Exploring the Application of Replay-Based Continuous Learning in a Machine Learning Pipeline

Fabian Rensing*, Lucy Ellen Lwakatare and Jukka K. Nurminen

Department of Computer Science, University of Helsinki

Abstract

Replay-based continuous learning (CL) methods are utilized to adapt deployed deep learning (DL) models to evolving data while minimizing catastrophic forgetting. However, the primary challenge is that most CL methods in the literature do not focus on real operational data and regression tasks. These key aspects are explored in this study, which applies and evaluates replay-based CL method in a machine learning pipeline that uses real operational data to retrain a DL model for a maritime use case. This pipeline was adapted to the application context based on experiments and discussions with domain experts, by enhancing the sample management. Our results show that even with a small set of replayed samples, the model performance can be improved and catastrophic forgetting can be limited.

Keywords

Continuous learning, machine learning pipeline, replay-based methods

1. Introduction

With the growing integration of technology in the maritime industry, data-driven approaches utilizing operational and environmental data from sensors onboard have become prevalent for machine learning (ML) -based performance monitoring [1]. In addition to the modelling challenge of accurately predicting ship fuel oil consumption (FOC), trained ML models deployed in real-world settings encounter data drifts due to factors such as biofouling [1]. To counteract decay in model accuracy caused by data drifts, the ML models are continuously retrained with the latest available data. A continuous learning (CL) strategy is recommended for upgrading deployed ML models, particularly when constrained by computational resources and data availability [2].

CL characterizes learning from dynamic data distributions and describes the ability of ML models to continuously learn from newly acquired data while retaining previously learned knowledge [3]. CL addresses the challenge of *catastrophic forgetting* where a model trained on a new dataset with a different distribution shows a reduced ability to retain previously learned knowledge [3]. Deep artificial neural network (ANN) models suffer from catastrophic forgetting and observe a drop in accuracy when previously learned representations are overwritten during parameter update after few iterations of training with the new dynamic data [4]. Generally, retraining the model with all data can address catastrophic forgetting, however, this practice is considered inefficient as it introduces significant storage and computation overheads among others [3].

Several CL methods have been proposed to address catastrophic forgetting, broadly categorized as parameter isolation/architecture-based, regularization-based, and replay-based methods [3, 4]. The replay-based CL approach explored in this study involves storing subsets of historical data (i.e., exemplars) and combining them with the latest data to retrain a model. A major issue with most studies on CL methods is their lack of focus on real-world operational data [5] and regression tasks [6]. This study addresses these research gaps. We chose a replay-based CL method because it allows the ML pipeline to be set up agnostically to the model architecture, enabling future extensibility and adaptability. We address these research questions (RQ):

- RQ1: How does the performance of a *replay-based CL* model compare to a model retrained using *all_data* and just the *current_data*?
- RQ2: How well does the replay-based CL model retain previous knowledge and generalize?
- RQ3: How to determine the optimal exemplar set?

The main contributions are twofold. First, we implement a replay-based CL method applied to a pre-trained ANN for FOC prediction using real-world ship sensor data. Additionally, a comparative analysis is performed alongside two baseline models, one trained with *all_data* and the other with *current_data*. Second, the paper proposes an approach to selecting an optimal exemplar set in the replay-based CL method, taking into consideration the requirements of the target application.

2. Continous Learning (CL)

CL aims to extract knowledge from an (infinite) stream of data to gradually extend the existing knowledge of the model without catastrophic forgetting [7]. CL literature introduces methods that try to balance the trade-offs between learning plasticity and memory stability [3]. A model with high stability is less flexible to changes during training, resulting in better retention of previous knowledge but less adaptation to new data. Conversely, a model with a higher plasticity can better adapt and integrate new knowledge but may lose previously learned knowledge. Thus, CL methods are evaluated from three aspects [3]: model's incremental quality (*Overall Performance*), model's ability not to forget (*Memory Stability*) and model's ability to learn from a dynamic data distribution (*Learning Plasticity*).

CL methods are extensively described in the literature (we refer the reader to [3]). One straightforward, practical approach is to store a few samples from the old data distributions and use them in subsequent training, as employed in replay-based CL methods [4]. Regularization-based CL methods do not consider historical data but rather freeze a copy of the old model to regularize parameter changes [3]. Parameter isolation or architecture-based CL methods assign different model parameters for different tasks [3].

