Dissertation

Multi-Model Deep Learning: Transformative Applications in Disaster Mitigation, Drug Discovery, and Autonomous Systems

(マルチモデル深層学習の減災、創薬、自律システムへの応用に関する研究)

Graduate School of
Natural Science & Technology
Kanazawa University

DIVISION OF ELECTRICAL ENGINEERING
AND COMPUTER SCIENCE

Student ID: 2124042002

Name: Willy Dharmawan

Chief Advisor: Hidetaka Nambo

Year and Month of Submission: 2024/January

DECLARATION OF ORIGINALITY

This dissertation is submitted in partial fulfillment of the requirements for the award of the degree of Doctor of Engineering at Kanazawa University. While registered as a candidate for the above degree, I have not been registered for any other research award. The results and conclusions embodied in this dissertation are the work of the named candidate and have not been submitted for any other academic award.

Willy Dharmawan

Submission date: January 2024

ABSTRACT

This dissertation explores applying multi-model deep learning to address a triad of real-world problems covering some aspects of SDGs: disaster mitigation, drug discovery, and system control. By harnessing deep learning techniques, this research strives to offer innovative solutions that transcend traditional boundaries and substantially improve each domain.

In disaster mitigation, the research presents a novel multi-model deep learning architecture for enhancing tsunami early warning systems. Concerning the threat of the giant tsunami caused by megathrust earthquakes in the Mentawai island area, we are collaborating with the National Research and Innovation Agency Indonesia (BRIN) to develop tsunami tides prediction in the shallow water area, which has ambient noise level-dependent property. From this case, we develop deep learning using an RNN model that accommodates time series data.

In the case of drug discovery, the study investigates the potential of deep learning to expedite the identification of protein-ligand binding pairs. By amalgamating various data sources of molecular structure and its properties, our preliminary experiments show that a 3D-CNN-based network can identify protein-ligand binding specificity. The current inventions are the pdbind filtered protein-ligand dataset and comprehensive evaluation of the ResNet variant to classify the binary over internal and external datasets.

Additionally, the dissertation delves into applying multi-model deep learning in autonomous system scenarios, especially self-driving cars and surface vehicles. As part of research sustainability from master research, this study uses a combination of CNN, RNN, Reinforcement Learning, and algorithms to accomplish specific tasks.

The outcomes of this research reveal the heuristic performance of its application. By leveraging the versatility of deep neural networks, the dissertation demonstrates the capacity to extract intricate patterns from heterogeneous data sources using multiple preprocessing and create holistic models that outperform traditional methods. The synthesis of insights from tsunami mitigation, drug discovery, and system control highlights the overarching versatility of multi-model deep learning in solving a range of intricate real-world challenges.

ACKNOWLEDGMENTS

Thanks to Allah SWT, who eased all of my work in this Doctoral Program despite many big obstacles things happened in the process.

To my nuclear family members, Wakiyo, Dwi Lestari, Ulfanie, and Kasirah, without whom this document would not exist in its present form.

To Nambo sensei, I am so grateful for your guidance. Sensei's kindness is so limitless. I hope we can always collaborate to create high-quality research.

To the lab-mate members, especially Hamid-san, it is a pleasure to be able to work together despite the short time. I owe you a favor, Hamid-san. I hope we can work together in the future.

To Mext Scholarship, I am grateful for all of the allowances for sponsorship.

To my valuable friends, Mery Diana, Afwan, Erik, Kholqillah, Danu, Hari, Hendra, Jono, Dedi, Firman, and Arbi, I thank you for your help and material and moral support in this Doctoral Program.

To my Vietnamese friend, Loc-san, who helped me share knowledge in drug discovery, I want to say thank you for your help and support.

To all of the support from branch family members, acquaintances at Kanazawa University, and Indonesia friends in the Gakusee Community, I am grateful for all your support.

CONTENTS

LIST OF FIGURES

LIST OF FORMULAS

LIST OF TABLES

DECLARATION OF AUTHORSHIP

Table 1. List of papers, publication status and authorship

Papers	Year	Title	Publication	Author/s
1	2023	Tsunami Tide Prediction in Shallow Water Using Recurrent Neural Networks: Model Implementation in the Indonesia Tsunami Early Warning System	Journal of Reliable Intelligent Environments, 19 Pages, Link	Willy Dharmawan, Mery Diana, Beti Tuntari, I. Made Astawa, Sasono Rahardjo, and Hidetaka Nambo
2	2023	Dynamic Path Planning for Unmanned Surface Vehicles with a Modified Neuronal Genetic Algorithm	Applied System Innovation, No. 6, 109, Pages 19, Link	Nur Hamid, Willy Dharmawan, and Hidetaka Nambo
3	2021	End-to-End Time Distributed Convolution Neural Network Model for Self-Driving Car in Moderate Dense Environment	Jurnal Teknologi Infomasi, Komunikasi dan Elektronika, Vol. 2, No.1, pp.8-13, Link	Willy Dharmawan, and Hidetaka Nambo
4	2022	Protein-Ligand Pair Interaction Prediction Using Wide Resnet For Virtual Drug Screening	Proceedings of the 22nd APIEMS, ID: OTH023	Willy Dharmawan, Pham Thi Loc, Nur Hamid, and Hidetaka Nambo

5	2022	Autonomous Evacuation Boat in Dynamic Flood Disaster Environment	International Conference on Advanced Computer Science and Information Systems, pp. 117-122, doi: 10.1109/ICACSIS56558.2022.9923446	Nur Hamid, Willy Dharmawan, and Hidetaka Nambo
6	2023	Neural Network-based Genetic Algorithm for Autonomous Boat Pathfinding	2022 International Conference on Advanced Computer Science and Information Systems, pp. 497-502, doi: 10.1109/ICKII58656.2023.10332606	Nur Hamid, Willy Dharmawan, and Hidetaka Nambo

Table 2. List of workshop paper, publication status and authorship

Papers	**Year**	**Title**	**Publication**	**Author/s**
1	2023	Developing self-driving car simulation wrapper for deep learning purposes	IEEJ Electronic Library, Link	Willy Dharmawan, Hidetaka Nambo.

I hereby certify that the authorship, as stated above, is an accurate record of the papers presented as part of this dissertation.

Willy Dharmawan
January 2024

CHAPTER 1: INTRODUCTION

1.1 Background

Artificial intelligence (AI), particularly machine learning, stands as a testament to the collaborative nature of modern research. This groundbreaking field operates at the intersection of computer science, mathematics, neuroscience, and other domains, giving rise to an interdisciplinary research landscape **[1]**. The quest for AI advancements is not only propelled by theoretical frameworks but is deeply rooted in the utilization of external data. Improving the performance of traditional machine learning, Deep Learning can take advantage of a diverse range of datasets, harnessing the power of real-world information to enhance the robustness and applicability of their models **[2,3]**. The symbiotic relationship between interdisciplinary collaboration and the incorporation of external data establishes a solid foundation for the evolution of AI, paving the way for innovative solutions to complex problems and pushing the boundaries of what is achievable in the realm of artificial intelligence **[4]**.

Deep learning has demonstrated profound efficacy in addressing and solving a myriad of real-world problems **[1]**. Many corporations, including Google, Microsoft, Nokia, etc., study it actively as it can provide significant results in different classification and regression problems and datasets **[5]**. One of the primary strengths lies in its ability to automatically extract intricate patterns and representations from vast and diverse datasets, enabling the development of models that can tackle complex challenges across various domains.

The domains of science and engineering are currently experiencing a profound shift towards transformative, transdisciplinary, and translational research methodologies, capturing substantial and escalating attention from researchers and scholars alike. This evolving paradigm signifies a departure from conventional, siloed approaches as the scientific community recognizes the transformative potential that lies at the intersection of diverse disciplines. Some areas **[5]** are sustainability research **[6]**, translational public health and translational medicine **[7,8]**, biomedical research **[9]**, and transformative digitalization **[5]**.

In this dissertation report, we will go through transdisciplinary artificial intelligence research with the goal of high-quality scientific research and application in artificial intelligence, which highlights the importance of our research and raises the research's impact. Some of our works collaborate with several government institutions and Universities, such as Infrastructure Technology Centers Ports and Coastal Dynamics, BPPT (Assessment and Application of Technology Research Organization)/BRIN (National Research and Innovation Agency), Kumamoto University, and Haiphong University of Medicine and Pharmacy Vietnam.

Our research foundation (refer to **Fig. 1.1**) is comprised of development, empowerment, and sustainability communities and partnerships, which align with the United Nations' SDGs (sustainable development goals). This modified methodological foundation framework **[10]** (**Fig. 1.1**) is designed to cover the sustainability of the research and push the transdisciplinary AI research progress toward both conceptual and practical.

1.2 Research Questions

Despite the variety of deep learning model implementations on various transdisciplinary knowledge, deep learning has some problems corresponding to the data type configuration of the input model and prediction. Based on the case presented in this dissertation, we can define our problem:

a. Sensor type input data (single array data) for regression problem (**Paper 1**: Tsunami Tide Prediction in Shallow Water Using Recurrent Neural Networks: Model Implementation in the Indonesia Tsunami Early Warning System).
    - Using input-output (Lookback-Prediction) pair model configuration, can the model predict the tide and tsunami spike through data pattern learning?
    - The primary issues of tide model prediction are prediction error and prediction time. Will the model be reliable in training and predicting processes denoted by Mean Square Error and Time elapsed?

- The main problem with our tide dataset is that it does not contain tsunami tide. This problem is caused by the unavailability of tsunami data in shallow water, which affects the output prediction of tides. Will this input data limitation have insufficient representability?

b. Multimodal sensor data (images, input direction state, speed limit, current speed, traffic light state) (**Paper 3**: End-to-End Time Distributed Convolution Neural Network Model for Self-Driving Car in Moderate Dense Environment)
   - The empirical approach implemented by Nvidia shows a great system performance **[11]** that can achieve 90% autonomy value. Nevertheless, we believe that in a more complex environment, steering angle alone is not enough for vehicle control. Lack of control, such as throttling and braking systems, will limit the potential application of the model. How do we solve this limitation?
   - Many factors in the self-driving car environment cannot be reflected solely through a single front-view camera sensor. How do compact parameters define the state of car driving?
   - The End-to-End temporal-based setup problem is an issue regarding how to set a temporal model on time steps. If we try to use a standard dense layer sequentially, the weights and biases might be changed, and it makes the output flattened with each time step mixed. How do we get the output layer separately by time steps?

c. Three-dimensional grid structures of protein and each feature definition (**Paper 4**: Protein-Ligand Pair Interaction Prediction Using Wide Resnet For Virtual Drug Screening)
   - What kind of features should be added along the grid structures data to represent the model?
   - What type of deep learning model is used for capturing these binding pocket data structures?
   - Will the proposed model be robust when tested outside of the reference dataset?

1.3 Research Objectives

For each paper presented in this dissertation, we can define the aim of our research as follows.

Paper 1: Tsunami Tide Prediction in Shallow Water Using Recurrent Neural Networks: Model Implementation in the Indonesia Tsunami Early Warning System.

- We maximize the lookback-prediction configuration by maximizing the GPU memory availability to define the best-set look-back and prediction parameters. Also, the tide prediction presented in the paper is a univariate time forecasting problem, which is relevant to the efficacy of RNN **[12, 13]**.
- Despite the fact that the RNN training phase cannot take advantage of parallel computation in GPU **[14]**, we experiment with various RNN-based model structures to achieve a closed real-time prediction in the inference time.
- We can consider the tide prediction as a regression case. Therefore, z-score analysis toward variability of the tsunami triggered by earthquake magnitude is required. This analysis is used to evaluate the sensitivity of the current algorithm.

Paper 3: End-to-End Time Distributed Convolution Neural Network Model for Self-Driving Car in Moderate Dense Environment.

- The central aim of this project is to develop an End-to-End DNN-based autonomous car model that can map states and images into necessary car control, such as steering angle, braking, and throttling. Making a set of discrete values from car control that can map the continuous value of the sensor is implemented to create a smoother driving experience.
- We address this compact state parameter with the implementation of a discrete state such as go left, right, go straight, and lane follows and define these into three parameters: output, steering angle, throttle, and brake. This setup is done to achieve end-to-end self-driving cars with multi-modalities.
- We implement a distributed layer to wrap the model. This layer is added to get output separately by time steps.

Paper 4: Protein-Ligand Pair Interaction Prediction Using Wide Resnet For Virtual Drug Screening.

- In this preliminary paper, we simplify the features representation by adopting Ken Dill's lattice protein folding model as features, which classifies the residue of the protein-pair ligand. This definition is the first approach to prove the ability of deep learning models for the case of protein-ligand prediction classification.
- The target of the research is to understand the capability of a model of a stacked convolutional neural network with a residual layer on protein-ligand binding identification.
- We can identify the robustness of the proposed model by comparing the performance on the external dataset. This process is also used to validate our dataset.

1.4 Research Contribution

In this dissertation document, we can summarize the research contribution of each of the papers.

Paper 1: Tsunami Tide Prediction in Shallow Water Using Recurrent Neural Networks: Model Implementation in the Indonesia Tsunami Early Warning System.

- Combination of multistage preprocessing and RNN-based deep-neural network on tide data for solving tide prediction modeling in shallow water cases. This model is intended to get a better tide prediction and human interpretation.
- The suitable tides preprocessing for shallow water cases that can reduce the noise of the tides data and accommodate the neural network input.
- Empirical insight of various RNN models, vanilla RNN, LSTM, and GRU, approach on a case of tsunami detection based on tide prediction.
- Experimentations on defining look-back and forward parameter scenarios on shallow water tides prediction models.
- Z-score analysis toward variability of the synthetic tsunami triggered by earth magnitude. This analysis evaluates the sensitivity of the current model.

- Prototype application implemented in Indonesia Early Warning System Server in Indonesia.

Paper 3: End-to-End Time Distributed Convolution Neural Network Model for Self-Driving Car in Moderate Dense Environment.

- The alternative of the PilotNet Model, an end-to-end time distributed-based model for self-driving cars, is a combination of the CNN FCN-LSTM approach with multiple inputs and states and multiple output predictions. Light context-based spatial using depth-separable convolution is also tested in this research **[15]** as part of range model completion on a self-driving car case.
- Early prototype version of a self-driving car wrapper using the Carla simulator as an engine. This wrapper is developed to help autonomous car developers and programmers implement their deep learning algorithms in the Carla simulator (refer to paper workshop 1**, Table 2**).

Paper 4: Protein-Ligand Pair Interaction Prediction Using Wide Resnet For Virtual Drug Screening.

- Post-processing pdbbind dataset for deep learning.
- A comprehensive evaluation of deep learning configurations experiment on protein-pair ligand prediction.

1.5 Dissertation Overview

While Chapter 2 focuses on the research foundation of this dissertation, Chapters 3, 4, and 5 are about published papers with roles as leading authors. In Chapter 6, we talk about conclusions and recommendations for future works. Finally, Chapter 7 gives information for co-author contribution.

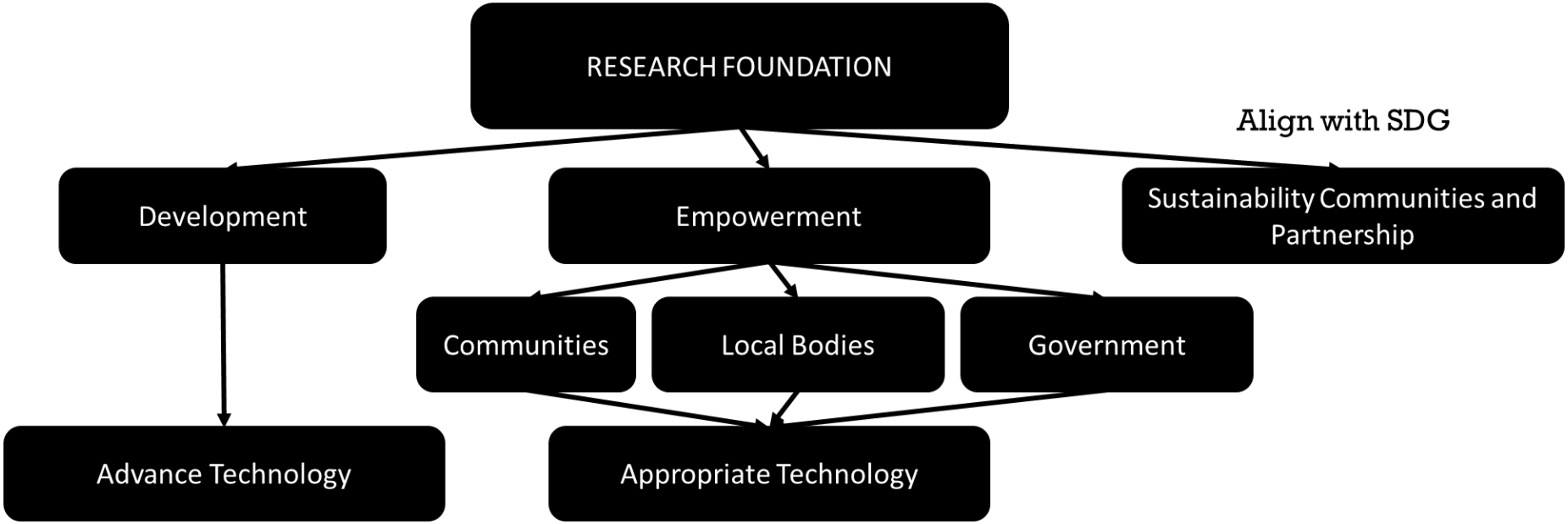


Fig. 1.1 Research Foundation Overview

CHAPTER 2: RESEARCH PHILOSOPHY, APPROACH, AND METHOD

2.1 Research Philosophy and Approach

The baseline of our research agenda always follows the global goals, SDG (Sustainable Development Goals). Our work should contribute to helping solve global problems or advance global aspirations. This approach also refers to a new generation of transdisciplinary, transformational, and translational AI/Data Science (DS) **[4]**. Based on this foundation, we aim to highlight the novelty and importance of our research, as well as raise the research impact.

This SDG's contribution is done in our research by collaborating with some institutions, such as the National Research and Innovation Agency (BRIN), Indonesia Government, Haiphong University of Medicine and Pharmacy Vietnam, and some private companies in the past. The main point of collaboration is to enhance the global partnership for sustainable development.

The research baseline is accomplished by dividing the research schema into two goals: long-term and short-term. The long-term refers to continuous research, which becomes the research scheme's primary focus. This long-term research has a higher achievement and requires perpetual exploration to get a good result. Drug discovery presented in this dissertation is part of the long-term agenda. It takes a lot of time to do a post-processing dataset. However, as part of SDG's good health and well-being, we also should submerge in the drug discovery problem. This starting point will later be used to develop Denovo ligands from specified protein binding on drug discovery. The design ligand will help to identify lead compounds for the specificity of the protein. This result is substantially supportive when the designated lead compounds are toxic and scarce.

The self-driving car research is also part of the long-term research agenda. It is the working continuity from 2018 (Master Degree Research) as part of research sustainability. However, the trend of the research has declined in 2021 **[16]**. Nevertheless, our work contributes to self-driving car-wrappers further developing self-driving car simulation.

In comparison, the short-term study refers to the current problem, which requires faster solutions for specific applications. Tide prediction on shallow water cases is a progressed problem for deploying cable-based tsunameters (CBTs) close to the Sipora shallow water area. It requires faster implementation as a disaster cannot wait at a specific time.

Nevertheless, the heuristic aim is not only part of our research philosophy. A sustainable innovative methodology has also become the central part of our research in deep learning to achieve better performance compared to the previous methods.

2.2 Research Method

All of the developmental work can be separated into five major works: simulation development, data capturing, model training and validation, hyperparameter setup, and testing and evaluation. These works are accomplished using Python programming in Jupyter lab environments.

CHAPTER 3: PAPER 1: Tsunami Tide Prediction in Shallow Water Using Recurrent Neural Networks: Model Implementation in the Indonesia Tsunami Early Warning System

3.1 Foreword

The starting point of this research is based on the threat of the giant tsunami caused by megathrust earthquakes in the Mentawai island area **[20]**. The subduction zone of this area has the characteristic of a low water depth of about 80m, which is considered to have considerable ambient noise. Therefore, it requires Mathematical modeling that can represent dynamics from nonlinear behavior (Time Series Modelling).

Moreover, the unavailability of tsunami datasets in shallow water raises a problem, particularly a classification problem. Therefore, we design the model for the regression case. With the help of Infrastructure Technology Centers Ports and Coastal Dynamics BPPT Indonesia for providing tsunami synthetic, we used z-score to identify the spike of the tides.

