Dynamic Multi-Modal Particle Swarm Optimization for Training Neural Network Ensembles Under Concept Drift

Chris H. Langeveldt, *Student Member, IEEE*
Computer Science Division
Stellenbosch University
Stellenbosch, South Africa
chl@sun.ac.za

Andries P. Engelbrecht, *Senior Member, IEEE*
Computer Science Division
Stellenbosch University
Stellenbosch, South Africa
engel@sun.ac.za

***Abstract*—Artificial neural networks (NNs) often struggle to maintain performance when concept drift, a phenomenon where the underlying data distribution changes over time, occurs. While traditional training approaches based on backpropagation struggle to maintain performance under drift conditions, ensemble learning combined with dynamic multi-modal (DMM) optimization offers a promising alternative. This paper investigates the applicability of multi-quantum swarm optimization (MQSO), a DMM particle swarm optimization (PSO) variant, for training NN ensembles in dynamic environments. MQSO is evaluated against traditional gradient-based methods and standard quantum swarm optimization. MQSO achieves superior performance on lower-dimensional problems with simpler decision boundaries, effectively tracking multiple optima. However, gradient-based methods with retraining outperform MQSO on higher-dimensional, non-linear problems due to computational constraints when using MQSO. The findings further show that retraining mechanisms are essential for maintaining performance under concept drift. It is also clear that DMM PSO is particularly effective for tracking rapidly changing decision boundaries.**

***Index Terms*—Concept drift, dynamic optimization, ensemble learning, multi-modal optimization, neural networks, particle swarm optimization.**

I. Introduction

In the real world, data is constantly evolving, and machine learning models must adapt to these changes to maintain their accuracy and reliability. As an example, consider the meals you prepare for your child. Over time the taste preferences of your child will change and what they loved as a toddler might be completely rejected a few years later. If you continue serving the same meals for years, your child will likely be dissatisfied. Similarly, in machine learning, the underlying relationship between input data (what you cook) and the targets (what your child enjoys) can shift over time. This phenomenon, known as *concept drift*, poses significant challenges for classification models. Traditional neural networks (NNs), typically trained on static data, often suffer from performance degradation when deployed in dynamic environments [1], [2].

To address this, dynamic adaptation mechanisms and ensemble learning are often employed. Ensemble methods combine the predictions of multiple models. Ensembles can maintain a diverse set of solutions that may be effective under different data distributions, making them well-suited to drift scenarios [3], [4]. However, effective training of these ensembles requires an optimization algorithm capable of locating and tracking multiple optima in a changing loss landscape.

Particle swarm optimization (PSO), inspired by collective swarm behavior, offers a meta-heuristic alternative to gradient-based NN training [5]. Variants of PSO have been developed to handle multi-modal optimization problems [6], [7], as well as dynamic optimization problems, where optima change over time [8], [9]. The multi-quantum swarm optimization (MQSO) algorithm combines both capabilities: maintaining multiple swarms to discover diverse ensemble members while employing charged particles to track moving optima [9], [10].

This paper proposes the use of MQSO to train NN ensembles in concept drift scenarios. The mechanisms within MQSO should allow independent swarms to track different optima in the NN loss landscape, thereby maintaining ensemble diversity and accuracy as the underlying data distribution changes. This approach is evaluated against gradient-based baselines across varying drift magnitudes and frequencies. The novel contributions of this work include the application of MQSO for NN ensemble training under concept drift, systematic evaluation against traditional methods, and insights into the strengths and limitations of DMM PSO in dynamic learning environments.

To facilitate reproducibility and further research, the code used in this study is publicly available via a Zenodo repository at https://doi.org/10.5281/zenodo.19442418.

The remainder of this paper is structured as follows. Section II provides background on the relevant concepts. Section III details the experimental design. Section IV presents results and discussion. Finally, Section V concludes with key findings and future research directions.

II. Background

This section provides an overview of the key concepts relevant to this study.

A. Neural Networks

NNs are computational models inspired by biological neural structures, designed to process information in order to learn complex patterns and relationships within data [11], [12]. The fundamental architecture used in this study is the feedforward neural network (FNN), where information flows in one direction from the input to the output layer. NNs consist of interconnected layers of nodes called neurons, each applying a weighted sum followed by an activation function to its inputs. The goal of a NN training algorithm is to optimize its parameters (i.e. weights and biases) to minimize a loss function that quantifies the discrepancy between predicted and actual outputs.