Published in the Proceedings of the Workshops of the EDBT/ICDT 2025 Joint Conference (March 25-28, 2025), Barcelona, Spain

*Corresponding author.

fabian.rensing@alumni.helsinki.fi (F. Rensing);
lucy.lwakatare@helsinki.fi (L. E. Lwakatare);
jukka.k.nurminen@helsinki.fi (J. K. Nurminen)

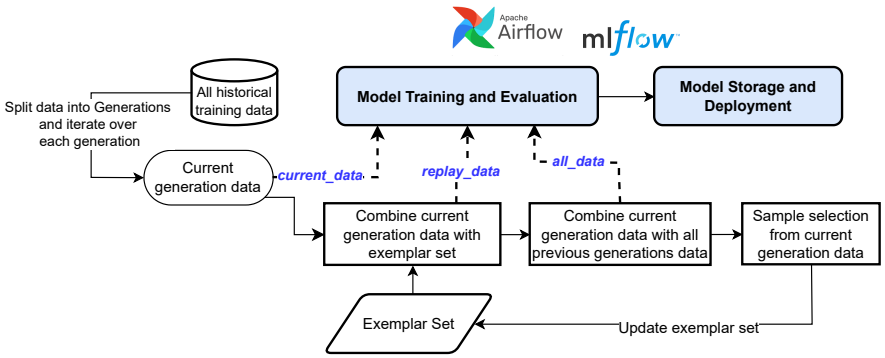

Figure 1: Overview of our experiment ML training pipeline

Replay-based CL methods select and store a small subset of past data (i.e., exemplars), in limited storage space to be replayed during training sessions along with new data [3, 4]. Alternatively to this reharsal approach, the exemplar set is used as constraint generators to characterize valid gradients [4]. The exemplar set size is fixed for all training sessions. After each session, new exemplars from the latest dataset replace an equal number of the oldest exemplars [7]. Given the fixed memory size of the exemplar set, selecting and discarding exemplars is crucial [4, 3]. Exemplars can be chosen and removed randomly, or through advanced methods that evaluate and discard the least valuable ones to optimize space [8, 4].

3. Experiment

The ML training pipeline depicted in Figure 1 illustrates our experimental setup for testing the replay-based CL method. The implementation code is available on GitHub[1].

3.1. Training Data and Pipeline

The training data consisted of historical sensor time series data collected from an anonymized container ship over two years. The historical experiment dataset includes over 175,000 rows, with data points spaced 5 minutes apart. The dataset's columns are detailed in Table 1. Utilizing a timestamp column, the experiment dataset was divided into five equal parts, each serving as a distinct training dataset for the pre-trained ANN model. These divisions are henceforth referred to as training generations. Data distribution plots for each training generations are in the GitHub repository.

In the replay-based approach, samples from previous datasets are kept in an exemplar set and *replayed* during model training alongside new data. The experiments are set up to first train an ANN model with the data of generation zero. In our study, the architecture of an optimal ANN model for predicting ship FOC, as discussed in [9], was adopted and utilized. Post generation zero, before retraining begins, the new generation data is mixed with the exemplar set. After retraining, new exemplars are added from the recent data to the set, with an equivalent number of old exemplars removed.

The training pipeline manages data generations, handles training data and exemplar sets, and follows a conventional training process, with each dataset split into training and validation sets using an 80% split ratio. The models are trained from scratch, with hyperparameters, such as the number of hidden layers, remaining unchanged. The number of training epochs is fixed at 30 for all models, except for the generation zero model, which undergoes training for 80 epochs. The trained models were evaluated based on their calculated mean squared error (MSE) when predicting on the validation sets for each generation. The architecture and hyperparameter configurations of the pre-trained ANN model are presented in Table 1.

[1]https://github.com/UH-MLOps/replay-cl-in-ml-pipelines

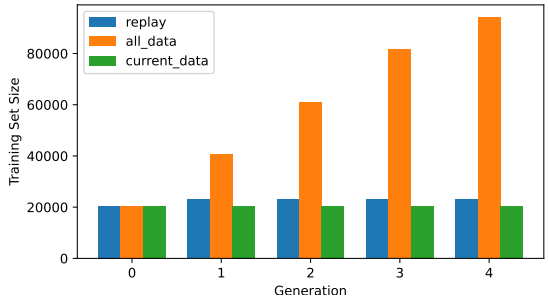

Figure 2: Training set sizes per model and generation.