3.2 Published Paper

Tsunami Tide Prediction in Shallow Water Using Recurrent Neural Networks: Model Implementation in the Indonesia Tsunami Early Warning System

Willy Dharmawan, Mery Diana, Beti Tuntari, I. Made Astawa, Sasono Rahardjo, and Hidetaka Nambo

3.2.1 Abstract

Near-field tides prediction for tsunami detection in the coastal area is a significant problem of the cable-based tsunami meter system in north Sipora, Indonesia. The problem is caused by its shallow water condition and the unavailability of an applicable model or research for tsunami detection in this area. The problem foundation of shallow water area is its ambient noise level dependent property that requires preprocessing to improve its feature representation. Moreover, because this

shallow water is close to the land area, we must consider a model that can accommodate low prediction time for a Tsunami Early Warning System. Therefore, we propose a recurrent neural network (RNN) model because of its reliable performance for time series forecasting. Our report evaluates variants of the RNN model (the vanilla RNN, LSTM and GRU models) in tides prediction and z-score analysis for tsunami identification. The GRU model overwhelms the other two variants in error scores and time processed (training and prediction). It can achieve median error score distribution of $7.8 \times 10{-}5$ on the L1000-P250 configuration with time prediction under 0.1 s. This lower-time prediction is necessary to ensure the early warning system is going well. Moreover, the GRU model can identify all synthetic tsunami tide spikes (compared with the ground truth result) from magnitude 7.2–8.2 by applying a z-score on the GRU's prediction.

Keywords · Recurrent neural network · Deep neural network · Shallow water body · Tides prediction · Tsunami early warning system

3.2.1 Introduction

There is already quite much proof of how devastating the tsunami impacted the land of Indonesia, which brought in significant loss of material and human lives. Take one of the recent tsunami events as an example **[17]**, tsunami Palu Dongala (2018) records a total loss of 20.89 trillion rupiahs and 4340 people died. tsunami Palu Dongala (2018) records a total loss of 20.89 trillion rupiahs, and 4340 people died. Moreover, the impact will be worsened because of the stop of the economic growth in the post-disaster.

One of the attempts to reduce the number of tsunami strike victims, Indonesia has developed a Tsunami Early Warning System (TEWS), which started in 2005 **[18]** (post-Aceh's tsunami). Recently, predicated on the president's instruction, article 5 number 93 2019, as part of strengthening disaster mitigation, the Agency for the Assessment and Application of Technology or BPPT Indonesia developed Cable Based Tsunami-meter (CBT) **[19]**. This system will adopt the SMART concept, Scientific Monitoring, And Reliable Telecommunication, which incorporates a monitoring function of the tsunami, earthquake, climate, ocean condition, and sea level with telecommunication capabilities.

Concerning the threat of the giant tsunami caused by megathrust earthquakes in the Mentawai island area **[20]**, BPPT initiated the deployment of the CBT system in Sipora. The subduction zone of this area has a characteristic of a low water depth of about 80 m, which consider having considerable ambient noise **[21, 22]**. Multiple occurrences of bottom bounce path in the sound channel and uncertain seafloor properties, including sound speed, density, and attenuation, make continental shelves environment has significant external noise, which muddles up the data measurement retrieved from Bottom Pressure Recorder (BPR). This ambient noise level-dependent feature makes shallow water environments more challenging to analyze and model **[21]**.

Meanwhile, tides prediction is a time series problem in which the output is the sequence prediction within some margin of error. The traditional modelings are mainly parametric based, such as AutoRegressive (AR) **[23]**, exponential smoothing **[24, 25]** or structural time series model **[26]**. However, it has also been found that many of these real-time series modelings seem to follow nonlinear behavior **[27]** and are insufficient to represent their dynamics **[27–29]**. Therefore, another approach using a different mathematical representation of the nonlinearity present in the data is suggested to overcome this problem **[27, 29, 30]**.

Notably, the emergence of artificial neural networks (ANN) adopting this approach have been widely used for the prediction of various complex system **[31–33]**. They can identify and learn the complicated nonlinear relationship between system variables, showing more accurate results than linear regression techniques **[34]**.

Among these various techniques, recurrent neural network (RNN) can detect a pattern in the data sequence **[35]**. This ability differentiates from Feedforward Neural Networks, which pass information through the network without cycles. The RNN has cycles and transmits information back into itself, which extends Feedforward Networks to account for previous information. Despite this advantage, RNN suffers from vanishing or exploding gradient in long-term dependency **[35]**. This problem motivated the introduction of long–short-term memory units (LSTMs) **[36]** for handling the vanishing gradient problem. LSTM

has become popular in time series forecasting **[36]**. Compared to deep Boltzmann machines, graph-structured recurrent neural networks, and convolutional neural networks, LSTM-NN-based deep learning performs better **[37]** for time series forecasting. It can extract robust patterns for an input feature space and effectively handle Multiple Input Multiple Output System (MIMO) systems in Deep Neural Networks (DNN). Moreover, the LSTM system can take nonlinear systems due to their specialized LSTM cell that performs better after learning. However, LSTM has some drawbacks related to its complicated unit and more data necessary to learn effectively **[38]**. Therefore Gated Recurrent Unit is proposed as a simpler hidden unit to compute and implement **[38]**.

Nonetheless, recent research on time forecasting shows unforeseeable CNN **[39–41]** to solve time series problems. However, this problem is still limited to the classification as output, not time sequences which is the output of tides prediction. In addition, the wide range variability of the data set must also be experimented with for a solid model hyperparameter.

Based on all these studies, we select RNN as our model foundation. Some factors that support this are as follows:

- Tides prediction is a univariate time forecasting problem relevant to the efficacy of RNN **[12, 13]**.
- The output of tides prediction is temporal-dependent sequence data. RNN is suitable for sequence learning from the features **[42, 43]**.
- Though for the training phase, RNN still cannot take advantage of parallel computation in GPU **[44]**, RNN can still achieve a closed real-time prediction in the inference time, which is around one second or less, depending on the GPU.

Thus, our work is considered novel since the unavailability of tides prediction model and study in shallow water areas for tsunami prediction purposes. Our contribution will be as follows:

- Combination of multistage preprocessing and RNN based deep-neural network on tide data for solving tide prediction modeling in shallow water

cases. This model is intended to get a better tide prediction and human interpretation.

- The suitable tides preprocessing for shallow water cases that can reduce the noise of the tides data and accommodate the neural networks input.
- Empirical insight of various RNN models, vanilla RNN, LSTM, and GRU, approach on a case of tsunami detection based on tide prediction.
- Experimentations on defining look-back and forward parameter scenarios on shallow water tides prediction models.
- Z-score analysis toward variability of the synthetic tsunami triggered by earth magnitude. This analysis evaluates the sensitivity of the current model.

Finally, solving this near-field tsunami forecasting in the coastal area is urgently required to reduce casualties. Indonesia experienced a tsunami caused by coastal volcano eruptions in 2018 **[45]**. Furthermore, Sumatra island's coastal region, especially Mentawai island, has considerable potential for megathrust earthquakes and landslides **[46]**. Therefore, the RNN tides prediction model proposed in this paper can become the required solution to mitigate this problem.

3.2.2 Related Works

National oceanic and atmospheric administration (NOAA) developed the tsunami detection algorithm under the deepocean assessment and reporting of tsunamis (DART) project using cubic polynomial **[47]**. While in [34], Beltrami tried to find a more efficient alternative tsunami detection algorithm by proposing an artificial neural network (ANN). Both of these algorithms use the data from the bottom pressure recorder (BPR) as a sensor to collect the sea level in the deep sea. Based on the comparison **[48]**, ANN methodology can predict tide and other regular patterns in the wave better than the DART.

Before **[48]**, Barman et al. **[49]** utilized non-linear regression in ANN to calculate the estimation time arrival (ETA) for predicting the tsunami travel time in the Indian Ocean. The ANN model could perform the rapid computation for ETA. The model proved its robustness in developing a real-time tsunami warning system for the Indian Ocean.

These efficacies of ANN encourage data-driven forecasting tsunami **[50]**. Romano et al. **[50]** utilized spatial values of maximum tsunami heights and tsunami arrival times (snapshots) computed through the TUNAMI-N2-NUS model. They achieved good accuracy and near-instantaneous fore forecasting of the maximum tsunami heights and arrival times for the entire computational domain.

Another variant of ANN is also adapted to estimate tsunami inundation **[51]**. Fauzi and Mizutani applied two machine learning models, a convolutional neural network and a multilayer perceptron, for real-time tsunami inundation forecasting in the Nankai region of Japan. They experimented using the hypothetical future Nankai megathrust earthquake with Atashika and Owase Bays in Japan as the study cases. The results show that the proposed methods are high-speed (less than 1 s) and comparable with nonlinear forward modeling.

Besides tsunami mitigation, another natural disaster, such as an earthquake, is also predicted usingANN **[52–54]**. In the most recent **[55]**, Kishore et al. used the LSTM to model the sequence of earthquakes. They used the trained model to predict the future trend of earthquakes and compared the LSTM with an ordinary Feed Forward Neural Network (FFNN) solution for the same problem. The result showed that the LSTM neural network was found to outperform the FFNN in the task of modeling the sequence of earthquakes.

Compared to all the previous work, our study regarding tsunami detection in shallow water case is considered a premiere. The challenging part of BPR data in shallow water areas is the muddled ambient noise, which requires signal processing to filter these out from the expected features. Various RNN networks are evaluated, serving as context learners that forecast the upcoming tides. Finally, the z-score will identify the tsunami spikes from the set of predicted tides.

3.2.3 Methodology

- **General algorithm design**

This project defines the primary solution through a block diagram comprising preprocessing, training model architecture, and tsunami identification. Preprocessing sequences involve feature scaling, vector shape matching, and

denoising. The training model architecture consists of the RNN stacked model and the dense layer, which map the features into a serial data prediction. Finally, the system will identify the tsunami from the prediction sequence of tides by smoothed z-score methodology, as shown in **Fig. 3.1**.

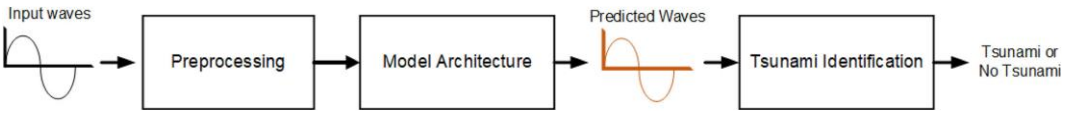


Fig. 3.1 General block diagram of designed system

- **Preprocessing**

The first part of the section (**Fig. 3.2**) is a scaler based on a percentile that will improve distribution data scaling. The process is unaffected by significant marginal outliers, which commonly occur in a noisy data environment. From this section onward, 0–1 normalization is required to match the RNN input layer. Eventually, the processing system applies a lowpass filter to reduce ambient noise. The LPF design parameter follows **[56]** Oceanographical Engineering Textbook allowing tsunami data to be captured with a frequency less than or equal to 0.01 Hz. The order of the filter is also set to 9 to reduce stopband ripple maximally, as shown in **Fig. 3.3**.

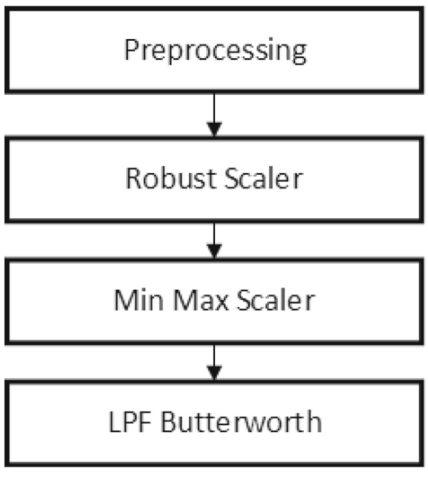


Fig. 3.2 Preprocessing block diagram

On the other hand, based on the general block diagram section (**Fig. 3.1**), we can write the system in pseudocode. Before utilizing the function, training data are fitted into the scaler to capture data traits (mean, variance, interquartile range, etc.). This trait can be saved into.bin format and loaded in the function.

Algorithm 1 Preprocessing

$data \Leftarrow tides$
$N \Leftarrow filter\_order$
$C \Leftarrow cutoff\_coefficient$
$trait_1 \Leftarrow robust\_trait$
$trait_2 \Leftarrow normal\_trait$
$robust\_data = trait_1.transform(tides)$
$normal\_data = trait_1.transform(robust\_data)$
$filter\_data = filter(N, C, normal\_data)$

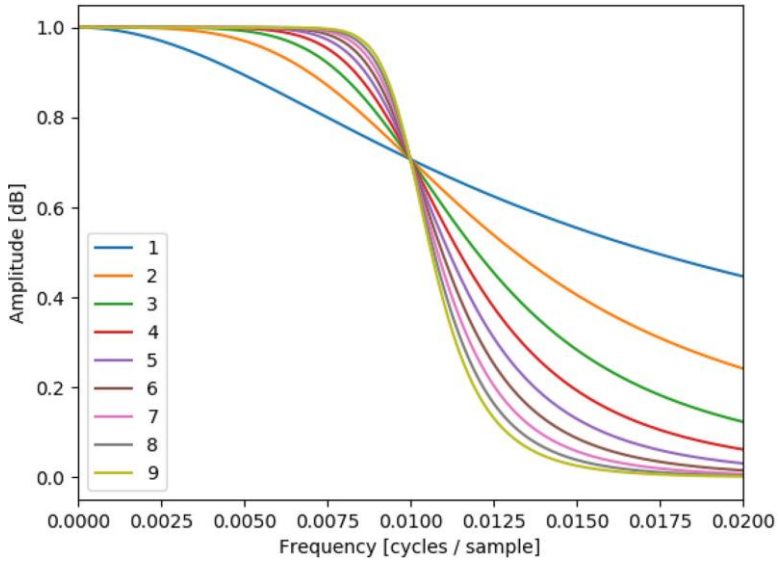


Fig. 3.3 Butterworth filter frequency response on 0.01Hz cutoff frequency

- **Model architecture**

The target of the design model **(Fig. 3.4)** is a vector of future predictions. This mechanism can be achieved by applying a stack of RNNs, followed by a dense layer. A drop-out layer shall be attached to the sequence to reduce overfitting cases.

Some variables define each of the functions. The input x needs to be in three-dimensional size, in which the vector should be reshaped into $(height \times weight \times 1)$. The n_units represents the number of hidden units denoting the number of dimensional output space while the $n\_samples$ symbolizes the number

of data input that becomes the previous data context. Finally, the predicted output is an array with time sequence size mapped using a dense layer.

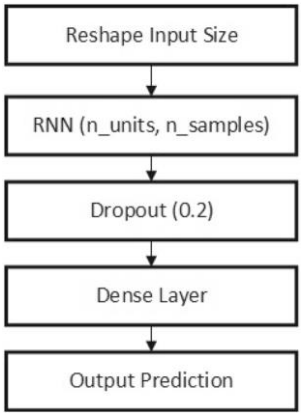


Fig. 3.4 Model network layers design

- **Model prediction algorithm**

This part (**Algorithm 2**) defines the basic algorithm to train and test the model prediction. The windowing LSTM with look-back variation value becomes the core of the algorithm. The program's first segment defines the number of *look − backs*, the number of predictions and reshapes input x into three-dimensional input ($samples, timesteps, features$), and y into two-dimensional information. The *look – back* parameter is the number of data points in prior timesteps, which become part of this project analysis. Finally, the model predicts the tide, then $data\_input$ and $predictions$ variables are updated.

Algorithm 2 Prediction

```
data_input ⇐ [number_of_lookback]
index ⇐ 0
model ⇐ load('model')
while true do
    index++
    if index < number_of_lookback then
        data_input.append(new_data)
    else
        prediction ⇐ model.predict(data_input)
        predictions.append(prediction)
        data_input.pop(0).append(new_data)
    end if
end while
```

3.2.4 Technical background

- **Recurrent neural network**

In most of the literature, a hidden unit in RNN can be formulated as follows **[57]**:

$$h(t) = f(h^{t-1}, x(t); \theta) \quad (1)$$

Referring to the (1) equation, h is a hidden unit function, x(t) is the current input, and θ is the parameter of the function f . This equation is recurrent because h at time t refers to the same definition at time $t-1$. There are several examples of the design pattern of RNN, yet to ease the exposition, we focus on the basic form of recurrent networks, hidden-to-hidden recurrent connection, which refers to **Fig. 3.5** **[58]**

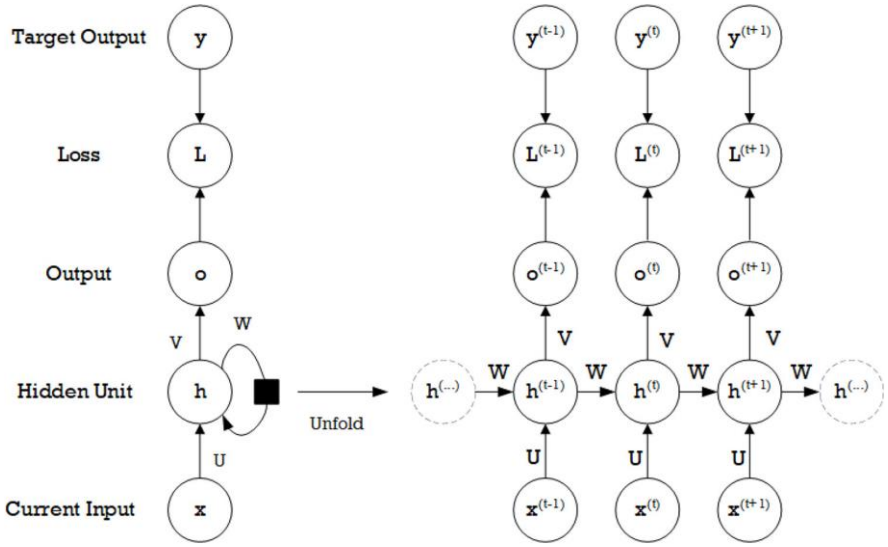


Fig. 3.5 Computational graph representation of RNN basic form, including training loss computation

From **Fig. 3.5**, we can see that the idea of RNN is to pass or connect previous information to the present task. This process is beneficial because it needs sequence information to be processed, such as a video frame.However, in this "long-term dependencies" case, Hochreiter **[59]** and Bengio **[60]** found some fundamental reasonswhy it is difficult. Therefore, basic RNNs fail to learn "long-term dependencies". Nevertheless, the variants of RNN architecture called gated RNNs, including LSTM and Gated Recurrent Units (GRUs), are introduced to tackle this problem. Especially, LSTM, which was introduced by Hochreiter and

Schmidhuber **[59]**, has been quite popular nowadays, as many researchers use it because of the efficacy in many different applications **[58]**.

LSTM introduces a new element called cell state c, which comprises the forget gate (f_t), input gate (u_t), and output gate (o_t). According to its name, forget gate determines whether the previous data is diminished. In contrast, the input gate evaluates the information to be carried over in the sequence, and the output gate decides the next hidden state value from the previous data. We can define each of the gates in the following equation:

$$u_t = \sigma(W_u h_{t-1} + I_u x_t + b_u) \quad (2)$$

$$f_t = \sigma(W_f h_{t-1} + I_f x_t + b_f) \quad (3)$$

$$o_t = \sigma(W_o h_{t-1} + I_o x_t + b_o) \quad (4)$$

Each of the formulae at the time step t, W_f, W_u, W_o, I_f, I_u and I_o are weight parameters on the corresponding gate, while variables, b_f, b_u and b_o, are bias alongside the gate. Thus, the cell candidate ($\tilde{c}_t$), current hidden state (h_t), and current cell state (c_t) can be formulated as below:

$$\tilde{c}_t = tanh(W_c h_{t-1} + I_c x_t + b_c) \quad (5)$$

$$c_t = f_t \odot c_{t-1} + u_t \odot \tilde{c}_t \quad (6)$$

$$h_t = o_t \odot tanh(c_t) \quad (7)$$

$$y_t = \sigma(W_y h_t + b_y) \quad (8)$$

where variables, W_c and I_c, represent weight parameters on the cell and variable b_c is bias alongside the cell.