The backpropagation algorithm, using gradient descent [13], is the standard training method used to find optimal NN parameter values [14], [15]. Backpropagation computes the gradient of the loss function with respect to each weight by applying the chain rule of calculus, allowing efficient updates through iterative adjustments. It is, however, susceptible to local minima and requires differentiable loss and activation functions.

B. Ensemble Learning

Ensemble learning improves prediction accuracy and robustness by aggregating the outputs of multiple base learners. The success of an ensemble relies on the diversity-accuracy trade-off: members must be sufficiently accurate while making uncorrelated errors. Ensembles can be homogeneous (i.e. same base learner type) or heterogeneous (i.e. different types).

Common ensemble generation methods include bagging, boosting, and stacking [16]–[18]. Bagging creates diversity through bootstrap sampling, while boosting focuses on difficult instances by reweighting data points. Stacking combines predictions from multiple models using a meta-learner.

Predictions from ensemble members can be combined through various methods, such as majority voting for classification or averaging for regression. More sophisticated techniques involve weighted voting or using a meta-learner to learn optimal combination strategies.

In the context of NNs, ensembles can mitigate overfitting and provide better generalization than single models [19], [20]. It is important to maintain diversity among ensemble members, which can be achieved through different NN initializations, architectures, training data subsets, or optimization methods. This research leverages ensemble methods to maintain multiple diverse NNs capable of adapting to concept drift.

C. Concept Drift

A concept can simply be defined as a mapping between input features and the target variable. When this mapping changes over time, concept drift occurs [21]. Machine learning models must adapt to these changes in the data distribution to maintain performance.

Drift can be categorized along several dimensions including the drift subject, drift magnitude and drift frequency. The drift subject refers to the aspect of the data that is changing. The three main types of drift subjects are real drift, virtual drift, and novel class appearance [22]. Real drift, also known as class drift, occurs when the relationship between input features and the target variable changes. Real drift is the most common form of concept drift, and generally the type referred to when discussing concept drift. This is also the type of drift primarily focused on in this study. Virtual drift, also known as covariate drift, occurs when the distribution of input features changes over time, while the relationship between input features and the target variable remains the same. Novel class appearance refers to the emergence of new classes in the data that were not present during the initial training of the model.

Drift magnitude refers to the extent of change in the data distribution. The magnitude of drift can vary from small, gradual changes to large, abrupt shifts. Drift frequency refers to how often concept drift occurs in the data stream. Some environments may experience frequent drift, while others may have infrequent or periodic drift events.

Drift detection and adaptation are critical components of handling concept drift. Drift detection methods monitor the data stream for changes in distribution or performance. Adaptation mechanisms update the model to accommodate these changes. Common adaptation strategies include retraining the model on recent data [23], [24], partial updates [25], and ensemble methods that maintain a diverse set of models [26], [27].

D. Particle Swarm Optimization

PSO is an iterative population-based stochastic optimization technique inspired by the social behavior of birds flocking [5]. For standard PSO, candidate solutions (particles) traverse the search space, updating their positions based on their personal best experience and the global best experience of the swarm.

PSO has been successfully applied in the context of NN training for various tasks. One common application is the use of PSO for hyperparameter optimization, where PSO is employed to search for optimal configurations of learning rates, network architectures, and regularization parameters [28], [29]. PSO has also been used to train the weights and biases of NNs directly [30], [31], providing an alternative to gradient-based methods. This approach uses PSO to search for an optimal set of weights and biases that minimizes the loss function of the NN on a training dataset.

The use of PSO for NN training offers several advantages, including the ability to escape local minima and the flexibility to handle non-differentiable loss functions. However, PSO can be computationally expensive, especially for large NNs, due to the fitness evaluations of multiple particles in each iteration.

When applied to dynamic environments, standard PSO faces several limitations. These limitations generally fall into two categories, i.e. change detection and adaptation mechanisms. Standard PSO lacks inherent change detection mechanisms, making it challenging to identify when the environment has changed. Without this capability, the swarm may continue to exploit outdated information, leading to suboptimal performance. Additionally, standard PSO does not possess ef-

fective adaptation mechanisms to respond to changes in the environment. Once the environment changes, the swarm may struggle to reorient itself towards new optima, resulting in slow convergence and reduced solution quality.

