to infiltrate, cause harm, or steal sensitive information from the targeted system. In the realm of cybersecurity, the detection of malware plays a crucial role in preventing potential data breaches and thwarting cyberattacks. Effectively identifying and combating malware is of utmost importance in safeguarding the integrity and confidentiality of sensitive data.

To address these security challenges effectively, it is crucial to prioritize the development of a sophisticated malware detection system, leveraging the capabilities of machine learning (ML) and deep learning (DL) [3]. Cybersecurity experts firmly believe that ML/DL-based anti-malware software will significantly improve the detection of new malware types and enhance existing scanning engines. Machine learning and deep learning offer distinct advantages in malware detection and analysis tools. Traditional machine learning techniques rely on feature engineering and feature representation techniques, which demand domain knowledge [4]. Currently, deep learning, an enhanced neural network model, has outperformed traditional machine learning in various tasks such as speech recognition, pattern identification, image

classification, and many other classification tasks. Deep learning automatically extracts features at each layer, as it has the capability of learning high-level feature representations through layered training from the lowest to the highest levels [5].

In previous research, several studies on malware detection have been conducted using various techniques. For instance, Saxe and Berlin [6] proposed a malware classification system constructed using a neural network with two hidden layers. These hidden layers were composed of 1024 Parametric Rectified Linear Units (PReLU), while the output layer utilized a sigmoid neuron to classify instances as either malware or benign. They have achieved a 95% detection rate with a 0.1% false positive rate when running an experiment on a 400,000 samples dataset. Huda et al. [7] proposed multilayer perceptron artificial neural network trained with backpropagation algorithm to determine an Android application is malware or benign. The experimental results show that the proposed method is effective, with a relatively high accuracy in recognizing existing malware samples. Pan, et al. [8] proposed a malware classification approach based on dynamic analysis, using dynamic analysis to obtain behavior profiles which are then used to abstract malware features and input the Back Propagation (BP) Neural Network model. The experimental results show the technique is effective and accurate. The methodology has an average accuracy of up to 86%, with non-malware categories predicting accuracy up to 99%. This suggests that the methodology can be easily extended for malware detection.

Makandar and Patrot [9] explored the classification of malware samples by converting them into grayscale images and using texture features for analysis. They extracted a total of 512 feature vectors using Gabor Wavelet Transform (GWT) and the Generic Fourier Descriptor (GIST) to observe the malware behavior. From these feature vectors, they selected 320 two-dimensional features and applied an Artificial Neural Network (ANN) for classification. The choice of using a feed-forward artificial neural network was driven by its capability to identify hidden patterns within complex data, such as images, sounds, and signals, while also providing implementation flexibility. For the training process, the authors utilized the backpropagation algorithm. Despite their dedicated efforts, they encountered challenges in effectively handling high-dimensional feature vectors, which resulted in a suboptimal accuracy of 96.35% for the malware classification task. Babak et al. [10] developed a neural network model for classifying PE files, employing the Gradient Descent algorithm with a decayed learning rate to update the weights during training. The model utilized the softmax activation function for the output neurons. With a dataset comprising 4,000 samples (3,000 malicious PE files and 1,000 benign), the implemented neural network achieved impressive performance metrics. The average accuracy attained was 97.8%, with a precision of 97.6% and a recall of 96.6%. These findings demonstrate the efficacy of their approach in accurately classifying PE files into malicious and benign categories. The utilization of the Gradient Descent algorithm and softmax activation function showcases the model's ability to handle complex data and effectively distinguish between potentially harmful and benign files.

Previous research has demonstrated the promising efficacy of Back Propagation Neural Networks (BPNNs) in malware detection, owing to their ability to learn intricate patterns and relationships within data. Nonetheless, the performance of BPNNs is heavily dependent on hyperparameters, which dictate the network's architecture and learning process. In this study, our objective is to optimize the malware detection process by fine-tuning key hyperparameters of the Back Propagation Neural Network through Hyperparameter Tuning. Our focus centers on three critical hyperparameters: the dropout rate, the number of hidden units in each layer, and the number of hidden layers. Leveraging dropout regularization helps prevent overfitting and enhances the model's generalization capability. Systematically evaluating various combinations of these hyperparameters, we endeavor to identify the optimal settings that will yield improved accuracy in malware detection.

To achieve this optimization, we propose a novel approach by utilizing a Back Propagation Neural Network (BPNN) with dropout regularization. The inherent capability of BPNN architecture in handling complex data makes it a suitable candidate for classifying malware. While the existing literature has explored the use of Artificial Neural Networks (ANNs) for malware classification, the specific application of BPNNs with dropout regularization to optimize malware detection using Back Propagation Neural Networks and Hyperparameter Tuning remains relatively underexplored. This study seeks to make a substantial contribution to the advancement of cybersecurity systems against malicious threats, ultimately culminating in an optimized and efficient malware detection solution.

2. RESEARCH METHOD

2.1. Research Design

The research design for this study is structured to optimize malware detection using Back Propagation Neural Network (BPNN) in conjunction with Hyperparameter Tuning. The process follows a systematic and well-defined flow aimed at enhancing the accuracy and efficiency of malware detection. The first step involves acquiring the dataset containing malware detection data. For this purpose, publicly available data was collected, specifically utilizing the dataset for the classification of malware with PE Headers sourced from github.com [11]. This dataset serves as the foundation for classifying PE files into two categories: malware and benign

(legitimate). The dataset used in the study is extensive, comprising over 138,000 instances. Among these, 96,724 samples were identified as malware, while 41,323 samples were labeled as benign (legitimate).

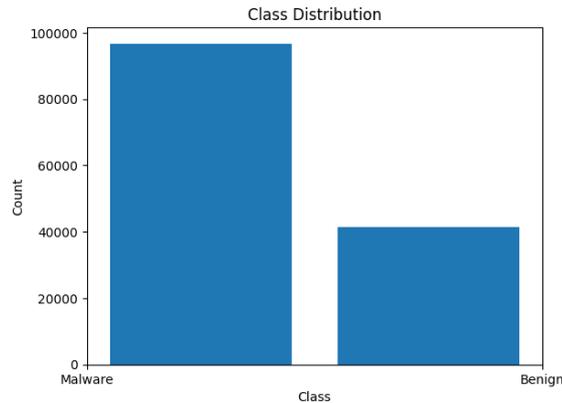


Figure 1. Class Distribution

Once the dataset is obtained, preprocessing techniques are applied to cleanse the data and prepare it for further analysis. This involves handling missing values and normalizing the dataset. The normalization method employed is MinMaxScaler. This method transforms each value in the dataset to a range between 0 and 1, where 0 represents the minimum value in the dataset, and 1 represents the maximum value. During the normalization process using MinMaxScaler, each value in the dataset is subtracted by the minimum value in the dataset, and the result is divided by the difference between the maximum and minimum values [12]. As a result, the range of values for all attributes in the dataset becomes uniform, allowing attributes with different value ranges to be treated fairly during analysis and modeling [12].

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (1)$$

The preprocessed dataset is divided into two subsets: the training set and the testing set. The training set is used for model training, allowing the Back Propagation Neural Network (BPNN) to learn from the data. The testing set, on the other hand, is used to evaluate the model's performance on new and independent data. The data is split in an 80-20 ratio, with 80% allocated to the training set and 20% to the testing set. This ensures that the model learns from a significant portion of the data and can effectively identify intricate patterns and relationships. The testing set provides an unbiased assessment of the model's ability to generalize to new and unseen data, reflecting its real-world performance.

The core of the research design lies in the architecture of the BPNN model and the utilization of Dropout Regularization. The model consists of multiple layers, each comprising a certain number of neurons, with different activation functions. The input layer of the neural network is determined by the number of features present in the preprocessed dataset. In this study, the number of input neurons corresponds to the number of features, facilitating the extraction of relevant information from the data. Subsequently, the model includes multiple hidden layers, each containing a specified number of hidden units. These hidden layers play a vital role in learning intricate patterns and relationships within the data, enabling the model to detect and classify malware instances effectively. The number of hidden layers and hidden units are hyperparameters that undergo tuning during the optimization process.

To mitigate the risk of overfitting, dropout regularization is applied after each hidden layer. Dropout refers to the process of dropping out units (both hidden and visible) in an artificial neural network. Dropping out units involves temporarily removing these units from the neural network along with all the connections that lead to and from the respective units [13]. The dropout rate, which is another hyperparameter, determines the proportion of neurons that are randomly deactivated during each training iteration. By using this regularization technique, the model becomes more robust and avoids excessive reliance on specific neurons, thereby enhancing its generalization capability.

Furthermore, the activation function used in the input layer and hidden layers is Rectified Linear Unit (ReLU), which allows the model to handle non-linear relationships in the data effectively. The ReLU activation function, proposed by [14], is founded on robust biological and mathematical principles. Initially introduced in 2011, it demonstrated significant advancements in training deep neural networks. The function operates by setting values below 0 to 0, effectively using $f(x) = \max(0, x)$. In simpler terms, it outputs 0 when $x < 0$ and a linear function when $x \geq 0$.

$$f(x) = \max(0, x) = \{xi, \text{if } xi \geq 0; 0, \text{if } xi < 0 \quad (2)$$

The output layer, which is responsible for handling the binary classification problem (distinguishing between malware and benign samples), uses the sigmoid activation function. The sigmoid function, also known as the logistic function, is a non-linear activation function that maps the output of a neuron to a range between 0 and 1 [15] [16] [17]. This mapping allows the sigmoid activation function to interpret the output as a probability, indicating the likelihood of a particular class being present in the input data. Mathematically, the sigmoid activation function can be expressed as follows:

$$f(x) = \frac{1}{1+e^{-x}} \quad (3)$$

where x is the input to the activation function, and $f(x)$ is the output in the range $[0, 1]$. When the input x is positive, the output $f(x)$ tends to 1, and when x is negative, the output $f(x)$ tends to 0. This property makes the sigmoid function suitable for binary classification problems, where the goal is to classify data into one of two distinct classes.

The advantages of using the sigmoid activation function in the output layer for binary classification tasks are twofold. Firstly, the sigmoid function offers a natural interpretation of the output as a probability score, facilitating the establishment of a decision threshold, typically set at 0.5, for making class predictions. In our study, we implemented this by classifying input data as benign if the output probability is greater than 0.5, and as malware if the probability is less than or equal to 0.5. This thresholding mechanism simplifies the decision-making process, promoting clarity and logical reasoning in binary classification tasks.

Secondly, the sigmoid activation function's ability to compress the output in the range $[0, 1]$ helps mitigate the issue of vanishing gradients during backpropagation [17]. Vanishing gradients occur when the gradient becomes too small as it propagates backward through the layers during training. This can hinder the learning process, especially in deep networks. The sigmoid function's derivative ranges from 0 to 0.25, which helps avoid extremely small gradients compared to other activation functions with larger derivative ranges.

In order to optimize the performance of the neural network model, we employ the Adam optimizer, an adaptive learning rate algorithm widely used in training deep neural networks [18]. The Adam optimizer combines the advantages of both the Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSprop) [19]. By dynamically adjusting the learning rates for each parameter during training, Adam is particularly effective in handling sparse gradients and non-stationary objective functions [20] [21], leading to faster convergence and better generalization on unseen data.

The key advantage of the Adam optimizer lies in its ability to maintain two moving average estimates: the first moment (mean) of the gradients and the second moment (uncentered variance) of the gradients [19]. These estimates are calculated using exponential moving averages and bias correction, effectively mitigating the impact of noisy gradients and improving optimization performance [19]. Furthermore, Adam incorporates bias correction during the initial optimization steps, especially beneficial when dealing with parameters having large gradients. This correction ensures a smoother optimization process, preventing overshooting or oscillation during parameter updates.

The Adam optimizer updates the model's parameters based on the gradients of the loss function with respect to those parameters. It combines the concepts of momentum and RMSprop to achieve adaptive learning rates for each parameter during training. The formulas for updating the parameters using the Adam optimizer are as follows [20]:

1. Moving average of the first moment (mean) of the gradients (m):

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \quad (4)$$

2. Moving average of the second moment (uncentered variance) of the gradients (v):

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \quad (5)$$

3. Bias-corrected first moment estimate:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (6)$$

4. Bias-corrected second moment estimate:

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (7)$$

5. Update rule for each parameter (θ):

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \cdot \hat{m}_t \quad (8)$$

where:

m_t represents the moving average of the gradients' first moment at time step t .

v_t represents the moving average of the gradients' second moment at time step t .

g_t represents the gradients of the loss function with respect to the parameters at time step t .

β_1 and β_2 are the exponential decay rates for the first and second moment estimates, respectively (typically set to 0.9 and 0.999).

η is the learning rate, which controls the step size of the parameter updates.

ϵ is a small constant (usually 10^{-8}) added to the denominator for numerical stability.

The Adam optimizer is renowned for its adaptive learning rate capability, which efficiently optimizes deep neural networks by adjusting learning rates based on historical gradients [18]. This adaptability, coupled with the integration of momentum and RMSprop concepts, firmly establishes Adam as a powerful and widely employed algorithm in the field of deep learning. Notably, its effectiveness extends to large datasets and high-dimensional feature spaces, rendering it a valuable asset for deep learning research [19].

In this research, we employ the binary cross-entropy loss function as a vital element in training neural networks for binary classification tasks. The loss function, also known as a cost function or objective function in machine learning, quantifies the disparity between the model's predicted values and the true labels within the training data. The primary purpose of the loss function is to guide the optimization process by providing a measure of how well the model is performing on the task at hand [22].