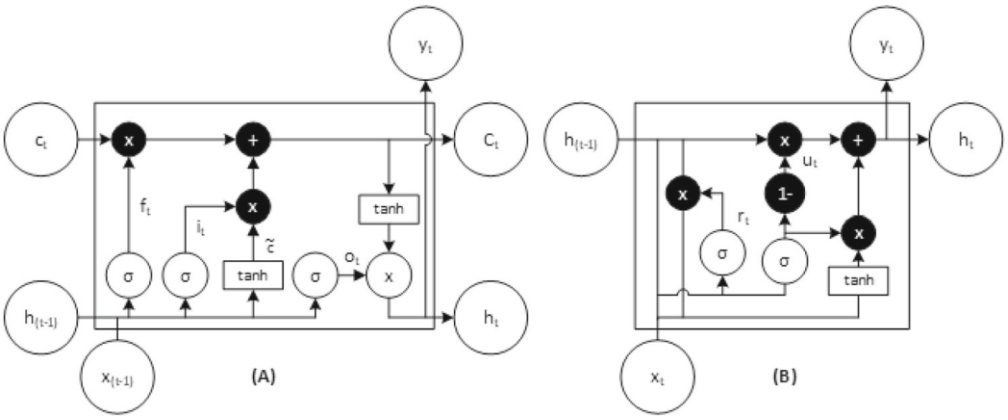


Fig. 3.6 General block diagram of A LSTM and B GRU

In comparison with LSTM (shown in **Fig. 3.6**), GRU replaces the three's LSTM gates into two gates: the update z_t and reset r_t gates. The update gate helps themodel control the new state's number from a copy of the previous state, while the reset gate intuitively controls how much past information to forget. The GRU unit is defined as the set of the equation below:

$$z_t = \sigma(W_z\, x_t + U_z h_{t-1} + b_z) \quad (9)$$

$$r_t = \sigma(W_r\, x_t + U_r h_{t-1} + b_r) \quad (10)$$

$$\widetilde{h_t} = tanh(W_h\, x_t + (r_t \odot h_{t-1})U_h + b_h) \quad (11)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h} \quad (12)$$

$$y_t = \sigma(W_y\, h_t + b_y) \quad (13)$$

From empirical insight **[61]**, GRUs overcome LSTM network performance for low complexity sequences and vice versa. This performance **[61]** corresponds to the size of the learning rate for each complexity rate (low and high) of seed strings. LSTM networks perform better for similar forecasting on higher complexity of seed strings.

- **Data Acquisition**

INA CBT Sipora consists of two sensors on Ocean Bottom Unit (OBU) and an optical cable under the sea (**Fig. 3.7**). One sensor is the Bottom Pressure Recorder (BPR), a pressure transducer measuring tide periodically. Three parameters are captured per second, DateTime, water column height, and temperature (**Fig. 3.8**).

Fig. 3.7 Component of OBU Sipora

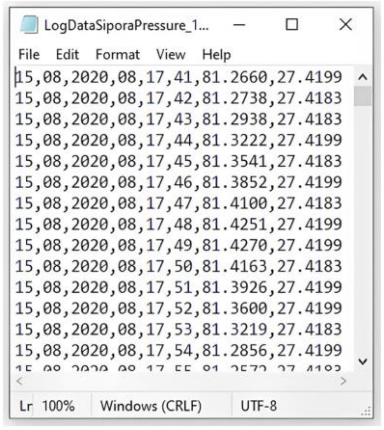


Fig. 3.8 Periodic tide data acquisition

- **Testing data**

The testing data preparation consists of modelling shallow water tsunamis using the Tunami-F1 model **[62]** and injecting the tsunami model into actual capture data. The work of Infrastructure Technology Centers Ports and Coastal Dynamics BPPT Indonesia helped the tides prediction model for tsunami identification by providing shallow water tsunami data tests. They simulate dummy tsunamis generated by an earthquake ranging from 6.4 to 8.2 magnitude (**Fig. 3.9**).

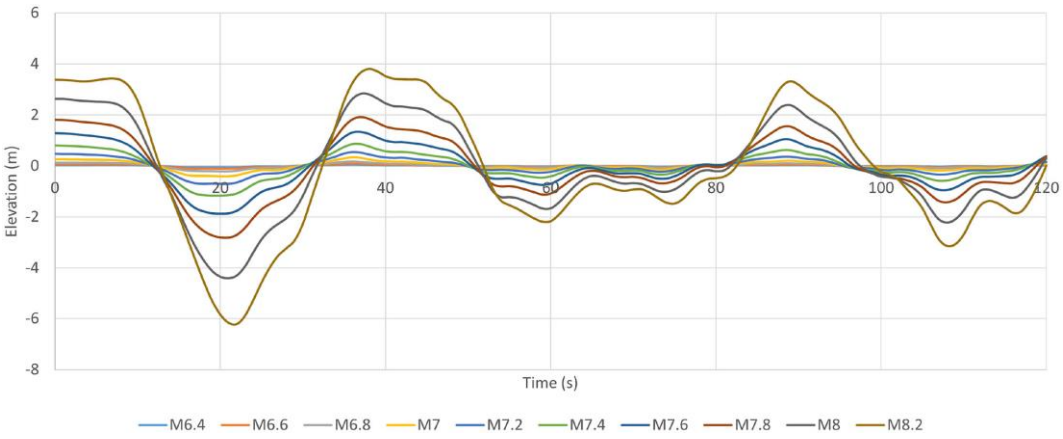


Fig. 3.9 Tsunami caused by an earthquake with the variability of magnitude

After the tsunami-generated by earthquake variabilities are produced, the modeled tsunami is injected into the test data. Some steps are to follow for injecting tsunami data into data tests and real-time captured tide data. The process starts by interpolating the tsunami-generated data by matching the model time sampling. The resulting wave will have zero paddings conforming to the array dimension. Finally,

the modeled tsunami superposes test data, which yields injected tsunami waves, as presented in **Fig. 3.10.**

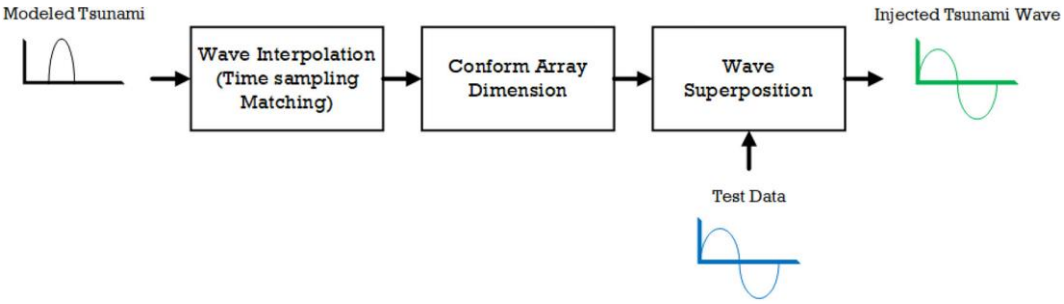


Fig. 3.10 Block diagram process of tsunami injection

- **Tsunami identification**

Because of the low amount of the actual shallow water tsunami data set (only one was generated), the model cannot be expected to solve the classification problem. Instead, the generated data prediction will use a smoothed z-score to determine the tides as a tsunami or not. Z-score is a standard methodology used in forecasting problems to identify the trend from the prediction. This score indicates how many standard deviations an observation on each i is above or below the mean:

$$Z_i = (x_i - \bar{x})/\sigma \quad (14)$$

3.2.5 Result and analysis

All test procedures are performed through Python 3.8 with Keras, Tensorflow, numpy, pandas, scikit-learn, and matplotlib third-party library. On top of that, these program specifications are supported by GPU Nvidia A6000 as part of the computer platform in Artificial Intelligence Laboratory Kanazawa. After obtaining the results, we performed two analyses to evaluate our designed performances. Those are look−back prediction and z-score tsunami identification analysis.

- **Preprocessing procedural testing**

In **Fig. 3.11**, the training data set comprises input data, a periodic tidal wave captured continuously every second for 5 days. Later, these data go into two scalers, robust, and min–max scaler. A robust scaler transforms the data input by removing the median and scaling the data according to the interquartile range (IQR)

(**Fig. 3.12**). The IQR ranges between the 1stquartile and 3rdquartile. This process makes the distribution of data robust to outliers. A min−max scaler is then applied to adjust the value range from 0 to 1, which is required for LSTM input data.

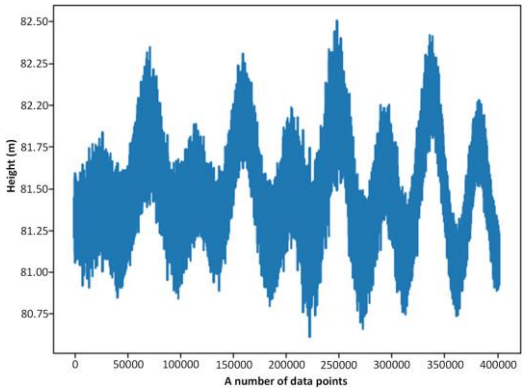


Fig. 3.11 Training data set

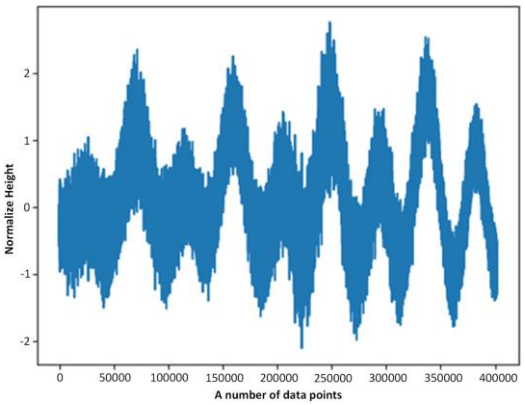


Fig. 3.12 Robust scaler input transformation

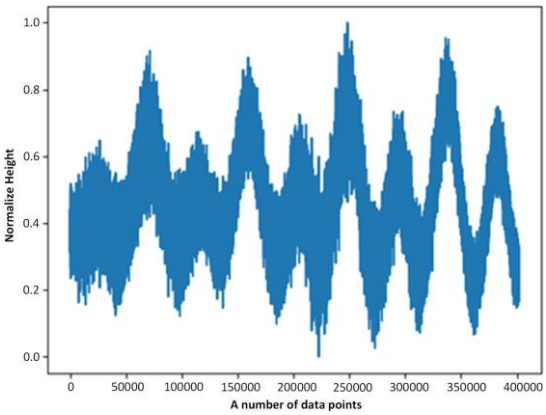


Fig. 3.13 Min–max scaler input transformation

After that, the Butterworth filter refines the normalized waves to reduce data noise form (shown in **Fig. 3.13**). From **Fig. 3.14**, data noise is decreased heavily into a smoother appearance.

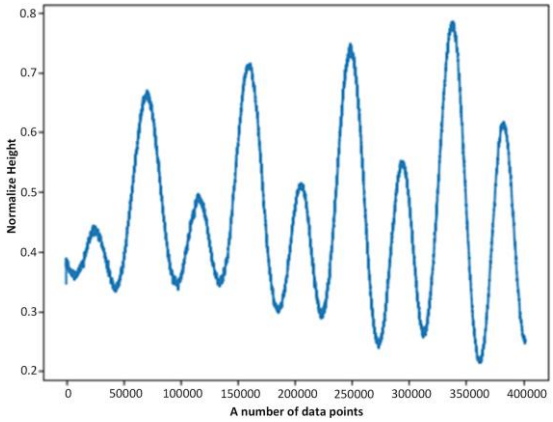


Fig. 3.14 Low-pass filter Butterworth output

- **Model and validation data set**

We sample six daily Sipora tides log data from Sipora OBU and divide them into a 5:1 ratio of a data set for training and testing. This testing data become the basis of dummy data for the superposition of tsunami-generated earthquake variation.

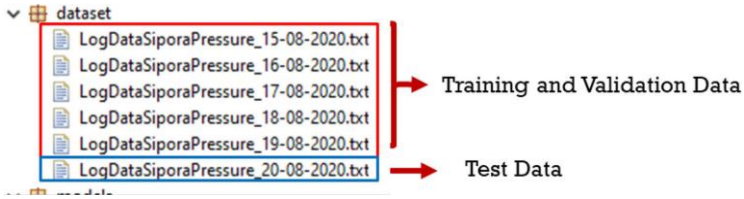


Fig. 3.15 Data set composition

There are 401,696 data points for the training set and validation. These data are split into 0.8 training and 0.2 test data (**Fig. 3.15**). Arbitrarily, we choose a configuration from a particular model to represent the "Training loss VS Validation loss" output. The training process uses Mean Squared Error (MSE) as a loss function. It converges with a training loss of 1.11×10^{-4} and a validation loss of 2.23×10^{-4}. The difference in these numbers indicates that the model is neither underfitting nor overfitting because of its small margin in the region 10^{-4} (refer to **Fig. 3.16**).

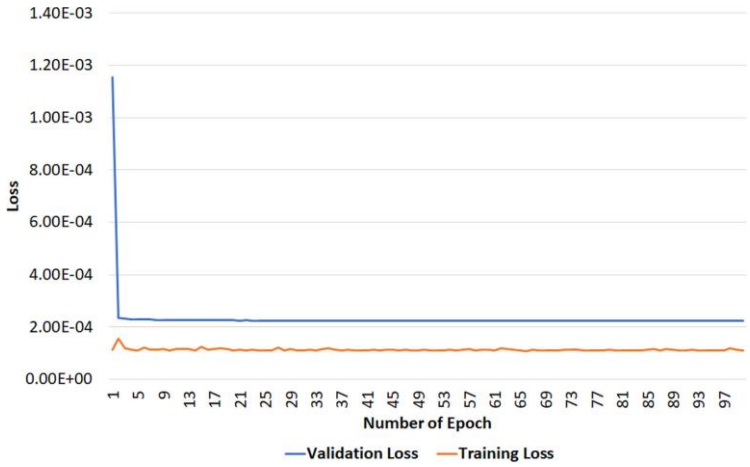
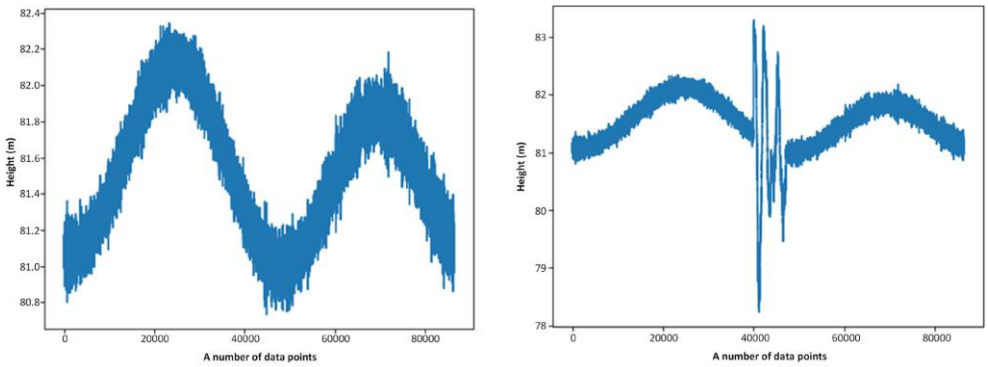


Fig. 3.16 Model train versus validation loss (from LSTM training process)

This model training is finished with 100 epochs and batch size 256-this process executes 100 units of the RNN stacked model. Adam optimizer is also applied with the learning rate $1e^{-3}$ and decay rate $1e^{-5}$ to improve convergence speed.

- **Testing data preparation**

Afterwe collect the necessary model data, the trainingmodel is saved into .h5 format. Then, the model is tested with an external data set referred to as the B section. Before the prediction, the data test embedded tsunami synthetic are prepared by using the explained algorithm in the methodology part, section 5 (**Fig. 3.17**).



(a) Without tsunami (b) Injected synthetic tsunami

Fig. 3.17 Data test

When all necessary data tests are gathered, the upcoming process will be a prediction. From this prediction, we can compute the error rate for performance

evaluation. Moreover, prediction and data input are also plotted to know how well the RNN model filters out the ambient noise (**Fig. 3.18**).

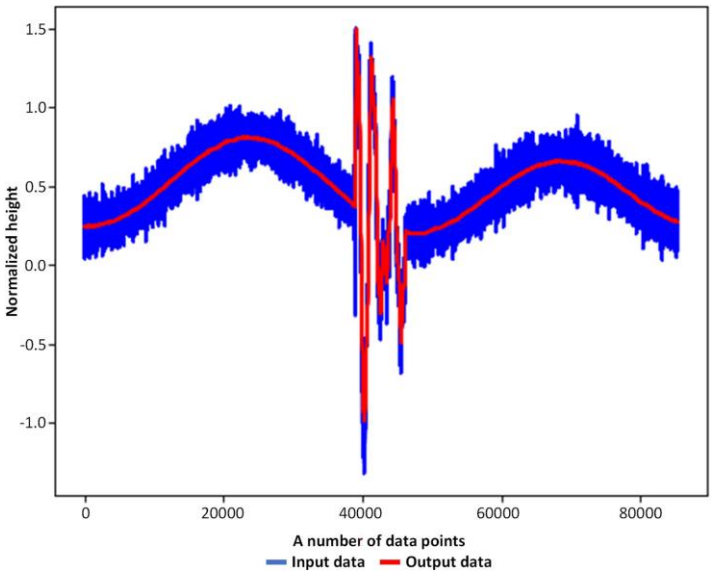


Fig. 3.18 Sequence prediction test on tsunami data injected induced by earthquake on mag. 7.8

- **Look-back variation analysis**

A set of look-back and prediction parameter configurations is tested. Moreover, to choose the look-back parameter,we need to empirically assess the computer's capability to execute multiple arrays related to GPU resources. In this experiment, the composition number of look-back and prediction points that can be executed maximally is 1000 nodes of look-back and 1000 predictions. Higher configuration points can raise Out of Memory (OOM) errors caused by insufficient memory. Initially, vanilla RNN networks are implemented to see how classic RNN work in the tide prediction application. In this experiment, as the input data will be continuous tides with temporal dependent, we apply a stateful setting on the RNN networks. Consequently, the network can learn the previous batches. Mini batches are also done for the input to ensure all the sequences are processed.

Vanilla RNN shown in **Fig. 3.19** can achieve a median from MSE score distribution of 1.14×10^{-4} on Look-back 1000 and Prediction 250 configuration. The median of the MSE score distribution is used as a pointer because the mean of the MSE score is skewed as a result of outliers, as shown in **Fig. 3.19**. It also can

be seen that each ratio L250-P250, L500-P100, and L500-P500 has the same range of 10^{-4} as the most minimum median in the MSE distribution. On the other hand, the other's ratio shows MSE scores in the range of 10^{-2} and 10^{-3}. Besides its performance, from **Fig. 3.19**, L1000-P50, L1000-P500, and L1000-P1000 configurations are missing because "NaN" errors occur during prediction. We consider this instability to be caused by vanilla RNN's insufficient representability for capturing the complexity of the tides.

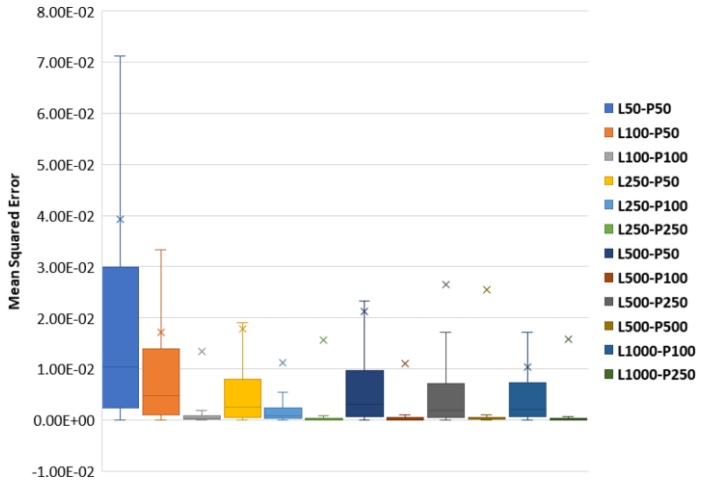


Fig. 3.19 Error distribution of tides prediction of Vanilla RNN model

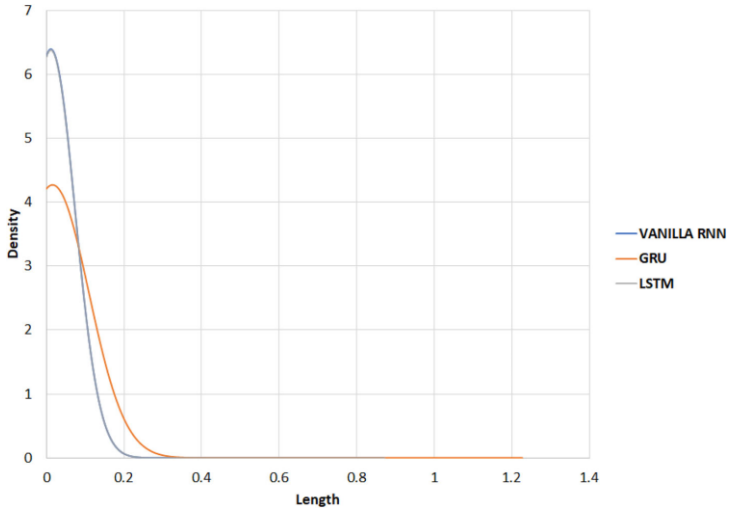


Fig. 3.20 Density distribution normalized graph of MSE distribution on L1000-P250

Surprisingly, according to **Fig. 3.20**, LSTM has a similar MSE distribution on our tides prediction with vanilla RNN on the ratio of L1000-P250.

