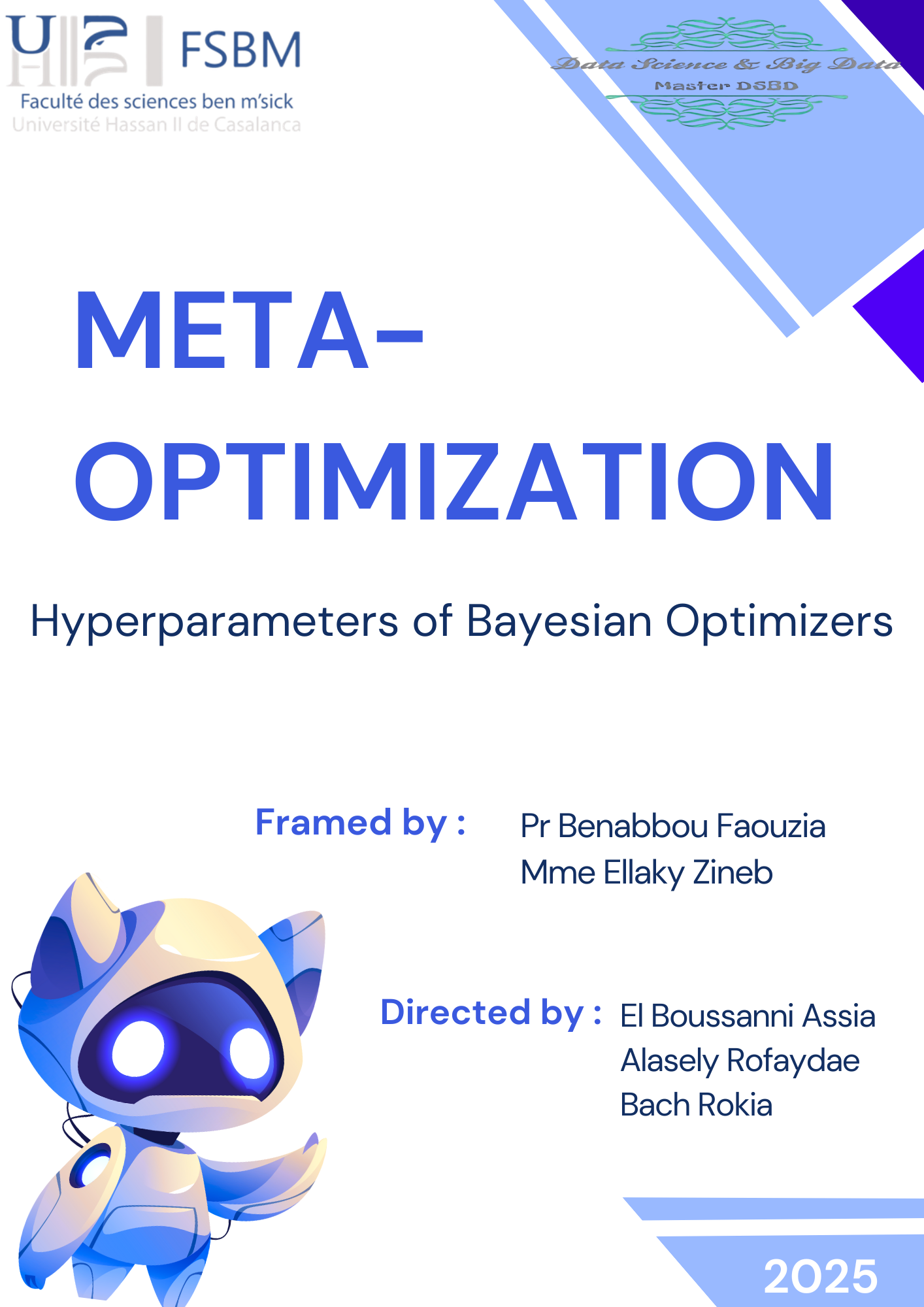
****

Table des matières

[*Abstract* 3](#_Toc202822210)

[Introduction 4](#_Toc202822211)

[Literature Review 5](#_Toc202822212)

[1- Classical Optimization Techniques 5](#_Toc202822213)

[2- The Rise of Intelligent Methods 5](#_Toc202822214)

[3- Application in Time Series Models 6](#_Toc202822215)

[4- Limitations in Existing Approaches 6](#_Toc202822216)

