Empirical Comparison of Deep Learning Algorithm Performances on Rapidminer and Tensorflow for Classification Problems

1st Utomo Pujianto
Electrical Engineering and Informatics Department
State University of Malang
Malang, Indonesia
utomo.pujianto.ft@um.ac.id

2nd Alvian Rahmadani Saputra
Electrical Engineering and Informatics Department
State University of Malang
Malang, Indonesia
alvian.rahmadani.2005356@students.um.ac.id

***Abstract*— The usage of deep learning algorithms in machine learning has been attracting huge attention in various scientific and industrial applications. This has triggered the emergence of many frameworks used to implement deep learning algorithms. This research aims to conduct an empirical test and comparison between the performance of deep learning algorithms implemented in the RapidMiner and TensorFlow in the context of machine learning. Empirical experimental methods are carried out using datasets relevant to classification problems to analyze the performance of deep learning algorithms. Evaluation is conducted by evaluating performance parameters such as accuracy, precision, recall and execution time. Using 50 datasets, the statistical test results show a significant performance difference between the deep learning models in RapidMiner and TensorFlow, with the TensorFlow model having an advantage in average accuracy and precision of 0.908 and average recall of 0.906. Meanwhile RapidMiner is superior in model training and validation execution time, with an average time of 1.01 seconds compared to 110.98 seconds for TensorFlow. The conclusions of this research provide valuable insight into selecting deep learning platforms and algorithms that suit specific needs in machine learning.**

Keywords — RapidMiner, TensorFlow, comparison, classification, statistic test

I. INTRODUCTION

In recent years, machine learning has been used to enable computers to effectively perform activities typically carried out by humans [1]. This is possible because the algorithms used in machine learning can learn from the provided data and make predictions on previously unseen data consistently [2]. Initially, the use of machine learning technology was limited to science laboratories, but now this technology is widely and commercially used. Many industry sectors implement these algorithms, such as healthcare, education, finance, and others [3]. According to Berggren, et al., the advancement of machine learning technology is due to the rapid development of the algorithms used and increasingly complex neural network models [4].

One widely used algorithm, the deep learning algorithm, uses concepts based on artificial neural networks (ANNs) that outperform traditional machine learning models in many applications [5]. With the advent of increasingly large and complex data, deep learning has become a very effective solution in dealing with data science problems, particularly classification problems.

The deep learning algorithm utilizes multiple layers that can amplify important aspects of the input to distinguish features and suppress irrelevant variations, making this algorithm effective for classification problems [5]. For its application in data science, this algorithm is implemented into various data science frameworks and software platforms.

TensorFlow is one of the most popular and widely used frameworks by the community of machine learning researchers and practitioners [6]. In addition to TensorFlow, there are various other platforms and tools that are also used in deep learning implementations, such as RapidMiner. RapidMiner is a data analysis platform that allows users to easily build and deploy machine learning models without the need for in-depth programming knowledge.

Although there are many deep learning algorithm platforms and frameworks available, each has certain advantages and disadvantages. This is especially true for deep learning models trained on TensorFlow and RapidMiner. Both have the capability to train deep learning models, but the performance levels of the models can vary in certain cases. Therefore, it is important to conduct empirical comparisons to understand their relative performance in the context of classification problems. The results of this research can provide valuable insights for data science practitioners in selecting the most suitable algorithm and platform for specific applications, which in turn can enhance the efficiency and quality of the solutions produced.

II. LITERATURE REVIEW

A. Empirical Test

Knowledge acquired through the human senses is the definition of empiricism [7]. Based on that definition, empirical testing is a test method based on direct observation or practical experience through the five human senses. The goal is to collect evidence or data that can be used as a basis for drawing conclusions.

While empirical research is research whose conclusions are obtained through experiments or by collecting first-hand data [8]. Empirical testing is often used to test the hypothesis and the theory of a study by gathering empiric data through observations or experiments.