However, compared to vanilla RNN, the LSTM model tremendously improved the median MSE score (**Fig. 3.21**) by reaching 7.96 $\times$ 10^{-5} on a ratio of L500-P250. On the same ratio, L1000-P250, it also shows improvement to 8.14 $\times$ 10^{-5}.Other ratios denote better scores than the vanilla RNN model.

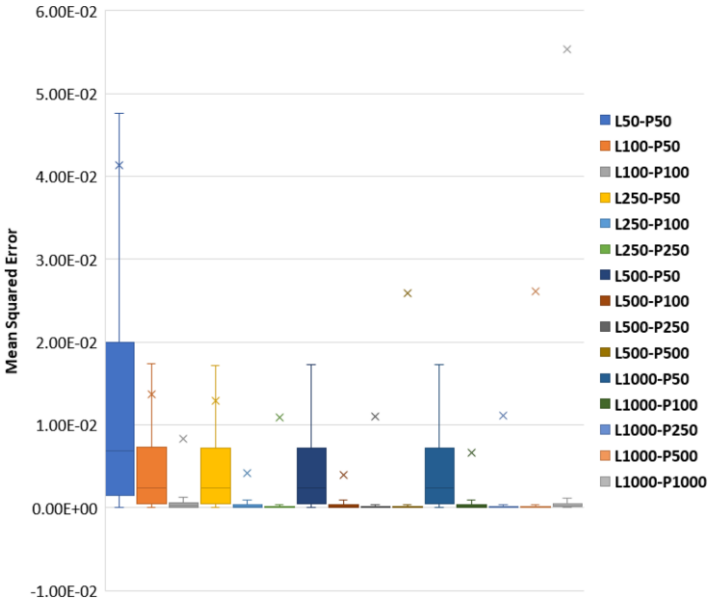


Fig. 3.21 Error distribution of tides prediction of LSTM model

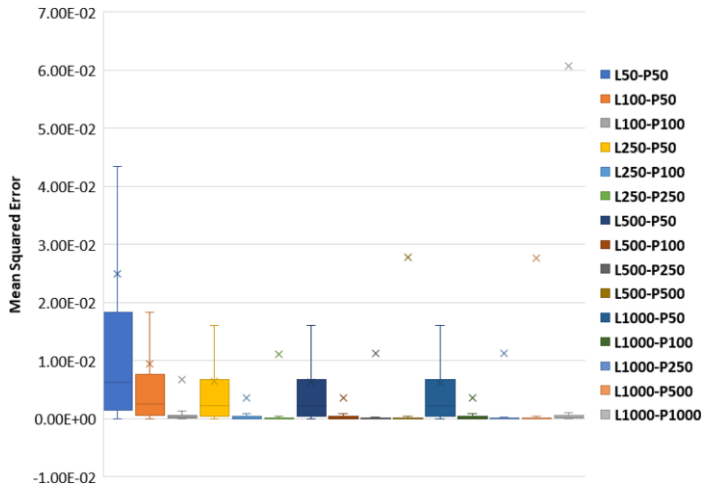


Fig. 3.22 Error distribution of tides prediction of GRU model

Finally, the GRUmodel shows the best performance compared to the others. It can pull off themedian in the error score distribution to 7.8 $\times$ 10^{-5} on the L1000-P250 configuration (**Fig. 3.22**), which is also smaller than the other two models (vanilla RNN and LSTM). Overall performance, GRU exhibits a higher but

close error score to the LSTM model (9 configurations higher than the LSTM model (**Table 3**)). This improvement in error score indicates that the tides prediction problem has a low complexity sequence which, in this case, GRU has better performance and efficiency (**Table 3** and **Fig. 3.23**).

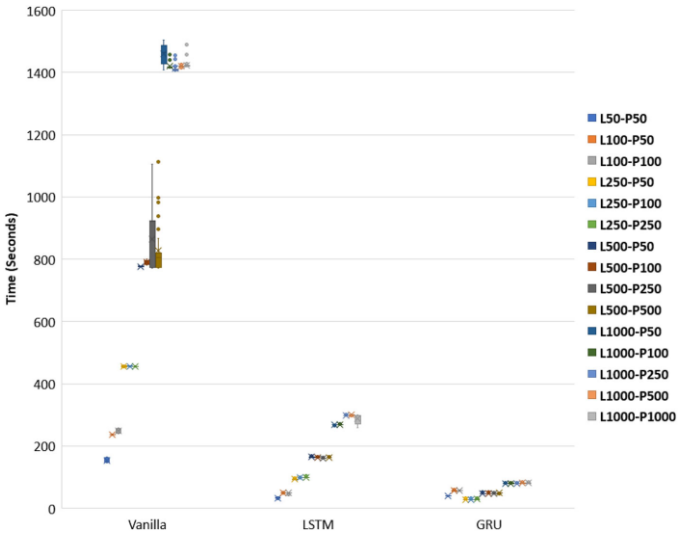


Fig. 3.23 Vanilla RNN VS LSTM VS GRU training time for one epoch distribution

Table 3 Median of MSE distribution for each of model configuration

Model	L50-P50	L100-P50	L100-P100	L250-P50	L250-P100
Vanilla RNN	1.02×10^{-2}	4.69×10^{-3}	4.15×10^{-4}	2.54×10^{-2}	8.28×10^{-4}
LSTM	6.91×10^{-3}	2.42×10^{-3}	2.33×10^{-4}	2.35×10^{-3}	1.6×10^{-4}
GRU	6.35×10^{-3}	2.54×10^{-3}	2.82×10^{-4}	2.25×10^{-3}	1.65×10^{-4}
	L250-P250	L500-P50	L500-P100	L500-P250	L500-P500
Vanilla RNN	1.88×10^{-4}	3.1×10^{-3}	1.95×10^{-4}	1.78×10^{-3}	2.28×10^{-4}
LSTM	8.29×10^{-5}	2.35×10^{-3}	1.62×10^{-4}	7.96×10^{-5}	1.13×10^{-4}
GRU	8.36×10^{-5}	2.25×10^{-3}	1.59×10^{-4}	7.92×10^{-5}	1.2×10^{-4}
	L1000-P50	L1000-P100	L1000-P250	L1000-P500	L1000-P1000
Vanilla RNN	NaN	2.03×10^{-3}	1.14×10^{-4}	NaN	NaN
LSTM	2.36×10^{-3}	1.58×10^{-4}	8.14×10^{-5}	1.18×10^{-4}	3.01×10^{-4}
GRU	2.26×10^{-3}	1.57×10^{-4}	7.8×10^{-5}	1.17×10^{-4}	3.09×10^{-4}

This efficiency refers to the training and prediction time. As for the context of tides prediction in the tsunami application (**Fig. 3.22**), our model should predict as fast as possible to ensure enough time for the information to be conveyed on the shore. Nonetheless, the training time for one epoch depends on the layer type, number of hidden units, network depth, input data dimension, and model hyperparameter. From our experiments, GRU shows remarkable efficiency in training and predicting time. All GRU model configuration accomplishes the training process for under 90 s per epoch (**Fig. 3.23**). This result is also directly proportional to the prediction time of one sequence output which set off all the configurations under 0.1 s (**Fig. 3.24**). Second best in efficiency, The LSTM model finishes the training process for one epoch up to 302 s and prediction of 0.352 s (**Fig. 24**). This performance evaluation is relevant to the [**61**] for less complex sequence problem. The last model, vanilla RNN, is the palest in time performance compared to others. It takes up to 1490 s to finish one epoch training and 1.14 s to predict the sequence of tides. Nevertheless, this performance showcases validation on RNN model comparison, which is relevant to the previous research [**33, 36, 58, 61**].

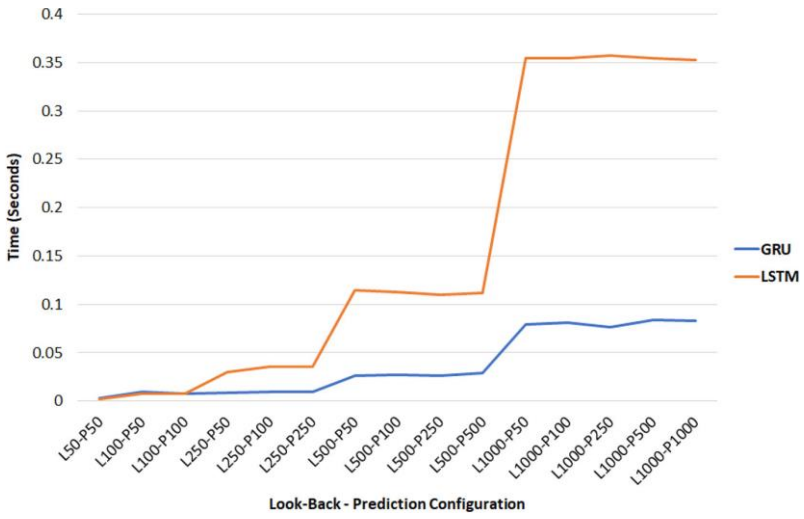


Fig. 3.24 GRU VS LSTM time prediction

- **Z-score analysis**

Z-score detects significant variations of the spikes from the expected tides. This methodology can work using the means and standard deviation of normal tides.

Then, the threshold value is applied to the standard deviation as a margin to the expected tidal wave distribution.

This methodology is tested on the divergence of tsunami synthetic. In this experiment,we empirically set the threshold of 2.7 of standard deviation as it shows the test's false error. The fluctuation can be identified by "1" as a rising tide and "−1" as a downward spike (**Fig. 3.25**).

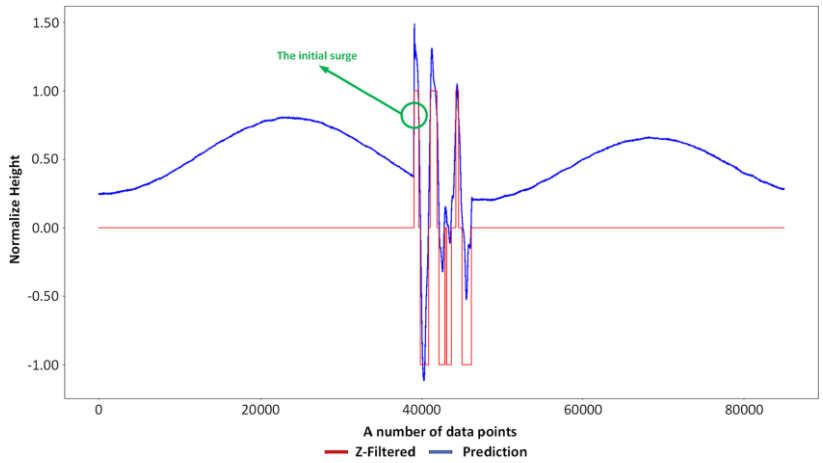


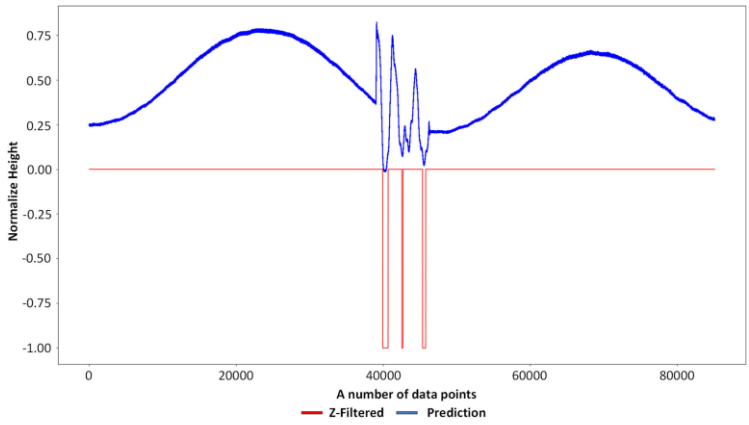
Fig. 3.25 Z-score tsunami spike identification on the data test with magnitude 7.8

Our experiment also evaluates the z-score on each model prediction. **Table 4** shows the performance of the tides predictionmodel processed in z-score to identify a surge of tides or tsunami caused by an earthquake of various magnitude. The shallow water tsunami of LSTM and GRU prediction can be completely identified in corresponding to the number of peaks detected in ground truth prediction. Still, vanilla RNN prediction misses two tsunami tides on magnitude 7.4 as shown in **Fig. 3.26**.

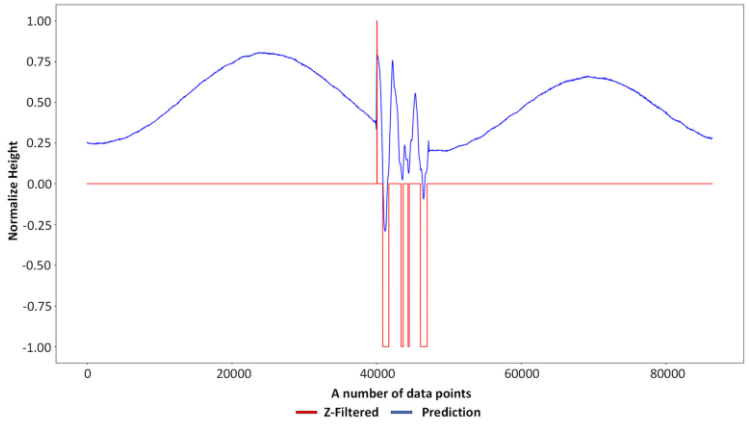
Table 4 Tsunami tides identification for each model

Magnitude	Ground Truth Det	Vanilla RNN Det	LSTM Det	GRU Det
7.2	2	2	2	2
7.4	5	3	5	5
7.6	6	6	6	6
7.8	7	7	7	7
8	6	6	6	6
8.2	6	6	6	6

Furthermore, the z-score methodology cannot identify tsunami spikes in the magnitude 6.4–7 range. This outcome is supported by the fact that thewaves embedded in the tsunami on that range are blended exceptionally well; the waveform is hardly noticed. The z-score can determine the number of fluctuations caused by the synthetic tsunami in the other magnitude span. The higher the magnitude cause, the easier z-score can recognize the sudden change of tides. Moreover, it can detect the tsunami's initial surge on the majority of magnitude except for magnitude 7.2 and 7.4 (only for vanilla RNN),which is vital to knowwhen the tsunami starts (shown in **Fig. 3.27**).



(a) Vanilla RNN



(b) Ground Truth

Fig. 3.26 Comparison of z-score processed

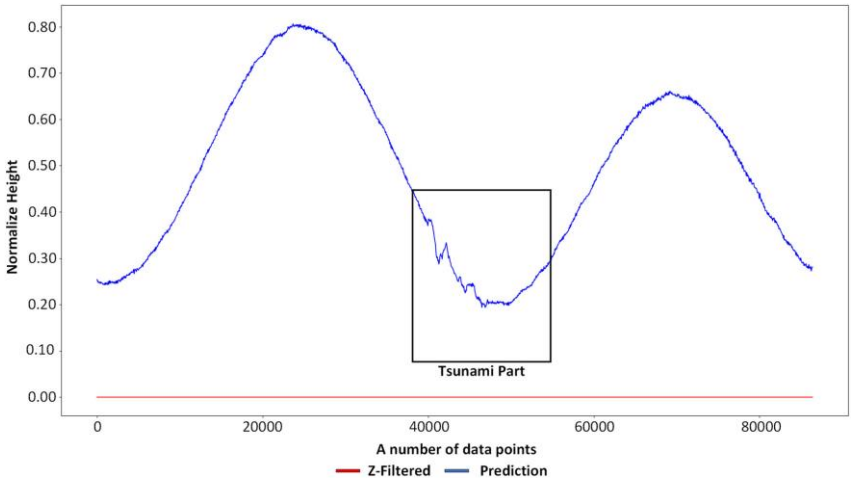


Fig. 3.27 Tides embedded synthetic tsunami induced by earthquake with a magnitude of 6.4

3.2.6 Conclusion and future works

Applying a time series model to tide prediction problems on shallow water requires multiple mechanisms to identify tsunami spikes. The procedures start from the preliminary operation on the input waves. Two-step features scaling, robust, and min–max scaler are applied to capture the wave's distribution and adjust to theRNNvariant input array requirement. Then, the Butterworth filter with 9 order and 0.01Hz cutoff frequency work in the sequence to filter out the ambient noise. This process continues to the time series model prediction of RNN and its variation. Finally, the z-score will determine whether these waves are possibly tsunamis.

We find that tides prediction is a low-complexity sequence problem corresponding to the performance evaluation of GRU, which is better than the LSTM model. GRU score lowest MSE median of 7.8×10−5 on the L1000-P250 configuration. It also exhibits the best efficiency by accomplishing the training process for under 90 s per epoch and the prediction process for under 0.1 s for all test configurations. Besides, in z-analysis, GRU and LSTM prediction show complete identification of tides.This result indicates that the GRU model suits the tides prediction problem.

Incorporating a z-score in the surge of tides identification is due to the limitation of actual tsunami data in shallow water areas on newly deployed CBT Sipora, in which the model needs sufficient data for classification problems. In the future, incorporating an accelerometer in time series data input and prediction will improve the tides prediction model and cover the lack of capability in determining tsunami spikes caused by lower earthquake magnitude.

In addition, it is also worth mentioning recent developments of a deep transformer model **[63–65]**, which has shown state-of-the-art performance in time series forecasting problems. This algorithm introduces the self-attention method, which can overcome the "short-term memory" problem over infinite long sequences **[64]**. This approach should also be included in the next study to find a better tides prediction model to improve efficiency and accuracy.

Acknowledgements - We express our gratitude to Infrastructure Technology Centers Ports and Coastal Dynamics BPPT and task force 5 of the Indonesian Tsunami EarlyWarning System,WidjoKongko. Dr.-Ing, for providing tsunami synthetic evoked by earthquake with the variability of magnitude. Furthermore, we also convey our thankfulness to the photonics laboratory for facilitating us with the said computer in the experiment. Finally, we appreciate all of the Indonesian CBT team's hard works in supplying the data set for the training and the test.

CHAPTER 4: PAPER 3: End-to-End Time Distributed Convolution Neural Network Model for Self-Driving Car in Moderate Dense Environment

4.1 Foreword

Supervised learning of spatial based autonomous car requires images as an input as well as a state of the current condition. Therefore, we need to capture images from the designated environment. Generally, we start to develop an autonomous car environment using Carla based simulator as well as adding and modifying the program so that it can capture and label the program automatically. While we finish the program, we make a batch-based trainer, to make an h5 model. Then, we test the model and grab the evaluation parameter for analysis.

4.2 Published Paper

End-to-End Time Distributed Convolution Neural Network Model for Self Driving Car in Moderate Dense Environment

Willy Dharmawan, and Hidetaka Nambo

4.2.1 Abstract

Vehicle control in Autonomous Car requires the following command to make sure that the car can accomplish a specific task, such as taking a turn, stop on the traffic light, following lanes, and changing lanes. This serial command indicates that a self-driving car should not be addressed as a context-based problem that theoretically needs a temporal system that can accommodate multiple frames.

Based on this added complexity of the problem, we propose a network that can accommodate the sequential input of images. Thus, we apply a time distributed model of Convolutional Neural Network (CNN), to recognize a visual problem, followed by LSTM that can capture temporal state dependencies.

By modifying the Carla environment, we can capture frame per frame images with detailed information of throttling, speed, steering angle, brake, and some states such as direction, speed limit, and traffic light state. We use the Carla

control agent so that it can automatically capture all of the images from the camera and those of information. We demonstrate that this rough approach can perform well in the Carla environment with moderate dense traffic. It can reach the destination faster than the ground truth and standard convolution model in just 93.978 seconds. Although the driver agent performance is a bit rough with around 13.27 of speed above score, it shows a better steering control, which means better stability.

Keywords—Time Distributed, LSTM, CNN, Carla, Autonomous Car

4.2.2 Introduction

Autonomous car system, which split up into spatial perception and control module, has become a part of the development of vehicle control rule-based solution **[66], [67], [68], [69], [70]**, as well as a traditional solution **[71]**. Then, in the recent year, Nvidia proposed a newer solution **[72,73]**, applying a high dimensional feature extraction using Convolutional Neural Network (CNN) and map the result into steering control.

This efficacy of learning action policies from mapping pixel images is appealing because it directly mimics the demonstrated performance, without accounting to the agent environment. Even though it can learn vehicle control from the input of images, we think that self-driving car does not just rely on single information of images. It is a sequence problem that requires multiple data of images. It shows in **[74]** that a sequential based model has slightly better accuracy in comparison to just CNN based model. This result shows in the implementation of Fully Connected Network-Long Short-Term Memory (FCN-LSTM) architecture on discrete action driving experimentation. Therefore, in this work, we explore a time distributed based model which combines CNN and LSTM, accommodating multiple input and multiple outputs.