Table 1
Summary of experiment data and model architecture

Dataset (anonymized container ship)
• *Input Features*: speed over ground; course over ground; draft; wind (speed, direction); wind wave (height, direction, period), swell (height, direction, period); water depth; current (speed, direction) • *Output:* Fuel mass flow in Kg/s (min:0, max:2.691, std:0.68)
Model (feedforward NN) architecture
• Hidden layers: 10 with 10 neurons in each layer • Activation function: Leaky ReLU; Optimizer: ADAM

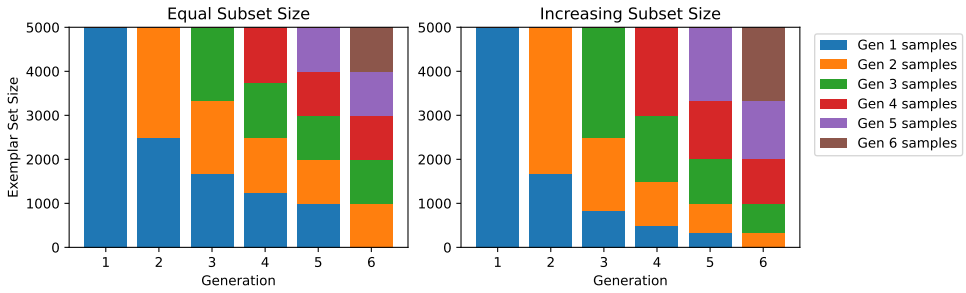

Figure 3: Subset sizes in the exemplar set for the two different operation modes. 6 generations of data, exemplar set size $M = 5000$, and a limit $n = 5$ generations that are stored.

3.2. Lower and Upper Boundary Models

The main difference between all models is the amount of training data used during the retraining runs. The upper boundary model, named *all_data*, is trained on the current generation data and all historical datasets of previously seen generations combined. The *all_data* model is expected to perform best since it is trained on all available datasets.

The lower boundary model, named *current_data*, is trained only with the new dataset available in the current generation. This model acts as the lower bound and is expected to perform poorly compared to the others, demonstrating the effect of catastrophic forgetting because training data is limited to the current generation. For this model, the training pipeline is set up with an exemplar set size of zero such that no historical samples are collected. Figure 2 shows the sizes of the training sets that are used for each model generation. The training set always consists of the current generation training data plus the samples stored in the exemplar set. The different sizes of the datasets are expected to influence the runtime of each training and the resulting model's performance. For reproducibility, the pipeline ensures that all libraries that use randomness are seeded before the experiment runs.

3.3. Replay-based CL Approach

In each generation, t, the model is trained based on a new dataset D_t combined with the exemplar set $E_{1:t-1} = \{E_1, E_2, \ldots, E_{t-1}\}$. The exemplar set comprises selected samples from the previous training datasets. The training objective for generation t is minimize the total loss L_t:

$$L_t = \sum_{\{x,y\} \in D_t \cup E_{1:t-1}} L(x, y, \theta_t)$$

L is the loss function that calculates the model error given input data x, the expected output y, and the model parameters θ_t. This error is calculated for every input in the combined dataset and summed to give the total loss L_t. Once the model is trained, the current dataset D_t gets sampled for exemplars, which are added to the exemplar set. The training pipeline automatically evaluates the trained model before the next training generation starts, where the same model is retrained on the next dataset.

3.3.1. Exemplar Set and Selection

The exemplar set is used for storing and replaying samples collected from previous datasets. An important property is that the exemplar set always stores a fixed amount of samples, meaning old ones are dropped from the set when new samples are selected. An exemplar set $E_{1:t-1}$ at generation t consists of subsets that contain samples from previous generations data $E_{1:t-1} = \{E_1, E_2, \ldots, E_{t-1}\}$, where each subset E_i contains samples from the i-th generation dataset. These subsets' combined size equals the total size M set for the exemplar set.

$$M = \sum_{i=0}^{t-1} |E_i|$$

The choice of M is critical when setting up the pipeline. To reduce the amount of data and computation during the training process, M should be as small as possible while still fulfilling the goal of mitigating catastrophic forgetting [7]. Previous literature has no established approaches for determining a good value for M. Researchers [7] have mentioned and used a size of 1% of the whole training dataset. This study uses an exemplar set size of 2% (4078 rows) of the entire generation dataset, as further explained below.