E. Quantum Swarm Optimization

To address the limitations of PSO in dynamic environments, Blackwell and Branke introduced quantum swarm optimization (QSO) [9]. Inspired by the quantum model of the atom, QSO replaces the deterministic velocity update with probabilistic position sampling. The swarm consists of *neutral* particles, which facilitate exploitation, and *charged* particles, which sample positions within a "cloud radius" around the global best solution. This continuous exploration allows the swarm to track moving optima effectively.

QSO has been applied to various optimization problems, including FNN training in dynamic environments [32]. The charged particles in QSO enable the swarm to adapt to changes in the loss landscape, maintaining performance as the underlying data distribution shifts. However, QSO is primarily designed for single-modal optimization and may be suboptimal for scenarios requiring the tracking of multiple optima.

F. Multi-Quantum Swarm Optimization

MQSO extends QSO to handle multi-modal landscapes by maintaining multiple independent quantum swarms, each capable of tracking a distinct peak in the search space [9]. MQSO can be visualized in three dimensions as multiple clouds of charged particles, each centered around a different optimum, with neutral particles facilitating exploitation within each swarm. Fig. 1 shows a conceptual illustration of MQSO with three swarms each tracking a different optimum.

Blackwell and Branke later introduced mechanisms to promote diversity among swarms and prevent premature convergence [10]. To prevent redundant convergence, MQSO implements an *exclusion* mechanism: if two swarms move within a specific radius of each other, the less fit swarm is re-initialized. Additionally, an *anti-convergence* mechanism resets the worst-performing swarm if all swarms stagnate, ensuring continuous exploration [10]. This makes MQSO ideal for training NN ensembles, as each swarm can independently optimize a diverse ensemble member.

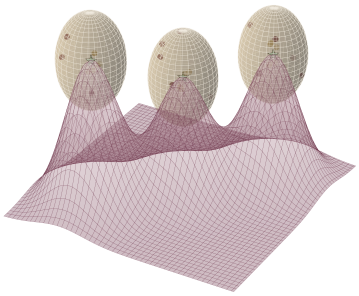


Fig. 1. Conceptual Illustration of Multi-Quantum Swarm Optimization

In the context of NN ensemble training, each particle in MQSO represents a complete NN, with the position vector of the particle representing the weights and biases of the NN. MQSO naturally promotes ensemble diversity, because different swarms converge to different minima in the loss landscape.

III. Experimental Design

This section outlines the experimental setup used to evaluate MQSO for training NN ensembles under concept drift.

A. Experimental Datasets

Two synthetic datasets commonly used in concept drift research were selected for evaluation, i.e. the streaming ensemble algorithm (SEA) dataset [33] and the random radial basis function (RBF) dataset [34]. Both datasets allow controlled manipulation of drift characteristics, facilitating systematic analysis.

SEA Dataset: The SEA dataset provides binary classification with three continuous features. The decision boundary is defined by $y = 1$ if $x_0 + x_1 > \epsilon(t)$, otherwise $y = 0$, where $\epsilon(t)$ is a time-varying threshold and x_2 is noise. Features were sampled uniformly from $[0, 10]$ with 10% label noise. The dataset for each experiment contained 10,000 samples.

Random RBF Dataset: This dataset provides multi-class classification (5 classes) with 10 features. Classes are defined by clusters around 20 randomly positioned centroids, with samples drawn from Gaussian distributions around each centroid. This creates complex, overlapping decision boundaries. Features were sampled from $[0, 1]$. Each experiment contained 10,000 samples.

B. Concept Drift Simulation

Drift was simulated along two dimensions, i.e. magnitude and frequency. Three levels of magnitude corresponding to gradual, moderate, and abrupt modifications were tested. Three levels of frequency corresponding to infrequent, periodic, and frequent drift occurrences were tested.

For the SEA dataset, drift was introduced by randomly varying the threshold $\epsilon(t)$ at specified intervals. For the random RBF dataset, centroids moved through the feature space at speeds corresponding to the drift magnitude in random directions.

Combined with a static (no drift) baseline, this yields 10 experimental configurations per dataset. Table I summarizes the drift configurations used in the experiments.