Binary cross-entropy is a specific type of loss function well-suited for binary classification tasks, where the objective is to categorize data into one of two possible classes (e.g., malware or benign). The binary cross-entropy loss function computes the cross-entropy between the predicted probabilities and the true binary labels of the data samples. It assesses the dissimilarity between the model's predictions and the actual labels, taking into account both positive and negative samples. The formula for binary cross-entropy is given as:

$$\text{Binary Cross - Entropy} = -\frac{1}{N} \sum_{i=1}^N (y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)) \quad (9)$$

One of the advantages of using binary cross-entropy in the research being conducted is its suitability for binary classification tasks and its compatibility with the sigmoid activation function in the output layer. The sigmoid activation function restricts the model's output to a probability value between 0 and 1, allowing for the interpretation of predictions as class probabilities for binary classification. This aligns well with the binary cross-entropy loss, which measures the alignment between predicted probabilities and true binary labels, incentivizing the model to produce confident predictions for the correct class.

Additionally, binary cross-entropy provides a clear and interpretable measure of the model's performance. Lower values of binary cross-entropy indicate better alignment of the model's predictions with the true binary labels, implying improved classification accuracy. Thus, the choice of binary cross-entropy as the loss function in the research supports effective training of the neural network and accurate classification of samples into the two classes, namely malware and benign, enabling reliable analysis and detection in the domain of malware detection.

The model in this research undergoes training and hyperparameter tuning using a custom loop with GridSearchCV (Grid Search Cross-Validation), a powerful technique for hyperparameter optimization in machine learning [23]. GridSearchCV systematically explores a specified hyperparameter grid to identify the combination that yields the best model performance on the testing data. The hyperparameters being tuned in this research are the dropout rate, number of hidden units, and number of hidden layers. The dropout rate is a regularization technique that randomly drops out a fraction of neurons during training to prevent overfitting. The number of hidden units refers to the number of neurons in each hidden layer, and the number of hidden layers determines the depth and complexity of the neural network architecture.

GridSearchCV exhaustively searches all possible hyperparameter combinations within the grid. Each combination is trained and evaluated through cross-validation, where the dataset is split into multiple folds, and each fold acts as a validation set. This approach ensures a robust and unbiased evaluation of the model's performance [24]. The key advantage of using GridSearchCV in this research is its ability to automate hyperparameter tuning and explore a wide range of combinations, leading to the identification of optimal hyperparameters for improved testing data accuracy. GridSearchCV systematically compares various hyperparameter combinations, aiding researchers in making informed decisions regarding neural network architecture and configuration. Furthermore, GridSearchCV's evaluation on multiple folds of the dataset ensures the selection of hyperparameters that generalize well to new data and minimize overfitting risks.

Once the best model has been selected, the training process proceeds with the epochs and learning rate specified for the experimentation. In this research journal, we conduct a series of experiments to explore the impact of different learning rates on the model's performance. Specifically, we test four learning rates: 0.1, 0.05, 0.01, and 0.001. The learning rate plays a crucial role in the optimization process, determining the step size for updating the model's parameters during training. By examining various learning rates, we aim to identify the most suitable rate that maximizes the model's convergence and classification accuracy. Additionally, we perform the training process for two different epoch values, namely 100 and 200. This variation in the number of epochs allows us to observe how the model's performance evolves over time and how additional training iterations affect its accuracy and other evaluation metrics.

For the evaluation of the model's performance, we choose essential metrics, including accuracy, precision, recall, and F1-score. These metrics serve as comprehensive measures of the model's classification capabilities in distinguishing between malware and benign samples. Accuracy provides an overall assessment of the model's correctness, while precision evaluates its ability to correctly identify positive samples. Recall measures the model's capability to detect all positive instances in the dataset, and F1-score strikes a balance between precision and recall, providing a more nuanced understanding of the model's performance.

By systematically analyzing the model's behavior and performance across different learning rates and epochs using these evaluation metrics, our research aims to gain valuable insights into the optimal hyperparameter configurations that lead to superior malware detection accuracy. The findings from this study will contribute to advancing the field of cybersecurity and fostering the development of more robust and accurate malware detection systems.

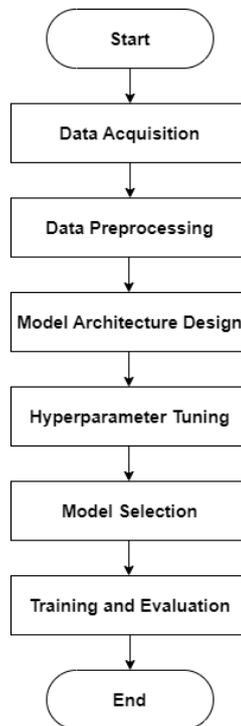


Figure 2. Research Design Flowchart

2.2. Model Evaluation

The model evaluation process is crucial to assess the effectiveness of the proposed method. It aims to measure the model's capability in accurately classifying malware and benign samples. The evaluation involves employing well-established performance metrics, namely accuracy, precision, recall, and F1-scores, to provide an objective and comprehensive analysis of the model's performance.

Accuracy assesses the overall correctness of the model's predictions by measuring the percentage of correctly classified samples. Precision evaluates the model's ability to correctly identify positive samples (benign) out of all predicted positive instances. Recall measures the model's capacity to detect all positive samples from the entire dataset of positive samples. F1-score strikes a balance between precision and recall, providing a harmonic mean that indicates the model's overall effectiveness in binary classification tasks.

Table 1. Performance Metrics for Evaluation

Metric	Formula	Description
True Positive (TP)	count()	Number of samples correctly classified as benign
False Positive (FP)	count()	Number of samples incorrectly classified as benign
True Negative (TN)	count()	Number of samples correctly classified as malware
False Negative (FN)	count()	Number of samples incorrectly classified as malware
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	Proportion of correctly predicted samples
Precision	$\frac{TP}{TP + FP}$	How well the model predicts benign samples as benign
Recall/TPR	$\frac{TP}{TP + FN}$	How well the model is able to find all benign samples
F1-score	$2 \times \frac{Precision \times Recall}{Precision + Recall}$	Harmonic mean of precision and recall, measures the balance between precision and recall

By utilizing these performance metrics, the research ensures a rigorous and systematic evaluation of the model's classification capabilities. The results obtained from the evaluation process will play a significant role in validating the proposed approach and making informed decisions about its practical applicability in the field of malware detection.

3. RESULTS AND ANALYSIS

In this section, we present the outcomes and analysis of our research aimed at optimizing malware detection using the Back Propagation Neural Network (BPNN) in conjunction with hyperparameter tuning. Following the selection of the best model through GridSearchCV, we conducted multiple experiments by varying the learning rates and epochs. Specifically, we tested learning rates of 0.1, 0.05, 0.01, and 0.001, while setting epochs to 100 and 200.

The experimental results from the various learning rates and epochs are presented in Table 2. The table provides a comprehensive analysis of the model's performance, presenting key metrics such as accuracy, precision, recall, and F1-score. These metrics play a crucial role in evaluating the model's capability to accurately detect and distinguish between malware and benign samples. Additionally, we meticulously recorded the computational time required for each experiment, allowing us to thoroughly assess the model's training efficiency.

Table 2. Experimental Results of Malware Detection using BPNN with Hyperparameter Variations

learning rate	Epoch	Best Hyperparameter according to GridSearchCV			Acc (%)	Prec (%)	Recall (%)	F1-score (%)	Computational Time (s)
		Dropout Rate	Number of Neurons in the Hidden Layer	Number of Hidden Layers					
0.1	50	0.2	20	1	97	97	97	97	1144.91
0.05	50	0.2	20	1	97	97	97	96	1189.57
0.01	50	0.1	10	2	97	97	97	97	734.31
0.001	50	0.2	20	2	97	97	97	97	1290.57
0.1	100	0.1	20	1	97	97	97	97	2292.28
0.05	100	0.1	20	1	97	97	97	97	2346.54
0.01	100	0.1	20	1	97	97	97	97	1421.77
0.001	100	0.1	20	2	97	97	97	97	1581.69
0.1	200	0.1	20	1	98	98	98	98	6325.65
0.05	200	0.1	20	2	96	97	96	97	5414.67
0.01	200	0.1	20	2	97	97	97	97	7464.18
0.001	200	0.2	20	1	97	97	97	97	5459.82

From the table, we observed that different hyperparameter values influence the model's performance in detecting malware. For instance, varying the learning rate and Epochs led to variations in the model's accuracy, precision, recall, and F1-score. Notably, higher values for these metrics indicate a more accurate and reliable detection system. Furthermore, the Dropout Rate, Number of Neurons in Hidden Layer, and Number of Hidden Layers also played a crucial role in shaping the model's performance. By tuning these hyperparameters, the model achieved significant improvements in detecting both malware and benign samples.

Upon analyzing the table, several noteworthy trends become evident. Firstly, the hyperparameter combination with a learning rate of 0.1, 200 epochs, and a dropout rate of 0.1, along with a single hidden layer containing 20 neurons, consistently produced the highest performance across all metrics, achieving an outstanding accuracy, precision, recall, and F1-score of 98%. Moreover, this configuration demonstrated a reasonable computational time of 6325.65 seconds, striking an effective balance between computational efficiency and model performance.

Secondly, the combination with a learning rate of 0.01, 50 epochs, and a dropout rate of 0.2, along with a single hidden layer containing 20 neurons, also exhibited competitive results, achieving 97% accuracy, precision, recall, and F1-score. Additionally, this configuration demonstrated a relatively low computational time of 734.31 seconds, further highlighting its efficiency.

Nevertheless, it is essential to consider other combinations as well. For instance, the hyperparameter combination with a learning rate of 0.05, 100 epochs, and a dropout rate of 0.1, along with a single hidden layer containing 20 neurons, achieved 97% performance across all metrics, while requiring a computational time of 2346.54 seconds. Although the computational time is higher compared to the previously mentioned configurations, it still offers acceptable performance.

On the other hand, some configurations, such as a learning rate of 0.05 with 200 epochs and a dropout rate of 0.1, along with two hidden layers, each containing 20 neurons, resulted in slightly lower performance, with accuracy, precision, recall, and F1-score falling in the range of 96-97%. Although the computational time was moderate at 5414.67 seconds, this combination may not be the most optimal choice for achieving the highest recall and F1-score.

The computational time varied significantly between experiments due to the influence of hyperparameter configurations. Longer training times were observed for configurations with more epochs and complex network structures. The increased number of epochs necessitates more iterations during the training process, consuming additional computational resources and time. Similarly, network structures with more hidden layers and neurons result in higher parameter counts, leading to increased computations during forward and backward passes. Consequently, configurations with more complex structures require longer training times. Conversely, configurations with fewer epochs and simpler network structures may offer shorter training times but could compromise optimal performance.

In summary, the best hyperparameter combination should be determined by a careful trade-off between performance metrics (accuracy, precision, recall, F1-score) and computational time. The configuration with a learning rate of 0.1 and epochs of 200 stands out as the most promising choice, offering the highest overall performance and computational efficiency. Nevertheless, other combinations should also be considered, as they may provide viable alternatives depending on the specific requirements and constraints of the malware detection system.



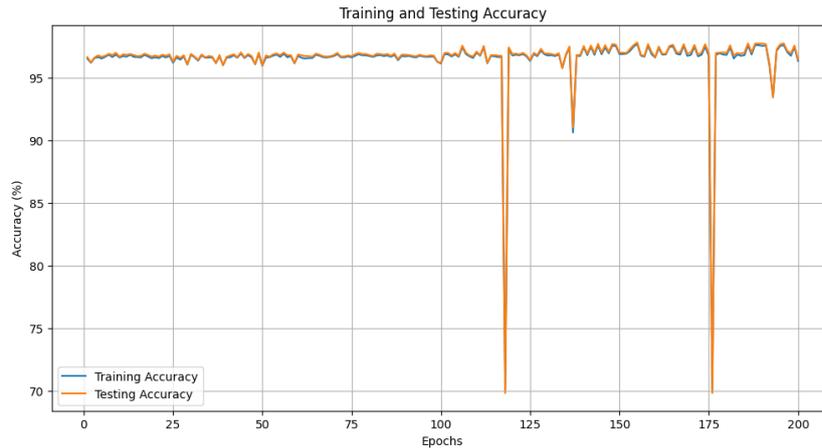


Figure 3. Comparison graph of accuracy for two different experimental settings. In condition (a), a learning rate of 0.1 was used in combination with 200 epochs, while in condition (b), a learning rate of 0.05 was employed with 200 epochs.

The implications of this finding are significant for malware detection using neural network models. The hyperparameter combination with a learning rate of 0.1 and 200 epochs demonstrates superior performance and computational efficiency. To comprehensively validate the advantages of this configuration, further research is warranted, utilizing a dataset with a greater diversity of features. Evaluating the model's performance on a more varied dataset will enhance the reliability and applicability of the findings to real-world scenarios. By conducting this additional research, the study can provide stronger support for the proposed hyperparameter configuration, leading to more confident decisions in developing high-performing and efficient malware detection systems.

4. CONCLUSION

In this research, we addressed the critical challenge of optimizing malware detection using the Back Propagation Neural Network (BPNN) in conjunction with hyperparameter tuning. With the rapid expansion of the internet and the consequent rise in cyber threats, developing a sophisticated malware detection system leveraging machine learning (ML) and deep learning (DL) techniques has become a priority to safeguard computer systems and networks.

