**Vietnam General Confederation of Labor**

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**

**A red and blue logo

Description automatically generated**

**FINAL REPORT**

**MACHINE LEARNING**

*Instructor*: **Mr. LE ANH CUONG**

*Student*: **Dang Thanh Dat -** *Mssv***: 521H0213**

*Class* **: 21H50301**

*Year***: 25**

# **HO CHI MINH CITY, 2023**

**Vietnam General Confederation of Labor**

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**

**A red and blue logo

Description automatically generated**

**FINAL REPORT**

**MACHINE LEARNING**

*Instructor*: **Mr. LE ANH CUONG**

*Student*: **Dang Thanh Dat -** *Mssv***: 521H0213**

*Class* **: 21H50301**

*Year***: 25**

# **HO CHI MINH CITY, 2023**

# ACKNOWLEDGEMENT

The first sincere thanks we want to give to Mr. Le Anh Cuong, who enthusiastically taught and worked tirelessly to give us enough tools and skills to complete this report. He played an important role in improving our mathematical logic and knowledge. The second thanks we would like to give to the teachers of the Department of Information Technology of Ton Duc Thang University for giving us the opportunity to do this report. Nevertheless, our report will have some errors, we are open to receive any feedback from teachers so that we can improve our report.

Finally, we wish you good health and success in your noble career.

*Ho Chi Minh city, 21st December, 2023*

*Dat*

*Dang Thanh Dat*

**THIS PROJECT WAS COMPLETED AT**

**TON DUC THANG UNIVERSITY**

We fully declare that this is our own project and is guided by Mr. Le Anh Cuong; The research contents and results in this topic are honest and have not been published in any form before.

**Should any frauds be found, I will take full responsibility for the content of my report.** Ton Duc Thang University is not related to copyright and copyright violations caused by me during the implementation process (if any).

*Ho Chi Minh city, 11th October, 2023*

*Author*

*(Sign and write full name)*

# CONFIRMATION AND ASSESSMENT SECTION

**Instructor confirmation section**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Ho Chi Minh , 2023*

*(Sign and write full name)*

**Evaluation section for grading instructor**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Ho Chi Minh , 2023*

*(Sign and write full name)*

**Summary**

**Comparison of Machine Learning Model Training Optimizers:**

This section examines various algorithms including Gradient Descent, Stochastic Gradient Descent (SGD), Mini-batch Gradient Descent, and Adaptive Methods such as Adagrad, RMSprop, and Adam. It explores the advantages, limitations, and specific uses of each optimizer in improving the training and efficiency of machine learning models.

**Exploring Continual Learning and Test Production in Machine Learning:**

The emphasis in Continual Learning is on a model's capacity for ongoing adaptation and learning, maintaining previously acquired knowledge. Test Production is focused on developing appropriate evaluation metrics and test datasets, which are especially vital in situations where there is a shift in data distributions or when learning new tasks without the original training data.

# INDEX

[ACKNOWLEDGEMENT 3](#_Toc153398414)

[CONFIRMATION AND ASSESSMENT SECTION 5](#_Toc153398415)

[SUMMARY 6](#_Toc153398416)

[INDEX 7](#_Toc153398417)

[Task 1: 8](#_Toc153398418)

**[1.Exploring and Contrasting Diverse Optimization Techniques for Training Machine Learning Models.](#_Toc153398419)** [8](#_Toc153398419)

[2.](#_Toc153398420) **[Investigating Continuous Learning and Test Development in Building a Machine Learning Solution for Problem Solving.](#_Toc153398420)** [11](#_Toc153398420)

**Continuous Learning (Lifelong Learning)**

**[Test Development for Machine Learning Solutions](#_Toc153398422)**

**Task 1**

## **Exploring and Contrasting Diverse Optimization Techniques for Training Machine Learning Models.**

Understanding and comparing optimization methods (Optimizer) in training machine learning models is an important topic in the field of deep learning and machine learning. Here's an overview of some popular optimization methods and how they differ from each other:

1. **Gradient Descent**: This is the most basic method, in which the model weights are updated in the opposite direction to the gradient of the loss function. The main weakness of Gradient Descent is that it updates the weights after calculating the gradient for the entire dataset, making it ineffective with large data.

Advantages:

-Simple Concept: Gradient Descent is easy to understand and implement. It forms the basis for more advanced optimization algorithms.

-Global Convergence: In convex optimization problems, Gradient Descent guarantees convergence to the global minimum of the loss function.

Disadvantages:

-Computational Intensity: Computationally expensive, especially with large datasets, as it requires the calculation of gradients for the entire dataset in each iteration.

-Memory Requirements: Since it processes the entire dataset, it may demand a large amount of memory, making it impractical for big data.

1. **Stochastic Gradient Descent (SGD)**: In SGD, weights are updated for each  mple or a small group of samples (mini-batch), which increases learning speed and reduces required memory. However, this approach can lead to instability in the learning process.