TABLE I
Drift Configurations

Dim.	Level	SEA Drift Configuration	Random RBF Drift Configuration
Magn.	Gradual	0.5–1.0 change of $\epsilon(t)$	0.5–1.0 change speed
	Moderate	1.5–2.5 change of $\epsilon(t)$	1.5–2.5 change speed
	Abrupt	3.5–5.0 change of $\epsilon(t)$	3.5–5.0 change speed
Freq.	Infrequent	Every 5 000 samples	Every 5 000 samples
	Periodic	Every 2 500 samples	Every 2 500 samples
	Frequent	Every 1 000 samples	Every 1 000 samples

C. Algorithms Evaluated

A total of six algorithms were evaluated, including two PSO variants and four gradient-based baselines. The PSO variants were QSO, and MQSO. The gradient-based baselines included standard backpropagation (BP), BP with retraining (BP_RE), BP with dropout (BP_DR), and BP_RE with dropout (BP_RE_DR). Dropout was added to simulate an implicit ensemble effect to provide a baseline comparison for the explicit ensembles trained via MQSO.

For MQSO, each particle represented a complete NN with its position vector encoding all NN weights and biases. Each swarm independently optimized its particles to train a separate NN ensemble member. Predictions from ensemble members were combined through averaging the outputs for each class, with the final prediction being the class with the highest average probability.

D. Hyperparameter Tuning

All algorithms underwent hyperparameter tuning to ensure fair comparison. NN architecture (i.e. hidden layer size) was tuned on static datasets using standard backpropagation and kept constant across all algorithms to ensure comparability.

For backpropagation variants, Adam optimizer [35] parameters (i.e. learning rate, β_1, β_2) and dropout rates were tuned iteratively on static datasets. For QSO and MQSO, particle composition and cloud radius across different drift magnitudes were tuned. MQSO-specific parameters (i.e. exclusion radius and number of swarms) were tuned on static datasets with reduced sample sizes to manage computational costs.

All hyperparameter values are summarized in Table II.

E. Evaluation Metrics

Performance was evaluated using various metrics. Traditional metrics such as loss, accuracy, and F1-score were computed on both training and test sets. For the SEA dataset, binary cross-entropy loss was used, while for the random RBF dataset, categorical cross-entropy loss, weighted and macro-averaged F1-scores were computed. These traditional classification metrics provide an overview of algorithm performance at each time window, but do not provide a metric to quantify the overall adaptation effectiveness of the algorithms.

The relative error distance (RED) metric quantifies the overall performance of algorithms on dynamic optimization problems across various time windows [36]. The RED combines a performance metric, for example loss, across all time windows into a single measure. The RED metric is defined as

$$\text{RED}_m = \sqrt{\sum_{t=1}^{T}(1 - b_{t,m})^2} \Big/ \sqrt{T}$$

where $b_{t,m}$ is the performance value of the metric m (e.g., loss) at time window t, and T is the total number of time windows. This formulation expects $b_{t,m}$ to be a value between 0 and 1, where 1 represents the best possible performance. The RED approach is used to quantify all other metrics across the time windows, with smaller values indicating better adaptation effectiveness.

Each experiment consisted of 30 independent trials with different random seeds. Statistical validation employed Mann-Whitney U tests for pairwise comparisons, with algorithms ranked based on the RED values of each metric on the testing data. Mean performance with 95% confidence intervals provided primary comparison metrics.

TABLE II
OPTIMAL HYPERPARAMETER VALUES

Algorithm	**Parameter**	**Optimal Value**	
		SEA	RBF
All algorithms	Hidden layer size	3	14
BP-based algorithms	Learning rate	0.01	0.001
	β_1	0.9	0.99
	β_2	0.999	0.999
BP dropout variants	Dropout rate	0.1	0.1
QSO and MQSO	Particles (neutral, quantum)	(5, 5)	(5, 5)
	Cloud radius	0.1	0.01
MQSO	Number of swarms	10	10
	Exclusion radius	1.5	6

IV. RESULTS AND DISCUSSION

This section presents and discusses the experimental results.