In literature, the exemplar set is often implemented such that each subset has the same number of samples: $|E_i| = \frac{M}{t}$. To always maintain the total size M of the exemplar set, the subsets' size must be updated after each new sampling. This means subsets already in the exemplar are reduced to free up space for adding new exemplars from the current dataset. The downside of this approach is that the number of new samples introduced gets smaller with every generation, limiting the influence of each sample set during future training sessions.

For this study, domain experts suggested that in the context of ship performance prediction current historical data is more relevant than older data when training a model. Given this knowledge, the exemplar set was extended with two new functionalities. First, a limit n on the number of subsets kept in the exemplar set was implemented. This way, the number of subsets can grow up to n stored subsets before the oldest subset gets dropped when new data is sampled, keeping the number of stored sets constant. This also ensures static subset sizes that will stay constant over multiple generations when the limit of n is reached. Secondly, a different operation mode for the exemplar set was implemented. In this new mode, the subsets are not kept at equal sizes, and newer data is prioritized by storing more samples from recent datasets. The following formula was implemented to calculate the subset sizes at generation t when the exemplar set contains $t - 1$ subsets:

$$|E_i| = M * \frac{2i}{t(t-1)},\ i \in [1,\ t-1]$$

For example, in generation $t = 4$ the calculated subset sizes are $|E_{1:t-1}| = |\{\frac{1}{6}M, \frac{2}{6}M, \frac{3}{6}M\}| = M$.

Figure 3 shows these functionalities in our experiment setup. The setup has six generations of data, the total exemplar set size is $M = 5000$, and the limit is set to $n = 5$ generations that are kept in the exemplar set. After generation 5 the oldest subset for generation 1 is removed because of the set limit.

4. Results

This section presents experiment results collected with the training pipeline. In all belwo evaluations and plots, the base model represents generation zero, and the retrained models represent generations one to four. Thus, generation zero shows the performance of the shared base model, while generations one to four show the assessments of the retrained models using the different training setups.

4.1. Replay-based CL Model Performance (RQ1)

RQ1 evaluates how the *replay* model compares to the other two baseline models trained either on *all_data* and *current_data*. Figure 4 shows the performance (MSE) of the three models against the current generation validation set.

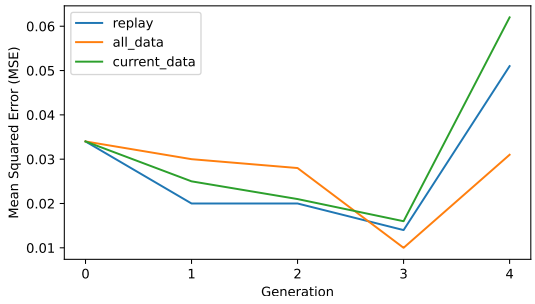

Figure 4: Model performance per generation evaluated on the generations validation set.

The *replay* model has the lowest MSE in generations one and two, but it gets overtaken by the *all_data* model in generations three and four. The plot shows the best performance for all three models in generation three and the poorest in generation four. The poor performance is likely due to the changes in data patterns/distributions in the training and validation datasets. In our GitHub[2] repo we provide distribution plots for the three most influential sensor measurements, that show a high overlap between the distributions of training and testing data in generation three, and a different distributions between training and test data in generation four. In summary, the replay-based CL approach has a slightly better but comparable performance to the *current_data* approach. The *all_data* approach is worse in the first generations compared to the *replay* and *current_data* approach but then improves in the third and fourth generations, beating the other two approaches.

When considering a production setup, the runtime of a training run is significant to evaluate since a long training process binds computing resources, which could accumulate high costs of operation when the training pipeline is run repeatedly over a long period, performing many lengthy retrainings. A model's training runtime depends on the model's architecture, i.e., the number of neurons in the model and the amount of training data used.