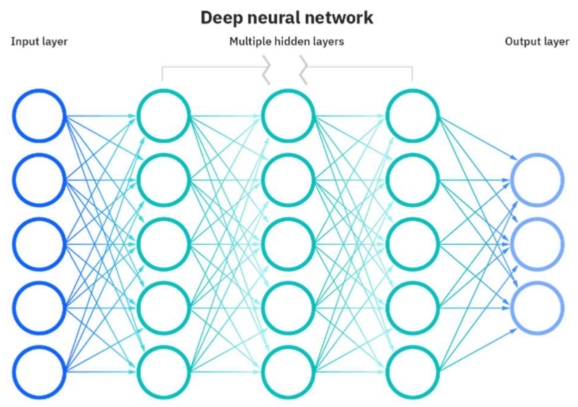
B. Classification

Classification is a way of grouping data according to the characteristics or characteristics of the data. Classification in machine learning is a learning process in which a model or algorithm uses the given data to predict the category or class of a new instance. The goal is to group objects or events into predefined categories based on the observed features or attributes. Widely used classification algorithms include [9]neural networks, decision trees, Bayesian networks, Support Vector Machine (SVM), and k-nearest neighbors [10].

C. Deep Learning

Deep learning is a subset of machine learning that focuses on artificial neural networks inspired by the structure and function of the human brain [11]. The technology allows computers to learn autonomously by analyzing data examples and identifying complex patterns, making it possible to perform tasks such as image recognition, natural language processing, and highly accurate predictions [5].

Deep learning is relatively more effective than traditional machine learning algorithms because of its optimized neural network architecture that can train models efficiently as well as perform hierarchical representation abstractions on multi-dimensional data [12]. Another advantage of neural network architecture is its ability to classify large classes of data with very high accuracy [13]. Despite the huge computational demands, the success of deep learning in solving difficult problems has made it the dominant approach in various artificial intelligence applications.



Artificial neural networks that implement deep learning architecture are called Deep Neural Networks (DNNs). DNN uses multiple layers with optimized algorithms and architectures [12]. Data is fed into this network and processed through a series of layers whose job is to extract important features from the data. Each layer changes the data representation in such a way that ultimately, the model can make predictions or identify complex patterns [14].

Each layer has interconnected neurons, and each connection has a weight that determines the importance of the information being passed. In the initial stage, these weights are set randomly. Then, the data is passed through the network, and the generated predictions are compared with the actual labels. The prediction errors, or errors, are then reduced through a process called backpropagation, where the model updates these weights using optimization algorithms such as Stochastic Gradient Descent (SGD) [5]. This process is repeated multiple times with different data to improve the accuracy of the model.

D. RapidMiner

RapidMiner is an open-source software platform for data mining that can be used as a stand-alone framework or integrated into other software as a data mining tool [15]. The software was designed by the company RapidMiner, which has been acquired by the company Altair. The platform provides an intuitive interface and powerful tools for investigating and analyzing datasets of varying scale and complexity. The deep learning operator in RapidMiner is an implementation of the H2O deep learning algorithm. The algorithm is based on a multilayered feedforward neural network trained with stochastic gradient descent (SGD) using the concept of backpropagation [16].

E. TensorFlow

TensorFlow is an open-source software platform developed by Google Brain Team for machine learning and artificial intelligence [17]. According to Abadi et. al., TensorFlow is the successor to the DistBelief framework which is Google's first artificial neural network trainer system [18]. TensorFlow was first introduced in 2015 and has since become one of the key tools in machine learning model development in areas including natural language processing, computer vision, speech recognition, and more. TensorFlow uses a unified dataflow graph to represent all computations and states in machine learning algorithms, including every operation, parameter and state update settings, and pre-processing inputs [18].

Users use APIs to build models that will later be converted into internal compute graphs on TensorFlow runtime [19]. TensorFlow uses Keras as a wrapper, which means Keras serves as a high-level interface (high-level API) to build and train models neural network on top of TensorFlow [20]. Since TensorFlow 2.0, Keras is fully integrated into TensorFlow and is now known as `tf.keras`.

