Fisheries Harvest Prediction using Genetic Algorithm Optimized of Gated Recurrent Unit

Adelwin Herman [a,1], Alvi Syahrini Utami [b,2], Annisa Darmawahyuni [b,3,*]
[a] Student of Informatics Engineering, Sriwijaya University, Palembang, Indonesia
[b] Lecture. Computer Science Faculty, Sriwijaya University, Palembang, Indonesia
[1] adeherman514@gmail.com*; [2] alvisyahrini@ilkom.unsri.ac.id; [3] riset.annisadarmawahyuni@gmail.com
* corresponding author

ARTICLE INFO

Article history
Received 19 March 2024
Revised 20 August 2024
Accepted 28 August 2024

Keywords
Neuroevolution
Econometric and Time Series Analysis
Evolutionary Machine Learning
Soft Computing
Metaheuristic Numerical Optimization

ABSTRACT

Indonesia is a maritime country with most of the population living near water areas. Water products are a common commodity often consumed cheaply, and food is therefore one of the primary human needs. Fishery harvest predictions are needed to control prices, prepare seeds, and ensure stable sales and consumption. The reason for choosing GRU for this prediction is that classical methods, commonly used in econometrics or time series analysis, were previously prevalent. GRU requires fewer operations than LSTM. Instead of training with an optimization algorithm relying on backpropagation and gradients, metaheuristic optimization in the form of a GA is used. GA does not require gradient information and is expected to avoid local optima. The total average MSE obtained is 9.55%.

1. Introduction

Indonesia is a maritime country with most of the population living near on water areas and aquatic products are a common commodity that is often consumed cheaply and therefore food is one of the primary human needs, so predictions of fisheries harvests are needed in order to control prices, prepare seeds, so that sales remain stable, consumption has no problem, etc. Examples that have predicted fisheries harvests include [1] for catfish with Support Vector Regression (SVR).

Gated Recurrent Unit (GRU) was chosen in this prediction because other methods such as Seasonal Autoregressive Integrated Moving Average (SARIMA) are often used, and GRU is similar to Long Short-Term Memory (LSTM) but requires fewer operations so it is fast and economical and can even outperform LSTM based on journal [2] and [3]. How to train GRU using a Genetic Algorithm (GA) (one of the metaheuristics) rather than gradient descent or the like because it does not involve objective function gradients, is popular (there are more than ten thousands of citations according to [4], can avoid local optima (Seiffert, 2001), robust [5], also considerations [6], and [7]; according to two twin studies [8] and [9], GRU whose weights are searched with fuzzy logic expert system controlled GA can produce a more generalized model.

Research on fishery harvest prediction using GRU which is optimized with a GA will be presented with the hope that several benefits can be obtained such as decision making (price control, just harvest predictions, cultivation management, or others).

2. Literature Study

a. Genetic Algorithm as Part of Evolutionary Optimization Algorithm
GA is (metaheuristic, population, and evolutionary based) one of evolutionary optimization algorithms by John Henry Holland (1975). The description of algorithm which can be made wider for almost many evolutionary algorithms (EA) is:

1. Define objectives and constraints, boundary/search area, allocation of chromosome (searched variable, vector of real number, integers, or discrete structure indices), allocation (number) of individuals, maximum epoch (generation), tolerance limit of evaluation values, mating and mutation properties (such as allocation (number) of mating and mutation probabilities for both individuals and chromosomes, etc), etc.;
2. Generate the first some individuals (solutions) with random numbers along the chromosome (the variable being searched) in the specified range.
3. Repeat until the epoch is complete or passes the tolerance limit of the evaluation values from the stages below:
   a. Evaluation and Survival:
      i. If necessary, perform individual decoding / postprocessing;
      ii. Evaluate all individuals by entering all parameters (individuals) into the constraint and objective function one by one, sort the results;
   b. Selection of individuals to be carried to the next generation;
   c. Crossover and mutation: With the probability that certain individuals and chromosomes are taken and supported by other properties, do crossover some of the previous individuals (only half of the pair of chromosomes are taken), so that their children become new individuals, then mutate some of the individuals (only half of the chromosomes), there is several options for how to carry out this mating and mutation.
   d. Population change.
4. Gated Recurrent Unit as Part of Deep Learning
   GRU is a RNN cell by [10] with this structure for a time in a layer.

   - Reset

$$z_t = sigmoid(w_z * [x_t, h_{t-1}] + b_z) \quad (1)$$

   - Forget

$$r_t = sigmoid(w_r * [x_t, h_{t-1}] + b_r) \quad (2)$$

   - Next hidden candidate

$$c_t = tanh(w_c * [x_t, r \odot h_{t-1}] + b_c) \quad (3)$$

   - Next hidden

$$h_t = ((1 - z_t) \odot h_{t-1}) + (z_t \odot c_t) \quad (4)$$

   How to use RNN is nearly the same as MLP; while MLP uses a generalized linear model (GLM) for a layer and has no time feature, RNN uses its own which evolved from GLM and has time feature. However, it's not enough to only rely on RNN, output must be processed in a GLM layer with no activation function using this formula.

$$\hat{y_t} = o_t = w_o * h_t + b_o \quad (5)$$

   Since it has a time feature, RNN has its own I/O (input and output) management by time; this is how RNN does forward propagation.