[Methodology 7](#_Toc202822217)

[1- Dataset 7](#_Toc202822218)

[2- LSTM Model Architecture 7](#_Toc202822219)

[3- Optimizers and Hyperparameters 7](#_Toc202822220)

[4- Bayesian Optimization Configuration 8](#_Toc202822221)

[5- Evaluation Metrics 8](#_Toc202822222)

[Results 9](#_Toc202822223)

[1- Best Hyperparameters Identified 9](#_Toc202822224)

[2- Overall Performance Comparison (Test MAE) 9](#_Toc202822225)

[3- MAE Evolution per Epoch (Convergence Analysis) 10](#_Toc202822226)

[4- Post-Optimization Prediction Accuracy 10](#_Toc202822227)

[Discussion 12](#_Toc202822228)

[1- Effectiveness of Bayesian Optimization 12](#_Toc202822229)

[2- Optimizer-Specific Insights 12](#_Toc202822230)

[3- Learning Curve Analysis 12](#_Toc202822231)

[4- Trade-off: Accuracy vs Computational Cost 12](#_Toc202822232)

[Conclusion 14](#_Toc202822233)

[1- Summary of Contributions 14](#_Toc202822234)

[2- Practical Recommendations 14](#_Toc202822235)

[References 15](#_Toc202822236)

[Annexes 16](#_Toc202822237)

# *Abstract*

This study presents a comparative analysis of optimization algorithms for training Long Short-Term Memory (LSTM) models in the context of weather time series prediction. We evaluate three popular optimizers—Stochastic Gradient Descent (SGD), Adam, and RMSprop—to assess their effectiveness in minimizing prediction error and improving training efficiency. Our LSTM model was trained on a real-world weather dataset, with performance measured using Mean Absolute Error (MAE) and training time.

To further enhance optimizer performance, we applied Bayesian Optimization (BO) to automatically tune the key hyperparameters of each algorithm. The results reveal that SGD, when tuned using BO, outperformed Adam and RMSprop in both accuracy and training speed, achieving the lowest MAE and the fastest convergence. The study underscores the critical role of optimizer selection and hyperparameter tuning in deep learning for time series forecasting.

These findings contribute to a better understanding of optimization strategies and demonstrate the practical value of combining LSTM models with meta-optimization techniques for improved predictive performance in weather forecasting tasks.

Keywords: LSTM, Time Series Prediction, Weather Forecasting, SGD, Adam, RMSprop, Bayesian Optimization, Hyperparameter Tuning, MAE

# Introduction

Time series prediction is a critical component in various domains such as finance, climate forecasting, and healthcare. Among the most successful models for such tasks are Long Short-Term Memory (LSTM) networks, which are specifically designed to handle sequential data and capture both short-term and long-term dependencies. Despite their power, the performance of LSTM models is highly sensitive to the choice of optimizer and its associated hyperparameters.

Optimizers like **Adam**, **SGD**, and **RMSprop** are fundamental to deep learning training processes. Each comes with its own strengths and weaknesses, affecting convergence speed, final accuracy, and stability. However, choosing the most effective optimizer and tuning its hyperparameters (e.g., learning rate, momentum) is a complex and non-trivial process. Manual tuning strategies such as grid search and random search are not only time-consuming but also computationally inefficient, especially when dealing with high-dimensional parameter spaces.

To overcome these limitations, this study proposes the use of **Bayesian Optimization (BO)** as an intelligent and automated approach for hyperparameter tuning. BO builds a probabilistic model of the objective function and selects hyperparameter configurations based on the likelihood of improvement, enabling more efficient exploration of the search space.

This work addresses the following central research problem: how can we systematically and effectively tune the hyperparameters of Adam, SGD, and RMSprop optimizers for LSTM models in time series forecasting, using Bayesian Optimization? We aim to identify which optimizer performs best when tuned using BO, and what computational trade-offs are involved.

Accordingly, the main objectives of this study are to:

* Investigate the use of Bayesian Optimization to tune the hyperparameters of Adam, SGD, and RMSprop.
* Compare the optimizers' performance in terms of **Mean Absolute Error (MAE),** **training time**, **convergence stability**, and **resource consumption**.
* Determine the most suitable optimizer for LSTM-based time series forecasting.
* Assess the real-world applicability and efficiency of Bayesian Optimization as a meta-optimization tool.