F. Measurement Metrics

The measurement metrics of the model used in this experiment are accuracy, weighted mean precision, weighted mean recall and execution time. Accuracy measures how close the measured value is to the actual value [11]. Accuracy describes the percentage of correctness of a model in predicting its actual class. Precision measures the proportion of correct positive predictions versus the number of positive predictions. This shows how often the model is correct when predicting positive classes. Weighted mean precision measures the average precision for each class. Recall measures the proportion of positive cases that are correctly predicted as positive by the model, compared to the total actual positive cases. Recall is also known as sensitivity. Weighted mean recall measures the average recall for each class. Execution time is used to measure how fast a process in a program executes commands. This metric becomes a standard for determining the efficiency of a model in the training phase or making inferences or predictions.

G. Normality Test

The results of the experiment log must be checked for data distribution first. The Shapiro-Wilk test is a normality test method for determining whether a sample is normally distributed [21]. This test tests the null hypothesis where the sample $x_1, \ldots, x_n$ comes from a normally distributed population. The p-value is obtained by comparing W the statistics with the values available in the Shapiro-Wilk test table regarding *W the p-value* for a given sample size.

- H_0 the data sample has a normal distribution and H_1 the data sample is not normally distributed.
- If the SW test result > 0.05, then H_0 is accepted and H_1 is rejected. Data is not normally distributed.
- If the SW test result > 0.05, then H_0 is rejected and H_1 is accepted. Data is not normally distributed.

H. Statistical Test

Statistical tests were conducted on experimental results to test the hypothesis of this study. The essence of statistical testing is to compare what has been observed in the experiment with what would be expected if the null hypothesis was correct [22]. In this study, the statistical test used depends on the distribution of experimental data. If the experimental data is normally distributed, the statistical test used is paired T-Test. Conversely, if the experimental data is not normally distributed, the statistical test used is the Wilcoxson signed-rank test.

III. Methods

A research method is a series of systematic steps or approaches used by researchers to collect, analyze, and interpret data. Empirical testing methods involve the collection and analysis of real data to test hypotheses or theories. This research uses four stages, namely: Data Preparation, Pre-processing, Training & Validation, Evaluation. These stages can be seen at Figure 1.

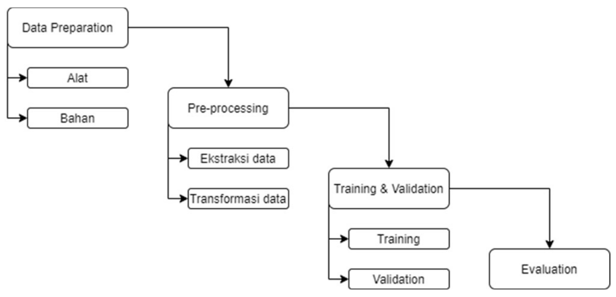


Figure 1. Research Methods used.

A. Data Preparation

This research uses dataset material collected from public dataset provider sites such as Kaggle (kaggle.com) and UCI Machine Learning (archive.ics.uci.edu) repositories. The criteria for datasets that are worthy of being used as test material from this study include the following:

- csv or xlsx dataset file format
- $100 < X < 9999$, where X is the number of instances of a dataset
- Number of numeric attributes > number of nominal attributes
- Usability values > 7 for datasets from Kaggle
- Missing values < 5 instances

B. Pre-processing

Pre-processing is the process of preparing data before it goes into a particular model or algorithm. The purpose of pre-processing is to ensure that the data is in a consistent, clean, and compliant form with the model or analysis to be performed. The pre-processing steps for each dataset in this study are divided into two, data extraction and data transformation.

Data extraction is the earliest step in pre-processing. Data from the dataset needs to be fed into the pipeline of the framework to be used. The extracted data must go through a transformation process in order to optimize the data that will be used to train the model and prevent errors in the subsequent process. The data transformation process includes feature selection, data duplication removal, data normalization, missing values replacement, data type change, and others.