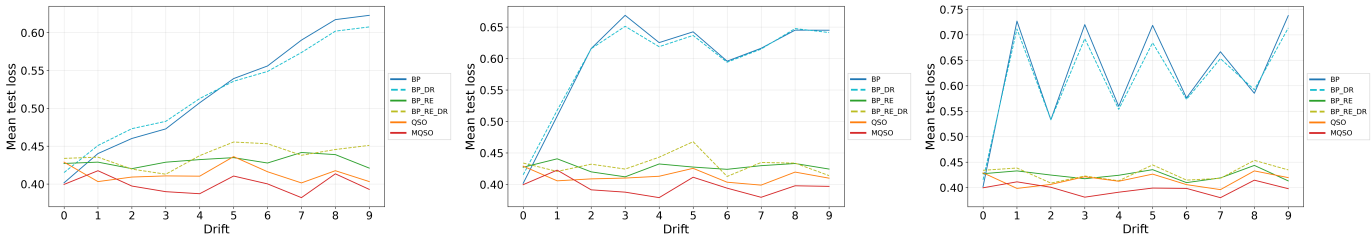
A. SEA Dataset Performance

Table III summarizes the average performance rankings across all SEA drift scenarios. MQSO consistently achieved superior performance, with an average training loss rank of 1.00 and strong performance for test loss (average rank of 2.06), test accuracy (average rank of 1.44), and test F1-score (average rank of 1.78). QSO also demonstrated excellent performance, particularly for accuracy and F1-score metrics. Both quantum-inspired algorithms outperformed backpropagation variants, particularly as drift frequency increased. BP variants with retraining (BP_RE and BP_RE_DR) showed improved performance over their non-retraining counterparts, highlighting the importance of adaptive mechanisms in concept drift scenarios. However, they still lagged behind the quantum-inspired approaches. Fig. 2 further illustrates these findings by showing the average test loss over time for the SEA dataset under frequent drift conditions.

B. Random RBF Dataset Performance

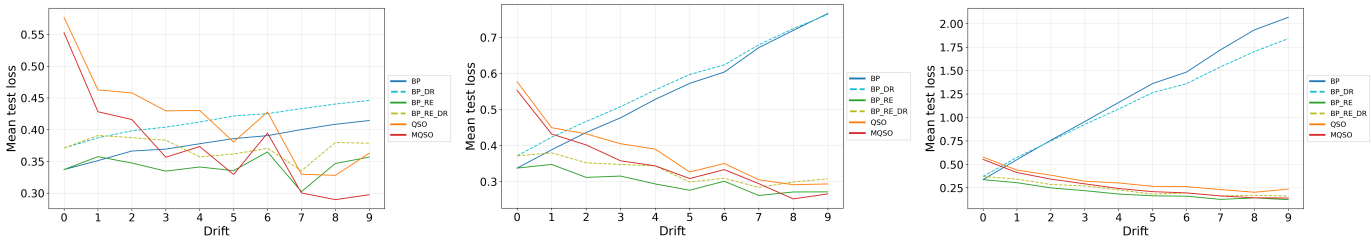
The RBF dataset reveals different algorithm strengths, as seen in Table IV. Here, BP_RE outperformed all other algorithms across all metrics, achieving average ranks of 1.28 for training loss, 1.56 for test loss and accuracy, 1.50 for weighted F1-score, and 1.78 for macro F1-score. This suggests that for higher-dimensional, non-linear problems, gradient-based methods with retraining might be more effective at adapting to concept drift than PSO variants. MQSO still outperformed standard QSO, indicating that the multi-swarm architecture provides benefits even in more complex landscapes.

The primary limitation of the QSO-based approaches is the high dimensionality of the RBF problem combined with computational constraints on the number of iterations. PSO-based methods generally require more function evaluations to



(a) Gradual Drift (b) Moderate Drift (c) Abrupt Drift

Fig. 2. SEA Dataset Test Loss Over Time Under Frequent Drift Conditions



(a) Gradual Drift (b) Moderate Drift (c) Abrupt Drift

Fig. 3. Random RBF Dataset Test Loss Over Time Under Frequent Drift Conditions

TABLE III
AVERAGE RANKS ON SEA DATASET (SMALLER IS BETTER)

Algorithm	Train Loss	Test Loss	Accuracy	F1-Score
BP	2.67	5.22	5.50	5.06
BP_DR	3.89	5.56	5.50	4.89
BP_RE	3.56	2.11	3.33	3.50
BP_RE_DR	5.11	2.89	3.61	4.06
QSO	4.78	3.17	1.61	**1.72**
MQSO	**1.00**	**2.06**	**1.44**	1.78