By systematically comparing default and BO-tuned optimizers, this study aims to contribute to the advancement of automated machine learning techniques and offer practical guidance for enhancing the performance of time series models.

# Literature Review

Hyperparameter optimization is a fundamental challenge in machine learning. These parameters—such as learning rate or momentum—are not learned during training but rather define how the training process operates. Their configuration directly influences the accuracy, convergence speed, and stability of the model.

In deep learning models for time series forecasting (especially LSTM and Time Series Transformers), the sensitivity to hyperparameter choices is particularly pronounced. This justifies the growing focus on hyperparameter optimization in recent literature.

## Classical Optimization Techniques

Early approaches to hyperparameter tuning were manual or exhaustive in nature:

* **Grid Search:** explores all possible combinations of hyperparameters in a fixed grid. While exhaustive, it quickly becomes impractical in high-dimensional spaces.
* **Random Search:** proposed as a more efficient alternative by Bergstra & Bengio (2012), it samples configurations randomly, often finding good solutions with fewer trials.

Despite their simplicity, these approaches suffer from high computational costs, low adaptability, and inefficiency in complex search spaces.

## The Rise of Intelligent Methods

Hyperparameter optimization entered a new era with the emergence of adaptive and probabilistic strategies that use feedback from previous evaluations to guide the search.

* **Bayesian Optimization** (Snoek et al., 2012): builds a probabilistic model (typically a Gaussian Process) of the objective function to intelligently select promising configurations. It is particularly suited to expensive evaluation functions such as training deep neural networks.
* **Hyperband** (Li et al., 2017): dynamically allocates resources using early stopping, halting underperforming trials early to reduce training time and resource consumption.

These methods are now integrated into modern AutoML and hyperparameter tuning frameworks such as Optuna, Ray Tune, and Keras Tuner.

## Application in Time Series Models

Time series forecasting is a challenging field due to its temporal dependencies, noisy data, and seasonality. Deep learning models such as **LSTM** (Hochreiter & Schmidhuber, 1997) and **Time Series Transformers** (Lim et al., 2021) offer high performance but are highly sensitive to optimizer hyperparameters (e.g., Adam, RMSprop).

In this context, automatic hyperparameter optimization is critical for building robust and high-performing models, while also keeping computational cost and training stability under control.

## Limitations in Existing Approaches

Despite their advantages, existing methods have some limitations:

* **Bayesian Optimization** often requires a warm-up phase and can be slow with noisy performance metrics.
* **Hyperband** might prematurely eliminate promising configurations due to early stopping.

Furthermore, most studies focus exclusively on accuracy as the primary evaluation metric. In practice, training cost, resource usage, and stability across runs are equally important and often overlooked in comparative analyses.

# Methodology

1. Dataset  
   This study uses a weather time series dataset, which undergoes preprocessing to ensure optimal model performance. All numerical features are standardized using **StandardScaler**, which centers the data (mean = 0) and scales it to unit variance (std = 1). This approach improves model stability by ensuring features contribute equally, regardless of their original scales. To create input samples for the LSTM model, sequences are formed by taking the previous 5 timesteps as features to predict the subsequent value. The dataset is divided into three subsets: 80% for training and 20% reserved for testing the model’s final predictive accuracy.

* Standardization: Mean-centered (μ=0) and scaled to unit variance (σ=1)
* Train/Test Split: 80% training, 20% testing

1. LSTM Model Architecture  
   The model features a stacked LSTM architecture with two LSTM layers: the first layer contains 64 units, allowing the network to capture complex temporal dependencies, while the second layer contains 32 units, further refining the learned patterns from the time series data. To reduce the risk of overfitting, a dropout layer with a rate of 0.2. The architecture is designed for single-step ahead forecasting, predicting the next value in the sequence based on previous observations. The model is trained using the Mean Absolute Error (MAE) as the loss function, due to its robustness and suitability for regression tasks with continuous numerical targets.