Our study focused on fine-tuning key hyperparameters of the BPNN model, namely the dropout rate, the number of neurons in hidden layers, and the number of hidden layers, to enhance the accuracy of malware detection. To achieve this, we employed a Back Propagation Neural Network (BPNN) with dropout regularization, as it showed promise in handling complex data and classification tasks. The core of the research focused on hyperparameter optimization using GridSearchCV, a powerful technique for systematically exploring various combinations of hyperparameters to identify the optimal configuration. The experiments involved varying learning rates and epochs to understand their impact on the model's performance.

Through rigorous experimentation and analysis, we identified the hyperparameter combination with a learning rate of 0.1 and 200 epochs as the most promising choice. This configuration consistently demonstrated outstanding performance, achieving an accuracy, precision, recall, and F1-score of 98% across all metrics. Moreover, it exhibited reasonable computational efficiency, striking an effective balance between model performance and training time. The implications of our findings are significant for the field of malware detection using neural network models. The proposed hyperparameter configuration with a learning rate of 0.1 and 200 epochs exhibits superior performance and computational efficiency, making it a valuable asset in developing effective malware detection systems.

However, further research and validation on datasets with greater feature diversity are warranted to strengthen the reliability and applicability of the results in real-world scenarios. In conclusion, our research successfully optimized malware detection using the BPNN with hyperparameter tuning. By leveraging deep learning techniques and systematically fine-tuning key hyperparameters, we developed a more robust and accurate malware detection solution. The proposed hyperparameter configuration demonstrates outstanding performance while maintaining computational efficiency, contributing to the field of cybersecurity and fortifying the security of computer systems and networks against malicious threats.

Future research in this area should focus on validating the proposed hyperparameter configuration on diverse datasets and exploring the integration of other advanced ML and DL techniques to further enhance the model's capabilities. Despite the promising results, our research acknowledges the limitations inherent in the

dataset used for experimentation. To address this concern, utilizing more diverse and comprehensive datasets can ensure the model's robustness. Additionally, other hyperparameter optimization techniques beyond GridSearchCV should be explored to fine-tune the model's performance even further.

By adopting the insights from this study, the field of cybersecurity can benefit from more reliable and efficient malware detection systems, ultimately fortifying the security of computer systems and networks against the ever-evolving landscape of cyber threats.

REFERENCES

- [1] "Internet and social media users in the world 2023 | Statista." <https://www.statista.com/statistics/617136/digital-population-worldwide/> (accessed Jul. 01, 2023).
- [2] D. K. Bhattacharyya and J. K. Kalita, *Network Anomaly Detection: ML Perspective*. 2014.
- [3] L. Li *et al.*, "On Locating Malicious Code in Piggybacked Android Apps," *J. Comput. Sci. Technol.*, vol. 32, no. 6, pp. 1108–1124, 2017, doi: 10.1007/s11390-017-1786-z.
- [4] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust Intelligent Malware Detection Using Deep Learning," *IEEE Access*, vol. 7, no. c, pp. 46717–46738, 2019, doi: 10.1109/ACCESS.2019.2906934.
- [5] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, vol. 13-17-Aug, pp. 1225–1234, 2016, doi: 10.1145/2939672.2939753.
- [6] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," *Proc. 2015 10th Int. Conf. Malicious Unwanted Softw.*, 2016.
- [7] F. Al Huda, W. F. Mahmudy, and H. Tolle, "Android malware detection using backpropagation neural network," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 4, no. 1, pp. 240–244, 2016, doi: 10.11591/ijeecs.v4.i1.pp240-244.
- [8] Z.-P. Pan, C. Feng, and C.-J. Tang, "Malware Classification Based on the Behavior Analysis and Back Propagation Neural Network," *ITM Web Conf.*, vol. 7, p. 02001, 2016, doi: 10.1051/itmconf/20160702001.
- [9] A. Makandar and A. Patrot, "Malware analysis and classification using Artificial Neural Network," *Int. Conf. Trends Autom. Commun. Comput. Technol. I-TACT 2015*, 2016, doi: 10.1109/ITACT.2015.7492653.
- [10] B. B. Rad, M. K. H. Nejad, and M. Shahpasand, "Malware classification and detection using artificial neural network," *J. Eng. Sci. Technol.*, vol. 13, no. Special Issue on ICCSIT 2018, pp. 14–23, 2018.
- [11] D. Jayanth, "Malware-Detection-in-PE-files-using-Machine-Learning," 2021. [https://github.com/DasariJayanth/Malware-Detection-in-PE-files-using-Machine-Learning/tree/master/PE_Header\(exe%2Cdllfiles\)](https://github.com/DasariJayanth/Malware-Detection-in-PE-files-using-Machine-Learning/tree/master/PE_Header(exe%2Cdllfiles))
- [12] G. Ciaburro, V. Kishore Ayyadevara, and A. Perrier, *Hands-On Machine Learning on Google Cloud Platform*, vol. 6, no. August. Packt Publishing Ltd, 2018.
- [13] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 6, pp. 1929–1958, 2014, doi: 10.1016/0010-4361(73)90803-3.
- [14] R. H. R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature*, vol. 405, no. 6789, pp. 947–951, 2000, doi: 10.1038/35016072.
- [15] M. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015. doi: 10.1108/978-1-83909-694-520211010.
- [16] F. Chollet, *Deep Learning with Python*, First Edit. Manning Publications, 2017. doi: 10.1007/978-1-4842-5364-9.
- [17] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing*, vol. 503, pp. 92–108, 2022, doi: 10.1016/j.neucom.2022.06.111.
- [18] M. Liu, D. Yao, Z. Liu, J. Guo, and J. Chen, "An Improved Adam Optimization Algorithm Combining Adaptive Coefficients and Composite Gradients Based on Randomized Block Coordinate Descent," *Comput. Intell. Neurosci.*, vol. 2023, pp. 1–14, 2023, doi: 10.1155/2023/4765891.
- [19] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd Int. Conf. Learn. Represent.*, pp. 1–15, 2015.
- [20] S. Ruder, "An overview of gradient descent optimization algorithms," pp. 1–14, 2016, [Online]. Available: <http://arxiv.org/abs/1609.04747>
- [21] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journa lof Mach. Learn. Res.*, vol. 12, pp. 2121–2159, 2011.
- [22] M. C. Dickson, A. S. Bosman, and K. M. Malan, "Hybridised Loss Functions for Improved Neural Network Generalisation," *Pan-African Artif. Intell. Smart Syst. PAAISS 2021. Lect. Notes Inst. Comput. Sci. Soc. Informatics Telecommun. Eng.*, vol. 405, pp. 169–181, 2022, doi: 10.1007/978-3-030-93314-2_11.
- [23] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, 2020, doi: 10.1016/j.neucom.2020.07.061.
- [24] E. Elgeldawi, A. Sayed, A. R. Galal, and A. M. Zaki, "Hyperparameter tuning for machine learning algorithms used for arabic sentiment analysis," *Informatics*, vol. 8, no. 4, pp. 1–21, 2021, doi: 10.3390/informatics8040079.

BIBLIOGRAPHY OF AUTHORS

Annisa Arrumaisha Siregar, currently an active final semester student in the Department of Electrical Engineering, specializing in the Telecommunication Engineering Study Program, at State Polytechnic of Sriwijaya. The author is interested in the fields of artificial intelligence and cyber security.



Sopian Soim, currently a lecturer in the Department of Electrical Engineering, specializing in the Telecommunication Engineering Study Program, at State Polytechnic of Sriwijaya. The author completed a Bachelor's Degree in Electrical Engineering at Universitas Sriwijaya in 1997 and then pursued a Master's Degree in Telecommunication Engineering at Institut Teknologi Sepuluh Nopember, graduating in 2007. The author's research interests lie in the areas of wireless communication, data science, artificial intelligence, and cyber security.



Mohammad Fadhli, currently a lecturer in the Department of Electrical Engineering, specializing in the Telecommunication Engineering Study Program, at State Polytechnic of Sriwijaya. The author completed a Bachelor's Degree in Electronics Engineering at Universitas Negeri Padang in 2013 and then pursued a Master's Degree in Multimedia Telecommunication at Institut Teknologi Sepuluh Nopember, graduating in 2015. The author's research interests lie in the areas of wireless communication, wireless sensor networks, and the Internet of Things.

ANGGARAN BIAYA

No	Material	Justifikasi Pemakaian	Kuantitas	Harga Satuan (Rp)	Harga Peralatan Penunjang (Rp)
1	Pembuatan Proposal Pra-Tugas Akhir	Alat Persyaratan Pra-TA	5 buah	30.000,-	150.000,-
2	Pembuatan Tugas Akhir	Alat Persyaratan TA	5 buah	70.000,-	350.000,-
3	Publikasi, Seminar dll	Alat Persyaratan TA	1 kali	1.000.000,-	1.000.000,-
4	ATK	Penunjang	1 set	200.000,-	200.000,-
TOTAL					1.700.000,-

SOURCE CODE

1. Algoritma Back Propagation Neural Network

1.1. Source code algoritma Back Propagation Neural Network Tanpa Optimasi Hyperparameter Tuning

```
from math import exp
from random import seed, random
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.metrics import confusion_matrix,
classification_report, precision_score, recall_score, f1_score,
roc_curve, auc
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
import time
import datetime

from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)

# Load dataset
dataset = pd.read_excel('/content/gdrive/My Drive/Colab
Notebooks/bpnn_d2/final_pe_data-Malware-Detection.xlsx')
dataset = dataset.dropna(axis=1) # untuk menghapus kolom jika
semua isinya adalah nan
dataset = dataset.dropna(axis=0) # untuk menghapus baris jika
semua adalah nan

dataset = dataset.sample(frac=1)
dataset = dataset.values[:, :]
min_max_scaler = MinMaxScaler()
train_scale = min_max_scaler.fit_transform(dataset)
print(train_scale)
y = dataset[:, -1]

# Split dataset menjadi data latih dan data uji
train_data, test_data = train_test_split(train_scale,
test_size=0.2, random_state=42)

# Jumlah data latih
jumlah_data_latih = len(train_data)
print("Jumlah Data Latih:", jumlah_data_latih)

# Jumlah data uji
jumlah_data_uji = len(test_data)
print("Jumlah Data Uji:", jumlah_data_uji)

# Initialize a network
def initialize_network(n_inputs, n_hidden, n_outputs):
    network = list()
    hidden_layer = [{'weights': [random() for _ in range(n_inputs
```

```

+ 1]}} for _ in range(n_hidden)]
    network.append(hidden_layer)
    output_layer = [{'weights': [random() for _ in range(n_hidden
+ 1)]] for _ in range(n_outputs)]
    network.append(output_layer)
    return network

# Calculate neuron activation for an input
def activate(weights, inputs):
    activation = weights[-1]
    for i in range(len(weights) - 1):
        activation += weights[i] * float(inputs[i])
    return activation

# Transfer neuron activation
def transfer(activation):
    return 1.0 / (1.0 + exp(-activation))

# Forward propagate input to a network output
def forward_propagate(network, row):
    inputs = row
    for layer in network:
        new_inputs = []
        for neuron in layer:
            activation = activate(neuron['weights'], inputs)
            neuron['output'] = transfer(activation)
            new_inputs.append(neuron['output'])
        inputs = new_inputs
    return inputs

# Calculate the derivative of a neuron output
def transfer_derivative(output):
    return output * (1.0 - output)

# Backpropagate error and store it in neurons
def backward_propagate_error(network, expected):
    for i in reversed(range(len(network))):
        layer = network[i]
        errors = list()
        if i != len(network) - 1:
            for j in range(len(layer)):
                error = 0.0
                for neuron in network[i + 1]:
                    error += neuron['weights'][j] *
neuron['delta']
                errors.append(error)
            else:
                for j in range(len(layer)):
                    neuron = layer[j]
                    errors.append(expected[j] - neuron['output'])
                for j in range(len(layer)):
                    neuron = layer[j]
                    neuron['delta'] = errors[j] *
transfer_derivative(neuron['output'])

```

```

# Update network weights with error
def update_weights(network, row, l_rate):
    for i in range(len(network)):
        inputs = row[:-1]
        if i != 0:
            inputs = [neuron['output'] for neuron in network[i -
1]]
        for neuron in network[i]:
            for j in range(len(inputs)):
                neuron['weights'][j] += l_rate * neuron['delta'] *
float(inputs[j])
                neuron['weights'][-1] += l_rate * neuron['delta']

# Make Class Predictions on Training Data and Testing Data
prediksi_output_train = []
prediksi_output_test = []

# Train a network for a fixed number of epochs
def train_network(network, train, test, l_rate, n_epoch,
n_outputs):
    # Initialize lists to store epoch-wise metrics
    epoch_losses_train = []
    epoch_accuracies_train = []
    epoch_precisions_train = []
    epoch_recalls_train = []
    epoch_f1_scores_train = []

    epoch_losses_test = []
    epoch_accuracies_test = []
    epoch_precisions_test = []
    epoch_recalls_test = []
    epoch_f1_scores_test = []

    confusion_mats_train = [] # List untuk menyimpan Confusion
Matrix pada setiap epoch
    confusion_mat_train = None
    confusion_mat_test = None

    for epoch in range(n_epoch):
        sum_error_train = 0

        # Reset predictions for the current epoch
        data_output_predict_train = [] # List untuk menyimpan
hasil prediksi pada setiap data (training)
        data_output_real_train = [] # List untuk menyimpan label
aktual pada setiap epoch (training)

        for row in train:
            outputs_train = forward_propagate(network, row)
            expected_train = [0 for _ in range(n_outputs)]
            expected_train[int(row[-1])] = 1

            max_index_train =
outputs_train.index(max(outputs_train))
            data_output_predict_train.append(max_index_train)
            data_output_real_train.append(int(row[-1]))