Figure 5 shows the average training runtime for each training approach. The experiments were conducted on an average laptop with no specialized hardware, e.g. GPU, which could speed up the training process. The training runtimes for the *all_data* model are increasing in every generation, which is explained by the increasing training dataset in every generation (see Figure 2). While this is an unfavourable behaviour regarding computing time, the results in the section 4.2 show that the *all_data* model performs the best for mitigating catastrophic forgetting. This shows an important trade-off between training runtime and model performance.

The training runtimes for the *replay* and *current_data* models are constant in each retraining since their training dataset are also of constant size. The only difference between the two models is the slightly increased runtime of the *replay* model, which can be explained by the 3500 rows of exemplars added to the current generation training data.

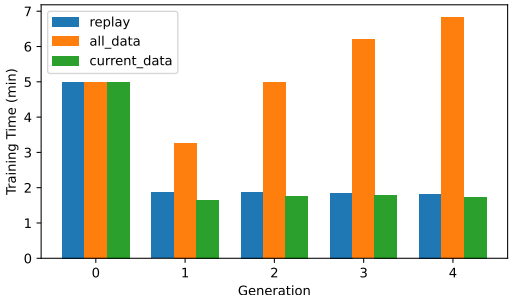

Figure 5: Average model training time in minutes.

4.2. Catastrophic Forgetting and Generalization (RQ2)

RQ2 evaluates the capability of the replay-based CL approach to mitigate catastrophic forgetting while also generalizing its learned knowledge effectively to predict unseen future events successfully. The evaluation of catastrophic forgetting is done by testing the replay model on historical data. If the performance is lower than before the retraining, the model has 'forgotten' what it has learned. To evaluate this, each model generation is evaluated against the validation data of generation zero. The experiment results assessed by the MSE are shown in Figure 6.

The *replay* model in generation one performs slightly worse than the other two models but then improves its performance in later generations. This shows that catastrophic forgetting can be limited even with a limited exemplar set that gets replayed during training. Comparing the performance of the *replay* and the *all_data* model, it can be seen

[2]https://github.com/UH-MLOps/replay-cl-in-ml-pipelines

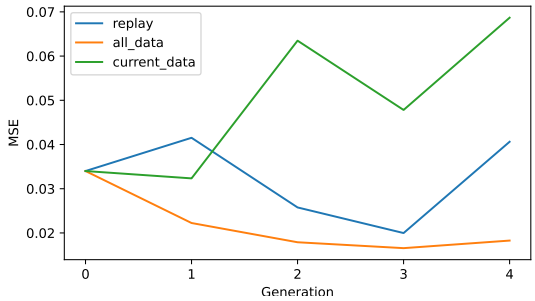

Figure 6: Performance of each model generation validated against the validation set of generation zero.

that the *replay* model performs only slightly worse in most generations, which is impressive considering the significant size difference of the training datasets between the models.

The Evaluation of model generalization is done by the performance of the replay model in predicting the unseen validation data of generation four. Figure 7 shows the result of these experiments.

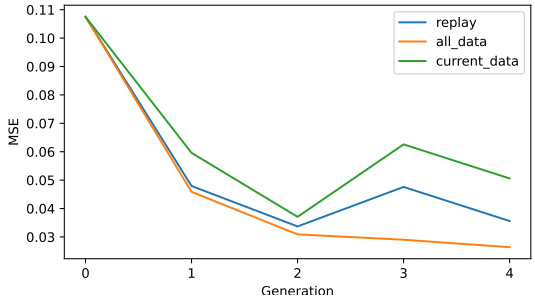

Figure 7: Performance of each model generation validated against the validation set of the last generation.

The experiments show good generalization results for all three models, which confirms that all models successfully learn the data patterns of the sensor data. The *replay* model generalization performance is close to the *all_data* model and overall better than the *current_data* model. The worst overall performing model of the retrained models is the *current_data* model, especially in generation three, with an error of 0.06. Interestingly, there is a significant improvement in the performance of the *replay* model compared to the *current_data* model, considering the only difference between the two are the 3500 rows of historical data available to the *replay* model. This limited set of historical samples brings the accuracy of the *replay* model close to the performance of the *all_data* model, which has access to all historical data for training.

