VIETNAM GENERAL CONFEDERATION OF LABOR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**

Logo

Description automatically generated

**FINAL PROJECT EXAM**

**MACHINE LEARNING**

*Instructor:* **PhD. Lê Anh Cường**

*Student:* **Vi Thành Đạt**

*Student ID:* **521H0390**

*Class:* **21H50301**

**HO CHI MINH CITY, 2023**

VIETNAM GENERAL CONFEDERATION OF LABOR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**

Logo

Description automatically generated

**FINAL PROJECT EXAM**

**MACHINE LEARNING**

*Instructor:* **PhD. Lê Anh Cường**

*Student:* **Vi Thành Đạt**

*Student ID:* **521H0390**

*Class:* **21H50301**

**HO CHI MINH CITY, 2023**

# ACKNOWLEDGEMENT

**THE PROJECT WAS COMPLETED**

**AT TON DUC THANG UNIVERSITY**

I would like to assure you that this is my own project and guided by Mr. Lê Anh Cường. The research contents and results in this topic are honest and have not been published in any form before. The data in the tables for analysis, comments, and evaluations collected by the author himself from different sources are clearly stated in the references section.

In addition, the project also uses some comments, reviews as well as figures of other authors and other organizations with quotes and annotations of origin.

If any fraud is detected, I would like to take full responsibility for the content of my project. Ton Duc Thang University is not involved in copyright or copyright violations caused by me in the process (if any).

*Ho Chi Minh City, 10 December 2023*

*Author*

*(sign and write your full name)*

*Vi Thành Đạt*

**FACULTY ENDORSEMENTS AND REVIEWS**

**Phần xác nhận của GV hướng dẫn**

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

Tp. Hồ Chí Minh, ngày tháng năm

(ký và ghi họ tên)

**Phần đánh giá của GV chấm bài**

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

Tp. Hồ Chí Minh, ngày tháng năm

(ký và ghi họ tên)

# TABLE OF CONTENTS

Contents

[ACKNOWLEDGEMENT 1](#_Toc153309625)

[TABLE OF CONTENTS 1](#_Toc153309626)

[LIST OF SYMBOLS AND ABBREVIATIONS 1](#_Toc153309627)

[CHAPTER 1: INTRODUCTION 2](#_Toc153309628)

[CHAPTER 2: 2](#_Toc153309629)

[CHAPTER 3: 3](#_Toc153309630)

[REFERENCE 3](#_Toc153309631)

# LIST OF SYMBOLS AND ABBREVIATIONS

# CHAPTER 1: OPTIMIZERS

**Introduction:**

The process of training deep learning models involves the use of optimization techniques. The loss function, which gauges how effectively a model can predict a given dataset, is what they have to minimize by altering the model's parameters. Choosing one of the many optimization techniques available can have a big impact on how well the model performs.

In deep learning, optimizers are algorithms that modify a model's parameters in order to minimize a loss function. Model training performance and speed can be significantly impacted by the optimizer selection. Some of the most popular deep learning optimizers will be introduced in this report, including: Gradient Descent (GD), Stochastic Gradient Descent (SGD), Mini-Batch Gradient Descent, Momentum Based Gradient Descent, Adagrad, RMSprop, Adam.

1. **Gradient Descent (GD):**

* Gradient descent is by far the most popular optimization strategy used in machine learning and deep learning at the moment. It is used when training data models, can be combined with every algorithm and is easy to understand and implement. This algorithm is based on finding the extremum (maximum or minimum) of a function by calculating the derivative and moving in the decreasing direction of the gradient.

A diagram of a weight loss

Description automatically generated

* 1. Funtion of one variable:

- The Gradient Descent algorithm begins by initializing the variable x at a random position, moves x in the opposite direction to the derivative of f(x), and then finds the minimum of the function. This process will be carried out again until a predetermined threshold is met.

Update fomula in Gradient Descent:

Where:

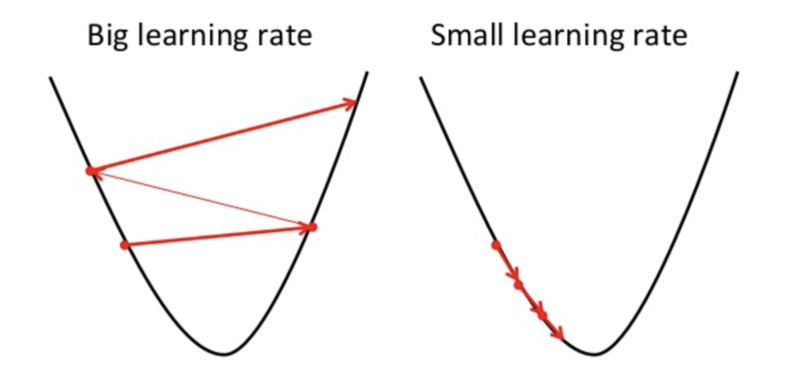
* is the value at the second step
* is the learning rate (learning speed)
* is derivative of the function in
  1. Multi-variable function:

- For a function of many variable , this algorithm will calculate the gradient (derivative vector) of the function at the random point   then move opposite direction to this Gradient.