TABLE IV
AVERAGE RANKS ON RANDOM RBF DATASET (SMALLER IS BETTER)

Algorithm	Train Loss	Test Loss	Accuracy	Weighted F1-Score	Macro F1-Score
BP	2.00	3.72	3.72	3.72	3.62
BP_DR	4.17	4.11	4.33	4.33	4.00
BP_RE	**1.28**	**1.56**	**1.56**	**1.50**	**1.78**
BP_RE_DR	3.17	2.11	2.11	2.17	2.28
QSO	5.78	5.11	5.06	5.06	5.11
MQSO	4.61	4.39	4.22	4.22	4.22

converge in high-dimensional spaces compared to gradient-based methods. However, analysis showed that as drift frequency increased (effectively allowing more iterations relative to the changing concept), the relative performance of MQSO improved, narrowing the gap with BP_RE.

Fig. 3 illustrates the average test loss over time for the RBF dataset under frequent drift conditions, highlighting the superior adaptation of BP_RE, as well as the performance improvements of MQSO and QSO over time.

V. CONCLUSIONS

This paper investigated the application of MQSO for training NN ensembles under concept drift scenarios. Through systematic experiments on two synthetic datasets with controlled drift characteristics, MQSO was compared against multiple baseline algorithms.

The results demonstrate that algorithm selection must be informed by problem complexity. MQSO achieved superior performance on the SEA dataset across all drift scenarios but struggled on the more complex RBF dataset where BP_RE emerges as the top performer. The primary limitation of MQSO is computational complexity, which becomes prohibitive as problem dimensionality increases. This limitation was confirmed by the improved relative performance of MQSO as drift frequency increases, effectively allowing more iterations between concept changes. Given sufficient computational resources, MQSO could potentially close the performance gap on higher-dimensional problems.

The critical importance of retraining mechanisms emerged consistently across all experiments. Algorithms without adaptation capabilities exhibited poor performance in drift scenarios, confirming that single-training approaches are inadequate for dynamic environments.

Several promising directions for future work include: (1) evaluating MQSO on real-world concept drift problems to validate findings from synthetic experiments; (2) extending evaluation to advanced NN architectures beyond single-hidden-layer FNN; (3) exploring adaptive ensemble configuration strategies that adjust swarm parameters based on observed drift characteristics; and (4) investigating computational optimizations for MQSO to improve scalability.

An interesting idea that emerged throughout this study in order to improve computational efficiency, is to employ a mini-batch training strategy within MQSO. Transitions between mini-batches functionally behave like concept drift, allowing the dynamic capabilities of MQSO to naturally adapt to the stochastic noise introduced by mini-batch training.

Ultimately, this study establishes that for problems with moderate dimensionality and frequent drift, MQSO offers a robust alternative to gradient-based training. While backpropagation with periodic retraining remains more practical for high-dimensional constraints, the dynamic tracking capabilities of MQSO provide a specialized solution for maintaining model relevance in rapidly evolving data streams.

ACKNOWLEDGMENT

The authors acknowledge the use of AI-assisted tools for identifying relevant literature, editing code, and improving the readability and language during the preparation of this manuscript. The specific systems used include Gemini 3, Claude Sonnet 4.5, and Perplexity.

REFERENCES

[1] S. M. Jameel, M. A. Hashmani, H. Alhussain, M. Rehman, and A. Budiman, "A critical review on adverse effects of concept drift over machine learning classification models," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 1, pp. 206–211, 2020.

[2] N. Jourdan, T. Bayer, T. Biegel, and J. Metternich, "Handling concept drift in deep learning applications for process monitoring," *Procedia of the International Academy for Production Engineering*, vol. 120, no. 1, pp. 33–38, 2023.

[3] D. Brzezinski and J. Stefanowski, "Reacting to different types of concept drift: The accuracy updated ensemble algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 81–94, 2013.

[4] L. L. Minku, A. P. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 5, pp. 730–742, 2009.

[5] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the International Conference on Neural Networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[6] R. Brits, A. P. Engelbrecht, and F. van den Bergh, "A niching particle swarm optimizer," in *Proceedings of the Conference on Simulated Evolution and Learning*, 2002.