Advantages:

-Faster Learning: Updating weights based on individual samples or small batches speeds up the learning process compared to Batch Gradient Descent, especially for large datasets.

-Reduced Memory Requirements: Since it processes only a subset of the data at a time, SGD requires less memory compared to Batch Gradient Descent, making it more suitable for large datasets.

-Potential for Online Learning: Well-suited for online learning scenarios where new data is continuously added, as it updates weights incrementally.

-Escaping Local Minima: The stochastic nature can help the algorithm escape from local minima due to the random sampling of data points.

Disadvantages:

-High Variability: The use of individual samples or small batches introduces a high level of variability in the updates, leading to a noisy convergence path. This can make it harder to reach a stable and optimal solution.

-Less Efficient Convergence: The high variability can result in slower convergence or oscillations around the minimum, especially in the presence of noise.

-Difficulty in Choosing Learning Rate: The choice of an appropriate learning rate becomes crucial, as too large a learning rate can lead to divergence, and too small a learning rate can slow down convergence.

-Not Always the Global Minimum: While it can escape local minima, there's no guarantee that SGD will converge to the global minimum in non-convex optimization problems.

**3**. **Mini-batch Gradient Descent**: This is a balance between Gradient Descent and SGD. It updates the weights after each subgroup of samples, which reduces the instability of SGD while maintaining efficiency.

Advantages:

-Efficiency: Like SGD, mini-batch gradient descent is more computationally efficient compared to batch gradient descent, as it processes a small random subset of the data in each iteration.

-Reduced Variability: Mini-batches help to reduce the high variability of updates seen in pure SGD. This can lead to a smoother convergence path, making it easier to find a stable solution.

-Memory Efficiency: It requires less memory than batch gradient descent since it doesn't need to store the entire dataset in memory.

-Applicability to Parallel Computing: Mini-batch processing is well-suited for parallel computing environments, as the computation can be distributed across multiple processors or GPUs.

Disadvantages:

-Not Always Optimal for All Datasets: The choice of the mini-batch size is crucial and might require some experimentation. An inappropriate size can lead to suboptimal convergence or increased computational overhead.

-Sensitivity to Learning Rate: Like SGD, mini-batch gradient descent is sensitive to the learning rate hyperparameter. Careful tuning is necessary to ensure convergence.

-Possible Local Minima: While less prone to the high variability of SGD, mini-batch gradient descent may still face challenges in escaping local minima in non-convex optimization problems.

**4. Momentum**: Momentum is an improvement of SGD, it accumulates past gradients to determine the current update direction, helping to speed up learning and minimize getting stuck at saddle points.

Advantages:

-Accelerated Convergence: Momentum helps to accelerate the convergence of the optimization process by accumulating past gradients. This allows the optimizer to maintain a more consistent and directed update direction, especially in the presence of noise or curvature in the loss landscape.

-Overcoming Saddle Points: Momentum is effective in overcoming the problem of getting stuck in saddle points. The accumulated momentum allows the optimizer to "roll through" flat regions of the loss surface.

-Smoothing Effect: The momentum term acts as a smoothing factor, reducing oscillations in the optimization path and providing more stable updates.

Disadvantages:

-Sensitivity to Hyperparameter: The performance of momentum is sensitive to the choice of its hyperparameter (often denoted as β or γ). An inappropriate value can lead to suboptimal convergence or instability.

-Possible Overfitting: In some cases, high momentum values might cause the optimizer to overshoot the minimum, leading to oscillations or instability.

1. **Adagrad:** Adagrad adjusts the learning rate automatically for each parameter. It reduces learning speed quickly and effectively for sparse data and problems with large gradients.

Advantages:

-Adaptive Learning Rates: Adagrad adapts the learning rate for each parameter individually based on the historical gradient information. This allows it to automatically reduce the learning rate for parameters that have large gradients and increase it for parameters with small gradients.

-Effective for Sparse Data: Adagrad is particularly effective for sparse data or problems where some features have infrequent occurrences. It automatically assigns smaller learning rates to parameters associated with frequently occurring features.

-Accommodation of Large Gradients: Adagrad is suitable for problems with large gradients, as it scales the learning rates according to the historical gradient magnitudes.

Disadvantages:

-Monotonic Learning Rate Decrease: One drawback of Adagrad is that the learning rates for each parameter monotonically decrease during training. This can result in overly aggressive reductions in the learning rate, making it slow to converge in the later stages of training.

-Accumulation of Squared Gradients: Adagrad accumulates the squared gradients over time, leading to an increase in the denominator of the learning rate update formula. This accumulation can result in very small learning rates, effectively slowing down learning.

-Memory Requirements: The algorithm needs to store the sum of squared gradients for each parameter, leading to increased memory requirements.