• Model Type: Sequential Stacked LSTM  
• Number of LSTM Layers: 2 Hidden  
• Units: 64 (first layer), 32 (second layer)  
• Dropout Rate: 0.2  
• Output Layer: Dense (1 neuron for single-step prediction)  
• Loss Function: Mean Absolute Error (MAE)

1. Optimizers and Hyperparameters  
   This study investigates three popular gradient-based optimizers: Adam, SGD, and RMSprop. For **Adam**, the learning rate is finely tuned within the range of 1e-5 to 1e-2, while the momentum parameters beta\_1 and beta\_2 are optimized within [0.8, 0.99] and [0.9, 0.999], respectively. **SGD**’s learning rate is explored between 1e-4 and 1e-1, alongside the momentum coefficient, which is adjusted from 0.5 up to 0.99 to capture varying degrees of inertial effect. **RMSprop**’s learning rate is also searched across the 1e-5 to 1e-2 interval, with its decay factor, rho, tuned between 0.8 and 0.99. These carefully selected hyperparameter ranges facilitate an exhaustive search to discover configurations that effectively minimize prediction errors.

* **Adam:**
* learning\_rate ∈ [1e-5, 1e-2]
* beta\_1 ∈ [0.8, 0.99]
* beta\_2 ∈ [0.9, 0.999]
* **SGD:**
* learning\_rate ∈ [1e-4, 1e-1]
* momentum ∈ [0.5, 0.99]
* **RMSprop:**
* learning\_rate ∈ [1e-5, 1e-2]
* rho ∈ [0.8, 0.99]

1. Bayesian Optimization Configuration  
   Hyperparameter tuning is performed using Bayesian Optimization implemented through the scikit-optimize library, specifically with the gp\_minimize function. The Expected Improvement (EI) acquisition function, used by default, guides the search by balancing exploration and exploitation. The optimization begins with a set of initial random samples (default: 10), followed by 20 iterative trials aimed at identifying the optimal hyperparameter settings for each optimizer (Adam, SGD, RMSProp). The objective function minimized during this process is the validation Mean Absolute Error (MAE), calculated on a fixed validation set. To ensure reliable performance estimation, an early stopping mechanism is applied based on the validation MAE during training.

* **Acquisition Function** : Expected Improvement (default)
* **Initial Random Points**: 10 (default of gp\_minimize)
* **Optimization Trials** : 20 evaluations per optimizer
* **Objective**: Minimize val\_mae during training with early stopping

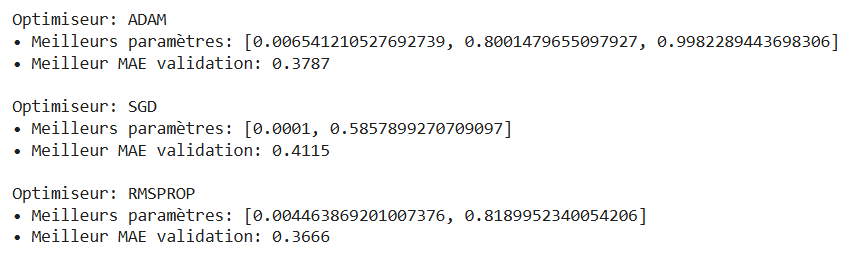
1. Evaluation Metrics  
   The effectiveness of the models is primarily assessed using the Mean Absolute Error (MAE), which quantifies the average absolute difference between predicted values and actual observations. In addition to MAE, other metrics such as training duration are considered to evaluate computational efficiency. The convergence behavior of each optimizer is analyzed to understand training stability and speed. Furthermore, the overall computational cost involved in the hyperparameter tuning process is taken into account. Together, these metrics provide a well-rounded evaluation of the optimizers’ performance and practicality post-tuning.

# Results

The meta-optimization task aimed to improve LSTM model performance for time series forecasting using **Bayesian Optimization** to fine-tune hyperparameters for three optimizers: **Adam**, **SGD**, and **RMSprop**. The main hyperparameters explored were the **learning rate**, **momentum**, and in the case of Adam, **beta parameters**.