[7] C. Castillo, G. Nitschke, and A. Engelbrecht, "Niche particle swarm optimization for neural network ensembles," in *Advances in Artificial Life: Darwin Meets von Neumann*, ser. Lecture Notes in Computer Science, G. Kampis, I. Karsai, and E. Szathmáry, Eds. Springer, 2011, vol. 5778, pp. 399–407.

[8] T. Blackwell and P. Bentley, "Dynamic search with charged swarms," in *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann, 2002, pp. 19–26.

[9] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," in *Applications of Evolutionary Computing*, ser. Lecture Notes in Computer Science, G. R. Raidl, S. Cagnoni, J. Branke, D. W. Corne, R. Drechsler, Y. Jin, C. G. Johnson, P. Machado, E. Marchiori, F. Rothlauf, G. D. Smith, and G. Squillero, Eds. Springer, 2004, vol. 3005, pp. 489–500.

[10] ——, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 459–472, 2006.

[11] W. S. McCulloch and W. H. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, no. 1, pp. 115–133, 1943.

[12] F. Rosenblatt, "The perceptron: A perceiving and recognizing automaton," Cornell Aeronautical Laboratory, Tech. Rep. 85-460-1, 1957.

[13] A. Cauchy, "Méthode générale pour la résolution des systemes d'équations simultanées," *Comptes Rendus de l'Académie des Sciences*, vol. 25, pp. 536–538, 1847.

[14] P. J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Ph.D. dissertation, Harvard University, Cambridge, MA, 1974.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[16] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[17] R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, no. 2, pp. 197–227, 1990.

[18] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[19] L. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[20] M. P. Perrone and L. N. Cooper, "When networks disagree: Ensemble methods for hybrid neural networks," in *How we learn; How we remember: Toward an understanding of brain and neural systems: Selected papers of Leon N Cooper*. World Scientific, 1995, pp. 342–358.

[21] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proceedings of the Brazilian Symposium on Artificial Intelligence*. Springer, 2004, pp. 286–295.

[22] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, "Characterizing concept drift," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 964–994, 2016.

[23] S. H. Bach and M. A. Maloof, "Paired learners for concept drift," in *Proceedings of the Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 23–32.

[24] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the SIAM International Conference on Data Mining*. SIAM, 2007, pp. 443–448.

[25] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the Seventh ACM International Conference on Knowledge Discovery and Data Mining*. ACM, 2001, pp. 97–106.

[26] A. Bifet, G. Holmes, and B. Pfahringer, "Leveraging bagging for evolving data streams," in *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2010, pp. 135–150.

[27] F. Chu and C. Zaniolo, "Fast and light boosting for adaptive mining of data streams," in *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2004, pp. 282–292.

[28] P. R. Lorenzo, J. Nalepa, M. Kawulok, L. S. Ramos, and J. R. Pastor, "Particle swarm optimization for hyper-parameter selection in deep neural networks," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2017, pp. 481–488.

[29] C. Ren, N. An, J. Wang, L. Li, B. Hu, and D. Shang, "Optimal parameters selection for BP neural network based on particle swarm optimization: A case study of wind speed forecasting," *Knowledge-Based Systems*, vol. 56, pp. 226–239, 2014.

[30] A. Ismail and A. P. Engelbrecht, "Global optimization algorithms for training product unit neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 1. IEEE, 2000, pp. 132–137.

[31] F. van den Bergh and A. P. Engelbrecht, "Cooperative learning in neural networks using particle swarm optimizers," *South African Computer Journal*, vol. 2000, no. 26, pp. 84–90, 2000.

[32] A. S. Rakitianskaia and A. P. Engelbrecht, "Training feedforward neural networks with dynamic particle swarm optimisation," *Swarm Intelligence*, vol. 6, no. 3, pp. 233–270, 2012.

[33] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proceedings of the Seventh ACM International Conference on Knowledge Discovery and Data Mining*. ACM, 2001, pp. 377–382.

[34] M. J. D. Powell, *Radial basis functions for multivariable interpolation: a review*. Clarendon Press, 1987, pp. 143—-167.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations*, 2015.

[36] S. A. G. Van der Stockt, G. Pamparà, A. P. Engelbrecht, and C. W. Cleghorn, "Performance analysis of dynamic optimization algorithms using relative error distance," *Swarm and Evolutionary Computation*, vol. 66, p. 100930, 2021.