Update fomula in Gradient Descent:

Where:

* is the value   at the second step
* is the learning rate (learning speed)
* is the gradient of the function in
  1. Learning rate:
* How big the steps gradient descent takes into the direction of the local minimum are determined by the learning rate, which figures out how fast or slow we will move towards the optimal weights.
* For the gradient descent algorithm to reach the local minimum we must set the learning rate to an appropriate value, which is neither too low nor too high. This is important because if the steps it takes are too big, it may not reach the local minimum because it bounces back and forth between the convex function of gradient descent (see left image below). If we set the learning rate to a very small value, gradient descent will eventually reach the local minimum but that may take a while (see the right image).



* So, the learning rate should never be too high or too low for this reason. You can check if your learning rate is doing well by plotting it on a graph.
  1. Advantages:
     + Very simple to implement.
     + Can work well with a well-tuned learning rate.
  2. .Disadvantages:
     + It can converge slowly, especially for complex models or large datasets.
     + Sensitive to the choice of learning rate.

1. **Momentum Based Gradient Descent:**

**-** There are a few problems that can occur when using Gradient Descent: local minima, saddle points, plateaus, oscillations, slow convergence, stochasticity,vanishing or exploding gradients.

* 1. Local Minima:
* Gradient descent can get stuck in local minima, points that are not the global minimum of the cost function but are still lower than the surrounding points. This can occur when the cost function has multiple valleys, and the algorithm gets stuck in one instead of reaching the global minimum.

A graph of a graph of a graph

Description automatically generated with medium confidence

* 1. Saddle Points:
* A saddle point is a point in the cost function where one dimension has a higher value than the surrounding points, and the other has a lower value. Gradient descent can get stuck at these points because the gradients in one direction point towards a lower value, while those in the other direction point towards a higher value.

A graph of a saddle point

Description automatically generated

* 1. Plateaus:
* A plateau is a region in the cost function where the gradients are very small or close to zero. This can cause gradient descent to take a long time or not converge.

A diagram of a diagram

Description automatically generated

* 1. Oscillations:
* Oscillations occur when the learning rate is too high, causing the algorithm to overshoot the minimum and oscillate back and forth.

A diagram of a diagram of a curve

Description automatically generated

* 1. Slow Convergence:
* Gradient descent can converge very slowly when the cost function is complex or has many local minima. This means the algorithm may take a long time to find the global minimum.

A graph of a function

Description automatically generated

* 1. Stochasticity:
* In stochastic gradient descent, the cost function is evaluated at random samples from the data set. This introduces randomness into the algorithm, making converging to a global minimum more difficult.

A diagram of a diagram

Description automatically generated

* 1. Vanishing or Exploding:
* Deep neural networks with many layers can suffer from vanishing or exploding gradients. This occurs when the gradients become very small or large, respectively, as they are backpropagated through the layers. This can make it difficult for the algorithm to update the weights and biases.

A diagram of a diagram of a diagram

Description automatically generated with medium confidence

* 1. How can Momemtum fix the problems
* For some of these issues, gradient descent with momentum can be used. To give the optimization method momentum as it descends the loss function, it adds a portion of the prior weight update to the current weight update. By doing so, the algorithm may be able to avoid oscillations and escape from saddle points and local minima, allowing it to converge more quickly.
* Momentum helps to,
* Escape local minima and saddle points
* Aids in faster convergence by reducing oscillations
* Smooths out weight updates for stability
* Reduces model complexity and prevents overfitting
* Can be used in combination with other optimization algorithms for improved performance.
* Used and Applied in Gradient Descent:
  + To applied GD with momentum, we can update weights as follows:
  + Where:
    - is the momentum term
    - is momentum hyperparameter ( typically set to 0.9)
    - learning\_rate is learning rate
    - gradient is the gradient of the loss function to the weights
* Momentum Base Gradient Descent work: Gradient descent with momentum is an optimization algorithm that helps accelerate the convergence of gradient descent by adding a momentum term to the weight update. The momentum term is based on the previous weight update and helps the algorithm build momentum as it descends the loss function.
* Here is the math behind momentum-based gradient descent:
* Let's say we have a set of weights w and a loss function L, and we want to use gradient descent to find the weights that minimize the loss function. The standard gradient descent update rule is:
* To incorporate gradient descent with momentum into this update rule, we can add a momentum term v that is based on the previous weight update:
* Where:
  + is learning rate
  + gradient is the gradient of the loss function to the weights.
  + rho is the momentum hyperparameter (typically set to 0.9).
  + The momentum term v can be interpreted as the "velocity" of the optimization algorithm, and it helps the algorithm build momentum as it descends the loss function. This can help the algorithm escape from local minima and saddle points and can also help the algorithm converge faster by avoiding oscillations.