The differences of time distributed model with stacked CNN are the LSTM layer part and sequential input in time steps. The agent will learn a control parameter from the subsequent information. The model extracts the high dimension features, and LSTM will determine the importance of each feature. From this

process, the model will yield the control parameter. This methodology fits with the autonomous car problem because it is a sequential problem that needs multiple images to know the best policy.

Many factors in the self-driving car environment cannot be reflected solely through a single front-view camera sensor. So that, we address this problem into a discrete state such as go left, right, go straight, and lane-follow and define these into three parameters output, steering angle, throttle, and brake. Also, the Carla environment provides some parameters, such as direction state, speed, speed limit, and traffic light state in the lane. Thereby, we can just simply make some adjustments on the code and create a program to acquire our training data and test our model. In contrast, it requires another algorithm or mechanism to get those specific states of the agent.

There is also a problem regarding on how to set a temporal model on time steps. If we try to use a standard dense layer sequentially, the weights and biases might be changed, and it makes the output flattened with each time step mixed. Thus, we use the time distributed layer to wrap the model, to get output separately by time steps.

We evaluate this proposed model by comparing it to a spatial based model and ground truth data on how it differs in test performances in the town environment. The experiment results show that time Distributed model has better stability in steering angle in comparison to just a regular stacked CNN model. However, this combination of CNN-FCN-LSTM does not perform well in controlling speed, because it has the highest score in above speed control. Consequently, it can reach the predefined destination faster than the ground truth and CNN based model.

4.2.3 End-To-End Self Driving Car

End-To-End Learning has been known as a straight forward way and more into brute force technique for solving a complex problem by taking advantage of deep learning structure **[75]**. It comprises of several layers that constitute an artificial neural network, which imitates a distributed approach to solve a problem.

The autonomous car is one of the remarkable examples of a complicated problem. To define this particular system, Alexandru Serban et al. create a multi-layers diagram **[76]**. It starts with the extraction of sensor fusion data to get relevant features (e.g., object detection), then the car will have information on the surrounding environment (world model). From this onward, the system will generate a behavior model that is useful for decision making in the planning layer. Finally, the control layer interfaces the control through the actuator.

The end-to-end learning model simplifies these multilevel problems. The first attempt in 1989, Autonomous Land Vehicle in a Neural Network (ALVINN) **[77]**, manages to do well in simple roads with few obstacles. It just comprised of a shallow network that predicted actions from pixel inputs. Nevertheless, this successful show the potential of neural networks for autonomous navigation.

With the development of deep learning, mainly CNN based model **[78]**, Nvidia proposed a similar idea that fully utilizes convolutional networks **[73][79]** to extract the features from driving images. This stacked convolutional layer is adjusted so that it can map the high dimensional features into vehicle control, a steering angle. It shows a successful result with a straightforward scenario such as highway lane following and driving in obstacle-free courses.

Following that basis, this mechanism expands with a multi-modal multi-task framework **[72]**, which combines CNN based networks with FCN-LSTM. They address end-to-end steering control with new speed prediction. They use Udacity and SAIC dataset to evaluate their model and shows that their model has a better mean absolute error in steering angle prediction.

The recent works focus on the variation of CNN based model on various performance evaluation. There is an ablation test of imitation learning in a successive percentage introduced by F. Codevilla et al. **[80]**. On multi-task learning, S. Chowdhuri et al. **[81]** presents multi-modal multi-task learning with a comprehensive evaluation of multiple types of datasets. This model develops into Multi-task learning from Demonstration (MT-LfD) **[82]** combining ResNet and Soft Attention to measure visual affordance.

Our work wraps the multi-modal and multi-task function with the Nvidia basis model on context learning. Driving behavior becomes the evaluation parameter. These parameters are defined through the accessible information in testing simulation.

4.2.4 Methodology

- **Working Flow**

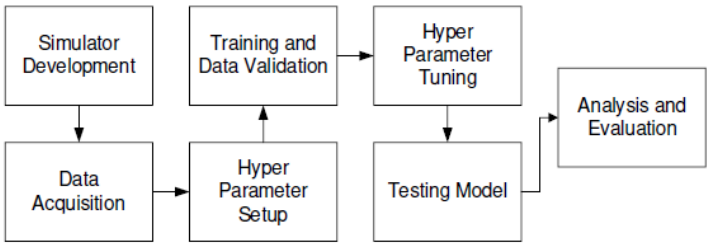


Fig. 4.1. Project working flow

This work (**Fig. 4.1**) starts from setting up and overwritten simulator program tailoring up to the test designed. It requires an understanding of the Carla environment to add some of the codes into Carla's python classes. Some of these classes are roaming_agent, basic_agent, local_planner, and autonomous_controller. These sections are essential parts to adjust to implementing an autonomous car algorithm.

All of these modifications are also useful for helping the user in capturing and labeling the data to create a training set and test set, which commonly requires a lot of time. When the dataset is ready, predefined hyperparameters need to be configured, such as number of epochs, learning rate, batch size, loss function, etc. Later, the obtained data is trained and validated with the ratio of 7:3, follows K-fold crossvalidation recommendation to get a good proportional dataset. The engineering process is taken place in the tuning process to achieve the best model performance.

After all parts of the design are set, training and data validation, as well as testing model, are executed. We track and save the results to multiple training logs. Through these saved data, the model is analyzed and evaluated.

- **Simulation Environment**

In this work, we simulate the model in Carla Simulator **[83]** version 0.9.5, which has a numerous improvement in comparison to 0.8.4 version, such as python API, waypoint ID, autopilot system, etc. We also use a navigation-based Proportional Integral Derivative (PID) controller, which employs a control loop on sensor feedback for the autonomous system. It is hardcoded AI that works imitating the human driver, such as stopping when red light, running while green light, keeping the speed up in the different lane, and distance up to the front car. Moreover, we can record all acquired data, such as agent states and parameters, images, and configuration information.

Specifically, we emulate the model in the Town02 map with clear noon weather conditions. By detail, the weather parameter has cloudiness 15, precipitation 0, wind intensity 0.35, sun azimuth angle 0, and sun altitude angle 75. We also set up an overall number of vehicles into 50 random means of transport, ranging from regular cars, bicycles, motorbikes, and trucks. These vehicles spawn in different positions. Their movement is also spreading randomly and follow the way of an expert agent moved. So, it follows the traffic rule and speed limit.

In the environment, there are varieties of landmarks, such as buildings, trees, warehouses, high buildings, open space areas, and many more. All these landmarks are part of selfdriving car challenges, how the agent can withstand the irregularities in the environment as well as keeping the lane and distance with non-player vehicles while maintaining the rule such as speed limit and traffic light.

Non-player vehicles have deterministic behavior. This kind of unrealistic nature will make an expert agent easily derives its policy. As our test focuses on driving experience on a different model to the target goal, we don’t put stochastic nature to these vehicles. Instead, apart from the standard capture (without added noise), temporal noise is added to the expert agent to cover all of the possible states in the test session.

All of these data are captured on the PC with a slightly higher spec, i7 6700 3.4 GHz, ram 16 GB, GPU 1080 GTX in python 3.7. The PC is also equipped

with 3rd parties library for training and testing purposes (Tensorflow-gpu, Keras, and library dependencies). Our test on this computer can achieve performance around 30 to 115 fps depends on the non-player crowd.

Fig. 4.2. Example of the arbitrary route generated randomly in the Carla environment.

- **Model Architecture**

The design of our model architecture comprised of Long-term Recurrent Convolutional Network design **[80]** with some tuning and idea of the Pilotnet model **[73]**. It includes three control parameter which defines the state of the autonomous agent.

Firstly, we crop and resize the input image into 66 x 200 x 3, adhering Nvidia designed model **[73]** as it has been tuned in that way to achieve a good result. Then, it follows with HSV convert, to remove noisy information. After that, the six stacked convolutional layers will extract images into high level features of data. The main differences between normal and time distributed are:

- Time distributed is a wrapper which applies time slice of an input. In other words, the layer put on a dense layer on n timesteps depends on the configuration.

- The weights and biases will not change until the batch of the sample concerning time steps shifts into another distribution. This part is also consequent with a standard layer without time steps.

After flattening the output of CNN, the four states (direction, speed limit, speed, and traffic light state) are concatenated. In the following part, LSTM will function as a context descriptor on the sequence of images. Finally, a fully connected layer will map these into the driving control parameter.

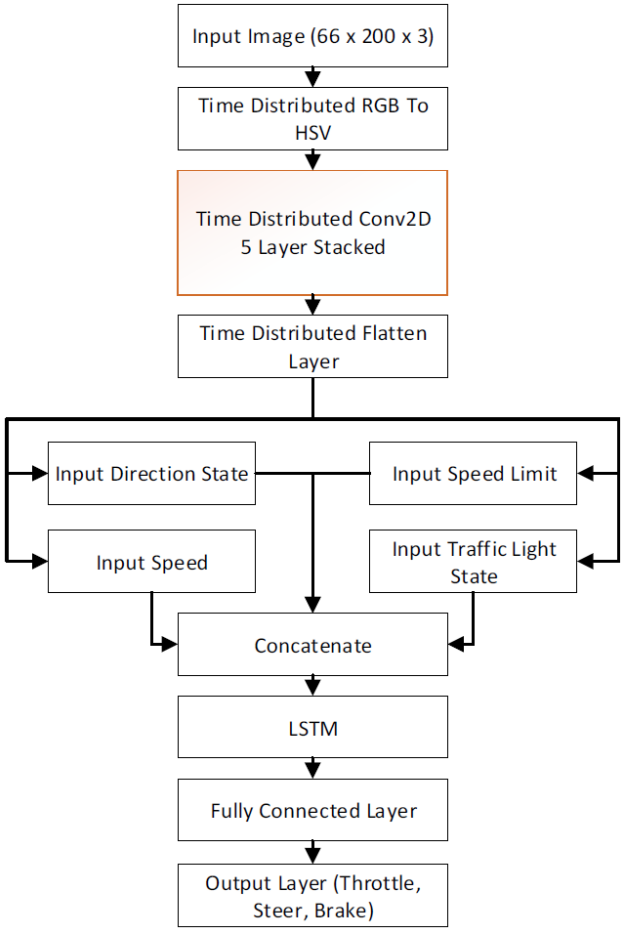


Fig. 4.3. Design of Time Distributed model which comprised of 5 stacked Conv2D layer and LSTM.

- **Training Data Acquisition**

Consider a controller that interacts with the environment over discrete time steps. At each time step t, the controller receives an observation o_t and takes an action a_t . The training data is a set of observation action pairs with respect to:

$$D = \{\langle ot, at \rangle\}_{t=1}^{N} \quad (15)$$

generated by the expert agent. The assumption is that the expert has accomplished the driving from a certain point until it reaches the destination. The autonomous agent will mimic the path or lane that the expert usually used to reach the goal.

We make our recorder to capture the driving images and another parameter. We use the sensor RGB or front view camera provided by Carla. To avoid redundancy in image acquisition, we sampled ten frames per second (FPS) for every episode.

There are three types of images that we captured, training data in default setup condition with another vehicle, without vehicle and noise added. These three differences in the dataset are necessary for the sake of covering all the possible positions of the agent so that it can recover from an unwanted situation or place. We acquired this data by randomly set up start location and stop location.

Fig. 4.4. The sample image, captured through the front view camera in Carla's environment.

4.2.5 Result And Analysis

In this section, we provide the result of our test simulation, which started from model training until experiment analysis.

- **Model Training and Validation**

After a long process of data acquisition, we obtained 101,360 images with detail such as frame number, speed, throttle, brake, speed limit, traffic light state,

direction, simulation time, etc. Then, we filter out into just six parameters, which have a composition of direction state, traffic light state, speed limit state, and speed as X, while throttle, steer and brake work as Y. Using Keras, we train our model with the following configuration.

Table 5. Time Distributed Based Model Configuration

Hyper Parameter	*Configuration*
Learning Rate	0.00012
Batch Size	16
Loss	Mean Square Error
Optimizer	Adam

In this training process, the total number of parameters is 802,199. We use the Early stopping module to reduce overfitting as well as get the best number of epochs in training before it stops improving. In our first experiment with the Nvidia model, the training ends after it reaches 40 epochs, with training loss 0.0054 and validation loss 0.024. The result shows that the model can converge very well. Furthermore, it can achieve training loss ~ validation loss, which means that the model is neither overfitting nor underfitting.

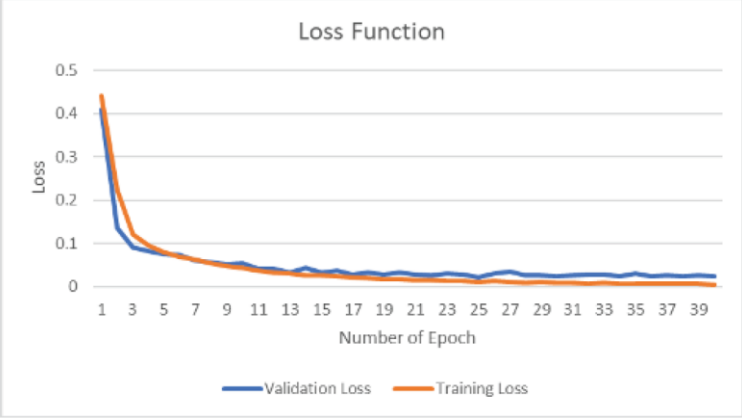


Fig. 4.5. Loss function graph of Time-distributed based model training.

- **Route Data Statistics**

We also compile data statistics of the route which the agent passes. Based on Figure 5, the road dominantly shows a straight path with some movement of steering angle. In **Fig. 6**, it also indicates that the PID or the ground truth agent is

braking quite often in the test route, as there are many vehicles around, even though it experiences red light not so often. On the other hand, the course mostly has a lane with a 30 Km/h speed limit with dominance in the lane-follow path.

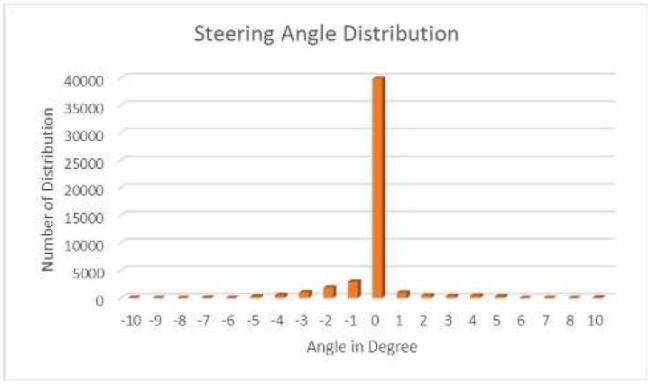


Fig. 4.6. Steering angle distribution in training data.

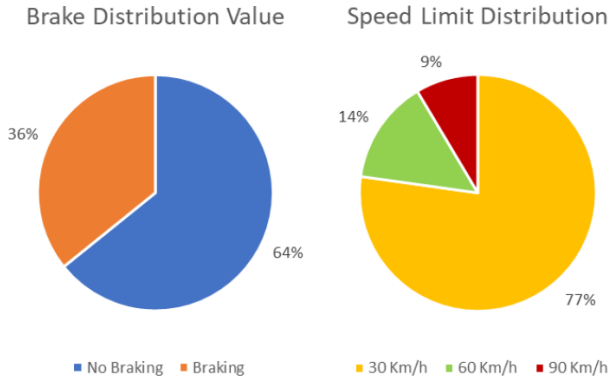


Fig. 4.7. Brake and speed limit distribution value on the designated routes in simulation.

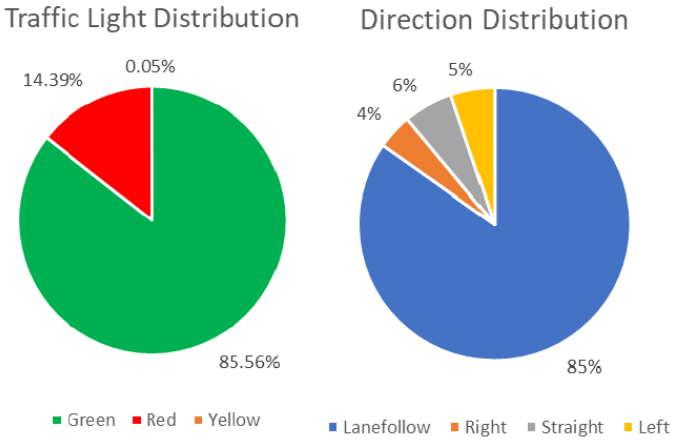


Fig. 4.8. Percentage of encountered traffic light distribution and direction distribution experienced by the agent on the designated routes.

- **Visualization Activations**

One of the means to know how our deep CNN can successfully classify or predict the input images is by knowing what our CNN model sees the input. We can visualize the intermediate actions with Keras model that we have trained and take batches of images as input. This representation also gives a view of how our CNN model decomposed images into a visual concept of features.

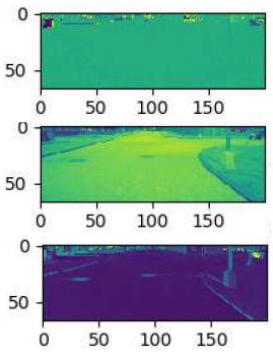


Fig. 4.9. The output of HSV converter from RGB images before getting into the CNN layer.

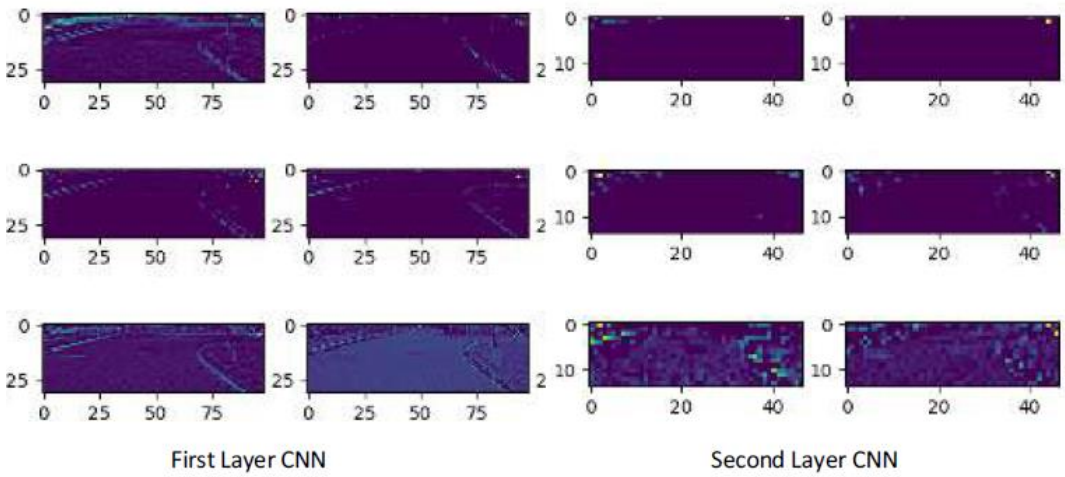


Fig. 4.10. Sample of visualization activation after pass-through from the first to the second layer of CNN.

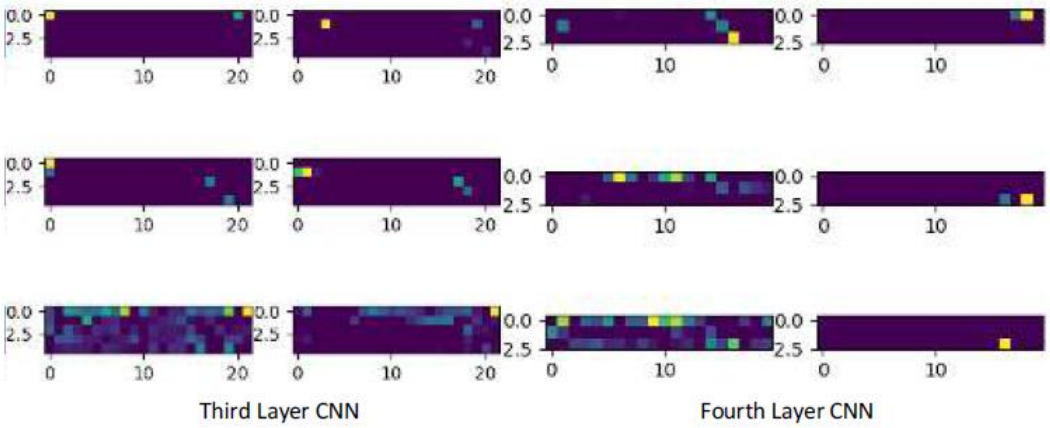


Fig. 4.11. Sample of visualization activation after pass through the third to the fourth layer of CNN.