C. Training & Validation

The training and validation phase is a stage where deep learning models get formed, trained, and validated. Pre-processed data goes inside the input pipeline to train the model. RapidMiner and TensorFlow have similar ways to train and validate models, but they differ in their execution methods. The configuration of TensorFlow parameters is adjusted as closely as possible to the configuration of RapidMiner. The hyperparameter comparison details of the two frameworks can be viewed on Table 1.

Table 1 Hyperparameter Comparison of Both Frameworks

Hyperparameters	RapidMiner	TensorFlow
Activation	Rectifier	ReLU (Rectified Linear Unit)
Hidden layer size	2 layers 50 neurons each	2 layers 50 neurons each
Epochs	10	10
Optimizer	ADADELTA	ADADELTA
Epsilon	1.0E-8	1.0E-8
Rho	0.99	0.99
L1 regularization	1.0E-5	1.0E-5
Trains samples per iteration	-2 (Auto-tune)	Online SGD

D. Evaluation

The results of the experiment in the form of accuracy, weighted mean recall, weighted mean precision, and execution time are recorded in Google Sheets and their averages are calculated. Average calculation aims to represent the performance of the framework for the entire dataset. The next step is to test the normality of the distribution using the Shapiro-Wilk test with a significant standard value or $\alpha=0.05$. If the normality test results show that the experiment log data has a normal distribution, then a parametric test is performed in the form of T or T-Test. On the contrary, if the experiment's log data does not have a normal distribution, then a Wilcoxson signed-rank test is used.

IV. Results and Discussion

A. Data Preparation

The datasets collected from the public repositories of Kaggle and UCI Machine learning amount to 50. The files from the datasets are downloaded and collected into folders and their metadata are recorded in Google Sheets. All the datasets used are stored on Google Drive by name. Details of the dataset characteristics based on the spread of instance number, number of attributes, and number of classes are visualized on the following histogram.

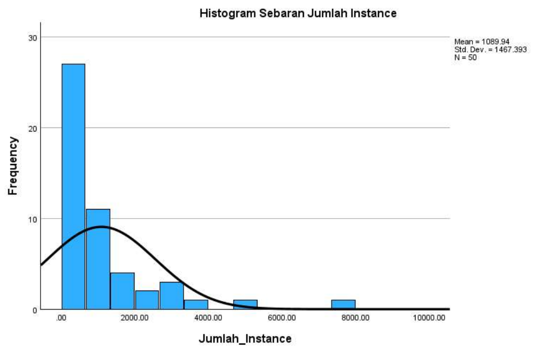


Figure 2. Histogram spread of the number of instances across the dataset.

From this histogram, it can be concluded that most instances are in a lower range of values, closer to the minimum value (104) than the maximum value (8000). Data distribution tends to be skewed to the right, with few datasets having a high number of instances. All this shows that the majority of the dataset has a relatively low number of instances.

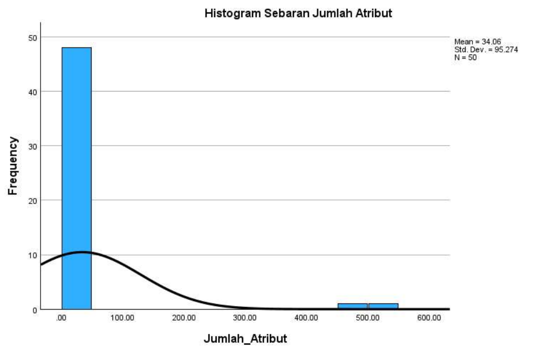


Figure 3. Histogram distribution of the number of attributes across the dataset.

The above histogram shows the spread of the number of attributes. From the available data, it can be seen that most datasets have a low number of attributes (close to zero), while only a few data have a high number of attributes (up to 532). This shows that most datasets have few attributes, with only some datasets having many attributes.