   - Many to one:
     - Layers (input, hidden, and output; hereinafter simply called hidden) are tensors of size (layer, time, batch, feature), at 0th time all hidden are initialized to 0, for all times (except 0th) at the first layer (input), although not hidden, is all initialized as input data that has been preprocessed. For all subsequent layers at the same time the same applies.
     - For all hidden layers with pointer l layers, for all l inputs with pointer i time:
       i. Insert inputs (l, i) and hidden (l, i-1) to the GRU gate.

ii. Does the input run out and there are no more layers after that? If so, a new hidden one has recently made its way to MLP; or, all these hidden layers will be input to the layer after (l + 1) and do it the same way.
iii. If training, calculate the error and change the weights; or post-processing.

- Many to many with an input mechanism that directly produces output (not used for this research):
  - Initialize hidden (at 0th time) with all 0;
  - Every single piece of input, after passing through all layers, will be passed to the MLP and that is the result;
  - The RNN stops when there is no more input.
- One to many:
  - Same as many to many above, however:
    i. There is only one input;
    ii. The results that come out become the next input.
- Many to many or Seq2Seq decoder encoder (in the case of this research, the vanilla/plain version is used without attention mechanism and beam search):
  - In the encoder stage, it is the same as many to one, but there is no need to output to the MLP;
  - At the decoder stage, the same as one to many, the difference is:
    i. Hidden is initialized with a hidden tail-end time encoder;
    ii. The first input is all 0.

As same as another ML based on regression, this objective function must be fulfilled.

$$\min_{\theta} J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda_1 \sum_{j=1}^{p}|\theta_j| + \lambda_2 \sum_{j=1}^{p}\theta_j^2 \tag{6}$$

The first order is MSE to make sure that the nonlinear multiple equation system is right; the second is L1 regularization, the third is L2 regularization, these make sure to prevent overfit or underfit. How to read that equation is "search thetas (weights and biases; as multivariable) that can minimize function J as low as possible".

5. Neuroevolution as Part of Soft Computing and Evolutionary ML
   Just using traditional ML methodology sometimes can be not enough, that's why using metaheuristic can help traditional ML to get the better model. Evolutionary ML and Soft Computing try to mix two or more different methodologies at same time such as neuroevolution in training and building architecture of artificial neural network (ANN) or ML based on regression. The need for BP comes from what the optimizer wants, if the optimizer needs gradient of J while J is a composite function, then chain rule is needed for practicality and gradient of J of a weight or bias can be reusable (this is BP means). In training cases, since common metaheuristics never rely on gradients of objective function, BP is not needed; Stephen [11] at his paper write that there is correlation between using BP/GD (deterministic) and evolutionary algorithm/computation (EA/EC) (metaheuristic); using non-BP for ANN or ML based on regression training is not new but rare due to gradient is not a problem for training according to. By using GA for training GRU, expected to not be trapped in local optima (minima).

3. Methodology

The research boundaries are:

- GA is used to find weights and biases of GRU;
- The data obtained is only in the period 2019 - 2022 which has been pre-processed, but the 2022 data has not been completed. If the data visualization is analyzed, there does not appear to be a special pattern, especially seasonality, in all species, so the calculation results may not be very hopeful;
- Involves custom instance libraries, which play a major and important role:
  - PyMOO (metaheuristic) to find weights and biases;

    - Keras (deep learning) with the optimizer replaced by PyMOO.
- The program is made limited to research, even only univariate;
- Using iterative methods in software creation;

4. Result and Discussion

This below configuration which (depicted as image) was obtained by random search (as first research from overall three research) was used for the main research (this is the third research from overall three researches) which the specification is:

- Configuration choosen:
    - No. 18;
    - 4 data inputs and 4 data outputs;
    - 0.0001 L1 and 0.001 L2 regularization constant;
    - Mutation hyperparameters: 0.3 prob_var, 0.3 prob, 0.7 eta, do at_least_once;
    - Crossover hyperparameters: 0.5 prob_var, 0.3 prob_exch, 0.9 prob_bin, 0.5 prob, 0.3 eta;
    - MSE of train and test (split 70:30) for searching hyperparameters are 12% and 1.1%.
- Giant snakehead data is used for overall first and second research; each configuration (total 20 configurations) took 50 epochs (100 - 400 seconds) with 50 individuals.
    - On main research, it's modified into 60 epochs and 50 individuals due to second research (please refer to the post and pre main research analysis part).
- 2 time-series validation on second and main research.
- For all research, preventing negative values when predicting, using ReLU (alternatively sigmoid) activation function on the output layer can help (remember that there is no backpropagation, no further effect aside this).