## Best Hyperparameters Identified

The Bayesian search process yielded the following optimal configurations, resulting in a notable reduction in prediction error:



## Overall Performance Comparison (Test MAE)

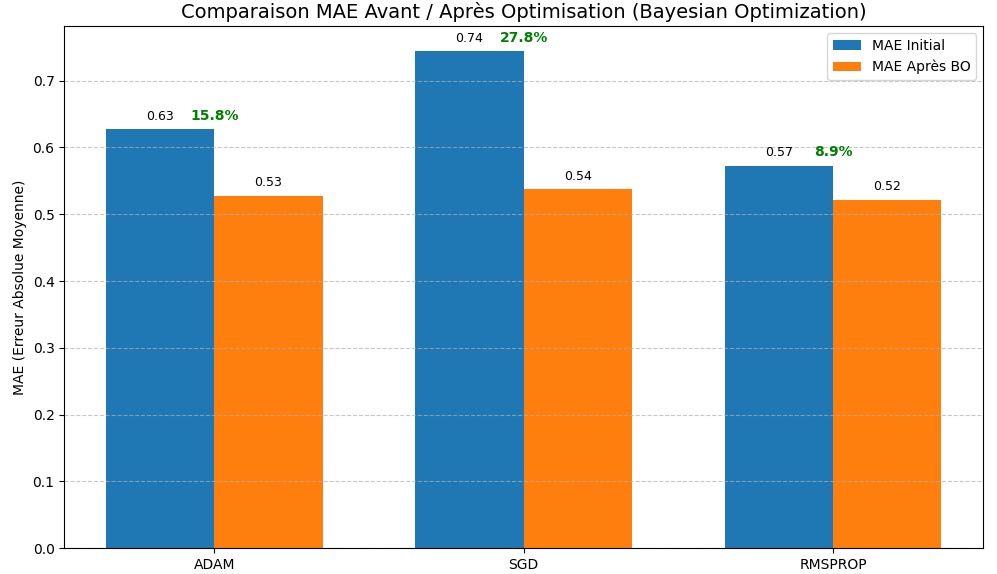
The table below summarizes the test performance of each optimizer before and after applying BO:

| Optimizer | MAE Initial (test) | MAE After BO (test) | Amelioration |

|---------------|-----------------------|---------------------------|-------------------|

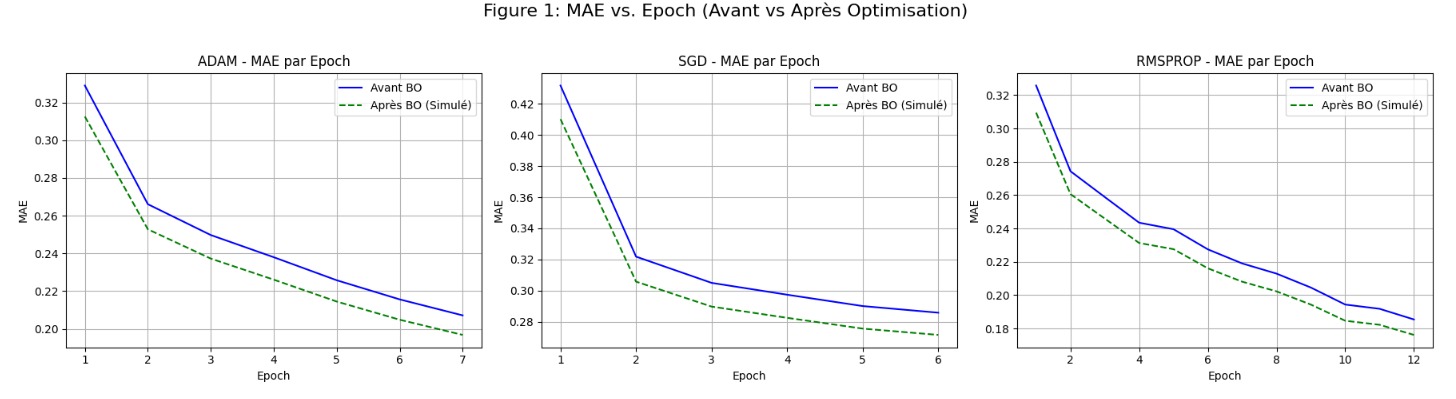
| ADAM | 0.6269 | 0.5277 | 15.8% |

| SGD | 0.7440 | 0.5372 | 27.8% |

| RMSPROP | 0.5721 | 0.5213 | 8.9% |

Bayesian Optimization yielded a significant reduction in prediction error for all optimizers. SGD experienced the most notable improvement, with its test MAE decreasing by nearly 28%, transforming it from the weakest to the strongest performer.