A diagram of a graph

Description automatically generated

1. **Adagrad:**

* An adaptive learning rate is used by the optimization algorithm Adagrad for each parameter. Based on the previous gradient data, the learning rate is modified, giving parameters that receive more updates a higher learning rate and those that receive fewer updates a lower learning rate. The following is one way to write the update rule:
* Where:
  + G is matrix that accumulates the square of gradients
  + is a small constant added to avoid division by zero
* Advantages:
  + It can work well with sparse data.
  + Automatically adjusts learning rates based on parameter updates.
* Disadvantages:
  + Can converge too slowly for some problems.
  + Can stop learning altogether if the learning rates become too small.

1. **RMSprop:**

* RMSProp is an optimization algorithm similar to Adagrad, but it uses an exponentially decaying average of the squares of the gradients rather than the sum. This helps to reduce the monotonic learning rate decay of Adagrad and improve convergence. We can write the update rule as follows:
* Where:
  + G is matrix that accumulates the square of gradients
  + is a small constant added to avoid division by zero
  + β is a decay rate hyperparameter.
* Advantages:
  + Can work well with sparse data.
  + Automatically adjusts learning rates based on parameter updates.
* Disadvantages:
  + Can converge too slowly for some problems.
  + Can stop learning altogether if the learning rates become too small.

1. **Adam:**

* Adam (Adaptive Moment Estimation) is an algorithm that allows calculation of adaptive learning speed for each weight. Adam not only stores the squared average of previous gradients such as Adadelta, but also the moment average. Values and ,. Calculated by the formula:
* Where:
  + is non-negative weights (typically we set )
  + If and are initialized as 0 vectors, these values ​​tend to lean towards 0, especially when and are approximately equal to 1. Therefore, to overcome this, these values ​​are estimated by way:A mathematical equations on a white background

    Description automatically generated
  + Then update the weights according to the formula:

A black and white image of a mathematical equation

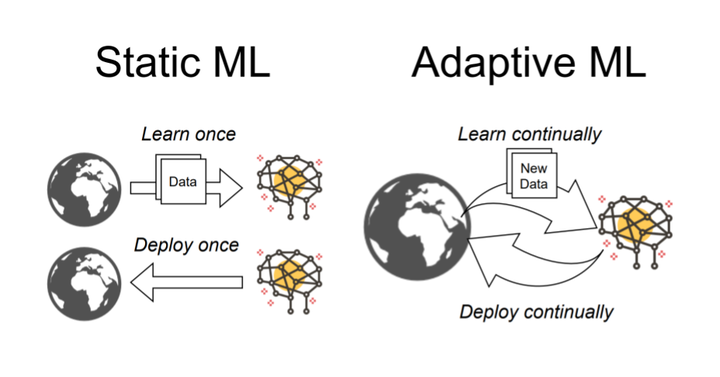
Description automatically generated

* Advantages:
  + Can converge faster than other optimization algorithms.
  + Can work well with noisy data.
* Disadvantages:
  + It may require more tuning of hyperparameters than other algorithms.
  + May perform better on some types of problems.

# CHAPTER 2: CONTINUAL LEARNING AND TEST PRODUCTION

1. Continual Learning
   1. Introduction:

* Continual Learning (also known as Incremental Learning, Life-long Learning) is a concept to learn a model for a large number of tasks sequentially without forgetting knowledge obtained from the preceding tasks, where the data in the old tasks are not available anymore during training new ones.



* In conventional machine learning, a model is supposed to do a single task after being trained on a fixed dataset. Nevertheless, this method presents issues when the tasks and data are dynamic and ever-changing, since the model needs to be able to adjust and gain knowledge from fresh data over time. This is where CL comes into play, allowing the model to learn new things all the time without losing what it already knows.
* The problem of catastrophic forgetting, where a model trained on many tasks needs to retain information from prior tasks when exposed to new data, is one of the major issues in CL. Regularization and memory-augmented networks are two methods that have been developed to address this issue.
* Regularization is a technique used to prevent overfitting to fresh data by introducing limits into the learning process. Alternatively, memory-augmented networks incorporate memory elements that retain data from prior tasks and leverage this data to enhance performance on subsequent tasks.
* An significant factor in a model's adaptability is also its architecture. Certain models include designs that make it easier for them to absorb new data and knowledge, making them more adaptive and flexible. For instance, modular architectures boost the model's flexibility in adjusting to new tasks and data by enabling the independent training and adaptation of its many components.
* The adaptability of a model is also influenced by the availability of task-specific data. Because they have more data to work with when making predictions, models that have access to vast volumes of task-specific data are better equipped to adapt and learn. To increase their flexibility and capacity to generalize to new tasks, some models are trained on enormous volumes of data.
* To sum up, CL is a significant area in machine learning that deals with the problem of developing models that are constantly learning and adapting to new situations and sets of data without losing track of their prior knowledge. The architecture of the model and the application of methods like regularization and memory-augmented networks are key factors in how flexible CL models are.
  1. Use case:
* Continual learning poses some inherent benefits that can serve the following use cases.