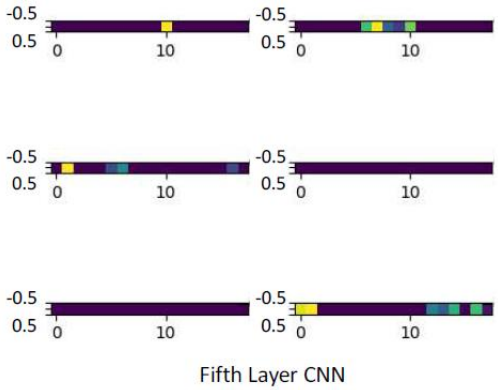


Fig. 4.12. Sample of visualization activation after pass through the fifth layer of CNN.

Fig. 4.13. Sample of output classification using the trained model.

- **Test Simulation**

We run the simulation and arbitrarily set up the route concerning the map of Town02. We use a PID based autonomous sensor (manually coded Artificial

Intelligent (AI)) as a ground truth. We evaluate the performances of time distribute model by comparing with ground truth, and CNN based **[73]** data test simulation. The data is sampled through ten of successful driving from setup position A to B. Through all of this round of the simulation, we calculate steering score, which we obtain from a mean of summing absolute steering value. There are also speed above and below the score, obtained through calculating the mean of the difference between speed and speed limit value.

A higher steering score means that the agent moves more agitated, and the speed score indicates haste of the agent. Consequently, a higher speed below the rating shows how good the agent can stick to the speed limit concerning the current lane. Conversely, higher in speed above score, point out that the car cannot keep up with the speed limit.

Table 6. Mean of performance comparison of time distributed model to ground truth data and stacked cnn model

Perf. Parameter	*Ground Truth*	*Time Distributed*	*Stacked CNN*
Steering Score	1.34	3.71	4.21
Speed Score	6.264	9.65	10.5
Speed Below Score	18.013	15.269	17.02
Speed Above Score	3.605	13.274	8.418
Real Time Duration (s)	141.303	109.1841	138.586
Total Crossed Lines	0	6	3

According to Table 2, ground truth result shows an excellent example of performance with zero total crossed lines, low steering score, and low-speed control. The agent can also control the speed limit problem very well, by having a high low speed below the score and low speed above score, which adheres to the traffic rule. Although time generally distributed inferior in comparison to this ground truth data, the model poses a better result in comparison to stacked CNN. It has a better steering score, and speed score compares to stacked CNN, but it has low control in the speed limit, which causes to cross lines many times. Consequently, the time

distributed agent shows a peculiar result in the time of reaching the destination. It remarks the best time to reach the goal. This unexpected behavior comes from how time distributed agent handles the speed in every lane with a different speed limit.

In the following table, a time distributed agent can achieve 93.978 s in the same route with good steering score and speed score. Nevertheless, it traverses the line 8 times with a high speed above score. Whereas stacked CNN can achieve the best time 98.29 with a slightly high steering score, but it is more stable by not crossing the line at all.

Table 7. The fastest test of an agent on a different model

Perf. Parameter	*Time Distributed*	*Stacked CNN*
Steering Score	3.9	4.4
Speed Score	8.91	8.61
Speed Below Score	14.83	13.14
Speed Above Score	14.42	8.29
Real Time Duration (s)	93.978	98.29
Total Crossed Lines	8	0

4.2.6 Conclusion And Future Works

In this paper, we develop and evaluate end-to-end time distributed based model, a combination of CNN FCN-LSTM with multiple input and states, and multiple-output prediction. There are four states included in this test, direction, speed limit, traffic light, and speed. There is also a parameter that we predict from all these states, Steering, Braking, and Throttling. Basically, from our experiment, these three parameters are sufficient to define the movement of the cars.

Experiments show that the time distributed agent can reach the destination exceptionally faster than the PID control based navigation and stacked CNN based model. As a result, it yields a low speed below the score and a higher speed above score. Though, it also displays that the agent is less agitated and better speed score than the stacked CNN model.

Overall, from the steering and speed score, the result indicates that time distributed model slightly better than the stacked CNN model. Still, it suffers from

speed limit management problems. Although finished faster in some cases is the top priority, stability in casual car driving would be far more critical.

On the other hand, we find that we should add more variables into the data set, such as weather and day (night, afternoon and morning), to know the robustness of time Distributed model toward a change of visual light perception and noisy environment. This robustness should also be tested with a stochastic nature non-player agent with a more comprehensive evaluation of driving performance.

Another thing that we concern about is the absence of pedestrians, which is also part of the city environment. While on our test, Carla 0.9.5 still not yet provides pedestrian features, so that in the future work, this part should be added.

CHAPTER 5: PAPER 4: Protein-Ligand Pair Binding Prediction Using Wide Resnet For Virtual Drug Screening

5.1 Foreword

This research is encouraged to be part of the artificial intelligence research agenda as part of dedication toward humanity, with a goal of accelerating lead compound discovery. In our experience, a big challenge in drug discovery is that it takes much time to preprocess raw data and train the model (a total of ~1.5 years). Therefore, the research paper presented will be a preliminary research approach to the protein-ligand binary classification case.

5.2 Published Paper

Protein-Ligand Pair Binding Prediction Using Wide Resnet For Virtual Drug Screening

Willy Dharmawan, and Hidetaka Nambo

5.2.1 Abstract

Deep Learning approaches have successfully addressed diverse problems, specifically drug discovery. Some models such as Front Propagated Wave, Multiple Neural Networks (NN) including Convolutional Neural Networks have been applied to different data representations of the input. Most of the algorithms seem to apply these model bases for heuristic purposes. Therefore, a comprehensive evaluation of the model structure and a combination of the residual network are necessary to improve the predictor performance further. Specifying the proper model hyperparameter in deep learning is needed for binary classification problems, especially in the case of drug-target binding affinity prediction. Some approaches can be exerted, such as filter size variation, adding more depth, model configuration exploration (learning rate, regularization function, backward algorithm, etc.), to changing the network width and residual function part. This paper focuses on finding the best hyperparameter setting on a wide ResNet model, including the depth and wide changing variability. From our experiment, adding the width of the network can improve the area under the curve to a 0.991 score compared to the

ResNet model with identity functions of 0.987. It is also shown that wide ResNet with B(3, 1, 3) configuration shows the best performance in our experiment set.

Keywords: Deep Learning, Drug Discovery, Wide Resnet, binary classification, Drug Target Binding Affinity Prediction.

5.2.2 Introduction

Over the past few years, an efficient way of mining large-scale chemistry data from vast amounts of available compound activity and biomedical data **[84, 85]** has become a crucial problem in Drug Discovery. Apart from the established methodology used in Quantitative Structure-Activity Relationship (QSAR), such as Support Vector Machine (SVM) **[86]**, Random Forest (RF) **[87]**, Neural Networks (NN) **[88]**, Deep Learning (DL) has started to be researched.

Basically, DL utilizes continuous improvement of computer power to accommodate the increased amounts of data. Compared to the previous methodology, which relies on single-layer NN, DL is more flexible. It can be tailored into multi-layer networks to solve a more complex problem.

Drug Discovery is a lengthy and costly process that uses various tools from various fields. Some areas **[89]**, including genomics, proteomics, cellular and organismic methodologies, are developed to facilitate this process. This molecular recognition between proteins or receptors with ligands plays a vital role in biological processes, such as enzyme catalysis. There are many ways DL can assess this recognition part. One of them is structural prediction which stimulates the grand challenge of the protein folding problem **[90]**.

A protein is a sequence formed by a combination of 20 amino acids. The protein folding problem dictates its three-dimensional atomic structure from this amino acid chain. This work is the first part of assessing how DL, which in our case is a three-dimensional Convolutional Neural Network (3D-CNN) based model, can capture the volumetric context of protein-ligand binding. This work adopts Ken Dill's **[91]**, Hydrophobic-Polar (HP) protein folding model that becomes features in defining protein and ligand structure.

The process of how we use these features consists of three phases, 3D coordinates construction or Voxelization, features symbolization, and 3D box modeling. Through the Protein Data Bank (PDB) data, we extract and interpolate the coordinates to create voxel data of protein. Each point in the grid coordinate will be mapped into an integer value that defines the HP symbol. Finally, we model it into a 3D box of protein-ligand binding by removing the unwanted point from a certain distance to the binding pocket between protein and ligand. These matrix features will become a dataset for a wide ResNet (WRN) model designed in this project.

The raw dataset comprises pairs of protein-ligand complex, taken from the pdbind version 2018 dataset **[92, 93, 94, 95, 96]** using the refined set, which is more selective **[94]** compared to the general set. To simplify the extraction and simulation, we filter into a single chain of amino acids in the dataset.

In this paper, we try to find the best hyperparameter setting on the residual network, including L1-layer input size, depth, and wide variability. Two types of residual blocks become our reference, considering minor errors on the CIFAR-10 test **[97]**, B(3,3), and B(3,1,3). These various residual block convolution operations are applied to the stacked 3D-CNN-based model. We evaluate and validate the model with binary cross-entropy and k-fold cross-validation. Stochastic gradient descent Nesterov is also used to improve training accuracy.

Besides, regardless to the protein-ligand binding overall structure, we also test the scaling ratio and filtering number of atom for the further evaluation of deep learning capability for the increase number of missing features. The range of each parameter is determined empirically. The range of scaling factor is 1 to 4, whereas the maximum distance from the center to outermost atom is 10 – 16.

Each different L1-layer input setting of the test experiments captures some performances parameter such as precision, sensitivity, and specificity, which are later used to determine F1 and Area Under Curve (AUC). From our experiments, WRN with a configuration of B(3,1,3) K4 Depth 10 shows the best result compared to another design with an F1 score of 0.9688 and AUC of 0.989. In addition, this configuration can achieve 0.99 AUC in the L1-64 configuration.

5.2.3 Related Works

Assessing the interaction between protein and ligand through scoring function based on forcefields or knowledge from existing protein-ligand complex structures is very important in molecular docking programs. One of the works using machine learning techniques is eMatchSite **[98]**. It offers robustness toward structure distortion and high accuracy of binding sites. However, it uses a template-based system, which may not work well on unseen data.

Binding site detection can also be solved using advanced deep learning, such as CNN. One of the examples is researched by Ragoza et al. **[99]**. He tried constructing protein-ligand binding coordinates into a 3D voxel with a resolution of 0.5 Å. By adopting a multi-layer CNN-based Caffe DL framework, his scoring outperformed Autodock Vina **[100]** on the CSAR inter-target pose-prediction dataset **[101]** but has bad performance in the intra-target ranking of poses. Following that, in the same year, A CNN-based protein-binding site predictor, Deepsite **[102]**, reports a better accuracy than Fpocket **[103]** and Concavity **[104]** on the sc-PDB database of the binding site.

Another approach to protein-ligand prediction, Marta M. **[105]** proposed a protein-ligand binding affinity predictor called Pafnuci. It works as a scoring function and reports a reliable prediction of relevant features, which outperforms all state-of-the-art functions tested by the CASF-2013 author **[106]**. In addition, a predictor on a CNN-based framework, KDEEP **[107]**, which includes the pharmacophoric properties of protein structure into voxel, can achieve a better score in binding affinity prediction than another method against the experimental value provided by PDBind **[108]**.

Finally, L. Pu et al. **[109]** published protein-ligand binding pocket classification using Deep3D. It successfully detects and classifies Nucleotide and Heme-binding sites with 95% accuracy. They achieve this result by employing a multi-layer 3D-CNN on a protein-binding pocket grid similar to VGG-network architecture.

Regardless of the primary purpose of ligand-protein binding pocket interaction prediction, which is identifying physiological ligands for orphans receptor **[108]**, our works concentrate on evaluating a deep learning model in capturing protein-ligand binding sites features through 3D coordinates of atom. Most of the paper does not comprehensively assess on DL sides and sticks to one framework of a deep learning network. While in our project, we are profoundly focusing on using the diverse hyperparameter setting of the network model and residual connection application for protein-ligand binary classification.

The reason for applying residual connection to the deep learning model is to overcome difficulties of the deeper NN model, such as exploding/vanishing gradient and degradation **[97]**. The transfer learning will also become more efficient because it can utilize the extracted features better than the inception architecture **[109]**. Also, the deployed residual link can speed up network convergence **[110]**.

5.2.4 Methodology

Assessing the interaction between protein and ligand through scoring function based on forcefields or knowledge from existing protein-ligand complex structures is very important in molecular docking programs. One of the works using machine learning techniques is eMatchSite **[98]**. It offers robustness toward structure distortion and high accuracy of binding sites. However, it uses a template-based system, which may not work well on unseen data.

- **HP Model**

Ken Dill proposed an on-lattice protein folding model in 1985 **[91]** to simplify the tertiary structure of PSP (Protein Structure Prediction). This HP model utilizes hydrophilic (attracted to water) and hydrophobic as a generalization of amino acids, which are represented as the two-symbol alphabet “H” (Hydrophobic) and “P” (Polar).

Table 8. Hydrophobic/Polar classification of the 20 amino acids.

Name	Sym.	Class	Name	Sym.	Class
Alanine	Ala	H	Leucine	Leu	H
Arginine	Cys	P	Lysine	Lys	P
Asparagine	Asn	P	Methionine	Met	H
Aspartic Acid	Asp	P	Phenylalanine	Phe	H
Cysteine	Cys	P	Pronine	Pro	H
Glutamic Acid	Glu	P	Serine	Ser	P
Glutamine	Gln	P	Threonine	Thr	P
Glycine	Gly	P	Trytophan	Trp	H
Histidine	His	P	Tyrosine	Tyr	P
Isoleucine	Ile	H	Valine	Val	H

- **Voxelization Protein-Ligand Binding Box**

We start constructing the protein-ligand complex pair structure using 3d coordinates in PDB data. After that, using the Table I model, we map the amino acids into HP representation. Later, these symbols will be encoded into integer encoding, which acts as features.

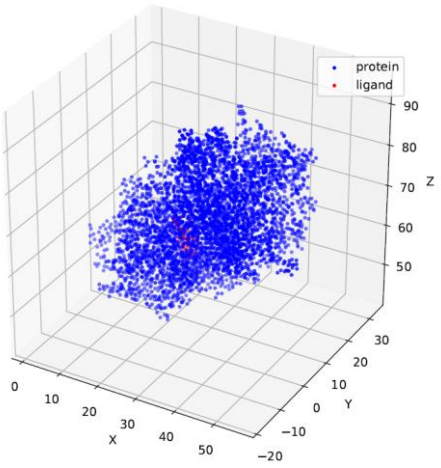


Fig. 5.1. Three dimensions construction Protein-Ligand Pair Complex on 1a28 into 3D grid structures.

The next stage is scaling and translating the protein-ligand pair voxel. This transformation is done to adjust the protein-ligand voxel for the grid box model. Finally, some grid points are filtered according to the set max distance.

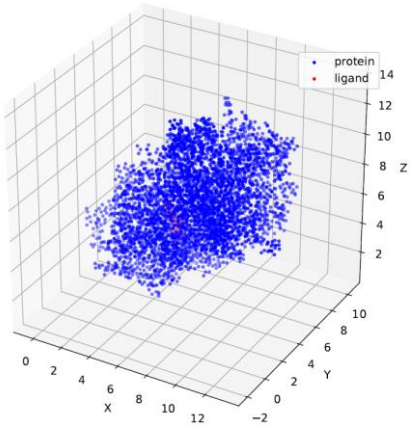


Fig. 5.2. Translation and scaling transformation of protein-ligand pair 1a28 voxel.

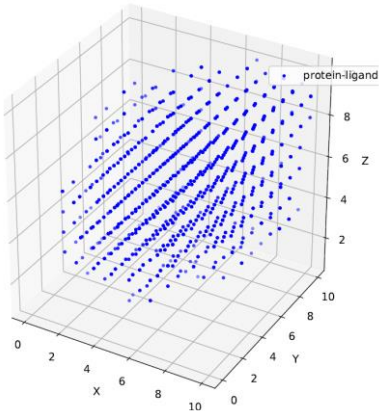


Fig. 5.3. Grid box of protein-ligand 1a28 pair voxel.

- **CNN Over Volume (3D CNN)**

CNN, introduced by LeCun **[111]**, is primarily used to process the data with a grid-like topology. These neural networks use convolution in place of general multiplication described in the artificial neuron section. Typically, a convolutional network comprises three stages **[112]**. The first stage, the convolutional layer, perform convolutions to yield a set of linear activation. In the second stage, the convolved features run into non-linear activation functions. Finally, through the pooling layer, the features are down-sampled.

Refer to Goodfellow et al. **[112]**, assume that a three-dimensional feature F convolves with a three-dimensional kernel K corresponding to the following formula,

$$K * F(i,j) = \sum_{C=0}^{2} \sum_{m}^{M} \sum_{n}^{N} K(m,n,c) I(i+m, i+n, c) \quad (16)$$

From the equation, we can infer that M, N, and C are height, width, and channel, while (i,j) are convolution pointers.

- **Wide-Resnet**

Introduced by S. Zagoruyko and N. Komodakis **[97]**, the wide residual network addresses the deep residual network, diminishing feature reuse, which makes the network very slow. This problem starts from identity mapping, which allows training very deep networks and becoming vulnerable to not learning anything during training. As the gradient flow over identity mapping or residual block, it is possible to avoid learning useful representations or get little information. They propose widening ResNet blocks to overcome this problem.

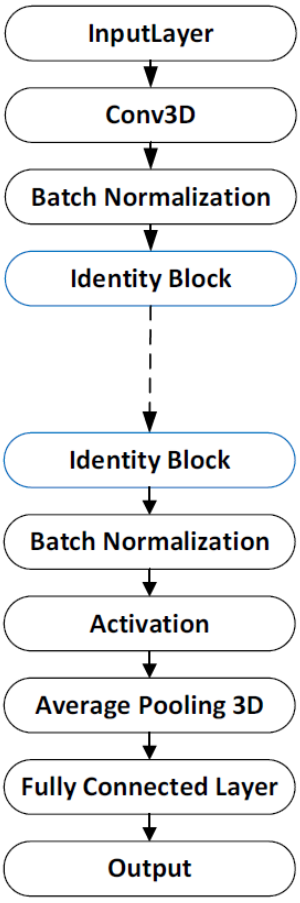


Fig. 5.4. The implemented WRN network model in this paper.

From their report **[97]**, WRN provides an effective way of improving performance compared to a deeper network. It shows a more efficient number of depth and processing time. For instance, their model improves over [30] by having 50 times fewer layers and is twice faster.

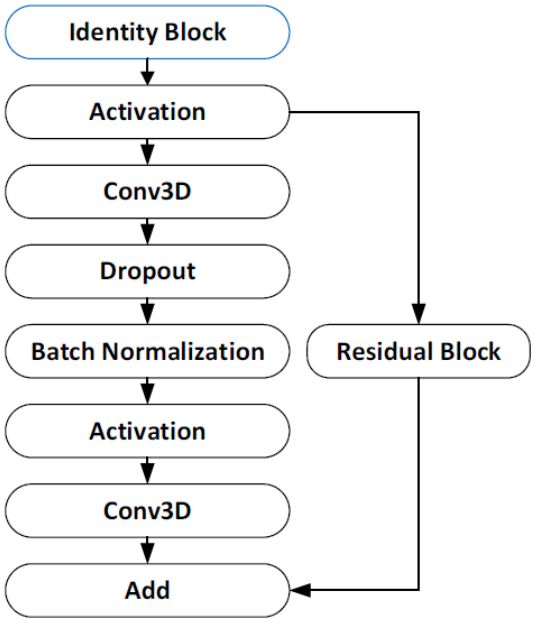


Fig. 5.5. Representation of Identity Block implemented in this work.

According to **Fig. 5.4**, We can state residual block with identity mapping as in **[97]**:

$$x_{i+1} = x_i + \mathcal{F}(x_i, \mathcal{W}_i) \quad (17)$$

where x_{i+1} and x_i respectively are input and output of the i-th unit in the residual network, $\mathcal{F}$ is residual function parametrized by $\mathcal{W}$. Report [14] has shown various residual block has been tested on CIFAR-10, B(3, 3), B(3, 1, 3), B(1, 3, 1), B(1, 3), B(3, 1), and B(3, 1, 1). Residual blocks of B(3,3), (3,1,3), and B(3,1) are the three best results in different block type tests, achieving a score of less than 6% error on the CIFAR-10 test.

- **Data Set Preparation**

As explained in the introduction section, the dataset is the 2018 version **[92, 93, 94, 95, 96]** which consists of 16,151 protein-ligand pair complexes, filtered out into 4,463 for the refined set. In our experiment, we filter out the single

polypeptide chain in the protein to ease the feature extraction of amino acid and protein-ligand interaction model training. Therefore, we sum up 3,496 protein-ligand pairs for training and validation. In addition, we also prepared 350 pairs of data for test sessions excluded from the training process.