**6.RMSprop:** RMSprop is a variant of Adagrad that solves the problem of dropping the learning rate too quickly by using a moving average of the square of the gradient to adjust the learning rate.

Advantages:

-Adaptive Learning Rates: Similar to Adagrad, RMSprop adapts the learning rate for each parameter individually based on historical gradient information.

-Addressing Monotonic Learning Rate Decrease: RMSprop introduces a moving average of the squared gradients, which helps mitigate the issue of the learning rate dropping too quickly over time.

-Effective for Non-Stationary Objectives: RMSprop is particularly effective for non-stationary objectives where the optimal learning rate may change during training.

-Memory Efficiency: Compared to Adagrad, RMSprop addresses the issue of excessive memory requirements by using a moving average of squared gradients instead of accumulating them.

Disadvantages:

-Sensitivity to Hyperparameters: The performance of RMSprop is sensitive to the choice of hyperparameters, such as the decay rate for the moving average.

-Lack of Momentum Term: RMSprop does not include a momentum term, which can be a limitation in certain scenarios where momentum is beneficial.

**7. Adam**: Adam combines ideas from RMSprop and Momentum. It calculates moving averages of gradients and gradient squares, which improves stability and learning speed.

Advantages:

-Adaptive Learning Rates: Adam adapts the learning rate for each parameter individually, similar to RMSprop and Adagrad, by calculating moving averages of both gradients and squared gradients.

-Efficient Momentum: Adam incorporates a momentum term, allowing it to maintain a moving average of past gradients. This helps accelerate convergence and smooth out noisy updates.

-Effective for Sparse Gradients: Adam is effective for problems with sparse gradients, as it uses moving averages to handle variations in gradient magnitudes.

-Automatic Bias Correction: Adam performs automatic bias correction for the moving averages, addressing potential bias towards zero in the early iterations.

Disadvantages:

-Sensitivity to Hyperparameters: Like other adaptive optimization algorithms, Adam's performance can be sensitive to the choice of hyperparameters, such as the learning rate, decay rates, and initial moment estimates.

-Memory Requirements: Adam requires additional memory to store the moving averages of gradients and squared gradients.

Each optimization method has its advantages and disadvantages, and the choice will depend on the specific problem and the type of data being worked with. Experimenting with many different optimization methods to find the most suitable method for a specific problem is an important part of the process of training a machine learning model.

## **2. Investigating Continuous Learning and Test Development in Building a Machine Learning Solution for Problem Solving.**

Investigating Continuous Learning and Test Development in building a machine-learning solution for problem-solving involves understanding and implementing strategies that ensure a machine-learning model remains effective and accurate over time, particularly in changing environments. Here's a detailed look at each aspect:

**Continuous Learning (Lifelong Learning)**

1.Definition: Continuous or lifelong learning in machine learning refers to the ability of a model to continuously learn from new data, adapting to new conditions and integrating new knowledge without forgetting previous learnings.

2.Importance: This is crucial in dynamic environments where data distributions change over time, requiring the model to evolve rather than become obsolete.

3. Techniques:

   - Incremental Learning: Updating the model with new data without retraining from scratch.

   - Transfer Learning Applying knowledge gained from one problem to new but related problems.

   - Regularization Techniques: Preventing overfitting to new data while retaining knowledge of old data (e.g., Elastic Weight Consolidation).

   - Rehearsal Methods: Mixing old and new data in training to prevent forgetting.

   - Meta-Learning: Learning to learn, where the model optimizes its learning strategy over time.

**Test Development for Machine Learning Solutions**

1. Purpose: Testing in machine learning ensures that models perform well not only on training data but also on unseen data, maintaining their accuracy and reliability in real-world scenarios.

2. Test Data Creation:

   - Realistic Scenarios: Creating test datasets that closely mimic real-world scenarios the model will encounter.

   - Changing Distributions: Including data that represent shifts or drifts in the input distribution over time.

3. Evaluation Strategies:

   - Cross-Validation: Using multiple subsets of data to evaluate the model's performance.

   - A/B Testing: Comparing the performance of different models or different versions of a model in a controlled environment.

   - Performance Metrics: Utilizing various metrics like accuracy, precision, recall, F1 score, ROC-AUC for classification problems, and MSE, and RMSE for regression tasks.

4. Continuous Monitoring and Updating:

   - Model Drift Monitoring: Regularly checking for changes in model performance due to evolving data patterns.

   - Feedback Loops: Incorporating user feedback and real-world outcomes to refine the model.

5. Ethical and Fairness Considerations:

   - Ensuring the test datasets and evaluation strategies account for fairness and bias, avoiding discriminatory or unethical outcomes.

In summary, building a machine learning solution that leverages continuous learning and robust test development involves creating models that can adapt over time, are rigorously tested against realistic and evolving scenarios, and are continuously monitored and updated to ensure their relevance and fairness.