```

```

        if max_index_train != int(row[-1]):
            sum_error_train += 1

        backward_propagate_error(network, expected_train)
        update_weights(network, row, l_rate)

    sum_error_train /= len(train)
    sum_error_train *= 100
    epoch_losses_train.append(sum_error_train)
    epoch_accuracies_train.append(100 - sum_error_train)

    # Append predictions to prediksi_output_train
    prediksi_output_train.append(data_output_predict_train)

    # Evaluation on training data
    prediksi_output_train_epoch =
data_output_predict_train.copy() # Salin hasil prediksi untuk
epoch saat ini

    # Calculate evaluation metrics for training data
    precision_train = precision_score(data_output_real_train,
prediksi_output_train_epoch, zero_division=1)
    recall_train = recall_score(data_output_real_train,
prediksi_output_train_epoch, zero_division=1)
    f1_train = f1_score(data_output_real_train,
prediksi_output_train_epoch, zero_division=1)

    epoch_precisions_train.append(precision_train)
    epoch_recalls_train.append(recall_train)
    epoch_f1_scores_train.append(f1_train)

    # Reset predictions for the current epoch
current data_output_predict_test = [] # Reset predictions for the
epoch
data_output_real_test = [] # Reset labels for the current
epoch
    sum_error_test = 0

    for row in test:
        outputs_test = forward_propagate(network, row)
        expected_test = [0 for _ in range(n_outputs)]
        expected_test[int(row[-1])] = 1

        max_index_test = outputs_test.index(max(outputs_test))
        data_output_predict_test.append(max_index_test)
        data_output_real_test.append(int(row[-1]))

        if max_index_test != int(row[-1]):
            sum_error_test += 1

    sum_error_test /= len(test)
    sum_error_test *= 100
    epoch_losses_test.append(sum_error_test)
    epoch_accuracies_test.append(100 - sum_error_test)

```

```

# Append predictions to prediksi_output_test
prediksi_output_test.append(data_output_predict_test)

# Evaluation on testing data
prediksi_output_test_epoch = data_output_predict_test #
Simpan hasil prediksi untuk epoch saat ini

# Calculate evaluation metrics for testing data
precision_test = precision_score(data_output_real_test,
prediksi_output_test_epoch, zero_division=1)
recall_test = recall_score(data_output_real_test,
prediksi_output_test_epoch, zero_division=1)
f1_test = f1_score(data_output_real_test,
prediksi_output_test_epoch, zero_division=1)

epoch_precisions_test.append(precision_test)
epoch_recalls_test.append(recall_test)
epoch_f1_scores_test.append(f1_test)

confusion_mat_train =
confusion_matrix(data_output_real_train,
prediksi_output_train_epoch)
confusion_mats_train.append(confusion_mat_train) #
Menambahkan Confusion Matrix ke dalam list

print('> epoch=%d, lrate=%.3f, error_train=%.3f %%,
error_test=%.3f %%' % (epoch, l_rate, sum_error_train,
sum_error_test))

# Print "Selesai" after all epochs are completed
print("Selesai")

# Calculate the final Confusion Matrix (Training)
confusion_mat_train = confusion_matrix(data_output_real_train,
prediksi_output_train[-1])

# Calculate the final Confusion Matrix (Testing)
confusion_mat_test = confusion_matrix(data_output_real_test,
prediksi_output_test[-1])

# Visualize Training and Testing Loss
plt.figure()
plt.plot(range(n_epoch), epoch_losses_train, label='Training')
plt.plot(range(n_epoch), epoch_losses_test, label='Testing')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training and Testing Loss')
plt.legend()
plt.show()
print()

# Visualize Training and Testing Accuracy
plt.figure()
plt.plot(range(n_epoch), epoch_accuracies_train,
label='Training')
plt.plot(range(n_epoch), epoch_accuracies_test,

```

```

label='Testing')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and Testing Accuracy')
plt.legend()
plt.show()
print()

# Visualize Training and Testing Precision
plt.figure()
plt.plot(range(n_epoch), epoch_precisions_train,
label='Training')
plt.plot(range(n_epoch), epoch_precisions_test,
label='Testing')
plt.xlabel('Epoch')
plt.ylabel('Score')
plt.title('Training and Testing Precision')
plt.legend()
plt.show()
print()

# Visualize Training and Testing Recall
plt.figure()
plt.plot(range(n_epoch), epoch_recalls_train,
label='Training')
plt.plot(range(n_epoch), epoch_recalls_test, label='Testing')
plt.xlabel('Epoch')
plt.ylabel('Score')
plt.title('Training and Testing Recall')
plt.legend()
plt.show()
print()

# Visualize Training and Testing F1-score
plt.figure()
plt.plot(range(n_epoch), epoch_f1_scores_train,
label='Training')
plt.plot(range(n_epoch), epoch_f1_scores_test,
label='Testing')
plt.xlabel('Epoch')
plt.ylabel('Score')
plt.title('Training and Testing F1-score')
plt.legend()
plt.show()
print()

# Print the final Confusion Matrix (Training)
print("Confusion Matrix (Training):")
print(confusion_mat_train)
print()

# Print the final Classification Report (Training)
classification_report_train =
classification_report(data_output_real_train,
prediksi_output_train[-1])
print("Classification Report (Training):")

```

```

print(classification_report_train)
print()

# Print the final Confusion Matrix (Testing)
print("Confusion Matrix (Testing):")
print(confusion_mat_test)
print()

# Print the final Classification Report (Testing)
classification_report_test =
classification_report(data_output_real_test,
prediksi_output_test[-1])
print("Classification Report (Testing):")
print(classification_report_test)

return confusion_mat_train, confusion_mat_test

# Test backpropagation algorithm
seed(1)

n_inputs = len(train_data[0]) - 1
n_outputs = len(set([row[-1] for row in train_data]))
network = initialize_network(n_inputs, 10, n_outputs)
network_awal = network.copy()

start_time = time.time()

l_rate = 0.05
n_epoch = 100
prediksi_output_train = [] # Inisialisasi list untuk prediksi
output train
prediksi_output_test = [] # Inisialisasi list untuk prediksi
output test
confusion_mat_train, confusion_mat_test = train_network(network,
train_data, test_data, l_rate, n_epoch, n_outputs)
output_training = train_data[:, -1]
output_testing = test_data[:, -1]

end_time = time.time()
computing_time = end_time - start_time
print("Lamanya waktu komputasi: {:.2f}
detik".format(computing_time))

# Konversi ke format jam:menit:detik
computing_time_formatted =
str(datetime.timedelta(seconds=int(computing_time)))
print("Lamanya waktu komputasi:
{}").format(computing_time_formatted))

# Calculate the confusion matrix for training data
confusion_mat_train = confusion_matrix(output_training,
prediksi_output_train[-1])

# Plot confusion matrix with class labels for training data
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_mat_train, annot=True, fmt="d",

```

```

cmap="Blues")
plt.title("Confusion Matrix (Training)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
print()

# Calculate the confusion matrix for testing data
confusion_mat_test = confusion_matrix(output_testing,
prediksi_output_test[-1])

# Plot confusion matrix with class labels for testing data
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_mat_test, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix (Testing)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
print()

# Kurva AUC (Area Under the Curve) for training data
fpr, tpr, thresholds = roc_curve(output_training,
prediksi_output_train[-1])
roc_auc = auc(fpr, tpr)

# Kurva ROC (Receiver Operating Characteristic) for training data
plt.figure()
plt.plot(fpr, tpr, label='ROC curve (area= %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (Training)')
plt.legend(loc="lower right")
plt.show()
print()

# Kurva AUC (Area Under the Curve) for testing data
fpr_test, tpr_test, thresholds_test = roc_curve(output_testing,
prediksi_output_test[-1])
roc_auc_test = auc(fpr_test, tpr_test)

# Kurva ROC (Receiver Operating Characteristic) for testing data
plt.figure()
plt.plot(fpr_test, tpr_test, label='ROC curve (area = %0.2f)' %
roc_auc_test)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (Testing)')
plt.legend(loc="lower right")
plt.show()
print()

```

```

# Visualize the neural network model
import matplotlib.pyplot as plt

def visualize_network(network):
    # Count the number of neurons in each layer
    num_inputs = len(network[0][0]['weights']) - 1
    num_hidden = len(network[0])
    num_outputs = len(network[1])

    # Create a new figure
    plt.figure(figsize=(8, 6))

    # Draw scatter plot for input layer
    plt.scatter([1] * num_inputs, range(num_inputs), color='blue',
label=f'Input Layer ({num_inputs} neurons)')

    # Draw scatter plot for hidden layer
    plt.scatter([2] * num_hidden, range(num_hidden),
color='orange', label=f'Hidden Layer ({num_hidden} neurons)')

    # Draw scatter plot for output layer
    plt.scatter([3] * num_outputs, range(num_outputs),
color='green', label=f'Output Layer ({num_outputs} neurons)')

    # Connect neurons with lines for input to hidden layer
    for i in range(num_inputs):
        for j in range(num_hidden):
            plt.plot([1, 2], [i, j], color='gray')

    # Connect neurons with lines for hidden to output layer
    for i in range(num_hidden):
        for j in range(num_outputs):
            plt.plot([2, 3], [i, j], color='gray')

    # Set the axis labels and title
    plt.xlabel('Layer')
    plt.ylabel('Neuron Index')
    plt.title('Neural Network Architecture')

    # Set the legend
    plt.legend(loc='best')

    # Remove x-axis ticks
    plt.xticks([])

    # Show the plot
    plt.show()

# Visualize the network architecture
visualize_network(network_awal)

# Save the model to a file using pickle
with open('/content/gdrive/My Drive/Colab
Notebooks/bpnn_d2/bpnn_d2_10_0.05_100/bpnn_d2_10_0.05_100.pkl',
'wb') as f:

```

```

        pickle.dump(network, f)

# Load the model from the file using pickle
with open('/content/gdrive/My Drive/Colab
Notebooks/bpnn_d2/bpnn_d2_10_0.05_100/bpnn_d2_10_0.05_100.pkl',
'rb') as f:
    model = pickle.load(f)

# Print the loaded model
print(model)

```

1.2. Source code algoritma Back Propagation Neural Network Dengan Optimasi Hyperparameter Tuning Menggunakan Grid Search

```

# Import library
import tensorflow as tf
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix,
classification_report, precision_score, recall_score, f1_score
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import time
import datetime
import seaborn as sns
from sklearn.metrics import roc_auc_score, roc_curve

from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)

# Load dataset
dataset = pd.read_excel('/content/gdrive/My Drive/Colab
Notebooks/bpnn_adam_d2/final_pe_data-Malware-Detection.xlsx')
dataset = dataset.dropna(axis=1)
dataset = dataset.dropna(axis=0)

dataset = dataset.sample(frac=1)
dataset = dataset.values[:, :]
min_max_scaler = MinMaxScaler()
train_scale = min_max_scaler.fit_transform(dataset)
print(train_scale)
y = dataset[:, -1]

# Grafik Data Visualization
plt.hist(y, bins=2, rwidth=0.8)
plt.xlabel('Class')
plt.ylabel('Count')
plt.title('Class Distribution')
plt.xticks(np.arange(2), ['Malware', 'Benign'])
plt.show

```

```

# Split dataset menjadi data latih dan data uji
train_data, test_data = train_test_split(train_scale,
test_size=0.2, random_state=42)

# Jumlah data latih
jumlah_data_latih = len(train_data)
print("Jumlah Data Latih:", jumlah_data_latih)

# Jumlah data uji
jumlah_data_uji = len(test_data)
print("Jumlah Data Uji:", jumlah_data_uji)

# Initialize a network using TensorFlow with dropout
regularization
def initialize_network(n_inputs, n_hidden_units, n_hidden_layers,
n_outputs, dropout_rate):
    model = tf.keras.Sequential()
    model.add(tf.keras.layers.Dense(n_hidden_units,
activation='relu', input_shape=(n_inputs,)))

    # Add dropout regularization after each hidden layer
    for _ in range(n_hidden_layers - 1):
        model.add(tf.keras.layers.Dropout(dropout_rate))
        model.add(tf.keras.layers.Dense(n_hidden_units,
activation='relu'))

    model.add(tf.keras.layers.Dropout(dropout_rate))
    model.add(tf.keras.layers.Dense(n_outputs,
activation='sigmoid')) # Update activation to 'sigmoid'

    return model

# Define the custom metrics as functions
def precision(y_true, y_pred):
    y_true = tf.cast(y_true, tf.float32)
    y_pred = tf.cast(y_pred, tf.float32)

    true_positives =
tf.keras.backend.sum(tf.keras.backend.round(tf.keras.backend.clip(
y_true * y_pred, 0, 1)))
    predicted_positives =
tf.keras.backend.sum(tf.keras.backend.round(tf.keras.backend.clip(
y_pred, 0, 1)))
    return true_positives / (predicted_positives +
tf.keras.backend.epsilon())

def recall(y_true, y_pred):
    y_true = tf.cast(y_true, tf.float32)
    y_pred = tf.cast(y_pred, tf.float32)

    true_positives =
tf.keras.backend.sum(tf.keras.backend.round(tf.keras.backend.clip(
y_true * y_pred, 0, 1)))
    possible_positives =
tf.keras.backend.sum(tf.keras.backend.round(tf.keras.backend.clip(
y_true, 0, 1)))