Table 1. Result of The Main Research

Species	Val. No.	% After Training MSE			Species	Val. No.	% After Training MSE		
		Train	Validation	Test			Train	Validation	Test
Giant snakehead	1	12.5	17.15	25.8	**Carp**	1	3	2.5	6.4
	2	15.9	3.19	3.88		2	3.4	1.2	7.1
Tambaqui	1	6.1	9.9	4.4	**King prawn**	1	9.2	14.1	6.5
	2	8.4	15.9	8.4		2	7.89	10.3	11.2
H. nemurus	1	8.1	9.3	19.3	**Nile tilapia**	1	2.9	6.8	18.3
	2	9.7	22.7	28.05		2	4.48	17.69	17.48
Tiger prawn	1	0.4	0.7	39.8	**Shark catfish**	1	4	5.1	11.15
	2	2	10	25.89		2	3.2	2.5	3.8
Giant gourami	1	3.5	7.9	5.3	**Walking catfish**	1	11.3	15.3	14.2
	2	2.2	4.2	4.8		2	7.8	4	5.1

This is the post training overall MSE.

Table 2. Overall MSE

At Validation	% Post Training MSE				
	Means			The Most	
	Train	Validation	Test	Lowest	Highest
1	6.1	8.875	15.115	0.4	39.8
2	6.497	9.168	11.57	1.2	28.05
Overall	**6.289**	**9.0215**	**13.3425**		
	9.55				

Post and pre main research training analysis:

- Sometimes, test and validation errors are not adjacent in some epochs.
  - Contrary to previous (second) research with modified configuration with 100 individuals and 100 epochs (around 2000 seconds), both errors are nearly the same. At that configuration, first validation has high error and rarely found the best optimum which has low validation or test error, and starts to rarely find the best optimum after the 50th epoch at second validation.
- After training, the average species had low errors; but for the first validation it was higher due to insufficient data. In the case of tiger prawns, an overfit occurred for both validation because the nature of the data suddenly had a pattern at the latest time, GA-GRU did not expect something like this (natural), it could be due to truncation of training and test data where the test data got the latest time data.
- The ups and downs of training progress can be considered normal because:
  - GA is a global search and many solutions can change quickly with optima shifts, so the application of history is necessary to capture the best optima. Compare this with gradient derivatives or similar which are only a single solution.
  - L1 and L2 regularization forcing optima shifts.
- These are the plots of the research on the next pages. Legend:
  - Training progress on the first (a) and second (c) validation.
    - x axes is epochs, y axes is MSE.
    - Blue line means training error, green for test error, and orange for validation error.
  - Predictions in the first (b) and second (d) validation.
    - x axes is time, y axes is quantity.
    - Blue line means the real training data, orange for machine output on training data, green for prediction (the mechanism is using autoregressive, concatenating the real data with m data outputs, initialized with the n last piece of data as inputs).
  - The species name in the images are written in biological-latin name.