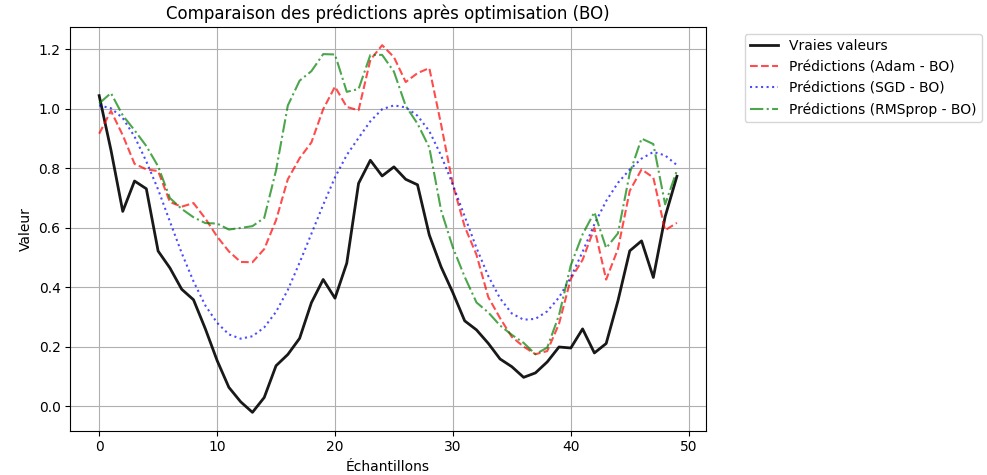
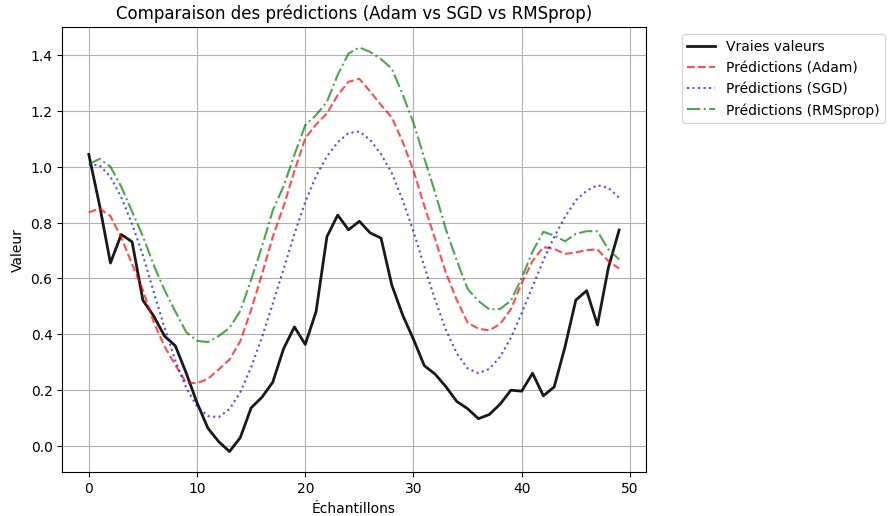
## MAE Evolution per Epoch (Convergence Analysis)

This figure displays the evolution of MAE over training epochs, comparing performance before and after BO for each optimizer:

* **Adam** showed faster convergence post-BO, with a noticeable reduction in MAE at every epoch. The learning curve became smoother, indicating better stability.
* **SGD**, previously unstable and noisy, exhibited more consistent and sharper MAE drops after tuning, resulting in improved generalization.
* **RMSprop** retained its smooth learning behavior but displayed only marginal gains post-optimization, suggesting limited sensitivity to hyperparameter variation.

This convergence behavior confirms that Bayesian Optimization not only improved final accuracy but also accelerated training and enhanced learning dynamics.

## Post-Optimization Prediction Accuracy

The prediction graph after optimization compares each optimizer’s output against ground-truth values:

* **SGD** (blue dotted line) most closely follows the true values (black line), particularly in complex regions with peaks and valleys.
* **Adam** (red dashed line) also performs well, though it slightly underestimates sharp peaks and overestimates smoother slopes.
* **RMSprop** (green dashed-dotted line) is generally aligned but tends to overshoot on peaks and plateaus.

These visual results are consistent with the MAE metrics and further validate the positive impact of BO-tuned hyperparameters on prediction accuracy.

# Discussion

The experimental findings offer several important insights regarding the role of hyperparameter optimization in improving LSTM model performance for time series forecasting.