```

```

    return true_positives / (possible_positives +
tf.keras.backend.epsilon())

def f1_score(y_true, y_pred):
    y_true = tf.cast(y_true, tf.float32)
    y_pred = tf.cast(y_pred, tf.float32)

    p = precision(y_true, y_pred)
    r = recall(y_true, y_pred)
    return 2 * (p * r) / (p + r + tf.keras.backend.epsilon())

# Create a function to build the model for GridSearchCV
def build_model(dropout_rate=0.1, n_hidden_units=10,
n_hidden_layers=2):
    n_inputs = len(train_data[0]) - 1
    n_outputs = len(set([row[-1] for row in train_data]))
    model = initialize_network(n_inputs, n_hidden_units,
n_hidden_layers, n_outputs, dropout_rate)

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.1
),
              loss='binary_crossentropy', # Keep the loss
function as 'binary_crossentropy'
              metrics=['accuracy']) # Include accuracy in
metrics

    return model

print("looking for the best hyperparameters for model")

# Custom loop for hyperparameter tuning and training
param_grid = {
    'dropout_rate': [0.1, 0.2],
    'n_hidden_units': [10, 20],
    'n_hidden_layers': [1, 2]
}

best_model = None
best_accuracy = 0.0
best_params = {}

start_time_total = time.time()

for dropout_rate in param_grid['dropout_rate']:
    for n_hidden_units in param_grid['n_hidden_units']:
        for n_hidden_layers in param_grid['n_hidden_layers']:
            start_time = time.time()

            # Initialize model
            n_inputs = len(train_data[0]) - 1
            n_outputs = len(set([row[-1] for row in train_data]))
            model = initialize_network(n_inputs, n_hidden_units,
n_hidden_layers, n_outputs, dropout_rate)

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.1

```

```

),
                                loss='binary_crossentropy',
                                metrics=['accuracy'])

    # Convert true labels to one-hot encoded vectors
    train_labels_onehot =
tf.keras.utils.to_categorical(train_data[:, -1], num_classes=2)
    test_labels_onehot =
tf.keras.utils.to_categorical(test_data[:, -1], num_classes=2)

    # Train the model
    history = model.fit(train_data[:, :-1],
train_labels_onehot, epochs=10, batch_size=32, verbose=0)

    # Evaluate the model on the test set
    loss_test, acc_test = model.evaluate(test_data[:, :-
1], test_labels_onehot, verbose=0)

    # Update best model if accuracy improves
    if acc_test > best_accuracy:
        best_accuracy = acc_test
        best_model = model
        best_params = {'dropout_rate': dropout_rate,
'n_hidden_units': n_hidden_units, 'n_hidden_layers':
n_hidden_layers}

    end_time = time.time()
    computing_time = end_time - start_time
    print("Dropout Rate: {:.2f}, Hidden Units: {}, Hidden
Layers: {}, Testing Accuracy: {:.3f}%, Time: {:.2f}
seconds".format(
        dropout_rate, n_hidden_units, n_hidden_layers,
acc_test * 100, computing_time))

end_time_total = time.time()
computing_time_total = end_time_total - start_time_total
print("\nTotal computing time: {:.2f}
seconds".format(computing_time_total))
print("Best Hyperparameters: ", best_params)
print("Best Testing Accuracy: {:.3f}%".format(best_accuracy *
100))

# Train the best model
start_time = time.time()
epoch_losses_train = []
epoch_accuracies_train = []
epoch_precisions_train = []
epoch_recalls_train = []
epoch_f1_scores_train = []

epoch_losses_test = []
epoch_accuracies_test = []
epoch_precisions_test = []
epoch_recalls_test = []
epoch_f1_scores_test = []

```

```

# Convert true labels to one-hot encoded vectors
train_labels_onehot = tf.keras.utils.to_categorical(train_data[:,
-1], num_classes=2)
test_labels_onehot = tf.keras.utils.to_categorical(test_data[:, -
1], num_classes=2)

for epoch in range(200):
    start_time_epoch = time.time()

    # Train the model for one epoch
    history = best_model.fit(train_data[:, :-1],
train_labels_onehot, epochs=1, batch_size=32, verbose=0)

    # Calculate metrics manually after each epoch
    y_train_pred = best_model.predict(train_data[:, :-1])
    y_test_pred = best_model.predict(test_data[:, :-1])

    loss_train, acc_train = best_model.evaluate(train_data[:, :-
1], train_labels_onehot, verbose=0)
    loss_test, acc_test = best_model.evaluate(test_data[:, :-1],
test_labels_onehot, verbose=0)

    # Calculate custom metrics using one-hot encoded vectors
    precision_train = precision(train_labels_onehot, y_train_pred)
    recall_train = recall(train_labels_onehot, y_train_pred)
    f1_score_train = f1_score(train_labels_onehot, y_train_pred)

    precision_test = precision(test_labels_onehot, y_test_pred)
    recall_test = recall(test_labels_onehot, y_test_pred)
    f1_score_test = f1_score(test_labels_onehot, y_test_pred)

    epoch_losses_train.append(loss_train)
    epoch_accuracies_train.append(acc_train * 100)
    epoch_precisions_train.append(precision_train)
    epoch_recalls_train.append(recall_train)
    epoch_f1_scores_train.append(f1_score_train)

    epoch_losses_test.append(loss_test)
    epoch_accuracies_test.append(acc_test * 100)
    epoch_precisions_test.append(precision_test)
    epoch_recalls_test.append(recall_test)
    epoch_f1_scores_test.append(f1_score_test)

    end_time_epoch = time.time()
    computing_time_epoch = end_time_epoch - start_time_epoch
    print("> Epoch={}, Dropout Rate: {:.2f}, Hidden Units: {},
Hidden Layers: {}, Loss (train): {:.3f}, Accuracy (train):
{:.3f}%, Precision (train): {:.3f}, Recall (train): {:.3f}, F1-
score (train): {:.3f}, Loss (test): {:.3f}, Accuracy (test):
{:.3f}%, Precision (test): {:.3f}, Recall (test): {:.3f}, F1-score
(test): {:.3f}, Time per Epoch: {:.2f} seconds".format(
    epoch + 1, best_params['dropout_rate'],
best_params['n_hidden_units'], best_params['n_hidden_layers'],
    loss_train, acc_train * 100, precision_train,
recall_train, f1_score_train,
    loss_test, acc_test * 100, precision_test, recall_test,

```

```

f1_score_test, computing_time_epoch))

end_time = time.time()
computing_time_total = end_time - start_time
print("\nTotal computing time: {:.2f}
seconds".format(computing_time_total))

# Konversi ke format jam:menit:detik
computing_time_formatted =
str(datetime.timedelta(seconds=int(computing_time_total)))
print("Lamanya waktu komputasi:
{}".format(computing_time_formatted))

# Print best testing accuracy
print("Best Testing Accuracy: {:.3f}%".format(best_accuracy *
100))

# Visualization of metrics
epochs = list(range(1, 201))

# Visualize Training and Testing Loss
plt.figure(figsize=(12, 6))
plt.plot(epochs, epoch_losses_train, label='Training Loss')
plt.plot(epochs, epoch_losses_test, label='Testing Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training and Testing Loss')
plt.legend()
plt.grid(True)
plt.show()

# Visualize Training and Testing Accuracy
plt.figure(figsize=(12, 6))
plt.plot(epochs, epoch_accuracies_train, label='Training
Accuracy')
plt.plot(epochs, epoch_accuracies_test, label='Testing Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy (%)')
plt.title('Training and Testing Accuracy')
plt.legend()
plt.grid(True)
plt.show()

# Visualize Training and Testing Precision
plt.figure(figsize=(12, 6))
plt.plot(epochs, epoch_precisions_train, label='Training
Precision')
plt.plot(epochs, epoch_precisions_test, label='Testing Precision')
plt.xlabel('Epochs')
plt.ylabel('Precision')
plt.title('Training and Testing Precision')
plt.legend()
plt.grid(True)
plt.show()

```

```

# Visualize Training and Testing Recall
plt.figure(figsize=(12, 6))
plt.plot(epochs, epoch_recalls_train, label='Training Recall')
plt.plot(epochs, epoch_recalls_test, label='Testing Recall')
plt.xlabel('Epochs')
plt.ylabel('Recall')
plt.title('Training and Testing Recall')
plt.legend()
plt.grid(True)
plt.show()

# Visualize Training and Testing F1-score
plt.figure(figsize=(12, 6))
plt.plot(epochs, epoch_f1_scores_train, label='Training F1-score')
plt.plot(epochs, epoch_f1_scores_test, label='Testing F1-score')
plt.xlabel('Epochs')
plt.ylabel('F1-score')
plt.title('Training and Testing F1-score')
plt.legend()
plt.grid(True)
plt.show()

# Final Confusion Matrix and Classification Report for Training
Data
y_pred_train_prob = best_model.predict(train_data[:, :-1])
y_pred_train = np.argmax(y_pred_train_prob, axis=1)

confusion_matrix_train = confusion_matrix(train_data[:, -1],
y_pred_train)
classification_report_train = classification_report(train_data[:, -1], y_pred_train)

# Final Confusion Matrix and Classification Report for Testing
Data
y_pred_test_prob = best_model.predict(test_data[:, :-1])
y_pred_test = np.argmax(y_pred_test_prob, axis=1)

confusion_matrix_test = confusion_matrix(test_data[:, -1],
y_pred_test)
classification_report_test = classification_report(test_data[:, -1], y_pred_test)

# Print the final Confusion Matrix (Training)
print("Confusion Matrix (Training Data):\n",
confusion_matrix_train)
print()

# Print the final Classification Report (Training)
print("Classification Report (Training Data):\n",
classification_report_train)
print()

```

```

# Print the final Confusion Matrix (Testing)
print("Confusion Matrix (Testing Data):\n", confusion_matrix_test)
print()

# Print the final Classification Report (Testing)
print("Classification Report (Testing Data):\n",
      classification_report_test)
print()

# Confusion matrix for testing data
y_pred_test_prob = best_model.predict(test_data[:, :-1])
y_pred_test = np.argmax(y_pred_test_prob, axis=1)
confusion_mat_test = confusion_matrix(test_data[:, -1],
y_pred_test)

# Confusion matrix for training data
y_pred_train_prob = best_model.predict(train_data[:, :-1])
y_pred_train = np.argmax(y_pred_train_prob, axis=1)
confusion_mat_train = confusion_matrix(train_data[:, -1],
y_pred_train)

# Plot confusion matrix with class labels for training data
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_mat_train, annot=True, fmt="d",
            cmap="Blues")
plt.title("Confusion Matrix (Training)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# Plot confusion matrix with class labels for testing data
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_mat_test, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix (Testing)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# Calculate the confusion matrix for training data
y_pred_train_prob = best_model.predict(train_data[:, :-1])
y_pred_train = np.argmax(y_pred_train_prob, axis=1)
confusion_mat_train = confusion_matrix(train_data[:, -1],
y_pred_train)

# Plot confusion matrix with class labels for training data
class_labels = ['Malware', 'Benign']
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_mat_train, annot=True, fmt="d",
            cmap="Blues", xticklabels=class_labels, yticklabels=class_labels)
plt.title("Confusion Matrix (Training)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# Calculate the confusion matrix for testing data
y_pred_test_prob = best_model.predict(test_data[:, :-1])

```

```

y_pred_test = np.argmax(y_pred_test_prob, axis=1)
confusion_mat_test = confusion_matrix(test_data[:, -1],
y_pred_test)

# Plot confusion matrix with class labels for testing data
class_labels = ['Malware', 'Benign']
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_mat_test, annot=True, fmt="d", cmap="Blues",
xticklabels=class_labels, yticklabels=class_labels)
plt.title("Confusion Matrix (Testing)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# Predict probabilities for training and testing data
y_probs_train = best_model.predict(train_data[:, :-1])
y_probs_test = best_model.predict(test_data[:, :-1])

# Calculate AUC for training and testing data
auc_train = roc_auc_score(train_data[:, -1], y_probs_train[:, 1])
auc_test = roc_auc_score(test_data[:, -1], y_probs_test[:, 1])

# Calculate ROC curve for training and testing data
fpr_train, tpr_train, _ = roc_curve(train_data[:, -1],
y_probs_train[:, 1])
fpr_test, tpr_test, _ = roc_curve(test_data[:, -1],
y_probs_test[:, 1])

# Plot AUC and ROC curves for training data
plt.figure(figsize=(8, 6))
plt.plot(fpr_train, tpr_train, label=f'Training AUC =
{auc_train:.3f}')
plt.plot([0, 1], [0, 1], 'k--') # Diagonal reference line
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve (Training Data)')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()

# Plot AUC and ROC curves for testing data
plt.figure(figsize=(8, 6))
plt.plot(fpr_test, tpr_test, label=f'Testing AUC =
{auc_test:.3f}')
plt.plot([0, 1], [0, 1], 'k--') # Diagonal reference line
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve (Testing Data)')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()

# Visualize the network architecture
def visualize_network_architecture(model):
    # Get the weights and biases of the model
    weights = []