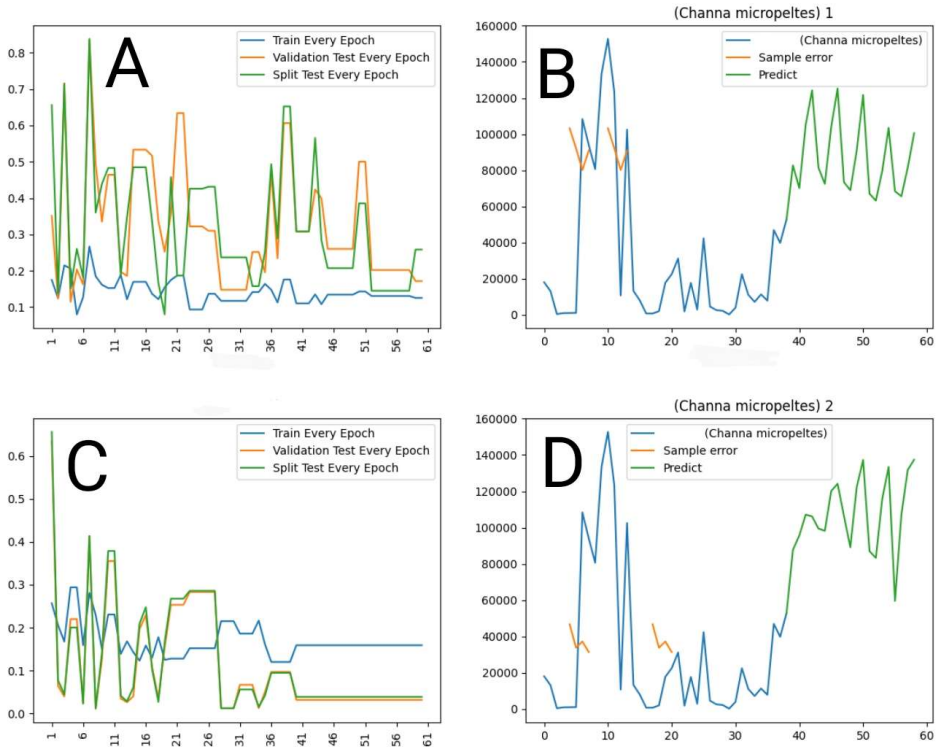

Fig. 1. Giant Snakehead

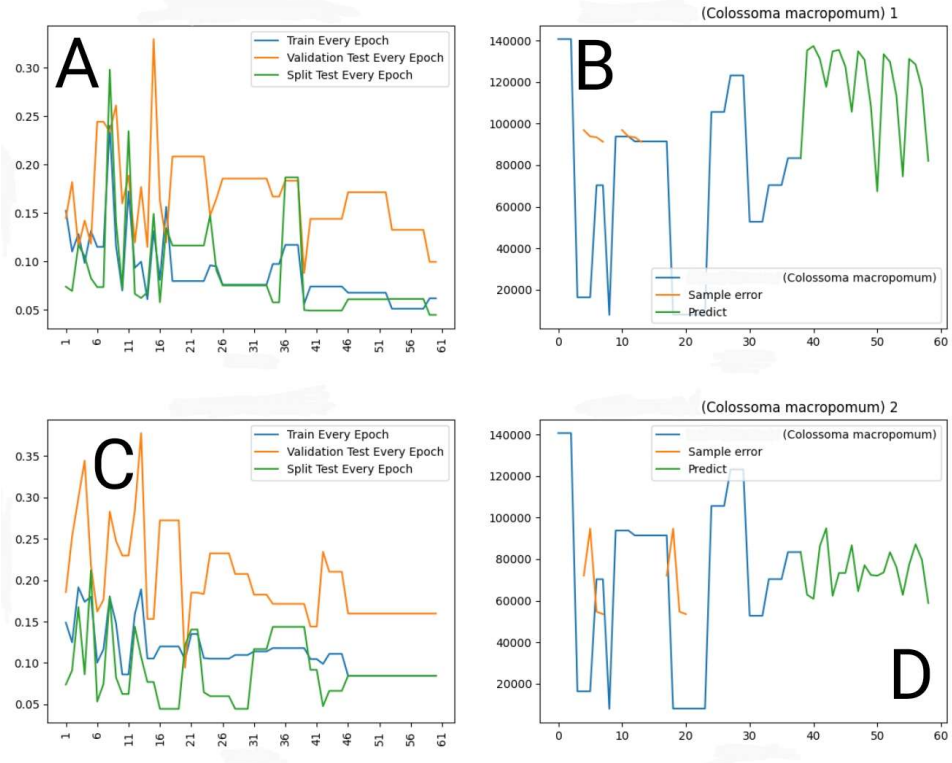

Fig. 2. Tambaqui

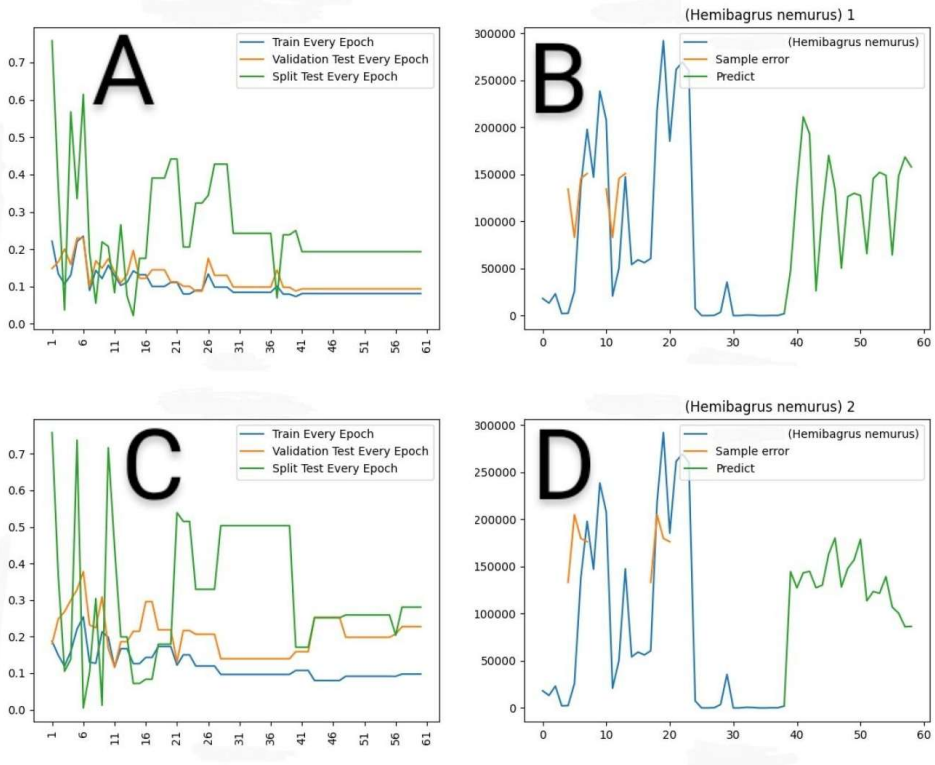

Fig. 3. *Hemibragus nemurus*

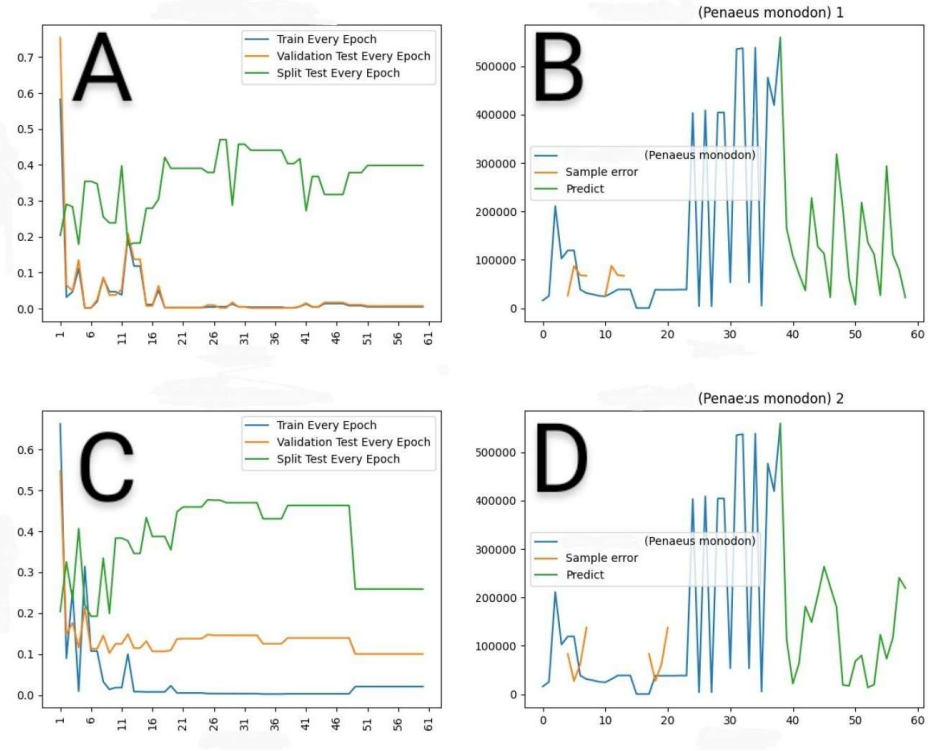

Fig. 4. Tiger Prawn

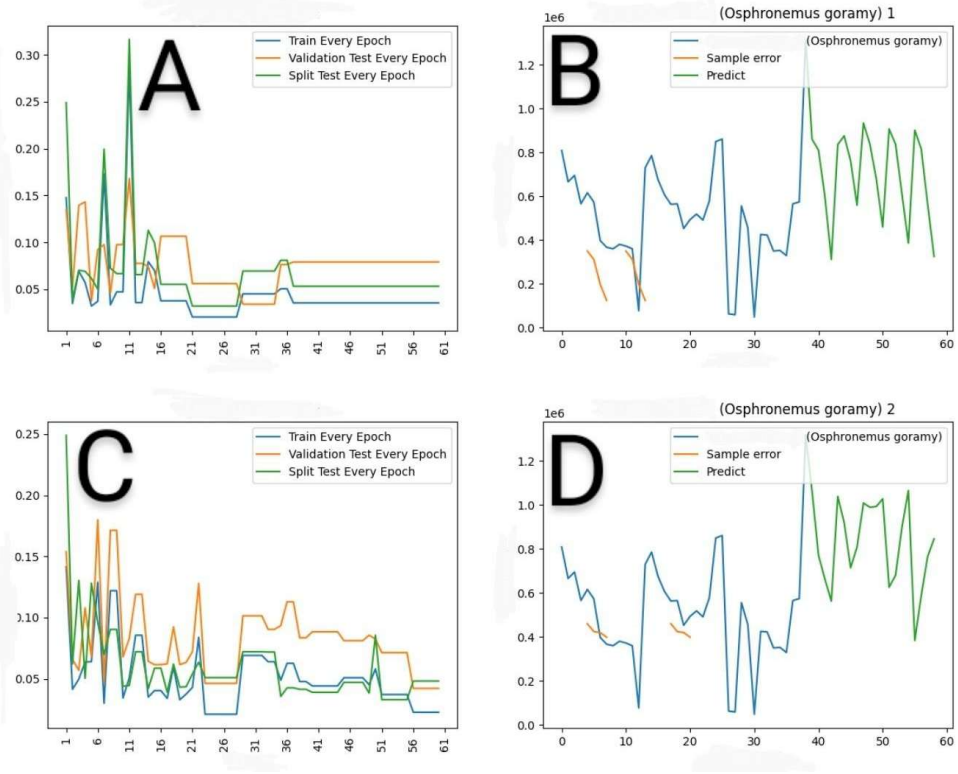

Fig. 5. Giant Gourami

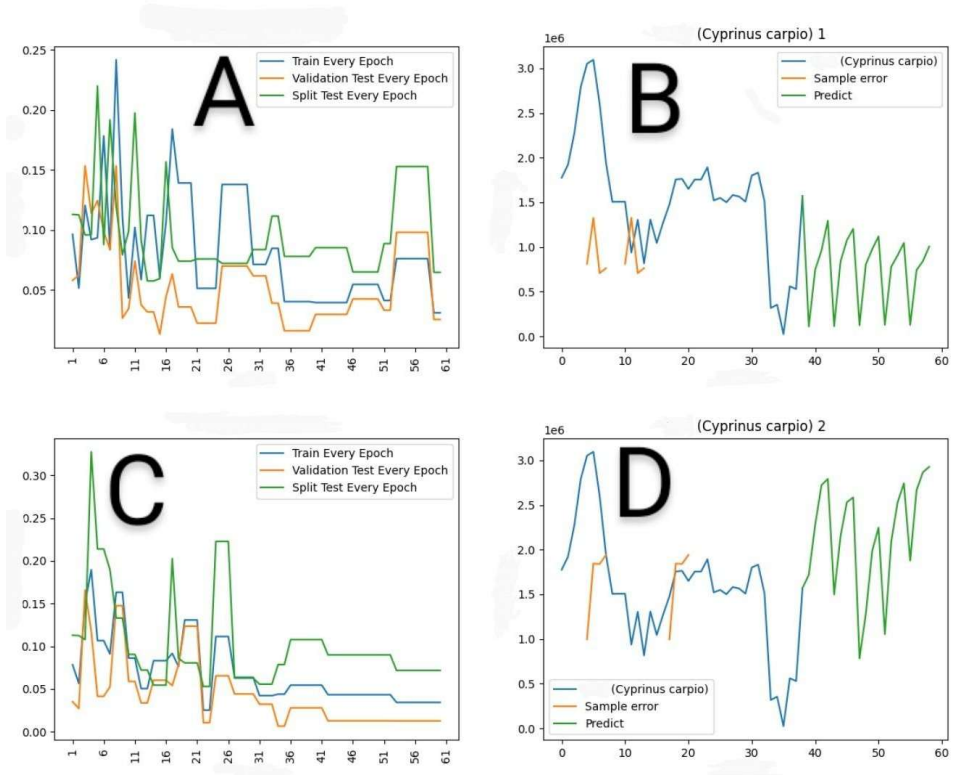

Fig. 6. Carp

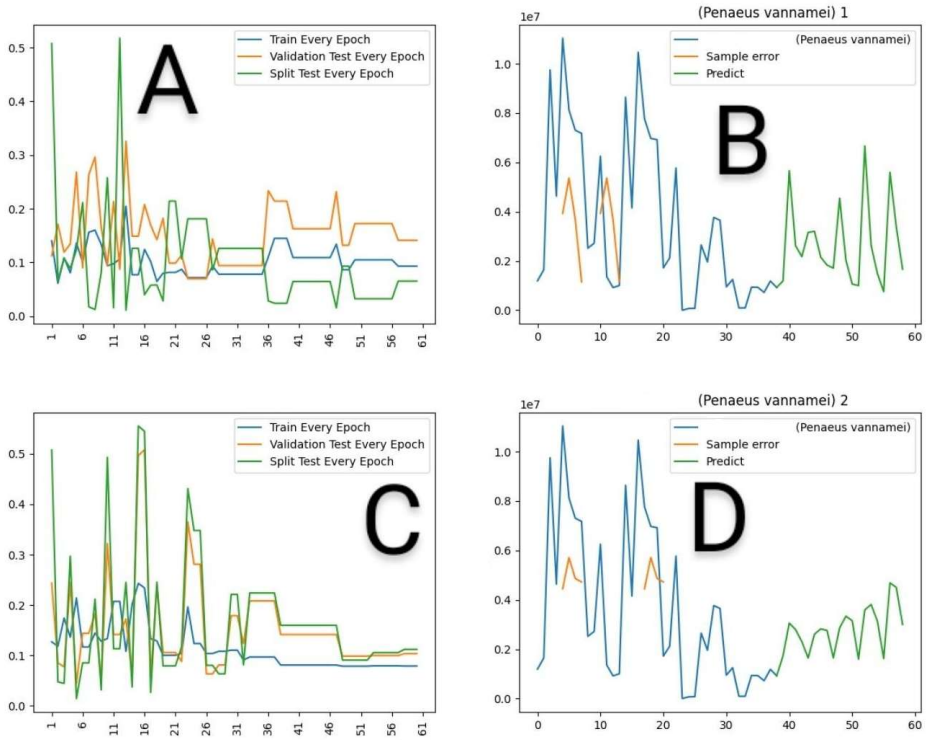

Fig. 7. King Prawn

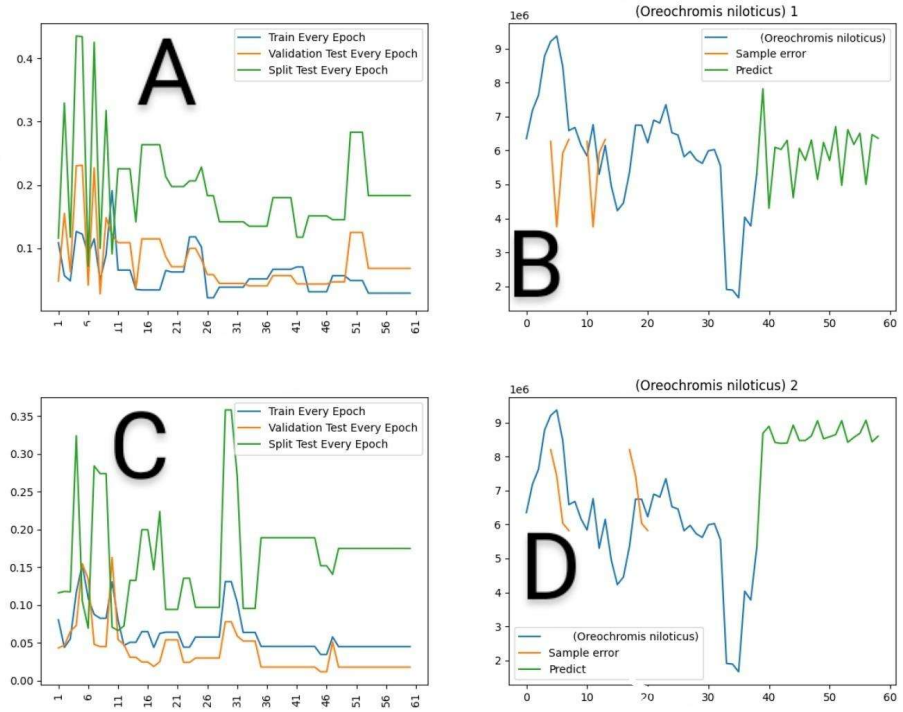

Fig. 8. Nile Tilapia

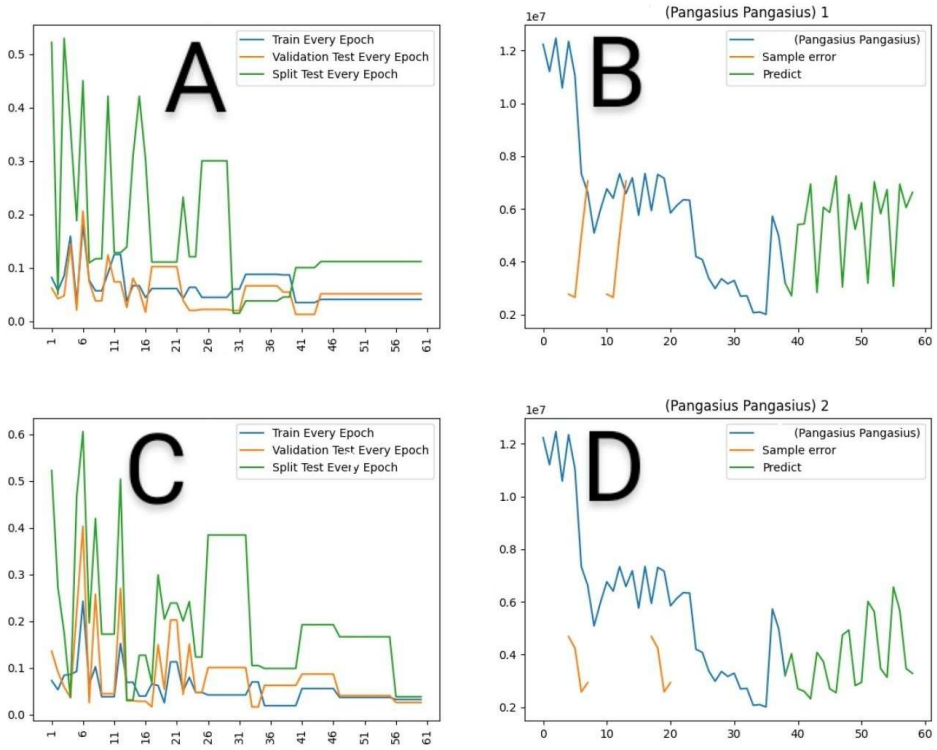

Fig. 9. Shark Catfish

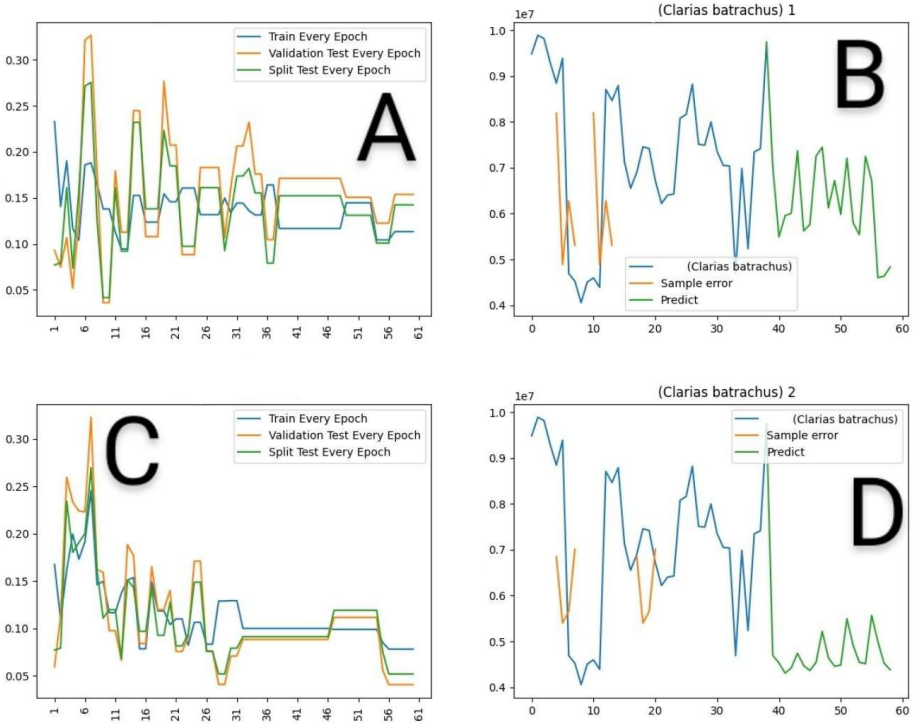

Fig. 10. Walking Catfish

5. Conclusion

Even though it is successful in making predictions, due to the lack of research resources, further research may be needed