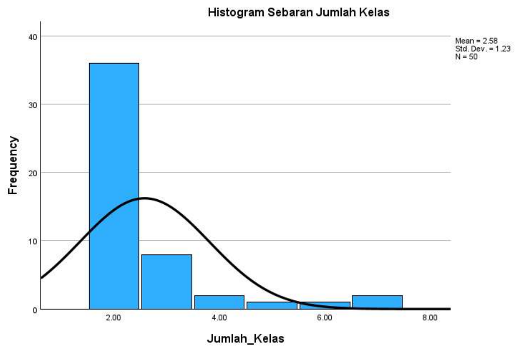


Figure 4. Histogram of class distribution across datasets.

Figure 4 is a histogram that depicts the distribution of class counts across the entire dataset. This histogram reveals that the majority of the data falls within the first class interval (0-2), indicating a left-skewed distribution as most of the data is concentrated towards the lower end of the scale and decreases in frequency as it moves towards higher class numbers. This suggests that the majority of the dataset tends to have a low number of classes.

B. Pre-processing

Datasets that have been stored on Google Drive will be extracted to each framework. All datasets must go through pre-processing before being fed into the deep learning model training pipeline.

- In RapidMiner all datasets are extracted by importing data. Data importing is done with a special interactive window from RapidMiner to manage how a dataset is extracted.
- Datasets that have been extracted and transformed will be stored in the local repository of RapidMiner.
- The dataset in TensorFlow is extracted using the help of a python library called pandas.
- TensorFlow utilizes pandas library and TensorFlow layer pre-processing to transform data.

C. Training & Validation

Deep learning models are trained from each dataset that has gone through the pre-processing stage. Both frameworks are configured to directly validate the model after training using stratified cross-validation. Data on measurement metrics that have been collected such as accuracy, weighted mean recall, weighted mean precision and execution time are also recorded in Google Sheets as experiment logs. This process is repeated as many times as the number of datasets used in the study.

D. Evaluation

Experimental logs that have been collected in the previous process are evaluated results such as rechecking experimental logs, calculating averages, normality tests and statistical tests.

Table 2. Average Metric Comparison of Both Frameworks

Metric	RapidMiner	TensorFlow
Accuracy	0.8682	0.9087
Weighted mean recall	0.8302	0.9087

Metric	RapidMiner	TensorFlow
Weighted mean precision	0.8451	0.9062
Execution Time	1.0164 seconds	110.9878 seconds

Based on the results obtained, TensorFlow excels in three model measurement metrics, which are accuracy, weighted mean recall and weighted mean precision with an average of 0.9 or 90% compared to RapidMiner which has an average of below 0.9 with a range of 0.83 to 0.86. While RapidMiner excels on the execution time metric with an average of only 1.01 seconds compared to TensorFlow which has an average of 110.98 seconds or 1 minute 50 seconds.

The normality test is performed by applying the Shapiro-Wilk test to all four measurement metrics to test the distribution of the data. Here are the results for the RapidMiner and TensorFlow frameworks.

TABLE 3. NORMALITY TEST RESULTS FOR THE RAPIDMINER FRAMEWORK

Metric	Statistics	Df	Sig.
Accuracy	0.859	50	< 0.001
Weighted mean recall	0.879	50	< 0.001
Weighted mean precision	0.837	50	< 0.001
Execution Time	0.678	50	< 0.001

TABLE 4. NORMALITY TEST RESULTS FOR TENSORFLOW FRAMEWORK

Metric	Statistics	Df	Sig.
Accuracy	0.844	50	< 0.001
Weighted mean recall	0.844	50	< 0.001
Weighted mean precision	0.828	50	< 0.001
Execution Time	0.674	50	< 0.001

The results of the Shapiro-Wilk test showed that the statistical value for all measurement metrics was less than the p-value used for this study. Therefore, H_0 is rejected and H_1 is accepted so that it can be assumed that accuracy, weighted mean recall, weighted mean precision, and execution time do not have a normal data distribution. Based on this, the statistical test used is the Wilcoxon Signed-Rank test.