```

```

biases = []
for layer in model.layers:
    # Skip dropout layers
    if 'dropout' in layer.name:
        continue
    weights.append(layer.get_weights()[0])
    biases.append(layer.get_weights()[1])

# Create a new figure
plt.figure(figsize=(8, 6))

# Draw scatter plot for input layer
num_inputs = len(train_data[0]) - 1
plt.scatter([1] * num_inputs, range(1, num_inputs + 1),
color='blue', label=f'Input Layer ({num_inputs} neurons)')

# Draw scatter plot for hidden layers
for i, (weight, bias) in enumerate(zip(weights[:-1], biases[:-1])):
    num_neurons_hidden = len(weight)
    plt.scatter([2] * num_neurons_hidden, range(1,
num_neurons_hidden + 1), color='orange', label=f'Hidden Layer
{i+1} ({num_neurons_hidden} neurons)')

# Draw scatter plot for output layer
num_outputs = len(set([row[-1] for row in train_data]))
plt.scatter([3] * num_outputs, range(1, num_outputs + 1),
color='green', label=f'Output Layer ({num_outputs} neurons)')

# Connect neurons with lines for input to hidden layer
for i, weight in enumerate(weights[:-1]):
    for j in range(len(weight)):
        for k in range(len(weight[0])):
            plt.plot([1, 2], [j + 1, k + 1], 'gray',
linewidth=0.5)

# Connect neurons with lines for hidden to output layer
for j in range(len(weights[-1][0])):
    for k in range(len(weights[-1])):
        plt.plot([2, 3], [k + 1, j + 1], 'gray',
linewidth=0.5)

# Set the axis labels and title
plt.xlabel('Layer')
plt.ylabel('Neuron Index')
plt.title('Neural Network Architecture')

# Set the legend
plt.legend(loc='best')

# Remove x-axis ticks
plt.xticks([])

# Show the plot
plt.show()

```

```

# Visualize the network architecture
visualize_network_architecture(best_model)

# Save the model to an HDF5 file
model_save_path = '/content/gdrive/My Drive/Colab
Notebooks/bpnn_adam_d2/bpnn_adam_d2_0.1_200/bpnn_adam_d2_0.1_200.h
5'
best_model.save(model_save_path)

# Load the model from the HDF5 file
loaded_model = tf.keras.models.load_model(model_save_path)

# Print the loaded model
print(loaded_model)

```

2. Website Pendeteksi Malware

2.1.app.py

```

from flask import Flask, request, render_template
import pandas as pd
from sklearn import preprocessing
import numpy as np
import pefile

app = Flask(__name__)
from keras.models import load_model
# Load the model from the file
model = load_model('bpnn_adam_d2_0.1_200.h5')

dataset = pd.read_excel('final_pe_data-Malware-Detection.xlsx')
dataset = dataset.dropna(axis=1) # untuk menghapus kolom jika
semua isinya adalah nan
dataset = dataset.dropna(axis=0) # untuk menghapus baris jika
semua adalah nan

min_max_scaler = preprocessing.MinMaxScaler()
dataset = min_max_scaler.fit_transform(dataset);
dataset = dataset[1:,:]

@app.route("/", methods=["GET", "POST"])
def index():
    return render_template("index.html")

@app.route("/process", methods=["POST"])
def process():
    try:
        uploaded_file = request.files['file']
        if uploaded_file.filename != '':
            uploaded_file.save(uploaded_file.filename)
            pe = pefile.PE(uploaded_file.filename)
            dataset = np.zeros(15)

```

```

dataset[0] = pe.FILE_HEADER.Machine
dataset[1] = pe.FILE_HEADER.SizeOfOptionalHeader
dataset[2] = pe.FILE_HEADER.Characteristics
dataset[3] = pe.OPTIONAL_HEADER.MajorLinkerVersion
dataset[4] = pe.OPTIONAL_HEADER.MinorLinkerVersion
dataset[5] = pe.OPTIONAL_HEADER.SizeOfCode
dataset[6] = pe.OPTIONAL_HEADER.SizeOfInitializedData
dataset[7] =
pe.OPTIONAL_HEADER.SizeOfUninitializedData
dataset[8] = pe.OPTIONAL_HEADER.AddressOfEntryPoint
dataset[9] = pe.OPTIONAL_HEADER.BaseOfCode
dataset[10] = pe.OPTIONAL_HEADER.BaseOfData
dataset[11] = pe.OPTIONAL_HEADER.ImageBase
dataset[12] = pe.OPTIONAL_HEADER.SectionAlignment
dataset[13] = pe.OPTIONAL_HEADER.FileAlignment
dataset[14] =
pe.OPTIONAL_HEADER.MajorOperatingSystemVersion
print (dataset)
dataset = dataset.reshape(1, -1)
dataset = min_max_scaler.transform(dataset)
print (dataset)
dataset=dataset[:, [0,1,2,3,4,5,6,7,8,9,10,12,13,14]]

hasil = model.predict (dataset)
print (hasil)

if (hasil[0][0] ==1 ):
    hasil = "Benign"
    print ("Benign")
else:
    hasil = "Malware"
    print ("Malware")
    print (pe.dump_info ())
    return render_template ("process.html", hasil = hasil)
except Exception as e:
    print (e)
    return render_template ("index.html")

@app.route ("/home")
def home ():
    return render_template ("index.html")

if __name__ == "__main__":
    app.run ()

```

2.2.index.html

```

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta content="width=device-width, initial-scale=1.0"
name="viewport">

```

```

<title>Malware Detector</title>
<meta content="" name="description">
<meta content="" name="keywords">

<!-- stylesheets -->
<!-- bootstrap css -->
<link rel="stylesheet" type="text/css"
href="css/bootstrap.min.css">
<!-- Responsive-->
<link rel="stylesheet" href="css/responsive.css">
<!-- Tweaks for older IEs-->
<link rel="stylesheet"
href="https://netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-
awesome.css">
<!-- Favicons -->
<link href="../../static/img/logo-1.png" rel="icon">
<!-- Google Fonts -->
<link
href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,4
00,400i,600,600i,700,700i|Jost:300,300i,400,400i,500,500i,600,600i
,700,700i|Poppins:300,300i,400,400i,500,500i,600,600i,700,700i"
rel="stylesheet">
<!-- Vendor CSS Files -->
<link href="../../static/vendor/aos/aos.css" rel="stylesheet">
<link href="../../static/vendor/bootstrap/css/bootstrap.min.css"
rel="stylesheet">
<link href="../../static/vendor/bootstrap-icons/bootstrap-
icons.css" rel="stylesheet">
<link href="../../static/vendor/boxicons/css/boxicons.min.css"
rel="stylesheet">
<link href="../../static/vendor/glightbox/css/glightbox.min.css"
rel="stylesheet">
<link href="../../static/vendor/remixicon/remixicon.css"
rel="stylesheet">
<link href="../../static/vendor/swiper/swiper-bundle.min.css"
rel="stylesheet">
<!-- Template Main CSS File -->
<link href="../../static/css/style.css" rel="stylesheet">

</head>

<body>

<!-- ===== Header ===== -->
<header id="header" class="fixed-top ">
  <div class="container d-flex align-items-center">

    <!-- <h1 class="logo me-auto"><a href="index.html">MALDET
</a></h1> -->
    <p href="index.html" class="logo me-auto">
MALDET</p>

    <nav id="navbar" class="navbar">
      <ul>

```

```

        <li><a class="nav-link scrollto active"
href="#home">Home</a></li>
        <li><a class="nav-link scrollto"
href="#about">About</a></li>
        <li><a class="nav-link scrollto"
href="#services">Services</a></li>
        <li><a class="nav-link scrollto" href="#me">Get to Know
Me</a></li>
        <li><a class="getstarted scrollto" href="#services">Get
Started</a></li>
    </ul>
    <i class="bi bi-list mobile-nav-toggle"></i>
</nav><!-- .navbar -->

</div>
</header><!-- End Header -->

<!-- ===== Home Section ===== -->
<section id="home" class="d-flex align-items-center">

    <div class="container">
        <div class="row">
            <div class="col-lg-6 d-flex flex-column justify-content-
center pt-4 pt-lg-0 order-2 order-lg-1" data-aos="fade-up" data-
aos-delay="200">
                <h1 class="p-50">Malware Detector</h1>
                <p>Layanan web ini menawarkan fitur deteksi malware yang
dapat membantu pengguna dalam mengidentifikasi apakah file dengan
ekstensi .exe pada perangkat Windows adalah file yang tidak
berbahaya (benign) atau berpotensi berbahaya (malicious).</p>
                <div class="d-flex justify-content-center justify-
content-lg-start">
                    <a href="#services" class="btn-get-started
scrollto">Get Started</a>
                </div>
            </div>
            <div class="col-lg-6 order-1 order-lg-2 home-img" data-
aos="zoom-in" data-aos-delay="200">
                
            </div>
        </div>
    </div>
</section><!-- End Home -->

<main id="main">
<!-- ===== About Section ===== -->
<section id="about" class="about">
    <div class="container" data-aos="fade-up">

        <div class="section-title">
            <h2>About</h2>
        </div>

        <div class="row content">

```

```

        <div>
            <p>
                Layanan web ini menawarkan fitur deteksi malware
                yang dapat membantu pengguna dalam mengidentifikasi apakah file
                dengan ekstensi .exe pada perangkat Windows adalah file yang tidak
                berbahaya (benign) atau berpotensi berbahaya (malicious). Dengan
                menggunakan algoritma sederhana, kami menganalisis karakteristik
                file .exe dan memberikan hasil deteksi yang cukup akurat dan dapat
                diandalkan. Dengan demikian, Anda dapat dengan mudah menghindari
                potensi ancaman dan menjaga keamanan perangkat Anda dengan lebih
                baik.
            </p>
        </div>

    </div>
</section><!-- End About Section -->

<!-- ===== Services Section ===== -->
<section id="services" class="services section-bg">
    <div class="container" data-aos="fade-up">

        <div class="section-title">
            <h2>Services</h2>
            <form action="/process" method="POST"
            enctype="multipart/form-data">
                <input type="file" name="file" class="drop-file
                scrollto">
                <span style="color: #47b2e4;"><b><i>*.exe file
                only</i></b></span>
                <button type="submit" class="detecting-
                process">Detecting Process</button>
            </form>
        </div>
    </div>

</div>
</section><!-- End Services Section -->

<!-- ===== Get to Know Me Section ===== -->
<section id="me" class="me section-bg justify-content-center">
    <div class="container" data-aos="fade-up">

        <div class="section-title">
            <h2>Get to Know Me</h2>
        </div>

        <div class="row">
            <div class="me" data-aos="zoom-in" data-aos-delay="100">
                <div class="member d-flex align-items-center justify-
                content-center">
                    <div class="pic"></div>
                    <div class="member-info">
                        <h4>Annisa Arrumaisha Siregar</h4>
                        <span>NIM. 061940351944</span>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>

```

```

        <ul class="social mb-0 list-inline mt-3">
            <li class="list-inline-item">
                <a target="_blank"
href="https://github.com/annisaarrumaisha/" class="social-link">
                    <i class="fa fa-github"></i>
                </a>
            </li>
            <li class="list-inline-item">
                <a target="_blank"
href="https://www.instagram.com/arrumaiiisha/" class="social-
link">
                    <i class="fa fa-instagram"></i>
                </a>
            </li>
            <li class="list-inline-item">
                <a target="_blank"
href="https://www.linkedin.com/in/annisaarrumaishasiregar"
class="social-link">
                    <i class="fa fa-linkedin"></i>
                </a>
            </li>
        </ul>
    </div>
</div>
</div>
</section><!-- End Get to Know Me Section -->

<div id="preloader"></div>
<a href="#" class="back-to-top d-flex align-items-center
justify-content-center"><i class="bi bi-arrow-up-short"></i></a>

<!-- Vendor JS Files -->
<script src="../../static/vendor/aos/aos.js"></script>
<script
src="../../static/vendor/bootstrap/js/bootstrap.bundle.min.js"></scri
pt>
<script
src="../../static/vendor/glightbox/js/glightbox.min.js"></script>
<script src="../../static/vendor/isotope-
layout/isotope.pkgd.min.js"></script>
<script src="../../static/vendor/swiper/swiper-
bundle.min.js"></script>
<script
src="../../static/vendor/waypoints/noframework.waypoints.js"></script
>
<script src="../../static/vendor/php-email-
form/validate.js"></script>

<!-- Template Main JS File -->
<script src="../../static/js/main.js"></script>

```

```
</body>
```

```
</html>
```

2.3.process.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0"
name="viewport">

  <title>Detection Process</title>
  <meta content="" name="description">
  <meta content="" name="keywords">