regarding the efficacy of using metaheuristics as a replacement for the optimizer (deterministic) built in to DL libraries, such as benchmarking between several types of optimization; or there is a need for certain additions in terms of time series. The results of training (it can be seen from table 2 that the average total error is low at 9.55%, and also from previous ten images) appear to be assessed skeptically and may even be suspected of being overfit and/or underfit (also shown in two data samples from each all ten models have same result). If someone basically wants to replace the optimizer from deterministic to probabilistic/metaheuristic, it is highly recommended to implement a history system and not implement early stopping; with a history system, someone can take models at certain epochs that they feel are the best. It is very necessary to continue research both for the same and different cases, so that from various perspectives, a better common ground (conclusion) can be drawn, considering that neuroevolution in training problems is rarely researched.

References

[1] C. M. Suprapto, "Prediksi Hasil Panen Budidaya Ikan Lele Dari Mitra Panen Menggunakan Algoritma Support Vector Regression (Studi Kasus: PT. Adma Digital Solusi)," UPN Veteran Jawa Timur, 2023.

[2] J. Noh, H. J. Park, J. S. Kim, and S. J. Hwang, "Gated recurrent unit with genetic algorithm for product demand forecasting in supply chain management," *Mathematics*, vol. 8, no. 4, 2020, doi: 10.3390/math8040565.