Tests were also conducted using Statistical Package and Service Solutions (SPSS) software. The test scenario is defined as follows.

TABLE 5. SCENARIO OF STATISTICAL TEST

Scenario #	RapidMiner and TensorFlow Deep Learning Model Comparison
Scenario #1	Accuracy RapidMiner and TensorFlow
Scenario #2	Weighted mean recall RapidMiner and TensorFlow
Scenario #3	Weighted mean precision RapidMiner and TensorFlow
Scenario #4	Execution time RapidMiner and TensorFlow

After obtaining results from all four test scenarios, further analysis using the average metric comparison results was performed to determine the difference in performance in the two frameworks.

E. Discussion

The normality test results show that all measurement metrics do not have a normal data distribution, so the test Wilcoxon signed rank applied for statistical testing. Test results based on scenarios can be viewed at Table 6.

TABLE 6. WILCOXON SIGNED-RANK TEST RESULTS

Scenario #	Asymp. Sig. (2-tailed)	Conclusion
Scenario #1	< 0.001	There is a significant difference
Scenario #2	< 0.001	There is a significant difference
Scenario #3	< 0.001	There is a significant difference
Scenario #4	< 0.001	There is a significant difference

Based on statistical test results that can be seen on Table 6, all measurement metrics have statistical values less than α or 0.05 which means that the null hypothesis (H_0) is rejected and the alternative hypothesis (H_a) is accepted, so it can be assumed that there is a significant performance difference between RapidMiner and TensorFlow. This is in line with the average calculation results which show that there is a relatively large gap between the measurement metrics of the RapidMiner and TensorFlow models.

Deep learning models trained on the TensorFlow framework are significantly more accurate than models trained on RapidMiner. This is evidenced by statistical tests and the average of the three model measurement metrics (accuracy, weighted mean recall, and weighted mean precision) is superior to RapidMiner. This result is also reinforced by the Wilcoxon signed rank statistical test which shows that there are significant differences between the three metrics in each framework.

However, the RapidMiner framework can train deep learning models relatively fast compared to TensorFlow. A comparison of the average execution time between the two frameworks shows that RapidMiner can train and validate models with an average time of 1 second compared with TensorFlow, which takes an average of 1 minute and 49 seconds. We assume that this difference is due to the model training procedures used by RapidMiner that are different from those of TensorFlow as RapidMiner utilizes parallelism in training models [16].

V. CONCLUSIONS

The results of Wilcoxon signed-rank test show that the statistical values for accuracy, weighted mean recall, weighted mean precision, and execution time are all less than the specified significant value of 0.05. Therefore, the null hypothesis is rejected, and the alternative hypothesis is accepted, so it can be concluded that there is a significant difference between the performance of deep learning models trained by RapidMiner and TensorFlow.

REFERENCES

[1] M. M. Taye, "Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions," *Computers*, vol. 12, no. 5, May 2023, doi: 10.3390/computers12050091.

[2] Z. Ge, Z. Song, S. X. Ding, and B. Huang, “Data Mining and Analytics in the Process Industry: The Role of Machine Learning,” *IEEE Access*, vol. 5, pp. 20590–20616, Sep. 2017, doi: 10.1109/ACCESS.2017.2756872.

[3] M. I. Jordan and T. M. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science (1979)*, vol. 349, no. 6245, pp. 255–260, Jul. 2015, doi: 10.1126/science.aaa8415.

[4] K. Berggren *et al.*, “Roadmap on emerging hardware and technology for machine learning,” *Nanotechnology*, vol. 32, no. 1, p. 012002, Jan. 2021, doi: 10.1088/1361-6528/aba70f.

[5] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[6] K. Filus and J. Domańska, “Software vulnerabilities in TensorFlow-based deep learning applications,” *Comput Secur*, vol. 124, 2023, doi: 10.1016/j.cose.2022.102948.