  <!-- stylesheets -->
  <!-- bootstrap css -->
  <link rel="stylesheet" type="text/css"
href="css/bootstrap.min.css">
  <!-- Responsive-->
  <link rel="stylesheet" href="css/responsive.css">
  <!-- Tweaks for older IEs-->
  <link rel="stylesheet"
href="https://netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-
awesome.css">
  <!-- Favicons -->
  <link href="../static/img/logo-1.png" rel="icon">
  <!-- Google Fonts -->
  <link
href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,4
00,400i,600,600i,700,700i|Jost:300,300i,400,400i,500,500i,600,600i
,700,700i|Poppins:300,300i,400,400i,500,500i,600,600i,700,700i"
rel="stylesheet">
  <!-- Vendor CSS Files -->
  <link href="../static/vendor/aos/aos.css" rel="stylesheet">
  <link href="../static/vendor/bootstrap/css/bootstrap.min.css"
rel="stylesheet">
  <link href="../static/vendor/bootstrap-icons/bootstrap-
icons.css" rel="stylesheet">
  <link href="../static/vendor/boxicons/css/boxicons.min.css"
rel="stylesheet">
  <link href="../static/vendor/glightbox/css/glightbox.min.css"
rel="stylesheet">
  <link href="../static/vendor/remixicon/remixicon.css"
rel="stylesheet">
  <link href="../static/vendor/swiper/swiper-bundle.min.css"
rel="stylesheet">
  <!-- Template Main CSS File -->
  <link href="../static/css/style.css" rel="stylesheet">

  <style>
    .spinner-border {
```

```

        margin-top: 50px;
    }
</style>
</head>

<body>

    <!-- ===== Header ===== -->
    <header id="header" class="fixed-top ">
        <div class="container d-flex align-items-center">

            <p href="index.html" class="logo me-auto">
MALDET</p>

            <nav id="navbar" class="navbar">
                <ul>
                    <li><a class="home scrollto" href="home">Home</a></li>
                </ul>
                <i class="bi bi-list mobile-nav-toggle"></i>
            </nav><!-- .navbar -->

        </div>
    </header><!-- End Header -->

    <!-- ===== Detection Process Section ===== -->
    <section id="Detection Process" class="process process-bg">
        <div class="container" data-aos="fade-up">
            <div class="d-flex align-items-center justify-content-center">
                {% if hasil == 'Malware' %}
                
                {% else %}
                
                {% endif %}
            </div>

            <div class="d-flex align-items-center justify-content-center">
                <p class="process-hasil" style="text-align: center;">{{
hasil }}</p>
            </div>

            {% if hasil == 'Malware' %}
            <div class="row m-4 p-2">
                <div class="col-xl-3 col-md-6 d-flex align-items-stretch"
data-aos="zoom-in" data-aos-delay="100">
                    <div class="icon-box">
                        <h4><a href="">Hapus file secara permanen</a></h4>
                        <p class="process-p">Segera hapus file tersebut secara
permanen dari sistem Anda. Jangan menjalankannya atau membuka file
tersebut karena dapat menyebabkan kerusakan pada sistem atau
pencurian data.</p>
                    </div>
                </div>
            </div>
        </div>
    </section>

```

```

<div class="col-xl-3 col-md-6 d-flex align-items-stretch mt-4 mt-md-0" data-aos="zoom-in" data-aos-delay="200">
  <div class="icon-box">
    <h4><a href="">Ganti kata sandi</a></h4>
    <p class="process-p">Setelah menghapus malware, disarankan untuk mengganti semua kata sandi yang digunakan di laptop Anda. Ini termasuk kata sandi untuk akun sistem operasi, akun email, jejaring sosial, perbankan online, dan situs web lainnya. Gunakan kata sandi yang kuat dan berbeda untuk setiap akun.</p>
  </div>
</div>

<div class="col-xl-3 col-md-6 d-flex align-items-stretch mt-4 mt-xl-0" data-aos="zoom-in" data-aos-delay="300">
  <div class="icon-box">
    <h4><a href="">Backup data</a></h4>
    <p class="process-p">Jika Anda memiliki data penting di laptop, segera lakukan backup ke perangkat eksternal atau cloud storage yang aman. Ini akan membantu Anda memulihkan data jika terjadi kerusakan atau kehilangan akibat serangan malware.</p>
  </div>
</div>

<div class="col-xl-3 col-md-6 d-flex align-items-stretch mt-4 mt-xl-0" data-aos="zoom-in" data-aos-delay="400">
  <div class="icon-box">
    <h4><a href="">Perbarui perangkat lunak dan lindungi diri</a></h4>
    <p class="process-p">Setelah menghapus malware, pastikan untuk memperbarui perangkat lunak antivirus Anda ke versi terbaru dan pastikan perangkat lunak lainnya juga diperbarui. Selain itu, perhatikan praktik keamanan yang baik seperti tidak mengklik tautan atau lampiran yang mencurigakan, memperbarui sistem operasi, menggunakan firewall, dan melakukan pemindaian rutin untuk mencegah infeksi yang lebih lanjut.</p>
  </div>
</div>
{% endif %}
</div>
</section>
<!-- End Detection Process Section -->

<div id="preloader"></div>
<a href="#" class="back-to-top d-flex align-items-center justify-content-center"><i class="bi bi-arrow-up-short"></i></a>

<!-- Vendor JS Files -->
<script src="../static/vendor/aos/aos.js"></script>
<script src="../static/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
<script src="../static/vendor/glightbox/js/glightbox.min.js"></script>
<script src="../static/vendor/isotope-

```

```

layout/isotope.pkgd.min.js"></script>
  <script src="../../static/vendor/swiper/swiper-
bundle.min.js"></script>
  <script
src="../../static/vendor/waypoints/noframework.waypoints.js"></script
>
  <script src="../../static/vendor/php-email-
form/validate.js"></script>

  <!-- Template Main JS File -->
  <script src="../../static/js/main.js"></script>

</body>
</html>

```

2.4.style.css

```

/*-----
# General
-----*/
body {
  font-family: "Open Sans", sans-serif;
  color: #444444;
}

p{
  color: #fff;
}

a {
  color: #47b2e4;
  text-decoration: none;
}

h1,
h2,
h3,
h4,
h5,
h6 {
  font-family: "Jost", sans-serif;
}

/*-----
# Preloader
-----*/
#preloader {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
}

```

```

z-index: 9999;
overflow: hidden;
background: #37517e;
}

@keyframes animate-preloader {
  0% {
    transform: rotate(0deg);
  }

  100% {
    transform: rotate(360deg);
  }
}

/*-----
# Back to top button
-----*/

.back-to-top {
  position: fixed;
  visibility: hidden;
  opacity: 0;
  right: 15px;
  bottom: 15px;
  z-index: 996;
  background: #47b2e4;
  width: 40px;
  height: 40px;
  border-radius: 50px;
  transition: all 0.4s;
}

.back-to-top i {
  font-size: 24px;
  color: #fff;
  line-height: 0;
}

.back-to-top:hover {
  background: #6bc1e9;
  color: #fff;
}

.back-to-top.active {
  visibility: visible;
  opacity: 1;
}

/*-----
# Header
-----*/

#header {
  transition: all 0.5s;
  z-index: 997;
  padding: 15px 0;
}

```

```

#header.header-scrolled,
#header.header-inner-pages {
  background: rgba(40, 58, 90, 0.9);
}

#header .logo {
  font-size: 30px;
  margin: 0;
  padding: 0;
  line-height: 1;
  font-weight: 500;
  letter-spacing: 2px;
  text-transform: uppercase;
}

#header .logo a {
  color: #fff;
}

#header .logo img {
  max-height: 40px;
}

/*-----
# Navigation Menu
-----*/
/**
 * Desktop Navigation
 */
.navbar {
  padding: 0;
}

.navbar ul {
  margin: 0;
  padding: 0;
  display: flex;
  list-style: none;
  align-items: center;
}

.navbar li {
  position: relative;
}

.navbar a,
.navbar a:focus {
  display: flex;
  align-items: center;
  justify-content: space-between;
  padding: 10px 0 10px 30px;
  font-size: 15px;
  font-weight: 500;
  color: #fff;
  white-space: nowrap;
}

```

```
    transition: 0.3s;
}

.navbar a i,
.navbar a:focus i {
    font-size: 12px;
    line-height: 0;
    margin-left: 5px;
}

.navbar a:hover,
.navbar .active,
.navbar .active:focus,
.navbar li:hover>a {
    color: #47b2e4;
}

.navbar .getstarted,
.navbar .getstarted:focus {
    padding: 8px 20px;
    margin-left: 30px;
    border-radius: 50px;
    color: #fff;
    font-size: 14px;
    border: 2px solid #47b2e4;
    font-weight: 600;
}

.navbar .getstarted:hover,
.navbar .getstarted:focus:hover {
    color: #fff;
    background: #31a9e1;
}

.navbar .dropdown ul {
    display: block;
    position: absolute;
    left: 14px;
    top: calc(100% + 30px);
    margin: 0;
    padding: 10px 0;
    z-index: 99;
    opacity: 0;
    visibility: hidden;
    background: #fff;
    box-shadow: 0px 0px 30px rgba(127, 137, 161, 0.25);
    transition: 0.3s;
    border-radius: 4px;
}

.navbar .dropdown ul li {
    min-width: 200px;
}

.navbar .dropdown ul a {
    padding: 10px 20px;
```

```
font-size: 14px;
text-transform: none;
font-weight: 500;
color: #0c3c53;
}

.navbar .dropdown ul a i {
font-size: 12px;
}

.navbar .dropdown ul a:hover,
.navbar .dropdown ul .active:hover,
.navbar .dropdown ul li: hover>a {
color: #47b2e4;
}

.navbar .dropdown: hover>ul {
opacity: 1;
top: 100%;
visibility: visible;
}

.navbar .dropdown .dropdown ul {
top: 0;
left: calc(100% - 30px);
visibility: hidden;
}

.navbar .dropdown .dropdown: hover>ul {
opacity: 1;
top: 0;
left: 100%;
visibility: visible;
}

@media (max-width: 1366px) {
.navbar .dropdown .dropdown ul {
left: -90%;
}

.navbar .dropdown .dropdown: hover>ul {
left: -100%;
}
}

/**
 * Mobile Navigation
 */
.mobile-nav-toggle {
color: #fff;
font-size: 28px;
cursor: pointer;
display: none;
line-height: 0;
transition: 0.5s;
}
```

```
.mobile-nav-toggle.bi-x {
  color: #fff;
}

@media (max-width: 991px) {
  .mobile-nav-toggle {
    display: block;
  }

  .navbar ul {
    display: none;
  }
}

.navbar-mobile {
  position: fixed;
  overflow: hidden;
  top: 0;
  right: 0;
  left: 0;
  bottom: 0;
  background: rgba(40, 58, 90, 0.9);
  transition: 0.3s;
  z-index: 999;
}

.navbar-mobile .mobile-nav-toggle {
  position: absolute;
  top: 15px;
  right: 15px;
}

.navbar-mobile ul {
  display: block;
  position: absolute;
  top: 55px;
  right: 15px;
  bottom: 15px;
  left: 15px;
  padding: 10px 0;
  border-radius: 10px;
  background-color: #fff;
  overflow-y: auto;
  transition: 0.3s;
}

.navbar-mobile a,
.navbar-mobile a:focus {
  padding: 10px 20px;
  font-size: 15px;
  color: #37517e;
}

.navbar-mobile a:hover,
.navbar-mobile .active,
```

```

.navbar-mobile li:hover>a {
  color: #47b2e4;
}

.navbar-mobile .getstarted,
.navbar-mobile .getstarted:focus {
  margin: 15px;
  color: #37517e;
}

.navbar-mobile .dropdown ul {
  position: static;
  display: none;
  margin: 10px 20px;
  padding: 10px 0;
  z-index: 99;
  opacity: 1;
  visibility: visible;
  background: #fff;
  box-shadow: 0px 0px 30px rgba(127, 137, 161, 0.25);
}

.navbar-mobile .dropdown ul li {
  min-width: 200px;
}

.navbar-mobile .dropdown ul a {
  padding: 10px 20px;
}

.navbar-mobile .dropdown ul a i {
  font-size: 12px;
}

.navbar-mobile .dropdown ul a:hover,
.navbar-mobile .dropdown ul .active:hover,
.navbar-mobile .dropdown ul li:hover>a {
  color: #47b2e4;
}

.navbar-mobile .dropdown>.dropdown-active {
  display: block;
}

/*-----
# Home Section
-----*/

#home {
  width: 100%;
  height: 80vh;
  background: #37517e;
}

#home .container {
  padding-top: 72px;
}

```

```
#home h1 {
  margin: 0 0 10px 0;
  font-size: 48px;
  font-weight: 700;
  line-height: 56px;
  color: #fff;
}

#home h2 {
  color: rgba(255, 255, 255, 0.6);
  margin-bottom: 50px;
  font-size: 24px;
}

#home p {
  text-align: justify;
  text-justify: inter-word;
}

#home .btn-get-started {
  font-family: "Jost", sans-serif;
  font-weight: 500;
  font-size: 16px;
  letter-spacing: 1px;
  display: inline-block;
  padding: 10px 28px 11px 28px;
  border-radius: 50px;
  transition: 0.5s;
  margin: 10px 0 0 0;
  color: #fff;
  background: #47b2e4;
}

#home .btn-get-started:hover {
  background: #209dd8;
}

#home .animated {
  animation: up-down 2s ease-in-out infinite alternate-reverse
  both;
}

@media (max-width: 991px) {
  #home {
    height: 100vh;
    text-align: center;
  }