[3] U. I. Arfianti, D. C. R. Novitasari, N. Widodo, M. Hafiyusholeh, and W. D. Utami, "Sunspot Number Prediction Using Gated Recurrent Unit (GRU) Algorithm," *IJCCS (Indonesian J. Comput. Cybern. Syst.*, vol. 15, no. 2, 2021, doi: 10.22146/ijccs.63676.

[4] A. Sunyoto, A. Arifianto, R. Rismala, and others, "Evolutionary Machine Learning: Pembelajaran Mesin Otonom Berbasis Komputasi Evolusioner," 2023.

[5] R. Mahajan and G. Kaur, "Neural Networks using Genetic Algorithms," *Int. J. Comput. Appl.*, vol. 77, no. 14, 2013, doi: 10.5120/13549-1153.

[6] P. Tao, Z. Sun, and Z. Sun, "An Improved Intrusion Detection Algorithm Based on GA and SVM," *IEEE Access*, vol. 6, 2018, doi: 10.1109/ACCESS.2018.2810198.

[7] V. Jayaraj and S. Raman, "A Genetic Algorithm Optimized Multilayer Perceptron for Software Defecr Prediction," *Int. J. Adv. Technol. Eng. Sci.*, vol. 4, no. 2, pp. 132–141, 2016.

[8] C. Bai, "AGA-GRU: An Optimized GRU Neural Network Model Based on Adaptive Genetic Algorithm," in *Journal of Physics: Conference Series*, 2020, vol. 1651, no. 1. doi: 10.1088/1742-6596/1651/1/012146.

[9] C. Bai, "AGA-LSTM: An Optimized LSTM Neural Network Model Based on Adaptive Genetic Algorithm," in *Journal of Physics: Conference Series*, 2020, vol. 1570, no. 1. doi: 10.1088/1742-6596/1570/1/012011.

[10] K. Cho *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2014. doi: 10.3115/v1/d14-1179.

[11] S. Whitelam, V. Selin, S. W. Park, and I. Tamblyn, "Correspondence between neuroevolution and gradient descent," *Nat. Commun.*, vol. 12, no. 1, 2021, doi: 10.1038/s41467-021-26568-2.