[7] B. Bahar, “Model Pengujian Akurasi Berbasis Empiris Pada Algoritma A-Priori,” *Jutisi : Jurnal Ilmiah Teknik Informatika dan Sistem Informasi*, vol. 8, no. 2, pp. 45–56, 2019, doi: 10.35889/jutisi.v8i2.350.

[8] R. A. Alcívar Espín and J. D. Alcívar Espín, “THE EMPIRICAL RESEARCH: FOUNDATIONS FOR THE CORRECT APPLICATION,” in *EDULEARN19 Proceedings*, Jul. 2019, pp. 8784–8790. doi: 10.21125/edulearn.2019.2188.

[9] A. P. Wibawa, M. G. A. Purnama, M. F. Akbar, and F. A. Dwiyanto, “Metode-metode Klasifikasi,” in *Prosiding Seminar Ilmu Komputer dan Teknologi Informasi*, 2018, pp. 134–138.

[10] A. A. Soofi and A. Awan, “Classification Techniques in Machine Learning: Applications and Issues,” *Journal of Basic & Applied Sciences*, vol. 13, pp. 459–465, Jan. 2017, doi: 10.6000/1927-5129.2017.13.76.

[11] J. Arunadevi, S. Ramya, and M. R. Raja, “A study of classification algorithms using Rapidminer,” *International Journal of Pure and Applied Mathematics*, vol. 119, no. 19, pp. 15977–15988, Jun. 2018, [Online]. Available: https://www.researchgate.net/publication/325718529

[12] A. Shrestha and A. Mahmood, “Review of Deep Learning Algorithms and Architectures,” *IEEE Access*, vol. 7, pp. 53040–53065, 2019, doi: 10.1109/ACCESS.2019.2912200.

[13] U. Pujianto, D. Setyadi, and M. I. Akbar, “Prediction of stock purchase decisions using artificial neural network method,” *Applied Engineering and Technology*, vol. 1, no. 2, 2022, doi: 10.31763/aet.v2i1.686.

[14] S. Pouyanfar *et al.*, “A survey on deep learning: Algorithms, techniques, and applications,” *ACM Comput Surv*, vol. 51, no. 5, 2018, doi: 10.1145/3234150.

[15] L. Kovács and H. Ghous, “Efficiency comparison of Python and RapidMiner,” *Multidiszciplináris Tudományok*, vol. 10, no. 3, pp. 212–220, 2020, doi: 10.35925/j.multi.2020.3.26.

[16] A. Candel, E. Ledell, and A. Bartz, *Deep Learning with H2O*, vol. 6. 2024. [Online]. Available: http://h2o.ai/resources/

[17] B. Pang, E. Nijkamp, and Y. N. Wu, “Deep Learning With TensorFlow: A Review,” *Journal of Educational and Behavioral Statistics*, vol. 45, no. 2, pp. 227–248, Apr. 2020, doi: 10.3102/1076998619872761.

[18] M. Abadi *et al.*, “TensorFlow: A system for large-scale machine learning,” in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*, USENIX Association, 2016, pp. 265–283.

[19] H. Dai, X. Peng, X. Shi, L. He, Q. Xiong, and H. Jin, “Reveal training performance mystery between TensorFlow and PyTorch in the single GPU environment,” *Science China Information Sciences*, vol. 65, no. 1, 2022, doi: 10.1007/s11432-020-3182-1.

[20] B. T. Chicho and A. B. Sallow, “A Comprehensive Survey of Deep Learning Models Based on Keras Framework,” *Journal of Soft Computing and Data Mining*, vol. 2, no. 2, 2021, doi: 10.30880/jscdm.2021.02.02.005.

[21] P. Royston, “Approximating the Shapiro-Wilk W-test for non-normality,” *Stat Comput*, vol. 2, no. 3, 1992, doi: 10.1007/BF01891203.

[22] D. Curran-Everett, “Explorations in statistics: Hypothesis tests and P values,” *American Journal of Physiology - Advances in Physiology Education*, vol. 33, no. 2, 2009, doi: 10.1152/advan.90218.2008.