  #home .animated {
    animation: none;
  }

  #home .home-img {
    text-align: center;
  }
}
```

```

#home .hero-img img {
  width: 50%;
}
}

@media (max-width: 768px) {
#home h1 {
  font-size: 28px;
  line-height: 36px;
}

#home h2 {
  font-size: 18px;
  line-height: 24px;
  margin-bottom: 30px;
}

#home .home-img img {
  width: 70%;
}
}

@media (max-width: 575px) {
#home .home-img img {
  width: 80%;
}

#home .btn-get-started {
  font-size: 16px;
  padding: 10px 24px 11px 24px;
}
}

@keyframes up-down {
  0% {
    transform: translateY(10px);
  }

  100% {
    transform: translateY(-10px);
  }
}

/*-----*/
# Sections General
/*-----*/

section {
  padding: 60px 0;
  /* overflow: hidden; */
}

.section-bg {
  background-color: #f3f5fa;
}

```

```
.section-title {
  text-align: center;
  padding-bottom: 30px;
}

.section-title h2 {
  font-size: 32px;
  font-weight: bold;
  text-transform: uppercase;
  margin-bottom: 20px;
  padding-bottom: 20px;
  position: relative;
  color: #37517e;
}

.section-title h2::before {
  content: "";
  position: absolute;
  display: block;
  width: 120px;
  height: 1px;
  background: #ddd;
  bottom: 1px;
  left: calc(50% - 60px);
}

.section-title h2::after {
  content: "";
  position: absolute;
  display: block;
  width: 40px;
  height: 3px;
  background: #47b2e4;
  bottom: 0;
  left: calc(50% - 20px);
}

.section-title p {
  margin-bottom: 0;
}

/*-----
# About
-----*/

.about .content h3 {
  font-weight: 600;
  font-size: 26px;
}

.about .content ul {
  list-style: none;
  padding: 0;
}

.about .content ul li {
  padding-left: 28px;
}
```

```

    position: relative;
}

.about .content ul li+li {
    margin-top: 10px;
}

.about .content ul i {
    position: absolute;
    left: 0;
    top: 2px;
    font-size: 20px;
    color: #47b2e4;
    line-height: 1;
}

.about .content p{
    color : #444444;
    text-align: justify;
    text-justify: inter-word;
}

.about .content p:last-child {
    margin-bottom: 0;
}

.about .content .btn-learn-more {
    font-family: "Poppins", sans-serif;
    font-weight: 500;
    font-size: 14px;
    letter-spacing: 1px;
    display: inline-block;
    padding: 12px 32px;
    border-radius: 4px;
    transition: 0.3s;
    line-height: 1;
    color: #47b2e4;
    animation-delay: 0.8s;
    margin-top: 6px;
    border: 2px solid #47b2e4;
}

.about .content .btn-learn-more:hover {
    background: #47b2e4;
    color: #fff;
    text-decoration: none;
}

/*-----
# Services
-----*/

#services .drop-file,
#services .drop-file:focus {
    padding: 8px 20px;
    margin-left: 30px;
    border-radius: 50px;
}

```

```
    color: #47b2e4;
    font-size: 14px;
    border: 2px solid #47b2e4;
    font-weight: 600;
}

#services .drop-file:hover,
#services .drop-file:focus:hover {
    color: #fff;
    background: #31a9e1;
}

#services .detecting-process,
#services .detecting-process:focus {
    padding: 8px 20px;
    margin-left: 30px;
    border-radius: 50px;
    color: #ff0000;
    font-size: 14px;
    border: 2px solid #ff0000;
    font-weight: 600;
}

#services .detecting-process:hover,
#services .detecting-process:focus:hover {
    color: #fff;
    background: #ff0000;
}

.file-description {
    margin-left: 10px;
    color: #47b2e4;
    font-weight: bold;
}

.services .icon-box {
    box-shadow: 0px 0 25px 0 rgba(0, 0, 0, 0.1);
    padding: 50px 30px;
    transition: all ease-in-out 0.4s;
    background: #fff;
}

.services .icon-box .icon {
    margin-bottom: 10px;
}

.services .icon-box .icon i {
    color: #47b2e4;
    font-size: 36px;
    transition: 0.3s;
}

.services .icon-box h4 {
    font-weight: 500;
    margin-bottom: 15px;
}
```

```

    font-size: 24px;
}

.services .icon-box h4 a {
    color: #37517e;
    transition: ease-in-out 0.3s;
}

.services .icon-box p {
    line-height: 24px;
    font-size: 14px;
    margin-bottom: 0;
}

.services .icon-box:hover {
    transform: translateY(-10px);
}

.services .icon-box:hover h4 a {
    color: #47b2e4;
}

/*-----
# Get to Know Me
-----*/

.me{
    justify-content: center;
}

.me.member {
    position: relative;
    box-shadow: 0px 2px 15px rgba(0, 0, 0, 0.1);
    padding: 30px;
    border-radius: 5px;
    background: #fff;
    transition: 0.5s;
    height: 100%;
}

.me .member .pic {
    overflow: hidden;
    width: 180px;
    border-radius: 50%;
}

.me .member .pic img {
    transition: ease-in-out 0.3s;
}

.me .member:hover {
    transform: translateY(-10px);
}

.me .member .member-info {
    padding-left: 30px;
}

```

```
.me .member h4 {
  font-weight: 700;
  margin-bottom: 5px;
  font-size: 20px;
  color: #37517e;
}

.me .member span {
  display: block;
  font-size: 15px;
  padding-bottom: 10px;
  position: relative;
  font-weight: 500;
}

.me .member span::after {
  content: "";
  position: absolute;
  display: block;
  width: 50px;
  height: 1px;
  background: #cbd6e9;
  bottom: 0;
  left: 0;
}

.me .member p {
  margin: 10px 0 0 0;
  font-size: 14px;
}

.me .member .social {
  margin-top: 12px;
  display: flex;
  align-items: center;
  justify-content: flex-start;
}

.me .member .social a {
  transition: ease-in-out 0.3s;
  display: flex;
  align-items: center;
  justify-content: center;
  border-radius: 50px;
  width: 32px;
  height: 32px;
  background: #eff2f8;
}

.me .member .social a i {
  color: #37517e;
  font-size: 16px;
  margin: 0 2px;
}

.me .member .social a:hover {
```

```
background: #47b2e4;
}

.me .member .social a:hover i {
  color: #fff;
}

.me .member .social a+a {
  margin-left: 8px;
}

/*-----
# Navigation Home
-----*/

.navbar .home,
.navbar .home:focus {
  padding: 8px 20px;
  margin-left: 30px;
  border-radius: 50px;
  color: #fff;
  font-size: 14px;
  border: 2px solid #47b2e4;
  font-weight: 600;
}

.navbar .home:hover,
.navbar .home:focus:hover {
  color: #fff;
  background: #31a9e1;
}

/*-----
# Detection Process
-----*/

.process {
  padding: 60px 0;
  height: 1020px;
  /* overflow: hidden; */
}

.process-bg {
  background-color: #37517e;
}

.process .icon-box {
  box-shadow: 0px 0 25px 0 rgba(0, 0, 0, 0.1);
  padding: 50px 30px;
  transition: all ease-in-out 0.4s;
  background: #fff;
}

.process .icon-box .icon {
  margin-bottom: 10px;
}

.process .icon-box .icon i {
```

```

    color: #47b2e4;
    font-size: 36px;
    transition: 0.3s;
  }

  .process .icon-box h4 {
    font-weight: 500;
    margin-bottom: 15px;
    font-size: 24px;
    text-align: justify;
  }

  .process .icon-box h4 a {
    color: #37517e;
    transition: ease-in-out 0.3s;
  }

  .process .icon-box p {
    line-height: 24px;
    font-size: 14px;
    margin-bottom: 0;
  }

  .process .icon-box:hover {
    transform: translateY(-10px);
  }

  .process .icon-box:hover h4 a {
    color: #47b2e4;
  }

  .process-hasil{
    font-size: 32px;
    font-weight: bold;
    color: #fff
  }

  .process-p {
    text-align: justify;
    color: #000;
  }

```

2.5.main.js

```

(function() {
  "use strict";

  /**
   * Easy selector helper function
   */
  const select = (el, all = false) => {
    el = el.trim()
    if (all) {

```

```

    return [...document.querySelectorAll(el)]
  } else {
    return document.querySelector(el)
  }
}

/**
 * Easy event listener function
 */
const on = (type, el, listener, all = false) => {
  let selectEl = select(el, all)
  if (selectEl) {
    if (all) {
      selectEl.forEach(e => e.addEventListener(type, listener))
    } else {
      selectEl.addEventListener(type, listener)
    }
  }
}

/**
 * Easy on scroll event listener
 */
const onscroll = (el, listener) => {
  el.addEventListener('scroll', listener)
}

/**
 * Navbar links active state on scroll
 */
let navbarlinks = select('#navbar .scrollto', true)
const navbarlinksActive = () => {
  let position = window.scrollY + 200
  navbarlinks.forEach(navbarlink => {
    if (!navbarlink.hash) return
    let section = select(navbarlink.hash)
    if (!section) return
    if (position >= section.offsetTop && position <=
(section.offsetTop + section.offsetHeight)) {
      navbarlink.classList.add('active')
    } else {
      navbarlink.classList.remove('active')
    }
  })
}
window.addEventListener('load', navbarlinksActive)
onscroll(document, navbarlinksActive)

/**
 * Scrolls to an element with header offset
 */
const scrollto = (el) => {
  let header = select('#header')
  let offset = header.offsetHeight

  let elementPos = select(el).offsetTop

```

```

window.scrollTo({
  top: elementPos - offset,
  behavior: 'smooth'
})
}

/**
 * Toggle .header-scrolled class to #header when page is
scrolled
 */
let selectHeader = select('#header')
if (selectHeader) {
  const headerScrolled = () => {
    if (window.scrollY > 100) {
      selectHeader.classList.add('header-scrolled')
    } else {
      selectHeader.classList.remove('header-scrolled')
    }
  }
  window.addEventListener('load', headerScrolled)
  onscroll(document, headerScrolled)
}

/**
 * Back to top button
 */
let backtotop = select('.back-to-top')
if (backtotop) {
  const toggleBacktotop = () => {
    if (window.scrollY > 100) {
      backtotop.classList.add('active')
    } else {
      backtotop.classList.remove('active')
    }
  }
  window.addEventListener('load', toggleBacktotop)
  onscroll(document, toggleBacktotop)
}

/**
 * Mobile nav toggle
 */
on('click', '.mobile-nav-toggle', function(e) {
  select('#navbar').classList.toggle('navbar-mobile')
  this.classList.toggle('bi-list')
  this.classList.toggle('bi-x')
})

/**
 * Mobile nav dropdowns activate
 */
on('click', '.navbar .dropdown > a', function(e) {
  if (select('#navbar').classList.contains('navbar-mobile')) {
    e.preventDefault()
    this.nextElementSibling.classList.toggle('dropdown-active')
  }
}

```

```

}, true)

/**
 * Scroll with ofset on links with a class name .scrollto
 */
on('click', '.scrollto', function(e) {
  if (select(this.hash)) {
    e.preventDefault()

    let navbar = select('#navbar')
    if (navbar.classList.contains('navbar-mobile')) {
      navbar.classList.remove('navbar-mobile')
      let navbarToggle = select('.mobile-nav-toggle')
      navbarToggle.classList.toggle('bi-list')
      navbarToggle.classList.toggle('bi-x')
    }
    scrollTo(this.hash)
  }
}, true)

/**
 * Scroll with ofset on page load with hash links in the url
 */
window.addEventListener('load', () => {
  if (window.location.hash) {
    if (select(window.location.hash)) {
      scrollTo(window.location.hash)
    }
  }
});

/**
 * Preloader
 */
let preloader = select('#preloader');
if (preloader) {
  window.addEventListener('load', () => {
    preloader.remove()
  });
}

/**
 * Initiate glightbox
 */
const glightbox = GLightbox({
  selector: '.glightbox'
});

/**
 * Skills animation
 */
let skillsContent = select('.skills-content');
if (skillsContent) {
  new Waypoint({
    element: skillsContent,
    offset: '80%',

```

```

        handler: function(direction) {
            let progress = select('.progress .progress-bar', true);
            progress.forEach((el) => {
                el.style.width = el.getAttribute('aria-valuenow') + '%';
            });
        }
    })
}

/**
 * Portfolio isotope and filter
 */
window.addEventListener('load', () => {
    let portfolioContainer = select('.portfolio-container');
    if (portfolioContainer) {
        let portfolioIsotope = new Isotope(portfolioContainer, {
            itemSelector: '.portfolio-item'
        });

        let portfolioFilters = select('#portfolio-filters li', true);

        on('click', '#portfolio-filters li', function(e) {
            e.preventDefault();
            portfolioFilters.forEach(function(el) {
                el.classList.remove('filter-active');
            });
            this.classList.add('filter-active');

            portfolioIsotope.arrange({
                filter: this.getAttribute('data-filter')
            });
            portfolioIsotope.on('arrangeComplete', function() {
                AOS.refresh()
            });
        }, true);
    }
});

/**
 * Initiate portfolio lightbox
 */
const portfolioLightbox = GLightbox({
    selector: '.portfolio-lightbox'
});

/**
 * Portfolio details slider
 */
new Swiper('.portfolio-details-slider', {
    speed: 400,
    loop: true,
    autoplay: {
        delay: 5000,
        disableOnInteraction: false
    },
},

```

```
    pagination: {
      el: '.swiper-pagination',
      type: 'bullets',
      clickable: true
    }
  });

  /**
   * Animation on scroll
   */
  window.addEventListener('load', () => {
    AOS.init({
      duration: 1000,
      easing: "ease-in-out",
      once: true,
      mirror: false
    });
  });
})()
```