4.3. Exemplar Set Size (RQ3)

RQ3 investigates the optimal exemplar set by conducting multiple experiments with different sizes to evaluate the selected exemplar set size. The selected size of 3500 rows is compared against a set half its size with 1750 rows and a set twice its size with 7000 rows. Since the primary goal of replay-based CL is to reduce the possibility of catastrophic forgetting, the same evaluation as in Section 4.3 is performed to assess the model's performance on the validation data of generation zero. Figure 8 shows the prediction performance of three replay-based models.

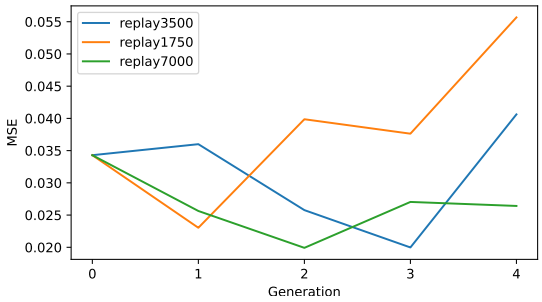

Figure 8: Prediction performance of the three different replay models against the validation set of generation zero.

Although the *replay1750* model starts well in generation one, it quickly loses prediction accuracy in later generations, showing the effect of catastrophic forgetting in a similar pattern as the *current_data* model in Figure 6. This confirms the expectation that an exemplar set with only 1750 rows is insufficient in the scenario of our study. The *replay3500* and the *replay7000* model show better performances across all generations. The *replay7000* model performs best except in generation three, showing a relatively constant mean squared error between 0.020 and 0.026 across all generations.

5. Discussion and Conclusion

This section discusses the results and concludes the study with general observations. From the findings presented for RQ1 in Section 4.1, the replay-based approach showed good performance across all training generations. This is a significant result since the model's main task in a production scenario is to accurately predict the ship's current performance until new data is available. The replay-based CL approach has a lower error rate than the *current_data* approach in all training generations. The replay-based approach also has the smallest combined error over all training generations out of the three approaches. This shows that the samples in the exemplar set help the *replay* model successfully improve its prediction performance in contrast to the *current_data* approach with no access to historical data.

Interestingly, the *all_data* model performs worst out of the three in generations one and two. A possible explanation for this is that the *all_data* model likely has higher model stability compared to the other two model types, meaning it is less likely to change during training. This stability is caused by the extensive dataset on which it gets trained. Since all historical data is present during training, the model is optimized to predict all data with an equally high accuracy. In contrast to that, the *replay* approach (and the *current_data* approach) focuses the training on the current generation data and can therefore achieve a higher accuracy on this smaller current dataset. Considering the suggestions of the domain expert mentioned in Section 3.3.1 regarding the use of historical data for model training, it can be argued that higher model plasticity seen in the *replay* and *current_data* approaches is favorable, because it allows

the model to adapt closer to the current dataset. Generally, changes in performance are expected in the context of maritime ship performance prediction since ships experience a broad range of weather conditions and operational settings over their lifetime. If specific patterns are not contained in a model's training data, it is nearly impossible for the model to accurately estimate a new operational condition [10]. For example, a model trained on calm weather data will lack the capability to accurately estimate a ship's fuel consumption in a high winds scenario.

RQ2 (Section 4.2) aimed to evaluate the effect of catastrophic forgetting on the three compared models. In the context of ship performance prediction, the model must accurately predict a ship's performance in various operational conditions. Since not all conditions are encountered within a limited timespan of the ship's lifetime, the prediction model must remember previously encountered conditions even if they have not been part of the last training data. As expected, the *current_data* model performed worst in this test since it did not consider historical data during retraining and successively forgot knowledge it had learned in earlier generations. More importantly, the difference in performance were compared between the *replay* and *all_data* models. The *replay* model performed exceptionally, considering its limited dataset compared to the *all_data* model. Considering the domain context of the models, this result shows successful mitigation of forgetting across many generations when using a replay-based CL approach. The slightly worse performance of the *replay* model is explained by the limited amount of historical data the model has access to during training. These two differences in performance and size of training data highlight the trade-off that has to be considered when setting up a CL training pipeline, which is also discussed by [8]: Which performance is expected after training? What amount of training data (and therefore training runtime) is within defined limits? The answer to both questions depends on the specific usage context and available resources for the pipeline operation. Considering the findings of training runtimes, it can be concluded that an unconstrained training set that grows with every generation, as seen for the *all_data* model, will not be feasible for a production setup. Therefore, the question about the dataset size should be reasoned on a fixed size.