A diagram of a diagram of a learning process

Description automatically generated

* 1. Type of coutinual learning:
* Task-based Continual Learning: Using this approach, a version picks up a series of unique responsibilities over time. The objective of the model is to adapt to every new task while maintaining awareness of previously discovered responsibilities. This class comprises methods such as Progressive Neural Networks (PNN) and Elastic Weight Consolidation (EWC).
* Class-incremental Learning: The specialty of class-incremental mastering is handling new courses or informational classes over time while maintaining comprehension of previously seen teachings. This is typical of programs such as image recognition, where new object training is added on a regular basis. For class-incremental mastering, techniques such as iCaRL (Incremental Classifier and Representation Learning) are employed.
* Domain-incremental Learning: This type of learning is concerned with adjusting to new domain names or record distributions. In autonomous robotics, for instance, a robot can wish to adjust to various surroundings. To address this situation, area-incremental learning and area variation techniques are applied.
  1. Step of continual learning:
* Initialization
* Task Sеquеncing
* Training on a Task
* Rеgularization for Knowlеdgе Prеsеrvation
* Knowlеdgе Distillation
* Tеsting and Evaluation
* Storing Knowlеdgе
* Task Switching
* Itеrativе Lеarning
* Monitoring and Adaptation
* Hypеrparamеtеr Tuning
* Tеrmination or Expansion
* Rеal-world Dеploymеnt
  1. Advantages of Continual Learning:
* Incremental Learning: Learning new information over time usually involves teaching a model new fact in little steps. This is known as continuous learning. This means that without retraining on the entire dataset, the version must adjust to new statistics.
* Memory and Forgetting: Models of persistent studying seek strategies to prevent catastrophic forgetting, in which learners fail to meet previously identified duties while learning about new ones, as well as mechanisms to help learners retain and preserve important information from prior reviews.
* Task Sequences: The terms of the series in which tasks are encountered can change as a result of ongoing situational learning. While some may also have a more dynamic or unpredictable sequence, others may also have a fixed order of commitments.
* Regularization and Stabilization: Model weights are regularized and stabilized using strategies like synaptic intelligence (SI) and elastic weight consolidation (EWC) to prevent you from making significant changes as you learn new responsibilities and to help you maintain prior knowledge.
* Replay and Experience Replay:  To strengthen and solidify the knowledge gained during earlier tasks, replay mechanisms entail going back and retraining on records or stories that have passed.
* Transfer of Learning: One of the main problems with permanent mastery is that one cannot learn new tasks without drawing on prior experience. Methods like first-class tuning and characteristic reuse could be useful.

1. Test production

In a perfect world, releasing a product would probably be bug-free. But in reality, it's not that simple. Therefore, both product managers and developers need to ensure there is a process in place to help detect any product issues. This is where testing production comes in.

* 1. Test in production
* Testing in production is a software development practice of running different tests on your product when it’s in a live environment in real time.
* It allows you to test new code changes on live users rather than a test or staging environment.
* This type of testing is not meant to be a replacement QA team or eliminating a unit test or integration test. In other words, it is not supposed to replace testing before production but to complement these tests.
  1. Big risks of testing in production
* Deploying bad code
* May accidentally leak sensitive data
* It can possibly cause system overload
* Mess up your tracking and analytics
* Risk releasing a poorly designed product or feature
  1. Should still test in production
* Collect real-world data and feedback: Testing in production allows you to collect user data in terms of users’ engagement with your new features. This enables the collection of valuable feedback from the customers that matters the most, which in turn would allow you to optimize the user experience based on this feedback.
* Uncover bugs: Since you’re testing on live users, you would be able to discover any bugs or issues that you may have otherwise missed in the development stage. Thus, you can ensure your new products and features are stable and capable of handling a high volume of real-world usage.
* Higher quality releases: Because you’re receiving continuous feedback from your users, developers can improve the products resulting in high quality releases that meet your customers’ needs and expectations.
* Support a larger strategy of incremental release: Because you’re receiving continuous feedback from your users, developers can improve the products resulting in high quality releases that meet your customers’ needs and expectations.
  1. Practice method:
* CI/CD
* A/B Testing
* Phased Rollouts
* Canary Deployments
* Blue/green deployments
* Usability Testing
* Smoke & Sanity Testing

# REFERENCE