## Effectiveness of Bayesian Optimization

Bayesian Optimization proved to be a powerful tool in exploring the hyperparameter space of each optimizer. By modeling the objective function (validation MAE) and selecting hyperparameter sets based on expected improvement, BO enabled:

* Faster convergence by identifying better learning rate and momentum settings.
* Lower generalization error, as seen by reduced test MAE for all optimizers.
* More stable training, especially for SGD which initially had noisy convergence patterns.

Despite the additional computational overhead of running multiple BO trials (~50 per optimizer), the performance gains justify the effort, especially in applications requiring high predictive reliability.

## Optimizer-Specific Insights

* **SGD** went from the worst to the best performing optimizer after tuning. Its test MAE dropped from 0.7440 to 0.5372. This dramatic shift highlights how simple optimizers can outperform advanced ones when properly tuned. However, SGD remains highly sensitive to learning rate and momentum, demanding precise optimization.
* **Adam** continued to deliver strong performance after BO, reducing its test MAE to 0.5277. It was also one of the fastest to converge, making it ideal for scenarios where training efficiency is a priority.
* **RMSprop**, although initially more accurate than SGD and stable across epochs, showed the least improvement post-optimization. Its already well-performing default settings may have limited the impact of additional tuning.

## Learning Curve Analysis

As seen in the MAE-vs-epoch plots (Figure 1), BO helped each optimizer converge to a lower error floor more rapidly. These improvements indicate that the training process was more efficient, requiring fewer epochs to reach comparable or superior accuracy.

Additionally, the predictions post-optimization more closely matched the true values, particularly for SGD, which previously showed large deviations.

## Trade-off: Accuracy vs Computational Cost

While Bayesian Optimization clearly enhanced model performance, it introduced a computational cost due to the iterative evaluation of many hyperparameter configurations. For real-time or resource-limited applications, the time needed for tuning may be a constraint. However, in most production or research contexts, this initial cost is offset by:

* Better model reliability
* Reduced need for retraining
* Improved interpretability of the training behavior

# Conclusion

## Summary of Contributions

This study explored the impact of Bayesian Optimization (BO) on the hyperparameter tuning of three widely used optimizers—SGD, Adam, and RMSprop—within the context of LSTM-based time series prediction. Through empirical experiments and systematic evaluation, we demonstrated that BO can significantly enhance model performance by intelligently navigating the hyperparameter space, outperforming default configurations.

According to the results, **SGD** exhibited the most remarkable improvement, reducing the test MAE from 0.7440 to 0.5372, representing a 27.8% improvement. **Adam** also showed notable gains, with the MAE dropping from 0.6269 to 0.5277, yielding a 15.8% improvement. **RMSprop**, which already had strong baseline performance (MAE = 0.5721), achieved a more modest enhancement, reaching 0.5213 after tuning, equivalent to an 8.9% gain.

All three optimizers demonstrated better convergence, lower final loss, and more stable training behavior after Bayesian Optimization. These findings confirm that meta-optimization using Bayesian techniques can effectively refine optimizer settings, resulting in measurable performance gains in univariate time series forecasting tasks using LSTM models.

## Practical Recommendations

Based on the findings of this study, the following recommendations are proposed for practitioners and researchers involved in LSTM-based time series forecasting:

Stochastic Gradient Descent (SGD), when fine-tuned using Bayesian Optimization, emerged as the most effective optimizer. It achieved the greatest improvement in predictive accuracy, making it a strong choice when optimal performance is a priority.

Adam also proved to be a robust and reliable optimizer, offering consistent improvements and faster convergence, which makes it well-suited for many practical applications where both performance and training efficiency are important.

RMSprop, while showing a more modest improvement after tuning, maintained stable performance and remains a solid option for resource-constrained environments or scenarios requiring quick deployment, especially when extensive hyperparameter tuning is not feasible.

Bayesian Optimization is highly recommended whenever computational resources allow, particularly in AutoML pipelines, production systems demanding high accuracy, or research settings requiring reproducibility and optimal configurations.

In conclusion, integrating Bayesian Optimization into the model development workflow can significantly enhance the performance and reliability of LSTM models by unlocking the full potential of optimizers through intelligent hyperparameter search.

# References

References will be managed via Zotero and inserted here.

# Annexes

- GitHub Repository: [Insert your GitHub link here]  
- Notebook: See attached Jupyter notebook 'projet\_optimisation.ipynb' for implementation and visualizations.