RQ3 (Section 4.3) considers different sizes for the exemplar set used in a replay-based CL approach. Most literature [7], do not state a specific approach to determine the exemplar set size. Only [7] mentioned their exemplar set size, which was used as the starting point of this study. Regarding the exemplar selection approach, a common practice is to randomly select exemplars while also maintaining an equal size of exemplar subsets [7]. However, the common practice was refined in this study based on inputs from domain experts that favoured recent data points in the exemplar set. As a result, the exemplar set kept more samples of recent generations and only a few samples from older generations. The latter consideration aimed to select data points that accurately show a ship's performance in various weather conditions while being sensitive not to include outliers and sensor errors. As noted by researchers, samples with higher deviations, specific patterns, and worse predictions are important to mitigate catastrophic forgetting [4, 8].

The experiments show that the best-performing model for mitigating catastrophic forgetting is the *replay7000* model, which has the most training data. However, this model also has the longest training runtime. It was decided to use the *replay3500* model with an exemplar set size of 3500 for the experiments since its performance regarding mitigation of catastrophic forgetting was considered as satisfactory and also its quicker training time was determined as beneficial for experiments on the limited available hardware. As mentioned above and also supported by [8], this trade-off must be considered individually for each application scenario. Considering the application scenario of the study, a significant factor that should be considered in a production setup is the number of ships for which this pipeline will be operated. For one ship, increasing the exemplar set size and training runtime might be feasible to improve prediction performance. However, when considering a production scenario where the pipeline trains models for multiple ships, it has to be determined if the execution environment has the necessary resources to handle the increased amounts of data and computational loads that accumulate for the ships.

Acknowledgments

The work was funded by the VesselAI EU project[3], enabling the collaboration between the Uni. of Helsinki and NAPA[4].

References

[1] P. Gupta, A. Rasheed, S. Steen, Ship performance monitoring using machine-learning 254 (2022) 111094.

[2] I. Prapas, B. Derakhshan, A. R. Mahdiraji, V. Markl, Continuous training and deployment of deep learning models, Datenbank-Spektrum 21 (2021) 203–212.

[3] L. Wang, X. Zhang, H. Su, J. Zhu, A comprehensive survey of continual learning: Theory, method and application, IEEE Transactions on Pattern Analysis and Machine Intelligence (2024) 1–20.

[4] B. Bagus, A. Gepperth, An Investigation of Replay-based Approaches for Continual Learning, in: 2021 International Joint Conference on Neural Networks, 2021, pp. 1–9.

[5] J. Hurtado, D. Salvati, R. Semola, M. Bosio, V. Lomonaco, Continual learning for predictive maintenance: Overview and challenges, Intelligent Systems with Applications (2023) 200251.

[6] Y. He, B. Sick, Clear: An adaptive continual learning framework for regression tasks, AI Perspectives 3 (2021) 2.

[7] S. Gao, H. Zhang, C. Gao, C. Wang, Keeping Pace with Ever-Increasing Data: Towards Continual Learning of Code Intelligence Models, in: 45th International Conference on Software Engineering, 2023, pp. 30–42.

[8] G. Merlin, V. Lomonaco, A. Cossu, A. Carta, D. Bacciu, Practical Recommendations for Replay-Based Continual Learning Methods, in: Image Analysis and Processing Workshops, Springer, 2022, pp. 548–559.

[9] Z. Zhao, Cost-Effective Decision Making in Weather Routing using Machine Learning-generated Simulation Data, 2023. URL: http://hdl.handle.net/10138/565800.

[10] A. Coraddu, L. Oneto, F. Baldi, F. Cipollini, M. Atlar, S. Savio, Data-driven ship digital twin for estimating the speed loss caused by the marine fouling 186 (2019) 106063. doi:10.1016/j.oceaneng.2019.05.045.

[3] https://cordis.europa.eu/project/id/957237

[4] https://www.napa.fi/