- **Test Environment and Configuration**

Model training and testing were configured in a Linux environment with the GPU of Nvidia RTX A6000. The program was also deployed with a framework of TensorFlow-Keras in python 3.6.

On the other hand, according to **[97]** performance result, the experiments consist of three different model types with hyperparameter variability as follows:

1. B(3,3): original (basic) 3 x 3 convolution layer
2. B(3,1,3): original (basic) with additional 1x1 convolutional layer.
3. Basic Resnet with identity connection.

The model was trained under a hyperparameter of 1000 epochs, batch size of 128, and loss function of binary cross-entropy. Moreover, to get a better result, the test employed Stochastic Gradient Descent (SGD) Nesterov with a learning rate of 0.0001, momentum of 0.9, and decay rate of 0.0001. Finally, the model checkpoint feature is used to achieve the best training model.

5.2.5 Result and Analysis

- **B(3,3) Residual Connection**

All the models are tested in different hyperparameter settings, such as depth d (10, 16, 30) and widening factor k (2, 4, 8). The model is validated using k-fold cross-validation of 0.8 training and 0.2 testing data composition over three different L1 filters, 16, 64, and 256. Fig. 6 shows one of the model training with a configuration of k 2, d 10. The validation accuracy for each filter respectively is 0.941, 0.954, and 0.956, which shows that the trained model is neither underfitting nor overfitting.

In addition, based on the graph in Fig. 6, the validation test on each of the L1 filter sizes has closed each other, especially on the size of 64 and 256. This result

also applies to the different model settings. These indicate that the size of the L1 input doesn't significantly affect the model training accuracy and loss of B(3,3) residual connection.

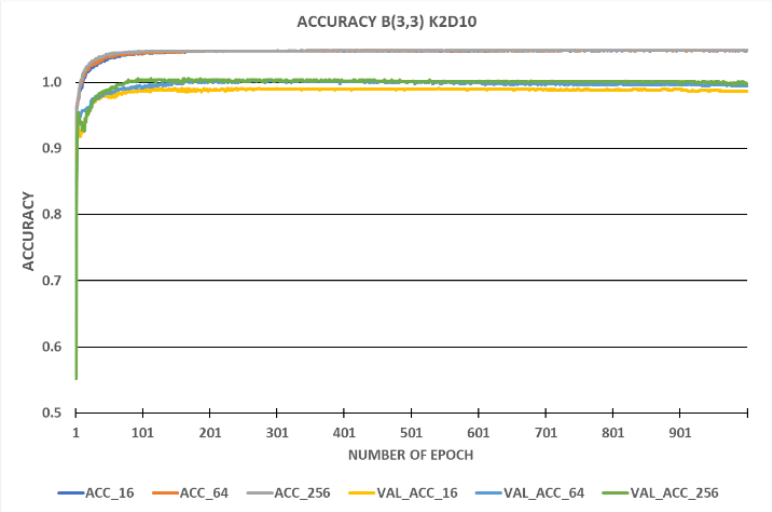


Fig. 5.6. B(3,3) residual function model accuracy comparison on the different L1 filters with ACC as accuracy and VAL as validation.

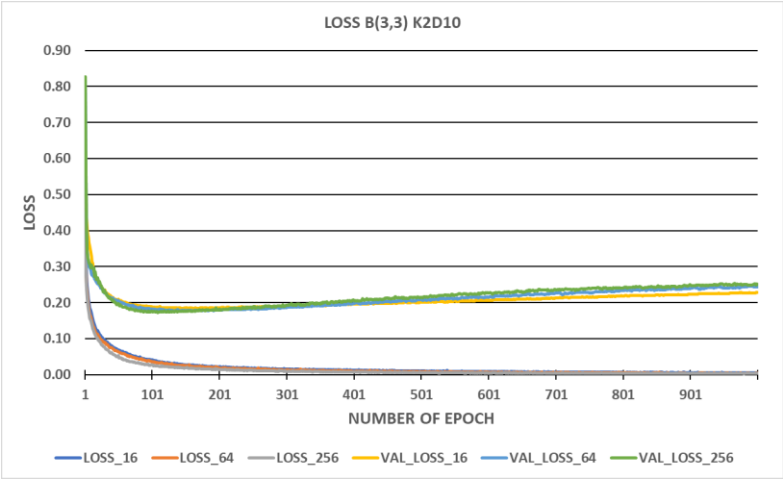


Fig. 5.7. B(3,3) residual function model loss comparison on the different L1 filters with ACC as accuracy and VAL as validation.

- **B(3,1,3) Residual Connection**

The same experiment configuration from the previous residual test is also applied in B(3,1,3) residual connection. Compared to B(3,3) residual connection, the accuracy discrepancy and loss of the L1-16 layer becomes clearer among other

sizes. The model converges toward the training data before 100 epoch and yields validation loss of 0.949, 0.962, and 0.96 for L1-16, L1-64, and L1-256.

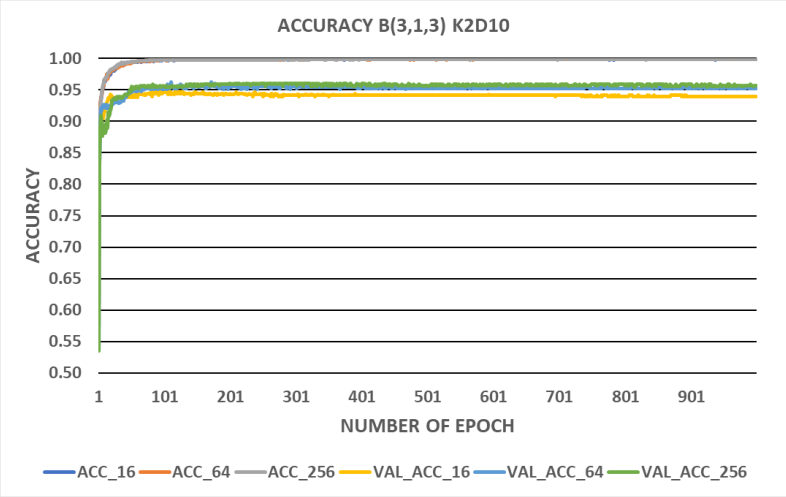


Fig. 5.8. B(3, 1, 3) residual function model accuracy comparison on the different L1 filters with ACC as accuracy and VAL as validation.

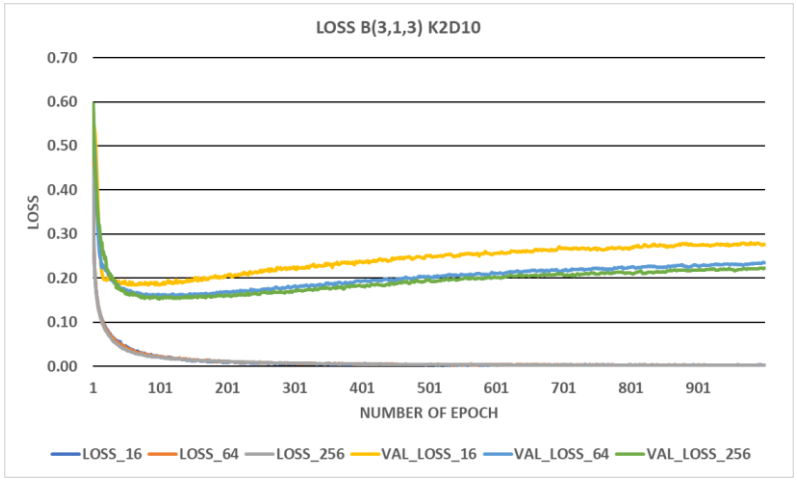


Fig. 5.9. B(3, 1, 3) residual function model loss comparison on the different L1 filters with ACC as accuracy and VAL as validation.

We can infer from **Fig. 7** and **Fig. 9** that a further increased number of epochs will make the line deviate from convergence. However, the saved model will be the last best update model because of the model checkpoint feature.

- **External Data Testing**

To understand which model can work well in capturing the protein-ligand binding site feature, we test the arbitrary chosen filtered data testing outside the

training and validation set with our best-saved model. Moreover, we also make a comparison to the identity residual or basic resnet.

Besides, in this premiere report, we restrict the grid box size to 8 x 8 x 8 x 2, which is smaller than the previous research **[108]**. All the atoms inside this grid box are mapped into the HP model to become the features.

The testing session used 350 pairs of protein-ligand data outside the training dataset. These data are augmented into positive and negative data with a ratio of 1:1. The program's execution applies arbitrarily chosen random seeds so that the result will be more consistent.

Table 9. Performance measurement of basic resnet with identity function (prec. = precision, sens. = sensitivity)

Residual Blocks	L1	Prec.	Sens.	F1	AUC
Resnet 10 Depth	16	0.9398	0.9828	0.9608	0.981
	64	0.9202	0.9885	0.95316	0.984
	256	0.95	0.9771	0.9633	0.987
Resnet 16 Depth	16	0.9109	0.994	0.9508	0.98
	64	0.935	0.988	0.961	0.975
	256	0.9202	0.988	0.953	0.982

After multiple model testing on the test dataset, we capture and calculate some key performances such as F1 and AUC. F1 score can be computed using precision and sensitivity, while AUC is from True Positive Rate (TPR)/Sensitivity and False Positive Rate (FPR).

Later, we filter out some best score performances for each residual connection (mark with orange color), ResNet D10 L1-256, B(3,3) K2D10 L1-64, B(3,3) K4D10 L1-256, and B(3, 1, 3) K4D10 L1-64-256. However, in the entire experiments, the residual connection of B(3,3) K4D10 L1-256 shows the best performance overall.

Referring to **[113]**, going deeper in the network means higher capacity, which should demonstrate improvement in experimental performance. However, based on our test, smaller resolutions of grid box cause degradation of performance of the higher number of layers. This case means more features are necessary to achieve better performance.

Table 10. Performance measurement of wrn with residual function of b(3,3)

(prec. = precision, sens. = sensitivity)

Residual Blocks	L1	Prec.	Sens.	F1	AUC
B(3,3) K_2_D_10	16	0.955	0.971	0.9631	0.987
	64	0.965	0.954	0.957	0.99
	256	0.945	0.988	0.966	0.988
B(3,3) K_2_D_16	16	0.934	0.982	0.958	0.983
	64	0.928	0.971	0.949	0.976
	256	0.95	0.977	0.963	0.985
B(3,3) K_4_D_10	16	0.9602	0.965	0.962	0.989
	64	0.939	0.982	0.9608	0.989
	256	0.971	0.965	0.9684	0.99
B(3,3) K_4_D_16	16	0.9301	0.988	0.958	0.979
	64	0.939	0.982	0.9608	0.985
	256	0.944	0.977	0.9606	0.983
B(3,3) K_6_D_10	16	0.954	0.954	0.954	0.989
	64	0.9447	0.977	0.9606	0.987
	256	0.9494	0.965	0.9575	0.987
B(3,3) K_6_D_16	16	0.9251	0.988	0.9558	0.983
	64	0.929	0.982	0.955	0.981
	256	0.9301	0.988	0.9584	0.979

Table 11. Performance measurement of wrn with residual function of b(3,1,3)

(prec. = precision, sens. = sensitivity)

Residual Blocks	L1	Prec.	Sens.	F1	AUC
B(3,1,3) K_2_D_10	16	0.965	0.954	0.959	0.987
	64	0.939	0.982	0.96	0.985
	256	0.96	0.971	0.965	0.987
B(3,1,3) K_2_D_16	16	0.9392	0.9714	0.955	0.986
	64	0.9255	0.9942	0.958	0.986
	256	0.9304	0.9942	0.9613	0.989
B(3,1,3) K_4_D_10	16	0.9402	0.9885	0.9637	0.991
	64	0.9453	0.9885	0.9664	0.99
	256	0.9606	0.9771	0.9688	0.989
B(3,1,3) K_4_D_16	16	0.9109	0.9942	0.9508	0.977
	64	0.9206	0.9942	0.956	0.979
	256	0.9354	0.9942	0.9639	0.984
B(3,1,3) K_6_D_10	16	0.9542	0.9542	0.9542	0.989
	64	0.9447	0.977	0.9606	0.987
	256	0.9494	0.9657	0.9575	0.987
B(3,1,3) K_6_D_16	16	0.939	0.982	0.96089	0.979
	64	0.925	0.994	0.958677	0.979
	256	0.939	0.9828	0.96089	0.982

An insufficient sample of data occurrence can also be seen in **Fig. 5.7** and **Fig. 5.9.** Longer convergence process of validation test leads to data underfit. Nevertheless, the best-saved model can overcome this problem by saving the last best model performance, which won't update the worse version.

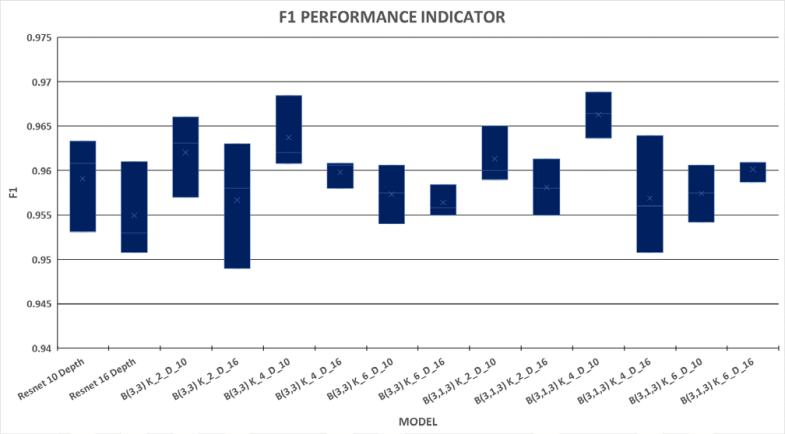


Fig. 5.10. Model loss comparison on the different L1 filters with ACC as accuracy and VAL as validation.

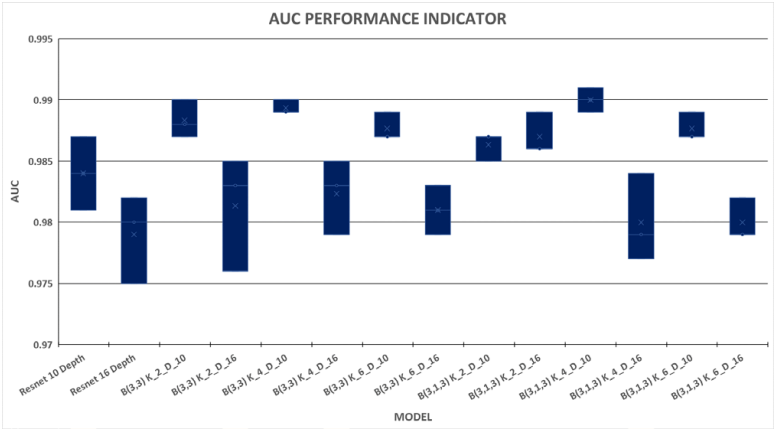


Fig. 5.11. Model loss comparison on the different L1 filters with ACC as accuracy and VAL as validation.

Table 12. Model performance top-5 f1 measurement ranking

Model	F1
B(3,1,3) K_4_D_10_L1-256	0.9688
B(3,3) K_4_D_10_L1-256	0.9684
B(3,1,3) K_4_D_10_L1-64	0.9664
B(3,3) K_2_D_10_L1-256	0.966
B(3,1,3) K_2_D_10_L1-256	0.965

Overall, fig. 10 shows that adding wide or k can improve F1 performance better than without k. ResNet with a wide of 4 and a residual network of B(3, 1, 3) can achieve the highest F1 score, 0.9688. Besides, wide-ResNet has better performance for each depth, which confirms that the protein-ligand binding prediction model can achieve better accuracy by going wide. Nonetheless, the degradation performance of each of the models can still be seen for an increased number of depths.

On the other performance parameter, **Fig. 5.11** shows the corresponding result to fig. 10. WRN with residual of B(3,1,3) parameter of k=4 and L1-filter size 16 yield 0.991 AUC. Despite its performance degradation, the wider network (k=6) also produces a closed result with k=4.

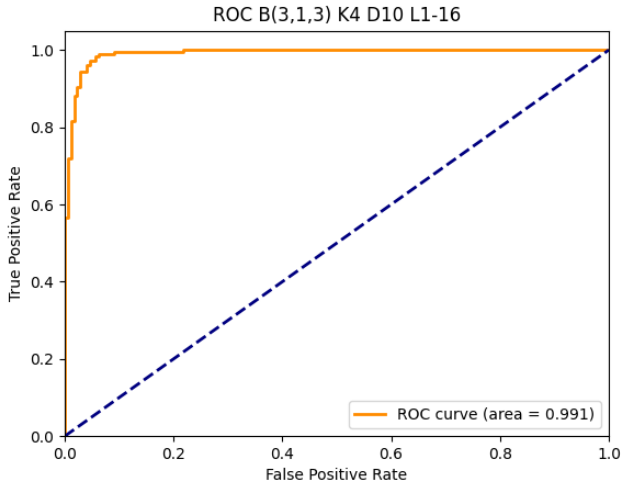


Fig. 5.12. ROC Graph of the highest AUC score in the experiments

5.2.6 Conclusion And Future Works

Our report focuses on the evaluation performance of multiple models of residual networks on protein-ligand binding binary classification problems with hyperparameter setting variability. Wider networks with two types (B(3,3) and B(3, 1, 3)) of residual connection are proposed **[97]**. Eventually, we also compare the performance result with the basic residual network (ResNet with identity function). The experiments use F1 and AUC as the key parameter for assessment.

Our tests indicate that making a wider network improved the model performance. This result can be seen in the overall performance of WRN on each

depth. The model of B(3,1,3) K_4_D_10 points out an AUC of 0.991 and F1 of 0.9688 on L1 sizes of 16 and 256.

On the other hand, performance degradation occurs on the deeper network (16 depth). Based on this practical result, we assume that the lower dimension of the voxel decreases the number of features which reduces the complexity of the dataset. Consequently, the shallower model depth shall perform better in this scenario.

Nevertheless, additional experiment scenarios should be inserted in future works. Variability of voxel size scenario can assist the work to get a better model hyperparameter, which affects the prediction performance. The second course of action can be adding noise to the dataset experiment. In this way, we can evaluate the model's robustness. Finally, adding another block module, such as Squeeze-and-Excitation (SE) block **[114]** or using a better way to scale up the network **[115]**, are also encouraged to get a more efficient and improved performance of the model.

CHAPTER 5: RESEARCH SUMMARY

5.1 Reflections

Despite the overstretching work presented in this paper, we aim to apply transdisciplinary, transformational, and translational artificial intelligence to show more impactful work and improve research quality. Even though a three-year doctoral degree is still insufficient to achieve more optimal output for all the papers presented in this dissertation, the research cornerstone has been established. Research sustainability should be carried out to have more optimized results on each of the case solutions.

5.2 Conclusion and Future Works

For each paper presented in this dissertation, we can summarize these papers as follows:

Paper 1: Tsunami Tide Prediction in Shallow Water Using Recurrent Neural Networks: Model Implementation in the Indonesia Tsunami Early Warning System.

- Our collaboration shows the successful implementation of multi-stage preprocessing, RNN regression, and z-score identification for artificial tsunami identification in shallow water areas of north Sipora.
- Despite its unavailability data test (shallow water tsunami), we accomplished this project using shallow water synthetic tsunami data. If actual shallow water tsunami data is available, we can extend this project into a classification case for further performance evaluation and analysis.
- The next direction of the deep learning model for this case is to use a multi-transformer **[63-65]** for another performance comparison.

Paper 3: End-to-End Time Distributed Convolution Neural Network Model for Self-Driving Car in Moderate Dense Environment.

- In this paper, we develop and evaluate an end-to-end time distributed-based model, a combination of CNN FCN-LSTM with multiple inputs and states and multiple output predictions.
- The agent (car) behavior is evaluated on particular pre-defined performance parameters, such as steering score, speed score, speed below the score,

speed above score, driving duration to the designated position, and total crossed traffic lines.

- For further development, multi-modal input transformers should be applied in more complex driving environments.
- To achieve transformative and translational research, simulation to reality becomes the next direction of this research, including collaboration for developing a real self-driving car.

Paper 4: Protein-Ligand Pair Interaction Prediction Using Wide Resnet For Virtual Drug Screening.

- This preliminary paper has shown the potential of deep learning ability in binary classification for protein-ligand binding prediction. Despite the preliminary research approach to this problem, we contribute to new post-processing protein-ligand pdbbind. The subsequent publication will attach the validation of this dataset through benchmarking with another available dataset.
- This research becomes the cornerstone of our research in drug discovery. We are still progressing this research with additional features and algorithm improvements.

5.3 Contribution as Co-author Papers

In all co-authored papers, I was involved as lead researcher adviser for the majority, recommending the research direction. Moreover, I was also involved in the literature review, writing, and analysis of all these papers. In addition, in this autonomous surface vehicle, I was involved in the majority of the progress of the translational development of the surface vehicle prototype. The target goal of this creation is to implement surface autonomous vehicles virtually and practically in a real-world environment.

REFERENCES

[1] Sarker IH. Machine learning: Algorithms, real-world applications and research directions. SN Computer. Science. 2021;2(3):1–21.

[2] Sarker IH, Kayes ASM, Badsha S, Alqahtani H, Watters P, Ng A. Cybersecurity data science: an overview from machine learning perspective. J Big Data. 2020;7(1):1–29.

[3] Xin Y, Kong L, Liu Z, Chen Y, Li Y, Zhu H, Gao M, Hou H, Wang C. Machine learning and deep learning methods for cybersecurity. IEEE Access. 2018;6:35365–81.

[4] Cao, L. Trans-AI/DS: transformative, transdisciplinary and translational artificial intelligence and data science. Int J Data Sci Anal 15, 119–132 (2023). https://doi.org/10.1007/s41060-023-00383-y

[5] Karhunen J, Raiko T, Cho KH. Unsupervised deep learning: a short review. In: Advances in independent component analysis and learning machines. 2015; p. 125–42.

[6] Renn, O.: Transdisciplinarity: synthesis towards a modular approach. Futures 130, 102744 (2021)

[7] Butler, D.: Translational research: crossing the valley of death. Nature 453, 840–842 (2008)

[8] Dankwa-Mullan, I., Rhee, K., Stoff, D., Pohlhaus, J., Sy, F., Stinson, N.J., Ruffin, J.: Moving toward paradigm-shifting research in health disparities through translational, transformational, and transdisciplinary approaches. Am. J. Public Health 100(Suppl 1), 19–24 (2010)

[9] Trevors, J.T., Pollack, G.H., Saier, M.H., Masson, L.: Transformative research: definitions, approaches and consequences. Theory Biosci. 131, 117–123 (2012)

[10] Sianipar, Corinthias P. M. & Dowaki, Kiyoshi & Yudoko, Gatot. (2015). Technological solution for vulnerable communities: Questioning the sustainability of Appropriate Technology. IOP Conference Series Earth and Environmental Science. 23. 012008. 10.1088/1755-1315/23/1/012008.

[11] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang et al., “End to end learning for self-driving cars,” arXiv preprint arXiv:1604.07316, 2016.

[12] Petneházi G (2019) Recurrent neural networks for time series forecasting. https://doi.org/10.48550/arXiv.1901.00069

[13] Smyl S (2020) A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. Int J Forecast 36(1):75–85. https://doi.org/10.1016/j.ijforecast.2019.03.017

[14] Bradbury J, Merity S, Xiong C, Socher R (2016) Quasi-recurrent neural networks. https://doi.org/10.48550/arXiv.1611.01576

[15] Willy Dharmawan and Hidetaka Nambo. 2020. End-to-End Xception Model Implementation on Carla Self Driving Car in Moderate Dense Environment. In Proceedings of the 2019 2nd Artificial Intelligence and Cloud Computing Conference (AICCC '19). Association for Computing Machinery, New York, NY, USA, 139–143. https://doi.org/10.1145/3375959.3375969

[16] Parekh, D.; Poddar, N.; Rajpurkar, A.; Chahal, M.; Kumar, N.; Joshi, G.P.; Cho, W. A Review on Autonomous Vehicles: Progress, Methods and Challenges. Electronics 2022, 11, 2162. https://doi.org/10.3390/electronics11142162

[17] Indonesian National Board for DisasterManagement (2018) Laporan kinerja tahun 2018. https://bnpb.go.id/uploads/24/laporankinerja-bnpb-2018.pdf

[18] Harig S, Immerz A, Griffin J, Weber B, Babeyko A, Rakowsky N, Hartanto D, Nurokhim A, Handayani T, Weber R et al (2019) The tsunami scenario database of the Indonesia Tsunami EarlyWarning System (InaTEWS): evolution of the coverage and the involved modeling approaches. Pure Appl Geophys 177:1379–1401

[19] Center of Electronic (2020) BPPT: term of reference: program penguatan dan pengembangan Indonesia Tsunami Early Warning System (Ina-TEWS). Puspiptek Serpong, South Tangerang, Indonesia

[20] Singh SC, Hananto N, Mukti M, Robinson DP, Das S, Chauhan A, Carton H, Gratacos B, Midnet S, Djajadihardja Y et al (2011) Aseismic zone and earthquake segmentation associated with a deep subducted seamount in Sumatra. Nat Geosci 4(5):308–311

[21] Zkrausk (1986) Ambient noise in shallow water: a survey of the unclassified literature. https://apps.dtic.mil/sti/tr/pdf/ADA167696.pdf

[22] Knobles D, Joshi S, Gaul R, Graber H,Williams N (2008) Analysis of wind-driven ambient noise in a shallow water environment with a sandy seabed. J Acoust Soc Am 124(3):157–162

[23] Box GE, Jenkins GM (1976) Time series analysis, control, and forecasting. Holden Day, San Francisco, CA 3226(3228):10

[24] Gardner ES Jr (1985) Exponential smoothing: the state of the art. J Forecast 4(1):1–28

[25] Winters PR (1960) Forecasting sales by exponentially weighted moving averages. Manag Sci 6(3):324–342

[26] Harvey AC (1990) Forecasting, structural time series models and the Kalman filter. Cambridge University Press, Cambridge. https://doi.org/10.1017/CBO9781107049994

[27] Tealab A (2018) Time series forecasting using artificial neural networks methodologies: a systematic review. Fut Comput Inform J 3(2):334–340

[28] Granger CW, Terasvirta T et al (1993) Modelling non-linear economic relationships. OUP Catalogue, Oxford

[29] Clements MP, Franses PH, Swanson NR (2004) Forecasting economic and financial time-series with non-linear models. Int J Forecast 20(2):169–183

[30] Teräsvirta T (1994) Specification, estimation, and evaluation of smooth transition autoregressive models. J Am Stat Assoc 89(425):208–218

[31] Haider SA, Naqvi SR, Akram T, Umar GA, Shahzad A, Sial MR, Khaliq S, KamranM(2019) LSTM neural network based forecasting model for wheat production in Pakistan. Agronomy 9(2):72

[32] Naqvi SR, Akram T, Iqbal S, Haider SA, Kamran M, Muhammad N (2018) A dynamically reconfigurable logic cell: from artificial neural networks to quantum-dot cellular automata. Appl Nanosci 8(1):89–103

[33] Naqvi SR, Akram T, Haider SA, Kamran M, Shahzad A, Khan W, Iqbal T, Umer HG (2018) Precision modeling: application of metaheuristics on current-voltage curves of superconducting films. Electronics 7(8):138

[34] Rather AM, Agarwal A, Sastry V (2015) Recurrent neural network and a hybrid model for prediction of stock returns. Expert Syst Appl 42(6):3234–3241

[35] Schmidt RM (2019) Recurrent neural networks (RNNs): a gentle introduction and overview. https://doi.org/10.48550/arXiv.1912.05911

[36] Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780

[37] Tan M, Santos Cd, Xiang B, Zhou B (2015) LSTM-based deep learning models for non-factoid answer selection. arXiv preprint arXiv:1511.04108

[38] Cho K, Merrienboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder–decoder for statistical machine translation. https://doi.org/10.48550/arXiv.1912.05911

[39] Lai G, Chang W-C, Yang Y, Liu H (2018) Modeling long- and short-term temporal patterns with deep neural networks. https://doi.org/10.48550/arXiv.1703.07015

[40] Shih S-Y, Sun F-K, Lee H-y (2019) Temporal pattern attention for multivariate time series forecasting. https://doi.org/10.48550/arXiv.1809.04206

[41] Dudek G, Smyl S, Pełka P (2022) Recurrent neural networks for forecasting time series with multiple seasonality: a comparative study. https://doi.org/10.48550/arXiv.2203.09170

[42] Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. https://doi.org/10.48550/arXiv.1409.3215

[43] Yin W, Kann K, Yu M, Schütze H (2017) Comparative study of CNN and RNN for natural language processing. https://doi.org/10.48550/arXiv.1702.01923

[44] Bradbury J, Merity S, Xiong C, Socher R (2016) Quasi-recurrent neural networks. https://doi.org/10.48550/arXiv.1611.01576

[45] 31. Grilli ST, Tappin DR, Carey S,Watt SF,Ward SN, Grilli AR, Engwell SL, Zhang C, Kirby JT, Schambach L et al (2019) Modelling of the tsunami from the December 22, 2018 lateral collapse of Anak Krakatau volcano in the Sunda Straits, Indonesia. Sci Rep 9(1):1–13

[46] McCloskey J, Antonioli A, Piatanesi A, Sieh K, Steacy S, Nalbant S, Cocco M, Giunchi C, Huang J, Dunlop P (2008) Tsunami threat in the Indian Ocean from a future megathrust earthquake west of Sumatra. Earth Planet Sci Lett 265(1–2):61–81

[47] Mofjeld H (1997) Tsunami detection algorithm. http://nctr.pmel.noaa.gov/tda_documentation.html

[48] Beltrami GM (2008) An ANN algorithm for automatic, real-time tsunami detection in deep-sea level measurements. Ocean Eng 35(5–6):572–587

[49] Barman R, Prasad Kumar B, Pandey PC, Dube SK (2006) Tsunami travel time prediction using neural networks. Geophys Res Lett 33(16). https://doi.org/10.1029/2006GL026688

[50] Romano M, Liong S-Y, Vu MT, Zemskyy P, Doan CD, Dao MH, Tkalich P (2009) Artificial neural network for tsunami forecasting. J Asian Earth Sci 36(1):29–37

[51] Fauzi A, Mizutani N (2020) Machine learning algorithms for realtime tsunami inundation forecasting: a case study in Nankai region. Pure Appl Geophys 177(3):1437–1450. https://doi.org/10.1007/s00024-019-02364-4

[52] Negarestani A, Setayeshi S, Ghannadi-Maragheh M, Akashe B (2002) Layered neural networks based analysis of radon concentration and environmental parameters in earthquake prediction. J Environ Radioact 62(3):225–233

[53] Ozerdem MS, Ustundag B, Demirer RM (2006) Self-organized maps based neural networks for detection of possible earthquake precursory electric field patterns. Adv Eng Softw 37(4):207–217

[54] Ni Y, Zhou X, Ko J (2006) Experimental investigation of seismic damage identification using PCA-compressed frequency response functions and neural networks. J Sound Vib 290(1–2):242–263

[55] Bhandarkar T, Satish N, Sridhar S, Sivakumar R, Ghosh S et al (2019) Earthquake trend prediction using long short-term memory RNN. Int J Electr Comput Eng 9(2):1304–1312

[56] Wiegel RL (1964) Oceanographical engineering. Prentice-Hall, Inc., Englewood Cliffs, NJ

[57] Le Cun Y, Jackel LD, Boser B, Denker JS, Graf HP, Guyon I, Henderson D, Howard RE, Hubbard W (1989) Handwritten digit recognition: applications of neural network chips and automatic learning. IEEE Commun Mag 27(11):41–46

[58] GoodfellowI, BengioY, Courville A, BengioY(2016) Deep learning, vol 1. MIT Press, Cambridge

[59] Hochreiter S (1991) Untersuchungen zu dynamischen neuronalen netzen. Diploma, Technische Universität München 91(1)

[60] Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. IEEE Trans Neural Netw 5(2):157–166

[61] Chen RC, Güttel S (2021) A comparison of LSTM and GRU networks for learning symbolic sequences. CoRR arxiv:2107.02248

[62] Imamura F, Yalciner A, Ozyurt G (2006) Tsunami modelling manual (tunami model). IOC Manuals and Guides (30)

[63] Wu N, Green B, Ben X, O'Banion S (2020) Deep transformer models for time series forecasting: the influenza prevalence case. CoRR arxiv:2001.08317

[64] Zhou H, Zhang S, Peng J, Zhang S, Li J, Xiong H, ZhangW(2020) Informer: Beyond efficient transformer for long sequence timeseries forecasting. CoRR arxiv: 2012.07436

[65] Grigsby J, Wang Z, Qi Y (2021) Long-range transformers for dynamic spatiotemporal forecasting. CoRR arXiv:2109.12218

[66] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, “Deepdriving: Learning affordance for direct perception in autonomous driving,” in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp.

[67] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue et al., “An

[68] empirical evaluation of deep learning on highway driving,” arXiv preprint arXiv:1504.01716, 2015.

[69] A. Gurghian, T. Koduri, S. V. Bailur, K. J. Carey, and V. N. Murali, Deeplanes: End-to-end lane position estimation using deep neural networks,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2016, pp. 38–45.

[70] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, “3d traffic scene understanding from movable platforms,” IEEE transactions on pattern analysis and machine intelligence, vol. 36, no. 5, pp. 1012–1025, 2014.

[71] H. Zhang, A. Geiger, and R. Urtasun, “Understanding high-level semantics by modeling traffic patterns,” in Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 3056–3063.

[72] Z. Yang, Y. Zhang, J. Yu, J. Cai, and J. Luo, “End-to-end Multi-Modal Multi-Task Vehicle Control for Self-Driving Cars with Visual Perceptions,” arXiv:1801.06734v2, 2018.

[73] M. Bojarski, et al., “End to end learning for self-driving cars,” arXiv preprint arXiv:1604.07316, 2016.

[74] H. Xu, Y. Gao, F. Yu, and T. Darrell, “End-to-end learning of driving models from large-scale video datasets,” arXiv preprint arXiv:1612.01079, 2016.

[75] T. Glasmachers, “Limits of end-to-end learning,” arXiv preprint arXiv:1704.08305, 2017.

[76] A.C. Serban, E.. Poll, and J. Visser, “A Standard Driven Software Architecture for Fully Autonomous Vehicles,” 2018 IEEE International Conference on Software Architecture Companion (ICSAC), IEEE, 2018.

[77] D. A. Pomerleau, “Alvinn: An autonomous land vehicle in a neural network,” in Advances in neural information processing systems, 1989, pp. 305–313.

[78] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, 2012, pp. 1097–1105.

[79] M. Bojarski, et al., “Explaining how a deep neural network trained with end-to-end learning steers a car,” arXiv 2017, arXiv:1704.07911.

[80] F. Codevilla, M. Muller, A. Lopez, V. Koltun, and A. Dosovitskiy, “End-to-end Driving via Conditional Imitation Learning,” International Conference on Robotics and Automation (ICRA), 2018.

[81] S. Chowdhuri, T. Pankaj, and K. Zipser, “MultiNet: Multi-Modal Multi-Task Learning for Autonomous Driving,” arXiv:1709.05581, 2019.

[82] A. Mehta, S. Adithya, and S. Anbumani, “Learning end-to-end autonomous driving using guided auxiliary supervision,” arXiv 2018, arXiv:1808.10393.

[83] A. Dosovitskiy, G. Ros, F. Codevilla, A. L´opez, and V. Koltun, “CARLA: An open urban driving simulator,” In Conference on Robot Learning (CoRL), 2017.

[84] G. Papadatos, A. Gaulton, A. Hersey, and J. P. Overington, “Activity, assay and target data curation and quality in the ChEMBL database,” J. Comput. Aided Mol. Des., 29 (2015), pp. 885-896.

[85] S. Kim, et al, “PubChem substance and compound databases,” Nucleic Acids Res., 44 (2016), pp. D1202-D1213.

[86] C. Cortes, and V. Vapnik, “Support-vector networks,” Mach. Learn., 20, 1995, pp. 273-297.

[87] T.K. Ho, “The random subspace method for constructing decision forests,” IEEE Trans. Pattern Anal. Mach. Intell., 20 (1998), pp. 832-844.

[88] D.W. Salt, N. Yildiz, D.J. Livingstone, and C.J. Tinsley, “The use of artificial neural networks in QSAR,” Pestic. Sci., 36 (1992), pp. 161-170.

[89] Y. M. Jhanker, M. F. Kadir, R. I. Khan and R. Hasan, “Proteomics in Drug Discovery,” Journal of Applied Pharmaceutical Science, 2012. doi: 10.7324/JAPS.2012.2801.

[90] K. A. Dill, S. B. Ozkan, M. S. Shell, and T. R. Weikl. The protein folding problem. Annual review of biophysics 2008, 37, 289–316. doi:10.1146/annurev.biophys.37.092707.153558.

[91] K.A. Dill, “Theory for the folding and stability of globular proteins,” Biochemistry 1985, doi: 10.1021/bi00327a032.

[92] Z.H. Liu, et al. Acc. Chem. Res.2017, 50, 302-309. (PDBbind v.2016)

[93] Z.H. Liu, et al. Bioinformatics, 2015, 31, 405-412. (PDBbind v.2014)

[94] L. Yan, et al. J. Chem. Inf. Model.,2014, 54, 1700-1716. (PDBbindv.2013 & CASF-2013)

[95] T. J. Cheng, et al. J. Chem. Inf. Model.,2009, 49, 1079-1093. (PDBbindv.2007 & CASF-2007)

[96] R. X. Wang, et al. J. Med. Chem. 2005, 48, 4111-4119; J. Med. Chem.2004, 47, 2977-2980. (early versions)

[97] S. Zagoruyko, N. Komodakis, “Wide Residual Networks,” arXiv:1605.07146v4 [cs.CV]

[98] M. Brylinski, “eMatchSite: sequence order-independent structure alignments of ligand binding pockets in protein models,” PLoS Comput Biol. 2014, doi: 10.1371/journal.pcbi.1003829.

[99] M. Ragoza, et al., “Protein–ligand scoring with convolutional neural networks,” J. Chem. Inf. Model. 57, 2017, pp. 942-957.

[100] O. Trott, A.J. Olson, “AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient

optimization, and multithreading," J. Comput. Chem., vol. 31, 2010, pp. 455-461.

[101] [18] J.B. Dunbar Jr, et al, "CSAR benchmark exercise of 2010: selection of the protein–ligand complexes," J. Chem. Inf. Model., vol. 51, 2011, pp. 2036-2046.

[102] J. Jimenez, S. Doerr, G. Martinez-Rossell, A.S. Rose, and G.D. Fabritiis, "DeepSite: protein-binding site predictor using 3D-convolutional neural networks," J. Bioinformatics, 2017, doi: 10.1093/bioinformatics/btx350.

[103] V.L. Guilloux, P. Schmidtke, and P. Tuffery, "Fpocket: an open source platform for ligand pocket detection," BMC Bioinformatics 2009, doi: 10.1186/1471-2105-10-168.

[104] J.A. Capra, R.A. Laskowski, J.M. Thornton, M Singh M, T.A. Funkhouser, "Predicting protein ligand binding sites by combining evolutionary sequence conservation and 3D structure," PLoS Comput Biol. 2009, doi: 10.1371/journal.pcbi.1000585.

[105] M.M. Stepniewska-Dziubinska, P. Zielenkiewicz, and P. Siedlecki, "Development and evaluation of a deep learning model for protein–ligand binding affinity prediction," J. Bioinformatics, vol. 34, 2018, pp. 3666-3674, doi: 10.1093/bioinformatics/bty374.

[106] J. Jimenez, M. Skalic, G. Martinez-Rosell, and G. D Fabritiis , "KDEEP: Protein-Ligand Absolute Binding Affinity Prediction via 3D-Convolutional Neural Networks," J Chem Inf Model, 2018, doi: 10.1021/acs.jcim.7b00650.

[107] Z. Liu, Y. Li, Han L, Li J, Liu J, Zhao Z, et al. PDB-wide collection of binding data: current status of the PDBbind database. Bioinformatics, 2015, doi10.1093/ bioinformatics/btu626.

[108] L. Pu, R.G. Govindaraj, J.M. Lemoine, H.C. Wu, and M. Brylinski, "DeepDrug3D: Classification of ligand-binding pockets in proteins with a convolutional neural network," PLoS Comput Biology 15(2): e1006718, 2019, doi: 10.1371/journal.pcbi.1006718.

[109] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” CoRR, abs/1512.03385, 2015.

[110] C. Szegedy, S. Ioffe, and V. Vanhoucke, “Inception-v4, inception resnet and the impact of residual connections on learning. abs/1602.07261, 2016.

[111] Y. LeCun, et al.,” Handwritten digit recognition: Applications of neural networks chipsand automatic learning,” Proceedings of the IEEE, 1998 86(11):2278–2324.

[112] I. Goodfellow and Y. Bengio and A. Courville,” Deep Learning,” MIT Press 2016.

[113] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” CoRR, abs/1603.05027, 2016.

[114] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[115] M. Tan, and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” International conference on machine learning. PMLR, 2019.